网易云信 Netease IM SDK 10.9.10

产品文档

文档版本：20250630

保留所有权

目录

1 接入流程

本章节介绍使用 10.x.x 系列版本网易云信即时通讯 SDK（NetEase IM SDK，简称 NIM SDK）的基本接入流程概述，您可以参考对应的文档完成 NIM SDK 接入，为应用项目实现即时通讯功能。

1.1 支持平台

10.x.x 系列版本 NIM SDK 适配了以下客户端开发平台和开发框架，其中，鸿蒙（HarmonyOS）平台适配仅在 10.x.x 系列版本 NIM SDK 中提供。您可以前往 资源下载 页面下载对应版本的 SDK 和 Demo。

1.2 流程概述

note note 使用网易云信即时通讯 10.x.x 系列版本客户端 NIM SDK 时，请同时调用配套的 服务端 API（RESTful），**请勿** 调用 9.x.x 及更低版本系列客户端 NIM SDK 配套的服务端 API。

步骤	操作
创建应用并获取 App Key	在 网易云信控制台 中创建应用，获取到该应用的密钥（App Key），后续集成 SDK 时在代码中填入使用。
开通 IM 产品	在网易云信控制台中，为已创建的应用开通 IM 产品服务。
注册 IM 账号	- 如果您有具体的业务需求，那么可以通过服务端接口（/im/v2/accounts）注册 IM 账号，具体请参考 注册 IM 账号。
集成 SDK	根据项目具体的开发平台或开发框架，将 NIM SDK 快速集成到您的项目中，填入 AppKey 等开发者身份验证信息。
初始化 SDK	初始化时，可配置第三方离线推送服务、会话已读多端同步、群消息已读和融合存储等重要功能。
登录 IM 账号	根据业务选择所需的鉴权方式，使用第三步中您创建的 IM 账号完成用户账号登录验证。
功能实现	实现即时通讯主要功能模块：

2 开始使用

本章节介绍了接入网易云信即时通讯（Instant Messages，简称 IM）产品国内单元的流程。如果您暂未明确具体的需求细节，请联系您的网易云信商务经理，沟通您的业务背景。

2.1 准备工作

根据本章节操作前，请确保您已经完成了以下设置：

1. 注册 网易云信开发者账号，并能 登录 网易云信控制台。
2. 创建应用，并获取到了应用密钥（App Key）。详细步骤，请参考 创建应用。

2.2 第一步：开通产品

网易云信 IM 产品分为免费版和套餐包版。

- 套餐包支持安卓、iOS、Web、电脑桌面等主流平台，提供单聊、群聊、文字、图片、语音、视频、地理位置等消息类型，同时支持自定义消息，满足常见的个性化需求。
- 免费版能注册的 IM 账号数上限为 100 个，套餐包没有限制。

使用网易云信 IM 套餐包功能前，您需要在网易云信控制台开通 IM 套餐包服务。开通步骤如下：

1. 在 网易云信控制台 首页 **应用管理** 列表下，选择并单击一款应用。

2. 在 **应用配置** 页面，找到并单击 **IM 即时通讯** 下的 **免费试用** 或 **正式开通**。

note note 因产品服务升级，仅部分存量客户能开通 IM 旧版套餐包（即专业版和极速版）。如果您有相关需求，请联系您的网易云信商务经理。

2.2.1 方式一：免费试用

如需免费试用 IM，则在 **免费试用** 界面指定 IM 节点的服务区域为 **国内** 或 **海外**（海外节点适用

有出海及国际化业务场景的客户），然后单击 **立即试用**。

note notice - 免费试用期限为 7 个自然日，试用结束后默认当前应用为 **IM 即时通讯 免费版**，您将无法使用 IM 套餐包支持的功能。 - 试用结束后，您可以单击 **套餐升级** 将免费版升级为套餐包，具体开通步骤请参考下文正式开通。

2.2.2 方式二：正式开通

1. 单击 **IM 即时通讯** 下的 **正式开通**。
2. 在 **套餐对比** 界面指定 IM 节点的服务区域为 **国内**。
3. 仔细查看套餐详情，并单击您适用的产品套餐。

4. 鼠标继续下滑，按应用的实际需求配置服务。
5. 按照控制台会生成的预估费用，设置不少于预估费用的 **充值金额**。

   关于具体的计费策略，请参考 计费方式 或咨询您的网易云信商务经理。

6. 勾选协议条款，并单击 **提交订单**。

7. 单击 **立即支付** 完成订单支付。

note note 请确保账号余额中有足够的金额开通服务。

7. 开通完成后基本信息页面 IM 即时通讯套餐包状态为 **已开通**，您可以单击 **功能配置** 继续开通和配置具体功能。

2.3 第二步：注册 IM 账号

网易云信 IM 账号是网易云信 IM 系统中用户的唯一身份标识。通过 IM 账号，用户可以在应用中使用各种 IM 服务。您在将自身应用与网易云信 IM 对接时，需要注册 IM 账号与自身应用中的账号相对应。您可以根据业务需要，注册多个网易云信 IM 账号来构成应用的账号体系。

为了保护用户账号的隐私和安全，建议您区分 IM 测试账号和 IM 业务账号。

2.3.1 类型一：业务账号

如果您有具体的业务需求，那么可以通过服务端 API 注册 IM 业务账号。详情请参考 注册 IM 账号。

note notice - 通过服务端 API 注册的账号不会出现在 网易云信控制台 的账号列表中。 - 通过服务端 API 注册账号成功后，网易云信将返回 account_id 与 token，请在应用服务器上维护 account_id 与 token 信息列表。

2.3.2 类型二：测试账号

如果您只需要体验产品或者快速测试功能，那么您可以在 网易云信控制台 创建 IM 测试账号。

1. 在 网易云信控制台 首页 **应用管理** 列表下，选择一款应用，进入 **应用配置** 页面。

2. 单击 **IM 即时通讯 套餐包** 下的 **功能配置** 按钮，进入 IM 即时通讯配置页。

3. 在顶部选择 **基础功能** 页签，单击 **测试账号管理** 下的 **子功能配置**。

4. 单击 **新建账号**，在弹窗填写 **账号**、**昵称**、**密码** 后，单击 **确定**。

   在 网易云信控制台 创建的 IM 账号信息，与服务端 OpenAPI 账号信息字段的映射关系如下：

   - **账号**：account_id
   - **密码**：token
   - **昵称**：name

5. （可选）对于创建好的测试账号，您可以在 网易云信控制台 修改昵称，重置密码，以及禁用操作。

2.4 第三步：配置功能

1. 在 网易云信控制台 首页 **应用管理** 列表下，选择一款应用，进入 **应用配置** 页面。
2. 单击 **IM 即时通讯 套餐包** 下的 **功能配置** 按钮进入 IM 即时通讯配置页。

