**OMTP协议说明文档**

OMTP(Open Message Transport Protocol)是一套自定义IM消息传输协议。它工作在TCP/IP协议族上基于客户端-服务器即时通讯系统设计的消息通讯协议，数据编码方式为基于Tars/PB的二进制类型。

**设计思想**

1 将业务数据进行抽象归纳，分为频繁变动和稳定不变两部分；

2 IM通道分离频繁变动的业务；

3 IM 通道从复杂多变的业务中解放出来；

4 OMTP协议只负责底层IM通道和稳定不变的业务部分；

5 当业务频繁变化时，OMTP协议最大限度保持不变；

6 增加业务的可扩展性，定制化特性；

**协议包含的业务**

IM协议的定义主要包含如下业务：

• 用户登录及身份认证

• 退出登录

• 用户被踢

• 心跳

• 业务通知消息发送/响应

• 单聊消息发送/响应

• 群聊消息发送/响应

• 最新消息会话，未读数获取请求/响应

• 历史消息获取/响应

• 数据同步请求/响应

• 开始旁听会话请求/响应

• 结束旁听会很请求/响应

• 群聊创建请求/响应

• 群聊解散请求/响应

• 群聊加人请求/响应

• 群聊减人请求/响应

• 群聊名称，头像等修改请求/响应

**OMTP协议字段**

**协议格式**

OMTP(Open Message Transport Protocol)控制报文由两部分组成，按照图示描述的顺序：  
OMTP控制报文的结构：

**字段**

**长度**

**说明**

Length

4字节

报文长度（包含本字段4字节）

Header

可变

报文头，tars格式二进制数据

Payload

可变

消息体，tars格式二进制数据

struct OTIMPack

{

0 require OTIMHeader header;

1 require vector<byte> payload;

};

**报头 Header**

每个OMTP控制报文都包含一个报头。

**格式**

**字段**

**类型**

**长度**

**说明**

type

short

2字节

报文的类型

version

short

2字节

协议版本号

flags

long

4-8字节

详见下文flags说明

packId

string

可变长度

报文Id，建议为UUID

struct OTIMHeader

{

0 require short type;

1 require short version;

2 require long flags;

3 require string packId;

};

**flags说明**

**含义**

**长度**

**说明**

ACK

1 bit

是否为请求响应，0:请求,1:响应；

DUP

1 bit

是否重发，0:不重复，1:重复

COUNTER

1 bit

是否计数，0:不计，1:计数

OVERRIDE

1 bit

是否覆写，0:正常,1:覆写

REVOKE

1 bit

是否撤回，0:正常,1:撤回

HIGHPRJ

1 bit

高优先级，0:正常,1:高优先级

COMPRESS

3 bits

压缩算法，00:无压缩，01:zlib

CRYPTO

3 bits

加密算法，00:无加密，01:AES

RESERVED

13-51 bits

保留

//ACK|DUP|COUNTER|OVERRIDE|REVOKE|HIGHPRJ|COMPRESS|CRYPTO|RESERVED

struct PT\_FLAGS\_BITS{

unsigned int ack:1;

unsigned int dup:1;

unsigned int counter:1;

unsigned int overwrite:1;

unsigned int revoke:1;

unsigned int highPRJ:1;

unsigned int compress:3;

unsigned int crypto:3;

unsigned int reserved:20;

};

