TCONND 概要设计文档

腾讯科技（深圳）有限公司

修订历史记录

日期	备注	版本	作者
<2008-11-23>		<创建>	hardway
<2008-11-25>		TSF4G_TCONND_01_0001	hardway
<2008-12-22>		TSF4G_TCONND_01_0002	hardway
<2008-12-27>		TSF4G_TCONND_01_0003	hardway
<2009-01-06>		TSF4G_TCONND_01_0004	hardway
<2009-01-08>		TSF4G_TCONND_01_0005	hardway
<2009-02-17>		TSF4G_TCONND_01_0006	hardway
<2009-05-13>		TSF4G_TCONND_01_0007	hardway
<2009-08-31>		TSF4G_TCONND_01_0008	hardway

目录

1 引言.....5
1.1 编写目的.....5
1.2 定义.....5
1.3 参考资料.....5
2 功能介绍.....5
2.1 需求描述.....5
2.2 系统拓扑图.....5
2.3 功能详解.....5
2.4 功能设计.....6
2.5 内部定义.....6
3 交互图.....6
3.1 直接通信.....6
3.2 加密通信.....7
3.2.1 签名签证过程.....7
3.2.2 重新连接.....9
3.2.3 释放连接.....11
3.2.4 路由跳转.....12
4 协议定义.....13
4.1 Client<->TConnd 协议定义.....13
4.1.1 签名验证请求消息.....14
4.1.2 签名验证应答消息.....17
4.1.3 连接握手消息.....17
4.1.4 用户连接应答信息.....18
4.1.5 重连信息.....18
4.1.6 用户连接排队信息。.....19
4.1.7 更改密钥信息.....19
4.1.8 通信信息.....20
4.1.9 未加密下行包.....20
4.2 TConnd<->LogicSvr 协议定义.....20
4.2.1 连接建立消息.....22
4.2.2 重连消息.....22
4.2.3 连接断开消息.....23
4.2.4 跳转预分配连接.....24
4.2.5 通信包.....24

4.2.6 设置路由信息..... 25
4.2.7 设置连接限制信息..... 25
4.2.8 心跳协议 tconnd<->server..... 26
4.3 TSF4G_TCONND_02_0000..... 26
4.4 TSF4G_TCONND_01_0008..... 26

1 引言

1.1 编写目的

本文档主要用于介绍了 tconnd 基本功能,交互流程及开发使用接口。适用于 tconnd.使用者及 tconnnd 接口开发者。

1.2 定义

- TCONND TSF4G connector daemon 服务器接入进程
- LogicSvr 逻辑服务器,tconnd 提供接入服务的对象。
- TDR TSF4G Data Representation,数据描述的组件。
- TBUS TSF4G 进程间通信组件
- TAPP TSF4G 应用程序框架

1.3 参考资料

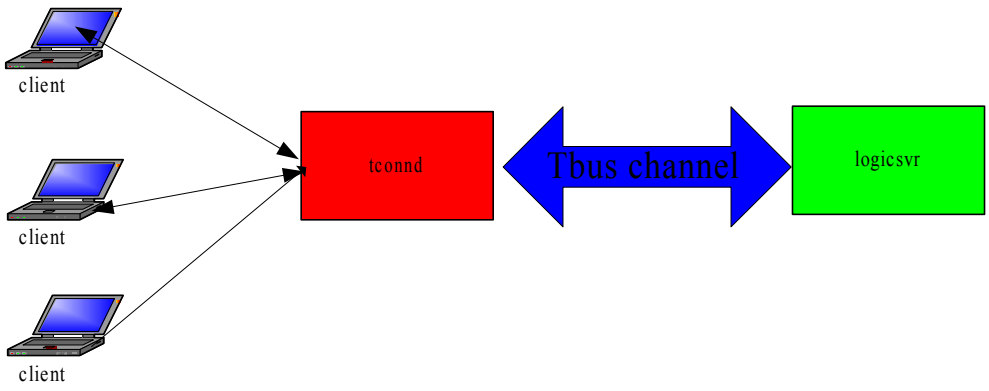
1. TDR 参考文档
2. TBUS 参考文档
3. TAPP 参考文档

2 功能介绍

2.1 需求描述

TCONND 主要满足 LogicSvr 网络接入的功能,实现客户端连接管理,通信消息组包的功能。

2.2 系统拓扑图



2.3 功能详解

Tconnd 的主要功能如下:

- 客户端网络连接管理,包括 Socket 连接建立,连接限制及连接释放通知 LogicSvr,连接

流量控制等。

- 消息组包机制,目前提供四种类型的消息组包策略。

  1) 二进制消息组包:应用使用 TDR 描述通信协议,TConnd 根据元数据相关长度字段分割二进制流。

  2) 文本消息(Null 结尾消息)组包:TConnd 根据'\0'字符分割消息。

  3) 直接转发:Tconnd 不组包,收到字节流,直接转发到 LogicSvr.一般不用

  4) 统一登录消息组包:应用使用 Tconnd 预定义了通信协议头,根据协议头定义分割消息。

  5) 字符消息:根据特定的字符分割消息

  6) 字符串分包:根据特定的字符串分割消息

- Tconnd 提供统一登陆消息协议头,使用统一登陆协议头可以支持 QQ 号码方式(0x82 协议，统一登录协议)签名认证和通信加密。
- 支持消息排队机制,使用统一登陆消息协议,还可以定时通知客户端排队信息。

2.4 功能设计

- TConnd 前端支持 TCP/UDP 连接,使用 Epoll 方式处理网络连接。
- Tconnd 从 Socket 中收取数据,并根据消息组包策略将字节流分割成一个个逻辑业务包,放入 TBUS 通道。
- Tconnd 与 LogicSvr 使用 TBus 通信,从消息队列取出数据包通过 Tbus 发送到 LogicSvr,同时从 Tbus 中收取数据发送到客户端。
- Tconnd 与 LogicSvr 之间 tbus 通道中的消息封装了一个 tconnd 消息头部，logicsvr 不能直接使用 tbus 的接口收发数据，而必须使用与 tconnd 配套 tconnapi 进行数据收发。

