即时通信

IMLib / IMKit

Flutter 5.X

2024-08-30

开发指导

更新时间:2024-08-30

融云 IMLib for Flutter 已支持 Android、iOS 跨平台开发。

- 如仍使用旧版 SDK（指低于 Flutter 5.2.4 版本），请阅读 SDK 变更说明。
- SDK 从 5.6.12 开始，不再支持 Web 平台，更多信息

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

功能	单聊（Private）	普通群组（Group）	超级群（UltraGroup）	聊天室（Chatroom）
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 »

平台兼容性

IM 客户端 SDK 支持主流移动操作平台，客户端功能在多端基本保持一致，支持多平台互通。以下数据基于 5.X 版本 SDK。

平台/框架	接口语种	支持架构	说明
Android	Java	armeabi-v7a、arm64-v8a、x86、x86-64	系统版本 4.4 及以上
iOS	Objective-C	真机：arm64、armv7。模拟器： arm64（5.4.7+）、x86_64	系统版本 9.0 及以上
Web	Javascript	---	---
Electron	Javascript	详见下方 **Electron 版本与架构支持**	Electron 11.1.x、14.0.0、16.0.x、20.0.x
Flutter	dart	---	Flutter 2.0.0 及以上
React Native	Typescript	-	react-native 0.60 及以上
uni-app	Javascript	---	uni-app 2.8.1 及以上
Unity	C#	armeabi-v7a、arm64-v8a	---

- **Electron 版本与架构支持：**

  Electron 框架需要通过 Web 端 SDK 的 Electron 模块支持（详见 Electron 集成方案），适用于开发运行在 Windows、Linux、MacOS 平台的桌面版即时通讯应用。下表列出了目前已支持的 Electron 版本、桌面操作系统版本及 CPU 架构：

Electron 版本	平台	支持架构	备注
Electron 11.1.x	Windows	ia32 (x86)	`win32-ia32`
Electron 11.1.x	Linux	x64	`linux-x64`
Electron 11.1.x	Linux	arm64	`linux-arm64`
Electron 11.1.x	Mac	x64	`darwin-x64`
Electron 14.0.0	Windows	ia32 (x86)	`win32-ia32`
Electron 14.0.0	Mac	x64	`darwin-x64`
Electron 16.0.x	Windows	ia32 (x86)	`win32-ia32`
Electron 16.0.x	Mac	x64	`darwin-x64`
Electron 20.0.x	Windows	ia32 (x86)	`win32-ia32`
Electron 20.0.x	Mac	x64	`darwin-x64`
Electron 20.0.x	Mac	arm64	`darwin-arm64`

版本支持

IM 客户端 SDK 针对各平台/框架提供的最新版本如下（--- 表示暂未支持）：

SDK/平台	Android	iOS	Web	Electron	Flutter	React Native	Unity	uni-app	小程序
IMLib	5.6.x	5.6.x	5.9.x	5.9.x	5.4.x	5.2.x	5.1.x	5.4.x	5.9.x
IMKit	5.6.x	5.6.x	5.9.x	---	---	---	---	---	---
Global IM UIKit	1.0.x	1.0.x	1.0.x	1.0.x	---	---	---	---	---

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。

SDK 变更说明 停止维护旧版 Flutter 5.X SDK 的说明

更新时间:2024-08-30

> 提示
>
> 仅适用于已集成 Flutter 5.X 旧版 SDK (`rongcloud_im_plugin`) 的客户。

Flutter 5.X 旧版 SDK (`rongcloud_im_plugin`) 已不再进行维护，旧版开发文档不再公开提供，建议已集成旧版 SDK 的客户迁移到新版本 SDK。新版本 SDK 整体进行了优化重构。所有新功能都将在新版 SDK 中进行添加。

新集成的客户，请直接使用新版 SDK（rongcloud_im_wrapper_plugin ）。详见快速上手。

迁移方法

Flutter IMLib 5x 旧版 SDK 与新版 SDK 不兼容，为帮助已集成 Flutter 5.X 旧版 SDK 的客户迁移到新版 SDK（rongcloud_im_wrapper_plugin ），我们整理了新旧版本 SDK API 使用差异。请开发者根据自身 API 使用情况与 API 差异，合理安排开发周期。详细内容可参考以下文档：

新旧 API 对照速查表

注意事项

1. 新版本 SDK 与旧版本（指低于 Flutter 5.2.4 版本）定义的自定义消息机制互相不兼容，无法正确解析。
2. 新版本 SDK 更换了仓库地址，旧的仓库不在进行维护。

其他区别

1. 新版本 SDK 的接口需要使用 `RCIMIWEngine` 对象进行调用，且有完整的生命周期，所以构建对象需要开发者进行保存，以便后续接口调用。旧版本 SDK 的接口直接通过调用 `RongIMClient` 提供的静态方法。
2. 新版本 SDK 接口大部分的结果处理，都不在当前接口进行体现，需要开发者实现对应的方法回调进行处理，且如果接口调用多次，SDK 都会触发同一个回调，但回调中会返回接口调用参数，需要开发者根据参数来进行区分。旧版本部分接口回调可在接口内直接获取。
3. 新版本 SDK 如果多处设置同一个监听，只有最后一次设置才会生效。为避免其余监听失效而引起问题，建议全局只设置一次。

支持 callback 回调参数

Flutter SDK 从 5.3.1 版本开始，部分接口新增 callback 参数。在传入 callback 参数后，仅通过 callback 参数触发回调，不再触发其他回调。

平台适配说明

更新时间:2024-08-30

融云即时通讯服务提供 Flutter 支持 Android、iOS 跨平台开发。

快速上手

更新时间:2024-08-30

本教程是为了让新手快速了解融云即时通讯能力库（IMLib）。在本教程中，您可以体验集成 SDK 的基本流程和 IMLib 的基础通信能力。

前置条件

- 注册开发者账号。注册成功后，控制台会默认自动创建您的首个应用，默认生成开发环境下的 App Key，使用国内数据中心。
- 获取开发环境的应用 App Key。如不使用默认应用，请参考 如何创建应用，并获取对应环境 App Key 和 App Secret。

> 提示
>
> 每个应用具有两个不同的 App Key，分别对应开发环境与生产环境，两个环境之间数据隔离。在您的应用正式上线前，可切换到使用生产环境的 App Key，以便上线前进行测试和最终发布。

- 阅读 SDK 变更说明。SDK 在 5.2.4、5.4.0 均有较大变更，请确保您已充分了解相关差异，例如接口回调的方式变更等。

环境要求

- **Dart**：2.15.0 ≦ dart < 3.0.0
- **Flutter**： ≧ 2.5.0

Demo 项目

融云 IMLib for Flutter SDK 提供了一个不含任何聊天界面的 Demo 项目，集中演示了各个接口的调用方法。

https://github.com/rongcloud/im-flutter-wrapper

导入 SDK

融云在 Flutter 平台以插件库形式提供 SDK，其中已包含 dart 层与 Android/iOS 平台层代码。

1. 在项目的 `pubspec.yaml` 中添加依赖。SDK 最新版本可以查询 官方仓库。

```
dependencies:
flutter:
sdk: flutter

rongcloud_im_wrapper_plugin: x.y.z
```

2. 在项目路径执行 `flutter pub get` 来下载相关插件。

3. 导入头文件。

```
import 'package:rongcloud_im_wrapper_plugin/rongcloud_im_wrapper_plugin.dart';
```

关闭 Android 自动混淆

如果需要配置混淆，请参考 Android 文档 。

如果不需要配置混淆，请关闭自动混淆。否则执行 `flutter build apk` 时，Flutter 会自动对 Android 代码进行混淆编译，导致 so 库找不到的错误。

```
android {
buildTypes {
release {
// Enables code shrinking, obfuscation, and optimization for only
// your project's release build type.
minifyEnabled false

// Enables resource shrinking, which is performed by the
// Android Gradle plugin.
shrinkResources false
}
}
...
}
```

初始化

在使用 SDK 所有功能之前，必须先调用此方法初始化 SDK。SDK 内的接口都需要通过 RCIMIWEngine 的对象进行调用，RCIMIWEngine 对象只能通过 `create` 进行获取，所以在调用 `create` 方法获取到实体对象后，开发者需要自行进行保存，以便后续接口调用。

初始化时需要传入上文获取的 App Key。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
RCIMIWEngine engine = await RCIMIWEngine.create(appKey, options);
```

引擎配置请参见引擎配置。

获取用户 Token

用户 Token 是与用户 ID 对应的身份验证令牌，是应用程序的用户在融云的唯一身份标识。应用客户端在使用融云即时通讯功能前必须与融云建立 IM 连接，连接时必须传入 Token。

在实际业务运行过程中，应用客户端需要通过应用的服务端调用 IM Server API 申请取得 Token。详见 Server API 文档 注册用户。

在本教程中，为了快速体验和测试 SDK，我们将使用控制台「北极星」开发者工具箱，从 API 调试页面调用 获取 Token 接口，获取到 **userId** 为 1 的用户的 Token。提交后，可在返回正文中取得 Token 字符串。

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{"code":200,"userId":"1","token":"gxld6GHx3t1eDxof1qtxxYrQcjkbhl1V@sgyu.cn.example.com;sgyu.cn.example.c
```

建立 IM 连接

1. 使用 onConnectionStatusChanged 监听 IM 连接状态的变化，连接状态发生变化时返回 RCIMIWConnectionStatus。详见连接状态监听。

```
engine?.onConnectionStatusChanged = (RCIMIWConnectionStatus? status) {
//...
};
```

2. 使用上方获取的 Token，模拟 userId 为 1 的用户连接到融云服务器。调用结果会直接通过 connect 方法中传入的 RCIMIWConnectCallback 返回。

```
RCIMIWConnectCallback? callback = RCIMIWConnectCallback(onDatabaseOpened: (int? code) {
//...
}, onConnected: (int? code, String? userId) {
//...
});

int? ret = await engine?.connect(token, timeout, callback:callback);
```

SDK 已实现自动重连机制，请参见连接。

监听消息

实现此功能需要开发者实现消息监听回调。

设置消息接收监听器，用于接收所有类型的实时或者离线消息。

1. `message` 接收到的消息对象
2. `left` 当客户端连接成功后，服务端会将所有补偿消息以消息包的形式下发给客户端，最多每 200 条消息为一个消息包，即一个 Package, 客户端接受到消息包后，会逐条解析并通知应用。left 为当前消息包（Package）里还剩余的消息条数
3. `offline` 消息是否离线消息
4. `hasPackage` 是否在服务端还存在未下发的消息包

```
engine?.onMessageReceived = (RCIMIWMessage? message, int? left, bool? offline, bool? hasPackage) {
//...
};
```

发送消息

```
RCIMIWTextMessage? textMessage = await engine.createTextMessage(
conversationType,
targetId,
channelId,
text,
);

engine.sendMessage(message);
```

监听消息发送结果

```
engine?.onMessageAttached = (RCIMIWMessage? message) {
//...
};

engine?.onMessageSent = (int? code, RCIMIWMessage? message) {
//...
};
```

退出登录

```
int? ret = await engine?.disconnect(receivePush);
```

引擎销毁

```
await engine?.destroy();
```

导入 SDK 环境要求

更新时间:2024-08-30

- **Dart**：2.15.0 ≧ dart < 3.0.0
- **Flutter**： ≧ 2.5.0

安装 SDK

融云在 Flutter 平台以插件库形式提供 SDK，其中已包含 dart 层与 Android/iOS 平台层代码。

1. 在项目的 `pubspec.yaml` 中添加依赖。

```
dependencies:
flutter:
sdk: flutter

rongcloud_im_wrapper_plugin: x.y.z
```

> 提示
>
> 请将 `x.y.z` 替换为要使用的版本。SDK 最新版本可以查询 官方仓库。为明确 Flutter SDK 版本对原生版本 SDK 的引用关系，从新版 SDK（rongcloud_im_wrapper_plugin）推出开始，Flutter SDK 版本号的前三位会和原生 SDK 版本前三位保持一致。例如 Flutter SDK 版本为 5.2.4，那么其使用的原生版本 SDK 也为 5.2.4。反之亦然。

2. 在项目路径执行 `flutter pub get` 来下载相关插件。

3. 导入头文件。

```
import 'package:rongcloud_im_wrapper_plugin/rongcloud_im_wrapper_plugin.dart';
```

关闭 Android 平台自动混淆

如果需要配置混淆，请参考 Android 文档。

如果不需要配置混淆，请关闭自动混淆。否则执行 `flutter build apk` 时，Flutter 会自动对 Android 代码进行混淆编译，导致 so 库找不到的错误。

```
android {
buildTypes {
release {
// Enables code shrinking, obfuscation, and optimization for only
// your project's release build type.
minifyEnabled false

// Enables resource shrinking, which is performed by the
// Android Gradle plugin.
shrinkResources false
}
}
...
}
```

引擎配置

更新时间:2024-08-30

IM 引擎提供以下配置，可按需修改。配置必须在初始化时提供。

数据中心配置

如果您的应用不使用中国国内数据中心（例如使用海外数据中心，或者使用私有部署），必须在初始化之前修改 IMLib SDK 连接的服务地址为对应数据中心地址。否则 SDK 默认连接中国国内数据中心服务地址。

使用海外数据中心的详细说明请参见配置海外数据中心服务地址 。

导航地址配置

私有部署的导航服务器地址，naviServer 必须为有效的服务器地址。默认不需要配置，如需配置，请联系商务。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
options.naviServer = '配置的导航地址';
```

媒体服务配置

私有部署的媒体服务器地址，即文件和图片的上传地址。默认不需要配置，如需配置，请联系商务。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
options.fileServer = '配置的文件服务地址';
```

配置统计服务器的信息

设置统计服务器的信息，仅限独立数据中心使用。默认不需要配置，如需配置，请联系商务。

statisticServer 必须为有效的服务器地址，否则会造成推送等业务不能正常使用。

格式说明：
1、如果使用 https，则设置为 https://cn.xxx.com:port 或 https://cn.xxx.com 格式，其中域名部分也可以是 IP，如果不指定端口，将默认使用 443 端口。
2、如果使用 http，则设置为 cn.xxx.com:port 或 cn.xxx.com 格式，其中域名部分也可以是 IP，如果不指定端口，将默认使用 80 端口。（iOS 默认只能使用 HTTPS 协议。如果您使用 http 协议，请在 iOS 工程中配置 ATS 权限）

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
options.statisticServer = '配置的统计服务器';
```

推送配置

推送是常见的基础功能。IMLib SDK 已集成融云自有推送，以及多家第三方推送，且会在 SDK 初始化后触发。因此，客户端的推送配置必须在初始化之前提供。详细说明请参见启用推送。

Android 推送信息配置

开发者在使用 Android 推送时，需要配置下面信息。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
RCIMIWPushOptions pushOptions = RCIMIWPushOptions.create(
idMI: '',
appKeyMI: '',
appIdMeizu: '',
appKeyMeizu: '',
appKeyOPPO: '',
appSecretOPPO: '',
enableHWPush: true,
enableFCM: true,
enableVIVOPush: true,
);
options.pushOptions = pushOptions;
```

是否踢出当前正在重连的设备

用户级别配置，可按需设置。设置断线重连时是否踢出当前正在重连的设备。要求 App Key 已开通用户级别配置。详见重连机制与重连互踢。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
options.kickReconnectDevice = true;
```

压缩策略配置

开发者可以自行配置 SDK 在发送图片、小视频时原始数据和缩略图的压缩配置

参数说明

参数名	类型	描述
originalImageQuality	int	原图压缩比
originalImageMaxSize	int	原图最长边的最大宽度
originalImageSize	int	原图大小限制 配置发送图片时，如果图片大小不超过则发送原图
thumbnailQuality	int	缩略图压缩比例
thumbnailMaxSize	int	缩略图压缩宽、高
thumbnailMinSize	int	缩略图压缩最小宽、高
sightCompressHeight	int	小视频压缩高度，建议使用16的倍数
sightCompressWidth	int	小视频压缩宽度，建议使用16的倍数
locationThumbnailQuality	int	位置消息缩略图压缩比例

参数名	类型	描述
locationThumbnailWidth	int	位置消息压缩的宽度
locationThumbnailHeight	int	位置消息压缩的高度

代码示例

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
RCIMIWCompressOptions compressOptions = RCIMIWCompressOptions.create(
originalImageQuality: 80,
originalImageMaxSize: 1080,
originalImageSize: 200,
thumbnailQuality: 80,
thumbnailMaxSize: 500,
thumbnailMinSize: 200,
sightCompressHeight: 160,
sightCompressWidth: 160,
);
options.compressOptions = compressOptions;
```

初始化

更新时间:2024-08-30

在使用 SDK 其它功能前，必须先进行初始化。

获取 App Key

您必须拥有正确的 App Key，才能进行初始化。

您可以控制台，查看您已创建的各个应用的 App Key。

如果您拥有多个应用，请注意选择应用名称（下图中标号 1）。另外，融云的每个应用都提供用于隔离生产和开发环境的两套独立 App Key / Secret。在获取应用的 App Key 时，请注意区分环境（生产 / 开发，下图中标号 2）。

> 提示
>
> - 如果您并非应用创建者，我们建议在获取 App Key 时确认页面上显示的数据中心是否符合预期。
> - 如果您尚未向融云申请应用上线，仅可使用开发环境。

初始化

开发者通过 `create` 方法来获取 IM 的实例对象。

> 提示
>
> SDK 内的接口都需要通过 `RCIMIWEngine` 的对象进行调用，`RCIMIWEngine` 对象只能通过 `create` 进行获取，所以在调用 `create` 方法获取到实体对象后，开发者需要自行进行保存，以便后续接口调用。

参数说明

参数名	类型	说明
appKey	String	开发者在 控制台 获取的 Appkey
options	RCIMIWEngineOptions	引擎相关配置

代码示例

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
RCIMIWEngine engine = await RCIMIWEngine.create(appKey, options);
```

监听连接状态 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。

方法

```
Function(RCIMIWConnectionStatus? status)? onConnectionStatusChanged;
```

参数说明

参数名	参数类型	描述
status	RCIMIWConnectionStatus	SDK 与融云服务器的连接状态

代码示例

```
engine?.onConnectionStatusChanged = (RCIMIWConnectionStatus? status) {
//...
};
```

连接

更新时间:2024-08-30

应用客户端成功连接到融云服务器后，才能使用融云即时通讯 SDK 的收发消息功能。

融云服务端在收到客户端发起的连接请求后，会根据连接请求里携带的用户身份验证令牌（Token 参数），判断是否允许用户连接。

前置条件

- 通过服务端 API 注册用户（获取 Token）。融云客户端 SDK 不提供获取 Token 方法。应用程序可以调用自身服务端，从融云服务端获取 Token。
  - 取得 Token 后，客户端可以按需保存 Token，供后续连接时使用。具体保存位置取决于应用程序客户端设计。如果 Token 未失效，就不必再向融云请求 Token。
  - Token 有效期可在控制台进行配置，默认为永久有效。即使重新生成了一个新 Token，未过期的旧 Token 仍然有效。Token 失效后，需要重新获取 Token。如有需要，可以主动调用服务端 API 作废 Token。
- 建议应用程序在连接之前设置连接状态监听。
- SDK 已完成初始化。

> 提示
>
> **请不要在客户端直接调用服务端 API 获取 Token。获取 Token 需要提供应用的 App Key 和 App Secret。客户端如果保存这些凭证，一旦被反编译，会导致应用的 App Key 和 App Secret 泄露。所以，请务必确保在应用服务端获取 Token。**

连接 IM 服务器

请根据应用的业务需求与设计，自行决定合适的时机（登陆、注册、或其他时机以免无法进入应用主页），向融云聊天服务器发起连接请求。

方法

```
Future<int> connect(String token, int timeout, {RCIMIWConnectCallback? callback});
```

参数说明

参数名	参数类型	描述
token	String	调用 server api 获取到的 token
timeout	int	连接超时时间，单位：秒。

参数名	参数类型	描述
callback	RCIMIWConnectCallback	链接事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
RCIMIWConnectCallback? callback = RCIMIWConnectCallback(onDatabaseOpened: (int? code) {
//...
}, onConnected: (int? code, String? userId) {
//...
});

int? ret = await engine?.connect(token, timeout, callback:callback);
```

回调方法

- **onDatabaseOpened**

数据库打开的回调

```
Function(int? code)? onDatabaseOpened;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常

代码示例

```
engine?.onDatabaseOpened = (int? code) {
//...
};
```

- **onConnected**

收到连接状态的回调

```
Function(int? code, String? userId)? onConnected;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userId	String	链接成功的用户 ID

代码示例

```
engine?.onConnected = (int? code, String? userId) {
//...
};
```

> 提示
>
> 1. SDK 本身有重连机制，在一个应用生命周期内不须多次调用 `connect()` 。否则可能触发多个回调，触发导致回调被清除。
> 2. 在 `code` 为 **31004** 的情况下，您需要请求您的服务器重新获取 Token 并建立连接，但是注意避免无限循环，以免影响 App 用户体验。

token 无效(31004)排查方案

原因	排查方案
token 错误	检查客户端初始化使用的 AppKey 和您服务器获取 Token 使用的 AppKey 是否一致
token 过期	检查开发者是否在 控制台 设置了 Token 过期时间，过期之后需要请求您的服务器重新获取 Token 并再次用新的 Token 建立连接

错误码

连接错误码参见原生平台错误码定义 Android 错误码 和 iOS 错误码

断开连接

更新时间:2024-08-30

开发者在连接融云服务之后，在需要用户切换、用户注销的操作时，可通过下面方法断开连接，并可根据此方法来指定在用户断开连接之后是否接收消息推送。

> 提示
>
> SDK 在前后台切换或者网络出现异常都会自动重连，会保证连接的可靠性。除非 App 逻辑需要登出，否则不需要调用此方法进行手动断开。

方法

```
Future<int> disconnect(bool receivePush);
```

参数说明

参数名	参数类型	描述
receivePush	bool	退出后是否接收 push，true:断开后接收远程推送，false:断开后不再接收远程推送

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
int? ret = await engine?.disconnect(receivePush);
```

重连机制与重连互踢　　自动重连机制

更新时间:2024-08-30

SDK 内已实现自动重连机制，一旦连接成功，SDK 的重连机制将立即开始生效，并接管所有的重连处理。当因为网络原因断线时，SDK 内部会尝试重新建立连接，不需要您做额外的连接操作。

可能导致 SDK 断线重连的异常情况如下：

- **弱网环境**：可能出现 SDK 不停重连的情况。因为客户端 SDK 和融云服务端之间存在连接保活机制，一旦因如果网络太差导致心跳超时，SDK 就会触发重连操作，尝试重连直到连接成功。
- **无网环境**：SDK 的重连机制会暂停。一旦网络恢复，SDK 会进行重连操作。

> 提示
>
> 一旦触发连接错误的回调，SDK 将退出重连机制。请根据具体的状态码自行处理。

重连时间间隔

SDK 尝试重连时，时间间隔逐次变大，分别是 0s, 0.25s, 0.5s, 1s, 2s, 4s, 8s, 16s, 32s。之后每 64s 重试一次。

当 APP 切换到前台或者网络状态发生变化，重连时间会按照上面的时间间隔从头开始，保证这种情况下能尽快的连接成功。

主动退出重连机制

应用主动断开连接后，SDK 将退出重连机制，不再尝试重连。

重连互踢策略

重连互踢策略用于控制 SDK 自动重连成功时是否需要下线的设备。

即时通讯业务默认仅允许同一用户账号在单台移动端设备上登录。后登录的移动端设备一旦连接成功，则自动踢出之前登录的设备。在部分情况下，SDK 的重连机制可能会导致后登录设备无法正常在线。

例如，默认的重连互踢策略可能导致以下情况：

1. 用户张三尝试在移动设备 A 上登录，但因 A 设备网络不稳定导致未连接成功，触发了 SDK 的自动重连机制。
2. 用户此时尝试换到移动设备 B 上登录。B 设备连接成功，且用户可通过 B 设备正常使用即时通讯业务。
3. A 设备网络稳定之后，SDK 重连成功。因此时 A 设备为后上线设备，导致 B 设备被踢出。

修改 App 用户的重连互踢策略

如果 App 用户希望在以上场景中将重连成功的 A 设备下线，同时保持 B 设备登录，可以修改当前用户（User ID）的**重连互踢策略**。

> 提示
>
> **使用该接口要求该 App Key 已启用登录时判断用户其他端登录状态提示（移动端）。如需启用该服务，请提交工单。**

设置断线重连时是否踢出重连设备。该配置属于引擎配置，此方法需要在引擎初始化之前设置。

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
options.kickReconnectDevice = true;
RCIMIWEngine engine = await RCIMIWEngine.create(appKey, options);
```

参数说明

参数	类型	说明
kickReconnectDevice	bool	是否踢出正在重连的设备。详见下方 `kickReconnectDevice` **参数详细说明**。

`kickReconnectDevice` **参数详细说明**：

- 设置为 `true`：

  如果重连时发现已有别的移动端设备在线，将不再重连，不影响已正常登录的移动端设备。

- 设置为 `false`：

  如果重连时发现已有别的移动端设备在线，将踢出已在线的移动端设备，使当前设备上线。

多端同时在线

更新时间:2024-08-30

多端同时在线是指同一用户账号从多个平台同时连接到融云即时通讯服务的功能。默认情况下，融云即支持多端设备同时在线。该功能无需开通即可使用。

> 默认多端设备之间不会进行消息同步。如有需要，请开通多设备消息同步服务。

多端登录限制说明

默认的情况下，同一用户账号可在移动端、Web 端、桌面端、小程序端最多一个设备上同时在线。

平台类别	限制	IM SDK 支持平台列表
移动端	默认仅支持一个移动端设备连接。如需支持移动端多设备登录，请提交工单申请开通移动多端服务。	Android、iOS、Flutter、React Native、uni-app、Unity
桌面端	默认仅支持一个桌面端设备连接。	Electron 框架（通过 Web 端 SDK 的 Electron 模块支持）
Web 端	默认仅支持一个 Web 页面连接（每个浏览器标签页认为是一个连接）。在控制台自助开通多设备消息同步服务后，自动支持多 Web 页面连接。	Web
小程序端	默认仅支持一个小程序连接。如需支持小程序多设备登录，请提交工单申请开通小程序多端服务。	小程序

多设备消息同步

更新时间:2024-08-30

在即时通讯业务中，同一用户账号可能在多个设备上登录。**多设备消息同步**是融云服务端提供的一项服务，可用于同一用户账号的多个设备之间同步收发消息。

默认情况下，融云不会在设备之间同步消息。新消息被某一端设备收取后，其他端无法收取该消息。

适用场景

在融云即时通讯业务中，多设备消息同步适用于以下情况：

- 同一用户账号在多设备上同时在线（无论是否为同一端），希望同步收发消息。例如，用户可能拥有多个移动端设备，如两个 Android 设备、一个 iOS 设备。

  > 注意：融云默认已支持多端同时在线，同一用户账号可在移动端、Web 端、桌面端、小程序端最多一个设备上同时在线。但是如果需要允许 App 用户同时在多个移动端设备或多个小程序端上在线，需要分别提交工单申请，详见多端同时在线。

- 同一用户账号换设备登录（无论是否曾在该设备登录过），希望同步收发的消息记录。例如用户从 Android 设备下线后，换到另一个设备从 Web 端登录。
- 同一用户账号在当前设备卸载重装 App，希望同步收发消息记录。

支持在多设备间同步的消息

并非所有消息均支持多设备消息同步。**状态消息**仅支持在多设备同时在线时同步接收，不在线的设备无法通过**多设备消息同步**收到消息。

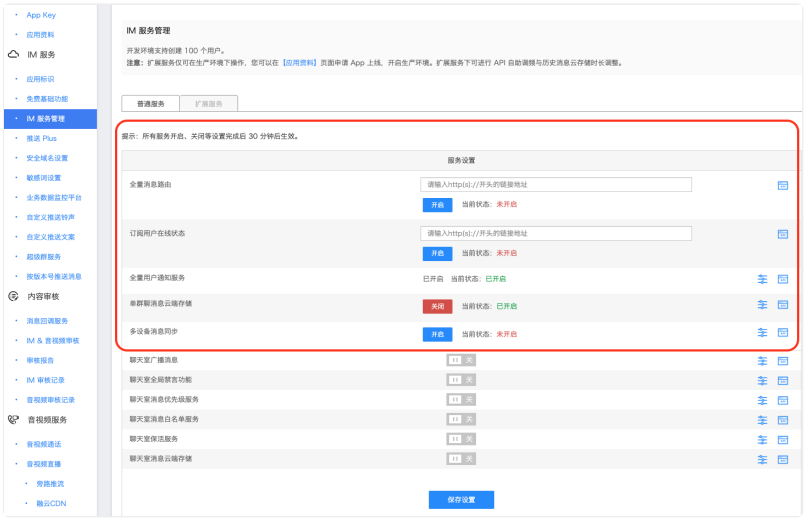
以下情况均属于状态消息：

- 融云内置消息类型中定义为状态消息类型的消息。内置状态类型消息的具体包括：正在输入状态消息（`RC:TypSts`）
- 自定义的状态消息类型的消息。详见各个客户端「自定义消息」文档。
- 使用服务端 API 状态消息接口发送的所有消息（不区分消息类型）均不支持同步。具体的 API 接口为发送单聊状态消息（`/statusmessage/private/publish.json`）、发送群聊状态消息（`/message/group/publish.json`）。

开通服务

请前往控制台，在 IM 服务管理 页面的**普通服务**标签下开通**多设备消息同步**服务。该服务在开发环境免费使用，默认为关闭状态。生产环境预存费用后才可开通服务。

> 服务开启、关闭设置完成后 30 分钟内生效。



对其他功能或业务的影响

多设备消息同步服务的状态对即时通讯业务中的离线补偿[?]、撤回消息、聊天室业务等有影响。

对离线补偿的影响

控制台的多设备消息同步服务包含了融云服务端离线补偿机制[?]的开关。

开通多设备消息同步服务后，融云服务端自动为 App 启用离线补偿机制。离线补偿机制会在以下场景触发：

- 换设备登录。用户在新设备上登录后（无论是否曾在该设备登录过），SDK 可获取最近指定天数（默认离线补偿天数为 1 个自然日）在其他终端上发送和接收过的消息。
- 应用卸载重装。消息与会话列表是存储在本地数据库，用户卸载 App 时会删除本地数据库。用户重新安装 App 后并再次连接时，会触发融云服务端离线补偿机制，SDK 可获取最近指定天数（默认离线补偿天数为 1 个自然日）在其他终端上发送和接收过的消息。

> 注意：在 换设备登录 或 应用卸载重装 场景下，离线补偿机制仅可获取到最近（默认离线补偿天数为 1 天，最大 7 天）的会话。早于该天数的会话无法通过离线补偿机制获取。因此，离线补偿后的会话列表可能与原设备上或卸载前的会话列表并不一致（您可能会有丢失部分会话的错觉）。

