即时通信

IMLib / IMKit

Web 2.X

2024-08-30

2.X 停止维护声明 2.X 停止维护声明

更新时间:2024-08-30

尊敬的融云开发者：

融云已不再为 Web/Electron 平台 2.X 版本的 IM SDK 提供技术支持。IM SDK 2.X 版本已于 2023.4.24 起停止维护。详见以下公告：

【下线通知】融云于 2023 年 4 月 24 日停止维护 2.X 和 2.X 版本 SDK

如果您有相关需求，或者您的应用还在使用 Web 平台的 IM 2.X SDK，参见以下提示：

- 新项目请直接集成 IM 5.X SDK。
- 已集成 IM 2.X 版本 SDK 的现有项目建议尽快升级到最新的 5.X 版本 SDK。详见 Web 升级文档 / Electron 升级文档。
- IM 2.X SDK 停止维护后，还在使用的客户端功能不受任何影响，但融云不再提供技术支持。建议您考虑上述方式尽快升级。
- 已替换为 Adapter 版本 SDK 的项目可以继续收到问题修复，但 Adapter SDK 无新功能。详见关于 Adapter 的说明。

如有任何疑问，欢迎提交工单。感谢您对融云的使用和支持！

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]
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 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 2x 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 2x SDK 的客户可以通过 `RongIMLib-v2-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 2x 停止维护。

替换方式

详见升级说明。

SDK 快速集成 SDK 快速集成

更新时间:2024-08-30

关于停止维护 IMLib v2 旧版 SDK 的声明

> 提示
>
> - **Web IMLib v2 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 v2 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 2x SDK 的客户可以通过 `RongIMLib-v2-Adapter` 无缝替换升级。详见 升级说明。**
> - **未来我们将在 `RongIMLib-v2-Adapter` 上进行问题修复，但不会增加新功能。**

导入 SDK

NPM 引入（推荐）

1. 依赖安装

```
npm install @rongcloud/imlib-v2-adapter
```

2. 代码集成。

```
// 非 ESModule
const RongIMLib = require('@rongcloud/imlib-v2-adapter')
// ESModule
import * as RongIMLib from '@rongcloud/imlib-v2-adapter'
```

CDN 引入方式详见引入 SDK。

初始化

> 从 融云开发者控制台 注册后可得到 Appkey。

开发者在使用融云 SDK 所有功能之前，开发者必须先调用此方法初始化 SDK。 在应用整个生命周期中，开发者只需要将 SDK 初始化一次。

```
RongIMLib.RongIMClient.init('appKey'); //控制台 -> 基本信息 获取
```

设置监听

需先设置监听再连接服务器。

```
/* 连接状态监听器 */
RongIMClient.setConnectionStatusListener({
onChanged: function (status) {
/* status 标识当前连接状态 */
console.log('连接成功');
}
});
/* 消息监听器 */
RongIMClient.setOnReceiveMessageListener({
onReceived: function (message) {
console.log(message);
}
});
```

连接融云

- `token` ：用户令牌，相当于当前用户连接融云的身份凭证。在连接融云服务器之前，需要 App Server 通过融云 Server API 获取 Token，客户端获取到这个 Token 即可连接融云服务器。
- `onSuccess` ：连接成功回调，会返回 `token` 对应的 `userId` 。
- `onError` ：连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致。
- `onTokenIncorrect` ：`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接。

```
/* 控制台获取或 Server API */
var token = 'token';
RongIMClient.connect(token, {
onSuccess: function(userId) {
console.log('连接成功, 用户 ID 为', userId);
// 连接已成功, 此时可通过 getConversationList 获取会话列表并展示
},
onTokenIncorrect: function() {
console.log('连接失败, 失败原因: token 无效');
},
onError: function(errorCode) {
console.log('连接失败, 失败原因: ', errorCode);
}
});
```

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**单群聊消息云端存储**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。

获取会话列表需要在 connect 成功后执行。

```
RongIMClient.getInstance().getConversationList({
onSuccess: function(list) {
// list => 会话列表集合
// 获取成功，此处可通过处理 list 在 UI 上展示会话列表
},
onError: function(error) {
// do something
}
}, null);
```

发送消息

> 发送单聊文本消息需要在 connect 成功后执行。

向用户 B 发送一条文本消息，发送成功会走 onSuccess 回调函数，发送失败会走 onError 失败回调，用户可根据 errorCode 进行异常信息调试。

```
var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息' });
var conversationType = RongIMLib.ConversationType.PRIVATE; // 单聊
var targetId = 'userB'; // 用户 B 的 userId
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送成功');
},
onError: function (errorCode, message) {
console.log(errorCode)
}
});
```

接收消息

用户 B 连接服务器成功后，可通过消息监听函数 RongIMClient.setOnReceiveMessageListener 接收用户 A 发送的消息,收到消息会自动触发 onReceived 回调函数。

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**单群聊消息云端存储**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。

获取历史消息需要在 connect 成功后执行。

```
var conversationType = RongIMLib.ConversationType.PRIVATE; //单聊, 其他会话选择相应的消息类型即可
var targetId = "userA"; // 想获取自己和谁的历史消息，targetId 赋值为对方的 userId。类型： string
var timestamp = 0; // 若从头开始获取历史消息，请赋值为 0；
var count = 20; // 每次获取的历史消息条数，范围 0-20 条，可以多次获取
RongIMLib.RongIMClient.getInstance().getHistoryMessages(conversationType, targetId, timestamp, count, {
onSuccess: function(list, hasMsg) {
// list => Message 数组。
// hasMsg => 是否还有历史消息可以获取。
},
onError: function(error) {
console.log('GetHistoryMessages, errorcode:' + error);
}
});
```

断开连接

断开当前用户连接。调用后不再接收消息, 不可发送消息, 不可获取历史消息, 不可获取会话列表

在下次连接融云成功后，会收取上次离线后的消息，离线消息最多可以保存 7 天。

```
RongIMClient.getInstance().disconnect();
```

引入 SDK 引入 SDK 关于停止维护 IMLib v2 旧版 SDK 的声明

更新时间:2024-08-30

提示

- **Web IMLib v2 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 v2 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 2x SDK 的客户可以通过 `RongIMLib-v2-Adapter` 无缝替换升级。详见 升级说明。**
- **未来我们将在 `RongIMLib-v2-Adapter` 上进行问题修复，但不会增加新功能。**

环境要求

浏览器兼容性要求如下：

Chrome	Firefox	Safari	IE	Edge	QQ 浏览器	微信 浏览器	Android
✔	✔	✔	9+	✔	✔	✔	2.3.6+

导入 SDK

NPM 引入（推荐）

1. 依赖安装

```
# 如有，请移除旧版本依赖
npm rm @rongcloud/imlib-v2 @rongcloud/engine
# 安装 adapter-v2
npm install @rongcloud/engine@latest @rongcloud/imlib-v2-adapter@latest -S
```

2. 代码集成

```
// 非 ESModule
const RongIMLib = require('@rongcloud/imlib-v2-adapter')
// ESModule
import * as RongIMLib from '@rongcloud/imlib-v2-adapter'
```

CDN 引入

推荐使用 RongIMLib-v2-Adapter 的最新版，如下：

```
<script src="https://cdn.ronghub.com/RongIMLib-v2-Adapter-5.9.5.prod.js"></script>
```

参考资源

- Web SDK API 示例: https://rongcloud.github.io/web-imlib-v2-test/index.html

初始化 初始化 关于停止维护 IMLib v2 旧版 SDK 的声明

更新时间:2024-08-30

> 提示
>
> - **Web IMLib v2 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 v2 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 2x SDK 的客户可以通过 `RongIMLib-v2-Adapter` 无缝替换升级。详见 升级说明。**
> - **未来我们将在 `RongIMLib-v2-Adapter` 上进行问题修复，但不会增加新功能。**

功能描述

融云 SDK 需要开发者在工程中调用下面方法来初始化 SDK。在应用整个生命周期中，开发者只需要将 SDK 初始化一次。

> 警告
>
> 1. **请在开发功能之前从 融云开发者控制台 注册得到的 Appkey，通过调用 `RongIMLib.RongIMClient.init` 传入 AppKey，进行 SDK 初始化。**
> 2. **开发者在使用融云 SDK 所有功能之前，开发者必须先调用此方法初始化 SDK。 在页面的整个生命周期中，开发者只需要将 SDK 初始化一次。**

API 参考：LogLevel

参数说明

参数	类型	必填	说明	最低版本	废弃版本
appKey	String	是	应用的唯一标识	2.0.0	
~~dataAccessProvider~~	Object	否	仅在桌面版有效，Web 可忽略	2.0.0	2.6.0
options	Object	否	参数配置	2.0.0	

`options` 参数说明

参数	类型	必填	说明	最低版本	废弃版本
navi	String	否	私有部署 navi 导航	2.0.0	
~~api~~	String	否	私有部署 api 地址	2.0.0	2.6.0
~~protobuf~~	String	否	私有部署 protobuf 地址	2.0.0	2.6.0
logLevel	LogLevel	否	输出日志等级	2.8.0	

参数	类型	必填	说明	最低版本	废弃版本
logStdout	Function	否	修改默认日志输出函数	2.8.0	
checkCA	Boolean	否	PC 端的 https 证书认证开关，默认为 true	2.9.4	
readReceiptTimeout	Number	否	web 端群聊天回执状态本地存储过期时间，默认为：1，最大为：15，单位：天	2.9.4	

`LogLevel` 类型说明

关键字	值	说明
DEBUG	0	调试
INFO	1	信息
WARN	2	警告
ERROR	3	错误
NONE	1000	不展示任何日志

`logStdout` 方法参数说明

参数	类型	说明
logLevel	LogLevel	输出日志等级
content	string	日志内容

代码示例

```
var appKey = 'kj29chm026yyn';

var options = {
navi:'', // 私有部署配置，公有云用户可忽略
api:'', // 私有部署配置，公有云用户可忽略
logLevel: 0
}
RongIMLib.RongIMClient.init(appkey, null, options);
```

设置监听　　设置监听 设置状态监听

更新时间:2024-08-30

当 SDK 与融云服务器的连接状态发生变化时，开发者可通过下面方法进行处理。

参数说明

参数	类型	必填	说明	最低版本
param	Function	是	当前连接状态变化时, 触发状态监听器的回调	2.0.0

代码示例

5.6.1 版本之前

```
var params = {
onChanged: function (status) {
// status 标识当前连接状态
switch (status) {
case RongIMLib.ConnectionStatus.CONNECTED:
console.log('连接成功');
break;
case RongIMLib.ConnectionStatus.CONNECTING:
console.log('正在连接');
break;
case RongIMLib.ConnectionStatus.DISCONNECTED:
console.log('断开连接');
break;
case RongIMLib.ConnectionStatus.KICKED_OFFLINE_BY_OTHER_CLIENT:
console.log('其他设备登录, 本端被踢');
break;
case RongIMLib.ConnectionStatus.DOMAIN_INCORRECT:
console.log('域名不正确, 请至控制台查看安全域名配置');
break;
case RongIMLib.ConnectionStatus.NETWORK_UNAVAILABLE:
console.log('网络不可用, 此时可调用 reconnect 进行重连');
break;
default:
console.log('链接状态为', status);
break;
}
}
}

RongIMClient.setConnectionStatusListener(params);
```

链接状态码说明：

状态	说明	枚举值
RongIMLib.ConnectionStatus.CONNECTED	连接成功	0

状态	说明	枚举值
RongIMLib.ConnectionStatus.CONNECTING	连接中	1
RongIMLib.ConnectionStatus.DISCONNECTED	连接已断开	2
RongIMLib.ConnectionStatus.NETWORK_UNAVAILABLE	网络错误	3
RongIMLib.ConnectionStatus.CONNECTION_CLOSED	连接已关闭	4
RongIMLib.ConnectionStatus.KICKED_OFFLINE_BY_OTHER_CLIENT	其他设备登录(被踢)	6
RongIMLib.ConnectionStatus.WEBSOCKET_UNAVAILABLE	WebSocket 不可用	7
RongIMLib.ConnectionStatus.DOMAIN_INCORRECT	域名错误(需通过控制台检查安全域名设置)	12
RongIMLib.ConnectionStatus.REQUEST_NAVI	正在请求导航	201
RongIMLib.ConnectionStatus.RESPONSE_NAVI	请求导航成功	202
RongIMLib.ConnectionStatus.RESPONSE_NAVI_ERROR	请求导航失败	203
RongIMLib.ConnectionStatus.RESPONSE_NAVI_TIMEOUT	请求导航超时	204

5.7.0 版本之后

```
var params = {
onChanged: function (status, code) {
// status 标识当前连接状态， code 表示连接断开原因
switch (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：' + 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：' + code);
break;
}
}
}

RongIMClient.onConnectionStatusChange(params);
```

错误码类型: ErrorCode

设置消息监听

当 SDK 在接收到消息时，开发者可通过下面方法进行处理。

> ⚠ 警告
>
> 音频消息(HQVoiceMessage)支持版本: **RongIMLib 2.5.0** 及以上. 低版本音频消息(HQVoiceMessage)可通过自定义消息 实现

参数说明

参数	类型	必填	说明	最低版本
param	Function	是	当接收到消息时，触发消息监听器的回调	2.0.0

代码示例

```
var params = {
// 接收到的消息
onReceived: function (message) {
console.info(message);
}
};
RongIMClient.setOnReceiveMessageListener(params);
```

设置会话状态监听器

当 SDK 收到会话状态改变时，开发者可通过下面方法进行处理。

参数说明

参数	类型	必填	说明	最低版本
param	Function	是	当接收到会话状态改变时，触发会话状态监听器的回调	2.5.8

代码示例

```
var params = {
//status 标识当前监听到的会话状态
onChanged: function(status) {
console.log(status);
}
};
RongIMClient.setConversationStatusListener(params);
```

设置离线消息拉取完成监听

当 SDK 离线消息拉取完成时，开发者可通过下面方法进行处理。

代码示例

```
var params = {
onFinished: function() {
console.log("离线消息拉取完成");
}
};
RongIMClient.setPullOffLineFinished(params);
```

设置消息敏感词监听

代码示例

```
/**
* 敏感词监听
*/
const listener = {
onReceived(info){
const messageId = info.blockedMessageUId
const conversationType = info.conversationType
const targetId = info.targetId
const blockType = info.blockType
}
}
RongIMClient.setMessageBlockedListener (listener)
```

设置聊天室监听

代码示例

从 2.10.0 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。

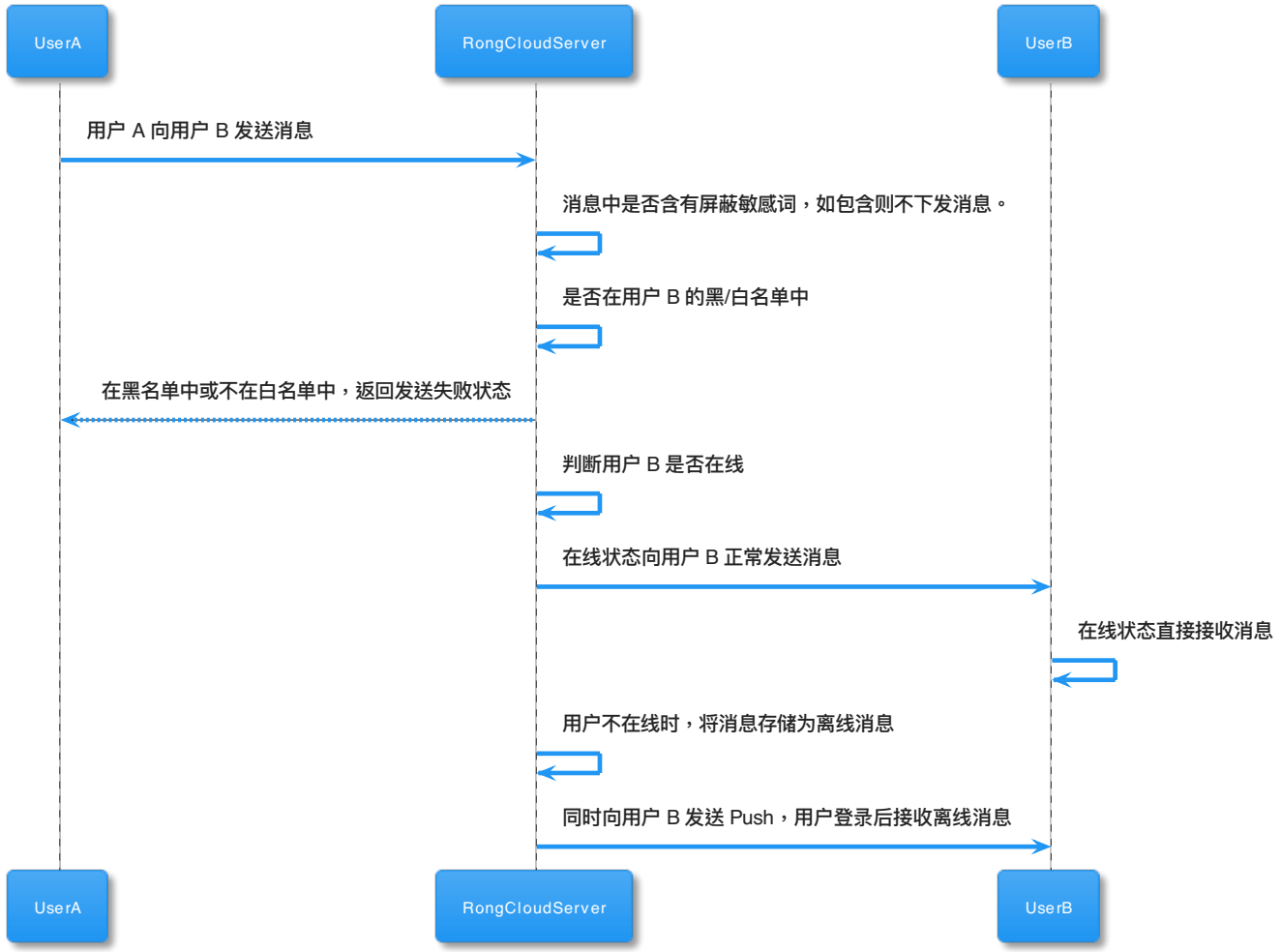
```
/**
* 聊天室监听
*/
const listener = {
onChanged(info){
// 监听到的聊天室 KV 更新
const updatedEntries = info.updatedEntries
// 加入或退出的聊天室成员
const userChange = info.userChange
// 销毁的聊天室 ID
const chatroomDestroyedId = info.chatroomDestroyed
}
}
RongIMClient.setChatRoomStatusListener(listener)
```

单聊介绍 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

消息结构说明

以下为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`
和			
和			
` 替换成空。			
imageUri	String	是	图片上传到图片存储服务器后的地址。
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、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 KB。
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

> 提示
>
> 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：bytes。
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
和			
和			
替换成空。			
duration	Int	是	视频时长，单位：秒。
size	String	是	视频大小单位 bytes。
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。

位置消息

ObjectName	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]

消息结构：

发送位置消息时 content 参数的 JSON 结构如下：

```
{
"content":"bhZPzJXimRwrtvc=",
"latitude":39.9139,
"longitude":116.3917,
"poi":"北京云中融信网络科技有限公司",
"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	表示位置图片缩略图，格式为 JPG，以 Base64 进行 Encode 后需要将所有 `
和			
和			
` 替换成空。			
latitude	String	是	位置的纬度值。
longitude	String	是	位置的经度值。
poi	String	是	表示位置的 poi 信息。
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。

提示小灰条消息

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

消息结构：

发送小灰条消息时 content 参数的 JSON 结构如下：

```
{
"message":"请在聊天中注意人身财产安全",
"extra":""
}
```

属性说明：

名称	类型	必传	说明
message	String	是	提示条消息内容。
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。

资料变更通知消息

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

消息结构：

发送用户资料变更消息时 content 参数的 JSON 结构如下：

```
{
"operation":"Update",
"data":"{\"nickname\":\"韩梅梅\", \"hometown\":\"beijing\"}",
"extra":""
}
```

属性说明：

名称	类型	必传	说明
operation	String	是	资料通知操作，可以自行定义。
data	String	是	操作的数据。
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。

联系人(好友)通知消息

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

消息结构：

发送加好友消息时 content 参数的 JSON 结构如下：