2.5 内部定义

Listerner:监听器, 包括监听 IP 和端口等信息。

Serializer:序列化器,包括对端 LogicSvr 通道 IP 等信息。

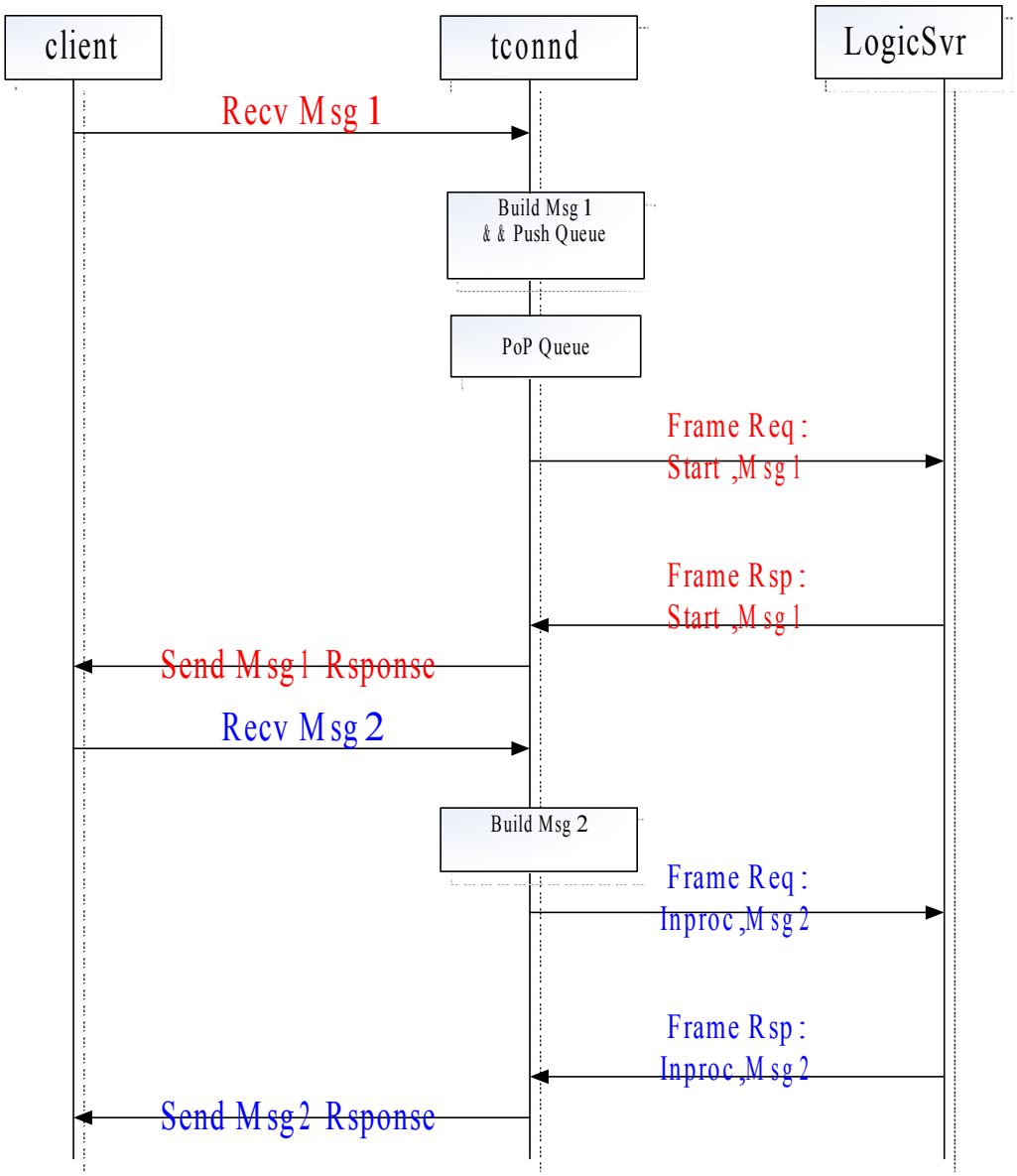
PDU:协议数据单元,描述消息协议及对应 TDR 文件。

NetTrans:网络传输通道,一个网络传输通道由 Listerner/Serializer/PDU 三方面的属性来定义。TCONND 支持多个 NetTrans 定义,在运行时生成多个实例,Nettrans 实例通过名字关联到 Listerner/Serializer/PDU,多个 Listerner 和多个 Serializer 可以定义成相同的名字。因此一个 NetTrans 可以对应多个 Listerner/Serializer 和一个 PDU.