如需修改离线消息补偿的天数，可提交工单。建议谨慎设置离线补偿天数，当单用户消息量超大时，可能会因为补偿消息量过大，造成端上处理压力较大。

对 Web 平台连接数的影响

开通多设备消息同步服务后，可额外支持多 Web 页面连接（每个浏览器标签页也认为是一个连接），最多 10 个。

对撤回消息的影响

- 未开通多设备消息同步服务时，多端之间无法同步撤回的消息。
- 开通多设备消息同步服务后，消息发送端一旦撤回消息，如果用户在其他端在线，则其他端同步撤回该条已发送消息。如果用户在其他端不在线时，则在用户登录后同步撤回已发送的消息。

对服务端 API 发送消息的影响

通过服务端 API 发送消息时，部分接口可通过 `isIncludeSender` 指定消息发送者可否在客户端接收该已发消息。

- 未开通多设备消息同步服务时，仅在发送者已登录客户端（在线）的情况下，通过 Server API 发送的消息可即时同步到发送者的在线客户端，无法同步到离线的客户端。
- 开通多设备消息同步服务后，如果发送者未登录客户端（离线），通过 Server API 发送的消息可在再次上线时同步到发送者的在线客户端。

用户概述

更新时间:2024-08-30

App 用户需要接入融云服务，才能使用即时通讯服务。对于融云来说，用户是指持有由融云分发的有效 Token，接入并使用即时通讯服务的 App 用户。

注册用户

应用服务端（App Server）应向融云服务端提供 App 用户的用户 ID（`userId`），以向融云换取唯一用户 Token。对融云来说，这个以 `userId` 获取 Token 的步骤即注册用户，且必须通过调用 Server API 来完成。

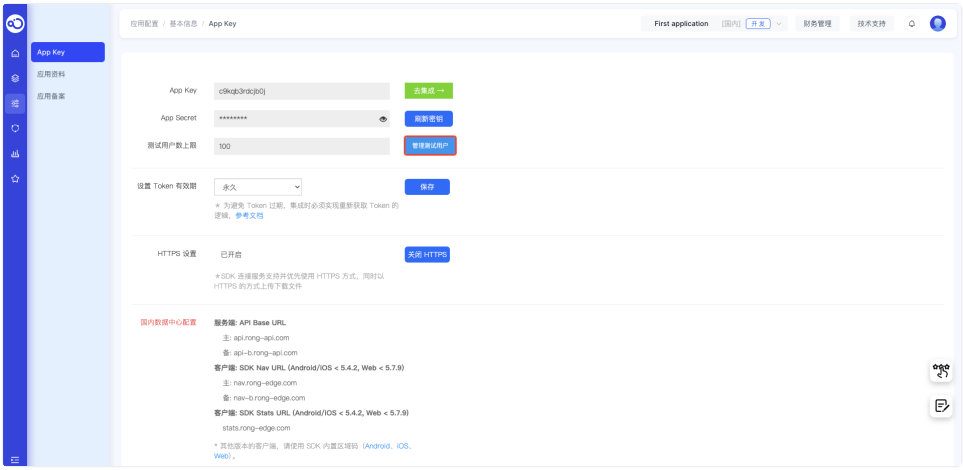
应用客户端必须持有有效 Token，才能成功连接到融云服务端，使用融云即时通讯服务。当 App 客户端用户向服务器发送登录请求时，服务器会查询数据库以检查连接请求是否匹配。

注册用户数限制

- 开发环境中的注册用户数上限为 100 个。
- 在 生产环境中，升级为 **IM 旗舰版**或 **IM 尊享版**后不限制注册用户数。

删除用户

删除用户是指在应用的开发环境中，通过控制台删除已注册的测试用户，以控制开发环境中的测试用户总数。生产环境不支持该操作。



注销用户

注销用户是指在融云服务中删除用户数据。App 可使用该能力实现自身的用户销户功能，满足 App 上架或合规要求。

融云返回注销成功结果后，与用户 ID 相关数据即被删除。您可以向融云查询所有已注销用户的 ID。如有需要，您可以重新激活已被注销的用户 ID（注意，用户个人数据无法被恢复）。

仅 IM Server API 提供上述能力。

用户信息

用户信息泛指用户的昵称、头像，以及群组的群昵称、群头像等数据。融云服务端不提供用户信息托管维护服务。

IMLib SDK 中未提供与用户信息相关的客户端接口。

好友关系

App 用户之间的好友关系需要由应用服务器（App Server）自行维护。融云不会同步或保存 App 端的好友关系数据。

如果需要对客户端用户之间的消息收发行为进行限制（例如，App 的所有 `userId` 泄漏，导致某个恶意用户可越过好友关系向任意用户发送消息），可以考虑使用用户白名单服务。用户一旦开启并设置白名单，则仅可接收来自该白名单中用户的消息。

用户管理接口

功能分类	功能描述	客户端 API	服务端 API
注册用户	使用 App 用户的用户 ID 向融云换取 Token。	不提供该 API	注册用户
删除用户	参见上文删除用户。	不提供该 API	不提供该 API
废弃 Token	废弃在特定时间点之前获取的 Token。	不提供该 API	作废 Token
注销用户	注销用户是指在融云服务中停用用户 ID，并删除用户个人数据。	不提供该 API	注销用户
查询已注销用户	获取已注销的用户 ID 列表。	不提供该 API	查询已注销用户
重新激活用户 ID	在融云服务中重新启用已注销用户的 ID。	不提供该 API	重新激活用户 ID
设置融云服务端的用户信息	设置在融云推送服务中使用的用户名称与头像。	不提供该 API	未提供单独的设置接口。在注册用户时必须提供用户信息。
获取融云服务端的用户信息	获取用户在融云注册的信息，包括用户创建时间和服务端的推送服务使用的用户名称、头像 URL。	不提供该 API	获取信息
修改融云服务端的用户信息	修改在融云推送服务中使用的用户名称与头像。	不提供该 API	修改信息
封禁用户	禁止用户连接到融云即时通讯服务，并立即断开连接。可按时长解封或主动解封。查询被封禁用户的用户 ID、封禁结束时间。	不提供该 API	添加封禁用户、解除封禁用户、查询封禁用户
查询用户在线状态	查询某用户的在线状态。	不提供该 API	查询在线状态
加入黑名单	在用户的黑名单列表中添加用户。在 A 用户黑名单的用户无法向 A 发送消息。	加入黑名单	加入黑名单
移出黑名单	在用户的黑名单中移除用户。	移出黑名单	移出黑名单

功能分类	功能描述	客户端 API	服务端 API
查询黑名单	查询某用户（userId）是否已被当前用户加入黑名单。	查询用户是否在黑名单中	不提供该 API
获取黑名单列表	获取用户的黑名单列表。	获取黑名单列表	查询黑名单
用户白名单	用户一旦开启并设置白名单，则仅可接收来自该白名单中用户的消息。	不提供该 API	开启用户白名单、用户白名单状态查询、添加白名单、移出白名单、查询白名单

黑名单管理

更新时间:2024-08-30

将用户加入黑名单之后，将不再收到对方发来的任何单聊消息。

- 加入黑名单为单向操作，例如：用户 A 拉黑用户 B，代表 B 无法给 A 发消息（错误码 `405`）。但 A 向 B 发消息，B 仍然能正常接收。
- 单个用户的黑名单总人数存在上限，具体与计费套餐有关。IM 旗舰版与 IM 尊享版上限为 3000 人，其他套餐详见功能对照表中的服务限制。
- 调用服务端 API 发送单聊消息默认不受黑名单限制。如需启用限制，请在调用 API 时设置 `verifyBlacklist` 为 `1`。

加入黑名单

将某个用户加入黑名单。

方法

```
Future<int> addToBlacklist(String userId, {IRCIMIWAddToBlacklistCallback? callback});
```

参数说明

参数名	参数类型	描述
userId	String	用户 Id
callback	IRCIMIWAddToBlacklistCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

回调方法

- **onBlacklistAdded**

```
Function(int? code, String? userId)? onBlacklistAdded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userId	String	用户 ID

代码示例

```
engine?.onBlacklistAdded = (int? code, String? userId) {
//...
};
```

移除黑名单

将某个用户从黑名单中移出。

方法

```
Future<int> removeFromBlacklist(String userId, {IRCIMIWRemoveFromBlacklistCallback? callback});
```

参数说明

参数名	参数类型	描述
userId	String	用户 Id
callback	IRCIMIWRemoveFromBlacklistCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

回调方法

- **onBlacklistRemoved**

```
Function(int? code, String? userId)? onBlacklistRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userId	String	用户 ID

代码示例

```
engine?.onBlacklistRemoved = (int? code, String? userId) {
//...
};
```

查询用户是否在黑名单中

获取某用户是否在黑名单中。

方法

```
Future<int> getBlacklistStatus(String userId, {IRCIMIWGetBlacklistStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
userId	String	用户 Id
callback	IRCIMIWGetBlacklistStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

回调方法

- **onBlacklistStatusLoaded**

```
Function(int? code, String? userId, RCIMIWBlacklistStatus? status)? onBlacklistStatusLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userId	String	用户 ID
status	RCIMIWBlacklistStatus	当前状态

代码示例

```
engine?.onBlacklistStatusLoaded = (int? code, String? userId, RCIMIWBlacklistStatus? status) {
//...
};
```

获取黑名单列表

获取当前用户设置的黑名单列表。

方法

```
Future<int> getBlacklist({IRCIMIWGetBlacklistCallback? callback});
```

返回值

参数名	参数类型	描述
callback	IRCIMIWGetBlacklistCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

回调方法

- **onBlacklistLoaded**

```
Function(int? code, List<String>? userIds)? onBlacklistLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userIds	List<String>	用户 ID 集合

代码示例

```
engine?.onBlacklistLoaded = (int? code, List<String>? userIds) {
//...
};
```

消息介绍　　消息基本结构

更新时间:2024-08-30

属性名	类型	说明
conversationType	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	所属会话的业务标识，长度限制 20 字符，仅在超级群下生效
messageType	RCIMIWMessageType	消息的类型
messageId	int	本地存储的消息的唯一值（数据库索引唯一值）
messageUId	String	服务器消息唯一 ID（在同一个 Appkey 下全局唯一）
direction	RCIMIWMessageDirection	消息的方向
receivedTime	int	消息的接收时间（Unix 时间戳、毫秒）
sentTime	int	消息的发送时间（Unix 时间戳、毫秒）
receivedStatus	RCIMIWReceivedStatus	消息的接收状态
sentStatus	RCIMIWSentStatus	消息的发送状态
senderUserId	String	消息的发送者 ID
expansion	Map	消息扩展信息列表，该属性在消息发送时确定，发送之后不能再做修改。 扩展信息只支持单聊和群组，其它会话类型不能设置扩展信息。 默认消息扩展字典 key 长度不超过 32 ，value 长度不超过 4096 ，单次设置扩展数量最大为 20，消息的扩展总数不能超过 300
offLine	bool	是否是离线消息，只在接收消息的回调方法中有效，如果消息为离线消息，则为 YES ，其他情况均为 NO
groupReadReceiptInfo	RCIMIWGroupReadReceiptInfo	群阅读回执状态
userInfo	RCIMIWUserInfo	消息携带的用户信息
mentionedInfo	RCIMIWMentionedInfo	消息的 @ 信息
pushOptions	RCIMIWMessagePushOptions	消息推送配置，仅可在发送方进行配置，接收方为空
extra	String	消息的附加字段，设置后接收方收到消息后也可以拿到数据。

消息类型与构造方法

IMLib 的消息类型（RCIMIWMessageType）枚举提供了预定义的文本、语音、图片等消息类型，并且支持 App 通过 `RCIMIWMessageType.custom` 实现自定义消息。

各类型消息默认会在客户端本地存储，计入未读消息数。以下类型除外：

- 撤回消息类型（`RCIMIWMessageType.recall`）：客户端本地存储，但不计入未读消息数
- 命令消息类型（`RCIMIWMessageType.command`）：客户端不进行本地存储，不计入未读消息数

- 命令通知消息类型（`RCIMIWMessageType.commandNotification`）：客户端本地存储，不计入未读消息数。
- 自定义消息类型（`RCIMIWMessageType.custom`）：由 App 自定义存储策略（RCIMIWCustomMessagePolicy）进行控制。

文本消息

类型：RCIMIWMessageType.text

专有可访问属性：

属性名	类型	说明
text	String	文本内容

构建文本消息

```
RCIMIWTextMessage? message = await engine.createTextMessage(
type,
targetId,
channelId,
text,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型，
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
text	String	文本内容

语音消息

类型：RCIMIWMessageType.voice

专有可访问属性：

属性名	类型	说明
duration	int	语音的长度，单位：秒

构建语音消息

```
RCIMIWVoiceMessage? message = await engine.createVoiceMessage(
type,
targetId,
channelId,
path,
duration,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
path	String	语音消息的本地路径，必须为有效路径
duration	int	语音消息的消息时长

图片消息

类型：`RCIMIWMessageType.image`

专有可访问属性：

属性名	类型	说明
thumbnailBase64String	String	图片的缩略图数据
original	bool	是否为原图

构建图片消息

```
RCIMIWImageMessage? message = await engine.createImageMessage(
type,
targetId,
channelId,
path,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
path	String	图片消息的本地路径，必须为有效路径

文件消息

类型：`RCIMIWMessageType.file`

专有可访问属性：

属性名	类型	说明
name	String	文件名
fileType	String	文件类型
size	int	文件大小，单位为 Byte

构建文件消息

```
RCIMIWFileMessage? message = await engine.createFileMessage(
type,
targetId,
channelId,
path,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
path	String	文件消息的本地路径，必须为有效路径

小视频消息

类型：RCIMIWMessageType.sight

专有可访问属性：

属性名	类型	说明
duration	int	视频时长
size	int	视频大小
name	String	视频的名称
thumbnailBase64String	String	缩略图数据

构建小视频消息

```
RCIMIWSightMessage? message = await engine.createSightMessage(
type,
targetId,
channelId,
path,
duration,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
path	String	小视频消息的本地路径，必须为有效路径
duration	int	小视频消息的视频时长

GIF 图消息

类型：RCIMIWMessageType.gif

专有可访问属性：

属性名	类型	说明
dataSize	int	GIF 图的大小，单位字节
width	int	GIF 图的宽
height	int	GIF 图的高

构建 GIF 消息

```
RCIMIWGIFMessage? message = await engine.createGIFMessage(
type,
targetId,
channelId,
path
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
path	String	GIF 消息的本地路径

撤回消息

撤回消息不可以主动构建发送，需要撤回消息请调用撤回消息接口

类型：`RCIMIWMessageType.recall`

专有可访问属性：

属性名	类型	说明
admin	bool	是否是管理员操作
deleted	bool	是否删除
recallTime	int	被撤回的原始消息的发送时间（毫秒）
recallActionTime	int	撤回动作的时间（毫秒）
originalMessage	RCIMIWMessage	被撤回的原消息

引用消息

类型：`RCIMIWMessageType.reference`

专有可访问属性：

属性名	类型	说明
text	String	引用文本

属性名	类型	说明
referenceMessage	RCIMIWMessage	被引用的消息

构建引用消息

```
RCIMIWReferenceMessage? message = await engine.createReferenceMessage(
type,
targetId,
channelId,
message,
text,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
referenceMessage	RCIMIWMessage	引用的消息
text	String	引用的文本内容

位置消息

类型：RCIMIWMessageType.location

专有可访问属性：

属性名	类型	说明
longitude	double	经度信息
latitude	double	纬度信息
poiName	String	POI 信息
thumbnailPath	String	缩略图地址

构建位置消息

```
RCIMIWLocationMessage? message = await engine?.createLocationMessage(
type,
targetId,
channelId,
double.parse(longitude),
double.parse(latitude),
poiName,
path,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
longitude	double	经度
latitude	double	纬度
poiName	String	POI 信息
thumbnailPath	String	缩略图本地路径，必须为有效路径

命令消息

类型：`RCIMIWMessageType.command`

专有可访问属性：

属性名	类型	说明
name	String	命令的名称
data	String	命令的扩展数据，可以为任意字符串，如存放您定义的json数据。

Flutter SDK 目前不提供 RCIMIWCommandMessage 命令消息的 create 方法，仅支持接收该类型消息。如需要在客户端发送此消息，需要在 Flutter 层先定义此消息的包装类。详见知识库文档Flutter SDK 如何发送命令消息和命令通知消息？。

命令通知消息

类型：`RCIMIWMessageType.commandNotification`

专有可访问属性：

属性名	类型	说明
name	String	命令提醒的名称
data	String	命令提醒消息的扩展数据，可以为任意字符串，如存放您定义的 json 数据。

Flutter SDK 目前不提供 RCIMIWCommandNotificationMessage 命令通知消息的 create 方法，仅支持接收该类型消息。如需要在客户端发送此消息，需要在 Flutter 层先定义此消息的包装类。详见知识库文档Flutter SDK 如何发送命令消息和命令通知消息？。

自定义消息

类型：`RCIMIWMessageType.custom`

专有可访问属性：

属性名	类型	说明
identifier	String	自定义消息的标识符
policy	RCIMIWCustomMessagePolicy	自定义的消息存储策略
fields	Ma<String, String>	自定义消息的键值对

构建自定义消息

```
RCIMIWCustomMessage? message = await engine.createCustomMessage(
type,
targetId,
channelId,
policy,
messageIdentifier,
fields,
);
```

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
policy	RCIMIWCustomMessagePolicy	消息的存储策略
messageIdentifier	String	消息的标识符，需唯一
fields	Ma<String, String>	消息的内容键值对

无效类型

当接收到无法识别的消息时，会变为无效类型的消息，此消息不可主动构建发送

类型：`RCIMIwMessageType.unknown`

专有可访问属性：

属性名	类型	描述
rawData	String	消息数据
objectName	String	消息的标识

发送消息

更新时间:2024-08-30

本文介绍了 IMLib 如何发送消息。

客户端 SDK 发送消息存在频率限制，每秒最多只能发送 5 条消息。

发送普通消息

开发者在发送文本消息、引用消息、自定义消息时，可以使用下面接口发送。

方法

```
Future<int> sendMessage(RCIMIWMessage message, {RCIMIWSendMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	发送的消息实体
callback	RCIMIWSendMessageCallback	发送消息的事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
RCIMIWSendMessageCallback? callback = RCIMIWSendMessageCallback(onMessageSaved: (RCIMIWMessage? message)
{
//...
}, onMessageSent: (int? code, RCIMIWMessage? message) {
//...
});

int? ret = await engine?.sendMessage(message, callback:callback);
```

回调方法

- **onMessageAttached**

消息存入数据库的监听

```
Function(RCIMIWMessage? message)? onMessageAttached;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	发送的消息

代码示例

```
engine?.onMessageAttached = (RCIMIWMessage? message) {
//...
};
```

- **onMessageSent**

消息发送结果的监听

```
Function(int? code, RCIMIWMessage? message)? onMessageSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	发送的消息

代码示例

```
engine?.onMessageSent = (int? code, RCIMIWMessage? message) {
//...
};
```

发送媒体消息

开发者在发送图片消息、语音消息、视频消息、文件消息、GIF 消息时，可以使用下面接口发送。

方法

```
Future<int> sendMediaMessage(RCIMIWMediaMessage message, {RCIMIWSendMediaMessageListener? listener});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	发送的媒体消息实体
listener	RCIMIWSendMediaMessageListener	发送媒体消息的事件监听

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
RCIMIWSendMediaMessageListener? listener = RCIMIWSendMediaMessageListener(onMediaMessageSaved:
(RCIMIWMediaMessage? message) {
//...
}, onMediaMessageSending: (RCIMIWMediaMessage? message, int? progress) {
//...
}, onSendingMediaMessageCanceled: (RCIMIWMediaMessage? message) {
//...
}, onMediaMessageSent: (int? code, RCIMIWMediaMessage? message) {
//...
});

int? ret = await engine?.sendMediaMessage(message, listener:listener);
```

回调方法

- **onMediaMessageAttached**

  消息存入数据库的监听

```
Function(RCIMIWMediaMessage? message)? onMediaMessageAttached;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	发送的消息

代码示例

```
engine?.onMediaMessageAttached = (RCIMIWMediaMessage? message) {
//...
};
```

- **onMediaMessageSending**

  消息发送进度的监听

```
Function(RCIMIWMediaMessage? message, int? progress)? onMediaMessageSending;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	发送的消息
progress	int	发送的进度

代码示例

```
engine?.onMediaMessageSending = (RCIMIWMediaMessage? message, int? progress) {
//...
};
```

- **onMediaMessageSent**

  消息发送结果的监听

```
Function(int? code, RCIMIWMediaMessage? message)? onMediaMessageSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMediaMessage	发送的消息

代码示例

```
engine?.onMediaMessageSent = (int? code, RCIMIWMediaMessage? message) {
//...
};
```

接收消息 设置消息监听

更新时间:2024-08-30

1. 设置消息接收监听器。所有接收到的消息都会在此接口方法中回调。
2. 建议在应用生命周期内注册消息监听。
3. 不支持设置多个监听器。为了避免内存泄露，请在不需要监听的时候将监听器置空。

方法

```
Function(RCIMIWMessage? message, int? left, bool? offline, bool? hasPackage)? onMessageReceived;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	接收到的消息对象
left	int	当客户端连接成功后，服务端会将所有补偿消息以消息包的形式下发给客户端，最多每 200 条消息为一个消息包，即一个 Package. 客户端接受到消息包后，会逐条解析并通知应用。left 为当前消息包（Package）里还剩余的消息条数
offline	bool	消息是否离线消息
hasPackage	bool	是否在服务端还存在未下发的消息包

代码示例

```
engine?.onMessageReceived = (RCIMIWMessage? message, int? left, bool? offline, bool? hasPackage) {
//...
};
```

注意事项

- 接收消息注意:
  1. 针对接收离线消息时，服务端会将 200 条消息打成一个包发到客户端，客户端对这包数据进行解析。
  2. hasPackage 标识是否还有剩余的消息包，left 标识这包消息解析完逐条抛送给 App 层后，剩余多少条。
- 如何判断离线消息收完：
  1. hasPackage 为 false 和 left 为 0。
  2. hasPackage 为 false 标识当前正在接收最后一包（200条）消息，left 为 0 标识最后一包的最后一条消息也已接收完毕。

获取历史消息　　开通服务

更新时间:2024-08-30

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。

提示：请注意区分历史消息记录与离线消息[?]。融云针对单聊、群聊、系统消息默认提供最多 7 天（可调整）的离线消息缓存服务。客户端上线时 SDK 会自动收取离线期间的消息，无需 App 层调用 API。详见管理离线消息存储配置。

获取历史消息

开发者可以通过此接口来获取某个会话的历史消息。

方法

```
Future<int> getMessages(RCIMIWConversationType type, String targetId, String? channelId, int sentTime,
RCIMIWTimeOrder order, RCIMIWMessageOperationPolicy policy, int count, {IRCIMIWGetMessagesCallback?
callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
sentTime	int	当前消息时间戳
order	RCIMIWTimeOrder	获取消息的方向。BEFORE：获取 sentTime 之前的消息 （时间递减），AFTER：获取 sentTime 之后的消息 （时间递增）
policy	RCIMIWMessageOperationPolicy	消息的加载策略。LOCAL：只加载本地，REMOTE：只加载远端，LOCAL_REMOTE：本地远端都加载
count	int	获取的消息数量，0 < count <= 20
callback	IRCIMIWGetMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetMessagesCallback? callback = IRCIMIWGetMessagesCallback(onSuccess: (List<RCIMIWMessage>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getMessages(type, targetId, channelId, sentTime, order, policy, count,
callback:callback);
```

回调方法

- **onMessagesLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? sentTime,
RCIMIWTimeOrder? order, List<RCIMIWMessage>? messages)? onMessagesLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
sentTime	int	当前消息时间戳
order	RCIMIWTimeOrder	获取消息的方向。BEFORE：获取 sentTime 之前的消息 （时间递减），AFTER：获取 sentTime 之后的消息 （时间递增）
messages	List<RCIMIWMessage>	获取到的消息集合

代码示例

```
engine?.onMessagesLoaded = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, int? sentTime, RCIMIWTimeOrder? order, List<RCIMIWMessage>? messages) {
//...
};
```

通过 messageId 获取消息

开发者可以通过此接口来获取某条消息。

方法

```
Future<int> getMessageById(int messageId, {IRCIMIWGetMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
messageId	int	消息的 messageId，可在消息对象中获取
callback	IRCIMIWGetMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetMessageCallback? callback = IRCIMIWGetMessageCallback(onSuccess: (RCIMIWMessage? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getMessageById(messageId, callback:callback);
```

通过 messageUId 获取消息

开发者可以通过此接口来获取某条消息。

方法

```
Future<int> getMessageByUId(String messageUId, {IRCIMIWGetMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值。
callback	IRCIMIWGetMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetMessageCallback? callback = IRCIMIWGetMessageCallback(onSuccess: (RCIMIWMessage? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getMessageByUId(messageUId, callback:callback);
```

插入消息　　插入单条消息

更新时间:2024-08-30

- 通过该功能在本地会话中插入一条消息。
- 插入的消息必须是可存储消息，否则报参数错误异常。
- 消息插入功能插入的消息仅支持配置会话类型、会话 ID、频道 ID、消息的发送方向、消息的发送时间以及各类消息独有的可访问属性。
- 默认不配置的情况下插入的是一条发送方的已经发送成功的消息

方法

```
Future<int> insertMessage(RCIMIWMessage message, {IRCIMIWInsertMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	插入的消息
callback	IRCIMIWInsertMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWInsertMessageCallback? callback = IRCIMIWInsertMessageCallback(onMessageInserted: (int? code,
RCIMIWMessage? message) {
//...
});

int? ret = await engine?.insertMessage(message, callback:callback);
```

回调方法

- **onMessageInserted**

  接口调用结果的监听

```
Function(int? code, RCIMIWMessage? message)? onMessageInserted;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	插入的消息

代码示例

```
engine?.onMessageInserted = (int? code, RCIMIWMessage? message) {
//...
};
```

插入多条消息

- 通过该功能在本地会话中插入多条消息。
- 插入的消息必须是可存储消息，否则报参数错误异常。
- 消息插入功能插入的消息仅支持配置会话类型、会话 ID、频道 ID、消息的发送方向、消息的发送时间以及各类消息独有的可访问属性。
- 默认不配置的情况下插入的是一条发送方的已经发送成功的消息
- 建议每次最多传入 10 条，当有一条失败时，所有的消息都会插入失败

方法

```
Future<int> insertMessages(List<RCIMIWMessage> messages, {IRCIMIWInsertMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage>	插入的消息集合
callback	IRCIMIWInsertMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWInsertMessagesCallback? callback = IRCIMIWInsertMessagesCallback(onMessagesInserted: (int? code,
List<RCIMIWMessage>? messages) {
//...
});

int? ret = await engine?.insertMessages(messages, callback:callback);
```

回调方法

- **onMessagesInserted**

  接口调用结果的监听

```
Function(int? code, List<RCIMIWMessage>? messages)? onMessagesInserted;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messages	List<RCIMIWMessage>	插入的消息集合

代码示例

```
engine?.onMessagesInserted = (int? code, List<RCIMIWMessage>? messages) {
//...
};
```

删除消息

更新时间:2024-08-30

单聊会话、群聊会话的参与者可删除会话中的消息。您可以删除指定的一条或一组消息，删除某个会话的全部消息历史，或删除会话中早于指定时间点的消息。

消息送达后会直接存入本地数据库，您可以调用接口从本地数据库中删除消息。消息仅从本地数据库中删除，保存在服务端的消息将不受影响。如果您已经开通历史消息云存储服务，仍然可以从服务端拉取到历史消息。

> 提示
>
> - App 用户的单聊会话、群聊会话、系统会话的消息默认仅存储在本地数据库中，仅支持从本地删除。如果 App（App Key/环境）已开通单群聊消息云端存储，该用户的消息还会保存在融云服务端（默认 6 个月），可从远端历史消息记录中删除消息。
> - 针对单聊会话、群聊会话，如果通过任何接口以传入时间戳的方式删除远端消息，服务端默认不会删除对应的离线消息补偿（该机制仅会在打开多设备消息同步开关后生效）。此时如果换设备登录或卸载重装，仍会因为消息补偿机制获取到已被删除的历史消息。如需彻底删除消息补偿，请提交工单，申请开通删除服务端历史消息时同时删除多端补偿的离线消息。如果以传入消息对象的方式删除远端消息，则服务端一定会删除消息补偿中的对应消息。
> - 针对单聊会话、群聊会话，如果 App 的管理员或者某普通用户希望在所有会话参与者的历史记录中彻底删除一条消息，应使用撤回消息功能。消息成功撤回后，原始消息内容会在所有用户的本地与服务端历史消息记录中删除。
> - 客户端 SDK 不提供删除聊天室消息的接口。当前用户的聊天室本地消息在退出聊天室时会被自动删除。开通聊天室消息云端存储服务后，如需清除全部用户的聊天室历史消息，可使用服务端 API 清除消息。

删除指定消息集合

您可以从本地数据库中删除指定的一条或一组消息。这种方式仅需要提供待删除消息 ID 数组，不区分会话类型。

方法

```
Future<int> deleteLocalMessages(List<RCIMIWMessage> messages, {IRCIMIWDeleteLocalMessagesCallback?
callback});
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage>	消息集合

参数名	参数类型	描述
callback	IRCIMIWDeleteLocalMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWDeleteLocalMessagesCallback? callback =
IRCIMIWDeleteLocalMessagesCallback(onLocalMessagesDeleted: (int? code, List<RCIMIWMessage>? messages) {
//...
});

int? ret = await engine?.deleteLocalMessages(messages, callback:callback);
```

回调方法

- **onLocalMessagesDeleted**

```
Function(int? code, List<RCIMIWMessage>? messages)? onLocalMessagesDeleted;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messages	List<RCIMIWMessage>	删除的消息集合

代码示例

```
engine?.onLocalMessagesDeleted = (int? code, List<RCIMIWMessage>? messages) {
//...
};
```

删除某个会话的指定消息集合

此方法会将本地和远端的消息一起进行删除

> 提示
>
> 只有已开通单群聊消息云存储服务的应用才可以操作。您可以前往控制台开通服务。

方法

```
Future<int> deleteMessages(RCIMIWConversationType type, String targetId, String? channelId,
List<RCIMIWMessage> messages, {IRCIMIWDeleteMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
messages	List<RCIMIWMessage>	消息集合
callback	IRCIMIWDeleteMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWDeleteMessagesCallback? callback = IRCIMIWDeleteMessagesCallback(onMessagesDeleted: (int? code,
List<RCIMIWMessage>? messages) {
//...
});

int? ret = await engine?.deleteMessages(type, targetId, channelId, messages, callback:callback);
```

