XEngine Message Service Docment

目录

[XEngine Message Service Docment 1](#_Toc22770)

[Preface 4](#_Toc31431)

[Reader 4](#_Toc14159)

[Overview 4](#_Toc24595)

[Associate Module 4](#_Toc15712)

[一 Technical structure 4](#_Toc1694)

[1.1 TCP 4](#_Toc4005)

[1.2 HTTP 4](#_Toc28087)

[1.3 WEBSOCKET 4](#_Toc12733)

[二 Configure Env 5](#_Toc11751)

[2.1 WINDOWS 5](#_Toc30692)

[2.1.1 Configure Environment 5](#_Toc22762)

[2.1.2 complie and run 5](#_Toc18835)

[2.2 LINUX 6](#_Toc2648)

[2.2.1 Evnironment Configure 6](#_Toc26108)

[2.2.2 Complie and Run 6](#_Toc13998)

[2.3 Version Requirements 6](#_Toc8557)

[2.3.1 System Version 6](#_Toc29597)

[2.3.2 Software Version 6](#_Toc2799)

[三 Interface Protocol 7](#_Toc3014)

[3.1 TCP 7](#_Toc9100)

[3.1.1 Post Protocol 7](#_Toc325)

[3.1.2 Get Protocol 8](#_Toc23850)

[3.1.3 Delete Protocol 9](#_Toc32192)

[3.1.4 Create Topic 10](#_Toc1244)

[3.1.5 Delete Topic 11](#_Toc25563)

[3.1.6 Subscribe Topic 12](#_Toc7445)

[3.1.7 Notification Message 14](#_Toc5427)

[3.1.8 Get Packet Set 14](#_Toc3623)

[3.1.9 Serial Number Get 15](#_Toc22867)

[3.1.10 User Authorize 16](#_Toc2025)

[3.2 HTTP 18](#_Toc30022)

[3.2.1 Request 18](#_Toc1415)

[3.2.2 Reply 18](#_Toc16553)

[3.3 WEBSOCKET 19](#_Toc11458)

[3.4 Data distribution protocol 19](#_Toc11377)

[3.4.1 Query Topic 19](#_Toc19489)

[3.4.2 Create Topic 20](#_Toc4292)

[3.4.3 Delete Topic 21](#_Toc6133)

[四 Configure Description 22](#_Toc648)

[4.1 Service Configure 22](#_Toc1040)

[4.1.1 basic configure 22](#_Toc13984)

[4.1.2 Max Configure 23](#_Toc18595)

[4.1.3 Time Configure 23](#_Toc26613)

[4.1.4 Log Configure 23](#_Toc21152)

[4.1.5 Database Configure 23](#_Toc7852)

[4.1.6 Authorize Configure 23](#_Toc9358)

[五 Advanced configuration 24](#_Toc27204)

[appendix 24](#_Toc4797)

[Appendix 1 Type Define 24](#_Toc28017)

[Appendix 2 Protocol Define 24](#_Toc24108)

[Appendix 3 Transformation Definition 24](#_Toc20966)

[Appendix 4 update log 24](#_Toc2987)

|  |  |  |  |
| --- | --- | --- | --- |
| File Status：  [ ] Draft  [√] Release | File Name： | XEngine Message Service Docment | |
| Be A Version： | V2.2 | |
| Released： | 2022-02-28 | |
| Writer： qyt | | |

# Preface

## Reader

Development ,test

## **Overview**

This document contains related technical descriptions and interface definitions

## Associate Module

The service used XEngine as Network Toolki.if you want to use code,you have to installed XEngine

And This service used library for jsoncpp

# 一 Technical structure

Should be bind Three port when Start service.these are tcp message port and http message port and websocket port

## TCP

The TCP protocol uses the standard protocol header of XEngine to realize message communication. The TCP protocol is faster and has more functions.

## HTTP

Send and recv Message through post.You only need to send the URL of the address and port, and then load the JSON. HTTP does not support subscription mode

### WEBSOCKET

Websocket can provide message support for the HTTP front end, using the standard WEBSOCKET protocol, and the type using TEXT. WEBSOCKET messages can be realized with JSON with the same load as HTTP, and WEBSOCKET messages support subscription mode.

# 二 Configure Env

## 2.1 WINDOWS

Need to download XEngine.

Complie and run and debug by vs.

Download address:https://gitee.com/xyry/libxengine

https://github.com/libxengine/xengine

### 2.1.1 Configure Environment

Follow the instructions in the XEngine Readme file to configure the environment.

### 2.1.2 complie and run

When you complete with configuration.you can come in code path.open XEngine\_MQServiceApp.sln by vs

If environment not have error.complie is succesed.contrain 5 module and 1 exe program

And you need copy file under XEngine\_Release to your complied dir.next step copy file under XEngine depend module to your complied dir.

If not have error,you can see follow the infomation



Note: You can run the program directly, the system will prompt you what you need, you can directly enter the XEngine directory to search.

## 2.2 LINUX

### 2.2.1 Evnironment Configure

If you use linux.you must running on ubuntu(20.04) or centos(8.x)...