3. 在顶部选择 **基础功能**、**全局功能** 或其他页签，开启或配置所需功能。

请参考下文了解 IM 基础功能和全局功能的功能说明情况。

2.4.1 基础功能

您需要单独开通和配置的基础功能说明，请参考如下列表：

- ż：表示默认开启、默认支持、有默认设置
- C：表示默认不开启、默认不支持、无默认设置

基础功能	功能介绍	默认设置
IM 服务区域		C
信令	开启信令服务	C
API 调试	API 调试测验	ż
测试账号管理	通过网易云信控制台创建的账号管理。单击 **子功能配置** 创建和管理 IM 账号。	ż
登录策略	设置用户登录应用服务器或网易云信服务器的方式。单击 **编辑** 设置 登录鉴权方式。	ż 静态 Token
多端登录模式	设置用户在不同端的登录模式。单击 **编辑** 设置 多端登录与互踢。默认为：	ż
第三方回调	单击 **编辑** 设置第三方回调的默认服务器地址。	无
iOS 应用角标未读数上限	针对 iOS 应用角标的未读消息数上限配置，最大配置为 99。	0
应用服务器 IP 白名单	仅支持设置的 IP 地址访问，多用于安全防护功能。	C
第三方厂商消息分类	开启后会将所有通过网易云信的消息默认配置为配置字段用以各平台的消息分类（单独设置的以设置的为准）。单击 **子功能配置** 设置各推送厂商的消息分类。	无
推送通知 TTL（Time to Live）	设置推送离线消息的存活时间，填写推送通知服务必须向终端节点发送消息的秒数（60 秒 ~ 604800 秒/7 天）。	默认不填写
自定义推送	配置自定义推送文案（通知栏现实的文案），最多可配 100 种自定义文案，每种自定义文案用一个自定义类型来标识。	C
添加好友逻辑配置	开启后，用户直接调用同意协议或 API 添加好友时，系统将进行判断对方是否有发起好友申请。	C
单聊消息配置	设置非好友关系是否允许发送单聊消息。	ż
消息扩展字段上限调整	单条消息可扩展字段的长度上	1024

基础功能	功能介绍	默认设置
	限。	
消息 attach 字段大小上限	修改发送图片/语音/视频/文件消息的 attach 字段上限。	4096
消息漫游	设置用户是否可拉取云端漫游消息。	C
消息撤回时长	设置用户发送消息后可在多长时间内操作撤回。	120 秒
撤回消息覆盖策略	设置用户撤回消息时，是否需要覆盖原始消息的推送。	默认不覆盖
PC 端在线，强推消息策略配置	设置当 PC 端在线时阻止强推消息推到手机端。	C
被拉黑时被拉黑者无法唤起呼叫	开启后被拉黑者无法呼叫对方。	C
IM 离线原因抄送配置	开通后抄送内 IM 的 logoutReason 字段可区分是被踢还是主动登出还是掉线。	C
聊天室离线原因抄送配置	开通后抄送内聊天室的 logoutReason 字段可区分是被踢还是主动登出还是掉线。	C
拉黑场景可以正常下发消息已读回执通知	开启后若已拉黑对方，则对方的已读回执仍可以发送。	C

2.4.2 全局功能

您需要单独开通和配置的全局功能说明，请参考如下列表：

- ż：表示默认开启、默认支持、有默认设置
- C：表示默认不开启、默认不支持、无默认设置

基础功能	功能介绍	默认值
好友数	单击 **编辑** 设置单用户可添加好友数上限。	3000 人
在线状态订阅	可获取用户的当前在线状态（一般用于用户列表、会话界面显示用户在线或离线）。	C
单向消息删除	单向删除指一方删除不影响其他方的消息存储。	C
会话置顶	会话置顶指服务端维护需要个性化配置的会话。	C
全员广播	即 广播通知，单条消息可发送给所有用户。	C
历史消息	单击 **编辑** 设置用户可查询的历史消息范围。	ż默认 1 年
高保障消息抄送	当客户服务器遇到阻塞或故障不能接受消息时，网易云信会缓存需要抄送的消息，待客户自有服务器恢复正常后重新抄送，保障抄送消息完整到达。该功能为增值服务。	C

基础功能	功能介绍	默认值
客户端反垃圾	客户端反垃圾用于配置聊天时希望被过滤的敏感词、违禁词，开通后可在子功能配置进行应用级别的客户端词条配置。该功能为增值服务。	C
登录登出事件记录查询	可随时查询用户的登录登出事件。该功能为增值服务。	C
会话消息标记	需要实现 PIN 消息 功能时，需开通此功能。类似 Slack 应用里 pin（标记）的能力，可针对某条消息做单独标记，并在标记列表单独展示。该功能为增值服务。	C
会话消息回复	需要实现 Thread 消息功能时，需开通此功能。可根据某条消息单独进入一个独立会话序。该功能为增值服务。	C
消息快捷评论	开启后，可针对某条消息直接进行点表情操作（每条消息最多点亮 100 个表情）。该功能为增值服务。	C
全文云端消息检索	开启后，针对关键词搜索，可提供应用级别的 全文检索消息。该功能为增值服务。	C
IM 存储清理任务管理	IM 存储提交系统任务自动清理。单击 **子功能配置** 新建和管理清理任务。	ż

3 快速实现消息收发

网易云信即时通讯（IM）产品提供一整套即时通讯基础能力，助您快速实现多样化的即时通讯场景。本章节主要介绍通过集成客户端 SDK（NetEase IM SDK，简称 NIM SDK）并调用 API，快速实现收发消息功能。

3.1 支持平台

本章节内容适用的开发平台或框架如下所示，涉及的接口请参考下文 相关接口 章节：

安卓	iOS	macOS/Windows	Web/uni-app/小程序	Node.js/Electron	鸿蒙	Flutter

3.2 环境准备

NIM SDK 兼容的平台和框架系统版本如下所示：

- 兼容安卓 5.0 及以上版本。
- 兼容 iOS 9.0 及以上版本，可使用 iPhone/iPad 真机或模拟器。
- 兼容鸿蒙 SDK API 11 及以上，运行环境 HarnomyOS NEXT 2.1.2.5（Canary1）以上。兼容 DevEco Studio NEXT Developer Beta1（5.0.3.300）及以上。
- 兼容 Windows7 及以上，支持 x86_64、x86 架构。
- 兼容 macOS 10.13 及以上，支持 x86_64、arm64 架构。
- 兼容微软 IE（9+）、谷歌 Chrome（4+）、微软 Edge（12+）、Mozilla Firefox（11 + ）、苹果 Safari（5+）、DCloud uni-app、微信/支付宝/百度/抖音小程序。
- 兼容 Dart 2.17.0 ~ 4.0.0 版本。