回调方法

- **onMessagesDeleted**

  接口调用结果的监听

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
List<RCIMIWMessage>? messages)? onMessagesDeleted;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。

参数名	参数类型	描述
messages	List<RCIMIWMessage>	删除的消息集合

代码示例

```
engine?.onMessagesDeleted = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, List<RCIMIWMessage>? messages) {
//...
};
```

通过时间戳清除某个会话的消息

> 提示
>
> 当删除远端消息时，只有已开通单群聊消息云存储服务的应用才可以操作。您可以前往控制台开通服务。

方法

```
Future<int> clearMessages(RCIMIWConversationType type, String targetId, String? channelId, int
timestamp, RCIMIWMessageOperationPolicy policy, {IRCIMIWClearMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	清除消息截止时间戳，0 <= recordTime <= 当前会话最后一条消息的 sentTime, 0 清除所有消息，其他值清除小于等于 recordTime 的消息
policy	RCIMIWMessageOperationPolicy	清除的策略
callback	IRCIMIWClearMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearMessagesCallback? callback = IRCIMIWClearMessagesCallback(onMessagesCleared: (int? code) {
//...
});

int? ret = await engine?.clearMessages(type, targetId, channelId, timestamp, policy, callback:callback);
```

回调方法

- **onMessagesCleared**

  接口调用结果的监听

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? timestamp)?
onMessagesCleared;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	时间戳

代码示例

```
engine?.onMessagesCleared = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, int? timestamp) {
//...
};
```

撤回消息 撤回消息

更新时间:2024-08-30

撤回指定消息，只有已发送成功的消息可被撤回。

默认情况下，融云对撤回消息的操作者不作限制。如需限制，可考虑以下方案：

- App 客户端自行限制撤回消息的操作者。例如，不允许 App 业务中的普通用户撤回他人发送的消息，允许 App 业务中的管理员角色撤回他人发送的消息。
- 如需避免用户撤回非本人发送的消息，可以提交工单 申请打开**IMLib SDK 只允许撤回自己发送的消息**。从融云服务端进行限制，禁止用户撤回非本人发送的消息。

方法

```
Future<int> recallMessage(RCIMIWMessage message, {IRCIMIWRecallMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	需要被撤回的消息
callback	IRCIMIWRecallMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRecallMessageCallback? callback = IRCIMIWRecallMessageCallback(onMessageRecalled: (int? code,
RCIMIWMessage? message) {
//...
});

int? ret = await engine?.recallMessage(message, callback:callback);
```

回调方法

- **onMessageRecalled**

  接口调用结果的监听

```
Function(int? code, RCIMIWMessage? message)? onMessageRecalled;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	撤回的消息

代码示例

```
engine?.onMessageRecalled = (int? code, RCIMIWMessage? message) {
//...
};
```

监听消息撤回事件

用于监听已接收的消息被撤回的事件。当接受到的某条消息被撤回时，会通过此监听器回调。

方法

```
Function(RCIMIWMessage? message)? onRemoteMessageRecalled;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	原本的消息会变为撤回消息

代码示例

```
engine?.onRemoteMessageRecalled = (RCIMIWMessage? message) {
//...
};
```

搜索消息　　根据关键字搜索

更新时间:2024-08-30

根据关键字搜索指定会话中的消息

方法

```
Future<int> searchMessages(RCIMIWConversationType type, String targetId, String? channelId, String
keyword, int startTime, int count, {IRCIMIWSearchMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
keyword	String	搜索的关键字
startTime	int	查询 beginTime 之前的消息， 传 0 时从最新消息开始搜索，从该时间往前搜索。
count	int	查询的数量，0 < count <= 50。
callback	IRCIMIWSearchMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSearchMessagesCallback? callback = IRCIMIWSearchMessagesCallback(onSuccess: (List<RCIMIWMessage>?
t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.searchMessages(type, targetId, channelId, keyword, startTime, count,
callback:callback);
```

回调方法

- **onMessagesSearched**

  接口调用结果的监听

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, String? keyword,
int? startTime, int? count, List<RCIMIWMessage>? messages)? onMessagesSearched;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
keyword	String	搜索的关键字
startTime	int	查询 beginTime 之前的消息
count	int	查询的数量
messages	List<RCIMIWMessage>	查询到的消息集合

代码示例

```
engine?.onMessagesSearched = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, String? keyword, int? startTime, int? count, List<RCIMIWMessage>? messages) {
//...
};
```

根据关键字搜索指定会话中某个时间段的消息

方法

```
Future<int> searchMessagesByTimeRange(RCIMIWConversationType type, String targetId, String? channelId,
String keyword, int startTime, int endTime, int offset, int count,
{IRCIMIWSearchMessagesByTimeRangeCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
keyword	String	搜索的关键字
startTime	int	开始时间
endTime	int	结束时间

参数名	参数类型	描述
offset	int	偏移量
count	int	返回的搜索结果数量，0 < count <= 50。
callback	IRCIMIWSearchMessagesByTimeRangeCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSearchMessagesByTimeRangeCallback? callback = IRCIMIWSearchMessagesByTimeRangeCallback(onSuccess:
(List<RCIMIWMessage>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.searchMessagesByTimeRange(type, targetId, channelId, keyword, startTime,
endTime, offset, count, callback:callback);
```

回调方法

- **onMessagesSearchedByTimeRange**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, String? keyword,
int? startTime, int? endTime, int? offset, int? count, List<RCIMIWMessage>? messages)?
onMessagesSearchedByTimeRange;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
keyword	String	搜索的关键字
startTime	int	开始时间
endTime	int	结束时间
offset	int	偏移量
count	int	查询的数量
messages	List<RCIMIWMessage>	查询到的消息集合

代码示例

```
engine?.onMessagesSearchedByTimeRange = (int? code, RCIMIWConversationType? type, String? targetId,
String? channelId, String? keyword, int? startTime, int? endTime, int? offset, int? count,
List<RCIMIWMessage>? messages) {
//...
};
```

根据用户 id 搜索指定会话中的消息

方法

```
Future<int> searchMessagesByUserId(String userId, RCIMIWConversationType type, String targetId, String?
channelId, int startTime, int count, {IRCIMIWSearchMessagesByUserIdCallback? callback});
```

参数说明

参数名	参数类型	描述
userId	String	用户 id
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
startTime	int	查询记录的起始时间， 传 0 时从最新消息开始搜索，从该时间往前搜索。
count	int	返回的搜索结果数量 0 < count <= 50。
callback	IRCIMIWSearchMessagesByUserIdCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSearchMessagesByUserIdCallback? callback = IRCIMIWSearchMessagesByUserIdCallback(onSuccess:
(List<RCIMIWMessage>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.searchMessagesByUserId(userId, type, targetId, channelId, startTime, count,
callback:callback);
```

回调方法

- **onMessagesSearchedByUserId**

  接口调用结果的监听

```
Function(int? code, String? userId, RCIMIWConversationType? type, String? targetId, String? channelId,
int? startTime, int? count, List<RCIMIWMessage>? messages)? onMessagesSearchedByUserId;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
userId	String	用户 id
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
startTime	int	查询记录的起始时间
count	int	查询的数量
messages	List<RCIMIWMessage>	查询到的消息集合

代码示例

```
engine?.onMessagesSearchedByUserId = (int? code, String? userId, RCIMIWConversationType? type, String?
targetId, String? channelId, int? startTime, int? count, List<RCIMIWMessage>? messages) {
//...
};
```

单聊消息回执　　发送已读回执

更新时间:2024-08-30

当 A 给 B 发送一条消息，B 用户调用发送阅读回执接口之后，A 用户可在回执监听中收到已读通知。

方法

```
Future<int> sendPrivateReadReceiptMessage(String targetId, String? channelId, int timestamp,
{IRCIMIWSendPrivateReadReceiptMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	该会话中已读的最后一条消息的发送时间戳
callback	IRCIMIWSendPrivateReadReceiptMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSendPrivateReadReceiptMessageCallback? callback =
IRCIMIWSendPrivateReadReceiptMessageCallback(onPrivateReadReceiptMessageSent: (int? code) {
//...
});

int? ret = await engine?.sendPrivateReadReceiptMessage(targetId, channelId, timestamp,
callback:callback);
```

回调方法

- **onPrivateReadReceiptMessageSent**

  接口调用结果的监听

```
Function(int? code, String? targetId, String? channelId, int? timestamp)?
onPrivateReadReceiptMessageSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	时间戳

代码示例

```
engine?.onPrivateReadReceiptMessageSent = (int? code, String? targetId, String? channelId, int?
timestamp) {
//...
};
```

设置消息回执监听器

设置消息回执监听器，用于接收消息回执。

方法

```
Function(String? targetId, String? channelId, int? timestamp)? onPrivateReadReceiptReceived;
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	已阅读的最后一条消息的 sendTime

代码示例

```
engine?.onPrivateReadReceiptReceived = (String? targetId, String? channelId, int? timestamp) {
//...
};
```

群聊消息回执

更新时间:2024-08-30

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。

发起回执请求

发起消息已读回执请求。

方法

```
Future<int> sendGroupReadReceiptRequest(RCIMIWMessage message,
{IRCIMIWSendGroupReadReceiptRequestCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	需要请求已读回执的消息
callback	IRCIMIWSendGroupReadReceiptRequestCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSendGroupReadReceiptRequestCallback? callback =
IRCIMIWSendGroupReadReceiptRequestCallback(onGroupReadReceiptRequestSent: (int? code, RCIMIWMessage?
message) {
//...
});

int? ret = await engine?.sendGroupReadReceiptRequest(message, callback:callback);
```

回调方法

- **onGroupReadReceiptRequestSent**

  接口调用结果的监听

```
Function(int? code, RCIMIWMessage? message)? onGroupReadReceiptRequestSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	需要请求已读回执的消息

代码示例

```
engine?.onGroupReadReceiptRequestSent = (int? code, RCIMIWMessage? message) {
//...
};
```

监听群聊回执请求

方法

```
Function(String? targetId, String? messageUId)? onGroupMessageReadReceiptRequestReceived;
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
messageUId	String	消息的 messageUid

代码示例

```
engine?.onGroupMessageReadReceiptRequestReceived = (String? targetId, String? messageUId) {
//...
};
```

响应回执请求

收到了回执请求，接收者需要响应该请求，通知对方已阅读此消息。可以一次响应同一会话中的多条消息。

方法

```
Future<int> sendGroupReadReceiptResponse(String targetId, String? channelId, List<RCIMIWMessage>
messages, {IRCIMIWSendGroupReadReceiptResponseCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
messages	List<RCIMIWMessage>	会话中需要发送已读回执的消息列表
callback	IRCIMIWSendGroupReadReceiptResponseCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

示例代码

```
IRCIMIWSendGroupReadReceiptResponseCallback? callback =
IRCIMIWSendGroupReadReceiptResponseCallback(onGroupReadReceiptResponseSent: (int? code,
List<RCIMIWMessage>? message) {
//...
});

int? ret = await engine?.sendGroupReadReceiptResponse(targetId, channelId, messages, callback:callback);
```

回调方法

- **onGroupReadReceiptResponseSent**

  接口调用结果的监听

```
Function(int? code, String? targetId, String? channelId, List<RCIMIWMessage>? messages)?
onGroupReadReceiptResponseSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
messages	List<RCIMIWMessage>	会话中需要发送已读回执的消息列表

代码示例

```
engine?.onGroupReadReceiptResponseSent = (int? code, String? targetId, String? channelId,
List<RCIMIWMessage>? messages) {
//...
};
```

监听群聊回执响应

方法

```
Function(String? targetId, String? messageUId, Map? respondUserIds)?
onGroupMessageReadReceiptResponseReceived;
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
messageUId	String	收到回执响应的消息的 messageUId
respondUserIds	Ma<String, Long>	会话中响应了此消息的用户列表。其中 key： 用户 id ； value： 响应时间。

代码示例

```
engine?.onGroupMessageReadReceiptResponseReceived = (String? targetId, String? messageUId, Map?
respondUserIds) {
//...
};
```

消息扩展　　适用场景

更新时间:2024-08-30

原始消息增加状态标识的需求，都可使用消息扩展。

- 消息评论需求，可通过设置原始消息扩展信息的方式添加评论信息。
- 礼物领取、订单状态变化需求，通过此功能改变消息显示状态。 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。
- 通过该功能可以对 `message` 设置扩展信息。
- 仅支持单聊、群聊会话类型，不支持聊天室类型。
- 每次设置消息扩展将会产生内置通知消息，频繁设置扩展会产生大量消息。

设置消息可扩展

在 构建消息 之后设置 message 的 expansion 属性，默认为不开启，设置值之后会自动开启。在发送消息的时候可以设置，发出去之后不可更改。

代码示例

```
RCIMIWImageMessage? message = await engine.createImageMessage(
type,
targetId,
channelId,
path,
);
message?.expansion = {};
```

更新消息扩展信息

1. 更新消息扩展信息。
2. 消息发送后调用。

方法

```
Future<int> updateMessageExpansion(String messageUId, Map expansion,
{IRCIMIWUpdateMessageExpansionCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值

参数名	参数类型	描述
expansion	Ma<String, String>	要更新的消息扩展信息键值对，类型是 HashMap；Key 支持大小写英文字母、数字、部分特殊符号 + = - _ 的组合方式，不支持汉字。Value 可以输入空格
callback	IRCIMIWUpdateMessageExpansionCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWUpdateMessageExpansionCallback? callback =
IRCIMIWUpdateMessageExpansionCallback(onMessageExpansionUpdated: (int? code) {
//...
});

int? ret = await engine?.updateMessageExpansion(messageUId, expansion, callback:callback);
```

回调方法

- **onMessageExpansionUpdated**

  接口调用结果的监听

```
Function(int? code, String? messageUId, Map? expansion)? onMessageExpansionUpdated;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messageUId	String	消息的 messageUid
expansion	Ma<String, String>	要更新的消息扩展信息键值对，类型是 HashMap

代码示例

```
engine?.onMessageExpansionUpdated = (int? code, String? messageUId, Map? expansion) {
//...
};
```

删除消息扩展信息

1. 删除消息扩展信息中特定的键值对。
2. 消息发送后调用。

方法

```
Future<int> removeMessageExpansionForKeys(String messageUId, List<String> keys,
{IRCIMIWRemoveMessageExpansionForKeysCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值
keys	List<String>	消息扩展信息中待删除的 key 的列表，类型是 ArrayList
callback	IRCIMIWRemoveMessageExpansionForKeysCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveMessageExpansionForKeysCallback? callback =
IRCIMIWRemoveMessageExpansionForKeysCallback(onMessageExpansionForKeysRemoved: (int? code) {
//...
});

int? ret = await engine?.removeMessageExpansionForKeys(messageUId, keys, callback:callback);
```

回调方法

- **onMessageExpansionForKeysRemoved**

接口调用结果的监听

```
Function(int? code, String? messageUId, List<String>? keys)? onMessageExpansionForKeysRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messageUId	String	消息的 messageUid

参数名	参数类型	描述
keys	List<String>	消息扩展信息中待删除的 key 的列表，类型是 ArrayList

代码示例

```
engine?.onMessageExpansionForKeysRemoved = (int? code, String? messageUId, List<String>? keys) {
//...
};
```

消息扩展更新监听

方法

```
Function(Map? expansion, RCIMIWMessage? message)? onRemoteMessageExpansionUpdated;
```

参数说明

参数名	参数类型	描述
expansion	Ma<String, String>	消息扩展信息中更新的键值对，只包含更新的键值对，不是全部的数据。如果想获取全部的键值对，请使用 message 的 expansion 属性。
message	RCIMIWMessage	发生变化的消息

代码示例

```
engine?.onRemoteMessageExpansionUpdated = (Map? expansion, RCIMIWMessage? message) {
//...
};
```

消息扩展移除监听

方法

```
Function(RCIMIWMessage? message, List<String>? keys)? onRemoteMessageExpansionForKeyRemoved;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	发生变化的消息
keys	List<String>	消息扩展信息中删除的键值对 key 列表

代码示例

```
engine?.onRemoteMessageExpansionForKeyRemoved = (RCIMIWMessage? message, List<String>? keys) {
//...
};
```

@ 消息 功能描述

更新时间:2024-08-30

开发者在群组会话中，可以发送 @ 消息。用于提示所有群成员或者是指定的某些用户。

类型说明

@ 消息类：RCIMIWMentionedInfo

属性说明：

属性名	类型	说明
type	RCIMIWMentionedType	@ 提醒的类型
userIdList	List<String>	@ 的用户 ID 列表
mentionedContent	String	包含 @ 提醒的消息，本地通知和远程推送显示的内容

代码示例

```
// 1、创建一条消息
RCIMIWTextMessage? textMessage = await engine.createTextMessage(
conversationType,
targetId,
channelId,
text,
);
// 2、创建 @ 的配置
RCIMIWMentionedInfo mentionedInfo = RCIMIWMentionedInfo.create(
type: mentionedType,
userIdList: users,
mentionedContent: '包含 @ 提醒的消息，本地通知和远程推送显示的内容',
);
// 3、将创建的 @ 配置设置到消息中
textMessage.mentionedInfo = mentionedInfo;
// 4、发送消息
int code = await engine?.sendMessage(textMessage) ?? -1;
```

消息下载 功能描述

更新时间:2024-08-30

- SDK 提供多媒体文件的下载、取消、暂停功能。
- 多媒体消息的发送，请参见消息发送 中的「发送媒体消息」部分。

下载多媒体消息

下载多媒体文件。

方法

```
Future<int> downloadMediaMessage(RCIMIWMediaMessage message, {RCIMIWDownloadMediaMessageListener?
listener});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	需要下载的媒体消息实体
listener	RCIMIWDownloadMediaMessageListener	下载媒体消息的事件监听

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
RCIMIWDownloadMediaMessageListener? listener =
RCIMIWDownloadMediaMessageListener(onMediaMessageDownloading: (RCIMIWMediaMessage? message, int?
progress) {
//...
}, onDownloadingMediaMessageCanceled: (RCIMIWMediaMessage? message) {
//...
}, onMediaMessageDownloaded: (int? code, RCIMIWMediaMessage? message) {
//...
});

int? ret = await engine?.downloadMediaMessage(message, listener:listener);
```

回调方法

- **onMediaMessageDownloading**

接口调用结果的监听

```
Function(RCIMIWMediaMessage? message, int? progress)? onMediaMessageDownloading;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	下载的消息
progress	int	下载的进度

代码示例

```
engine?.onMediaMessageDownloading = (RCIMIWMediaMessage? message, int? progress) {
//...
};
```

- **onMediaMessageDownloaded**

接口调用结果的监听

```
Function(int? code, RCIMIWMediaMessage? message)? onMediaMessageDownloaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMediaMessage	下载的消息

代码示例

```
engine?.onMediaMessageDownloaded = (int? code, RCIMIWMediaMessage? message) {
//...
};
```

取消多媒体消息

取消多媒体消息下载。

方法

```
Future<int> cancelDownloadingMediaMessage(RCIMIWMediaMessage message,
{IRCIMIWCancelDownloadingMediaMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMediaMessage	需要取消下载的媒体消息实体
callback	IRCIMIWCancelDownloadingMediaMessageCallback	取消下载媒体消息的事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

回调方法

- **onDownloadingMediaMessageCanceled**

接口调用结果的监听

```
Function(int? code, RCIMIWMediaMessage? message)? onDownloadingMediaMessageCanceled;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMediaMessage	取消下载的消息

代码示例

```
engine?.onDownloadingMediaMessageCanceled = (int? code, RCIMIWMediaMessage? message) {
//...
};
```

管理离线消息存储配置

更新时间:2024-08-30

即时通讯业务支持修改 App 级别的离线消息配置。

> 提示
>
> 离线消息配置仅适用于单聊、群聊。聊天室、超级群因业务特性不支持离线消息，因此无离线消息配置。

了解离线消息

离线消息是指当用户不在线时收到的消息。融云服务端会自动为用户保留离线期间接收的消息，默认的离线消息保留时长为 7 天。7 天内客户端如果上线，服务端会直接将离线消息发送到该接收端。如果 7 天内客户端都没有上线，服务端将抛弃掉过期的消息。

即时通讯业务下并非所有会话类型都支持离线消息：

- **支持离线消息**：单聊、群聊、系统消息
- **不支持离线消息**：聊天室、超级群

App 级别离线消息配置

如需修改 App 级别设置，请提交工单。

App 级别的离线消息配置如下：

- 单聊离线消息存储时长：默认存储 7 天。设置范围为 1 - 7 天。配置修改将影响 App 下所有单聊会话。
- 群聊离线消息存储时长：默认存储 7 天。设置范围为 1 - 7 天。配置修改将影响 App 下所有群聊会话。
- 群组离线消息存储数量：默认存储 7 天内的所有群消息。配置修改将影响 App 下所有群聊会话。

会话介绍

更新时间:2024-08-30

会话是指融云 SDK 根据每条消息的发送方、接收方以及会话类型等信息，自动建立并维护的逻辑关系，是一种抽象概念。

会话类型

融云支持多种会话类型，以满足不同业务场景需求。客户端 SDK 通过 RCIMIWConversationType 枚举来表示各类型会话，各枚举值代表的含义参考下表：

枚举	说明
invalid	暂不支持，SDK 保留类型，开发者不可使用
private	单聊会话
group	群聊会话
chatroom	聊天室会话
system	系统会话
ultraGroup	超级群会话

单聊会话

指两个用户一对一进行聊天，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。

单聊类型会话里的消息会保存在客户端本地数据库中。

群组会话

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。每个群最大人数上限为 3000 人，App 内的群组数量没有限制。

群组类型会话里的消息会保存在客户端本地数据库中。

超级群会话

超级群（UltraGroup）提供了一种新的群组业务形态，支持在群会话中创建独立的频道（channel），超级群的消息数据（会话、消息、未读数）和群组成员支持分频道进行聚合，各个频道之间消息独立。

超级群类型会话里的消息会保存在客户端本地数据库中，更多内容请参见超级群概述。

聊天室会话

聊天室成员不设用户上限，海量消息并发即时到达，用户退出聊天室后不会再接收到任何聊天室中的消息，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。

SDK 不保存聊天室消息，在退出聊天室时会清空此聊天室所有数据，更多内容请参见聊天室概述。

系统会话

系统会话是指利用系统帐号向用户发送消息从而建立的会话关系，此类型会话可以是通过调用广播接口发送广播来建立，也可以是加好友等单条通知消息而建立的会话。

会话实体类

客户端 SDK 中封装的会话实体类是 RCIMIWConversation ，所有会话相关的信息都从该实体类中获取。

下表列出了 RCIMIWConversation 中提供的主要成员

属性	类型	描述
conversationType	RCIMIWConversationType	会话类型，参考上文详细描述。
targetId	String	会话 Id，单聊时为接收方 Id，群组会话中为群组 Id，聊天室会话中为聊天室 Id，系统会话为开发者指定的系统账号 Id。
channelId	String	频道 ID
unreadCount	int	当前会话未读消息数量
mentionedCount	int	本会话里自己被 @ 的消息数量
firstUnreadMsgSendTime	int	会话第一条未读消息的时间戳，仅对超级群生效。
top	bool	是否置顶
draft	String	会话里保存的草稿信息
notificationLevel	RCIMIWPushNotificationLevel	免打扰级别
lastMessage	RCIMIWMessage	会话中在客户端本地存储的最后一条消息的消息内容。

获取会话

更新时间:2024-08-30

客户端 SDK 会根据收发的消息在本地数据库中生成对应会话。您可以从本地数据库获取 SDK 生成的会话列表。

获取会话列表

通过以下接口分页获取 SDK 在本地数据库生成的会话列表。获取到的会话列表按照时间倒序排列，置顶会话会排在最前。

方法

```
Future<int> getConversations(List<RCIMIWConversationType> conversationTypes, String? channelId, int
startTime, int count, {IRCIMIWGetConversationsCallback? callback});
```

参数说明

参数名	参数类型	描述
conversationTypes	List<RCIMIWConversationType >	会话类型
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可
startTime	int	时间戳（毫秒），获取小于此时间戳的会话，传 0 为查询最新数据。在分页获取会话时，传入回调结果中最后一条会话数据的 operationTime，来获取下一页数据。
count	int	查询的数量， 0 < count <= 50
callback	IRCIMIWGetConversationsCallback	获取会话列表事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWGetConversationsCallback? callback = IRCIMIWGetConversationsCallback(onSuccess:
(List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversations(conversationTypesInt, channelId, startTime, count,
callback:callback);
```

回调方法

- **onConversationsLoaded**

获取会话列表的结果回调

```
Function(int? code, List<RCIMIWConversationType>? conversationTypes, String? channelId, int? startTime,
int? count, List<RCIMIWConversation>? conversations)? onConversationsLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用
startTime	int	时间戳（毫秒）
count	int	查询的数量
conversations	List<RCIMIWConversation>	查询到的会话集合

代码示例

```
engine?.onConversationsLoaded = (int? code, List<RCIMIWConversationType>? conversationTypes, String?
channelId, int? startTime, int? count, List<RCIMIWConversation>? conversations) {
//...
};
```

获取指定会话

获取某个会话的详细信息。

方法

```
Future<int> getConversation(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWGetConversationCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可

参数名	参数类型	描述
callback	IRCIMIWGetConversationCallback	获取会话事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWGetConversationCallback? callback = IRCIMIWGetConversationCallback(onSuccess:
(RCIMIWConversation? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversation(type, targetId, channelId, callback:callback);
```

回调方法

- **onConversationLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
RCIMIWConversation? conversation)? onConversationLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
conversation	RCIMIWConversation	获取到的会话

代码示例

```
engine?.onConversationLoaded = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, RCIMIWConversation? conversation) {
//...
};
```

获取未读列表

获取指定类型的含有未读消息的会话列表，支持单聊、群聊、系统会话。获取到的会话列表按照时间倒序排列，置顶会话会排

在最前。

方法

```
Future<int> getUnreadConversations(List<RCIMIWConversationType> conversationTypes,
{IRCIMIWGetUnreadConversationsCallback? callback});
```

参数说明

参数名	参数类型	描述
conversationTypes	List<RCIMIWConversationType>	支持单聊、群聊、系统会话
callback	IRCIMIWGetUnreadConversationsCallback	获取会话列表事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码。

代码示例

```
IRCIMIWGetUnreadConversationsCallback? callback = IRCIMIWGetUnreadConversationsCallback(onSuccess:
(List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUnreadConversations(conversationTypesInt, callback:callback);
```

常见问题

Q1: 当用户卸载重新安装后发现会话列表为空或者部分会话丢失

A1: 由于会话列表是从本地数据库获取的, 是在 SDK 内部数据库存储的，所以当用户在卸载的时候会删除本地数据库，导致重新安装后会话列表为空，而会出现部分会话的原因是因为开启了 离线消息补偿 功能，具体操作是在后台购买 多设备消息同步 功能，这个功能里默认涵盖了今天(当天 **0** 点)的离线消息补偿，所以当用户在新设备登陆会触发离线消息补偿功能，从而获取到部分会话，造成部分会话丢失的错觉。如果想要更多天的离线消息补偿，可 提交工单 修改，最多支持 7 天，设置时间过长，当单用户消息量超大时，可能会因为补偿消息量过大，造成端上处理压力的问题。

会话未读数　　获取所有会话未读数

更新时间:2024-08-30

获取所有会话类型(除聊天室外)的未读数。

方法

```
Future<int> getTotalUnreadCount(String? channelId, {IRCIMIWGetTotalUnreadCountCallback? callback});
```

参数说明

参数名	参数类型	描述
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetTotalUnreadCountCallback	获取所有未读数事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetTotalUnreadCountCallback? callback = IRCIMIWGetTotalUnreadCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getTotalUnreadCount(channelId, callback:callback);
```

回调方法

- **onTotalUnreadCountLoaded**

```
Function(int? code, String? channelId, int? count)? onTotalUnreadCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
count	int	未读数量

代码示例

```
engine?.onTotalUnreadCountLoaded = (int? code, String? channelId, int? count) {
//...
};
```

获取指定会话未读数

获取指定会话(除聊天室外)的所有未读消息个数。

方法

```
Future<int> getUnreadCount(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWGetUnreadCountCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetUnreadCountCallback	获取会话未读数事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUnreadCountCallback? callback = IRCIMIWGetUnreadCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUnreadCount(type, targetId, channelId, callback:callback);
```

回调方法

- **onUnreadCountLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? count)?
onUnreadCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
count	int	未读数量

代码示例

```
engine?.onUnreadCountLoaded = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, int? count) {
//...
};
```

按会话类型获取未读数

获取多个指定会话类型(除聊天室外)的未读数。

方法

```
Future<int> getUnreadCountByConversationTypes(List<RCIMIWConversationType> conversationTypes, String?
channelId, bool contain, {IRCIMIWGetUnreadCountByConversationTypesCallback? callback});
```

参数说明

参数名	参数类型	描述
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
contain	bool	是否包含免打扰消息的未读消息数。
callback	IRCIMIWGetUnreadCountByConversationTypesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUnreadCountByConversationTypesCallback? callback =
IRCIMIWGetUnreadCountByConversationTypesCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUnreadCountByConversationTypes(conversationTypesInt, channelId, contain,
callback:callback);
```

回调方法

- **onUnreadCountByConversationTypesLoaded**

```
Function(int? code, List<RCIMIWConversationType>? conversationTypes, String? channelId, bool? contain,
int? count)? onUnreadCountByConversationTypesLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
contain	bool	是否包含免打扰消息的未读消息数。
count	int	未读数量

代码示例

```
engine?.onUnreadCountByConversationTypesLoaded = (int? code, List<RCIMIWConversationType>?
conversationTypes, String? channelId, bool? contain, int? count) {
//...
};
```

清除指定会话未读数

清除指定会话(除聊天室外)的未读数。

方法

```
Future<int> clearUnreadCount(RCIMIWConversationType type, String targetId, String? channelId, int
timestamp, {IRCIMIWClearUnreadCountCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	该会话已阅读的最后一条消息的发送时间戳，清除所有传入当前最新时间戳
callback	IRCIMIWClearUnreadCountCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearUnreadCountCallback? callback = IRCIMIWClearUnreadCountCallback(onUnreadCountCleared: (int?
code) {
//...
});

int? ret = await engine?.clearUnreadCount(type, targetId, channelId, timestamp, callback:callback);
```

回调方法

- **onUnreadCountCleared**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? timestamp)?
onUnreadCountCleared;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	该会话已阅读的最后一条消息的发送时间戳

代码示例

```
engine?.onUnreadCountCleared = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, int? timestamp) {
//...
};
```

加载所有@未读数

加载某个会话的所有@未读数

方法

```
Future<int> getUnreadMentionedCount(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWGetUnreadMentionedCountCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetUnreadMentionedCountCallback	获取会话中未读的 @ 消息数量事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

代码示例

