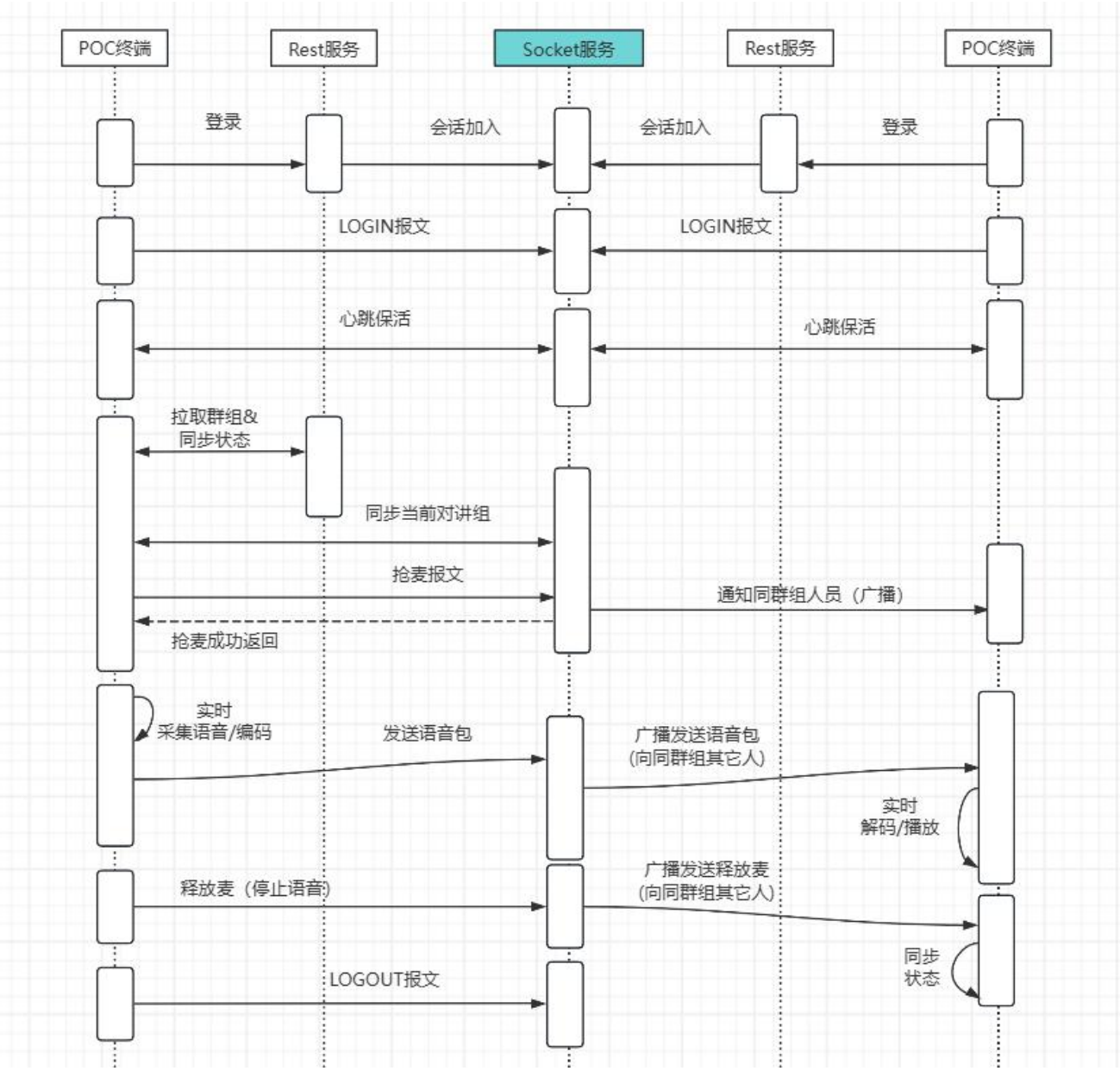
目 录

1、语音对讲业务时序图



流程说明

1）登录：指用帐号和密码登录 rest 服务器，获取 token,由 rest 服务器通知 socket 服务用户上线并进行会话管理

2）心跳保活：POC 终端定期向 Socket 服务器发送心跳报文

3）上传 LOGIN 报文到 Socket 服务器

4）拉取群组信息：登录成功后，要调用接口获取当前帐号参与的群组信息 （群组含固定群组和临时群组）

5）POC 端初始化完成后，要向 socket 服务器上报参与的当前群组 REPORT 报文信息

6）POC 端要对讲时，首先要向 socket 服务器发送抢麦报文，成功响应后才能开始采集语音

7）实时采集语音的基本流程
设置采样频率、声道和编码标准- > 采集语音- > 编码 -> 发送 socket 服务器

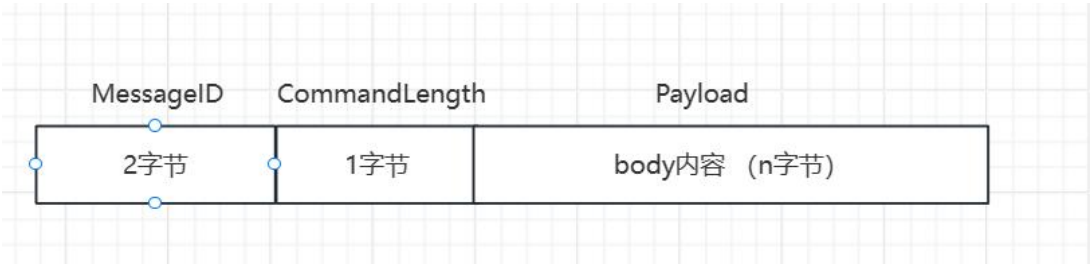
8）Socket 服务端管理了群组会话，当发送端的语音到来时，会向当前会话群组内的其它成员广播语音包

9）接收端实时播放语音的基本流程
监听 socket 接收消息 -> 根据 socket 消息 ID 解析出语音包头 -> 解码 -> 发送音箱（或听筒）播放

10）对讲语音结束：发送端释放麦克风后，向 socket 服务器发送“释放麦”报文，socket 服务器通知同群组其它人员

11）同群组其它人员同步本地状态，如麦空闲等，整个语音对讲流程结束

2、 Socket 报文编码格式

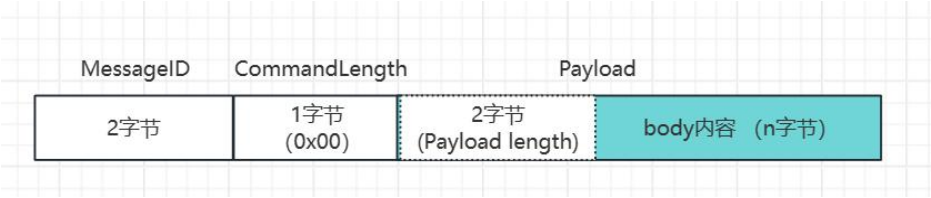


报文没有采用魔幻报文头、报文尾和校验码(客户可以在 socket 报文编解码类中自行扩展)，较简单易懂

由三大部分组成

（一）**MessageID**：消息 ID，short 类型整数，占 2 个字节，用来表示报文的消息类型，附表有 messageId 的一览表

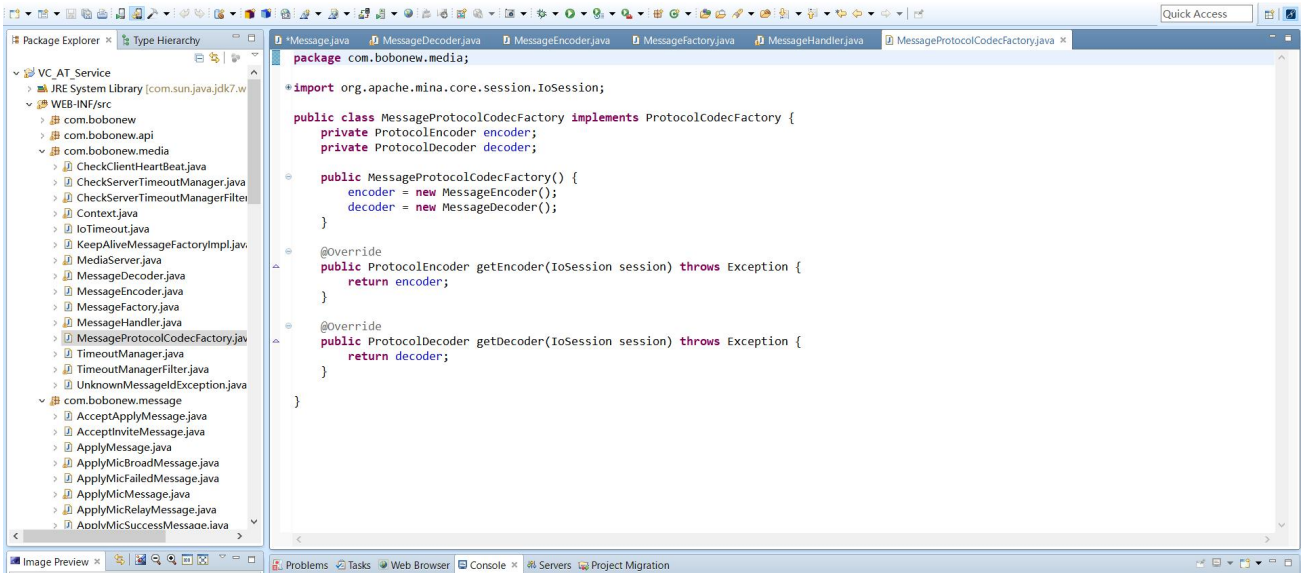
（二）**CommandLength**：当消息是属于某些信令类型时，即 Payload 负载部分内容不到 255 个字节时，由该字节表示 Payload 中 byte 数组的长度，如果 Payload 负载部分内容大于 255 个字节时，该字段为 0，则 Payload 部分字节长度由前面 2 位组成，如下图