**报文类型**

enum PACK\_TYPE

{

PT\_NONE = 0, //00-09 保留

PT\_PING = 10, //心跳

PT\_LOGIN = 11, //登录认证

PT\_LOGOUT = 12, //登出

PT\_KICKOUT = 13, //踢出

PT\_MSG\_SINGLE\_CHAT = 14,//单聊消息

PT\_MSG\_GROUP\_CHAT = 15,//群聊消息

PT\_MSG\_BIZ\_NOTIFY = 16,//业务通知消息

PT\_MSG\_CTRL = 17, //消息控制指令（比如撤销，删除等）

PT\_MSG\_READ = 18, //消息已读

PT\_HOTSESSION\_SYNC = 19, //热会话同步

PT\_HIGH\_PRIOR\_MSG\_SYNC = 20, //高优先级消息同步

PT\_HISTORY\_MSG\_PULL = 21, //拉历史消息

PT\_SESSION\_MONITOR\_START = 22,//监控会话开始

PT\_SESSION\_MONITOR\_STOP = 23,//监控会话结束

PT\_SESSION\_MONITOR\_SYNC = 24, //监控会话同步（同步登陆用户的会话）

PT\_SYNC\_DATA\_CMD = 25, //同步数据指令,根据该指令可以客户端可以进行各种增量数据同步（比如会话，好友记录，群聊等）

PT\_GROUPCHAT\_SYNC = 40, //同步我的群(我创建，加入，监听的群里)

PT\_GROUPCHAT\_CREATE = 41, //创建群聊

PT\_GROUPCHAT\_JION = 42, //加入群聊

PT\_GROUPCHAT\_QUIT = 43, //退出群聊

PT\_GROUPCHAT\_DISMISS = 44, //解散群聊

PT\_GROUPCHAT\_UPDATE\_CREATOR = 45, //换群主

PT\_GROUPCHAT\_INFO\_UPDATE = 46, //更新群资料

PT\_GROUPCHAT\_MEMBERS\_GET = 47, //获取群成员

PT\_FRIEND\_ADD = 51, //添加好友

PT\_FRIEND\_DEL = 52, //删除好友

PT\_FRIEND\_SYNC = 53, //同步好友

PT\_USERINFO\_GET = 54, //获取用户信息

PT\_USERINFO\_UPDATE = 55, //用户信息更新

PT\_USERATTRIBUTE\_SET = 56, //设置用户属性，比如昵称

PT\_USERATTRIBUTE\_GET = 57, //获取用户属性，比如昵称

PT\_SESSIONATTRIBUTE\_SET = 58, //设置会话属性，比如置顶，免打扰

};

**报文体Payload**

根据不同type（报文类型），Payload内容不同；  
根据不同的format，采用不同传输数据格式；  
报文包含的内容信息及简单流程详见下文。

**报文类型详细说明**

**PT\_PING 心跳**

• 报文类型值：10

• flags ACK位: 0

• Payload：无

**PT\_LOGIN 登录请求**

• 报文类型：11

• flags ACK位: 0

• Payload：LoginReq

**字段名称**

**类型**

**说明**

clientId

string

用户名

username

string

用户名

password

string

密码

devicetype

DEVICE\_TYPE

设备类型：1：ios，2：android，3：windows PC,4:Mac OS

deviceName

string

设备名称

deviceId

string

设备id

version

string

app版本号，如：“3.2.1”

enum DEVICE\_TYPE

{

DEVICE\_NONE = 0,

DEVICE\_IOS = 1,

DEVICE\_ANDROID = 2,

DEVICE\_MACOS = 3,

DEVICE\_WINDOWS = 4,

DEVICE\_LINUX = 5,

};

/\* 登录请求 \*/

struct LoginReq

{

0 require string clientId ; //用户UserId

1 require string userName ; //登录名（手机号）

2 optional string password ; //密码,token（如：a502c991-9ab9-4823-bde9-520d9169545b）

3 optional DEVICE\_TYPE deviceType; //设备类型：1：ios，2：android，3：windows PC,4:Mac OS 5:Linux

4 optional string deviceName; //设备名称

5 optional string deviceId; //设备唯一标识

6 optional string version; //客户端的版本号

};

**PT\_LOGIN 登录响应**

• 报文类型值：11

• flags ACK位: 1

• Payload：LoginResp

**字段名称**

**类型**

**说明**

code

int

登录返回码

desc

string

错误原因

struct LoginResp

{

0 require int code; //认证结果码值

1 optional string clientId;

2 optional string extraData;

};

