XEngine消息服务文档

目录

[XEngine消息服务文档 1](#_Toc54)

[前言 4](#_Toc5887)

[阅读者 4](#_Toc8827)

[概述 4](#_Toc32114)

[相关模块 4](#_Toc28023)

[一 技术结构 4](#_Toc28441)

[1.1 TCP 4](#_Toc14289)

[1.2 HTTP 4](#_Toc27842)

[1.3 WEBSOCKET 4](#_Toc30932)

[1.4 心跳 4](#_Toc2907)

[1.5协议解释 5](#_Toc18758)

[二 配置环境 5](#_Toc8693)

[2.1 WINDOWS 5](#_Toc18375)

[2.1.1 配置环境 5](#_Toc20770)

[2.1.2 编译运行 5](#_Toc25935)

[2.2 LINUX 5](#_Toc30203)

[2.3 MacOS 5](#_Toc652)

[三 接口协议 6](#_Toc28300)

[3.1 TCP 6](#_Toc19748)

[3.1.1 投递包协议 6](#_Toc640)

[3.1.2 获取包协议 7](#_Toc1074)

[3.1.3 删除包协议 8](#_Toc6715)

[3.1.4 创建主题 9](#_Toc17778)

[3.1.5 删除主题 11](#_Toc32715)

[3.1.6 通知消息 12](#_Toc20723)

[3.1.7 主题绑定 12](#_Toc85)

[3.1.9 编号获取 14](#_Toc24400)

[3.1.10 用户验证 15](#_Toc14822)

[3.1.11 用户注册 16](#_Toc18113)

[3.1.12 用户注销 16](#_Toc207)

[3.1.13 消息修改 17](#_Toc19829)

[3.1.14 主题修改 18](#_Toc24831)

[3.1.15 未读消息 19](#_Toc15393)

[3.2 WEBSOCKET 22](#_Toc8940)

[3.2.1 请求 22](#_Toc6568)

[3.2.2 回复 23](#_Toc14606)

[3.2 HTTP 23](#_Toc316)

[3.2.1 查询用户列表 23](#_Toc25088)

[3.2.2 查询主题列表 24](#_Toc20087)

[四 配置说明 25](#_Toc27832)

[4.1 服务器配置 25](#_Toc30515)

[4.1.1 基本配置 25](#_Toc14202)

[4.1.2 最大配置 25](#_Toc26924)

[4.1.3 时间配置 25](#_Toc1907)

[4.1.4 日志配置 25](#_Toc12724)

[4.1.5 数据库配置 26](#_Toc1569)

[4.1.6 HTTP代理配置 26](#_Toc10237)

[4.2 数据库配置 26](#_Toc23276)

[4.2.1 用户数据库 26](#_Toc22219)

[五 高级配置 26](#_Toc19102)

[六 其他功能 26](#_Toc2179)

[6.1 回复配置 26](#_Toc21414)

[6.2 HTTP代理 27](#_Toc12538)

[附录 27](#_Toc13627)

[附录1 类型定义 27](#_Toc2670)

[附录2 协议定义 27](#_Toc227)

[附录3 转换定义 27](#_Toc10123)

[附录4 更新历史 27](#_Toc4330)

|  |  |  |  |
| --- | --- | --- | --- |
| 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： | V3.8 | |
| 发布日期： | 2023-08-16 | |
| 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

提供HTTP接口用于服务器信息查询和管理.

## WEBSOCKET

Websocket类型使用TEXT.负载JSON即可实现,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

如果使用LINUX来开发运行,那么你需要在UBUNTU或者RockyLinux下面才可以使用,UBUNTU需要22.04 以上系统.RockyLinux需要9.x版本(兼容类Centos)

## 2.3 MacOS

MacOS需要13以及以上版本,编译和运行方式可以参考Linux.

# 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.

wReserve:如果是定时发布,协议头的此字段可以表示自身是否接受

#### 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

byVersion 表示负载的内容格式,具体可参考ENUM\_XENGINE\_PROTOCOLHDR\_PAYLOAD\_TYPE

#### 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

byVersion返回的内容表示负载的类型,参考:ENUM\_XENGINE\_PROTOCOLHDR\_PAYLOAD\_TYPE

### 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\_REQTOPICCREATE

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\_REPTOPICCREATE

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\_REQTOPICDELETE

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\_REPTOPICDELETE

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 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

### 3.1.7 主题绑定

绑定后可以接受通知和不设置序列号取消息,支持跳转到指定序列号开始获取

#### 3.1.7.1 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\_REQTOPICBIND

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;