3 交互图

3.1 直接通信

直接通信适用于二进制消息/文本消息情况。

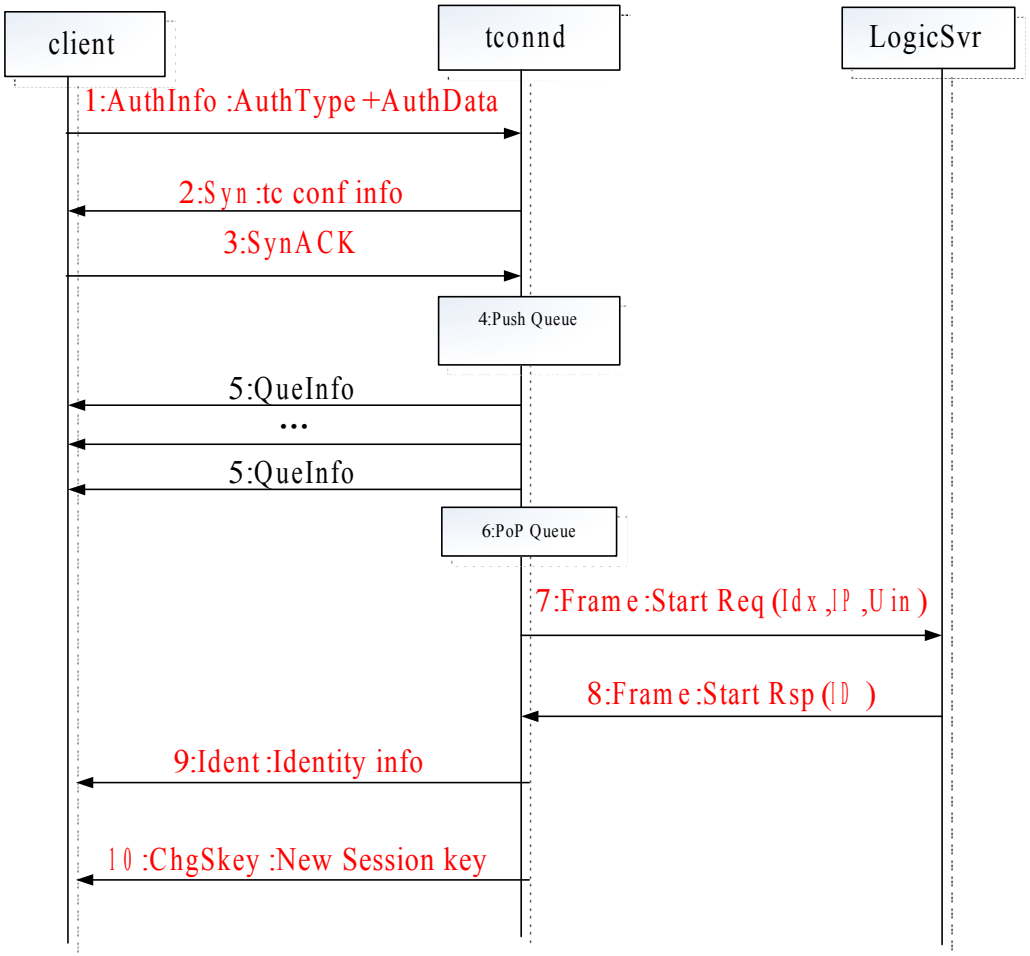


- Tconnd 从 accept 连接连接,根据分包策略组包并将连接放入队列
- Tconnd 根据服务器容量现在从队列中放出连接收取数据打包发送到 LogicSvr, 同时从 BUS 读取数据解包发送到 Client.
- 第一个数据包,Tconnd 会在 Frame 协议头中标识 Start,之后标识 Inproc。

3.2 加密通信

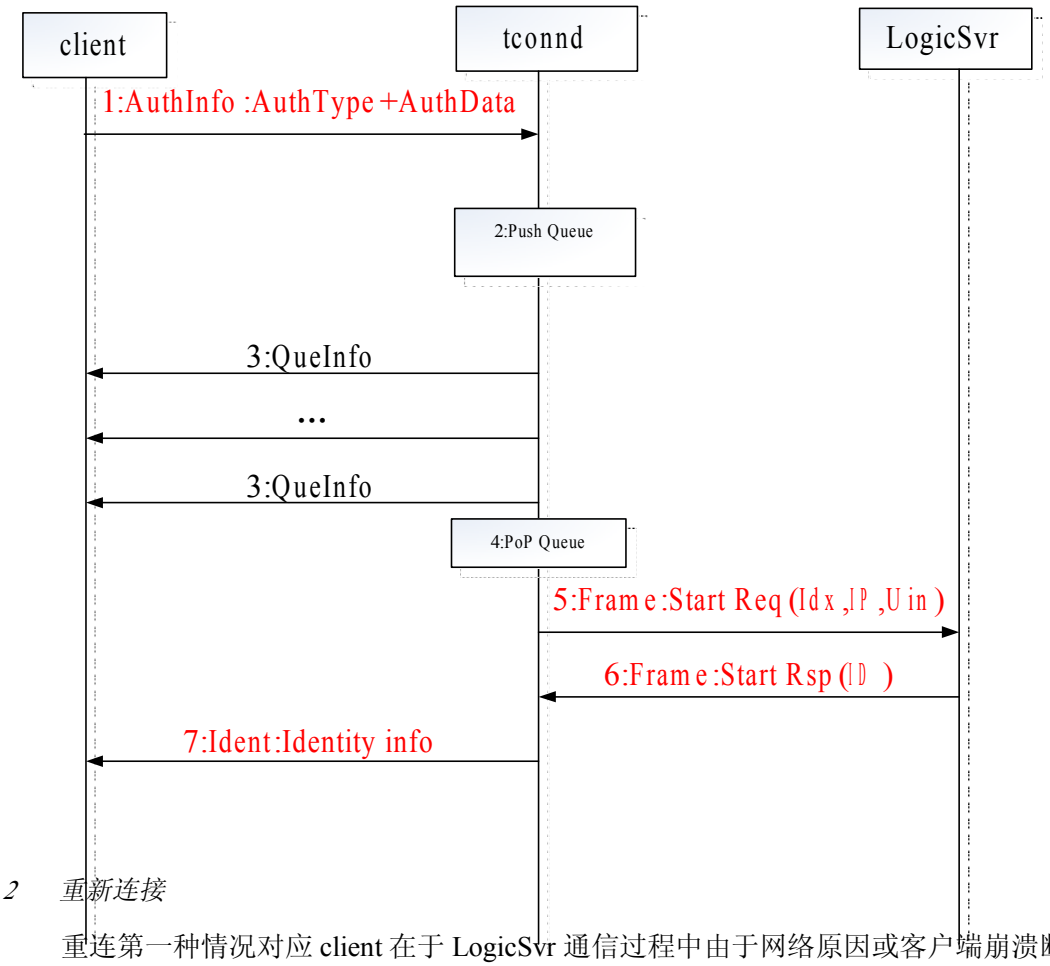
加密通信适用于统一登录消息类型。包括签名验证过程,加密通信过程,断线重连,关闭连接通知等情况。

3.2.1 签名签证过程



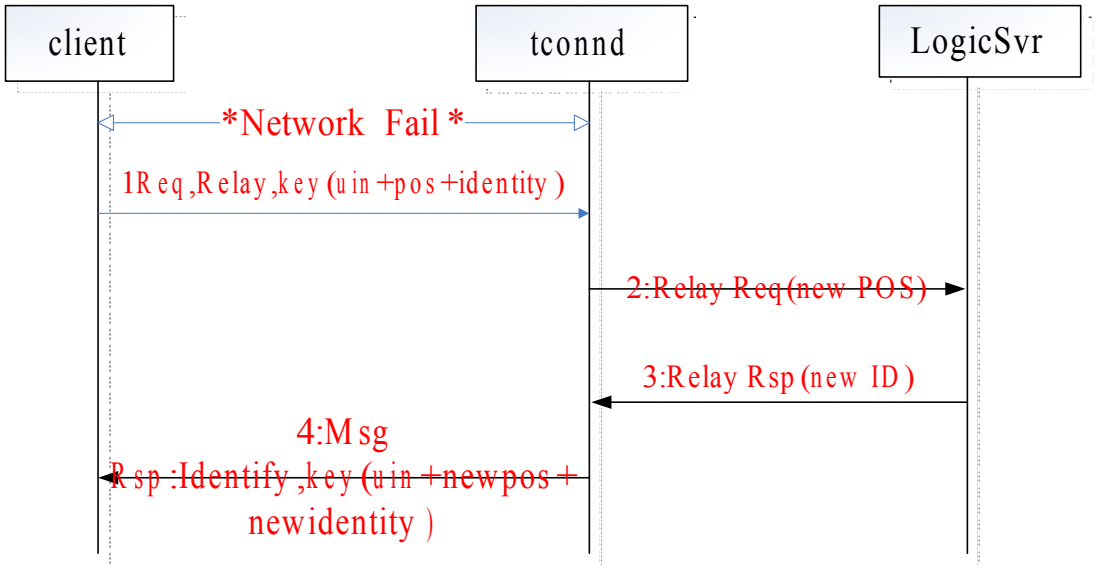
- 1:Client 发送带有 QQ 签名的包头信息,包括 session key 和签名信息，加密信息具体结构参考 tconnd 开发文档接口文档部分。只有包头信息。
- 2:若签名信息通过验证,则 tconnd 下发连接凭证握手信息,否则则断开连接.
- 3:客户端携带连接凭证请求三次握手,若连接信息一致,则建立连接成功。
- 4(tconnd 可配置): 若连接在排队中,通知 client 端排队信息。
- 5:排到了队首,出队列
- 6:向服务器发送连接建立包,包括连接索引信息,IP,端口 uin 等信息.
- 7.:服务器必须发送连接建立回应包,建议回应包此时置连接在 server 端的索引标识。
- 8.发送连接建立返回消息给客户端,包括连接凭证信息和连接索引,
- 9.发送通信新密钥信息到 client。通信信息需用使用新 session key 加密通信。

