**NN协议文档**

**编 写 人**：刘自然

**编写日期**：2020年04月22日

**文档修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **说明** | **日期** | **变更人** |
| 1 | 创建文档 | 2020.04.22 | 刘自然 |
| 2 | 文档修改 | 2020.04.28 | 黎正红 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 消息类型定义

#define PACKET\_MAXLEN 1400

#define PACKET\_HEADLEN 5

#define MSG\_MAXLEN 1200

#define TOCKEN\_MAXLEN 64

#define DEVICEID\_MAXLEN 32

typedef enum \_CMsgType

{

E\_CMD\_PACKET\_UNKNOWN = 0,

E\_CMD\_PACKET\_HEARBEAT = 1,

E\_CMD\_PACKET\_HEARBEATACK = 2,

E\_CMD\_PACKET\_BIND = 3,

E\_CMD\_PACKET\_BINDACK =4,

E\_CMD\_PACKET\_OFFLINE = 5,

E\_CMD\_PACKET\_OFFLINEACK = 6,

E\_CMD\_PACKET\_KICKOFF = 7,

E\_CMD\_PACKET\_KICKOFFACK = 8,

E\_CMD\_P2PMSG = 9， //私聊消息类型

E\_CMD\_GROUPMSG = 10，//群聊消息类型

E\_CMD\_CHATROOMMSG =11，//聊天室消息类型

E\_CMD\_MSGACK = 12, //ACK消息类型

E\_CMD\_ENTERCHATROOM=13,//进入聊天室

E\_CMD\_ENTERCHATROOMACK=14,//进入聊天室ACK

E\_CMD\_EXITCHATROOM=15,//退出聊天室

E\_CMD\_EXITCHATROOMACK=16,//退出聊天室ACK

} CMsgType;

# REDIS相关列表说明

## TCP网关相关的REDIS列表：

upmsglisttcp,存储上行消息，表结构为list。

downcmdlisttcp,存储上行消息的ACK，表结构为list。

downmsglisttcp,存储下行消息，表结构为list。

upcmdlisttcp,存储下行消息的ACK，表结构为list。

## WebSocket网关相关的REDIS列表：

upmsglistjson,存储上行消息，表结构为list。

downcmdlistjson,存储上行消息的ACK，表结构为list。

downmsglistjson,存储下行消息，表结构为list。

upcmdlistjson,存储下行消息的ACK，表结构为list。

## 用户状态REDIS说明

### 以UserId为主键的SET

value为DeviceId。

### 以”UserId-DeviceId”为主键的HSET

状态：HSET里有KEY为STAT，VALUE为0或1

网关类型：HSET里有KEY为GTYPE，VALUE为TCP或WEB

# **通讯协议**

通讯协议：主要是网关与客户端的TCP通讯、WEBSOCKET通讯的协议。

## 消息结构

### 数据类型

二进制块：变长INT + 二进制数据块

**变长INT说明：**

一种变长的整型数值，字节顺序如下：

字节a

高位

字节b

字节c

字节d

低位

字节1为必选字节。

其比特序的高2位（掩码0xC0）为flag标记。

|  |  |  |
| --- | --- | --- |
| **flag** | **描述** | **取值范围** |
| 0x00 | 字节a低6位表示具体数值 | 0~63 |
| 0x01 | 字节a低6位 + 字节b表示具体数值 | 64~16383 |
| 0x10 | 字节a低6位 + 字节b + 字节c 表示具体数值 | 16384~4194303 |
| 0x11 | 字节a低6位 + 字节b + 字节c + 字节d 表示具体数值 | 4194304~1073741823 |

### 消息组成

消息由“通讯头”+“消息体”组成，如下图：

serialNo

msgLen

body

….

…

flag

Head

Body

msgType

### 通讯头(Head)结构：

|  |  |  |
| --- | --- | --- |
| **字段** | **描述** | **长度** |
| msgLen | 消息长度 | 定长，2字节，表示总长度 |
| msgType | 消息类型 | 定长，1字节 |
| serialNo | 消息流水号 | 定长，1字节 |
| flag | 标记值 | 定长，1字节  0000 0000 不加密消息  0000 0001 rc4加密消息  0000 0011 加密+上行客户端版本号参数  0000 0111 加密+上行客户端版本号+添加网络状态参数 |
| body | 消息内容 | 变长，由size可计算出长度。 |

### 具体通讯协议

////////下面几个是通讯协议，body直接使用的是结构体

typedef struct \_Communication\_Head

{

unsigned short msgLen;

unsigned char msgType; // 消息类型

char serialNo;

char flag;

} CCommunicationHead;

// 心跳

typedef struct \_CHearBeat

{

unsigned int userId;

