

Logsrv 用户指南

编写：蒋勇
审核：
版本：ver 2.2.8 build 2026-6-4

更新记录：

2026-6-4 修正 BUG 和增加 TLS1.3 支持
2026-1-11 增加日至文件大小配置
2025-8-14 修正删除空目录问题
2024-8-13 增加每个域名多 ip 支持和 65 查询。
2024-8-3 增加 DNS 服务作为子域名解析用法
2024-7-26 增加 DNS 服务和 level_self 配置
2023-11-30 增加多用户日志访问配置
2023-8-8 修正笔误，调整 WEB 页面字体
2019-2-28 初版

内容目录

1. 概述	1
1.1 系统结构	1
1.2 运行环境	2
2. 安装	3
2.1 基本配置	3
2.2 用户账号配置	5
2.3 Windows 安装	7
2.4 Linux 下安装	8
3 日志读取协议	10
3.1 协议层次	10
3.2 TLS1.2/1.3 规范	10
3.3 websocket 规范	11
3.3.1 websocekt 压缩扩展	11
3.3.2 连接 url 地址	12
3.3.3 兼容性	12
3.4 JSON 规范	12
3.5 命令消息定义	13
3.5.1 请求/应答报文	13
3.5.2 登录	14
3.5.3 读取日志文件列表 ws_loglist	16
3.5.4 读取日志 ws_logget	18
3.5.4 读取日志用于显示 ws_logview	19
4 日志提交协议	22
4.1 日志提交协议定义	22
4.1.1 报头	22
4.1.2 日志内容	22
5 协议实现实例	23
6 DNS 服务	24
6.1 解决什么问题	24
6.2 实现原理	24
6.3 DNS 服务开启	24
6.4 本地 DNS 配置	26
6.4.1 windows 系统	26
6.4.2 Linux 系统	27
6.5 本地 ipv6 自动提交	27
6.6 测试	28
6.7 DNS 记录管理命令	28
6.7.1 提交/更新	28
6.7.2 删除	29
6.7.3 查找	30
6.7.4 读取	31
6.8 子域名解析服务用法	31

1.概述

Logsrv 是一个跨平台可独立部署的集中日志服务、DNS 服务、HTTP 服务软件。为其他软件提供日志写入，查阅，下载服务，支持文本，视频，图像日志。

- 1) 使用 C++11 开发，跨平台，支持 windows/Linux，含 aarch64 架构 CPU（包括国产 CPU 和嵌入式系统），Linux 系统（含国产麒麟，银河等 Linux 系统）。
- 2) 支持分仓和混仓模式，关联的不同软件日志可以分目录存储或者混合存储以便分析顺序逻辑。
- 3) 支持 ERR,WRN,INF,MOR,DBG,ALL 五个级别。
- 4) 支持 WEB 页面查阅，下载，支持 IPv6 接入。
- 5) 可配置按照时间和空间门限自动删除过期日志文件。
- 6) 文本日志支持高亮着色。
- 7) web 页面支持视频播放，图像显示
- 8) 支持超大文件下载。
- 9) 可配置映射多个 httproot 目录，当作 HTTP/HTTPS 下载服务器使用。
- 10) 可配置 DNS 服务，提供 ipv4/ipv6 私有域名解析服务。

1.1 系统结构

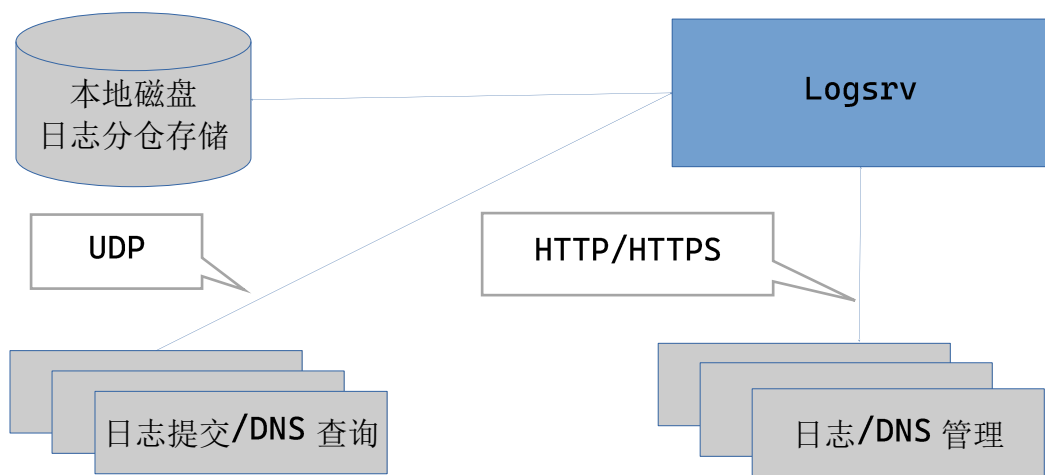


图 1.1 Logsrv 系统结构图

注：

日志生产者各种产生日志的程序。

日志使用者为专门的日志分析程序或者使用 **Logsrv** 提供的 **WEB** 服务的客户端浏览器微软 **edge**, **chrome**, **Firefox** 等。

1.2 运行环境

Windows:

windows server 2012 及以上, **Windows 10/11**, **Windows7/8**(不推荐)。

Linux:

X86_64 平台: 内核 **3.10** 及以上, **glibc 2.17** 及以上, **libstdc++ 6.0.19** 及以上。

Arrch64 平台: 内核 **4.4** 及以上, **glibc 2.23** 及以上, **libstdc++ 6.0.21** 及以上。

2.安装

Windows 安装为 **service** 后台服务，自注册安装。Linux 安装为 **systemd** 后台服务，提供 **shell** 安装脚本，安装前先规划并配置 **logsrv.ini** 文件。