#### 3.1.7.2 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\_REPTOPICBIND

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

### 3.1.13 消息修改

#### 3.1.13.1 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\_REQMSGMODIFY

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:根据消息Key和序列号来确定修改的消息

XENGINE\_PROTOCOL\_XMQ

#### 3.1.13.2 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\_REPMSGMODIFY

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

### 3.1.14 主题修改

#### 3.1.14.1 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\_REQTOPICMODIFY

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

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:XMQ协议只需要填写主题名,末尾跟上要修改的主题名字符串即可

XENGINE\_PROTOCOL\_XMQ + tocpiname

#### 3.1.14.2 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\_REPTOPICMODIFY

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

### 3.1.15 未读消息

#### 3.1.15.1 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\_REQUNREAD

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XMQ)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:只需要填写主题名,也可以不填表示获取所有

XENGINE\_PROTOCOL\_XMQ

#### 3.1.15.2 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\_REPUNREAD

unPacketSize = JSON

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

{

"Array":[

{

"Array":[

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":9,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 10:46:13",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey",

"tszQueuePublishTime":""

},

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":10,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 10:46:13",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey",

"tszQueuePublishTime":""

}

],

"Count":2,

"Name":"XEngine\_NotifyKey"

},

{

"Array":[

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":6,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 15:10:43",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey2",

"tszQueuePublishTime":""

},

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":7,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 15:10:44",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey2",

"tszQueuePublishTime":""

},

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":8,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 15:10:45",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey2",

"tszQueuePublishTime":""

},

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":9,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 15:10:46",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey2",

"tszQueuePublishTime":""

},

{

"byMsgType":5,

"nMsgLen":12,

"nQueueGetTime":0,

"nQueueSerial":10,

"tszMsgBuffer":"123456789aaa",

"tszQueueCreateTime":"2023-01-04 15:10:46",

"tszQueueLeftTime":"",

"tszQueueName":"XEngine\_NotifyKey2",

"tszQueuePublishTime":""

}

],

"Count":5,

"Name":"XEngine\_NotifyKey2"

}

],

"Count":2

}

## 3.2 WEBSOCKET

WebSocket协议负载JSON,返回也是,WebSocket请求需要心跳协议来包活,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

头字段的内容都与TCP的请求相对应.

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,byVersion表示负载类型,参考协议头的定义,如果是二进制,需要先base64编码后传输.

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":{

"nPayLen":6,

"tszPayData":"123456"

}

}

### 3.2.2 回复

{

"unOperatorType":7,

"unOperatorCode":7001,

"wReserve":0,

"byVersion":2,

"st\_MQProtocol":{

"tszMQKey":"主题名",

"nSerial":0,

"nKeepTime":0,

"nGetTimer":0,

"nPubTime":0

},

"st\_Payload":{

"nPayType":5,

"nPayLen":6,

"tszPayData":"123456"

}

}

## 3.2 HTTP

HTTP用作管理接口,可以通过HTTP协议来管理服务器

### 3.2.1 查询用户列表

#### 3.2.1.1 请求

方法:GET

地址:http://127.0.0.1:5202/api?function=get&method=user

#### 3.2.1.2 回复

{

"Array":[

{

"nIDNumber":511000000000101000,

"nPhoneNumber":13699999999,

"nUserLevel":0,

"nUserState":0,

"tszCreateTime":"2023-08-11 13:36:54",

"tszEMailAddr":"486179@qq.com",

"tszLoginTime":"2023-08-11 13:37:22",

"tszUserName":"123123aa",

"tszUserPass":"123123"

}

],

"Count":1,

"code":0

}

### 3.2.2 查询主题列表

#### 3.2.2.1 请求

方法:GET

地址:http://127.0.0.1:5202/api?function=get&method=topic

#### 3.2.2.2 回复

{

"Array":[

{

"tszTopicName":"XEngine\_CommKey"

},

{

"tszTopicName":"XEngine\_NotifyKey"

}

],

"Count":2,

"code":0

}

# 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

### 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

### 4.1.6 HTTP代理配置

XPass配置

* nTimeout:超时时间,单位秒
* tszPassRegister:注册代理地址,比如:http://127.0.0.1:5600/api/v1/register
* tszPassUNReg:注销代理地址,比如:http://127.0.0.1:5600/api/v1/unregister
* tszPassLogin:登录代理地址,比如:http://127.0.0.1:5600/api/v1/login
* tszPassLogout:登出代理地址,比如:http://127.0.0.1:5600/api/v1/logout

## 4.2 数据库配置

配置文件:XEngine\_DBConfig.json 用于数据库操作的一些配置

### 4.2.1 用户数据库

#### 4.2.1.1 用户时间表

* bPubClear:定时发布完成是否清理数据库

# 五 高级配置

# 六 其他功能

## 6.1 回复配置

目前支持设置请求回复或者不回复的协议有:

* POST
* CREATE
* NOTIFY

## 6.2 HTTP代理

支持HTTP代理的协议有:

* 登录
* 登出
* 注册
* 注销

# 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 |