XEngine网络验证服务文档

目录

[XEngine网络验证服务文档 1](#_Toc8769)

[前言 5](#_Toc31706)

[阅读者 5](#_Toc30880)

[概述 5](#_Toc31695)

[相关模块 5](#_Toc816)

[一 技术结构 5](#_Toc499)

[1.1 目录结构 5](#_Toc12264)

[1.2 程序目录 5](#_Toc11387)

[1.3 用户级别 6](#_Toc21470)

[1.4 协议解释 6](#_Toc13584)

[1.5 加密通信 6](#_Toc16213)

[二 配置环境 6](#_Toc15142)

[2.1 WINDOWS 6](#_Toc19273)

[2.1.1 配置环境 6](#_Toc21965)

[2.1.2 编译运行 6](#_Toc29796)

[2.2 LINUX 7](#_Toc20839)

[2.2.1 环境配置 7](#_Toc30328)

[2.2.2 编译运行 7](#_Toc24484)

[2.3 MacOS 7](#_Toc13559)

[2.4 老版本升级 7](#_Toc29865)

[三 TCP接口协议 7](#_Toc24577)

[3.1 登录协议 8](#_Toc8800)

[3.1.1 请求 8](#_Toc763)

[3.1.2 回复 8](#_Toc2870)

[3.2 超时通知 9](#_Toc20687)

[四 Websocket接口协议 9](#_Toc4052)

[4.1 用户验证协议 10](#_Toc25416)

[4.1.1 请求 10](#_Toc14229)

[4.1.2 回复 10](#_Toc3618)

[4.2 超时通知 10](#_Toc4556)

[五 HTTP管理接口 10](#_Toc11208)

[5.1 客户端管理接口 11](#_Toc32278)

[5.1.1 获取用户 11](#_Toc14551)

[5.1.2 用户列表 12](#_Toc31812)

[5.1.3 关闭客户 13](#_Toc31964)

[5.1.4 修改用户 13](#_Toc26795)

[5.2 序列号管理接口 14](#_Toc12778)

[5.2.1 列举序列号 14](#_Toc28632)

[5.2.2 插入序列号 15](#_Toc27131)

[5.2.3 删除序列号 16](#_Toc7417)

[5.3 用户接口 16](#_Toc10297)

[5.3.1 用户删除 16](#_Toc16508)

[5.3.2 用户注册 17](#_Toc5533)

[5.3.3 用户充值 18](#_Toc6319)

[5.3.4 找回密码 18](#_Toc13195)

[5.3.5 获取时间 19](#_Toc13421)

[5.3.6 快速验证 20](#_Toc17495)

[5.4 Token协议 20](#_Toc2071)

[5.4.1 登录 21](#_Toc8400)

[5.4.2 续期 21](#_Toc12136)

[5.4.3 关闭 22](#_Toc11079)

[5.5 功能开关 22](#_Toc27371)

[5.5.1 获取功能列表 22](#_Toc21419)

[5.5.2 设置功能开关 23](#_Toc7705)

[5.6 黑名单协议 24](#_Toc32178)

[5.6.1 插入 24](#_Toc31593)

[5.6.2 删除 25](#_Toc24877)

[5.6.3 查询 25](#_Toc13782)

[5.6.4 修改 26](#_Toc32236)

[5.7 CDKEY系统 27](#_Toc2985)

[5.7.1 创建CDKey 27](#_Toc18404)

[5.7.2 注册CDKey 29](#_Toc23693)

[5.7.3 验证CDKey 29](#_Toc5496)

[5.8 公告协议 29](#_Toc19759)

[5.8.1 插入 30](#_Toc26900)

[5.8.2 删除 30](#_Toc19707)

[5.8.3 列举 31](#_Toc14449)

[5.9 动态验证码 31](#_Toc14013)

[5.9.1 获取 32](#_Toc10052)

[5.10 临时验证协议 32](#_Toc19231)

[5.10.1 列举 32](#_Toc6184)

[5.10.2 删除 33](#_Toc31850)

[5.10.3 修改 34](#_Toc19964)

[六 三方验证 34](#_Toc31983)

[6.1 用户登录 34](#_Toc26895)

[6.1.1 请求 34](#_Toc19960)

[6.1.2 回复 35](#_Toc9934)

[6.2 用户登出 35](#_Toc15177)

[6.2.1 请求 35](#_Toc16342)

[6.2.2 回复 35](#_Toc20323)

[6.3 用户超时 35](#_Toc24467)

[6.3.1 请求 35](#_Toc28460)

[6.3.2 回复 35](#_Toc18899)

[七 高级功能 35](#_Toc32694)

[7.1 分布式验证 35](#_Toc9719)

[7.2 多端登录 36](#_Toc20264)

[7.3 序列卡说明 36](#_Toc25767)