2.1 基本配置

Windows 和 Linux 下配置是一样的，只是注意两个系统下目录写法有所区别。下面使用 Linux 下配置为例说明。

#日志配置

[log]

#日志提交协议,目前仅支持 **udp** 协议

protocol = udp://0.0.0.0:999

#protocol_ipv6 = udp://[::]:999

#日志存储的跟目录

path = /home/logsrv/

#日志存储级别[err,wrn,msg,mor,dbg,all]之一

level = all

#level_self 是 ver2.1.5 build 2004-7-26 新加属性，logsrv 自己输出的日志级别

level_self = dbg

#日志存储模式,[mix,split] 文件存储模式, mix:混合存储; split:分开存储

save_mode = split

#日志文件保存有效期, 0 表示长期有效,大于 0 表示过期天数.超期的将被删除

valid_days = 180

#日志文件大小配置,单位 KB, 配置范围 200-2048, 默认值 512

file_size = 512

```
# max number files per cabin, >= 10, default 1000
```

```
cabin_max_files = 300
```

```
#web 服务配置,用于查看下载日志,配置证书使用 HTTPS, 否则用 HTTP
```

```
[webserver]
```

```
#服务协议, 如果使用 https, 需要配置协议为 https://0.0.0.0:999
```

```
protocol = http://0.0.0.0:999
```

```
protocol_ipv6 = http://[::]:999
```

```
maphttproot = VIDEOS:/data/videos
```

```
maphttproot = PHOTOS:/data/photos
```

```
maphttproot = DOCS:/data/docs
```

```
#根证书,可选
```

```
ca_root = ; root certificate file
```

```
#服务器证书, 标准 X.509 服务器证书全路径文件名,含 RSA 公钥
```

```
ca_server = ; server certificate file
```

```
#服务器私钥, 服务器 RSA 私钥全路径文件名
```

```
ca_private = ; server private key file
```

```
#日志管理者
```

```
username = admin ; web 登录账号
```

```
userpswd = admin ; web 登录密码
```

整个配置 `ini` 文件分为两个小节, `[log]` 小节为日志提交服务, `[webserver]` 为日志查询下载服务配置。 `[DNS]` 小节为 DNS 服务, 参见第 6 章 DNS 服务。

[log]小节的 `protocol` 字段, 如果配置 `udp://127.0.0.1:999` 则表示只有本机的应用可以提交日志。

使用了默认的 999 端口, 提交是 UDP 通道, 默认没开启 `ipv6`; 查询下载使用 TCP 通道, 默认开启了 `ipv6`, 可同时支持 `ipv4` 和 `ipv6` 的接入。防火墙注意要例外 999 端口的 UDP 和 TCP 协议。

[webserver]中 `maphttproot` 可以配置多个, 映射多个不同的目录供 `logsrv` 的 `http/https` 提供下载服务。用冒号分为两段, 前面为显示名, 后面为全路径发布目录, 例:

```
maphttproot = VIDEOS:/data/videos
```

在 `pubfile.html` 页面显示, 如图 2 多目录发布 VIDEOS 目录视频在线播放服务。



图 2 多 HTTP/HTTPS root 视频目录发布

2.2 用户账号配置

在 `logsrv fileversion 2.0.0.7` 版及以后版本里, 可以通过配置 `loguaer.json` 文件增加访问日志的用户账号, 每个账号可以配置能访问的日志仓名(一级子目录名)。默认的 `loguser.json` 如下, 配置了一个 `rdb` 账号只能访问 `rdbsrv` 的日志。

```
{  
  "note": "这是 Logsrv 的受限用户配置文件",  
  "users": [  

```

```

{
  "name": "rdb",
  "pswd": "rdb",
  "cabins": [
    "sip3"
  ]
}
]
}

```

其中 **users** 字段是一个对象数组，可以添加用户账号。用户下的 **cabins** 字段是一个字符串数组，可以添加需要访问的仓名，*表示全部。如下例子添加一个 **ioserver** 账号可访问 **rdbsrv** 和 **io** 日志，添加一个 **log** 账号可以访问全部日志：

```

{
  "note": "这是 Logsrv 的受限用户配置文件",
  "users": [
    {
      "name": "sip3",
      "pswd": "sip3",
      "cabins": [
        "sip3"
      ]
    },
    {
      "name": "ioserver",
      "pswd": "ioserver",
      "cabins": [
        "sip3",
        "io"
      ]
    },
  ]
}

```

```

{
    "name": "log",
    "pswd": "log",
    "cabins": [
        "*"
    ]
}
]
}

```

注1) loguser.json 编码为 UTF8

注2) [webserver]中 maphttproot 发布的公共文件不受用户账号限制。

2.3 Windows 安装

配置正确确认无误后，开启防火强例外端口。将 /logsrv/logsrv_win64 目录拷贝到目标系统 C 盘。用管理员权限在 c:/logsrv_win64 目录下打开命令行窗口运行如下命令。

```

Microsoft Windows [版本 10.0.22621.1778]
(c) Microsoft Corporation。保留所有权利。

C:\Users\kipway>cd /logsrv_win64

C:\logsrv_win64>dir
驱动器 C 中的卷没有标签。
卷的序列号是 DE55-B359

C:\logsrv_win64 的目录
2023/06/10  10:03    <DIR>          .
2023/06/10  10:03    <DIR>          httpdoc
2023/06/09  09:42             1,792,000 logsrv.exe
2023/06/06  11:06              1,195 logsrv.ini
2023/05/30  14:53             13,794 mime.ini
                3 个文件          1,806,989 字节
                2 个目录 782,304,710,656 可用字节

C:\logsrv_win64>logsrv -install

