**Internal Push Service技术简介**

**V1.0**

目录

[1. 架构概述 3](#_Toc5282128)

[2. 架构详细设计 3](#_Toc5282129)

[2.1 FPS 3](#_Toc5282130)

[2.2 MEC 6](#_Toc5282131)

# 架构概述

Internal Push Service部分功能主要包括两方面内容，对应到源码包中，对应fps和mec两包代码内容，分别简介如下：

FPS（Free Push Service）：基于开源项目mosquito进行二次开发的服务端程序，通讯协议采用IBM提出的MQTT消息传输协议。主要用于服务端与客户端的长链接。

MEC（Message Exchange Center）：主要负责VoIP系统中的消息交换，如新通话（NCL）、新消息（MMS）等。

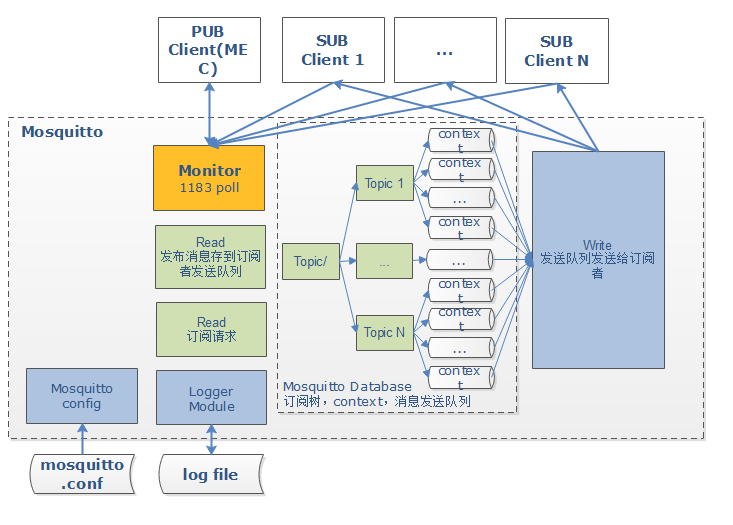
下面章节进行详细说明。

# 2. 架构详细设计

## 2.1 FPS

FPS负责推送通知给手机端用户，相应的手机端程序中，包含了支持MQTT协议的客户端程序（已经包含在so库中，使用c库方式进行封装），它是基于开源mosquito项目开发，通讯协议使用IBM的MQTT协议。

技术组成如下：



* PUB Client（MEC）：MEC连接FPS的客户端，MEC作为MQTT的发布者角色；
* SUB Client：对应手机客户端，作为MQTT的订阅者角色；
* Monitor：FPS进程启动监听端口，处理来自MEC及客户端的长链接；

### 2.1.1 报文格式

报文详细请参阅网上MQTT标准协议格式，下文简述系统中较关注的部分

|  |
| --- |
| Fixed header |
| Variable header |
| Payload |

Fix header：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte 1 | Message type | | | | DUP flag | Qos level | | RETAIN |
| Byte 2 | Remaining Length | | | | | | | |

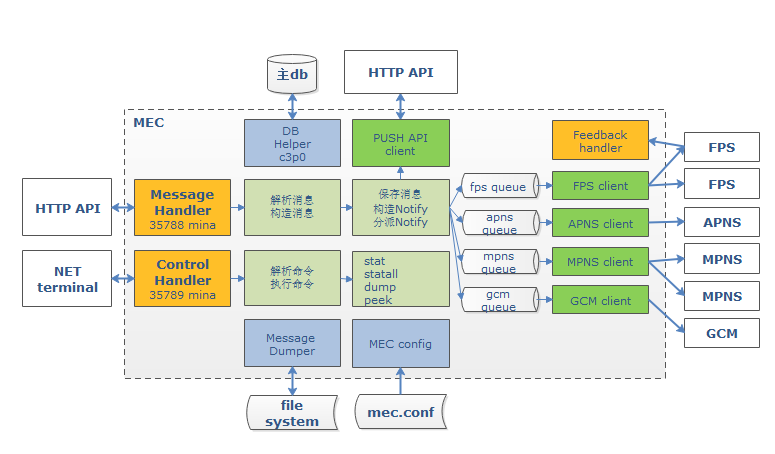
Message Type：

|  |  |  |
| --- | --- | --- |
| **Mnemonic** | **Enumeration** | **Description** |
| Reserved | 0 | Reserved |
| CONNECT | 1 | Client request to connect to Server |
| CONNACK | 2 | Connect Acknowledgment |
| PUBLISH | 3 | Publish message |
| PUBACK | 4 | Publish Acknowledgment |
| PUBREC | 5 | Publish Received (assured delivery part 1) |
| PUBREL | 6 | Publish Release (assured delivery part 2) |
| PUBCOMP | 7 | Publish Complete (assured delivery part 3) |
| SUBSCRIBE | 8 | Client Subscribe request |
| SUBACK | 9 | Subscribe Acknowledgment |
| UNSUBSCRIBE | 10 | Client Unsubscribe request |
| UNSUBACK | 11 | Unsubscribe Acknowledgment |
| PINGREQ | 12 | PING Request |
| PINGRESP | 13 | PING Response |
| DISCONNECT | 14 | Client is Disconnecting |
| Reserved | 15 | Reserved |

...

