即时通信

IMLib / IMKit

Web 4.X

2024-08-30

开发指导

更新时间:2024-08-30

欢迎使用融云即时通讯。本页面简单介绍了融云即时通讯架构、服务能力和 SDK 产品。

架构与服务

融云提供的即时通讯服务，不需要在 App 之外建立并行的用户体系，不用同步 App 下用户信息到融云，不影响 App 现有的系统架构与帐号体系，与现有业务体系能够实现完美融合。

融云的架构设计特点：

- 无需改变现有 App 的架构，直接嵌入现有代码框架中；
- 无需改变现有 App Server 的架构，独立部署一份用于用户授权的 Service 即可；
- 专注于提供通讯能力，使用私有的二进制通信协议，消息轻量、有序、不丢消息；
- 安全的身份认证和授权方式，无需担心 SDK 能力滥用（盗用身份的垃圾消息、垃圾群发）问题。

融云即时通讯产品支持单聊、群聊、超级群、聊天室 多种业务形态，提供丰富的客户端和服务端接口，大部分能力支持开箱即用。

业务类型介绍

单聊（Private）业务即一对一聊天。普通群组（Group）业务类似微信的群组。超级群与聊天室业务均不设用户总数上限。超级群（UltraGroup）[1] 类似 Discord，提供了一种新的群组业务形态，在超级群中提供公有/私有频道、用户组等功能，适用于构建超级社区。聊天室（Chatroom）只有在线用户可接收消息，广泛适用于直播、社区、游戏、广场交友、兴趣讨论等场景。融云的 IMKit 为 Android/iOS/Web 平台的单聊、普通群组业务提供了开箱即用的 UI 组件，其他情况下可以使用 IMLib SDK 构建您的业务体验。

单聊、群组、超级群、聊天室的主要差异如下：

功能	单聊（Private）	普通群组（Group）	超级群（UltraGroup）[1]	聊天室（Chatroom）
场景类比	类似微信私聊	类似微信群组	类似 Discord	聊天室
特性/优势	支持离线消息推送和历史消息记录漫游	支持离线消息推送和历史消息记录漫游，可用于兴趣群、办公群、客服服务沟通等	不限成员数量；支持修改已发消息；提供公有/私有频道、用户组等社群功能	不限成员数量；只有在线用户可接收消息，退出时清除本地历史消息
开通服务	不需要	不需要	需要	不需要
UI 组件	IMKit[2]	IMKit[2]	不提供	不提供
创建方式	无需创建	服务端 API	服务端 API	服务端 API；客户端加入时可自动创建
销毁/解散方式	不适用	服务端 API	服务端 API	服务端 API；具有自动销毁机制[3]

功能	单聊（Private）	普通群组（Group）	超级群（UltraGroup）	聊天室（Chatroom）
成员数量限制	不适用	群成员数上限 3000	不限	不限
用户加入限制	不适用	不限	最多加入 100 个群，每个群中可加入 50 个频道	默认仅可加入 1 个聊天室，可自行关闭限制[4]
获取加入前的消息	不适用	默认不允许，可关闭限制	默认不允许，可关闭限制	客户端加入聊天室即可获取最新消息，最多 50 条
客户端发送消息频率	每个客户端 5 条/秒[5]	每个客户端 5 条/秒[5]	每个客户端 5 条/秒[5]	每个客户端 5 条/秒[5]
服务端发送消息频率	6000 条/分钟[6]	20 条/秒[6]	100 条/秒[6]	100 条/秒[6]
扩展消息	支持	支持	支持	不支持
修改消息	不支持	不支持	支持	不支持
消息可靠度	100% 可靠	100% 可靠	100% 可靠	超出服务端消费上限的消息将被主动抛弃[7]
消息本地存储	移动端、PC 端支持	移动端、PC 端支持	移动端、PC 端支持	不支持
消息云端存储	需开通，可存储 6 - 36 个月[8]	需开通，可存储 6 - 36 个月[8]	默认存储 7 天，提供 3 - 36 个月存储服务[8]	需开通，可存储 2 - 36 个月[8]
离线缓存消息	默认 7 天离线消息缓存	默认 7 天离线消息缓存	不支持	不支持
消息本地搜索	支持	支持	支持	不支持
离线推送通知	支持	支持	支持，可调整推送频率	不支持

脚注：

1. 超级群业务仅限 IM 尊享版 使用。
2. IMKit 已支持 Android/iOS/Web 端。
3. 聊天室具有自动销毁机制。默认情况下，如果聊天室在指定时间内（默认 1 个小时）没有人说话，且没有人加入聊天室时，会把聊天室内所有成员踢出聊天室并销毁聊天室。您可以灵活调整聊天室的存活条件与存活时间。
4. 可允许单个用户加入多个聊天室，参考知识库文档：开通单个用户加入多个聊天室 。
5. 客户端不区分业务类型整体限制 5 条消息/秒，可付费上调。
6. 此处为服务端 API 默认频率，可付费上调。详细限频信息参见 API 接口列表 。
7. 聊天室消息量较大时，超出服务端消费上限的消息将被主动抛弃。您可通过用户白名单、消息白名单、自定义消息级别等服务，改变消息抛弃策略。如果用户在聊天室的用户白名单内，该用户所发送的消息在消息量大时也不会被抛弃。如需了解服务端消费上限与如何改变消息抛弃策略，可参见服务端文档消息优先级服务 、聊天室白名单服务 。
8. 参考知识库文档：单聊、群聊、聊天室、超级群在融云端历史消息存储时间分别是多长？ 。

前往融云产品文档 · 即时通讯 »

高级与扩展功能

IM 服务支持的高级与扩展功能，包括但不限于以下项目：

- **用户管理**：例如用户封禁、用户黑名单（拉黑）、用户白名单，群组及聊天室禁言、聊天室成员封禁等。
- **在线状态订阅**：将用户每一个终端在线、离线或登出后的状态，同步给应用开发者指定的服务器地址。
- **多设备在线消息同步**：同时支持桌面端、移动端、以及多个 Web 端之间的消息在线同步。
- **全量消息路由**：支持将单聊、群组、聊天室、超级群等的消息数据同步到应用开发者指定的服务器地址。
- **内容审核**：支持设置敏感词列表，过滤或替换消息中的敏感词。利用消息回调服务，可将消息先转发到应用开发者指定的服务器地址，由应用服务器判定是否可发送给目标接收者。
- **推送服务**：融云负责对接厂商推送平台，已覆盖小米、华为、荣耀、OPPO（适用于一加、realme）、vivo、魅族、FCM、APNs 手机系统级推送通道。支持标签推送、多种推送场景、推送统计、全量用户通知等特性。

> 部分功能需要在控制台开通服务后方可使用。部分为收费增值服务，详见即时通讯计费细则。

客户端 SDK

融云即时通讯（IM）客户端 SDK 提供丰富的组件与接口，大部分能力支持开箱即用。配合 IM 服务端 API 接口，可满足丰富的业务特性要求。

在集成融云 SDK 之前，我们建议使用快速上手教程与示例项目进行评估。

如何选择 SDK

IMLib 与 IMKit 是融云 IM 服务提供的两款经典的客户端 SDK。客户端功能在不同平台间基本保持一致。

- **IMLib** 是即时通讯能力库，封装了通信能力和会话、消息等对象。不含任何 UI 界面组件。

  IMLib 已支持绝大部分主流平台及框架，如 Android、iOS、Web、Flutter、React Native、Unity、微信小程序等。

- **IMKit** 是即时通讯界面库，集成了会话界面，并且提供了丰富的自定义功能。

  IMKit 已支持 Android、iOS 与 Web（要求 Web 5.X 版本）。

您可以根据业务需求进行选择：

- 基于 IMLib 开发应用，将融云即时通讯能力嵌入应用中，并自行开发产品的 UI 界面。
- 基于 IMKit 开发应用，将 IMKit 提供的界面组件直接集成到产品中，自定义界面组件功能，节省开发时间。您还可以使用融云提供的独立功能插件扩展 IMKit 的功能。

> 前往融云产品文档 · 客户端 SDK 体系 · IMLib · IMKit »

即时通讯服务端

即时通讯服务端提供一套 API 接口与多种语言的开源 SDK。

服务端 API

您可以使用服务端 API 将融云服务集成到您的即时通讯服务体系中，构建您即时通讯 App 的后台服务系统。例如，向融云获取用户身份令牌 (Token)，从 App 产品服务端向用户发送/撤回消息，或管理禁言用户列表。

> 前往融云即时通讯服务端 API 文档 · 集成必读 »

服务端 SDK

融云提供提供多个语言版本的开源服务端 SDK：

- server-sdk-java (GitHub) · (Gitee)
- server-sdk-php (GitHub) · (Gitee)
- server-sdk-go (GitHub) · (Gitee)

控制台

使用控制台，您可以对开发者账户和应用进行管理，开通高级服务，查看应用数据报表，和计费数据。

> 部分 IM 功能必须开通服务后方可使用。详见控制台服务管理页面。

即时通讯数据

如需在融云服务端长期存储单聊会话、群聊会话、聊天室会话的历史消息，您可以开通消息云存储服务。默认的长期存储时长与业务类型相关，可按需调整。该服务存储的数据仅供客户端获取历史消息时使用。

如果需要获取全部用户的消息历史，请开通 Server API 历史消息日志下载。开通后可使用服务端 API 获取最多三天的消息日志。

除此之外，您还可以开通全量消息路由服务，实时将消息同步到您的业务服务器。

您可以前往控制台的数据统计页面，查看即时通讯用户统计、业务统计、消息统计、业务健康检查等数据。开通相应服务后，还能获取如业务数据分析等数据。

融云不会利用客户的数据。同时融云提供完善的数据隐私保护策略。参见 SDK 隐私政策。

关于 Adapter 的说明

更新时间:2024-08-30

> 建议新客户使用 5x 系列版本进行集成。

Adapter 版本说明

对于过去已经集成 IMLib 4x 版本的客户，将从 2022年5月5日 起将转为使用 Adapter 方式进行支持。集成旧版 4x SDK 的客户可以通过 `RongIMLib-v4-Adapter` 无缝替换升级。未来我们将在 Adapter 上进行问题修复，但不会增加新功能。同时，原有 IMLib 4x 停止维护。

替换方式

详见升级说明。

SDK 快速集成 关于停止维护 IMLib v4 旧版 SDK 的声明

更新时间:2024-08-30

注意:

- **Web IMLib v4 版本目前已停止维护，建议您优先选择最新的 IMLib 版本。**
- 已集成 IMLib v4 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 4x SDK 的客户可以通过 `RongIMLib-v4-Adapter` 无缝替换升级。详见 升级说明。
- 未来我们将在 `RongIMLib-v4-Adapter` 上进行问题修复，但不会增加新功能。

兼容说明

Chrome	Firefox	Safari	IE	Edge	QQ 浏览器	微信 浏览器	Android
✔	✔	✔	9+	✔	✔	✔	4.4+

导入 SDK

IMLib 4.0 底层使用 Typescript 进行了重构，对 Typescript 的使用者提供了友好的类型化支持，推荐开发者使用 Typescript 进行业务开发以提升代码健壮性及可维护性。

NPM 引入（推荐）

原 `@rongcloud/imlib-v4` 包已停止维护，请使用 `@rongcloud/imlib-v4-adapter` 替代。

1. 依赖安装

```
npm rm @rongcloud/imlib-v4 && npm install @rongcloud/imlib-v4-adapter
```

2. 代码集成

```
// 非 ESModule
// const RongIMLib = require('@rongcloud/imlib-v4') 需修改为
const RongIMLib = require('@rongcloud/imlib-v4-adapter')
// ESModule
// import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
```

CDN 引入方式请参见引入 SDK。

App Key

App Key 是使用 IMLib 进行即时通讯功能开发的必要条件，也是应用的唯一性标识。在集成使用 IMLib 之前，请务必先通过控制台 注册并获取开发者的专属 App Key。

只有在 App Key 相同的情况下，不同用户之间的消息才能互通。

初始化

IMLib 提供的所有能力基于 IMLib 初始化后获取的实例对象，因此在使用 IMLib 的能力之前，**必须先调用 IMLib 的初始化接口，且务必保证该接口在应用全生命周期内仅被调用一次。**

```
// 应用初始化以获取 RongIMLib 实例对象，请务必保证此过程只被执行一次,连接类型推荐使用comet
const im = RongIMLib.init({ appkey: '<Your-App-Key>'});
```

> 后续所有代码示例中的 im 均指通过初始化获取到的 RongIMLib 实例对象

设置监听

初始化完成后，应在建立连接之前对 im 对象添加事件监听器，及时获取相关事件通知。

```
// 添加事件监听
im.watch({
// 监听会话列表变更事件
conversation (event) {
// 假定存在 getExistedConversationList 方法，以获取当前已存在的会话列表数据
const conversationList = getExistedConversationList()
// 发生变更的会话列表
const updatedConversationList = event.updatedConversationList;
// 通过 im.Conversation.merge 计算最新的会话列表
const latestConversationList = im.Conversation.merge({ conversationList, updatedConversationList })
},
// 监听消息通知
message (event) {
// 新接收到的消息内容
const message = event.message;
},
// 监听 IM 连接状态变化
status (event) {
console.log('connection status:', event.status);
},
// 监听聊天室 KV 数据变更
chatroom (event) {
/**
* 聊天室 KV 存储数据更新
* @example
* [
* {
* "key": "name",
* "value": "我是小融融",
* "timestamp": 1597591258338,
* "chatroomId": "z002",
* "type": 1 // 1: 更新 ( 含:修改和新增 ) 、2: 删除
* },
* ]
*/
const updatedEntries = event.updatedEntries
},
expansion (event) {
/**
* 更新的消息拓展数据
* @example {
* expansion: { key: 'value' }, // 设置或更新的扩展值
* messageUId: 'URIT-URIT-ODMF-DURR' // 设置或更新扩展的消息 uid
* }
*/
const updatedExpansion = event.updatedExpansion;
/**
* 删除的消息拓展数据
* @example {
* deletedKeys: ['key1', 'key2'], // 设置或更新的扩展值
* messageUId: 'URIT-URIT-ODMF-DURR' // 设置或更新扩展的消息 uid
* }
*/
const deletedExpansion = event.deletedExpansion;
}
});
```

建立 IM 连接

App Key 是应用的唯一性标识，Token 则是用户的唯一性标识，是用户连接融云 IM 服务所必需的身份凭证。Token 一般由开发者的应用服务器调用融云 Server API 获取 Token 接口获取之后，由应用服务器下发到应用客户端。

以下示例代码假定客户端已获取 Token

```
im.connect({ token: '<Your-Token>' }).then(user => {
console.log('链接成功, 链接用户 id 为: ', user.id);
}).catch(error => {
console.log('链接失败: ', error.code, error.msg);
});
```

获取会话列表

Web 端不具备持久化的数据存储能力，无法在本地持久化存储历史消息记录与会话列表，因此需要从融云服务端获取数据。从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 **IM 旗舰版**或 **IM 尊享版**可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。

该功能需要在调用 `im.connect()` 并且建立连接成功之后执行。

IMLib 通过会话数据中的 `conversationType` 与 `targetId` 两个属性值来标识会话的唯一性，对于两个属性的定义如下：

1. `conversationType` 用来标识会话类型（如：单聊、群聊...），其值为 `RongIMLib.CONVERSATION_TYPE` 中的常量定义
2. `targetId` 用来标识与本端进行对话的人员或群组 Id：

- 当 `conversationType` 值为 `RongIMLib.CONVERSATION_TYPE.PRIVATE`，`targetId` 为对方用户 Id
- 当 `conversationType` 值为 `RongIMLib.CONVERSATION_TYPE.GROUP`，`targetId` 为当前群组 Id
- 当 `conversationType` 值为 `RongIMLib.CONVERSATION_TYPE.CHATROOM`，`targetId` 为聊天室 Id

```
// 获取会话列表
im.Conversation.getList().then(conversationList => {
console.log('获取会话列表成功', conversationList);
}).catch(error => {
console.log('获取会话列表失败: ', error.code, error.msg);
});
```

发送消息

该功能需要在调用 `im.connect()` 并且建立连接成功之后执行。

IMLib 内置消息类型可通过 RongIMLib.MESSAGE_TYPE 获取其常量定义

```
// 获取指定会话的抽象实例，对于会话的操作基于此实例完成
const conversation = im.Conversation.get({
// targetId
targetId: '<TargetId>',
// 会话类型：RongIMLib.CONVERSATION_TYPE.PRIVATE | RongIMLib.CONVERSATION_TYPE.GROUP
type: '<Conversation-Type>'
});
// 向会话内发消息
conversation.send({
// 消息类型，其中 RongIMLib.MESSAGE_TYPE 为 IMLib 内部的内置消息类型常量定义
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
// 消息内容
content: {
content: 'Hello RongCloud' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
}).catch(error => {
console.log('发送文字消息失败', error.code, error.msg);
});
```

接收消息

当本端作为消息接收的一方，所接收的消息将通过 im.watch() 注册的消息监听向业务层抛出。具体可参考上述 设置监听 部分

获取历史消息

> Web 端不具备持久化的数据存储能力，无法在本地持久化存储历史消息记录与会话列表，因此需要从融云服务端获取数据。从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 **IM 旗舰版**或 **IM 尊享版**可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。
> 该功能需要在调用 im.connect() 并且建立连接成功之后执行。

```
const conversation = im.Conversation.get({
targetId: '<TargetId>',
type: '<Conversation-Type>'
});
const option = {
// 获取历史消息的时间戳，默认为 0，表示从当前时间获取
timestamp: +new Date(),
// 获取条数，有效值 1-100，默认为 20
count: 20,
};
conversation.getMessages(option).then(result => {
const list = result.list; // 获取到的消息列表
const hasMore = result.hasMore; // 是否还有历史消息可获取
console.log('获取历史消息成功', list, hasMore);
}).catch(error => {
console.log('发送文字消息失败', error.code, error.msg);
});
```

断开连接

> 断开当前用户连接，连接断开后无法接收消息、发送消息、获取历史消息、获取会话列表...
> 在下次连接融云成功后，会收取上次离线后的消息，离线消息默认保存 7 天。

```
im.disconnect().then(() => console.log('断开链接成功'));
```

引入 SDK 关于停止维护 IMLib v4 旧版 SDK 的声明

更新时间:2024-08-30

注意:

- **Web IMLib v4 版本目前已停止维护，建议您优先选择最新的 IMLib 版本。**
- **已集成 IMLib v4 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 4x SDK 的客户可以通过 `RongIMLib-v4-Adapter` 无缝替换升级。详见 升级说明。**
- **未来我们将在 `RongIMLib-v4-Adapter` 上进行问题修复，但不会增加新功能。**

兼容说明

Web

Chrome	Firefox	Safari	IE	Edge	QQ 浏览器	微信 浏览器	Android
✔	✔	✔	9+	✔	✔	✔	4.4+

导入 SDK

IMLib 4.0 底层使用 Typescript 进行了重构，对 Typescript 的使用者提供了友好的类型化支持，推荐开发者使用 Typescript 进行业务开发以提升代码健壮性及可维护性。

NPM 引入

原 `@rongcloud/imlib-v4` 包已停止维护，请使用 `@rongcloud/imlib-v4-adapter` 替代。

1. 依赖安装

```
# 如有，请移除旧版本依赖
npm rm @rongcloud/imlib-v4 @rongcloud/engine
# 安装 RongIMLib-v4-Adapter
npm install @rongcloud/engine@latest @rongcloud/imlib-v4-adapter@latest -S
```

2. 代码集成

```
// 非 ESModule
// const RongIMLib = require('@rongcloud/imlib-v4') 需修改为
const RongIMLib = require('@rongcloud/imlib-v4-adapter')
// ESModule
// import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
```

CDN 引入

推荐使用 RongIMLib-v4-Adapter 的最新版，如下：

```
<script src="https://cdn.ronghub.com/RongIMLib-v4-Adapter-5.9.5.prod.js"></script>
```

初始化 功能描述

更新时间:2024-08-30

IMLib 提供的所有能力基于 IMLib 初始化后获取的实例对象，因此在使用 IMLib 的能力之前，**必须先调用 IMLib 的初始化接口，且务必保证该接口在应用全生命周期内仅被调用一次。**

> ⚠ 警告
>
> App Key 是使用 IMLib 进行即时通讯功能开发的必要条件，也是应用的唯一性标识。在集成使用 IMLib 之前，请务必先通过 控制台 注册并获取开发者的专属 App Key。
>
> 只有在 App Key 相同的情况下，不同用户之间的消息才能互通。

API 参考：init

代码示例

```
// 应用初始化以获取 RongIMLib 实例对象，请务必保证此过程只被执行一次
const im = RongIMLib.init({ appkey: '<Your-App-Key>' });
```

设置监听 设置监听

更新时间:2024-08-30

初始化完成后，应在建立连接之前对 im 对象添加事件监听器，及时获取相关事件通知。

> ⚠ 警告
>
> 当多次调用 im.watch 对同一事件进行监听时，只有最新添加的监听函数才会被执行。

API 参考：watch

参数说明

参数	类型	必填	说明	最低版本
message	Function	否	接收消息时触发，结构体参考	3.0.0
status	Function	否	IM 链接状态变更时触发，链接状态码说明 **5.7.0** 版本将废弃该方法	3.0.0
connection	Function	否	IM 链接状态变更时触发，链接状态码说明	5.7.0
conversation	Function	否	会话列表有变更时触发	3.0.0
chatroom	Function	否	聊天室 KV 数据变化时触发 说明：从 4.5.0 开始，支持在用户加入、退出聊天室发送通知，该功能需要提交工单申请开通	3.0.6
expansion	Function	否	消息扩展数据变更时触发	3.0.7
pullFinished	Function	否	离线消息拉取完成时触发	4.5.0
messageBlocked	Function	否	本端发送的消息如果被融云服务端屏蔽，触发该监听	4.5.0
tag	Function	否	当用户在其它端添加移除更新标签时会触发,该回调不会给当前操作设备回调，只会给其他的多端设备回调	4.5.0

代码示例

1. 所有代码示例中的 `im` 均指通过 `RongIMLib.init()` 获取的实例对象
2. 以下示例中假定存在 `getExistedConversationList` 方法，以获取当前已存在的会话列表数据

```
// 添加事件监听
im.watch({
// 监听会话列表变更事件，触发时机：会话状态变化（置顶、免打扰）、会话未读数变化（未读数增加、未读数清空）、会话 @ 信息、会话最后一条消息变化
conversation (event) {
// 假定存在 getExistedConversationList 方法，以获取当前已存在的会话列表数据
const conversationList = getExistedConversationList()
// 发生变更的会话列表
const updatedConversationList = event.updatedConversationList;
```

```
// 通过 im.Conversation.merge 计算最新的会话列表
const latestConversationList = im.Conversation.merge({ conversationList, updatedConversationList })
},
// 监听消息通知
message (event) {
// 新接收到的消息内容
const message = event.message;
},
// 监听 IM 连接状态变化， 5.6.1 版本之前如此，5.7.0 版本之后该监听废弃，见下方使用说明
status (event) {
console.log('connection status:', event.status);
},
// 监听聊天室 KV 数据变更，进入、退出聊天室，销毁聊天室
chatroom (event) {
/**
* 聊天室 KV 存储数据更新
* @example
* [
* {
* "key": "name",
* "value": "我是小融融",
* "timestamp": 1597591258338,
* "chatroomId": "z002",
* "type": 1 // 1: 更新 ( 含:修改和新增 ) 、2: 删除
* },
* ]
*/
const updatedEntries = event.updatedEntries
},
expansion (event) {
/**
* 更新的消息扩展数据
* @example {
* expansion: { key: 'value' }, // 设置或更新的扩展值
* messageUId: 'URIT-URIT-ODMF-DURR' // 设置或更新了扩展的消息 uid
* }
*/
const updatedExpansion = event.updatedExpansion;
/**
* 删除的消息扩展数据
* @example {
* deletedKeys: ['key1', 'key2'], // 删除的扩展值
* messageUId: 'URIT-URIT-ODMF-DURR' // 删除了扩展的消息 uid
* }
*/
const deletedExpansion = event.deletedExpansion;
},
// 监听离线消息拉取完成
pullFinished () {
console.log('拉取离线消息完成')
},
// 本端发送的消息被屏蔽时触发
messageBlocked (event) {
const messageId = event.blockedMessageUId
const conversationType = event.conversationType
const targetId = event.targetId
const blockType = event.blockType
},
// 该回调不会给当前操作设备回调，只会给其他的多端设备回调
// 当用户在其它端添加移除更新标签时会触发此监听器，用于多端之间的信息同步
tag(){
console.log('标签发生变化')
}
});
```

5.7.0 版本之后连接状态监听说明

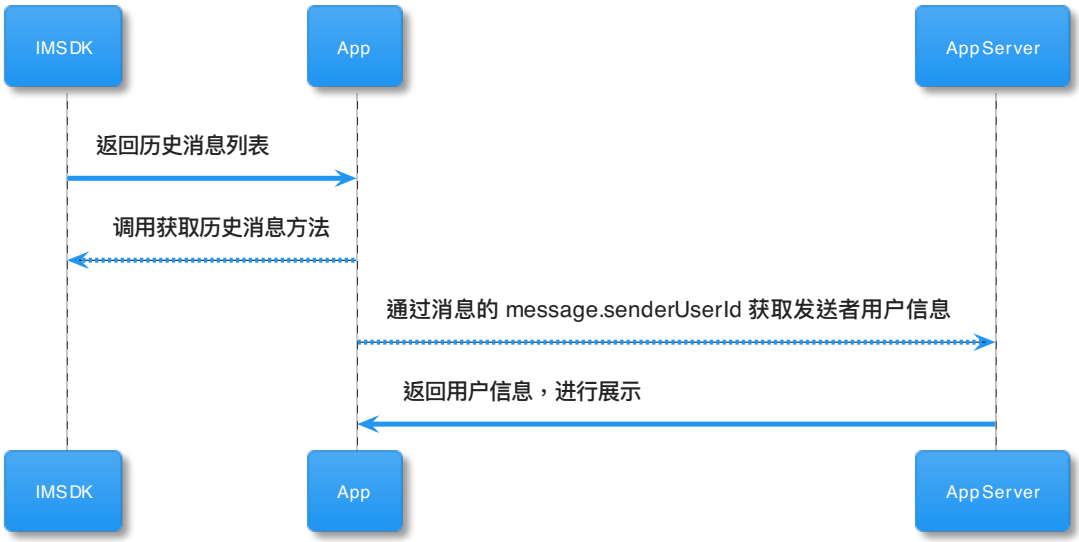
```
im.watch({
// 监听 IM 连接状态变化
connection (event) {
// status 标识当前连接状态， code 表示连接断开原因
switch (event.status) {
case RongIMLib.RCConnectionStatus.CONNECTED:
console.log('连接成功');
break;
case RongIMLib.RCConnectionStatus.CONNECTING:
console.log('正在连接');
break;
case RongIMLib.RCConnectionStatus.DISCONNECTED:
console.log('断开连接, 错误码：' + event.code);
break;
case RongIMLib.RCConnectionStatus.SUSPENDED:
// SDK 内部会重连
console.log('连接断开，内部重连中，错误码：' + event.code);
break;
},
});
```

显示用户信息 在历史消息中显示

更新时间:2024-08-30

从 App Server 获取用户信息

1. 您的 App Server 封装获取用户信息接口。
2. 通过 message.senderUserId 获取发送者 ID。
3. 将发送者 ID 传入 App Server 暴露的接口中，以获取对应的用户信息。
4. 将用户信息展示到页面中。



发消息时携带用户信息

> 携带的用户信息保存在消息中。如果用户修改了用户信息，已经发送的消息携带的用户信息不会同步更新。

1. 获取当前用户(也就是发送者)的用户信息。
2. 发消息时携带当前用户信息。
3. 展示消息时，通过消息体内的用户信息进行展示。

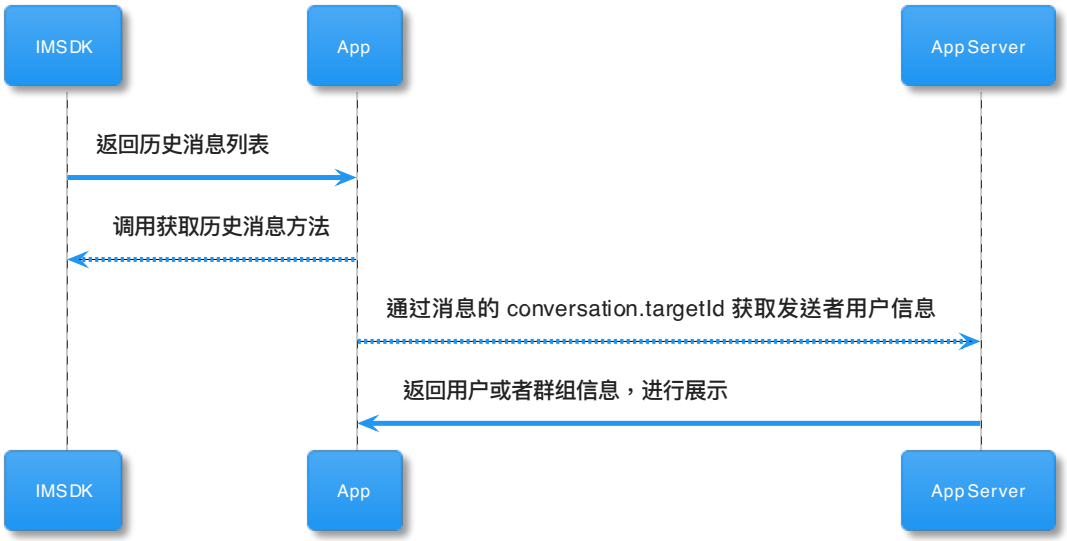
代码示例

```
const conversation = im.Conversation.get({
targetId: 'user1',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: 'Hello RongCloud', // 文本内容
user: {
id: 'user2',
name: '用户二',
portraitUri: 'https://www.rongcloud.cn/images/newVersion/log_wx.png'
},
extra: '' // 附加信息
}
}).then(message => {
console.log('发送文字消息成功', message);
});
```