```
{
"operation":"Request",
"sourceUserId":"123",
"targetUserId":"456",
"message":"我是小艾，能加一下好友吗？",
"extra":""
}
```

属性说明：

名称	类型	必传	说明
operation	String	是	联系人操作的指令，官方针对 `operation` 属性定义了 "Request", "AcceptResponse", "RejectResponse" 几个常量，也可以由开发者自行扩展。
sourceUserId	String	是	发出通知的用户 Id。
targetUserId	String	是	单聊会话为接收通知的用户 Id，群聊、聊天室会话为会话 Id。
message	String	是	表示请求或者响应消息，如添加理由或拒绝理由。
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。

显示用户信息 显示用户信息

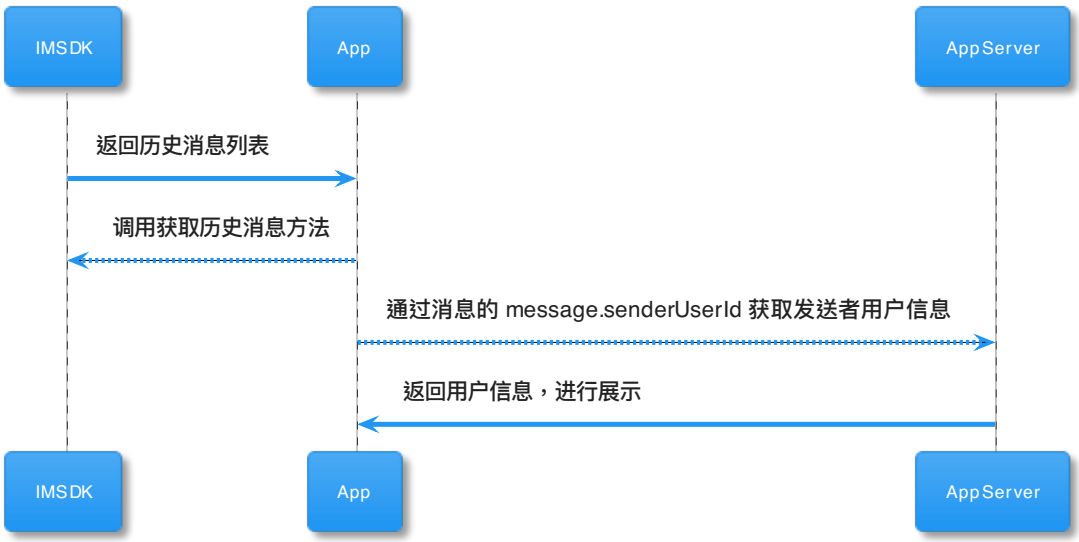
更新时间:2024-08-30

> 提示
>
> 单群聊中用户信息需要开发者自行处理。SDK 不处理用户信息。

历史消息显示用户信息

通过开发者 App Server 获取用户信息

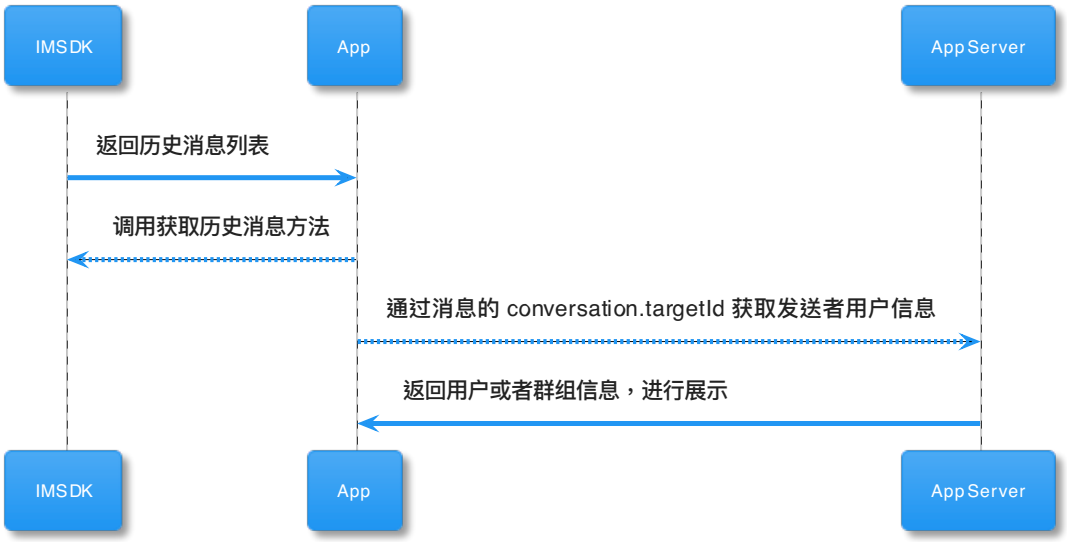
1. 开发者 App Server 封装获取用户信息接口
2. 通过 message.senderUserId 获取发送者 id
3. 将发送者 id 传入 App Server 暴露的接口中，获取对应用户信息
4. 将用户信息展示到页面中



会话列表显示用户信息

通过开发者 App Server 获取用户信息

1. 开发者 App Server 封装获取用户或群组信息接口
2. 通过 conversation.targetId 获取用户或者群组 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


IMSDK
App
AppServer
返回历史消息列表
调用获取历史消息方法
通过消息的 conversation.targetId 获取发送者用户信息
返回用户或者群组信息，进行展示
IMSDK
App
AppServer

连接服务　　连接服务 功能描述

更新时间:2024-08-30

连接融云服务器之前，需要 App Server 通过融云 Server API 获取 Token，客户端获取到这个 Token 即可连接融云服务器。

⚠ 警告

1. 连接方法必须在执行初始化之后调用。详见初始化文档。
2. 连接方法必须在设置状态监听器和消息监听器之后调用。详见设置监听文档。
3. 除初始化、监听以外，所有方法都必须在 connect 成功之后 再调用
4. 默认一个用户只支持一个页面连接，开通 多设备消息同步 即可支持多页面连接

参数说明

参数	类型	必填	说明	最低版本
token	String	是	用户的唯一标识	2.0.0
callback	Object	是	重连回调对象	2.0.0
callBack.onSuccess	Function	是	连接成功回调，会返回 `token` 对应的 `userId`	2.0.0
callBack.onError	Function	是	连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.0.0
callBack.onTokenIncorrect	Function	是	`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接	2.0.0

代码示例

```
RongIMClient.connect('<Your-Token>', {
onSuccess: function(userId) {
console.log('连接成功，用户 ID 为', userId);
// 连接已成功，此时可通过 getConversationList 获取会话列表并展示
},
onTokenIncorrect: function() {
console.log('连接失败，失败原因：token 无效');
},
onError: function(errorCode) {
console.log('连接失败，失败原因：', errorCode);
}
});
```

重连逻辑 重连逻辑

更新时间:2024-08-30

功能描述

在 v2.6.0 及以上版本中，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 RongIMClient.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 RongIMClient.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 RongIMClient.reconnect() 进行重新连接。

在 v2.5.x 版本中 SDK 内无重连机制，重新连接调用时机：

1. 网络不好，导致连接频繁断开后不会重新连接。
2. 长时间断网后，IM不会重新连接。
3. 交互连接频繁断开（网络环境：WIFI 或者联通、移动 3/4G ），交替切换网络后，IM不会重新连接。

在 v2.9.4 及以上版本中调用 RongIMClient.reconnect() 前需先调用 RongIMClient.disconnect() 方法，否则会报错误 35007。

参数说明

参数	类型	必填	说明	最低版本	废弃版本
callback	Object	是	重连回调对象	2.3.3	
callBack.onSuccess	Function	是	连接成功回调，会返回 `token` 对应的 `userId`	2.3.3	
callBack.onError	Function	是	连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.3.3	
callBack.onTokenIncorrect	Function	是	`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接	2.3.3	
~~config~~	Object	否	重连配置	2.3.3	2.6.0

代码示例

```
var callback = {
onSuccess: function(userId) {
console.log('reconnect success. ' + userId);
},
onTokenIncorrect: function() {
console.log('token 无效');
},
onError: function(errorCode) {
console.log(errorCode);
}
};
RongIMClient.reconnect(callback);
```

断开连接 断开连接 断开连接

更新时间:2024-08-30

断开当前用户的连接。调用后将不再接收消息，不可发送消息，不可获取历史消息，不可获取会话列表。

> 提示
>
> 在下次连接融云成功后，会收取上次离线后的消息，离线消息最多可以保存 7 天。

代码示例

```
RongIMClient.getInstance().disconnect();
```

退出登录

切换用户时使用，示例见: Web SDK API 示例

代码示例

```
RongIMClient.getInstance().logout();
```

多端同时在线 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。

好友关系 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 获取全部会话 获取本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供获取本地会话列表接口。

获取远端会话列表

会话列表生成规则 : 服务端会根据消息来生成初始会话列表, 在收到消息和发送消息之后服务端会更新会话列表。

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。

API 参考：getConversationList

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	是	获取的会话类型, 获取所有会话类型传 null	2.3.3
count	Number	否	获取会话数量	2.3.3

`conversationTypes` 枚举值说明

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

回调参数说明

`onSuccess` 说明：

回调参数	类型	说明
list	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
conversationType	Number	会话类型
targetId	String	单聊会话 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识，融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息，消息结构详见消息数据结构
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间
isTop	Boolean	会话置顶状态
notificationStatus	Number	会话免打扰状态

代码示例

```
var count = 150;
var callback = {
onSuccess: function(list) {
console.log('获取会话列表成功', list);
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, count);
```

⚠ 警告

返回的会话列表 list 长度是在传递的 count 基础上进一步筛选 conversationTypes 的结果，数量会小于等于 count。

常见问题

Q1: 切换用户后获取会话列表，会获取到不属于当前用户的会话

A1: 解决方案

在退出之前先清除掉缓存在调用 logout

RongIMClient.getInstance().clearCache();

RongIMClient.getInstance().logout();

Q2: 会话列表的名字和头像怎么维护

A2: 解决方案

1、用户信息在您的服务器维护，例如：用户 Id、头像、名称等

2、通过会话列表中的 targetId、senderUserId 在您应用服务器获取用户信息

3、使用用户信息与会话列表通过 targetId、senderUserId 做为对应关系将数据合并

4、将列表渲染至页面

说明：

targetId: 接收方用户或群组 ID

senderUserId: 消息发送人 ID

删除全部会话　　删除全部会话 清除本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供清除本地会话列表接口。

清除远端会话列表

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	清除的会话类型, 不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

获取指定会话　　获取指定会话 功能描述

更新时间:2024-08-30

此方法获取的为本地缓存数据，必须在调用 getConversationList 之后再调用。

API 参考：getConversation

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，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

回调参数说明

返回值	返回类型	说明
conversation	Object	返回获取的会话信息

`conversation` 属性说明

字段名	类型	说明
conversationType	Number	会话类型
targetId	String	单聊会话 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识, 融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';
RongIMClient.getInstance().getConversation(conversationType, targetId, {
onSuccess: function(conversation) {
if (conversation) {
console.log('获取指定会话成功', conversation);
}
},
onError: function (error) {
console.log('获取指定会话失败:', error)
},
});
```

删除指定会话 删除指定会话 删除指定会话

更新时间:2024-08-30

API 参考：removeConversation

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = PRIVATE;
var targetId = '单聊会话 ID';
var callback = {
onSuccess: function() {
console.log('删除指定会话成功');
},
onError: function(error) {
console.log('删除指定会话失败', error);
}
};
RongIMClient.getInstance().removeConversation(conversationType, targetId, callback);
```

按会话类型删除

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

参数	类型	必填	说明	最低版本
conversationTypes	Array	否	清除的会话类型，不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

会话草稿信息 会话草稿信息 获取草稿

更新时间:2024-08-30

获取内存中存储的草稿信息。

API 参考：getTextMessageDraft

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，单聊会话传入 PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0

回调参数说明

回调参数	类型	说明
draft	String	返回草稿信息

代码示例

```
var conversationType = PRIVATE;
var targetId = '单聊会话 ID';
var draft = RongIMClient.getInstance().getTextMessageDraft(conversationType, targetId);
console.log('草稿信息为: ', draft);
```

保存草稿

草稿存储在内存中，如刷新或者关闭页面会导致草稿丢失。

API 参考：saveTextMessageDraft

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，单聊会话传入 PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
draftText	String	是	草稿信息	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	保存草稿接口操作状态

代码示例

```
var conversationType = PRIVATE;
var targetId = '单聊会话 ID';
var draftText = '草稿信息';
var bool = RongIMClient.getInstance().saveTextMessageDraft(conversationType, targetId, draftText);
```

删除草稿

API 参考：clearTextMessageDraft

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，单聊会话传入 PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	删除草稿接口操作状态

代码示例

```
var conversationType = PRIVATE;
var targetId = '单聊会话 ID';
var bool = RongIMClient.getInstance().clearTextMessageDraft(conversationType, targetId);
```

常见问题

Q1: Web 端设置了草稿，获取会话列表没有取到草稿？

A1: 草稿为本地存储的，如果您需要显示需要您按照您的逻辑做下 UI 渲染。

会话未读数　　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

> ⚠ 警告
>
> 1. 清除浏览器缓存会导致会话未读数不准确
> 2. 会话未读数为本地操作，换端登录会话未读数不会同步
> 3. 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

API 参考：getTotalUnreadCount

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	会话类型	2.9.5
includeMuted	Boolean	否	是否包含免打扰会话，默认为：false	2.9.5

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE];
var callback = {
onSuccess: function(count) {
console.log('获取所有会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取所有会话未读消息数失败', error);
}
};
RongIMClient.getInstance().getTotalUnreadCount(callback, conversationTypes);
```

获取单个会话未读数

方式一：调用 getUnreadCount 方法来获取会话未读数。

方式二：通过 conversation.unreadMessageCount 获取。

API 参考：getUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';
var callback = {
onSuccess: function(count) {
console.log('获取指定会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话未读消息数失败', error);
}
};
RongIMLib.RongIMClient.getInstance().getUnreadCount(conversationType, targetId, callback);
```

按会话类型获取未读数

API 参考：getConversationUnreadCount

参数说明

参数	类型	必填	说明	最低版本	废弃版本
conversationTypes	Array	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0	2.6.0
callback	Object	是	回调对象	2.2.0	2.6.0
callback.onSuccess	Function	是	成功回调	2.2.0	2.6.0
callback.onError	Function	是	失败回调	2.2.0	2.6.0

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE];
var callback = {
onSuccess: function(count) {
console.log('获取指定会话类型总未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话类型总未读消息数失败', error);
}
};
RongIMClient.getInstance().getConversationUnreadCount(conversationTypes, callback);
```

清除单个会话未读数

API 参考：clearUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';
var callback = {
onSuccess: function() {
console.log('清除指定会话未读消息数成功');
},
onError: function(error) {
console.log('清除指定会话未读消息数失败', error);
}
};
RongIMClient.getInstance().clearUnreadCount(conversationType, targetId, callback);
```

清除全部会话未读数

> 最低版本：2.10.4

```
RongIMClient.getInstance().clearAllUnreadCount()
```

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

本端清除会话未读数后，通过在会话中发送一条同步消息，通知其他端。其他端收取到会话中的同步消息后，调用相应方法清除本地会话未读消息数。

1. 本端阅读会话中的消息后，调用 clearUnreadCount() 清除会话未读消息数。
2. 清除成功后，在会话中发送 SyncReadStatusMessage 类型消息，同步阅读状态。
3. 其他端接收到 SyncReadStatusMessage 类型消息后，调用 clearUnreadCount() 方法，清除本地的会话未读数。（在 v2.10.4 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
var im = RongIMClient.getInstance();
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId';
im.clearUnreadCount(conversationType, targetId, {
onSuccess: function () {
// 从消息里获取服务器端时间，以最近一条已读 message 为准
var msg = {
lastMessageSendTime: message.sentTime
};
msg = new RongIMLib.SyncReadStatusMessage(msg);
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function () {
console.log('发送同步消息成功', message);
},
onError: function () {}
});
},
onError: function (errorCode) {}
});
```

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 clearUnreadCount() 成功后不需要再在发送 SyncReadStatusMessage 类型消息。
var clearUnreadCount = RongIMClient.getInstance().clearUnreadCount;
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId';
clearUnreadCount(conversationType, targetId, {
onSuccess: function () {},
onError: function (errorCode) {}
});
```

会话免打扰　　会话免打扰 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.5.8
targetId	String	是	单聊会话 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId';
var statusItem = {
notificationStatus: 1
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

会话置顶　　会话置顶 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.5.8
targetId	String	是	单聊会话 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId';
var statusItem = {
isTop: true
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

会话标签 会话标签

更新时间:2024-08-30

> SDK 2.X 版本从 2.8.0 开始支持会话标签功能。

会话标签用于对会话进行分组。用户可设置会话标签，用于会话的分组显示。

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。

管理标签

SDK 支持创建标签、编辑标签，获取标签列表，以及移除标签。SDK 创建的标签可用于对会话进行管理。

SDK 创建的标签会被同步到服务端。

> 关于如何使用标签管理会话，详见使用标签管理会话。

创建标签

每个用户最多可以创建 20 个标签。

API 参考：createTag

参数说明

参数	类型	必填	说明
tag	Object	是	标签信息
tag.tagId	String	是	标签 Id，标签唯一标识，长度不超过 10 个字
tag.tagName	String	是	标签名称，长度不超过 15 个字
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('创建标签成功')
},
onError: function(error) {
console.log('创建标签失败', error)
}
}
RongIMClient.getInstance().createTag(tag,callback)
```

移除标签

API 参考：removeTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tagId = '<标签Id>'
var callback = {
onSuccess: function() {
console.log('删除标签成功')
},
onError: function(error) {
console.log('删除标签失败', error)
}
}
RongIMClient.getInstance().removeTag(tagId,callback)
```

编辑标签

参数说明

参数	类型	必填	说明
tag	Object	是	标签信息
tag.tagId	String	是	标签 Id，标签唯一标识，长度不超过 10 个字
tag.tagName	String	是	标签名称，长度不超过 15 个字
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调

参数	类型	必填	说明
callback.onError	Function	是	失败回调

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('编辑标签成功')
},
onError: function(error) {
console.log('编辑标签失败', error)
}
}
RongIMClient.getInstance().updateTag(tag,callback)
```

获取标签列表

API 参考：getTagList

参数说明

参数	类型	必填	说明
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

成功回调

成功回调 `ITagInfo[]` 的参数说明：

参数	返回类型	说明
tagInfo	ITagInfo[]	标签列表信息，详见下方 `ITagInfo` 属性说明

- `ITagInfo` 属性说明

参数	类型	说明
tagId	String	标签 Id
tagName	String	标签名称
conversationCount	Number	标签下的会话数量
createdTime	Number	标签创建的时间戳

代码示例

```
var callback = {
onSuccess: function(tagInfo) {
console.log(tagInfo)
console.log('获取标签列表成功')
},
onError: function(error) {
console.log('获取标签列表失败', error)
}
}
RongIMClient.getInstance().getTagList(callback)
```

用户标签多端同步

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。

代码示例

```
/**
* 设置会话标签更改时的回调
* 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。
* 收到此回调，可调用 getTagList 获取最新标签信息
* 请在初始化之后，连接之前调用该方法
*/
var tagListener = {
onChanged:function(){
return
}
}
RongIMClient.setTagListener(tagListener)
```

使用标签管理会话

SDK 支持使用一个或多个标签标记会话，可用于会话的分组显示、获取会话列表等操作，并提供了多个接口用于移除会话上的标签。

会话标签还支持一系列高级功能，包括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，以及在特定标签下设置某个会话置顶。

添加会话到一个标签

每个标签下最多可以添加 1000 个会话。

API 参考：addTagForConversations

参数说明

参数	类型	必填	说明
tagId	String	是	标签信息
conversations	IConversationOption []	是	要添加的会话列表。详见下方 `IConversationOption` 属性说明。
callback	Object	是	回调对象

参数	类型	必填	说明
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('添加成功')
},
onError: function(error) {
console.log('添加失败', error)
}
}
var conversations =[{
type:1,//会话类型
targetId:'<目标Id>'
}
]
RongIMClient.getInstance().addTagForConversations(tagId,conversations,callback)
```

删除指定标签中某些会话

`removeTagForConversations` 接口可从特定标签下移除指定多个会话，也可以理解为“为多个会话移除单个标签”。

API 参考：removeTagForConversations

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
conversations	IConversationOption[]	是	要从指定标签下删除的会话列表。详见下方 `IConversationOption` 属性说明。
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagId = '<标签Id>'
var callback = {
onSuccess: function() {
console.log('删除成功')
},
onError: function(error) {
console.log('删除失败', error)
}
}
var conversations =[{
type:1,//会话类型
targetId:'<目标Id>'
}]
RongIMClient.getInstance().removeTagForConversations(tagId,conversations,callback)
```

删除指定会话中的某些标签

`removeTagsForConversation` 接口可为特定会话移除指定的一个或多个标签。

API 参考：removeTagsForConversation

参数说明

参数	类型	必填	说明
conversation	IConversationOption	是	会话。详见下方 `IConversationOption` 属性说明。
tagIds	Array	是	所有标签
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明。

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagIds = ['<标签Id>']
var callback = {
onSuccess: function() {
console.log('删除成功')
},
onError: function(error) {
console.log('删除失败', error)
}
}
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
RongIMClient.getInstance().removeTagsForConversation(conversation,tagIds,callback)
```

获取指定会话下的所有标签

API 参考：getTagsForConversation

参数说明

参数	类型	必填	说明
conversation	IConversationOption	是	会话。详见下方 IConversationOption 属性说明。
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- IConversationOption 属性说明。

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

成功回调

成功回调 IConversationTag 的参数说明：

参数	返回类型	说明
tagId	String	标签 Id
tagName	String	标签名称
createdTime	Number	标签创建的时间戳
isTop	Boolean	是否置顶

代码示例

```
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
var callback = {
onSuccess: function(IConversationTag) {
console.log('获取成功',IConversationTag)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getTagsForConversation(conversation,callback)
```

分页获取指定标签下的会话列表

getConversationListByTag 可从服务端分页获取会话列表，并返回分页查询结果中带有指定标签的会话。

API 参考：getConversationListByTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
count	Number	是	单次分页查询时需要从服务端获取的会话数量
startTime	Number	是	会话中最后一条消息时间戳
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tagId = '<标签Id>'
var count = 10
var startTime = 1649404018538
var callback = {
onSuccess: function(IV2ConversationContainTag) {
console.log('获取成功后会话列表',IV2ConversationContainTag)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getConversationListByTag(tagId,count,startTime,callback)
```

按标签获取未读消息数

API 参考：getUnreadCountByTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
containMuted	Number	是	是否包含已设置为免打扰状态的会话
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

成功回调

成功回调参数说明：

参数	返回类型	说明
count	Number	获取未读消息数

代码示例

```
var tagId = '<标签Id>'
var containMuted = true
var callback = {
onSuccess: function(count) {
console.log('成功获取未读消息数',count)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getUnreadCountByTag(tagId,containMuted,callback)
```

设置标签中会话置顶

`setConversationStatusInTag` 可设置会话在标签下为置顶状态。

设置成功后，可在分页获取指定标签下会话列表（`getConversationListByTag`）返回的结果中查看相关属性。

- **注意，`setConversationStatusInTag` 接口仅用于在特定标签下会话是否置顶。**
- **如需设置会话在会话列表中置顶，请使用 `setConversationStatus` 接口（SDK ≧ 2.5.8）。**

API 参考：setConversationStatusInTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
conversation	IConversationOption	是	会话。详见下方 `IConversationOption` 属性说明。
status	Object	是	置顶状态