**PT\_LOGOUT 登出**

• 报文类型：12

• flags ACK位: 0

• Payload：无

**PT\_LOGOUT 登出响应**

• 报文类型：12

• flags ACK位: 1

• Payload：无

**PT\_KICKOUT 踢出**

• 报文类型：13

• flags ACK位: 0

• Payload：KickOutReq

struct KickOutReq

{

0 require int deviceType; //准备要踢掉的 deviceType

1 require string deviceId; //准备要踢掉的 设备类型deviceId，

};

**PT\_KICKOUT 踢出响应**

• 报文类型：13

• flags ACK位: 1

• Payload：无

**PT\_MSG\_SINGLE\_CHAT 单聊消息**

• 报文类型：14

• flags ACK位: 0

• Payload：MsgReq

**字段名称**

**类型**

**说明**

seqid

long

序列号，排序用，服务端产生

timestamp

long

时间戳

sessionid

string

会话id，发送方根据约定规则填写，具体规则后文详述

from

string

发送者：用户id

to

string

接收者：用户id

status

int

消息状态，0:正常，1:撤回，2:删除

contentType

int

消息内容类型：1：文本， 2：语音，3：图片，5：位置，6:视频, 7：文件 8：富媒体

content

Format（header里面定义格式）

根据contentType，有不同字段，下文详细说明

struct MsgReq

{

0 require string sessionId; //会话Id

1 optional long seqId ; //由服务器端填写,客户端用来排序.

2 optional long timestamp ; //消息时间,由服务器端来填写.用来在客户端显示

3 require string from ; //消息发送者的名片Id或appid

4 require string to ; //消息接收者的名片Id或者群组Id

5 optional string pushInfo ; //推送信息IOS-Push用,此字段无值表示不需要push

6 optional int contentType ; //content 类型，文本，语音，图片等等

7 optional string content ; //内容可为:pb/json/text

8 optional int status ; //消息状态 0:正常,1:删除,2:撤销

};

**PT\_MSG\_SINGLE\_CHAT 单聊消息响应**

• 报文类型：14

• flags ACK位: 1

• Payload：MsgAck

**字段名称**

**类型**

**说明**

code

int

返回值

sessionId

string

会话Id

seqid

long

序列号，排序用，服务端产生

timestamp

long

时间戳

struct MsgAck

{

0 require int code;

1 require string sessionId;

2 require long seqId;

3 require long timestamp; //消息类型

};

**PT\_MSG\_GROUP\_CHAT 群聊消息**

• 报文类型：15

• flags ACK位: 0

• Payload：MsgReq（同单聊）

**字段名称**

**类型**

**说明**

seqid

long

序列号，排序用，服务端产生

timestamp

long

时间戳

sessionid

string

会话id，发送方根据约定规则填写，具体规则后文详述

from

string

发送者

to

string

接收者：群地址

status

int

消息状态，0:正常，1:撤回，2:删除

contentType

int

消息内容类型：1：文本， 2：语音，3：图片，5：位置，6:视频, 7：文件 8：富媒体

content

Format（header里面定义格式）

根据contentType，有不同字段，下文详细说明

**PT\_MSG\_GROUP\_CHAT 群聊消息响应**

• 报文类型：15

• flags ACK位: 1

• Payload：MsgAck(同单聊）

**字段名称**

**类型**

**说明**

seqid

long

序列号，排序用，服务端产生

timestamp

long

时间戳

code

int

返回值

**PT\_MSG\_BIZ\_NOTIFY 业务通知消息请求**

• 报文类型：16

• flags ACK位: 0

• Payload：MsgReq（同单聊）

注：此条指令仅仅通知客户端出发相应同步，不做数据传输。

**PT\_MSG\_BIZ\_NOTIFY 业务通知消息响应**

• 报文类型：16

• flags ACK位: 1

• Payload：MsgAck(同单聊）

**PT\_SYNC\_DATA\_CMD**

**字段名称**

**类型**

**说明**

cmd

long

同步业务指令

**PT\_MSG\_CTRL 消息控制（比如撤销，删除等）**

• 报文类型：17

• flags ACK位: 0

• Payload：MsgControl

**字段名称**

**类型**

**说明**

command

int

1.撤销，2.删除

sessionId

string

会话Id

packId

string

目标消息packId

struct MsgControl

{

0 require int command ; //1.撤销，2.删除

1 require string sessionId ; //会话Id

2 require string packId ; //目标消息packId

3 require long seqId ; //目标消息seqId

};

