**Nppclient概要设计**

**目前Ngi和CSS使用osp进行交互，需改成MQ，所以为NGI封装Nppclient实现MQ通信。**

**Nppclient提供给NGI的接口：**

**1、void nppclient\_init(const s8 \*pchMqIp, const u16 wPort)**

接口操作：

MQ初始化，创建消费者，生产者，以及创建对应异常/正常的处理函数。

nppClient侧的MQ消费者 EX: ngi.nppngi.ex

K: ngi.nppngi.k:时间戳

Q: ngi.nppngi.q:时间戳

NPP侧的MQ消费者 EX：npp.nginpp.ex

K: npp.nginpp.k

Q：npp.nginpp.q

Nppclient和NPP之间走的是MQ的RPC消息,后面的时间戳来区分回给哪个q，使消息投递给不同的ngi。RPC消息的rpcid,以及q存储在TNppReqHeadInfo里面.

设置SIP终端的消息处理回调函数，Sip终端发过来的消息类型为NU\_MAU\_TRANSPARENT\_CHANNEL\_REQ, 通过此回调函数将对不同消息类型做不同的处理。

注册打印；

1. **BOOL32 SendMsgToNpp(void \*pBuf, u32 dwBufLen, const u16 wMsgType)**

**目前pBuf内容主要为TPasConfMsg, Ngi以此接口发送请求消息，消息发送成功返回1，失败为0。**

目前以的css中的nussn代替NPP，所以nussn需要创建生产者消费者。

**2.1 终端召集会议**

**Nppclient与NPP交互走MQRPC通信**

**Nppclient内容：**

**·获取终端类型；**

**·获取TCreateConfReq信息；**

**·完善reqinfo;**

消息号：NU\_MAU\_CREATECONF\_REQ

NPPclient将pBuf内容取出，并获取部分会议信息,转为JSON发送给CSS

发送的JSON字段:

TConfInfo、TConfInfoEx、TNppReqHeadInfo、TVmpModuleInfo、TAutorecAttrb、TMiniPollInfo、TMiniVipInfo、TMixModule、”mt”:

{

"pollparam": {},

"dualmode": "1",

"hdumodule": "0",

"audioformat": [ ],

"speakersrc": "0",

"confname": "1234560000091鐨勪細璁�",

"resolution": "32",

"isportconf": "0",

"encryptkey": "",

"hasxmpu": "0",

"duration": "240",

"chairmanalias": "",

"cascadeconf": {},

"mt": [],

"reqheadinfo": { },

"confE164": "6560026",

"encryptmode": "0",

"confid": "00000000000000000000000000000000",

"psw": "",

"speakeralias": "",

"pollmodule": "1",

"creatorname": "1234560000091",

"allsilence": "0",

"moid": "5r4sr4dtntocdtyesk7noksp",

"enckeymanual": "1",

"domainname": "涓婄爺鎵€",

"bandwidth": "1024",

"vipmt": [],

"mixmode": "1",

"vacinterval": "0",

"type": "MAU\_MCU\_CREATECONF\_REQ",

"allmute": "0",

"callmode": "0",

"callinterval": "0",

"calltimes": "0",

"sat": "0",

"vmpmodule": "1",

"publicconf": "0",

"openmode": "1",

"nodisturb": "0",

"videoprior": "0",

"mixmodule": "1",

"starttime": "2016-08-19 17:11:02",

"endtime": "1970-01-01 12:00:00",

"mixparam": {},

"maxjoinedmt": "192",

"onereforming": "0",

"videoformat": [],

"secvideoformat": [ ],

"vmpparam": { },

"autorecattrb": {},

"isrecdstream": "0"

}

**CSS侧内容：**

·终端权限域权限的判断（CheckAndGetMtDomain）；

获取到域名，**MOID**之后需要设置到JSON字段中。（Tconfinfo，ReqInfo中需要）

·更新创会参数（updateCreateConfParam）；

需要将更新后的数据设置到JSON中。

·向VC发送终端创会消息，VC部分不变。

**召集会议ACK和NACK**

CSS发往nppclient JSON：

{

"type": "MAU\_NU\_CREATECONF\_ACK/NACk”,

"nppreqheadinfo": {}

}

Nppclient收到JSON字段之后，将JOSN字段转为二进制数据，通过回调函数发送给NGI。

**2.2终端加入会议**

**同样Nppclient与npp交互走MQrpc消息。**

NPPclient侧内容：

·获取TjoinConfReq；

·构造TreqHeadInfo;