参数	类型	必填	说明
status.isTop	Boolean	是	是否置顶
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagId = '<标签Id>'
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
let status = {
isTop:true
}
var callback = {
onSuccess: function() {
console.log('设置成功')
},
onError: function(error) {
console.log('设置失败', error)
}
}
RongIMClient.getInstance().setConversationStatusInTag(tagId,conversation,status,callback)
```

用户会话标签多端同步

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。

代码示例

```
var tagListener = {
onChanged:function(){
return
}
}
RongIMClient.setConversationTagListener(tagListener)
```

消息发送　　消息发送 功能描述

更新时间:2024-08-30

消息属性	消息描述	消息属性	消息描述
消息类名	各端消息名	ObjectName	传输层名称，与消息类名一一对应
存储属性	存储 / 不存储	计数属性	计数 / 不计数
离线属性	缓存 / 不缓存	消息尺寸	128 KB
推送属性	是/否	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，需开通历史消息云存储服务，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。

发送普通消息

API 参考：sendMessage

参数需按顺序传入，顺序为参数说明中的字段顺序。

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

`config` 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0

参数	类型	必填	说明	最低版本
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 `isStatusMessage` 替代该参数。在 `isStatusMessage` 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE; // 群聊, 其他会话选择相应的消息类型即可
var targetId = '接收方的 userId'; // 目标 Id
var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息'});
var callBack = {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
};
var isMentioned = false; // @ 消息
var pushContent = 'user 发送了一条消息'; // Push 显示内容
var methodType = null; // 已经逐步废弃，填写null即可
var pushData = null; // Push 通知时附加信息, 可不填
var config = {
isVoipPush: true, // 发送 voip push
disableNotification: true // 发送静默消息， 设置为 true 后移动端不会收到 Push 信息和本地通知提醒
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, methodType, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE
targetId	String	单聊会话 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	Number	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, PC 端有效，Web 端无效
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

字段名	类型	说明
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

文本消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TextMessage	RC:TxtMsg	存储	计数	存储	推送	消息内容

TextMessage 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 textMessageInfo = {
content: 'hello RongCloud!',
extra: '附加信息'
}
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
});
```

图文消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RichContentMessage	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

属性名称	属性类型	是否必填	属性说明
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 msg = new RongIMLib.RichContentMessage({
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'
});
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送富文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送富文本消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

代码示例

```
var textMessageInfo = { content: '□' }
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送 Emoji 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 Emoji 消息失败', errorCode);
}
});
```

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

```
<!-- HTTP -->
<script src="http://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- HTTPS -->
<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-v2-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

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
LocationMessage	RC:LBSMsg	存储	计数	存储	推送	[位置]

LocationMessage 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 locatMessageInfo = {
latitude: 40.0317727,
longitude: 116.4175057,
poi: '融云',
content: '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE', // 位置图片 base64
}
var msg = new RongIMLib.LocationMessage(locatMessageInfo);

var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送位置消息成功', message);
},
onError: function (errorCode) {
console.log('发送位置消息失败', errorCode);
}
});
```

正在输入状态消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TypingStatusMessage	RC:TypSts	不存储	不计数	不存储	不推送	无

TypingStatusMessage 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 typingContentType = 'RC:TxtMsg';
var msg = new RongIMLib.TypingStatusMessage({ typingContentType: typingContentType});

var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 Id
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送已读通知成功', message);
},
onError: function (errorCode) {
console.log('发送已读通知失败', errorCode);
}
});
```

发送媒体消息

API 参考：sendMessage

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0

参数	类型	必填	说明	最低版本
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

config 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 isStatusMessage 替代该参数。在 isStatusMessage 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId'; // 目标 ID

var base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});

var callBack = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};

var isMentioned = false; // @ 消息
var pushContent = 'user 发送了一条消息'; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填
var config = {
isVoipPush: true // 发送 voip push
};

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，单聊会话传入 RongIMLib.ConversationType.PRIVATE

字段名	类型	说明
targetId	String	单聊会话 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	Number	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, PC 端有效，Web 端无效
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
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

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，文件存储到优先级高的云存储，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。

API 参考：sendMessage

消息发送

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
ImageMessage	RC:ImgMsg	存储	计数	存储	推送	[图片]

ImageMessage 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 base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 Id
var callback = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

图片上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.IMAGE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

语音消息

> 提示
>
> - 语音上传请参照 文件上传 进行是实现。
> - HQVoiceMessage 消息支持版本： **v2.5.0** 及以上。

API 参考：sendMessage

消息说明

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

HQVoiceMessage 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 messageInfo = {
remoteUrl: 'http://rongcloud-file.ronghub.com/1e0e4743249cd9653b.aac',//此地址为模拟地址仅作为示例使用
duration: 22,
extra: '附加信息'
}
var msg = new RongIMLib.HQVoiceMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送语音消息成功', message);
},
onError: function (errorCode) {
console.log('发送语音消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

消息发送

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

FileMessage 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	文件尺寸，单位: Byte
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 msg = new RongIMLib.FileMessage({
name: 'RongIMLib.js', // 文件名,
size: 1024, // 文件大小单位 bytes
type: 'js', // 文件类型
fileUrl: 'https://cdn.ronghub.com/RongIMLib.js' // 文件地址
});
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID
var callback = {
onSuccess: function (message) {
console.log('发送文件消息成功', message);
},
onError: function (errorCode) {
console.log('发送文件消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

文件上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.FILE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

小视频消息

> ⚠ 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。
> 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**。

API 参考：sendMessage

消息说明

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

SightMessage 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
和		
和		
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 params = {
content: "/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
sightUrl: "http://rongcloud...com/video...",
duration: 10,
size: 734320,
name: "video_xx.mp4",
}
var msg = new RongIMLib.SightMessage(params);
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送小视频消息成功', message);
},
onError: function (errorCode) {
console.log('发送小视频消息失败', errorCode);
}
});
```

GIF 消息

API 参考：sendMessage

消息说明

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

GIFMessage 参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 bytes
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 messageInfo ={
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
var msg = new RongIMLib.GIFMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.PRIVATE; //选择相应的会话类型即可
var targetId = '单聊会话 ID';
var callback = {
onSuccess: function (message) {
console.log('发送 GIF 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 GIF 消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

发送自定义消息

注册

> ⚠ 警告
>
> 1. 注册自定义消息代码必须在发送、接收该自定义消息之前
> 2. 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
> 3. 注册的消息类型名，必须多端一致，否则消息无法互通
> 4. 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
messageName	String	消息名称
objectName	String	消息类型名
mesasgeTag	Object	存储，计数标识
searchProp	string[]	搜索字段，web 端无需设置，搜索字段值设置为数字时取值范围为 (-Math.pow(2, 64), Math.pow(2, 64)) 且为整数

代码示例

```
var messageName = 'PersonMessage'; // 消息名称
var objectName = 's:person'; // 消息类型名，请按照此格式命名
var isCounted = true; // 消息计数
var isPersited = true; // 消息保存
var mesasgeTag = new RongIMLib.MessageTag(isCounted, isPersited);
var searchProp = ['name', 'age']; // 搜索字段中的属性名
RongIMClient.registerMessageType(messageName, objectName, mesasgeTag, searchProp);
```

发送

API 参考：sendMessage

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';
var msg = new RongIMClient.RegisterMessage.PersonMessage({ name: 'zhang', age: 12 });

var callback = {
onSuccess: function (message) {
console.log('发送自定义消息成功', message);
},
onError: function (errorCode) {
console.log('发送自定义消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

配置消息推送

MessagePushConfig 属性介绍

> ⚠ 警告
>
> 从 SDK 2.8.0 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。

API 参考：sendMessage

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）
category	String	iOS 平台通知分类

参数	类型	说明
richMediaUri	String	iOS 平台推送自定义的通知栏消息右侧图标 URL

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 conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID'; // 目标 ID
var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息'});

var callbacks = {
onSuccess: function (message) {
console.log('发送消息成功', message);
},
onError: function (errorCode) {
console.log('发送消息失败', errorCode);
}
}
let config = {
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"
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callbacks, null, null, null,
null, config);
```

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)。

具体方法请参见 Emoji 消息。

消息接收 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，详细可参考 设置消息监听 文档。

历史消息获取　　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。

API 参考：getHistoryMessages

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，单聊会话传入 RongIMLib.ConversationType.PRIVATE	2.2.0
targetId	String	是	单聊会话 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
objectName	String	否	消息类名,仅桌面版解决方案有用	2.3.4
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
hasMsg	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
conversationType	Number	会话类型
targetId	String	单聊会话 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

字段名	类型	说明
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.ConversationType.PRIVATE;
var targetId = '单聊会话 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);
}
});
```

消息回执　　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。

发送回执

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

API 参考：sendMessage

参数说明

输入参数说明

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

回调参数说明

请参考 message 属性说明

代码示例

```
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

var msg = new RongIMLib.ReadReceiptMessage({ messageUId: messageUId, lastMessageSendTime:
lastMessageSendTime, type: type });
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '接收方的 userId'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送已读通知成功', message);
},
onError: function (errorCode) {
console.log('发送已读通知失败', errorCode);
}
});
```

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 ReadReceiptMessage 消息，收到后按需处理即可。消息监听。

消息撤回 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。

API 参考：sendRecallMessage

参数说明

输入参数说明

属性名称	属性类型	是否必填	属性说明
recallMessage	Object	是	需要撤回的消息对象
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

回调参数说明

请参考 message 属性说明

代码示例

```
// recallMessage 为需要撤回的消息对象
RongIMClient.getInstance().sendRecallMessage(recallMessage, {
onSuccess: function (message) {
console.log('撤回成功', message);
},
onError: function (errorCode) {
console.log('撤回失败', errorCode);
}
});
```

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 RecallCommandMessage 消息，收到后按需处理即可。

详见「设置监听」文档中的设置消息监听。

消息转发 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　　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

API 参考：deleteRemoteMessages

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传 RongIMLib.ConversationType.PRIVATE	2.5.3
targetId	String	是	单聊会话 ID	2.5.3
messages	Object	是	要删除的消息数组, 不能超过 100 条	2.5.3
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

`messages` 参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	2.3.5
sentTime	Number	是	消息发送时间	2.3.5
messageDirection	Number	是	消息发送方向	2.3.5

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = '单聊会话 ID';
/*
message 对象可通过历史消息获取
messageUId、sentTime、messageDirection 必传，且必须正确
*/
var messages = [
{ messageUId: 'BETR-GIM7-TNO5-89QU', sentTime: 1575965764383, messageDirection: 1 },
{ messageUId: 'AEGR-CFN7-QFJO-8OC0', sentTime: 1575965744371, messageDirection: 2 }
];
RongIMLib.RongIMClient.getInstance().deleteRemoteMessages(conversationType, targetId, messages, {
onSuccess: function() {
console.log('清除成功');
},
onError: function(error) {
console.log('清除失败', error);
}
});
```

通过时间戳删除

API 参考：clearRemoteHistoryMessages

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，单聊会话传 RongIMLib.ConversationType.PRIVATE	2.3.5
targetId	String	是	单聊会话 ID	2.3.5
timestamp	Number	是	清除时间点，该时间之前的消息将被清除	2.3.5

代码示例

```
var params = {
conversationType: RongIMLib.ConversationType.PRIVATE,
targetId: '单聊会话 ID',
timestamp: 1513308018122 // 可取 sentTime，收发消息和历史消息中都有 sentTime 字段
};
RongIMLib.RongIMClient.getInstance().clearRemoteHistoryMessages(params, {
onSuccess: function() {
console.log('清除成功');
},
onError: function(error) {
console.log('清除失败', error);
}
});
```

消息扩展 消息扩展

更新时间:2024-08-30

消息扩展功能可为消息增加基于 Key/Value 的状态标识。消息的扩展信息可在发送前、后设置或更新，可用于实现消息评论、礼物领取、订单状态变化等业务需求。

一条消息是否可携带或可设置扩展信息，由发送消息时 Message 的可扩展（canIncludeExpansion）属性决定，发送后无法修改该属性。因此消息发送前需要将消息设置为可扩展，才能对该条消息进行扩展信息添加、删除等操作。

单条消息单次最多可设置 20 个扩展信息 KV 对，总计不可超过 300 个扩展信息 KV 对。在并发情况下如出现设置超过 300 个的情况，超出部分会被丢弃删除。

消息扩展（Key、Value 扩展信息）会被存储。如已开通历史消息云存储功能，从服务端获取的历史消息也会携带已设置的扩展信息。

- 2.x SDK 从 `2.5.12` 版本开始支持消息扩展功能。
- 仅支持单聊、群聊会话类型。不支持聊天室和系统会话。

实现思路

以订单状态变化为例，可通过消息扩展改变消息显示状态。以订单确认为例：

1. 当用户购买指定产品下单后，商家需要向用户发送订单确认信息。可在发送消息时，将 `canIncludeExpansion` 属性设置为可扩展，同时设置用于标识订单状态的 Key 和 Value。例如，在用户未确认前，可用一对 Key/Value 表示该订单状态为「未确认」。
2. 用户点击确认（或其他确认操作）该订单消息后，订单消息状态需要变更为「已确认」。此时，可通过 `updateMessageExpansion` 更新此条消息的扩展信息，标识为已确认状态，同时更改本地显示的消息样式。
3. 发送方通过消息扩展状态监听，获取指定消息的状态变化，根据最新扩展信息显示最新的订单状态。

消息评论、礼物领取可参照以上实现思路：

- 礼物领取：可通过消息扩展改变消息显示状态实现。例如，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。
- 消息评论：可通过设置原始消息扩展信息的方式添加评论信息。

设置监听

```
RongIMClient.setMessageExpansionListener({
onUpdated: function(updatedExpansion) { // 新增、修改扩展监听
/*
updatedExpansion: {
expansion: { key: value },
messageUId: 'BKCP-VBUT-0006-9GPP'
}
*/
var expansion = updatedExpansion.expansion; // 新增、修改的扩展信息
var messageUId = updatedExpansion.messageUId; // 扩展消息的 messageUId
},
onDeleted: function(deletedExpansion) { // 删除扩展监听
/*
deletedExpansion: {
deletedKeys: ['key1', 'key2'],
messageUId: 'BKCP-VBUT-0006-9GPP'
}
*/
let deletedKeys = deletedExpansion.deletedKeys; // 删除扩展的 keys
let messageUId = deletedExpansion.messageUId; //扩展消息的 messageUId
}
})
```

设置消息是否支持扩展

发消息前，设置 CanIncludeExpansion 属性为 true，打开消息的可扩展属性。在 expansion 中设置发消息时需要携带的扩展信息。

API 参考：sendMessage

> **注意：**
>
> - 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。
> - `CanIncludeExpansion` 属性在发送消息的时候可以设置，消息发出去之后不可更改。
> - 会话列表最后一条消息仅保存发送时携带的扩展，后续更新、删除扩展不会影响会话列表最后一条消息的扩展内容。

参数说明:

参数	类型	必填	默认值	说明
canIncludeExpansion	Boolean	否	false	是否支持扩展。
expansion	Object	否	--	扩展内容。详见 `expansion` 参数说明。

- `expansion` 参数说明
  - 单次最大设置扩展信息键值对 20 对。单条消息可设置最多 300 个扩展信息。
  - Key 支持大小写英文字母、数字、 特殊字符`+ = - _` 的组合方式，不支持汉字。最大 32 个字符。
  - （SDK < 5.2.0）Value 最大 64 个字符。

- （SDK ≧ 5.2.0）Value 最大 4096 个字符。

代码示例

```
var conversationType = RongIMLib.ConversationType.PRIVATE; // 群聊，其他会话选择相应的消息类型即可
var targetId = '接收方的 userId'; // 目标 Id
var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息'});
var config = {
canIncludeExpansion: true, //是否支持扩展
expansion: { key: '这是 value' } //消息直接携带的扩展
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
}, false, null, null, null, config);
```

更新扩展

更新消息的扩展内容

API 参考：updateMessageExpansion

参数说明:

参数	类型	必填	默认值	说明
expansion	Object	是	---	扩展内容
message	Object	是	---	原消息体
callback	Object	是	---	回调对象
callback.onSuccess	Function	是	---	成功回调
callback.onError	Function	是	---	失败回调

代码示例:

```
var expansion = {
key: '这是 value'
};
var message = {
conversationType: 1,
targetId: 'user1',
messageUId: 'BK96-PP18-OIQ6-9GPP',
canIncludeExpansion: true
// ....
};
RongIMClient.getInstance().updateMessageExpansion(expansion, message, {
onSuccess: function () {
console.log('设置成功')
},
onError: function (reason) {
console.log('设置失败', reason)
}
});
```

删除扩展

API 参考：removeMessageExpansionForKey

参数说明:

参数	类型	必填	默认值	说明
keys	Array	是	--	删除的 keys
message	Object	是	--	原消息体
callback	Object	是	--	回调对象
callback.onSuccess	Function	是	--	成功回调
callback.onError	Function	是	--	失败回调

代码示例:

```
var keys = ['key1', 'key2'];
var message = {
conversationType: 1,
targetId: 'user1',
messageUId: 'BK96-PP18-OIQ6-9GPP'
//...
};
RongIMClient.getInstance().removeMessageExpansionForKey(keys, message, {
onSuccess: function () {
console.log('删除成功')
},
onError: function (reason) {
console.log('删除失败')
}
});
```

群聊介绍 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。

功能	描述
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。

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

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
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

功能	群组（group）	聊天室（Chatroom）
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

显示用户信息 显示用户信息

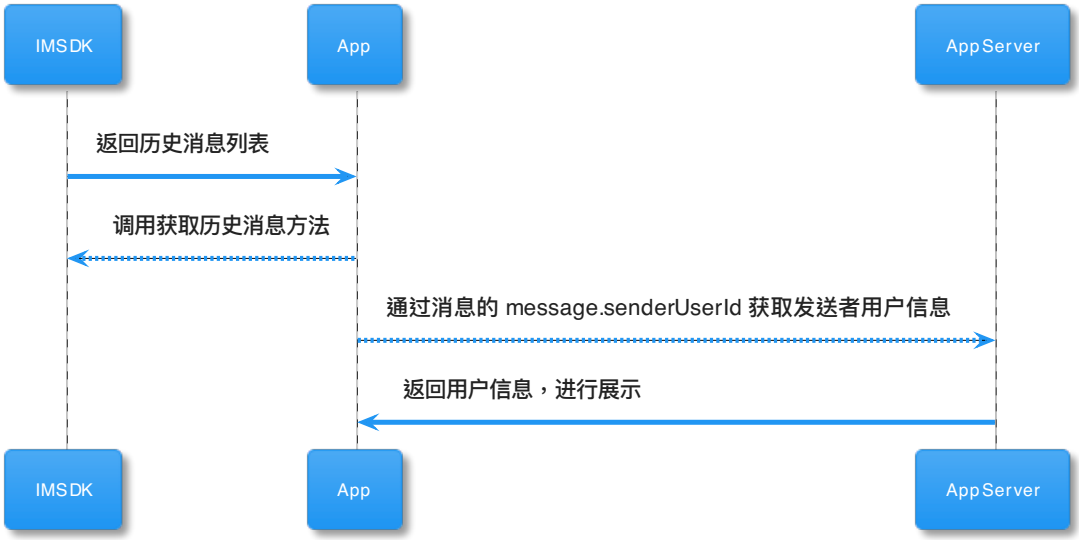
更新时间:2024-08-30

> 提示
>
> 单群聊中用户信息需要开发者自行处理。SDK 不处理用户信息。

历史消息显示用户信息

通过开发者 App Server 获取用户信息

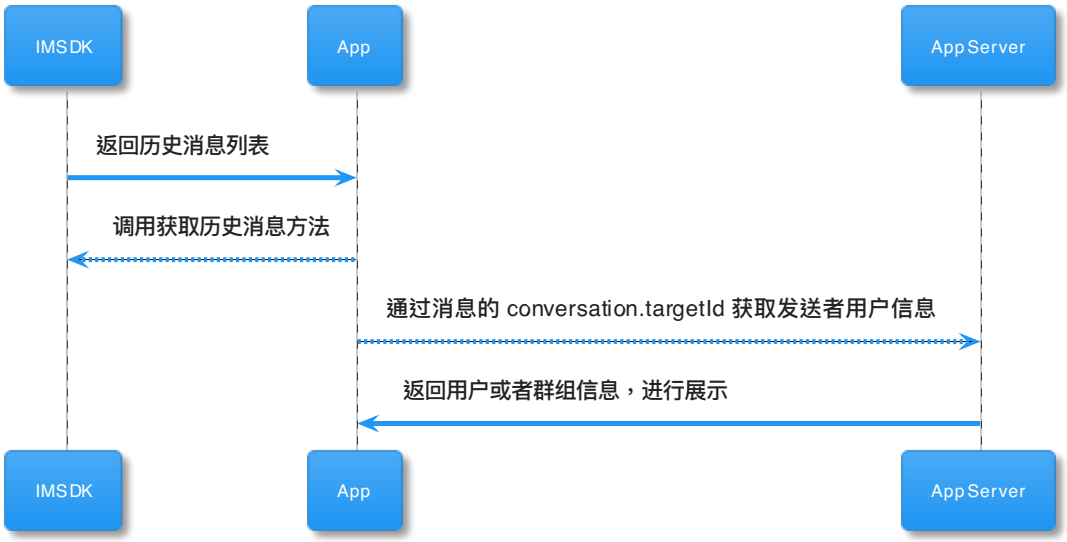
1. 开发者 App Server 封装获取用户信息接口
2. 通过 message.senderUserId 获取发送者 id
3. 将发送者 id 传入 App Server 暴露的接口中，获取对应用户信息
4. 将用户信息展示到页面中



会话列表显示用户信息

通过开发者 App Server 获取用户信息

1. 开发者 App Server 封装获取用户或群组信息接口
2. 通过 conversation.targetId 获取用户或者群组 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


IMSDK
App
AppServer
返回历史消息列表
调用获取历史消息方法
通过消息的 conversation.targetId 获取发送者用户信息
返回用户或者群组信息，进行展示
IMSDK
App
AppServer

连接服务 连接服务 功能描述

更新时间:2024-08-30

连接融云服务器之前，需要 App Server 通过融云 Server API 获取 Token，客户端获取到这个 Token 即可连接融云服务器。

⚠ 警告

1. 连接方法必须在执行初始化之后调用。详见初始化文档。
2. 连接方法必须在设置状态监听器和消息监听器之后调用。详见设置监听文档。
3. 除初始化、监听以外，所有方法都必须在 connect 成功之后 再调用
4. 默认一个用户只支持一个页面连接，开通 多设备消息同步 即可支持多页面连接

参数说明

参数	类型	必填	说明	最低版本
token	String	是	用户的唯一标识	2.0.0
callback	Object	是	重连回调对象	2.0.0
callBack.onSuccess	Function	是	连接成功回调，会返回 `token` 对应的 `userId`	2.0.0
callBack.onError	Function	是	连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.0.0
callBack.onTokenIncorrect	Function	是	`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接	2.0.0

代码示例

