XEngine消息服务文档

目录

[XEngine消息服务文档 1](#_Toc17914)

[前言 4](#_Toc27607)

[阅读者 4](#_Toc30137)

[概述 4](#_Toc471)

[相关模块 4](#_Toc19360)

[一 技术结构 4](#_Toc8584)

[1.1 TCP 4](#_Toc19698)

[1.2 HTTP 4](#_Toc11906)

[1.3 WEBSOCKET 4](#_Toc23770)

[1.4 心跳 4](#_Toc7526)

[1.5协议解释 5](#_Toc27383)

[二 配置环境 5](#_Toc16085)

[2.1 WINDOWS 5](#_Toc3832)

[2.1.1 配置环境 5](#_Toc30478)

[2.1.2 编译运行 5](#_Toc16959)

[2.2 LINUX 5](#_Toc28541)

[2.2.1 环境配置 5](#_Toc12220)

[2.2.2 编译运行 5](#_Toc31172)

[2.3 MacOS 6](#_Toc4941)

[2.4 版本要求 6](#_Toc22531)

[2.4.1 系统版本 6](#_Toc11620)

[2.4.2 软件需求 6](#_Toc31848)

[三 接口协议 6](#_Toc13524)

[3.1 TCP 6](#_Toc8792)

[3.1.1 投递包协议 6](#_Toc16356)

[3.1.2 获取包协议 8](#_Toc30117)

[3.1.3 删除包协议 9](#_Toc10014)

[3.1.4 创建主题 10](#_Toc24753)

[3.1.5 删除主题 11](#_Toc14196)

[3.1.6 订阅主题 12](#_Toc8300)

[3.1.7 通知消息 14](#_Toc30382)

[3.1.8 取包设置 14](#_Toc30100)

[3.1.9 编号获取 15](#_Toc5194)

[3.1.10 用户验证 17](#_Toc17569)

[3.1.11 用户注册 18](#_Toc12837)

[3.1.12 用户注销 18](#_Toc16947)

[3.2 HTTP 19](#_Toc19369)

[3.2.1 请求 19](#_Toc30740)

[3.2.2 回复 20](#_Toc30562)

[3.3 WEBSOCKET 21](#_Toc3834)

[四 配置说明 21](#_Toc24090)

[4.1 服务器配置 21](#_Toc27570)

[4.1.1 基本配置 21](#_Toc23533)

[4.1.2 最大配置 21](#_Toc25188)

[4.1.3 时间配置 22](#_Toc8954)

[4.1.4 日志配置 22](#_Toc6397)

[4.1.5 数据库配置 22](#_Toc13462)

[五 高级配置 22](#_Toc14866)

[附录 22](#_Toc27956)

[附录1 类型定义 22](#_Toc25529)

[附录2 协议定义 22](#_Toc89)

[附录3 转换定义 22](#_Toc23070)

[附录4 更新历史 23](#_Toc27984)

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布 | 文件名称： | XEngin消息服务文档 | |
| 适用版本： | V3.3 | |
| 发布日期： | 2022-08-05 | |
| 拟 制： qyt | | |

# 前言

## 阅读者

开发人员,测试人员

## **概述**

此文档包含了存储服务的相关技术说明和接口定义!

## 相关模块

此服务使用了XEngine作为开发包.要使用此服务代码必须配置安装好XEngine开发环境.

此服务还使用了第三方模块jsoncpp

# 一 技术结构

服务器启动会绑定三个端口,一个TCP消息端口,一个HTTP消息端口和WEBSOCKET端口

## TCP

TCP协议使用了XEngine的标准协议头来实现消息通信.TCP协议速度更快,功能更多.

## HTTP

消息通过通过POST方法提交,只需要发送地址端口的URL,然后负载JSON即可.HTTP不支持订阅模式.

## WEBSOCKET

Websocket可以为HTTP前端提供消息支持,使用标准WEBSOCKET协议,类型使用TEXT.负载和HTTP一样的JSON即可实现WEBSOCKET消息,WEBSOCKET消息支持订阅模式.

## 心跳

为了保持会话的持续,心跳必须按照固定时间发送.

## 1.5协议解释

TCP私有协议采用的是XEngine定义的协议,你可以参考XEngine的协议文档详细了解协议的定义与解释.

# 二 配置环境

## 2.1 WINDOWS

需要下载XEngine.

通过VS来编写代码,编译,调试.

### 2.1.1 配置环境

按照XEngine Readme文件的说明配置环境

### 2.1.2 编译运行

在完成环境配置后.你可以进入代码目录.使用VS打开XEngine\_MQServiceApp.sln,然后编译.

如果环境没有错误,编译会直接成功.包含5个模块和1个EXE程序

成功后,你需要拷贝XEngine\_Release下的文件到你编译的目录下.运行XEngine\_MQServiceApp.exe即可

注意：XEngine环境可以通过VSCopy脚本拷贝到你的编译目录下,前提是你配置好你的XEngine环境..

