XEngine网络存储服务文档

目录

[XEngine网络存储服务文档 1](#_Toc11469)

[前言 4](#_Toc25870)

[阅读者 4](#_Toc3966)

[概述 4](#_Toc30916)

[相关模块 4](#_Toc7324)

[一 技术结构 4](#_Toc1480)

[1.1 上传接口 4](#_Toc12568)

[1.2 下载接口 4](#_Toc3041)

[1.3 管理接口 4](#_Toc29132)

[二 配置环境 5](#_Toc30693)

[2.1 WINDOWS 5](#_Toc11769)

[2.1.1 配置环境 5](#_Toc8916)

[2.1.2 编译运行 5](#_Toc28274)

[2.2 LINUX 6](#_Toc32438)

[2.2.1 环境配置 6](#_Toc24447)

[2.2.2 编译运行 6](#_Toc1939)

[2.3 版本要求 7](#_Toc29310)

[2.3.1 系统版本 7](#_Toc250)

[2.3.2 软件需求 7](#_Toc15807)

[三 接口协议 7](#_Toc3315)

[3.1 查询协议 7](#_Toc5183)

[3.1.1 文件列表 7](#_Toc20305)

[3.2 三方接口 8](#_Toc23589)

[3.2.1 用户验证 8](#_Toc10430)

[3.2.2 上传代理 8](#_Toc8746)

[3.2.3 完成通知 9](#_Toc12020)

[四 配置说明 9](#_Toc6491)

[4.1 基本配置 9](#_Toc29701)

[4.2 最大配置 9](#_Toc29143)

[4.3 时间配置 9](#_Toc31906)

[4.4 日志配置 10](#_Toc1644)

[4.5 数据库配置 10](#_Toc1463)

[4.6 存储配置 10](#_Toc8268)

[4.7 代理配置 10](#_Toc31710)

[4.7.1 验证代理 10](#_Toc18730)

[4.7.2 完成通知 10](#_Toc23411)

[4.8 限制配置 11](#_Toc10110)

[4.9 版本配置 11](#_Toc9815)

[附录 11](#_Toc32205)

[附录1 类型定义 11](#_Toc17563)

[附录2 协议定义 11](#_Toc27925)

[附录3 转换定义 11](#_Toc4008)

[附录4 更新历史 11](#_Toc23516)

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布 | 文件名称： | XEngine网络存储服务文档 | |
| 适用版本： | V1.3 | |
| 发布日期： | 2021-07-23 | |
| 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

采用的是HTTP协议来作为基础通信协议.上传,下载,管理接口都是HTTP

服务器启动会绑定三个端口,来作为区分上传,下载,管理.

三个端口不可以跨端口使用,每个端口做的事情是固定的

支持第三方服务接口,支持NGINX作为上传和下载引擎.

## 上传接口

上传接口采用HTTP实现,上传接口需要使用HTTP的PUT方法实现.

上传接口不需要使用form-data头字段.

上传的路径需要由用户通过接口创建或者指定.

上传的路径为PUT /dir/name HTTP/1.1 取中间的URL地址

## 下载接口

下载接口通过GET实现.可以通过查询得到NGINX的下载地址

下载的文件通过 GET /dir/name HTTP/1.1 中间的URL地址获得

## 管理接口

管理接口通过POST实现,POST可以为空的负载信息,可以带JSON作为负载信息.根据每个接口的不同,负载的内容也不同,但是都是通过JSON来实现的

管理接口需要使用 /api/type/name 固定三元组形势提供.表示API版本,API类型,API名称.

# 二 配置环境

## 2.1 WINDOWS

需要下载XEngine.

通过VS2019来编写代码,编译,调试.

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

### 2.1.1 配置环境

下载XEngine后,如果你是压缩包下载,需要解压,解压后,添加用户环境变量

你需要在你的系统当中,添加下面两个用户环境变量

* XEngine\_Include  XEngine的头文件目录地址
* XEngine\_Library  XEngine的库目录文件地址

比如:



你还需要JSONCPP环境,你可以通过VCPKG来安装JSONCPP.