在会话列表中显示

通过 App Server 获取用户信息。

1. 您的 App Server 封装获取用户或群组信息接口
2. 通过 conversation.targetId 获取用户 ID 或者群组 ID。
3. 将用户或者群组 ID 传入 App Server 暴露的接口中，获取对应用户或群组信息
4. 将信息展示到页面中。



连接服务 功能描述

更新时间:2024-08-30

在应用的整个生命周期，此方法只需要调用一次，之后无论是网络异常或者 App 有前后台的切换，SDK 都会自动重连，直到开发者主动断开连接。

Token 是用户连接融云 IM 服务所必需的身份凭证。Token 一般由开发者的应用服务器调用融云 Server API 获取 Token 接口获取之后，由应用服务器下发到应用客户端。

> ⚠ 警告
>
> 1. 连接方法必须在执行初始化之后调用。
> 2. 除初始化、监听以外，所有方法都必须在 connect 连接成功之后 再调用。
> 3. 默认一个用户只支持一个页面连接，开通 多设备消息同步 可支持多页面连接。

API 参考：connect

代码示例

> 以下示例代码假定客户端已获取 Token

```
im.connect({ token: '<Your-Token>' }).then(user => {
console.log('链接成功, 链接用户 id 为: ', user.id);
}).catch(error => {
console.log('链接失败: ', error.code, error.msg);
});
```

链接失败错误码说明

重连逻辑

更新时间:2024-08-30

重连说明

SDK 内已实现重连机制，在应用的整个生命周期内，开发者只需要调用一次 im.connect() 建立连接。当网络异常中断时，SDK 内部会尝试重新建立连接，业务层无需进行其他操作

当业务层使用 im.disconnect() 主动断开连接后，希望以断开前的身份重新进行连接时，可使用 im.reconnect() 进行重新连接：

```
im.reconnect().then(user => {
console.log('重新链接成功, 链接用户 id 为: ', user.id);
}).catch(error => {
console.log('链接失败: ', error.code, error.msg);
});
```

在 v4.4.4 及以上版本中，调用 im.reconnect() 前需先调用 im.disconnect() 方法，否则会报错误：35007。

链接失败错误码说明

断开连接 断开连接

更新时间:2024-08-30

业务层通过调用 im.disconnect() 主动断开连接，连接断开后，业务层将不会再接收消息通知，且无法发送消息、拉取历史消息、获取会话列表。

> 连接断开后所有发送至该用户的消息被定义为离线消息，当用户重新建立连接后，SDK 会拉取最多 7 天的离线消息补发给业务层。

连接断开后，业务可通过调用 im.reconnect() 方法以断开前的身份重新建立连接。

API 参考：disconnect

代码示例

```
im.disconnect().then(() => console.log('断开链接成功'));
```

多端同时在线

更新时间:2024-08-30

默认的情况下，融云仅支持 1 个 Web 端、1 个 桌面端、1 个移动端同时在线。

开通多设备消息同步功能后，可以支持 Web 端、桌面端和移动端之间的消息同步。且开通此功能后，可以同时支持多个 Web 端同时在线。重新登录时，获取当天收发的所有消息。

开通方式

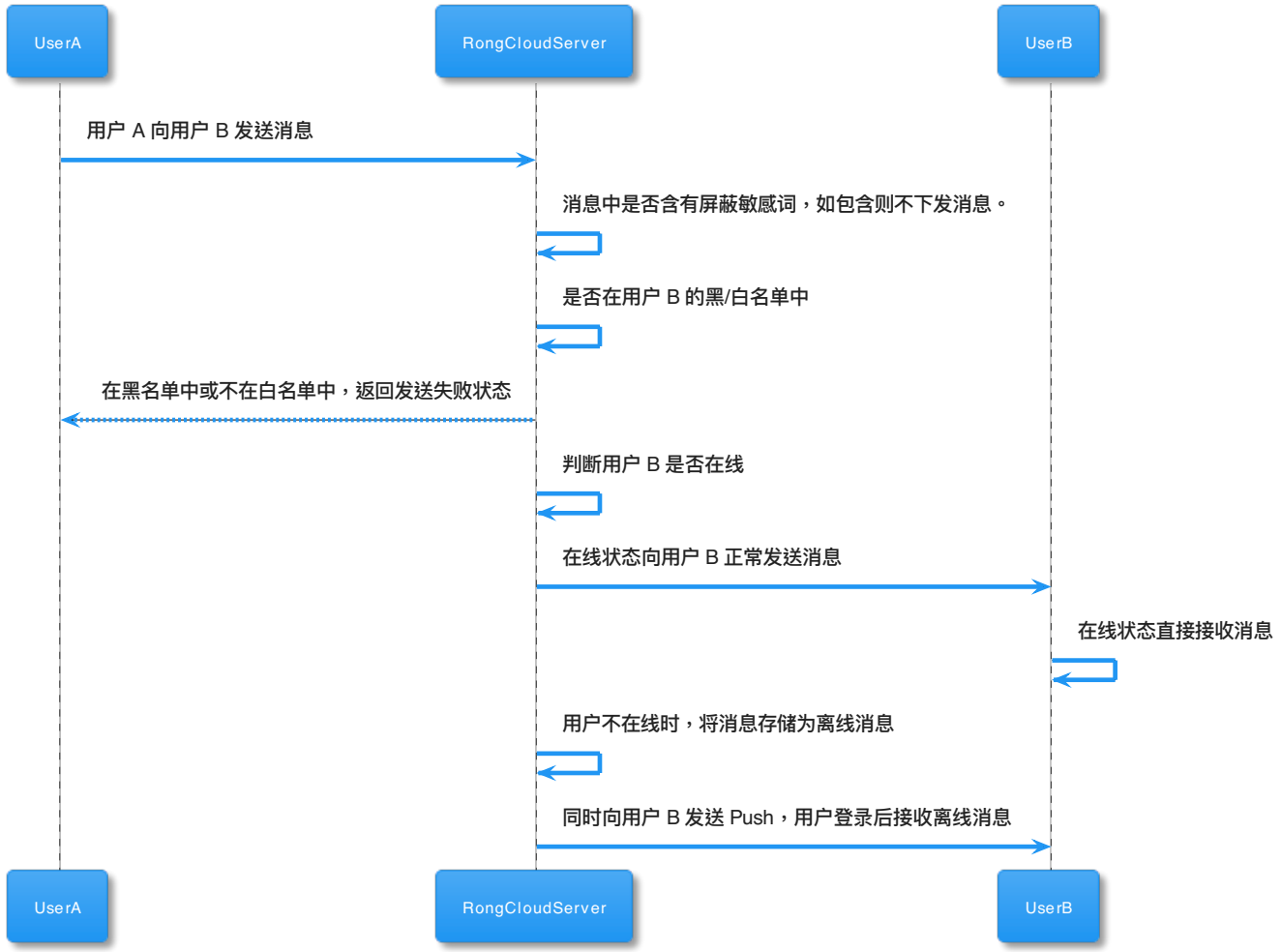
- 开发环境，默认为关闭状态，开启后 30 分钟内生效。
- 生产环境，**IM 旗舰版**或**IM 尊享版**可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。服务开启、关闭设置完成后 30 分钟内生效。

单聊介绍 概述

更新时间:2024-08-30

适用于应用内两个用户之间一对一聊天方式，两个用户间可以是好友也可以是陌生人，融云不对用户的关系进行维护管理，会话关系由融云负责建立并保持。

消息发送时序图：



主要功能

功能	描述
离线消息	支持离线消息存储，存储时间可设置（1 ～ 7 天），默认存储 7 天。
消息提醒	离线状态，单聊中有新消息时，支持 Push 通知。
本地存储	存储在移动端本地，提供本地消息搜索功能。
历史消息	提供服务端消息存储存功能，需开通单群聊消息云存储，默认存储时长为 6 个月。
消息删除	支持按会话删除本地和存储在服务器的指定消息或会话中全部历史消息。

功能	描述
消息搜索	支持按关键字或用户搜索本地指定会话的消息内容。
消息阅读回执	发送单聊消息后如需要查看消息的阅读状态，可以使用此功能来发送阅读回执请求。
消息撤回	消息发送成功后，在有效时间内可撤回该条消息，默认可撤回时间为 2 分钟，时间可配置。
单聊会话免打扰	可设置指定的单聊会话，收到新的消息后是否进行提醒，默认进行新消息提醒。
单聊黑名单	不想接收到某一用户的消息时，可将此用户加入到黑名单中，应用中的每个用户都可以设置自己的黑名单列表
单聊白名单	对用户之间相互发送消息有限制要求的客户，可使用用户白名单功能，将用户加入白名单后，才能收到该用户发送的单聊消息

注：用户白名单服务与用户黑名单服务不能同时使用，融云默认开启的是用户黑名单服务，如需要开通白名单服务请提交工单申请开通。服务开通 30 分钟后生效，同时黑名单服务不再生效。

消息类型

消息类型	描述
文字消息	用来发送文字类消息，其中可以包括表情、超链接（会自动识别），客户端收到消息后计入未读消息数、进行存储。
语音消息	发送高质量的短语音消息，录制的语音文件存储到融云服务端，语音文件格式为 AAC，时长上限为 60 秒，客户端收到消息后计入未读消息数、进行存储。
图片消息	用来发送图片类消息，客户端收到消息后计入未读消息数、进行存储。图片缩略图格式为 JPG，大小建议不超过 100k。
GIF 图片消息	用来发送 GIF 动态图片消息，客户端收到消息后计入未读消息数、进行存储。
图文消息	用来发送图文消息，包含一个标题，一段文字内容和一张图片，客户端收到消息后计入未读消息数、进行存储。
文件消息	用来发送文件类消息，客户端收到消息后计入未读消息数、进行存储。
位置消息	用来发送地理位置消息，客户端收到消息后计入未读消息数、进行存储。
小视频消息	用来发送小视频消息，支持录制发送及选择本地视频文件发送两种方式，录制时长不超过 10 秒，本地选择视频文件方式时长不超过 2 分钟，小视频消息小视频文件格式为 .mp4，客户端收到消息后计入未读消息数、进行存储。
合并转发消息	IMKit SDK 中支持将多条消息合并为一条消息进行发送，合并后的消息以 HTML 文件的方式存储到融云服务端，客户端收到消息后计入未读消息数、进行存储。红包、阅后即焚及自定义消息的合并转发功能
命令消息	用来发送通用的指令通知消息，消息内可以定义任意 JSON 内容，与通用命令通知消息的区别是不存储、不计数，此类型消息没有 Push 通知。
自定义消息	融云内置消息类型，无法满足客户业务需求时，可通过自定义消息类型进行实现，接收自定义消息的格式解析及展示处理需要开发者自行实现

内置消息说明

请参见融云服务端文档内置消息类型说明。

好友关系

更新时间:2024-08-30

融云为了客户隐私考虑，既不同步又不保存用户的好友关系。所以，所有用户的好友关系都需要开发者自己实现、管理维护。

如果使用的是 IMKit 进行集成，则会话及好友列表中显示好友的昵称及头像信息，需要 App 设置一个用户信息提供者给 IMKit，以便 IMKit 通过用户信息提供者，来实现在聊天界面和会话列表页中显示好友的昵称和头像。

获取全部会话　　获取本地会话列表

更新时间:2024-08-30

Web 端不具备持久化的数据存储能力，故无法提供获取本地会话列表相关功能

获取远端会话列表

> Web 端不具备持久化的数据存储能力，无法在本地持久化存储历史消息记录与会话列表，因此需要从融云服务端获取数据。从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 **IM 旗舰版**或 **IM 尊享版**可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。
> 该功能需要在调用 `im.connect()` 并且建立连接成功之后执行。
> 服务器会话列表存储上限为 `1000` 条会话

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
count	Number	否	300	想要获取的会话列表数量，默认值 `300`，最大值 `1000`	3.0.0
startTime	Number	否	0	获取会话列表所使用的时间戳，需要精确到毫秒	3.0.7.1
order	Number	否	0	会话排序方式	3.0.7.1

`order` 说明：

value	说明
0	按会话生成时间倒序排列，即获取早于 startTime 生成的会话列表
1	按会话生成时间正序排列，即获取晚于 startTime 生成的会话列表

`startTime` 补充说明：

- startTime 为 0 且 order 为 0 时, startTime 表示当前时间点，接口将返回最新的会话列表
- startTime 为 0 且 order 为 1 时, startTime 表示会话列表的创建时间，即最早的一条消息的产生时间，接口将返回最早的会话列表数据

回调参数说明

回调参数	类型	说明
conversationList	Array	会话列表，会话参数说明请参照 conversation 属性说明

- conversation 属性说明：

字段名	类型	说明
type	Number	会话类型，见 type 枚举值说明
targetId	String	接收方的 userId
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息，详见 latestMessage
hasMentioned	Boolean	是否包含 @ 自己的消息
mentionedInfo	Object	@ 信息，见 mentionedInfo 结构说明
notificationStatus	Number	当前会话免打扰状态
isTop	Boolean	当前会话免置顶状态

- type 枚举值说明

会话类型	说明	枚举值
RongIMLib.CONVERSATION_TYPE.PRIVATE	单聊	1
RongIMLib.CONVERSATION_TYPE.GROUP	群聊	3
RongIMLib.CONVERSATION_TYPE.CHATROOM	聊天室	4
RongIMLib.CONVERSATION_TYPE.SYSTEM	系统	6

- mentionedInfo 结构说明：

字段名	类型	说明
type	Number	@ 类型，1: @ 所有人， 2: @ 部分人
userIdList	Array	被 @ 的用户 id 列表

代码示例

```
im.Conversation.getList({
count: 30,
startTime: 0,
order: 0
}).then(conversationList => {
console.log('获取会话列表成功', conversationList);
});
```

删除全部会话　　清除会话列表

更新时间:2024-08-30

SDK 不提供批量删除接口，如需事先清除多个会话，可循环调用 删除指定会话 接口来实现。

获取指定会话

更新时间:2024-08-30

可以通过获取会话列表得到会话列表信息，通过 Conversation.get 获取会话实例进行消息发送等操作。

参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	接收方的 userId	3.0.0
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	3.0.0

代码示例

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
const conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
```

conversation 属性说明：

字段名	类型	说明
type	Number	会话类型
targetId	String	接收方的 userId

删除指定会话　　删除指定会话

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 `conversation.destory()` 删除会话。

> 当收到新消息或者发送消息时会重新生成该会话。

代码示例

```
// conversation 会话实例
const conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.destory().then(() => console.log('删除会话成功'));
```

会话草稿信息 获取草稿

更新时间:2024-08-30

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	接收方的 userId	4.0.0
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.getDraft().then(function (draft) {
console.log('获取指定会话草稿成功', draft)
})
```

保存草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	接收方的 userId	4.0.0
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	4.0.0

参数说明

参数	类型	必填	说明	最低版本
draft	String	是	草稿内容	4.0.0

代码示例

```
var draft = '这是会话草稿内容';
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type:RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.setDraft(draft).then(function () {
console.log('设置指定会话草稿成功')
})
```

删除草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	接收方的 userId	4.0.0
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.deleteDraft().then(function () {
console.log('删除指定会话草稿成功')
})
```

会话未读数　　获取所有会话未读数

更新时间:2024-08-30

会话未读数指某一个会话中未读消息的数量

1. 清除浏览器缓存会导致会话未读数不准确
2. 会话消息未读数存储在 WebStorage 中, 若浏览器不支持或禁用 WebStorage，未读消息数将不会保存，浏览器页面刷新未读消息数将不会存在

参数说明

参数	类型	必填	说明	最低版本
includeMuted	Boolean	否	是否包含免打扰会话，默认为: false	4.4.5
conversationTypes	Array	否	会话类型	4.4.5

代码示例

```
const conversationTypes = [RongIMLib.CONVERSATION_TYPE.SYSTEM]
im.Conversation.getTotalUnreadCount(false, conversationTypes).then(function(
totalUnreadCount
) {
console.log('获取未读总数成功', totalUnreadCount)
})
```

清除全部会话未读数

最低版本：4.5.4

```
im.Conversation.readAll()
```

清除单个会话未读数

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.read().then(function(){
console.log('清除未读数成功');
});
```

获取指定会话未读数

代码示例

```
let conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});

conversation.getUnreadCount().then(function(count) {
console.log('获取指定会话未读数成功', count);
})
```

多端同步未读数

未读消息存在 localStorage 中，未读消息数是针对当前端的未读消息数，服务器不存未读消息数量。

实现方案

1. 调用 conversation.read() 清除未读数。
2. 清除成功后发送 `RC:SRSMsg` 类型消息进行未读数同步。
3. 其他端接受到 `RC:SRSMsg` 类型消息，调用 `conversation.read()` 方法进行本地未读数清除。（在 v4.5.4 版本之后收到该消息时 sdk 内部会清理未读数，无需用户主动调用）

代码示例

清除端

```
// 清除未读数
let conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});

conversation.read().then(function() {
console.log('清除指定会话未读数成功' );
// 发送同步消息
conversation.send({
messageType: 'RC:SRSMsg',
content: {
//从消息里获取服务器端时间，以最近一条已读 message 为准
lastMessageSendTime: message.sentTime
}
}).then(function(message){
console.log('发送同步消息成功', message);
});
});
```

同步端

```
// 其他端在消息监听中接收到同步消息后，调用清除未读数做更新处理
// 收到同步消息进行未读数清除操作 调用 conversation.getUnreadCount() 成功后不需要再在发送 `RC:SRSMsg` 类型消息。
let conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});

conversation.read().then(function(count) {
console.log('获取指定会话未读数成功', count);
});
```

会话免打扰　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	3.0.0
targetId	String	是	接收方的 userId	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
let statusItem = {
notificationStatus: 1,
};
conversation.setStatus(statusItem).then(function() {
console.log('设置免打扰成功');
})
```

会话置顶　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，单聊 传入 RongIMLib.CONVERSATION_TYPE.PRIVATE	3.0.0
targetId	String	是	接收方的 userId	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE,
})
let statusItem = {
isTop: true,
}
conversation.setStatus(statusItem).then(function() {
console.log('设置置顶成功')
})
```

会话标签

更新时间:2024-08-30

> SDK 从 4.5.0 版本开始支持会话标签功能。

会话标签用于对会话进行分组。用户设置会话标签，用于会话的分组显示。

用户可以按照标签获取会话列表，按照标签获取标签下所有会话的消息未读数，在特定标签下设置某个会话置顶。

每个用户最多可以创建 20 个标签，每个标签下最多可以添加 1000 个会话。

同一个会话可以设置多个不同的标签。

标签管理功能接口

创建

```
im.Tag.create({'tagId','tagName'}).then(code => {
// 创建标签成功
if(code === 0) {

}
})
```

移除

```
im.Tag.get(tagId).then(tag => {
tag.remove().then(data => {
// 删除标签成功
if(code === 0) {

}
})
})
```

编辑

```
im.Tag.get(tagId).then(tag => {
tag.update('tagName').then(data => {
// 更新标签成功
if(code === 0) {

}
})
})
```

获取标签列表

```
im.Tag.getTagInfoList().then(result => {
// 获取标签列表
if(!result.code) {
console.log(result)
}
})
```

会话标签功能

添加会话到标签

```
const type = RongIMLib.ConversationType.PRIVATE
// 会话数组
const conversations = [{type, targetId: 'targetId'}]
im.Tag.get(tagId).then(tag => {
tag.addConversations(conversations).then(code => {
// 添加成功
if(code === 0) {

}
})
})
```

删除标签中某些会话

```
const type = RongIMLib.ConversationType.PRIVATE
// 会话数组
const conversations = [{type, targetId: 'targetId'}]
im.Tag.get(tagId).then(tag => {
tag.removeConversations(conversations).then(code => {
// 删除成功
if(code === 0) {

}
})
})
```

删除会话中的标签

```
const type = RongIMLib.ConversationType.PRIVATE
im.Conversation.get({type,'targetId'}).removeTags(['tagId']).then(code => {
// 删除成功
if(code === 0) {

}
})
```

获取会话下的标签

```
const type = RongIMLib.ConversationType.PRIVATE
im.Conversation.get({type,'targetId'}).getTags().then(result => {
// 获取标签列表
if(!result.code) {
console.log(result)
}
})
```

分页获取指定标签下会话列表

```
// 开始时间
const startTime = 0
const count = 10
im.Tag.get(tagId).then(tag => {
tag.getConversationList(startTime,count).then(result => {
// 获取会话列表
if(!result.code) {
console.log(result)
}
})
})
```

按标签获取未读消息数

```
// 是否包含免打扰
const isIncludeNotNotification = true
im.Tag.get(tagId).then(tag => {
tag.getUnreadCount(isIncludeNotNotification).then(result => {
// 获取未读消息数
if(!result.code) {
console.log(result)
}
})
})
```

设置标签中会话置顶

> ⚠ 警告
>
> 注意：此处是在标签下会话的置顶，而不是会话的置顶。【这个属性在根据标签获取的会话中可以看到】

```
// 会话数组
const conversation = {type: RongIMLib.CONVERSATION_TYPE.PRIVATE,targetId: 'targetId'}
const isTopInTag = true
im.Tag.get(tagId).then(tag => {
tag.updateConversationIsTop(conversation, isTopInTag).then(code => {
// 更新成功
if(code === 0) {

}
})
})
```

消息发送 属性描述

更新时间:2024-08-30

消息属性	描述
消息类名	各端消息名
存储属性	存储 / 不存储
离线属性	缓存 / 不缓存
推送属性	是/否
ObjectName	传输层名称，与消息类名一一对应
计数属性	计数 / 不计数
消息尺寸	128 KB
推送内容	详见各消息类送方式

存储属性

存储属性	存储分类	支持平台	详细描述
存储	客户端	Android、iOS	发送、接收该消息后，本地数据库存储 Web 端 和 小程序端因本地存储不可靠，不支持客户端消息存储，但可通过历史消息云存储服务获取历史记录
存储	云端	Android、iOS、Web	默认不在云端进行存储，需开通单群聊消息云存储服务，开通后，可在融云服务端存储 6 个月的历史消息，供客户端按需拉取
不存储	客户端	Android、iOS	发送、接收该消息后，本地数据库不存储
不存储	云端	Android、iOS、Web	无论是否开通历史消息云存储服务，该消息均不存储

计数属性

接收收到消息时，会话是否累计未读数。

计数属性	支持平台	详细描述
计数	iOS、Android、Web	会话未读消息数 + 1，该属性只影响会话列表未读数计数，App 应用角标可根据每个会话列表未读数累加获得
不计数	iOS、Android、Web	会话未读消息数不变

离线属性

接收人当前不在线时，是否进行离线缓存。

离线属性	详细描述
存储	消息进行离线缓存，默认 7 天。接收人在 7 天内上线，均可接收到该消息。超过 7 天后，消息被离线缓存淘汰。如有需要，可通过云端存储拉取到该消息
不存储	消息不进行离线缓存，所以只有接收人在线时，才可收到该消息。该消息不进行历史消息云存储、不进入云端存储（Log 日志）、不进行消息同步（消息路由）

推送属性

接收人是否接收推送，当离线属性为 存储 时，该属性生效。离线属性为 不存储 时属性无效。

由于 Web、小程序、PC 端没有推送平台，无法收到推送提醒。

推送属性	平台	推送方式	详细描述
推送	iOS、Android	APNs、华为、小米、魅族、OPPO、vivo、FCM、融云	当有离线缓存消息时，进行远程推送提醒，内容为该推送提醒显示的内容
不推送	iOS、Android	--	当有离线缓存消息时，不进行远程推送提醒

⚠ 警告

1. 发送消息必须在成功连接融云服务器成功之后进行。
2. 每秒最多发送 5 条消息。

消息结构

字段名	类型	说明
type	Number	会话类型
targetId	String	接收方的 userId
senderUserId	String	发送者 id
content	Object	消息内容
messageType	String	消息标识
messageUId	String	服务端存储的消息 id
messageDirection	Number	消息方向。 发送: 1, 接收: 2
isOffLineMessage	Boolean	是否为离线消息
sentTime	Number	消息在融云服务端的发送时间
receivedTime	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
isPersited	Boolean	消息是否存储在服务端
isCounted	Boolean	消息是否计数
disableNotification	Boolean	消息是否静默，静默消息不会发送 Push 信息和本地通知提醒

参数说明

参数	类型	必填	默认值	说明	最低版本
messageType	String	是	--	消息类型. 与移动端 ObjectName 一致	3.0.0
content	Object	是	--	消息内容	3.0.0
isPersited	Boolean	否	true	是否存储在服务端	3.0.0
isCounted	Boolean	否	true	是否计数. 计数消息接收端接收后未读数加 1	3.0.0
pushContent	String	否	--	Push 显示内容	3.0.0
pushData	String	否	--	Push 通知时附加信息	3.0.0
isVoipPush	String	否	false	为 ture 时, 对端不在线的 iOS 会收到 Voip Push. Android 无影响	3.0.0
isStatusMessage	Boolean	否	false	是否发送状态消息，设置为 **true** 后 **isPersited** 和 **isCounted** 属性失效	3.0.0
disableNotification	Boolean	否	false	是否发送静默消息，设置为 **true** 后不会发送 **Push** 信息和本地通知提醒	3.0.5

发送代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
},
isPersited: true,// 是否存储在服务端,默认为 true
isCounted: true, // 是否计数. 计数消息接收端接收后未读数加 1，默认为 true
pushContent:'user 发送了一条消息', // Push 显示内容
pushData: 'Push 通知时附加信息', // Push 通知时附加信息, 可不填
isStatusMessage: false , // 设置为 true 后 isPersited 和 isCounted 属性失效
disableNotification: false, // 设置为 true 后移动端不会收到 Push 信息和本地通知提醒
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

发送消息

文本消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:TxtMsg	存储	计数	存储	推送	消息内容

参数说明

属性名称	属性类型	是否必填	属性说明
content	String	是	文本消息内容
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: 'Hello RongCloud' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

图文消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ImgTextMsg	存储	计数	存储	推送	消息内容

参数说明

属性名称	属性类型	是否必填	属性说明
content	String	是	消息的文字内容
title	String	是	消息的标题
imageUri	String	是	图片上传到服务器的 url
url	String	是	富文本消息点击后打开的 URL
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.RICH_CONTENT, // 'RC:ImgTextMsg'
content: {
title: '消息的标题',
content: '消息的文字内容',
imageUri: '图片上传到服务器的 url',
url: '富文本消息点击后打开的 URL'
}
}).then(function(message){
console.log('发送图文消息成功', message);
});
```

Emoji 消息

- web 端发送 Emoji 消息，开发者直接使用 文本消息 发送即可。
- 融云提供 Emoji 插件，内置了 128 个 Emoji 表情的图片，做消息输入框的表情选项，也可自行扩展配置。
- 发消息时，必须直接发送 Emoji 原生字符. 如：□，转换方法： `symbolToEmoji`。
- Web SDK 接收消息时接收到的是 Unicode 编码格式，如：”ef600” 需要转化才能正确显示原生 Emoji。

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: '□' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

Emoji 插件

插件兼容性

Chrome	Firefox	Safari	IE	Edge	iPhone	Android
30+	30+	10+	7+	✔	iOS 8.0+ 的Safari浏览器以及微信浏览器	4.4+系统的Chrome浏览器以及微信浏览器

Emoji 插件引入

```
<!-- 非压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- 压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.min.js"></script>
```

Emoji 代码示例：https://rongcloud.github.io/web-emoji-demo/src/index.html

⚠ 警告

1. 使用 `import * as RongIMLib from '@rongcloud/imlib-v4-adapter'` 方式引入 SDK 时表情插件调用需要使用 `window` 前缀。例如：`window.RongIMLib.RongIMEmoji.init()`
2. RongEmoji 插件仅支持 cdn 引入方式，暂不支持 npm 引入

Emoji 初始化：默认参数初始化

```
RongIMLib.RongIMEmoji.init();
```

Emoji 初始化：自定义表情配置初始化

`config` 参数说明：

参数	类型	必填	说明	最低版本
size	Number	否	表情大小, 默认 24, 建议 18 - 58	2.2.6
url	String	否	Emoji 背景图片 url	2.2.6
lang	String	否	Emoji 对应名称语言, 默认 zh	2.2.6
extension	Object	否	扩展表情	2.2.6

```
// 表情信息可参考 http://unicode.org/emoji/charts/full-emoji-list.html
var config = {
size: 25,
url: '//f2e.cn.ronghub.com/sdk/emoji-48.png',
lang: 'en',
extension: {
dataSource: {
u1F914: { // 自定义 u1F914 对应的表情
en: 'thinking face', // 英文名称
zh: '思考', // 中文名称
tag: '', // 原生 Emoji
position: '0 0' // 所在背景图位置坐标
}
},
url: '//cdn.ronghub.com/thinking-face.png' // 新增 Emoji 背景图 url
}
};
RongIMLib.RongIMEmoji.init(config);
```

获取列表

```
var list = RongIMLib.RongIMEmoji.list;
/*list => [{
unicode: 'u1F600',
emoji: '',
node: span,
symbol: '[笑嘻嘻]'
}]
*/
```

Emoji 转文字

在不支持原生 Emoji 渲染时，可显示对应名称，适用于消息输入。

```
var message = '😁测试 Emoji';
// 将 message 中的原生 Emoji 转化为对应名称
RongIMLib.RongIMEmoji.emojiToSymbol(message);
// => '[笑嘻嘻][露齿而笑]测试 Emoji'
```

文字转 Emoji

发送消息时，消息体里必须使用原生 Emoji 字符。

```
var message = '[笑嘻嘻][露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为原生 Emoji
RongIMLib.RongIMEmoji.symbolToEmoji(message);
// => '😁测试 Emoji'
```

Emoji 转 HTML

Web SDK 接收消息后，消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示。

```
var message = '\uf600测试 Emoji';
// 将 message 中的原生 Emoji（包含 Unicode ）转化为 HTML
RongIMLib.RongIMEmoji.emojiToHTML(message);
// => "<span class='rong-emoji-content' name='[笑嘻嘻]'></span>测试 Emoji"
```

文字 转 HTML