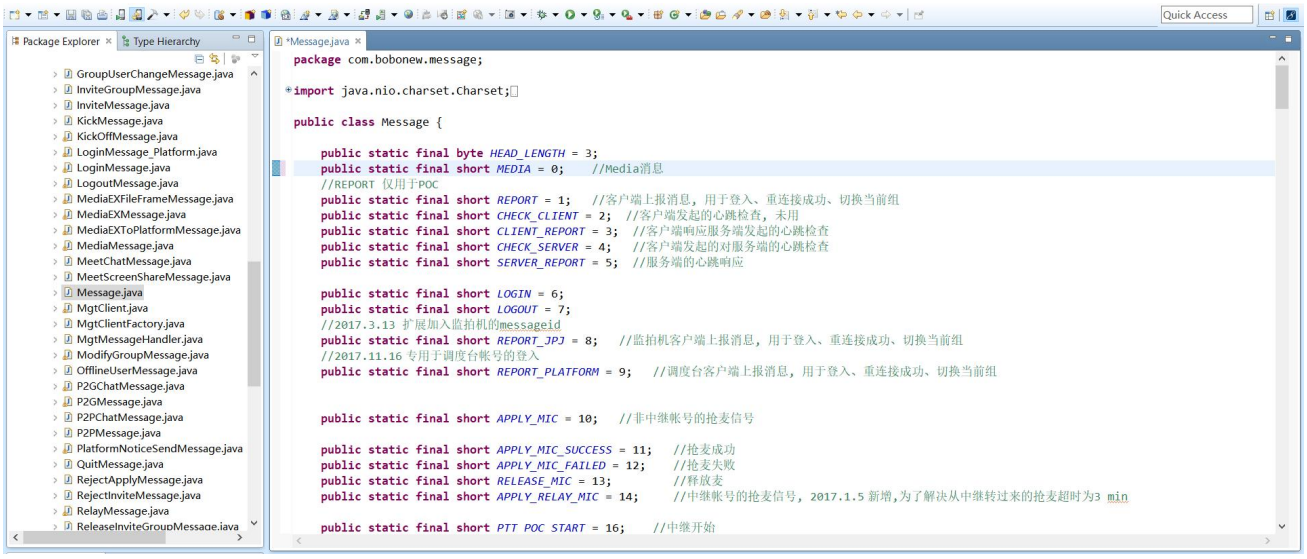


（三） Payload：是消息的有效负载部分，一般由业务消息决定，业务消息一般分为：*心跳消息、信令消息、状态报文消息、语音包消息*等，其包含的内容和编码因业务消息类型而异。

3、 Socket 报文消息

所有的消息报文，不管哪种类型，都按报文格式统一编解码，Socket 服务器采用 Java netty/mina 框架开发，适用于高并发、高性能的场景，所有的消息编解码易扩展易定制



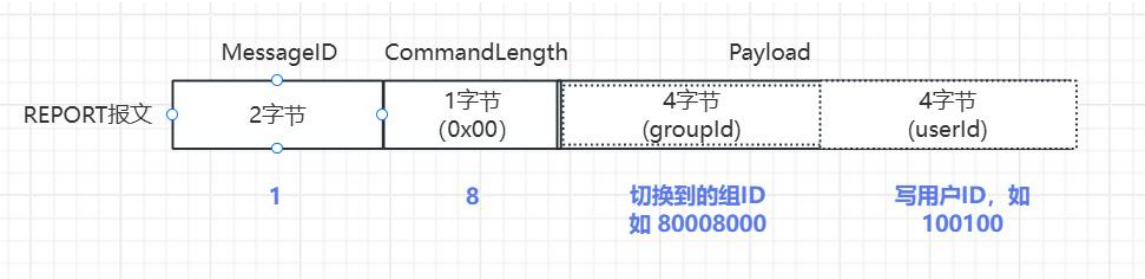


1. REPORT 报文【messageId=1】

目的： 表示指定用户切换到某个组（固定组或临时组）,注：仅终端用

建议用时： 在终端切换到某个组后，要立即向 Socket 服务上报该报文

报文格式：



2. REPROT_PLATFORM 报文【messageId=9】

目的： 表示指定用户切换到某个组（固定组或临时组）,注：仅调度台用

建议用时： 在调度台切换到某个组后，要立即向 Socket 服务上报该报文

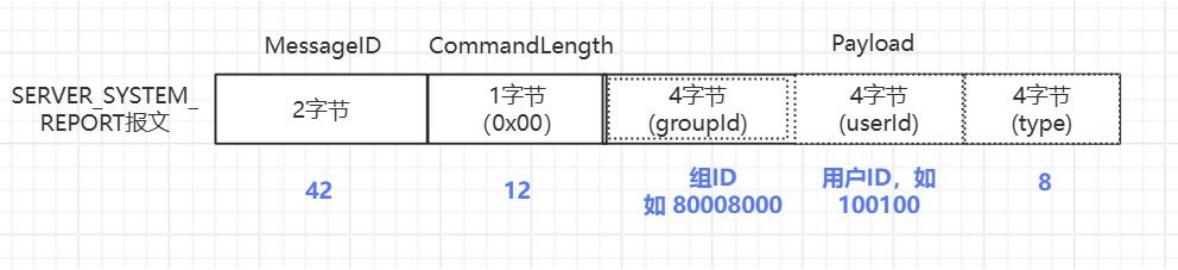
报文格式： **同【REPORT】报文**

3. SERVER_SYSTEM_REPORT【messageId=42】

目的： 表示用户某种状态，具体状态由业务枚举而定，一般用于服务端向终端广播其它终端的状态，从而达到终端之间的状态的感知和同步（如上线、下线、讲话中、麦闲等状态）

建议用时： 该报文一般是由其它报文服务端处理业务逻辑时而触发生成（例如：张三登录了，服务端将发送此报文通知其它同组用户），然后由服务端发给终端或调度台。

报文格式：



Payload 解释：

groupId: 表示某个组 ID，终端或调度台收到时，明确知道是哪个组

userId: 表示某个用户 ID

type: 如下表

Type 命称	int 值	用途	备注
SYS_MSSAGE_TALK_START	**1**	某用户开始说话	
SYS_MSSAGE_TALK_STOP	**2**	某用户停止说话	
SYS_MSSAGE_IN_GROUP	**3**	某用户进入某群组	**实际不用这个状态**

SYS_MSSAGE_OUT_GROUP	**4**	某用户离开某群组	
SYS_MSSAGE_REJECT_INVITE	**5**	某用户拒绝邀请	
SYS_MSSAGE_ENTER_PRESON	**6**	某用户同意单聊邀请	
SYS_MSSAGE_EXIT_PRESON	**7**	某用户拒绝单聊邀请，	**删除方删除临时组，也是这个消息**
SYS_MSSAGE_ONLINE_PRESON	**8**	某用户上线	**同时表示用户进入某个组这个状态有双重含义，终端切换组时，服务端也转发此状态值**
SYS_MSSAGE_OFFLINE_PRESON	**9**	某用户掉线或主动离线	
SYS_MSSAGE_TALK_START_TOPOC	**10**	转 POC 某用户开始说话	
SYS_MSSAGE_TALK_STOP_TOPOC	**11**	转 POC 某用户停止说话	
SYS_MSSAGE_TALK_INCALL	**12**	呼叫中对方正在通话	
TYPE_TOPOC_START_MIC	**16**	申请中继台成功	**中继终端的指令，普通终端收不到**
TYPE_TOPOC_FAIL_MIC	**17**	申请中继台失败或释放中继台	**中继终端的指令，普通终端收不到**
TYPE_TOPOC_RELEASE_SUCCESS_MIC	**18**	申请中继台的麦成功	**中继终端的指令，普通终端收不到**
TYPE_TOPOC_RELEASE_FAIL_MIC	**19**	申请中继台失败或释放中继台	**中继终端的指令，普通终端收不到**

