XEngine开发框架说明文档

SDK开发文档

目录

[一 概述 6](#_Toc204868466)

[1.1 开发目的 6](#_Toc204868467)

[1.2 联系方式 6](#_Toc204868468)

[1.2.1 BUG提交 6](#_Toc204868469)

[1.2.2 意见建议 6](#_Toc204868470)

[1.2.3 联系我们 6](#_Toc204868471)

[1.2.4 商业合作 6](#_Toc204868472)

[1.3 系统环境 6](#_Toc204868473)

[1.3.1 Windows环境 6](#_Toc204868474)

[1.3.2 Linux环境 7](#_Toc204868475)

[1.3.3 MacOS环境 7](#_Toc204868476)

[1.3.4 Android环境 7](#_Toc204868477)

[1.3.5 IOS环境 7](#_Toc204868478)

[1.4 支持的开发工具 7](#_Toc204868479)

[1.5 版本说明 7](#_Toc204868480)

[1.5.1 发布版本说明 7](#_Toc204868481)

[1.5.2 版本号说明 7](#_Toc204868482)

[1.6 名词定义 8](#_Toc204868483)

[二 发展历程 8](#_Toc204868484)

[2.1 更新历史 8](#_Toc204868485)

[2.2 版本发展 8](#_Toc204868486)

[2.2.1 V1:堆积代码(2010.02 - 2011.08) 8](#_Toc204868487)

[2.2.2 V2:目标明确(2012.05 - 2016.05) 8](#_Toc204868488)

[2.2.3 V3:稳定架构(2017.03 - 2017.12) 9](#_Toc204868489)

[2.2.4 V4:迈向成熟(2018.01 - 2018.12) 9](#_Toc204868490)

[2.2.5 V5:稳定兼容(2019.01 - 2019.12) 9](#_Toc204868491)

[2.2.6 V6:商业应用(2020.01 - 2021.04) 9](#_Toc204868492)

[2.2.7 V7:统一合并(2021.04 - 2023.02) 10](#_Toc204868493)

[2.2.8 V8:国际标准(2023.03 - 2024.12) 10](#_Toc204868494)

[2.2.9 V9:安全可控(2024.12-) 10](#_Toc204868495)

[2.2.10 V10:暂无规划 10](#_Toc204868496)

[三 跨平台性 10](#_Toc204868497)

[3.1 跨平台说明 10](#_Toc204868498)

[四 特殊说明 11](#_Toc204868499)

[4.1 错误处理 11](#_Toc204868500)

[4.2 网络开发 11](#_Toc204868501)

[4.3 报毒说明 11](#_Toc204868502)

[4.4 崩溃处理 11](#_Toc204868503)

[五 开始开发 12](#_Toc204868504)

[5.1 部署环境 12](#_Toc204868505)

[5.2 加载头文件 12](#_Toc204868506)

[5.3 调用API 13](#_Toc204868507)

[5.5 连接到库 14](#_Toc204868508)

[5.6 使用例子 14](#_Toc204868509)

[六：网络引擎图 15](#_Toc204868510)

[6.1 网络引擎结构图 15](#_Toc204868511)

[6.1.1 音视频编解码组件结构图 15](#_Toc204868512)

[6.1.2 基础组件结构图 15](#_Toc204868513)

[6.1.3 客户端组件结构图 16](#_Toc204868514)

[6.1.4 核心组件结构图 16](#_Toc204868515)

[6.1.5 帮助组件结构图 16](#_Toc204868516)

[6.1.6 网络组件结构图 17](#_Toc204868517)

[6.1.7 标准组件结构图 17](#_Toc204868518)

[6.1.8 流媒体组件结构图 17](#_Toc204868519)

[6.1.9 系统组件结构图 18](#_Toc204868520)

[6.2 网络引擎模块图 19](#_Toc204868521)

[七：目录结构 19](#_Toc204868522)

[7.1 主目录结构 19](#_Toc204868523)

[7.2 引擎目录 19](#_Toc204868524)

[7.3 组件目录 20](#_Toc204868525)

[7.4 模块结构 20](#_Toc204868526)

[7.4.1 XEngine\_AvCodec 20](#_Toc204868527)

[7.4.2 XEngine\_BaseLib 20](#_Toc204868528)

[7.4.3 XEngine\_Client 21](#_Toc204868529)

[7.4.4 XEngine\_Core 21](#_Toc204868530)

[7.4.5 XEngine\_HelpComponents 21](#_Toc204868531)

[7.4.6 XEngine\_NetHelp 22](#_Toc204868532)

[7.4.7 XEngine\_Rfccomponets 22](#_Toc204868533)

[7.4.8 XEngine\_StreamMedia 23](#_Toc204868534)

[7.4.9 XEngine\_SystemSdk 24](#_Toc204868535)

[八 授权信息 24](#_Toc204868536)

[8.1 授权支持 24](#_Toc204868537)

[8.2 代码仓库 24](#_Toc204868538)

[8.3 开源项目 25](#_Toc204868539)

[九 参与贡献 25](#_Toc204868540)

[十 FAQ 25](#_Toc204868541)

[十一 版权信息 25](#_Toc204868542)

[11.1 版权说明 25](#_Toc204868543)

[11.2 开源库 26](#_Toc204868544)

[附录 26](#_Toc204868545)

[附录1 公用头文件 26](#_Toc204868546)

[附录2 协议头定义 26](#_Toc204868547)

[附录3 特别鸣谢 26](#_Toc204868548)

|  |  |  |  |
| --- | --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布 | 文件名称： | XEngine开发文档 | |
| 当前版本： | V9.25.0.1001 | |
| 完成日期： | 2025-08-01 | |
| 拟 制： qyt | | |

# 一 概述

在阅读此文档之前,你需要先阅读Readme文件,重复内容不会在这个文档显示.

## 开发目的

参考readme的项目介绍

## 1.2 联系方式

### 1.2.1 BUG提交

你可以通过发送电子邮件的方式或者论坛提交给我们反驳BUG。Email：486179@qq.com 或者论坛 bbs.xyry.org,感谢大家的支持。

### 1.2.2 意见建议

如果有任何意见和建议都可以通过EMAIL方式发送给我们，电子邮件地址可以使用上面的地址提交即可。

当然你也可以直接与我们交流引擎方面的意见和好的想法。QQ:486179

### 1.2.3 联系我们

1. 网站:https://www.xyry.org http://www.libxengine.com
2. 论坛:https://bbs.xyry.org
3. QQ:486179
4. 邮件:486179@qq.com

### 1.2.4 商业合作

如果有合作请求,也可以与我们联系

## 系统环境

我们支持大多数主流系统发布:Windows 10以及以上,Ubuntu22.04 以及以上,Fedora 41以及以上,Debian 12以及以上,Rockylinux 9(Centos 9)以及以上,Macos13以及以上,这些系统都支持Arm64和x86\_64.Windows包括32位支持.还支持移动平台的Android和IOS的ARM64版本

请注意:GIT仓库只包含Windows的x86-64平台,Linux的Ubuntu24.04 x86-64平台,MacOS的x86-64平台.如果你使用的是其他系统平台,那么需要再GITHUB的发布页面单独下载后解压运行安装脚本才可以使用

### 1.3.1 Windows环境

Windows环境可以支持x86和x64以及ARM64模块.默认是x64的,x86和ARM64需要单独下载和配置

Windows环境目前使用的是Unicode环境的UTF8字符集代码格式

### 1.3.2 Linux环境

我们推荐使用最新版本运行,并且使用x86-64的系统！

注意:由于linux系统的库环境版本不如windows齐全,其他linux发行版本都需要单独处理.如果你有其他系统版本需求,请与我们联系.我们将单独定做!