**PT\_MSG\_CTRL 消息控制响应**

• 报文类型：17

• flags ACK位: 1

• Payload：CommonResp

**字段名称**

**类型**

**说明**

code

int

错误码

desc

string

错误信息

struct CommonResp

{

0 require int code ; //错误码

1 require string desc ; //错误信息

};

**PT\_MSG\_READ 消息已读**

• 报文类型：18

• flags ACK位: 0

• Payload:MsgReaded

**字段名称**

**类型**

**说明**

sessionId

string

会话Id

seqId

long

起始seqId

struct MsgReaded

{

0 require string sessionId ; //会话Id

1 require long seqId ; //已读消息seqId

};

**PT\_MSG\_READ 消息已读响应**

• 报文类型：18

• flags ACK位: 1

• Payload:CommonResp

**PT\_HOTSESSION\_SYNC 热会话同步请求**

• 报文类型：19

• flags ACK位: 0

• Payload:HotSessionReq

**字段名称**

**类型**

**说明**

timestamp

long

时间戳

struct HotSessionReq

{

0 require long timestamp ; //单位微秒，本地拉取会话时间戳或者断点时间戳

};

**PT\_HOTSESSION\_SYNC热会话同步响应**

• 报文类型：19

• flags ACK位: 1

• Payload：HotSessionResp

**字段名称**

**类型**

**说明**

code

int

返回码

timestamp

long

时间戳

sessions

Vector

热会话列表

• HotSessionItem

**字段名称**

**类型**

**说明**

sessionId

string

会话id

readSeqId

long

已读seqId

unreadCount

Int

未读数

lastMsgs

Vector

最新n条消息

struct HotSessionItem

{

0 require string sessionId; // 会话Id

1 require long readSeqId; //会话已读游标

2 require int unreadCount = 0; //某会话未读数，默认值0

3 optional vector<OTIMPack> lastMsgs; //该会话最新一条可显示的msg,normal\_top\_msg至少存在一条

4 optional string attribute; //会话属性集

};

struct HotSessionResp

{

0 require int code = 0; //返回码 类型

1 require long timestamp ;

2 require vector<HotSessionItem> sessions; //所有会话的状态信息

};

**PT\_HIGH\_PRIOR\_MSG\_SYNC 高优先级消息同步请求**

• 报文类型：20

• flags ACK位: 0

• Payload: MsgHighPrioritySyncReq

**字段名称**

**类型**

**说明**

seqId

long

当前最新seqId

count

int

获取数量

struct MsgHighPrioritySyncReq

{

0 require long seqId;

1 require int count = 200;//获取数量

};

**PT\_PRIOR\_MSG\_SYNC 高优先级消息同步响应**

• 报文类型：20

• flags ACK位: 1

• Payload:MsgGetHighPriorityResp

**字段名称**

**类型**

**说明**

code

int

错误码

desc

string

错误描述

lastSeqId

long

最新seqId

msgs

vector

消息

struct MsgHighPrioritySyncResp

{

0 require int code; //服务请求的结果码

1 optional string desc; //服务请求的结果描述

2 require long lastSeqId;

3 optional vector<OTIMPack> msgs; //消息列表

};

**PT\_HISTORY\_MSG\_PULL 历史消息请求**

• 报文类型：21

• flags ACK位: 0

• Payload

**字段名称**

**类型**

**说明**

sessionId

string

会话id

seqId

long

起始seqid

count

int

消息数量

struct HistoryMsgPullReq

{

0 optional string sessionId;