```
IRCIMIWGetUnreadMentionedCountCallback? callback = IRCIMIWGetUnreadMentionedCountCallback(onSuccess:
(int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUnreadMentionedCount(type, targetId, channelId, callback:callback);
```

回调方法

- **onUnreadMentionedCountLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? count)?
onUnreadMentionedCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
count	int	未读数量

代码示例

```
engine?.onUnreadMentionedCountLoaded = (int? code, RCIMIWConversationType? type, String? targetId,
String? channelId, int? count) {
//...
};
```

删除会话 删除指定会话

更新时间:2024-08-30

- 从会话列表中移除指定会话，不删除会话内的消息。
- 会话内再来一条消息，该会话之前的消息还存在，方便用户查看。
- 如果需要移除指定会话，并删除会话内的消息，请同时调用删除会话内消息接口。

方法

```
Future<int> removeConversation(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWRemoveConversationCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可
callback	IRCIMIWRemoveConversationCallback	移除会话事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveConversationCallback? callback = IRCIMIWRemoveConversationCallback(onConversationRemoved:
(int? code) {
//...
});

int? ret = await engine?.removeConversation(type, targetId, channelId, callback:callback);
```

回调方法

- **onConversationRemoved**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId)?
onConversationRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。

代码示例

```
engine?.onConversationRemoved = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId) {
//...
};
```

按会话类型删除

按指定的会话类型删除会话，并删除会话内的消息。

方法

```
Future<int> removeConversations(List<RCIMIWConversationType> conversationTypes, String? channelId,
{IRCIMIWRemoveConversationsCallback? callback});
```

参数说明

参数名	参数类型	描述
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWRemoveConversationsCallback	移除会话列表事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveConversationsCallback? callback =
IRCIMIWRemoveConversationsCallback(onConversationsRemoved: (int? code) {
//...
});

int? ret = await engine?.removeConversations(conversationTypesInt, channelId, callback:callback);
```

回调方法

- **onConversationsRemoved**

```
Function(int? code, List<RCIMIWConversationType>? conversationTypes, String? channelId)?
onConversationsRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。

代码示例

```
engine?.onConversationsRemoved = (int? code, List<RCIMIWConversationType>? conversationTypes, String?
channelId) {
//...
};
```

删除全部会话

SDK 内部没有清除全部会话的方法，您可使用 按会话类型删除, 传入所有会话类型即可。

该方法会同时删除会话和消息。聊天室和超级群 UI 层建议客户单独展示，所以删除时可不传入。

代码示例

```
engine.removeConversations(
[RCIMIWConversationType.private, RCIMIWConversationType.group, RCIMIWConversationType.system],
null,
);
```

会话草稿 保存草稿

更新时间:2024-08-30

- 保存一条草稿内容至指定会话。
- 保存草稿会更新会话的 `sentTime`，该会话会排在列表前部。

方法

```
Future<int> saveDraftMessage(RCIMIWConversationType type, String targetId, String? channelId, String draft, {IRCIMIWSaveDraftMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
draft	String	草稿的文字内容。
callback	IRCIMIWSaveDraftMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSaveDraftMessageCallback? callback = IRCIMIWSaveDraftMessageCallback(onDraftMessageSaved: (int? code) {
//...
});

int? ret = await engine?.saveDraftMessage(type, targetId, channelId, draft, callback:callback);
```

回调方法

- **onDraftMessageSaved**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, String? draft)?
onDraftMessageSaved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
draft	String	草稿信息

代码示例

```
engine?.onDraftMessageSaved = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, String? draft) {
//...
};
```

获取草稿

获取草稿内容。

方法

```
Future<int> getDraftMessage(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWGetDraftMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetDraftMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetDraftMessageCallback? callback = IRCIMIWGetDraftMessageCallback(onSuccess: (String? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getDraftMessage(type, targetId, channelId, callback:callback);
```

回调方法

- **onDraftMessageLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, String? draft)?
onDraftMessageLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常 接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
draft	String	草稿信息

代码示例

```
engine?.onDraftMessageLoaded = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId, String? draft) {
//...
};
```

删除草稿

清除某个会话的草稿信息。

方法

```
Future<int> clearDraftMessage(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWClearDraftMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWClearDraftMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
Future<int> clearDraftMessage(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWClearDraftMessageCallback? callback});
```

回调方法

- **onDraftMessageCleared**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId)?
onDraftMessageCleared;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。

代码示例

```
engine?.onDraftMessageCleared = (int? code, RCIMIWConversationType? type, String? targetId, String?
channelId) {
//...
};
```

会话置顶

更新时间:2024-08-30

SDK 提供设置会话是否置顶接口，置顶的状态将会被同步到服务端，切换设备后置顶状态也会一并同步下来。

设置会话置顶

方法

```
Future<int> changeConversationTopStatus(RCIMIWConversationType type, String targetId, String? channelId,
bool top, {IRCIMIWChangeConversationTopStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
top	bool	是否置顶
callback	IRCIMIWChangeConversationTopStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeConversationTopStatusCallback? callback =
IRCIMIWChangeConversationTopStatusCallback(onConversationTopStatusChanged: (int? code) {
//...
});

int? ret = await engine?.changeConversationTopStatus(type, targetId, channelId, top, callback:callback);
```

回调方法

- **onConversationTopStatusChanged**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, bool? top)?
onConversationTopStatusChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
top	bool	是否置顶

代码示例

```
engine?.onConversationTopStatusChanged = (int? code, RCIMIWConversationType? type, String? targetId,
String? channelId, bool? top) {
//...
};
```

获取会话置顶状态

通过此方法获取指定会话的置顶状态。

方法

```
Future<int> getConversationTopStatus(RCIMIWConversationType type, String targetId, String? channelId,
{IRCIMIWGetConversationTopStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetConversationTopStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetConversationTopStatusCallback? callback = IRCIMIWGetConversationTopStatusCallback(onSuccess:
(bool? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversationTopStatus(type, targetId, channelId, callback:callback);
```

回调方法

- **onConversationTopStatusLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, bool? top)?
onConversationTopStatusLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
top	bool	是否置顶

代码示例

```
engine?.onConversationTopStatusLoaded = (int? code, RCIMIWConversationType? type, String? targetId,
String? channelId, bool? top) {
//...
};
```

获取置顶会话列表

获取置顶会话列表

方法

```
Future<int> getTopConversations(List<RCIMIWConversationType> conversationTypes, String? channelId,
{IRCIMIWGetTopConversationsCallback? callback});
```

参数说明

参数名	参数类型	描述
conversationTypes	List<RCIMIWConversationType>	会话类型集合

参数名	参数类型	描述
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetTopConversationsCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetTopConversationsCallback? callback = IRCIMIWGetTopConversationsCallback(onSuccess:
(List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getTopConversations(conversationTypesInt, channelId, callback:callback);
```

回调方法

- **onTopConversationsLoaded**

```
Function(int? code, List<RCIMIWConversationType>? conversationTypes, String? channelId,
List<RCIMIWConversation>? conversations)? onTopConversationsLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
conversations	List<RCIMIWConversation>	加载的会话集合

代码示例

```
engine?.onTopConversationsLoaded = (int? code, List<RCIMIWConversationType>? conversationTypes, String?
channelId, List<RCIMIWConversation>? conversations) {
//...
};
```

置顶状态同步

SDK 提供了会话状态（置顶或免打扰）同步机制，通过设置会话状态同步监听器，当在其它端修改会话状态时，可在本端实

时监听到会话状态的改变。

方法

```
Function(RCIMIWConversationType? type, String? targetId, String? channelId, bool? top)?
onConversationTopStatusSynced;
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。 频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
top	bool	是否置顶

代码示例

```
engine?.onConversationTopStatusSynced = (RCIMIWConversationType? type, String? targetId, String?
channelId, bool? top) {
//...
};
```

输入状态　　发送输入状态消息

更新时间:2024-08-30

1. 用户正在输入的时候，向对方发送正在输入的状态。
2. 只支持单聊会话类型。

方法

```
Future<int> sendTypingStatus(RCIMIWConversationType type, String targetId, String? channelId, String currentType);
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
currentType	String	当前的状态

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
int? ret = await engine?.sendTypingStatus(type, targetId, channelId, currentType);
```

设置输入状态监听器

触发时机：当前会话正在输入的用户有变化时

方法

```
Function(RCIMIWConversationType? type, String? targetId, String? channelId, List<RCIMIWTypingStatus>? userTypingStatus)? onTypingStatusChanged;
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
userTypingStatus	List<RCIMIWTypingStatus>	发生状态变化的集合

RCIMIWTypingStatus介绍

属性名	类型	说明
userId	String	当前正在输入的用户 ID
contentType	String	当前正在输入的消息类型名，为发送方调用发送接口时传入的 currentType
sentTime	int	输入时间

代码示例

```
engine?.onTypingStatusChanged = (RCIMIWConversationType? type, String? targetId, String? channelId,
List<RCIMIWTypingStatus>? userTypingStatus) {
//...
};
```

管理标签信息数据 管理标签信息数据

更新时间:2024-08-30

SDK 从 5.6.11 版本开始支持创建标签。

本文描述 App 如何使用 RCIMIWEngine 下的接口创建和管理标签信息数据。客户端 SDK 支持用户创建标签信息（RCIMIWTagInfo），用于对会话进行标记分组。每个用户最多可以创建 20 个标签。App 用户创建的标签信息数据会同步到融云服务端。

标签信息（RCIMIWTagInfo）的定义如下：

参数	类型	说明
tagId	NSString	标签唯一标识，字符型，长度不超过 10 个字。
tagName	NSString	长度不超过 15 个字，标签名称可以重复。
timestamp	long long	时间戳由 SDK 内部协议栈提供。

提示

本文仅描述如何管理标签信息数据。关于如何为会话设置标签、以及如何按标签获取会话数据，请参见设置与使用会话标签。

创建标签信息

创建标签，每个用户最多可以创建 20 个标签。

方法

```
Future<int> createTag(String tagId, String tagName, {IRCIMIWCreateTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
tagName	String	长度不超过 15 个字，标签名称可以重复。
callback	IRCIMIWCreateTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWCreateTagCallback? callback = IRCIMIWCreateTagCallback(onTagCreated: (int? code) {
//...
});

int? ret = await engine?.createTag(tagId, tagName, callback:callback);
```

移除标签信息

移除标签。移除标签信息时只需要传入 RCIMIWTagInfo 中的 tagId。

方法

```
Future<int> removeTag(String tagId, {IRCIMIWRemoveTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
callback	IRCIMIWRemoveTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWRemoveTagCallback? callback = IRCIMIWRemoveTagCallback(onTagRemoved: (int? code) {
//...
});

int? ret = await engine?.removeTag(tagId, callback:callback);
```

编辑标签信息

更新标签信息。仅支持修改 RCIMIWTagInfo 中的标签名称（tagName）字段。长度不超过 15 个字，标签名称可以重复。

方法

```
Future<int> updateTagNameById(String tagId, String newName, {IRCIMIWUpdateTagNameByIdCallback?
callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
newName	String	长度不超过 15 个字，标签名称可以重复。
callback	IRCIMIWUpdateTagNameByIdCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWUpdateTagNameByIdCallback? callback = IRCIMIWUpdateTagNameByIdCallback(onTagNameByIdUpdated:
(int? code) {
//...
});

int? ret = await engine?.updateTagNameById(tagId, newName, callback:callback);
```

获取标签信息列表

获取当前用户已创建的标签信息。成功回调中会返回 RCIMIWTagInfo 的列表。

方法

```
Future<int> getTags({IRCIMIWGetTagsCallback? callback});
```

参数说明

参数名	参数类型	描述
callback	IRCIMIWGetTagsCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWGetTagsCallback? callback = IRCIMIWGetTagsCallback(onSuccess: (List<RCIMIWTagInfo>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getTags(callback:callback);
```

设置与使用会话标签 设置与使用会话标签

更新时间:2024-08-30

- SDK 从 5.6.11 版本开始支持会话标签功能。
- 在为会话设置标签前，请确保已创建标签信息。详见管理标签信息数据。
- 本功能不适用于聊天室、超级群。

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。如果标签下已添加 1000 个会话，继续在该标签下添加会话仍会成功，但会导致最早添加标签的会话被移除标签。

场景描述

会话标签常实现 App 用户对会话进行分组的需求。创建标签信息（RCIMIWTagInfo）后，App 用户可以为会话设置一个或多个标签。

设置标签后，可以利用会话的标签数据实现会话的分组获取、展示、删除等特性。还可以获取指定标签下所有会话的消息未读数，或在特定标签下设置某个会话置顶。

- **场景 1**：对会话列表中的每个会话打 tag，类似企业微信会话列表中的外部群，部门群，个人群等 tag。
- **场景 2**：通讯录根据 tag 来分组，类似 QQ 好友列表中的家人，朋友，同事分组等。
- **场景 3**：前两个场景的结合，按照 tag 来进行会话列表分组，类似 Telegram 的会话列表分组。

使用标签标记会话

在创建标签信息（RCIMIWTagInfo）后，App 用户可以使用标签标记会话。SDK 将用标签标记会话的操作视为将会话添加到标签中。

支持以下操作：

- 标记会话，即将一个或多个会话添加到指定标签。
- 从标签中移除一个或多个会话。
- 为指定会话移除一个或多个标签。

将一个或多个会话添加到指定标签

SDK 将用标签标记会话的操作视为将会话添加到标签中。您可以将多个会话添加到一个标签。指定标签时只需要传入 RCIMIWTagInfo 中的 `tagId`。

方法

```
Future<int> addConversationToTag(String tagId, RCIMIWConversationType type, String targetId,
{IRCIMIWAddConversationToTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
callback	IRCIMIWAddConversationToTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWAddConversationToTagCallback? callback =
IRCIMIWAddConversationToTagCallback(onConversationToTagAdded: (int? code) {
//...
});

int? ret = await engine?.addConversationToTag(tagId, type, targetId, callback:callback);
```

从指定标签下移除会话

App 用户可能需要携带指定标签的会话中移除一个或多个会话。例如，在所有添加了「培训班」标签的会话中移除与「Tom」的私聊会话。SDK 将该操作视为从指定标签中移除会话。移除成功后，会话仍然存在，但不再携带该标签。

方法

```
Future<int> removeConversationFromTag(String tagId, RCIMIWConversationType type, String targetId,
{IRCIMIWRemoveConversationFromTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
callback	IRCIMIWRemoveConversationFromTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWRemoveConversationFromTagCallback? callback =
IRCIMIWRemoveConversationFromTagCallback(onConversationFromTagRemoved: (int? code) {
//...
});

int? ret = await engine?.removeConversationFromTag(tagId, type, targetId, callback:callback);
```

为指定会话中移除标签

App 用户可能为指定会话中添加了多个标签。SDK 支持一次移除单个或多个标签。移除时需要传入所有待移除 RCIMIWTagInfo 的 tagId 列表。

方法

```
Future<int> removeTagsFromConversation(RCIMIWConversationType type, String targetId, List<String>
tagIds, {IRCIMIWRemoveTagsFromConversationCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
tagIds	List<String>	标签集合
callback	IRCIMIWRemoveTagsFromConversationCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWRemoveTagsFromConversationCallback? callback =
IRCIMIWRemoveTagsFromConversationCallback(onTagsFromConversationRemoved: (int? code) {
//...
});

int? ret = await engine?.removeTagsFromConversation(type, targetId, tagIds, callback:callback);
```

获取指定会话的所有标签

获取指定会话携带的所有标签。获取成功后，回调中返回 [RCConversationTagInfo] 的列表。每个 [RCConversationTagInfo] 中包含对应的标签信息 RCIMIWTagInfo 和置顶状态信息（会话是否在携带该标签信息的所有会话中置顶）。

方法

```
Future<int> getTagsFromConversation(RCIMIWConversationType type, String targetId,
{IRCIMIWGetTagsFromConversationCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
callback	IRCIMIWGetTagsFromConversationCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功，具体结果需要实现接口回调；非 0 代表当前接口调用操作失败，不会触发接口回调。详细错误参考错误码。

代码示例

```
IRCIMIWGetTagsFromConversationCallback? callback = IRCIMIWGetTagsFromConversationCallback(onSuccess:
(List<RCIMIWConversationTagInfo>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getTagsFromConversation(type, targetId, callback:callback);
```

按标签操作会话数据

SDK 支持对携带指定标签的会话进行操作。App 用户为会话添加标签后，可以实现以下操作：

- 配合使用[会话置顶]功能，可以实现在携带指定标签的会话中置顶会话。
- 按标签获取会话列表，即获取携带指定标签的所有会话。
- 按标签获取未读消息数。
- 清除标签对应会话的未读消息数。
- 删除标签对应的会话。

在携带指定标签的会话中置顶

App 可以使用会话标签按照业务需求对会话进行分类和展示。如果需要在同一类会话（携带同一标签的所有会话）中置顶显示会话，可以使用会话置顶功能。

方法

```
Future<int> changeConversationTopStatusInTag(String tagId, RCIMIWConversationType type, String targetId,
bool top, {IRCIMIWChangeConversationTopStatusInTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
top	bool	是否置顶
callback	IRCIMIWChangeConversationTopStatusInTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeConversationTopStatusInTagCallback? callback =
IRCIMIWChangeConversationTopStatusInTagCallback(onConversationTopStatusInTagChanged: (int? code) {
//...
});

int? ret = await engine?.changeConversationTopStatusInTag(tagId, type, targetId, top,
callback:callback);
```

分页获取本地指定标签下会话列表

以会话中最后一条消息时间戳为界，分页获取本地指定标签下会话列表。

方法

```
Future<int> getConversationsFromTagByPage(String tagId, int timestamp, int count,
{IRCIMIWGetConversationsCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
timestamp	int	会话的时间戳。获取这个时间戳之前的会话列表。首次可传 0，后续可以使用返回的 RCConversation 对象的 operationTime 属性值，作为下一次查询的 startTime。

参数名	参数类型	描述
count	int	获取的数量。当实际取回的会话数量小于 count 值时，表明已取完数据。
callback	IRCIMIWGetConversationsCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetConversationsCallback? callback = IRCIMIWGetConversationsCallback(onSuccess:
(List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversationsFromTagByPage(tagId, timestamp, count, callback:callback);
```

按标签获取未读消息数

获取携带指定标签的所有会话的未读消息数。

方法

```
Future<int> getUnreadCountByTag(String tagId, bool contain, {IRCIMIWGetUnreadCountCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
contain	bool	是否包含免打扰会话。
callback	IRCIMIWGetUnreadCountCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUnreadCountCallback? callback = IRCIMIWGetUnreadCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUnreadCountByTag(tagId, contain, callback:callback);
```

清除标签对应会话的未读消息数

清除携带指定标签的所有会话的未读消息数。

方法

```
Future<int> clearMessagesUnreadStatusByTag(String tagId, {IRCIMIWClearMessagesUnreadStatusByTagCallback?
callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
callback	IRCIMIWClearMessagesUnreadStatusByTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearMessagesUnreadStatusByTagCallback? callback =
IRCIMIWClearMessagesUnreadStatusByTagCallback(onSuccess: (bool? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.clearMessagesUnreadStatusByTag(tagId, callback:callback);
```

删除标签对应的会话

删除指定标签下的全部会话，同时解除这些会话和标签的绑定关系。删除成功后，会话不再携带指定的标签。这些会话收到新消息时，会产生新的会话。

方法

```
Future<int> clearConversationsByTag(String tagId, bool deleteMessage,
{IRCIMIWClearConversationsByTagCallback? callback});
```

参数说明

参数名	参数类型	描述
tagId	String	标签唯一标识，字符型，长度不超过 10 个字。
deleteMessage	bool	是否删除消息
callback	IRCIMIWClearConversationsByTagCallback	事件回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearConversationsByTagCallback? callback = IRCIMIWClearConversationsByTagCallback(onSuccess:
(bool? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.clearConversationsByTag(tagId, deleteMessage, callback:callback);
```

您可以通过 deleteMessage 参数配置是否同时清除这些会话对应的本地消息。

- 如果不删除会话对应的本地消息，再接收到新消息时，可以看到历史聊天记录。
- 如果删除会话对应的本地消息，再接收到新消息时，无法看到历史聊天记录。如果开通了单群聊消息云存储服务，服务端仍保存有消息历史。如需删除，请使用删除服务端历史消息接口。

免打扰功能概述

更新时间:2024-08-30

「免打扰功能」用于控制用户在客户端设备离线时，是否可针对离线消息接收推送通知。

- **客户端为离线状态**：会话中有新离线消息时，用户默认通过推送通道收到消息且默认弹出通知。设置免打扰后，融云服务端不会为相关消息触发推送。
- **客户端在后台运行**：会话中有新消息时，用户直接收到消息。如果使用 IMLib，您需要自行判断 App 是否在后台运行，并根据业务需求自行实现本地通知弹窗。

前提条件

请在使用「免打扰功能」前检查是否已集成第三方推送，是否已为用户启用了推送服务。

免打扰设置维度

客户端 SDK 支持对单聊、群聊、系统会话业务进行以下多个维度的免打扰设置：

- App 的免打扰设置
- 按指定会话类型设置免打扰级别
- 按会话设置免打扰级别
- 全局免打扰

App 的免打扰设置

以 App Key 为单位，设置整个应用所有用户的默认免打扰级别。默认未设置，等同于全部消息都接收通知。该级别的配置暂未在控制台开放，如有需要，请提交工单。

- **全部消息均通知**：当前 App 下的用户可针对任何消息接收推送通知。
- **未设置**：默认全部消息都通知。
- **仅 @ 消息通知**：当前 App 下，仅针对提及（@）指定用户和群组全体成员的消息向离线用户发送推送通知。
- **仅 @ 指定用户通知**：当前 App 下，用户仅针对提及（@）当前用户的消息接收推送通知。例如：仅张三会接收且仅接收“@张三 Hello” 的消息的通知。
- **仅 @群全员通知**：当前 App 下，用户仅针对提及（@）群组全体成员的消息接收推送通知。
- **都不接收通知**：当前 App 下，用户不针对任何消息接收推送通知，即任何离线消息都不会触发推送通知。
- **除 @ 消息外群聊消息不发推送**：当前 App 下，用户针对单聊消息、提及（@）指定用户的消息、和提及（@）群组全体成员的消息接收推送通知。

融云服务端判断是否需要推送时，App 级别的免打扰配置的优先级最低。如果存在以下任何一种用户级别的免打扰配置，以用户级别配置为准：

- 按指定会话类型设置免打扰级别

- 按会话设置免打扰级别
- 全局免打扰

按会话类型设置免打扰级别

> 提示
>
> 该功能属于用户级别设置。

客户端 SDK 免打扰级别配置，定义在 RCIMIWPushNotificationLevel 枚举类中，允许用户为会话类型（单聊、群聊、超级群、系统会话）配置触发推送通知的消息类别，或完全关闭通知。提供以下六个级别：

枚举	说明
allMessage	与融云服务端断开连接后，当前用户可针对指定类型会话中的所有消息接收通知。
none	未设置。未设置时均为此初始状态。
mention	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）全部群成员的消息接收通知。
blocked	当前用户针对指定类型的会话中的任何消息都不接收推送通知。

具体设置方法详见按会话类型设置免打扰。

融云服务端判断是否需要为用户发送推送通知时，如果同时存在以下任何一种用户级别的免打扰配置，以下列配置为准：

- 按会话设置免打扰级别
- 全局免打扰

按会话设置免打扰级别

> 提示
>
> 该功能属于用户级别设置。

客户端 SDK 提供 RCIMIWPushNotificationLevel，允许用户为会话配置触发通知的消息类别，或完全关闭通知。提供以下六个级别：

枚举	说明
allMessage	与融云服务端断开连接后，当前用户可针对指定类型会话中的所有消息接收通知。
none	未设置。未设置时均为此初始状态。
mention	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）全部群成员的消息接收通知。

枚举	说明
blocked	当前用户针对指定类型的会话中的任何消息都不接收推送通知。

具体设置方法详见按会话设置免打扰。

融云服务端判断是否需要为用户发送推送通知时，如果该用户已配置全局免打扰，则已全局免打扰的配置细节为准。

全局免打扰

客户端 SDK 提供 RCIMIWPushNotificationQuietHoursLevel，允许用户配置何时接收通知以及触发通知的消息类别。提供了以下三个级别：

枚举	说明
none	未设置。如未设置，SDK 会依次查询消息所属群的用户级别免打扰设置及其他非用户级别设置，再判断是否需要推送通知。
mentionMessage	与融云服务端断开连接后，当前用户仅在指定时段内针对指定会话中提及（@）当前用户和全体群成员的消息接收通知。
blocked	当前用户在指定时段内针对任何消息都不接收推送通知。

具体设置方法详见全局免打扰。

免打扰设置的优先级

针对单聊、群聊、系统会话、融云服务端会遵照以下顺序搜索免打扰配置。优先级从左至右依次降低，以优先级最高的配置为准判断是否需要触发推送：

全局免打扰设置（用户级）> 指定会话类型的免打扰设置（用户级）> 指定会话的免打扰设置（用户级）> App 级的免打扰设置

API 接口列表

下表描述了适用于单聊、群聊、系统会话的免打扰配置 API 接口。

免打扰配置维度	客户端 API	服务端 API
设置指定时段内，应用全局的免打扰级别。	详见全局免打扰。	详见 设置用户免打扰时段
设置指定类型会话的免打扰级别	详见按会话类型设置免打扰。	详见设置会话类型免打扰。
设置指定会话的免打扰级别	详见按会话设置免打扰。	详见设置会话免打扰。
设置 App 级免打扰级别	客户端 SDK 不提供 API。	服务端不提供该 API。

按会话设置免打扰　　设置会话的消息免打扰状态

更新时间:2024-08-30

方法

即时通讯业务用户为指定会话或超级群设置免打扰级别，支持单聊、群聊、超级群会话。

```
Future<int> changeConversationNotificationLevel(RCIMIWConversationType type, String targetId, String?
channelId, RCIMIWPushNotificationLevel level, {IRCIMIWChangeConversationNotificationLevelCallback?
callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型。请注意以下限制： • **超级群会话类型**：如在 2022.09.01 之前开通超级群业务，默认不支持为单个超级群会话所有消息设置免打扰级别（“所有消息”指所有频道中的消息和不属于任何频道的消息）。该接口仅设置指定超级群会话（`targetId`）中不属于任何频道的消息的免打扰状态级别。如需修改请提交工单。 • **聊天室会话类型**：不支持，因为聊天室消息默认不支持消息推送提醒。
targetId	String	会话 ID/超级群ID
channelId	String	超级群的会话频道 ID。其他类型传 null 即可。 • 如果传入频道 ID，则针对该指定频道设置消息免打扰级别。如果不指定频道 ID，则对所有超级群消息生效。 • **注意**：2022.09.01 之前开通超级群业务的客户，如果不指定频道 ID，则默认传 "" 空字符串，即仅针对指定超级群会话（`targetId`）中不属于任何频道的消息设置免打扰状态级别。如需修改请提交工单。
level	RCIMIWPushNotificationLevel	免打扰级别，参考免打扰概述
callback	IRCIMIWChangeConversationNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeConversationNotificationLevelCallback? callback =
IRCIMIWChangeConversationNotificationLevelCallback(onConversationNotificationLevelChanged: (int? code) {
//...
});

int? ret = await engine?.changeConversationNotificationLevel(type, targetId, channelId, level,
callback:callback);
```

回调方法

- **onConversationNotificationLevelChanged**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
RCIMIWPushNotificationLevel? level)? onConversationNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onConversationNotificationLevelChanged = (int? code, RCIMIWConversationType? type, String?
targetId, String? channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

获取会话的免打扰状态

查询当前用户为指定会话设置的免打扰级别

方法

```
Future<int> getConversationNotificationLevel(RCIMIWConversationType type, String targetId, String?
channelId, {IRCIMIWGetConversationNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型。请注意以下限制： • 超级群会话类型：如在 2022.09.01 之前开通超级群业务，默认不支持为单个超级群会话所有消息设置免打扰级别（“所有消息”指所有频道中的消息和不属于任何频道的消息）。该接口仅设置指定超级群会话（`targetId`）中不属于任何频道的消息的免打扰状态级别。如需修改请提交工单。 • 聊天室会话类型：不支持，因为聊天室消息默认不支持消息推送提醒。
targetId	String	会话 ID/超级群ID
channelId	String	超级群的会话频道 ID。其他类型传 null 即可。 • 如果传入频道 ID，则针对该指定频道设置消息免打扰级别。如果不指定频道 ID，则对所有超级群消息生效。 • 注意：2022.09.01 之前开通超级群业务的客户，如果不指定频道 ID，则默认传 "" 空字符串，即仅针对指定超级群会话（`targetId`）中不属于任何频道的消息设置免打扰状态级别。如需修改请提交工单。
callback	IRCIMIWGetConversationNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetConversationNotificationLevelCallback? callback =
IRCIMIWGetConversationNotificationLevelCallback(onSuccess: (RCIMIWPushNotificationLevel? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversationNotificationLevel(type, targetId, channelId, callback:callback);
```

回调方法

- **onConversationNotificationLevelLoaded**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
RCIMIWPushNotificationLevel? level)? onConversationNotificationLevelLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
level	RCIMIWPushNotificationLevel	当前会话的消息通知级别

代码示例

```
engine?.onConversationNotificationLevelLoaded = (int? code, RCIMIWConversationType? type, String?
targetId, String? channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

获取免打扰状态列表

获取所有设置了消息免打扰的会话列表。

方法

```
Future<int> getBlockedConversations(List<RCIMIWConversationType> conversationTypes, String? channelId,
{IRCIMIWGetBlockedConversationsCallback? callback});
```

参数说明

```
Future<int> getBlockedConversations(List<RCIMIWConversationType> conversationTypes, String? channelId,
{IRCIMIWGetBlockedConversationsCallback? callback});
```

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetBlockedConversationsCallback? callback = IRCIMIWGetBlockedConversationsCallback(onSuccess:
(List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getBlockedConversations(conversationTypesInt, channelId, callback:callback);
```

回调方法

- **onBlockedConversationsLoaded**

```
Function(int? code, List<RCIMIWConversationType>? conversationTypes, String? channelId,
List<RCIMIWConversation>? conversations)? onBlockedConversationsLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
conversationTypes	List<RCIMIWConversationType>	会话类型集合
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
conversations	List<RCIMIWConversation>	获取到的会话集合

代码示例

```
engine?.onBlockedConversationsLoaded = (int? code, List<RCIMIWConversationType>? conversationTypes,
String? channelId, List<RCIMIWConversation>? conversations) {
//...
};
```

按会话类型设置免打扰　　设置会话的消息免打扰状态

更新时间:2024-08-30

即时通讯业务用户为指定会话或超级群设置免打扰级别

方法