```
var message = '[露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为 HTML
RongIMLib.RongIMEmoji.symbolToHTML(message);
// => "<span class='rong-emoji-content' name='[露齿而笑]'>😁</span>测试 Emoji"
```

位置消息

消息说明

⚠ 警告

messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:LBSMsg	存储	计数	存储	推送	[位置]

参数说明

属性名称	属性类型	是否必填	属性说明
longitude	Number	是	经度
latitude	Number	是	纬度
poi	String	是	位置信息
content	String	是	位置缩略图,图片需要是不带前缀的 base64 字符串
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
var latitude = 40.0317727;
var longitude = 116.4175057;
var poi = '北苑路 融云';
var content = '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE'; // 位置图片 base64
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.LOCATION, // 'RC:LBSMsg'
content: {
latitude: latitude,
longitude: longitude,
poi: poi,
content: content
}
}).then(function(message){
console.log('发送位置消息成功', message);
});
```

正在输入状态消息

消息说明

⚠ 警告

messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:TypSts	不存储	不计数	不存储	不推送	无

参数说明

属性名称	属性类型	是否必填	属性说明
typingContentType	String	是	正在输入的消息 ObjectName
data	String	否	携带信息

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: 'RC:TypSts',
content: {
typingContentType: 'RC:TxtMsg' // 正在输入的消息类型
},
isStatusMessage: true // 正在输入建议发送状态消息，不存储、不计数，且仅在线用户可收到
}).then(function(message){
console.log('发送正在输入消息成功', message);
});
```

图片消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。
- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。
- 小程序、uni-app 必须使用各平台官方的上传插件。

发送图片消息

- **消息说明**

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ImgMsg	存储	计数	存储	推送	[图片]

- **参数说明**

属性名称	属性类型	是否必填	属性说明
content	String	是	图片的略缩图，必须是 base64 字符串，类型必须为 jpg，base64 字符串大小不可超过 10 KB，base64 略缩图必须不带前缀
imageUriString		是	上传到服务器的 url. 用来展示高清图片

属性名称	属性类型	是否必填	属性说明
extra	String	否	附加信息，一般为消息不显示消息内容

- 代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.IMAGE, // 'RC:ImgMsg'
content: {
content: '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....', // // 压缩后的 base64 略缩图，用来快速展示图片
imageUri: 'https://www.rongcloud.cn/images/newVersion/log_wx.png' // 上传到服务器的 url，用来展示高清图片
}
}).then(function(message){
console.log('发送图片消息成功', message);
});
```

图片上传（Web）

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 window 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用 getFileToken 方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

1. 开始上传图片。

   - config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

   - **代码示例：**

     ```
     //创建 input 上传按钮
     <input id="uploadFile" type="file">
     ```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initImage(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name

参数	类型	必填	说明
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethodNumber		否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

图片上传（小程序）

小程序上传图片，必须使用小程序官方 API。代码示例：https://github.com/rongcloud/websdk-demo/tree/master/miniprogram-upload。

图片上传（uni-app）

uni-app 上传图片，必须使用 uni-app 官方 API。详见 uni-app 官方文档 uni.uploadFile(OBJECT)。

语音消息

⚠ 警告

1. SDK 目前不提供录音功能, 开发者需生成语音 url 后, 传入发送消息接口，发送语音消息
2. 语音上传请参照 文件上传 进行是实现。

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:HQVCMsg	存储	计数	存储	推送	[语音]

参数说明

属性名称	属性类型	是否必填	属性说明
remoteUrl	String	是	媒体内容上传服务器后的网络地址
type	String	否	编解码类型，默认为 aac 音频
duration	Number	否	语音消息的时长，最长为 60 秒（单位：秒）
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.HQ_VOICE, // 'RC:HQVCMsg'
content: {
remoteUrl: 'https://rongcloud-audio.cn.ronghub.com/audio_amr__RC-2020-03-17_42_1584413950049.aac?
e=1599965952&token=CddrKW5AbOMQaDRwc3ReDNvo3-sL_SO1fSUBKV3H:CDngyWj7ZApNmAfoecng7L_3SaU=', // 音频 url, 建
议格式: aac
duration: 6, // 音频时长
type: 'aac'
}
}).then(function(message){
console.log('发送语音消息成功', message);
});
```

文件消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。
- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。

发送文件消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:FileMsg	存储	计数	存储	推送	[文件] + 文件名，如：[文件] 123.txt

参数说明

属性名称	属性类型	是否必填	属性说明
name	String	是	文件名称
size	Number	是	文件大小，单位：bytes
type	String	是	文件类型
fileUrl	String	是	上传到服务器的 url
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
var name = 'RongIMLib.js'; // 文件名
var size = 1024; // 文件大小
var type = 'js'; // 文件类型
var fileUrl = 'https://cdn.ronghub.com/RongIMLib.js'; // 文件地址
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.FILE, // 'RC:FileMsg'
content: {
name: name,
size: size,
type: type,
fileUrl: fileUrl
}
}).then(function(message){
console.log('发送文件消息成功', message);
});
```

文件上传

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 window 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用 getFileToken 方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

4. 开始上传图片。

- config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

- **代码示例：**

```
//创建 input 上传按钮
<input id="uploadFile" type="file">
```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initFile(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name

参数	类型	必填	说明
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethodNumber		否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

小视频消息

如果 App Key 使用 **IM 旗舰版**或 **IM 尊享版**，文件存储时长默认为 180 天（不含小视频文件，小视频文件存储 7 天）。注意，**IM 商用版（已下线）默认存储 7 天。如需了解IM 旗舰版或 IM 尊享版的具体功能与费用，请参见融云官方价格说明页面及计费说明。**

> ⚠ 警告
>
> 1. 此消息类型 Web 端 SDK 仅支持解析展示，不提供录制。
> 2. 如 Web 端需要发送小视频消息，小视频录制需要开发者自行实现。

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:SightMsg	存储	计数	存储	推送	[小视频]

参数说明

参数	类型	说明
sightUrl	String	上传到文件服务器的小视频地址
content	String	小视频首帧的缩略图进行 Base64 编码的结果值，格式为 JPG，注意在 Base64 进行 Encode 后需要将所有 \r\n 和 \r 和 \n 替换成空
duration	Number	视频时长，单位：秒
size	Number	视频大小单位 bytes
name	String	发送端视频的文件名，小视频文件格式为 mp4。

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.SIGHT, // 'RC:SightMsg'
content: {
content: "/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
sightUrl: "http://rongcloud...com/video...",
duration: 10,
size: 734320,
name: "video_xx.mp4",
}
}).then(function(message){
console.log('发送小视频消息成功', message);
});
```

GIF 消息

消息说明

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:GIFMsg	存储	计数	存储	推送	[图片]

参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 byte
remoteUrl	String	GIF 图片的服务器地址
width	Number	GIF 图的宽
height	Number	GIF 图的高

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.GIF, // 'RC:GIFMsg'
content: {
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
}).then(function(message){
console.log('发送 GIF 消息成功', message);
});
```

发送自定义消息

1. 注册自定义消息

注意事项：

- 注册自定义消息代码必须在发送、接收该自定义消息之前
- 推荐将所有注册自定义消息代码放在 `init` 方法之后，`connect` 方法之前

- 注册的消息类型名，必须多端一致，否则消息无法互通
- 请勿使用 `RC:` 开头的类型名，以免和 SDK 默认的消息名称冲突

参数说明

参数	类型	说明
messageType	String	消息类型名
isPersited	Boolean	是否存储
isCounted	Boolean	是否计数

代码示例

```
// 初始化 IM
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);

var messageType = 's:person'; // 自定义消息类型
var isPersited = true; // 自定义消息存储属性
var isCounted = true; // 自定义消息计数属性
im.registerMessageType(messageType, isPersited, isCounted);
```

2. 发送自定义消息

参数说明

参数	类型	说明
messageType	String	消息类型名
content	Object	消息属性对象

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
}
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

配置消息推送

MessagePushConfig 属性介绍

> ⚠ 警告
>
> 从 SDK 4.0.4 版本开始支持此功能，针对每条 Message 都可以设置此属性，详细查看以下参数说明。

参数	类型	说明
pushTitle	String	推送标题，如果没有设置，会使用 SDK 默认的标题显示规则
pushContent	String	推送内容，如果没有，则使用发送消息的 pushContent，最后则会使用 SDK 默认的标题显示规则
pushData	String	远程推送附加信息，如果没有，则使用发送消息的 pushData
forceShowDetailContent	boolean	是否强制显示通知详情，当目标用户设置推送不显示消息详情时，可通过此参数，强制设置该条消息显示推送详情
disablePushTitle	boolean	iOS 平台是否禁用推送标题
iOSConfig	IOSConfig	iOS 平台相关配置
androidConfig	AndroidConfig	Android 平台相关配置

iOSConfig 属性介绍

参数	类型	说明
threadId	String	iOS 平台通知栏分组 ID，相同的 threadId 推送分为一组（iOS10 开始支持）
apnsCollapseId	String	iOS 平台通知覆盖 ID，apnsCollapseId 相同时，新收到的通知会覆盖老的通知，最大 64 字节（iOS10 开始支持）

AndroidConfig 属性介绍

参数	类型	说明
notificationId	String	Android 平台 Push 唯一标识，目前支持小米、华为推送平台，默认开发者不需要进行设置，当消息产生推送时，消息的 messageUId 作为 notificationId 使用
channelIdMi	String	小米的渠道 ID，该条消息针对小米使用的推送渠道，如开发者集成了小米推送，需要指定 channelId 时，可向 Android 端研发人员获取，channelId 由开发者自行创建
channelIdHW	String	华为的渠道 ID，该条消息针对华为使用的推送渠道，如开发者集成了华为推送，需要指定 channelId 时，可向 Android 端研发人员获取，channelId 由开发者自行创建
channelIdOPPO	String	OPPO 的渠道 ID，该条消息针对 OPPO 使用的推送渠道，如开发者集成了 OPPO 推送，需要指定 channelId 时，可向 Android 端研发人员获取，channelId 由开发者自行创建
typeVivo	String	VIVO 推送通道类型，开发者集成了 VIVO 推送，需要指定推送类型时，可进行设置。 目前可选值 "0"(运营消息) 和 "1"(系统消息)

Channel ID 需要由开发者进行创建，创建方式如下：

推送通道配置说明

华为	App 端，调用 Android SDK 创建 Channel ID 接口创建 Channel ID
小米	在小米开放平台管理台上创建 Channel ID 或通过小米服务端 API 创建
OPPO	App 端，调用 Android SDK 创建 Channel ID；在 OPPO 管理台登记该 Channel ID，保持一致性
vivo	调用服务端 API 创建 Channel ID

示例代码

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: 'Hello RongCloud' // 文本内容
},
pushConfig: {
pushTitle: "推送标题",
pushContent: "推送内容",
pushData: "推送附加消息",
disablePushTitle: false,
forceShowDetailContent: false,
iOSConfig: {
threadId: "threadId",
apnsCollapseId: "apnsCollapseId"
},
androidConfig: {
notificationId: "notificationId",
channelIdMi: "channelIdMi",
channelIdHW: "channelIdHW",
channelIdOPPO: "channelIdOPPO",
typeVivo: "typeVivo"
},
templateId: "templateId"
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

常见问题

Q1: 收到的表情信息无法解析， 表情显示有问题
A1: 解决方案：

1. Web 端展示 Emoji 时, 都需要调用 `emojiToHTML` 转成 HTML 或者使用 `symbolToEmoji` 将 Unicode 转化成原生 Emoji 字符
2. 发送消息时, 必须发送原生 Emoji 字符, 如果发送 HTML, 则认定发送的是字符串

Q2: 表情列表 每一个手机展示的表情不一样
A2: 解决方案：

1. 消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示
2. 不同浏览器, 不同设备, 展示的原生 Emoji 表情都不同
3. 如需多端展示 Emoji 一致, 需使用 `emojiToHTML` 转化为 HTML 后再展示(此方法为以图片形式展示)
   可参照考文档进行实现：emoji插件

消息接收　　功能描述

更新时间:2024-08-30

开发者可通过此接口拦截到 SDK 接收到的消息，并进行响应的业务操作。

实现方法

SDK 通过消息监听接收消息，详见消息监听。

历史消息获取　　本地获取

更新时间:2024-08-30

Web 没有本地存储，不提供本地获取方法。

远端获取

从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 IM 旗舰版或 IM 尊享版可开通该服务。具体功能与费用以融云官方价格说明页面及计费说明文档为准。

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
timestamp	Number	否	0	获取时间戳. 0 为从当前最新时间拉取 时间戳参数名对比 3.X 已修正为 `timestamp` 单位：毫秒	4.0.0
count	Number	否	20	获取条数, 范围 1 - 20	4.0.0
order	Number	否	0	获取顺序，默认为 0， 0 表示升序：获取消息发送时间比传入 sentTime 小 的消息 1 表示倒序：获取消息发送时间比传入 sentTime 大 的消息	4.0.0

回调参数说明

参数	类型	说明
list	Array	获取的历史消息列表，返回 message 列表
hasMore	Bool	是否还有历史消息可以获取

`message` 属性说明

字段名	类型	说明
`type`	Number	会话类型
`targetId`	String	接收方的 userId
`senderUserId`	String	发送者 id
`content`	Object	消息内容
`messageType`	String	消息标识
`messageUId`	String	服务端存储的消息 id
`messageDirection`	Number	消息方向。 发送: 1, 接收: 2
`isOffLineMessage`	Boolean	是否为离线消息
`sentTime`	Number	消息在融云服务端的发送时间
`receivedTime`	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
`isPersited`	Boolean	消息是否存储在服务端
`isCounted`	Boolean	消息是否计数
`isMentioned`	Boolean	是否为 @ 消息

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
var option = {
timestamp: +new Date(),
count: 20
};
conversation.getMessages(option).then(function(result){
var list = result.list; // 历史消息列表
var hasMore = result.hasMore; // 是否还有历史消息可以获取
console.log('获取历史消息成功', list, hasMore);
});
```

消息回执　　功能描述 功能描述

更新时间:2024-08-30

开发者可使用此功能实现消息已读未读功能的展示。

当 A 给 B 发送了一条消息，B 在未阅读之前 A 用户显示未读，当 B 用户阅读并调用发送回执接口之后，A 用户可在监听回执中收到通知，此时可根据对应的数据内容将发送的消息显示为已读。

发送回执

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ReadNtf	不存储	不计数	不存储	不推送	无

参数说明

属性名称	属性类型	是否必填	属性说明
messageUId	String	是	消息唯一 ID
lastMessageSendTime	Number	是	最后一条消息的发送时间
type	String	是	备用，默认赋值 1 即可

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
var messageUId = '1301-NBJQ-MK31-3417'; // 消息唯一 Id, message 中的 messageUid
var lastMessageSendTime = 1550719033312; // 最后一条消息的发送时间
var type = '1'; // 备用，默认赋值 1 即可
// 以上 3 个属性在会话的最后一条消息中可以获得
conversation.send({
messageType: 'RC:ReadNtf',
content: {
messageUId: messageUId,
lastMessageSendTime: lastMessageSendTime,
type: type
}
}).then(function(message){
console.log('发送已读通知消息成功', message);
});
```

接收回执

消息通过设置监听中的消息监听进行接收，消息监听中接收 RC:ReadNtf 消息，收到后按需处理即可。消息监听文档

消息撤回　　消息撤回

更新时间:2024-08-30

消息发送方可通过下面方法撤回已发送成功的消息。撤回指定消息后，原消息将被删除。

参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	3.0.0
sentTime	Number	是	消息发送时间	3.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});

conversation.recall({
messageUId: 'BH5T-JG24-C445-IKQM',
sentTime: 1585638211857
}).then(function(message){
console.log('撤回消息成功', message);
});
```

监听撤回

消息通过设置监听中的消息监听进行接收，消息监听中接收 RC:RcCmd 消息，收到后按需处理即可。消息监听文档

消息转发

更新时间:2024-08-30

消息转发调用发送消息接口发送即可，调用 SDK 发送消息接口需考虑 SDK 每秒 5 条消息的限制

超过每秒 5 条限制可使用 Server API 进行发送

消息删除　　本地删除

更新时间:2024-08-30

Web 没有本地存储，不提供本地删除功能。

远端删除

通过消息 ID 删除

参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	3.0.0
sentTime	Number	是	消息发送时间	3.0.0
messageDirection	Number	是	消息发送方向	3.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.deleteMessages([
{ messageUId: '2jJ9-KU1j-OLJG-29KL', sentTime: 1580869079801, messageDirection: 1 },
{ messageUId: '8UJ9-JU9j-WSJG-92K0', sentTime: 1580869078886, messageDirection: 1 }
]).then(function(){
console.log('删除历史消息成功');
});
```

通过时间戳删除

参数说明

参数	类型	必填	说明	最低版本
timestamp	Number	是	清除时间点，该时间之前的消息将被清除 时间戳参数名对比 3.X 已修正为 timestamp	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.clearMessages({
timestamp: +new Date()
}).then(function(){
console.log('清除历史消息成功');
});
```

消息扩展　　功能描述

更新时间:2024-08-30

> ⚠ 警告
>
> 从 3.0.7 版本开始支持单条消息扩展信息设置功能。

功能描述如下：

- 对已发送并且已设置扩展标识的消息，可设置扩展信息。
- 信息以 Key、Value 的方式进行存储，单条消息可设置 300 个扩展信息。
- 该功能仅支持单聊、群聊会话类型，暂不支持在聊天室中使用。
- 如已开通历史消息云存储功能，针对某一条消息设置的扩展信息，也会同时保存到该条消息的历史记录中。
- 每个终端在设置扩展信息时，如未达到上线都可以进行设置，所在并发情况下，会出现设置超过 300 的情况，超出部分会丢弃删除。
- 设置消息扩展后，会产生内置消息. 频繁设置扩展会产生大量消息，如果对消息量增长敏感，请谨慎使用此功能。

> ⓘ 提示
>
> **适用场景:**
>
> 需要对原始消息增加状态标识的需求, 都可使用消息扩展
>
> - 比如消息评论需求，可通过设置原始消息扩展信息的方式添加评论信息
> - 比如礼物领取、订单状态变化需求，通过此功能改变消息显示状态. 向用户发送礼物，默认为未领取状态，用户点击后可设置消息扩展为已领取状态

设置监听

Web 扩展监听需在 `im.watch` 中增加 `expansion` 事件，为避免设置重复，可移步 监听设置 查看实现方法

设置扩展

发送消息设置扩展内容

参数说明:

参数	类型	必填	默认值	说明	最低版本
messageType	String	是	--	消息类型	3.0.0
content	Object	是	--	消息内容	3.0.0
canIncludeExpansion	Boolean	否	false	是否携带扩展	3.0.7
expansion	Object	否	--	携带的扩展内容	3.0.7

代码示例:

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});

conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: { // 填写开发者定义的消息内容
content: '文字消息'
},
canIncludeExpansion: true, // 是否携带扩展
expansion: { // 携带的扩展内容
key: 'value',
key2: 'value2'
}
}).then((message) => {
console.log('发送携带扩展的文字消息成功', message);
}).catch((error) => {
console.log('发送携带扩展的文字消息失败', error);
});
```

⚠ 警告

扩展限制:

1. Key 支持英文、字母、数字、+、=、-、_。
2. Key 最大长度 32 字符。
3. 如果 SDK 版本 < 4.6.0，Value 最大长度 64 字符。如果 SDK 版本 ≧ 4.6.0，Value 最大 4096 个字符。
4. 单次调用最多设置 20 个键值对。
5. 单条消息最多设置 300 个键值对。

更新扩展

更新消息的扩展内容。

- 每次设置消息扩展将会产生内置通知消息，频繁设置扩展会产生大量消息。
- 每个终端在设置扩展信息时，如未达到上限都可以进行设置；在并发情况下，会出现设置超过 300 的情况，超出部分会被丢弃。

参数说明:

参数	类型	必填	说明	最低版本
expansion	Object	是	扩展内容	3.0.7
message	Object	是	原消息体	3.0.7

代码示例:

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var expansion = {
key: '新 value',
key2: '旧 value2',
key3: '新 value3'
};
var message = {
content: {
content: '文字消息'
},
messageUId: 'BK96-PP18-OIQ6-9GPP',
canIncludeExpansion: true,
expansion: {
key2: '旧 value2'
},
// .....
};
conversation.updateMessageExpansion(expansion, message).then(() => {
console.log('更新消息扩展成功');
}).catch((error) => {
console.log('更新消息扩展失败', error);
});
```

删除扩展

参数说明:

参数	类型	必填	说明	最低版本
keys	Array	是	删除的 keys	3.0.7
messageObject		是	原消息体	3.0.7

代码示例:

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var keys = ['key', 'key3'];
conversation.removeMessageExpansion(keys, message).then(() => {
console.log('删除消息扩展成功')
}).catch((reason) => {
console.log('删除消息扩展失败', reason)
});
```

群聊介绍　　概述

更新时间:2024-08-30

群组指两个以上用户一起进行聊天，群组成员信息由 App 提供并进行维系，融云只负责将消息传达给群组中的所有用户。

主要功能

功能	描述
离线消息	支持离线消息存储，存储时间可设置（1 ～ 7 天），默认存储 7 天。
消息提醒	离线状态，群组中有新消息时，支持 Push 通知。
本地存储	存储在移动端本地，提供本地消息搜索功能。
历史消息	提供服务端消息存储存功能，需开通单群聊消息云存储，默认存储时长为 6 个月。
消息删除	支持按会话删除本地和存储在服务器的指定消息或会话中全部历史消息。
消息搜索	支持按关键字或用户搜索本地指定会话的消息内容。
群消息阅读回执	发送群消息后如需要查看消息的阅读状态，可以使用此功能来发送阅读回执请求。
消息撤回	消息发送成功后，在有效时间内可撤回该条消息，默认可撤回时间为 2 分钟，时间可配置。
群聊会话免打扰	可设置指定的群聊会话，收到新的消息后是否进行提醒，默认进行新消息提醒。
创建群组	App 内的群组数量没有限制，默认一个群上限为 3000 人，可调整群上限，需要提交工单申请开通。
加入群组	每个群最大至 3000 人，一个用户可加入多个群组，没有限制。默认加入群组后，只能查看加入后群组中产生的消息，如需要查看加入前的消息，则需要开通单群聊消息云存储后，再开通查看加入前群消息功能
退出群组	将用户从群中移除，不再接收该群组的消息。
解散群组	将指定群组解散，所有成员都无法再接收该群的消息。
群成员查询	获取指定群组中群成员用户 Id。
刷新群组信息	目前支持更新群组名称。
同步用户所属群组	在集成融云前 App Server 已有群组数据，可使用此服务进行同步。
群组成员禁言	被禁言用户可以接收查看群组中其他用户消息，但不能通过客户端 SDK 发送消息。
群组整体禁言	指定群组所有成员不能发送消息，需要某些用户可以发言时，可将此用户加入到群禁言用户白名单中。
群组禁言用户白名单	群组被整体禁言后，禁言白名单中用户可以发送群消息。
发送群组消息	向群组中所有成员发送消息。
发送群组定向消息	向群中指定的一个或多个用户发送消息，群中其他用户无法收到该消息。
发送群组@消息	发送群组中指定群成员需要特别关注的消息。
发送群组状态消息	群组中在线用户会收到此条消息，离线用户不会再收到此条消息，状态消息不计数、不存储。

与聊天室的区别

融云提供群组与聊天室业务，其主要区别如下，客户可根据自己的业务场景进行选择：

功能	群组（group）	聊天室（Chatroom）
场景	类似微信的群组，无论是否在线都会接收消息	只有在线用户可接收消息，可用于直播、社区、游戏、广场交友、兴趣讨论等场景。
离线缓存消息	支持离线消息存储，存储时间可设置（1 ～ 7 天），默认存储 7 天。	无离线消息，只有在线用户才可收到聊天室消息
人数限制	默认一个群上限为 3000 人	聊天室人数无上限
消息提醒	离线状态，群组中有新消息时，支持远程推送（PUSH）通知	离开聊天室后不再接收消息
本地存储	移动端本地数据库存储，提供本地消息搜索接口	退出聊天室后同时删除本地聊天室消息，不支持消息搜索功能
云端存储	需开通单群聊消息云存储，可以提供 6 - 36 个月存储服务	需开通聊天室消息云存储，可以提供 2 - 36 个月存储服务

功能	群组（group）	聊天室（Chatroom）
用户加入限制	一个用户可加入多个群组，无限制	默认一个用户只能加入一个聊天室，加入多个聊天室功能可在控制台自行开通
加入后消息获取逻辑	默认加入群组后，只能查看加入后群组中产生的消息。如需要查看群历史消息，则需要开通单群聊消息云存储后，再开通“查看加入前群消息”功能	加入后可获取聊天室中最新的 50 条消息。
销毁/解散逻辑	需要通过 AppServer 自行调用解散群组接口。	提供销毁聊天室接口，可通过 AppServer 调用。同时聊天室中 1 小时内没有消息产生时，将自动销毁聊天室。
消息可靠度	100% 可靠，不丢消息。	消息量较大时，超出服务端消费上限的消息将被主动抛弃。您可通过用户白名单、消息白名单、自定义消息级别等服务，改变消息抛弃策略。如果用户在聊天室的用户白名单内，该用户所发送的消息在消息量大时也不会被抛弃。 如需了解服务端消费上限与如何改变消息抛弃策略，可参见服务端文档消息优先级服务、聊天室白名单服务。
相关接口调用	SDK 不提供群组管理功能接口，通过 Server API 提供群组功能接口。	SDK 和 Server API 同时提供功能接口，销毁聊天室操作只能通过 Server API 方式调用。
发送消息频率	每个客户端 5 条/秒；服务端调用，20 条/秒	每个客户端 5 条/秒；服务端调用，100 条/秒

消息类型

消息类型	描述
文字消息	用来发送文字类消息，其中可以包括表情、超链接（会自动识别），客户端收到消息后计入未读消息数、进行存储。
语音消息	发送高质量的短语音消息，录制的语音文件存储到融云服务端，语音文件格式为 AAC，时长上限为 60 秒，客户端收到消息后计入未读消息数、进行存储。
图片消息	用来发送图片类消息，客户端收到消息后计入未读消息数、进行存储。图片缩略图格式为 JPG，大小建议不超过 100k。
GIF 图片消息	用来发送 GIF 动态图片消息，客户端收到消息后计入未读消息数、进行存储。
图文消息	用来发送图文消息，包含一个标题，一段文字内容和一张图片，客户端收到消息后计入未读消息数、进行存储。
文件消息	用来发送文件类消息，客户端收到消息后计入未读消息数、进行存储。
位置消息	用来发送地理位置消息，客户端收到消息后计入未读消息数、进行存储。
小视频消息	用来发送小视频消息，支持录制发送及选择本地视频文件发送两种方式，录制时长不超过 10 秒，本地选择视频文件方式时长不超过 2 分钟，小视频消息小视频文件格式为 .mp4，客户端收到消息后计入未读消息数、进行存储。
合并转发消息	IMKit SDK 中支持将多条消息合并为一条消息进行发送，合并后的消息以 HTML 文件的方式存储到融云服务端，客户端收到消息后计入未读消息数、进行存储。红包、阅后即焚及自定义消息的合并转发功能
命令消息	用来发送通用的指令通知消息，消息内可以定义任意 JSON 内容，与通用命令通知消息的区别是不存储、不计数，此类型消息没有 Push 通知。
自定义消息	融云内置消息类型，无法满足客户业务需求时，可通过自定义消息类型进行实现，接收自定义消息的格式解析及展示处理需要开发者自行实现

内置消息说明

> 请参见融云服务端文档内置消息类型说明。

获取全部会话　　获取本地会话列表

更新时间:2024-08-30

Web 端不具备持久化的数据存储能力，故无法提供获取本地会话列表相关功能

获取远端会话列表

> Web 端不具备持久化的数据存储能力，无法在本地持久化存储历史消息记录与会话列表，因此需要从融云服务端获取数据。从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 IM 旗舰版或 IM 尊享版可开通该服务。具体功能与费用以融云官方价格说明页面及计费说明文档为准。
> 该功能需要在调用 `im.connect()` 并且建立连接成功之后执行。
> 服务器会话列表存储上限为 `1000` 条会话

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
count	Number	否	300	想要获取的会话列表数量，默认值 `300`，最大值 `1000`	3.0.0
startTime	Number	否	0	获取会话列表所使用的时间戳，需要精确到毫秒	3.0.7.1
order	Number	否	0	会话排序方式	3.0.7.1

`order` 说明：

value说明

0	按会话生成时间倒序排列，即获取早于 startTime 生成的会话列表
1	按会话生成时间正序排列，即获取晚于 startTime 生成的会话列表

`startTime` 补充说明：

- startTime 为 0 且 order 为 0 时, startTime 表示当前时间点，接口将返回最新的会话列表
- startTime 为 0 且 order 为 1 时, startTime 表示会话列表的创建时间，即最早的一条消息的产生时间，接口将返回最早的会话列表数据

回调参数说明

回调参数	类型	说明
conversationListArray		会话列表，会话参数说明请参照 `conversation` 属性说明

- `conversation` 属性说明：

字段名	类型	说明
type	Number	会话类型，见 `type` 枚举值说明

字段名	类型	说明
targetId	String	群组 ID
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息，详见 latestMessage
hasMentioned	Boolean	是否包含 @ 自己的消息
mentionedInfo	Object	@ 信息，见 `mentionedInfo` 结构说明
notificationStatus	Number	当前会话免打扰状态
isTop	Boolean	当前会话免置顶状态

- `type` 枚举值说明

会话类型	说明	枚举值
RongIMLib.CONVERSATION_TYPE.PRIVATE	单聊	1
RongIMLib.CONVERSATION_TYPE.GROUP	群聊	3
RongIMLib.CONVERSATION_TYPE.CHATROOM	聊天室	4
RongIMLib.CONVERSATION_TYPE.SYSTEM	系统	6

- `mentionedInfo` 结构说明：

字段名	类型	说明
type	Number	@ 类型，1: @ 所有人， 2: @ 部分人
userIdListArray		被 @ 的用户 id 列表

代码示例

```
im.Conversation.getList({
count: 30,
startTime: 0,
order: 0
}).then(conversationList => {
console.log('获取会话列表成功', conversationList);
});
```