同时签名验证流程也支持如下省略步骤二,三和十的流程,需要将 tconnd 三次握手的配置置为 0,配置参考 tconnd 运维使用文档。



3.2.2 重新连接

重连第一种情况对应 client 在于 LogicSvr 通信过程中由于网络原因或客户端崩溃断开连接,可以使用预定义协议断线重连，并设置重连类型,重连的请求不需要排队可以直接跟 LogicSvr 直接通信。

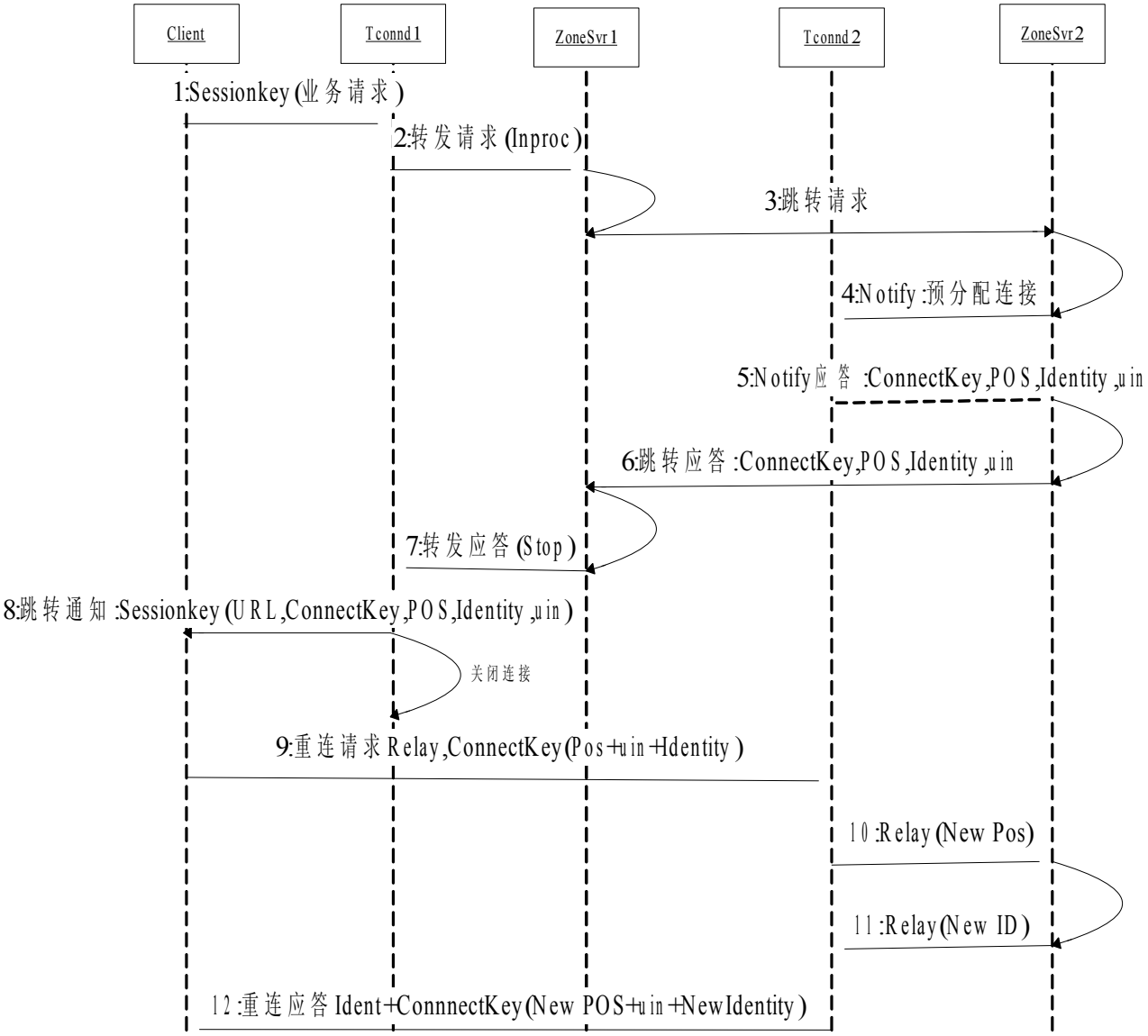


- 1:出现断线,客户端在 tconnd 释放连接前(不活动连接释放时间可以在 tconnd 配置)发送

断线重连请求

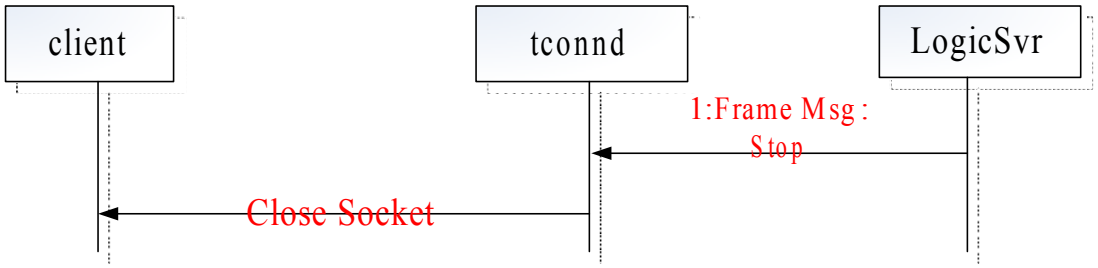
- 2:若上次连接信息尚未释放,且断线重连验证通过,向服务器发送断线重连通知.
- 3:服务端必须发送断线重连响应包.
- 4:通知客户端重连成功信息。

