**Push服务器api接口说明**

* 约束

1. 连接方式

Push服务器向前台提供一个socket端口，接受前台的连接，前台和Push服务器之间通过通信协议发送消息的形式，与Push服务器进行互动。

Push服务器和前台之间的消息为双向的，但本文仅说明前台到Push服务器的协议，后续的扩展可以添加Push服务器到前台的协议定义。

1. Push服务器不关心前台和客户端的协议，前台和客户端的协议可以通过MSG\_DELIVER所负载的消息来定义和扩展。Push服务器仅建立前台和客户端以及客户端之间的通信通道,.
2. 作为接入到该Push服务器系统的所有连接都需要确定一个唯一的ID。目前的接入主要有前台以及客户端，则前台和客户端都必须有唯一的ID，不接受相同ID的客户端接入。前台连接以IP地址为验证依据，只允许从特定（配置）的IP地址连接。
3. 必须对Push服务器提供一个数据库接口，通过该接口获取全网客户端信息，从而可实现全网广播的需求。
4. 该api接口说明适用于前台和客户端

* 接入流程

1. 前台广播接入流程
2. 前台以Tcp方式连接Push服务器集群的监听端口（端口号待定），其中前台为tcp client；目前只允许从特定（配置）的IP地址连接。
3. 当连接建立成功后，前台发送MSG\_LOGIN消息，进行登录【注:目前该步骤暂略】。
4. 前台向单个或多个客户端发送消息(MSG\_DELIVER)，或者发送广播消息(MSG\_BROADCAST)
5. 客户端接入流程
6. 客户端以TCP方式连接负载均衡服务器，获取连接服务器的IP及监听端口(发送MSG\_LOOKUP\_SERVER, 接收MSG\_SERVER\_ADDR)。获取成功后断开此连接。
7. 客户端以Tcp方式连接连接服务器的监听端口，其中客户端为tcp client；
8. 当连接建立成功后，前台发送MSG\_LOGIN消息，进行登录
9. 客户端等待接收Push服务端转发的各类消息
10. 客户端向单个或多个其他客户端发送消息(MSG\_DELIVER)

* 协议

//消息号

enum

{

\_MSG\_LOGIN=1, //客户端建立连接后，必须发送的第一条LOGIN消息

\_MSG\_PING=2, //客户端的ping包（防止socket长连接失效）

\_MSG\_DELIVER=3, //push服务器和客户端之间的发送的消息标志

\_MSG\_BROADCAST=4, //广播消息

\_MSG\_LOOKUP\_CLIENT=5, // PUSH服务器发送此消息查询客户端是否登录

\_MSG\_RESULT=6, // 返回此消息给PUSH服务器表示消息发送结果

\_MSG\_LOOKUP\_SERVER=101, //客户端发送此消息查询连接服务器地址

\_MSG\_SERVER\_ADDR=102, //服务端发送此消息给客户端回复服务器地址

};

enum

{

\_ALL\_BROADCAST = 1; //全部广播，不论是否在线

\_ONLINE\_BROADCAST =2; //仅在线客户广播

};

//客户端ID或前台ID

typedef unsigned long UID;

struct MSG\_HEAD

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。即上面的enum值

};

//登陆消息

struct MSG\_LOGIN

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_MSG\_LOGIN

UID ulUserId; //客户端唯一ID

long lAuthorCode; //验证码, 暂时无用，可设置为 0

};

struct MSG\_PING

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_MSG\_PING

UID ulUserId; //客户端唯一ID

};

//服务端接收到消息时，所打的时间戳

//年月日-时分秒（其中年占2byte）

struct SVR\_TIME\_STAMP

{

short YY;

char MM;

char DD;

char sp; //split char='-'

char HH;

char mm;

char ss;

};

//消息负载分发

struct MSG\_DELIVER

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_MSG\_DELIVER

UID ulSrcUid; //发送方唯一ID

SVR\_TIME\_STAMP timestamp //时间戳SVR\_TIME\_STAMP

int nTarAmount; //向单人（1）或多人（n）发送时，表示目标总数

UID ulTarUIds[1]; //目标唯一ID数组，有多少列多少

//UID ulTarUid1;

//UID ulTarUid2;

//UID ulTarUid3;

//char msginfo[1]; //所发送的具体消息

};

//广播消息

struct MSG\_BROADCAST

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_MSG\_BROADCAST

unsigned short usBcType; //广播模式

UID ulSrcUid; //发送方唯一ID（广播方）

SVR\_TIME\_STAMP timestamp //时间戳SVR\_TIME\_STAMP

char msginfo[1]; //所发送的具体消息

};

// 查询客户端是否登录

struct MSG\_LOOKUP\_CLIENT

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_MSG\_LOOKUP\_CLIENT

UID ulClientUid; //客户端ID

}

// 回复PUSH服务器端消息结果

struct MSG\_RESULT \*

{

unsigned short usSize; //消息总长

unsigned short usType; //消息类型。\_ MSG\_RESULT

unsigned short usMsgType; // 消息类型, 表示是哪个消息的结果

unsigned short usResult; // 结果: 0: 消息发送成功

// 或客户端登录

// 非0: 消息发送失败

// 客户端未登录或操作失败

UID ulClientUid; // 回复MSG\_LOOKUP\_CLIENT时, 非0

// 回复其他消息时， 无用， 值为0

};

\*: web端发送 MSG\_DELIVER, MSG\_BROADCAST, MSG\_LOOKUP\_CLIENT时，PUSH服务器都会回复MSG\_RESULT给WEB 端。MSG\_RESULT的第三个字段 usMsgType表示是哪个消息的RESULT， usResult表示消息发送/查询的结果，当回复MSG\_LOGIN\_CLIENT时， ulClientUid表示是哪个客户端的登录结果。

struct MSG\_LOOKUP\_SERVER

{

unsigned short usSize; //消息总长 = 4

unsigned short usType; //消息类型。\_ MSG\_LOOKUP\_SERVER

};

struct MSG\_SERVER\_ADDR

{

unsigned short usSize; //消息总长 = 4

unsigned short usType; //消息类型。\_ MSG\_ SERVER\_ADDR

unsigned char AA; // IP地址第一个字节

unsigned char BB; // IP地址第二个字节

unsigned char CC; // IP地址第三个字节

unsigned char DD; // IP地址第四个字节

unsigned short Port; //监听端口

};