4. CHECK_SERVER【messageId=4】

目的： 用于终端上报心跳给服务端用，心跳默认间隔是 10s，超过 2 个心跳周期（20s）没有上报心跳，服务端会认为其下线了，并广播给其它用户

建议用时： 在终端上报 Report 报文后，可以启动定时上报该心跳消息给服务端

报文格式：

5. SERVER_REPORT【messageId=5】

目的： 客户上报心跳后，服务端会响应此报文，如果超过指定时间没有响应，客户端可以认为服务端忙或离线状态。客户端要作重试机制

建议用时： 监听到响应报文后，客户端可以同步服务在线状态

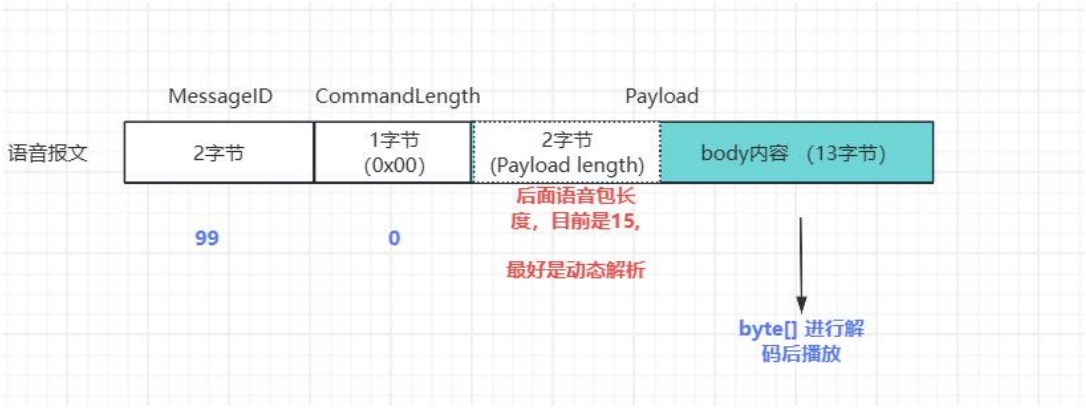
报文格式：

6. MEDIA_EX 【messageId=99】

目的： 实时对讲语音包，由服务端转发，向同组内其它用户发送语音包

建议用时：当讲话人获取到麦权时，由终端自动采集，按采样率(8000)和 amrnb 编码，每 20ms 发送一次语音包。

报文格式：

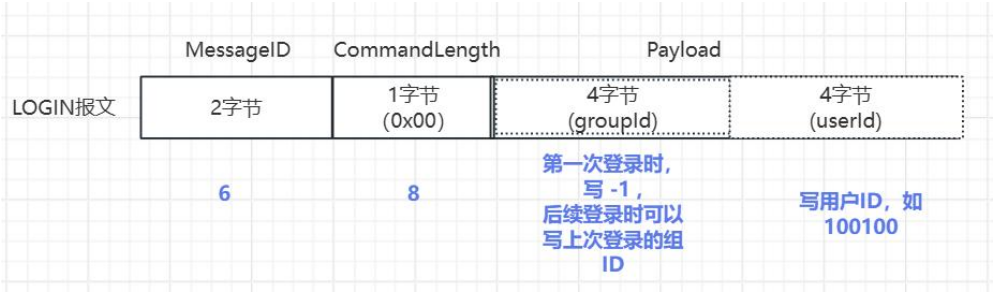


7. LOGIN 【messageId=6】

目的： 客户端登录报文，表示客户端登录的状态用，服务端需此报文管理终端的会话状态

建议用时： 一般用于，终端第一次登录时，或者终端没有加入任何固定组或临时组时发送用

报文格式：



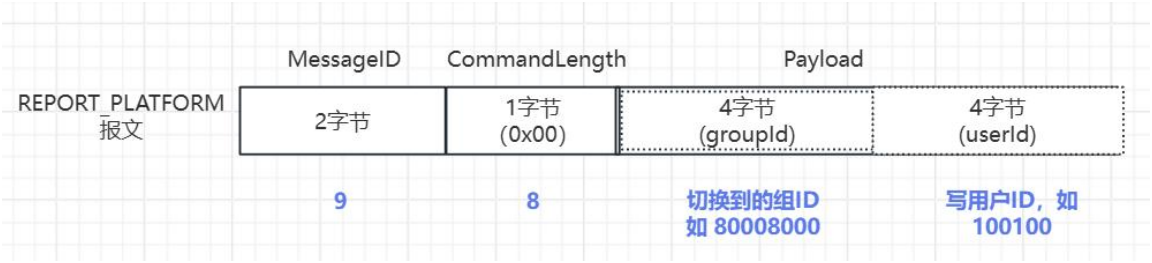
8. REPROT_PLATFORM 【messageId=9】

目的： 专用于调度端的状态报文，表示调度台所在组的状态用，服务端需此报文专门管理调度台的

会话和组状态

建议用时： 一般用于，调度台的第一次登录时和切换组时用

报文格式：



9. APPLY_MIC 【messageId=10】

目的： 专用于终端（非中继台）的抢麦报文，服务端以此识别终端在何组何人抢麦

建议用时： 一般用于终端（非中继台）用户要发送语音对讲先，先抢麦，等服务端确认权限后才能采集语音和发送

报文格式：

10. APPLY_MIC_SUCCESS 【messageId=11】

目的： 专用于服务端向终端（非中继台）用户返回抢麦成功报文

建议用时： 终端（非中继台）用户收到服务端响应的报文后才能采集语音和发送

报文格式：

11. APPLY_MIC_FAILED 【messageId=12】

目的： 专用于服务端向终端（非中继台）用户返回抢麦失败报文，失败原因可能是网络堵塞超时或者终端权限不够或者是组中有人正在讲话。

建议用时： 终端（非中继台）用户收到服务端响应的该报文后同步 UI 状态，表示抢麦失败

报文格式：

12. RELEASE_MIC 【messageId=13】

目的： 专用于终端（非中继台）用户释放麦报文，向服务器告之结束对讲。

建议用时： 终端（非中继台）用户要同步 UI 状态，表示对讲结束状态

报文格式：

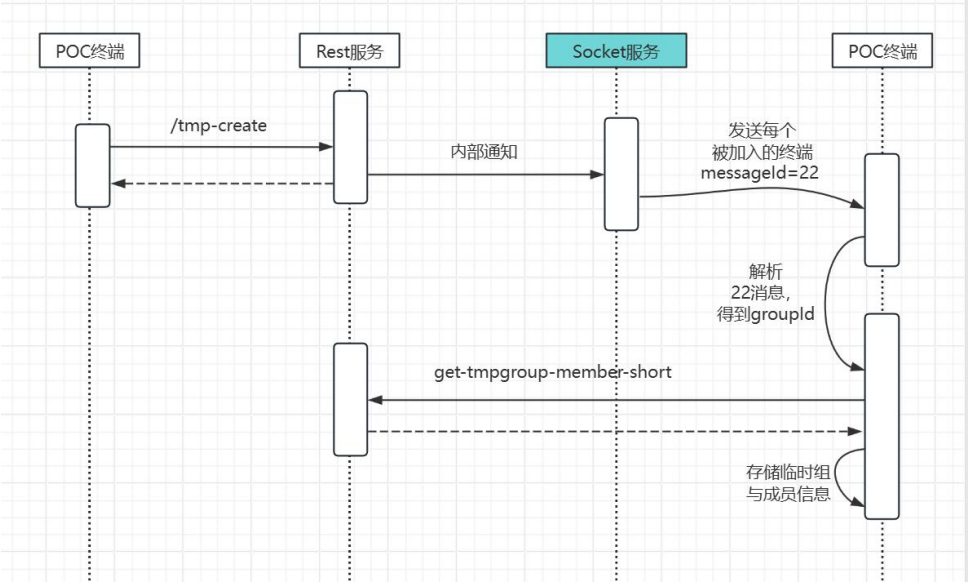
13. Todo.....

4、业务逻辑说明

1、创建临时组

用于创建临时组（多于 2 人）和单呼（2 人）的场景

时序图如下



说明：

1) 发起终端或调度台，调用 rest 的/tmp-create 接口，要求传入被拉入的终端 userId（也包括自己 userId）和临时组名称 （如果没有，服务端取时间戳）
2) rest 服务创建临时组成功后，返回发起终端的成功的临时组信息（包含 groupId 和成员）
3) rest 服务内部再异步通知 socket 服务，要求通知每个被拉的终端，messageId=22 的邀请消息
4) 被邀请终端收到 socket 22 消息后,解析出 groupId 后要求存储临时组信息
5) 被邀请终端再请求 rest 的 get-tmpgroup-member-short 接口
6) rest 返回结果后，被邀请终端存储临时组的成员信息