### 1.3.3 MacOS环境

MacOS一般来说,支持13以及以上版本.MACOS有x86\_64和arm64版本

想要在MACOS系统下使用XEngine,需要你优先配置xcode,在运行我们的脚本.

我们需要使用到brew安装依赖,你可以优先配置好或者使用我们脚本自动配置

### 1.3.4 Android环境

Android采用NDK编译,使用原生C/C++编写,需要的版本是ANDROID10以及以上系统并且需要arm64-v8a架构编译.

### 1.3.5 IOS环境

IOS环境我们使用的是XCODE SDK编译.原生C/C++代码.需要IOS版本在15或者以上,并且架构仅支持ARM64.

## 1.4 支持的开发工具

* 我们的SDK使用C/C++编写,所以对于支持C/C++系统的环境都支持
* 除了标准C/C++以外,也支持E,VB,JAVA,GO等语言调用

## 1.5 版本说明

### 1.5.1 发布版本说明

* PT：技术预览版本，拥有一些新技术新功能的加入，极其不稳定的版本。
* BT：测试版本，通过单元测试或者内部测试后的版本。
* RC：候选发布版本，已经可以正式使用的版本，其内部只有极少数的隐含错误。
* LTS：长期支持正式版本，可以用于正式商业项目(这个标识在发布版本的时候一般会被省略)。

### 1.5.2 版本号说明

我们的版本号有4个版本，其中依次从左往右，比如：k.mm.sss.ffff的意思为

* K =（KernelVersion）：内核版本号，表示框架最重要的版本号。除非重大技术架构核心变更，不然一般都不会更改此版本号。
* Mm = （MainVersion）：主要版本号，一般加核心模块更改或者技术架构更新的时候才会更改此版本号。
* Sss = （SubVersion）：次要版本号，在功能上做了变更或者增加删除功能模块的时候会更改此版本号。
* Ffff = （FixVersion）：修复更新版本号，只是在引擎更新优化和修复一些错误或者代码的时候才会更新此版本号。

## 1.6 名词定义

在核心服务引擎和网络客户端中.有下面的几个服务函数需要注意

* TCPXCore: Windows表示IOCP模型,Linux(Android)表示EPOLL模型,MacOS(IOS)代表Kqueue.
* TCPXPoll: Windows表示OVERLAPPED模型,Linux和Mac表示POLL模型

# 二 发展历程

## 2.1 更新历史

完整更新历史请参考官方网站的更新历史

## 2.2 版本发展

### 2.2.1 V1:堆积代码(2010.02 - 2011.08)

XEngine网络引擎成立于2010年初,编写这套引擎的目的是为了简化工作上面的一些重复的编码.

V1版本的编写,就是一直在往上面堆代码,然后实现各种各样的功能.是个极其不成熟的引擎,并且架构过于臃肿,毫无规划可言,无法满足日常需要,由于一直在新增加功能,也一直无法发布稳定版本.所以,导致更新两年后,V1版本的历史就结束了.

### 2.2.2 V2:目标明确(2012.05 - 2016.05)

V2版本的发布,在2012年年中的时候,由于V1版本过难于维护,而且功能杂乱,完全不是一个正常的网络通信引擎了,而是一个大杂烩一样的套件.所以,这个版本删除和重构了很多代码和功能模块.

V2版本虽然删除了很多与网络不相关的代码.但是架构上面还是有很多问题,这次V2版本更新,还是没有重构架构,只是对于模块代码上面进行了整理,但是,这是治标不治本的方式.所以,就算一直在维护,也是写写停停.导致在经过4年后的2016年初,彻底断更.

### 2.2.3 V3:稳定架构(2017.03 - 2017.12)

中间断更一年时间,这一年由于时间不合适,版本过于杂乱,感觉自己无心维护这版本,导致了这个引擎一直没有动过,后来有点多余的时间.而且,也不想放弃维护了这么久的引擎,终于,这套引擎重新架构了.

V3版本,引擎架构几乎全变,采用了最新的编译环境和编写方式.对于一些本身的网络功能,全部采用自写实现,统一了WINDOWS和LINUX的API等.删除了无用的代码和与网络不相关的功能,彻底重构了核心网络代码,使之引擎的架构目的明确.

通过这一版本,引擎架构基本不会在有什么变化,功能模块也逐渐稳定.终于可以适用于正式项目并且通过大多数功能模块都通过了验证.

### 2.2.4 V4:迈向成熟(2018.01 - 2018.12)

如果说V3版本是架构上面的稳定,那么V4版本就是功能模块上面的成熟.V4版本对大多数功能模块进行了优化和更新,这是为了让引擎更加适合商业和正式版本上线而做的改动.

目前的V4版本,对于以前的一些不合理或者还有缺陷的地方都进行了修正和优化,V4版本将是一个稳定的周期迭代版本.所以,在这个版本中,有很多功能是目前市面上很多商业软件没有的功能.并且,我们对于提供的大多数服务和APP例子代码都对这些功能进行了验证和使用.保证了引擎的稳定和成熟!

### 2.2.5 V5:稳定兼容(2019.01 - 2019.12)

经过V4版本的更新,已经对架构和设计有了比较成熟的模式,V4版本对于某些功能还是不能做到兼容和稳定.

V5版本,将在V4版本的基础上,对现有模块和跨平台函数进行修改,对老旧代码进行更新,对不支持跨平台的函数进行整改.对于没有作用的模块进行删除,对于需求大的模块进行添加.对版本发布进行管理.继续精简优化代码和模块与架构.使之更加强大稳定.

### 2.2.6 V6:商业应用(2020.01 - 2021.04)

V6版本将在V5的基础上面,不在进行大规模模块与组件的裁剪.除了开发计划里面的以外.

V6将更加注重商业应用上面的需求进行更新,并且对稳定性和性能进行进一步优化和修改.V6引擎将会是一个长期支持版本(LTS).可能会长达1-2年时间.我们将重点关注流媒体,编解码以及存储服务方面的内容.

从V6版本过后,网络通信引擎名称正式命名为XEngine.