3.3 前提条件

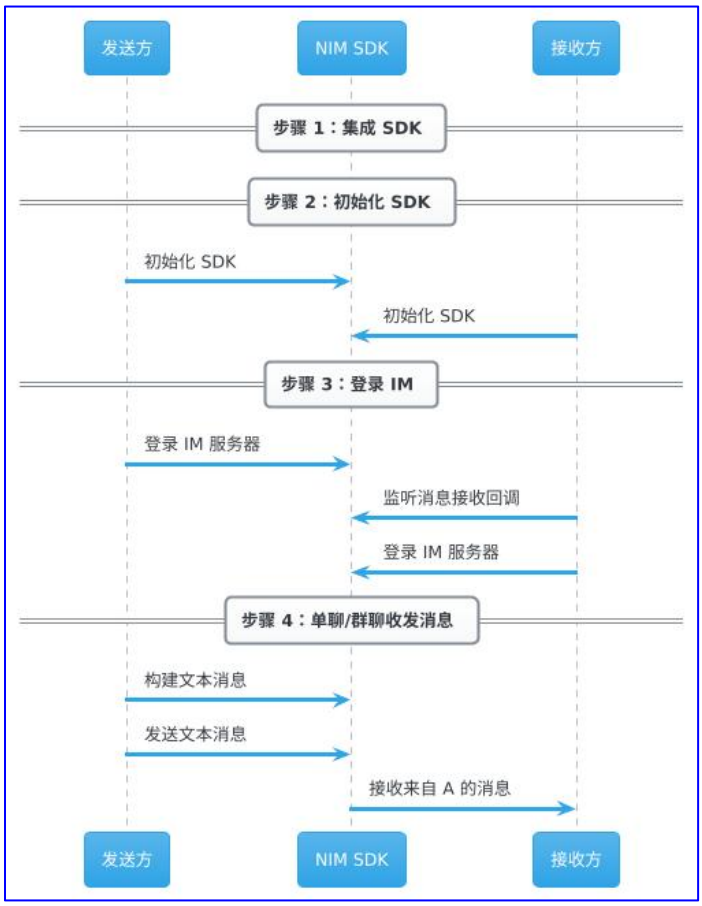
根据本章节操作前，请确保您已经完成了以下设置：

- 在 网易云信控制台 上 创建应用，获取应用密钥（App Key）。
- 注册 IM 账号，获取 IM 账号和 Token。
- 若使用聊天室功能，在 网易云信控制台 上 开通和配置聊天室功能 和调用服务端接口 /im/v2/chatrooms 创建聊天室。
- 若使用圈组功能，在 网易云信控制台 上 开通和配置圈组功能。