删除全部会话　　清除会话列表

更新时间:2024-08-30

SDK 不提供批量删除接口，如需事先清除多个会话，可循环调用 删除指定会话 接口来实现。

获取指定会话

更新时间:2024-08-30

可以通过获取会话列表得到会话列表信息，通过 Conversation.get 获取会话实例进行消息发送等操作。

参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	群组 ID	3.0.0
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.GROUP	3.0.0

代码示例

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
const conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
```

conversation 属性说明：

字段名	类型	说明
type	Number	会话类型
targetId	String	群组 ID

删除指定会话　　删除指定会话

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 `conversation.destory()` 删除会话。

> 当收到新消息或者发送消息时会重新生成该会话。

代码示例

```
// conversation 会话实例
const conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.destory().then(() => console.log('删除会话成功'));
```

会话草稿信息　　获取草稿

更新时间:2024-08-30

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	群组 ID	4.0.0
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.GROUP	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.getDraft().then(function (draft) {
console.log('获取指定会话草稿成功', draft)
})
```

保存草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	群组 ID	4.0.0
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.GROUP	4.0.0

参数说明

参数	类型	必填	说明	最低版本
draft	String	是	草稿内容	4.0.0

代码示例

```
var draft = '这是会话草稿内容';
var conversation = im.Conversation.get({
targetId: '群组 ID',
type:RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.setDraft(draft).then(function () {
console.log('设置指定会话草稿成功')
})
```

删除草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	群组 ID	4.0.0
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.GROUP	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.deleteDraft().then(function () {
console.log('删除指定会话草稿成功')
})
```

会话未读数　　获取所有会话未读数

更新时间:2024-08-30

会话未读数指某一个会话中未读消息的数量

1. 清除浏览器缓存会导致会话未读数不准确
2. 会话消息未读数存储在 WebStorage 中, 若浏览器不支持或禁用 WebStorage，未读消息数将不会保存，浏览器页面刷新未读消息数将不会存在

参数说明

参数	类型	必填	说明	最低版本
includeMuted	Boolean	否	是否包含免打扰会话，默认为: false	4.4.5
conversationTypes	Array	否	会话类型	4.4.5

代码示例

```
const conversationTypes = [RongIMLib.CONVERSATION_TYPE.SYSTEM]
im.Conversation.getTotalUnreadCount(false, conversationTypes).then(function(
totalUnreadCount
) {
console.log('获取未读总数成功', totalUnreadCount)
})
```

清除全部会话未读数

最低版本：4.5.4

```
im.Conversation.readAll()
```

清除单个会话未读数

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.read().then(function(){
console.log('清除未读数成功');
});
```

获取指定会话未读数

代码示例

```
let conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});

conversation.getUnreadCount().then(function(count) {
console.log('获取指定会话未读数成功', count);
})
```

多端同步未读数

未读消息存在 localStorage 中，未读消息数是针对当前端的未读消息数，服务器不存未读消息数量。

实现方案

1. 调用 conversation.read() 清除未读数。
2. 清除成功后发送 `RC:SRSMsg` 类型消息进行未读数同步。
3. 其他端接受到 `RC:SRSMsg` 类型消息，调用 `conversation.read()` 方法进行本地未读数清除。（在 v4.5.4 版本之后收到该消息时 sdk 内部会清理未读数，无需用户主动调用）

代码示例

> ⚠ 警告
>
> 群聊发送 RC:SRSMsg 消息必须传入 directionalUserIdList

清除端

```
// 清除未读数
let conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP,
})

conversation.read().then(function() {
console.log('清除指定会话未读数成功')
// 发送同步消息
conversation
.send({
messageType: 'RC:SRSMsg',
content: {
//从消息里获取服务器端时间，以最近一条已读 message 为准
lastMessageSendTime: message.sentTime,
},
// 定向发送给多端的自己，群里其他用户无需接收
directionalUserIdList: ['currentLoginUserId'],
})
.then(function(message) {
console.log('发送同步消息成功', message)
})
})
```

同步端

```
// 其他端在消息监听中接收到同步消息后，调用清除未读数做更新处理
// 收到同步消息进行未读数清除操作 调用 conversation.getUnreadCount() 成功后不需要再在发送 `RC:SRSMsg` 类型消息。
let conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP,
})

conversation.read().then(function() {
console.log('获取指定会话未读数成功')
})
```

会话免打扰　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.GROUP	3.0.0
targetId	String	是	群组 ID	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
let statusItem = {
notificationStatus: 1,
};
conversation.setStatus(statusItem).then(function() {
console.log('设置免打扰成功');
})
```

会话置顶　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，群聊 传入 RongIMLib.CONVERSATION_TYPE.{{ConversationType}}	3.0.0
targetId	String	是	群组 ID	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.{{ConversationType}},
})
let statusItem = {
isTop: true,
}
conversation.setStatus(statusItem).then(function() {
console.log('设置置顶成功')
})
```

会话标签

更新时间:2024-08-30

> SDK 从 4.5.0 版本开始支持会话标签功能。

会话标签用于对会话进行分组。用户设置会话标签，用于会话的分组显示。

用户可以按照标签获取会话列表，按照标签获取标签下所有会话的消息未读数，在特定标签下设置某个会话置顶。

每个用户最多可以创建 20 个标签，每个标签下最多可以添加 1000 个会话。

同一个会话可以设置多个不同的标签。

标签管理功能接口

创建

```
im.Tag.create({'tagId','tagName'}).then(code => {
// 创建标签成功
if(code === 0) {

}
})
```

移除

```
im.Tag.get(tagId).then(tag => {
tag.remove().then(data => {
// 删除标签成功
if(code === 0) {

}
})
})
```

编辑

```
im.Tag.get(tagId).then(tag => {
tag.update('tagName').then(data => {
// 更新标签成功
if(code === 0) {

}
})
})
```

获取标签列表

```
im.Tag.getTagInfoList().then(result => {
// 获取标签列表
if(!result.code) {
console.log(result)
}
})
```

会话标签功能

添加会话到标签

```
const type = RongIMLib.ConversationType.GROUP
// 会话数组
const conversations = [{type, targetId: 'targetId'}]
im.Tag.get(tagId).then(tag => {
tag.addConversations(conversations).then(code => {
// 添加成功
if(code === 0) {

}
})
})
```

删除标签中某些会话

```
const type = RongIMLib.ConversationType.GROUP
// 会话数组
const conversations = [{type, targetId: 'targetId'}]
im.Tag.get(tagId).then(tag => {
tag.removeConversations(conversations).then(code => {
// 删除成功
if(code === 0) {

}
})
})
```

删除会话中的标签

```
const type = RongIMLib.ConversationType.GROUP
im.Conversation.get({type,'targetId'}).removeTags(['tagId']).then(code => {
// 删除成功
if(code === 0) {

}
})
```

获取会话下的标签

```
const type = RongIMLib.ConversationType.GROUP
im.Conversation.get({type,'targetId'}).getTags().then(result => {
// 获取标签列表
if(!result.code) {
console.log(result)
}
})
```

分页获取指定标签下会话列表

```
// 开始时间
const startTime = 0
const count = 10
im.Tag.get(tagId).then(tag => {
tag.getConversationList(startTime,count).then(result => {
// 获取会话列表
if(!result.code) {
console.log(result)
}
})
})
```

按标签获取未读消息数

```
// 是否包含免打扰
const isIncludeNotNotification = true
im.Tag.get(tagId).then(tag => {
tag.getUnreadCount(isIncludeNotNotification).then(result => {
// 获取未读消息数
if(!result.code) {
console.log(result)
}
})
})
```

设置标签中会话置顶

> ⚠ 警告
>
> 注意：此处是在标签下会话的置顶，而不是会话的置顶。【这个属性在根据标签获取的会话中可以看到】

```
// 会话数组
const conversation = {type: RongIMLib.CONVERSATION_TYPE.PRIVATE,targetId: 'targetId'}
const isTopInTag = true
im.Tag.get(tagId).then(tag => {
tag.updateConversationIsTop(conversation, isTopInTag).then(code => {
// 更新成功
if(code === 0) {

}
})
})
```

消息发送　　属性描述

更新时间:2024-08-30

消息属性	描述
消息类名	各端消息名
存储属性	存储 / 不存储
离线属性	缓存 / 不缓存
推送属性	是/否
ObjectName	传输层名称，与消息类名一一对应
计数属性	计数 / 不计数
消息尺寸	128 KB
推送内容	详见各消息类送方式

存储属性

存储属性	存储分类	支持平台	详细描述
存储	客户端	Android、iOS	发送、接收该消息后，本地数据库存储 Web 端 和 小程序端因本地存储不可靠，不支持客户端消息存储，但可通过历史消息云存储服务获取历史记录
存储	云端	Android、iOS、Web	默认不在云端进行存储，需开通单群聊消息云存储服务，开通后，可在融云服务端存储 6 个月的历史消息，供客户端按需拉取
不存储	客户端	Android、iOS	发送、接收该消息后，本地数据库不存储
不存储	云端	Android、iOS、Web	无论是否开通历史消息云存储服务，该消息均不存储

计数属性

接收收到消息时，会话是否累计未读数。

计数属性	支持平台	详细描述
计数	iOS、Android、Web	会话未读消息数 + 1，该属性只影响会话列表未读数计数，App 应用角标可根据每个会话列表未读数累加获得
不计数	iOS、Android、Web	会话未读消息数不变

离线属性

接收人当前不在线时，是否进行离线缓存。

离线属性	详细描述
存储	消息进行离线缓存，默认 7 天。接收人在 7 天内上线，均可接收到该消息。超过 7 天后，消息被离线缓存淘汰。如有需要，可通过云端存储拉取到该消息
不存储	消息不进行离线缓存，所以只有接收人在线时，才可收到该消息。该消息不进行历史消息云存储、不进入云端存储（Log 日志）、不进行消息同步（消息路由）

推送属性

接收人是否接收推送，当离线属性为 *存储* 时，该属性生效。离线属性为 *不存储* 时属性无效。

由于 Web、小程序、PC 端没有推送平台，无法收到推送提醒。

推送属性	平台	推送方式	详细描述
推送	iOS、Android	APNs、华为、小米、魅族、OPPO、vivo、FCM、融云	当有离线缓存消息时，进行远程推送提醒，内容为该推送提醒显示的内容
不推送	iOS、Android	--	当有离线缓存消息时，不进行远程推送提醒

⚠ 警告

1. 发送消息必须在成功连接融云服务器 [connect] 成功之后进行。
2. 每秒最多发送 5 条消息。

消息结构

字段名	类型	说明
type	Number	会话类型
targetId	String	群组 ID
senderUserId	String	发送者 id
content	Object	消息内容
messageType	String	消息标识
messageUId	String	服务端存储的消息 id
messageDirection	Number	消息方向。 发送: 1, 接收: 2
isOffLineMessage	Boolean	是否为离线消息
sentTime	Number	消息在融云服务端的发送时间
receivedTime	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
isPersited	Boolean	消息是否存储在服务端
isCounted	Boolean	消息是否计数
disableNotification	Boolean	消息是否静默，静默消息不会发送 Push 信息和本地通知提醒
expansion	Object	携带的扩展内容

参数说明

参数	类型	必填	默认值	说明	最低版本
messageType	String	是	--	消息类型. 与移动端 ObjectName 一致	3.0.0
content	String	是	--	消息内容	3.0.0
isPersited	Boolean	否	true	是否存储在服务端	3.0.0
isCounted	Boolean	否	true	是否计数. 计数消息接收端接收后未读数加 1	3.0.0
pushContent	String	否	--	Push 显示内容	3.0.0
pushData	String	否	--	Push 通知时附加信息	3.0.0
isVoipPush	String	否	false	为 ture 时, 对端不在线的 iOS 会收到 Voip Push. Android 无影响	3.0.0
isStatusMessage	Boolean	否	false	是否发送状态消息，设置为 **true** 后 **isPersited** 和 **isCounted** 互斥	3.0.0
isMentioned	Boolean	否	false	是否为 @ 消息	3.0.0
mentionedType	Number	否	1	@ 类型 1: @ 所有人 2: @ 指定用户	3.0.0
mentionedUserIdList	Array	否		@ 用户 id 列表	3.0.0
directionalUserIdList	Array	否		接收定向消息的用户 id 列表. 仅群组有效	3.0.0
disableNotification	Boolean	否	false	是否发送静默消息，设置为 **true** 后不会发送 **Push** 信息和本地通知提醒	3.0.5
canIncludeExpansion	Boolean	否	false	是否允许扩展	3.0.7
expansion	Object	否	--	携带的扩展内容	3.0.7

发送代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP,
})
var expansion = {
key: '我是 value1',
key2: '我是 value2'
}; // 携带的扩展内容
conversation.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
},
isPersited: true,// 是否存储在服务端,默认为 true
isCounted: true, // 是否计数. 计数消息接收端接收后未读数加 1，默认为 true
pushContent:'user 发送了一条消息', // Push 显示内容
pushData: 'Push 通知时附加信息', // Push 通知时附加信息, 可不填
isStatusMessage: false // 设置为 true 后 isPersited 和 isCounted 属性失效
isMentioned: true,
mentionedType: RongIMLib.Mentioned_TYPE.ALL,
mentionedUserIdList:['user1'],
disableNotification: false, // 设置为 true 后不会发送 Push 信息和本地通知提醒
canIncludeExpansion: true, // 是否允许扩展
expansion: expansion
}).then(function(message){
console.log('发送 s:person 消息成功', message);
}).catch(function(error) {
console.log('发送 s:person 消息失败', error.code, error.msg);
});
```

发送消息

文本消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:TxtMsg 存储 计数 存储 推送 消息内容

参数说明

属性名称	属性类型	是否必填	属性说明
content	String	是	文本消息内容
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: 'Hello RongCloud' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

图文消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ImgTextMsg	存储	计数	存储	推送	消息内容

参数说明

属性名称	属性类型	是否必填	属性说明
content	String	是	消息的文字内容
title	String	是	消息的标题
imageUri	String	是	图片上传到服务器的 url
url	String	是	富文本消息点击后打开的 URL
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.RICH_CONTENT, // 'RC:ImgTextMsg'
content: {
title: '消息的标题',
content: '消息的文字内容',
imageUri: '图片上传到服务器的 url',
url: '富文本消息点击后打开的 URL'
}
}).then(function(message){
console.log('发送图文消息成功', message);
});
```

Emoji 消息

- web 端发送 Emoji 消息，开发者直接使用 文本消息 发送即可。
- 融云提供 Emoji 插件，内置了 128 个 Emoji 表情的图片，做消息输入框的表情选项，也可自行扩展配置。
- 发消息时，必须直接发送 Emoji 原生字符. 如：😊，转换方法： `symbolToEmoji`。
- Web SDK 接收消息时接收到的是 Unicode 编码格式，如：”ef600” 需要转化才能正确显示原生 Emoji。

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: '' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

Emoji 插件

插件兼容性

Chrome	Firefox	Safari	IE	Edge	iPhone	Android
30+	30+	10+	7+✔		iOS 8.0+ 的Safari浏览器以及微信浏览器	4.4+系统的Chrome浏览器以及微信浏览器

Emoji 插件引入

```
<!-- 非压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- 压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.min.js"></script>
```

> Emoji 代码示例：https://rongcloud.github.io/web-emoji-demo/src/index.html

> ⚠ 警告
>
> 1. 使用 `import * as RongIMLib from '@rongcloud/imlib-v4-adapter'` 方式引入 SDK 时表情插件调用需要使用 `window` 前缀。例如：`window.RongIMLib.RongIMEmoji.init()`
> 2. RongEmoji 插件仅支持 cdn 引入方式，暂不支持 npm 引入

Emoji 初始化：默认参数初始化

```
RongIMLib.RongIMEmoji.init();
```

Emoji 初始化：自定义表情配置初始化

`config` 参数说明：

参数	类型	必填	说明	最低版本
size	Number	否	表情大小，默认 24，建议 18 - 58	2.2.6
url	String	否	Emoji 背景图片 url	2.2.6
lang	String	否	Emoji 对应名称语言，默认 zh	2.2.6
extension	Object	否	扩展表情	2.2.6

```
// 表情信息可参考 http://unicode.org/emoji/charts/full-emoji-list.html
var config = {
size: 25,
url: '//f2e.cn.ronghub.com/sdk/emoji-48.png',
lang: 'en',
extension: {
dataSource: {
u1F914: { // 自定义 u1F914 对应的表情
en: 'thinking face', // 英文名称
zh: '思考', // 中文名称
tag: '', // 原生 Emoji
position: '0 0' // 所在背景图位置坐标
}
},
url: '//cdn.ronghub.com/thinking-face.png' // 新增 Emoji 背景图 url
}
};
RongIMLib.RongIMEmoji.init(config);
```

获取列表

```
var list = RongIMLib.RongIMEmoji.list;
/*list => [{
unicode: 'u1F600',
emoji: '',
node: span,
symbol: '[笑嘻嘻]'
}]
*/
```

Emoji 转文字

在不支持原生 Emoji 渲染时，可显示对应名称，适用于消息输入。

```
var message = '😁测试 Emoji';
// 将 message 中的原生 Emoji 转化为对应名称
RongIMLib.RongIMEmoji.emojiToSymbol(message);
// => '[笑嘻嘻][露齿而笑]测试 Emoji'
```

文字转 Emoji

发送消息时，消息体里必须使用原生 Emoji 字符。

```
var message = '[笑嘻嘻][露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为原生 Emoji
RongIMLib.RongIMEmoji.symbolToEmoji(message);
// => '😁测试 Emoji'
```

Emoji 转 HTML

Web SDK 接收消息后，消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示。

```
var message = '\uf600测试 Emoji';
// 将 message 中的原生 Emoji（包含 Unicode ）转化为 HTML
RongIMLib.RongIMEmoji.emojiToHTML(message);
// => "<span class='rong-emoji-content' name='[笑嘻嘻]'></span>测试 Emoji"
```

文字 转 HTML

```
var message = '[露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为 HTML
RongIMLib.RongIMEmoji.symbolToHTML(message);
// => "<span class='rong-emoji-content' name='[露齿而笑]'>😁</span>测试 Emoji"
```

位置消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:LBSMsg 存储 计数 存储 推送 [位置]
参数说明

属性名称属性类型是否必填属性说明

longitude	Number	是	经度
latitude	Number	是	纬度
poi	String	是	位置信息
content	String	是	位置缩略图,图片需要是不带前缀的 base64 字符串
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var latitude = 40.0317727;
var longitude = 116.4175057;
var poi = '北苑路 融云';
var content = '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE'; // 位置图片 base64
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.LOCATION, // 'RC:LBSMsg'
content: {
latitude: latitude,
longitude: longitude,
poi: poi,
content: content
}
}).then(function(message){
console.log('发送位置消息成功', message);
});
```

正在输入状态消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:TypSts 不存储 不计数 不存储 不推送 无
参数说明

属性名称	属性类型	是否必填	属性说明
typingContentType	String	是	正在输入的消息 ObjectName
data	String	否	携带信息

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: 'RC:TypSts',
content: {
typingContentType: 'RC:TxtMsg' // 正在输入的消息类型
},
isStatusMessage: true // 正在输入建议发送状态消息，不存储、不计数，且仅在线用户可收到
}).then(function(message){
console.log('发送正在输入消息成功', message);
});
```

图片消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。
- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。
- 小程序、uni-app 必须使用各平台官方的上传插件。

发送图片消息

- **消息说明**

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ImgMsg	存储	计数	存储	推送	[图片]

- **参数说明**

属性名称	属性类型	是否必填	属性说明
content	String	是	图片的略缩图，必须是 base64 字符串, 类型必须为 jpg，base64 字符串大小不可超过 10 KB，base64 略缩图必须不带前缀
imageUriString		是	上传到服务器的 url. 用来展示高清图片
extra	String	否	附加信息，一般为消息不显示消息内容

- **代码示例**

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.IMAGE, // 'RC:ImgMsg'
content: {
content: '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....', // // 压缩后的 base64 略缩图，用来快速展示图片
imageUri: 'https://www.rongcloud.cn/images/newVersion/log_wx.png' // 上传到服务器的 url. 用来展示高清图片
}
}).then(function(message){
console.log('发送图片消息成功', message);
});
```

图片上传（Web）

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 window 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用 getFileToken 方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

4. 开始上传图片。

- config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

- **代码示例：**

```
//创建 input 上传按钮
<input id="uploadFile" type="file">
```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initImage(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name

参数	类型	必填	说明
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethodNumber		否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

图片上传（小程序）

> 小程序上传图片，必须使用小程序官方 API。代码示例：https://github.com/rongcloud/websdk-demo/tree/master/miniprogram-upload。

图片上传（uni-app）

> uni-app 上传图片，必须使用 uni-app 官方 API。详见 uni-app 官方文档 uni.uploadFile(OBJECT)。

语音消息

> ⚠ 警告
>
> 1. SDK 目前不提供录音功能, 开发者需生成语音 url 后, 传入发送消息接口，发送语音消息
> 2. 语音上传请参照 文件上传 进行是实现。

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:HQVCMsg	存储	计数	存储	推送	[语音]

参数说明

属性名称	属性类型	是否必填	属性说明
remoteUrl	String	是	媒体内容上传服务器后的网络地址
type	String	否	编解码类型，默认为 aac 音频
duration	Number	否	语音消息的时长，最长为 60 秒（单位：秒）
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.HQ_VOICE, // 'RC:HQVCMsg'
content: {
remoteUrl: 'https://rongcloud-audio.cn.ronghub.com/audio_amr__RC-2020-03-17_42_1584413950049.aac?
e=1599965952&token=CddrKW5AbOMQaDRwc3ReDNvo3-sL_SO1fSUBKV3H:CDngyWj7ZApNmAfoecng7L_3SaU=', // 音频 url, 建
议格式: aac
duration: 6, // 音频时长
type: 'aac'
}
}).then(function(message){
console.log('发送语音消息成功', message);
});
```

文件消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。
- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。

发送文件消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:FileMsg	存储	计数	存储	推送	[文件] + 文件名，如：[文件] 123.txt

参数说明

属性名称	属性类型	是否必填	属性说明
name	String	是	文件名称
size	Number	是	文件大小，单位：bytes
type	String	是	文件类型
fileUrl	String	是	上传到服务器的 url
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var name = 'RongIMLib.js'; // 文件名
var size = 1024; // 文件大小
var type = 'js'; // 文件类型
var fileUrl = 'https://cdn.ronghub.com/RongIMLib.js'; // 文件地址
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.FILE, // 'RC:FileMsg'
content: {
name: name,
size: size,
type: type,
fileUrl: fileUrl
}
}).then(function(message){
console.log('发送文件消息成功', message);
});
```

文件上传

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 window 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用 getFileToken 方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

4. 开始上传图片。

- config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

- **代码示例：**

```
//创建 input 上传按钮
<input id="uploadFile" type="file">
```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initFile(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name

参数	类型	必填	说明
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethodNumber		否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

小视频消息

如果 App Key 使用 **IM 旗舰版**或 **IM 尊享版**，文件存储时长默认为 180 天（不含小视频文件，小视频文件存储 7 天）。注意，**IM 商用版（已下线）默认存储 7 天。如需了解IM 旗舰版或 IM 尊享版的具体功能与费用，请参见融云官方价格说明页面及计费说明。**

> ⚠ 警告
>
> 1. 此消息类型 Web 端 SDK 仅支持解析展示，不提供录制。
> 2. 如 Web 端需要发送小视频消息，小视频录制需要开发者自行实现。

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:SightMsg	存储	计数	存储	推送	[小视频]

参数说明

参数	类型	说明
sightUrl	String	上传到文件服务器的小视频地址
content	String	小视频首帧的缩略图进行 Base64 编码的结果值，格式为 JPG，注意在 Base64 进行 Encode 后需要将所有 \r\n 和 \r 和 \n 替换成空
duration	Number	视频时长，单位：秒
size	Number	视频大小单位 bytes
name	String	发送端视频的文件名，小视频文件格式为 mp4。

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.SIGHT, // 'RC:SightMsg'
content: {
content: "/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
sightUrl: "http://rongcloud...com/video...",
duration: 10,
size: 734320,
name: "video_xx.mp4",
}
}).then(function(message){
console.log('发送小视频消息成功', message);
});
```

GIF 消息

消息说明

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:GIFMsg	存储	计数	存储	推送	[图片]

参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 byte
remoteUrl	String	GIF 图片的服务器地址
width	Number	GIF 图的宽
height	Number	GIF 图的高

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.GIF, // 'RC:GIFMsg'
content: {
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
}).then(function(message){
console.log('发送 GIF 消息成功', message);
});
```

发送 @ 消息

> ⚠ 警告
>
> 1. 支持 @ 的会话类型: RongIMLib.CONVERSATION_TYPE.GROUP
> 2. 支持 @ 的消息类型: MESSAGE_TYPE.TEXT、MESSAGE_TYPE.IMAGE、MESSAGE_TYPE.LOCATION

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 填写开发者定义的 messageType
content: {
content: 'Hello RongCloud' // 文本内容
},
isMentioned: true,
mentionedType: RongIMLib.MENTIONED_TYPE.ALL,
mentionedUserIdList:['user1']
}).then(function(message){
console.log('发送消息成功', message);
});
```

发送定向消息

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 填写开发者定义的 messageType
content: {
content: 'Hello RongCloud' // 文本内容
},
directionalUserIdList:['user1']
}).then(function(message){
console.log('发送消息成功', message);
});
```

发送自定义消息

1. 注册自定义消息

注意事项：

- 注册自定义消息代码必须在发送、接收该自定义消息之前
- 推荐将所有注册自定义消息代码放在 `init` 方法之后，`connect` 方法之前
- 注册的消息类型名，必须多端一致，否则消息无法互通
- 请勿使用 `RC:` 开头的类型名，以免和 SDK 默认的消息名称冲突

参数说明

参数	类型	说明
messageType	String	消息类型名
isPersited	Boolean	是否存储
isCounted	Boolean	是否计数

代码示例

```
// 初始化 IM
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);

var messageType = 's:person'; // 自定义消息类型
var isPersited = true; // 自定义消息存储属性
var isCounted = true; // 自定义消息计数属性
im.registerMessageType(messageType, isPersited, isCounted);
```

2. 发送自定义消息

参数说明

参数	类型	说明
messageType	String	消息类型名
content	Object	消息属性对象

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
}
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

常见问题

> Q1: 收到的表情信息无法解析， 表情显示有问题
> A1: 解决方案：
>
> 1. Web 端展示 Emoji 时, 都需要调用 `emojiToHTML` 转成 HTML 或者使用 `symbolToEmoji` 将 Unicode 转化成原生 Emoji 字符
> 2. 发送消息时, 必须发送原生 Emoji 字符, 如果发送 HTML, 则认定发送的是字符串

> Q2: 表情列表 每一个手机展示的表情不一样
> A2: 解决方案：
>
> 1. 消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示

2. 不同浏览器, 不同设备, 展示的原生 Emoji 表情都不同
3. 如需多端展示 Emoji 一致, 需使用 `emojiToHTML` 转化为 HTML 后再展示(此方法为以图片形式展示)
   可参照考文档进行实现：[emoji 插件]

消息接收　　功能描述

更新时间:2024-08-30

开发者可通过此接口拦截到 SDK 接收到的消息，并进行响应的业务操作。

实现方法

消息通过设置监听总的消息监听进行接收，消息监听中接收的消息不区分消息类型。收到后按需处理即可。详见消息监听。

历史消息获取　　本地获取

更新时间:2024-08-30

Web 没有本地存储，不提供本地获取方法。

远端获取

从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 IM 旗舰版或 IM 尊享版可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
timestamp	Number	否	0	获取时间戳. 0 为从当前最新时间拉取 时间戳参数名对比 3.X 已修正为 `timestamp` 单位：毫秒	4.0.0
count	Number	否	20	获取条数, 范围 1 - 20	4.0.0
order	Number	否	0	获取顺序，默认为 0， 0 表示升序：获取消息发送时间比传入 sentTime 小 的消息 1 表示倒序：获取消息发送时间比传入 sentTime 大 的消息	4.0.0

回调参数说明

参数	类型	说明
list	Array	获取的历史消息列表，返回 message 列表
hasMore	Bool	是否还有历史消息可以获取

`message` 属性说明

字段名	类型	说明
`type`	Number	会话类型
`targetId`	String	群组 ID
`senderUserId`	String	发送者 id
`content`	Object	消息内容
`messageType`	String	消息标识
`messageUId`	String	服务端存储的消息 id
`messageDirection`	Number	消息方向。 发送: 1, 接收: 2
`isOffLineMessage`	Boolean	是否为离线消息
`sentTime`	Number	消息在融云服务端的发送时间
`receivedTime`	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
`isPersited`	Boolean	消息是否存储在服务端
`isCounted`	Boolean	消息是否计数
`isMentioned`	Boolean	是否为 @ 消息

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var option = {
timestamp: +new Date(),
count: 20
};
conversation.getMessages(option).then(function(result){
var list = result.list; // 历史消息列表
var hasMore = result.hasMore; // 是否还有历史消息可以获取
console.log('获取历史消息成功', list, hasMore);
});
```

消息回执 功能描述

更新时间:2024-08-30

开发者可使用此功能实现消息已读未读功能的展示。

当 A 给 B 发送了一条消息，B 在未阅读之前 A 用户显示未读，当 B 用户阅读并调用发送回执接口之后，A 用户可在监听回执中收到通知，此时可根据对应的数据内容将发送的消息显示为已读。

此功能目前仅在 GROUP 类型的会话中开放。用户可以对自己发送的消息发起阅读回执请求，发起后，可以看到有多少人阅读过这条消息。

发送回执请求

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:RRReqMsg	不存储	不计数	不存储	不推送	无

参数说明

属性名称	属性类型	是否必填	属性说明
content	Object	是	回执内容
messageUIdString	是	messageUId 为消息的唯一标识，通过 message.messageUId 可取到	

代码示例

```
var conversation = im.Conversation.get({
type: RongIMLib.CONVERSATION_TYPE.GROUP,
targetId: '群组 ID'
})
conversation.send({
messageType: 'RC:RRReqMsg',
content: {
messageUId: '消息UId'
}
});
```