## 2.2 MEC

MEC主要负责VoIP系统中的消息交换部分，组成如下：



* HTTP API：负责接收来自客户端的推送消息，接口为HTTP，它将接受到的消息，传送给MEC模块；
* Net Terminal：命令行方式（telnet）访问MEC，可进行相关操作
* Message Handler：负责消息通知的分发，和消息体的存储，接口为TCP，基本的操作逻辑有：

send：首先将收到的消息放入Message Center中，同时生成一条通知消息，通过APNS/FPS/FCM等推送服务，推送给各平台客户端，通知各平台来拉取消息；

get：从Message Center中得到某客户的所有消息（可配置消息过期策略，默认保存15天）；

ack：确认消息，确认过的消息，从Message Center中进行物理删除。

* DB Helper：使用c3p0连接池处理DB事务，会根据相关表进行MEC的配置查询等（base\_config/server\_config/user\_register\_info/fps\_info\_table/cs\_session\_table等表）；
* Message Dumper：MEC重启时会将当前存储的消息放到dump文件，重启之后从dump文件中恢复存储的消息。
* PUSH API Client：通过调用HTTP API接口(push\_message\_from\_server.php)，推送消息。
* XXX Client：为每种Push Service的Client端，负责将消息推送到相应的Push Service中。

### 2.2.1 消息推送流程

### 2.2.2 MEC API与MEC通信协议

MEC API模块与MEC Dispatcher之间使用TCP进行通信，采用文本格式的协议：

1. 命令/响应均放到同一行中，并且以“\n”结尾
2. 每个部分用空格分隔

### 2.2.3 发送消息（SND）

命令格式：

|  |
| --- |
| SND SENDER\_ID SENDER\_NAME DEST\_MID PUSH\_TO APPNAME PROPERTIES TTL TYPE MESSAGE CAPABILITY |

字段解释：

|  |  |  |
| --- | --- | --- |
| **方向** | **字段** | **格式和含义** |
| 发送 | SND | 表示这个一个SND（发送消息）指令 |
| SENDER\_ID | 发送者的唯一标识 |
| SENDER\_NAME | 暂时未用到 |
| DEST\_MID | 接收端MID，一次一个 |
| PUSH\_TO | 发送到哪种Push Service中。  1：APNS  2：C2DM  4：FPS（目前系统只有这种方式推送）  如果=0，则消息会被丢弃，并且返回SUCCESS。 |
| APPNAME | 应用程序名称，字符串格式。固定ipphone |
| PROPERTIES | 消息附加的信息，json 格式的字符串。如果没有附加消息，则填："{}" |
| TTL | 消息保存时间，单位为秒，如300即为保持5分钟 |
| TYPE | 消息类型，如 NCL、MMS 等，字符串类型 |
| MESSAGE | 消息正文，注意不能有空格，不能有\n |
| CAPABILITY | 接收者的capability，可以为空 |
| 返回值 | CODE | 返回值代码  0 SUCCESS （成功）  11000 INVALID\_COMMAND （命令格式不对） |
|  | DESCRIPTION | 返回值的描述 |

### 2.2.4 获取消息（GET）

命令格式：

|  |
| --- |
| GET MY\_MID APPNAME |

字段解释：

|  |  |  |
| --- | --- | --- |
| **方向** | **字段** | **格式和含义** |
| 发送 | GET | 表示这个一个GET（获取消息）指令 |
| MY\_MID | 获取消息者的MID |
| APPNAME | 应用程序名称，字符串格式，固定使用ipphone |
| 返回值 |  | 返回值格式见下面的说明 |

如果有消息，则返回：

|  |
| --- |
| 0 SUCCESS  // msg 是 key，内容分为两部分，用逗号隔开。前面是序号，后面是实际消息内容  msg=123,NIC,xxx  msg=124,NVM,xxx  msg=125,NVM,xxx  … … |

如果没有消息，则返回：

|  |
| --- |
| 11002 NO\_MESSAGE |

### 2.2.5 获取消息（FETCH）

FETCH和GET的区别在于：

* GET：当给调用方返回结果后，TCP连接并不马上终止，需要等待调用方主动Close连接。这种方式，有可能导致调用方无法得到完整的结果。
* FETCH：当给调用方返回结果后，立即关闭TCP连接。可保证调用方可得到完整的结果。

其它参数、返回值等，均和GET相同。

### 2.2.6 确认消息（ACK）

此部分用于客户端与服务器进行长链接，以接收服务端或其他客户端的实时推送。

|  |
| --- |
| ACK MY\_MID APPNAME SERIAL |

字段解释：

|  |  |  |
| --- | --- | --- |
| **方向** | **字段** | **格式和含义** |
| 发送 | ACK | 表示这个一个GET（获取消息）指令 |
| MY\_MID | 获取消息者的MID |
| APPNAME | 应用程序名称，字符串格式，ipphone |
| SERIAL | 确认的消息序号。所有序号 <= SERIAL的消息，将会被物理删除 |
| 返回值 |  | 有两个结果：   1. 该用户的队列中已经没有消息了，返回：0 SUCCESS 2. 该用户的队列中还有消息，返回：11003 HAS\_MESSAGE |

### 2.2.7 返回值和错误代码

返回值格式为：数字+空格+说明字符串。详细定义如下：

|  |  |  |
| --- | --- | --- |
| **错误代码** | **含义** | **说明** |
| 0 | SUCCESS | 成功 |
| 11000 | INVALID\_COMMAND | 非法的命令 |
| 11001 | INTERNAL\_ERROR | 服务器内部发生了异常，需要到MEC的error log文件去查看具体原因 |
| 11002 | NO\_MESSAGE | 没有消息。客户端执行GET操作时，如果没有对应用户的消息，则返回这个代码 |
| 11003 | HAS\_MESSAGE | 用户在执行了ACK将消息确认后，如果队列中又有了新的消息，则返回这个代码。客户端需要再取一次消息，直到返回11002为止。 |