3.4 流程概览

单聊/群聊收发消息

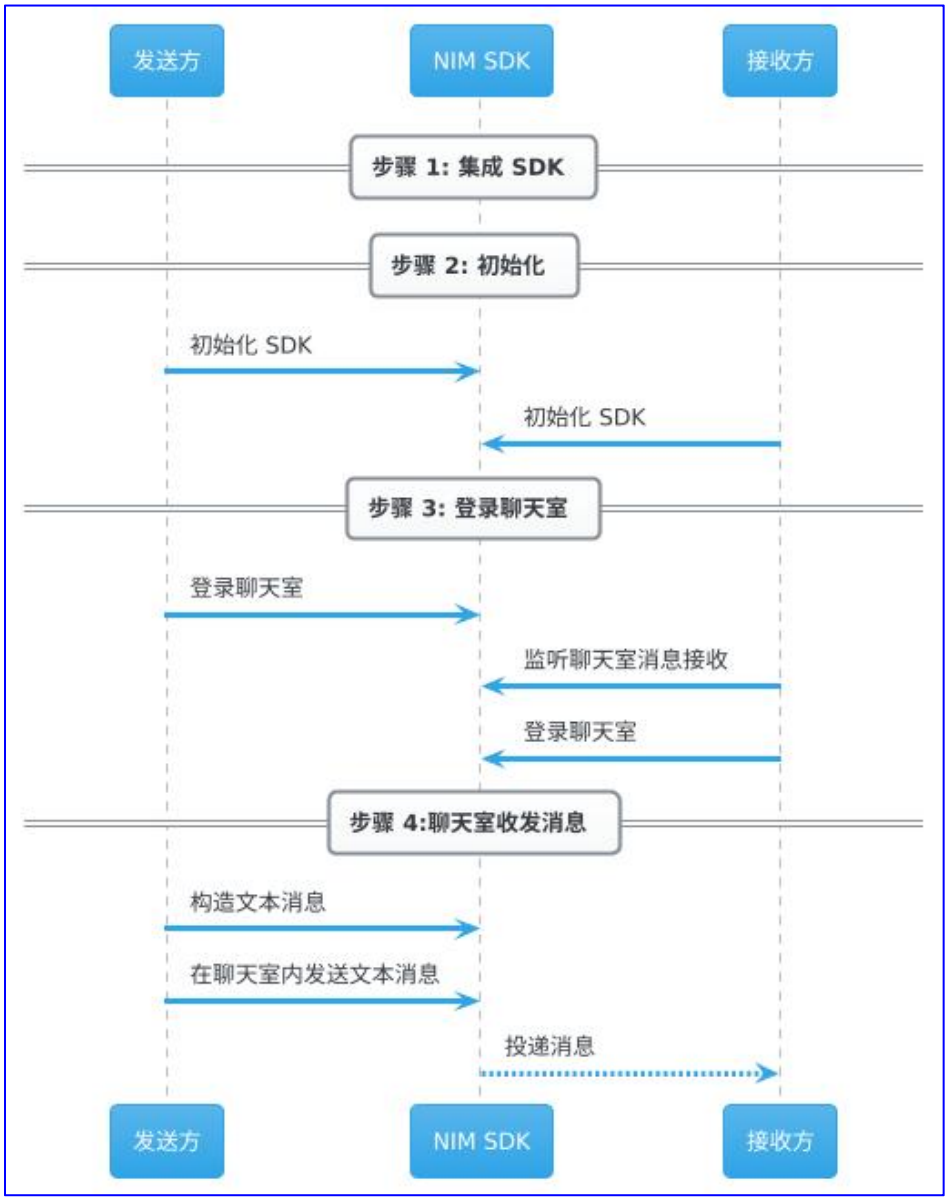
实现单聊/群聊收发消息的流程，可分为下图所示的步骤。



普通消息.png

聊天室收发消息

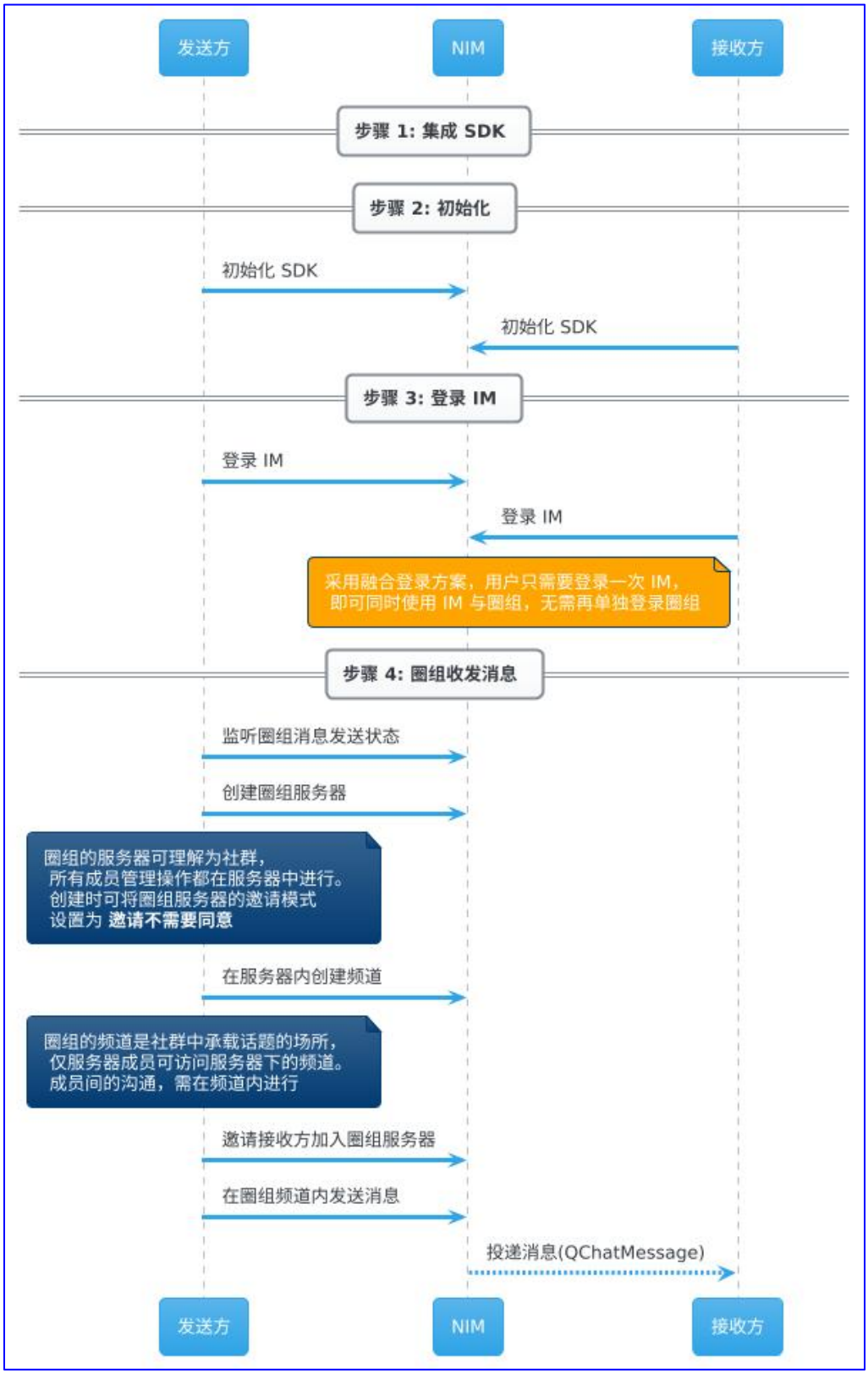
实现聊天室收发消息的流程，可分为下图所示的步骤。



聊天室.png

圈组收发消息

实现圈组收发消息的流程，可分为下图所示的步骤。



圈组.png

3.5 第一步：集成 SDK

各端 SDK 的集成请参考相关集成文档。

- 集成安卓 SDK
- 集成 iOS SDK
- 集成 macOS/Windows SDK
- 集成 Web/uni-app/小程序 SDK
- 集成鸿蒙 SDK
- 集成 Flutter SDK
- 集成 Node.js/Electron SDK

3.6 第二步：初始化 SDK

各端 SDK 的集成请参考相关集成文档。

- 安卓 SDK 初始化
- iOS SDK 初始化
- macOS/Windows SDK 初始化
- Web/uni-app/小程序 SDK 初始化
- 鸿蒙 SDK 初始化
- Flutter SDK 初始化

3.7 第三步：用户登录

3.7.1 登录 IM

客户端用户在使用网易云信即时通讯功能前需要先登录网易云信 IM 服务器。

note note 建议参考 IM 登录最佳实践 实现 IM 登录以及相应的上层应用逻辑。

调用 login 方法进行登录。以静态 Token 登录为例，示例代码如下：

安卓

```
NIMClient.getService(V2NIMLoginService.class).login("account", "token", null, new V2NIMSuccessCallback<Void>() {
    @Override
    public void onSuccess(Void unused) {
        // TODO
    }
},
    new V2NIMFailureCallback() {
    @Override
    public void onFailure(V2NIMError error) {
        int code = error.getCode();
        String desc = error.getDesc();
        // TODO
    }
});
```

iOS

```
- (void)login
{
    NSString *accountId = @"accountId";
    NSString *token = @"token";
    [[NIMSDK sharedSDK].v2LoginService login:accountId token:token
            option:nil
                success:^{
        NSLog(@"login succ");
    }
                failure:^(V2NIMError * _Nonnull error) {
        NSLog(@"login fail: error = %@", error);
    }];
}
```

macOS/Windows

```
V2NIMLoginOption option;
loginService.login(
    "accountId",
    "token",
    option,
    []() {
        // login succeeded
    },
    [](V2NIMError error) {
        // login failed, handle error
    });
```

Web/uni-app/小程序

```
try {
    await nim1.V2NIMLoginService.login("ACCOUNT_ID", "TOKEN", {
    "forceMode": false
    })
} catch (err) {
    // TODO failed, check code
    // console.log(err.code)
}
```

Node.js/Electron

```
await v2.loginService.login('accountId', 'token', {})
```

鸿蒙

```
try {
await nim.loginService.login("ACCOUNT_ID", "TOKEN", {
    forceMode: false
} as V2NIMLoginOption)
} catch (err) {
// TODO failed, check code
// console.log(err.code)
}
```

Flutter

```
final loginResult = await NimCore.instance.loginService.login(
    "ACCOUNT_ID", "TOKEN", NIMLoginOption());
```

其他登录方式请参考 登录登出 IM。

3.7.2 登录聊天室

1. 发送方和接收方调用 newInstance 方法创建聊天室实例。调用成功后，返回聊天室实例（instanceId），聊天室实例与聊天室（roomId）形成一一绑定关系。

   示例代码如下：

   安卓

```
V2NIMChatroomClient chatroomClient = V2NIMChatroomClient.newInstance();
```

   iOS

```
V2NIMChatroomClient *chatroomClient = [V2NIMChatroomClient newInstance];
```

   macOS/Windows

```
auto chatroomClient = V2NIMChatroomClient::newInstance();
    if (!chatroomClient) {
        // create instance failed
        // ...
        return;
    }
    auto instanceId = chatroomClient->getInstanceId();
    // save instanceId to cache
    // ...
```

   Web/uni-app/小程序

```
const chatroom = V2NIMChatroomClient.newInstance(
        {
            appkey: 'YOUR_APPKEY'
        }
    )
```

   Node.js/Electron

```
const chatroomClient = V2NIMChatroomClient.newInstance()
```

   鸿蒙

```
const context: common.Context = getContext(this).getApplicationContext()
    const chatroom = V2NIMChatroomClient.newInstance(context,
        {
            appkey: 'YOUR_APPKEY'
        }
    )
```

2. 发送方和接收方调用 addChatroomClientListener 方法注册聊天室登录相关监听器，包括聊天室连接状态变更、进出聊天室、被踢出聊天室。

   示例代码如下：

   安卓

```
chatroomClient.addChatroomClientListener(new V2NIMChatroomClientListener() {
        @Override
        public void onChatroomStatus(V2NIMChatroomStatus status, V2NIMError error) {
```

```
        }

        @Override
        public void onChatroomEntered() {

        }

        @Override
        public void onChatroomExited(V2NIMError error) {

        }

        @Override
        public void onChatroomKicked(V2NIMChatroomKickedInfo kickedInfo) {

        }
    });
```

iOS

```
@interface ClientListener : NSObject <V2NIMChatroomClientListener>
- (void)addToClient:(NSInteger)clientId;
@end

@implementation ClientListener
- (void)addToClient:(NSInteger)clientId
{
    V2NIMChatroomClient *instance = [V2NIMChatroomClient getInstance:clientId];
    [instance addChatroomClientListener:self];
}

- (void)onChatroomStatus:(V2NIMChatroomStatus)status
                   error:(nullable V2NIMError *)error
{

}
- (void)onChatroomEntered
{

}

- (void)onChatroomExited:(nullable V2NIMError *)error
{

}

- (void)onChatroomKicked:(V2NIMChatroomKickedInfo *)kickedInfo
{

}
@end
```

macOS/Windows

```
V2NIMChatroomClientListener listener;
    listener.onChatroomStatus = [](V2NIMChatroomStatus status, nstd::optional<V2NIMError> error) {
        // handle chatroom status
    };
```

```
    listener.onChatroomEntered = []() {
        // handle chatroom entered
    };
    listener.onChatroomExited = [](nstd::optional<V2NIMError> error) {
    // handle chatroom exited
    };
    listener.onChatroomKicked = [](V2NIMChatroomKickedInfo kickedInfo) {
        // handle chatroom kicked
    };
    chatroomClient.addChatroomClientListener(listener);
```

