XEngine网络验证服务文档

目录

[XEngine网络验证服务文档 1](#_Toc29080)

[前言 4](#_Toc30668)

[阅读者 4](#_Toc7354)

[概述 4](#_Toc32084)

[相关模块 4](#_Toc29397)

[一 技术结构 4](#_Toc23892)

[1.1 目录结构 4](#_Toc18946)

[1.2 程序目录 4](#_Toc6117)

[二 配置环境 5](#_Toc26915)

[2.1 WINDOWS 5](#_Toc8358)

[2.1.1 配置环境 5](#_Toc9366)

[2.1.2 编译运行 5](#_Toc21709)

[2.2 LINUX 5](#_Toc7617)

[2.2.1 环境配置 5](#_Toc19724)

[2.2.2 编译运行 5](#_Toc7147)

[2.3 版本要求 6](#_Toc26490)

[2.3.1 系统版本 6](#_Toc12533)

[2.3.2 软件需求 6](#_Toc4606)

[三 TCP接口协议 6](#_Toc12456)

[3.1 删除协议 6](#_Toc24777)

[3.1.1 请求 7](#_Toc30650)

[3.1.2 回复 7](#_Toc6622)

[3.2 注册协议 8](#_Toc10168)

[3.2.1 请求 8](#_Toc26056)

[3.2.2 回复 8](#_Toc26480)

[3.3 登录协议 9](#_Toc5852)

[3.3.1 请求 9](#_Toc29597)

[3.3.2 回复 9](#_Toc17344)

[3.4 充值协议 10](#_Toc27091)

[3.4.1 请求 10](#_Toc5911)

[3.4.2 回复 10](#_Toc18439)

[3.5 找回密码 11](#_Toc1395)

[3.5.1 请求 11](#_Toc11562)

[3.5.2 回复 11](#_Toc25604)

[3.6 获取时间 12](#_Toc2981)

[3.6.1 请求 12](#_Toc28066)

[3.6.2 回复 13](#_Toc17600)

[3.7 网络验证 13](#_Toc28948)

[3.7.1 请求 13](#_Toc7227)

[3.7.2 回复 14](#_Toc31442)

[3.8 获取用户 14](#_Toc11672)

[3.8.1 请求 14](#_Toc7569)

[3.8.2 回复 15](#_Toc6389)

[3.9 设置用户 15](#_Toc22608)

[3.9.1 请求 15](#_Toc2909)

[3.9.2 回复 16](#_Toc19913)

[3.10 公告协议 16](#_Toc9979)

[3.11 超时通知 16](#_Toc24691)

[四 Websocket接口协议 17](#_Toc12696)

[4.1 用户信息协议 17](#_Toc8047)

[4.1.1 请求 17](#_Toc25669)

[4.1.2 回复 17](#_Toc5153)

[4.2 用户验证协议 18](#_Toc19750)

[4.2.1 请求 18](#_Toc15678)

[4.2.2 回复 19](#_Toc31921)

[4.3 用户充值协议 19](#_Toc11156)

[4.3.1 请求 19](#_Toc16872)

[4.3.2 回复 20](#_Toc5599)

[4.4 用户时间协议 20](#_Toc5458)

[4.4.1 请求 20](#_Toc9310)

[4.4.2 回复 20](#_Toc4284)

[4.5 快速验证 21](#_Toc31245)

[4.5.1 请求 21](#_Toc523)

[4.5.2 回复 21](#_Toc28843)

[4.6 通告协议 21](#_Toc22956)

[4.6.1 请求 21](#_Toc26103)

[4.6.2 回复 21](#_Toc31069)

[4.7 过期协议 21](#_Toc11207)

[4.7.1 请求 22](#_Toc20070)

[4.7.2 回复 22](#_Toc31206)

[五 配置说明 22](#_Toc14974)

[5.1 基础配置 22](#_Toc30688)

[5.2 验证配置 22](#_Toc10476)