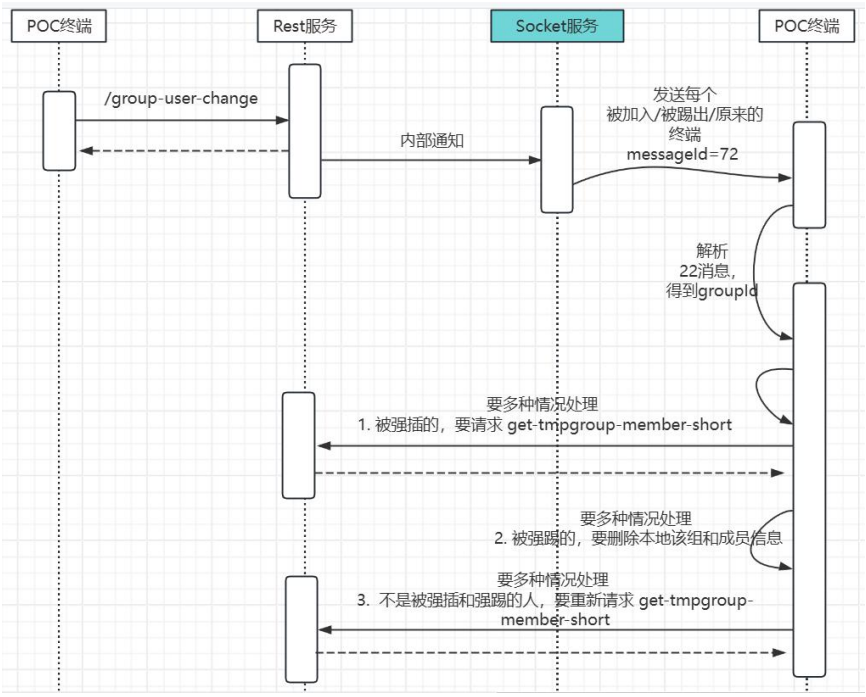
注意：建议终端至少要有 2 张表

组表（含有固定组和临时组） 组成员表（存储固定组或临时组的成员用）

2、临时组员变更（强插强踢）

可以用于临时组或固定组（多于 2 人）的强插和强踢的场景

时序图如下



说明：

1) 发起终端或调度台，调用 rest 的/group-user-change 接口，可以支持对临时组或固定组的人员强插和强踢操作，有具体参数

2) rest 服务处理组成员变更成功后，返回发起终端的 200 成功消息

3) rest 服务内部再异步通知 socket 服务，要求通知每个被强插/强踢/原来（没插和没踢）的终端，messageId=72 组内成员变更消息

4) 所有终端收到 72 消息要进行解析组 ID，组类型（固定组或临时组）、变更类型（强插或强踢）、变更成员 userId (多个会用 半角逗号 ，分隔)

5) 终端根据以下 3 种来处理

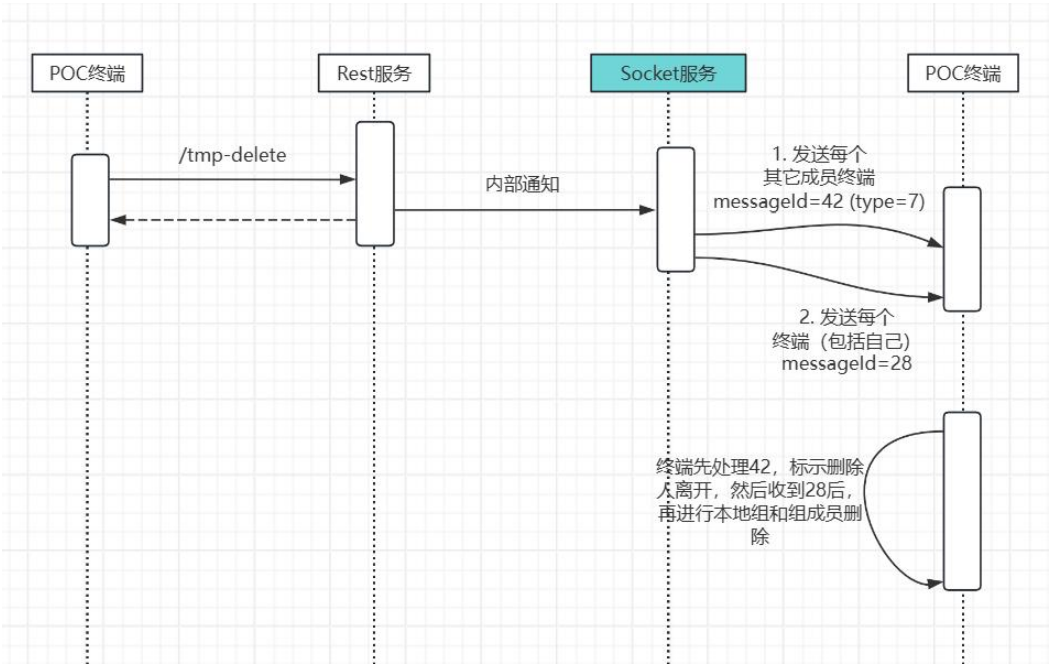
A、被强插的，要请求 tmp-query 和 get-tmpgroup-member-short 分别获取临时组的信息，返回的结果要分别存储本地，如果是固定组的被强插，要请求 /group/query 和 /query-group-member-short 分别获取固定组的信息，并分别存储本地

B、被强踢的，要求进行本地处理，分别删除组和组成员的信息

C、原来的不变，要重新请求，同 A 处理

3、临时组解散

可以用于临时组（多于 2 人）或单呼（2 人）解散的场景

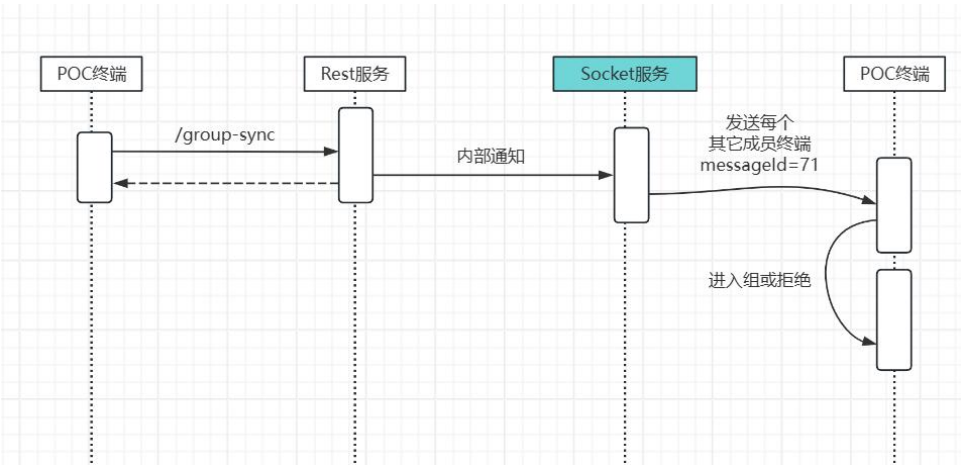


说明：

1) 发起终端或调度台，调用 rest 的/tmp-delete 接口，可以支持对临时组删除，固定组会禁止删除，临时组只有当初创建的人才有权限删除
2) rest 服务处理后，返回发起终端的 200 成功消息
3) rest 服务内部再异步通知 socket 服务， 会接连发送 42 和 28 两种指令给组内终端成员
4) 42 指令会被组内其它成员收到，其中消息包中 type=7，表示删除方离开组
5) 28 指令会被组内所有人收到，包括删除方，表示要求删除组和组成员

4、固定组或临时组同步

可以用于临时组（多于 2 人）或单呼（2 人）解散的场景



说明：

1) 发起终端或调度台，调用 rest 的/group-sync 接口，可以支持固定组或临时组的同步，一般用于向组内其它成员同步到当前组来

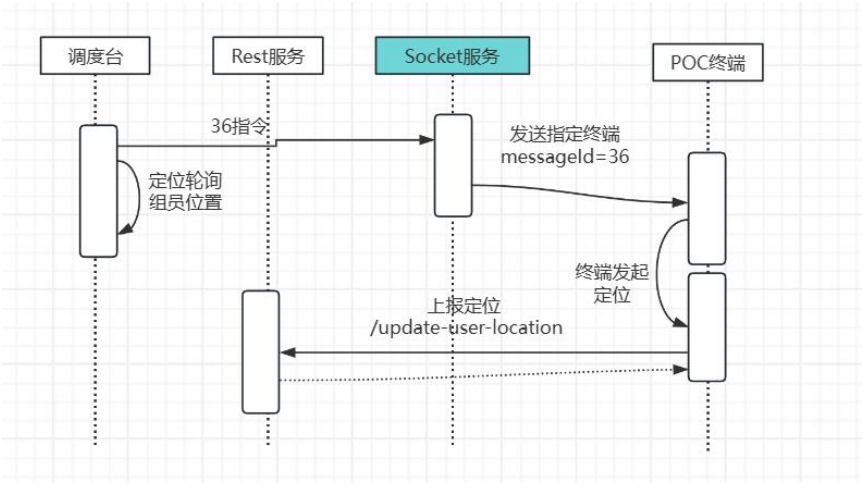
2) rest 服务收到请求，会返回 code=200 成功消息

3) rest 服务内部通知 socket 服务，由 socket 服务发 71 指令给组内其它成员

4) 终端收到 71 消息后，解析出 groupId，可以拒绝进入或进入指定组，进入指定组后要上报 Report 报文(messageId=1) 报告自己进入了该组

5、定位

目前是提供了 socket 指令 36 用于调度台对终端进行实时定位下达



说明：

1) 发起端为调度台，向 socket 服务发送 36 指令，可以指定成员下达定位请求

2) socket 服务向目标终端转发定位请求

3) 终端用户收到 36 指令后，进行终端定位

4) 终端定位完成后，再请求 rest 的/update-user-location 上报位置

5) 调度台的地图界面，有定时轮询组内成员的位置 (通过 rest 的 /location/queryByGroupId 接口)