Web/uni-app/小程序

```
chatroom.on("onChatroomStatus", function (status: V2NIMChatroomStatus, err?: V2NIMError) {})
    chatroom.on("onChatroomEntered", function () {})
    chatroom.on("onChatroomExited", function (err?: V2NIMError) {})
    chatroom.on("onChatroomKicked", function (kickedInfo: V2NIMChatroomKickedInfo) {})
```

Node.js/Electron

```
chatroom.on("chatroomStatus", function (status: V2NIMChatroomStatus, err?: V2NIMError) {})
    chatroom.on("chatroomEntered", function () {})
    chatroom.on("chatroomExited", function (err?: V2NIMError) {})
    chatroom.on("chatroomKicked", function (kickedInfo: V2NIMChatroomKickedInfo) {})
```

鸿蒙

```
const chatroom = this.getInstance(instanceId)
    chatroom.on("onChatroomStatus", (status: V2NIMChatroomStatus, err?: V2NIMError) => {
    })
    chatroom.on("onChatroomEntered", ()=> {
    })
    chatroom.on("onChatroomExited", (err?: V2NIMError) => {
    })
    chatroom.on("onChatroomKicked", (kickedInfo: V2NIMChatroomKickedInfo)=> {
    })
```

3. 在登录聊天室之前，需要先提前获取聊天室地址。可以通过以下两种方式获取：

   - 若当前客户端已 登录 IM，那么可以通过 getChatroomLinkAddress 方法获取指定聊天室的地址。

     示例代码如下：

     安卓

```
NIMClient.getService(V2NIMLoginService.class).getChatroomLinkAddress("123", new V2
  NIMSuccessCallback<List<String>>() {
            @Override
            public void onSuccess(List<String> result) {
                // get success
            }
        }, new V2NIMFailureCallback() {
            @Override
            public void onFailure(V2NIMError error) {
                // get failed
            }
        });
```

iOS

```
NSString *roomId = @"36";
  [NIMSDK.sharedSDK.v2LoginService getChatroomLinkAddress:roomId
                                                  success:^(NSArray<NSString *> *links) {
                                                    // get success
                                                  }
                                                  failure:^(V2NIMError *error) {
                                                    // get failed
                                                  }];
```

macOS/Windows

```
loginService.getChatroomLinkAddress(
      "roomId",
      [](nstd::vector<nstd::string> linkAddresses) {
        // handle link addresses
      },
      [](V2NIMError error) {
        // handle error
      });
```

Web/uni-app/小程序

```
const addressArray = await nim.V2NIMLoginService.getChatroomLinkAddress('36', isMini
  App)
```

Node.js/Electron

```
const linkAddresses = await v2.loginService.getChatroomLinkAddress('roomId')
```

- 若当前客户端未登录 IM，那么 SDK 无法获取聊天室服务器的地址，需要客户端向开发者应用服务器请求该地址，而应用服务器需要向网易云信服务器请求，然后将请求结果原路返回给客户端。具体请参考 获取聊天室地址 服务端 API。

4. 发送方和接收方调用 enter 方法登录聊天室。以静态 Token 登录为例，示例代码如下：

安卓

```
//创建 V2NIMChatroomClient(注意：不要每次都 newInstance，用完不再使用需要 destroyInstance)
  V2NIMChatroomClient chatroomClient = V2NIMChatroomClient.newInstance();
  //获取 chatroomClient 的实例 ID，可以缓存起来，后面通过 instanceId 可以得到 V2NIMChatroo
  mClient
  int instanceId = chatroomClient.getInstanceId()

  ……

  V2NIMChatroomLinkProvider chatroomLinkProvider = new V2NIMChatroomLinkProvider() {
    @Override
    public List<String> getLinkAddress(String roomId, String accountId) {
      return "聊天室 Link 地址";
    }
  };
  V2NIMChatroomEnterParams enterParams = V2NIMChatroomEnterParams.V2NIMChatroomEnter
  ParamsBuilder.builder(chatroomLinkProvider)
  .withAccountId("账号名")
  .withToken("静态 token")
  // 按需设置
```

```
    //.withRoomNick("进入聊天室后显示的昵称")
    //.withRoomAvatar("进入聊天室后显示的头像")
    //.withTimeout("进入方法超时时间")
    //.withServerExtension("用户扩展字段")
    //.withNotificationExtension("通知扩展字段，进入聊天室通知开发者扩展字段")
    //.withTagConfig("进入聊天室标签信息配置")
    //.withLocationConfig("进入聊天室空间位置信息配置")
    //.withAntispamConfig("用户资料反垃圾检测配置");
    .build();

    V2NIMChatroomClient chatroomClient = V2NIMChatroomClient.getInstance(instanceId);
    if(chatroomClient != null){
        chatroomClient.enter(roomId, enterParams,
            new V2NIMSuccessCallback<V2NIMChatroomEnterResult>() {
                @Override
                public void onSuccess(V2NIMChatroomEnterResult result) {
                    //进入成功
                }
            },
            new V2NIMFailureCallback() {
                @Override
                public void onFailure(V2NIMError error) {
                    //进入失败
                }
            });
    }
```

iOS

```
@interface V2NIMEnterChatroom: NSObject <V2NIMChatroomLinkProvider>
    @end
    @implementation V2NIMEnterChatroom
    - (void)enter
    {
    NSString *roomId = @"36";
    //创建 V2NIMChatroomClient(注意：不要每次都 newInstance，用完不再使用需要 destroyInstance)
    V2NIMChatroomClient *client = [V2NIMChatroomClient newInstance];
    //获取 chatroomClient 的实例 ID，可以缓存起来，后面通过 instanceId 可以得到 V2NIMChatroomClient
    NSInteger instanceId = client.getInstanceId;

    V2NIMChatroomEnterParams *enterParams = [[V2NIMChatroomEnterParams alloc] init];
    enterParams.linkProvider = self;
    enterParams.accountId = @"账号名";
    enterParams.token = @"静态 token";
    // 按需设置
    // enterParams.roomNick: 进入聊天室后显示的昵称
    // enterParams.roomAvatar: 进入聊天室后显示的头像
    // enterParams.timeout: 进入方法超时时间
    // enterParams.serverExtension: 用户扩展字段
    // enterParams.notificationExtension: 通知扩展字段，进入聊天室通知开发者扩展字段
    // enterParams.tagConfig: 进入聊天室标签信息配置
    // enterParams.locationConfig: 进入聊天室空间位置信息配置
```

```
    // enterParams.antispamConfig: 用户资料反垃圾检测配置

    V2NIMChatroomClient *chatroomClient = [V2NIMChatroomClient getInstance:instanceId];
    [chatroomClient enter:roomId
            enterParams:enterParams
               success:^(V2NIMChatroomEnterResult *result)
               {
                  //进入成功
               }
               failure:^(V2NIMError *error)
               {
                     //进入失败
               }];
    }

    - (nullable NSArray<NSString *> *)getLinkAddress:(NSString *)roomId
                                  accountId:(NSString *)accountId
    {
        return @[@"聊天室 Link 地址"];
    }
    @end
```

macOS/Windows

```
V2NIMChatroomEnterParams enterParams;
    enterParams.accountId = "accountId";
    enterParams.token = "token";
    enterParams.roomNick = "nick";
    enterParams.roomAvatar = "avatar";
    enterParams.linkProvider = [](nstd::string roomId, nstd::string account) {
        nstd::vector<nstd::string> linkAddresses;
        // get link addresses
        // ...
        return linkAddresses;
    };
    enterParams.serverExtension = "server extension";
    enterParams.notificationExtension = "notification extension";
    chatroomClient.enter(
        "roomId",
        enterParams,
        [](V2NIMChatroomEnterResult result) {
            // enter succeeded
        },
        [](V2NIMError error) {
            // enter failed, handle error
        });
```