响应回执请求

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:RRRspMsg	不存储	不计数	不存储	不推送	无

参数说明

属性名称	属性类型	是否必填	属性说明
receiptMessageDic	Object	是	回复已读通知信息

`receiptMessageDic` 说明

Object 的 key 为消息发送者的 userId，value 为回复的 messageUId 数组。可参考代码示例进行理解

代码示例

```
content = {
receiptMessageDic:{
'userId': ['消息UId']
}
};

var conversation = im.Conversation.get({
type: RongIMLib.CONVERSATION_TYPE.GROUP,
targetId: '群组 ID'
})
conversation.send({
messageType: 'RC:RRRspMsg',
content
});
```

消息撤回　　消息撤回

更新时间:2024-08-30

消息发送方可通过下面方法撤回已发送成功的消息。撤回指定消息后，原消息将被删除。

参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	3.0.0
sentTime	Number	是	消息发送时间	3.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});

conversation.recall({
messageUId: 'BH5T-JG24-C445-IKQM',
sentTime: 1585638211857
}).then(function(message){
console.log('撤回消息成功', message);
});
```

监听撤回

消息通过设置监听中的消息监听进行接收，消息监听中接收 RC:RcCmd 消息，收到后按需处理即可。消息监听文档

消息转发

更新时间:2024-08-30

消息转发调用发送消息接口发送即可，调用 SDK 发送消息接口需考虑 SDK 每秒 5 条消息的限制

超过每秒 5 条限制可使用 Server API 进行发送

消息删除　　本地删除

更新时间:2024-08-30

Web 没有本地存储，不提供本地删除功能。

远端删除

通过消息 ID 删除

参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	3.0.0
sentTime	Number	是	消息发送时间	3.0.0
messageDirection	Number	是	消息发送方向	3.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.deleteMessages([
{ messageUId: '2jJ9-KU1j-OLJG-29KL', sentTime: 1580869079801, messageDirection: 1 },
{ messageUId: '8UJ9-JU9j-WSJG-92K0', sentTime: 1580869078886, messageDirection: 1 }
]).then(function(){
console.log('删除历史消息成功');
});
```

通过时间戳删除

参数说明

参数	类型	必填	说明	最低版本
timestamp	Number	是	清除时间点，该时间之前的消息将被清除 时间戳参数名对比 3.X 已修正为 timestamp	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.clearMessages({
timestamp: +new Date()
}).then(function(){
console.log('清除历史消息成功');
});
```

消息扩展　　功能描述

更新时间:2024-08-30

> ⚠ 警告
>
> 从 3.0.7 版本开始支持单条消息扩展信息设置功能。

功能描述如下：

- 对已发送并且已设置扩展标识的消息，可设置扩展信息。
- 信息以 Key、Value 的方式进行存储，单条消息可设置 300 个扩展信息。
- 该功能仅支持单聊、群聊会话类型，暂不支持在聊天室中使用。
- 如已开通历史消息云存储功能，针对某一条消息设置的扩展信息，也会同时保存到该条消息的历史记录中。
- 每个终端在设置扩展信息时，如未达到上线都可以进行设置，所在并发情况下，会出现设置超过 300 的情况，超出部分会丢弃删除。
- 设置消息扩展后，会产生内置消息. 频繁设置扩展会产生大量消息，如果对消息量增长敏感，请谨慎使用此功能。

> ⓘ 提示
>
> **适用场景:**
>
> 需要对原始消息增加状态标识的需求, 都可使用消息扩展
>
> - 比如消息评论需求，可通过设置原始消息扩展信息的方式添加评论信息
> - 比如礼物领取、订单状态变化需求，通过此功能改变消息显示状态. 向用户发送礼物，默认为未领取状态，用户点击后可设置消息扩展为已领取状态

设置监听

Web 扩展监听需在 `im.watch` 中增加 `expansion` 事件，为避免设置重复，可移步 监听设置 查看实现方法

设置扩展

发送消息设置扩展内容

参数说明:

参数	类型	必填	默认值	说明	最低版本
messageType	String	是	--	消息类型	3.0.0
content	Object	是	--	消息内容	3.0.0
canIncludeExpansion	Boolean	否	false	是否携带扩展	3.0.7
expansion	Object	否	--	携带的扩展内容	3.0.7

代码示例：

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});

conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: { // 填写开发者定义的消息内容
content: '文字消息'
},
canIncludeExpansion: true, // 是否携带扩展
expansion: { // 携带的扩展内容
key: 'value',
key2: 'value2'
}
}).then((message) => {
console.log('发送携带扩展的文字消息成功', message);
}).catch((error) => {
console.log('发送携带扩展的文字消息失败', error);
});
```

⚠ 警告

扩展限制：

1. Key 支持英文、字母、数字、+、=、-、_。
2. Key 最大长度 32 字符。
3. 如果 SDK 版本 < 4.6.0，Value 最大长度 64 字符。如果 SDK 版本 ≧ 4.6.0，Value 最大 4096 个字符。
4. 单次调用最多设置 20 个键值对。
5. 单条消息最多设置 300 个键值对。

更新扩展

更新消息的扩展内容。

- 每次设置消息扩展将会产生内置通知消息，频繁设置扩展会产生大量消息。
- 每个终端在设置扩展信息时，如未达到上限都可以进行设置；在并发情况下，会出现设置超过 300 的情况，超出部分会被丢弃。

参数说明：

参数	类型	必填	说明	最低版本
expansion	Object	是	扩展内容	3.0.7
message	Object	是	原消息体	3.0.7

代码示例：

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var expansion = {
key: '新 value',
key2: '旧 value2',
key3: '新 value3'
};
var message = {
content: {
content: '文字消息'
},
messageUId: 'BK96-PP18-OIQ6-9GPP',
canIncludeExpansion: true,
expansion: {
key2: '旧 value2'
},
// .....
};
conversation.updateMessageExpansion(expansion, message).then(() => {
console.log('更新消息扩展成功');
}).catch((error) => {
console.log('更新消息扩展失败', error);
});
```

删除扩展

参数说明：

参数	类型	必填	说明	最低版本
keys	Array	是	删除的 keys	3.0.7
messageObject		是	原消息体	3.0.7

代码示例：

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
var keys = ['key', 'key3'];
conversation.removeMessageExpansion(keys, message).then(() => {
console.log('删除消息扩展成功')
}).catch((reason) => {
console.log('删除消息扩展失败', reason)
});
```

聊天室概述

更新时间:2024-08-30

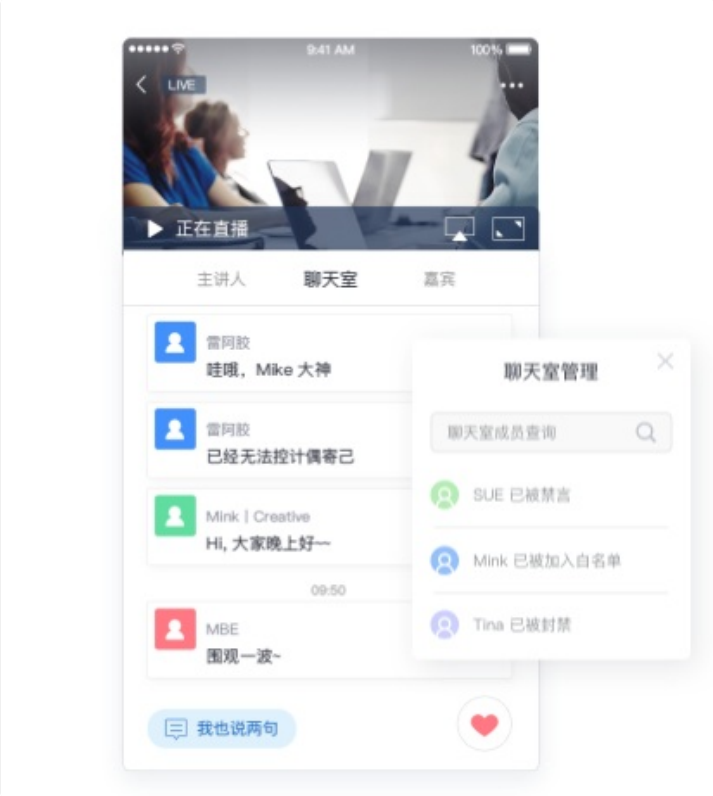
聊天室（Chatroom）提供了一种不设用户上限，支持高并发消息处理的业务形态，可用于直播、社区、游戏、广场交友、兴趣讨论等场景。聊天室业务要点如下：

- App Key 下可创建的聊天室数量没有限制，单个聊天室成员数量没有限制。
- 聊天室具有自动销毁机制，默认情况下所有聊天室会在不活跃（连续时间段内无成员进出且无新消息）达到 1 小时后踢出所有成员并自动销毁，可延长该时间，也可配置为定时自动销毁。详见服务端文档聊天室销毁机制。
- 聊天室具有离线成员自动退出机制。满足默认预设条件时，融云服务端会踢出聊天室成员，详见退出聊天室。
- 聊天室本地消息会在退出聊天室时删除。**IM 旗舰版** 与 **IM 尊享版**客户可选择启用**聊天室消息云端存储**功能，将消息存储在融云服务端。具体功能与费用以融云官方价格说明页面及计费说明文档为准。
- 聊天室不具备离线消息转推送功能，只有在线的聊天室成员可接收聊天室消息。

客户端 UI 框架参考设计

聊天室产品暂不提供聊天室会话专用的 UI 组件。您可以参考以下 UI 框架设计了解聊天室的设计思路。

- 下图**聊天室**标签中为聊天室消息列表。
- 下图**聊天室管理**窗口中展示了聊天室支持的部分能力，如禁言、封禁、白名单等。



服务配置

客户端 SDK 默认支持聊天室，不需要申请开通。

聊天室的部分基础功能与增值服务可以在控制台的免费基础功能和 IM 服务管理页面进行开通和配置。

客户端 SDK 使用须知

> IMKit 依赖 IMLib，因此 IMKit 具备 IMLib 的全部能力，包括聊天室。但请注意 IMKit 不提供开箱即用的聊天室会话 UI 组件。

聊天室功能接口

聊天室会话关系由融云负责建立并保持连接。SDK 提供加入、退出等部分聊天室管理接口。更多聊天室管理功能需要配合使用即时通讯服务端 API。下表描述了融云聊天室主要的功能接口。

功能分类	功能描述	客户端 API	融云服务端 API
创建与销毁聊天室	手动创建聊天室，或手动销毁聊天室。注意：客户端 SDK 无单独的创建聊天室 API。客户端不提供手动销毁聊天室 API。	不提供该 API	创建房间、销毁房间
加入与退出聊天室	加入聊天室时，如果聊天室未创建，则会创建聊天室再加入。如果已创建，则直接加入。	加入聊天室、退出聊天室	不提供该 API
查询聊天室房间与用户信息	• 查询聊天室房间的基础信息，包括聊天室 ID、名称、创建时间。 • 查询聊天室成员信息，支持获取聊天室成员用户 ID、加入时间，最多返回 500 个成员信息，支持按加入时间排序。	查询聊天室信息	查询房间信息
聊天室保活	添加一个或多个聊天室到聊天室保活列表。在保活列表中的聊天室不会被融云服务端自动销毁。	不提供该 API	保活房间
聊天室属性管理	在指定聊天室中设置自定义属性。比如在语音直播聊天室场景中，利用此功能记录聊天室中各麦位的属性；或在狼人杀等卡牌类游戏场景中记录用户的角色和牌局状态等。 聊天室属性以 Key-Value 的方式进行存储，支持设置、删除与查询属性，支持批量和强制操作。	聊天室属性	属性管理（KV）
封禁/解封聊天室用户	封禁一个或多个聊天室成员。被封禁成员将被踢出指定聊天室，并在封禁时间内不能再进入此聊天室中。	不提供该 API	成员封禁
聊天室用户白名单	需在IM 服务管理页面普通服务下开通后使用。 用户被加入某个聊天室的白名单后，在该聊天室消息量较大的情况下，该用户发送的消息不会被丢弃；并且用户也不会被融云服务端自动踢出该聊天室。	不提供该 API	[低级别消息/白名单]
发送聊天室消息	发送聊天室消息。	发送消息	发送聊天室消息
撤回聊天室消息	撤回聊天室消息。	撤回消息	消息撤回
获取聊天室历史消息	获取聊天室历史消息。	获取聊天室历史消息	历史消息日志

功能分类	功能描述	客户端 API	融云服务端 API
聊天室低级别消息	需在IM 服务管理页面普通服务下开通后使用。 如果消息类型在低级别消息列表中，该类型的消息全部视为低级别消息。当服务器负载高时，高级别的消息优先保留，低级别消息则优先丢弃。默认情况下，所有消息均为高级别消息。	不提供该 API	[低级别消息/白名单]
聊天室消息白名单	需在IM 服务管理页面普通服务下开通后使用。 如果消息类型在聊天室消息白名单中，该类型的消息全部受到保护，在聊天室消息量较大的情况下也不会被丢弃。	不提供该 API	[低级别消息/白名单]
聊天室成员禁言	在指定的某个聊天室中，禁言一个或多个成员。聊天室成员被禁言后，可以接收并查看聊天室中用户聊天信息，但不能通过往该聊天室内发送消息。	不提供该 API	单人禁言
全体成员禁言	设置某一聊天室全部成员禁言，或取消指定聊天室全部成员禁言状态。设置全体群成员禁言后，该聊天室的所有成员均不能通过客户端 SDK 往该群组内发送消息。	不提供该 API	全体禁言
全体禁言白名单	添加一个或多个群成员到聊天室全体成员禁言白名单。聊天室成员被添加到白名单后，即使该聊天室处于全体成员禁言状态，该成员仍可通过客户端 SDK 往该聊天室发送消息。	不提供该 API	全体禁言
全局禁言聊天室成员	需在IM 服务管理页面普通服务下开通后使用。 添加一个或多个用户到聊天室全局禁言列表中，列表中的用户在应用下的所有聊天室中都无法发送消息。	不提供该 API	全局禁言

与群组和超级群的区别

您可以通过以下文档了解业务类型之间的区别及所有功能：

- 即时通讯开发指导 · 业务类型介绍
- IM 尊享版、IM 旗舰版功能对照表

聊天室服务配置

更新时间:2024-08-30

聊天室业务本身不需要单独申请开通，但部分聊天室服务需要在控制台开通与配置，例如聊天室广播消息、聊天室消息云端存储、以及与聊天室相关的回调地址等。

聊天室服务配置主要在免费基础功能和 IM 服务管理页面。

免费基础功能

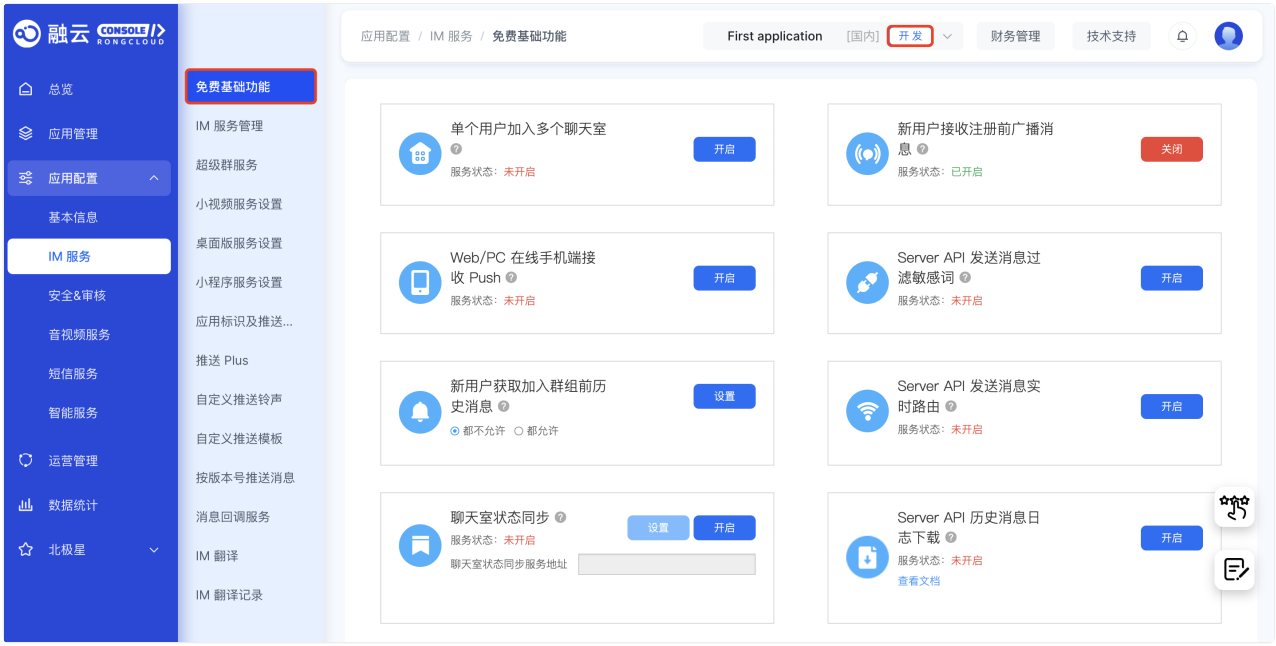
以下是聊天室业务提供的免费基础功能：

- 单个用户加入多个聊天室：默认一个用户只能加入一个聊天室中，开启后一个用户可以同时加入到多个聊天室中。
- 聊天室状态同步：聊天室状态同步是融云提供的服务端回调服务，需同时提供可正常访问的回调地址。配置成功后，在应用下聊天室发生状态变化时，将实时同步到开发者的应用服务器地址，目前支持的同步状态包括：创建、销毁、成员加入、成员退出聊天室。详见 IM 服务端文档「聊天室管理」下的聊天室状态同步。
- 加入聊天室获取指定消息设置：默认加入聊天室时可最多获取全部消息类型的最近 50 条消息，开启后可设置指定消息类型获取。
- 聊天室销毁等待时间：
  1. 可以支持配置不活跃聊天室销毁的等待时间，默认等待时间为1小时，即超过1小时不活跃即被销毁，客户可以按需调整这个时间，最长可设置24小时。
  2. 聊天室销毁时，会向聊天室成员发送聊天室销毁通知，以方便客户可以在聊天室销毁后，在终端自定义一些操作(依赖 5.1.1 及以后版本)。
  3. 聊天室增加 sessionid，在聊天室生存周期内保持不变，聊天室重建后重新生成。用于使用相同聊天室ID，多次开播时，客户端能区分出来。

- 聊天室属性自定义设置：可在指定聊天室中设置自定义属性，用于语音直播聊天室场景的会场属性同步或狼人杀等卡牌类游戏场景中记录用户的角色和牌局状态等，详见「聊天室业务」下的**聊天室属性管理（KV）**文档。如果 App 业务服务端需要融云提供聊天室属性变更数据同步，需要提供可正常访问的回调地址，配置成功后，自动开启融云提供的服务端回调服务，详见 IM 服务端文档「聊天室管理」下的聊天室属性同步。

修改服务配置

访问控制台免费基础功能页面，可调整聊天室业务相关的免费基础功能配置。



IM 旗舰版/尊享版功能

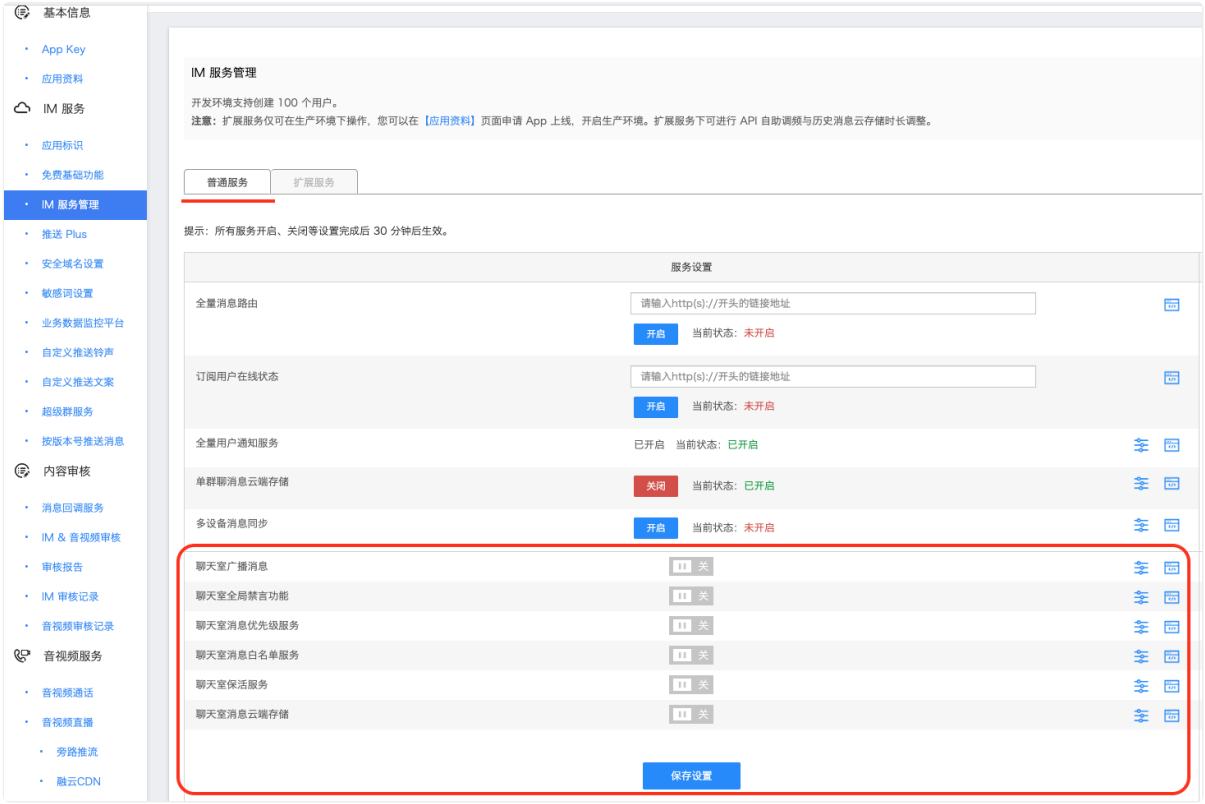
下图显示了控制台 IM 服务管理 页面与聊天室业务相关的普通服务配置。

> 开发环境下可以免费使用。生产环境下，IM 旗舰版或 IM 尊享版才能使用以下服务。

- 聊天室广播消息：向应用中的所有聊天室发送一条消息，单条消息最大 128k。详见 IM 服务端文档「消息管理」下的发送全体聊天室广播消息。
- 聊天室全局禁言功能：当不想让某一用户在所有聊天室中发言时，可将此用户添加到聊天室全局禁言中，被禁言用户可接收查看聊天室中用户聊天信息，但不能发送消息。详见 IM 服务端文档「聊天室用户管理」下的全局禁言用户。
- 聊天室消息优先级服务：在指定聊天室中设置指定类型的消息为低级别消息。当服务器负载高时低级别消息优先被丢弃，这样可以确保重要的消息不被丢弃。详见 IM 服务端文档「聊天室消息优先级服务」下的添加低级别消息。
- 聊天室白名单服务：开通后，可以使用以下功能对应的 Server API：
  - 聊天室用户白名单：可用于保护指定聊天室中的重要用户，支持按聊天室设置白名单用户。例如，App 业务中指定聊天室中的管理员、主播等重要角色的用户。
  - 聊天室消息白名单：可用于保护 App 下所有聊天室中的指定消息类型。例如 App 业务中自定义的红包消息。
- 聊天室保活服务：当聊天室中 1 小时无人说话，同时没有人加入聊天室时，融云服务端会自动把聊天室内所有成员踢出聊天室并销毁聊天室。保活的聊天室不会被自动销毁，可以调用 API 接口销毁聊天室。详见 IM 服务端文档「聊天室管理」下的保活聊天室。
- 聊天室消息云端存储：聊天消息保存在云端，用户进入聊天室后，可以查看聊天室中以前的消息，历史消息默认保存 2 个月。
- 加入聊天室获取指定消息配置：加入聊天室时只返回指定类型的消息，不返回其他类型的消息。

修改服务配置

访问开发后台 IM 服务管理 页面，切换到普通服务标签下，可启用以下聊天室服务配置开关。



您还可以在扩展服务标签下对部分服务的具体配置进行调整。

加入

更新时间:2024-08-30

默认同一用户不能同时加入多个聊天室，加入新的聊天室后，会自动退出之前的聊天室。

如需支持单个用户加入多个聊天室，请在控制台免费基础功能页面打开该配置。

加入聊天室

功能描述

1. 加入聊天室前需要先实例化。
2. 如果聊天室不存在，SDK 会创建聊天室并加入，如果已存在，则直接加入。
3. 调用加入聊天室接口时可以设置进入聊天室时的拉取消息数量。

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.0

加入参数说明

参数	类型	必填	说明	最低版本
countNumber		是	拉取消息数，由 **im.watch** 抛出 **-1** 表示不获取任何历史消息 **0** 表示不特殊设置而使用 SDK 默认的设置，默认为获取 **10** 条 传入其他值为具体获取的消息数量，最大值为 **50**	3.0.0

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

加入

```
chatRoom.join({
count: 20 // 进入后, 自动拉取 20 条聊天室最新消息
}).then(function() {
console.log('加入聊天室成功');
});
```

加入存在的聊天室

功能描述

1. 加入聊天室前需要先实例化。
2. 需加入 已存在 的聊天室。若聊天室不存在，则加入失败。
3. 调用加入聊天室接口时可以设置进入聊天室时的拉取消息数量。

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.0

加入参数说明

参数	类型	必填	说明	最低版本
countNumber		是	拉取消息数，由 **im.watch** 抛出 **-1** 表示不获取任何历史消息 **0** 表示不特殊设置而使用 SDK 默认的设置，默认为获取 **10** 条 传入其他值为具体获取的消息数量，最大值为 **50**	3.0.5

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

加入

```
chatRoom.joinExist({
count: 20 // 进入后, 自动拉取 20 条聊天室最新消息
}).then(function() {
console.log('加入聊天室成功');
});
```

退出

更新时间:2024-08-30

退出聊天室支持以下几种情况：

- **被动退出聊天室**：聊天室具有离线成员自动踢出机制。该机制被触发时，融云服务端会将用户踢出聊天室。用户如被封禁，也会被踢出聊天室。
- **主动退出聊天室**：客户端提供 API，支持由用户主动退出聊天室。

聊天室离线成员自动退出机制

聊天室具有离线成员自动退出机制。用户离线后，如满足以下默认预设条件，融云服务端会自动将该用户踢出聊天室：

- 从用户离线开始 30 秒内，聊天室中产生第 31 条消息时，触发自动踢出。
- 或用户已离线 30 秒后，聊天室有新消息产生时，触发自动踢出。

> **注意：**
>
> - 默认预设条件均要求聊天室中必须要有新消息产生，否则无法触发踢出动作。如果聊天室中没有消息产生，则无法将异常用户踢出聊天室。
> - 如需保护特定用户，即不自动踢出指定用户（如某些应用场景下可能希望用户驻留聊天室），可使用 Server API 提供的聊天室用户白名单功能。

主动退出聊天室

客户端用户可主动退出聊天室。通过 `ChatRoom.get` 获取聊天室实例，调用 `chatRoom.quit()` 退出聊天室。

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.quit().then(function() {
console.log('退出聊天室成功');
});
```

查询聊天室信息　　功能描述

更新时间:2024-08-30

获取聊天室的信息，包含部分成员信息和当前聊天室中的成员总数.

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
count	Number	否	获取人数，范围 0 - 20 1. 传入 **0** 获取到的聊天室信息将或仅包含成员总数，不包含具体的成员列表 2. 传入其他大于 **0** 的值返回聊天室信息，结果仅包含包含不多于 **20** 人的成员信息和当前成员总数。**最大值为 20**	3.0.0
order	Number	否	1. 排序方式，1 正序，2 倒序，可通过 `RongIMLib.CHATROOM_ORDER` 枚举获取 2.传入 **0** 获取到的聊天室信息将或仅包含成员总数，不包含具体的成员列表	3.0.0

回调参数说明

字段名	类型	说明
userCountNumber		聊天室用户数量
userInfos	Object	聊天室用户信息

`userInfo` 数据说明：

字段名	类型	说明
id	String	用户 id
time	Number	用户加入聊天室时间

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.getInfo().then(function(result){
var userCount = result.userCount;
var userInfos = result.userInfos;
console.log('获取聊天室信息成功', userCount, userInfos);
});
```

设置属性

更新时间:2024-08-30

聊天室属性（KV）管理接口用于在指定聊天室中设置自定义属性。

在语音直播聊天室场景中，可利用此功能记录聊天室中各麦位的属性；或在狼人杀等卡牌类游戏场景中记录用户的角色和牌局状态等。

功能局限

- 聊天室销毁后，聊天室中的自定义属性同时销毁。
- 每个聊天室中，最多允许设置 **100** 个属性信息，以 `Key-Value` 的方式进行存储。
- 客户端 SDK 未针对聊天室属性 KV 的操作频率进行限制。建议每个聊天室，每秒钟操作 `Key-Value` 频率保持在 **100** 次及以下（一秒内单次操作 100 个 KV 等同于操作 100 次）。

开通服务

使用聊天室属性（KV）接口要求开通聊天室属性自定义设置服务。您可以前往控制台的**免费基础功能**页面开启服务。

如果配置了服务端回调 URL，融云服务端会将应用下的聊天室属性变化（设置，删除，全部删除等操作）同步到指定的回调地址。详见服务端文档聊天室属性同步（KV）。

设置属性

仅聊天室中不存在此属性 或 属性设置者为自己时, 可设置成功

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.2

设置参数说明

参数	类型	必填	说明	最低版本
key	String	是	属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符	3.0.2
value	String	是	属性对应的值, 最大长度 4096 字符	3.0.2
isSendNotification	Boolean	否	设置成功后是否发送通知消息	3.0.2
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	3.0.2
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

设置

```
var key = 'role';
var value = 'Werewolf';
chatRoom.setEntry({
key: key,
value: value,
isAutoDelete: false,
isSendNotification: true,
notificationExtra: 'Change role'
}).then(function() {
console.log('设置聊天室属性成功');
});
```