[7.3.1 秒卡 36](#_Toc546)

[7.3.2 次数卡 36](#_Toc7224)

[7.3.3 天数卡 36](#_Toc4371)

[7.3.4 自定义日期卡 37](#_Toc22815)

[7.4 客户端SDK 37](#_Toc32081)

[7.5 CDKey本地验证 37](#_Toc2858)

[7.5.1 通过网络 37](#_Toc25598)

[7.5.2 通过本地 37](#_Toc15704)

[7.6 数据库 38](#_Toc1272)

[7.6.1 SQLITE 38](#_Toc30079)

[7.6.2 MYSQL 38](#_Toc18023)

[八 配置说明 38](#_Toc16935)

[8.1 基本配置文件 38](#_Toc16812)

[8.1.1 基础配置 38](#_Toc28741)

[8.1.2 最大配置 38](#_Toc3249)

[8.1.3 验证配置 39](#_Toc28599)

[8.1.4 登录配置 39](#_Toc19190)

[8.1.5 加密配置 39](#_Toc25776)

[8.1.6 数据库配置 39](#_Toc11217)

[8.1.7 日志配置 40](#_Toc5492)

[8.1.8 报告配置 40](#_Toc26537)

[8.2 功能开关配置 40](#_Toc7962)

[附录 40](#_Toc7241)

[附录1 类型定义 40](#_Toc28968)

[附录2 协议定义 41](#_Toc10355)

[附录3 转换定义 41](#_Toc20296)

|  |  |  |  |
| --- | --- | --- | --- |
| 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： | V3.13.0.1001 | |
| 发布日期： | 2024-07-15 | |
| 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,WEBSOCKET,HTTP协议来作为基础通信协议

服务器启动会绑定三个端口,用来处理验证协议.

## 1.1 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
* XEngine\_SQL:数据库备份目录

## 1.2 程序目录

我们提供了服务器和管理工具,在源代码的XEngine\_APPService目录中

* XEngine\_AuthorizeApp:Windows管理工具,依赖XEngine\_AuthorizeService
* XEngine\_AuthorizeService:网络验证主服务器

## 1.3 用户级别

用户一共有很多级别,需要自己区分.目前除了级别0不做验证其他都会处理.以后可能会用上所有级别,需要鉴别

-1为封禁.

0为最高级别

> 0 && < 10的都属于管理权限

> =10 \*\* <= 20的都属于VIP会员用户.充值用户应该在这个组别

>=20 的属于普通用户

## 1.4 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.

## 1.5 加密通信

加密通信是通过XCrypto技术实现的.加密通信只加密负载内容,不加密头,也就是说,无论是TCP的XENGINE协议头还是WEBSOCKET的通信头还是HTTP的头都不加密,加密只是加密负载内容

# 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.

### 2.1.1 配置环境

按照XEngine Readme文件的说明执行脚本配置环境

### 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.sln.

如果环境没有错误,编译会直接成功.包含7个模块和2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

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22.04 以上系统.RockyLinux需要9.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 MacOS

MacOS需要13以及以上版本,编译和运行方式可以参考Linux.

## 2.4 老版本升级

3.3版本升级到后续版本需要创建和修改数据库名称

需要创建Authorize\_BannedAddr和Authorize\_BannedAddr表.参考XEngine\_SQL/main.sql

修改表名称:AuthReg\_NetVer -> Authorize\_NetVer

AuthReg\_Serial -> Authorize\_Serial

AuthReg\_User -> Authorize\_User

3.10版本升级:需要修改临时试用表为新的表名称,字段也有修改,参考SQL文件

# 三 TCP接口协议

TCP接口协议用于适用于APP等应用程序验证.可用于登录操作,网络授权验证可以通过此接口进行.其他管理接口请参考HTTP接口.

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XCrypto算法.如果需要加密,请填充,否则请填充0

## 3.1 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

### 3.1.1 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

{

XCHAR tszUserName[64]; //用户名

XCHAR tszUserPass[64]; //密码

XCHAR tszDCode[8]; //动态码,最大8位

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

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

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

### 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\_REPLOGIN

unPacketSize = 0

byVersion = 0

byIsReply = FALSE

wReserve = 0

wPacketSerial = 0

wTail = XENGIEN\_COMMUNICATION\_PACKET\_PROTOCOL\_TAIL

#### 3.1.2.1 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
* 250:设备类型未知
* 251:用户不存在
* 252:密码错误
* 253:用户已经登录
* 254:用户没有权限
* 255:用户已过期
* 256:服务器内部错误

## 3.2 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.

JSON里面的unOperatorType参考TCP的unOperatorType,unOperatorCode参考TCP的unOperatorCode,这里填写的是10进制.请严格按照格式填写,目前没有写安全性验证代码