### 2.2.7 V7:统一合并(2021.04 - 2023.02)

V7版本的主要目的是为了合并WINDOWS,LINUX系统代码,不需要两套代码维护,维护一套代码可以面向x86 x64,arm等系统进行适配.可以让我们更加注重跨平台本身,而不是跨平台后的代码适配.

V7版本是一个重大的架构重构版本,这个版本的架构进行了完全的重构.经过了10年的发展,以往的架构已经不适合目前的需求,所以,这个版本的架构进行了重新设计.

### 2.2.8 V8:国际标准(2023.03 - 2024.12)

V8版本将以国际化和标准化为主.包括兼容平台和IDE环境,字符集编码环境等.

V8的版本还会对现在依赖库和技术进行升级,将匹配最新开源版本技术和标准.

### 2.2.9 V9:安全可控(2024.12-)

V9版本将以更加标准安全的方法开发框架,将采用最新的C++特性和安全函数来作为基本操作函数使用.让内存泄漏,安全问题得到最大程度的解决.

V9还将提供全平台支持,对于UBUNTU,ROCKYLINUX,WINDOWS,MACOS等系统的x86,x64,arm64等环境将提供全面支持

V9将是一个长期支持版本,其时间可能会在2-3年

### 2.2.10 V10:暂无规划

# 三 跨平台性

## 3.1 跨平台说明

我们的引擎不是所有API都支持跨平台，某些API可能会不支持某些平台,具体请参考代码的宏定义,如果不支持,你将无法调用此函数.

请注意,我们的平台宏定义使用了下面的类型定义,你可以根据需要使用

WINDOWS:\_MSC\_BUILD

MACOS:\_\_MACOS\_\_

Linux:UBUNTU:\_\_UBUNTU\_\_

RockyLinux(Centos家族群):\_\_CENTOS\_\_

Debian:\_\_DEBIAN\_\_

Fedora:\_\_FEDORA\_\_

ANDROID:\_\_ANDROID\_\_

IOS:\_\_IOS\_\_

# 四 特殊说明

## 4.1 错误处理

每一个模块，我们都有单独的错误表，通过指定的函数可以获取错误，一般的，除特殊指定，逻辑型返回值都有错误码，你可以通过查看头文件定义的获取最后一次发生的错误来获取错误码。如果你不想获取错误，可以不用关心、返回假才有错误码，真表示成功

每个模块获取错误码的方式都在 \*\_Define.h 的第一个函数 \*\_GetLastError() 函数,这个函数的参数是系统错误,可以为NULL,一般的这个参数导出的错误是linux errno的错误或者windows getlasterror()函数返回的错误,只有这个函数的返回值,才是引擎返回的错误.

返回的错误是16进制的,你需要通过%lX 来输出,或者自己转换为16进制.

## 4.2 网络开发

在你开发客户端程序的时候，你可以自己开发客户端,而不需要我们提供的客户端,但是某些模块需要专用的通信协议才能通信,这个你可以通过协议文档了解更多。

## 4.3 报毒说明

如果你发现我们的引擎被杀毒软件误报,你可以与我们联系,并且提供给我们是哪个杀毒软件在报毒,我们会及时与杀毒厂商联系.

## 4.4 崩溃处理

如果你发现调用我们的函数与预期不符,比如结果错误,或者崩溃,你们可以与我们联系,在此之前,我们建议你们仔细检查输入是否正确.

很多时候,我们的API函数内部会有一些异常错误处理判断.包括线程安全等,但并不是所有检查都能够覆盖全或者一定没有问题.如果你输入的数据错误,比如内存指针被破坏.或者在没有线程安全的函数中使用多线程,都可能造成结果不正确或者崩溃.

如果你们在使用过程中遇到问题,可以增加COREDUMP等方式调试,也可以联系我们报告问题.

# 五 开始开发

请先参考README文档环境前置部署

## 5.1 部署环境

Windows：如果你想应用我们的SDK，你可以拷贝相对应的依赖DLL到你程序的运行目录下即可。你编译出来的程序，运行的时候，系统会提示你缺少的DLL是什么，你可以在我们的发布目录下找到这些缺失的DLL。你也可以使用安装脚本选项1把模块安装到系统中,我们推荐你这样做!

Linux和MacOS：LINUX和MacOS版本现在拥有安装脚本，你可以自己运行此安装脚本来进行安装我们的开发环境，安装方式你可以通过-h 参数来获得帮助，此脚本安装会安装头文件和库.头文件和库会被安装到系统目录下,不需要特殊目录指定即可连接。

MacOS:MAC使用的同样是linux的安装脚本,MAC需要brew命令的支持,如果没有安装,我们会帮你安装,但是安装的Brew脚本服务器没有设置国内镜像,会很慢,所以我们推荐在运行脚本前先自己安装brew.此脚本安装会安装头文件和库.头文件和库会被安装到系统目录下,不需要特殊目录指定即可连接.

Android:需要拷贝依赖的SO到你的应用目录放到一起,安卓环境需要用户自己拷贝依赖项目

IOS:由于IOS的特殊性,你需要继续把dylib签名打包成xcframework后使用!

## 5.2 加载头文件

在安装配置好环境后,你可以进行代码编写了.使用我们的库的时候,需要加载头文件,每个模块都有单独的头文件,在加载我们的头文件的时候,你需要优先加载系统头文件然后加载我们提供的公用头文件,最后加载SDK的头文件,我们建议你最好使用单独新建立一个.h文件来加载和管理这些头文件.

比如你要使用我们的核心服务头文件,你使用我们的头文件就是以下方式.

#include <pch.h> //预编译头文件,Windows可能需要

#include <系统头文件.h>

Using namespace 名词空间;

#include <XEngine\_Include/XEngine\_CommHdr.h>

#include <XEngine\_Include/XEngine\_ProtocolHdr.h> //可能需要

#include <XEngine\_Include/XEngine\_Core/NetCore\_Define.h>

#include <XEngine\_Include/XEngine\_Core/NetCore\_Error.h>

#include “你的头文件”

## 5.3 调用API

我们的API都被定义在 \*\*\_Define.h头文件中，其中你打开此头文件可以看到很多定义。其中一个函数的格式如下。

上面是解释 下面是函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：HttpProtocol\_Server\_Init

函数功能：初始化HTTP服务器

 参数.一：lpszCodeFile

  In/Out：In

  类型：常量字符指针

  可空：N

  意思：HTTP状态码配置文件位置

 参数.二：lpszMimeFile

  In/Out：In

  类型：常量字符指针

  可空：N

  意思：HTTPMIME配置文件位置

 参数.三：nPoolCount

  In/Out：In

  类型：整数型

  可空：Y

  意思：任务池个数,应该配合线程池个数使用

 参数.四：nLimitCount

  In/Out：In

  类型：整数型

  可空：Y

  意思：每个包限制大小.0为不限制,单位XBYTE

 参数.五：bAllowOrigin

  In/Out：In

  类型：逻辑型

  可空：Y

  意思：跨域访问选项,默认为允许

返回值

  类型：逻辑型

  意思：是否成功

备注：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" XHANDLE HttpProtocol\_Server\_InitEx(LPCXSTR lpszCodeFile, LPCXSTR lpszMimeFile, int nPoolCount = 0, int nLimitCount = 0, bool bAllowOrigin = true);