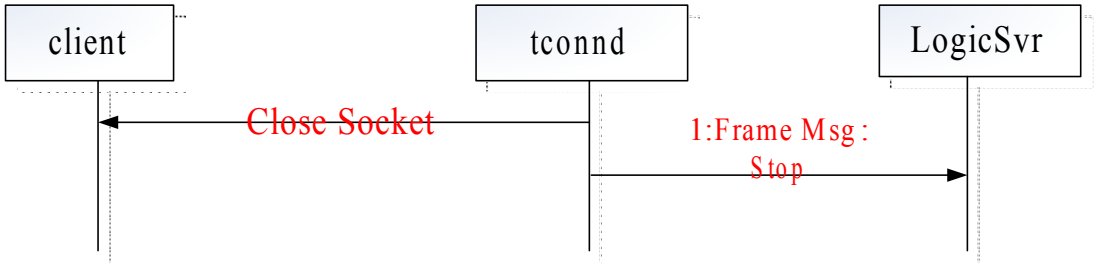
重连另外一种情况用于跨 tconnd，服务器之间的跳转,



3.2.3 释放连接

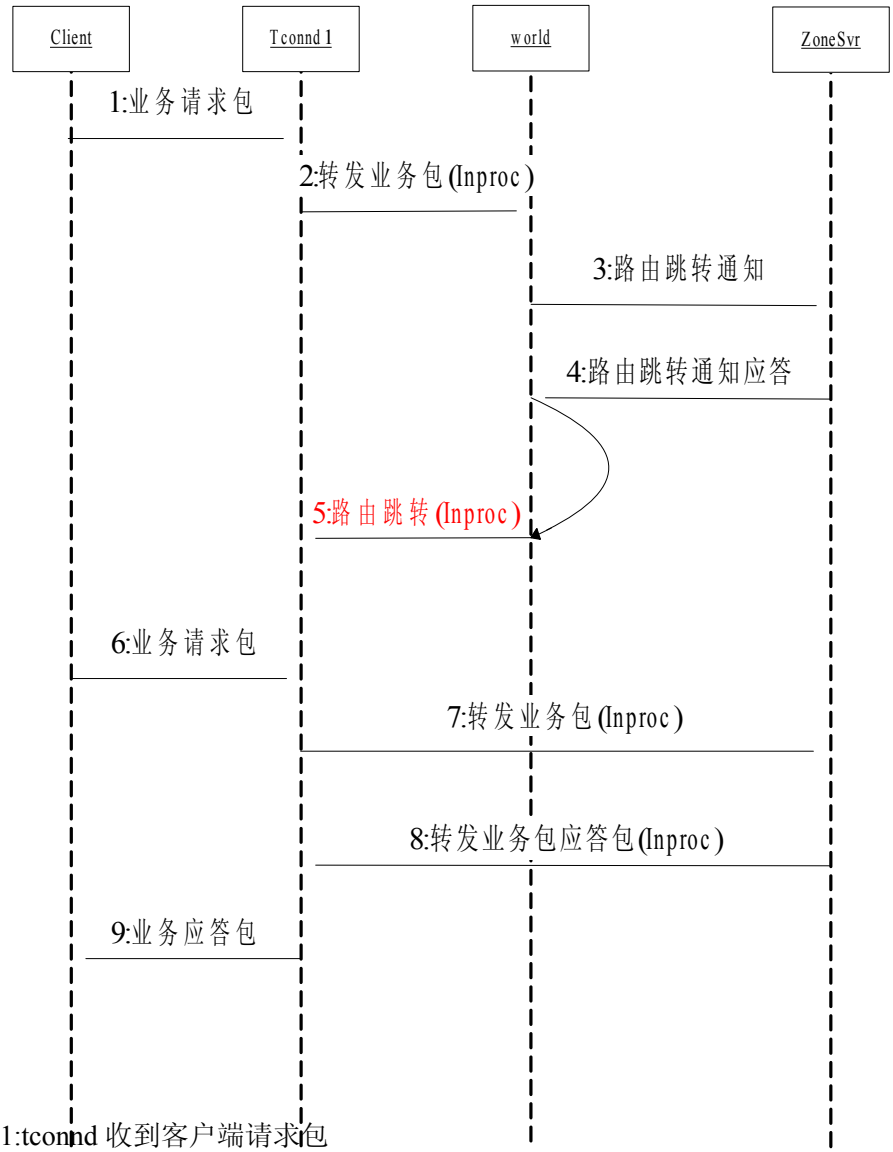
连接释放通知,有两种情况,一种是 LogicSvr 发送 Stop 包头通知 tconnd 关闭端口,释放连接,另一种情况是 tconnd 主动关闭连接,同时通知 LogicSvr.





3.2.4 路由跳转

路由跳转协议支持连接在通信过程中改变本 listerner 所对应的 Serializer,从而可以将客户端请求包路由到不同的服务器。路由跳转的场景如下:



1:tconnd 收到客户端请求包

2:转发请求到相应的服务器,

3:需要应用自定义协议;通知要跳转的服务器,并传递连接相关的信息

4：需要应用自定义协议,收到应答和服务器索引信息。

5:发送跳转指令到 tconnd

6-9:再次收到连接的业务包将转发到跳转的服务器.

4 协议定义

4.1 Client<->TConnd 协议定义

tconnd 与 Client 预定义包头结构如下,详情参考 tpdudef.h

struct tagTPDUFrame

```
{
    TPDUHEAD stHead;      //预定义包头
     char szBody[1];          //包体,应用可在这开始定义业务协议
};
struct tagTPDUHead
{
    TPDUBASE stBase;         //基本信息
    TPDUEXT stExt;            //扩展信息
};
struct tagTPDUBase
{
    unsigned char bMagic;                //
    unsigned char bVersion;              //版本号
    unsigned char bCmd;                  //扩展信息类型选择
//TPDU_CMD_None(0); 当 CMD 为此项时,扩展信息为空,通信包
//TPDU_CMD_CHGSKEY(1)：更改通信密钥
//TPDU_CMD_QUEINFO(2)：连接排队信息
//TPDU_CMD_AUTH(3);  签名验证信息
//TPDU_CMD_IDENT(4);  签名验证过程结束后返回的第一条信息
//TPDU_CMD_PLAIN(5); 当 CMD 为此项时,扩展信息为空,表示收到的这条消息
没有加密,只有下行包有该选项。
//TPDU_CMD_Relay(6); 客户端重连信息
//TPDU_CMD_Stop(7); 服务端断开连接信息，未用
//TPDU_CMD_SYN(8); 签名验证返回信息,下发 tconnd 配置信息
//TPDU_CMD_SYNACk(9); 签名验证请求握手信息
    unsigned char bHeadLen;              //预定义消息包头长度
    int iBodyLen;                        //应用协议长度
};
union tagTPDUExt
{
    TPDUEXTCHGSKEY stChgSkey;
    TPDUEXTQUEINFO   stQueInfo;
    TPDUEXTAUTHINFO stAuthInfo;
    TPDUEXTIDENT        stIdent;
    TPDUEXTRELAY       stRelay;
    TPDUEXTSTOP        stStop;
    TPDUEXTSYN stSyn;
```

```
    TPDUEXTSYNACK stSynAck;
};
```