5、42 状态消息涉及的业务

42 指令主要用于 socket 向终端发通知各种状态用的

目前用到的场景有：

序号	发起事件	发送目标	type 值	状态含义
1	服务检测到某终端心跳超时	会向终端最后一次 report 上报组的其它成员发送 42 通知	9	让同组内其它成员知道该终端离线了或主动退出了
2	终端断网被检测到 socket 连接中断了	同上	9	某种原因断开了
3	终端抢麦成功后，服务端下达通知	当前抢麦组内其它成员	1	表示当前组该终端要开始说话了
4	中继终端抢麦成功后(14 指令)	当前抢麦组内其它成员	1	表示组内的中继终端要开始说话了
5	终端发心跳(4 指	当前抢麦组内所有成员	8	通知组内成员终端上线了

	令)			
6	终端发 login(6 指令)	当前抢麦组内其它成员	8	通知组内成员该终端上线了
7	终端发 release_mic(13 指令)	当前抢麦组内其它成员	2	通知组内成员该终端停止讲话了
8	终端发 report(1 指令)	所进入组的其它成员	8	通知组内成员该终端进入了
9	终端发 logout(7 指令)主动退出应用	退出时组的其它成员	9	通知组内成员该终端退出了
10	终端说话超时，服务端主动掐断了	当前讲话组内的其它成员	2	通知组内成员该终端讲话结束了

6、麦权/麦权超时/麦权抢断业务

1）、麦权(也称话权)

目前系统麦权，共有 3 种：普通话权、优先话权、最高话权，
可以在人员管理时，进行麦权设定

话权优先级： 是指同组内，最高话权> 优先话权> 普通话权

A、当组内没有人说话，谁先抢到麦权时，谁就先说话，同等级麦权人无法再抢到麦

B、当组内低等级话权人在说话时，这时高权限人说话时，会立即打断 Ta 的话权

2）、麦权超时

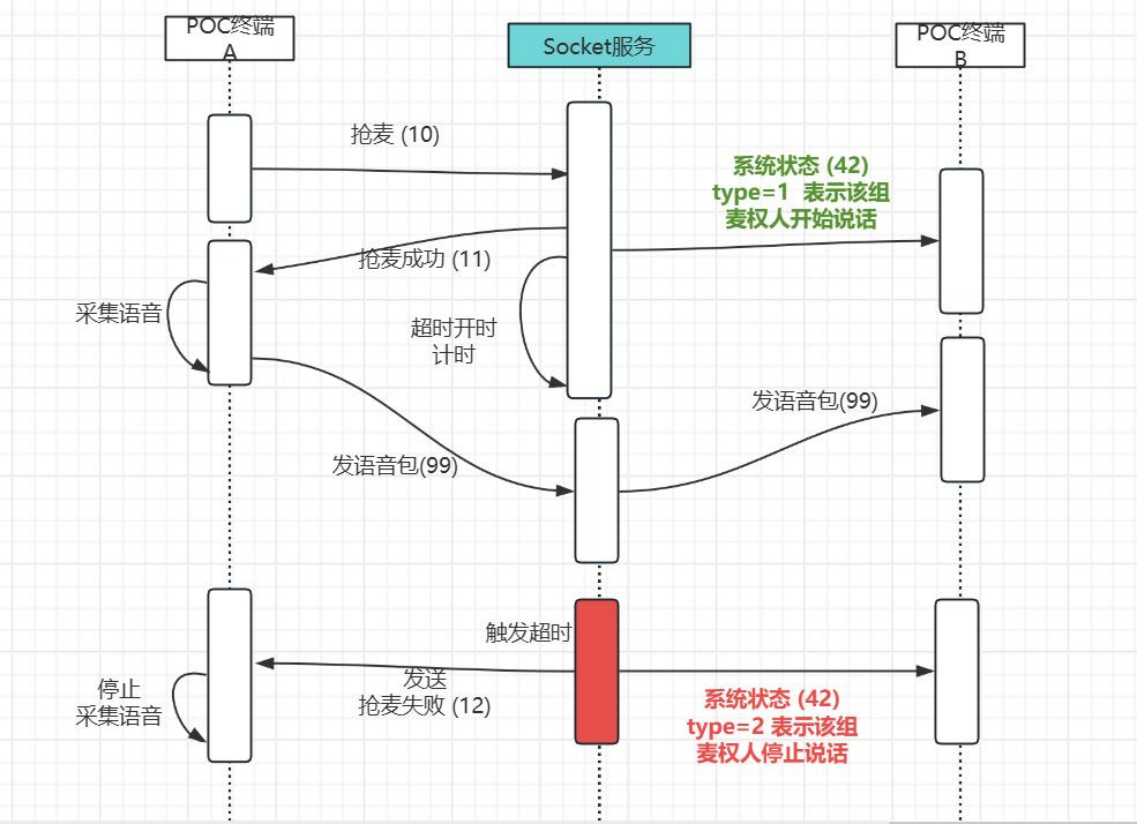
指某终端发出 10（抢麦指令）后，socket 服务经判断允许讲话后，服务端返回 11（抢麦成功指令）后，socket 服务开始计时，这个就叫做 “麦权超时”概念。

目前服务端的“麦权超时”概念一共有 3 种场景设定

场景	麦权超时时间	备注

普通对讲	60s （也可以根据具体场景来设定）	即调度台、普通终端参与的对讲，即一方获得抢麦成功后，允许讲话的时长
中继台讲话	60*3 s （也可以根据具体场景来设定）	一般用于中继台设备中转模拟对讲与数据对讲之间用
广播讲话	60*60 s （也可以根据具体场景来设定）	一般用于调度台发广播讲话时用

流程图：如下



说明：

1) POC 终端 A 发送抢麦(10)；

2) Socket 假设判断允许抢麦，返回抢麦成功(11), 并启动该终端麦权超时的计时，并且向该组其它成员发送系统状态消息（42），其中 type=1, 表示该组中的 userId（麦权人）开始说话；

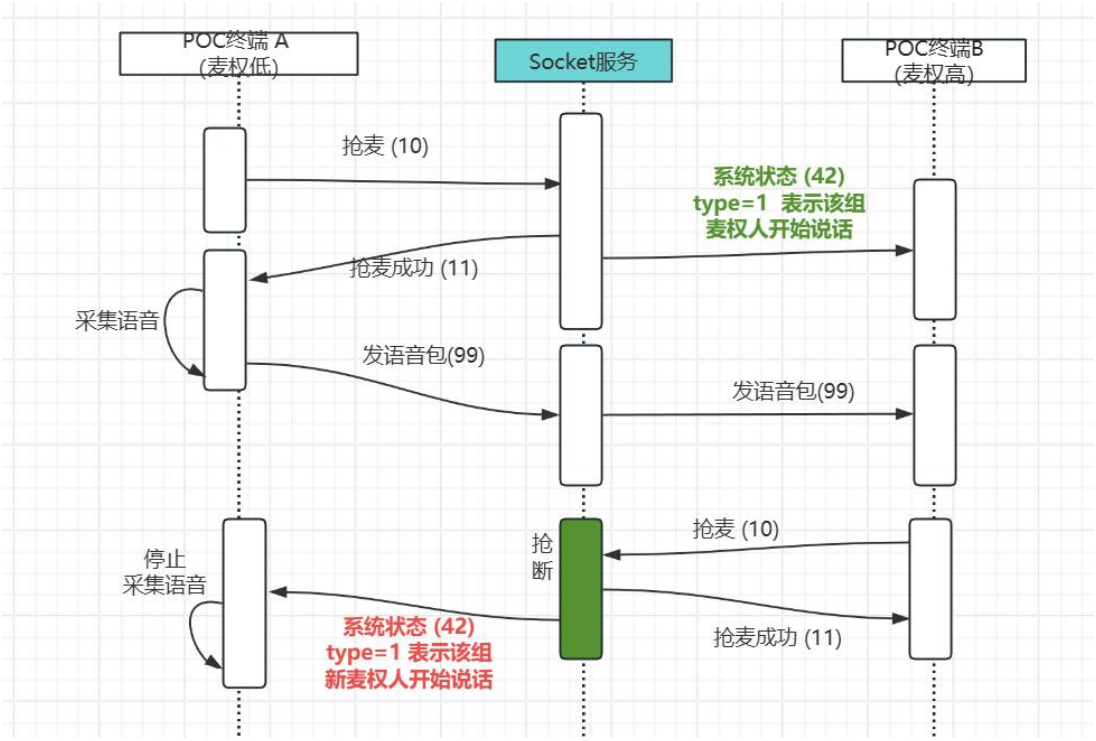
3) POC 终端 A 收到抢麦成功(11)消息后，开始采集语音，并发送语音包(99)；

4) Socket 服务收到 POC 终端 A 发来的语音包（99），并向同组内其它成员（例如 POC 终端 B）转发

5) 当 Socket 的麦权超时到来时，将向麦权人（假如这时麦权人还未释放麦）发送麦权失败（12）消息，同时向组内其它发送系统状态消息（42），其 type=2 表示该组麦权人停止说话