我们的注释都写的很详细，包括函数的意思和参数的意思.你只需要调用我们的头文件即可。

另外，你们可以在我们发布的例子目录下找到一些API对应的使用方式的例子。

## 5.5 连接到库

Windows的连接库方式,如果你是VS,并且在项目属性库目录添加了我们的环境变量,例如:x86是$(XEngine\_Lib32)或者x64是$(XEngine\_Lib64),那么可以直接连接.比如:

#pragma comment(lib,"XEngine\_BaseLib/XEngine\_BaseLib")

MacOS和Linux连接到库的方式是相同的,你如果调用了baselib的库函数,直接在编译的脚本加上库连接即可,比如:-lXEngine\_BaseLib

## 5.6 使用例子

我们提供了很多使用例子代码,你可以参考我们的例子来编写你需要的服务,.里面的例子包含的头文件路径不正确,可能需要你自己改动加载的头文件路径,并且你需要把依赖的DLL拷贝到例子编译出来的目录下能运行!

Windows可以直接用VS打开编译，LINUX和MAC需要使用命令编译，编译命令在代码中被注释了，你可以直接拷贝到终端编译。

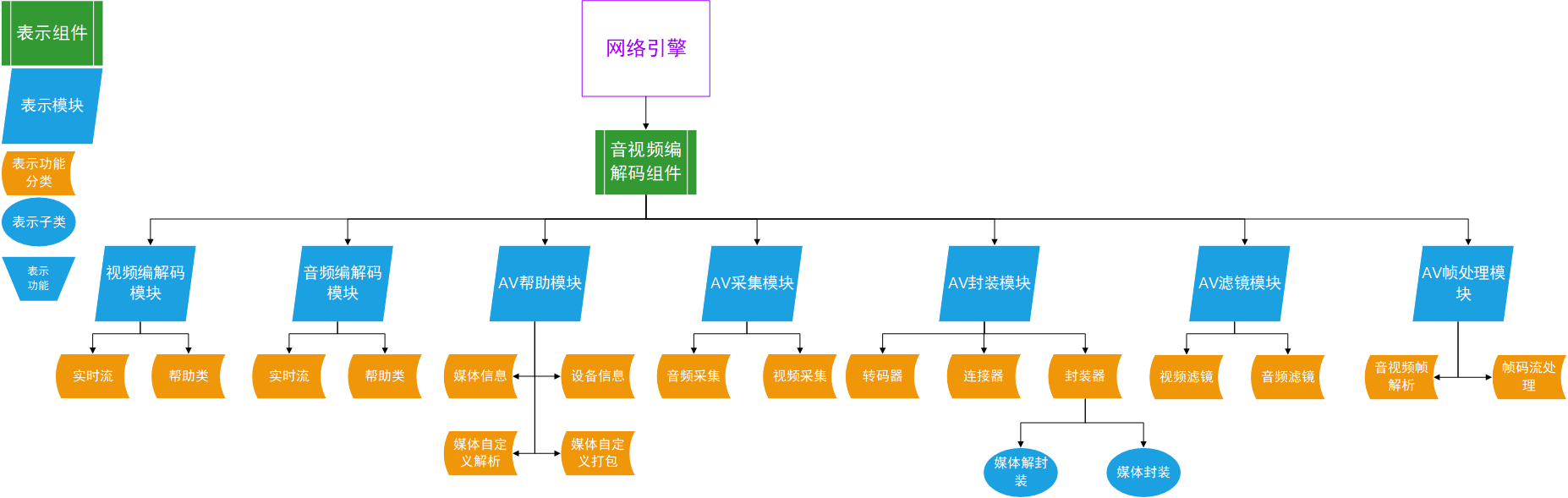
例子下载地址在我们的官方网站上面可以找到.