Web/uni-app/小程序

```
try {
        const chatroom = V2NIMChatroomClient.newInstance({
        appkey: 'YOUR_APPKEY',
        debugLevel: 'debug'
        })
        await chatroom.enter('YOUR_ROOM_ID', {
            accountId: 'YOUR_ACCOUNT_ID',
            token: 'YOUR_TOKEN'
```

```
        })
    } catch (err) {
        // TODO failed, check code
        // console.log(err.code)
    }
```

Node.js/Electron

```
const result = await chatroomClient.enter('your room id', {
        accountId: 'your account id',
        token: 'your token',
        roomNick: 'your room nick',
        linkProvider: (roomId, account) => {
            return ['chatroom link...']
        }
    })
    if (result) {
        console.error(result)
    }
```

鸿蒙

```
try {
        const chatroom = V2NIMChatroomClient.newInstance({
        appkey: 'YOUR_APPKEY',
        debugLevel: 'debug'
        })
        await chatroom.enter('YOUR_ROOM_ID', {
            accountId: 'YOUR_ACCOUNT_ID',
            token: 'YOUR_TOKEN'
        })
    } catch (err) {
        // TODO failed, check code
        // console.log(err.code)
    }
```

5. 登录聊天室成功后，调用 getChatroomService 方法获取聊天室服务。后续聊天室相关操作（聊天室成员、消息等）均在返回的 Service 类中实现。

   Web/uni-app/小程序可跳过此步骤。

   示例代码如下：

   安卓

```
V2NIMChatroomService chatroomService = chatroomClient.getService(ChatroomService.class)
```

   iOS

```
[[V2NIMChatroomClient getInstance:instanceId] getChatroomService];
```

   macOS/Windows

```
auto& chatroomService = client.getChatroomService();
```

   Node.js/Electron

```
const chatroomService = chatroomClient.getChatroomService()
```

   鸿蒙

```
const client: V2NIMChatroomClient = this.getInstance(instanceId)
    const ret = client.chatroomService
```

3.7.3 登录圈组

NIM SDK V10 已采用融合登录方案，用户只需要调用 login 方法登录一次，则可以同时使用 IM 与圈组，**无需再单独登录圈组服务器**。具体示例代码请参考 登录 IM。

note notice 调用 login 后调用 QChatService#login 方法将会报错。

3.8 第四步：收发消息

3.8.1 单聊/群聊收发消息

本节以发送发和接收方的消息交互为例，介绍快速实现单聊收发消息的流程。更多消息类型的收发，请参考 收发消息。

在 创建或加入群组 后，用户发送和接收消息的接口与单聊收发消息相同，区别在于会话类型的参数配置，TEAM 为高级群，SUPER_TEAM 为超大群。

1. 接收方注册消息监听器，监听消息接收回调事件 onReceiveMessages。示例代码如下：

安卓

```
V2NIMMessageService v2MessageService = NIMClient.getService(V2NIMMessageService.class);

    V2NIMMessageListener messageListener = new V2NIMMessageListener() {

        @Override
        public void onReceiveMessages(List<V2NIMMessage> messages) {

        }
    };
    v2MessageService.addMessageListener(messageListener);
```

iOS

```
[[[NIMSDK sharedSDK] v2MessageService] addMessageListener:listener];
```

macOS/Windows

```
V2NIMMessageListener listener;
    listener.onReceiveMessages = [](nstd::vector<V2NIMMessage> messages) {
        // receive messages
    };
    messageService.addMessageListener(listener);
```

Web/uni-app/小程序

```
nim.V2NIMMessageService.on("onReceiveMessages", function (messages: V2NIMMessage[]) {})
```

Node.js/Electron

```
v2.messageService.on("receiveMessages", function (messages: V2NIMMessage[]) {})
```

鸿蒙

```
nim.messageService.on("onReceiveMessages", function (messages: V2NIMMessage[]) {})
```

Flutter

```
subsriptions.add(
        NimCore.instance.messageService.onReceiveMessages.listen((event) {
    //do something
    }));
```

2. 发送方调用 createTextMessage 方法构建文本消息，然后调用 sendMessage 方法向接收方发送文本消息。示例代码如下：

安卓

```
V2NIMMessageService v2MessageService = NIMClient.getService(V2NIMMessageService.class);
    // 创建一条文本消息
    V2NIMMessage v2Message = V2NIMMessageCreator.createTextMessage("xxx");
    // 以单聊类型为例
    String conversationId = V2NIMConversationIdUtil.conversationId("xxx", V2NIMConversationType.
    V2NIM_CONVERSATION_TYPE_P2P);
    // 根据实际情况配置
    V2NIMMessageAntispamConfig antispamConfig = V2NIMMessageAntispamConfig.V2NIMMessag
    eAntispamConfigBuilder.builder()
    .withAntispamBusinessId()
    .withAntispamCheating()
    .withAntispamCustomMessage()
    .withAntispamEnabled()
    .withAntispamExtension()
    .build();

    // 根据实际情况配置
    V2NIMMessageConfig messageConfig = V2NIMMessageConfig.V2NIMMessageConfigBuilder.bui
    lder()
    .withLastMessageUpdateEnabled()
    .withHistoryEnabled()
    .withOfflineEnabled()
    .withOnlineSyncEnabled()
    .withReadReceiptEnabled()
    .withRoamingEnabled()
    .withUnreadEnabled()
    .build();
    // 根据实际情况配置
    V2NIMMessagePushConfig pushConfig = V2NIMMessagePushConfig.V2NIMMessagePushConfig
    Builder.builder()
    .withContent()
    .withForcePush()
    .withForcePushAccountIds()
    .withForcePushContent()
    .withPayload()
    .withPushEnabled()
    .withPushNickEnabled()
    .build();
    // 根据实际情况配置
    V2NIMMessageRobotConfig robotConfig = V2NIMMessageRobotConfig.V2NIMMessageRobotCo
    nfigBuilder.builder()
    .withAccountId()
    .withCustomContent()
```

```
    .withFunction()
    .withTopic()
    .build();
    // 根据实际情况配置
    V2NIMMessageRouteConfig routeConfig = V2NIMMessageRouteConfig.V2NIMMessageRouteCon
    figBuilder.builder()
    .withRouteEnabled()
    .withRouteEnvironment()
    .build();
    // 根据实际情况配置
    V2NIMSendMessageParams sendMessageParams = V2NIMSendMessageParams.V2NIMSendMes
    sageParamsBuilder.builder()
    .withAntispamConfig(antispamConfig)
    .withClientAntispamEnabled()
    .withClientAntispamReplace()
    .withMessageConfig(messageConfig)
    .withPushConfig(pushConfig)
    .withRobotConfig(robotConfig)
    .withRouteConfig(routeConfig)
    .build();
    // 发送消息
    v2MessageService.sendMessage(v2Message, conversationId, sendMessageParams,
    new V2NIMSuccessCallback<V2NIMSendMessageResult>() {
        @Override
        public void onSuccess(V2NIMSendMessageResult v2NIMSendMessageResult) {
            // TODO: 发送成功
        },
        new V2NIMFailureCallback() {
            @Override
            public void onFailure(V2NIMError error) {
                // TODO: 发送失败
            }
        });
    }
```