```
RongIMClient.connect('<Your-Token>', {
onSuccess: function(userId) {
console.log('连接成功, 用户 ID 为', userId);
// 连接已成功, 此时可通过 getConversationList 获取会话列表并展示
},
onTokenIncorrect: function() {
console.log('连接失败, 失败原因: token 无效');
},
onError: function(errorCode) {
console.log('连接失败, 失败原因: ', errorCode);
}
});
```

断开连接 断开连接 断开连接

更新时间:2024-08-30

断开当前用户的连接。调用后将不再接收消息，不可发送消息，不可获取历史消息，不可获取会话列表。

> 提示
>
> 在下次连接融云成功后，会收取上次离线后的消息，离线消息最多可以保存 7 天。

代码示例

```
RongIMClient.getInstance().disconnect();
```

退出登录

切换用户时使用，示例见: Web SDK API 示例

代码示例

```
RongIMClient.getInstance().logout();
```

重连逻辑 重连逻辑

更新时间:2024-08-30

功能描述

重新连接调用时机：

1. 网络不好，导致连接频繁断开后不会重新连接。
2. 长时间断网后，IM不会重新连接。
3. 交互连接频繁断开（网络环境：WIFI 或者联通、移动 3/4G ），交替切换网络后，IM不会重新连接。

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	重连回调对象	2.3.3
callBack.onSuccess	Function	是	即连接成功回调，会返回 `token` 对应的 `userId`	2.3.3
callBack.onError	Function	是	即连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.3.3
callBack.onTokenIncorrect	Function	是	即过 `token` 无效回调，是因为您在控制台设置了 `token` 过期时间，需要重新获取 `token` 并再次用新的 `token` 建立连接	2.3.3
config	Object	否	重连配置	2.3.3

`config` 参数说明

参数	类型	必填	说明	最低版本
auto	Boolean	否	是否自动重连, 默认 false	2.3.3
url	String	否	用于网络嗅探的地址, auto 为 true 时, 此参数必填	2.3.3
rate	Array	否	网络嗅探频率, 单位为毫秒, auto 为 true 时, 此参数必填	2.3.3

⚠ 警告

1. 不传 config 参数, 则为直接重连, 此时 reconnect 方法 必须在网络正常的情况下调用
2. 传入 config 参数, SDK 内部自动做网络嗅探处理, 当检测到网络正常时, 进行重连, 按照传入嗅探频率嗅探 10 次后会在 onError 中返回重连失败，如还需要重连请在 onError 中继续调用重连方法。

代码示例

```
var callback = {
onSuccess: function(userId) {
console.log('reconnect success. ' + userId);
},
onTokenIncorrect: function() {
console.log('token 无效');
},
onError: function(errorCode) {
console.log(errorCode);
}
};
var config = {
auto: true,
url: 'cdn.ronghub.com/RongIMLib-2.2.6.min.js?d=' + Date.now(),
rate: [100, 1000, 3000, 6000, 10000]
};
RongIMClient.reconnect(callback, config);
```

多端同时在线 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。

获取全部会话 获取全部会话 获取本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供获取本地会话列表接口。

获取远端会话列表

会话列表生成规则 : 服务端会根据消息来生成初始会话列表, 在收到消息和发送消息之后服务端会更新会话列表。

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。

API 参考：getConversationList

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	是	获取的会话类型, 获取所有会话类型传 null	2.3.3
count	Number	否	获取会话数量	2.3.3

`conversationTypes` 枚举值说明

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

回调参数说明

`onSuccess` 说明：

回调参数	类型	说明
list	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
conversationType	Number	会话类型
targetId	String	群组 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识, 融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息, 消息结构详见消息数据结构
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间
isTop	Boolean	会话置顶状态
notificationStatus	Number	会话免打扰状态

代码示例

```
var count = 150;
var callback = {
onSuccess: function(list) {
console.log('获取会话列表成功', list);
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, count);
```

⚠ 警告

返回的会话列表 list 长度是在传递的 count 基础上进一步筛选 conversationTypes 的结果，数量会小于等于 count。

常见问题

Q1: 切换用户后获取会话列表，会获取到不属于当前用户的会话

A1: 解决方案

在退出之前先清除掉缓存在调用 logout

RongIMClient.getInstance().clearCache();

```
RongIMClient.getInstance().logout();
```

Q2: 会话列表的名字和头像怎么维护

A2: 解决方案

1、用户信息在您的服务器维护，例如：用户 Id、头像、名称等

2、通过会话列表中的 targetId、senderUserId 在您应用服务器获取用户信息

3、使用用户信息与会话列表通过 targetId、senderUserId 做为对应关系将数据合并

4、将列表渲染至页面

说明：

targetId: 接收方用户或群组 ID

senderUserId: 消息发送人 ID

删除全部会话 删除全部会话 清除本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供清除本地会话列表接口。

清除远端会话列表

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	清除的会话类型, 不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

获取指定会话　　获取指定会话 功能描述

更新时间:2024-08-30

此方法获取的为本地缓存数据，必须在调用 getConversationList 之后再调用。

API 参考：getConversation

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，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

回调参数说明

返回值	返回类型	说明
conversation	Object	返回获取的会话信息

`conversation` 属性说明

字段名	类型	说明
conversationType	Number	会话类型
targetId	String	群组 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识, 融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
RongIMClient.getInstance().getConversation(conversationType, targetId, {
onSuccess: function(conversation) {
if (conversation) {
console.log('获取指定会话成功', conversation);
}
},
onError: function (error) {
console.log('获取指定会话失败:', error)
},
});
```

删除指定会话 删除指定会话 删除指定会话

更新时间:2024-08-30

API 参考：removeConversation

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = GROUP;
var targetId = '群组 ID';
var callback = {
onSuccess: function() {
console.log('删除指定会话成功');
},
onError: function(error) {
console.log('删除指定会话失败', error);
}
};
RongIMClient.getInstance().removeConversation(conversationType, targetId, callback);
```

按会话类型删除

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

参数	类型	必填	说明	最低版本
conversationTypes	Array	否	清除的会话类型，不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

会话草稿信息 会话草稿信息 获取草稿

更新时间:2024-08-30

获取内存中存储的草稿信息。

API 参考：getTextMessageDraft

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，群组会话传入 GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0

回调参数说明

回调参数	类型	说明
draft	String	返回草稿信息

代码示例

```
var conversationType = GROUP;
var targetId = '群组 ID';
var draft = RongIMClient.getInstance().getTextMessageDraft(conversationType, targetId);
console.log('草稿信息为: ', draft);
```

保存草稿

草稿存储在内存中，如刷新或者关闭页面会导致草稿丢失。

API 参考：saveTextMessageDraft

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，群组会话传入 GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
draftText	String	是	草稿信息	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	保存草稿接口操作状态

代码示例

```
var conversationType = GROUP;
var targetId = '群组 ID';
var draftText = '草稿信息';
var bool = RongIMClient.getInstance().saveTextMessageDraft(conversationType, targetId, draftText);
```

删除草稿

API 参考：clearTextMessageDraft

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，群组会话传入 GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	删除草稿接口操作状态

代码示例

```
var conversationType = GROUP;
var targetId = '群组 ID';
var bool = RongIMClient.getInstance().clearTextMessageDraft(conversationType, targetId);
```

常见问题

Q1: Web 端设置了草稿，获取会话列表没有取到草稿？

A1: 草稿为本地存储的，如果您需要显示需要您按照您的逻辑做下 UI 渲染。

会话未读数 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

> ⚠ 警告
>
> 1. 清除浏览器缓存会导致会话未读数不准确
> 2. 会话未读数为本地操作，换端登录会话未读数不会同步
> 3. 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

API 参考：getTotalUnreadCount

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	会话类型	2.9.5
includeMuted	Boolean	否	是否包含免打扰会话，默认为：false	2.9.5

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.GROUP];
var callback = {
onSuccess: function(count) {
console.log('获取所有会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取所有会话未读消息数失败', error);
}
};
RongIMClient.getInstance().getTotalUnreadCount(callback, conversationTypes);
```

获取单个会话未读数

方式一：调用 getUnreadCount 方法来获取会话未读数。

方式二：通过 conversation.unreadMessageCount 获取。

API 参考：getUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
var callback = {
onSuccess: function(count) {
console.log('获取指定会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话未读消息数失败', error);
}
};
RongIMLib.RongIMClient.getInstance().getUnreadCount(conversationType, targetId, callback);
```

按会话类型获取未读数

API 参考：getConversationUnreadCount

参数说明

参数	类型	必填	说明	最低版本	废弃版本
conversationTypes	Array	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0	2.6.0
callback	Object	是	回调对象	2.2.0	2.6.0
callback.onSuccess	Function	是	成功回调	2.2.0	2.6.0
callback.onError	Function	是	失败回调	2.2.0	2.6.0

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.GROUP];
var callback = {
onSuccess: function(count) {
console.log('获取指定会话类型总未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话类型总未读消息数失败', error);
}
};
RongIMClient.getInstance().getConversationUnreadCount(conversationTypes, callback);
```

清除单个会话未读数

API 参考：clearUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
var callback = {
onSuccess: function() {
console.log('清除指定会话未读消息数成功');
},
onError: function(error) {
console.log('清除指定会话未读消息数失败', error);
}
};
RongIMClient.getInstance().clearUnreadCount(conversationType, targetId, callback);
```

清除全部会话未读数

> 最低版本：2.10.4

```
RongIMClient.getInstance().clearAllUnreadCount()
```

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

本端清除会话未读数后，通过在会话中发送一条同步消息，通知其他端。其他端收取到会话中的同步消息后，调用相应方法清除本地会话未读消息数。

1. 本端阅读会话中的消息后，调用 clearUnreadCount() 清除会话未读消息数。
2. 清除成功后，在会话中发送 SyncReadStatusMessage 类型消息，同步阅读状态。
3. 其他端接收到 SyncReadStatusMessage 类型消息后，调用 clearUnreadCount() 方法，清除本地的会话未读数。（在 v2.10.4 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
>
> 群消息未读数同步必须发送定向消息，参数为 `config.userIds`

清除端

```
// 清除未读数
var im = RongIMClient.getInstance();
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
im.clearUnreadCount(conversationType, targetId, {
onSuccess: function () {
// 从消息里获取服务器端时间，以最近一条已读 message 为准
var msg = {
lastMessageSendTime: message.sentTime
};
var isMentioned = false; // @ 消息
var pushContent = null; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填
var config = {
// 群定向消息，仅发送给群中的自己，群内其他用户无需接收。 currentUserId 为当前登录的用户 ID
userIds: ['currentUserId']
}
msg = new RongIMLib.SyncReadStatusMessage(msg);
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function () {
console.log('发送同步消息成功', message);
},
onError: function () {}
}, isMentioned, pushContent, pushData, null, config);
},
onError: function (errorCode) {}
});
```

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 clearUnreadCount() 成功后不需要再在发送 SyncReadStatusMessage 类型消息。
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
var clearUnreadCount = RongIMClient.getInstance().clearUnreadCount;
clearUnreadCount(conversationType, targetId, {
onSuccess: function () {},
onError: function (errorCode) {}
});
```

会话免打扰 会话免打扰 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.5.8
targetId	String	是	群组 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '接收方的 userId';
var statusItem = {
isTop: true
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

会话置顶 会话置顶 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.5.8
targetId	String	是	群组 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '接收方的 userId';
var statusItem = {
isTop: true
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

会话标签 会话标签

更新时间:2024-08-30

> SDK 2.X 版本从 2.8.0 开始支持会话标签功能。

会话标签用于对会话进行分组。用户可设置会话标签，用于会话的分组显示。

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。

管理标签

SDK 支持创建标签、编辑标签，获取标签列表，以及移除标签。SDK 创建的标签可用于对会话进行管理。

SDK 创建的标签会被同步到服务端。

> 关于如何使用标签管理会话，详见使用标签管理会话。

创建标签

每个用户最多可以创建 20 个标签。

API 参考：createTag

参数说明

参数	类型	必填	说明
tag	Object	是	标签信息
tag.tagId	String	是	标签 Id，标签唯一标识，长度不超过 10 个字
tag.tagName	String	是	标签名称，长度不超过 15 个字
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('创建标签成功')
},
onError: function(error) {
console.log('创建标签失败', error)
}
}
RongIMClient.getInstance().createTag(tag,callback)
```

移除标签

API 参考：removeTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tagId = '<标签Id>'
var callback = {
onSuccess: function() {
console.log('删除标签成功')
},
onError: function(error) {
console.log('删除标签失败', error)
}
}
RongIMClient.getInstance().removeTag(tagId,callback)
```

编辑标签

参数说明

参数	类型	必填	说明
tag	Object	是	标签信息
tag.tagId	String	是	标签 Id，标签唯一标识，长度不超过 10 个字
tag.tagName	String	是	标签名称，长度不超过 15 个字
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调

参数	类型	必填	说明
callback.onError	Function	是	失败回调

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('编辑标签成功')
},
onError: function(error) {
console.log('编辑标签失败', error)
}
}
RongIMClient.getInstance().updateTag(tag,callback)
```

获取标签列表

API 参考：getTagList

参数说明

参数	类型	必填	说明
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

成功回调

成功回调 `ITagInfo[]` 的参数说明：

参数	返回类型	说明
tagInfo	ITagInfo[]	标签列表信息，详见下方 `ITagInfo` 属性说明

- `ITagInfo` 属性说明

参数	类型	说明
tagId	String	标签 Id
tagName	String	标签名称
conversationCount	Number	标签下的会话数量
createdTime	Number	标签创建的时间戳

代码示例

```
var callback = {
onSuccess: function(tagInfo) {
console.log(tagInfo)
console.log('获取标签列表成功')
},
onError: function(error) {
console.log('获取标签列表失败', error)
}
}
RongIMClient.getInstance().getTagList(callback)
```

用户标签多端同步

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。

代码示例

```
/**
* 设置会话标签更改时的回调
* 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。
* 收到此回调，可调用 getTagList 获取最新标签信息
* 请在初始化之后，连接之前调用该方法
*/
var tagListener = {
onChanged:function(){
return
}
}
RongIMClient.setTagListener(tagListener)
```

使用标签管理会话

SDK 支持使用一个或多个标签标记会话，可用于会话的分组显示、获取会话列表等操作，并提供了多个接口用于移除会话上的标签。

会话标签还支持一系列高级功能，包括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，以及在特定标签下设置某个会话置顶。

添加会话到一个标签

每个标签下最多可以添加 1000 个会话。

API 参考：addTagForConversations

参数说明

参数	类型	必填	说明
tagId	String	是	标签信息
conversations	IConversationOption []	是	要添加的会话列表。详见下方 `IConversationOption` 属性说明。
callback	Object	是	回调对象

参数	类型	必填	说明
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- IConversationOption 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tag = {
tagId: '<标签Id>',
tagName: '<新的标签名字>'
}
var callback = {
onSuccess: function() {
console.log('添加成功')
},
onError: function(error) {
console.log('添加失败', error)
}
}
var conversations =[{
type:1,//会话类型
targetId:'<目标Id>'
}
]
RongIMClient.getInstance().addTagForConversations(tagId,conversations,callback)
```

删除指定标签中某些会话

removeTagForConversations 接口可从特定标签下移除指定多个会话，也可以理解为“为多个会话移除单个标签”。

API 参考：removeTagForConversations

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
conversations	IConversationOption[]	是	要从指定标签下删除的会话列表。详见下方 IConversationOption 属性说明。
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- IConversationOption 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagId = '<标签Id>'
var callback = {
onSuccess: function() {
console.log('删除成功')
},
onError: function(error) {
console.log('删除失败', error)
}
}
var conversations =[{
type:1,//会话类型
targetId:'<目标Id>'
}]
RongIMClient.getInstance().removeTagForConversations(tagId,conversations,callback)
```

删除指定会话中的某些标签

`removeTagsForConversation` 接口可为特定会话移除指定的一个或多个标签。

API 参考：removeTagsForConversation

参数说明

参数	类型	必填	说明
conversation	IConversationOption	是	会话。详见下方 `IConversationOption` 属性说明。
tagIds	Array	是	所有标签
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明。

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagIds = ['<标签Id>']
var callback = {
onSuccess: function() {
console.log('删除成功')
},
onError: function(error) {
console.log('删除失败', error)
}
}
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
RongIMClient.getInstance().removeTagsForConversation(conversation,tagIds,callback)
```

获取指定会话下的所有标签

API 参考：getTagsForConversation

参数说明

参数	类型	必填	说明
conversation	IConversationOption	是	会话。详见下方 IConversationOption 属性说明。
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- IConversationOption 属性说明。

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

成功回调

成功回调 IConversationTag 的参数说明：

参数	返回类型	说明
tagId	String	标签 Id
tagName	String	标签名称
createdTime	Number	标签创建的时间戳
isTop	Boolean	是否置顶

代码示例

```
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
var callback = {
onSuccess: function(IConversationTag) {
console.log('获取成功',IConversationTag)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getTagsForConversation(conversation,callback)
```

分页获取指定标签下的会话列表

getConversationListByTag 可从服务端分页获取会话列表，并返回分页查询结果中带有指定标签的会话。

API 参考：getConversationListByTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
count	Number	是	单次分页查询时需要从服务端获取的会话数量
startTime	Number	是	会话中最后一条消息时间戳
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

代码示例

```
var tagId = '<标签Id>'
var count = 10
var startTime = 1649404018538
var callback = {
onSuccess: function(IV2ConversationContainTag) {
console.log('获取成功后会话列表',IV2ConversationContainTag)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getConversationListByTag(tagId,count,startTime,callback)
```

按标签获取未读消息数

API 参考：getUnreadCountByTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
containMuted	Number	是	是否包含已设置为免打扰状态的会话
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

成功回调

成功回调参数说明：

参数	返回类型	说明
count	Number	获取未读消息数

代码示例

```
var tagId = '<标签Id>'
var containMuted = true
var callback = {
onSuccess: function(count) {
console.log('成功获取未读消息数',count)
},
onError: function(error) {
console.log('获取失败', error)
}
}
RongIMClient.getInstance().getUnreadCountByTag(tagId,containMuted,callback)
```

设置标签中会话置顶

`setConversationStatusInTag` 可设置会话在标签下为置顶状态。

设置成功后，可在分页获取指定标签下会话列表（`getConversationListByTag`）返回的结果中查看相关属性。

- **注意，`setConversationStatusInTag` 接口仅用于在特定标签下会话是否置顶。**
- **如需设置会话在会话列表中置顶，请使用 `setConversationStatus` 接口（SDK ≧ 2.5.8）。**

API 参考：setConversationStatusInTag

参数说明

参数	类型	必填	说明
tagId	String	是	标签 Id
conversation	IConversationOption	是	会话。详见下方 `IConversationOption` 属性说明。
status	Object	是	置顶状态

参数	类型	必填	说明
status.isTop	Boolean	是	是否置顶
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

- `IConversationOption` 属性说明

参数	类型	必填	说明
type	Number	是	会话类型
targetId	String	是	目标 ID

代码示例

```
var tagId = '<标签Id>'
var conversation ={
type:1,//会话类型
targetId:'<目标Id>'
}
let status = {
isTop:true
}
var callback = {
onSuccess: function() {
console.log('设置成功')
},
onError: function(error) {
console.log('设置失败', error)
}
}
RongIMClient.getInstance().setConversationStatusInTag(tagId,conversation,status,callback)
```

用户会话标签多端同步

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。

代码示例

```
var tagListener = {
onChanged:function(){
return
}
}
RongIMClient.setConversationTagListener(tagListener)
```

消息发送　　消息发送 功能描述

更新时间:2024-08-30

消息属性	消息描述	消息属性	消息描述
消息类名	各端消息名	ObjectName	传输层名称，与消息类名一一对应
存储属性	存储 / 不存储	计数属性	计数 / 不计数
离线属性	缓存 / 不缓存	消息尺寸	128 KB
推送属性	是/否	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，需开通历史消息云存储服务，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。

发送普通消息

参数需按顺序传入，顺序为参数说明中的字段顺序。

API 参考：sendMessage

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

`config` 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0

参数	类型	必填	说明	最低版本
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 `isStatusMessage` 替代该参数。在 `isStatusMessage` 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP; // 群聊，其他会话选择相应的消息类型即可
var targetId = '群组 ID'; // 目标 Id

var isMentioned = true; // @ 消息
var mentioneds = new RongIMLib.MentionedInfo(); // @ 消息对象
mentioneds.type = RongIMLib.MentionedType.PART;
mentioneds.userIdList = [ 'user1', 'user2' ]; // @ 人员列表

var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息',mentionedInfo:
mentioneds });
var callBack = {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
};

var pushContent = 'user 发送了一条消息'; // Push 显示内容
var methodType = null; // 已经逐步废弃，填写 null 即可
var pushData = null; // Push 通知时附加信息，可不填

var config = {
userIds: [ 'user1', 'user2' ], // 群定向消息，仅发消息给群中的 user1 和 user2
isVoipPush: true, // 发送 voip push
disableNotification: true // 发送静默消息， 设置为 true 后移动端不会收到 Push 信息和本地通知提醒
};

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, methodType, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP
targetId	String	群组 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	Number	本地生成的消息 id
messageUId	String	服务端存储的消息 id

字段名	类型	说明
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, PC 端有效，Web 端无效
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
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

文本消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TextMessage	RC:TxtMsg	存储	计数	存储	推送	消息内容