### 2.2.2 Complie and Run

Configure complete.you can complie it.open terminal in you xengine\_storage dir and execute command.

complie:make

install:make FLAGS=InstallAll

clean:make FLAGS=CleanAll

If there is no error.you can see complied XEngine\_MQServiceApp file in XEngine\_Release

You can running at terminal.if there is no error,you can infomation:



## 2.3 Version Requirements

### 2.3.1 System Version

Minimum version requirements:

WINDOWS: win7 sp1

Ubuntu:20.04

Centos:8.x

### 2.3.2 Software Version

Minimum version requirements:

XEngine:V7.28

# 三 Interface Protocol

## 3.1 TCP

Please Note:The wReserver field of the protocol header will be used as the server reply processing result. 0 means the processing is successful, other values mean failure  
 whether protocol load is general protocol or HTTP protocol,they can be load json data.

### 3.1.1 Post Protocol

To use the message queue, you need to deliver a packet to the message queue server first, so that other programs can get a packet from the message queue service.

#### 3.1.1.1 Request

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

DATA in tail

#### 3.1.1.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

### 3.1.2 Get Protocol

#### 3.1.2.1 Requestion

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

#### 3.1.2.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

Protocol Body:if sucess,protocol follow get data

### 3.1.3 Delete Protocol

#### 3.1.3.1 Requestion

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

#### 3.1.3.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

### 3.1.4 Create Topic

Post a packet to message queue,you must create a topic,you have to set topic name,data can be enter the message queue

#### 3.1.4.1 Requestion

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

#### 3.1.4.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

### 3.1.5 Delete Topic

#### 3.1.5.1 Requestion

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

#### 3.1.5.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

### 3.1.6 Subscribe Topic

Subscribing to the topic allows the user to actively push a message to the subscribed client every time there is new content in the topic after the request is successfully subscribed

#### 3.1.6.1 Requestion

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

#### 3.1.6.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

### 3.1.7 Notification Message

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256];

\_int64x nSerial;

int nKeepTime;

int nGetTimer;

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

...body

### 3.1.8 Get Packet Set

Support jump to the specified serial number to start, support set reverse order get and order get

#### 3.1.8.1 Request

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256]; //set topic name

\_int64x nSerial; //get packet serial number,Zero gets the last or first value according to the order parameter

int nKeepTime; //1 is order, 0 is reverse order

int nGetTimer; //useless

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

#### 3.1.8.2 Reply

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256]; //set topic name

\_int64x nSerial; //get packet serial number

int nKeepTime; //1 is order, 0 is reverse order

int nGetTimer; //useless

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

### 3.1.9 Serial Number Get

Can be get specified topic count of number,start message number and tail message number

#### 3.1.9.1 Request

Protocol Header:

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

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXmq

{

*TCHAR* tszMQKey[256]; //topic name

\_int64x nSerial; //useless

int nKeepTime; //useless

int nGetTimer; //useless

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

#### 3.1.9.2 Reply

Protocol Header:

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\_MQNUMBER)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

typedef struct tag\_XEngine\_MQNumber

{

TCHAR tszMQKey[256]; //topic name

\_\_int64x nCount; //total number

\_\_int64x nFirstNumber; //start number

\_\_int64x nLastNumber; //tail number

}XENGINE\_MQNUMBER, \* LPXENGINE\_MQNUMBER;

### 3.1.10 User Authorize

This function is optional and can pass local authentication or HTTP proxy

#### 3.1.10.1 Request

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_AUTH\_REQLOGIN

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERAUTH)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

typedef struct tag\_XEngine\_Protocol\_Auth

{

CHAR tszUserName[64]; //User

CHAR tszUserPass[64]; //Pass

ENUM\_PROTOCOLCLIENT\_TYPE enClientType; //user type

ENUM\_PROTOCOLDEVICE\_TYPE enDeviceType; //device type

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

#### 3.1.10.2 Reply

Protocol Header:wReserve 0 is sucess,ohter fail

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_AUTH\_REPLOGIN

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

#### 3.1.10.3 HTTP Authorize

If it is HTTP or WEBSOCKET, then the load JSON format is as follows: and there is no st\_MQProtocol structure. The two fields of en are optional and can be Optional. The HTTP proxy verification request is the same, but all have only the main field.

{

....JSON Main...  
    **"st\_Auth"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"enClientType"**:**10**,  
        **"enDeviceType"**:**20**  
    }  
}

## 3.2 HTTP

HTTP support post requestion now.make sure event through json

Content of Requestion and reply same to tcp protocol

st\_Payload mean is load data.nPayType is 0 mean it is string,is 1 as base64 codec,if the value is 1, Then you need base64 decoding

If no have payload,,st\_Payload can not be exist.