[5.3 加密配置 22](#_Toc26143)

[5.4 邮件配置 23](#_Toc22955)

[5.5 数据库配置 23](#_Toc26892)

[5.6 日志配置 23](#_Toc31373)

[附录 23](#_Toc19979)

[附录1 类型定义 23](#_Toc9918)

[附录2 协议定义 23](#_Toc8590)

[附录3 转换定义 23](#_Toc26388)

[附录4 更新历史 24](#_Toc26261)

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布 | 文件名称： | XEngine网络验证服务文档 | |
| 适用版本： | V2.6 | |
| 发布日期： | 2022-02-18 | |
| 拟 制： qyt | | |

# 前言

## 阅读者

开发人员,测试人员

## **概述**

此文档包含了验证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.

# 一 技术结构

采用的是TCP协议来作为基础通信协议

服务器启动会绑定两个端口,用来处理验证协议.

## 目录结构

* XEngine\_Apps:应用例子代码目录
* XEngine\_Docment:文档目录
* XEngine\_Release:编译发布目录
* XEngine\_Source:源代码目录

## 程序目录

应用程序我们有两个版本,在源代码的XEngine\_APPService目录中,我们建议你使用带界面版本

* XEngine\_AuthorizeApp:WINDOWS带管理界面版本
* XEngine\_AuthorizeService:跨平台不带界面版本

# 二 配置环境

## 2.1 WINDOWS

需要下载XEngine.

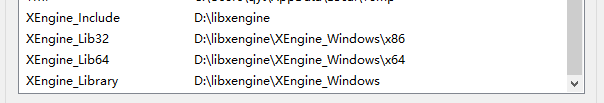
通过VS C++来编写代码,编译,调试.

下载地址:https://gitee.com/xyry/libxengine

https://github.com/libxengine/xengine

### 2.1.1 配置环境

按照XEngine Readme文件的说明执行脚本配置环境,成功会在你的系统环境变量中看到下面的值.



### 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.sln.

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.然后拷贝依赖的XEngine模块到你的编译目录下,运行XEngine\_AuthorizeApp.exe即可

注意：你可以直接运行程序，系统会提示你需要哪些，你直接进入XEngine目录搜索.

## 2.2 LINUX

LINUX只能使用XEngine\_AuthorizeService不带界面的版本

### 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(兼容相同版本)

### 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\_AuthorizeService

然后直接在终端运行它即可.

## 2.3 版本要求

### 2.3.1 系统版本

最低版本要求:

WINDOWS: win7 sp1

Ubuntu:20.04

Centos:8.x

### 2.3.2 软件需求

最低版本要求:

XEngine:V7.28

# 三 TCP接口协议

TCP接口协议用于适用于APP等应用程序验证.可用于登录,注册,注销,删除等操作.

请注意:验证协议使用了协议头的保留字段wReserve用于回复客户端请求,告诉客户端这次操作成功还是失败,0表示成功,其他表示失败

wCrypto表示加解密,目前仅仅支持类型4,XCrypto算法.如果需要加密,请填充4,否则请填充0

## 3.1 删除协议

用于删除一个用户

### 3.1.1 请求

请求一次删除用户,你需要使用协议头+注册协议来实现,这次协议里面的内容用于判断和验证用户是否有权限删除自己.当然,配合TOKEN使用,还可以实现权限分配.使用TOKEN来验证这个用户有没有权限删除另一个用户

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERINFO)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_Protocol\_UserReg

{

*TCHAR* tszUserName[64]; //用户名

*TCHAR* tszUserPass[64]; //密码

*TCHAR* tszEMailAddr[64]; //电子邮件地址

*TCHAR* tszLoginTime[64]; //登录时间

*TCHAR* tszCreateTime[64]; //注册时间

\_\_int64x nPhoneNumber; //电话号码

\_\_int64x nIDNumber; //身份证号

int nUserLevel; //用户等级

*int nUserState*; //状态

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

### 3.1.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

## 3.2 注册协议

### 3.2.1 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\_AUTH\_REQREGISTER

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERINFO)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_Protocol\_UserReg