TextMessage 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 textMessageInfo = {
content: 'hello RongCloud!',
extra: '附加信息'
}
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
});
```

图文消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RichContentMessage	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 msg = new RongIMLib.RichContentMessage({
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'
});
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送富文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送富文本消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

代码示例

```
var textMessageInfo = { content: '' }
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送 Emoji 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 Emoji 消息失败', errorCode);
}
});
```

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

```
<!-- HTTP -->
<script src="http://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- HTTPS -->
<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-v2-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

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
LocationMessage	RC:LBSMsg	存储	计数	存储	推送	[位置]

LocationMessage 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 locatMessageInfo = {
latitude: 40.0317727,
longitude: 116.4175057,
poi: '融云',
content: '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE', // 位置图片 base64
}
var msg = new RongIMLib.LocationMessage(locatMessageInfo);

var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送位置消息成功', message);
},
onError: function (errorCode) {
console.log('发送位置消息失败', errorCode);
}
});
```

正在输入状态消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TypingStatusMessage	RC:TypSts	不存储	不计数	不存储	不推送	无

TypingStatusMessage 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 typingContentType = 'RC:TxtMsg';
var msg = new RongIMLib.TypingStatusMessage({ typingContentType: typingContentType});

var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 Id
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送已读通知成功', message);
},
onError: function (errorCode) {
console.log('发送已读通知失败', errorCode);
}
});
```

发送 @ 消息

⚠ 警告

1. 支持 @ 的会话类型: RongIMLib.ConversationType.GROUP
2. 支持 @ 的消息类型: TextMessage、ImageMessage、VoiceMessage、RichContentMessage、LocationMessage

API 参考：sendMessage

参数说明

属性名称	属性类型	是否必填	属性说明
type	Number	是	@ 的类型，分为全部和部分两类，通过 RongIMLib.MentionedType 获取枚举值
userIdList	Array	是	@ 的用户列表

代码示例

```
var isMentioned = true; // @ 消息
var mentioneds = new RongIMLib.MentionedInfo(); // @ 消息对象
// 全部：RongIMLib.MentionedType.ALL, 部分：RongIMLib.MentionedType.PART
mentioneds.type = RongIMLib.MentionedType.PART;
mentioneds.userIdList = [ 'user1', 'user2' ]; // @ 人员列表

var msg = new RongIMLib.TextMessage({
content: 'hello RongCloud!',
mentionedInfo: mentioneds // 构建消息时传递实例化的 @ 消息对象。注意只有 支持 @ 的消息类型才生效
});
var conversationType = RongIMLib.ConversationType.GROUP; // 群聊, 其他会话选择相应的消息类型即可
var targetId = '群组 ID'; // 目标 Id

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
}, isMentioned);//isMentioned = true; 发送消息设置是否为 @ 消息为 true
```

发送定向消息

> ⚠ 警告
>
> 发送定向消息需要注意 sendMessage 方法参数 isMentioned，pushContent，pushData 都需要传递。

API 参考：sendMessage

参数说明

属性名称	属性类型	是否必填	属性说明
isVoipPush	Number	是	是否发送 voip push

属性名称	属性类型	是否必填	属性说明
userIds	Array	是	群定向消息，仅发消息给群中 userIds 数组内的用户

代码示例

```
var isMentioned = false; // @ 消息
var pushContent = null; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填

var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!'});
var conversationType = RongIMLib.ConversationType.GROUP; // 群聊，其他会话选择相应的消息类型即可
var targetId = '群组 ID'; // 目标 Id

var config = {
userIds: [ 'user1', 'user2' ], // 群定向消息，仅发消息给群中的 user1 和 user2
isVoipPush: true // 发送 voip push
};

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
}, isMentioned, pushContent, pushData, null, config);
```

发送媒体消息

API 参考：sendMessage

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

`config` 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 `isStatusMessage` 替代该参数。在 `isStatusMessage` 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

var base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});

var callBack = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};

var isMentioned = false; // @ 消息
var pushContent = 'user 发送了一条消息'; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填
var config = {
isVoipPush: true // 发送 voip push
};

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，群组会话传入 RongIMLib.ConversationType.GROUP
targetId	String	群组 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	Number	本地生成的消息 id

字段名	类型	说明
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, PC 端有效，Web 端无效
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
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

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，文件存储到优先级高的云存储，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。

API 参考：sendMessage

消息发送

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
ImageMessage	RC:ImgMsg	存储	计数	存储	推送	[图片]

ImageMessage 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 base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 Id
var callback = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

图片上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.IMAGE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

语音消息

> 提示
>
> - 语音上传请参照 文件上传 进行是实现。
> - HQVoiceMessage 消息支持版本： **v2.5.0** 及以上。

API 参考：sendMessage

消息说明

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

HQVoiceMessage 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 messageInfo = {
remoteUrl: 'http://rongcloud-file.ronghub.com/1e0e4743249cd9653b.aac',//此地址为模拟地址仅作为示例使用
duration: 22,
extra: '附加信息'
}
var msg = new RongIMLib.HQVoiceMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送语音消息成功', message);
},
onError: function (errorCode) {
console.log('发送语音消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

消息发送

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

FileMessage 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	文件尺寸，单位: Byte
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 msg = new RongIMLib.FileMessage({
name: 'RongIMLib.js', // 文件名,
size: 1024, // 文件大小单位 bytes
type: 'js', // 文件类型
fileUrl: 'https://cdn.ronghub.com/RongIMLib.js' // 文件地址
});
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID
var callback = {
onSuccess: function (message) {
console.log('发送文件消息成功', message);
},
onError: function (errorCode) {
console.log('发送文件消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

文件上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.FILE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

小视频消息

> ⚠ 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。
> 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 。**

API 参考：sendMessage

消息说明

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

SightMessage 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
和		
和		
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 params = {
content: "/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
sightUrl: "http://rongcloud...com/video...",
duration: 10,
size: 734320,
name: "video_xx.mp4",
}
var msg = new RongIMLib.SightMessage(params);
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送小视频消息成功', message);
},
onError: function (errorCode) {
console.log('发送小视频消息失败', errorCode);
}
});
```

GIF 消息

API 参考：sendMessage

消息说明

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

GIFMessage 参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 bytes
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 messageInfo ={
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
var msg = new RongIMLib.GIFMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.GROUP; //选择相应的会话类型即可
var targetId = '群组 ID';
var callback = {
onSuccess: function (message) {
console.log('发送 GIF 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 GIF 消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

发送自定义消息

注册

> ⚠ 警告
>
> 1. 注册自定义消息代码必须在发送、接收该自定义消息之前
> 2. 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
> 3. 注册的消息类型名，必须多端一致，否则消息无法互通
> 4. 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
messageName	String	消息名称
objectName	String	消息类型名
mesasgeTag	Object	存储，计数标识
searchProp	string[]	搜索字段，web 端无需设置，搜索字段值设置为数字时取值范围为 (-Math.pow(2, 64), Math.pow(2, 64)) 且为整数

代码示例

```
var messageName = 'PersonMessage'; // 消息名称
var objectName = 's:person'; // 消息类型名，请按照此格式命名
var isCounted = true; // 消息计数
var isPersited = true; // 消息保存
var mesasgeTag = new RongIMLib.MessageTag(isCounted, isPersited);
var searchProp = ['name', 'age']; // 搜索字段中的属性名
RongIMClient.registerMessageType(messageName, objectName, mesasgeTag, searchProp);
```

发送

API 参考：sendMessage

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
var msg = new RongIMClient.RegisterMessage.PersonMessage({ name: 'zhang', age: 12 });

var callback = {
onSuccess: function (message) {
console.log('发送自定义消息成功', message);
},
onError: function (errorCode) {
console.log('发送自定义消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

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)。具体方法请参见 Emoji 消息。

消息接收 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「设置监听」文档中的设置消息监听。

历史消息获取　　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。

API 参考：getHistoryMessages

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，群组会话传入 RongIMLib.ConversationType.GROUP	2.2.0
targetId	String	是	群组 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
objectName	String	否	消息类名,仅桌面版解决方案有用	2.3.4
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
hasMsg	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
conversationType	Number	会话类型
targetId	String	群组 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

字段名	类型	说明
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.ConversationType.GROUP;
var targetId = '群组 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);
}
});
```

消息回执 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

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
ReadReceiptRequestMessage	RC:RRReqMsg	不存储	不计数	不存储	不推送	无

API 参考：sendMessage

参数说明

属性名称	属性类型	是否必填	属性说明
messageUId	String	是	messageUId 为消息的唯一标识，通过 message.messageUId 可取到

代码示例

```
var msg = new RongIMLib.ReadReceiptRequestMessage({
// messageUId 为消息的唯一标识，通过 message.messageUId 可取到
messageUId: messageUId
});
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

// 将 msg 通过 sendMessage 接口发送即可
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送消息成功', message);
},
onError: function (errorCode) {
console.log('发送消息失败', errorCode);
}
});
```

响应回执请求

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
ReadReceiptResponseMessage	RC:RRRspMsg	不存储	不计数	不存储	不推送	无

API 参考：sendMessage

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 为对方用户的 userId，value 为回复的 messageUId 数组。可参考代码示例进行理解

代码示例

```
var msg = new RongIMLib.ReadReceiptResponseMessage({
receiptMessageDic: {
// userId01 为具体的用户 Id， messageUId 为 ReadReceiptRequestMessage 消息中的 messageUId
userId01: [messageUId]
}
});
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID'; // 目标 ID

// 将 msg 通过 sendMessage 接口发送即可
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送消息成功', message);
},
onError: function (errorCode) {
console.log('发送消息失败', errorCode);
}
});
```

⚠ 警告

1. 请求回执成功后可在 `message.receiptResponse` 属性查看回执情况。
2. 群回执存储在 localStorage，若需持久化存储可将群响应回执存储在应用服务器。

```
// 示例
// 例如，用户 A（userA）请求 messageId 为 JDKS-DKSL-FHDK-FSGH 的消息回执
// 用户 B（userB）和用户 C （userC）阅读了消息
var responses = {
'userA_JDKS-DKSL-FHDK-FSGH': 2
};
```

消息撤回 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。

API 参考：sendRecallMessage

参数说明

输入参数说明

属性名称	属性类型	是否必填	属性说明
recallMessage	Object	是	需要撤回的消息对象
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

回调参数说明

请参考 message 属性说明

代码示例

```
// recallMessage 为需要撤回的消息对象
RongIMClient.getInstance().sendRecallMessage(recallMessage, {
onSuccess: function (message) {
console.log('撤回成功', message);
},
onError: function (errorCode) {
console.log('撤回失败', errorCode);
}
});
```

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 RecallCommandMessage 消息，收到后按需处理即可。

详见「设置监听」文档中的设置消息监听。

消息转发 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 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

API 参考：deleteRemoteMessages

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传 RongIMLib.ConversationType.GROUP	2.5.3
targetId	String	是	群组 ID	2.5.3
messages	Object	是	要删除的消息数组, 不能超过 100 条	2.5.3
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

`messages` 参数说明

参数	类型	必填	说明	最低版本
messageUId	String	是	消息 uid	2.3.5
sentTime	Number	是	消息发送时间	2.3.5
messageDirection	Number	是	消息发送方向	2.3.5

代码示例

```
var conversationType = RongIMLib.ConversationType.GROUP;
var targetId = '群组 ID';
/*
message 对象可通过历史消息获取
messageUId、sentTime、messageDirection 必传，且必须正确
*/
var messages = [
{ messageUId: 'BETR-GIM7-TNO5-89QU', sentTime: 1575965764383, messageDirection: 1 },
{ messageUId: 'AEGR-CFN7-QFJO-8OC0', sentTime: 1575965744371, messageDirection: 2 }
];
RongIMLib.RongIMClient.getInstance().deleteRemoteMessages(conversationType, targetId, messages, {
onSuccess: function() {
console.log('清除成功');
},
onError: function(error) {
console.log('清除失败', error);
}
});
```

通过时间戳删除

API 参考：clearRemoteHistoryMessages

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，群组会话传 RongIMLib.ConversationType.GROUP	2.3.5
targetId	String	是	群组 ID	2.3.5
timestamp	Number	是	清除时间点，该时间之前的消息将被清除	2.3.5

代码示例

```
var params = {
conversationType: RongIMLib.ConversationType.GROUP,
targetId: '群组 ID',
timestamp: 1513308018122 // 可取 sentTime，收发消息和历史消息中都有 sentTime 字段
};
RongIMLib.RongIMClient.getInstance().clearRemoteHistoryMessages(params, {
onSuccess: function() {
console.log('清除成功');
},
onError: function(error) {
console.log('清除失败', error);
}
});
```

聊天室介绍 聊天室介绍 功能描述

更新时间:2024-08-30

聊天室成员不设用户上限，海量消息并发即时到达，用户退出聊天界面后即视为离开聊天室，不会再接收到任何聊天室中消息，没有推送通知功能。会话关系由融云负责建立并保持连接，通过 SDK 相关接口，可以让用户加入或者退出聊天室。

主要功能

功能	描述
离线消息	不支持离线消息，只有当前在线用户可收到聊天室中消息
人数限制	聊天室人数无上限
消息提醒	离线后不再接收聊天室中消息。
本地存储	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。
历史消息	提供服务端消息存储存功能，需开通聊天室消息云存储，默认存储时长为 2 个月。
创建聊天室	App 内的聊天室数量没有限制。
销毁聊天室	将指定聊天室解散，所有成员都无法再接收该聊天室的消息。
查询聊天室信息	查询聊天室基础信息，包括：聊天室 ID、名称、创建时间。
获取聊天室成员	获取的聊天室成员信息数，包括：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。
指定聊天室禁言	用户在指定聊天室中禁言，被禁言用户可以接收查看聊天室中用户聊天信息，但不能发送消息。
用户聊天室全局禁言	用户在应用中的所有聊天室中禁言，被禁言用户可以接收查看聊天室中用户聊天信息，但不能发送消息。
聊天室用户封禁	在 App 中如果想将某一用户踢出聊天室并在一段时间内不允许再进入聊天室时，可实现将用户对指定的聊天室做封禁处理，被封禁用户将被踢出聊天室，并在设定的时间内不能再进入聊天室中。
指定聊天室全局禁言	对指定聊天室做禁言处理，聊天室中所有用户都不能发送消息。
消息优化级	通过聊天室消息优先级接口，设置的消息类型为低优先级的消息，默认情况下全部为高的消息，当服务器负载高时低优先级的消息优先被丢弃，这样可以让出资源给高优先级的消息，确保重要的消息不被丢弃。
消息白名单	白名单中的消息类型受到保护，在聊天室消息量较大的情况下也不被丢弃。
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也不被丢弃。同时用户处于被保护状态，以避免在离线 30 秒后有新消息产生时或离线后聊天室中产生 30 条消息时被自动踢出聊天室。
聊天室保活	聊天室保活功能，可以确保聊天室在此状态下不被自动销毁，只能通过调用 API 接口销毁聊天室。
状态同步	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，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。
属性自定义	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，聊天室销毁后，聊天室中的自定义属性同时销毁。

功能	描述
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。

消息类型

功能	描述
离线消息	不支持离线消息，只有当前在线用户可收到聊天室中消息
人数限制	聊天室人数无上限
消息提醒	离线后不再接收聊天室中消息。
本地存储	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。
历史消息	提供服务端消息存储存功能，需开通聊天室消息云存储，默认存储时长为 2 个月。
创建聊天室	App 内的聊天室数量没有限制。
销毁聊天室	将指定聊天室解散，所有成员都无法再接收该聊天室的消息。
查询聊天室信息	查询聊天室基础信息，包括：聊天室 ID、名称、创建时间。
获取聊天室成员	获取的聊天室成员信息数，包括：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。
指定聊天室禁言	用户在指定聊天室中禁言，被禁言用户可以接收查看聊天室中用户聊天信息，但不能发送消息。
用户聊天室全局禁言	用户在应用中的所有聊天室中禁言，被禁言用户可以接收查看聊天室中用户聊天信息，但不能发送消息。
聊天室用户封禁	在 App 中如果想将某一用户踢出聊天室并在一段时间内不允许再进入聊天室时，可实现将用户对指定的聊天室做封禁处理，被封禁用户将被踢出聊天室，并在设定的时间内不能再进入聊天室中。
指定聊天室全局禁言	对指定聊天室做禁言处理，聊天室中所有用户都不能发送消息。
消息优化级	通过聊天室消息优先级接口，设置的消息类型为低优先级的消息，默认情况下全部为高的消息，当服务器负载高时低优先级的消息优先被丢弃，这样可以让出资源给高优先级的消息，确保重要的消息不被丢弃。
消息白名单	白名单中的消息类型受到保护，在聊天室消息量较大的情况下也不被丢弃。
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也不被丢弃。同时用户处于被保护状态，以避免在离线 30 秒后有新消息产生时或离线后聊天室中产生 30 条消息时被自动踢出聊天室。
聊天室保活	聊天室保活功能，可以确保聊天室在此状态下不被自动销毁，只能通过调用 API 接口销毁聊天室。
状态同步	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，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。
属性自定义	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，聊天室销毁后，聊天室中的自定义属性同时销毁。
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。

与群组的区别

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）

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
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

常见问题

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。

是否可以实现匿名登录

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。

断线情况如何处理

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

进入时如何获取最新消息

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。

点赞、礼物消息问题

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。

聊天室用户信息显示

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。详细请查看用户信息显示集成指南。

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。

聊天室管理功能

聊天室踢人功能实现

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。

设置聊天室禁言用户

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。

聊天室成员自动退出机制

聊天室中用户在离线 30 秒后有新消息产生时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。

聊天室用户白名单设置方法

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。

聊天室自动销毁机制

聊天室中 1 小时无人说话时，同时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。

聊天室不被自动销毁实现方式

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。

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

消息的分级与抛弃模型

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

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。

注：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。

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

多端情况下，一端退出聊天室其他终端是否同步退出聊天室

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。

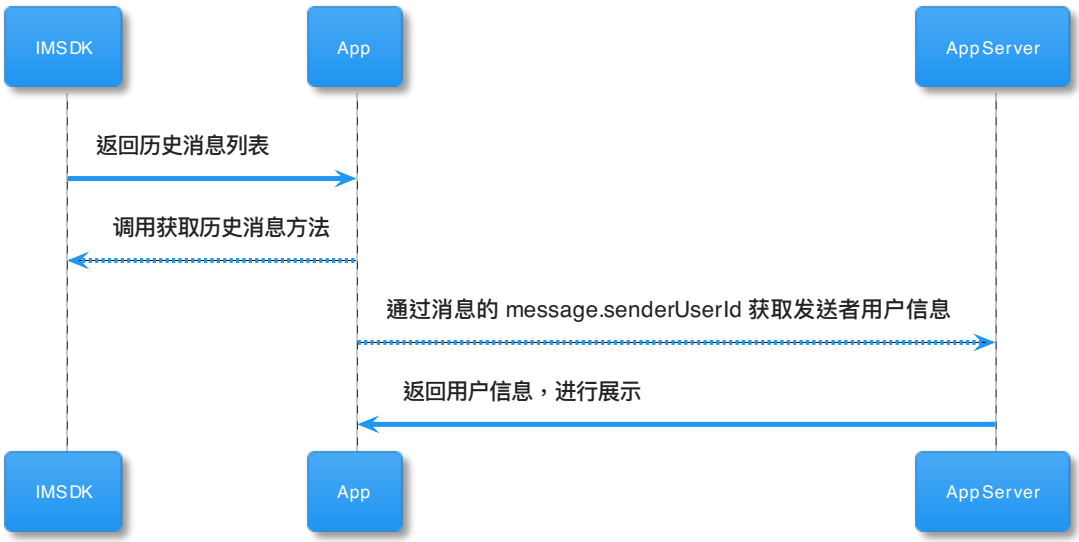
显示用户信息　　显示用户信息 历史消息显示用户信息

更新时间:2024-08-30

您可以使用下面任意任意一种方案实现用户信息的获取与显示。

通过开发者 App Server 获取用户信息

1. 开发者 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，获取对应用户信息。
4. 将用户信息展示到页面中。



通过发消息携带用户信息

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

> ⚠ 警告
>
> 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。

代码示例

```
var msg = new RongIMLib.TextMessage({
content: 'hello RongCloud!',
user : { // 当前用户(发送者) 信息
"id" : "user1",
"name" : "张三",
"portrait" : "https://cdn.ronghub.com/thinking-face.png"
},
});
var conversationType = RongIMLib.ConversationType.PRIVATE;
var targetId = 'user2'; // 目标 Id

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送消息成功, 用户信息为: ', message.content.user);
},
onError: function (errorCode) {
console.log('发送消息失败', errorCode);
}
});
```

连接服务　　连接服务 功能描述

更新时间:2024-08-30

连接融云服务器之前，需要 App Server 通过融云 Server API 获取 Token，客户端获取到这个 Token 即可连接融云服务器。

> ⚠ 警告
>
> 1. 连接方法必须在执行初始化之后调用。详见初始化文档。
> 2. 连接方法必须在设置状态监听器和消息监听器之后调用。详见设置监听文档。
> 3. 除初始化、监听以外，所有方法都必须在 connect 成功之后 再调用
> 4. 默认一个用户只支持一个页面连接，开通 多设备消息同步 即可支持多页面连接

参数说明

参数	类型	必填	说明	最低版本
token	String	是	用户的唯一标识	2.0.0
callback	Object	是	重连回调对象	2.0.0
callBack.onSuccess	Function	是	连接成功回调，会返回 `token` 对应的 `userId`	2.0.0
callBack.onError	Function	是	连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.0.0
callBack.onTokenIncorrect	Function	是	`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接	2.0.0

代码示例

```
RongIMClient.connect('<Your-Token>', {
onSuccess: function(userId) {
console.log('连接成功, 用户 ID 为', userId);
// 连接已成功, 此时可通过 getConversationList 获取会话列表并展示
},
onTokenIncorrect: function() {
console.log('连接失败, 失败原因: token 无效');
},
onError: function(errorCode) {
console.log('连接失败, 失败原因: ', errorCode);
}
});
```