Serial can not be empty When you get the message

wReserve feild Must existed When subscribe message protocol

### 3.2.1 Request

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

**"wReserve"**:**1**,

**"byVersion"**:**2**,  
    **"st\_MQProtocol"**:{  
        **"tszMQKey"**:**"主题名"**,  
        **"nSerial"**:**0**,  
        **"nKeepTime"**:**0**,  
        **"nGetTimer"**:**0**  
    },  
    **"st\_Payload"**:{  
        **"nPayType"**:**0**,  
        **"nPayLen"**:**6**,  
        **"tszPayData"**:**"123456"**  
    }  
}

### 3.2.2 Reply

{  
    **"unOperatorType"**:**7**,  
    **"unOperatorCode"**:**7001**,  
    **"wReserve"**:**0**,  
    **"st\_MQProtocol"**:{  
        **"tszMQKey"**:**"主题名"**,  
        **"nSerial"**:**0**,  
        **"nKeepTime"**:**0**,  
        **"nGetTimer"**:**0**  
    },  
    **"st\_Payload"**:{  
        **"nPayType"**:**0**,  
        **"nPayLen"**:**6**,  
        **"tszPayData"**:**"123456"**  
    }  
}

## 3.3 WEBSOCKET

WebSocket Protocol same load as http,return too,WebSocket request need ping and pong to heartbeat when is not have data change.

Websocket payload type only is TEXT.websocket can be used subscribe mode.

Websocket as long connection.can be not set serial whe you get message.

## 3.4 Data distribution protocol

There are a few things to note about data distribution services.

1:The DDS uses the broadcast protocol

2:DDS content uses multicast protocol and TCP protocol

3.Need to start multiple cross-network segment services, need central service support

### 3.4.1 Query Topic

#### 3.4.1.1 Request

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REQTOPICQUERY

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XDDS)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

st\_XDDSProtocol.tszTopic = “query topic name”

#### 3.4.1.2 Reply

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REPTOPICQUERY

unPacketSize = JSONSIZE

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

{  
    **"ListCount"**:**1**,  
    **"ListArray"**:[  
        {  
            **"tszTopic"**:**"aa"**,  
            **"tszDDSAddr"**:**"192.168.1.2"**,  
            **"bCreater"**:**1**,  
            **"bTcp"**:**0**,  
            **"nPort"**:**10000**  
        }  
    ]  
}

### 3.4.2 Create Topic

#### 3.4.2.1 Request

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REQTOPICCREATE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XDDS)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXdds

{

CHAR tszTopic[MAX\_PATH]; //Topic

CHAR tszDDSAddr[64]; //TCP is local address,UDP is cast address

BOOL bCreater; //TRUE

BOOL bTcp; //is TCP or udp

int nPort; //port

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

#### 3.4.2.2 Reply

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REPTOPICCREATE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XDDS)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:DDS structure information you requested

### 3.4.3 Delete Topic

#### 3.4.3.1 Request

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REQTOPICDELETE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XDDS)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:

typedef struct tag\_XEngine\_ProtocolXdds

{

CHAR tszTopic[MAX\_PATH]; //Topic

CHAR tszDDSAddr[64]; //TCP is local address,UDP is cast address

BOOL bCreater; //TRUE

BOOL bTcp; //is TCP or udp

int nPort; //port

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

#### 3.4.3.2 Reply

Protocol Header:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = DomainID

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

unOperatorCode = XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_DDS\_REPTOPICDELETE

unPacketSize = sizeof(XENGINE\_PROTOCOL\_XDDS)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Protocol Body:DDS structure information you requested

# 四 Configure Description

## 4.1 Service Configure

Basic Configure File:XEngine\_Config.json

### 4.1.1 basic configure

* bDeamon: 1 deamon process run 0 is terminal run
* tszTopic:Default Topic Name
* nTCPPort:tcp port
* nHttpPort:http port
* nWSPort:websocket port
* nBroadRVPort:DDS Protocol Recv Port
* NBroadSDPort:DDS Protocol Send Port

### 4.1.2 Max Configure

XMax Configure

* MaxClient Allow Max Client Count
* MaxQueue Allow Max Queue
* IOThread:network io process threads number
* nTCPThread:tcp process threads number
* nHttpThread:http process threads number
* nWSThread:websocket process thread number

### 4.1.3 Time Configure

XTime Configure

* bHBTime,1 is enable heartbeat,0 disable
* nDBMonth:database save time,default month
* nTimeCheck:check time
* nTCPTimeOut:how time check once
* nHttpTimeOut:same nTCPTimeOut
* nWSTimeOut:same to nTCPTimeOut

### 4.1.4 Log Configure

XLog Configure

* MaxSize:Log file size
* MaxCount:Log File Number
* LogLeave:Allow save level

### 4.1.5 Database Configure

XSql Configure

### 4.1.6 Authorize Configure

XAuth Configure

* nAuth:0 is close user authorize,1 use local user list,2 use HTTP pass
* AuthUser:local user list address
* AuthHttp:HTTP proxy address

# 五 Advanced configuration

# appendix

## Appendix 1 Type Define

reference file:XEngine\_CommHdr.h

## Appendix 2 Protocol Define

reference file:XEngine\_ProtocolHdr.h

## Appendix 3 Transformation Definition

reference file:XEngine\_Types.h only LINUX

## Appendix 4 update log