{

*TCHAR* tszUserName[64]; //用户名

*TCHAR* tszUserPass[64]; //密码

*TCHAR* tszEMailAddr[64]; //电子邮件地址

*TCHAR* tszLoginTime[64]; //登录时间

*TCHAR* tszCreateTime[64]; //注册时间

\_\_int64x nPhoneNumber; //电话号码

\_\_int64x nIDNumber; //身份证号

int nUserLevel; //用户等级

*int nUserState*; //是否在线

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

### 3.2.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

## 3.3 登录协议

登录协议是客户端和服务器在请求服务器交换数据的时候所需要做的第一个步骤.只有登录成功,才能使用服务器

### 3.3.1 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\_AUTH\_REQLOGIN

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; //客户端类型

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

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

### 3.3.2 回复

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 = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

#### 3.3.2.1 回复类别

服务器回复成功和失败,可以通过开发人员自定义,或者通过统一标识来确定.在这里.我们通过wReserver来确定,这个值在登录协议上面,返回的数值将表达以下意思:

* 0:成功
* 251:用户不存在
* 252:密码错误
* 253:用户没有权限
* 254:用户已过期
* 255:服务器内部错误

## 3.4 充值协议

用户充值协议是通过充值卡来充值的,充值卡需要通过我们的序列号模块来生成.

### 3.4.1 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\_AUTH\_REQPAY

unPacketSize = sizeof(AUTHREG\_PROTOCOL\_USERPAY)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct

{

*CHAR* tszUserName[XENGINE\_AUTHREG\_SERVICE\_SQL\_MAX\_USERNAME]; //用户

*CHAR* tszSerialNumber[128]; //序列号

}AUTHREG\_PROTOCOL\_USERPAY, \*LPAUTHREG\_PROTOCOL\_USERPAY;

### 3.4.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

## 3.5 找回密码

### 3.5.1 请求

协议头:请求的协议,用户名和身份证以及电话号码不能为空

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERINFO)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct tag\_XEngine\_Protocol\_UserReg

{

*TCHAR* tszUserName[64];

*TCHAR* tszUserPass[64];

*TCHAR* tszEMailAddr[64];

*TCHAR* tszLoginTime[64];

*TCHAR* tszCreateTime[64];

\_\_int64x nPhoneNumber;

\_\_int64x nIDNumber;

int nUserLevel;

*int nUserState*;

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

### 3.5.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERAUTH)

byVersion = 1

byIsReply = FALSE

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;

#### 2.5.2.1 错误定义

* 291:用户不存在
* 292:验证信息不正确

## 3.6 获取时间

获取时间表示获取用户所剩余的时间.需要登录

### 3.6.1 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\_AUTH\_REQGETTIME

unPacketSize = 0

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

### 3.6.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = sizeof(AUTHREG\_PROTOCOL\_TIME)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

typedef struct

{

CHAR tszUserAddr[XENGINE\_AUTHREG\_SERVICE\_SQL\_MAX\_USERNAME]; //地址

CHAR tszUserName[XENGINE\_AUTHREG\_SERVICE\_SQL\_MAX\_USERNAME]; //用户

CHAR tszLeftTime[64]; //过期日期

\_\_int64x nTimeLeft; //剩余时间

\_\_int64x nTimeONLine; //在线时间

ENUM\_AUTHREG\_GENERATESERIALTYPE enSerialType;

}AUTHREG\_PROTOCOL\_TIME, \* LPAUTHREG\_PROTOCOL\_TIME;

## 3.7 网络验证

临时网络验证协议,可以不需要注册登录即可使用,通过一段唯一的识别码(CPUID,主板ID,硬盘ID)来进行注册.此注册会发送到服务器,服务器会自动进行记录保存,并且识别是否过期,方便用户使用.