```

图 3 Windows 下安装

显示安装成功后，再次检查配置，然后到 **windows** 的[服务]里启动 **logsrv**。如果启动失败，一般是目录配置错误，检查[log]小节 **path** 字段是否正确，注意目录可以不预先创建，**Logsrv** 可以自动创建。



图 4 windows 下启动 **logsrv** 服务

2.4Linux 下安装

X86_64 和 **Aarch64** 平台安装方式完全一样。配置正确确认无误后，开启防火强例外端口。然后将安装目录拷贝到目标系统 **/home** 目录下(也可以是其他目录)。注意不要搞错了：**x86_64** 平台：**/logsrv/ logsrv_x86_64_inst**

Aarch64 平台：**/logsrv/ logsrv_aarch64_inst**

下面使用 **x86_64** 平台，假设目标系统 **IP** 为 **192.168.1.59**，使用 **windows** 的 **Linux** 子系统(可以保证执行文件不丢属性)

第一步：拷贝安装文件夹

```
scp -r logsrv_x86_64_inst root@192.168.1.59:/home/
```

第二步：**ssh** 到目标系统

```
ssh root@192.168.1.59
```

第三步：执行安装脚本

```
cd /home/logsrv_x86_64_inst
./install.sh
```

安装脚本会自动启动 **logsrv** 服务。

如果启动不成功，检查配置，主要检查 **path** 字段，还有 **999** 端口是否被占用等。后期可以用 **systemd** 的命令启停 **logsrv** 服务。例如：

```
systemctl stop logsrv.service
```

```
systemctl start logsrv.service
```

安装时如果使用其他终端管理软件拷贝安装文件夹，注意检查 **install.sh** 和 **logsrv** 的可执行属性是否存在。

UDP 和 **HTTP/HTTPS** 的服务端口和服务 **IPv4** 地址均可配置。如果 **UDP** 服务配置为 **127.0.0.1** 等环形地址则容许本机的应用写入日志。

- 日志读取下载和在线显示协议：传输层采用标准通道(**WS/WSS**)，交互数据描述层采用 **JSON** 对象表示。支持各种客户端使用，包括移动客户端，**WEB** 客户端，**C/S** 客户端接入。
- 日志提交协议，采用 **UDP** 协议，适合产生日志的应用程序提交日志。

本文给出的协议，可使用 **JavaScript**, **java**, **C#**, **C/C++**实现客户端 **API**。

3 日志读取协议

文本日志读取下载和在线显示协议：

传输层采用标准通道(**WS/WSS**)，交互数据描述层采用 **JSON** 对象表示。支持各种客户端使用，包括移动客户端，**WEB** 客户端，**C/S** 客户端接入。

3.1 协议层次

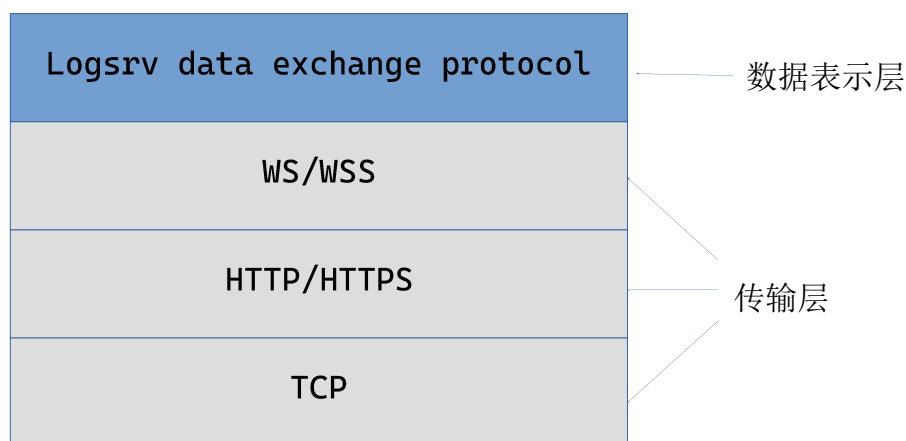


图 3.1 协议层次

HTTP：超文本协议。

HTTPS：这里是指基于 **TLS1.2** 私密通道的超文本协议

WS：从 **HTTP** 升级而来的 **websocket** 协议。

WSS：从 **HTTPS** 升级而来的 **websocket** 协议。

所有以上这些协议，**Logsrv** 都是原生支持，无需第三方库或者组件，支持裸系统部署。

3.2 TLS1.2/1.3 规范

安全性方面，**Logsrv** 支持基于 **TLS 1.2(RFC5246)**版本的安全通道的 **HTTPS** 和 **WSS** 规范，版本 **2.2.8** 之前支持 **TLS1.2** 的密码套件：

- **TLS_RSA_WITH_AES_128_CBC_SHA256** = { 0x00,0x3C };
- **TLS_RSA_WITH_AES_256_CBC_SHA256** = { 0x00,0x3D };
- **TLS_RSA_WITH_AES_128_CBC_SHA** = {0x00,0x2F};
- **TLS_RSA_WITH_AES_256_CBC_SHA** = {0x00,0x35};