} CHearBeat;

typedef struct \_CHearBeatAck

{

unsigned int userId;

} CHearBeatAck;

// 绑定

typedef struct \_CBind

{

unsigned int userId;

unsigned char deviceType;

char token[TOCKEN\_MAXLEN];

char deviceId[DEVICEID\_MAXLEN];

char versionNo[3];

} CBind;

typedef struct \_CBindAck

{

unsigned int userId;

char deviceId[DEVICEID\_MAXLEN];

} CBindAck;

// 下线

typedef struct \_COffLine

{

unsigned int userId;

char deviceId[DEVICEID\_MAXLEN];

} COffLine;

typedef struct \_COffLineAck

{

unsigned int userId;

char deviceId[DEVICEID\_MAXLEN];

} COffLineAck;

// 踢掉

typedef struct \_CKickOff

{

unsigned int userId;

char deviceId[DEVICEID\_MAXLEN];

} CKickOff;

typedef struct \_CKickOffAck

{

unsigned int userId;

char deviceId[DEVICEID\_MAXLEN];

} CKickOffAck;

//进入聊天室

typedef struct \_CEnterChatRoom

{

unsigned int userId;

unsigned int chatRoomId;

char deviceId[DEVICEID\_MAXLEN];

} CEnterChatRoom;

//进入聊天室ACK

typedef struct \_CEnterChatRoomAck

{

unsigned int userId;

unsigned int chatRoomId;

char deviceId[DEVICEID\_MAXLEN];

} CEnterChatRoomAck;

//退出聊天室

typedef struct \_CExitChatRoom

{

unsigned int userId;

unsigned int chatRoomId;

char deviceId[DEVICEID\_MAXLEN];

} CExitChatRoom;

]

//退出聊天室ACK

typedef struct \_CExitChatRoomAck

{

unsigned int userId;

unsigned int chatRoomId;

char deviceId[DEVICEID\_MAXLEN];

} CExitChatRoomAck;

# 消息协议

消息协议：业务服务间消息交互的协议。注意：客户端与网关交互消息协议结构为：通讯头 +“消息头”+“消息体”

### 消息组成

消息由“消息头”+“消息体”组成，注意：这里的消息体根据ecodetype有Protobuf、json、结构体等类型。如下图：

|  |
| --- |
| ecodeType |
| msgType |
| len |
| fromUid |
| toUid |
| timeStamp |
| deviceId |
| Body  ...  ... |

MsgHead

Body

### 消息头(MsgHead)字段说明：

TCP网关定长：

|  |  |  |
| --- | --- | --- |
| **字段** | **描述** | **长度** |
| ecodeType | 区分数据类型 | 定长，1字节 0: 结构体,1: protoBuf,2: json,3:绑定结构体 |
| msgType | 消息类型 | 定长，1字节 |
| len | 数据长度 | 定长，2字节 body结构的整体长度 |
| fromUid | 发送者ID | 定长，4字节 |
| toUid | 接受者ID | 定长，4字节 根据type判断是userId还是groupId |
| timeStamp | 时间戳 | 定长，8字节 |
| deviceId | 设备ID | 定长，32字节 |
| data | 消息内容 | 变长，由len可计算出长度。 |
| mqTag | MQ的标记 | 定长，8字节 |
| mqChanId | MQ通道ID | 定长，2字节 |

typedef struct \_CMsgHead

{

unsigned char ecodeType; //0: 结构体,1: protobuf,2: json,3:绑定结构体

unsigned char msgType ; // 消息类型,1 byte

unsigned short len;

unsigned int fromUid ; // 发送者ID

unsigned int toUid; // 接受者ID 根据type判断是userId还是groupId

long long timeStamp;

char deviceId[DEVICEID\_MAXLEN]; // 设备ID

unsigned long long  mqTag;  
  unsigned short  mqChanId;

} CMsgHead;

WebSocket JSON：

|  |  |  |
| --- | --- | --- |
| **字段** | **描述** | **备注** |
| ecodeType | 区分数据类型 | JSON， 0: 结构体,1: protobuf,2: json,3:绑定结构体 |
| msgType | 消息类型 | JSON， |
| len | 数据长度 | JSON， body结构的整体长度 |
| fromUid | 发送者id | JSON， |
| toUid | 接受者id | JSON， 根据type判断是userId还是groupId |
| timeStamp | 时间戳 | JSON， |
| deviceId | 设备ID | JSON， |
| data | 消息内容 | JSON，由len可计算出长度。 |
| mqTag | MQ的标记 | JSON, |
| mqChanId | MQ通道 | JSON, |

### 具体消息定义

具体消息定义详情见cmessage.proto