批量设置属性

对聊天室属性进行批量设置。注意，仅在以下情况下允许批量设置：

- 聊天室中不存在任何当前需要设置的属性。
- 如果当前需要设置的任何属性在聊天室中已存在，仅当这些属性设置者为本人时可修改，否则会设置失败。

如果设置失败，您可以使用下面的方法 forceSetEntry 进行强制设置，但一次仅能强制设置一个属性。

参数说明

实例化参数说明

参数类型	必填	说明	最低版本
id String	是	聊天室 id	4.5.0

设置参数说明

参数	类型	必填	说明	最低版本
entries	Object	是	属性集合，属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符；属性对应的值, 最大长度 4096 字符	4.5.0
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	4.5.0
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	4.5.0

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

设置

```
var key = 'role';
var value = 'Werewolf';
const option = {
entries: {name:'name'},
isAutoDelete: false,
notificationExtra: 'Change role'
}
chatRoom.setEntries(options).then(function() {
console.log('批量设置聊天室属性成功');
});
```

强制设置属性

强制修改/创建任意聊天室属性

参数说明

实例化参数说明

参数类型	必填	说明	最低版本
id String	是	聊天室 id	3.0.2

强制设置参数说明

参数	类型	必填	说明	最低版本
key	String	是	属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符	3.0.2
value	String	是	属性对应的值, 最大长度 4096 字符	3.0.2
isSendNotification	Boolean	否	设置成功后是否发送通知消息.	3.0.2
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	3.0.2
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

强制设置

```
var key = 'count';
var value = '8';
chatRoom.forceSetEntry({
key: key,
value: value,
isAutoDelete: false,
isSendNotification: true,
notificationExtra: 'Change Count'
}).then(function() {
console.log('强制设置聊天室属性成功');
});
```

删除属性　　删除属性

更新时间:2024-08-30

仅能删除自己设置的聊天室属性

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.2

设置参数说明

参数	类型	必填	说明	最低版本
key	String	是	属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符	3.0.2
isSendNotification	Boolean	否	删除成功后是否发送通知消息	3.0.2
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

删除

```
var key = 'role';
chatRoom.removeEntry({
key: key,
isSendNotification: true,
notificationExtra: 'Change role'
}).then(function() {
console.log('删除聊天室属性成功');
});
```

强制删除属性

强制删除任意聊天室属性

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.2

强制删除参数说明

参数	类型	必填	说明	最低版本
key	String	是	属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符	3.0.2
isSendNotification	Boolean	否	删除成功后是否发送通知消息	3.0.2
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

删除

```
var key = 'role';
chatRoom.forceRemoveEntry({
key: key,
isSendNotification: true,
notificationExtra: 'Change role'
}).then(function() {
console.log('删除聊天室属性成功');
});
```

获取属性　　获取单个属性

更新时间:2024-08-30

获取指定属性信息

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.2

获取参数说明

参数	类型	必填	说明	最低版本
key	String	是	属性名称, 支持英文字母、数字、+、=、-、_ 的组合方式, 最大长度 128 字符	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

获取

```
var key = 'role';
chatRoom.getEntry(key).then(function(value) {
console.log('获取聊天室属性信息成功', value);
});
```

获取所有属性

参数说明

实例化参数说明

参数	类型	必填	说明	最低版本
id	String	是	聊天室 id	3.0.2

代码示例

实例化

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
```

获取属性

```
chatRoom.getAllEntries().then(function(entries) {
console.log('获取所有聊天室属性信息成功', entries);
});
```

设置回调

更新时间:2024-08-30

Web 回调方法需在 `im.watch` 中增加 `chatroom` 事件，为避免设置重复，可移步 监听设置

消息发送　　属性描述

更新时间:2024-08-30

消息属性	描述
消息类名	各端消息名
存储属性	存储 / 不存储
离线属性	缓存 / 不缓存
推送属性	是/否
ObjectName	传输层名称，与消息类名一一对应
计数属性	计数 / 不计数
消息尺寸	128 KB
推送内容	详见各消息类送方式

存储属性

存储属性	存储分类	支持平台	详细描述
存储	客户端	Android、iOS	发送、接收该消息后，本地数据库存储 Web 端 和 小程序端因本地存储不可靠，不支持客户端消息存储，但可通过历史消息云存储服务获取历史记录
存储	云端	Android、iOS、Web	默认不在云端进行存储，需开通单群聊消息云存储服务，开通后，可在融云服务端存储 6 个月的历史消息，供客户端按需拉取
不存储	客户端	Android、iOS	发送、接收该消息后，本地数据库不存储
不存储	云端	Android、iOS、Web	无论是否开通历史消息云存储服务，该消息均不存储

计数属性

接收收到消息时，会话是否累计未读数。

计数属性	支持平台	详细描述
计数	iOS、Android、Web	会话未读消息数 + 1，该属性只影响会话列表未读数计数，App 应用角标可根据每个会话列表未读数累加获得
不计数	iOS、Android、Web	会话未读消息数不变

离线属性

接收人当前不在线时，是否进行离线缓存。

离线属性	详细描述
存储	消息进行离线缓存，默认 7 天。接收人在 7 天内上线，均可接收到该消息。超过 7 天后，消息被离线缓存淘汰。如有需要，可通过云端存储拉取到该消息
不存储	消息不进行离线缓存，所以只有接收人在线时，才可收到该消息。该消息不进行历史消息云存储、不进入云端存储（Log 日志）、不进行消息同步（消息路由）

推送属性

接收人是否接收推送，当离线属性为 `存储` 时，该属性生效。离线属性为 `不存储` 时属性无效。

由于 Web、小程序、PC 端没有推送平台，无法收到推送提醒。

推送属性	平台	推送方式	详细描述
推送	iOS、Android	APNs、华为、小米、魅族、OPPO、vivo、FCM、融云	当有离线缓存消息时，进行远程推送提醒，内容为该推送提醒显示的内容
不推送	iOS、Android	--	当有离线缓存消息时，不进行远程推送提醒

⚠ 警告

1. 发送消息必须在成功连接融云服务器成功之后进行。
2. 每秒最多发送 5 条消息。

消息结构

字段名	类型	说明
type	Number	会话类型
targetId	String	接收方的 userId
senderUserId	String	发送者 id
content	Object	消息内容
messageType	String	消息标识
messageUId	String	服务端存储的消息 id
messageDirection	Number	消息方向。 发送: 1, 接收: 2
isOffLineMessage	Boolean	是否为离线消息
sentTime	Number	消息在融云服务端的发送时间
receivedTime	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
isPersited	Boolean	消息是否存储在服务端
isCounted	Boolean	消息是否计数

参数说明

参数	类型	必填	默认值	说明	最低版本
messageType	String	是	--	消息类型. 与移动端 ObjectName 一致	3.0.0
content	String	是	--	消息内容	3.0.0
isPersited	Boolean	否	true	是否存储在服务端	3.0.0
isCounted	Boolean	否	true	是否计数. 计数消息接收端接收后未读数加 1	3.0.0
pushContent	String	否	--	Push 显示内容	3.0.0
pushData	String	否	--	Push 通知时附加信息	3.0.0
isVoipPush	String	否	false	为 ture 时, 对端不在线的 iOS 会收到 Voip Push. Android 无影响	3.0.0
isStatusMessage	Boolean	否	false	是否发送状态消息，**设置为 true 后 isPersited 和 isCounted 属性失效**	3.0.0

发送代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
},
isPersited: true,// 是否存储在服务端,默认为 true
isCounted: true, // 是否计数. 计数消息接收端接收后未读数加 1，默认为 true
pushContent:'user 发送了一条消息', // Push 显示内容
pushData: 'Push 通知时附加信息', // Push 通知时附加信息, 可不填
isStatusMessage: false // 设置为 true 后 isPersited 和 isCounted 属性失效
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

发送消息

文本消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:TxtMsg 存储 计数 存储 推送 消息内容
参数说明

属性名称属性类型是否必填属性说明
content String 是 文本消息内容
extra String 否 附加信息，一般为消息不显示消息内容
代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: 'Hello RongCloud' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

图文消息

消息说明

messageType 存储属性计数属性离线属性推送属性推送内容
RC:ImgTextMsg 存储 计数 存储 推送 消息内容
参数说明

属性名称属性类型是否必填属性说明
content String 是 图文内容
title String 是 图文标题
imageUri String 是 图片上传到服务器的 url
url String 是 富文本消息点击后打开的 URL
extra String 否 附加信息，一般为消息不显示消息内容
代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});

chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.RICH_CONTENT, // 'RC:ImgTextMsg'
content: {
title: '图文标题',
content: '图文内容',
imageUri: '图片上传到服务器的 url',
url: '富文本消息点击后打开的 URL'
}
}).then(function(message){
console.log('发送图文消息成功', message);
});
```

Emoji 消息

- web 端发送 Emoji 消息，开发者直接使用 文本消息 发送即可。

- 融云提供 Emoji 插件，内置了 128 个 Emoji 表情的图片，做消息输入框的表情选项，也可自行扩展配置。
- 发消息时，必须直接发送 Emoji 原生字符。如：😊，转换方法： `symbolToEmoji` 。
- Web SDK 接收消息时接收到的是 Unicode 编码格式，如："ef600" 需要转化才能正确显示原生 Emoji。

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 'RC:TxtMsg'
content: {
content: '😊' // 文本内容
}
}).then(function(message){
console.log('发送文字消息成功', message);
});
```

Emoji 插件

插件兼容性

Chrome	Firefox	Safari	IE	Edge	iPhone	Android
30+	30+	10+	7+✔		iOS 8.0+ 的Safari浏览器以及微信浏览器	4.4+系统的Chrome浏览器以及微信浏览器

Emoji 插件引入

```
<!-- 非压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- 压缩版 -->
<script src="https://cdn.ronghub.com/RongEmoji-2.2.10.min.js"></script>
```

> Emoji 代码示例：https://rongcloud.github.io/web-emoji-demo/src/index.html

> ⚠ 警告
>
> 1. 使用 `import * as RongIMLib from '@rongcloud/imlib-v4-adapter'` 方式引入 SDK 时表情插件调用需要使用 `window` 前缀。例如：`window.RongIMLib.RongIMEmoji.init()`
> 2. RongEmoji 插件仅支持 cdn 引入方式，暂不支持 npm 引入

Emoji 初始化：默认参数初始化

```
RongIMLib.RongIMEmoji.init();
```

Emoji 初始化：自定义表情配置初始化

config 参数说明：

参数	类型	必填	说明	最低版本
size	Number	否	表情大小, 默认 24, 建议 18 - 58	2.2.6
url	String	否	Emoji 背景图片 url	2.2.6
lang	String	否	Emoji 对应名称语言, 默认 zh	2.2.6
extension	Object	否	扩展表情	2.2.6

```
// 表情信息可参考 http://unicode.org/emoji/charts/full-emoji-list.html
var config = {
size: 25,
url: '//f2e.cn.ronghub.com/sdk/emoji-48.png',
lang: 'en',
extension: {
dataSource: {
u1F914: { // 自定义 u1F914 对应的表情
en: 'thinking face', // 英文名称
zh: '思考', // 中文名称
tag: '', // 原生 Emoji
position: '0 0' // 所在背景图位置坐标
}
},
url: '//cdn.ronghub.com/thinking-face.png' // 新增 Emoji 背景图 url
}
};
RongIMLib.RongIMEmoji.init(config);
```

获取列表

```
var list = RongIMLib.RongIMEmoji.list;
/*list => [{
unicode: 'u1F600',
emoji: '',
node: span,
symbol: '[笑嘻嘻]'
}]
*/
```

Emoji 转文字

在不支持原生 Emoji 渲染时，可显示对应名称，适用于消息输入。

```
var message = '😁测试 Emoji';
// 将 message 中的原生 Emoji 转化为对应名称
RongIMLib.RongIMEmoji.emojiToSymbol(message);
// => '[笑嘻嘻][露齿而笑]测试 Emoji'
```

文字转 Emoji

发送消息时，消息体里必须使用原生 Emoji 字符。

```
var message = '[笑嘻嘻][露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为原生 Emoji
RongIMLib.RongIMEmoji.symbolToEmoji(message);
// => '😁😁测试 Emoji'
```

Emoji 转 HTML

Web SDK 接收消息后，消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示。

```
var message = '\uf600测试 Emoji';
// 将 message 中的原生 Emoji（包含 Unicode ）转化为 HTML
RongIMLib.RongIMEmoji.emojiToHTML(message);
// => "<span class='rong-emoji-content' name='[笑嘻嘻]'>😁</span>测试 Emoji"
```

文字 转 HTML

```
var message = '[露齿而笑]测试 Emoji';
// 将 message 中的 Emoji 对应名称转化为 HTML
RongIMLib.RongIMEmoji.symbolToHTML(message);
// => "<span class='rong-emoji-content' name='[露齿而笑]'>😁</span>测试 Emoji"
```

位置消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:LBSMsg 存储 计数 存储 推送 [位置]

参数说明

属性名称	属性类型	是否必填	属性说明
longitude	Number	是	经度
latitude	Number	是	维度
poi	String	是	位置信息
content	String	是	位置缩略图,图片需要是不带前缀的 base64 字符串
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
var latitude = 40.0317727;
var longitude = 116.4175057;
var poi = '北苑路 融云';
var content = '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE'; // 位置图片 base64
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.LOCATION, // 'RC:LBSMsg'
content: {
latitude: latitude,
longitude: longitude,
poi: poi,
content: content
}
}).then(function(message){
console.log('发送位置消息成功', message);
});
```

正在输入状态消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:TypSts	不存储	不计数	不存储	不推送	无

参数说明

属性名称	属性类型	是否必填	属性说明
typingContentType	String	是	正在输入的消息 ObjectName
data	String	否	携带信息

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: 'RC:TypSts',
content: {
typingContentType: 'RC:TxtMsg' // 正在输入的消息类型
},
isStatusMessage: true // 正在输入建议发送状态消息，不存储、不计数，且仅在线用户可收到
}).then(function(message){
console.log('发送正在输入消息成功', message);
});
```

图片消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。

- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。

发送图片消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:ImgMsg	存储	计数	存储	推送	[图片]

参数说明

属性名称	属性类型	是否必填	属性说明
content	String	是	图片的略缩图，必须是 base64 字符串, 类型必须为 jpg，base64 字符串大小不可超过 10 KB，base64 略缩图必须不带前缀
imageUri	String	是	上传到服务器的 url. 用来展示高清图片
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.IMAGE, // 'RC:ImgMsg'
content: {
content: '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....', // 压缩后的 base64 略缩图，用来快速展示图片
imageUri: 'https://www.rongcloud.cn/images/newVersion/log_wx.png' // 上传到服务器的 url. 用来展示高清图片
}
}).then(function(message){
console.log('发送图片消息成功', message);
});
```

图片上传

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload 。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 `window` 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用 getFileToken 方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

4. 开始上传图片。

- config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

- **代码示例：**

```
//创建 input 上传按钮
<input id="uploadFile" type="file">
```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initImage(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名, 上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name

参数	类型	必填	说明
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethodNumber		否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.IMAGE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

语音消息

> ⚠ 警告
>
> 1. SDK 目前不提供录音功能, 开发者需生成语音 url 后, 传入发送消息接口，发送语音消息
> 2. 语音上传请参照 文件上传 进行是实现。

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:HQVCMsg	存储	计数	存储	推送	[语音]

参数说明

属性名称	属性类型	是否必填	属性说明
remoteUrl	String	是	媒体内容上传服务器后的网络地址
type	String	否	编解码类型，默认为 aac 音频
duration	Number	否	语音消息的时长，最长为 60 秒（单位：秒）
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.HQ_VOICE, // 'RC:HQVCMsg'
content: {
remoteUrl: 'https://rongcloud-audio.cn.ronghub.com/audio_amr__RC-2020-03-17_42_1584413950049.aac?
e=1599965952&token=CddrKW5AbOMQaDRwc3ReDNvo3-sL_SO1fSUBKV3H:CDngyWj7ZApNmAfoecng7L_3SaU=', // 音频 url, 建
议格式: aac
duration: 6, // 音频时长
type: 'aac'
}
}).then(function(message){
console.log('发送语音消息成功', message);
});
```

文件消息

- 发送图片消息只需要图片上传后的 url，和图片的略缩图。
- 融云提供上传插件，文件默认存储到七牛云，插件不包含发送消息。
- 插件提供的是上传方式，开发者也可通过自己的方式将文件上传至自己文件服务。
- 融云默认上传文件存储有效期为 `6` 个月， 上传的文件不支持迁移。

发送文件消息

消息说明

messageType	存储属性	计数属性	离线属性	推送属性	推送内容
RC:FileMsg	存储	计数	存储	推送	[文件] + 文件名，如：[文件] 123.txt

参数说明

属性名称	属性类型	是否必填	属性说明
name	String	是	文件名称
size	Number	是	文件大小，单位：bytes
type	String	是	文件类型
fileUrl	String	是	上传到服务器的 url
extra	String	否	附加信息，一般为消息不显示消息内容

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
var name = 'RongIMLib.js'; // 文件名
var size = 1024; // 文件大小
var type = 'js'; // 文件类型
var fileUrl = 'https://cdn.ronghub.com/RongIMLib.js'; // 文件地址
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.FILE, // 'RC:FileMsg'
content: {
name: name,
size: size,
type: type,
fileUrl: fileUrl
}
}).then(function(message){
console.log('发送文件消息成功', message);
});
```

文件上传

> Web 平台上传图片请使用以下步骤。您可以参考上传图片代码示例：https://github.com/rongcloud/rongcloud-web-im-upload。

Web 兼容性说明：

Chrome	Firefox	Safari	IE	Edge
49+	52+	✔	10+	✔

1. (ESModule) 如果您 ESModule 模式引入 IMLib，必须将 RongIMLib 挂载到 window 上。

```
// 建议尽快升级到 adapter SDK，import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
window.RongIMLib = RongIMLib
```

2. 引入上传插件。

```
<script src = "./send-data.js"></script>
<script src = "../upload.js"></script>
<script src="./init.js"></script>
```

3. 获取上传 token。

> 必须在 IM SDK 连接成功后调用getFileToken方法。

代码示例：

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
im.getFileToken(fileType).then(function(data) {
console.log('上传 token 为', data.token);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
```

4. 开始上传图片。

- config 参数说明：

参数	类型	必填	说明
domain	String	是	上传地址，默认为七牛: https://upload.qiniup.com
fileType	Number	是	上传类型
getTokenFunction		是	获取 token 回调

- 代码示例：

```
//创建 input 上传按钮
<input id="uploadFile" type="file">
```

```
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;

var config = {
domain: '',
fileType: fileType,
getToken: function(callback) {
im.getFileToken(fileType).then(function(data) {
callback(data.token, data);
}).catch(function(error) {
console.log('获取上传 token 失败', error);
});
}
};

var uploadCallbacks = {
onProgress: function(loaded, total) {
var percent = Math.floor(loaded / total * 100);
console.log('已上传: ', percent);
},
onCompleted: function(data) {
// 上传完成, 调用 getFileUrl 获取文件下载 url
console.log('上传成功: ', data);
},
onError: function(error) {
console.error('上传失败', error);
}
};

var uploadEl = document.getElementById("uploadFile");
uploadEl.onchange = function() {
var _file = this.files[0]; // 上传的 file 对象
UploadClient.initFile(config, function(uploadFile) {
uploadFile.upload(_file, uploadCallbacks);
});
}
```

5. 获取图片下载 url。

> fileType 值必须与 getFileToken 时传入的 fileType 值一致。

- **参数说明：**

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FILE_TYPE 获取
filename	String	否	上传后的文件名，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 ,对应属性 data.filename
oriname	String	否	文件原名，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取 , 对应属性 data.name
data	Object	否	uploadCallbacks 的 onCompleted 回调中返回的数据
uploadMethod	Number	否	上传方式，从上传成功返回的数据中取 `data.uploadMethod`(1:七牛，2: 阿里)，默认为七牛，SDK v4.1.0 以上支持

- **代码示例：**

```
// data 通过 uploadFile.upload 获取
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);
var fileType = RongIMLib.FILE_TYPE.FILE;
var filename = data.filename; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var oriname = data.name; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
var uploadMethod = data.uploadMethod; // 通过 uploadCallbacks 的 onCompleted 中返回的 data 获取
im.getFileUrl(fileType, filename, oriname, data, uploadMethod).then(function(data) {
console.log('文件 url 为: ', data.downloadUrl);
}).catch(function(error) {
console.log('获取文件 url 失败', error);
});
```

小视频消息

::: warning

1. 此消息类型 Web 端 SDK 仅支持解析展示，不提供录制。
2. 如 Web 端需要发送小视频消息，小视频录制需要开发者自行实现。
3. 如果 App Key 使用 **IM 旗舰版**或 **IM 尊享版**，文件存储时长默认为 180 天（不含小视频文件，小视频文件存储 7 天）。注意，**IM 商用版（已下线）默认存储 7 天。如需了解IM 旗舰版或 IM 尊享版的具体功能与费用，请参见融云官方价格说明页面及计费说明。**

:::

消息说明

> ⚠ 警告

messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:SightMsg 存储 计数 存储 推送 [小视频]

参数说明

参数	类型	说明
sightUrl	Number	上传到文件服务器的小视频地址
content	String	小视频首帧的缩略图进行 Base64 编码的结果值，格式为 JPG，注意在 Base64 进行 Encode 后需要将所有 \r\n 和 \r 和 \n 替换成空
duration	Object	视频时长，单位：秒
size	Object	视频大小单位 bytes
name	Function	发送端视频的文件名，小视频文件格式为 mp4。

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1',
})
chatRoom
.send({
messageType: RongIMLib.MESSAGE_TYPE.SIGHT, // 'RC:SightMsg'
content: {
content: '/9j/4AAQSkZ/2wB...hDSaSiimB//9k=',
sightUrl: 'http://rongcloud...com/video...',
duration: 10,
size: 734320,
name: 'video_xx.mp4',
},
})
.then(function(message) {
console.log('发送小视频消息成功', message)
})
```

GIF 消息

消息说明

> ⚠ 警告
>
> messageType 与移动端 ObjectName 相对应。

messageType存储属性计数属性离线属性推送属性推送内容
RC:GIFMsg 存储 计数 存储 推送 [图片]

参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 KB
remoteUrl	String	GIF 图片的服务器地址
width	Number	GIF 图的宽
height	Number	GIF 图的高

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: RongIMLib.MESSAGE_TYPE.GIF, // 'RC:GIFMsg'
content: {
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
}).then(function(message){
console.log('发送 GIF 消息成功', message);
});
```

发送自定义消息

参数说明

参数	类型	说明
messageType	String	消息类型名
content	Object	消息属性对象

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
},
isPersited: true,// 是否存储在服务端,默认为 true
isCounted: true // 是否计数. 计数消息接收端接收后未读数加 1，默认为 true
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

常见问题

Q1: 收到的表情信息无法解析， 表情显示有问题

A1: 解决方案：

1. Web 端展示 Emoji 时, 都需要调用 `emojiToHTML` 转成 HTML 或者使用 `symbolToEmoji` 将 Unicode 转化成原生 Emoji 字符
2. 发送消息时, 必须发送原生 Emoji 字符, 如果发送 HTML, 则认定发送的是字符串

Q2: 表情列表 每一个手机展示的表情不一样

A2: 解决方案：

1. 消息体内的原生 Emoji 字符会被解码为对应 Unicode 码，需调用转化方法才能正确显示
2. 不同浏览器, 不同设备, 展示的原生 Emoji 表情都不同
3. 如需多端展示 Emoji 一致, 需使用 `emojiToHTML` 转化为 HTML 后再展示(此方法为以图片形式展示)
   可参照考文档进行实现：emoji 消息

消息接收　　功能描述

更新时间:2024-08-30

开发者可通过此接口拦截到 SDK 接收到的消息，并进行响应的业务操作。

实现方法

消息通过设置监听总的消息监听进行接收，消息监听中接收的消息不区分消息类型。收到后按需处理即可。[设置监听]文档。

消息撤回　　消息撤回

更新时间:2024-08-30

消息发送方可通过下面方法撤回已发送成功的消息。撤回指定消息后，原消息将被删除。

参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	3.0.7
sentTime	Number	是	消息发送时间	3.0.7
user	Object	否	用户信息	3.0.7

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
chatRoom.recall({
messageUId: 'BH5T-JG24-C445-IKQM',
sentTime: 1585638211857,
user: { // 可不填，填入则消息 content 中携带 user 用户信息
id: 'user1',
name: '张三',
portrait: 'https://cdn.ronghub.com/thinking-face.png'
}
}).then(function(message){
console.log('撤回消息成功', message);
});
```

监听撤回

消息通过设置监听中的消息监听进行接收，消息监听中接收 RC:RcCmd 消息，收到后按需处理即可。消息监听文档。

历史消息获取　　功能描述

更新时间:2024-08-30

聊天室会话的历史消息可保存在云端，用户进入聊天室后，可以查看聊天室中以前的消息，历史消息默认保存 2 个月。该功能要求您已在控制台 IM 服务管理 页面为当前使用的 App Key 开启聊天室消息云端存储服务。**IM 旗舰版或IM 尊享版**可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
timestamp	Number	否	0	获取时间戳. 0 为从当前最新时间拉取 时间戳参数名对比 3.X 已修正为 `timestamp` 单位：毫秒	4.0.0
count	String	否	20	获取条数, 范围 1 - 20	4.0.0
order	Number	否	0	获取顺序，默认为 0， 0 表示升序：获取消息发送时间比传入 sentTime 小 的消息 1 表示倒序：获取消息发送时间比传入 sentTime 大 的消息	4.0.0

回调参数说明

参数	类型	说明
list	Array	获取的历史消息列表，返回 message 列表
hasMore	Bool	是否还有历史消息可以获取

`message` 属性说明

字段名	类型	说明
type	Number	会话类型
targetId	String	接收方的 userId
senderUserId	String	发送者 id
content	Object	消息内容
messageType	String	消息标识
messageUId	String	服务端存储的消息 id
messageDirection	Number	消息方向。 发送: 1, 接收: 2
isOffLineMessage	Boolean	是否为离线消息
sentTime	Number	消息在融云服务端的发送时间
receivedTime	Number	消息接收时间. isOffLineMessage 为 true 时, receivedTime 无效
isPersited	Boolean	消息是否存储在服务端
isCounted	Boolean	消息是否计数