版本 2.2.8 开始支持以下密码套件：

TLS1.3

TLS_AES_256_GCM_SHA384

TLS_CHACHA20_POLY1305_SHA256

TLS_AES_128_GCM_SHA256

TLS1.2

ECDHE-RSA-AES256-GCM-SHA384

ECDHE-RSA-AES128-GCM-SHA256

AES256-GCM-SHA384

AES128-GCM-SHA256

AES256-SHA256

即使不采用 TLS 安全私密通道，Logsrv 的登陆验证过程也是无密码交换的防重放攻击的登录方式。参见第 3.5.2 的登录验证。

3.3 websocket 规范

websocket 协议(RFC6455)是 html5 组成之一，是在 http 协议上的升级协议，浏览器中的 javascript 原生提供 websocket 客户端 API。使用 websocket 客户端可以实时的和服务器通信。

logsrv 提供了原生的 http 协议和 websocket 协议支持(不依赖任何第三方软件和第三方库,直接在 tcp 层提供协议支持)。http 协议为 1.1 版，websocket 协议为 13 版。

GET / HTTP/1.1

Sec-WebSocket-Version:13

为了便于描述，在 logsrv 中基于 websocket 的应用层协议命名为 logws 协议。websocket 协议支持文本和二进制模式，为方便 javascript 处理，logws 协议采用文本模式，服务器发送的报文采用字符串模式 JSON 对象。

由于 logsrv 提供的 http 协议是只读的，直接使用 C++实现，没有使用第三发库，没有 PUT、POST、DELETE 等命令，因此是绝对安全的。

3.3.1 websocekt 压缩扩展

logsrv 的 websocket 接口支持压缩扩展，特别适合云服务器等网络资源非常宝贵的场合。目前支持两种压缩扩展标准：

- **permessage-deflate**: 目前是最新的标准(RFC7692), **chrome,firefox** 等都支持。
- **deflate-frame**: 已废弃的标准, 目前只有 **IOS** 的 **safari** 在使用, 扩展名为 **x-webkit-deflate-frame**, **safari** 并不支持 **RFC7692** 制定的 **permessage-deflate** 压缩扩展, 为了节约移动端苹果用户的流量, **logsrv** 依旧支持这个标准。

3.3.2 连接 url 地址

WS_URL 例子:

"ws://192.168.1.61:999"

如果使用 **80** 端口则: **"ws://192.168.1.61"**

WSS_URL 例子:

"wss://192.168.1.61:999",

如果使用 **443** 端口: **"wss://192.168.1.61"**

和 HTTP 协议的 URL 中指定端口一样, 如果使用标准端口(**WS:80, WSS443**)则无需指定。

3.3.3 兼容性

支持 **html5** 标准的客户端浏览器均支持, 目前已测试兼容的客户端浏览器包括:

- **MicroSoft Edge for windows**
- **Chrome for windows/Linux, Chromium for Linux**
- **Firefox for windows/Linux**
- **Safari for IOS 9, 10, 11, 15, 16 ipad and iphone**
- **Android 8.0 及以上手机平板内置浏览器**

3.4 JSON 规范

JSON 规范使用 **RFC8259** 标准, 特殊字符按照下图转义。

To escape an extended character that is not in the Basic Multilingual Plane, the character is represented as a 12-character sequence, encoding the UTF-16 surrogate pair. So, for example, a string containing only the G clef character (U+1D11E) may be represented as `"\uD834\uDD1E"`.

```
string = quotation-mark *char quotation-mark

char = unescaped /
      escape (
        %x22 /           ; "    quotation mark  U+0022
        %x5C /           ; \    reverse solidus U+005C
        %x2F /           ; /    solidus          U+002F
        %x62 /           ; b    backspace        U+0008
        %x66 /           ; f    form feed        U+000C
        %x6E /           ; n    line feed         U+000A
        %x72 /           ; r    carriage return   U+000D
        %x74 /           ; t    tab               U+0009
        %x75 4HEXDIG ) ; uXXXX    U+XXXX

escape = %x5C           ; \

quotation-mark = %x22   ; "

unescaped = %x20-21 / %x23-5B / %x5D-10FFFF
```

需要特别提示的是：

- 字符串均为 **utf8** 编码的 **unicode** 字符串。
- 使用双引号分离字段名和字符串类型的字段值，字段值中出现双引号需要使用字符 `'\"'` 转义。
- 不能有注释。`//`和`/**/`都不能有（规范也没有）。
- 除了支持双引号的转义，其他转义严格按照 **RFC8259** 约定。

3.5 命令消息定义

3.5.1 请求/应答报文

每个请求报文必有两个字段：**request** 和 **seqno**，其他字段更具 **request** 不同而不同；**request** 是请求命令，字符串型。

seqno 是大于 0 的整数，客户端填写，服务端原样返回，用于 **request/response** 配对使用，建议每次请求 **seqno** 加 1，达到 **MAX_INT32(2,147,483,647)**后重新从 1 开始。

例如：

```
{
  "request": "ws_login",
  "seqno": 1,
  "name": "admin"
}
```

每个应答报文必有 3 个字段：**response**, **seqno**, **status**,其他字段更具 **request** 不同而不同；

response 字符串型，填写的客户端的 **request** 字段值。

seqno 整数，原样填写客户端请求报文中的值。

status 整数，处理结果状态，0 表示成功，其他为错误码。

例如：

```
{
  "response": "ws_login",
  "seqno": 1,
  "status": 0,
  "vals": "4F2BB96A64DE184DE4CA65765645B55D"
}
```

