XEngine后台控制服务文档

目录

[XEngine后台控制服务文档 1](#_Toc28277)

[前言 4](#_Toc8777)

[阅读者 4](#_Toc1769)

[概述 4](#_Toc9250)

[相关模块 4](#_Toc18644)

[一 技术结构 4](#_Toc3224)

[1.1 TCP 4](#_Toc30449)

[1.2 HTTP 4](#_Toc30056)

[二 配置环境 4](#_Toc26158)

[2.1 WINDOWS 4](#_Toc8521)

[2.1.1 配置环境 5](#_Toc4578)

[2.1.2 编译运行 5](#_Toc15306)

[2.2 LINUX 6](#_Toc26281)

[2.2.1 环境配置 6](#_Toc22841)

[2.2.2 编译运行 6](#_Toc8500)

[2.3 版本要求 6](#_Toc16052)

[2.3.1 系统版本 6](#_Toc18577)

[2.3.2 软件需求 6](#_Toc31698)

[三 接口协议 7](#_Toc18356)

[3.1 后台服务协议 7](#_Toc27273)

[3.1.1 协议处理 7](#_Toc2124)

[3.1.2 后台管理协议 7](#_Toc3890)

[3.2 RPC协议 10](#_Toc16672)

[3.2.1 请求 10](#_Toc25063)

[3.2.2 回复 11](#_Toc31888)

[四 配置说明 11](#_Toc5001)

[4.1 服务器配置 11](#_Toc16369)

[4.1.1 基本配置 11](#_Toc13357)

[4.1.2 时间配置 11](#_Toc17107)

[4.1.3 客户端配置 11](#_Toc3523)

[4.1.4 日志配置 12](#_Toc23928)

[4.1.5 版本配置 12](#_Toc23866)

[4.1.6 RPC配置 12](#_Toc20483)

[4.2 进程守护配置 12](#_Toc9576)

[五 高级配置 12](#_Toc20524)

[5.1 邮件报告 12](#_Toc29304)

[5.1.1 配置文件 13](#_Toc31096)

[5.1.2 生成配置 13](#_Toc22976)

[5.2 服务配置 13](#_Toc23619)

[5.2.1 Windows 13](#_Toc8474)

[5.2.2 Linux 13](#_Toc2599)

[附录 14](#_Toc4666)

[附录1 类型定义 14](#_Toc12927)

[附录2 协议定义 14](#_Toc24993)

[附录3 转换定义 14](#_Toc14360)

[附录4 更新历史 14](#_Toc3598)

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布 | 文件名称： | XEngin后台服务文档 | |
| 适用版本： | V3.1 | |
| 发布日期： | 2021-12-17 | |
| 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.也支持TCP和UDP

## TCP

TCP协议.TCP协议速度更快,功能更多.

## HTTP

控制消息通过通过POST方法获取

RPC使用HTTP作为通信协议

# 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/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.



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/

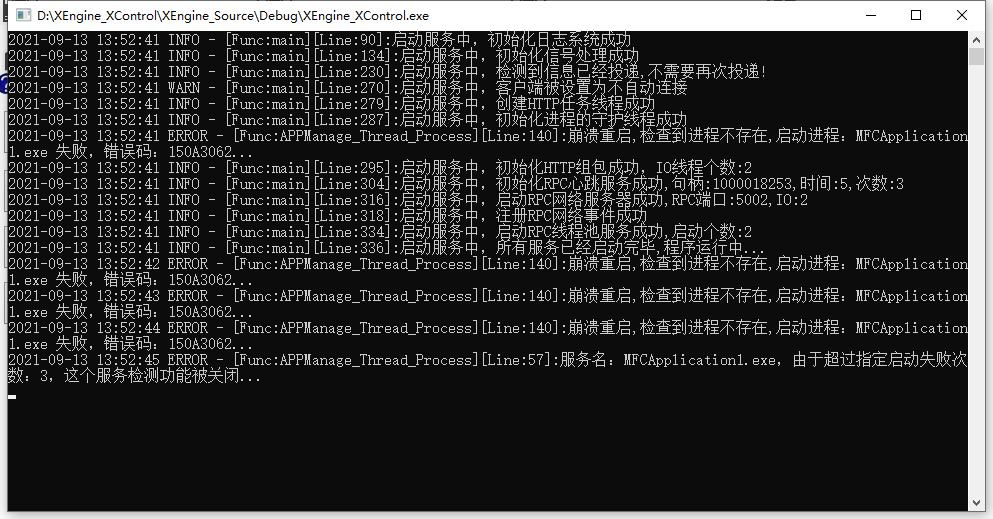
如果是自己下载,你需要自己安装和编译.并且在我们的项目中-属性-VC++目录 配置你的JSONCPP目录才可使用.

### 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.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3个模块和2个程序(LINUX只有一个)

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\_XControl.exe即可.如果没有错误,你可以看到下面的界面信息



注意：你可以使用VSCopy.bat脚本来实现自动拷贝依赖的XEngine模块到你的程序目录中.