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
var option = {
timestamp: +new Date(),
count: 20
};
chatRoom.getMessages(option).then(function(result){
var list = result.list; // 历史消息列表
var hasMore = result.hasMore; // 是否还有历史消息可以获取
console.log('获取聊天室历史消息成功', list, hasMore);
});
```

聊天室常见问题　　聊天室产品问题

更新时间:2024-08-30

融云 IM 聊天室中是否包含视频流？

IMLib 聊天室中不包含视频直播，包含与 IM 相关功能，如消息发送、聊天室控制等。

视频直播视频流的录制、播放可使用融云 RTC SDK 低延迟直播。

聊天室登录、重连问题

登录聊天室是否要先连接融云？

必须先连接融云，所有接口都是连接融云后才可以调用。

如何实现匿名登录？

从融云 SDK 的角度，都是需要确定一个 UserId 来与融云服务器连接，以保证其后所有的操作都有指向性。

但作为 App 应用层来讲，可以不暴露融云的 UserId，所谓匿名就是以一个随机的或者说不跟你们应用帐号绑定的 UserId 来登录。

如何处理断线情况？

进入聊天室后如果出现断开连接，当与融云服务器的连接恢复后，SDK 会尝试重连聊天室。重连过程不需要用户参与，用户可以设置聊天室状态监听，来获取聊天室重连的状态信息。

聊天室消息问题

如何在进入聊天室时获取最新消息？

对于同一个聊天室，只存储该聊天室的 50 条最新消息，也就是说移动端用户进入聊天室时，最多能够拉取到最新的 50 条消息。

可通过融云加入聊天室方法，设置获取历史消息条数，加入后可获取到对应条数的历史消息。

如何发送用户进入聊天室消息？

如用户加入聊天室时，需要发送“XX加入聊天室”的通知消息，可通过融云 Server API 发送聊天室消息接口，自行向聊天室中发送一条消息，可以在消息体中附加其他需要的信息属性。

如何向应用下所有聊天室发送消息？

在直播聊天场景下，如需要向应用下所有聊天室发送消息时，可使用聊天室广播消息功能，发送后所有在聊天室中的用户都将收到此条消息，如：聊天室管理通知、优惠活动系统通知等。

直播聊天室中有哪些常见消息类型？

常见消息类型包含：文字、语音、图片、点赞、礼物、弹幕及自定义消息实现的命令消息，用来辅助直播业务。

聊天室中的哪些功能可以用消息实现？

文字、语音、图片、点赞、礼物、弹幕、开播通知等功能均可使用消息实现。

直播聊天室中如何准确的使用消息类型？

由于当消息密集时，服务会自动抛弃消息，开发者设计消息类型一定要将文字、语音、图片、点赞、礼物等业务消息，单独定义消息类型使用，不要通过一个消息类型的不同属性实现功能。

如何发送展示点赞消息？

通过融云提供的自定义消息功能，定义礼物的消息类型，实现点赞功能，鉴于有些用户会连续点击，建议做消息合并机制处理。比如设置定时器，每 5 秒触发一次，将 5 秒内所有的点赞数一次性发出去，降低服务器压力，保证重要消息的畅达。

如何发送展示礼物消息？

通过融云提供的自定义消息功能，定义礼物的消息类型，实现礼物功能，展示方式就是界面上的一个 View，完全可以根据你们的需求来自定义。因为聊天室类型的应用 UI 都是特殊定制的，作为融云 SDK 来讲，只是负责消息通知，并不负责 UI 绘制。

什么是实时消息路由？

由融云服务端把指定 App Key 下产生所有的消息回调到到开发者服务器的过程叫实时消息路由，消息到达开发者服务器后便于对聊天室消息进行数据实时分析和挖掘。

调用客户端 SDK 接口发送消息为什么自己发的自己收不到？

自己发送的消息是 不会触发消息监听 的，自己将发送成功后的 Message 对象渲染到页面即可。

调用 Server API 接口发送消息为什么自己发的自己收不到？

Server API 发送的消息, 默认发送方是不会触发消息监听的，可以在本地模拟一条 Message 对象渲染到页面即可。

或者在使用 Server API 发消息时，isIncludeSender 填 1，则发送者也可收到发送的消息。

断网后再连接为何有时可收到消息，有时无法收到消息？

聊天室中用户在离线 30 秒后或离线后聊天室中产生 30 条消息时会被自动退出聊天室，如有需求不希望自动退出，可将用户加入到聊天室白名单中。

聊天室用户问题

如何显示用户昵称和头像？

融云不维护用户的信息，所以用户的昵称和头像信息都需要您自己来维护管理。如果您基于 IMKit 用户，可参见「用户信息显示」相关章节。

如何显示用户列表？

用户列表显示逻辑是根据您的需求来确定。举一个例子：聊天室人数从几百到十几万不等，比如需求定为只显示 10 个用户的头像，那就可以根据用户等级或者加入时间的顺序，选取 10 个在线用户的信息来显示。可以由应用自己的服务器来维护这个用户列表，不定时通知给全体成员，也可以每个用户加入/退出聊天室时发送状态消息，由端上接收并维护这个列表。显示的方法也是移动端进行 UI 绘制。

如何显示用户等级信息？

融云不维护用户的信息，所以关于用户信息和等级的显示，都需要您自己进行开发。通常有两种处理方式：

1. 在收到消息需要展示用户信息的时候，您可以通过您的 Server 端接口获取到用户信息来展示。
2. 用户在发送消息时把自己的信息附加到消息中，接收方通过解析消息获得用户信息。

一个用户是否可多个端同时加入聊天室？

融云默认支持一个 Android、iOS、Web、小程序用户同时在线，如果同种设备需要多端登录，需要单独开通。Web 每个浏览器 Tab 算一个端。

开通位置：控制台 -> 选择应用 -> 服务管理 -> IM 服务管理 -> 多设备消息同步

一个用户是否可以加入多个聊天室？

融云默认允许一个用户（不区分平台）同时加入一个聊天室，如有加入多聊天室需求，需至控制台开通。

开通位置: 控制台 -> 选择应用 -> 服务管理 -> 免费基础功能 -> 单个用户加入多个聊天室

什么是用户白名单？

白名单中用户发送的消息受到保护，在聊天室消息量较大的情况下不会被丢弃，可将重要用户加入用户白名单，保证重要用户发送消息不会被抛弃，如：主播、场控等。

聊天室管理功能

如何实现聊天室踢人功能？

可通过调用融云 Server API 聊天室封禁服务接口，实现将用户踢出聊天室。

如何实现在聊天室中禁言用户？

可通过融云 Server API 接口，设置聊天室禁言用户，被禁言用户不能在聊天室中发送消息，详细请查看聊天室成员禁言服务。

如何实现聊天室成员自动退出机制？

聊天室中用户离线后从第一条新消息产生开始计时，倒数 30 秒退出聊天室。或离线后聊天室中产生 30 条消息时会被自动退出聊天室。如应用场景中有驻留用户，不允许被自动退出聊天室，则需要将用户加入到聊天室白名单中。

如何设置聊天室用户白名单？

可通过融云 Server API 接口，设置聊天室白名单，详细请查看聊天室用户白名单服务。

聊天室如何销毁？

销毁方式有两种:

1. 主动销毁聊天室。该方式要求需要调用融云 Server API 的销毁房间接口

2. 或聊天室中一小时内无人说话，同时没有人加入聊天室时，融云服务端会自动把聊天室内所有成员踢出聊天室并销毁聊天室

如何避免聊天室被自动销毁？

可通过融云 Server API 接口，设置需要保活的聊天室，设置后聊天室不会被自动销毁，详细请查看聊天室保活服务。

如何实现用户如何同时加入多个聊天室？

默认同一用户不能同时加入多个聊天室，可通过提交工单方式开启用户同时加入多个聊天室功能。同时，在开启多设备消息同步情况下，多端用户可同时加入到多个聊天室。

如何动态获取聊天室在线人数？

调用 Server API 中的查询聊天室内用户方法，可通过返回值 `total` 获取聊天室中在线人数。

聊天室消息进阶功能

什么是消息优先级？

融云消息服务中，消息存在高级别、低级别之分 API，可设置消息界别的高低，默认全部为高级别，当服务器负载高时低级别消息优先被丢弃，让出资源给 高级别消息，确保高级别消息不被丢弃，消息多时可主动抛弃指定消息，如: 点赞消息

具体可参考文档: [低级别消息/白名单]

消息的分级与抛弃模型是什么？

当聊天室内人数众多，消息量会变得非常大，这时可能会出现服务端超过预设承载能力或者分发的消息量超过客户端消息接收能力的情况，这时，就需要引入消息分级机制。

融云并没有对任何消息进行抛弃，但是在消息量极大的情况下，比如 1 万人到百万左右的聊天室内，消息并发量极大的情况下，每个用户端能收发到的消息和体验已经很有限，因此消息抛弃指的是确保用户端总是能收到最重要的消息，因此不重要的消息看起来就像是被抛弃了。

在开发过程中，除官方的普通文本消息之外，开发者需要针对不同的消息类别定义不同的消息类型，以便通过消息的 ObjectName 设置消息分级。目前融云支持两级消息分类，分别是高优先级消息和低优先级消息。当发生消息抛弃行为时，优先抛弃低优先级消息。

消息量大时的丢弃策略是什么？

聊天室场景下，融云服务端默认单个聊天室中上行消息处理能力是每 200 毫秒 40 条，其中 20 条为高优先级消息使用配额，另外 20 条为高优先级和低优先级消息共同使用。

在聊天室消息量较大的情况下，融云服务器会按消息发送的时间顺序，将超出消费上限的最新消息丢弃，确保服务器稳定。

针对以上情况，为保证聊天室中重要消息不被丢弃，融云提供了以下服务：

1. 聊天室用户白名单功能，白名单中用户发送的消息受到保护，在聊天室消息量较大的情况下也不被丢弃。
2. 聊天室消息白名单功能，该名单中的消息受到保护，在聊天室消息量较大的情况下也不被丢弃。
3. 聊天室低级别消息设置，该功能为设置低优先级的消息类型，在聊天室消息量较大的情况下，此类型的消息将被优先抛弃，确保重要消息不被丢弃。

注：未设置情况下融云的所有消息均为高优先级消息。

以上功能设置后，服务端收到聊天室上行消息时，根据消息类型的设置状态，处理逻辑如下：

1. 上行消息为低优先级消息，则占用高优先级和低优先级共有的 20 条消息配额，如配额已经用完，之后收到的低优先级消息将被抛弃，不占用高优先级的配额。
2. 上行消息为默认高优先级消息，则先占用高优先级的 20 条消息配额，如配额已经用完，高优先级和低优先级共用的 20 条配额未占用完时，则占用高低优先级消息的共同配额，直到全部占用，之后收到的高、低优先级消息都将被抛弃。
3. 上行消息为设置的聊天室消息白名单中的消息或用户白名单中的用户发送的消息时，该类消息不会丢弃，但会占用每 200 毫秒 40 条的消息配额，优先占用高、低优先级消息共用的 20 条配额，其次占用高优先级消息的 20 条配额。配额被全部占用后，再收到高、低优先级的消息时都将被丢弃，但如收到白名单中的消息时则不会被丢弃，按时间顺序正常下发。

> 单个聊天室可消费的每 200 毫秒 40 条的上行消息配额，开通专有云后可进行配置

用户白名单、消息白名单、消息优先级三者均和消息有关，是否有先后顺序？

用户白名单、消息白名单、消息优先级提供了 3 种设置消息级别的不同维度，级别均为高级别，可根据业务需要按需使用

如何统计聊天室点赞消息数？

您可以通过开通服务端实时消息路由来实现此功能，用户在发送点赞消息时，这条消息同步给您的应用服务器。您的应用服务器通过消息类型来分析统计这些数据。

查看服务端实时消息路由功能介绍。

如何实现消息回看功能？

如果直播聊天室中需要实现消息回看功能，则需要开通聊天室消息云端存储功能。开通成功后，可通过接口获取聊天室中历史消息，消息回看 UI 界面的展示需要由您自己实现。

聊天室中存储的消息类型及自定义消息如何设置存储，请查看聊天室消息云端存储功能介绍。

多端情况下退出聊天室

在多端登录情况下，一端退出聊天室其他终端是否可同步退出聊天室？

在开通了多端同时登录情况下，用户登录多个终端，加入聊天室后，如在一端退出聊天室，其他端不会同步退出，仍然可以收到

聊天室中的消息，如果需要一端退出聊天室后，其他已登录的终端也一起退出聊天室，则需要在退出时发送一条命令消息通知其他端退出聊天室，此步骤需要开发者自行实现。

另外，需要注意如果用户登录终端 A 后，加入了聊天室并在聊天室的会话界面中，这时用户又登录了终端 B ，但未进入聊天室会话界面，这时 A 、B 两个终端都会收接聊天室的消息，只是 B 端的聊天室消息不进行展示。如用户在 A 端退出聊天室，这时融云认为用户在 B 端仍然在聊天室中，会继续向 B 端发送聊天室消息。如果需要 B 端同时也退出聊天室，则需要 A 端在退出时发送一条命令消息通知其他端退出聊天室，此步骤需要开发者自行实现。

系统通知介绍　　概述

更新时间:2024-08-30

一般系统消息特指会话类型（ConversationType）为“系统（SYSTEM）”的会话中的消息。

系统消息（System Message）是一种业务概念，是指利用系统帐号（非用户帐号，用户不可登录）向用户发送的消息，既可以是通过调用全量通知接口发送给所有人的消息，也可以是加好友提醒等单个用户通知消息。

> ⚠ 警告
>
> 系统通知消息，只能通过 Server API 进行发送，终端用户收到系统消息后，不支持消息回复功能。
>
> 具体消息发送方式详见：服务端集成-系统通知

主要功能

功能	描述
离线消息	支持离线消息存储，存储时间可设置（1 ～ 7 天），默认存储 7 天。
消息提醒	离线状态，系统会话有新消息时，支持 Push 通知。
本地存储	存储在移动端本地，提供本地消息搜索功能。
历史消息	提供服务端消息存储存功能，需开通单群聊消息云存储，默认存储时长为 6 个月。
单个用户通知	向单个或多个用户发送系统消息。
标签用户通知	向标签下的所有用户发送系统消息，用户标签属性需要开发者通过 Server API 接口进行设置。
全量用户落地通知	向应用下的所有用户发送广播消息，用户在线情况下会直接收到消息内容，未在线时此条消息会转为 Push 进行推送，同时该条消息做为离线消息进行存储，用户下次登录后会通过离线消息获取。
全量用户不落地通知	向应用下的所有用户发送远程 Push，无论用户是否正在使用应用，该通知只会展示在通知栏，通知内容不会做为消息存储到本地。

消息类型

消息类型	描述
文字消息	用来发送文字类消息，其中可以包括表情、超链接（会自动识别），客户端收到消息后计入未读消息数、进行存储。
语音消息	发送高质量的短语音消息，录制的语音文件存储到融云服务端，语音文件格式为 AAC，时长上限为 60 秒，客户端收到消息后计入未读消息数、进行存储。
图片消息	用来发送图片类消息，客户端收到消息后计入未读消息数、进行存储。图片缩略图格式为 JPG，大小建议不超过 100k。
GIF 图片消息	用来发送 GIF 动态图片消息，客户端收到消息后计入未读消息数、进行存储。
图文消息	用来发送图文消息，包含一个标题，一段文字内容和一张图片，客户端收到消息后计入未读消息数、进行存储。
文件消息	用来发送文件类消息，客户端收到消息后计入未读消息数、进行存储。
小视频消息	用来发送小视频消息，支持录制发送及选择本地视频文件发送两种方式，录制时长不超过 10 秒，本地选择视频文件方式时长不超过 2 分钟，小视频消息小视频文件格式为 .mp4，客户端收到消息后计入未读消息数、进行存储。
命令消息	用来发送通用的指令通知消息，消息内可以定义任意 JSON 内容，与通用命令通知消息的区别是不存储、不计数，此类型消息没有 Push 通知。
自定义消息	融云内置消息类型，无法满足客户业务需求时，可通过自定义消息类型进行实现，接收自定义消息的格式解析及展示处理需要开发者自行实现

消息结构示例

系统通知场景下融云内置消息类型说明：

文本消息

ObjectName存储属性计数属性离线属性推送属性推送内容
RC:TxtMsg 存储 计数 存储 推送 消息内容
消息结构：

发送文本消息时 content 参数的 JSON 结构如下：

```
{
"content":"Hello world!",
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":""
}
```

属性说明：

名称	类型	必传	说明
content	String	是	文字消息的文字内容，包括表情。
user	String	否	消息中携带的用户信息，详细查看 user 参数说明。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

user 属性说明：

名称	说明
id	发送用户 Id。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

图片消息

ObjectName存储属性计数属性离线属性推送属性推送内容
RC:ImgMsg 存储 计数 存储 推送 [图片]
消息结构：

发送图片消息时 content 参数的 JSON 结构如下：

```
{
"content":"bhZPzJXimRwrtvc=",
"imageUri":"http://p1.cdn.com/fds78ruhi.jpg",
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":""
}
```

属性说明：

名称	类型	必传	说明
content	String	是	图片缩略图，格式为 JPG，Base64 字符串长度建议为 5k，最大不超过 10k，注意在 Base64 进行 Encode 后需要将所有 \r\n 和 \r 和 \n 替换成空。
imageUri	String	是	图片上传到图片存储服务器后的地址。

名称	类型	必传	说明
user	String	否	消息中携带的用户信息，详细查看 user 参数说明。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

user 属性说明：

名称	说明
id	发送用户 Id。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

常见问题：

1、缩略图最大尺寸为：240 x 240 像素，以宽度和高度中较长的边不超过 240 像素等比压缩。

2、大图最大尺寸为：960 x 960 像素，以宽度和高度中较长的边不超过 960 像素等比压缩。

3、图片消息包括两个主要部分：缩略图和大图，如设置为原图发送则为缩略图和原图，缩略图直接 Base64 编码后放入 `content` 中，大图或原图首先上传到文件服务器（融云 SDK 中默认上传到七牛云存储），然后将云存储上的大图或原图地址放入消息体中。

4、发送图片消息时，需要自行上传图片文件到应用的文件服务器，生成地址后进行发送。

GIF 图片消息

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:GIFMsg	存储	计数	存储	推送	[图片]

消息结构：

发送图片消息时 content 参数的 JSON 结构如下：

```
{
"gifDataSize":34563,
"height":246,
"localPath":"/var/mobile/.../GIF_53",
"remoteUrl":"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.jpg",
"width":263,
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":""
}
```

属性说明：

名称	类型	必传	说明
gifDataSize	Int	是	GIF 图片文件大小，单位为字节 Byte。
localPath	String	是	下载 GIF 图片后存储在本地的图片地址。
remoteUrl	String	是	GIF 图片的服务器地址。
width	Int	是	GIF 图片宽度。
height	Int	是	GIF 图片高度。
user	String	是	消息中携带的用户信息，IMKit SDK 会话界面中优先显示消息中携带的用户信息，可去掉此属性。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

user 属性说明：

名称	说明
id	发送用户 Id。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

语音消息

> ⚠ 警告
>
> 从 SDK 2.9.19 版本开始支持 RC:HQVCMsg 语音消息功能，RC:HQVCMsg 语音消息与旧版本 SDK 不兼容，旧版本 SDK 无法收听新的语音消息。

新语音消息 RC:HQVCMsg 和旧版本语音消息 RC:VcMsg 不同的是将录制的音频数据存储到服务端，而消息体内只保存 URL。摆脱了消息体 128K 的大小限制，所以拥有更高音质。语音时长上限为 60 秒，客户端收到消息后计入未读消息数、进行存储。

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:HQVCMsg	存储	计数	存储	推送	[语音]

消息结构：

发送高质量语音消息时 content 参数的 JSON 结构如下：

```
{
"localPath":"/9j/4AAQSkZ/2wBaSiimB//9k=",
"remoteUrl":"http://p1.cdn.com/fds78ruhi.aac",
"duration":7,
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":""
}
```

参数说明：

参数	类型	必传	说明
localPath	String	否	采用 AAC 格式进行编码录制的媒体内容本地路径。
remoteUrl	String	是	媒体内容上传服务器后的网络地址。
duration	Int	是	语音消息的时长，最长为 60 秒（单位：秒）。
user	String	否	消息中携带的用户信息，详细查看 user 参数说明。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

user 参数说明：

名称	说明
id	发送用户 Id。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

常见问题

1、发送高质量语音消息时，需要自行生成 AAC 格式文件并上传文件到应用的文件服务器，生成地址后进行发送。

文件消息

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:FileMsg	存储	计数	存储	推送	[文件] + 文件名，如：[文件] 123.txt

消息结构：

发送文件消息时 content 参数的 JSON 结构如下：

```
{
"name":"file.txt",
"size":190184,
"type":"txt",
"fileUrl":"http://www.demo.com/am.ind",
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":""
}
```

属性说明：

名称	类型	必传	说明
name	String	是	文件名称。
size	String	是	文件大小，单位：Byte。
type	String	是	文件类型。
fileUrl	String	是	文件地址。
user	String	否	消息中携带的用户信息，详细查看 user 参数说明。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

user 属性说明：

名称	说明
id	发送用户 Id。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

常见问题

1、通过 Server API 发送文件消息时，需要自行上传文件到应用的文件服务器，生成文件地址后进行发送。

小视频消息

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:SightMsg	存储	计数	存储	推送	[小视频]

消息结构：

发送小视频消息时 content 参数的 JSON 结构如下：

```
{
"sightUrl":"http://rongcloud...com/video...",
"content":"/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
"duration":2,
"size":734320,
"name":"video_xx.mp4",
"user":
{
"id":"4242",
"name":"Robin",
"portrait":"http://example.com/p1.png",
"extra":"extra"
},
"extra":"extra"
}
```

属性说明：

名称	类型	必传	说明
sightUrl	String	是	上传到文件服务器的小视频地址。
content	String	是	小视频首帧的缩略图进行 Base64 编码的结果值，格式为 JPG，注意在 Base64 进行 Encode 后需要将所有 \r\n 和 \r 和 \n 替换成空。
duration	Int	是	视频时长，单位：秒。 • 如使用 IMKit 小视频插件进行录制，默认最长可录制 10 秒。 • 如选择本地视频文件，请注意服务端的默认视频时长上限为 2 分钟。如需调整上限，请联系商务。
size	String	是	视频大小单位 Byte。
name	String	是	发送端视频的文件名，小视频文件格式为 MP4（H.264+AAC）。
user	String	否	消息中携带的用户信息，详见 user 属性说明。
extra	String	否	扩展信息，可以放置任意的数据内容，也可以去掉此属性。

- user 属性说明：

名称	说明
id	发送用户 ID。
name	发送用户需要显示的名称。
portrait	发送用户需要显示的头象。
extra	扩展信息，可以放置任意的数据内容。

常见问题

1. 通过 Server API 发送视频消息时，需要自行上传视频文件到应用的文件服务器，生成文件地址后进行发送。
2. IMKit SDK 中目前支持播放的视频文件格式为 mp4，IMLib SDK 中播放功能需要开发者自行实现。

命令消息

ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RC:CmdMsg	不存储	不计数	存储	不推送	无

消息结构：

运营平台向终端发送指令信息时可使用此命令消息，消息中 content 参数的 JSON 结构如下：

```
{
"name":"AtPerson",
"data":"{\"sourceId\":\"9527\"}"
}
```

属性说明:

名称 类型 必传说明
nameString**是** 命令名称，可以自行定义。
data String**是** 命令的内容。

获取全部会话　　获取本地会话列表

更新时间:2024-08-30

Web 端不具备持久化的数据存储能力，故无法提供获取本地会话列表相关功能

获取远端会话列表

> Web 端不具备持久化的数据存储能力，无法在本地持久化存储历史消息记录与会话列表，因此需要从融云服务端获取数据。从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 IM 旗舰版或 IM 尊享版可开通该服务。具体功能与费用以融云官方价格说明页面及计费说明文档为准。
> 该功能需要在调用 `im.connect()` 并且建立连接成功之后执行。
> 服务器会话列表存储上限为 1000 条会话

参数说明

输入参数说明

参数	类型	必填	默认值	说明	最低版本
count	Number	否	300	想要获取的会话列表数量，默认值 `300`，最大值 `1000`	3.0.0
startTime	Number	否	0	获取会话列表所使用的时间戳，需要精确到毫秒	3.0.7.1
order	Number	否	0	会话排序方式	3.0.7.1

`order` 说明：

value	说明
0	按会话生成时间倒序排列，即获取早于 startTime 生成的会话列表
1	按会话生成时间正序排列，即获取晚于 startTime 生成的会话列表

`startTime` 补充说明：

- startTime 为 0 且 order 为 0 时, startTime 表示当前时间点，接口将返回最新的会话列表
- startTime 为 0 且 order 为 1 时, startTime 表示会话列表的创建时间，即最早的一条消息的产生时间，接口将返回最早的会话列表数据

回调参数说明

回调参数	类型	说明
conversationListArray		会话列表，会话参数说明请参照 `conversation` 属性说明

- `conversation` 属性说明：

字段名	类型	说明
type	Number	会话类型，见 `type` 枚举值说明
targetId	String	系统会话 ID

字段名	类型	说明
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息，详见 latestMessage
hasMentioned	Boolean	是否包含 @ 自己的消息
mentionedInfo	Object	@ 信息，见 `mentionedInfo` 结构说明
notificationStatus	Number	当前会话免打扰状态
isTop	Boolean	当前会话免置顶状态

- `type` 枚举值说明

会话类型	说明	枚举值
RongIMLib.CONVERSATION_TYPE.PRIVATE	单聊	1
RongIMLib.CONVERSATION_TYPE.GROUP	群聊	3
RongIMLib.CONVERSATION_TYPE.CHATROOM	聊天室	4
RongIMLib.CONVERSATION_TYPE.SYSTEM	系统	6

- `mentionedInfo` 结构说明：

字段名	类型	说明
type	Number	@ 类型，1: @ 所有人， 2: @ 部分人
userIdListArray		被 @ 的用户 id 列表

代码示例

```
im.Conversation.getList({
count: 30,
startTime: 0,
order: 0
}).then(conversationList => {
console.log('获取会话列表成功', conversationList);
});
```

删除全部会话　　清除会话列表

更新时间:2024-08-30

SDK 不提供批量删除接口，如需事先清除多个会话，可循环调用 删除指定会话 接口来实现。

获取指定会话

更新时间:2024-08-30

可以通过获取会话列表得到会话列表信息，通过 Conversation.get 获取会话实例进行消息发送等操作。

参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	系统会话 ID	3.0.0
type	Number	是	会话类型，系统通知 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	3.0.0

代码示例

```
// 注: im 实例通过 RongIMLib.init 获取(单个页面仅需初始化一次)
const conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
```

conversation 属性说明：

字段名	类型	说明
type	Number	会话类型
targetId	String	系统会话 ID

删除指定会话　　删除指定会话

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 `conversation.destory()` 删除会话。

> 当收到新消息或者发送消息时会重新生成该会话。

代码示例

```
// conversation 会话实例
const conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
conversation.destory().then(() => console.log('删除会话成功'));
```

会话草稿信息　　获取草稿

更新时间:2024-08-30

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	系统会话 ID	4.0.0
type	Number	是	会话类型，系统通知 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
conversation.getDraft().then(function (draft) {
console.log('获取指定会话草稿成功', draft)
})
```

保存草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	系统会话 ID	4.0.0
type	Number	是	会话类型，系统通知 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	4.0.0

参数说明

参数	类型	必填	说明	最低版本
draft	String	是	草稿内容	4.0.0

代码示例

```
var draft = '这是会话草稿内容';
var conversation = im.Conversation.get({
targetId: '系统会话 ID',
type:RongIMLib.CONVERSATION_TYPE.SYSTEM
});
conversation.setDraft(draft).then(function () {
console.log('设置指定会话草稿成功')
})
```

删除草稿

获取会话实例参数说明

参数	类型	必填	说明	最低版本
targetId	String	是	系统会话 ID	4.0.0
type	Number	是	会话类型，系统通知 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
conversation.deleteDraft().then(function () {
console.log('删除指定会话草稿成功')
})
```

会话未读数　　获取所有会话未读数

更新时间:2024-08-30

会话未读数指某一个会话中未读消息的数量

1. 清除浏览器缓存会导致会话未读数不准确
2. 会话消息未读数存储在 WebStorage 中, 若浏览器不支持或禁用 WebStorage，未读消息数将不会保存，浏览器页面刷新未读消息数将不会存在

参数说明

参数	类型	必填	说明	最低版本
includeMuted	Boolean	否	是否包含免打扰会话，默认为: false	4.4.5
conversationTypes	Array	否	会话类型	4.4.5

代码示例

```
const conversationTypes = [RongIMLib.CONVERSATION_TYPE.SYSTEM]
im.Conversation.getTotalUnreadCount(false, conversationTypes).then(function(
totalUnreadCount
) {
console.log('获取未读总数成功', totalUnreadCount)
})
```

清除全部会话未读数

最低版本：4.5.4

```
im.Conversation.readAll()
```

清除单个会话未读数

代码示例

```
var conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
conversation.read().then(function(){
console.log('清除未读数成功');
});
```

获取指定会话未读数

代码示例

```
let conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});

conversation.getUnreadCount().then(function(count) {
console.log('获取指定会话未读数成功', count);
})
```

多端同步未读数

未读消息存在 localStorage 中，未读消息数是针对当前端的未读消息数，服务器不存未读消息数量。

实现方案

1. 调用 conversation.read() 清除未读数。
2. 清除成功后发送 `RC:SRSMsg` 类型消息进行未读数同步。
3. 其他端接受到 `RC:SRSMsg` 类型消息，调用 `conversation.read()` 方法进行本地未读数清除。（在 v4.5.4 版本之后收到该消息时 sdk 内部会清理未读数，无需用户主动调用）

会话免打扰　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，系统 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	3.0.0
targetId	String	是	系统会话 ID	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM
});
let statusItem = {
notificationStatus: 1,
};
conversation.setStatus(statusItem).then(function() {
console.log('设置免打扰成功');
})
```

会话置顶　　会话置顶设置

更新时间:2024-08-30

1. 删除指定会话需要通过 `targetId` 和 `type` 获取到该会话的实例。
2. 调用 conversation.setStatus() 进行设置

获取会话实例参数说明

参数	类型	必填	说明	最低版本
type	Number	是	会话类型，系统 传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	3.0.0
targetId	String	是	系统会话 ID	3.0.0

设置参数statusItem说明

参数	类型	必填	说明	最低版本
notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	3.0.4
isTop	Boolean	否	是否置顶	3.0.4

代码示例

```
let conversation = im.Conversation.get({
targetId: '系统会话 ID',
type: RongIMLib.CONVERSATION_TYPE.SYSTEM,
})
let statusItem = {
isTop: true,
}
conversation.setStatus(statusItem).then(function() {
console.log('设置置顶成功')
})
```

消息接收　　功能描述

更新时间:2024-08-30

开发者可通过此接口拦截到 SDK 接收到的消息，并进行响应的业务操作。

实现方法

消息通过设置监听总的消息监听进行接收，消息监听中接收的消息不区分消息类型。收到后按需处理即可。详见消息监听。

历史消息获取　　本地获取

更新时间:2024-08-30

Web 没有本地存储，不提供本地获取方法。

远端获取

从远端获取单群聊历史消息是指从融云服务端获取历史消息，该功能要求 App Key 已启用融云提供的单群聊消息云端存储服务。您可以在控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。如果使用生产环境的 App Key，请注意仅 IM 旗舰版或 IM 尊享版可开通该服务。具体功能与费用以融云官方价格说明 页面及计费说明 文档为准。

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统通知传入 RongIMLib.CONVERSATION_TYPE.SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
timestamp	Number	是	获取时间戳, 0 为从当前时间拉取	2.2.0
count	Number	是	获取条数, 范围 1 - 20	2.2.0
order	Number	否	获取顺序，默认为 0， 0 表示升序：获取消息发送时间比传入 sentTime 小 的消息 1 表示倒序：获取消息发送时间比传入 sentTime 大 的消息	2.5.3

回调参数说明

参数	类型	说明
list	Array	获取的历史消息列表，返回 message 列表
hasMsgBool		是否还有历史消息可以获取

`message` 属性说明

字段名	类型	说明
conversationType	Number	会话类型
targetId	String	系统会话 ID
senderUserId	String	发送者 ID
content	Object	消息内容
objectName	String	消息的消息标识，融云内置消息以 "RC:" 开头
messageType	String	消息类型
messageId	String	本地生成的消息 ID
messageUId	String	服务端存储的消息 ID
messageDirection	Number	消息方向，发送: 1，接收: 2，枚举值通过 `RongIMLib.MessageDirection` 获取
offLineMessage	Boolean	是否为离线消息
sentStatus	Number	发送状态, 枚举值通过 `RongIMLib.SentStatus` 获取
sentTime	Number	消息在融云服务端的发送时间
receivedStatus	Number	接收状态, 枚举值通过 `RongIMLib.ReceivedStatus` 获取
receivedTime	Number	接收时间

代码示例

```
var conversationType = RongIMLib.CONVERSATION_TYPE.SYSTEM;
var targetId = '系统会话 ID';
var timestamp = 0;
var count = 20;
RongIMLib.RongIMClient.getInstance().getHistoryMessages(conversationType, targetId, timestamp, count, {
onSuccess: function(list, hasMsg) {
/*
list: 获取的历史消息列表
hasMsg: 是否还有历史消息可以获取
*/
console.log('获取历史消息成功', list);

},
onError: function(error) {
// 请排查：单群聊消息云存储是否开通
console.log('获取历史消息失败', error);
}
});
```