3.5.2 登录

登录采用无密钥交换的两次握手方式，即使不走 **WSS** 私密通道也无法截获密钥。

其原理为：客户端发送登录报文，带有自己的用户名(账号)，服务端回答一个随机字符串。客户端使用随机字符串加上自己密码的 **MD5** 字符串在做 **MD5** 散列计算，并将结果发给服务端，服务端用同样方法验证客户端是否合法。

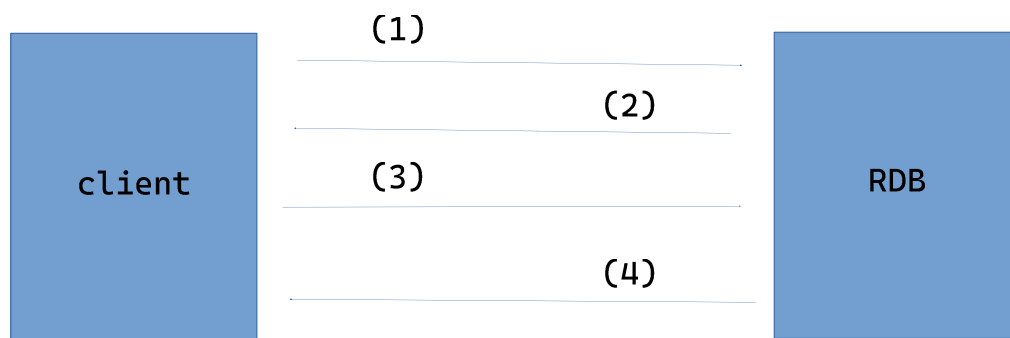


图 3.5.2 登录交互图

(1)client 发

送 `ws_login` 报文

```
{
  "request": "ws_login",
  "seqno": 1,
  "name": "admin"
}
```

其中 `name` 字段为 `client` 的用户名，这里使用 `admin` 作为例子。

(2)应答 `ws_login` 报文

```
{
  "response": "rdb_login",
  "seqno": 1,
  "status": 0,
  "vals": "4F2BB96A64DE184DE4CA65765645B55D"
}
```

其中 `vals` 为服务端发送的随机字符串。

如果失败，则会返回无此用户。例如：

```
{
  "response": "ws_login",
  "seqno": 1,
  "status": 33,
  "message": "no user!"
}
```

(3)`client` 使用随机数和自己的密码做 MD5 散列计算

这里假设密码也是"admin", 客户端在收到的 **vals** 字段的字符串的尾部加上自己密码的 MD5 的 32 字节 HEX 大写书写格式的字符串

("21232F297A57A5A743894A0E4A801FC3")组成新的字符串

(4F2BB96A64DE184DE4CA65765645B55D21232F297A57A5A743894A0E4A801FC3),

然后再计算新字符串的 MD5 摘要转换为 32 字节的 HEX 大写书写格式

("E1446FC255396A2AF49C0F8F6F5F8466")使用 **rdb_auth** 命令发送给 RDB:

```
{
  "request": "ws_auth",
  "seqno": 2,
  "vals": "E1446FC255396A2AF49C0F8F6F5F8466"
}
```

(4)应答验证

服务端会使用相同的算法计算验证字符串, 如何匹配表示客户端合法, 则会返回成功报文, 如下:

```
{
  "response": "ws_auth",
  "seqno": 2,
  "status": 0,
  "vals": "login success"
}
```

如果验证失败会返回密码错误报文, 如下:

```
{
  "response": "ws_auth",
  "seqno": 2,
  "status": 34,
  "message": "password error!"
}
```

具体登录过程可以参见 **logsrv** 里的 **javascript** 代码。

3.5.3 读取日志文件列表 **ws_loglist**

读取指定仓的日志列表, 这个功能用于查询目前有哪些仓和日志文件。使用这个命令可实现日志资源浏览器。

请求字段说明

KEY	VAL 类型	说明
-----	--------	----

request	string	"ws_loglist"
seqno	int	>0, 客户端填写, 服务端原样返回
cabin	string	仓名, 无此字段或者该字段为 nul 表示读取混合仓

应答字段说明

KEY	VAL 类型	说明
response	string	"ws_loglist"
seqno	int	>0, 客户端填写, 服务端原样返回
status	int	0 表示成功;
vals	object array	记录集, JSON 对象数组, 每个对象字段定义: "file" : 文件名; "dir" : int ; 是否是目录, 0: 文件; 1: 目录 "size": string ; 文件大小 "modify" : string; 修改日期

例子:

请求读取混仓:

```
{
  "request": "ws_loglist",
  "seqno": 1000
}
```

应答:

```
{
  "response": "ws_loglist",
  "seqno": 1000,
  "status": 0,
  "message": "OK",
  "vals": [{
    "file": "tstec3",
    "dir": 1,
```

```

        "size": "0",
        "modify": "2020-10-28 10:23:21"
    }, {
        "file": "log-20201103-0001.txt",
        "dir": 0,
        "size": "416",
        "modify": "2020-11-3 11:30:27"
    }, {
        "file": "log-20201102-0001.txt",
        "dir": 0,
        "size": "416",
        "modify": "2020-11-2 11:39:31"
    }
]
}

```

3.5.4 读取日志 ws_logget

读取指定文件名的日志，日志会使用 **base64** 编码返回。这个功能用于下载日志文件。

请求字段说明

KEY	VAL 类型	说明
request	string	"ws_logget"
seqno	int	>0, 客户端填写，服务端原样返回
cabin	string	仓名，无此字段或者该字段为 null 表示读取混合仓
filename	string	日志文件名