3）、麦权抢断

抢断是指高麦权在同组内打断低麦权人的讲话
流程：



6) POC 终端 A 发送抢麦(10)；

7) Socket 假设判断允许抢麦，返回抢麦成功(11)，并启动该终端麦权超时的计时，并且向该组其它成员发送系统状态消息（42），其中 type=1，表示该组中的 userId（麦权人）开始说话；

8) POC 终端 A 收到抢麦成功(11)消息后，开始采集语音，并发送语音包(99)；

9) Socket 服务收到 POC 终端 A 发来的语音包（99），并向同组内其它成员（例如 POC 终端 B）转发

10) POC 终端 B 的麦权高于 POC 终端 A，Ta 可以在前者讲话过程中抢麦

11) Socket 服务判断后，将麦权给到 POC 终端 B，并返回抢麦成功（11）

12) 同时 Socket 服务向其它人（包括先前麦权人 A）发送系统状态消息(42)，其它 type=1 表示新的麦权人开始讲话了

附表 1：<<MessageId 消息编号与消息类型一览表>>

MessageID 值(十进制)	消息类型	消息命名	Command Length	Payload 字段组成	备注
1	状态报文	*REPORT*	8	userId,groupId	客户端上报当前组的状态
42	状态报文	*SERVER_SYSTEM_REPORT*	12	userId,groupId,type	客户端上报自身状态，其中 type 字段表示具体状态枚举

4	心跳报文	*CHECK_SERVER*	0	空	客户端发起的对服务端的心跳检查
5	心跳报文	*SERVER_REPORT*	0	空	服务端的心跳响应
6	信令消息	*LOGIN*	8	userId,groupId	登录
7	信令消息	*LOGOUT*	8	userId,groupId	登出
9	信令消息	*REPROT_PLATFORM*	8	userId,groupId	客户端上报当前组的状态（调度台专用，**终端禁用**）
10	信令消息	*APPLY_MIC*	0	空	非中继帐号的抢麦信号
11	信令消息	*APPLY_MIC_SUCCESS*	0	空	抢麦成功
12	信令消息	*APPLY_MIC_FAILED*	0	空	抢麦失败
13	信令消息	*RELEASE_MIC*	0	空	释放麦
14	信令消息	*APPLY_RELAY_MIC*	0	空	中继帐号的抢麦信号
16	中继消息	*PTT_POC_START*	0	参考具体业务	中继开始，具体参考中继台业务说明
17	中继消息	*PTT_POC_STOP*	0	参考具体业务	中继结束，具体参考中继台业务说明
18	信令消息	*APPLY_BROAD_MIC*	0	空	调度台在广播模式下的抢麦专用，服务器将此类广播超时设为很长，因为广播的时长会长
20	信令消息	*KICK*	8	userId,groupId	踢人
21	信令消息	*QUIT*	8	userId,groupId	退出组
22	信令消息	*INVITE*	0	参考具体业务	邀请进对讲群
23	信令消息	*ACCEPT_INVITE*	0		接受邀请
24	信令消息	*REJECT_INVITE*	0		拒绝邀请
28	信令消息	*DELETE_GROUP*	4	groupId	解散组
29	信令消息	*CREATE_GROUP*	0	参考具体业务	创建组
36	信令消息	*GPS_COMMAND*	17	参考具体业务	调度台与 POC 之间 GPS 命令发送与接收
38	信令消息	*SOS_LOCATION*	0	参考具体业务	SOS 按下的定位报文
39	语音包消息	*SOS_MEDIA_EX*	0	参考具体业务	SOS 报文的语音数据包
43	信令消息	*INVITE_GROUP*	0	参考具体业务	某用户邀请单聊
44	信令消息	*INVITE_GROUP_RELEASE*	0	参考具体业务	某用户释放邀请单聊
45	信令消息	*KICK_OFF*	8	userId,groupId	同帐号的互相踢除，后登入的踢出

					前一个
46	信令消息	*CAMERA_SWITCH*	8	参考具体业务	远程监控时发出的前后摄像头切换
71	信令消息	*GROUP_SYNC*	0	参考具体业务	强制同步到指定组讲话
72	信令消息	*GROUP_USER_CHANGE*	0	参考具体业务	组内人员的变更报文，用于强插与强拆消息通知
75	信令消息	*AV_CHAT_NEW*	0	参考具体业务	一对一的音视频通话
76	信令消息	*AV_REMOTE_MONI*	0	参考具体业务	远程监控调取
99	语音包消息	*MEDIA_EX*	0	参考具体业务	
101	语音包消息	*MEDIA_EX_FILE_FRAME*	0	参考具体业务	广播音频帧的数据包
102	信令消息	*MEET_CHAT*	0	参考具体业务	视频会商

注：

1)上表中的 payload 中字段说明

userId: int32 类型整型，占 4 个字节，表示用户 ID

groupId: int32 类型整型，占 4 个字节，表示对讲组 ID

2) **中继消息:**

用于 POC 对讲与模拟对讲之间的中继转换用

3) 字节顺序

对于 short (2 个字节) int (4 个字节) long (8 个字节) 的整型类型，客户端与服务端统一用 ByteOrder.BIG_ENDIAN （大端字节顺序）

附表 2：<<MessageId 消息的 payload 结构及编码>>

1、messageId=1：report

payload 结构：

```
int userId; //用户 ID
int groupId; //组 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

2、messageId=42：server_system_report

payload 结构：

```
 int userId;   //用户 ID
int groupId;  //组 ID
int type;     //参考前文中的枚举说明
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
buffer.putInt(type);
```

3、messageId=4: check_server

payload 为空

4、messageId=5: server_report

payload 为空

5、messageId=6: login

payload 结构:

```
int groupId;  //登录组 ID
int userId;   //用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

6、messageId=7:logout

payload 结构:

```
int groupId;  //登出组 ID
int userId;   //用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

7、messageId=9: report_plateform (调度台专用上报进入组状态)

payload 结构:

```
int groupId;  //登录组 ID
int userId;   //用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

8、messageId=10：apply_mic

payload 为空

9、messageId=11:apply_mic_success

payload 为空

10、messageId=12：apply_mic_failed

payload 为空

11、messageId=13：release_mic

payload 为空

12、messageId=18: apply_broad_mic

payload 为空

注：该指令是调度台在广播模式下专用的抢麦指令，服务端用于识别将其超时加长，如 30 分钟

13、messageId=20: kick

payload 结构：

```
int groupId;  //登录组 ID
int userId;   //用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

注：已废弃指令，不建议用

14、messageId=21: quit

payload 结构：

```
int groupId;  //登录组 ID
int userId;   //用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

注：已废弃指令，不建议用，原先是作为某终端退出组，有点类似于 “微信用户退群”

15、messageId=22: invite

payload 结构：

```
int groupId;   //邀请的组 ID
byte groupNameLength;  //组名称长度,不超过 255 个字节
```

```
String groupName;  //邀请的组名称
int userId;     //被邀请的用户 ID
int inviteId;   //邀请者用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.put(groupNameLength);
buffer.put(groupName.getBytes(UTF_8));
buffer.putInt(userId);
buffer.putInt(inviteId);
```

16、messageId=23: accept_invite

payload 结构:

```
int groupId;   //接受邀请的组 ID
int userId;     //被邀请的用户 ID
byte userNameLength;  //接受者的姓名长度,不超过 255 个字节
String userName;  //接受者的姓名
int inviteId;   //邀请者用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
buffer.put(userNameLength);
buffer.put(userName.getBytes(Message.UTF_8));
buffer.putInt(inviteId);
```

注：已废弃指令，不建议用

17、messageId=24：reject_invite

payload 结构:

```
int groupId;   //接受邀请的组 ID
int userId;     //被邀请的用户 ID
byte userNameLength;  //接受者的姓名长度,不超过 255 个字节
String userName;  //接受者的姓名
int inviteId;   //邀请者用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
buffer.put(userNameLength);
buffer.put(userName.getBytes(Message.UTF_8));
buffer.putInt(inviteId);
```

注：已废弃指令，不建议用

18、messageId=28：delete_group

payload 结构:

```
int groupId;   //接受邀请的组 ID
```

编码

```
buffer.putInt(groupId);
```

19、**messageId=29：create_group**

payload 结构：

```
short msglen;  //payload 长度，要加上本身占的 2 个字节长度
int groupId;   //创建组 ID
int userId;    //创建者 ID
String userIds;   //被选中的用户 ID,多个用逗号分隔,要把自身创建者 ID 也包含在内
```

编码

```
buffer.putShort(msglen);
buffer.putInt(groupId);
buffer.putInt(userId);
buffer.put(userIds.getBytes(Message.UTF_8));
```

注：目前该指令，不能由终端直接调用

20、**messageId=36：gps_command**

payload 结构：

```
short    msglen;  //payload 长度
byte requestType;  //消息类型（0：请求  1：结果）
byte locationMode; //定位类型（0：自动 1：一般  2：高精）

int dispatcherId; //调度员 ID
int groupId; //当前群组 Id
int userId;  //请求的终端用户 ID
byte errorCode; //返回码（0：成功  1：终端 gps 权限未打开   2:数据库写入失败 ）
```