4.1.1 签名验证请求消息

方向:client->tconnd

命令字:TPDUHEAD.TPDUBASE.cmd=0x03 (TPDU_CMD_AUTH)

作用:连接建立的第一条请求消息,只要求发送预定义包头。

请求信息体:

```
struct tagTPDUExtAuthInfo
{
    int iEncMethod;
    //0 不加密 2 tea 算法 3 taes 算法
    /*  通信加解密类型 1.8 版本增加,*/
    int iServiceID;
   /*游戏 id, 2.0 版本增加*/
   int iAuthType;          //必须跟 tconnd 协议数据单元里的配置验证方式一致
   //TSEC_AUTH_NONE=0:不需要验证签名信息
   //QQV1 = 1;   0x37 签名验证方式,tconnd 现在不支持 0x37 验证协议,如果发送的
   请求包该项为 1,tconnd 仍按照 0x82 的方式验证。,
   //QQV2 =2 ;    0x82 验证方式。
   //QQUnified=3;  统一登录方式          //目前一般用统一登录方式

   TPDUEXTAUTHDATA stAuthData;
};
union tagTPDUExtAuthData
{
    TQQAUTHINFO stAuthQQV1;  //iAuthType =    QQV1(1)
    TQQAUTHINFO stAuthQQV2;    //iAuthType =   QQV2(2)
    TQQUnifiedAuthInfo    stAuthQQUnified  //iAuthType = QQUnified(3)
};
struct TQQAuthInfo
{
    //Ox82 第一段结构加密后长度,,密钥对应 tconnd 配置的 svrkey
    unsigned char bSignLen;

    //加密 Buffer 缓存
    unsigned char  szSignData[TQQ_MAX_SIGN_LEN];
```

```
        //第二段结构加密后长度，密钥对应 tconnd 配置的 svrkey2,暂未用
        unsigned char bSign2Len;

        //第二段加密区 Buffer
        unsigned char szSign2Data[TQQ_MAX_SIGN2_LEN];
    }

    //Ox82 第一段加密区结构
    struct tagTQQGameSig
    {
        char szGameKey[TQQ_KEY_LEN];     //Session Key
        char    szSvcBitmap[TQQ_SVCBITMAP_LEN];
        char    szSvcBitmapExt[TQQ_SVCBITMAP_EXT_LEN];
        unsigned long ulValidateBitmap;
        long lUin;                                          //QQ 号码,
        unsigned long ulTime;                               // 签名时间戳
        unsigned long ulUinFlag;
        int iClientIP;                              /*   客户端 IP ,2.0 版本增加该字段*/
    };

    //第二段加密区结构,
    struct tagTQQSigForS2
    {
        char chType;
        unsigned long ulValidateBitmap;
        long lUin;                          //QQ 号码
        unsigned long ulTime;

        //签名的 IP，若 tconnd 配置了验证 IP 选项,且连接 client 的 IP 不等于该签名的 IP,   则
验证不能通过
        unsigned long ulCltIP;
    };

    //统一签名方式
    struct tagTQQUnifiedSig
    {
```

```
    short nVersion;
    int iTime;
    short nEncryptSignLen;
    unsigned char szEncryptSignData[TQQ_UNIFIED_MAX_ENCSIGN_LEN];
};

struct tagTQQUnifiedEncrySig
{
    int iRandom;
    short nVersion;
    unsigned int dwUin;
    int iTime;
    int iSSOver;
    int iAppClientVer;
    int iClientIP;
    unsigned char szSessionKey[TQQ_KEY_LEN];
    short nUnifiedSig2Len;
    unsigned char szUnifiedSig2[TQQ_UNIFIED_MAX_ENCSIGN2_LEN];
    short nCustomInfoLen;
    unsigned char szCustomInfoData[TQQ_UNIFIED_CUSTOMINFO_LEN];
};

struct tagTQQUnifiedAuthInfo
{
    long lUin;
    TQQUNIFIEDSIG stQQUinSig;
};
```

响应:用户连接信息，cmd=0x08 。见下

若验证未通过,tconnd 直接断开 Socket 连接.

4.1.2 签名验证应答消息

方向:tconnd->client

命令字:TPDUHEAD.TPDUBASE.cmd=0x08 (TPDU_CMD_SYN)

作用:签名验证通过返回消息,返回连接凭证信息。

扩展消息内容:（使用 SessionKey 加密)

struct tagTPDUExtSyn

```
{
    unsigned char bLen;
    char szEncryptInfo[TPDU_MAX_ENCRYPTSYN_LEN];
};

struct tagTPDUSynInfo
{
    char szRandstr[TQQ_IDENT_LEN];
};
```

4.1.3 连接握手消息

方向:client->tconnd

命令字:TPDUHEAD.TPDUBASE.cmd=0x00 (TPDU_CMD_SYNACK)

作用:请求连接握手消息,

扩展消息内容:（使用 SessionKey 加密) ,消息内容与 tconnd 下发验证应答消息一致.

```
struct tagTPDUExtSynAck
{
    unsigned char bLen;
    unsigned char szEncryptSynInfo[TPDU_MAX_ENCRYPTSYNACK_LEN];
};
```

响应:不需要响应.tconnd 收到此消息.则三次握手完成.向 server 发送连接成功消息..

4.1.4 用户连接应答信息

方向:tconnd->client

命令字:TPDUHEAD.TPDUBASE.cmd=0x04 (TPDU_CMD_IDENT)

作用:tconnd 收到 server 连接成功返回消息后,,返回用户连接信息。

扩展消息内容:（使用 SessionKey 加密)

```
struct tagTPDUExtIdent
{
    int iLen;                                   //用户连接信息加密后长度
    char szEncryptIdent[TPDU_MAX_ENCRYPTIDENT_LEN];   //加密内容
};

struct tagTPDUIdentInfo
```

```
{
    int iPos;                                   //索引
    char szIdent[TQQ_IDENT_LEN];        // 连接凭证
};
```