此协议比本地验证更加安全.可以方便一些不想注册的用户使用.

### 3.7.1 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\_AUTH\_REQTRYVER

unPacketSize = IDLEN

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:填写唯一标识码,可以是CPUID等

### 3.7.2 回复

协议头:成功将返回下面的内容,表示时间还没有过期

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

#### 3.7.2.1 错误信息

* 0x2D1:表示用户的试用时间已经超过了
* 0x2D2:表示服务器没有启用此功能
* 0x2D3:服务器内部错误

## 3.8 获取用户

需要验证用户名和密码

### 3.8.1 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\_AUTH\_REQGETUSER

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; //客户端类型

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

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

### 3.8.2 回复

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERINFO)

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

Data(XENGINE\_PROTOCOL\_USERINFO)

## 3.9 设置用户

需要验证一些基本信息

### 3.9.1 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\_AUTH\_REQSETUSER

unPacketSize = sizeof(XENGINE\_PROTOCOL\_USERINFO)

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:

Data(XENGINE\_PROTOCOL\_USERINFO)

### 3.9.2 回复

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 1

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

## 3.10 公告协议

此协议只用于发送文本内容.用于通知用户消息.

协议头:

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = MSGLEN

byVersion = 1

byIsReply = TRUE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

协议体:协议体为要发送的内容.

## 3.11 超时通知

此协议为通知协议,服务器主动下发,当客户端收到此协议后需要主动断开连接,否则服务器会一直下发此通知.

此协议表示了客户端的时间到期,没有剩余时间了.

wHeader = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_HEADER

xhToken = 0

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

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

unPacketSize = 0

byVersion = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

# 四 Websocket接口协议

WEBSOCKET协议简称WS协议.

除了上面的TCP协议,WS协议可以支持HTML或前端网页处理授权验证,这样除了APP程序以外,我们还可以支持对网站进行授权验证.

前端网页的授权验证通过WEBSOCKET协议负载JSON实现.

## 4.1 用户信息协议

用于删除,注册,找回密码,用户设置协议

### 4.1.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8193**,  
    **"byIsReply"**:**1**,

**"wCrypto"**:**0**,  
    **"st\_UserInfo"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"tszEMailAddr"**:**"486179@qq.com"**,  
        **"nPhoneNumber"**:**1369999999**,  
        **"nIDNumber"**:**5000000000**  
    }  
}

### 4.1.2 回复

#### 4.1.2.1 用户删除,注册,设置

{  
    **"wHeader"**:**17**,  
    **"wTail"**:**255**,  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8197**,  
    **"byIsReply"**:**0**,  
    **"wCrypto"**:**0**,  
    **"wReserve"**:**0**  
}

#### 4.1.2.2 找回密码

{  
    **"wHeader"**:**17**,  
    **"wTail"**:**255**,  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8197**,  
    **"byIsReply"**:**0**,  
    **"wCrypto"**:**0**,  
    **"wReserve"**:**0**,  
    **"st\_UserAuth"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"enClientType"**:**10**,  
        **"enDeviceType"**:**25**  
    }  
}

## 4.2 用户验证协议

用于登录,获取用户协议

### 4.2.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8197**,  
    **"byIsReply"**:**1**,

**"wCrypto"**:**0**,  
    **"st\_UserAuth"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserPass"**:**"123123"**,  
        **"enClientType"**:**10**,  
        **"enDeviceType"**:**25**  
    }  
}