1 require long seqId; //起始seqId

2 require int count = 200;//获取数量

};

**PT\_HISTORY\_MSG\_PULL 历史消息响应**

• 报文类型：21

• flags ACK位: 1

• Payload：HistoryMsgPullResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

错误信息

sessionId

string

会话id

msgs

vector

消息内容

struct HistoryMsgPullResp

{

0 require CommonErrorCode errorCode;

1 require string sessionId; //起始sessionId

2 require vector<OTIMPack> msgs;

};

**PT\_SESSION\_MONITOR\_START 监控会话开始请求**

• 报文类型：22

• flags ACK位: 0

• Payload:SessionMonitor

**字段名称**

**类型**

**说明**

sessionId

string

会话id

struct SessionMonitor

{

0 require string sessionId;

};

**PT\_SESSION\_MONITOR\_START 监控会话开始响应**

• 报文类型：22

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_SESSION\_MONITOR\_STOP 监控会话开始请求**

• 报文类型：23

• flags ACK位: 0

• Payload:SessionMonitor

**字段名称**

**类型**

**说明**

sessionId

string

会话id

struct SessionMonitor

{

0 require string sessionId;

};

**PT\_SESSION\_MONITOR\_STOP 监控会话开始响应**

• 报文类型：23

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_SESSION\_MONITOR\_SYNC 监听会话同步请求**

• 报文类型：24

• flags ACK位: 0

• Payload：无

**PT\_SESSION\_MONITOR\_SYNC 监听会话同步响应**

• 报文类型：24

• flags ACK位: 1

• Payload：HotSessionResp

**PT\_SYNC\_DATA\_CMD 同步数据指令**

• 报文类型：25

• flags ACK位: 1

• Payload:SyncDataReq

**字段名称**

**类型**

**说明**

command

int

业务类型 1,好友列表，2，会话，3，群聊

content

string

业务数据

struct SyncDataReq

{

0 require int command;

1 require string content;

};

**PT\_SYNC\_DATA\_CMD 同步数据指令响应**

• 报文类型：25

• flags ACK位: 1

• Payload：CommonErrorCode

**PT\_GROUPCHAT\_SYNC 群聊同步请求**

• 报文类型：40

• flags ACK位: 0

• Payload:SyncGroupChatReq

**字段名称**

**类型**

**说明**

timestamp

long

同步时间戳

struct GroupChatSyncReq

{

0 require long timestamp;

};

**PT\_GROUPCHAT\_SYNC 群聊同步响应**

• 报文类型：40

• flags ACK位: 1

• Payload：GroupChatSyncResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

错误信息

timestamp

long

同步时间戳

groupchats

vector

GroupChatInfo 群聊列表

struct GroupChatInfo

{

0 require string groupId; // 班级群groupid由classid和classcode组合生成

1 require string name; // 群聊名字

2 optional string avatar; // 群聊头像

3 require string creatorId; // 群聊创建者，即群主userId

4 optional string desc; // 群聊描述

5 optional int memberLimit; // 最大成员数

6 optional long createTime; // 创建时间

7 optional long updateTime; // 更新时间

};

struct GroupChatSyncResp

{

0 require CommonErrorCode errorCode;

1 require long timestamp;

2 require vector<GroupChatInfo> groupchats;

3 require vector<string> delGroupChatIds;

};

**PT\_GROUPCHAT\_CREATE 群聊创建请求**

• 报文类型：41

• flags ACK位: 0

• Payload:GroupChatCreateReq

struct GroupChatCreateReq

{

0 require GroupChatInfo groupInfo;

1 require vector<string> memberIds;

};

**PT\_GROUPCHAT\_CREATE 群聊创建响应**

• 报文类型：41

• flags ACK位: 1

• Payload:GroupChatCreateResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

groupId

string

struct GroupChatCreateResp

{

0 require CommonErrorCode errorCode;