```
Future<int> changeConversationTypeNotificationLevel(RCIMIWConversationType type,
RCIMIWPushNotificationLevel level, {IRCIMIWChangeConversationTypeNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
level	RCIMIWPushNotificationLevel	消息通知级别
callback	IRCIMIWChangeConversationTypeNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeConversationTypeNotificationLevelCallback? callback =
IRCIMIWChangeConversationTypeNotificationLevelCallback(onConversationTypeNotificationLevelChanged: (int?
code) {
//...
});

int? ret = await engine?.changeConversationTypeNotificationLevel(type, level, callback:callback);
```

回调方法

- **onConversationTypeNotificationLevelChanged**

```
Function(int? code, RCIMIWConversationType? type, RCIMIWPushNotificationLevel? level)?
onConversationTypeNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onConversationTypeNotificationLevelChanged = (int? code, RCIMIWConversationType? type,
RCIMIWPushNotificationLevel? level) {
//...
};
```

多端同步阅读状态　　同步消息未读状态

更新时间:2024-08-30

多端登录时，通知其它终端同步某个会话的消息未读状态。

方法

```
Future<int> syncConversationReadStatus(RCIMIWConversationType type, String targetId, String? channelId,
int timestamp, {IRCIMIWSyncConversationReadStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	会话中已读的最后一条消息的发送时间戳
callback	IRCIMIWSyncConversationReadStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSyncConversationReadStatusCallback? callback =
IRCIMIWSyncConversationReadStatusCallback(onConversationReadStatusSynced: (int? code) {
//...
});

int? ret = await engine?.syncConversationReadStatus(type, targetId, channelId, timestamp,
callback:callback);
```

回调方法

- **onConversationReadStatusSynced**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId, int? timestamp)?
onConversationReadStatusSynced;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	会话中已读的最后一条消息的发送时间戳

代码示例

```
engine?.onConversationReadStatusSynced = (int? code, RCIMIWConversationType? type, String? targetId,
String? channelId, int? timestamp) {
//...
};
```

监听同步消息未读状态

当用户调用syncConversationReadStatus同步消息未读状态时，远端用户会收到该回调

方法

```
Function(RCIMIWConversationType? type, String? targetId, int? timestamp)?
onConversationReadStatusSyncMessageReceived;
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
timestamp	int	时间戳

代码示例

```
engine?.onConversationReadStatusSyncMessageReceived = (RCIMIWConversationType? type, String? targetId,
int? timestamp) {
//...
};
```

群组业务概述

更新时间:2024-08-30

群聊是即时通讯类应用中常见的多人通讯方式，一般包含两个及以上的用户。融云的群组业务支持丰富的群组成员管理、禁言管理等特性，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。

群组业务要点如下：

- 融云只负责将消息传达给群组中的所有用户，不维护群组成员的资料（头像、名称、群成员名片等），需要由开发者应用服务器维护。
- 创建/解散/加入/退出群组等群组管理操作，必须由 App 服务器请求融云服务端 API 实现。融云客户端 SDK 不提供相应方法。详见下方群组管理功能。
- App Key 下可创建的群组数量没有限制，单个群组默认成员上限为 3000 人。可提交工单修改群成员人数上限。
- 单个用户可加入的群组数量无限制。
- 从控制台 IM 服务管理页面为 App Key 开启单群聊消息云端存储服务后，可使用融云提供的消息存储服务，实现消息历史记录漫游。**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。

服务配置

客户端 SDK 默认支持群组业务，不需要申请开通。

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

群组管理功能

融云不会托管用户，也不管理群组的业务逻辑，因此群的业务逻辑全部需要在 App 服务器进行实现。

> 提示
>
> **群主、群管理员、群公告、邀请入群、群号搜索等均为群组业务逻辑，需在 App 侧自行实现。**

对于客户端开发人员来说，创建群组、解散等基础管理操作只需要与 App 自身的业务服务端交互即可，由 App 服务端负责调用相应的融云服务端 API（Server API）接口完成相关操作。

下表列出了融云服务端提供的群组基础管理接口。注意，客户端不提供群组管理的 API。

功能分类	功能描述	融云服务端 API
创建、解散群组	提供创建者用户 ID、群组 ID、和群名称，向融云服务端申请建群。如解散群组，则群成员关系不复存在。	创建群组、解散群组

功能分类	功能描述	融云服务端 API
加入、退出群组	加入群组后，默认可查看入群以后产生的新消息。退出群组后，不再接收该群的新消息。	加入群组、退出群组
修改融云服务端的群组信息	修改在融云推送服务中使用的群组信息。	刷新群组信息
查询群组成员	查询指定群组所有成员的用户 ID 信息。	查询群组成员
查询用户所在群组	根据用户 ID 查询该用户加入的所有群组，返回群组 ID 及群组名称。融云不存储群组资料信息，群组资料及群成员信息需要开发者在应用服务器自行维护，如应用服务端维护的用户群组关系有缺失时，可通过此接口来核对校验。	查询用户所在群组
同步用户所在群组	向融云服务端同步指定用户当前所加入的所有群组，防止应用中的用户群组信息与融云服务端的用户所属群信息不一致。如果在集成融云服务前 App Server 上已有群组及成员数据，第一次连接融云服务器时，可使用此接口向融云同步已有的用户与群组对应关系。	同步用户所在群组
群组单人禁言	在指定的单个群组中或全部群组中，禁言一个或多个用户。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。	单人禁言
群组全体禁言	将群组全体成员禁言。被禁言群组的所有成员均不能发送消息，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。	全体禁言
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。	全体成员禁言白名单

群聊消息功能

群聊消息功能与单聊业务类似，共用部分 API 及配置。

功能	描述	客户端 API	融云服务端 API
发送消息	可发送普通消息与媒体消息，例如文本、图片、GIF 等，或自定义消息。支持在发送消息时添加 @ 信息。	发送消息	发送群聊消息
发送群聊定向消息	可发送普通消息与媒体消息给群组中的指定的一个或多个成员，其他成员不会收到该消息。	发送群定向消息	发送群聊定向消息
接收消息	监听并实时接收消息，或在客户端上线时接收离线消息。	接收消息	不适用
群聊消息已读回执	发送群消息后如需要查看消息的阅读状态，需要先发送回执请求，在通过接受者的响应获取已读数据。	群聊消息回执	不适用
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 天内的所有群消息，支持调整存储时长与存储的群消息数量。	不提供该 API	不提供该 API
离线消息推送	离线状态下，群组中有新消息时，支持 Push 通知。	推送开发指南	不提供该 API
撤回消息	消息发送成功后可撤回该条消息。	撤回消息	撤回消息
本地搜索消息	消息存储在本地（移动端），支持按关键字或用户搜索本地指定会话的消息内容。	搜索消息	不提供该 API

功能	描述	客户端 API	融云服务端 API
获取历史消息	从本地数据库或远端获取历史消息。注意，从远端获取历史消息需要开通单群聊消息云存储服务，默认存储时长为 6 个月。	获取历史消息	不提供该 API
获取历史消息日志	融云服务端可以保存 APP 内所有会话的历史消息记录，历史消息记录以日志文件方式提供，并已经过压缩。您可以使用服务端 API 获取、删除指定 App 的历史消息日志	不提供该 API	获取历史消息日志
本地插入消息	在本地数据库中插入消息。本地插入的消息不会实际发送给服务器和对方。	插入消息	不适用
删除消息	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。	删除消息	消息清除
单群聊消息扩展	为原始消息增加状态标识（扩展数据为 KV 键值对），提供添加、删除、查询扩展信息的接口。	消息扩展	单/群聊消息扩展
自定义消息类型	如果内置消息类型满足不了您的需求，可以自定义消息类型。支持自定义普通消息类型与自定义媒体消息类型。	自定义消息类型	不适用

默认新入群的成员的用户仅可接收加入群组后产生的消息。如果需要查看入群之前的历史消息，请为 App Key 开启以下两项服务（请注意区分开发/生产环境）：

从控制台 IM 服务管理 页面开启**单群聊消息云端存储**服务。开启该服务后，可使用融云提供的消息存储服务，实现消息历史记录漫游。**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。

- 从控制台免费基础功能 页面开启**新用户获取加入群组前历史消息**。

群聊会话功能

群聊会话功能与单聊业务类似，共用部分 API 及配置。

功能	描述	客户端 API	融云服务端 API
获取会话	SDK 会根据收发的消息在本地数据库中生成对应会话。您可以从本地数据库获取 SDK 生成的会话列表。	获取会话	不提供该 API
处理会话未读消息数	获取或清除会话中的未读消息数，可用于 UI 展示。	会话未读数	不提供该 API
删除会话	从 SDK 生成的会话列表中删除一个会话或多个会话	删除会话	不提供该 API
会话草稿	保存一条草稿内容至指定会话。	会话草稿	不提供该 API
输入状态	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。	输入状态	不提供该 API
会话置顶	在会话列表中将指定会话置顶。	会话置顶	会话置顶
会话免打扰	控制用户在客户端设备离线时，是否可针对离线消息接收推送通知。支持按照会话或按会话类型设置免打扰。	免打扰功能概述	免打扰功能概述
多端同步阅读状态	在同一用户账户的多个设备间主动同步会话的阅读状态。	多端同步阅读状态	不适用

与聊天室和超级群的区别

您可以通过以下文档了解业务类型之间的区别及所有功能：

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

发送群定向消息

更新时间:2024-08-30

SDK 支持往群聊会话中发送定向消息。定向消息只会发送给指定用户，群聊会话中的其它用户不会收到这条消息。

当前仅支持发送普通消息，不支持发送媒体消息。

开通服务

使用发送群组定向消息功能无需开通服务。注意，如需将群组定向消息存入服务端历史消息记录，需要开通以下服务：

- 单群聊历史消息云存储服务，可前往控制台 IM 服务管理 页面为当前使用的 App Key 开启服务。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。
- 群定向消息云存储服务，需要提交工单 申请开通。

默认情况下，客户端发送与接收的群定向消息默认都不会存入历史消息服务，因此客户端调用获取历史消息的 API 时，从融云服务端返回的结果中不会包含当前用户发送、接收的群组定向消息。

发送群组定向普通消息

在群组中发送普通消息给指定的单个或多个用户。

方法

```
Future<int> sendGroupMessageToDesignatedUsers(RCIMIWMessage message, List<String> userIds,
{RCIMIWSendGroupMessageToDesignatedUsersCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	要发送的消息
userIds	List<String>	群成员集合
callback	RCIMIWSendGroupMessageToDesignatedUsersCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
RCIMIWSendGroupMessageToDesignatedUsersCallback? callback =
RCIMIWSendGroupMessageToDesignatedUsersCallback(onMessageSaved: (RCIMIWMessage? message) {
//...
}, onMessageSent: (int? code, RCIMIWMessage? message) {
//...
});

int? ret = await engine?.sendGroupMessageToDesignatedUsers(message, userIds, callback:callback);
```

回调方法

- **onGroupMessageToDesignatedUsersAttached**

  消息存入数据库的监听

```
Function(RCIMIWMessage? message)? onGroupMessageToDesignatedUsersAttached;
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	发送的消息内容

代码示例

```
engine?.onGroupMessageToDesignatedUsersAttached = (RCIMIWMessage? message) {
//...
};
```

- **onGroupMessageToDesignatedUsersSent**

  消息发送收到结果的回调

```
Function(int? code, RCIMIWMessage? message)? onGroupMessageToDesignatedUsersSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	发送的消息内容

代码示例

```
engine?.onGroupMessageToDesignatedUsersSent = (int? code, RCIMIWMessage? message) {
//...
};
```

聊天室概述

更新时间:2024-08-30

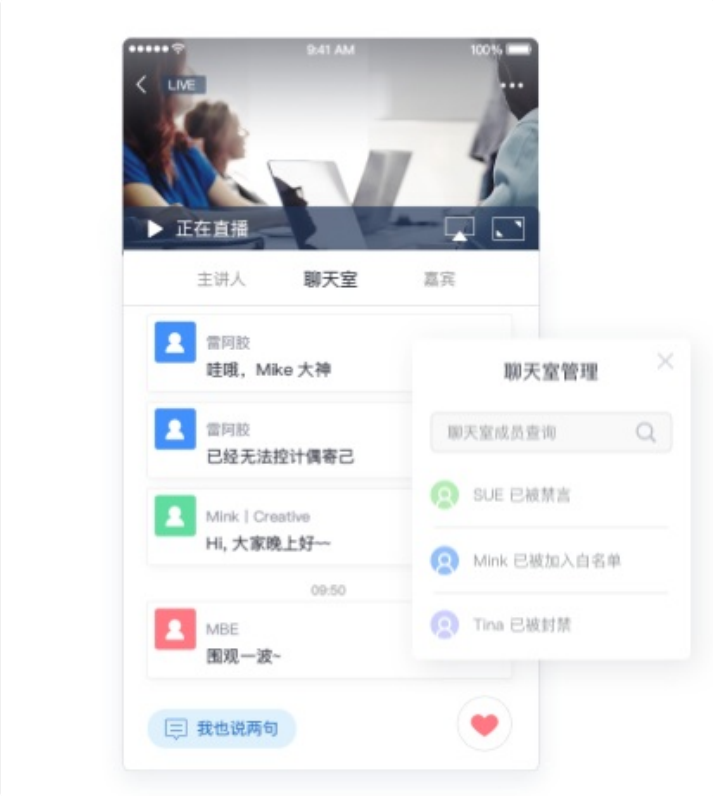
聊天室（Chatroom）提供了一种不设用户上限，支持高并发消息处理的业务形态，可用于直播、社区、游戏、广场交友、兴趣讨论等场景。聊天室业务要点如下：

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

客户端 UI 框架参考设计

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

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



服务配置

客户端 SDK 默认支持聊天室，不需要申请开通。

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

聊天室功能接口

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

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

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

与群组和超级群的区别

您可以通过以下文档了解业务类型之间的区别及所有功能：

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

聊天室服务配置

更新时间:2024-08-30

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

聊天室服务配置主要在**免费基础功能**和 **IM 服务管理**页面。

免费基础功能

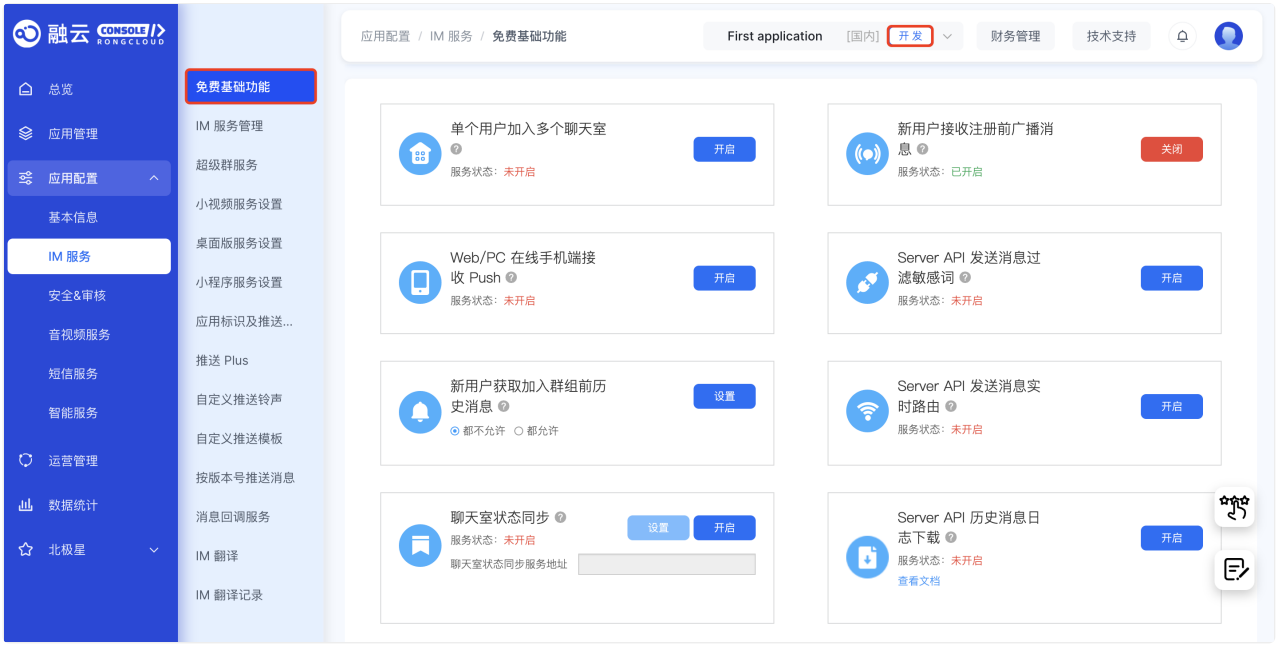
以下是聊天室业务提供的免费基础功能：

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

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

修改服务配置

访问控制台免费基础功能 页面，可调整聊天室业务相关的免费基础功能配置。



IM 旗舰版/尊享版功能

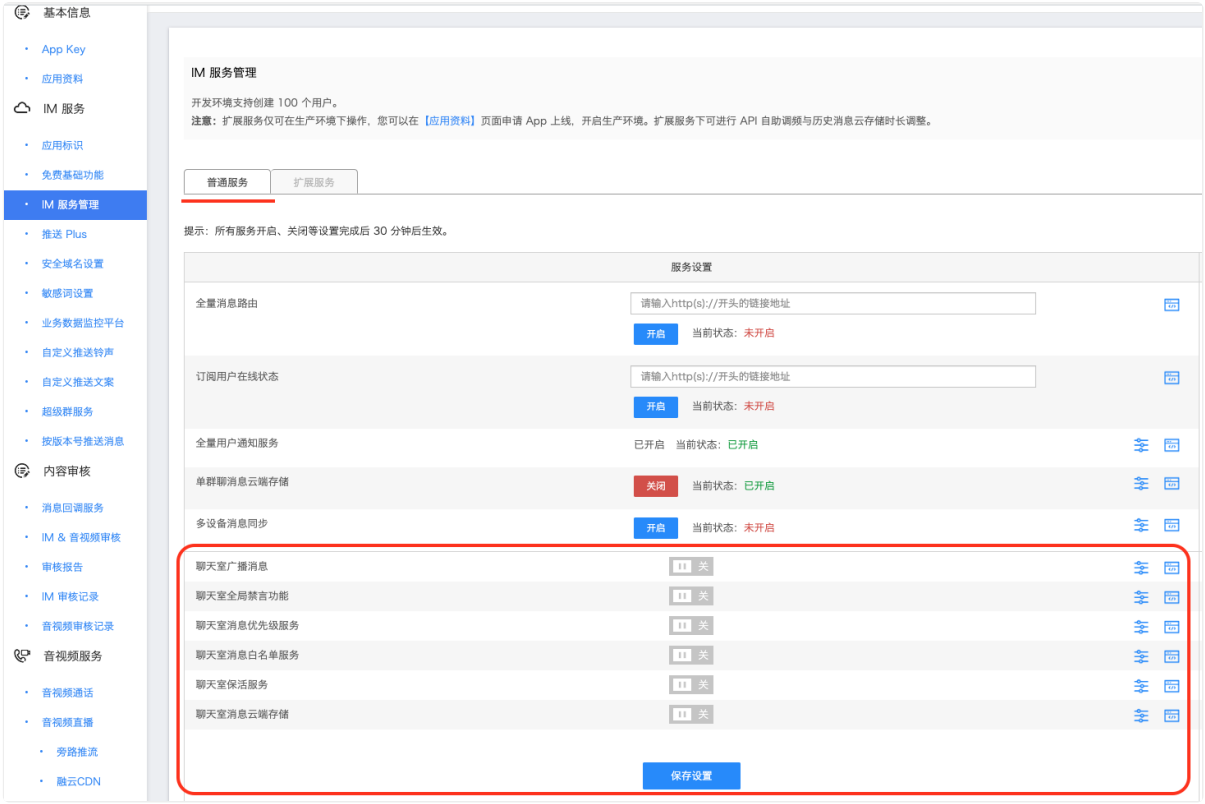
下图显示了控制台 IM 服务管理 页面与聊天室业务相关的普通服务配置。

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

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

修改服务配置

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



您还可以在扩展服务标签下对部分服务的具体配置进行调整。

加入聊天室

更新时间:2024-08-30

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

如需支持单个用户加入多个聊天室，请在控制台打开该配置，详见聊天室服务配置。

加入聊天室

方法

```
Future<int> joinChatRoom(String targetId, int messageCount, bool autoCreate,
{IRCIMIWJoinChatRoomCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
messageCount	int	进入聊天室拉取消息数目，-1 时不拉取任何消息，0 时拉取 10 条消息，最多只能拉取 50
autoCreate	bool	是否创建聊天室，TRUE 如果聊天室不存在，sdk 会创建聊天室并加入，如果已存在，则直接加入
callback	IRCIMIWJoinChatRoomCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWJoinChatRoomCallback? callback = IRCIMIWJoinChatRoomCallback(onChatRoomJoined: (int? code,
String? targetId) {
//...
});

int? ret = await engine?.joinChatRoom(targetId, messageCount, autoCreate, callback:callback);
```

回调方法

- **onChatRoomJoined**

```
Function(int? code, String? targetId)? onChatRoomJoined;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID

代码示例

```
engine?.onChatRoomJoined = (int? code, String? targetId) {
//...
};
```

- **onChatRoomJoining**

正在加入聊天室的回调

```
Function(String? targetId)? onChatRoomJoining;
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室 ID

代码示例

```
engine?.onChatRoomJoining = (String? targetId) {
//...
};
```

- SDK 提供聊天室重连机制，待链接状态恢复后，如果曾经加入过聊天室，没有退出，则重新加入聊天室。

退出聊天室

更新时间:2024-08-30

退出聊天室支持以下几种情况：

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

聊天室离线成员自动退出机制

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

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

> 提示
>
> - 默认预设条件均要求聊天室中必须要有新消息产生，否则无法触发踢出动作。如果聊天室中没有消息产生，则无法将异常用户踢出聊天室。
> - 如需修改默认行为对新消息的依赖，请提交工单申请开通**聊天室成员异常掉线实时踢出**。开通该服务后，服务端会通过 SDK 行为判断用户是否处于异常状态，最迟 5 分钟可以将异常用户踢出聊天室。
> - 如需保护特定用户，即不自动踢出指定用户（如某些应用场景下可能希望用户驻留聊天室），可使用 Server API 提供的聊天室用户白名单功能。

主动退出聊天室

客户端用户可主动退出聊天室。

方法

```
Future<int> leaveChatRoom(String targetId, {IRCIMIWLeaveChatRoomCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
callback	IRCIMIWLeaveChatRoomCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWLeaveChatRoomCallback? callback = IRCIMIWLeaveChatRoomCallback(onChatRoomLeft: (int? code,
String? targetId) {
//...
});

int? ret = await engine?.leaveChatRoom(targetId, callback:callback);
```

回调方法

- **onChatRoomLeft**

```
Function(int? code, String? targetId)? onChatRoomLeft;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID

代码示例

```
engine?.onChatRoomLeft = (int? code, String? targetId) {
//...
};
```

聊天室状态监听　　监听聊天室状态改变

更新时间:2024-08-30

- 当聊天室状态改变时会回调该方法
- 聊天室的状态改变包括聊天室被重置、用户调用IM Server API 手动销毁聊天室、IM Server 自动销毁聊天室

方法

```
Function(String? targetId, RCIMIWChatRoomStatus? status)? onChatRoomStatusChanged;
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
status	RCIMIWChatRoomStatus	聊天室变化的状态

代码示例

```
engine?.onChatRoomStatusChanged = (String? targetId, RCIMIWChatRoomStatus? status) {
//...
};
```

聊天室成员变化监听

- 当有用户加入、离开聊天室时会回调该方法

此功能需要提交工单开通后才能使用。

方法

```
Function(String? targetId, List<RCIMIWChatRoomMemberAction>? actions)? onChatRoomMemberChanged;
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
actions	List<RCIMIWChatRoomMemberAction>	发生的事件

代码示例

```
engine?.onChatRoomMemberChanged = (String? targetId, List<RCIMIWChatRoomMemberAction>? actions) {
//...
};
```

获取聊天室历史消息　　开通服务

更新时间:2024-08-30

使用获取聊天室远端历史记录功能要求开通 聊天室消息云端存储 服务。使用前请确认已开通服务。开通后聊天室历史消息保存在云端，默认保存 2 个月。

获取聊天室远端历史记录

当退出聊天室时，会清除本地聊天室历史消息。当再次加入聊天室时，如需获取之前的历史消息，可调用此接口。

方法

```
Future<int> getChatRoomMessages(String targetId, int timestamp, RCIMIWTimeOrder order, int count,
{IRCIMIWGetChatRoomMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
timestamp	int	起始的消息发送时间戳
order	RCIMIWTimeOrder	拉取顺序 0:倒序，1:正序
count	int	要获取的消息数量，0 < count <= 50。
callback	IRCIMIWGetChatRoomMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetChatRoomMessagesCallback? callback = IRCIMIWGetChatRoomMessagesCallback(onSuccess:
(List<RCIMIWMessage>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getChatRoomMessages(targetId, timestamp, order, count, callback:callback);
```

回调方法

- **onChatRoomMessagesLoaded**

```
Function(int? code, String? targetId, List<RCIMIWMessage>? messages, int? syncTime)?
onChatRoomMessagesLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
messages	List<RCIMIWMessage>	加载到的消息
syncTime	int	下次拉取的时间戳

代码示例

```
engine?.onChatRoomMessagesLoaded = (int? code, String? targetId, List<RCIMIWMessage>? messages, int?
syncTime) {
//...
};
```

聊天室属性管理（KV）

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

> 提示
>
> - 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）。

添加聊天室 KV

方法

```
Future<int> addChatRoomEntry(String targetId, String key, String value, bool deleteWhenLeft, bool overwrite, {IRCIMIWAddChatRoomEntryCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
key	String	聊天室属性名称，Key 支持大小写英文字母、数字、部分特殊符号 + = - _ 的组合方式，最大长度 128 个字符
value	String	聊天室属性对应的值，最大长度 4096 个字符
deleteWhenLeft	bool	用户掉线或退出时，是否自动删除该 Key、Value 值
overwrite	bool	如果当前 key 存在，是否进行覆盖

参数名	参数类型	描述
callback	IRCIMIWAddChatRoomEntryCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWAddChatRoomEntryCallback? callback = IRCIMIWAddChatRoomEntryCallback(onChatRoomEntryAdded: (int?
code) {
//...
});

int? ret = await engine?.addChatRoomEntry(targetId, key, value, deleteWhenLeft, overwrite,
callback:callback);
```

回调方法

- **onChatRoomEntryAdded**

```
Function(int? code, String? targetId, String? key)? onChatRoomEntryAdded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
key	String	聊天室属性名称

代码示例

```
engine?.onChatRoomEntryAdded = (int? code, String? targetId, String? key) {
//...
};
```

添加多个聊天室 KV

方法

```
Future<int> addChatRoomEntries(String targetId, Map entries, bool deleteWhenLeft, bool overwrite,
{IRCIMIWAddChatRoomEntriesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
entries	Ma<String, String>	聊天室属性
deleteWhenLeft	bool	用户掉线或退出时，是否自动删除该 Key、Value 值
overwrite	bool	是否强制覆盖
callback	IRCIMIWAddChatRoomEntriesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWAddChatRoomEntriesCallback? callback = IRCIMIWAddChatRoomEntriesCallback(onChatRoomEntriesAdded:
(int? code, Map? errors) {
//...
});

int? ret = await engine?.addChatRoomEntries(targetId, entries, deleteWhenLeft, overwrite,
callback:callback);
```

回调方法

- **onChatRoomEntriesAdded**

```
Function(int? code, String? targetId, Map? entries, Map? errorEntries)? onChatRoomEntriesAdded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
entries	Ma<String, String>	聊天室属性
errorEntries	Ma<String, int>	发生错误的属性

代码示例

```
engine?.onChatRoomEntriesAdded = (int? code, String? targetId, Map? entries, Map? errorEntries) {
//...
};
```

加载聊天室 KV

方法

```
Future<int> getChatRoomEntry(String targetId, String key, {IRCIMIWGetChatRoomEntryCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
key	String	聊天室属性键值
callback	IRCIMIWGetChatRoomEntryCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetChatRoomEntryCallback? callback = IRCIMIWGetChatRoomEntryCallback(onSuccess: (Map? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getChatRoomEntry(targetId, key, callback:callback);
```

回调方法

- **onChatRoomEntryLoaded**

```
Function(int? code, String? targetId, Map? entry)? onChatRoomEntryLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
entry	Ma<String, String>	获取到的属性。

代码示例

```
engine?.onChatRoomEntryLoaded = (int? code, String? targetId, Map? entry) {
//...
};
```

加载某个聊天室所有 KV

方法

```
Future<int> getChatRoomAllEntries(String targetId, {IRCIMIWGetChatRoomAllEntriesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
callback	IRCIMIWGetChatRoomAllEntriesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetChatRoomAllEntriesCallback? callback = IRCIMIWGetChatRoomAllEntriesCallback(onSuccess: (Map?
t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getChatRoomAllEntries(targetId, callback:callback);
```

回调方法

- **onChatRoomAllEntriesLoaded**

```
Function(int? code, String? targetId, Map? entries)? onChatRoomAllEntriesLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID

参数名	参数类型	描述
entries	Ma<String, String>	获取到的属性集合。

代码示例

```
engine?.onChatRoomAllEntriesLoaded = (int? code, String? targetId, Map? entries) {
//...
};
```

移除聊天室 KV

方法

```
Future<int> removeChatRoomEntry(String targetId, String key, bool force,
{IRCIMIWRemoveChatRoomEntryCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
key	String	聊天室属性键值
force	bool	是否强制删除
callback	IRCIMIWRemoveChatRoomEntryCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

示例代码

```
IRCIMIWRemoveChatRoomEntryCallback? callback =
IRCIMIWRemoveChatRoomEntryCallback(onChatRoomEntryRemoved: (int? code) {
//...
});

int? ret = await engine?.removeChatRoomEntry(targetId, key, force, callback:callback);
```

回调方法

- **onChatRoomEntryRemoved**

```
Function(int? code, String? targetId, String? key)? onChatRoomEntryRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
key	String	聊天室属性键值

代码示例

```
engine?.onChatRoomEntryRemoved = (int? code, String? targetId, String? key) {
//...
};
```

移除聊天室多个 KV

方法

```
Future<int> removeChatRoomEntries(String targetId, List<String> keys, bool force,
{IRCIMIWRemoveChatRoomEntriesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	聊天室会话 ID
keys	List<String>	聊天室属性
force	bool	是否强制覆盖
callback	IRCIMIWRemoveChatRoomEntriesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveChatRoomEntriesCallback? callback =
IRCIMIWRemoveChatRoomEntriesCallback(onChatRoomEntriesRemoved: (int? code) {
//...
});

int? ret = await engine?.removeChatRoomEntries(targetId, keys, force, callback:callback);
```