## 2.2 LINUX

### 2.2.1 环境配置

如果使用LINUX来开发运行,那么你需要在UBUNTU或者CENTOS下面才可以使用,UBUNTU需要20.04 以上系统.CENTOS需要8.x版本(兼容类Centos8版本)

### 2.2.2 编译运行

配置完环境后,你可以编译它,打开终端,进入源码目录执行以下命令

编译:make

安装:make FLAGS=InstallAll

清理:make FLAGS=CleanAll

如果没有错误,你可以在XEngine\_Release目录下看到编译的XEngine\_MQServiceApp

然后直接在终端运行它即可.

## 2.3 MacOS

MacOS需要12以及以上版本,编译和运行方式可以参考Linux.

## 2.4 版本要求

### 2.4.1 系统版本

最低版本要求:

WINDOWS: win7 sp1

Ubuntu:20.04

Centos:8.x

MacOS:12

### 2.4.2 软件需求

最低版本要求:

XEngine:V7.38

# 三 接口协议

## 3.1 TCP

请注意:协议头的wReserver字段将作为服务器回复处理结果使用.为0表示处理成功,其他值表示失败!  
 协议负载无论是通用协议还是HTTP协议,他们负载的内容都可以负载JSON.具体参考协议说明.

### 3.1.1 投递包协议

要使用消息队列,你需要先投递一个包给消息队列服务器,这样其他程序才能从消息队列服务中取得一个包.

#### 3.1.1.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQPOST

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ) + MSGLEN

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

末尾跟上DATA数据

nPubTime表示定时消息,只有订阅了这个消息并且在线的用户才会在这个时间段收到

#### 3.1.1.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPPOST

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.2 获取包协议

#### 3.1.2.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQGET

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.2.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPGET

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ) + MSGLEN

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

协议体:如果成功,协议后面跟上获取到的数据

### 3.1.3 删除包协议

#### 3.1.3.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQDEL

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.3.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPDEL

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.4 创建主题

要投递一个包到消息队列,你必须先创建一个主题,必须指明主题,数据才能进入消息队列.

#### 3.1.4.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQCREATE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.4.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPCREATE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.5 删除主题

#### 3.1.5.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQDELETE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.5.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPDELETE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.6 订阅主题

订阅主题允许用户在请求订阅成功后,每次主题有新内容,都会主动推送一次消息给订阅的客户端.

#### 3.1.6.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQNOTIFY

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.6.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPNOTIFY

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.7 通知消息

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_MSGNOTIFY

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ) + message

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

...body

### 3.1.8 取包设置

支持跳转到指定序列号开始获取

#### 3.1.8.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQSERIAL

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.8.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPSERIAL

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

### 3.1.9 编号获取

可以获取指定主题总消息个数,起始消息编号,结束消息编号

#### 3.1.9.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQNUMBER

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_ProtocolXmq

{

*CHAR* tszMQKey[256]; //此消息的KEY，不能为空

\_\_int64x nSerial; //包序列号

\_\_int64x nPubTime; //发布时间，根据自己需求配置时区

int nKeepTime; //保持时间

int nGetTimer; //可以获取的次数

}XENGINE\_PROTOCOL\_XMQ, \* LPXENGINE\_PROTOCOL\_XMQ;

#### 3.1.9.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_XMQ

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPNUMBER

unPacketSize = sizeof(XENGINE\_MQNUMBER)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_MQNumber

{

TCHAR tszMQKey[256]; //主题名

\_\_int64x nCount; //总个数

\_\_int64x nFirstNumber; //起始编码

\_\_int64x nLastNumber; //末尾编号

}XENGINE\_MQNUMBER, \* LPXENGINE\_MQNUMBER;

### 3.1.10 用户验证

#### 3.1.10.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQUSERLOG

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERAUTH)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_Protocol\_Auth

{

CHAR tszUserName[64]; //用户名

CHAR tszUserPass[64]; //密码

ENUM\_PROTOCOLCLIENT\_TYPE enClientType; //用户类型

ENUM\_PROTOCOLDEVICE\_TYPE enDeviceType; //设备类型

}XENGINE\_PROTOCOL\_USERAUTH, \* LPXENGINE\_PROTOCOL\_USERAUTH;

#### 3.1.10.2 回复

协议头:wReserve 0表示成功,其他值表示失败

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPUSERLOG

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

### 3.1.11 用户注册

#### 3.1.10.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQUSERREG

unPacketSize = sizeof(*XENGINE\_PROTOCOL\_USERINFO*)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