# 六：网络引擎图

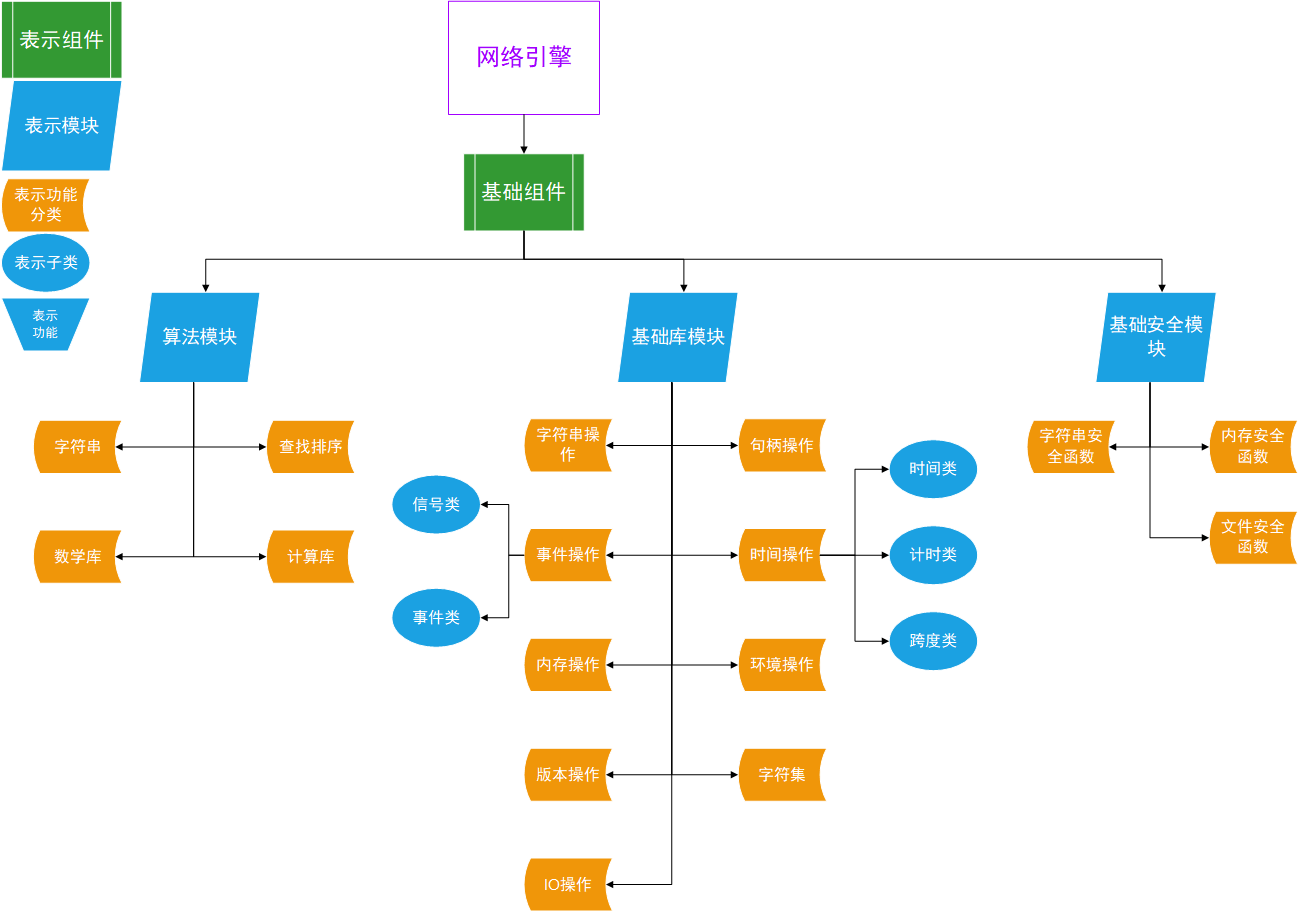
## 6.1 网络引擎结构图



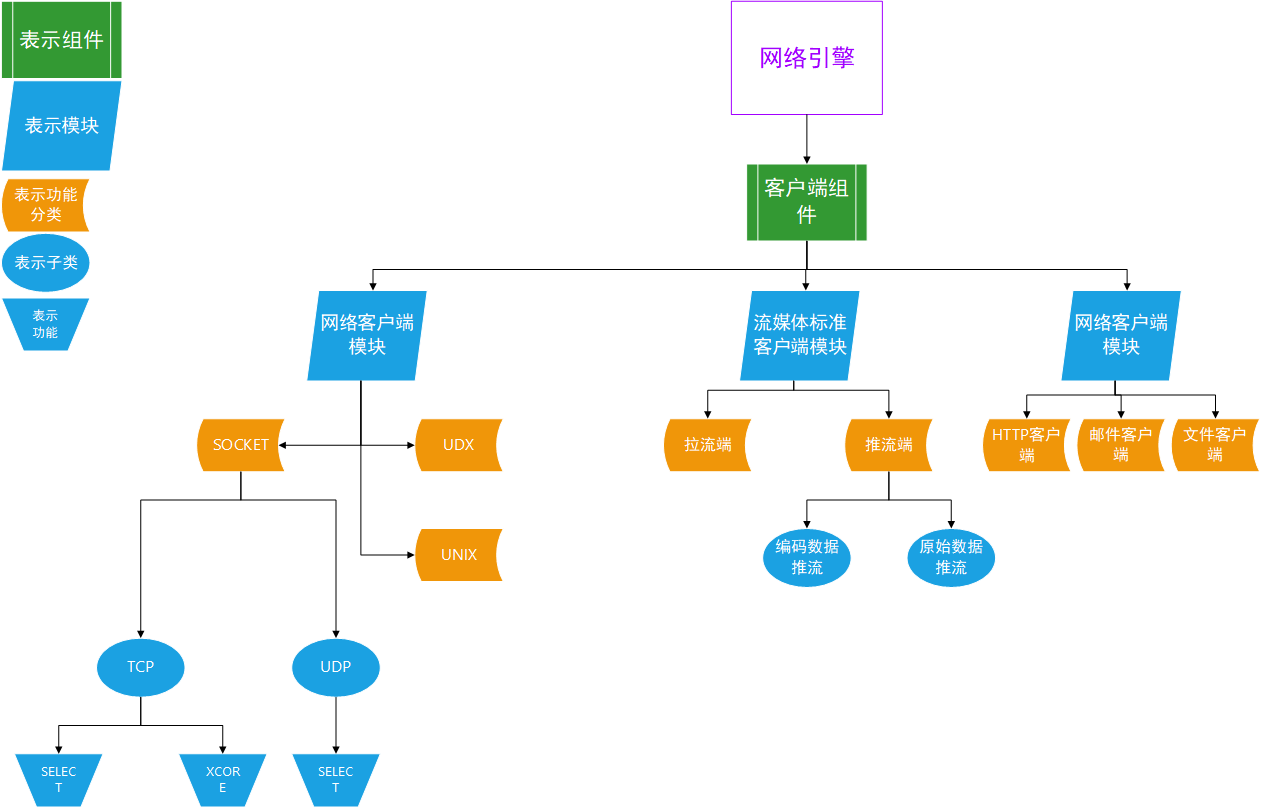
### 6.1.1 音视频编解码组件结构图



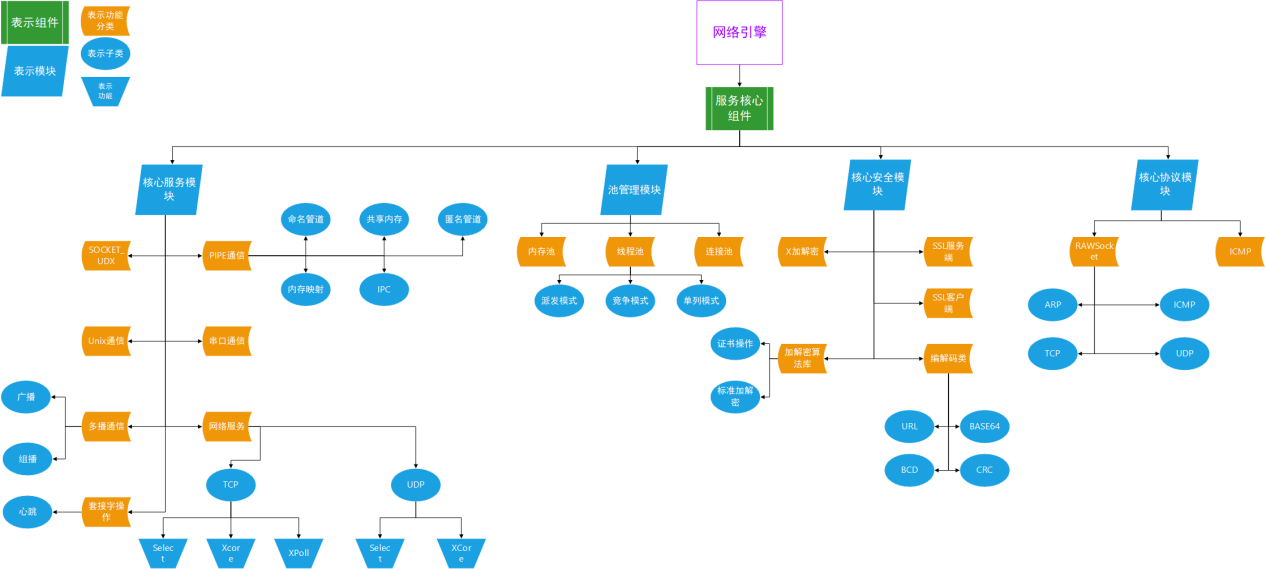
### 6.1.2 基础组件结构图



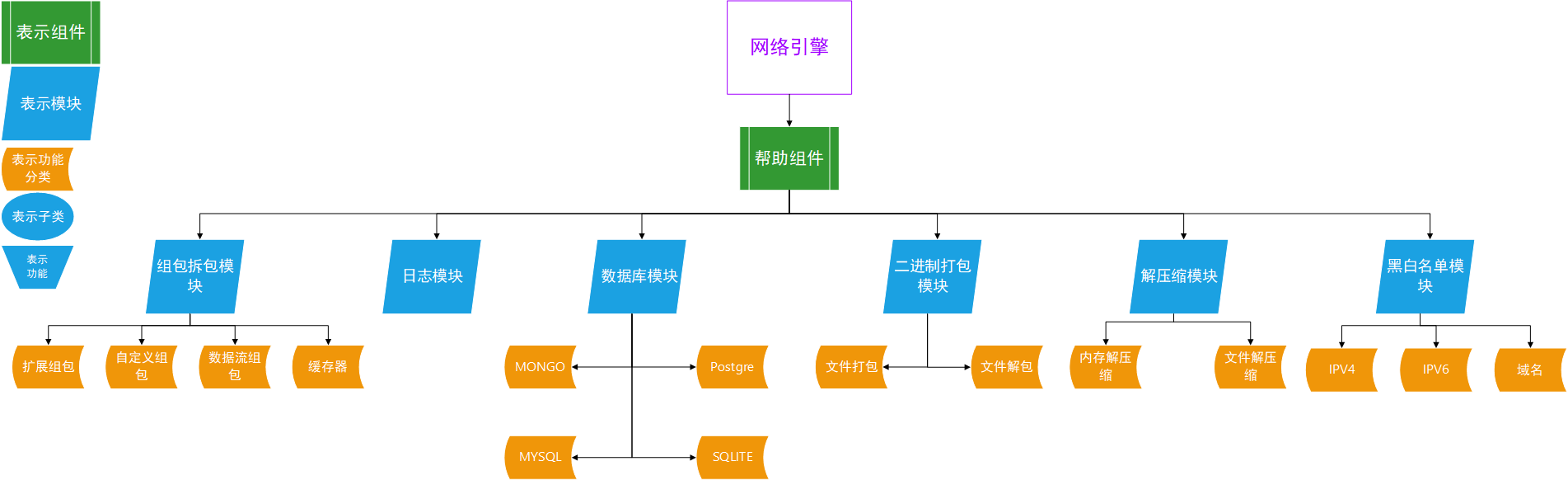
### 6.1.3 客户端组件结构图



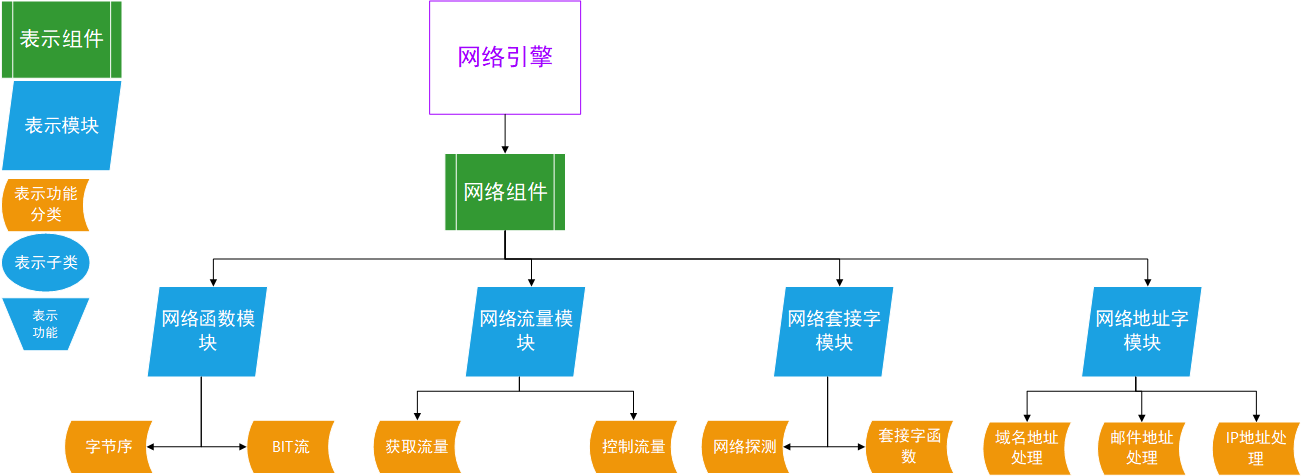
### 6.1.4 核心组件结构图



### 6.1.5 帮助组件结构图



### 6.1.6 网络组件结构图

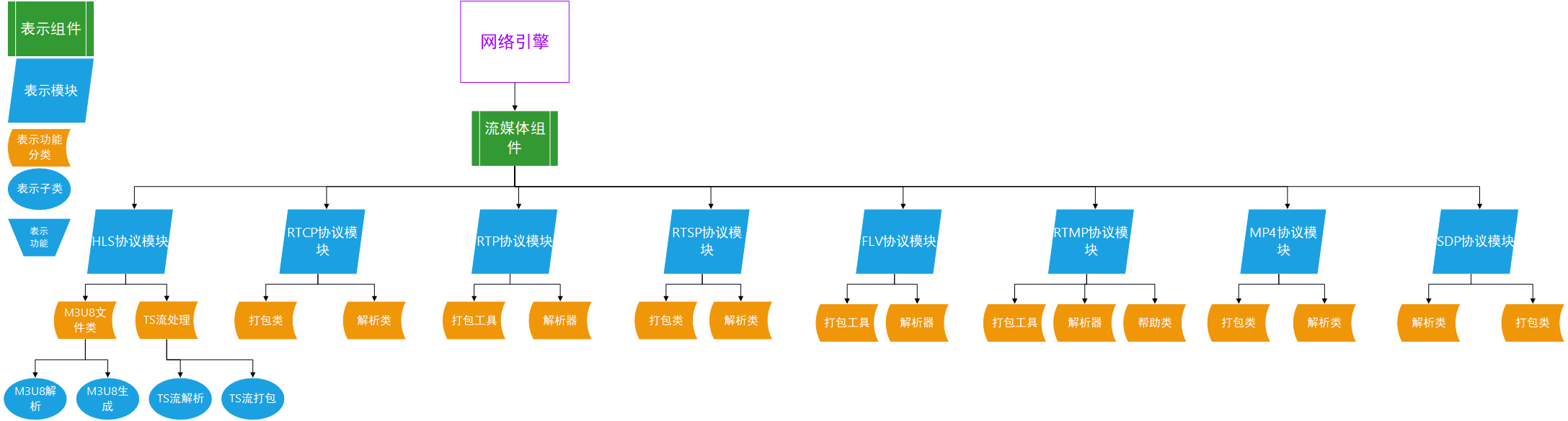


### 6.1.7 标准组件结构图

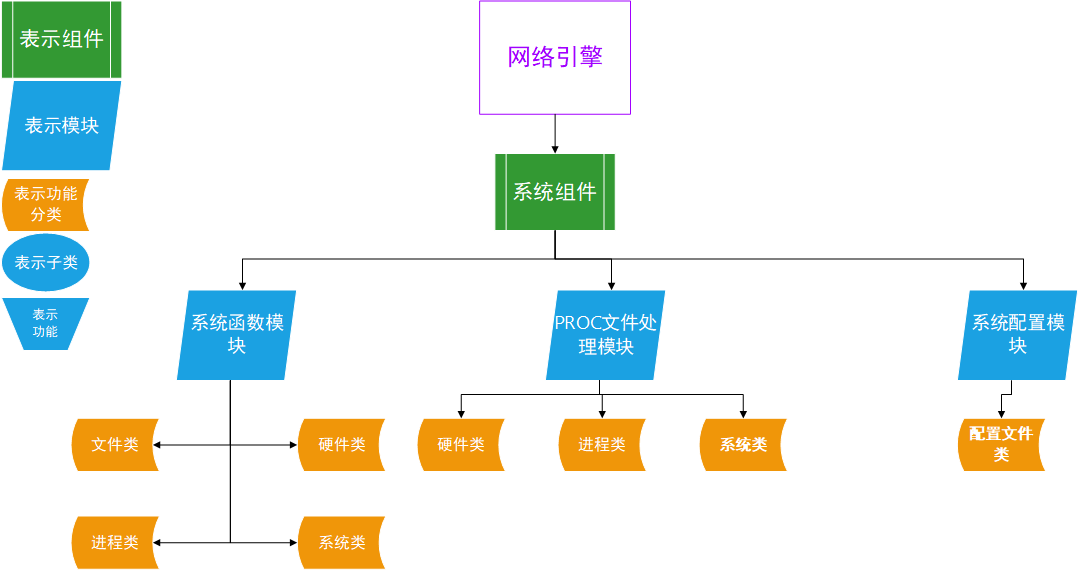
图示

AI 生成的内容可能不正确。

### 6.1.8 流媒体组件结构图



### 6.1.9 系统组件结构图



## 6.2 网络引擎模块图

图形用户界面

AI 生成的内容可能不正确。

# 七：目录结构

## 7.1 主目录结构

参考Read me

## 7.2 引擎目录

* XEngine\_AVCodec 音视频解码器
* XEngine\_Client 客户端组件
* XEngine\_Core 核心引擎服务组件
* XEngine\_HelpComponents 帮助引擎组件
* XEngine\_NetHelp 网络开发组件
* XEngine\_BaseLib 公有库基础组件
* XEngine\_RfcComponents RFC协议定义的组件开发目录
* XEngine\_StreamMedia 流媒体组件开发目录
* XEngine\_SystemSdk 系统特有组件

## 7.3 组件目录

组件目录下面只包含二进制模块文件,比如包含dll,lib,so等