断开连接 断开连接 断开连接

更新时间:2024-08-30

断开当前用户的连接。调用后将不再接收消息，不可发送消息，不可获取历史消息，不可获取会话列表。

> 提示
>
> 在下次连接融云成功后，会收取上次离线后的消息，离线消息最多可以保存 7 天。

代码示例

```
RongIMClient.getInstance().disconnect();
```

退出登录

切换用户时使用，示例见: Web SDK API 示例

代码示例

```
RongIMClient.getInstance().logout();
```

重连逻辑 重连逻辑

更新时间:2024-08-30

功能描述

重新连接调用时机：

1. 网络不好，导致连接频繁断开后不会重新连接。
2. 长时间断网后，IM不会重新连接。
3. 交互连接频繁断开（网络环境：WIFI 或者联通、移动 3/4G ），交替切换网络后，IM不会重新连接。

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	重连回调对象	2.3.3
callBack.onSuccess	String	是	即连接成功回调，会返回 `token` 对应的 `userId`	2.3.3
callBack.onError	String	是	即连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.3.3
callBack.onTokenIncorrect	String	是	即过 `token` 无效回调，是因为您在控制台设置了 `token` 过期时间，需要重新获取 `token` 并再次用新的 `token` 建立连接	2.3.3
config	Object	否	重连配置	2.3.3

`config` 参数说明

参数	类型	必填	说明	最低版本
auto	Boolean	否	是否自动重连, 默认 false	2.3.3
url	String	否	用于网络嗅探的地址, auto 为 true 时, 此参数必填	2.3.3
rate	Array	否	网络嗅探频率, 单位为毫秒, auto 为 true 时, 此参数必填	2.3.3

⚠ 警告

1. 不传 config 参数, 则为直接重连, 此时 reconnect 方法 必须在网络正常的情况下调用
2. 传入 config 参数, SDK 内部自动做网络嗅探处理, 当检测到网络正常时, 进行重连, 按照传入嗅探频率嗅探 10 次后会在 onError 中返回重连失败，如还需要重连请在 onError 中继续调用重连方法。

代码示例

```
var callback = {
onSuccess: function(userId) {
console.log('reconnect success. ' + userId);
},
onTokenIncorrect: function() {
console.log('token 无效');
},
onError: function(errorCode) {
console.log(errorCode);
}
};
var config = {
auto: true,
url: 'cdn.ronghub.com/RongIMLib-2.2.6.min.js?d=' + Date.now(),
rate: [100, 1000, 3000, 6000, 10000]
};
RongIMClient.reconnect(callback, config);
```

多端同时在线 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。

加入　　加入 功能描述

更新时间:2024-08-30

加入聊天室，需加入 已存在 的聊天室。若聊天室不存在，则加入失败。

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。

API 参考：joinChatRoom

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 id	2.2.0
count	Number	是	拉取聊天数, 范围 0 - 50	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var chatRoomId = 'chatroom1';
var count = 10;
var callback = {
onSuccess: function() {
console.log('加入聊天室成功');
},
onError: function(error) {
console.log('加入聊天室失败', error);
}
};
RongIMClient.getInstance().joinChatRoom(chatRoomId, count, callback);
```

退出 退出

更新时间:2024-08-30

API 参考：quitChatRoom

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 id	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var chatRoomId = "chatroom1"; // 聊天室 Id
var callback = {
onSuccess: function() {
console.log('退出聊天室成功');
},
onError: function(error) {
console.log('退出聊天室失败');
}
};
RongIMClient.getInstance().quitChatRoom(chatRoomId, callback);
```

查询聊天室信息 查询聊天室信息

更新时间:2024-08-30

API 参考：getChatRoomInfo

获取成员信息

获取成员信息

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 id	2.2.0
count	Number	是	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**	2.2.0
order	Number	是	排序方式, 1 正序, 2 倒序，可通过 `RongIMLib.GetChatRoomType` 枚举获取	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

回调参数说明

参数	类型	说明	最低版本
chatRoom	Object	聊天室信息	2.2.0
chatRoom.userInfos	Array	聊天室成员信息	2.2.0
chatRoom.userTotalNums	Number	聊天室总人数	2.2.0

代码示例

```
var chatRoomId = 'chatroom1';
var count = 10;
var order = RongIMLib.GetChatRoomType.REVERSE;
var callback = {
onSuccess: function(chatRoom) {
/*
chatRoom.userInfos 聊天室成员信息
chatRoom.userTotalNums 聊天室总人数
*/
console.log('获取聊天室信息成功', chatRoom);
},
onError: function(error) {
console.log('获取聊天室信息失败', error);
}
};
RongIMClient.getInstance().getChatRoomInfo(chatRoomId, count, order, callback);
```

设置属性 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

API 参考：setChatroomEntry

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3
chatroomEntry	Object	是	设置的属性信息	2.5.3

`chatroomEntry` 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	2.5.3

参数	类型	必填	说明	最低版本
value	Object	是	属性对应的值，最大长度 4096 字符	2.5.3
isSendNotification	Boolean	否	设置成功后是否发送通知消息	2.5.3
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	2.5.3
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
var key = 'role';
var value = 'Werewolf';
var chatroomEntry = {
key: key,
value: value,
isAutoDelete: false,
isSendNotification: true,
notificationExtra: 'Change role'
};
RongIMClient.getInstance().setChatroomEntry(chatRoomId, chatroomEntry, {
onSuccess: function() {
console.log('设置聊天室属性成功');
},
onError: function(error) {
// 请检查：是否开通聊天室属性自定义服务
console.log('设置聊天室属性失败', error);
}
});
```

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

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.10.0
options	Object	是	设置的属性信息	2.10.0

`options` 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	2.10.0
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	2.10.0

代码示例

```
var chatRoomId = 'chatroom1';
var options = {
entries: {
name: 'name'
}
isAutoDelete: true
}
RongIMClient.getInstance().setChatRoomEntries(chatRoomId, chatroomEntry, {
onSuccess: function() {
console.log('批量设置聊天室属性成功');
},
onError: function(error) {
// 请检查: 是否开通聊天室属性自定义服务
console.log('批量设置聊天室属性失败', error);
}
});
```

强制设置属性

强制修改/创建任意聊天室属性

API 参考：forceSetChatroomEntry

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3
chatroomEntry	Object	是	设置的属性信息	2.5.3

chatroomEntry 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	2.5.3
value	Object	是	属性对应的值, 最大长度 4096 字符	2.5.3
isSendNotification	Boolean	否	设置成功后是否发送通知消息.	2.5.3
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	2.5.3
isAutoDelete	Boolean	否	用户退出聊天室时是否清除此属性	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
var key = 'count';
var value = '8';
var chatroomEntry = {
key: key,
value: value,
isAutoDelete: false,
isSendNotification: true,
notificationExtra: 'Change Count'
};
RongIMClient.getInstance().forceSetChatroomEntry(chatRoomId, chatroomEntry, {
onSuccess: function() {
console.log('强制设置聊天室属性成功');
},
onError: function(error) {
// 请检查：是否开通聊天室属性自定义服务
console.log('强制设置聊天室属性失败', error);
}
});
```

删除属性 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

API 参考：removeChatroomEntry

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3
chatroomEntry	Object	是	设置的属性信息	2.5.3

chatroomEntry 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	2.5.3
isSendNotification	Boolean	否	删除成功后是否发送通知消息	2.5.3
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
var key = 'role';
var chatroomEntry = {
key: key,
isSendNotification: true,
notificationExtra: 'Change role'
};
RongIMClient.getInstance().removeChatroomEntry(chatRoomId, chatroomEntry, {
onSuccess: function() {
console.log('删除聊天室属性成功');
},
onError: function(error) {
// 请检查: 是否开通聊天室属性自定义服务
console.log('删除聊天室属性失败', error);
}
});
```

强制删除属性

强制删除任意聊天室属性

API 参考：forceRemoveChatroomEntry

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3
chatroomEntry	Object	是	设置的属性信息	2.5.3

`chatroomEntry` 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	2.5.3
isSendNotification	Boolean	否	删除成功后是否发送通知消息	2.5.3
notificationExtra	String	否	RC:chrmKVNotiMsg 消息中携带的附加信息	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
var key = 'role';
var chatroomEntry = {
key: key,
isSendNotification: true,
notificationExtra: 'Change role'
};
RongIMClient.getInstance().forceRemoveChatroomEntry(chatRoomId, chatroomEntry, {
onSuccess: function() {
console.log('强制删除聊天室属性成功');
},
onError: function(error) {
// 请检查: 是否开通聊天室属性自定义服务
console.log('强制删除聊天室属性失败', error);
}
});
```

获取属性 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

API 参考：getChatroomEntry

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3
key	String	是	聊天室属性名	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
var key = 'role';
RongIMClient.getInstance().getChatroomEntry(chatRoomId, key, {
onSuccess: function(value) {
console.log('获取聊天室属性信息成功', value);
},
onError: function(error) {
console.log('获取聊天室属性信息失败', error);
}
});
```

获取所有属性

API 参考：getAllChatroomEntries

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.5.3

代码示例

```
var chatRoomId = 'chatroom1';
RongIMClient.getInstance().getAllChatroomEntries(chatRoomId, {
onSuccess: function(entries) {
// entries 为聊天室内存储的所有属性信息
console.log('获取所有聊天室属性信息成功', entries);

},
onError: function(error) {
console.log('获取所有聊天室属性信息失败', error);
}
});
```

消息发送　　消息发送 功能描述

更新时间:2024-08-30

消息属性	消息描述	消息属性	消息描述
消息类名	各端消息名	ObjectName	传输层名称，与消息类名一一对应
存储属性	存储 / 不存储	计数属性	计数 / 不计数
离线属性	缓存 / 不缓存	消息尺寸	128 KB
推送属性	是/否	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，需开通历史消息云存储服务，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。

发送普通消息

参数需按顺序传入，顺序为参数说明中的字段顺序。

API 参考：sendMessage

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，聊天室会话传入 RongIMLib.ConversationType.CHATROOM	2.2.0
targetId	String	是	聊天室 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

`config` 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9

参数	类型	必填	说明	最低版本
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 `isStatusMessage` 替代该参数。在 `isStatusMessage` 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.CHATROOM; // 群聊，其他会话选择相应的消息类型即可
var targetId = '聊天室 ID'; // 目标 Id
var msg = new RongIMLib.TextMessage({ content: 'hello RongCloud!', extra: '附加信息'});
var callBack = {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
};
var isMentioned = false; // @ 消息
var pushContent = 'user 发送了一条消息'; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填
var config = {
isVoipPush: true // 发送 voip push
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, pushData, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，聊天室会话传入 RongIMLib.ConversationType.CHATROOM
targetId	String	聊天室 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	Number	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, PC 端有效，Web 端无效
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

字段名	类型	说明
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

文本消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TextMessage	RC:TxtMsg	存储	计数	存储	推送	消息内容

TextMessage 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 textMessageInfo = {
content: 'hello RongCloud!',
extra: '附加信息'
}
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送文本消息失败', errorCode);
}
});
```

图文消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
RichContentMessage	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

属性名称	属性类型	是否必填	属性说明
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 msg = new RongIMLib.RichContentMessage({
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'
});
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID';

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送富文本消息成功', message);
},
onError: function (errorCode) {
console.log('发送富文本消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

代码示例

```
var textMessageInfo = { content: '□' }
var msg = new RongIMLib.TextMessage(textMessageInfo);
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 id 和发送消息时间戳
console.log('发送 Emoji 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 Emoji 消息失败', errorCode);
}
});
```

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

```
<!-- HTTP -->
<script src="http://cdn.ronghub.com/RongEmoji-2.2.10.js"></script>
<!-- HTTPS -->
<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-v2-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

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
LocationMessage	RC:LBSMsg	存储	计数	存储	推送	[位置]

LocationMessage 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 locatMessageInfo = {
latitude: 40.0317727,
longitude: 116.4175057,
poi: '融云',
content: '/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDABsSFBcUERsXFhceHBsgKE', // 位置图片 base64
}
var msg = new RongIMLib.LocationMessage(locatMessageInfo);

var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送位置消息成功', message);
},
onError: function (errorCode) {
console.log('发送位置消息失败', errorCode);
}
});
```

正在输入状态消息

API 参考：sendMessage

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
TypingStatusMessage	RC:TypSts	不存储	不计数	不存储	不推送	无

TypingStatusMessage 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 typingContentType = 'RC:TxtMsg';
var msg = new RongIMLib.TypingStatusMessage({ typingContentType: typingContentType});

var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 Id
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
console.log('发送已读通知成功', message);
},
onError: function (errorCode) {
console.log('发送已读通知失败', errorCode);
}
});
```

发送媒体消息

API 参考：sendMessage

sendMessage 参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，聊天室会话传入 RongIMLib.ConversationType.CHATROOM	2.2.0
targetId	String	是	聊天室 ID	2.2.0
msg	Object	是	消息	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
isMentioned	Boolean	否	是否为 @ 消息	2.2.0
pushContent	String	否	Push 显示内容	2.2.0

参数	类型	必填	说明	最低版本
pushData	String	否	Push 通知时附加信息	2.2.0
~~methodType~~	Number	否	该参数已废弃	2.6.0
config	Object	否	其他设置项	2.5.3

config 说明：

参数	类型	必填	说明	最低版本
userIds	Array	否	接收定向消息的用户 id	2.2.0
isVoipPush	Boolean	否	为 true 时，对端不在线的 iOS 会收到 Voip Push，Android 无影响	2.5.3
disableNotification	Boolean	否	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	2.5.9
canIncludeExpansion	Boolean	否	是否支持扩展	2.6.0
expansion	Object	否	扩展内容	2.6.0
isStatusMessage	Boolean	否	是否为状态消息	2.6.0
~~isStatus~~	Boolean	否	该参数已废弃，请使用 isStatusMessage 替代该参数。在 isStatusMessage 有值的情况下，该参数将失效	2.6.0

代码示例

```
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID

var base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});

var callBack = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};

var isMentioned = false; // @ 消息
var pushContent = 'user 发送了一条消息'; // Push 显示内容
var pushData = null; // Push 通知时附加信息，可不填
var config = {
isVoipPush: true // 发送 voip push
};

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callBack, isMentioned,
pushContent, pushData, config);
```

message 属性说明

字段名	类型	说明
conversationType	Number	会话类型，聊天室会话传入 RongIMLib.ConversationType.CHATROOM
targetId	String	聊天室 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	Number	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, PC 端有效，Web 端无效
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
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

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，文件存储到优先级高的云存储，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。

API 参考：sendMessage

消息发送

消息说明

消息类名	ObjectName	存储属性	计数属性	离线属性	推送属性	推送内容
ImageMessage	RC:ImgMsg	存储	计数	存储	推送	[图片]

ImageMessage 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 base64Str = '/9j/4AAQSBAAD/2wBDDBAYEBAQEB....'; // 压缩后的 base64 略缩图，用来快速展示图片
var 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
var msg = new RongIMLib.ImageMessage({content: base64Str, imageUri: imageUri});
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 Id
var callback = {
onSuccess: function (message) {
console.log('发送图片消息成功', message);
},
onError: function (errorCode) {
console.log('发送图片消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

图片上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.IMAGE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

语音消息

> 提示
>
> - 语音上传请参照 文件上传 进行是实现。
> - HQVoiceMessage 消息支持版本： **v2.5.0** 及以上。

API 参考：sendMessage

消息说明

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

HQVoiceMessage 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 messageInfo = {
remoteUrl: 'http://rongcloud-file.ronghub.com/1e0e4743249cd9653b.aac',//此地址为模拟地址仅作为示例使用
duration: 22,
extra: '附加信息'
}
var msg = new RongIMLib.HQVoiceMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID

RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送语音消息成功', message);
},
onError: function (errorCode) {
console.log('发送语音消息失败', errorCode);
}
});
```

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。

API 参考：sendMessage

消息发送

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

FileMessage 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	文件尺寸，单位: Byte
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 msg = new RongIMLib.FileMessage({
name: 'RongIMLib.js', // 文件名,
size: 1024, // 文件大小单位 bytes
type: 'js', // 文件类型
fileUrl: 'https://cdn.ronghub.com/RongIMLib.js' // 文件地址
});
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID
var callback = {
onSuccess: function (message) {
console.log('发送文件消息成功', message);
},
onError: function (errorCode) {
console.log('发送文件消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

文件上传

兼容性

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

> 上传代码示例：https://github.com/rongcloud/rongcloud-web-im-upload

引入

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

上传 token

> 必须 IM SDK 连接成功后，才能使用此方法

API 参考：getFileToken

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型，通过 RongIMLib.FileType 获取
callbacks	Object	是	回调对象
fileName	String	否	原文件名

代码示例：

```
var fileType = RongIMLib.FileType.FILE;
RongIMClient.getInstance().getFileToken(fileType, {
onSuccess: function(data) {
console.log('上传 token 为', data.token);
},
onError: function(error) {
console.log('get file token error', error);
}
}, fileName)
```

开始上传

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
getToken	Function	是	获取 token 回调

代码示例：

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

```
var file;
var fileType = RongIMLib.FileType.IMAGE;
var uploadEl = document.getElementById("uploadFile");

var getFileType = function(filename) {
// 默认支持两种图片格式，可自行扩展
var imageType = {
'jpg': 1,
'png': 2,
}
var index = filename.lastIndexOf('.') + 1,
type = filename.substring(index);
return type in imageType ? 'image': 'file';
};

var config = {
domain: '',
fileType,
getToken: function(callback) {
var caback={
onSuccess: function(data){
callback(data.token);
},
onError: function(){
console.error('get file token error', error);
}
}
RongIMClient.getInstance().getFileToken(fileType, caback, fileName);
}
};
```

```
var callback = {
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 initType = {
file: function(_file){
config.fileType = RongIMLib.FileType.FILE;
UploadClient.initFile(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
},
image: function(_file){
UploadClient.initImage(config, function(uploadFile){
uploadFile.upload(_file, callback);
});
}
};


uploadEl.onchange = function() {
// 根据文件名选择不同的初始化方式
file = this.files[0]; // 上传的 file 对象;
initType[getFileType(file.name)](file);
}
```

下载 url

> fileType 值需与 getFileToken 时传入的 fileType 值一致

API 参考：getFileUrl

参数说明：

参数	类型	必填	说明
fileType	Number	是	上传类型, 通过 RongIMLib.FileType 获取
filename	String	是	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	是	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
callbacks	Object	是	回调对象
data	Object	否	uploadCallbacks.onCompleted 回调返回数据
uploadMethod	String	否	服务器类型，上传成功后可通过 uploadCallbacks 的 onCompleted 中返回的 data 获取，对应属性data.uploadMethod

代码示例：

```
// data 通过 uploadFile.upload 获取
var fileType ; // 文件类型
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;
RongIMClient.getInstance().getFileUrl(fileType, filename, oriname, {
onSuccess: function(data) {
console.log('文件 url 为: ', data.downloadUrl);
},
onError: function(error) {
console.log('get file url error', error);
}
}, data,uploadMethod)
```

小视频消息

> ⚠ 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。
> 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**。

API 参考：sendMessage

消息说明

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

SightMessage 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
和		
和		
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 params = {
content: "/9j/4AAQSkZ/2wB...hDSaSiimB//9k=",
sightUrl: "http://rongcloud...com/video...",
duration: 10,
size: 734320,
name: "video_xx.mp4",
}
var msg = new RongIMLib.SightMessage(params);
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID'; // 目标 ID
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, {
onSuccess: function (message) {
// message 为发送的消息对象并且包含服务器返回的消息唯一 ID 和发送消息时间戳
console.log('发送小视频消息成功', message);
},
onError: function (errorCode) {
console.log('发送小视频消息失败', errorCode);
}
});
```

GIF 消息

API 参考：sendMessage

消息说明

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

GIFMessage 参数说明

参数	类型	说明
gifDataSize	Number	GIF 图片文件大小，单位为 bytes
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 messageInfo ={
gifDataSize:34563,
height:246,
remoteUrl:"https://rongcloud-image.cn.ronghub.com/image_jpe64562665566.gif",
width:263,
}
var msg = new RongIMLib.GIFMessage(messageInfo);
var conversationType = RongIMLib.ConversationType.CHATROOM; //选择相应的会话类型即可
var targetId = '聊天室 ID';
var callback = {
onSuccess: function (message) {
console.log('发送 GIF 消息成功', message);
},
onError: function (errorCode) {
console.log('发送 GIF 消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

发送自定义消息

注册

> ⚠ 警告
>
> 1. 注册自定义消息代码必须在发送、接收该自定义消息之前
> 2. 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
> 3. 注册的消息类型名，必须多端一致，否则消息无法互通
> 4. 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
messageName	String	消息名称
objectName	String	消息类型名
mesasgeTag	Object	存储，计数标识
searchProp	string[]	搜索字段，web 端无需设置，搜索字段值设置为数字时取值范围为 (-Math.pow(2, 64), Math.pow(2, 64)) 且为整数

代码示例

```
var messageName = 'PersonMessage'; // 消息名称
var objectName = 's:person'; // 消息类型名，请按照此格式命名
var isCounted = true; // 消息计数
var isPersited = true; // 消息保存
var mesasgeTag = new RongIMLib.MessageTag(isCounted, isPersited);
var searchProp = ['name', 'age']; // 搜索字段中的属性名
RongIMClient.registerMessageType(messageName, objectName, mesasgeTag, searchProp);
```

发送

API 参考：sendMessage

代码示例

```
var conversationType = RongIMLib.ConversationType.CHATROOM;
var targetId = '聊天室 ID';
var msg = new RongIMClient.RegisterMessage.PersonMessage({ name: 'zhang', age: 12 });

var callback = {
onSuccess: function (message) {
console.log('发送自定义消息成功', message);
},
onError: function (errorCode) {
console.log('发送自定义消息失败', errorCode);
}
};
RongIMClient.getInstance().sendMessage(conversationType, targetId, msg, callback);
```

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)。具体方法请参见 Emoji 消息。

消息接收 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「设置监听」文档中的设置消息监听。

消息撤回 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。

API 参考：sendRecallMessage

参数说明

输入参数说明

属性名称	属性类型	是否必填	属性说明
recallMessage	Object	是	需要撤回的消息对象
callback	Object	是	回调对象
callback.onSuccess	Function	是	成功回调
callback.onError	Function	是	失败回调

回调参数说明

请参考 message 属性说明

代码示例