编码

```java
@Override
public IoBuffer toBuffer() {

    //msglen=17
    IoBuffer buffer = IoBuffer.allocate( HEAD_LENGTH + 17);
    buffer.putShort( Message.GPS_COMMAND );
    buffer.put( (byte)17 );

    buffer.putShort( (short) 17 );
    buffer.put(requestType);
    buffer.put(locationMode);
    buffer.putInt(dispatcherId);

    //groupId
    buffer.putInt(groupId);
    buffer.putInt(userId);
    buffer.put(errorCode);

    buffer.flip();
    logger.debug( "GPS_COMMAND:" + this.toString() );
    logger.debug( "GPS_COMMAND toBuffer:" + buffer.getHexDump() );

    return buffer;
```

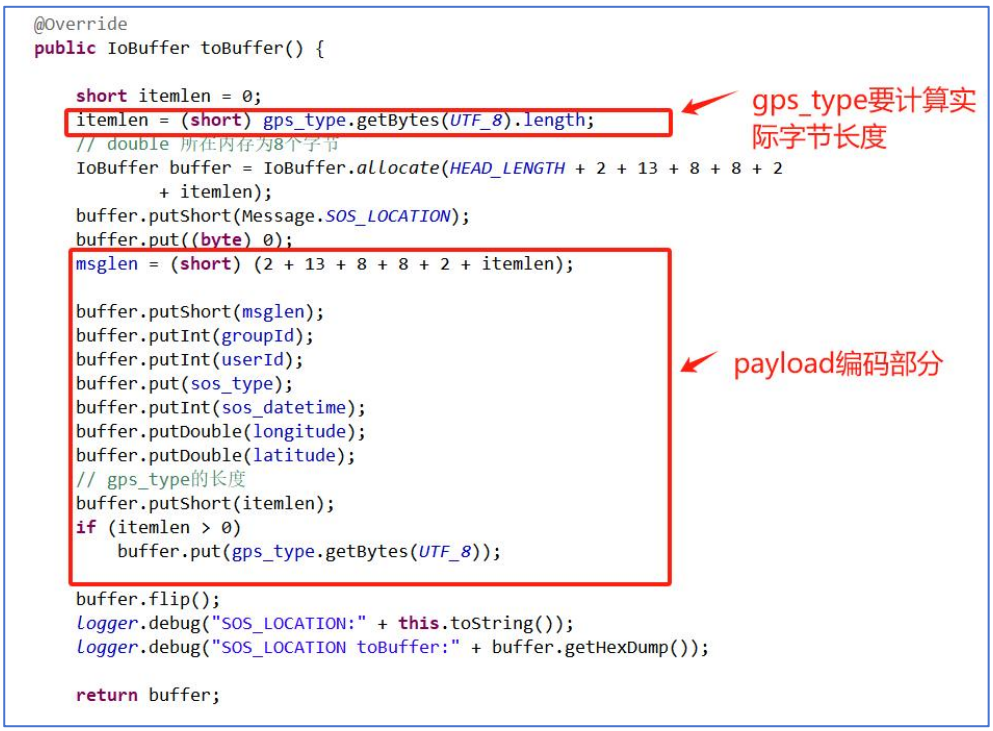
21、messageId=38: sos_location

payload 结构:

short msglen; //payload 长度
int groupId; //当前群组 Id,如果 sos_type=3 时，此处表示是调度台的 userid
int userId; //请求的终端用户 ID
byte sos_type; //SOS 类型，报警级别,1 表示 SOS，2 表示其它，3 表示是调度台发给 POC
int sos_datetime; // SOS 报警时间,秒数

double longitude; //经度
double latitude; //纬度
String gps_type; //定位类型: 如 baidu,gps,Google

编码



22、messageId=39: sos_media_ex

payload 结构：

```
short msgLength; //语音编码包字节数组的长度,即 payload 长度
byte[] media; //语音编码包字节数组

/**
 * 转发模式，0 表示转发给调度台,如果一个企业号内有多个调度台登入，即都转发， 1 表示转发到当前群组内所有人,除了自己
 */
byte transfer_mode;

/**
 * 表示发起端：0 表示 是 poc 端发起来的 sos 语音包， 1 表示的是调度端发起的 sos 语音包
 */
byte endpoint_type;

/**
 * sos 键按下时所在的当前群组
 */
int groupId;

/**
 * 用户帐号 ID， 1） 调度台发往 poc 的，userid 一定要写成目标用户的 userid
```

2) 是 poc 端发起来的 sos 语音包时,userid 为发起方的 userid
*/
int userId;

编码

```
@Override
public IoBuffer toBuffer() {
    IoBuffer buffer = IoBuffer.allocate(HEAD_LENGTH + 2 + 8 + 1 + 1
            + msgLength);

    buffer.putShort(this.getMessageId());
    buffer.put((byte) this.getLength());

    buffer.putShort((short) (msgLength + 2+ 8 + 1 + 1 ));
    buffer.put(media);
    // 包中后面加transfer_mode,groupId,userId
    buffer.put(this.endpoint_type);
    buffer.put(this.transfer_mode);
    buffer.putInt(this.getGroupId());
    buffer.putInt(this.getUserId());

    buffer.flip();
    return buffer;
}
```

实际写: buffer.put(0x00);

msgLength为media字节数组长度

经编码后的语音字节数组

23、messageId=43: invite_group

payload 结构:

```
int groupId;     //邀请组 ID
byte groupNameLength;  //组名称长度
String groupName;  //组名称
int userId;     //被邀请人 ID
int inviteId;  //邀请人 ID
int worktype;  //标志位
```

编码

```
buffer.putInt(groupId);
buffer.put(groupNameLength);
buffer.put(groupName.getBytes(UTF_8));
buffer.putInt(userId);
buffer.putInt(inviteId);
buffer.putInt(worktype);
```

注： 已废除指令

24、messageId=44: invite_group_release

payload 结构:

```
int groupId;     //邀请组 ID
```

```
byte groupNameLength;  //组名称长度
String groupName;  //组名称
int userId;    //被邀请人 ID
int inviteId;  //邀请人 ID
int worktype;  //标志位
```

编码

```
buffer.putInt(groupId);
buffer.put(groupNameLength);
buffer.put(groupName.getBytes(UTF_8));
buffer.putInt(userId);
buffer.putInt(inviteId);
buffer.putInt(worktype);
```

注： 已废除指令

25、messageId=45：kick_off

payload 结构：

```
int groupId;
int userId;
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

注： 该指令是，同帐号互踢用处，例如 A 帐号，已在一个终端上登录了，随后 A 帐号在其它终端或调度台上登录了，那么原来的终端上会收到 45 指令，你要登出整个应用并提示被别处终端登录了

26、messageId=46：camera_switch

payload 结构：

```
int groupId;
int userId;
```

编码

```
buffer.putInt(groupId);
buffer.putInt(userId);
```

注： 该指令是，调度台远程监控终端时，发出的前后摄像头切换用处，终端在收到后要处理切换摄像头后上传视频

27、messageId=71：group_sync

payload 结构：

```
int groupId;  //强制同步组 ID
byte groupNameLength;  //指 groupName 字段转成 utf-8 后字节数组的长度
String groupName;  //组名称
int userId;        //被同步的用户 ID
int inviteId;      //邀请同步的用户 ID
```

编码

```
buffer.putInt(groupId);
buffer.put(groupNameLength);
buffer.put(groupName.getBytes(UTF_8));
buffer.putInt(userId);
buffer.putInt(inviteId);
```

28、messageId=72：group_user_change

payload 结构：

```
int groupId;  //变更组的 ID

/**
 * 0: 表示固定组, 1:表示临时组
 */
int groupTypeId;

/**
 * 0: 表示强拆, 1: 表示强插
 */
int changeType;

/**
 * 整个消息的长度，即 payload 长度*/
short  msgLength;