1 require string groupId;

};

**PT\_GROUPCHAT\_JION 加入群聊请求**

• 报文类型：42

• flags ACK位: 0

• Payload:GroupChatJoinReq

**字段名称**

**类型**

**说明**

groupId

string

memberIds

vector

userId列表

struct GroupChatJoinReq

{

0 require string groupId;

1 require vector<string> memberIds;

};

**PT\_GROUPCHAT\_JION 加入群聊响应**

• 报文类型：42

• flags ACK位: 1

• Payload:GroupChatJoinResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

groupId

string

struct GroupChatJoinResp

{

0 require CommonErrorCode errorCode;

1 require string groupId;

};

**PT\_GROUPCHAT\_QUIT 退出群聊请求**

• 报文类型：43

• flags ACK位: 0

• Payload:GroupChatQuitReq

**字段名称**

**类型**

**说明**

groupId

string

operatorId

string

操作者userId

memberIds

vector

退出的userId列表

struct GroupChatQuitReq

{

0 require string groupId;

1 require string operatorId;

2 require vector<string> memberIds;

};

**PT\_GROUPCHAT\_QUIT 退出群聊响应**

• 报文类型：43

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_GROUPCHAT\_DISMISS 解散群聊请求**

• 报文类型：44

• flags ACK位: 0

• Payload:GroupChatDismissReq

**字段名称**

**类型**

**说明**

groupId

string

operatorId

string

操纵者userId

memberIds

vector

退出的userId列表

struct GroupChatDismissReq

{

0 require string groupId;

1 require string operatorId;

};

**PT\_GROUPCHAT\_DISMISS 解散群聊响应**

• 报文类型：44

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_GROUPCHAT\_CREATOR\_UPDATE 换群主请求**

• 报文类型：45

• flags ACK位: 0

• Payload:GroupChatCreatorUpdateReq

**字段名称**

**类型**

**说明**

groupId

string

operatorId

string

操纵者userId

memberIds

vector

退出的userId列表

struct GroupChatCreatorUpdateReq

{

0 require string groupId;

1 require string operatorId;

2 require string newCreatorId;

};

**PT\_GROUPCHAT\_CREATOR\_UPDATE 换群主响应**

• 报文类型：45

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_GROUPCHAT\_INFO\_UPDATE 更新群资料请求**

• 报文类型：46

• flags ACK位: 0

• Payload:GroupChatInfoUpdateReq

**字段名称**

**类型**

**说明**

operatorId

string

操作者userId

groupInfo

GroupChatInfo

群信息

struct GroupChatInfoUpdateReq

{

0 require string operatorId;

1 require GroupChatInfo groupInfo;

};

**PT\_GROUPCHAT\_INFO\_UPDATE 更新群资料响应**

• 报文类型：46

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_GROUPCHAT\_MEMBER\_GET 获取群成员请求**

• 报文类型：47

• flags ACK位: 0

• Payload:GroupChatMemberGetReq

**字段名称**

**类型**

**说明**

groupId

string

群Id

struct GroupChatMemberGetReq

{

0 require string groupId;

};

**PT\_GROUPCHAT\_MEMBER\_GET 获取群成员响应**

• 报文类型：47

• flags ACK位: 1

• Payload:GroupChatMemberGetResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

groupId

string

群Id

memberIds

vector

群成员userId列表

struct GroupChatMemberGetResp

{

0 require CommonErrorCode errorCode;

1 require string groupId;

2 require vector<string> memberIds;

};

**PT\_FRIEND\_ADD 添加好友请求**

• 报文类型：51

• flags ACK位: 0

• Payload:FriendAddReq

**字段名称**

**类型**

**说明**

friends

vector

好友FriendInfo列表

struct FriendInfo

{

0 require string userId;

1 require string friendId;

2 require string remark;//备注名

};

struct FriendAddReq

{

0 require vector<FriendInfo> friends;

};