iOS

```
// 创建一条文本消息
    V2NIMMessage *message = [V2NIMMessageCreator createTextMessage:@"v2 message"];
    V2NIMSendMessageParams *params = [[V2NIMSendMessageParams alloc] init];
    // 发送消息
    [[[NIMSDK sharedSDK] v2MessageService] sendMessage:message
                                        conversationId:@"conversationId"
                                                params:params
                                               success:^(V2NIMSendMessageResult * _Nonull result) {
                                                   // 发送成功回调
                                                   }
                                               failure:^(V2NIMError * _Nonnull error) {
                                                   // 发送失败回调，error 包含错误原因
                                                   }
    }];
```

macOS/Windows

```
// 以单聊类型为例
    auto conversationId = V2NIMConversationIdUtil::p2pConversationId("target_account_id");
```

```
    // 创建一条文本消息
    auto message = V2NIMMessageCreator::createTextMessage("hello world");
    auto params = V2NIMSendMessageParams();
    // 发送消息
    messageService.sendMessage(
        message,
        conversationId,
        params,
        [](V2NIMSendMessageResult result) {
            // send message succeeded
        },
        [](V2NIMError error) {
            // send message failed, handle error
        });
```

Web/uni-app/小程序

```
try {
    // 创建一条文本消息
    const message: V2NIMMessage = nim.V2NIMMessageCreator.createTextMessage("hello")
    // 发送消息
    const res: V2NIMSendMessageResult = await nim.V2NIMMessageService.sendMessage(messag
    e, 'test1|1|test2')
    // Update UI with success message.
    } catch (err) {
    // todo error
    }
```

Node.js/Electron

```
const message = v2.messageCreator.createTextMessage('Hello NTES IM')
    const result = await v2.messageService.sendMessage(message, conversationId, params, progres
    sCallback)
```

鸿蒙

```
try {
    // 创建一条文本消息
    const message: V2NIMMessage = nim.messageCreator.createTextMessage("hello")
    // 发送消息
    const res: V2NIMSendMessageResult = await nim.messageService.sendMessage(message, 'test
    1|1|test2')
    // todo Success
    } catch (err) {
    // todo error
    }
```

Flutter

```
await MessageCreator.createTextMessage(text);
    await NimCore.instance.messageService.sendMessage(message, conversationId, params);
```

目前 NIM SDK 支持多种消息类型，包括文本消息、图片消息、语音消息、视频消息、文件消息、地理位置消息、提示消息、通知消息以及自定义消息。具体请参考 收发消息。

3. 接收方通过 onReceiveMessages 回调收到文本消息。

3.8.2 聊天室收发消息

本节以发送方与接收方的消息交互为例，介绍通过 NIM SDK 快速实现聊天室收发消息的流程。

note note 其他类型收发消息相关详情，请参考 聊天室消息管理。

1. 接收方注册聊天室监听器，监听聊天室消息接收回调事件 onReceiveMessages。

   示例代码如下：

   安卓

```
V2NIMChatroomClient v2ChatroomClient = V2NIMChatroomClient.getInstance(instanceId);
    V2NIMChatroomService v2ChatroomService = v2ChatroomClient.getChatroomService();

    V2NIMChatroomListener listener = new V2NIMChatroomListener() {
        @Override
        public void onReceiveMessages(List<V2NIMChatroomMessage> messages) {

        }
    };

    v2ChatroomService.addChatroomListener(listener);
```

   iOS

```
@interface Listener: NSObject<V2NIMChatroomListener>
    - (void)addToService;
    @end

    @implementation Listener

    - (void)addToService
    {
        id <V2NIMChatroomService> service = [[V2NIMChatroomClient getInstance:1] getChatroomServi
    ce];
        [service addChatroomListener:self];
    }

    - (void)onReceiveMessages:(NSArray *)messages
    {

    }
    @end
```

   macOS/Windows

```
V2NIMChatroomListener listener;
    listener.onReceiveMessages = [](nstd::vector<V2NIMChatroomMessage> messages) {
        // handle receive messages
    };
    chatroomService.addChatroomListener(listener);
```

   Web/uni-app/小程序

```
chatroom.V2NIMChatroomService.on('onReceiveMessages', function (messages: V2NIMChatroomMe
    ssage[]){})
```

Node.js/Electron

```
chatroom.chatroomService.on('receiveMessages', function (messages: V2NIMChatroomMessage[]){})
```

鸿蒙

```
chatroom.chatroomService.on('onReceiveMessages', (messages: V2NIMChatroomMessage[]) => {})
```

2. 发送方调用 createTextMessage 方法，构建一条文本消息。并调用 sendMessage 方法，发送已构建的文本消息。

   示例代码如下：

   安卓

```
// 新建一个聊天室实例，注意：每次 newInstance 都会返回一个新的实例，实际使用中请一个聊天室对应一个 V2NIMChatroomClient 实例，使用中需要临时缓存
V2NIMChatroomClient v2ChatroomClient = V2NIMChatroomClient.newInstance();
// 获取聊天室服务
V2NIMChatroomService v2ChatroomService = v2ChatroomClient.getChatroomService();
// 创建一条文本消息
V2NIMChatroomMessage v2Message = V2NIMChatroomMessageCreator.createTextMessage("xxx");

V2NIMChatroomMessageConfig messageConfig = new V2NIMChatroomMessageConfig();
// 根据实际情况配置
// 设置是否需要在服务端保存历史消息，默认 true
// messageConfig.setHistoryEnabled(true);
// 设置是否是高优先级消息，默认 false
// messageConfig.setHighPriority(false);

V2NIMMessageRouteConfig routeConfig = V2NIMMessageRouteConfig.V2NIMMessageRouteConfigBuilder.builder()
// 根据实际情况配置
// .withRouteEnabled()
// .withRouteEnvironment()
.build();

V2NIMMessageAntispamConfig antispamConfig = V2NIMMessageAntispamConfig.V2NIMMessageAntispamConfigBuilder.builder()
// 根据实际情况配置
// .withAntispamBusinessId()
// .withAntispamCheating()
// .withAntispamCustomMessage()
// .withAntispamEnabled()
// .withAntispamExtension()
.build();

V2NIMSendChatroomMessageParams params = new V2NIMSendChatroomMessageParams();
// 设置消息相关配置
// params.setMessageConfig(messageConfig);
// 设置路由抄送相关配置
// params.setRouteConfig(routeConfig);
// 设置反垃圾相关配置
// params.setAntispamConfig(antispamConfig);
```

```
    // 是否开启本地反垃圾，默认 false
    // params.setClientAntispamEnabled(false);
    // 本地反垃圾的替换文本
    // params.setClientAntispamReplace("xxx");
    //设置聊天室定向消息接收方账号 ID 列表
    // params.setReceiverIds(receiverIds);
    // 设置消息的目标标签表达式
    // params.setNotifyTargetTags("xxx");
    // 设置位置信息
    // params.setLocationInfo(locationInfo);
    v2ChatroomService.sendMessage(v2Message,params,
    new V2NIMSuccessCallback<V2NIMSendChatroomMessageResult>() {
        @Override
        public void onSuccess(V2NIMSendChatroomMessageResult result) {
            // 发送成功
        }
    },
    new V2NIMFailureCallback() {
        @Override
        public void onFailure(V2NIMError error) {
            // 发送失败
        }
    },
    new V2NIMProgressCallback() {
        @Override
        public void onProgress(int progress) {
            // 发送进度
        }
    });
```