回调方法

- **onChatRoomEntriesRemoved**

```
Function(int? code, String? targetId, List<String>? keys)? onChatRoomEntriesRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
keys	List<String>	聊天室属性键值集合

代码示例

```
engine?.onChatRoomEntriesRemoved = (int? code, String? targetId, List<String>? keys) {
//...
};
```

聊天室时 KV 同步完成的回调

刚加入聊天室时 KV 同步完成的回调

方法

```
Function(String? roomId)? onChatRoomEntriesSynced;
```

参数说明

参数名	参数类型	描述
roomId	String	聊天室 ID

代码示例

```
engine?.onChatRoomEntriesSynced = (String? roomId) {
//...
};
```

聊天室 KV 变化的回调

如果刚进入聊天室时存在 KV，会通过此回调将所有 KV 返回，再次回调时为其他人设置或者修改 KV

方法

```
Function(RCIMIWChatRoomEntriesOperationType? operationType, String? roomId, Map? entries)?
onChatRoomEntriesChanged;
```

参数说明

参数名	参数类型	描述
operationType	RCIMIWChatRoomEntriesOperationType	操作的类型
roomId	String	聊天室 ID
entries	Ma<String, String>	发送变化的 KV

代码示例

```
engine?.onChatRoomEntriesChanged = (RCIMIWChatRoomEntriesOperationType? operationType, String? roomId,
Map? entries) {
//...
};
```

超级群概述

更新时间:2024-08-30

融云超级群（UltraGroup）提供了一种新的群组业务形态。超级群不设置群成员人数上限，允许用户在超级社群中建立社交关系、在海量信息中聚焦自己感兴趣的内容，帮助开发者打造高用户黏性的群体。超级群组成员最多可加入 100 个超级群，每个超级群下的不同频道之间共享一份超级群成员关系。App 内的超级群数量没有限制。

超级群业务的会话类型（RCIMIWConversationType）为 `ultraGroup`，用 `targetId` 表示超级群 ID，`channelId` 表示超级群频道 ID。

开通服务

超级群功能需要在控制台超级群服务页面开通。仅 **IM** 尊享版支持开通超级群服务。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。

如何使用频道

超级群支持在群会话中创建独立的频道（客户端由 `channelId` 指定、对应服务端的 `busChannel`），超级群的会话、消息、未读数等消息数据和群组成员支持分频道进行聚合，各个频道之间消息独立。

频道按类型区分为公有频道与私有频道。公有频道对所有超级群成员开放（无需加入）。该超级群的所有成员都会接收公有频道下的消息。私有频道仅对该频道成员列表上的用户开放。有关私有频道的详细介绍，可参见超级群私有频道概述。

超级群业务提供一个 ID 为 `RCDefault` 的默认频道。`RCDefault` 频道对所有超级群成员开放，不可转为私有频道。

对于 App 业务来说，如果仅需实现类似群聊的业务，可以利用超级群无成员上限的特性构建大于 3000 人的超大群。这种场景下，可以让所有消息都在 `RCDefault` 默认频道中进行收发。建议在调用客户端、服务端 API 时指定频道 ID 为 `RCDefault`。

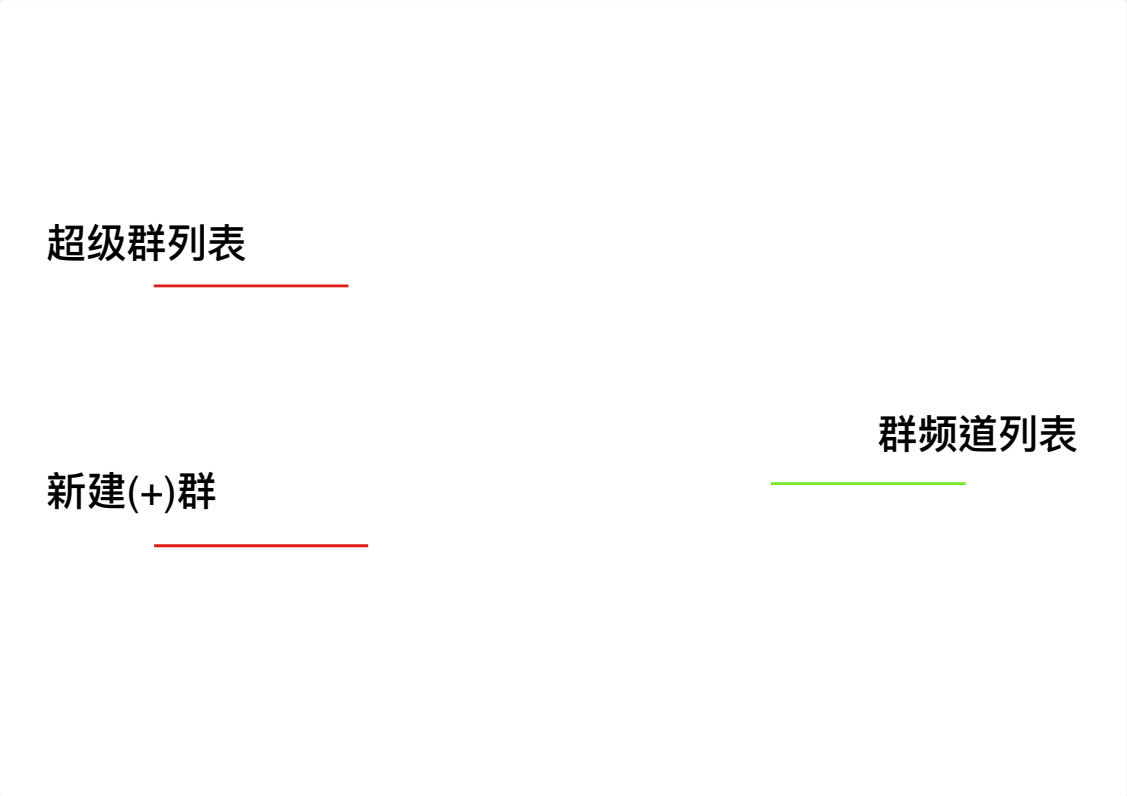
如果仅需实现类似 Discord 类业务，通过超级群频道功能构建子社区，推荐全部使用您自行创建的频道实现您的业务特性。默认频道（`RCDefault`）与自建频道的行为存在差异，全部使用自建频道可避免这种差异在实现 App 业务逻辑时造成限制。

> **提示**
>
> 如果您的 App / 环境在 2022.10.13 日之前开通超级群服务，则您的超级群服务中不存在 `RCDefault` 频道。在调用客户端、服务端 API 时如果不指定频道 ID，一般仅作用于不属于任何频道的消息，具体行为需参见各功能文档。融云支持客户调整服务至最新行为。该行为调整将影响客户端、服务端收发消息、获取会话、清除历史消息、禁言等多个功能。如有需要，请提交工单咨询详细方案。

客户端 UI 框架参考设计

超级群产品暂不提供 UI 组件。您可以参考以下 UI 框架设计了解超级群 App 的设计思路。

- 下图左侧红框中为超级群列表，即当前登录用户的超级群列表。红框底部的加号（+）按钮代表新建超级群。
- 上图绿框中为超级群频道（Channel）列表。超级群的每一个频道由频道 ID（`channelId`）指定。



超级群管理接口

融云不会托管用户，也不管理群组的业务逻辑，因此超级群的业务逻辑全部需要在 App 服务器进行实现。

对于客户端开发人员来说，创建群组、频道等基础管理操作只需要与 App 服务端交互即可。App 服务端需要调用相应的融云服务端 API（Server API）接口相关接口创建超级群、创建频道等其他管理操作。

下表列出了融云提供的超级群基础管理接口。

> 提示
>
> Server API 还提供超级群全体禁言、超级群用户禁言等更多管理接口。具体请参见 融云服务端超级群文档。

功能分类	功能描述	客户端 API	融云服务端 API
创建、解散超级群	提供创建者用户 ID、超级群 ID、和群名称，向融云服务端申请建群。如解散超级群，则群成员关系不复存在。	不提供该 API	创建超级群、解散超级群
加入、退出超级群	加入超级群后，默认可查看入群以后产生的新消息。退出超级群后，不再接受该群的新消息。	不提供该 API	加入超级群、退出超级群
修改融云服务端的超级群信息	修改在融云推送服务中使用的超级群信息。	不提供该 API	更新超级群信息
创建、删除群频道	在超级群会话中创建独立沟通的频道。如删除频道，将无法在频道中发送消息。	不提供该 API	创建频道、删除频道

功能分类	功能描述	客户端 API	融云服务端 API
查询群频道列表	加入超级群后，默认可查看入群以后产生的新消息。退出超级群后，不再接受该群的新消息。	不提供该 API	查询频道列表
变更群频道类型	超级群频道可以随时切换为公有频道或私有频道。	不提供该 API	变更频道类型
添加、删除私有频道成员	将超级群成员加入或移出指定频道的私有频道成员列表。在频道类型为私有频道时启用该成员列表的数据。	不提供该 API	添加私有频道成员、删除私有频道成员

创建超级群与频道

更新时间:2024-08-30

客户端 SDK 不提供创建超级群与创建群频道的接口。请使用融云服务端 API（Server API）的相关接口创建超级群、群频道，或进行其他管理操作。

> 提示
>
> 单个用户最多可以加入 100 个超级群。单个用户在每个群中最多可以加入或者创建 50 个频道。

本文仅简单介绍创建超级群与创建频道的基本流程。

创建超级群

创建超级群必须使用融云服务端 API（Server API），具体接口使用方法请参见服务端文档创建超级群。

基本流程

1. App 客户端请求 App 服务端（AppServer）创建超级群。
2. App 服务端调用融云 Server API 接口创建超级群，群组 ID 由 App 服务器自行生成
3. 超级群创建成功后，由 App 服务端返回给 App 客户端。

如何处理与展示超级群列表

AppServer 需要保存当前用户的超级群列表，并下发给 App，然后进行展示。

> 考虑到 App 由于自身业务需求，需要知晓当前用户的超级群列表（例如用户所在的超级群有等级权重等排序规则），而融云的超级群列表是通过消息产生的，因而两个列表可能不完全一样，建议由 App 按照自身业务保存用户的超级群列表。

创建群频道

创建超级群必须使用融云服务端 API（Server API），具体接口使用方法请参见服务端文档创建频道。

基本流程

如何处理与展示超级群频道列表

为方便在同一超级群下的不同用户按需看到自己需要的列表（例如用户对特定频道标星需要特别展示，APP 服务则需要按用

户记录），APP 服务应采取更为灵活的方式维护超级群的频道列表。

APP 服务端也可以按照需求将超级群分组（如超级群概述中的 UI 框架设计所示）。

获取频道列表

更新时间:2024-08-30

App 可按需使用客户端 SDK 与融云服务端 API 提供的能力，采取灵活的方式维护超级群的频道列表。

> 提示
>
> - 如果您的应用/环境在 2022.10.13 日及以后开通超级群服务，超级群业务中会包含一个 ID 为 `RCDefault` 的默认频道。如果发消息时不指定频道 ID，则该消息会发送到 `RCDefault` 频道中。在获取 `RCDefault` 频道的历史消息时，需要传入该频道 ID。
> - 如果您的应用/环境在 2022.10.13 日前已开通超级群服务，在发送消息时如果不指定频道 ID，则该消息不属于任何频道。获取历史消息时，如果不传入频道 ID，可获取不属于任何频道的消息。融云支持客户调整服务至最新行为。该行为调整将影响客户端、服务端收发消息、获取会话、清除历史消息、禁言等多个功能。如有需要，请提交工单咨询详细方案。

如何获取超级群全部频道列表

超级群下全部频道的列表可通过融云服务端 API (`/ultragroup/channel/get.json`) 获取。

注意，对单个用户来说，最多可以加入 100 个超级群，在每个超级群中最多可以加入或者创建 50 个频道。

> 提示
>
> 客户端 SDK 会通过频道中收发的消息在本地数据库中生成一个超级群频道列表。该列表仅包含了已在本地产生消息的频道，可能并非该超级群下全部频道的列表。

建议从 App 业务服务端维护超级群的频道列表。App 业务一般需要知晓当前用户的所加入的超级群，以及超级群下的频道列表，才能实现特定的业务功能。假设同一超级群下的不同用户需要展示个性化的频道列表（例如 App 需要在 UI 上展示用户标星的特定频道），则需要为用户保存超级群频道列表。

APP 服务端也可以按照需求将超级群分组（如超级群概述中的 UI 框架设计所示）。

获取本地指定超级群下的频道列表

客户端 SDK 会根据频道中收发的消息在本地数据库中生成对应频道的会话。您可以从本地数据库获取 SDK 生成的频道列表。获取到的频道列表按照时间倒序排列。

方法

```
Future<int> getConversationsForAllChannel(RCIMIWConversationType type, String targetId,
{IRCIMIWGetConversationsForAllChannelCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
callback	IRCIMIWGetConversationsForAllChannelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetConversationsForAllChannelCallback? callback =
IRCIMIWGetConversationsForAllChannelCallback(onSuccess: (List<RCIMIWConversation>? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversationsForAllChannel(type, targetId, callback:callback);
```

回调方法

- **onConversationsLoadedForAllChannel**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, List<RCIMIWConversation>?
conversations)? onConversationsLoadedForAllChannel;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
conversations	List<RCIMIWConversation>	获取到的会话集合

代码示例

```
engine?.onConversationsLoadedForAllChannel = (int? code, RCIMIWConversationType? type, String? targetId,
List<RCIMIWConversation>? conversations) {
//...
};
```

获取历史消息 获取历史消息

更新时间:2024-08-30

以下方法先从本地获取历史消息，本地有缺失的情况下会从服务端同步缺失的部分。当本地没有更多消息的时候，会从服务端拉取。

```
Future<int> getMessages(RCIMIWConversationType type, String targetId, String? channelId, int sentTime,
RCIMIWTimeOrder order, RCIMIWMessageOperationPolicy policy, int count, {IRCIMIWGetMessagesCallback?
callback});
```

该方法详细介绍参考loadMessages

从服务端获取特定批量消息

强制从服务端获取对应的消息。

1. 0 < messages 个数 ≦ 20
2. messages 所有数据必须是超级群类型且为同一个会话
3. message 有效值为 conversationType，targetId，channelId，messageUId，sentTime

方法

```
Future<int> getBatchRemoteUltraGroupMessages(List<RCIMIWMessage> messages,
{IRCIMIWGetBatchRemoteUltraGroupMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage >	获取的消息集合
callback	IRCIMIWGetBatchRemoteUltraGroupMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetBatchRemoteUltraGroupMessagesCallback? callback =
IRCIMIWGetBatchRemoteUltraGroupMessagesCallback(onSuccess: (List<RCIMIWMessage>? matchedMessages,
List<RCIMIWMessage>? notMatchedMessages) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getBatchRemoteUltraGroupMessages(messages, callback:callback);
```

回调方法

- **onBatchRemoteUltraGroupMessagesLoaded**

```
Function(int? code, List<RCIMIWMessage>? matchedMessages, List<RCIMIWMessage>? notMatchedMessages)?
onBatchRemoteUltraGroupMessagesLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
matchedMessages	List<RCIMIWMessage>	从服务获取的消息列表
notMatchedMessages	List<RCIMIWMessage>	非法参数或者从服务没有拿到对应消息

代码示例

```
engine?.onBatchRemoteUltraGroupMessagesLoaded = (int? code, List<RCIMIWMessage>? matchedMessages,
List<RCIMIWMessage>? notMatchedMessages) {
//...
};
```

删除消息

更新时间:2024-08-30

超级群会话消息存储在服务端（免费存储 7 天）和用户设备本地数据库。App 用户通过客户端 SDK 删除自己的历史消息，支持仅从本地数据库删除消息、或仅从融云服务端删除消息。

> 提示
>
> - 客户端的删除消息的操作均指从当前登录用户的历史消息记录中删除消息，不影响会话中其他用户的历史消息记录。
> - 如果 App 的管理员或者某普通用户希望在该 App 中彻底删除一条消息，例如在所有超级群成员的聊天记录中删除一条消息，应使用客户端或服务端的撤回消息功能。消息成功撤回后，原始消息内容会在所有用户的本地与服务端历史消息记录中删除。

删除所有频道指定时间之前的消息

删除本地单个超级群所有频道指定时间之前的消息。

方法

```
Future<int> clearUltraGroupMessagesForAllChannel(String targetId, int timestamp,
{IRCIMIWClearUltraGroupMessagesForAllChannelCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
timestamp	int	时间戳
callback	IRCIMIWClearUltraGroupMessagesForAllChannelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearUltraGroupMessagesForAllChannelCallback? callback =
IRCIMIWClearUltraGroupMessagesForAllChannelCallback(onUltraGroupMessagesClearedForAllChannel: (int?
code) {
//...
});

int? ret = await engine?.clearUltraGroupMessagesForAllChannel(targetId, timestamp, callback:callback);
```

回调方法

- **onUltraGroupMessagesClearedForAllChannel**

```
Function(int? code, String? targetId, int? timestamp)? onUltraGroupMessagesClearedForAllChannel;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
timestamp	int	时间戳

代码示例

```
engine?.onUltraGroupMessagesClearedForAllChannel = (int? code, String? targetId, int? timestamp) {
//...
};
```

删除特定频道指定时间之前的消息

删除本地单个超级群特定频道指定时间之前的消息。

方法

```
Future<int> clearUltraGroupMessages(String targetId, String? channelId, int timestamp,
RCIMIWMessageOperationPolicy policy, {IRCIMIWClearUltraGroupMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
timestamp	int	时间戳
policy	RCIMIWMessageOperationPolicy	清除策略

参数名	参数类型	描述
callback	IRCIMIWClearUltraGroupMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearUltraGroupMessagesCallback? callback =
IRCIMIWClearUltraGroupMessagesCallback(onUltraGroupMessagesCleared: (int? code) {
//...
});

int? ret = await engine?.clearUltraGroupMessages(targetId, channelId, timestamp, policy,
callback:callback);
```

回调方法

- **onUltraGroupMessagesCleared**

```
Function(int? code, String? targetId, String? channelId, int? timestamp, RCIMIWMessageOperationPolicy?
policy)? onUltraGroupMessagesCleared;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	时间戳
policy	RCIMIWMessageOperationPolicy	清除策略

代码示例

```
engine?.onUltraGroupMessagesCleared = (int? code, String? targetId, String? channelId, int? timestamp,
RCIMIWMessageOperationPolicy? policy) {
//...
};
```

从服务端删除

删除服务端单个超级群特定频道指定时间之前的消息。

方法

```
Future<int> clearUltraGroupMessages(String targetId, String? channelId, int timestamp,
RCIMIWMessageOperationPolicy policy, {IRCIMIWClearUltraGroupMessagesCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
timestamp	int	时间戳
policy	RCIMIWMessageOperationPolicy	清除策略
callback	IRCIMIWClearUltraGroupMessagesCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWClearUltraGroupMessagesCallback? callback =
IRCIMIWClearUltraGroupMessagesCallback(onUltraGroupMessagesCleared: (int? code) {
//...
});

int? ret = await engine?.clearUltraGroupMessages(targetId, channelId, timestamp, policy,
callback:callback);
```

回调方法

- **onUltraGroupMessagesCleared**

```
Function(int? code, String? targetId, String? channelId, int? timestamp, RCIMIWMessageOperationPolicy?
policy)? onUltraGroupMessagesCleared;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	时间戳

参数名	参数类型	描述
policy	RCIMIWMessageOperationPolicy	清除策略

代码示例

```
engine?.onUltraGroupMessagesCleared = (int? code, String? targetId, String? channelId, int? timestamp,
RCIMIWMessageOperationPolicy? policy) {
//...
};
```

修改消息

更新时间:2024-08-30

超级群消息发送后支持主动修改消息内容，仅可修改自己发送的消息。

不支持修改消息类型，即改前改后必须是同一类型。

修改消息

方法

```
Future<int> modifyUltraGroupMessage(String messageUId, RCIMIWMessage message,
{IRCIMIWModifyUltraGroupMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值
message	RCIMIWMessage	要修改的 message
callback	IRCIMIWModifyUltraGroupMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWModifyUltraGroupMessageCallback? callback =
IRCIMIWModifyUltraGroupMessageCallback(onUltraGroupMessageModified: (int? code) {
//...
});

int? ret = await engine?.modifyUltraGroupMessage(messageUId, message, callback:callback);
```

回调方法

- **onUltraGroupMessageModified**

```
Function(int? code, String? messageUId)? onUltraGroupMessageModified;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messageUId	String	消息的 messageUid

代码示例

```
engine?.onUltraGroupMessageModified = (int? code, String? messageUId) {
//...
};
```

监听远端消息修改

当接收到的消息发生变化时会回调此方法。

方法

```
Function(List<RCIMIWMessage>? messages)? onRemoteUltraGroupMessageModified;
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage>	被更新的消息集合

代码示例

```
engine?.onRemoteUltraGroupMessageModified = (List<RCIMIWMessage>? messages) {
//...
};
```

撤回消息

更新时间:2024-08-30

超级群业务中，消息发送方可撤回已发送成功的消息。撤回成功后，服务端即删除原始消息。

撤回指定消息，只有已发送成功的消息可被撤回。

撤回指定消息

方法

```
Future<int> recallUltraGroupMessage(RCIMIWMessage message, bool deleteRemote,
{IRCIMIWRecallUltraGroupMessageCallback? callback});
```

参数说明

参数名	参数类型	描述
message	RCIMIWMessage	需要撤回的消息
deleteRemote	bool	是否删除远端消息
callback	IRCIMIWRecallUltraGroupMessageCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRecallUltraGroupMessageCallback? callback =
IRCIMIWRecallUltraGroupMessageCallback(onUltraGroupMessageRecalled: (int? code) {
//...
});

int? ret = await engine?.recallUltraGroupMessage(message, deleteRemote, callback:callback);
```

回调方法

- **onUltraGroupMessageRecalled**

```
Function(int? code, RCIMIWMessage? message, bool? deleteRemote)? onUltraGroupMessageRecalled;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
message	RCIMIWMessage	撤回的消息
deleteRemote	bool	调用接口时传入的是否删除远端消息

代码示例

```
engine?.onUltraGroupMessageRecalled = (int? code, RCIMIWMessage? message, bool? deleteRemote) {
//...
};
```

监听远端消息撤回

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。

方法

```
Function(List<RCIMIWMessage>? messages)? onRemoteUltraGroupMessageRecalled;
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage>	撤回的消息集合

代码示例

```
engine?.onRemoteUltraGroupMessageRecalled = (List<RCIMIWMessage>? messages) {
//...
};
```

扩展消息

更新时间:2024-08-30

已发送的超级群消息可增加、修改、删除扩展信息。
原始消息增加状态标识的需求，都可使用消息扩展。

- 消息评论需求，可通过设置原始消息扩展信息的方式添加评论信息。
- 礼物领取、订单状态变化需求，通过此功能改变消息显示状态。 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。

更新消息扩展

方法

```
Future<int> updateUltraGroupMessageExpansion(String messageUId, Map expansion,
{IRCIMIWUpdateUltraGroupMessageExpansionCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值
expansion	Ma<String, String>	更新的消息扩展信息键值对，类型是 HashMap；Key 支持大小写英文字母、数字、部分特殊符号 + = - _ 的组合方式，不支持汉字。Value 可以输入空格。
callback	IRCIMIWUpdateUltraGroupMessageExpansionCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWUpdateUltraGroupMessageExpansionCallback? callback =
IRCIMIWUpdateUltraGroupMessageExpansionCallback(onUltraGroupMessageExpansionUpdated: (int? code) {
//...
});

int? ret = await engine?.updateUltraGroupMessageExpansion(messageUId, expansion, callback:callback);
```

回调方法

- **onUltraGroupMessageExpansionUpdated**

```
Function(int? code, Map? expansion, String? messageUId)? onUltraGroupMessageExpansionUpdated;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
expansion	Ma<String, String>	更新的消息扩展信息键值对
messageUId	String	消息的 messageUid

代码示例

```
engine?.onUltraGroupMessageExpansionUpdated = (int? code, Map? expansion, String? messageUId) {
//...
};
```

删除消息扩展

方法

```
Future<int> removeUltraGroupMessageExpansionForKeys(String messageUId, List<String> keys,
{IRCIMIWRemoveUltraGroupMessageExpansionForKeysCallback? callback});
```

参数说明

参数名	参数类型	描述
messageUId	String	消息的 messageUid，可在消息对象中获取，且只有发送成功的消息才会有值
keys	List<String>	消息扩展信息中待删除的 key 的列表，类型是 ArrayList
callback	IRCIMIWRemoveUltraGroupMessageExpansionForKeysCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveUltraGroupMessageExpansionForKeysCallback? callback =
IRCIMIWRemoveUltraGroupMessageExpansionForKeysCallback(onUltraGroupMessageExpansionForKeysRemoved: (int?
code) {
//...
});

int? ret = await engine?.removeUltraGroupMessageExpansionForKeys(messageUId, keys, callback:callback);
```

回调方法

- **onUltraGroupMessageExpansionForKeysRemoved**

```
Function(int? code, String? messageUId, List<String>? keys)? onUltraGroupMessageExpansionForKeysRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
messageUId	String	消息的 messageUid
keys	List<String>	消息扩展信息中待删除的 key 的列表

代码示例

```
engine?.onUltraGroupMessageExpansionForKeysRemoved = (int? code, String? messageUId, List<String>? keys)
{
//...
};
```

监听远端消息扩展更新

当远端消息的扩展信息被更新时，会回调此方法。

方法

```
Function(List<RCIMIWMessage>? messages)? onRemoteUltraGroupMessageExpansionUpdated;
```

参数说明

参数名	参数类型	描述
messages	List<RCIMIWMessage>	被更新的消息集合

代码示例

```
engine?.onRemoteUltraGroupMessageExpansionUpdated = (List<RCIMIWMessage>? messages) {
//...
};
```

获取未读数

更新时间:2024-08-30

返回的未读数最大值为 999。如果实际未读数超过 999，接口仍返回 999。

获取所有超级群会话的未读消息数

获取当前用户加入的所有超级群会话的未读消息数的总和。

方法

```
Future<int> getUltraGroupAllUnreadCount({IRCIMIWGetUltraGroupAllUnreadCountCallback? callback});
```

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupAllUnreadCountCallback? callback =
IRCIMIWGetUltraGroupAllUnreadCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupAllUnreadCount(callback:callback);
```

回调方法

- **onUltraGroupAllUnreadCountLoaded**

```
Function(int? code, int? count)? onUltraGroupAllUnreadCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
count	int	未读数量

代码示例

```
engine?.onUltraGroupAllUnreadCountLoaded = (int? code, int? count) {
//...
};
```

所有超级群会话中的未读 @ 消息数

获取当前用户加入的所有超级群会话中的未读 @ 消息数的总和。

方法

```
Future<int> getUltraGroupAllUnreadMentionedCount({IRCIMIWGetUltraGroupAllUnreadMentionedCountCallback?
callback});
```

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupAllUnreadMentionedCountCallback? callback =
IRCIMIWGetUltraGroupAllUnreadMentionedCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupAllUnreadMentionedCount(callback:callback);
```

回调方法

- **onUltraGroupAllUnreadMentionedCountLoaded**

```
Function(int? code, int? count)? onUltraGroupAllUnreadMentionedCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
count	int	未读数量

代码示例

```
engine?.onUltraGroupAllUnreadMentionedCountLoaded = (int? code, int? count) {
//...
};
```

获取指定会话的未读消息数

获取当前用户加入的所有超级群会话的未读消息数的总和。

方法

```
Future<int> getUltraGroupUnreadCount(String targetId, {IRCIMIWGetUltraGroupUnreadCountCallback?
callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
callback	IRCIMIWGetUltraGroupUnreadCountCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupUnreadCountCallback? callback = IRCIMIWGetUltraGroupUnreadCountCallback(onSuccess:
(int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupUnreadCount(targetId, callback:callback);
```

回调方法

- **onUltraGroupUnreadCountLoaded**

```
Function(int? code, String? targetId, int? count)? onUltraGroupUnreadCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	None
targetId	String	None
count	int	None

代码示例

```
engine?.onUltraGroupUnreadCountLoaded = (int? code, String? targetId, int? count) {
//...
};
```

取超级群会话中被 @ 的消息数

获取超级群会话中被 @ 的消息数

方法

```
Future<int> getUltraGroupUnreadMentionedCount(String targetId,
{IRCIMIWGetUltraGroupUnreadMentionedCountCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
callback	IRCIMIWGetUltraGroupUnreadMentionedCountCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupUnreadMentionedCountCallback? callback =
IRCIMIWGetUltraGroupUnreadMentionedCountCallback(onSuccess: (int? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupUnreadMentionedCount(targetId, callback:callback);
```

回调方法

- **onUltraGroupUnreadMentionedCountLoaded**

```
Function(int? code, String? targetId, int? count)? onUltraGroupUnreadMentionedCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
count	int	未读数量

代码示例

```
engine?.onUltraGroupUnreadMentionedCountLoaded = (int? code, String? targetId, int? count) {
//...
};
```

清除消息未读状态

更新时间:2024-08-30

超级群业务可在多个客户端之间同步消息阅读状态。

同步消息已读状态

调用同步已读状态接口会同时清除本地与服务端记录的消息的未读状态，同时服务端会将最新状态同步给同一用户账号的其他客户端。

- 如果指定了频道 ID（channelId），则标记该频道所有消息为全部已读，并同步其他客户端。
- 如果频道 ID 为空，则标记该超级群会话下所有不属于任何频道的消息为全部已读，并同步其他客户端。

> 提示
>
> 超级群暂不支持按时间戳同步已读状态。调用 syncUltraGroupReadStatus 会按指定参数的要求标记全部消息为已读。时间戳参数（timestamp）未使用，可传入任意数字。

方法

```
Future<int> syncUltraGroupReadStatus(String targetId, String? channelId, int timestamp,
{IRCIMIWSyncUltraGroupReadStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
timestamp	int	已读时间
callback	IRCIMIWSyncUltraGroupReadStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSyncUltraGroupReadStatusCallback? callback =
IRCIMIWSyncUltraGroupReadStatusCallback(onUltraGroupReadStatusSynced: (int? code) {
//...
});

int? ret = await engine?.syncUltraGroupReadStatus(targetId, channelId, timestamp, callback:callback);
```

回调方法

- **onUltraGroupReadStatusSynced**

```
Function(int? code, String? targetId, String? channelId, int? timestamp)? onUltraGroupReadStatusSynced;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
timestamp	int	已读时间

代码示例

```
engine?.onUltraGroupReadStatusSynced = (int? code, String? targetId, String? channelId, int? timestamp)
{
//...
};
```

监听其他端同步的消息未读状态

当接收到其他端同步的未读状态时，会回调此方法

方法

```
Function(RCIMIWConversationType? type, String? targetId, int? timestamp)?
onConversationReadStatusSyncMessageReceived;
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
timestamp	int	时间戳

代码示例