## 7.4 模块结构

### 7.4.1 XEngine\_AvCodec

编解码组件:用于音频和视频编解码开发和过滤器,转码等操作

#### 7.4.1.1 XEngine\_AudioCodec

音频编解码开发模块

#### 7.4.1.2 XEngine\_AVCollect

音视频采集模块,支持音频,桌面,摄像头采集

#### 7.4.1.3 XEngine\_AVHelp

音视频开发帮助模块,枚举视频,音频设备和获取支持的编解码器操作等

#### 7.4.1.4 XEngine\_VideoCodec

视频编解码开发模块

#### 7.4.1.5 XEngine\_AVFormat

音视频文件格式封装和格式转换模块

#### 7.4.1.6 XEngine\_AVFrame

音视频帧处理模块，分拆帧，帧转换

#### 7.4.1.6 XEngine\_AVFilter

音视频滤镜处理模块

### 7.4.2 XEngine\_BaseLib

基础组件:包含基本的时间,字符串,算法,事件,句柄等操作

#### 7.4.2.1 XEngine\_Algorithm

算法库开发模块,包含一些常用算法操作API

#### 7.4.2.2 XEngine\_BaseSafe

基础安全操作模块,提供字符串,内存的安全操作API函数

#### 7.4.2.2 XEngine\_BaseLib

基础模块,包含字符串,事件,句柄,时间,位等操作的API函数集

### 7.4.3 XEngine\_Client

客户端组件:包含常规TCP和UDP客户端开发,以及其他高级客户端开发SDK

#### 7.4.3.1 XClient\_Socket

网络客户端开发模块,这个客户端包含TCP,UDP,UDT,SCTP客户端操作,支持高性能,可靠,低延迟,等客户端网络开发API函数.如果不是UDT,SCTP,用户可以不适用这个模块,而自己适用标准SOCKET开发客户端.

#### 7.4.3.2 XClient\_Stream

用于流媒体协议客户端,可支持推流拉流