应答字段说明

KEY	VAL 类型	说明
response	string	"ws_logget"
seqno	int	>0, 客户端填写，服务端原样返回

status	int	0 表示成功;
vals	string array	第一个字符串为日志文件内容使用 base64 编码后的字符串

例子:

request:

```
{
  "request": "ws_logget",
  "seqno": 3002,
  "cabin": "testec3",
  "filename": "20200407-0001.txt"
}
```

response:

```
{
  "response": "ws_logget",
  "seqno": 3002,
  "status": 0,
  "message": "OK",
  "vals": ["ZGVtbyB0ZXh0IQ=="]
}
```

3.5.4 读取日志用于显示 ws_logview

读取指定文件名的日志，日志内容会先转为 **utf-8** 编码，然后 **base64** 编码返回。这个功能用于在线浏览日志。

请求字段说明

KEY	VAL 类型	说明
request	string	"ws_logview"

seqno	int	>0, 客户端填写, 服务端原样返回
cabin	string	仓名, 无此字段或者该字段为 null 表示读取混合仓
filename	string	日志文件名

应答字段说明

KEY	VAL 类型	说明
response	string	"ws_logview"
seqno	int	>0, 客户端填写, 服务端原样返回
status	int	0 表示成功;
vals	string array	第一个字符串为日志文件内容先转为 utf-8 编码, 然后 base64 编码的字符串。

例子:

request:

```
{
  "request": "ws_logview",
  "seqno" : 3003,
  "cabin": "testec3",
  "filename" : "20200407-0002.txt"
}
```

response:

```
{
  "response": "ws_logview",
  "seqno" : 3003,
  "status" : 0,
  "message" : "OK",
  "vals" : ["5Lit5paH5L6L5a2Q5pe15b+XIQ=="]
}
```


4 日志提交协议

日志提交协议采用 UDP 通道，默认端口 999，无连接信任模式。如果不想本机以外的写入，可以将配置文件 [log] 小节下的 `protocol` 由默认的 `udp://0.0.0.0:999` 改为 `protocol = udp://127.0.0.1:999`

每条日志受限 udp 报文大小，小于 65507，实际应用建议小于 32K。

4.1 日志提交协议定义

udp 报文分为报文头和日志内容。中间使用 '\n' 符分开，即 ASCII 码的 0x0A 字符。

报头	\n	日志内容
----	----	------

4.1.1 报头

报头是一个 utf-8 编码的 JSON 对象。有三个字段：

`order : string` 目前只有一个 "wlog" 命令

`level : Number`，日志级别；目前级别的定义如下：

```
#define CLOG_DEFAULT_ERR 10
#define CLOG_DEFAULT_WRN 20
#define CLOG_DEFAULT_MSG 30
#define CLOG_DEFAULT_MOR 31
#define CLOG_DEFAULT_DBG 40
#define CLOG_DEFAULT_ALL 999
```

只有 `level` 小于等于 `logsrv.ini` 中的配置才会落地。

`cabin : string` 仓名，必须要提供一个仓名，一般一个仓名代表一个日志提交者。

报头例子：

```
{"order": "wlog", "level": 30, "cabin": "rdbsrv"}
```

4.1.2 日志内容

不要求编码（可以为 utf8 编码或者 UBK 编码，不要混合编码，否则会导致显示乱码），不求语言，`logsrv` 会原样保存，建议统一规范使用 utf8 编码以便于跨系统平台显示阅读。

5 协议实现实例

日志读取实例可参见 `logsrv` WEB 页面中的 `javascript` 代码。

日志提交的 C++ 实现参见 `eclib3/ec_log.h` 中 `class ec::udplog`

6 DNS 服务

从 ver2.1.5 build2024-7-26 开始，logsrv 提供了 DNS 服务，符合 RFC1034, RFC1035, RFC3596 规范，使用 53 端口，用于私有域名或子域名解析服务。

6.1 解决什么问题

目前宽带通过 pppoe 拨号上网后，如果路由器支持 ipv6，即可获得一个公网 ipv6 地址子网，局域网中的每一台主机可以获得一个公网 ipv6 地址，异地计算机可以直接互联。但是每次 pppoe 拨号后，ipv6 地址前缀会变，导致局域网内的主机无法固定 ipv6 地址。好在路由器重拨不会很频繁，除非断电，一般都是一周才会重拨一次。重拨后路由器会为每台主机重新分配 ipv6 地址（一般只改变运营商提供的地址前缀，一般前缀 64bit）。ipv6 有 128bit 很长，不容易记忆。logsrv 的 DNS 服务提供 ipv6 私有域名解析服务，无需注册，部署在公网可以实现异地互联互通。

6.2 实现原理

在具有公网 ipv6 的云服务器上部署 logsrv，开启 DNS 服务，会在 53 端口提供 udp 服务，支持 A(ipv4), AAAA(ipv6) 记录查询，如果找到域名直接返回相应的查询结果，没有找到记录会转发到配置的公有 DNS 服务查询，并将查询结果返回客户端。

logsrv 使用日志的查询协议新增了 DNS 相关服务命令用于提交，查询，删除域名记录。可以通过 test.html 页面手动提交域名解析记录，或者通过另外一个本地服务程序 ipv6ds 自动采集本机公网 ipv6 提交到 logsrv，支持一个域名多个 ip。

6.3 DNS 服务开启

logsrv 新增一个 [dns] 小节用于配置 DNS 服务，服务 udp 端口 53 不需配置。注意防火墙例外 udp 53 端口。

[dns]