```
engine?.onConversationReadStatusSyncMessageReceived = (RCIMIWConversationType? type, String? targetId,
int? timestamp) {
//...
};
```

输入状态

更新时间:2024-08-30

应用程序可以向超级群中发送当前用户输入状态。超级群内收到通知的用户可以在 UI 展示 “ xxx 正在输入”。

> 提示
>
> 为保证最佳体验，建议在仅在人数小于 10,000 的超级群中使用该功能。

发送输入状态

应用程序可以在当前用户输入文本时调用 sendUltraGroupTypingStatus，发送当前用户输入状态。

方法

```
Future<int> sendUltraGroupTypingStatus(String targetId, String? channelId, RCIMIWUltraGroupTypingStatus typingStatus, {IRCIMIWSendUltraGroupTypingStatusCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
typingStatus	RCIMIWUltraGroupTypingStatus	输入状态
callback	IRCIMIWSendUltraGroupTypingStatusCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWSendUltraGroupTypingStatusCallback? callback =
IRCIMIWSendUltraGroupTypingStatusCallback(onUltraGroupTypingStatusSent: (int? code) {
//...
});

int? ret = await engine?.sendUltraGroupTypingStatus(targetId, channelId, typingStatus,
callback:callback);
```

回调方法

- **onUltraGroupTypingStatusSent**

```
Function(int? code, String? targetId, String? channelId, RCIMIWUltraGroupTypingStatus? typingStatus)?
onUltraGroupTypingStatusSent;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
typingStatus	RCIMIWUltraGroupTypingStatus	输入状态枚举

代码示例

```
engine?.onUltraGroupTypingStatusSent = (int? code, String? targetId, String? channelId,
RCIMIWUltraGroupTypingStatus? typingStatus) {
//...
};
```

监听用户输入状态变化

当超级群用户的输入状态发生变化时，会回调下面方法。为了减少服务端和客户端的压力，服务端会将一段时间内（例如 5 秒）用户输入事件汇总之后统一批量下发，所以回调以数组形式提供。应用程序收到通知时可以在 UI 展示 “ xxx 正在输入”。

方法

```
Function(List<RCIMIWUltraGroupTypingStatusInfo>? info)? onUltraGroupTypingStatusChanged;
```

参数说明

参数名	参数类型	描述
info	List<RCIMIWUltraGroupTypingStatusInfo>	正在输入的RCUltraGroupTypingStatusInfo列表（nil 表示当前没有用户正在输入）

代码示例

```
engine?.onUltraGroupTypingStatusChanged = (List<RCIMIWUltraGroupTypingStatusInfo>? info) {
//...
};
```

超级群免打扰功能概述

更新时间:2024-08-30

「免打扰功能」用于控制用户在客户端设备离线时，是否可针对离线消息接收推送通知。

- **客户端为离线状态**：会话中有新离线消息时，用户默认通过推送通道收到消息且默认弹出通知。设置免打扰后，融云服务端不会为相关消息触发推送。
- **客户端在后台运行**：会话中有新消息时，用户直接收到消息。如果使用 IMLib，您需要自行判断 App 是否在后台运行，并根据业务需求自行实现本地通知弹窗。

前提条件

请在使用「免打扰功能」前检查是否已集成第三方推送，是否已为用户启用了推送服务。

免打扰设置维度

客户端 SDK 支持超级群业务进行以下多个维度的免打扰设置：

- App 的免打扰设置
- 按超级群或频道设置默认免打扰级别
- 按指定会话类型设置免打扰级别
- 按会话设置免打扰级别
- 按超级群频道设置免打扰级别
- 全局免打扰

> 提示
>
> **超级群离线消息推送通知还会收到控制台超级群默认推送频率设置影响。详见开通超级群服务 。**

App 的免打扰设置

以 App Key 为单位，设置整个应用所有用户的默认免打扰级别。默认未设置，等同于全部消息都接收通知。该级别的配置暂未在控制台开放，如有需要，请提交工单。

- **全部消息均通知**：当前 App 下的用户可针对任何消息接收推送通知。
- **未设置**：默认全部消息都通知。
- **仅 @ 消息通知**：当前 App 下的用户仅针对提及（@）当前用户和提及所在群组全体成员的消息接收推送通知。
- **仅 @ 指定用户通知**：当前 App 下，用户仅针对提及（@）当前用户的消息接收推送通知。例如：仅张三会接收且仅接收“@张三 Hello” 的消息的通知。
- **仅 @群全员通知**：当前 App 下，用户仅针对提及（@）群组全体成员的消息接收推送通知。
- **都不接收通知**：当前 App 下，用户不针对任何消息接收推送通知，即任何离线消息都不会触发推送通知。

- **除 @ 消息外群聊消息不发推送**：当前 App 下，用户针对单聊消息、提及（@）指定用户的消息、和提及（@）群组全体成员的消息接收推送通知。

融云服务端判断是否需要推送时，如果存在以下任何一种用户级别的免打扰配置，以用户级别配置为准：

- 按指定会话类型设置免打扰级别
- 按会话设置免打扰级别
- 全局免打扰

如果不存在用户级别配置，则以消息所属超级群/频道的默认免打扰级别为准。App 级别的免打扰配置的优先级最低。

按超级群/频道设置默认免打扰级别

客户端 SDK 支持为指定超级群下的所有成员配置触发推送通知的消息类别，或完全关闭通知。客户端 SDK 提供 RCIMIWPushNotificationLevel，支持以下六个级别：

枚举值	数值	说明
allMessage	0	与融云服务端断开连接后，当前超级群的所有用户可针对指定超级群（或频道）中的所有消息接收通知。
none	1	未设置。未设置时均为此初始状态。在此状态下，如果超级群与群频道均为未设置，则认为超级群与频道的默认免打扰级别为全部消息都通知。
mention	2	与融云服务端断开连接后，当前超级群的所有用户仅针对指定超级群（或频道）中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	3	与融云服务端断开连接后，当前用户仅针对指定超级群（或频道）中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	4	与融云服务端断开连接后，当前用户仅针对指定超级群（或频道）中提及（@）全部群成员的消息接收通知。
blocked	5	当前用户针对指定超级群（或频道）中的任何消息都不接收推送通知。

具体设置方法详见设置群/频道默认免打扰。

为指定的超级群设置的默认免打扰逻辑，自动适用于群下的所有频道。如果针对频道另行设置了默认免打扰逻辑，则以该频道的默认设置为准。融云服务端判断是否需要为用户发送推送通知时，如果同时存在以下任何一种用户级别的免打扰配置，以下列配置为准：

- 按会话类型设置免打扰级别
- 按会话设置免打扰级别
- 按超级群频道设置免打扰级别
- 全局免打扰

按会话类型设置免打扰级别

客户端 SDK 免打扰级别配置，定义在 RCIMIWPushNotificationLevel 枚举类中，允许用户为会话类型（单聊、群聊、超级群、系统会话）配置触发推送通知的消息类别，或完全关闭通知。提供以下六个级别：

枚举值	数值	说明
allMessage	0	与融云服务端断开连接后，当前用户可针对指定类型会话中的所有消息接收通知。

枚举值	数值	说明
none	1	未设置。未设置时均为此初始状态。 注意：在此状态下，如果超级群与群频道均为未设置，则认为超级群与频道的默认免打扰级别为全部消息都通知。
mention	2	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	3	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	4	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）全部群成员的消息接收通知。
blocked	5	当前用户针对指定类型的会话中的任何消息都不接收推送通知。

具体设置方法详见按会话类型设置免打扰。

融云服务端判断是否需要为用户发送推送通知时，如果同时存在以下任何一种用户级别的免打扰配置，以下列配置为准：

- 按会话设置免打扰级别
- 按超级群频道设置免打扰级别
- 全局免打扰

按会话设置免打扰级别

> 提示
>
> 该功能属于用户级别设置。

客户端 SDK 提供 RCIMIWPushNotificationLevel，允许用户为会话配置触发通知的消息类别，或完全关闭通知。提供以下六个级别：

枚举值	数值	说明
allMessage	0	与融云服务端断开连接后，当前用户可针对指定类型会话中的所有消息接收通知。
none	1	未设置。未设置时均为此初始状态。 注意：在此状态下，如果超级群与群频道均为未设置，则认为超级群与频道的默认免打扰级别为全部消息都通知。
mention	2	与融云服务端断开连接后，当前用户仅针对指定会话中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	3	与融云服务端断开连接后，当前用户仅针对接收指定会话中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	4	与融云服务端断开连接后，当前用户仅针对指定会话中提及（@）全部群成员的消息接收通知。
blocked	5	当前用户针对指定会话中的任何消息都不接收推送通知。

具体设置方法详见按会话设置免打扰。

从 2022.09.01 开始，为指定的超级群设置的免打扰级别，自动适用于群下的所有频道。融云服务端判断是否需要为用户发送推送通知时，如果同时存在以下任何一种用户级别的免打扰配置，以下列配置为准：

- 按超级群频道设置免打扰级别
- 全局免打扰

按超级群频道设置免打扰级别

> 提示
>
> 该功能属于用户级别设置。

客户端 SDK 提供 RCIMIWPushNotificationLevel，允许用户为指定超级群频道配置触发通知的消息类别，或完全关闭通知。提供以下六个级别：

枚举值	数值	说明
allMessage	0	与融云服务端断开连接后，当前用户可针对指定超级群频道中的所有消息接收通知。
none	1	未设置。未设置时均为此初始状态。 注意：在此状态下，如果超级群与群频道均为未设置，则认为超级群与频道的默认免打扰级别为全部消息都通知。
mention	2	与融云服务端断开连接后，当前用户可针对指定超级群频道中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	3	与融云服务端断开连接后，当前用户可针对指定超级群频道中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	4	与融云服务端断开连接后，当前用户可针对指定超级群频道中提及（@）全部群成员的消息接收通知。
blocked	5	当前用户针对指定超级群频道中的任何消息都不接收推送通知。

具体设置方法详见按频道设置免打扰。

融云服务端判断是否需要为用户发送推送通知时，如果该用户已配置全局免打扰，则已全局免打扰的配置细节为准。

全局免打扰

客户端 SDK 提供 RCIMIWPushNotificationQuietHoursLevel，允许用户配置何时接收通知以及触发通知的消息类别。提供了以下三个级别：

枚举值	数值	说明
none	0	未设置。如未设置，SDK 会依次查询消息所属群的用户级别免打扰设置及其他非用户级别设置，再判断是否需要推送通知。
mentionMessage	1	与融云服务端断开连接后，当前用户仅在指定时段内针对指定会话中提及（@）当前用户和全体群成员的消息接收通知。
blocked	2	当前用户在指定时段内针对任何消息都不接收推送通知。

具体设置方法详见全局免打扰。

免打扰设置的优先级

针对超级群会话，融云服务端会遵照以下顺序搜索免打扰配置。优先级从左至右依次降低，以优先级最高的配置为准判断是否需要触发推送：

全局免打扰设置（用户级）> 指定超级群频道的免打扰设置（用户级）> 指定会话的免打扰设置（用户级）> 指定会话类型的免打扰设置（用户级）> 指定超级群频道的默认免打扰设置（超级群全员）> 指定超级群的免打扰设置（超级群全员）> App 级的免打扰设置

API 接口列表

下表描述了适用于超级群会话的免打扰配置 API 接口。

免打扰配置维度	客户端 API	服务端 API
设置指定时段内，应用全局的免打扰级别（用户级）	详见全局免打扰。	详见 设置用户免打扰时段
设置指定超级群频道的免打扰级别（用户级）	详见按频道设置免打扰。	详见设置会话免打扰。
设置指定会话的免打扰级别（用户级）	详见按会话设置免打扰。	详见设置会话免打扰。
设置指定类型会话的免打扰级别（用户级）	详见按会话类型设置免打扰。	详见设置会话类型免打扰。
设置指定超级群/频道的默认免打扰设置（超级群全体成员）	详见「超级群管理」下的设置群/频道默认免打扰。	详见「超级群管理」下的设置群/频道默认免打扰
设置 App 级免打扰级别	客户端 SDK 不提供 API。	服务端不提供该 API。

按频道设置免打扰

更新时间:2024-08-30

本文描述如何为超级群业务设置免打扰级别。

> 提示
>
> 即时通讯客户端 SDK 支持多维度、多级别的免打扰设置。
>
> - App 开发者可实现从 App Key、指定细分业务（仅超级群）、用户级别多个维度的免打扰功能配置。在融云服务端决定是否触发推送通知时，不同维度的优先级如下：用户级别设置 > 指定超级群频道的默认配置（仅超级群支持） > 指定超级群会话的默认配置（仅超级群支持） > App Key 级设置。
> - 用户级别设置下包含多个细分维度。在融云服务端决定是否触发推送通知时，如存在用户级别配置，不同细分维度的优先级如下：全局免打扰 > 按频道设置的免打扰 > 按会话设置的免打扰 > 按会话类型设置的免打扰。详见免打扰功能概述。

在免打扰设置生效时，客户端收到新消息时行为如下：

- 客户端在后台运行：会话中有新消息时，将不会进行通知提醒，但可以收到消息内容。
- 客户端为离线状态：会话中有新消息时，不会收到远程通知提醒，再次上线时可收取消息内容。

支持的免打扰级别

免打扰级别提供了针对不同 @ 消息的免打扰控制。指定频道的免打扰配置支持以下级别：

枚举值	说明
allMessage	所有消息均可进行通知。
none	未设置。未设置时均为此初始状态。 注意：在此状态下，如果超级群与群频道均为未设置，则认为超级群与频道的默认免打扰级别为全部消息都通知。
mention	仅针对 @ 消息进行通知，包括 @指定用户 和 @所有人
mentionUsers	仅针对 @ 指定用户消息进行通知，且仅针对被 @ 的指定的用户进行通知。 如：@张三，则张三可以收到推送； @所有人不会触发推送通知。
mentionAll	仅针对 @群全员进行通知，即只接收 @所有人的推送信息。
blocked	不接收通知，即使为 @ 消息也不推送通知。

管理超级群的免打扰设置

SDK 不提供一次性设置所有超级群频道免打扰的方法，需要开发者根据所有频道来遍历调用 [设置指定频道免打扰级别] 的方法，达到整个超级群免打扰的效果。

管理频道的免打扰设置

超级群（UltraGroup）支持在超级群的会话下创建独立的频道（channel），对消息数据（会话、消息、未读数）和群组成员分频道进行聚合。SDK 支持在超级群业务中的用户（userId）设置指定群频道（channelId）的免打扰级别。

设置指定频道免打扰级别

为当前用户设置超级群频道中的消息的免打扰级别。

方法

```
Future<int> changeConversationNotificationLevel(RCIMIWConversationType type, String targetId, String?
channelId, RCIMIWPushNotificationLevel level, {IRCIMIWChangeConversationNotificationLevelCallback?
callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
level	RCIMIWPushNotificationLevel	消息通知级别
callback	IRCIMIWChangeConversationNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeConversationNotificationLevelCallback? callback =
IRCIMIWChangeConversationNotificationLevelCallback(onConversationNotificationLevelChanged: (int? code) {
//...
});

int? ret = await engine?.changeConversationNotificationLevel(type, targetId, channelId, level,
callback:callback);
```

回调方法

- **onUltraGroupUnreadMentionedCountLoaded**

```
Function(int? code, String? targetId, int? count)? onUltraGroupUnreadMentionedCountLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
count	int	未读数量

代码示例

```
engine?.onUltraGroupUnreadMentionedCountLoaded = (int? code, String? targetId, int? count) {
//...
};
```

获取指定频道免打扰级别

获取为当前用户设置的超级群频道免打扰级别。

方法

```
Future<int> getConversationNotificationLevel(RCIMIWConversationType type, String targetId, String?
channelId, {IRCIMIWGetConversationNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
callback	IRCIMIWGetConversationNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetConversationNotificationLevelCallback? callback =
IRCIMIWGetConversationNotificationLevelCallback(onSuccess: (RCIMIWPushNotificationLevel? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getConversationNotificationLevel(type, targetId, channelId, callback:callback);
```

回调方法

- **onConversationNotificationLevelChanged**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
RCIMIWPushNotificationLevel? level)? onConversationNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onConversationNotificationLevelChanged = (int? code, RCIMIWConversationType? type, String?
targetId, String? channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

设置群/频道默认免打扰

更新时间:2024-08-30

超级群业务支持为指定的群，或群频道设置默认免打扰逻辑。默认免打扰逻辑对所有群成员生效，一般由超级群的管理员进行设置。

如果您希望从 App 服务端控制指定超级群，或指定群频道默认免打扰逻辑，可参考服务端 API 文档设置超级群/频道默认免打扰。

注意事项

- 在融云服务端判断是否需要推送超级群消息时，指定的超级群，或群频道的默认免打扰配置优先级均低于用户级别配置。如果存在任何用户级别的免打扰配置，则优先以用户级别免打扰配置为准进行判断。

> 提示
>
> 即时通讯业务免打扰功能的 用户级别设置 支持控制指定的单聊会话、群聊会话、超级群会话、超级群频道的免打扰级别，并可设置全局免打扰的时间段与级别。用户级别设置优先级如下：全局免打扰 > 按频道设置的免打扰 > 按会话设置的免打扰 > 按会话类型设置的免打扰。详见免打扰功能概述。

- 为指定的超级群设置的默认免打扰逻辑，自动适用于群下的所有频道。如果针对频道另行设置了默认免打扰逻辑，则以该频道的默认设置为准。

支持的免打扰级别

指定超级群或群频道的默认免打扰级别可设置为以下任一级别：

枚举	说明
allMessage	与融云服务端断开连接后，当前用户可针对指定类型会话中的所有消息接收通知。
none	未设置。未设置时均为此初始状态。
mention	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户和全体群成员的消息接收通知。
mentionUsers	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）当前用户的消息接收通知。例如：张三只会接收 “@张三 Hello” 的消息的通知。
mentionAll	与融云服务端断开连接后，当前用户仅针对指定类型的会话中提及（@）全部群成员的消息接收通知。
blocked	当前用户针对指定类型的会话中的任何消息都不接收推送通知。

设置指定超级群的默认免打扰级别

方法

```
Future<int> changeUltraGroupDefaultNotificationLevel(String targetId, RCIMIWPushNotificationLevel level,
{IRCIMIWChangeUltraGroupDefaultNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
level	RCIMIWPushNotificationLevel	消息通知级别
callback	IRCIMIWChangeUltraGroupDefaultNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeUltraGroupDefaultNotificationLevelCallback? callback =
IRCIMIWChangeUltraGroupDefaultNotificationLevelCallback(onUltraGroupDefaultNotificationLevelChanged:
(int? code) {
//...
});

int? ret = await engine?.changeUltraGroupDefaultNotificationLevel(targetId, level, callback:callback);
```

回调方法

- **onUltraGroupDefaultNotificationLevelChanged**

```
Function(int? code, String? targetId, RCIMIWPushNotificationLevel? level)?
onUltraGroupDefaultNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onUltraGroupDefaultNotificationLevelChanged = (int? code, String? targetId,
RCIMIWPushNotificationLevel? level) {
//...
};
```

查询指定超级群的默认免打扰级别

方法

```
Future<int> getUltraGroupDefaultNotificationLevel(String targetId,
{IRCIMIWGetUltraGroupDefaultNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
callback	IRCIMIWGetUltraGroupDefaultNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupDefaultNotificationLevelCallback? callback =
IRCIMIWGetUltraGroupDefaultNotificationLevelCallback(onSuccess: (RCIMIWPushNotificationLevel? t) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupDefaultNotificationLevel(targetId, callback:callback);
```

回调方法

- **onUltraGroupDefaultNotificationLevelLoaded**

```
Function(int? code, String? targetId, RCIMIWPushNotificationLevel? level)?
onUltraGroupDefaultNotificationLevelLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onUltraGroupDefaultNotificationLevelLoaded = (int? code, String? targetId,
RCIMIWPushNotificationLevel? level) {
//...
};
```

设置指定群频道的默认免打扰级别

方法

```
Future<int> changeUltraGroupChannelDefaultNotificationLevel(String targetId, String? channelId,
RCIMIWPushNotificationLevel level, {IRCIMIWChangeUltraGroupChannelDefaultNotificationLevelCallback?
callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
level	RCIMIWPushNotificationLevel	消息通知级别
callback	IRCIMIWChangeUltraGroupChannelDefaultNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeUltraGroupChannelDefaultNotificationLevelCallback? callback =
IRCIMIWChangeUltraGroupChannelDefaultNotificationLevelCallback(onUltraGroupChannelDefaultNotificationLev
Changed: (int? code) {
//...
});

int? ret = await engine?.changeUltraGroupChannelDefaultNotificationLevel(targetId, channelId, level,
callback:callback);
```

回调方法

- **onUltraGroupChannelDefaultNotificationLevelChanged**

```
Function(int? code, String? targetId, String? channelId, RCIMIWPushNotificationLevel? level)?
onUltraGroupChannelDefaultNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onUltraGroupChannelDefaultNotificationLevelChanged = (int? code, String? targetId, String?
channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

查询指定群频道的默认免打扰级别

方法

```
Future<int> getUltraGroupChannelDefaultNotificationLevel(String targetId, String? channelId,
{IRCIMIWGetUltraGroupChannelDefaultNotificationLevelCallback? callback});
```

参数说明

参数名	参数类型	描述
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
callback	IRCIMIWGetUltraGroupChannelDefaultNotificationLevelCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetUltraGroupChannelDefaultNotificationLevelCallback? callback =
IRCIMIWGetUltraGroupChannelDefaultNotificationLevelCallback(onSuccess: (RCIMIWPushNotificationLevel? t)
{
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getUltraGroupChannelDefaultNotificationLevel(targetId, channelId,
callback:callback);
```

回调方法

- **onUltraGroupChannelDefaultNotificationLevelLoaded**

```
Function(int? code, String? targetId, String? channelId, RCIMIWPushNotificationLevel? level)?
onUltraGroupChannelDefaultNotificationLevelLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用。
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onUltraGroupChannelDefaultNotificationLevelLoaded = (int? code, String? targetId, String?
channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

全局免打扰

更新时间:2024-08-30

SDK 支持为当前用户设置全局免打扰时段与免打扰级别。

- 该功能可设置一个从任意时间点（`HH:MM:SS`）开始的免打扰时间窗口。在再次设置或删除用户免打扰时间段之前，当次设置的免打扰时间窗口会每日重复生效。例如，App 用户希望设置永久全天免打扰，可设置 `startTime` 为 `00:00:00`，`period` 为 `1439`。
- 单个用户仅支持设置一个时间段，重复设置会覆盖该用户之前设置的时间窗口。

> 提示
>
> 在经 SDK 设置的全局免打扰时段内：
>
> - 如果客户端处于离线状态，融云服务端将不会进行推送通知。
> - 「全局免打扰时段」为用户级别的免打扰设置，且具有最高优先级。在用户设置了「全局免打扰时段」时，均以此设置的免打扰级别为准。

（推荐）在 App 自行实现本地通知处理时，如果检测到客户端 App 已转至后台运行，可通过 SDK 提供的全局免打扰接口决定是否弹出本地通知，以实现全局免打扰的效果。

支持的免打扰级别

为当前用户设置免打扰时间段时，可使用以下免打扰级别：

枚举值	数值	说明
none	0	未设置。如未设置，SDK 会依次查询消息所属群的用户级别免打扰设置及其他非用户级别设置，再判断是否需要推送通知。
mentionMessage	1	仅针对 @ 消息进行通知，包括 @指定用户 和 @所有人的消息。
blocked	2	不接收通知，即使为 @ 消息也不推送通知。

设置免打扰时间

设置消息通知免打扰时间。在免打扰时间内接收到消息时，会根据该接口设置的免打扰级别判断是否需要推送消息通知。

方法

```
Future<int> changeNotificationQuietHours(String startTime, int spanMinutes,
RCIMIWPushNotificationQuietHoursLevel level, {IRCIMIWChangeNotificationQuietHoursCallback? callback});
```

参数说明

参数名	参数类型	描述
startTime	String	开始消息免打扰时间，格式为 HH:MM:SS
spanMinutes	int	需要消息免打扰分钟数，0 < spanMinutes < 1440 （比如，您设置的起始时间是 00：00， 结束时间为 01:00，则 spanMinutes 为 60 分钟。设置为 1439 代表全天免打扰（23 60 + 59 = 1439））
level	RCIMIWPushNotificationQuietHoursLevel	消息通知级别
callback	IRCIMIWChangeNotificationQuietHoursCallback	事件回调。SDK 从 5.3.1 版本开始支持 callback 方式回调。从 5.4.0 版本废弃该接口的其他回调方式。如果传入了 callback 参数，仅触发 callback 回调。

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWChangeNotificationQuietHoursCallback? callback =
IRCIMIWChangeNotificationQuietHoursCallback(onNotificationQuietHoursChanged: (int? code) {
//...
});

int? ret = await engine?.changeNotificationQuietHours(startTime, spanMinutes, level, callback:callback);
```

回调方法

- **onConversationNotificationLevelChanged**

```
Function(int? code, RCIMIWConversationType? type, String? targetId, String? channelId,
RCIMIWPushNotificationLevel? level)? onConversationNotificationLevelChanged;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
type	RCIMIWConversationType	会话类型
targetId	String	会话 ID
channelId	String	频道 ID，仅支持超级群使用，其他会话类型传 null 即可。
level	RCIMIWPushNotificationLevel	消息通知级别

代码示例

```
engine?.onConversationNotificationLevelChanged = (int? code, RCIMIWConversationType? type, String?
targetId, String? channelId, RCIMIWPushNotificationLevel? level) {
//...
};
```

移除免打扰时间

您可以调用以下方法, level设置为 none 将免打扰时间段设置移除。

方法

```
Future<int> removeNotificationQuietHours({IRCIMIWRemoveNotificationQuietHoursCallback? callback});
```

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWRemoveNotificationQuietHoursCallback? callback =
IRCIMIWRemoveNotificationQuietHoursCallback(onNotificationQuietHoursRemoved: (int? code) {
//...
});

int? ret = await engine?.removeNotificationQuietHours(callback:callback);
```

回调方法

- **onNotificationQuietHoursRemoved**

```
Function(int? code)? onNotificationQuietHoursRemoved;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常

代码示例

```
engine?.onNotificationQuietHoursRemoved = (int? code) {
//...
};
```

获取免打扰时间

您可以通过以下方法获取免打扰时间段设置。在免打扰时间内接收到消息时，会根据当前免打扰级别判断是否需要推送消息通知。

方法

```
Future<int> getNotificationQuietHours({IRCIMIWGetNotificationQuietHoursCallback? callback});
```

返回值

返回值	描述
Future<int>	当次接口操作的状态码。0 代表调用成功 具体结果需要实现接口回调，非 0 代表当前接口调用操作失败，不会触发接口回调，详细错误参考错误码

代码示例

```
IRCIMIWGetNotificationQuietHoursCallback? callback = IRCIMIWGetNotificationQuietHoursCallback(onSuccess:
(String? startTime, int? spanMinutes, RCIMIWPushNotificationQuietHoursLevel? level) {
//...
}, onError: (int? code) {
//...
});

int? ret = await engine?.getNotificationQuietHours(callback:callback);
```

回调方法

- **onNotificationQuietHoursLoaded**

```
Function(int? code, String? startTime, int? spanMinutes, RCIMIWPushNotificationQuietHoursLevel? level)?
onNotificationQuietHoursLoaded;
```

参数说明

参数名	参数类型	描述
code	int	接口回调的状态码，0 代表成功，非 0 代表出现异常
startTime	String	开始消息免打扰时间
spanMinutes	int	已设置的屏蔽时间分钟数，0 < spanMinutes < 1440
level	RCIMIWPushNotificationQuietHoursLevel	消息通知级别

代码示例

```
engine?.onNotificationQuietHoursLoaded = (int? code, String? startTime, int? spanMinutes,
RCIMIWPushNotificationQuietHoursLevel? level) {
//...
};
```

推送概述

更新时间:2024-08-30

Flutter 仅封装了设置推送参数的接口，推送功能依赖 Android、iOS 平台的推送能力。

集成推送功能

请参考 Android、iOS 端的推送文档进行集成。

- Android 推送
- iOS 推送

设置 Android 第三方推送

```
class RCIMIWPushOptions {
// 小米推送 Id
String? idMI;
// 小米推送 appKey
String? appKeyMI;
// 魅族推送 Id
String? appIdMeizu;
// 魅族推送 appKey
String? appKeyMeizu;
// Oppo 推送 appKey
String? appKeyOPPO;
// Oppo 推送 appSecret
String? appSecretOPPO;
// 是否开启华为推送
bool? enableHWPush;
// 是否开启 FCM 推送
bool? enableFCM;
// 是否开启 Vivo 推送
bool? enableVIVOPush;
}
```

代码示例

```
RCIMIWEngineOptions options = RCIMIWEngineOptions.create();
RCIMIWPushOptions pushOptions = RCIMIWPushOptions.create(
idMI: AndroidPushInfo.idMI,
appKeyMI: AndroidPushInfo.appKeyMI,
appIdMeizu: AndroidPushInfo.appIdMeizu,
appKeyMeizu: AndroidPushInfo.appKeyMeizu,
appKeyOPPO: AndroidPushInfo.appKeyOPPO,
appSecretOPPO: AndroidPushInfo.appSecretOPPO,
enableHWPush: true,
enableFCM: false,
enableVIVOPush: true,
);
options.pushOptions = pushOptions;
RCIMIWEngine e = await RCIMIWEngine.create(appkey, options);
```

> 安卓厂商推送，除了调用上述接口外，还需要在原生层执行相关配置。安卓推送

iOS 推送

前期准备

参考 iOS 推送

请求推送权限

```
- (BOOL)application:(UIApplication )application didFinishLaunchingWithOptions:(NSDictionary
)launchOptions {
// App 启动后向用户请求推送权限
if ([[UIDevice currentDevice].systemVersion floatValue] >= 10.0) {
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
center.delegate = self;
[center requestAuthorizationWithOptions:(UNAuthorizationOptionAlert | UNAuthorizationOptionBadge |
UNAuthorizationOptionSound) completionHandler:^(BOOL granted, NSError * _Nullable error) {
if (granted) {
//点击允许
[center getNotificationSettingsWithCompletionHandler:^(UNNotificationSettings * _Nonnull settings) {

}];

} else {

}
}];
} else if ([[UIDevice currentDevice].systemVersion floatValue] > 8.0) {
//iOS8 - iOS10
[application registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:UIUserNotificationTypeAlert | UIUserNotificationTypeSound | UIUserNotificationTypeBadge
categories:nil]];
}

// 注册获得 device Token
[application registerForRemoteNotifications];
```

```
- (void)application:(UIApplication )application
didRegisterUserNotificationSettings:(UIUserNotificationSettings )notificationSettings{
[application registerForRemoteNotifications];
}
```