```
// recallMessage 为需要撤回的消息对象
RongIMClient.getInstance().sendRecallMessage(recallMessage, {
onSuccess: function (message) {
console.log('撤回成功', message);
},
onError: function (errorCode) {
console.log('撤回失败', errorCode);
}
});
```

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 RecallCommandMessage 消息，收到后按需处理即可。

详见「设置监听」文档中的设置消息监听。

历史消息获取　　历史消息获取 功能描述

更新时间:2024-08-30

> ⚠ 警告
>
> 该功能需开通 聊天室消息云存储服务 后才能使用

API 参考：getChatRoomHistoryMessages

参数说明

参数	类型	必填	说明	最低版本
chatRoomId	String	是	聊天室 ID	2.2.0
count	Number	是	获取消息数, 范围 0 - 20	2.2.0
order	Number	是	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	2.2.0

代码示例

```
var chatRoomId = 'chatroom1';
var count = 10;
var order = 0;
RongIMClient.getInstance().getChatRoomHistoryMessages(chatRoomId, count, order, {
onSuccess: function(list, hasMore) {
console.log('获取聊天室历史消息成功', list, hasMore);
},
onError: function(error) {
// 请检查: 是否开通聊天室消息云存储服务
console.log('获取聊天室历史消息失败', error);
}
});
```

显示用户信息 显示用户信息

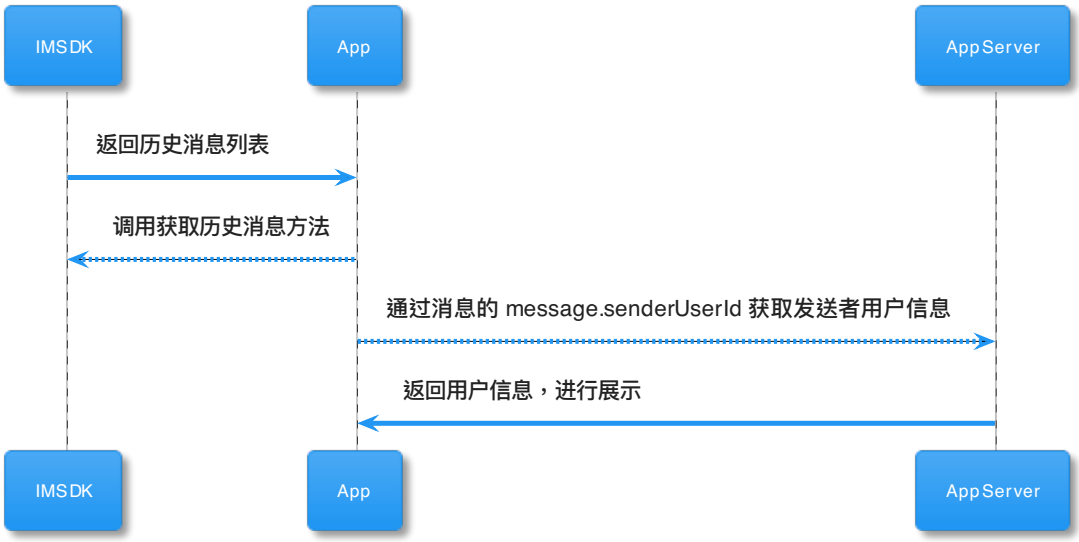
更新时间:2024-08-30

> 提示
>
> 单群聊中用户信息需要开发者自行处理。SDK 不处理用户信息。

历史消息显示用户信息

通过开发者 App Server 获取用户信息

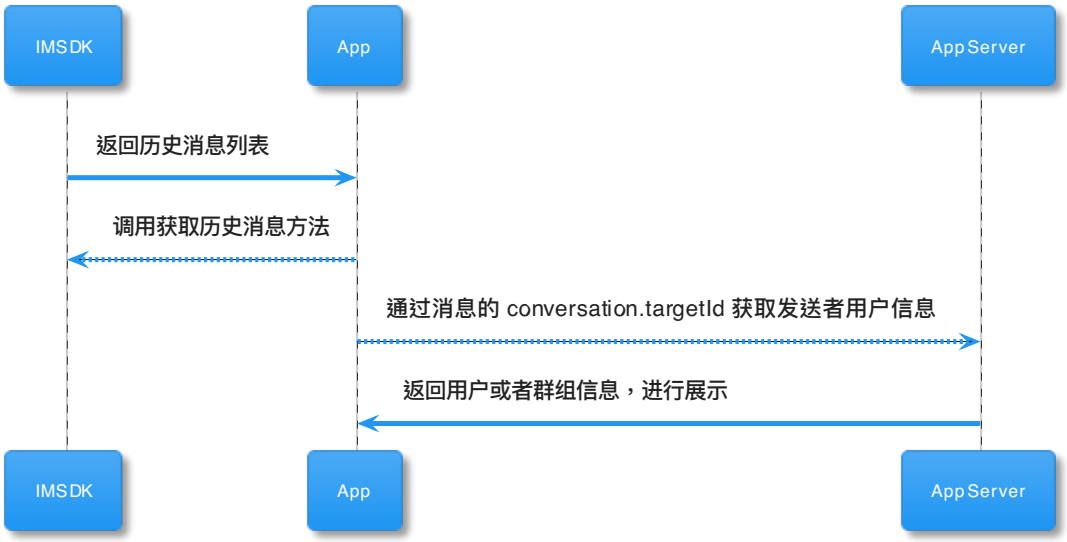
1. 开发者 App Server 封装获取用户信息接口
2. 通过 message.senderUserId 获取发送者 id
3. 将发送者 id 传入 App Server 暴露的接口中，获取对应用户信息
4. 将用户信息展示到页面中



会话列表显示用户信息

通过开发者 App Server 获取用户信息

1. 开发者 App Server 封装获取用户或群组信息接口
2. 通过 conversation.targetId 获取用户或者群组 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


IMSDK
App
AppServer
返回历史消息列表
调用获取历史消息方法
通过消息的 conversation.targetId 获取发送者用户信息
返回用户或者群组信息，进行展示
IMSDK
App
AppServer

连接服务 连接服务 功能描述

更新时间:2024-08-30

连接融云服务器之前，需要 App Server 通过融云 Server API 获取 Token，客户端获取到这个 Token 即可连接融云服务器。

⚠ 警告

1. 连接方法必须在执行初始化之后调用。详见初始化文档。
2. 连接方法必须在设置状态监听器和消息监听器之后调用。详见设置监听文档。
3. 除初始化、监听以外，所有方法都必须在 connect 成功之后 再调用
4. 默认一个用户只支持一个页面连接，开通 多设备消息同步 即可支持多页面连接

参数说明

参数	类型	必填	说明	最低版本
token	String	是	用户的唯一标识	2.0.0
callback	Object	是	重连回调对象	2.0.0
callBack.onSuccess	Function	是	连接成功回调，会返回 `token` 对应的 `userId`	2.0.0
callBack.onError	Function	是	连接失败回调，请您检查客户端初始化使用的 `AppKey` 和获取 `token` 用的 `AppKey` 是否一致	2.0.0
callBack.onTokenIncorrect	Function	是	`token` 无效回调，建议排查 控制台 是否设置了 Token 有效期，或重新获取 Token 再建立连接	2.0.0

代码示例

```
RongIMClient.connect('<Your-Token>', {
onSuccess: function(userId) {
console.log('连接成功，用户 ID 为', userId);
// 连接已成功，此时可通过 getConversationList 获取会话列表并展示
},
onTokenIncorrect: function() {
console.log('连接失败，失败原因：token 无效');
},
onError: function(errorCode) {
console.log('连接失败，失败原因：', errorCode);
}
});
```

断开连接 断开连接 断开连接

更新时间:2024-08-30

断开当前用户的连接。调用后将不再接收消息，不可发送消息，不可获取历史消息，不可获取会话列表。

> 提示
>
> 在下次连接融云成功后，会收取上次离线后的消息，离线消息最多可以保存 7 天。

代码示例

```
RongIMClient.getInstance().disconnect();
```

退出登录

切换用户时使用，示例见: Web SDK API 示例

代码示例

```
RongIMClient.getInstance().logout();
```

多端同时在线 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。

获取全部会话 获取全部会话 获取本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供获取本地会话列表接口。

获取远端会话列表

会话列表生成规则 : 服务端会根据消息来生成初始会话列表, 在收到消息和发送消息之后服务端会更新会话列表。

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。

API 参考：getConversationList

参数说明

输入参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	是	获取的会话类型, 获取所有会话类型传 null	2.3.3
count	Number	否	获取会话数量	2.3.3

`conversationTypes` 枚举值说明

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

回调参数说明

`onSuccess` 说明：

回调参数	类型	说明
list	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
conversationType	Number	会话类型
targetId	String	系统会话 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识，融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息，消息结构详见消息数据结构
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间
isTop	Boolean	会话置顶状态
notificationStatus	Number	会话免打扰状态

代码示例

```
var count = 150;
var callback = {
onSuccess: function(list) {
console.log('获取会话列表成功', list);
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, count);
```

⚠ 警告

返回的会话列表 list 长度是在传递的 count 基础上进一步筛选 conversationTypes 的结果，数量会小于等于 count。

常见问题

Q1: 切换用户后获取会话列表，会获取到不属于当前用户的会话

A1: 解决方案

在退出之前先清除掉缓存在调用 logout

RongIMClient.getInstance().clearCache();

RongIMClient.getInstance().logout();

Q2: 会话列表的名字和头像怎么维护

A2: 解决方案

1、用户信息在您的服务器维护，例如：用户 Id、头像、名称等

2、通过会话列表中的 targetId、senderUserId 在您应用服务器获取用户信息

3、使用用户信息与会话列表通过 targetId、senderUserId 做为对应关系将数据合并

4、将列表渲染至页面

说明：

targetId: 接收方用户或群组 ID

senderUserId: 消息发送人 ID

删除全部会话　　删除全部会话 清除本地会话列表

更新时间:2024-08-30

Web 没有本地数据库，不提供清除本地会话列表接口。

清除远端会话列表

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	清除的会话类型, 不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

获取指定会话　　获取指定会话 功能描述

更新时间:2024-08-30

此方法获取的为本地缓存数据，必须在调用 getConversationList 之后再调用。

API 参考：getConversation

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，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

回调参数说明

返回值	返回类型	说明
conversation	Object	返回获取的会话信息

`conversation` 属性说明

字段名	类型	说明
conversationType	Number	会话类型
targetId	String	系统会话 ID
latestMessageId	String	会话中最后一条消息 ID
objectName	String	会话中最后一条消息的消息标识, 融云内置消息以 "RC:" 开头
unreadMessageCount	Number	当前会话的未读消息数
latestMessage	Object	会话中最后一条消息
sentStatus	Number	会话中最后一条消息发送状态
sentTime	Number	会话中最后一条消息融云服务端的发送时间

代码示例

```
var conversationType = RongIMLib.ConversationType.SYSTEM;
var targetId = '系统会话 ID';
RongIMClient.getInstance().getConversation(conversationType, targetId, {
onSuccess: function(conversation) {
if (conversation) {
console.log('获取指定会话成功', conversation);
}
},
onError: function (error) {
console.log('获取指定会话失败:', error)
},
});
```

删除指定会话 删除指定会话 删除指定会话

更新时间:2024-08-30

API 参考：removeConversation

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统会话传入 SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = SYSTEM;
var targetId = '系统会话 ID';
var callback = {
onSuccess: function() {
console.log('删除指定会话成功');
},
onError: function(error) {
console.log('删除指定会话失败', error);
}
};
RongIMClient.getInstance().removeConversation(conversationType, targetId, callback);
```

按会话类型删除

> 提示
>
> conversationTypes 为 Array 类型。可以按类型进行删除，可传递多个，不填则清除所有会话。

API 参考：clearConversations

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.3.2
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

参数	类型	必填	说明	最低版本
conversationTypes	Array	否	清除的会话类型，不填则清除所有会话	2.3.2

conversationTypes 枚举值说明

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

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.PRIVATE, RongIMLib.ConversationType.GROUP];
RongIMClient.getInstance().clearConversations({
onSuccess: function(bool) {
console.log('清除会话成功',bool);
},
onError: function(error) {
console.log('清除会话失败', error);
}
}, conversationTypes);
```

会话草稿信息 会话草稿信息 获取草稿

更新时间:2024-08-30

获取内存中存储的草稿信息。

API 参考：getTextMessageDraft

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，系统会话传入 SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0

回调参数说明

回调参数	类型	说明
draft	String	返回草稿信息

代码示例

```
var conversationType = SYSTEM;
var targetId = '系统会话 ID';
var draft = RongIMClient.getInstance().getTextMessageDraft(conversationType, targetId);
console.log('草稿信息为: ', draft);
```

保存草稿

草稿存储在内存中，如刷新或者关闭页面会导致草稿丢失。

API 参考：saveTextMessageDraft

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，系统会话传入 SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
draftText	String	是	草稿信息	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	保存草稿接口操作状态

代码示例

```
var conversationType = SYSTEM;
var targetId = '系统会话 ID';
var draftText = '草稿信息';
var bool = RongIMClient.getInstance().saveTextMessageDraft(conversationType, targetId, draftText);
```

删除草稿

API 参考：clearTextMessageDraft

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，系统会话传入 SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0

回调参数说明

回调参数	类型	说明
bool	Boolean	删除草稿接口操作状态

代码示例

```
var conversationType = SYSTEM;
var targetId = '系统会话 ID';
var bool = RongIMClient.getInstance().clearTextMessageDraft(conversationType, targetId);
```

常见问题

Q1: Web 端设置了草稿，获取会话列表没有取到草稿？

A1: 草稿为本地存储的，如果您需要显示需要您按照您的逻辑做下 UI 渲染。

会话未读数　　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

> ⚠ 警告
>
> 1. 清除浏览器缓存会导致会话未读数不准确
> 2. 会话未读数为本地操作，换端登录会话未读数不会同步
> 3. 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

API 参考：getTotalUnreadCount

参数说明

参数	类型	必填	说明	最低版本
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0
conversationTypes	Array	否	会话类型	2.9.5
includeMuted	Boolean	否	是否包含免打扰会话，默认为：false	2.9.5

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.SYSTEM];
var callback = {
onSuccess: function(count) {
console.log('获取所有会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取所有会话未读消息数失败', error);
}
};
RongIMClient.getInstance().getTotalUnreadCount(callback, conversationTypes);
```

获取单个会话未读数

方式一：调用 getUnreadCount 方法来获取会话未读数。

方式二：通过 conversation.unreadMessageCount 获取。

API 参考：getUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.SYSTEM;
var targetId = '系统会话 ID';
var callback = {
onSuccess: function(count) {
console.log('获取指定会话未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话未读消息数失败', error);
}
};
RongIMLib.RongIMClient.getInstance().getUnreadCount(conversationType, targetId, callback);
```

按会话类型获取未读数

API 参考：getConversationUnreadCount

参数说明

参数	类型	必填	说明	最低版本	废弃版本
conversationTypes	Array	是	会话类型，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.2.0	2.6.0
callback	Object	是	回调对象	2.2.0	2.6.0
callback.onSuccess	Function	是	成功回调	2.2.0	2.6.0
callback.onError	Function	是	失败回调	2.2.0	2.6.0

代码示例

```
var conversationTypes = [RongIMLib.ConversationType.SYSTEM];
var callback = {
onSuccess: function(count) {
console.log('获取指定会话类型总未读消息数成功', count);
},
onError: function(error) {
console.log('获取指定会话类型总未读消息数失败', error);
}
};
RongIMClient.getInstance().getConversationUnreadCount(conversationTypes, callback);
```

清除单个会话未读数

API 参考：clearUnreadCount

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.2.0
targetId	String	是	系统会话 ID	2.2.0
callback	Object	是	回调对象	2.2.0
callback.onSuccess	Function	是	成功回调	2.2.0
callback.onError	Function	是	失败回调	2.2.0

代码示例

```
var conversationType = RongIMLib.ConversationType.SYSTEM;
var targetId = '系统会话 ID';
var callback = {
onSuccess: function() {
console.log('清除指定会话未读消息数成功');
},
onError: function(error) {
console.log('清除指定会话未读消息数失败', error);
}
};
RongIMClient.getInstance().clearUnreadCount(conversationType, targetId, callback);
```

清除全部会话未读数

> 最低版本：2.10.4

```
RongIMClient.getInstance().clearAllUnreadCount()
```

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

本端清除会话未读数后，通过在会话中发送一条同步消息，通知其他端。其他端收取到会话中的同步消息后，调用相应方法清除本地会话未读消息数。

1. 本端阅读会话中的消息后，调用 clearUnreadCount() 清除会话未读消息数。
2. 清除成功后，在会话中发送 SyncReadStatusMessage 类型消息，同步阅读状态。
3. 其他端接收到 SyncReadStatusMessage 类型消息后，调用 clearUnreadCount() 方法，清除本地的会话未读数。（在 v2.10.4 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 会话免打扰 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.5.8
targetId	String	是	系统会话 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.SYSTEM;
var targetId = '接收方的 userId';
var statusItem = {
isTop: true
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

会话置顶　　会话置顶 设置置顶

更新时间:2024-08-30

API 参考：setConversationStatus

参数说明

参数	类型	必填	说明	最低版本
conversationType	Number	是	会话类型，系统会话传入 RongIMLib.ConversationType.SYSTEM	2.5.8
targetId	String	是	系统会话 ID	2.5.8
statusItem	Object	是	设置对象	2.5.8
statusItem.isTop	Boolean	否	是否置顶	2.5.8
statusItem.notificationStatus	Number	否	是否免打扰： 1 开启免打扰 | 2 关闭免打扰	2.5.8
callback	Object	是	回调对象	2.5.8
callback.onSuccess	Function	是	成功回调	2.5.8
callback.onError	Function	是	失败回调	2.5.8

代码示例