Nppclient发送给css的JSON字段：

{

"type": "NU\_MAU\_JOINCONF\_REQ",

"nppreqheadinfo": { }

}

CSS侧内容：

·终端权限域权限的判断；（CheckAndGetMtDomian）；

获取到域名，MOID之后需要设置到JSON字段中；（Reqinfo中）

·CSS在nussn中需要判断终端入会行为（加入已召开会议/创预约会议/创公共模板）

·如果是加入已召开会议，直接向cmu发送入会请求,使用css现有逻辑

发往cmussn的JSON字段{

"type": "MAU\_MCU\_ADDMT\_REQ",

"addtype": "",

"nppreqheadinfo": {}

}

·如果是终端创建预约会，使用现有逻辑，nussn在redis上查找是否存在预约会议，存在则向VC发送创建预约会议消息：

**消息号**：MAU\_MAU\_ MTCREATECONFBYBOOKCONF \_REQ

**消息内容**：nppreqheadinfo

·如果是终端创建公共模板，使用现有逻辑，nussn在获取到公共模板信息之后，直接向vc发送创公共模板消息：

**消息号**：MAU\_MAU\_ MTCREATECONFBYPUBLICTEMP \_REQ

**消息内容**：nppreqheadinfo

CSS回复nppclient的JSON字段：（仍使用当前逻辑）

{

"type": "MAU\_NU\_JOINCONF\_ACK/NACk”,

" nppreqheadinfo ": {}

}

nppclient收到JSON字段之后，改写为二进制数据，通过回调函数发送给NGI。

**2.3 级联呼叫个人模板**

**Nppclient与npp交互走MQrpc消息。**

**nppclient侧内容：**

**·构造TNppReqHeadInfo;（将TPasConfMsg内容取出）**

**发送给CSS的JSON：**

{

"type": "NU\_MAU\_CREATECONFBYTEMP\_REQ",

" nppreqheadinfo ": { }

}

**CSS（nussn）侧内容：**

**·查找个人模板中是否存在该模板；**

**·完善TReqHeadInfo，并加入请求MAP列表中;**

**·域信息判断；**

**·设置做大入会终端数到tconfinfo；**

发送创会信息到VC，VC部分不动。

**CSS级联呼叫ACK/NACK 发送给**nppclient**JSON：**

{

"type": "MAU\_NU\_CREATECONFBYTEMP\_ACK/NACk”,

" nppreqheadinfo ": {}

}

nppclient收到之后将nppreqheadinfo转为TPasConfMsg发送给NGI。

**2.4 终端获取会议列表（即时/预约/公共模板）**

**获取会议列表,Nppclient与npp交互走MQ消息，但不走RPC，因为除sip终端之外的其他终端消息回复类型为ntf，在会议列表的回复中需要带上nppreqheadinfo来获取信息确认需要回复给哪个终端。**

nppclient侧内容:

·构造tNppReqHeadInfo(主要内容包括：ProctocolType,ProductId,VersionId,E164,mtreqid)

发送给CSS的JSON字段

{

"type": "NU\_MAU\_GETCONFLIST\_REQ/NU\_MAU\_GETTEMPLATELIST/NU\_MAU\_GETBOOKLIST\_REQ",

“getall”:”1”, CV25M终端独有.其他终端为0

" nppreqheadinfo ": { }

}

CSS侧内容：保持现有逻辑

·获取终端型号；

·终端信息域信息校验；

CSS从redis上获取到即时会议列表，预约会议列表的JSON字段，六个一包发往nppclient；CSS需要将从Map中获取到的公共模板列表转为JSON字段并六个一包发给nppclient。再由nppclient转为二进制数据发给NGI

获取所有会议列表时，CSS将所有列表转为JOSN并六个一包发送给nppclient。

JOSN字段

**{**

**“type”:”** NU\_MAU\_GETCONFLIST\_NTF**”,**

" nppreqheadinfo ": { },

“conflist”:””

**}**

**2.5 获取会议详情信息**

**Nppclient与npp交互走MQrpc消息。**

NPPclient侧内容:

·构造tNppReqHeadInfo(主要内容包括：ProctocolType,ProductId,VersionId,E164,mtreqid)

发送给CSS的JSON字段

{

"type": "NU\_MAU\_GETCONFDETAIL\_REQ",

" nppreqheadinfo ": { }

“confdetailtype:”:”1”, 1为即时会议，2为预约会议，3为模板

}

CSS侧内容：保持现有逻辑

·终端信息域信息校验；

npp回复nppclient JSON:

{

"type": "NU\_MAU\_GETCONFDETAIL\_ACK",

" nppreqheadinfo ": { }

“confdetail”:””

}

CSS将获取到的会议信息TconfDetailInfo设置到JOSN中，发送给nppclient,nppclient转为二进制数据发送给NGI。

**2.6 获取最大/可用会议数**

**Nppclient与npp交互走MQrpc消息。**

NPP侧内容:

·构造tReqHeadInfo(主要内容包括：ProctocolType,ProductId,VersionId,E164,mtreqid)

发送给CSS的JSON字段

{

"type": "NU\_MAU\_GETCONFNUM\_REQ",

" nppreqheadinfo ": { }

}

CSS侧内容:

·保持现有逻辑

**CSS获取到最大可用会议数设置到JOSN字段中，发送给nppclient**

**{**

"type": "NU\_MAU\_GETCONFNUM\_ACK",

" nppreqheadinfo ": { }

“maxlicensedconfnum”:””,

“licensedconfnum”:””

**}**

**备注：对于sip终端的消息，消息号为NU\_MAU\_TRANSPARENT\_CHANNEL\_REQ/ACK/NACK,请求REQ信息，通过回调函数对不同的消息做出处理，响应回复消息（ACK/NACK）由nppclient在处理为sip终端时，将回复以上消息号。**

1. **设置消息响应回调函数接口**

**nppclient设置一个回调函数typedef void(\*nppclient\_callback)(void \*pBuf, u32 dwBufLen, const u16 wEvent) 目前pBuf内容主要为TPasConfMsg**

**提供一个设置回调函数接口给NGI,该回调函数用于npp回复消息给ngi：**

**void nppclient\_set\_callback(nppclient\_callback cb);**

1. **返初始化函数**

**Void Nppclient\_finit();**

**接口功能：程序退出释放资源。主要有mq退出函数等。**

1. **CNppClientMqStatus GetNppMqStatus() nppclient获取MQ消费者,生产者创建状态。**

**返回nppclient的生产者和消费者的状态。**

1. **设置断链，连接成功回调函数**

**typedef void(\*nppclient\_connectcb)(BOOL32 bConnected);**

**void nppclient\_set\_connectCB(nppclient\_connectcb cb);**

**参数为1，成功建链；为0，断链；**

**交互时序图：**