## 4.1 用户验证协议

用于登录

### 4.1.1 请求

{

"unOperatorType":2,

"unOperatorCode":8197,

"byIsReply":1,

"wCrypto":0,

"st\_UserAuth":{

"tszUserName":"123123aa",

"tszUserPass":"123123",

"enClientType":10,

"enDeviceType":25

}

}

### 4.1.2 回复

{  
    **"wHeader"**:**17**,  
    **"wTail"**:**255**,  
    **"unOperatorType"**:**2**,  
    **"unOperatorCode"**:**8198**,  
    **"wCrypto"**:**0**,  
    **"wReserve"**:**0**  
}

## 4.2 超时通知

Code值为:XENGINE\_COMMUNICATION\_PROTOCOL\_OPERATOR\_CODE\_AUTH\_TIMEDOUT

# 五 HTTP管理接口

此接口用于HTTP RESTFUL ,以后可以通过此接口来管理服务器

HTTP接口只会返回200成功,错误码需要通过负载的json获取.code和msg.

## 5.1 客户端管理接口

管理需要进行Token验证,你需要先登录得到Token值,并且附加到Json的Root(根节点)当中才可以使用.

得到Token的值参考5.4节的协议

{

"xhToken":100011213

}

### 5.1.1 获取用户

#### 5.1.1.1 请求

方法:POST

接口:http://IP:PORT/auth/client/get

负载:

{

"xhToken":100011213,

"st\_UserInfo":{

"tszUserName":"用户名"

}

}

#### 5.1.1.2 回复

{

"code":0,

"msg":"success",

"st\_UserTable":{

"tszLeftTime":"还剩下多少时间",

"tszHardCode":"硬件吗",

"enSerialType":1,

"enDeviceType":0,

"st\_UserInfo":{

"tszUserName":"用户名",

"tszUserPass":"密码",

"tszEMailAddr":"电子邮件",

"tszLoginTime":"登录时间",

"tszCreateTime":"注册时间",

"nPhoneNumber":1369999999,

"nIDNumber":5111010101011,

"nUserLevel":5,

"nUserState":0

}

}

}

### 5.1.2 用户列表

#### 5.1.2.1 请求

方法:POST

接口:http://IP:PORT/auth/client/list

负载:

{

"xhToken":100011213,

"Online":false,

"PosStart":0,

"PosEnd":100

}

#### 5.1.2.2 回复

{

"Array":[

{

"enDeviceType":0,

"enSerialType":1,

"st\_UserInfo":{

"nIDNumber":511025111111111100,

"nPhoneNumber":1366666666,

"nUserLevel":5,

"nUserState":0,

"tszCreateTime":"2022-05-27 10:09:17",

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

"tszLoginTime":"",

"tszUserName":"123123aa",

"tszUserPass":"123123"

},

"tszHardCode":"ND2N09DN1N11",

"tszLeftTime":"81"

}

],

"code":0,

"msg":"success"

}

### 5.1.3 关闭客户

#### 5.1.3.1 请求

方法:POST

接口:http://IP:PORT/auth/client/close

负载:

{

"xhToken":100011213,

"st\_UserInfo":{

"tszUserName":"用户名"

}

}

#### 5.1.3.2 回复

{

"code":0,

"msg":"success"

}

### 5.1.4 修改用户

#### 5.1.4.1 请求

方法:POST

接口:http://IP:PORT/auth/client/modify

负载:

{

"xhToken":100011213,

"st\_UserTable":{

"enSerialType":1,

"st\_UserInfo":{

"nIDNumber":511025111111111100,

"nPhoneNumber":1366666666,

"nUserLevel":5,

"tszCreateTime":"2022-05-27 10:09:17",

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

"tszUserName":"123123aa",

"tszUserPass":"123123"

},

"tszHardCode":"ND2N09DN1N11",

"tszLeftTime":"81"

}

}

#### 5.1.4.2 回复

{

"code":0,

"msg":"success"

}

## 5.2 序列号管理接口

需要Token,参考5.1

### 5.2.1 列举序列号

#### 5.2.1.1 请求

方法:POST

接口:http://IP:PORT/auth/serial/list

负载:

{

"xhToken":100011213,

"PosStart":0,

"PosEnd":100

}

#### 5.2.1.2 回复

{

"Array":[

{

"bIsUsed":0,

"enSerialType":2,

"tszCreateTime":"2022-08-11 14:45:58",

"tszMaxTime":"2",

"tszSerialNumber":"XAUTH-XYRYD-MRLJ2-YYF85-UQLKI-31115-98XVA-LQYKX-12744-KFQ38",

"tszUserName":"NOT"

}

],

"code":0,

"msg":"success"

}

### 5.2.2 插入序列号

#### 5.2.2.1 请求

方法:POST

接口:http://IP:PORT/auth/serial/list

负载:

{

"xhToken":100011213,

"st\_SerialInfo":{

"enSerialType":1,

"nNumberCount":5,

"nSerialCount":2,

"tszHasTime":"20"

}

}

enSerialType:代表序列号类型

nNumberCount:序列号字段个数

nSerialCount:序列号个数

tszHasTime:拥有的时间

#### 5.2.2.2 回复

{

"code":0,

"msg":"success"

}

### 5.2.3 删除序列号

#### 5.2.3.1 请求

方法:POST

接口:http://IP:PORT/auth/serial/delete

负载:

{

"xhToken":100011213,

"Array":[

{

"tszSerialNumber":"XAUTH-XYRYD-MRLJ2-YYF85-UQLKI-31115-98XVA-LQYKX-12744-KFQ38"

}

]

}

#### 5.2.3.2 回复

{

"code":0,

"msg":"success"

}

## 5.3 用户接口

用户接口可以不需要登录而直接使用

### 5.3.1 用户删除

用于删除一个用户

#### 5.3.1.1 请求

方法:POST

接口:http://IP:PORT/auth/user/delete

负载:

{

"st\_UserInfo" :

{

"nIDNumber" : 511025111111111111,

"nPhoneNumber" : 1366666666,

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

"tszUserName" : "123123aa",

"tszUserPass" : "123123"

}

}

#### 5.3.1.2 回复

{

"code" : 0,

"msg" : "success"

}

### 5.3.2 用户注册

#### 5.3.2.1 请求

方法:POST

接口:http://IP:PORT/auth/user/register

负载:

{

"st\_UserTable" :

{

"st\_UserInfo" :

{

"nIDNumber" : 511025111111111111,

"nPhoneNumber" : 1366666666,

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

"tszUserName" : "123123aa",

"tszUserPass" : "123123",

"nUserLevel":5

},

"tszHardCode" : "2FDWAD02JD2091"

}

}

#### 5.3.2.2 回复

{

"code":0,

"msg":"success"

}

### 5.3.3 用户充值

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.

#### 5.3.3.1 请求

方法:POST

接口:http://IP:PORT/auth/user/pay

负载:

{

"st\_UserPay" :

{

"tszSerialNumber" : "XAUTH-XYRYM-ZAA10-59NKW-KOFLP-35838-ZXC36-ZOVTH",

"tszUserName" : "123123aa"

}

}

#### 5.3.3.2 回复

{

"code":0,

"msg":"success"

}

### 5.3.4 找回密码

#### 5.3.4.1 请求

方法:POST

接口:http://IP:PORT/auth/user/pass

负载:

{

"st\_UserInfo" :

{

"nIDNumber" : 511025111111111111,

"nPhoneNumber" : 1366666666,

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

"tszUserName" : "123123aa"

}

}

#### 5.3.4.2 回复

{

"code":0,

"msg":"success",

"st\_UserAuth":{

"enClientType":0,

"enDeviceType":0,

"tszUserName":"123123aa",

"tszUserPass":"123123"

}

}

### 5.3.5 获取时间

获取时间表示获取用户所剩余的时间

#### 5.3.5.1 请求

方法:POST

接口:http://IP:PORT/auth/user/time

负载:

{

"st\_UserAuth" :

{

"tszUserName" : "123123aa",

"tszUserPass" : "123123"

}

}

#### 5.3.5.2 回复

{

"st\_UserTime":{

"enDeviceType":0,

"enSerialType":1,

"nNetType":0,

"nTimeLeft":2,

"nTimeONLine":3,

"tszLeftTime":"2022-08-25 13:56:36",

"tszUserAddr":"127.0.0.1:50168",

"tszUserName":"123123aa"

}

}

### 5.3.6 快速验证

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.

#### 5.3.6.1 请求

方法:POST

接口:http://IP:PORT/auth/user/try

负载:

{

"st\_VERTemp":{

"tszSerial":"cpuid:112233"

}

}

#### 5.3.6.2 回复

{

"code":0,

"msg":"success"

}

## 5.4 Token协议

用于管理接口权限验证管理

### 5.4.1 登录

用于获取Token值,这个登录支持管理接口和授权验证登录.