#### 7.4.3.3 XClient\_APIHelp

网络客户端操作,HTTP,EMAIL等等

### 7.4.4 XEngine\_Core

核心组件:包括各种高性能网络服务,高性能线程池和网络相关基础协议开发等

#### 7.4.4.1 XEngine\_Core

网络服务开发模块,里面包含了所有可用服务开发的API接口,是整套引擎的核心模块,你可以在这里面找到TCP,UDP各种高性能开发模型API接口,也可以找到无线通信,红外,蓝牙开发接口,还可以找到其他各种通信方式,广播,组播的开发接口,等等,具体可以查看导出的API说明.

#### 7.4.4.2 XEngine\_ManagePool

管理池模块,这里面包含内存池,线程池,连接池的开发API接口

#### 7.4.4.3 XEngine\_Cryption

安全加解密开发模块,包含SSL服务与客户端API接口和加解密等API接口

#### 7.4.4.4 XEngine\_Protocol

RFC协议开发接口,一些标准的,小型RFC接口定义使用这个模块开发.

### 7.4.5 XEngine\_HelpComponents

帮助组件:包括协议组包拆包,二进制组包拆包,解压缩,数据库相关SDK开发

#### 7.4.5.1 HelpComponents\_BINPack

二进制打包解包API工具,可以将多个文件打包成一个文件或者解包成多个文件.

#### 7.4.5.2 HelpComponents\_Compress

解压缩API开发模块

#### 7.4.5.3 HelpComponents\_DataBase

数据库开发模块,支持SQLITE,MYSQL,MSSQL,MARAIDB,POSTPRESQL,MONGODB数据库

#### 7.4.5.4 HelpComponents\_Packets

数据流,组包,解包模块,可以把TCP留组包,或者解包,TCP是流式套接字,必须通过这个模块操作你才能处理一个完成的包,不然你接受数据的时候可能会造成沾包的情况发生.