4.1.5 重连信息

方向:Client->tconnd

命令字:TPDU_CMD_RELAY = 6

作用:重新连接服务器

请求消息体:

```
struct tagTPDUExtRelay
{
    int iRelayType;             //重连类型  1:跳转  2：断线  3:crash 重连
    int    iOldPos                      //重连索引
    int iLen;                           //重连信息加密后长度
    char szEncryptIdent[TPDU_MAX_ENCRYPTIDENT_LEN];   //加密内容
};
    //连接信息结构
struct tagTQQUserIdent
{

    long lUin;                  //QQ 号码
    int iPos;                       //       用户连接在 tconnd 的索引,
    char szIdent[TQQ_IDENT_LEN];            //服务端生成的用户连接验证随机串
};
```

应答：cmd=04 参见 4.1.2

若重连验证失败,则直接断开连接

4.1.6 用户连接排队信息。

方向:tconnd->client

命令字:TPDUHEAD.TPDUBASE.cmd=0x02 (TPDU_CMD_QUEINFO)

作用:Tconnd 通知 Client 排队的情况

消息内容体内容:(没有加密)

```
struct tagTPDUExtQueInfo
{
    int iPos;
    int iMax;
```

};

请求字段:

- iPos: //在等待队列中的位置
- iMax: //等待队列的长度

响应:无

4.1.7 更改密钥信息

方向:tconnd->client

命令字:TPDUHEAD.TPDUBASE.cmd=0x01 (TPDU_CMD_CHGSKEY)

作用:Tconnd 通知 Client 更改通信的密钥(使用旧 Sessionkey 加密),新密钥的长度为 16 个字节。tconnd 下发连接应答消息之后会下发更改密钥信息.

消息体:

```
struct tagTPDUExtChgSkey
{
    short nType;
    short nLen;
    char szEncryptSkey[TPDU_MAX_ENCRYPTSKEY_LEN];
};
```

请求字段:

- nType:置为 0
- nLen://使用原 sessionkey 加密新密钥后 buffer 的长度
- szEncryptSkey: //密文 buffer

响应:无

4.1.8 通信信息

方向:tconnd<->client

命令字:TPDUHEAD.TPDUBASE.cmd=0x00 (TPDU_CMD_None)

作用:业务通信包,无扩展包头信息。

另外:

4.1.9 未加密下行包

方向:tconnd->client

命令字:cmd=0x05(TPDU_CMD_Plain)

作用:表示该下行消息体没有加密。

4.2 TConnd<->LogicSvr 协议定义

Tconnd 和 logicSvr 通信之间封装了一个包头,详情可以参考(tframehead.h)

```
struct tagtframehead
{
```

char chVer; //版本信息,默认为零

char chCmd; //命令子,消息类型

char chExtraType; //是否带 IP 信息

char chTimeStampType; //是否带时间戳信息

int iID; //逻辑服务器索引,只读,下行组播包可以及控制包(heartbeat,setconnlimit,notify 可以忽略该字段

int iConnIdx; //连接在 tconnd 的索引,下行组播包及控制包(heartbeat,setconnlimit,notify）可以忽略该字段

TFRAMEHEADDATA stExtraInfo; //IP 信息

TTIMESTAMPDATA stTimeStamp; //时间戳信息

TFRAMECMDDATA stCmdData; //消息内容

};

- chVer:版本信息:默认填零
- chCmd;命令字,消息类型

TFRAMEHEAD_CMD_START" value="0" //连接建立包

TFRAMEHEAD_CMD_STOP" value="1" //连接断开包

TFRAMEHEAD_CMD_INPROC" value="2" //业务通信包

TFRAMEHEAD_CMD_RELAY" value="3" //断线重连包

TFRAMEHEAD_CMD_NOTIFY" value="4"//跳转预分配连接

TFRAMEHEAD_CMD_SETROUTING value="5" //设置路由包

TFRAMEHEAD_CMD_RSP_SETROUTING" value="6" //设置路由回应包

TFRAMEHEAD_CMD_ADJUSTCONNLIMIT" value="7" //设置连接限制

TFRAMEHEAD_CMD_RSP_ADJUSTCONNLIMIT" value="8" //设置连接限制回应包

TFRAMEHEAD_CMD_HEARTBEAT = 9, /* 服务器检查 tconnd 心跳包,2.0 版本增加 */

- ExtraType:附加信息选项

"TFRAMEHEAD_EXTRA_IP" value="1" 当 ExtraType=1 时,stExtraInfo 为 IP 信息,其它值时 stExtraInfo 为空,tconnd 发送 start 包和 relay 上行包的时候会带上客户端 IP 信息,如果对应的下行返回包置了包头 ID 索引字段,则后续 tconnd 发送 INPROC 上行包不再带 IP 信息,否则 Tconnd 发送 INPROC 请求包会携带连接客户端 IP 信息,此时要求 logicsvr 下行返回包也需要携带连接客户端 IP 信息,否则 tconnd 发送下行包不会验证通过。

- chTimeStampType:时间戳信息， 该选项由 tconnd 配置。

TFRAMEHEAD_TIMESTAMP_TIMEVAL =1,当 chTimeStampType=1,stTimeStamp 保存 Tconnd 收到客户端上行包时间戳信息,否则 stTimeStamp 结构为空。

- iID:客户端连接在 LogicSvr 的索引,tconnd 发送给 LogicSvr 初始值为-1.一般 Svr 在

start 包和 relay 返回下行包中需要置该字段并保存该字段为客户端连接在 server 的索引,如果没有置该字段,则返回给 tconnd 包头需要带客户端 IP 信息。对于组播包和控制包(heartbeart,notify,setconnlimit)可以忽略该字段。

- iConnIdx:用户连接在 Tconnd 的索引,该字段只读.tconnd 发送给 LogicSvr 的上行包中会携带该字段,对于组播包和控制包(heartbeart,notify,setconnlimit)可以忽略该字段。
- stExtraInfo:附加信息
- stTimeStamp:时间戳信息
- stCmdData:消息内容

消息结构:

```
union tagTFrameCmdData
{
TFRAMECMDSTARTstStart;       //TFRAMEHEAD_CMD_START,
TFRAMECMDSTOP stStop;       // TFRAMEHEAD_CMD_STOP,
TFRAMECMDINPROC stInProc;       //TFRAMEHEAD_CMD_INPROC,
TFRAMECMDRELAY stRelay;     //TFRAMEHEAD_CMD_RELAY,
TFRAMECMDSTOP stNotify;              //TFRAMEHEAD_CMD_NOTIFY,
TFRAMECMDSETROUTING stSetRouting;
  /* TFRAMEHEAD_CMD_SETROUTING,     1.8 版本增加*/
    TFRAMECMDSETROUTINGRSP stSetRoutingRsp;
  /* TFRAMEHEAD_CMD_RSP_SETROUTING,   1.8 版本增加    */
    TFRAMECMDSETCONNLIMIT stSetConnLimit;
    /* TFRAMEHEAD_CMD_SETCONNLIMIT,     */
    TFRAMECMDSETCONNLIMITRSP stSetConnLimitRsp;
    /* TFRAMEHEAD_CMD_RSP_SETCONNLIMIT,   1.8 版本增加  */
};
```