#use port 53 , udp, 这一句是纯注释

#dns 自己域名，用于提供本机的反向查询和 NS 查询

dns_name = dns.yourname.net

#绑定的 ipv4 服务 ip, 不填表示不提供 ipv4 的 dns 服务

#dns_bindipv4 = 0.0.0.0

#绑定的服务 **ipv6**，不填表示不提供 **ipv6dns** 服务，可以填写 **::** 或者本机公网 **ipv6** 地址

dns_bindipv6 = 240e:1330:250e:a501::169

#反向查询用的本机的 **ipv4**，填写公网对外 **ipv4**，不填表示不提供反向查询

#dns_ptripv4 = 139.178.111.36

#反向查询用的本机的 **ipv6**，填写公网对外 **ipv6**，不填表示不提供反向查询

dns_ptripv6 = 240e:1330:250e:a501::169

#当本地查询失败时，转发到公共的 **dns** 服务 **ipv4**，默认阿里 **223.5.5.5**

dns_serveripv4 = 223.5.5.5

#当本地查询失败时，转发到公共的 **dns** 服务 **ipv6**，默认 阿里 **2400:3200::1**

dns_serveripv6 = 2400:3200::1

#提交 **dns** 解析信息的账号

dns_user = dnsup

dns_pswd = dnsup

#支持 **https** 的端口，不填表示没有，多个用逗号分开

dnshttpsports = 999,1001

上面的配置中，只配置 **ipv6** 服务，没配置 **ipv4** 服务

一般的云服务器使用 **ip a** 命令能看到公网 **ipv6**，如果看不到，**dns_bindipv6** 要配置为本地 **any** 地址，即两个冒号。

一般的云服务器尤其时国内的，使用 **ip a** 看不到公网 **ipv4**，**dns_bindipv4** 需要配置为 **0.0.0.0**，

dns_bindipv6 和 dns_ptripv4 都需要配置本机的公网 IP。

6.4 本地 DNS 配置

如果做为子域名解析(参见 6.8)，不需要本地配置，请直接跳过。

6.4.1 windows 系统

将首选 dns 服务器配置为 logsrv 的公网 ip，以 ipv6 为例：

编辑 DNS 设置

关

备用 DNS

DNS over HTTPS

关

IPv6

☒ 开

首选 DNS

2408:4002:1175:2fcd3:fb99:2b8e:bbe

DNS over HTTPS

关

备用 DNS

DNS over HTTPS

关

如果需要解析 **ipv4** 域名, 同样需要将 **ipv4** 的首选 DNS 服务器填写 **logsrv** 的公网 **ipv4**, 注意 windows 系统中, 如果公网 **logsrv** 只开启 **ipv4** 的 DNS 服务, 是无法解析 **ipv6** 域名的, 因为 Windows 不会向 **ipv4** DNS 服务器发送 AAAA 查询命令。

6.4.2 Linux 系统

包括 WSL 系统, 修改 `/etc/resolv.conf` 第一个 `nameserver` 为 **logsrv** 的 **ipv6** 公网地址。如下例

```
nameserver 2408:3002:1275:220b:fcd3:fb99:32b2:4bbc
```

```
nameserver 8.8.8.8
```

具体修改方式请 Google 查找。

6.5 本地 ipv6 自动提交

另外提供了一个后台服务程序 **ipv6ddns** 负责定时采集本机公网 **ipv6** 地址, 使用配置的域名将 AAAA 记录提交到 **Logsrv** 的域名解析库。

ipv6ddns 后台服务程序例子配置

#本机私有域名

```
hostname = server1.yourdns.net
```

#本机的日志服务器, 只是用于 **ipv6ds** 记录日志

```
logurl = udp://127.0.0.1:999/ipv6ddns?level=dbg
```

#DNS 服务器(公网上的 **logsrv**)用于提交域名 AAAA 记录的 **ws** 服务 url, 也就是 **logsrv** 的公网上的日志服务 url, 走 **websocket** 通道, 有证书用 **wss**, 没有用 **ws**, **ip** 地址可以用注册域名。

```
dnsurl = ws://139.178.111.36:999
```

#DNS 服务器登录账号

```
dnsuser = dnsup
```

#DNS 服务器登录密码

`dnspswd = dnssup`

本机私有域名 `hostname` 可以随意定，一般用三个字母表示根区，比如

`"myo"` 表示我的 `office`

`"myh"` 表示我的 `home`

如果是作为子域名解析，则严格按照子域名格式填写，比如 `server1.yourdns.net`

6.6 测试

公网 `logsrv`，本机 `ipv6ddns` 正常运行，检查日志没有错误后。先用 `nslookup` 测试，测试时指定 DNS 服务器为 `logsrv` 的 `ipv6`。正常后，再使用 `ping` 测试。

若 `nslookup` 测试不通过，在 `logsrv` 的 `test.html` 页面发送如下命令看有没有域名记录：

```
{"request": "ws_dnslist", "seqno": 1001, "name": "*"}
```

`nslookup` 测试通过，但是 `ping` 测试不通过，一般是本机首选 `dns` 配置不对。

一切正常后，远程终端，远程桌面都可以使用私有域名了。

例：

```
ssh root@server.myo
```

6.7 DNS 记录管理命令

协议同 3 章日志读取协议，登录后，发送 JSON 命令对域名记录做增删改查，`ip6vddns` 也是通过下面的命令自动提交域名记录。

可以通过 `logsrv` 新增的 `test.html` 页面测试或手动管理 DNS 记录。升级时需要将 `test.html` 复制到 `logsrv` 安装目录 `httpdoc` 子目录下。

