XEngine网络验证服务文档

目录

[XEngine网络验证服务文档 1](#_Toc31115)

[前言 4](#_Toc6543)

[阅读者 4](#_Toc12569)

[概述 4](#_Toc26797)

[相关模块 4](#_Toc22664)

[一 技术结构 4](#_Toc20896)

[1.1 目录结构 4](#_Toc10003)

[1.2 程序目录 4](#_Toc21624)

[1.3 用户级别 5](#_Toc13117)

[1.4 协议解释 5](#_Toc20865)

[二 配置环境 5](#_Toc14514)

[2.1 WINDOWS 5](#_Toc10259)

[2.1.1 配置环境 5](#_Toc27199)

[2.1.2 编译运行 5](#_Toc13634)

[2.2 LINUX 5](#_Toc6231)

[2.2.1 环境配置 5](#_Toc30897)

[2.2.2 编译运行 6](#_Toc30981)

[2.3 MacOS 6](#_Toc6482)

[2.4 版本要求 6](#_Toc1089)

[2.4.1 系统版本 6](#_Toc4319)

[2.4.2 软件需求 6](#_Toc12279)

[三 TCP接口协议 6](#_Toc15430)

[3.1 登录协议 7](#_Toc28999)

[3.1.1 请求 7](#_Toc16487)

[3.1.2 回复 7](#_Toc21704)

[3.2 超时通知 8](#_Toc28118)

[四 Websocket接口协议 8](#_Toc19917)

[4.1 用户验证协议 9](#_Toc25461)

[4.1.1 请求 9](#_Toc7430)

[4.1.2 回复 9](#_Toc28456)

[4.2 超时通知 9](#_Toc28913)

[五 HTTP管理接口 9](#_Toc25848)

[5.1 客户端管理接口 10](#_Toc14063)

[5.1.1 获取用户 10](#_Toc10322)

[5.1.2 用户列表 11](#_Toc2565)

[5.1.3 关闭客户 12](#_Toc16459)

[5.1.4 修改用户 12](#_Toc21639)

[5.2 序列号管理接口 13](#_Toc22298)

[5.2.1 列举序列号 13](#_Toc9282)

[5.2.2 插入序列号 14](#_Toc17941)

[5.2.3 删除序列号 14](#_Toc27879)

[5.3 用户接口 15](#_Toc6911)

[5.3.1 用户删除 15](#_Toc4794)

[5.3.2 用户注册 16](#_Toc10350)

[5.3.3 用户充值 16](#_Toc10200)

[5.3.4 找回密码 17](#_Toc638)

[5.3.5 获取时间 18](#_Toc22671)

[5.3.6 快速验证 18](#_Toc4675)

[5.4 Token协议 19](#_Toc10144)

[5.4.1 登录 19](#_Toc32142)

[5.4.2 续期 20](#_Toc24931)

[5.4.3 关闭 20](#_Toc19963)

[六 三方验证 21](#_Toc9500)

[6.1 用户登录 21](#_Toc22893)

[6.1.1 请求 21](#_Toc7695)

[6.1.2 回复 21](#_Toc16218)

[6.2 用户登出 21](#_Toc4623)

[6.2.1 请求 21](#_Toc10654)

[6.2.2 回复 22](#_Toc22936)

[6.3 用户超时 22](#_Toc13478)

[6.3.1 请求 22](#_Toc13754)

[6.3.2 回复 22](#_Toc16947)

[七 高级功能 22](#_Toc23135)

[7.1 分布式验证 22](#_Toc21257)

[八 配置说明 22](#_Toc30637)

[8.1 基础配置 22](#_Toc10056)

[8.2 最大配置 23](#_Toc9111)

[8.3 验证配置 23](#_Toc8759)

[8.4 登录配置 23](#_Toc19676)

[8.5 加密配置 23](#_Toc5584)

[8.6 数据库配置 23](#_Toc27388)

[8.7 日志配置 24](#_Toc31568)

[附录 24](#_Toc31477)

[附录1 类型定义 24](#_Toc2452)

[附录2 协议定义 24](#_Toc5826)

[附录3 转换定义 24](#_Toc24328)

[附录4 更新历史 24](#_Toc9713)

|  |  |  |  |
| --- | --- | --- | --- |
| 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： | V3.2.0.1001 | |
| 发布日期： | 2022-09-23 | |
| 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

服务器启动会绑定三个端口,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

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

## 用户级别

用户一共有0-5个级别,-1为封禁.

0为最高级别,5为最低.默认注册级别为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 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5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 MacOS

MacOS需要12以及以上版本,编译和运行方式可以参考Linux.

## 2.4 版本要求

### 2.4.1 系统版本

最低版本要求:

WINDOWS: win7 sp1

Ubuntu:20.04

Centos:8.x

MacOS:12

### 2.4.2 软件需求

最低版本要求:

XEngine:V7.38

# 三 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

{

*CHAR* tszUserName[64]; //用户名

*CHAR* tszUserPass[64]; //密码

ENUM\_PROTOCOLCLIENT\_TYPE enClientType; //客户端类型,可0

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

}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

后面的协议不在单独列举xhToken字段.

{

"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:

{

"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:

#### 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/get

负载:

{

"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:

{

"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:无

#### 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:

{

"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:

{

"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\_UserTry":{

"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

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

注意:如果是普通用户接口,你需要update更新token并且你需要通过5.3.5获取用户超时,除非你设置了三方验证,否则不会主动通知你超时.

### 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,如果是授权验证用户,那么关闭这个TOKEN相当于退出使用.服务器会自动更新时间.也就是,权限级别大于1的用户将更新时间.

#### 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"

}

# 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

# 八 配置说明

配置文件:XEngine\_Config.json

## 8.1 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

## 8.2 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

## 8.3 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

## 8.4 登录配置

配置信息:XLogin

* bMultiLogin:是否允许多端登录
* 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
* nHTTPAuthTime:HTTP授权验证的超时时间,单位秒
* tszPassLogin:用户登录
* tszPassLogout:用户登出
* tszPassTimeout:用户超时

## 8.5 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

## 8.6 数据库配置

配置信息:XSql

* tszSQLite:数据库地址

## 8.7 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

# 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 |