4.2.1 连接建立消息

方向:tconnd-->LogicSvr

命令字:chCmd= 0x00,

作用:tconnd 和 logicsvr 交互的第一次请求和应答消息。

响应:同请求包,一般需更改包头的 ID 字段,chCmd= 0x00,LogicSvr->tconnd

消息体内容:

```
struct tagTFrameCmdStart
{
    //tconnd 验证方式
    int iAuthType;
    / /TSEC_AUTH_NONE=0:不验证签名信息
```

```
        //QQV1 = 1;     0x37 签名验证方式,tconnd 现在不支持 0x37 验证协议,如果发送的
        请求包该项为 1,则 tconnd 仍按照 0x82 的方式验证。,
        //QQV2 =2 ;      0x82 验证方式。
        //QQUnified=3,   统一登录方式
        TFRAMEAUTHDATA stAuthData;
        int iWaitTime;          //连接排队时间
    };
    union tagTFrameAuthData
    {
    TFRAMEAUTHQQ stAuthQQV1;            /* TSEC_AUTH_QQV1,     */
    TFRAMEAUTHQQ stAuthQQV2;            /* TSEC_AUTH_QQV2,     */
    TFRAMEAUTHQQ stAuthQQUnified        /* TSEC_AUTH_QQUnified,     */
    };
    struct tagTFrameAuthQQ
    {
        long lUin;
    };
```

请求字段:

- Uin:QQ 号码

返回包字段::一般只需更改包头的 ID 字段,

4.2.2 重连消息

方向:tconnd->LogicSvr

命令字::chCmd= 0x03

作用:通知 LogicSvr 用户重新连接

响应:同请求包,一般需更改包头的 ID 字段,chCmd= 0x03

消息体内容:

```
    struct tagTFrameCmdRelay
    {
        long lUin;
        int iRelayType;                 //重连类型: 1:跳转  2：断线  3:crash 重连

        TFRAMEIDENT stOld;          //原连接信息
        TFRAMEIDENT stNew;          //新连接信息
    };
    struct tagTFrameIdent
    {
```

```
    int iID;                //用户在 LogicSvr 的索引
    int iConnIdx;           //用户在 tconnd 的  连接索引
};
```

请求包字段:

- Uin：QQ 号码
- stOld:原连接信息
  - iID:一般为-1 的情况为跨服跳转,其它应该为断线重连的情况
  - iConndidx：原来的连接索引.
- stNew:新的连接信息。
  - iID:为-1
  - iConndidx: 新的连接索引,更请求包头里 idx 字段为一致。

返回包字段: 返回包只需要在包头置新的 ID,不需要更改消息体的内容。

4.2.3 连接断开消息

方向:tconnd<->LogicSvr

命令字::chCmd= 0x03

作用:通知用户断开连接

响应:无,消息可以是 tconnd 通知 LogicSvr 也可以 LogicSvr 到 tconnnd。

消息体内容:

```
struct tagTFrameCmdStop
{
    int iReason;
};
```

返回包:无需返回

4.2.4 跳转预分配连接

方向:LogicSvr->tconnd

命令字:chCmd = 0x04;

作用:为跳转预分配连接

响应:同请求包 chCmd = 0x04

消息体内容:

```
    struct tagTFrameCmdStop
    {
    Long    uin;                //请求包只需要置 uin
    Int     connidx;
    Char    Szkey[16]
    Char    szidentity[16];
    unsigned char bISetRouting;     //是否设置路由的服务器名,一般需要
```

TROUTINGOPTION stRouting; //bISetRouting 非零时有效,指定路由的 serlizername

};

请求包字段:

- Uin：预分配连接 Uin 号
- bISetRouting:路由指示
- stRouting:路由信息,结构参考 4.2.6 节内容

返回包字段

- Conidx：连接索引,同时在包头也会值该字段。
- Szkey：重连密钥
- Szidentity：重连凭证。

4.2.5 通信包

方向:tconnd->LogicSvr

命令字::chCmd= 0x02

作用:业务通信包

响应:同请求包 chCmd = 0x04

消息体内容:该结构内容只有返回包即(LogicSvr->tconnd)才有意义

```
struct tagTFrameCmdInProc
{
    char chValid;
    char chNoEnc;
    short nCount;
    TFRAMEIDENT    astIdents[TFRAMEHEAD_MAX_BATCH_IDENT];
};
```

请求消息字段::

Tconnd 发送通信请求包时,该结构中的信息无意义

返回消息体:定义同请求包

返回消息字段:.

- chValid:当 LogicSvr 需要使用业务组播消息时,该项须置为一,此时 tconnd 会使用组播方式发送。若该字段不为一。则 tconnd 作单播处理.
- chNoEnc:若该项为真时,表示该下行消息 tconnd 不需要加密
- ncount：用户信息数组个数,当 chValid 字段为一即组播方式时有效,组播连接个数
- astIdents[TFRAMEHEAD_MAX_BATCH_IDENT];用户信息数组,单播的时候可以不填。

4.2.6 设置路由信息

方向:LogicSvr->tconnd

命令字::chCmd= 0x05

作用:设置连接发送包目的地

响应:chCmd= 0x06

消息体内容:

```
struct tagTFrameCmdSetRouting
{
    char szSerializerName[SERLIZER_NAME_LEN];   //序列器名,对应逻辑服务器
    int iID;                                    /*    连接在跳转服务端上的索引, */
};
```

响应内容:

```
struct tagTFrameCmdSetRoutingRsp
{
    int iResult;                                /*零为成功,其它为错误码*/
};
```

4.2.7 设置连接限制信息

方向:LogicSvr->tconnd

命令字::chCmd= 0x07

作用:设置传输连接限制

响应:chCmd= 0x05

消息体内容:

```
struct tagTFrameCmdSetRouting
{
    char szSerializerName[SERLIZER_NAME_LEN];   //序列器名,对应逻辑服务器
    int iID;                                    /*    连接在跳转服务端上的索引, */
};
```

响应内容:

```
struct tagTFrameCmdSetRoutingRsp
{
    int iResult;                                /*零为成功,其它为错误码*/
```

};

4.2.8 心跳协议 tconnd<->server

包体内容为空

5 版本变更

5.1 TSF4G_TCONND_02_0000

- Tconnd 与服务器直接增加心跳协议

5.2 TSF4G_TCONND_01_0008

- Tconnd 与服务器之间增加设置连接跳转路由协议
- tconnd 与服务器之间增加设置连接限制协议