## 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\_LINEnv.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

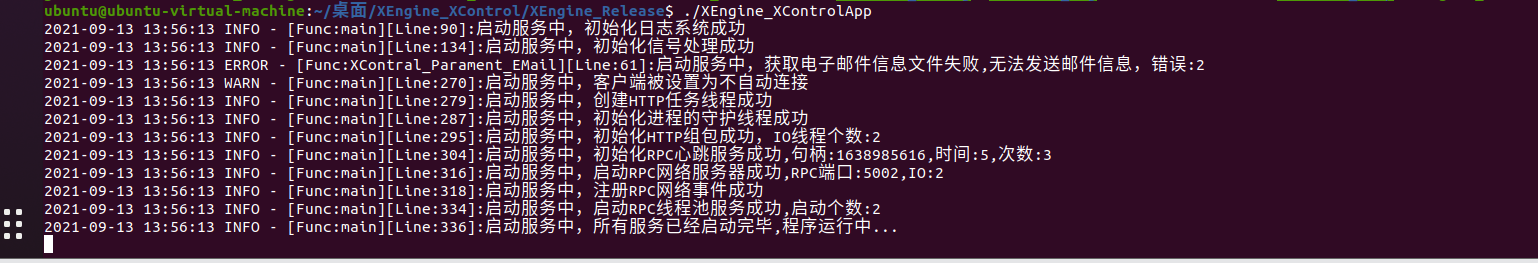
编译:make

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\_XControlApp

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:V7.21

JsonCpp:V1.9.4

# 三 接口协议

## 3.1 后台服务协议

后台服务协议用于后台程序服务,适用于XEngine\_XControl服务,可以通过此服务对一个机器进行管理.后台服务协议没有回复确认包

后台服务协议采用HTTP RESTFUL API.请求-应答模式

### 3.1.1 协议处理

注意:每种协议的JSON里面都需要带上”unOperatorCode”:0x5001 字段.这个字段代表需要处理的协议类型.这个类型可以通过协议头找到,比如0x5001表示下载.

### 3.1.2 后台管理协议

#### 3.1.2.1 下载协议

{  
    **"DownloadUrl"**:**"[http://www.xyry.org/file.exe](http://www.xyry.org/file.exe" \t "https://www.json.cn/_blank)"**,  
    **"SaveUrl"**:**"C:\1\file.exe"**  
}

#### 3.5.2.2 删除文件协议

{  
    **"DeleteFile"**:**"C:\1\file.exe"**  
}

#### 3.1.2.3 删除文件夹协议

{  
    **"DeleteDir"**:**"C:\1\"**  
}

#### 3.1.2.4 上传文件协议

上传文件协议使用FTP进行上传文件,你必须准备好一个FTP服务器,当指令被客户端收到后,就会立即上传文件.

{  
    **"UPLoadFile"**:**"C:\1\file.exe"**,  
 **"UPLoadUrl"**:**"ftp://192.168.0.1/1.rar,用户名密码格式：ftp://ftpuser:123123@192.168.1.103/1.rar"**  
}

#### 3.1.2.5 获取文件列表协议

##### 3.1.2.5.1 请求

{  
    **"PostUrl"**:**"[http://www.xyry.org/api/listfile](http://www.xyry.org/api/listfile" \t "https://www.json.cn/json/_blank)"**,  
    **"FilePath"**:**"C:\1\file.exe"**  
}

##### 3.1.2.5.2 回复

如果成功,那么将发送一个HTTP POST请求到你指定的POSTURL地址中.并且负载如下格式的JSON内容

{  
    **"Count"**:**2**,  
    **"List"**:[  
        {  
            **"FileName"**:**"1.exe"**,  
            **"FileDir"**:**"C:\1\"**  
        },  
        {  
            **"FileName"**:**"2.exe"**,  
            **"FileDir"**:**"C:\1\"**  
        }  
    ]  
}

#### 3.1.2.6 创建进程

协议体:ExecShow,1代表前端显示,0代表运行到后台

{  
    **"ExecShow"**:**1**,  
    **"ExecFile"**:**"C:\1\1.exe"**  
}

#### 3.1.2.7 结束进程

{  
    **"tszProcessName"**:**"1.exe"**  
}

#### 3.1.2.8 发送消息

此协议暂时不支持LINUX版本

{  
    **"MessageBox"**:**"弹出消息"**  
}

#### 3.1.2.9 关闭系统

协议体:支持的类型参考系统模块的SystemApi\_System\_Shutdown函数参数

{  
    **"ShutDownType"**:1  
}

#### 3.1.2.10 执行命令

{  
    **"tszExecCmd"**:**"要执行的命令"**  
}

#### 3.1.2.11 连接

{  
    **"tszIPAddr"**:**"ip地址"**,  
    **"nPort"**:**32**,  
    **"nIPType"**:**2**  
}

nIPType表示TCP还是UDP,需要按照socket头文件定义类型来设置

## 3.2 RPC协议

目前RPC协议采用HTTP+JSON方式,没有使用protobuf等模式,将在以后加入支持protobuf.