**PT\_FRIEND\_ADD 添加好友响应**

• 报文类型：51

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_FRIEND\_DEL 删除好友请求**

• 报文类型：52

• flags ACK位: 0

• Payload:FriendDelReq

**字段名称**

**类型**

**说明**

friends

vector

好友FriendInfo列表

struct FriendDelReq

{

0 require vector<FriendInfo> friends;

};

**PT\_FRIEND\_DEL 删除好友响应**

• 报文类型：52

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_FRIEND\_SYNC 同步好友请求**

• 报文类型：53

• flags ACK位: 0

• Payload:FriendDelReq

**字段名称**

**类型**

**说明**

timestamp

long

增量时间戳

struct FriendSyncReq

{

0 require long timestamp;

};

**PT\_FRIEND\_SYNC 同步好友响应**

• 报文类型：53

• flags ACK位: 1

• Payload:FriendSyncResp

struct FriendSyncResp

{

0 require CommonErrorCode errorCode;

1 require vector<FriendInfo> friends;

};

**PT\_USERINFO\_GET 获取用户资料请求**

• 报文类型：54

• flags ACK位: 0

• Payload:UserInfoGetReq

**字段名称**

**类型**

**说明**

userIds

vector

用户id列表

struct UserInfoGetReq

{

0 require vector<string> userIds;

};

**PT\_USERINFO\_GET 获取用户资料响应**

• 报文类型：54

• flags ACK位: 1

• Payload:UserInfoGetResp

**字段名称**

**类型**

**说明**

errorCode

CommonErrorCode

userInfos

vector

UserInfo 列表

struct UserInfoGetResp

{

0 require CommonErrorCode errorCode;

1 require vector<UserInfo> userInfos;

};

struct UserInfo

{

0 require string userId;

1 require string name;

2 require string avator;

3 require string mobile;

4 require long birthday;//timestamp

};

**PT\_USERINFO\_UPDATE 更新用户资料请求**

• 报文类型：55

• flags ACK位: 0

• Payload:UserInfoUpdateReq

**字段名称**

**类型**

**说明**

userInfo

UserInfo

用户资料

struct UserInfoUpdateReq

{

0 require UserInfo userInfo;

};

**PT\_USERINFO\_UPDATE 更新用户资料响应**

• 报文类型：55

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_USERATTRIBUTE\_SET 设置用户属性，比如昵称**

• 报文类型：56

• flags ACK位: 0

• Payload:UserAttrSetReq

**字段名称**

**类型**

**说明**

attribute

UserAttribute

属性UserAttribute

struct UserAttribute

{

0 require string userId;

1 require string friendId;

2 require string attrName;

3 require string attrValue;

};

struct UserAttrSetReq

{

0 require UserAttribute attribute;

};

**PT\_USERATTRIBUTE\_SET 设置用户属性响应**

• 报文类型：56

• flags ACK位: 1

• Payload:CommonErrorCode

**PT\_USERATTRIBUTE\_GET 获取用户属性请求**

• 报文类型：57

• flags ACK位: 0

• Payload:无

**PT\_USERATTRIBUTE\_GET 获取用户属性响应**

• 报文类型：57

• flags ACK位: 1

• Payload:UserAttrGetResp

struct UserAttrGetResp

{

0 require CommonErrorCode errorCode;

1 require vector<UserAttribute> attributes;

};

**PT\_SESSIONATTRIBUTE\_SET 设置会话属性，比如置顶，免打扰**

• 报文类型：58

• flags ACK位: 0

• Payload:SessionAttrSetReq

**字段名称**

**类型**

**说明**

userId

string

sessionId

string

attrName

string

属性名

attrValue

string

属性值

struct SessionAttrSetReq

{

0 require string userId;

1 require string sessionId;

2 require string attrName;

3 require string attrValue;

};

**PT\_SESSIONATTRIBUTE\_SET 更新会话属性响应**

• 报文类型：58

• flags ACK位: 1

• Payload:CommonErrorCode