```
var conversationType = RongIMLib.ConversationType.SYSTEM;
var targetId = '接收方的 userId';
var statusItem = {
isTop: true
};
RongIMClient.getInstance().setConversationStatus(conversationType, targetId, statusItem, {
onSuccess: function() {
console.log('设置会话状态成功')
},
onError: function(error) {
console.log('设置会话状态失败', error)
}
})
```

低版本实现

⚠ 警告

SDK 2.5.8 版本以下可使用如下方法设置

设置置顶

Web SDK 没有本地数据库，不提供 设置会话置顶 接口。如需要实现 会话置顶 功能可参照如下解决方案。

会话置顶、消息免打扰实现思路：

1. 把置顶、免打扰的会话存在自己的服务器，当前用户 + conversationType + targetId 可以确定一个唯一的会话。
2. 渲染会话列表前先自己的服务器获取置顶、免打扰的会话，与 getConversationList 返回的会话列表通过比对 conversationType + targetId 进行合并。
3. 群组信息、用户信息需要在自己的服务器。

取消置顶

如您需要实现 **取消会话置顶** 功能可根据您 **设置会话置顶** 功能的实现对应实现取消会话置顶。

消息接收 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「设置监听」文档中的设置消息监听。

历史消息获取　　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。

API 参考：getHistoryMessages

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，系统会话传入 RongIMLib.ConversationType.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
objectName	String	否	消息类名,仅桌面版解决方案有用	2.3.4
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
hasMsg	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
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

字段名	类型	说明
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.ConversationType.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 更新日志 关于停止维护 IMLib v2 旧版 SDK 的声明

更新时间:2024-08-30

提示

- **Web IMLib v2 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 v2 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 2x SDK 的客户可以通过 `RongIMLib-v2-Adapter` 无缝替换升级。详见 升级说明。**
- **未来我们将在 `RongIMLib-v2-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。
3. 修复了 Electron 本地插入 RC:RcNtf 消息的发送状态异常的问题。
4. 修复了在 Electron 平台，RTC 信令发送和解析失败的问题。
5. 修复了连接时连续收到多个 30021 导致子进程崩溃的问题。
6. 修复了应用退出，子进程会重启的问题。

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。

v5.9.6

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。

v5.9.2

发布日期：2023/12/13

问题修复：

1. 修复了 Electron 在 Windows 平台发送消息接口响应延迟高的问题。
2. 修复了 Electron 在 Windows 平台拉大量离线消息时应用卡顿的问题。

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。
4. Electron 平台修复发送撤回消息后,再次拉到撤回消息时原始消息被修改两次的问题。
5. 修复 Electron 的 Windows 平台退出时卡死的问题。
6. 优化消息量大时，在 Windows 平台会导致应用卡顿问题。

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 ELectron 平台 CMP 连接失败后未重连的问题。
4. 修复推送配置中单独设置 iOSConfig 或者 androidConfig 不生效的问题。
5. 修复 insertMessage 接口多余的校验参数 messageType 的问题。

v5.7.2

发布日期：2023/02/07

问题修复：

1. 修复了 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

问题修复：

1. 修复了 ELectron 平台获取会话列表中 hasMentioned 字段错误的问题。
2. 修复了 ELectron 平台获取消息中 isMentioned 字段错误的问题。

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 `setConnectionStatusListener`，请使用 `onConnectionStatusChange`。

v5.6.1

发布日期：2022/11/18

优化：

1. 在 Electron 平台，收到撤回消息时 SDK 内部将被撤回消息更新为小灰条消息 `RC:RcNtf`。

v5.6.0

发布日期：2022/11/04

问题修复：

1. 修复了断网重连时如果 token 过期，应用层收不到状态通知的问题。
2. 修复了多端登录时 ELectron 端收到消息的 offLineMessage 为 true 的问题。
3. 修复了 Web 端多端登录情况下，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。
4. 修复了 Electron 平台插入本地消息时（`insertMessage`），因传入的 message 中指定了服务端消息 ID（`messageUId`），导致消息可能重复的问题。
5. 修复了 Electron 平台引用消息和图文消息无法被搜索的问题。
6. 修复了 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. 在 ELectron 平台，主进程 @rongcloud/electron 初始化时，强制要求传参 appkey，否则初始化失败，详见 主进程初始化。
2. 修改 ELectron 平台扩展 .node 包的下载方式，详见：安装 .node 文件。
3. Web 端不再支持 Comet 连接模式，仅支持 Websocket 连接。

v5.5.5

发布日期：2022/10/01

问题修复：

1. 修复了导航数据变更通知向前兼容报错问题。

非兼容性变更：
setMessageContent 接口增加了 callback 参数回调。

v5.5.2

发布日期：2022/09/09

问题修复：

1. 修复了升级到 5.5.0 版本时，Electron 中数据库会话列表丢失的问题。

v5.5.1

发布日期：2022/09/01

问题修复：

1. 修复了在 Electron 中发送自定义消息时，content 中数字大于 32 位可能导致崩溃的问题。

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. 修复在 electron 平台多个窗口同时调用 connect 方法时，部分窗口没有返回结果的问题
4. 修复在 electron 平台主进程会报 window is not defined 的错误
5. 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 user 和 extra 字段

RongIMLib 版本

2.11.3

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

2.11.2

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。

2.11.1

发布日期：2022/02/17

问题修复：

1. 修复环境中 console 无法使用时导致 SDK 无法使用的问题
2. 修复已读回执消息未通知的问题

2.11.0

发布日期：2022/01/07

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

2.10.4

发布日期：2021/12/30

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

新增功能：

1. 新增清除全部未读数接口

2.10.3

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

2.10.2

发布日期：2021/11/25

问题修复：

1. 针对 `Electron-Solution` 包用户，修复开通「允许加入多个聊天室」功能时，断线重连后可能重新加入错误的聊天室的问题

2.10.1

发布日期：2021/11/04

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。

2.10.0

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

问题修复：

1. 修复加入聊天室错误时没有状态码的问题

2.9.10

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

2.9.9

发布日期：2021/09/24

问题修复：

1. 修复 Electron-Solution 中多窗口时切换用户后获取 userId 可能不对的问题

功能优化：

1. 优化消息拉取功能

2.9.8

发布日期：2021/09/10

问题修复：

1. 修复切换用户时会话列表最后一条消息可能不对的问题
2. 修复推送可能丢失 pushData 字段的问题
3. 修复 Electron-Solution 中获取已删除的会话时，会话状态不对的问题

功能优化：

1. 解除端上不允许发系统消息的限制

2.9.7

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 Electron-Solution 中发送回执后接受状态没更新的问题

3. 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

2.9.6

发布日期：2021/08/13

问题修复：

1. 修复撤回消息时会话最后一条消息可能没有更新的问题

2. 修复撤回消息时多端同步到的会话信息可能错误的问题

3. 修复收到直发消息时清理未读数可能不能清空的问题

2.9.5

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

2.9.4

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. 增加 PC 端请求 navi 时的证书认证开关

2. 优化重连逻辑

3. 增加群聊天回执状态本地存储过期时间配置

2.9.3

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

2.9.2

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

2.9.1

发布日期：2021/06/11

问题修复

1. 修复可能会发送多个 ping 的问题

功能优化：

1. 优化 navi 存储策略

2.9.0

发布日期：2021/06/03

新增功能

1. 新增 typing 状态通知

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. 接收消息内容添加 pushConfig 字段

5. 会话列表增加 matchCount 字段

6. 重定向失败时继续重连

7. 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

2.8.6

发布日期：2021/05/27

功能优化：

1. searchConversationByContent 方法增加返回 matchCount 字段

2. 增加 appkey 类型校验

问题修复：

1. 修复重新连接后未拉取离线消息问题

2. 修复发送消息失败时 messageId 返回错误问题

2.8.5

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

2.8.4

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

2.8.3

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

2.8.2

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

功能优化：

1. 发送消息失败时返回消息内容，增加 sentTime 参数
2. 解决小程序平台打包体积过大问题

2.8.1

发布日期：2021/04/15

问题修复：

1. 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：

1. 更新日志输出格式

2.8.0

发布日期：2021/04/12

新增功能：

1. 会话分组功能：通过给会话增加 tag 标签以便于对会话进行归类管理
2. 支持移动端 push 推送内容可配置
3. 优化对服务器侧主动断开连接后的重连处理逻辑
4. 支持新版本 Electron 桌面端解决方案 - @rongcloud/electron-solution
5. 不再支持老版本桌面解决方案

2.7.6

发布日期：2021/03/09

问题修复：

1. 补全 RongIMClient.MessageType 以兼容 2.5 及之前的 SDK 版本

2.7.5

发布日期：2021/03/08

问题修复：

1. 修复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. 修复 web 端多组织功能引起的获取 Conversation 实例数据失败
3. 修复获取单一 Conversation 会话实例时频繁拉取会话列表问题
4. 修复构建脚本错误导致 IMLib 文件重复打包 engine 依赖造成代码冗余问题
5. 修复获取历史消息时，若 timestamp 为 0、count 为 1 时无法获取数据问题

2.7.4

发布日期：2021/02/24

问题修复：

1. 修复聊天室拉取消息重复问题

2.7.3

发布日期：2021/02/05

问题修复：

1. 修复时间戳取值错误导致聊天室拉取消息可能重复问题
2. 修复接收状态消息更新收件箱时间戳导致部分消息丢失问题

2.7.2

发布日期：2021/01/28

问题修复：

1. 修复多端情况下服务器通知拉取聊天室消息，未加入聊天室的端报错的问题

功能优化：

1. 兼容小程序平台

2.7.0

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

2.6.2

发布日期：2020/12/16

问题修复：

1. 修复了同时集成 RTC SDK 时，IM 无法收到消息问题

2.6.1

发布日期：2020/12/11

新增功能：

1. 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. 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-v2

2. 提升了 SDK 代码健壮性、稳定性

3. 提升了 SDK 性能，减少不必要的异步任务

2.5.14

发布日期：2020/11/25

功能优化：

1. 优化了重连失败时的处理逻辑

2.5.13

发布日期：2020/11/23

问题修复：

1. 修复了偶现的发送完消息导致缓存中会话免打扰状态被改为免打扰的问题

2. 修复了重复设置会话状态报 26002 的问题

2.5.12

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。

2.5.11

发布日期：2020/09/04

问题修复：

1. 修复了 SDK 部分 BUG。

2.5.10

发布日期：2020/08/19

问题修复：

1. 修复了 SDK 对部分 Emoji 特殊字符解析不正确的问题

功能优化：

1. 添加了 CMP 连接超时时间，10 秒没有响应连接超时

2.5.9

发布日期：2020/07/21

新增功能：

1. 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

问题修复：

1. 修复了多端断网重连后拉取消息时间不一致的问题
2. 修复了发送合并转发消息 content 中缺少 conversationType 字段的问题

2.5.8

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

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。

问题修复：

1. 修复了调用 disconnect 或 logout 后未终止当前重连的逻辑
2. 优化了获取会话列表功能

2.5.7

发布日期：2020/05/08

问题修复：

1. 修复了开发者代码报错影响 SDK 消息抛出的问题

2.5.6

发布日期：2020/04/10

新增功能：

1. 增加了动态导航功能，使融云 IM 服务更加稳定
2. 增加了发送消息时，对消息体大小超限增加 30016 提示
3. 针对文字、图片、语音、高清语音消息增加阅后即焚 burnDuration 字段
4. 增加了引用消息(ReferenceMessage)、动态图片消息(GIFMessage)、小视频消息(SightMessage)内置消息类型
5. 增加了发送状态消息功能

功能优化：

1. 根据传入导航匹配链接协议头
2. 修正了在线消息的接收时间字段(receiveTime)

2.5.5

发布日期：2020/02/17

问题修复：

1. 修复了聊天室广播消息偶现丢失的问题

2.5.4

发布日期：2020/01/10

新增功能：

1. 长轮训链接前增加 cmp 嗅探逻辑(小程序基于 Web，且包含多 cmp 地址)

功能优化：

1. 优化了聊天室设置/删除自定义属性返回值

2. 桌面版状态码与 Web 状态码对齐

2.5.3

发布日期：2019/12/18

新增功能：

1. 增加了聊天室属性自定义存储功能，查看文档

2. 增加了删除指定的一条或者一组服务端历史消息功能，查看文档

3. 增加了按时间向前获取 N 条历史消息功能，查看文档。

4. 允许多次注册状态监听器、消息监听器

问题修复：

1. 修复了多个 Tab 页登录同一用户，未读数重复增加的问题。

2. 修复了 iOS 9 浏览器链接失败的问题。

3. 修复了开发者二次打包后代码编译错误的问题。

2.5.2

发布日期：2019/11/14

问题修复：

1. 修复了 SDK BUG 提升了稳定性

2.5.1

发布日期：2019/11/01

问题修复：

1. 修复了 SDK BUG 提升了稳定性

2.5.0

发布日期：2019/06/11

新增功能：

1. Web 端支持多导航逻辑

问题修复：

1. 修复了 SDK BUG 提升了稳定性

2.4.0

发布日期：2019/03/30

功能优化：

1. 优化了消息拉取机制
2. 增加了对 RTCLib SDK 3.0.0 版本的支持
3. 对 CallLib SDK 核心音视频引擎进行了升级，升级后新版本与之前版本不兼容，旧版本仍然可以使用。参考知识库文档

2.3.5

发布日期：2018/12/18

新增功能：

1. 新增了清除总未读消息数方法(clearTotalUnreadCount)

问题修复：

1. 修复了重连后未读数错误问题
2. 修复了偶发的无法收到即时消息问题

2.3.4

发布日期：2018/11/13

功能优化：

1. 优化了消息收发逻辑
2. 修复了 SDK 部分 BUG

2.3.3

发布日期：2018/08/09

问题修复：

1. 优化了导航缓存逻辑
2. 修复部分 BUG，增强了 SDK 稳定性

2.3.2

发布日期：2018/06/01

问题修复：

1. 优化了复合连接
2. 修复部分 BUG，增强了 SDK 稳定性

2.3.1

发布日期：2018/04/12

功能优化：

1. 优化了链接，支持复合链接策略
2. 优化了聊天室接收消息逻辑

2.3.0

发布日期：2017/12/19

新增功能：

1. 修复了聊天室消息偶尔重复的问题
2. 新增了清除服务端单群聊历史消息功能
3. 新增了发送群定向消息功能

2.2.9

发布日期：2017/11/30

问题修复：

1. 修复了多端同步消息 RecallCommandMessage 无会话类型的问题。
2. 修复了获取当前连接状态偶尔不准确的问题。

2.2.8

发布日期：2017/08/28

问题修复：

1. 修复了不刷新页面重复加入聊天室无法获取最新聊天内容的问题。
2. 修复了可以撤回其他用户发送的消息的问题。

功能优化：

1. 优化了错误信息日志，RongIMClient.init(appkey, null, {showError: true}) 开启，默认关闭。

2.2.7

发布日期：2017/08/17

问题修复：

1. 修复了一些 BUG ，增强了 SDK 稳定性

2.2.6

发布日期：2017/07/25

新增功能：

1. 增加了聊天室获取历史消息功能

问题修复：

1. 修复了删除会话时存在的 BUG
2. 修复了聊天室拉取消息时的 BUG

功能优化：

1. 进行了模块化加载代码的优化

2.2.5

发布日期：2017/02/13

新增功能：

1. 新增获取聊天室历史消息方法 getChatRoomHistoryMessages。

问题修复：

2. 修复了历史消息接口 count 为 1 ，第二次获取消息缺失的问题。
3. 修复了 α 和 β 等特殊字符乱码的问题。
4. 更新了 isPullSendBox 心跳拉取过程中拉取发件箱标识修改为 False。

2.2.4

发布日期：2016/09/27

功能优化：

1. 将导航、protobuf 等内部依赖参数化、可配置。

2. 优化了 protobuf 引入逻辑。

3. 优化了 Comet、WebSocket 自动识别。

4. 优化了 HTTP、HTTPS 自动识别。

2.2.3

发布日期：2016/09/06

新增功能：

1. 增加了群组、讨论组中阅读消息回执功能。

2. 优化了群组、讨论组未读消息数同步功能。

2.2.2

发布日期：2016/08/18

新增功能：

1. 增加了发送文件消息功能。

2.2.1

发布日期：2016/07/27

新增功能：

1. 支持用户加入多个聊天室。

2.2.0

发布日期：2016/07/15

新增功能：

1. 增加 WebSQL 本地存储，为 PC 端应用程序提供了本地存储方案。

2. 聊天室增加获取实时在线人数方法。

升级说明　　升级说明 关于停止维护 IMLib v2 旧版 SDK 的声明

更新时间:2024-08-30

提示

- **Web IMLib v2 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 v2 版本的用户，转为使用 Adapter 方式进行支持。集成旧版 2x SDK 的客户可以通过 `RongIMLib-v2-Adapter` 无缝替换升级。详见 升级说明。**
- **未来我们将在 `RongIMLib-v2-Adapter` 上进行问题修复，但不会增加新功能。**

IMLib 2.x 替换为 v2-adapter

NPM 包变更

原 `@rongcloud/imlib-v2` 包已停止维护，请使用 `@rongcloud/imlib-v2-adapter` 替代。

```
# 移除旧版本依赖
npm rm @rongcloud/imlib-v2 @rongcloud/engine
# 安装 RongIMLib-v2-Adapter
npm install @rongcloud/engine@latest @rongcloud/imlib-v2-adapter@latest -S
```

同时需要在集成代码中修改包引用。

```
// import * as RongIMLib from '@rongcloud/imlib-v2' 需修改为
import * as RongIMLib from '@rongcloud/imlib-v2-adapter'

// const RongIMLib = require('@rongcloud/imlib-v2') 需修改为
const RongIMLib = require('@rongcloud/imlib-v2-adapter')
```

CDN 链接引入

原 `RongIMLib-2.x.x.prod.js` SDK 已停止维护，请使用 `RongIMLib-v2-Adapter` 的最新版本替代。

> `RongIMLib-v2-Adapter` 的最新版本可参见引入 SDK。

关于 v2.5 及更早版本的升级说明

IMLib 2.5 及更早版本，需注意如下变更：

1. SDK 不再支持公众号与客服插件相关接口！
2. IM 链接因网络问题意外中断后，SDK 会自动重连，应用层无需再调用 `reconnect` 方法。
3. 若 `'RC:ProfileNtf'`、`'RC:CmdNtf'`、`'RC:InfoNtf'` 类型消息的 `content.data` 字段为 Json 字符串，需自行解析。
4. SDK 不再兼容 IE 浏览器 6 至 8 版本！

废弃功能

会话

废弃功能

因本地存储不准确，不再维护以下会话相关接口

废弃方法	描述
~~getConversationUnreadCount~~	按会话类型获取会话未读数
~~clearConversations~~	按会话类型删除会话
~~clearTotalUnreadCount~~	清除所有会话未读数

兼容方法

`getConversationUnreadCount` 兼容示例代码

```
// 1、调用 getConversationList 方法获取所有会话
var targetTypes = [1,3]; //目标会话类型
var callback = {
onSuccess: function(list) {
// 2、根据返回的会话列表中 conversationType 方法字段筛选出目标会话
var targetConvers = list.filter(function(conver) {
return targetTypes.indexOf(conver.conversationType.toString()) > -1
});
// 3、根据筛选出的会话计算未读数
var targetUnreadCount = 0
targetConvers.forEach(function(conver) {
targetUnreadCount += conver.unreadMessageCount
});
console.log('按会话类型获取会话未读数成功', targetUnreadCount);
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, 1000);
```

`clearConversations` 兼容示例代码

```
// 1、调用 `getConversationList` 方法获取所有会话
var targetTypes = [1,3]; //目标会话类型
var callback = {
onSuccess: function(list) {
// 2、根据返回的会话列表中 `conversationType` 字段筛选出目标会话
var targetConvers = list.filter(function(conver) {
return targetTypes.indexOf(conver.conversationType) > -1
});
// 3、循环调用 removeConversation 方法删除会话
var targetUnreadCount = 0
targetConvers.forEach(function(conver) {
RongIMClient.getInstance().removeConversation(conver.conversationType, conver.targetId, { onSuccess:
function () { }, onError: function () { } });
});
console.log('按会话类型删除会话成功');
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, 1000);
```

clearTotalUnreadCount 兼容示例代码

```
// 1、调用 `getConversationList` 方法获取所有会话
var targetTypes = [1,3]; //目标会话类型
var callback = {
onSuccess: function(list) {
// 2、循环调用 `clearUnreadCount` 方法清除未读数
list.forEach(function(conver) {
RongIMClient.getInstance().clearUnreadCount(conver.conversationType, conver.targetId, { onSuccess:
function () { }, onError: function () { } });
});
console.log('按会话类型删除会话成功');
},
onError: function(error) {
console.log('获取会话列表失败', error);
}
}
RongIMClient.getInstance().getConversationList(callback, null, 1000);
```

讨论组

废弃功能

废弃讨论组相关接口

废弃方法	描述
~~createDiscussion~~	创建讨论组
~~getDiscussion~~	获取讨论组信息
~~quitDiscussion~~	退出讨论组
~~addMemberToDiscussion~~	加入讨论组
~~removeMemberFromDiscussion~~	将指定成员移除讨论租

废弃方法	描述
~~setDiscussionInviteStatus~~	设置讨论组邀请状态
~~setDiscussionName~~	设置讨论组名称

状态码 状态码

更新时间:2024-08-30

状态码	说明
0	连接成功
1	连接中
2	连接断开
3	网络不可以用，此处可触发重连，详细请参考 重连文档。
4	连接关闭
6	当前账户在其他 Web 设备登录，本机会被踢掉线
7	websocket 连接失败
8	AppKey 被封禁或已删除
9	用户被封禁
12	域名错误
20	初始化时填写的 AppKey 不正确，请在控制台获取。
201	开始请求导航
202	请求导航结束
203	请求导航失败
204	请求导航超时

状态码	说明
1101	互踢次数过多，此时可能出现：在其它他设备登陆有 reconnect 逻辑
-1	未知错误
-3	参数错误
405	已被对方加入黑名单，消息发送失败。
407	未在对方的白名单中，消息发送失败。
20407	群组 Id 无效
20106	该用户处于单聊禁言状态，禁止发送单聊消息。
20604	发送消息频率过高，1秒钟最多只允许发送 5 条消息，详细请联系商务，电话：13161856839。
20605	信令被封禁。如遇到该错误，请提交工单。
20607	调用超过频率限制，请稍后再试。
21501	发送的消息中包含敏感词 （发送方发送失败，接收方不会收到消息）
21502	消息中敏感词已经被替换 （接收方可以收到被替换之后的消息）
22406	当前用户不在群组中
22408	当前用户在群组中已被禁言
23406	当前用户不在聊天室中，请加入聊天室在调用和聊天室相关方法
23407	获取用户失败
23408	当前用户在聊天室中已被禁言
23409	已被踢出并禁止加入聊天室。被禁止的时间取决于服务端调用踢出接口时传入的时间。

状态码	说明
23410	聊天室不存在
23411	聊天室成员超限，开发者可以【 提交工单 】申请聊天室人数限制变更
23412	聊天室接口参数不正确。请确认参数是否为空或者有效。
23414	聊天室云存储业务未开通，开通请【 提交工单 】
23423	聊天室属性个数超限，单个聊天室默认上限为 100 个
23424	没有权限修改聊天室中已存在的属性值
23425	聊天室属性设置频率超限，单个聊天室每秒默认设置上限为 100 次
23426	未开通聊天室属性自定义设置。请在控制台免费基础功能页面开通聊天室属性自定义设置。
23427	聊天室属性不存在
25101	撤回消息参数不正确。请确认撤回消息参数是否正确的填写
25102	未开通历史消息云存储
30001	当前连接不可用（连接已经被释放）。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30002	当前连接不可用。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30003	客户端发送消息请求，融云服务端响应超时。
30004	导航 HTTP 发送失败。如果是偶尔出现此错误，SDK 会做好自动重连，开发者无须处理。对于 iOS 平台，如果一直连接不上，应该是您没有设置好 ATS。ATS 默认只使用 HTTPS 协议，当 HTTP 协议被禁止时 SDK 会一直 30004 错误。您可以在我们 iOS 开发文档中搜索到 ATS 设置。
30005	请求连接导航地址失败。SDK 会做好自动重连，开发者无须处理。如果使用私有云持续报此问题，请确认导航地址是否正确。
30006	请求连接导航地址后，接收数据失败。SDK 会做好自动重连，开发者无须处理。如果使用私有云持续报此问题，请确认导航地址是否正确。
30007	导航 HTTP 请求失败。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。

状态码	说明
30008	导航 HTTP 返回数据格式错误。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30009	导航数据解析后，其中不存在有效 IP 地址。如果使用私有云出现此问题，请排查导航数据是否正常。
30010	Socket 不存在，一般由于没有连接（connect）引起的。
30011	Socket 连接被断开，主要有两种情况，一是用户主动调用 disconnect 之后，Socket 被服务器断开；二是中间路由原因等导致 Socket 断开。
30012	PING 失败。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30013	PING 超时。 建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30014	信令发送失败。建立连接的临时错误码，SDK 会做好自动重连，开发者无须处理。
30016	消息大小超限，消息体最大 128 KB
31000	做 connect 连接时，收到的 ACK 超时
33001	未调用 init 初始化函数，请参考关于初始化的文档说明
33003	调用接口时传入的参数不正确
33006	连接中，再调用 connect 被拒绝
33007	未开通历史消息云存储服务。可在控制台 开启服务。
34002	小视频时间长度超出限制，默认小视频时长上限为 2 分钟
34003	GIF 消息文件大小超出限制， 默认 GIF 文件大小上限是 2 MB
35001	群组异常信息
35002	匹配群信息异常
35006	公有云包不允许连接私有云环境

状态码	说明
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
36001	加入聊天室 ID 为空
36002	加入聊天室失败
36003	拉取聊天室历史消息失败
36004	聊天室 KV 未找到
38001	获取草稿失败
38002	保存草稿失败
38003	删除草稿失败