*XENGINE\_PROTOCOL\_USERINFO*

#### 3.1.10.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPUSERREG

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

### 3.1.12 用户注销

#### 3.1.10.1 请求

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REQUSERDEL

unPacketSize = sizeof(*XENGINE\_PROTOCOL\_USERINFO*)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

*XENGINE\_PROTOCOL\_USERINFO*

#### 3.1.10.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

unOperatorType = ENUM\_XENGINE\_COMMUNICATION\_PROTOCOL\_TYPE\_AUTH

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_MQ\_REPUSERDEL

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

## 3.2 HTTP

HTTP目前支持POST请求,通过负载JSON来确定请求事项

HTTP使用TOKEN来保持会话,在登录后得到的TOKEN,每次请求都必须带上

请求和回复的内容都与TCP的值相对应.

st\_MQProtocol 在消息负载的时候才存在

st\_Auth 在登录的时候才存在

st\_User 在注册和删除用户的时候存在

st\_Payload表示负载的数据,nPayType为0表示是字符串,1为BASE64编码,如果是1,那么服务器接受到数据就需要BASE64解码,tszPaydata为负载的具体数据.

如果没有负载内容,st\_Payload可以不存在.

wReserve字段在订阅消息协议的时候必须存在

### 3.2.1 请求

{  
    **"unOperatorType"**:**7**,  
    **"unOperatorCode"**:**7001**,  
    **"wReserve"**:**1**,  
    **"byVersion"**:**2**,

**"xhToken"**:**2123123123123**,  
    **"st\_MQProtocol"**:{  
        **"tszMQKey"**:**"主题名"**,  
        **"nSerial"**:**0**,  
        **"nKeepTime"**:**0**,  
        **"nGetTimer"**:**0**,  
        **"nPubTime"**:**0**  
    },  
    **"st\_Auth"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"enClientType"**:**10**,  
        **"enDeviceType"**:**20**  
    },  
    **"st\_User"**:{  
        **"nUserLevel"**:**0**,  
        **"nUserState"**:**0**,  
        **"nPhoneNumber"**:**1333333**,  
        **"nIDNumber"**:**511111**,  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"tszEMailAddr"**:**"41adawd@1111.com"**  
    },  
    **"st\_Payload"**:{  
        **"nPayType"**:**0**,  
        **"nPayLen"**:**6**,  
        **"tszPayData"**:**"123456"**  
    }  
}

### 3.2.2 回复

{  
    **"unOperatorType"**:**7**,  
    **"unOperatorCode"**:**7001**,  
    **"wReserve"**:**0**,

**"xhToken"**:**2123123123123**,  
    **"st\_MQProtocol"**:{  
        **"tszMQKey"**:**"主题名"**,  
        **"nSerial"**:**0**,  
        **"nKeepTime"**:**0**,  
        **"nGetTimer"**:**0**,

**"nPubTime"**:**0**  
    },  
    **"st\_Payload"**:{  
        **"nPayType"**:**0**,  
        **"nPayLen"**:**6**,  
        **"tszPayData"**:**"123456"**  
    }  
}

## 3.3 WEBSOCKET

WebSocket协议负载同HTTP,返回也是,WebSocket请求需要PING PONG来包活心跳,在没有数据交换的时候.

WebSocket的负载数据类型只能是TEXT.WEBSOCKET可以使用订阅模式

WEBSOCKET作为长连接,其操作方式与TCP相同,只是负载类型为JSON

# 四 配置说明

## 4.1 服务器配置

基础配置文件:XEngine\_Config.json

### 4.1.1 基本配置

* bDeamon: 1为守护进程后台运行0为前台运行
* tszTopic:默认公有主题名称
* nTCPPort:TCP通信端口
* nHttpPort:HTTP通信端口
* nWSPort:WEBSOCKET 通信端口

### 4.1.2 最大配置

XMax 配置

* MaxClient 最大允许客户端
* MaxQueue 最大允许队列
* IOThread:网络IO处理线程个数
* nTCPThread:TCP处理线程个数
* nHttpThread:HTTP处理线程个数
* nWSThread:WEBSOCKET处理线程个数

### 4.1.3 时间配置

XTime配置

* nDBMonth:数据库保存时间,默认月
* nSessionTime:客户端超时时间

### 4.1.4 日志配置

XLog 配置

* MaxSize:日志文件大小
* MaxCount:日志文件个数
* LogLeave:允许保存的级别

### 4.1.5 数据库配置

XSql 配置.需要MYSQL的支持

* SQLAddr:数据库地址
* SQLPort:数据库端口
* SQLUser:用户名
* SQLPass:密码

# 五 高级配置

# 附录

## 附录1 类型定义

参考文件 XEngine\_CommHdr.h

## 附录2 协议定义

参考文件XEngine\_ProtocolHdr.h

## 附录3 转换定义

参考文件 XEngine\_Types.h

## 附录4 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新说明 | 修改者 | 适用版本 |