iOS

```
// 通过实例 ID 获取聊天室服务
    id <V2NIMChatroomService> service = [[V2NIMChatroomClient getInstance:instanceId] getChatroo
    mService];
    // 创建一条文本消息
    V2NIMChatroomMessage *message = [V2NIMChatroomMessageCreator createTextMessage:@"xx
    x"];
    V2NIMChatroomMessageConfig *messageConfig = [V2NIMChatroomMessageConfig new];
    // 根据实际情况配置
    // 设置是否需要在服务端保存历史消息，默认 true
    // messageConfig.historyEnabled = YES;
    // 设置是否是高优先级消息，默认 false
    // messageConfig.highPriority = NO;
    V2NIMMessageRouteConfig *routeConfig = [V2NIMMessageRouteConfig new];
    // 根据实际情况配置
    // routeConfig.routeEnabled
    // routeConfig.routeEnvironment
    V2NIMMessageAntispamConfig *antispamConfig = [V2NIMMessageAntispamConfig new];
    // 根据实际情况配置
    // antispamConfig.antispamBusinessId
    // antispamConfig.antispamCheating
    // antispamConfig.antispamCustomMessage
    // antispamConfig.antispamEnabled
    // antispamConfig.antispamExtension
```

```
V2NIMSendChatroomMessageParams *params = [V2NIMSendChatroomMessageParams new];
// 设置消息相关配置
// params.messageConfig = messageConfig;
// 设置路由抄送相关配置
// params.routeConfig = routeConfig;
// 设置反垃圾相关配置
// params.antispamConfig = antispamConfig;
// 是否开启本地反垃圾，默认 false
// params.clientAntispamEnabled = false;
// 本地反垃圾的替换文本
// params.clientAntispamReplace = @"xxx";
// 设置聊天室定向消息接收方账号 ID 列表
// params.receiverIds = receiverIds;
// 设置消息的目标标签表达式
// params.notifyTargetTags = @"xxx";
// 设置位置信息
// params.locationInfo = locationInfo;
[service sendMessage:message
            params:params
            success:^(V2NIMSendChatroomMessageResult *result)
            {
                // 发送成功
            }
            failure:^(V2NIMError *error)
            {
                // 发送失败
            }
            progress:^(NSUInteger progress)
            {
                // 上传进度
            }];
```

macOS/Windows

```
// 创建一条文本消息
auto message = V2NIMChatroomMessageCreator::createTextMessage("hello world");
auto params = V2NIMSendChatroomMessageParams();
// 发送消息
chatroomService.sendMessage(
    message,
    params,
    [](V2NIMSendChatroomMessageResult result) {
        // send message succeeded
    },
    [](V2NIMError error) {
        // send message failed, handle error
    },
    [](uint32_t progress) {
        // upload progress
    });
```

Web/uni-app/小程序

```
await chatroom.V2NIMChatroomService.sendMessage(
    message,
```

```
    // V2NIMSendChatroomMessageParams
    {
      locationInfo: {x: 0, y: 100, z: 0}
    },
    progress: (percentage) => {console.log('上传进度: ' + percentage)}
)
```

Node.js/Electron

```
const message = V2NIMChatroomMessageCreator.createTextMessage('Hello NTES IM')
await chatroomService.sendMessage(message, {})
```

鸿蒙

```
// 准备代发送的消息
const msg: V2NIMChatroomMessage = this.chatroomClient.messageCreator.createTextMessage
(text)
// 发送聊天室消息时的参数
const params: V2NIMSendChatroomMessageParams = {
// 配置参数，如
locationInfo: {x: 0, y: 100, z: 0}
}
// 发送进度回调，如上传附件时由该 cb 回调
const progressCb = (percentage: number) => {
this.messageSetProgress(imgMsg, percentage)
console.info(`onUploadProgress: ${JSON.stringify(percentage)}`)
}
// send
const msgRes: V2NIMSendChatroomMessageResult = await this.chatroomClient.chatroomService.
sendMessage(msg, params, progressCb)
```

3. 接收方通过 onReceiveMessages 回调收到聊天室消息。

3.8.3 圈组收发消息

圈组收发消息相关详情，请参考

- 安卓 圈组收发消息
- iOS 圈组收发消息
- macOS/Windows 圈组收发消息
- Web/uni-app/小程序 圈组收发消息
- Flutter 圈组收发消息

3.9下一步

为保障通信安全，如果您在调试环境中的使用的是网易云信控制台生成的测试用 IM 账号 和 token，请确保在后续的正式生产环境中，将其替换为通过 IM 新版服务端 API 生成的正式 IM 账号和 token。

3.10 相关接口

安卓/iOS/macOS/Windows

API	说明

API	说明
addMessageListener	注册消息相关监听器
addChatroomClientListener	注册聊天室实例监听器
addChatroomListener	注册聊天室监听器
V2NIMLoginService.login	登录 IM
newInstance	构造聊天室实例
getChatroomLinkAddress	获取聊天室连接地址
enter	进入聊天室
getChatroomService	获取聊天室服务
V2NIMChatroomService	聊天室服务类
V2NIMConversationType	会话类型
V2NIMMessageCreator.createTextMessage	创建一条文本消息
V2NIMMessageService.sendMessage	发送消息
V2NIMChatroomMessageCreator.createTextMessage	创建一条聊天室文本消息
V2NIMChatroomService.sendMessage	发送聊天室消息
QChatService#login	登录圈组（旧版本接口）

Web/uni-app/小程序/Node.js/Electron/鸿蒙

API	说明
V2NIMMessageService.on	注册消息相关监听器
V2NIMChatroomClient.on	注册聊天室实例监听器
V2NIMChatroomService.on	注册聊天室监听器
V2NIMLoginService.login	登录 IM
newInstance	构造聊天室实例
getChatroomLinkAddress	获取聊天室连接地址
enter	进入聊天室
getChatroomService	获取聊天室服务（除 Web 端）
V2NIMChatroomService	聊天室服务类
V2NIMConversationType	会话类型
V2NIMMessageCreator.createTextMessage	创建一条文本消息
V2NIMMessageService.sendMessage	发送消息
V2NIMChatroomMessageCreator.createTextMessage	创建一条聊天室文本消息
V2NIMChatroomService.sendMessage	发送聊天室消息
QChatService#login	登录圈组（旧版本接口）

API	说明
NIMMessageService.listen	注册消息相关监听器
NIMLoginService.login	登录 IM
NIMConversationType	会话类型
NIMMessageCreator.createTextMessage	创建一条文本消息
NIMMessageService.sendMessage	发送消息
QChatService#login	登录圈组（旧版本接口）

Flutter