更新日志　　关于停止维护 IMLib v4 旧版 SDK 的声明

更新时间:2024-08-30

注意：

- **Web IMLib v4 版本目前已停止维护，建议您优先选择最新的 IMLib 版本。**
- **已集成 IMLib v4 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 4x SDK 的客户可以通过 `RongIMLib-v4-Adapter` 无缝替换升级。详见 升级说明。**
- **未来我们将在 `RongIMLib-v4-Adapter` 上进行问题修复，但不会增加新功能。**

Adapter 版本

v5.10.1

发布日期：2024/06/28

问题修复：

1. 修复非群聊会话中会可能会携带 mentionedInfo 的问题

v5.9.9

发布日期：2024/06/05

问题修复：

1. 修复了实时日志请求 URL 有特殊字符导致请求失败的问题。
2. 修复了日志数据库升级可能会报错的问题。

v5.9.8

发布日期：2024/04/29

问题修复：

1. 修复了 Web 端拉取消息后处理异常时导致不再拉取消息的问题。

v5.9.7

发布日期：2024/04/29

问题修复：

1. 修复了 Web 端拉取消息后处理异常时导致不再拉取消息的问题。

v5.9.6

发布日期：2024/03/29

问题修复：

1. 修复了重连报 30021 时没有重连的问题。
2. 修复了主动撤回消息后，在消息监听中收到重复的撤回消息通知的问题。

v5.9.0

发布日期：2023/11/23

问题修复：

1. 小程序平台不再请求动态导航地址
2. 修复可能收不到敏感词拦截通知的问题
3. 修复发送@消息时，会话中的@字段错误的问题

v5.8.0

发布日期：2023/07/3

问题修复：

1. 修复断网重连偶现导致触发心跳问题。

v5.7.8

发布日期：2023/05/11

问题修复：

1. uniapp 打包 app 链接不上
2. IE 浏览器不再支持日志存储, 因为 indexDB 不支持 getAllKeys 方法

v5.7.7

发布日期：2023/04/21

问题修复：

1. 修复获取免打扰列表 notificationLevel 值 undefined。
2. 修复 Electron 平台获取全部会话列表无法获取系统会话的问题。

v5.7.5

发布日期：2023/04/12

问题修复：

1. 优化 5.4.7 之前版本禁用资源 pb 报错。

v5.7.4

发布日期：2023/03/30

问题修复：

1. 修复无法获取到未设置免打扰状态会话的未读数的问题。
2. 修复在 web 平台，会收到自己设置的聊天室 kv 的通知的问题。
3. 修复偶现 `Cannot read property 'kvStorage' of null` 的问题。
4. 修复断网重连后再发消息时，偶发消息监听中收到自己发送的消息的问题。

v5.7.3

发布日期：2023/03/02

问题修复：

1. 修复发送 @ 消息后发送方自己收到 @ 消息的会话变更问题
2. 修复切换用户后会话状态还使用的前一个用户的数据问题
3. 修复推送配置中单独设置 iOSConfig 或者 androidConfig 不生效的问题

v5.7.2

发布日期：2023/02/07

问题修复：

1. 修复 getCurrentConnectionStatus 接口返回状态类型错误问题

优化：

1. Web 端本地会话状态缓存上限优化，最大支持存储 1000 条会话状态

v5.7.1

发布日期：2023/01/10

问题修复：

1. 修复在火狐浏览器中的 indexDB 兼容问题
2. 修复断网重连时调用 disconnect 无法断开连接的问题

3. 修复调用 `removeChatRoomEntry` 后，其他人收到的 KV 数据更新类型（ChatroomEntryType）为 `UPDATE` 的问题。修复后，KV 更新类型为 `DELETE`。
4. 修复 Electron 平台插入消息时设置的消息扩展字段 `canIncludeExpansion`，`expansion` 与返回数据中不一致的问题
5. 修复 Electron 平台发起 http 请求报错的问题

v5.7.0

发布日期：2022/12/01

优化：

1. 断网重连时，如果被聊天室封禁，则不再尝试加入该聊天室
2. 断网重连情况下，SDK 内部重新加入聊天室时拉取的历史消息数量为加入时传入的值，默认为 10

非兼容性变更：

1. 连接状态监听函数变更，废弃 `status`，请使用 `connection`

v5.6.0

发布日期：2022/11/04

问题修复：

1. 修复断网重连时如果 token 过期，应用层收不到状态通知的问题
2. 修复多端登录情况下，本端未加入聊天室时，会收到其他端加入聊天室后发送的消息问题
3. 修复 Web 平台收到位置共享功能的 `RC:RLQuit`、`RC:RLJoin` 消息时，在控制台报错的问题

非兼容性变更：

1. Web 端不再支持 Comet 连接模式，仅支持 Websocket 连接

v5.4.5

发布日期：2022/08/18

问题修复：

1. 修复用户多端登录情况下设置会话状态会导致 Web 端收到重复通知的问题
2. 修复小程序平台 HTTP 请求的 header 字段错误的问题
3. 修复加入多个聊天室时，后加入的聊天室 KV 拉取异常的问题

v5.3.4

发布日期：2022/06/20

问题修复：

1. 修复频繁设置会话置顶或会话免打扰状态导致 26002 错误的问题。

v5.3.3

发布日期：2022/06/02

问题修复：

1. 修复可能会丢失会话类型为 ConversationType.RTC_ROOM 的直发消息的问题。
2. 修复获取会话列表为空时，返回报错的问题。
3. 修复升级 5.0 后会话未读数无法清除的问题。

优化：

1. 优化撤回消息计数。

v5.3.2

发布日期：2022/05/20

优化：

1. 优化聊天室获取消息及扩展属性信息机制。

v5.3.1

发布日期：2022/05/19

问题修复：

1. 优化重连逻辑，修复网络异常时可能无法重连的问题
2. 修复 comet 连接时拉取消息报错的问题
3. 修复 App Key 未开启超级群服务时，SDK 断网重连后会异常拉取超级群消息的问题

其他：

1. 优化连接逻辑

v5.3.0

发布日期：2022/04/29

问题修复：

1. 修复收到广播消息后，断开连接再重复连接，会再次收到广播消息的问题
2. 修复在 IE 11 浏览器中调用 disconnect 方法报错的问题

其他：

1. 发送撤回消息的消息体中可携带 extra 字段

RongIMLib 版本

4.6.3

发布日期：2022/04/15

问题修复：

1. 修复极少数情况下会丢失会话类型为 `ConversationType.RTC_ROOM` 的消息的问题
2. 修复 https 协议时无法上报日志的问题

4.6.2

发布日期：2022/04/07

问题修复：

1. 修复接受广播消息可能重复的问题。
2. 修复获取会话列表为空时，返回报错的问题。

4.6.1

发布日期：2022/03/17

问题修复：

1. 修复在小程序中重连失败时无法继续重连的问题

4.6.0

发布日期：2022/02/17

问题修复：

1. 修复在单聊中发送 @ 消息时，接收方收到该消息时可能会报错的问题
2. 修复环境中 console 无法使用时导致 SDK 无法使用的问题

4.5.4

发布日期：2021/12/28

问题修复：

1. 修复会话列表中 latestMessage 为 null 时（一般在会话中最新消息在 Web 客户端本地被删除时出现）报错的问题
2. 修复支付宝小程序中请求导航返回值解析报错的问题

功能优化：

1. 本端主动清理未读数时不再通知会话变更

新增功能：

1. 新增清除全部未读数接口

4.5.3

发布日期：2021/12/09

问题修复：

1. 修复断线重连时可能收消息延迟的问题
2. 修复连接之前多个 ping 等待造成连接延迟的问题
3. 修复切换用户后，后登录用户使用前一用户的内存数据拉取消息的问题
4. 修复消息体内 user.portraitUri 字段多端不一致问题，推荐使用 portrait 字段

4.5.2

发布日期：2021/11/25

功能优化：

1. 连接时优先使用缓存导航。

4.5.1

发布日期：2021/11/04

新增功能：

1. 撤回消息时，会触发会话列表变更事件，开发者可在收到事件通知后刷新 UI。

问题修复：

1. 修复在极少数情况下，断线重连后可能无法拉取离线消息的问题。

4.5.0

发布日期：2021/10/22

新增功能：

1. 新增发送敏感词时通知功能
2. 新增批量设置和删除聊天室属性能力
3. 新增用户未加入聊天室时，支持获取聊天室属性信息
4. 新增用户加入、退出聊天室通知能力，需要客户开通后支持，可提交工单申请开通
5. 新增聊天室销毁通知能力

5. 新增会话标签功能

问题修复：

1. 修复加入聊天室错误时没有状态码的问题

4.4.10

发布日期：2021/10/14

问题修复：

1. 修复多端同步状态消息时可能报错的问题

4.4.9

发布日期：2021/09/24

功能优化：

1. 优化消息拉取功能

4.4.8

发布日期：2021/09/10

问题修复：

1. 修复推送可能丢失 pushData 字段的问题

功能优化：

1. 解除端上不允许发系统消息的限制

新增功能：

1. 增加 Conversation.getInfo 接口

4.4.7

发布日期：2021/08/26

问题修复：

1. 修复收到非 UpStreamMessage 的消息信令时可能报错的问题

2. 修复在服务端发送消息时，本人可能会收到重复消息的问题

功能优化：

1. 优化连接时返回具体错误码

2. 增加在小程序环境请求导航

4.4.6

发布日期：2021/08/13

问题修复：

1. 修复撤回消息时多端同步到的会话信息可能错误的问题

2. 修复收到直发消息时清理未读数可能不能清空的问题

4.4.5

发布日期：2021/07/30

问题修复：

1. 修复获取未读总数时未过滤免打扰会话的问题

2. 修复单聊会话中对方可能收到 RC:SRSMsg 类型消息的问题

3. 修复频繁删除KV时切换聊天室可能会导致不再通知KV变化的问题

新增功能：

1. 增加离线消息拉取完成通知

4.4.4

发布日期：2021/07/15

问题修复：

1. 修复一些情况下可能报 IDBKeyRange 错误问题

功能优化：

1. 优化重连逻辑

4.4.3

发布日期：2021/07/02

问题修复：

1. 修复在小程序中发送消息可能报错问题

4.4.2

发布日期：2021/07/01

问题修复：

1. 修复日志报错问题
2. 修复发送消息后刷新页面重新连接，可能会重复收到本端发送消息的问题
3. 修复聊天室拉取消息时可能拉取到加入前消息的问题
4. 修复 comet 连接重连后不发 pullMsg 问题
5. 调用 disconnect 时清除重连定时器

4.4.1

发布日期：2021/06/11

功能优化：

1. 优化 navi 存储策略

问题修复：

1. 修复可能会发送多个 ping 的问题

4.4.0

发布日期：2021/06/03

新增功能：

1. 增加 typing 消息, 新增 typing 状态通知
2. 推送扩展新增字段

功能优化：

1. 适配头条和百度小程序
2. 小程序支持 comet 连接
3. 发送消息失败时返回消息内容
4. 重定向失败时继续重连
5. 优化日志输出

问题修复：

1. 修复监听连接状态改变为 success 时调用接口报 30001 的问题

其他

1. npm 包不再支持 IE，如需支持请用 CDN 包

4.3.6

发布日期：2021/05/27

问题修复：

1. 修复重新连接后未拉取离线消息问题

4.3.5

发布日期：2021/05/20

功能优化：

1. 心跳间隔和超时改为15s
2. 自定义 Navi 地址过滤 '/'
3. 禁止插件重复初始化

4.3.4

发布日期：2021/05/07

问题修复：

1. 修复连接失败时不能再次调用连接的问题
2. 修复在 uni-app 中，在 Android 和 iOS 平台编译不过的问题
3. 修复在微信小程序中，在使用 2.16.0 及以下版本的基础调试库时报错的问题

4.3.3

发布日期：2021/04/29

问题修复：

1. 增加内置消息类型 'RC:GIFMsg'
2. 修复小程序无法使用 comet 连接问题
3. 修复小程序获取导航为 null 问题

4.3.2

发布日期：2021/04/23

问题修复：

1. 修复通知拉取消息时消息状态错误问题
2. 修复消息的 isStatusMessage 参数判断错误问题
3. 修复断开网络30s内重连成功时，ping 会产生多个 timer 的问题
4. 修复多端或换端登录情况下,拉取离线补偿过程中发送消息可能导致拉取时间戳错误,导致丢失部分发件箱消息
5. 修复发送普通群组消息后，会把会话@信息清空的问题，增加清空未读数时清空@信息

4.3.1

发布日期：2021/04/15

问题修复：

修复连接 ping 逻辑错误导致 ping 超时也不会主动中断连接

功能优化

更新日志输出格式

4.3.0

发布日期：2021/04/12

新增功能：

1. 针对移动端支持了发送单条消息配置推送内容功能
2. 优化了对服务器侧主动断开连接后的重连处理逻辑
3. 支持新版本 Electron 桌面端解决方案 - @rongcloud/electron-solution

4.2.6

发布日期：2021/03/17

问题修复：

1. 对 TypeScript 开发者增加 IAReceivedConversation 接口定义暴露

4.2.5

发布日期：2021/03/08

问题修复：

1. 修复了加入房间时不获取历史消息，断线重连后出现聊天室消息断档问题

2. 优化了获取单一 Conversation 会话实例时频繁拉取会话列表问题

3. 修复了构建脚本错误导致 IMLib 文件重复打包 engine 依赖造成代码冗余问题

4. 修复了获取历史消息时，若 timestamp 为 0、count 为 1 时无法获取数据问题

5. 补全了 TS 开发者依赖的接口及类定义声明

4.2.4

发布日期：2021/02/24

问题修复：

1. 修复了聊天室拉取消息重复问题

4.2.3

发布日期：2021/02/07

问题修复：

1. 修复了聊天室拉取消息偶现的重复消息问题

2. 修复了接收状态消息时更新了本地收件箱时间，导致拉取消息时可能存在消息丢失的问题

4.2.2

发布日期：2021/01/28

问题修复：

1. 修复了多端情况下一端在聊天室中，另一端未加入聊天室获取聊天室消息出错的问题。

4.2.1

发布日期：2021/01/25

问题修复：

1. 修复了 SDK 中部分 BUG。

4.2.0

发布日期：2021/01/20

问题修复：

1. 修复了聊天室属性监听无法触发问题

2. 修复了退出聊天室异常问题

3. 修复了接收消息体里静默消息字段展示错误问题

功能优化

1. 浏览器最低兼容到 IE 9

2. 优化了导航连接逻辑

4.1.1

发布日期：2020/12/29

问题修复：

1. 修复了群已读消息状态多端同步时，其他群成员也会收到状态同步的问题。

4.1.0

发布日期：2020/12/11

新增功能：

1. 进一步优化了上传文件服务，当发送富媒体消息时如图片、文件、小视频等，上传文件失败情况下会自动切换到备份服务器进行存储，用户无感知。

问题修复：

1. 修复发送的状态消息被存储、计数问题

4.0.1

发布日期：2020/11/27

问题修复：

1. 修复了 SDK 链接导航时内部上报版本号错误的问题

2. 补齐了小视频内置消息类型

4.0.0

发布日期：2020/11/20

新增功能：

1. 新增了会话草稿功能

2. 增加了对 TypeScript 的类型支持，提供 .d.ts 声明文件

功能优化：

1. 支持了 NPM 模块集成，NPM 仓库包名为 `@rongcloud/imlib-v4`

2. 提升了 SDK 代码健壮性、稳定性

3. 提升了连接速度及连接成功率，减少不必要的网络嗅探

4. 提升了 SDK 性能，减少不必要的异步任务

3.0.7.4 Dev

发布日期：2020/12/07

问题修复：

1. 修复了加入多个聊天室后，无法退出的问题

3.0.7.3 Dev

发布日期：2020/12/07

问题修复：

1. 增加了 25102 错误码提示，表示单群聊消息云存储服务未开通，无法获取服务端会话列表

3.0.7.2 Dev

发布日期：2020/11/24

问题修复：

1. 修复支付宝小程序 localStorage 取值错误问题

3.0.7.1 Dev

发布日期：2020/10/14

新增功能：

1. 支持了分页获取会话列表功能

3.0.7 Dev

发布日期：2020/09/18

新增功能：

1. 针对单条消息增加了消息扩展属性设置功能，消息发送前需要设置为可扩展后，才能对该条消息进行扩展信息添加。查看文档

2. 长轮询方式支持了会话置顶、免打扰设置功能

3. 聊天室模块添加 reall 撤回消息方法。查看文档

问题修复：

1. 更新了会话列表 mentionedInfo 字段存储信息，存储会话中最新 @当前用户的用户 ID

2. 修复了长轮询方式重复连接的问题

3. 合并转发模板 修复 Web 无展示嵌套合并转发消息问题

3.0.6 Dev

发布日期：2020/08/19

新增功能：

1. 连接成功后，接收消息中添加该消息其他端是否已经接收过的标识字段 receivedStatus

2. 增加了聊天室 KV 属性变化监听能力，聊天室中属性每次变化时都会同步监听状态

问题修复：

1. 修复了 uni-app 在微信小程序下环境判断错误的问题

2. 修复了 SDK 对部分 Emoji 特殊字符解析不正确的问题

3. 修复了多端情况下被踢后未断开 WebSocket 连接，再次连接后重复收消息问题

4. 修复了获取到的会话列表的会话结构中 hasMetioned 未生效的问题

功能优化：

1. 增加了 Navi 导航请求超时时间 10 秒

2. 增加了 CMP 超时时间 10 秒

3.0.5 Dev

发布日期：2020/07/21

新增功能：

1. 增加了新的加入聊天室接口，如聊天室不存在则加入不成功

2. 增加了静默消息功能，发送单条消息时支持设置该条消息没有通知

3. 完成了对 uni-app 框架的适配，可通过 uni-app 框架实现多平台研发

问题修复：

1. 修复了多端同步的单群聊状态消息，SDK 未向用户抛出的问题

2. 修复了读取某会话消息时，再次接收这个会话新消息，查看此会话未读数时错误的问题

3. SDK 状态监听器抛出时序问题处理，防止 SDK 内部逻辑未处理完成，抛出连接成功后用户调用任意接口报错

功能优化：

1. 优化了 SDK 断网重连后，自动重新加入聊天室的逻辑

3.0.4 Dev

发布日期：2020/06/19

新增功能：

1. 增加了会话免打扰及会话置顶多端状态同步功能

2. 增加了获取指定会话未读数方法 getUnreadCount

功能优化：

1. 优化了上传文件服务，当发送富媒体消息时如图片、文件、小视频等，上传文件失败情况下会自动切换到备份服务器进行存储，用户无感知。

2. 自动重连逻辑优化，防止 im.watch 重复监听 conversation、message、status 事件

3.0.3 Dev

发布日期：2020/06/02

功能优化：

1. 优化了 SDK 在 FILE 协议下的连接

3.0.2 Dev

发布日期：2020/05/22

新增功能：

1. IMLib SDK 增加了聊天室属性自定义存储功能，查看功能文档

3.0.1 Dev

发布日期：2020/05/07

新增功能：

1. 实现了对音视频 SDK 3.2.2 及以上版本的兼容
2. 增加了获取上传文件 Token 接口 getFileToken 及获取上传文件地址 Url 接口 getFileUrl
3. 增加了获取用户 ID 接口 getConnectionUserId
4. 增加了获取连接状态接口 getConnectionStatus
5. 适配了微信小程序环境

问题修复：

1. 修复了聊天室中消息方向显示不正确的问题
2. 修复了收到撤回消息时，会话列表变化监听未执行的问题
3. 开发者代码报错影响 SDK 消息抛出

3.0.0 Dev

发布日期：2020/03/31

新版本发布：

1. 体积相比 SDK 2.0 减少 50%
2. 兼容性健壮，兼容 Chrome、Firefox、Safari、IE5-11、Edge、微信浏览器、Android 2.3.6+ 等浏览器
3. 以会话为模型的全新 API 设计，集成更便捷
4. 内部代码高内聚低耦合，模块依赖清晰

升级说明

更新时间:2024-08-30

- 本文面向 **IMLib v3/v4 开发者**，旨在为 IMLib 开发者升级至 v4-adapter 提供引导。
- 对于使用 **IMLib v2 的用户来说，不建议升级至 v4-adapter 版本**。

关于停止维护 IMLib v4 旧版 SDK 的声明

注意：

- **Web IMLib v4 版本目前已停止维护，建议您优先选择最新的 IMLib 版本。**
- 已集成 IMLib v4 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 4x SDK 的客户可以通过 `RongIMLib-v4-Adapter` 无缝替换升级。详见 **升级说明**。
- 未来我们将在 `RongIMLib-v4-Adapter` 上进行问题修复，但不会增加新功能。

IMLib 4.x 替换为 v4-adapter

浏览器兼容性

Chrome	Firefox	Safari	Edge	QQ 浏览器	微信 浏览器	Android	IE
✔	✔	✔	✔	✔	✔	4.4+	9+

引入 SDK

修改 NPM 包

原 `@rongcloud/imlib-v4` 包已停止维护，请使用 `@rongcloud/imlib-v4-adapter` 替代。

1. 依赖安装

```
# 移除旧版本依赖
npm rm @rongcloud/imlib-v4 @rongcloud/engine
# 安装 adapter-v4
npm install @rongcloud/engine@latest @rongcloud/imlib-v4-adapter@latest -S
```

2. 代码集成

```
// CMD
// const RongIMLib = require('@rongcloud/imlib-v4') 需修改为
const RongIMLib = require('@rongcloud/imlib-v4-adapter')

// ES
// import * as RongIMLib from '@rongcloud/imlib-v4' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
```

修改 CDN 引入链接

原 RongIMLib-4.x.x.prod.js SDK 已停止维护，请使用 RongIMLib-v4-Adapter 的最新版本替代。

> RongIMLib-v4-Adapter 的最新版本可参见引入 SDK。

IMLib 3.x 升级 v4-adapter

引入 SDK

修改 NPM 包

1. 依赖安装

```
npm install @rongcloud/engine@latest @rongcloud/imlib-v4-adapter@latest -S
```

2. 代码集成

```
// CMD
const RongIMLib = require('@rongcloud/imlib-v4-adapter')

// ES
import * as RongIMLib from '@rongcloud/imlib-v4-adapter'
```

修改 CDN 引入链接

原 RongIMLib-3.x.x.prod.js SDK 已停止维护，请使用 RongIMLib-v4-Adapter 的最新版本替代。

> RongIMLib-v4-Adapter 的最新版本可参见引入 SDK。

接口变更

发送自定义消息

变更流程说明

发送自定义消息由直接发送变更为：先注册自定义消息再发送自定义消息，详细如下

1. 注册自定义消息

> ⚠ 警告
>
> - 注册自定义消息代码必须在发送、接收该自定义消息之前
> - 推荐将所有注册自定义消息代码放在 `init` 方法之后，`connect` 方法之前
> - 注册的消息类型名，必须多端一致，否则消息无法互通
> - 请勿使用 `RC:` 开头的类型名，以免和 SDK 默认的消息名称冲突

代码示例

```
// 初始化 IM
var config = {
appkey: ' ',
};
var im = RongIMLib.init(config);

// 注册自定义消息
var messageType = 's:person'; // 自定义消息类型
var isPersited = true; // 自定义消息存储属性
var isCounted = true; // 自定义消息计数属性
im.registerMessageType(messageType, isPersited, isCounted);
```

2. 发送自定义消息

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE
});
conversation.send({
messageType: 's:person', // 填写开发者定义的 messageType
content: { // 填写开发者定义的消息内容
name: 'RongCloud',
age: 12
}
}).then(function(message){
console.log('发送 s:person 消息成功', message);
});
```

自定义消息存储、计数属性说明

发消息：

1. 是否是内置消息，内置消息以内置消息的存储、计数配置为准
2. 是否是已注册的消息，已注册的消息以注册时的存储、计数配置为准
3. 其他消息，以消息发送接口传参为准，未传 isCounted、isPersited 时均默认为 false

收消息：

1. 是否是内置消息，内置消息以内置消息的存储、计数配置为准
2. 是否是已注册消息，已注册的消息以注册时的存储、计数配置为准
3. 其他消息，以接收的服务器数据为准

发送 @ 消息

变更输入参数说明

废弃参数	当前参数	类型	必填	默认值	说明	最低版本
~~isMentioned~~	**isMentioned**	Boolean	否	false	是否为 @ 消息	4.0.0
~~mentionedType~~	**mentionedType**	Number	否	1	@ 类型 1: @ 所有人 2: @ 指定用户	4.0.0
~~mentionedUserIdList~~	**mentionedUserIdList**	Array	否	--	@ 用户 id 列表	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '群组 ID',
type: RongIMLib.CONVERSATION_TYPE.GROUP
});
conversation.send({
messageType: RongIMLib.MESSAGE_TYPE.TEXT, // 填写开发者定义的 messageType
content: {
content: 'Hello RongCloud' // 文本内容
},
isMentioned: true,
mentionedType: RongIMLib.MENTIONED_TYPE.ALL,
mentionedUserIdList:['user1']
}).then(function(message){
console.log('发送消息成功', message);
});
```

获取历史消息

1. 单聊、群聊获取历史消息

变更输入参数说明

废弃参数	当前参数	类型	必填	默认值	说明	最低版本
~~timestamp~~	timestamp	Number	否	0	获取时间戳. 0 为从当前最新时间拉取	4.0.0

代码示例

```
var conversation = im.Conversation.get({
targetId: '接收方的 userId',
type: RongIMLib.CONVERSATION_TYPE.PRIVATE // 群聊可传入 RongIMLib.CONVERSATION_TYPE.GROUP
});
var option = {
timestamp: +new Date(), // 获取时间戳
count: 20 // 获取条数
};
conversation.getMessages(option).then(function(result){
var list = result.list; // 历史消息列表
var hasMore = result.hasMore; // 是否还有历史消息可以获取
console.log('获取历史消息成功', list, hasMore);
});
```

变更回调参数 list.message 属性说明

废弃字段	当前字段名	类型	说明
~~isMentioned~~	isMentioned	Boolean	是否为 @ 消息

2. 聊天室获取历史消息

变更输入参数说明

废弃参数	当前参数	类型	必填	默认值	说明	最低版本
~~timestamp~~	timestamp	Number	否	0	获取时间戳. 0 为从当前最新时间拉取	4.0.0

代码示例

```
var chatRoom = im.ChatRoom.get({
id: 'chatRoom1'
});
var option = {
timestamp: +new Date(), // 获取时间戳
count: 20 // 获取条数
};
chatRoom.getMessages(option).then(function(result){
var list = result.list; // 历史消息列表
var hasMore = result.hasMore; // 是否还有历史消息可以获取
console.log('获取聊天室历史消息成功', list, hasMore);
});
```

变更回调参数 list.message 属性说明

废弃字段	当前字段	类型	说明
~~isMentioned~~	isMentioned	Boolean	是否为 @ 消息

获取会话列表

变更回调参数 conversationList.conversation 属性说明

废弃字段	当前字段	类型	说明
~~hasMentioned~~	hasMentioned	Boolean	是否包含 @ 自己的消息
~~isMentioned~~	isMentioned	Boolean	是否为 @ 消息
~~mentionedInfo~~	mentionedInfo	Object	@ 信息

废弃属性

非标准接口

旧版本 SDK 顶级变量下暴露了 SDK 内部使用的部分接口，不建议开发者直接持有引用，具体如下：

- `RongIMLib.Logger` 日志上传接口
- `RongIMLib.common` 公共方法集
- `RongIMLib.env` 环境配置

状态码　　链接错误码

更新时间:2024-08-30

调用 connect 失败后, 返回的错误码说明

状态码	说明
31002	AppKey 错误
31003	服务器错误，在小程序端或桌面端未开通对应服务时也会返回该错误。
31004	Token 无效
31005	App 校验未通过（开通了 App 校验功能，但是校验未通过）
31006	链接重定向
31008	AppKey 被封禁或被删除
31009	连接失败，一般因为用户已被封禁。
31012	安全域名错误, 请至控制台查看设置的安全域名。前往控制台查看安全域名设置。
33006	连接中，再调用 connect 被拒绝
34001	连接已存在, 请勿重复链接
35006	公有云包不允许连接私有云环境
35007	调用 reconnect() 前需先调用 disconnect() 方法
35008	不支持的平台类型，如果使用小程序、桌面端，请检查是否已开通对应服务。
35009	Web 端设置安全域名后，连接端域名不在安全域名范围内。前往控制台查看安全域名设置。
35010	开启“禁止把已在线客户端踢下线“开关后，该错误码标识已有同类型端在线，禁止链接

链接状态码

链接过程中, im.watch status 抛出的状态码说明

状态码	说明
0	链接成功
1	正在连接中

状态码	说明
2	用户主动断开链接
3	网络不可用, SDK 内部会自动重连
4	Socket 不可用, SDK 内部会自动重连
6	被其他端踢掉
9	用户被封禁
12	域名错误

业务错误码

状态码	说明
-1	请求超时(一般为本地网络问题)
-2	SDK 内部错误
-3	开发者参数传入错误
405	已被对方加入黑名单
20106	该用户处于单聊禁言状态，禁止发送单聊消息。
20604	发送频率过快
20605	信令被封禁。如遇到该错误，请提交工单。
20606	不支持的操作
20607	调用超过频率限制，请稍后再试。
22406	不在群组中
22408	在群组中被禁言
23406	不在聊天室中
23408	已在聊天室中被禁言
23409	已被踢出并禁止加入聊天室
23410	聊天室不存在
23411	聊天室成员超限
23412	聊天室参数无效

状态码说明

23414 聊天室云存储业务未开通

25101 撤回消息失败

25102 未开通历史消息云存储

25103 清除历史消息时，传递的时间戳大于当前系统时间

25105 清除历史消息时遇到内部异常，请提交工单确认原因

26001 push 设置参数无效

21501 发送的消息中包含敏感词

21502 消息中敏感词已经被替换

30001 用户未连接成功，需连接成功后再执行

30007 导航请求失败(一般为本地网络问题)

30010 CMP 嗅探失败(一般为本地网络问题)

33003 消息扩展参数错误

33007 未开通单群聊历史消息云存储

34008 消息未设置可扩展参数(原消息 canIncludeExpansion 为 false)

34009 发送扩展消息失败

34010 消息扩展 key 或 value 超出限制