也可以自己下载:https://github.com/open-source-parsers/jsoncpp/

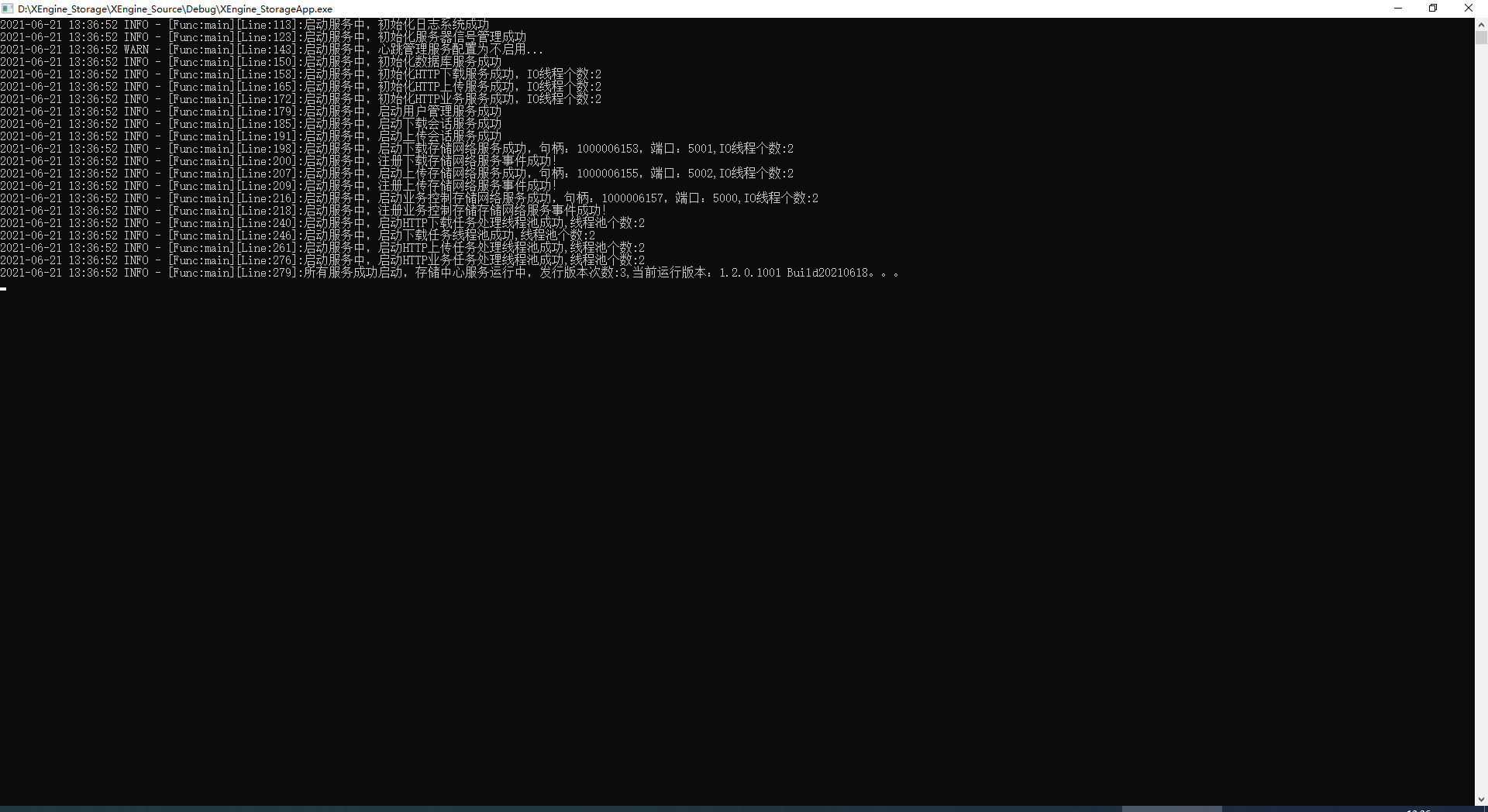
如果是自己下载,你需要自己安装和编译.并且在我们的项目中-属性-VC++目录 配置你的JSONCPP目录才可使用.

### 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VS2019打开XEngine\_StorageApp.sln,然后选择x86 debug 编译.

如果环境没有错误,编译会直接成功.包含5个模块和一个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模块与JSONCPP模块到你的编译目录下,运行XEngine\_StorageApp.exe即可.如果没有错误,你可以看到下面的界面信息



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

### 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STREAM版本)

在下载完毕后,你可能需要解压,解压后,在目录里面找到脚本安装文件并且执行下面的命令:

sudo XEngine\_RunEnv.sh -i 3

即可完成XEngine的环境配置.

当然,还需要JSONCPP的环境,你可以通过命令安装

Ubuntu:sudo apt install libjsoncpp-devel -y

Centos:sudo dnf install jsoncpp-devel -y

### 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\_StorageApp

然后直接在终端运行它即可.如果没有错误,你可以看到下面的信息:



## 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:V7.14

JsonCpp:V1.9.2

# 三 接口协议

我们的接口都是POST接口协议,也就是业务端口.