设置 deviceToken

```
- (void)application:(UIApplication )application didRegisterForRemoteNotificationsWithDeviceToken:(NSData
)deviceToken {
// Flutter SDK 内部已经执行了 iOS 原生 SDK setDeviceToken 方法，用户无需调用
}
```

配置消息的推送属性

更新时间:2024-08-30

在发送消息时，您可以通过设置消息的 pushOptions 对象，对单条消息的推送行为进行个性化配置。

示例代码

```
RCIMIWTextMessage? textMessage = await engine?.createTextMessage(
conversationType,
targetId,
channelId,
text,
);

RCIMIWAndroidPushOptions androidPushOptions = RCIMIWAndroidPushOptions.create();
// 此处可配置 Android 参数

RCIMIWIOSPushOptions iosPushOptions = RCIMIWIOSPushOptions.create();
// 此处可配置 iOS 参数

textMessage.pushOptions = RCIMIWMessagePushOptions.create(
disableNotification: false,
pushContent: arg['pushContent'],
pushData: arg['pushData'],
androidPushOptions: androidPushOptions,
iOSPushOptions: iosPushOptions
);
code = await engine?.sendMessage(message);
```

pushOptions 属性说明

参数	类型	说明
disableNotification	bool	通知栏是否屏蔽通知标题，true 不显示通知标题，false 显示通知标题，默认情况下融云单聊消息通知标题为用户名、群聊消息为群名称，设置后不会再显示通知标题。
disablePushTitle	bool	是否屏蔽通知标题。此属性只针目标用户为 iOS 平台时有效，Android 第三方推送平台的通知标题为必填项，所以暂不支持。
pushTitle	string	推送标题，此处指定的推送标题优先级最高。如不设置，可参考用户内容类消息格式 中对各内置消息类型默认推送通知标题与推送通知内容的说明。
pushContent	string	推送内容。此处指定的推送内容优先级最高。如不设置，可参考用户内容类消息格式 中对各内置消息类型默认推送通知标题与推送通知内容的说明。
pushData	String	远程推送附加信息。如不设置，则使用消息发送参数中设置的 pushData 值。

参数	类型	说明
forceShowDetailContent	bool	是否越过目标客户端的配置，强制在推送通知内显示通知内容（pushContent）。 客户端设备可通过 setPushContentShowStatus 设置在接收推送通知时仅显示类似「您收到了一条通知」的提醒。发送消息时，可设置 forceShowDetailContent 为 1 越过该配置，强制目标客户端在此条消息的推送通知中显示推送内容。
iOSPushOptions	RCIMIWIOSPushOptions	iOS 平台推送通知配置。目标端设备为 iOS 平台设备时，适用该配置。详细说明见 iOSPushOptions 属性说明。
androidPushOptions	RCIMIWAndroidPushOptions	Android 平台推送通知配置。目标端设备为 Android 平台设备时，适用该配置。详细说明见 RCIMIWAndroidPushOptions 属性说明。
templateId	String	推送模板 ID。设置后根据目标用户通过 setPushLanguageCode 设置的语言环境，匹配模板中设置的语言内容进行推送，未匹配成功时使用默认内容进行推送。 模板内容在控制台 > 自定义推送文案中进行设置，具体操作请参见配置和使用自定义多语言推送模板(./push-template)。
voIPPush	bool	如果对端设备是 iOS，设置 isVoIPPush 为 True，会走 VoIP 通道推送 Push。

- iOSPushOptions 属性说明

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）。
category	String	iOS 富文本推送的类型开发者自己定义，自己在 App 端进行解析判断，与 richMediaUri 一起使用，当设置 category 后，推送时默认携带 mutable-content
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）。
richMediaUri	String	iOS 富文本推送内容的 URL，与 category 一起使用。

- androidPushOptions 属性说明

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。
pushTypeVIVO	RCIMIWVIVOPushType	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。
collapseKeyFCM	String	FCM 推送的通知分组 ID。SDK 5.1.3 及以上版本支持该字段。注意，如使用该字段，请确保控制台的 FCM 推送配置中推送方式为通知消息方式。
imageUrlFCM	String	FCM 推送的通知栏右侧图标 URL。如果不设置，则不展示通知栏右侧图标。SDK 5.1.3 及以上版本支持该字段。注意，如使用该字段，请确保控制台的 FCM 推送配置鉴权方式为证书，推送方式为通知消息方式。

参数	类型	说明
importanceHW	String	华为推送消息级别
imageUrlHW	String	华为通知栏消息右侧大图标 URL，如果不设置，则不展示通知栏右侧图标。
imageUrlMi	String	小米 Large icon 链接
channelIdFCM	String	FCM 通知的频道 ID，该应用程序必须使用此频道 ID 创建一个频道，然后才能收到带有该频道 ID 的任何通知。

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。

外部链接

您可以通过以下链接了解更多与 Channel ID 相关的信息：

- 小米推送：小米推送消息分类新规
- 华为推送：自定义通知渠道
- OPPO 推送：推送私信通道申请
- vivo 推送：消息分类功能说明

用户级推送配置

更新时间:2024-08-30

用户级别推送配置是指针对 App 当前登录用户的推送配置。

> 提示
>
> - 用户级别推送配置区别于 App Key 级别推送配置。App Key 级别的推送配置针对 App 下所有用户。您可以在控制台调整部分 App Key 级别的推送服务配置。
> - 用户级别推送配置要求 App Key 已开通用户级别功能设置。如需开通，请提交工单。

设置用户推送语言偏好

为当前登录用户设置推送通知的展示语言偏好。在用户未设置偏好的情况下，使用 App Key 级别的 **Push** 语言设置。

融云内置消息类型的默认推送内容中含有部分格式文本字符串。例如，默认情况下用户收到单聊会话的文件消息推送时，推送通知内容中将显示简体中文字符串“[文件]”。如果用户将自己的推送语言偏好修改为美国英语 en_US，则再接收到文件消息时，通知内容中的格式文本字符串将为“[File]”。

上例中的 “[文件]” “[File]” 即格式文本字符串。目前融云服务端为内置消息类型的推送内容提供了格式文本字符串，支持简体中文 zh_CN、美国英语 en_US、阿拉伯语 ar_SA。

接口说明

方法：

```
Future<int> changePushLanguage(
String language,
)
```

调用示例：

```
int? code = await engine?.changePushLanguage(
language,
);
```

设置成功后，当前用户接收内置消息类型的推送通知时，推送内容中的格式文本字符串会根据对应语种进行调整。

参数说明

参数	类型	说明
language	String	设置推送通知显示的语言。目前融云支持的内置推送语言为 zh_CN、en_US、ar_SA。自定义推送语言请与 控制台 > 自定义推送文案 中的语言标识保持一致。

目前融云支持的内置推送语言为简体中文 zh_CN、美国英语 en_US、阿拉伯语 ar_SA。App 可以配合使用自定义多语言推送模板功能，可以实现在一条推送通知中支持更多推送语言。

您可以修改 App Key 在融云的默认推送语言配置。如有需要，请提交工单申请更改 App Key 级别的 **Push 语言**。

设置用户推送通知详情偏好

在用户未设置偏好的情况下，默认会展示推送通知内容。该功能支持当前登录用户设置自己的推送通知中是否需要展示推送通知的内容详情。

接口说明

所属类：RCIMIWEngine

方法：

```
Future<int> changePushContentShowStatus(
bool showContent,
)
```

调用示例：

```
int? code = await IMEngineManager().engine?.changePushContentShowStatus(
showContent,
);
```

如果设置为不显示详情，推送通知将显示为格式文本字符串“您收到了一条通知”（该格式文本字符串支持简体中文 zh_CN、美国英语 en_US、阿拉伯语 ar_SA）。

参数说明

参数	类型	说明
showContent	bool	是否显示推送详情。 true, 显示详情 false, 不显示详情，推送通知将显示为“您收到了一条通知”

请注意，发送消息时可以指定 forceShowDetailContent 强制越过消息接收者的该项配置，强制显示推送通知内容详情。以下列出了部分平台的配置：

- Android：`Message` 的 MessagePushConfig 属性。参见 Android 端「发送消息」文档中的 **MessagePushConfig 属性说明**。
- iOS：`RCMessage` 的 RCMessagePushConfig 属性。参见 iOS 端「APNs 推送开发指南」下的**配置消息的推送属性**。
- Web：IPushConfig
- IM Server API：若接口提供 `forceShowPushContent` 参数，则支持该功能。

您也可以修改 App Key 在融云的默认配置。如有需要，请提交工单申请更改 App Key 级别的 **推送通知详情**。关闭后，所有推送通知均默认不显示推送内容详情。

设置用户多端时接收推送偏好

为当前登录用户设置在 Web 端或 PC 端在线时，离线的移动端设备是否需要接收推送通知。请注意，该接口仅在 App Key 已开通 **Web/PC 在线手机端接收 Push** 服务后可用。

您可以前往控制台的免费基础功能 页面修改 App Key 级别配置。

- 如果 App Key 未开通 **Web/PC 在线手机端接收 Push**，则所有 App 用户在 Web 端或 PC 端在线时，不在线的移动端不会收到推送。不支持 App 用户修改自己的偏好。
- 如果 App Key 已开通 **Web/PC 在线手机端接收 Push**，当前登录用户可自行关闭或开启该行为。

接口说明

方法

```
Future<int> changePushReceiveStatus(
bool receive,
)
```

调用示例：

```
int? code = await engine?.changePushReceiveStatus(
receive,
);
```

参数说明

参数	类型	说明
receive	bool	其它端在线时，移动端是否接受推送 true, 接受推送 false, 不接受推送

自定义多语言推送模板

更新时间:2024-08-30

融云支持通过推送模板功能实现多语言推送。服务端会根据 App 用户通过客户端上报的推送语言，从指定推送模板中匹配对应语言的推送内容进行远程推送。

> 提示
>
> - 客户端 SDK 从 5.0.0 版本开始，支持在发消息时指定使用一个推送模板。
> - App 用户需要通过客户端 SDK 设置接收推送的语言，否则服务端会默认使用 App Key 级别的 **Push 语言**设置为其匹配推送文案。

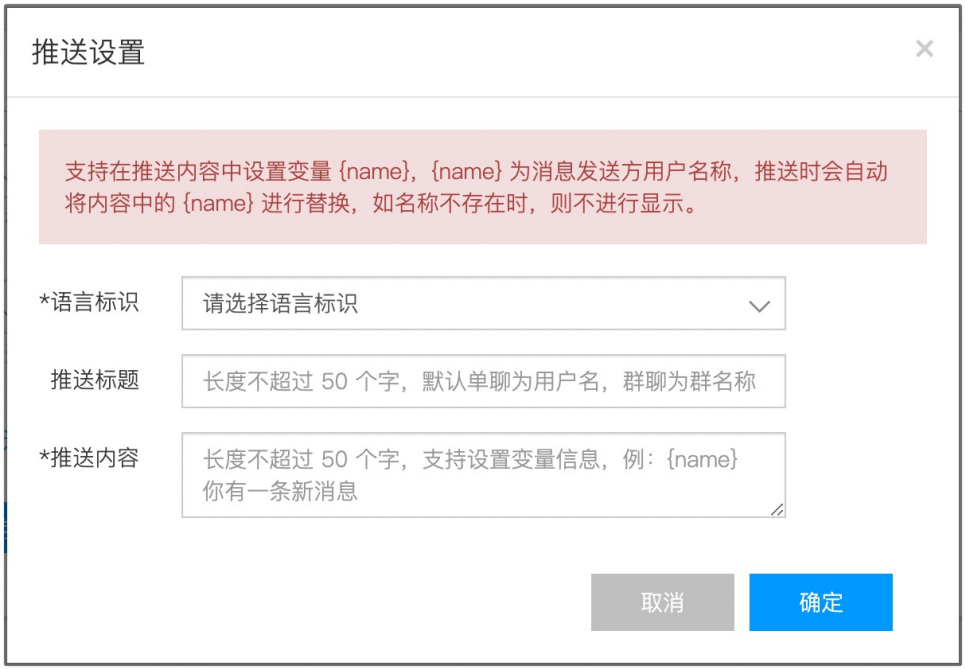
创建多语言推送模板

在使用推送模板功能前，必须在控制台创建多语言推送模板。

1. 访问控制台 自定义推送文案 页面，创建推送模板，最多可创建 100 个。

   - **推送模板 ID**：推送模板的唯一标识。发送消息时，若使用该推送模板，需填入此模板 ID。
   - **推送模板名称**：设置推送模板的名称。
   - **推送内容设置**：每个推送模板中支持设置多个语言的推送内容，每种语言对应一条推送标题和一条推送内容。

2. 添加推送内容。您可以按**语言标识**逐个设置对应的推送标题与推送内容。例如，模板 `asia` 中可同时添加 zh_CN、zh_TW、zh_HK、ja_JP、ko_KR 等多种语言的对应文案。



- **语言标识**：从下拉菜单中选择一个语言标识，如 en_US，以添加与该语言匹配的文案。
- **推送标题**：设置对应语言的通知栏标题。可选参数，默认单聊为用户名，群聊、超级群为群名称。请注意发消息时指定的推送标题会覆盖模板中的标题，导致推送模板中设置的标题不生效。
- **推送内容**：设置对应语言的通知栏显示内容。请注意发消息时指定的推送内容会覆盖模板中的内容，导致推送模板中设置的内容不生效。

创建模板后 30 分钟生效。

使用自定义推送模板

客户端 SDK 支持在发送消息时，在消息推送属性（`pushOptions`）中传入模版 ID（`templateId`），为消息启用指定的多语言推送模板。设置后，如该消息触发推送，服务端会根据指定推送模板中的多语言内容进行推送。

如果接收方客户端设置了推送语言，服务端将从推送模板中匹配用户上报的推送语言进行远程推送。如果用户未设置，或无法匹配用户推送语言，服务端根据 App 在融云服务端的默认推送语言内容进行推送。

客户端 SDK 在设置接收推送的语言时，语言标识需要控制台的语言标识一致，才能匹配成功。例如，客户端设置推送语言为美式英语 en-US，而推送模板中美式英语的表示法为 en_US，则无法成功匹配。

> 提示
>
> - 关于如何指定推送模版 ID（`templateId`），详见 配置消息的推送属性。注意，如需使用推送模板功能，请将消息推送属性中推送标题与推送内容留空。
> - 关于客户端如何设置接收推送的语言，详见 用户级推送配置 > 推送语言。
> - 您可以修改 App Key 在融云的默认推送语言配置。如有需要，请提交工单申请更改 App Key 级别的 **Push 语言**。

内容审核概述

更新时间:2024-08-30

即时通讯支持对 IM 内容进行审核。

- 即时通讯（IM）服务已内置敏感词机制。注意，敏感词机制仅是一种基础保护机制，且仅限于文本内容（默认最多 50 个敏感词），不可替代专业内容审核服务。
- 融云的内容审核服务产品中的 IM 审核服务，可为 IM 内容提供全面的保障与支持，支持审核文本、图片、语音片段、小视频，精准识别敏感信息。
- 如需自行实现审核或对接第三方审核服务，可以使用**消息回调服务**。

如果消息因被判定违规导致无法下发收件人，默认情况下消息发送者不会收到通知。如果 App 希望通知消息发送者消息已被拦截，可提交工单开通**含敏感词消息屏蔽状态回调发送端**，并在客户端设置监听（要求 Android/iOS SDK 版本 ≧ 5.1.4，Web ≧ 5.0.2）。详见敏感信息拦截回调。

敏感词机制

> 提示
>
> - 客户端不提供针对该功能的管理接口，仅提供回调接口，可在消息被判定为不下发时通知消息发送方。详见敏感信息拦截回调。
> - 如需审核文本（支持语义检测）、图片、语音片段、小视频，建议使用融云提供的内容审核服务产品。

敏感词机制是一种基础保护机制，仅支持对文本消息内容中的敏感词进行识别与过滤。对命中敏感词的消息，您可以选择进行屏蔽该消息（不会下发给接收方），或按指定规则替换消息中的敏感词后再进行下发。

目前支持的敏感词过滤语言包括：中文、英文、日语、德语、俄语、韩语、阿拉伯语。

您可以通过以下方式管理 App Key 下开发环境或生产环境的敏感词：

功能描述	客户端 API	融云服务端 API	控制台
添加敏感词，支持设置替换内容	不提供该 API	添加敏感词	敏感词设置页面
移除敏感词	不提供该 API	移除敏感词	敏感词设置页面
批量移除敏感词	不提供该 API	批量移除敏感词	敏感词设置页面
获取敏感词列表，支持获取设置的替换内容	不提供该 API	获取敏感词列表	敏感词设置页面

默认行为

- 默认最多设置 50 个敏感词。

- 默认仅针对从客户端 SDK 发送的消息生效。
- 默认仅支持识别官方内置的文本消息类型（消息标识为 `RC:TxtMsg`）中的敏感词。支持单聊、群聊、聊天室、超级群会话。超级群中文本消息修改后的内容默认也会敏感词识别、拦截或过滤。

调整配置

- **IM 旗舰版或 IM 尊享版**可以在控制台 IM 服务管理 页面的**扩展服务**标签下自行调整敏感词上限数。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。
- 如果您对使用服务端 API 发送的消息进行敏感词过滤，可以在控制台的免费基础功能界面 打开 **Server API 发送消息过滤敏感词**开关。
- 如果您需要对自定义消息类型启用敏感词机制，可以在敏感词设置 页面点击**设置自定义消息**。提供自定义消息的消息类型的 `ObjectName`，及该消息类型下内容（Content）JSON 结构中对应的键值 Key，即可对该 Key 所对应的 Value 值进行敏感词过滤处理。

IM 内容审核服务

> 提示
>
> 客户端不提供针对该功能的管理接口，仅提供回调接口，可在消息被判定为不下发时通知消息发送方。详见敏感信息拦截回调。

如果您希望全面审核 IM 内容，可以使用融云的内容审核服务产品，该产品提供 **IM 审核服务**与**音视频审核服务**。

IM 审核针对即时通讯业务，具体可提供以下能力：

- 审核文本内容
- 审核图片
- 审核语音片段
- 审核小视频
- 审核自定义消息类型（需要提交工单申请）
- 审核超级群业务中的消息修改
- 从控制台查看审核报告
- 从控制台查询 IM 审核记录
- 审核结果回调

您可以在控制台的 IM & 音视频审核 页面开通 **IM 审核服务**，配置接收审核结果回调的地址。详见服务端文档审核结果回调 。

IM 内容审核计费

内容审核服务为付费服务，开发环境可免费体验，生产环境下需预存才能使用服务。具体计费说明详见资费标准 · IM 审核 。

消息回调服务

如果您希望对接自己的审核系统或其他第三方内容审核服务，可以使用消息回调服务 。

消息回调服务（原模版路由）提供一种消息过滤机制。您可以根据发送用户 ID、接收用户 ID、消息类型、会话类型等参数，将相应的消息同步到您指定的服务器。超级群业务中，修改消息内容、更新消息扩展也支持通过消息回调同步到您指定的服务器。

消息同步到您指定的服务器后，可以使用您自己的审核系统执行内容审核，也可以对接其他第三方审核系统。融云服务端会根据您应用服务器返回的响应结果，决定是否将消息下发、是否替换消息中的内容，以及如何进行内容替换。

您可以通过控制台的消息回调服务页面管理 App Key 下开发环境或生产环境的消息回调服务状态和路由规则。

关于如何创建路由规则，以及回调参数的具体说明，请参见消息回调服务文档。

消息回调服务计费

费用以融云官方价格说明页面及计费说明文档为准。

敏感信息拦截回调

更新时间:2024-08-30

默认情况下，消息发送方无法感知消息是否已被融云审核服务拦截。如果 App 希望在消息因触发审核规则而无法下发时通知消息发送方，可开通使用敏感信息拦截回调服务。

融云的内容审核服务（包括消息敏感词、IM 审核服务、消息回调服务），可能在以下情况下拦截消息：

- 文本消息内容命中了融云内置的消息敏感词，导致消息不下发给接收方。
- 文本消息内容命中了您自定义的消息敏感词（屏蔽敏感词），导致消息不下发给接收方。
- 消息命中了 **IM 审核服务**，或**消息回调服务**设置的审核规则，导致消息不下发给接收方。

开通服务

如有需求，请提交工单，申请开通**含敏感词消息屏蔽状态回调发送端**。

设置敏感词拦截监听器

您可以通过 onMessageBlocked 方法设置敏感词拦截监听器，监听到被拦截的消息以及拦截原因。

方法

```
Function(RCIMIWBlockedMessageInfo? info)? onMessageBlocked;
```

调用示例

```
engine?.onMessageBlocked = (RCIMIWBlockedMessageInfo? info) {
//...
};
```

参数说明

参数	类型	说明
info	RCIMIWBlockedMessageInfo	被拦截消息详细信息

- RCIMIWBlockedMessageInfo 里包含了被拦截消息的相关信息，您可以通过下表列出的方法获取：

方法名称	说明
conversationType	获取被拦截消息所在会话的会话类型

方法名称	说明
targetId	获取被拦截消息所在的会话 Id
blockMsgUId	获取被拦截消息的唯一 Id
blockType	获取消息被拦截的原因。
extra	获取被拦截消息的附加信息

新旧 API 对照速查表

更新时间:2024-08-30

旧版本接口	新版本接口
init	create
setAndroidPushConfig	在 create 中配置
connect	connect（开发文档）
disconnect	disconnect（开发文档）
setServerInfo	在 create 中配置
setStatisticServer	在 create 中配置
sendMessage	sendMessage（开发文档）sendMediaMessage（开发文档）
sendMessageCarriesPush	sendMessage（开发文档）sendMediaMessage（开发文档）
sendMessageWithCallBack	sendMessage（开发文档）sendMediaMessage（开发文档）
sendIntactMessageWithCallBack	sendMessage（开发文档）sendMediaMessage（开发文档）
sendDirectionalMessage	sendGroupMessageToDesignatedUsers（开发文档）
sendDirectionalMessageWithOption	sendGroupMessageToDesignatedUsers（开发文档）
cancelSendMediaMessage	cancelSendingMediaMessage
getHistoryMessage	getMessages（开发文档）
getHistoryMessages	getMessages（开发文档）
clearHistoryMessages	clearMessages（开发文档）
getMessage	getMessages（开发文档）
getConversationList	getConversations（开发文档）
getConversationListByPage	getConversations（开发文档）
getConversation	getConversation（开发文档）
removeConversation	removeConversation（开发文档）
clearMessagesUnreadStatus	clearUnreadCount（开发文档）
joinChatRoom	joinChatRoom（开发文档）
joinExistChatRoom	joinChatRoom（开发文档）
quitChatRoom	leaveChatRoom（开发文档）
getRemoteHistoryMessages	getMessages（开发文档）
getMessages	getMessages（开发文档）

旧版本接口	新版本接口
getMessages	getMessages（开发文档）
insertIncomingMessage	insertMessage（开发文档）
insertOutgoingMessage	insertMessage（开发文档）
batchInsertMessage	insertMessages（开发文档）
deleteMessages	clearMessages（开发文档）
deleteMessageByIds	deleteLocalMessages（开发文档）
getFirstUnreadMessage	getFirstUnreadMessage
getTotalUnreadCount	getTotalUnreadCount（开发文档）
getUnreadCount	getUnreadCount（开发文档）
getUnreadCountConversationTypeList	getUnreadCountByConversationTypes（开发文档）
setConversationNotificationStatus	changeConversationNotificationLevel（开发文档）
getConversationNotificationStatus	getConversationNotificationLevel（开发文档）
getBlockedConversationList	getBlockedConversations（开发文档）
setConversationToTop	changeConversationTopStatus（开发文档）
addToBlackList	addToBlacklist（开发文档）
removeFromBlackList	removeFromBlacklist（开发文档）
getBlackListStatus	getBlacklistStatus（开发文档）
getBlackList	getBlacklist（开发文档）
sendReadReceiptMessage	sendPrivateReadReceiptMessage（开发文档）
sendReadReceiptRequest	sendGroupReadReceiptRequest（开发文档）
sendReadReceiptResponse	sendGroupReadReceiptResponse（开发文档）
syncConversationReadStatus	syncConversationReadStatus（开发文档）
setNotificationQuietHours	changeNotificationQuietHours（开发文档）
removeNotificationQuietHours	removeNotificationQuietHours（开发文档）
getNotificationQuietHours	getNotificationQuietHours（开发文档）
getUnreadMentionedMessages	getUnreadMentionedMessages
setChatRoomEntry	addChatRoomEntry（开发文档）
forceSetChatRoomEntry	addChatRoomEntry（开发文档）
getChatRoomEntry	getChatRoomEntry（开发文档）
getAllChatRoomEntries	getAllChatRoomEntries（开发文档）
removeChatRoomEntry	removeChatRoomEntry（开发文档）
forceRemoveChatRoomEntry	removeChatRoomEntry（开发文档）
setChatRoomEntries	addChatRoomEntries（开发文档）
removeChatRoomEntries	removeChatRoomEntries（开发文档）
recallMessage	recallMessage（开发文档）
getTextMessageDraft	getDraftMessage（开发文档）
saveTextMessageDraft	saveDraftMessage（开发文档）

旧版本接口	新版本接口
searchConversations	searchConversations
searchMessages	searchMessages （开发文档）
sendTypingStatus	sendTypingStatus （开发文档）
downloadMediaMessage	downloadMediaMessage （开发文档）
deleteRemoteMessages	deleteMessages （开发文档）
clearMessages	clearMessages （开发文档）
setMessageReceivedStatus	changeMessageReceiveStatus
setMessageSentStatus	changeMessageSentStatus
clearConversations	removeConversations （开发文档）
setReconnectKickEnable	在 create 中配置
cancelDownloadMediaMessage	cancelDownloadingMediaMessage （开发文档）
getRemoteChatRoomHistoryMessages	getChatRoomMessages （开发文档）
imageCompressConfig	在 create 中配置
getMessage	getMessageById （开发文档）
getMessageByUId	getMessageByUId （开发文档）
updateMessageExpansion	updateMessageExpansion （开发文档）
removeMessageExpansionForKey	removeMessageExpansionForKeys （开发文档）
syncUltraGroupReadStatus	syncUltraGroupReadStatus （开发文档）
getConversationListForAllChannel	getConversationsForAllChannel （开发文档）
getUltraGroupUnreadMentionedCount	getUltraGroupUnreadMentionedCount （开发文档）
sendUltraGroupTypingStatus	sendUltraGroupTypingStatus （开发文档）
deleteUltraGroupMessagesForAllChannel	clearUltraGroupMessagesForAllChannel （开发文档）
deleteUltraGroupMessages	clearUltraGroupMessages （开发文档）
deleteRemoteUltraGroupMessages	clearUltraGroupMessages （开发文档）
modifyUltraGroupMessage	modifyUltraGroupMessage （开发文档）
updateUltraGroupMessageExpansion	updateUltraGroupMessageExpansion （开发文档）
removeUltraGroupMessageExpansion	removeUltraGroupMessageExpansion （开发文档）
recallUltraGroupMessage	recallUltraGroupMessage （开发文档）
getBatchRemoteUrtraGroupMessages	getBatchRemoteUltraGroupMessages （开发文档）
setNotificationQuietHoursLevel	changeNotificationQuietHours （开发文档）
getNotificationQuietHoursLevel	getNotificationQuietHours （开发文档）
setConversationChannelNotificationLevel	changeConversationNotificationLevel （开发文档）
getConversationChannelNotificationLevel	getConversationNotificationLevel （开发文档）
getConversationNotificationLevel	getConversationNotificationLevel （开发文档）
setConversationNotificationLevel	changeConversationNotificationLevel （开发文档）
setConversationTypeNotificationLevel	changeConversationTypeNotificationLevel （开发文档）
getConversationTypeNotificationLevel	getConversationTypeNotificationLevel

旧版本接口	新版本接口
setUltraGroupConversationDefaultNotificationLevel	changeUltraGroupDefaultNotificationLevel（开发文档）
getUltraGroupConversationDefaultNotificationLevel	getUltraGroupDefaultNotificationLevel（开发文档）
setUltraGroupConversationChannelDefaultNotificationLevel	changeUltraGroupChannelDefaultNotificationLevel（开发文档）
getUltraGroupConversationChannelDefaultNotificationLevel	getUltraGroupChannelDefaultNotificationLevel（开发文档）
getUltraGroupUnreadCount	getUltraGroupUnreadCount（开发文档）
getUltraGroupAllUnreadCount	getUltraGroupAllUnreadCount（开发文档）
getUltraGroupAllUnreadMentionedCount	getUltraGroupAllUnreadMentionedCount（开发文档）

旧版本回调	新版本回调
onMessageBlocked	onMessageBlocked
onConnectionStatusChange	onConnectionStatusChanged
messageExpansionDidUpdate	onRemoteMessageExpansionUpdated
messageExpansionDidRemove	onRemoteMessageExpansionForKeyRemoved
onMessageSend	onMessageSent
onMessageReceived	onMessageReceived
onMessageReceivedWrapper	onMessageReceived
onJoinChatRoom	onChatRoomJoined
onChatRoomReset	onChatRoomStatusChanged
onChatRoomDestroyed	onChatRoomStatusChanged
onQuitChatRoom	onChatRoomLeft
chatRoomKVDidSync	onChatRoomEntriesSynced
chatRoomKVDidUpdate	onChatRoomEntriesChanged
chatRoomKVDidRemove	onChatRoomEntriesChanged
onReceiveReadReceipt	onPrivateReadReceiptReceived
onMessageReceiptRequest	onGroupMessageReadReceiptRequestReceived
onMessageReceiptResponse	onGroupMessageReadReceiptResponseReceived
onTypingStatusChanged	onTypingStatusChanged
onRecallMessageReceived	onRemoteMessageRecalled
onDatabaseOpened	onDatabaseOpened

部分功能的实现方案参考

接口功能	实现方案
config	在创建引擎时提供配置信息
getChatRoomInfo	开发者服务端调用融云 Server API 进行查询，客户端根据服务端内容来处理
getConnectionStatus	开发者可以根据链接状态变化全局进行处理
onUploadMediaProgress	开发者在sendmediamessage 中可以直接处理
forwardMessageByStep	直接调用 sendMessage 处理

接口功能	实现方案
addMessageDecoder	参考 https://help.rongcloud.cn/t/topic/954
conversationDigest	参考 https://help.rongcloud.cn/t/topic/954
onDataReceived	参考 https://help.rongcloud.cn/t/topic/954
objectName 如何判断	参考 https://help.rongcloud.cn/t/topic/956

状态码

更新时间:2024-08-30

Flutter 错误码定义：

错误码	描述
0	成功
-101	参数错误
-102	引擎未初始化 或 引擎已销毁
-103	原生 SDK 抛出的错误
-104	未知错误

原生平台错误码定义：

- Android 错误码
- iOS 错误码