#### 5.4.1.1 请求

方法:GET

接口:http://IP:PORT/api?function=login&user=123123aa&pass=123123&device=36

参数:用户名,密码,设备类型,参考:ENUM\_PROTOCOLDEVICE\_TYPE

负载:无

#### 5.4.1.2 回复

{

"code":0,

"msg":"success",

"nTimeout":3600,

"tszTimeEnd":"2022-08-25 15:19:12",

"tszTimeStart":"2022-08-25 14:19:12",

"xhToken":1000030115

}

Token是有过期时间的,过期后不能使用管理接口

### 5.4.2 续期

如果想要继续使用这个Token,可以使用续期协议进行续期,前提是TOKEN必须存在

#### 5.4.2.1 请求

方法:GET

接口:http://IP:PORT/api?function=update&token=1000030115

负载:无

#### 5.4.2.2 回复

{

"code" : 0,

"msg" : "success",

"nTimeout" : 3600,

"tszTimeEnd" : "2022-08-25 15:31:42",

"tszTimeStart" : "2022-08-25 14:31:42",

"xhToken" : 1000032558

}

### 5.4.3 关闭

如果不想用TOKEN,可以关闭,也可以等超时自动销毁,

#### 5.4.3.1 请求

方法:GET

接口:http://IP:PORT/api?function=close&token=1000030115

负载:无

#### 5.4.3.2 回复

{

"code" : 0,

"msg" : "success"

}

## 5.5 功能开关

功能开关接口只是临时的功能,重启后无效.

### 5.5.1 获取功能列表

#### 5.5.1.1 请求

方法:POST

接口:http://IP:PORT/auth/switch/get

负载:

{

"xhToken":12345679

}

#### 5.5.1.2 回复

{

"msg":"success",

"code":0,

"bSwitchDelete":true,

"bSwitchRegister":true,

"bSwitchLogin":true,

"bSwitchPass":true,

"bSwitchPay":true,

"bSwitchTime":true,

"bSwitchCDKey":true,

"bSwitchNotice":true,

"bSwitchDCode":false,

"bSwitchMulti":false,

"bSwitchTry":true,

"bSwitchBanned":true

}

### 5.5.2 设置功能开关

#### 5.5.2.1 请求

方法:POST

接口:http://IP:PORT/auth/switch/set

负载:

{

"xhToken":123456789,

"st\_SwitchInfo":{

"bSwitchDelete":true,

"bSwitchRegister":true,

"bSwitchLogin":true,

"bSwitchPass":true,

"bSwitchPay":true,

"bSwitchTime":true,

"bSwitchCDKey":true,

"bSwitchNotice":true,

"bSwitchDCode":false,

"bSwitchMulti":false,

"bSwitchTry":true,

"bSwitchBanned":true

}

}

#### 5.5.2.2 回复

{

"msg":"success",

"code":0

}

## 5.6 黑名单协议

可以禁止IP地址登录或者禁止指定用户名注册

### 5.6.1 插入

#### 5.6.1.1 请求

方法:POST

接口:http://IP:PORT/auth/banned/insert

负载:

{

"xhToken":123456789,

"st\_Banned":{

"bEnable":true,

"tszIPAddr":"要禁用的IP地址",

"tszUserName":"要禁用的用户名,ip或者用户名选择一项",

"tszLeftTime":"过期日期"

}

}

#### 5.6.1.2 回复

{

"msg":"success",

"code":0

}

### 5.6.2 删除

#### 5.6.2.1 请求

方法:POST

接口:http://IP:PORT/auth/banned/delete

负载:

{

"xhToken":123456789,

"st\_Banned":{

"tszIPAddr":"禁用的IP地址",

"tszUserName":"禁用的用户名,ip或者用户名选择一项"

}

}

#### 5.6.2.2 回复

{

"msg":"success",

"code":0

}

### 5.6.3 查询

#### 5.6.3.1 请求

方法:POST

接口:http://IP:PORT/auth/banned/list

负载:

{

"xhToken":123456789

}

#### 5.6.2.2 回复

{

"ArrayAddr":[

{

"nID":1,

"bEnable":true,

"tszIPAddr":"192.168.1.1",

"tszLeftTime":"2023-12-29 15:00:20",

"tszCreateTime":"2022-12-29 15:00:20"

}

],

"ArrayUser":[

{

"nID":1,

"bEnable":true,

"tszLeftTime":"2023-12-29 15:00:20",

"tszCreateTime":"2022-12-29 14:57:03",

"tszUserName":"123123"

},

{

"nID":2022,

"bEnable":true,

"tszLeftTime":"2023-12-29 15:00:20",

"tszCreateTime":"adadad",

"tszUserName":"2"

}

],

"CountAddr":1,

"CountUser":2,

"code":0,

"msg":"success"

}

### 5.6.4 修改

#### 5.6.4.1 请求

方法:POST

接口:http://IP:PORT/auth/banned/modify

负载:

{

"xhToken":123456789,

"st\_Banned":{

"bEnable":true,

"tszIPAddr":"要禁用的IP地址",

"tszUserName":"要禁用的用户名,ip或者用户名选择一项",

"tszLeftTime":"过期日期"

}

}

#### 5.6.4.2 回复

{

"msg":"success",

"code":0

}

## 5.7 CDKEY系统

### 5.7.1 创建CDKey

#### 5.7.1.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/cdkey/create

负载:

{

"tszAddr": "服务器地址",

"nPort": 5000,

"st\_AuthAppInfo": {

"tszAppName": "自定义名称",

"tszAppVer": "自定义版本",

"nExecTime": 0,

"bInit": false

},

"st\_AuthRegInfo": {

"tszHardware": "硬件吗",

"tszCreateTime": "CDKEY创建时间",

"tszRegisterTime": "CDKEY注册时间",

"tszLeftTime": "CDKEY拥有时间",

"tszStartTime": "CDKEY启动时间",

"tszExpiryTime": "CDKEY过期时间",

"enSerialType": 1,

"enRegType": 1,

"enHWType": 1,

"enVModeType": 1

},

"st\_JsonSerialInfo": {

"tszTimeSerial": "XAUTH-XXX-XXX-XXX",

"nTimeCount": 10,

"tszDataSerial": "XAUTH-XX-XX-X-X",

"tszDataTime": "2022-01-01 11:11:10",

"bTimeAdd": false,

"tszUNLimitSerial": "XAUTH-X-X-X"

},

"st\_AuthUserInfo": {

"tszUserName": "注册的用户名",

"tszUserContact": "联系方式",

"tszCustom": "自定义,可选参数"

}

}

#### 5.7.1.2 回复

[Connection]

tszAddr=http://app.xyry.org

nPort=5501

[AppInfo]

tszAppName=XEngine

tszAppVer=1.0.0.1001

nExecTime=0

bInit=0

[AuthReg]

nHasTime=0

enHWType=1

enRegType=2

enSerialType=3

enVModeType=1

tszHardware=5501012NE21N

tszCreateTime=2024-07-15 14:06:20

tszRegisterTime=

tszLeftTime=

tszStartTime=

tszExpiryTime=

[AuthSerial]

tszTimeSerial=XAUTH-CS3GW-KFNQD-06481-O13KP-REPGG-63723

nTimeCount=9999

tszDataSerial=XAUTH-YT2YY-YCQPE-67536-MNMKS-VWSRE-12183

tszDataTime=2025-07-15 14:06:20

bAddTime=0

tszUNLimitSerial=XAUTH-FTSYS-FADPO-16623-CE67A-BOYGP-77135

[AuthUser]

tszUserName=qyt

tszUserContact=486179@qq.com

tszCustom=

### 5.7.2 注册CDKey

#### 5.7.2.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/cdkey/auth

负载:创建返回的CDKEY

#### 5.7.2.2 回复

回复授权成功的CDKey文件内容

### 5.7.3 验证CDKey

#### 5.7.3.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/cdkey/ver

负载:

要验证的CDKey内容

#### 5.7.3.2 回复

成功返回验证的CDKey失败返回错误信息

## 5.8 公告协议

公告协议的字符串需要UTF8格式

### 5.8.1 插入

#### 5.8.1.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/notice/insert

负载:

{

"xhToken":123456789,

"st\_Notice":{

"tszContext":"公告内容"

}

}

#### 5.8.1.2 回复

{

"msg":"success",

"code":0

}

### 5.8.2 删除

#### 5.8.2.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/notice/delete

负载:

{

"xhToken":123456789,

"st\_Notice":{

"nID":1

}

}

#### 5.8.2.2 回复

{

"msg":"success",

"code":0

}

### 5.8.3 列举

#### 5.8.3.1 请求

方法:GET

参数:无

地址:http://127.0.0.1:5302/auth/notice/list

负载:无

#### 5.8.3.2 回复

{

"Array":[

{

"nID":1,

"tszContext":"公告1",

"tszCreateTime":"2023-02-24 11:24:26"

},

{

"nID":2,

"tszContext":"公告2",

"tszCreateTime":"2023-02-24 11:27:30"

}

],

"Count":2,

"code":0,

"msg":"success"

}

## 5.9 动态验证码

可以在登录的时候启用动态验证码,防止暴力破解.

启用动态验证码你必须使用返回的token来填充你的登录协议,否则验证将失败

### 5.9.1 获取

#### 5.9.1.1 请求

方法:GET

参数:无

地址:http://127.0.0.1:5302/api?function=dcode&user=get

负载:无

#### 5.9.1.2 回复

{

"code":0,

"msg":"success",

"nDynamicCode":803798,

"nTimeout":30,

"tszTimeEnd":"2023-04-18 16:56:14",

"tszTimeStart":"2023-04-18 16:55:44",

"xhToken":3411647435

}

## 5.10 临时验证协议

### 5.10.1 列举

#### 5.10.1.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/try/list

负载:

{

"xhToken":123456789,

"PosStart":0,

"PosEnd":100

}

#### 5.10.1.2 回复

{

"Array" :

[

{

"enVMode" : 2,

"nID" : 2,

"nLTime" : 4,

"nVTime" : 5,

"tszVDate" : "2024-01-09 14:22:42",

"tszVSerial" : "C6DS090081035"