## 3.1 查询协议

### 3.1.1 文件列表

查询接口:/api/query/file

负载内容:可空或者如下:

{  
    **"lpszTimeStart"**:**"开始时间,可NULL"**,  
    **"lpszTimeEnd"**:**"结束时间,可NULL"**,  
    **"lpszFileName"**:**"文件名,可NULL"**,  
    **"lpszFileHash"**:**"文件HASH,可NULL"**  
}

## 3.2 三方接口

三方接口是用来接入第三方服务器的接口,目前支持NGINX的nginx upload module模块上传接口以及NGINX的下载代理转发接口

### 3.2.1 用户验证

用户验证目前仅仅支持BASIC的HTTP验证.你可以通过配置我们的UserList.txt文件来实现用户验证.也可以使用HTTP PASS代理来实现验证.

#### 3.2.1.1 本地验证

在XEngine\_Config文件夹的UserList.txt里面有用户列表.

每一行代表一个用户,中间用空格分开,前面表示用户名,后面表示密码.你只需要配置这个文件即可实现HTTP 基本的验证.没有验证通过将无法继续操作.

#### 3.2.1.2 代理验证

代理验证是由服务器发送一条HTTP POST消息给一个指定地址,返回200才表示成功,其他值表示失败.

发送的HTTP POST代理地址由用户指定,负载内容由服务器指定,负载为JSON格式,内容如下:

{  
    **"lpszPostUrl"**:**"客户端提交的URL"**,  
    **"lpszClientAddr"**:**"客户端的IP地址"**,  
    **"lpszUser"**:**"用户"**,  
    **"lpszPass"**:**"密码"**  
}

如果成功,那么需要返回200.

### 3.2.2 上传代理

nginx上传是用的nginx upload module 实现的.在你所有安装配置成功后,需要使用

proxy\_pass <http://192.168.1.7:5000/Api/Event/UPFile;> 指向我们的服务器.服务器会接受到代理转发的结果,并且返回.

### 3.2.3 完成通知

完成通知表示服务器在接受到一个上传和下载请求处理完毕后,是否需要给指定服务发送一个HTTP POST协议的通知.可以通过配置文件配置.

完成通知的内容负载是相同的,根据URL来区分,负载信息如下:

{  
    **"lpszFileName"**:**"文件名"**,  
    **"lpszFileHash"**:**"文件HASH,可能为NULL"**,  
    **"lpszClientAddr"**:**"请求的客户端地址"**,  
    **"nFileSize"**:**33333**  
}

# 四 配置说明

## 4.1 基本配置

* bDeamon: 1为守护进程后台运行0为前台运行
* nCenterPort:业务处理端口,HTTP POST处理端口
* nStorageDLPort:下载端口GET 协议
* nStorageUPPort:上传端口 PUT协议

## 4.2 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
* CenterThread:业务处理线程个数
* nStorageUPThread:上传处理线程个数
* nStorageDLThread:下载处理线程个数

## 4.3 时间配置

XTime配置

* bHBTime是否启用,1为启用0不启用心跳
* nDBMonth:数据库保存时间,默认月
* nTimeCheck:检测次数
* nStorageTimeOut:间隔多久检查一次

## 4.4 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

## 4.5 数据库配置

XSql 配置.需要MYSQL的支持

## 4.6 存储配置

XStorage 配置

* nUseMode:0,使用自己,1下载重定向技术.2上传,3业务,4所有
* nHashMode:HASH算法,1MD5,2HASH1,具体值查看OPENSSL的定义
* bRename:是否自动改名和路径,对于NGINX上传的文件有效
* tszHttpAddr:HTTP 业务服务重定向地址
* tszNginAddr:NGINX上传和下载服务重定向地址
* tszFileDir:保存的路径

## 4.7 代理配置

XProxy 配置

### 4.7.1 验证代理

XProxyAuth 配置

* bAuth:是否启用验证
* tszUserList: 用户列表地址
* tszAuthProxy:使用远程验证,通过HTTP POST来验证,参考3.2.1.2,如果为空表示使用本地列表,否则请填写一个HTTP地址

### 4.7.2 完成通知

XProxyPass 配置

* bUPGet:是否启用上传完成通知
* bDLGet:是否启用下载完成通知
* tszUPPass:用户上传完成通知地址
* tszDLPass:用户下载完成通知地址

注意:配置了这个,服务将等待你的返回结果,并且根据是否是200来返回成功和失败给客户端

## 4.8 限制配置

XLimit 配置

* nMaxUPLoad:最大上传速率.0不限制,单位字节(BYTE)
* nMaxDNLoad:最大下载速率,同上

## 4.9 版本配置

XVer 配置:用户显示当前版本号

# 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 只有LINUX下才有!

## 附录4 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新说明 | 修改者 | 适用版本 |