### 4.2.2 回复

#### 4.2.2.1 登录

参考4.1.2.1

#### 4.2.2.2 获取用户

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8203**,  
    **"byIsReply"**:**1**,  
    **"wCrypto"**:**0**,  
    **"st\_UserInfo"**:{  
        **"tszUserName"**:**"用户名"**,  
        **"tszUserPass"**:**"用户密码"**,  
        **"tszCreateTime"**:**"注册时间"**,  
        **"tszEMailAddr"**:**"邮件"**,  
        **"tszLoginTime"**:**"最后登录时间"**,  
        **"nIDNumber"**:**5151515151**,  
        **"nPhoneNumber"**:**13131313**,  
        **"nUserLevel"**:**1**,  
        **"nUserState"**:**0**  
    }  
}

## 4.3 用户充值协议

用于用户充值协议

### 4.3.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8199**,  
    **"byIsReply"**:**1**,

**"wCrypto"**:**0**,  
    **"st\_UserPay"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszSerialNumber"**:**"123123"**  
    }  
}

### 4.3.2 回复

参考4.1.2.1节

## 4.4 用户时间协议

用于获取剩余时间

### 4.4.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8203**,  
    **"byIsReply"**:**1,**

**"wCrypto"**:**0**  
}

### 4.4.2 回复

{  
    **"wHeader"**:**17**,  
    **"wTail"**:**255**,  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8204**,  
    **"byIsReply"**:**0**,  
    **"wCrypto"**:**0**,  
    **"wReserve"**:**0**,  
    **"st\_UserTime"**:{  
        **"tszUserName"**:**"123123aa"**,  
        **"tszUserAddr"**:**"123123"**,  
        **"tszLeftTime"**:**"2021-12-22 10:11:23"**,  
        **"nTimeLeft"**:**25**,  
        **"nTimeONLine"**:**10**,  
        **"enDeviceType"**:**1**,  
        **"enSerialType"**:**1**  
    }  
}

## 4.5 快速验证

功能参考3.6节内容

### 4.5.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8203**,  
    **"byIsReply"**:**1**,  
    **"wCrypto"**:**0**,  
    **"lpszPayload"**:**"唯一硬件码或者其他序列值"**  
}

### 4.5.2 回复

回复内容参考4.1.2.1

## 4.6 通告协议

协议内容功能参考3.10

### 4.6.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**12272**,  
    **"byIsReply"**:**1**,  
    **"wCrypto"**:**0**,  
    **"lpszPayload"**:**"发送的内容"**  
}

### 4.6.2 回复

无

## 4.7 过期协议

协议功能参考3.11

### 4.7.1 请求

{  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**12287**,  
    **"byIsReply"**:**0**,  
    **"wCrypto"**:**0**  
}

### 4.7.2 回复

无

# 五 配置说明

配置文件:XEngine\_Config.ini

## 5.1 基础配置

配置信息:ServiceConfig

* TCPPort:TCP监听的端口
* WSPort:websocket端口
* ThreadPool:允许的线程
* UserVerTimed:用户允许的验证时间
* AutoStart:是否自动运行
* TimeNotify:超时通知设置,为1为一直通知直到客户端主动关闭

## 5.2 验证配置

配置信息:Verification,快速验证不需要登录,连接后直接请求就可以得到验证授权信息

* VerTime:快速验证时间
* VerMode:快速验证模式
* TryTime:试用时间
* TryMode:试用模式

## 5.3 加密配置

配置信息:Crypto,建议使用加密传输,否则很容易被破解

* Enable:是否启用
* Pass:密码,仅允许数字

## 5.4 邮件配置

配置信息:SmtpConfig

* bSmtpEnable:0不启用,1启用,启用后只返回成功失败给客户端,密码发送到邮箱
* SmtpService:电子邮件服务器
* SmtpFromAddr:回复地址
* SmtpUser:用户名
* SmtpPass:密码

## 5.5 数据库配置

配置信息:XSql

* tszSQLite:数据库地址

## 5.6 日志配置

配置信息:XLog 目前仅仅适用于不带界面版本

* nMaxSize:最大日志文件大小
* nMaxCount:文件备份个数
* nLogLeave:日志级别
* tszLogFile:日志保存路径

# 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 |