}

],

"Count" : 1,

"code" : 0,

"msg" : "success"

}

### 5.10.2 删除

#### 5.10.2.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/try/delete

负载:

{

"xhToken":123456789,

"st\_VERTemp":{

"tszVSerial":"C6DS090081035"

}

}

#### 5.10.2.2 回复

{

"msg":"success",

"code":0

}

### 5.10.3 修改

#### 5.10.3.1 请求

方法:POST

参数:无

地址:http://127.0.0.1:5302/auth/try/modify

负载:

{

"xhToken":123456789,

"st\_VERTemp":{

"enVMode":2,

"nID":1,

"nLTime":5,

"nVTime":5,

"tszVDate": "2024-01-09 14:22:42",

"tszVSerial":"C6DS090081035"

}

}

#### 5.10.3.2 回复

{

"msg":"success",

"code":0

}

# 六 三方验证

从3.2版本后,验证服务可以通过三方验证接口来处理用户登录登出和超时等,如果你想要自己管理用户表,那么你需要使用这个功能,如果启用了这个功能,那么服务器的用户数据库将不在有作用

三方验证采用的是HTTP POST方法,将根据配置文件提交到指定的URL

## 6.1 用户登录

### 6.1.1 请求

方法:POST

接口:配置文件tszPassLogin

负载:5.3.4的回复JSon

### 6.1.2 回复

HTTP 200代表成功,其他值代表失败.

如果成功,你需要回复我们一个完成的用户信息表,参考5.1.4 JSON结构,需要填充所有字段.

## 6.2 用户登出

### 6.2.1 请求

方法:POST

接口:配置文件tszPassLogout

负载:5.3.5的回复JSon

### 6.2.2 回复

系统不关心回复信息

## 6.3 用户超时

### 6.3.1 请求

方法:POST

接口:配置文件tszPassTimeout

负载:5.3.5的回复JSon

### 6.3.2 回复

系统不关心回复内容

# 七 高级功能

## 7.1 分布式验证

分布式验证默认开启,如果用户想要分布式验证,可以通过post的auth/pass/接口提交到我们服务器,服务器将通过验证后返回给验证端.

服务器可以处理登录,注销,到期

你也可以通过6大节来作为验证转发服务.新版本提供了更为灵活的验证方式

## 7.2 多端登录

你可以再Switch的配置文件中打开多端登录开关.

多端登录现在支持分钟和次数卡,多端登录设备类型必须正确填写,他是根据登录的设备类型来确认多端登录类型的.

多端登录的分钟可以合并到一起,次数会每次登录就减去一.

自定义和天数卡按照到期时间处理,不建议开启.

## 7.3 序列卡说明

在序列号管理页面,有一些字段需要填写,其中序列号数量代表生成的序列号的字段数,比如4就是 xxx-xxx-xx-xxx 5就是:xxx-xxx-xxx-xxx-xxx

生成卡数量是要生成多少张卡.卡拥有时间根据类型有所不同,参考下面的信息

不支持用户更换充值卡类型,更换会直接删除以往的时间

### 7.3.1 秒卡

SECOND类型,秒卡,代表用户在线秒数.卡拥有时间可以直接填写值,比如填写60,代表用户可在线60秒.

开启多端登录后可以合并用时,如果有两个用户在线,那么会翻倍减少可用时间,比如一个用户可以在线60秒,两个用户只有30秒.

### 7.3.2 次数卡

TIME类型,次数卡,代表可用次数,每次登录后会减1,这种类型卡和在线时间无关.也就是说登录后无论在线多久都不会过期.除非下次登录才会验证.

这种卡类型同样适用于多端登录,每次登录减少一次可用登录次数.

卡拥有时间就是可以使用的次数,可以直接填写值,比如60,就是代表可以登录60次.

### 7.3.3 天数卡

DAY类型.天数卡,按照天扣除,设置拥有天数.当天登录无论多少次,都只扣除一天的天数,比如拥有5天,当天5号登录过,扣除一次,剩余4天,那么5号当天无论登录多少次都剩余4天.

### 7.3.4 自定义日期卡

CUSTOM卡类型,自定义时间卡,卡拥有时间为到期日期,请按照固定格式填写,如:yyyy-mm-dd hh:mm:ss,比如到期时间2023-11-20 10:10:01.

此卡类型不合并用时将不做任何处理,开启后,没有效果,只是允许了多端登录而已,请用户自己考虑是否开启此选项.

## 7.4 客户端SDK

客户端SDK:AuthorizeModule\_Client 模块,此模块封装了登录接口,可以用于处理登录和超时判断.

通过AuthClient\_Connector\_Connect链接上服务器后.在通过AuthClient\_Connector\_Login函数登录.登录成功后可以通过AuthClient\_Connector\_GetAuth函数来判断验证是否超时还是离线.