### 3.2.1 请求

请求方法:POST

请求路径:无

请求内容:

{  
    **"XRpc\_FunctionName"**:**"XEngine\_AddMethod"**,  
    **"XRpc\_ParamCount"**:**2**,  
    **"XRpc\_ParameteArray"**:[  
        {  
            **"ParameteLen"**:**4**,  
            **"ParameteType"**:**3**,  
            **"ParameteValue"**:**1**  
        },  
        {  
            **"ParameteLen"**:**4**,  
            **"ParameteType"**:**3**,  
            **"ParameteValue"**:**2**  
        }  
    ],  
    **"XRpc\_ReturnType"**:**3**  
}

名词解释:

* XRpc\_FunctionName:要调用的函数名称
* XRpc\_ParamCount:参数个数
* XRpc\_ParameteArray:参数列表
* ParameteLen:参数大小,比如4 sizeof(int)
* ParameteType:参数类型,参考:ENUM\_PROTOCOL\_XRPC\_PARAMETE\_TYPE
* ParameteValue:参数值
* XRpc\_ReturnType:返回值类型,参考:ENUM\_PROTOCOL\_XRPC\_PARAMETE\_TYPE

### 3.2.2 回复

HTTPCode 代表处理结果.负载内容如下:

{  
    **"XRpc\_FunctionName"**:**"XEngine\_AddMethod"**,  
    **"XRpc\_ReturnLen"**:**4**,  
    **"XRpc\_ReturnType"**:**3**,  
    **"XRpc\_ReturnValue"**:**3**  
}

# 四 配置说明

## 4.1 服务器配置

配置文件:XEngine\_Config.json

### 4.1.1 基本配置

* tszTaskUrl:HTTP任务地址
* tszTmpFile:临时文件保存地址
* tszAPPDeamon:服务进程守护的本地程序
* bAutoStart:是否设置自动启动
* bHideWnd:启动自动隐藏窗口

### 4.1.2 时间配置

ClientTime配置

* nCheckTime:进程检查等待时间,单位秒
* nErrorTime:最大允许进程启动错误次数

### 4.1.3 客户端配置

ClientConfig配置

* bEnable:是否启用
* nIPType:网络类型,TCP还是UDP
* nPort:端口
* tszIPAddr:服务器地址

### 4.1.4 日志配置

LogConfig配置

* nMaxSize:日志大小
* nMaxCount:日志备份个数
* nLogLeave:日志打印级别
* tszLogFile:日志输出路径

### 4.1.5 版本配置

VersionList 数组

### 4.1.6 RPC配置

RPCConfig配置

* nPort:RPC服务端口
* nThread:RCP服务线程
* nClient:最大运行客户端个数
* nTimeCheck:超时检查次数
* nTimeOut:超时时间

## 4.2 进程守护配置

配置文件:XEngine\_ConfigList.json

ListCount:守护进程个数

ListArray:进程列表

* APPName:进程名称
* APPPath:进程路径
* APPAutoStart:是否自动启动
* APPService:是否是服务
* APPReTime:设置自动重启,时间,秒
* APPEnable:是否启用此配置

# 五 高级配置

## 5.1 邮件报告

邮件报告是在程序第一次启动的时候给指定邮件地址发送一封报告邮件.

你需要在发布软件前就配置好.

这个配置会被程序加密,因为设计到一些敏感信息.

### 5.1.1 配置文件

配置文件:XControl\_EMail.ini

[Email]

* SmtpAddr=邮箱服务器
* SmtpUser=用户名
* SmtpPass=密码
* SmtpFrom=发送者邮件地址

[SendTo]

* AddrList=要报告的地址,可以多个使用;分割

### 5.1.2 生成配置

你可以通过程序来生成配置文件.执行程序加参数XEngine\_XControl.exe -c 加密邮件配置,无参数.待加密文件:./XControl\_Config/XControl\_EMail.ini 加密后的文件:./XControl\_Config/XControl\_EMail.ini.dat

注意:你必须按照格式放置文件.加密完成后可以删除XControl\_EMail.ini而保留XControl\_EMail.dat

## 5.2 服务配置

### 5.2.1 Windows

Windows服务配置需要用到XEngine\_Deamon程序.这个程序只在WINDOWS下有用.此程序通过服务运行

守护进程在配置文件XEngine\_Config.json的tszAPPDeamon

配置使用管理员运行控制台

安装服务执行XEngine\_Deamon.exe /install

删除服务执行XEngine\_Deamon.exe /uninstall

### 5.2.2 Linux

Linux使用xControl.service脚本来设置服务

首先拷贝程序所有文件到/usr/sbin/XEngine\_XControl 目录下.

执行命令:

启动服务:sudo systemctl start xControl

自动启动:sudo systemctl enable xControl

停止服务:sudo systemctl stop xControl

# 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 |