#### 7.4.5.5 HelpComponents\_XLog

日志模块,可以打印日志,并且保存日志,备份日志.服务器最好使用这个模块来记录日志信息.

#### 7.4.5.6 HelpComponents\_WBlackList

黑白名单策略模块,可以对IPV4和域名进程过滤和配置策略

### 7.4.6 XEngine\_NetHelp

网络组件:与网络相关的协议与工具开发SDK

#### 7.4.6.1 NetHelp\_APIHelp

网络帮助函数,字节序转换,网络流操作等

#### 7.4.6.2 NetHelp\_APIFlow

网络流量处理模块,支持获取流量信息和流量控制

#### 7.4.6.3 NetHelp\_XSocket

网络API封装接口,IP库,抓包.流量获取等API接口

#### 7.4.6.4 NetHelp\_APIAddr

网络地址相关操作解析函数库,比如IP地址,邮件,域名地址操作解析

### 7.4.7 XEngine\_Rfccomponets

标准组件:用于RFC标准组织推出的协议所开发的模块.

#### 7.4.7.1 RfcComponents\_ProxyProtocol

代理协议开发接口模块,可以开发代理服务器或者客户端,包括SOCKS5和HTTP隧道道理

#### 7.4.7.2 RfcComponents\_WSProtocol

HTTP WEBSOCKET开发接口,支持发送和接受WEBSOCKET协议.

#### 7.4.7.3 RfcComponents\_SIPProtocol

SIP客户端服务器模块,支持SIP协议操作

#### 7.4.7.4 RfcComponents\_SnmpProtocol

SNMP协议客户端封装模块,使用此模块快速进行SNMP协议开发

#### 7.4.7.5 RfcComponents\_NatProtocol

NAT客户端模块,用于内网进行穿墙打洞服务

#### 7.4.7.6 RfcComponents\_HttpProtocol

HTTP服务器开发模块

#### 7.4.7.7 RfcComponents\_SSDPProtocol

SSDP协议操作接口,通过这个模块的API,你可以对SSDP协议和UPNP相关操作

#### 7.4.7.8 RfcComponents\_MQTTProtocol

MQTT协议操作模块,支持协议的解析和封包

#### 7.4.7.9 RfcComponents\_NTPProtocol

NTP时间同步协议模块,支持解析和封包

#### 7.4.7.10 RfcComponents\_DNSProtocol

DNS域名协议解析封包模块

### 7.4.8 XEngine\_StreamMedia

流媒体组件:用于流媒体协议客户端服务器协议相关开发的接口模块

#### 7.4.8.1 StreamMedia\_RtspProtocol

RTSP协议处理模块,支持RTSP协议解析和组包

#### 7.4.8.2 StreamMedia\_RtpProtocol

RTP协议处理模块,支持RTP协议解析和打包

#### 7.4.8.3 StreamMedia\_RtcpProtocol

RTCP协议处理模块,支持RTCP协议解析和打包

#### 7.4.8.4 StreamMedia\_HLSProtocol

HLS直播,录播协议处理与TS文件协议处理模块

#### 7.4.8.5 StreamMedia\_FLVProtocol

FLV协议解析打包模块,支持解析FLV流为单帧或者打包FLV流

#### 7.4.8.6 StreamMedia\_RTMPProtocol

RTMP协议解析打包模块,支持解析RTMP流为单帧或者打包RTMP流

#### 7.4.8.7 StreamMedia\_SDPProtocol

SDP封包解包开发模块

#### 7.4.8.8 StreamMedia\_MP4Protocol

MP4媒体文件解封装模块

### 7.4.9 XEngine\_SystemSdk

系统组件:用于系统平台相关的SDK开发组件

#### 7.4.9.1 XEngine\_SystemApi

系统模块:封装了大部分操作,进程,线程,网络,界面的API接口操作,还有CPU,内存等信息获取的API接口,具体查看导出的接口定义.

#### 7.4.9.2 XEngine\_ProcSdk

ProcFile文件信息读取模块,用于获取linux proc文件系统信息.

#### 7.4.9.3 XEngine\_SystemConfig

系统配置操作模块.包含系统配置读写,配置文件读写等操作

# 八 授权信息

## 8.1 授权支持

如果你对我们的框架敢兴趣,想要更加了解或者需要技术支持,可以访问我们的网站:https://www.xyry.org/auth.html

## 8.2 代码仓库

在V9.x版本我们代码总行数18W+,其中代码10W+,注释8W+.可以让您迅速上手.

代码全是核心关键代码,无业务相关.代码质量非常高.

一键部署：脚本化自动安装环境,完全根据系统的仓库安装依赖库而不需要自己编译.可以更好的兼容系统,目前最优环境解决方案.

Windows使用VCPKG

Ubuntu使用apt,RockyLinux使用dnf

MacOS使用brew

良好的文档支持:我们有良好的文档支持,可以方便你快速了解引擎

最好的代码结构与注释:采用统一代码结构,方便阅读,并且有良好的代码注释,保证函数代码注释覆盖率达到99%.

## 8.3 开源项目

我们目前正在拥抱开源,以往的几个服务器现在已经作为开源发布了.如果你有兴趣,可以查看下面的项目地址

GITEE:https://gitee.com/libxengine 不推荐

GITHUB:https://github.com/libxengine 推荐

# 九 参与贡献

你也可以通过参加开源贡献的方式加入我们,进入我们的框架代码开发.获得框架完整代码资料.

前提通过加入我们的开源项目,提供优质的开源代码,我们将在考察后邀请你加入

你至少参与两个主要开源项目并且每个项目至少有20次有效的代码提交.

# 十 FAQ

有关FAQ常见问题解答和其他更新内容,请关注我们微信公众号



# 十一 版权信息

## 11.1 版权说明

如果你觉得我们侵犯了你的权益,请发邮件到486179@qq.com 给我们,确认后我们将在24小时之内删除!

## 11.2 开源库

我们的引擎依赖了部分开源库,我们使用的开源库列表可以参考下面的列表

1. Curl
2. FFMpeg
3. MongoDB
4. MariaDB
5. Openssl
6. PostgreSQL
7. Zlib
8. Sqlite
9. Nghttp2

# 附录

## 附录1 公用头文件

公用头文件是使用我们引擎需要优先加载的头文件.

加载方式:预编译头文件->系统头文件->公用头文件->网络引擎头文件->项目头文件

公用头文件名为:XEngine\_CommHdr.h

## 附录2 协议头定义

协议头定义文件是我们现在使用的协议头,在某些时候需要用到这个头文件,可以参考:XEngine\_ProtocolHdr.h.如果需要系统头文件,需要放到公用头文件后面加载

## 附录3 特别鸣谢

我们由衷感谢以下人员对于我在开发这个引擎当中所作出的不可磨灭的贡献：

|  |  |  |  |
| --- | --- | --- | --- |
| 排名不分先后 | 姓名（英文名） | 联系方式 | 贡献类型 |
|  |  |  |  |