在E语言例子中就使用了此模块来实现验证.需要把这个模块进行x86-release编译才可以使用.

请注意:密码只能是数字或者字母.在不使用的时候需要关闭.

你需要依赖以下模块才能正常使用此模块的函数,你也可以通过编译选项来减少依赖模块.

XEngine\_BaseLib.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_TCP)(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_HTTP)

XEngine\_Algorithm.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_TCP)

XEngine\_OPenSsl.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_TCP)

XClient\_Socket.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_TCP)

XClient\_APIHelp.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_HTTP)

XEngine\_SystemApi.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_HTTP)

jsoncpp.dll(\_XAUTH\_BUILD\_SWITCH\_CLIENT\_HTTP)

## 7.5 CDKey本地验证

CDKEY可以网络验证和本地验证,无论网络验证还是本地验证,CDKEY文件应该加密.网络验证应该按照服务端要求传递明文或者密文数据

### 7.5.1 通过网络

可以通过网络创建和验证,我们不推荐通过网络授权CDKEY,因为不好控制或者说现阶段还不行.

### 7.5.2 通过本地

这是我们推荐的方法,由用户创建一个CDKEY到本地文件,然后把文件发送给管理员,管理员得到文件后通过管理程序打开这个CDKEY进行授权.

#### 7.5.2.1 用户自验证

在创建CDKEY的时候可以写序列信息,这样,用户可以自己实现注册CDKEY而完全不需要管理人员干预.

可以调用Authorize\_CDKey\_UserRegister这个函数实现用户自注册,使用者可以自己编写一套界面程序实现,可参考我们的例子实现

## 7.6 数据库

我们支持两种数据库,一个SLQITE一个MYSQL,必须也只能选择一种数据库进行使用

### 7.6.1 SQLITE

SQLITE是本地数据库,轻量级用户可以选择这个,非常的方便,我们提供了默认的数据库文件,你不需要特定的操作就可以直接启动服务端.

### 7.6.2 MYSQL

MYSQL需要自己配置环境,并且导入XEngine\_SQL目录下的mysql.sql数据库文件才可以使用.

# 八 配置说明

## 8.1 基本配置文件

配置文件:XEngine\_Config.json

### 8.1.1 基础配置

* nTCPPort:TCP监听的端口
* nWSPort:websocket端口
* nHTTPPort:HTTP端口
* bDeamon:守护进程模式
* bTimeNotify:超时通知设置,为true为一直通知直到客户端主动关闭

### 8.1.2 最大配置

配置信息:XMax

* nMaxClient:最大客户端数量
* nMaxQueue:最大队列数量
* nIOThread:网络IO线程
* nTCPThread:TCP业务线程
* nWSThread:WEBSOCKET业务线程
* nHTTPThread:HTTP业务线程

### 8.1.3 验证配置

配置信息:XVerification,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

* nUserTimeout:用户登录超时时间
* nTokenTimeout:Token超时时间,单位秒
* nVerTime:快速验证时间
* nVerMode:快速验证模式
* nTryTime:试用时间
* nTryMode:试用模式
* st\_XCDKey:示例用CDKEY配置

### 8.1.4 登录配置

配置信息:XLogin

* bHTTPAuth:是否启用HTTP授权验证,需要单独的HTTP心跳
* bPassAuth:是否启用三方验证
* nMultiMode:多端登录模式,0按照分类验证,1按照子类型,2不限制类型
* st\_PassUrl:HTTP三方验证接口
* st\_MultiLogin:可以使用的多端登录的客户端支付类型

### 8.1.5 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

* bEnable:是否启用
* nPass:密码,仅允许数字

### 8.1.6 数据库配置

配置信息:XSql

* nSQLType:0 使用SQLITE,1使用MYSQL
* SQLAddr:MYSQL数据库地址
* SQLPort:MYSQL端口
* SQLUser:MYSQL用户
* SQLPass:MYSQL密码
* SQLFile:SQLITE数据库地址

### 8.1.7 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

### 8.1.8 报告配置

XReport配置

* bEnable:是否启用
* tszServiceName:服务名称
* tszAPIUrl:报告地址

## 8.2 功能开关配置

配置文件:XEngine\_SwitchConfig.json

* bSwitchDelete:用户注销开关
* bSwitchRegister:用户注册开关
* bSwitchLogin:用户登录开关
* bSwitchPay:用户充值开关
* bSwitchPass:用户找回密码开关
* bSwitchTime:用户获取时间开关
* bSwitchCDKey:CDKey授权功能开关
* bSwitchNotice:公告功能开关
* bSwitchDCode:动态验证码功能开关
* bSwitchMulti:多端登录功能开关
* bSwitchTry:临时试用功能开关
* bSwitchBanned:是否启用黑名单

# 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