/**
 * 目标用户 userid 的内容(多个 userid 以, (半角逗号)隔开)
 */
private String useridlist;
```

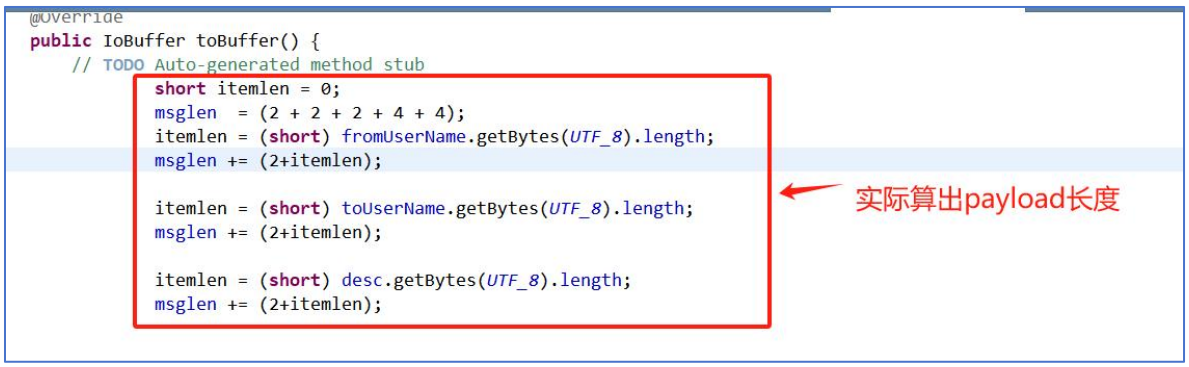
编码

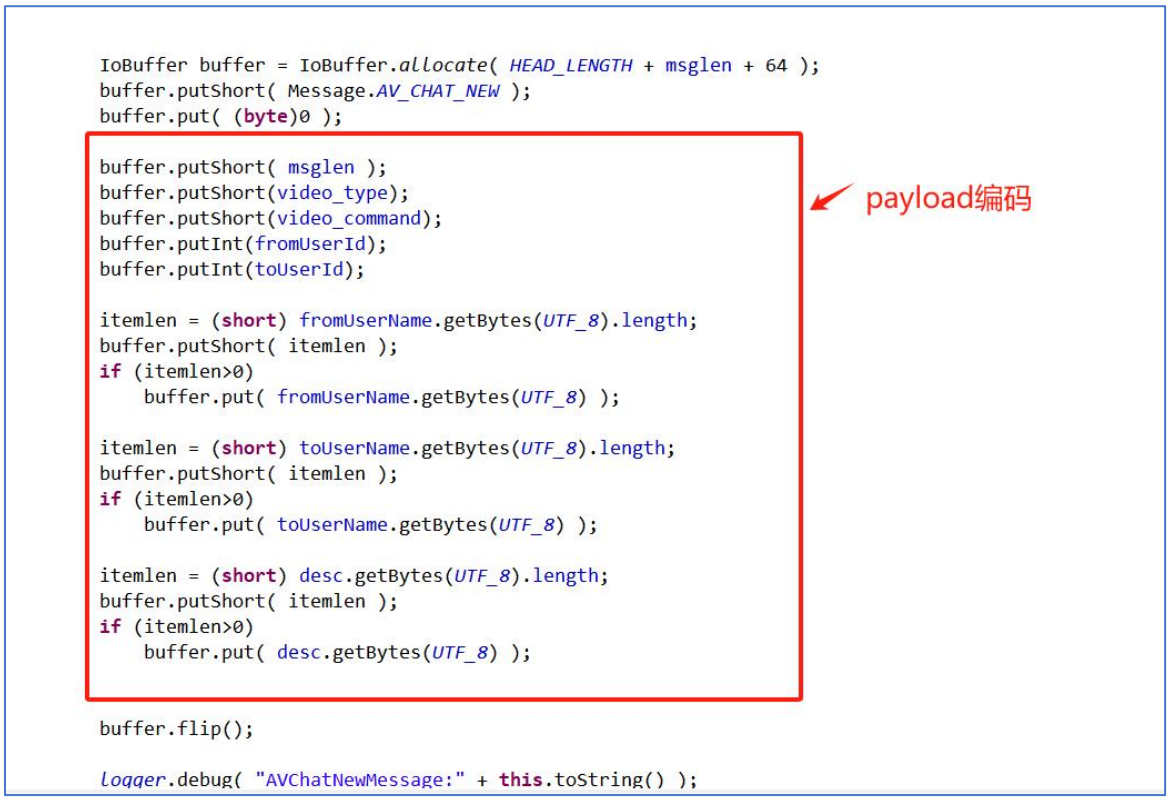
29、messageId=75：av_chat_new

payload 结构：

```
short     msglen;   //payload 长度
short     video_type;    //视频类型：1、语音通话  2、视频通话
short     video_command; //视频命令，  1：请求  2：应答  3：拒绝
int       fromUserId;    //发出音视频邀请方的 ID
int       toUserId;      //被邀请方的 ID
String    fromUserName;  //发出音视频邀请方的用户名称
String    toUserName;    //被邀请方的用户名称
String    desc;          //备注说明，可空
```

编码






```
IoBuffer buffer = IoBuffer.allocate( HEAD_LENGTH + msglen + 64 );
buffer.putShort( Message.AV_CHAT_NEW );
buffer.put( (byte)0 );

buffer.putShort( msglen );
buffer.putShort(video_type);
buffer.putShort(video_command);
buffer.putInt(fromUserId);
buffer.putInt(toUserId);

itemlen = (short) fromUserName.getBytes(UTF_8).length;
buffer.putShort( itemlen );
if (itemlen>0)
    buffer.put( fromUserName.getBytes(UTF_8) );

itemlen = (short) toUserName.getBytes(UTF_8).length;
buffer.putShort( itemlen );
if (itemlen>0)
    buffer.put( toUserName.getBytes(UTF_8) );

itemlen = (short) desc.getBytes(UTF_8).length;
buffer.putShort( itemlen );
if (itemlen>0)
    buffer.put( desc.getBytes(UTF_8) );

buffer.flip();

logger.debug( "AVChatNewMessage:" + this.toString() );
```

注：用于一对一的音视频通话

30、messageId=76：av_remote_moni

payload 结构：

```
short msglen;  //payload 长度
/**
 * 1: 开启监控    2: 结束监控
 */
byte commandType;
int fromUserId;   //发送调取监控方的用户 ID
String jsonStr;   //业务数据的 json 格式字符串,要看调度台与终端之间的业务约定内容
```

编码

```java
@Override
public IoBuffer toBuffer() {
    IoBuffer buffer = IoBuffer.allocate(HEAD_LENGTH + msglen);
    buffer.putShort(Message.AV_REMOTE_MONI);
    buffer.put((byte) 0);

    buffer.putShort(msglen);
    buffer.put(commandType);
    buffer.putInt(fromUserId);
    buffer.put(jsonStr.getBytes(UTF_8));
    buffer.flip();
    return buffer;

}
```

payload长度=
本身2+1+4+
jsonStr的实际长度

注：用于调度台远程静默方式调取终端的监控画面

31、messageId=99：media_ex

payload 结构：

short msgLength; //payload 长度，本身占 2 字符，实际=2+media 字节数组的长度

byte[] media; //经编码后的语音包字节数组

编码

```java
@Override
public IoBuffer toBuffer() {
    IoBuffer buffer = IoBuffer.allocate(HEAD_LENGTH +  msgLength);

    buffer.putShort(this.getMessageId());
    buffer.put((byte) 0);

    buffer.putShort((short) (msgLength));
    buffer.put(media);

    buffer.flip();
    return buffer;
}
```

payload编码

32、messageId=101：media_ex_file_frame

payload 结构：

short msgLength; //payload 长度，本身占 2 字符，实际=2+media 字节数组的长度
byte[] media; //经编码后的语音包字节数组

编码

```java
@Override
public IoBuffer toBuffer() {
    IoBuffer buffer = IoBuffer.allocate(HEAD_LENGTH +  msgLength);

    buffer.putShort(this.getMessageId());
    buffer.put((byte) 0);

    buffer.putShort((short) (msgLength));
    buffer.put(media);

    buffer.flip();
    return buffer;
}
```

payload编码

注：该指令本质上与 99 是一样的，只不过专用来标识是调度台发过来的广播用的，终端要识别出来并处理

33、messageId=102：meet_chat

payload 结构：

short msglen; //payload 长度
short meet_type; // 会议类型：1、纯语音会议 2、音视频会议(含音频和视频)
short meet_command; // 命令，1：请求 2：同意应答 3:拒绝应答 4:超时未应答 5：主动退出会议 6：召集方结束会议
short room_id; //会议室房间号

int fromUserId;
int toUserId;
String fromUserName;
String toUserName;
String desc;

编码

```java
@Override
public IoBuffer toBuffer() {

    short itemlen = 0;
    msglen = (2 + 2 + 2 + 4 + 4 + 2);
    itemlen = (short) fromUserName.getBytes(UTF_8).length;
    msglen += (2 + itemlen);

    itemlen = (short) toUserName.getBytes(UTF_8).length;
    msglen += (2 + itemlen);

    itemlen = (short) desc.getBytes(UTF_8).length;
    msglen += (2 + itemlen);
```

计算实际的payload长度

```
IoBuffer buffer = IoBuffer.allocate(HEAD_LENGTH + msglen);
buffer.putShort(Message.MEET_CHAT);
buffer.put((byte) 0);

buffer.putShort(msglen);
buffer.putShort(meet_type);
buffer.putShort(meet_command);
buffer.putShort(room_id);

buffer.putInt(fromUserId);
buffer.putInt(toUserId);

itemlen = (short) fromUserName.getBytes(UTF_8).length;
buffer.putShort(itemlen);
if (itemlen > 0)
    buffer.put(fromUserName.getBytes(UTF_8));

itemlen = (short) toUserName.getBytes(UTF_8).length;
buffer.putShort(itemlen);
if (itemlen > 0)
    buffer.put(toUserName.getBytes(UTF_8));

itemlen = (short) desc.getBytes(UTF_8).length;
buffer.putShort(itemlen);
if (itemlen > 0)
    buffer.put(desc.getBytes(UTF_8));

buffer.flip();

logger.debug("MeetChatMessage:" + this.toString());
logger.debug("MeetChatMessage toBuffer:" + buffer.getHexDump());
```

payload编码