6.7.1 提交/更新

每次提交一个记录，用于新增和更新，管理者和 `dns_user` 有权限。

请求命令 `"ws_dnupdate"`

```
{  
    "request": "ws_dnupdate",  
    "seqno": 1000,  
    "item": {
```

```

        "name": "hpegen10.lisz",
        "ipv6s": ["240e:3130:250e:a500::f9", "240e:3130:250e:a500::fa"],
        "ipv4s": ["139.124.102.23"],
        "httpsports": [999, 10001]
    }
}

```

ipv6s, ipv4s 字段是 **ip** 字符串数组, 没有的不提供, 至少提供一个。

支持 HTTPS, **qtype=65** 查询, **httpsports** 字段为支持 **https** 的端口数组, 可选。

成功应答:

```

{
    "response": "ws_dnsupdate",
    "seqno": 1000,
    "status": 0,
    "message": "success"
}

```

如果失败, **status** 不为 0, **message** 为错误信息。

6.7.2 删除

每次删除一个记录, 仅管理者有权限。

请求命令 **"ws_dnsdelete"**

```

{
    "request": "ws_dnsdelete",
    "seqno": 1001,
    "name": "home214.lisz"
}

```

name 字段为需要删除的域名。

成功应答:

```

{
    "response": "ws_dnsdelete",
    "seqno": 1001,
}

```

```
    "status": 0,  
    "message": "success"  
}
```

如果失败，`status` 不为 0，`message` 为错误信息。

6.7.3 查找

查找匹配的记录，返回一批，管理者和 `dns_user` 有权限。

请求命令 `"ws_dnslist"`

```
{  
    "request": "ws_dnslist",  
    "seqno": 1002,  
    "name": "*"  
}
```

`name` 字段为需要匹配的域名匹配串，可以是完整域名，或是带 `*?` 的字符串

成功应答：

```
{  
    "response": "ws_dnslist",  
    "seqno": 1002,  
    "status": 0,  
    "items": [  
        {  
            "name": "gen10s.jyh",  
            "ipv6": "240e:1330:250e:a501::f412",  
            "timestamp": "2024-07-23T16:27:38.140+08:00"  
        },  
        {  
            "name": "nuc3350.jyo",  
            "ipv6s": ["240e:1330:450e:f5c1::1f96"],  
            "timestamp": "2024-07-23T16:28:02.083+08:00"  
        }  
    ]  
}
```

```
    ]  
}
```

如果失败，`status` 不为 0，`message` 为错误信息。

6.7.4 读取

读取指定域名的记录，返回一个，管理者和 `dns_user` 有权限。

请求命令 `"ws_dnsget"`

```
{  
    "request": "ws_dnsget",  
    "seqno": 1003,  
    "name": "gen10s.jyh"  
}
```

`name` 为需要读取的域名。

成功应答：

```
{  
    "response": "ws_dnsget",  
    "seqno": 1003,  
    "status": 0,  
    "item": {  
        "name": "gen10s.jyh",  
        "ipv6s": ["240e:1330:250e:a501::f412"],  
        "timestamp": "2024-07-23T16:31:38.136+08:00"  
    }  
}
```

如果失败，`status` 不为 0，`message` 为错误信息。

6.8 子域名解析服务用法

如果在公有云上有一个已经注册的域名，比如 `"kiptest.net"`，则可以在上面安装 `logsrv` 的 DNS 服务来解析其子域名，例如 `"mysql1.kiptest.net"`，这种方法是标准用法，不需要对客户端机器做任何设置（不需要更改首选 DNS 服务器）。下面以阿里云为例描述操作。

原理：子域名使用 NS 记录，指向我们自己的 DNS 服务器(logsrv)。

基本条件：在阿里云上有一个 ECS 或轻量服务器，有 ipv4 固定 ip, 开启 ipv6 固定 IP。并且注册一个域名 “kiptest.net” 指向这个服务器。

在阿里云的域名管理页面里：

- 1) 添加一个 AAAA 记录指向子域名 “dns.kiptest.net” , 记录值为这个云服务器的 ipv6 地址。
- 2) 添加一个 A 记录指向子域名 “dns.kiptest.net” , 记录值为这个服务器的 ipv4 地址。这个步骤可选，logsrv 开启 ipv4 的 DNS 服务才需要。
- 3) 上两步完成就有了一个域名解析的子域名，用于添加其他子域名。现在需要添加一个 “mysql1.kiptest.net” 子域名由我们的 logsrv 的 DNS 服务来解析。只需添加一个 NS 记录，意思就是让 “mysql1.kiptest.net” 由指向的 DNS 服务器来解析，其记录值为 “dns.kiptest.net”。设置完成后，最多等 10 分钟。
“mysql1.kiptest.net” 就是一个任何主机都可以访问的合法子域名。

☐	主机记录 ?	记录类型 ?	解析请求来源(isp) ?	记录值 ?	TTL ?
☐	dns	AAAA	默认	2408:.....:fcd3:fb99:22b2:4b.....	10 分钟
☐	dns	A	默认	139.....30	10 分钟
☐	hpgen10	NS	默认	dns.l.....net	10 分钟

图 6-8-1 子域名配置截图

由于域名 **kiptest.net** 已经注册备案，其下面的子域名不需要再备案。上面截图 **hpgen10** 位置填写 **mysql1**

为了防止 **mysql1** 主机的 **ipv6** 地址因路由器拨号发生变化，还是需要 **ipv6ds** 后台服务监测 **ipv6** 变化并将变化后的 **ipv6** 地址同步到 **logsrv** 的域名库。