



应用负载均衡产品 API V3.9R110 手册



适配软件版本：V3.9R110

上海弘积信息科技有限公司

修订记录

日期	修改人	修改内容
2020-03-23	lutaijie	修改 WEB 自定义证书部分
2020-03-23	lihai	SLB 全局软关机现在仅支持 node，不再支持 vip
2020-12-15	Sushijie	添加 PRO 类型健康检查，vrrp L3peer 部分
2021-01-15	Xingyanhu	增加 hosts、管理员 disable；修改 TCP profile
2021-07-1	Sushijie	增加了日志配置模块
2021-08-1	xintang	HTTP 模板增加 URL 限速参数
2021-08-16	Lihai	修改添加了 SNAT pool 部分参数；HTTP 模板增加了插入客户端 IP 部分
2022-1-15	Lihai	修改了 SLB 策略/规则表中部分参数范围
2022-1-15	Sushijie	修改了系统/网络联通/ping 的部分参数和参数范围
2022-1-15	Yejianru	修改了 GSLB 组的 ID 参数范围和域名服务的 weight 参数范围
2022-1-15	Lihai	增加了部分 SLB 健康检查的公共参数，主要针对 pro 类型的健康检查
2022-1-15	Niejuanjuan	修改了 SSL 卸载的客户端卸载和服务端卸载两个模块
2022-2-10	Sushijie	增加了 VRRP 手动同步配置部分以及触发配置同步命令
2022-2-10	Lihai	增加了 SLB 个别虚拟服务中的模板参数
2022-2-10	Lihai	增加了 SLB 模板中的虚拟服务模板；修改了 TCP 模板的 timeout 参数
2022-2-10	Yejianru	修改了网络 IPv4 扩展列表中 timerange 的参数范围
2022-2-10	Sushijie	增加了网络 IPv6 访问列表模块
2022-2-10	Zhangkuo	修改了 SLB 服务池的个别参数以及 cookie 保持的个别参数
2022-2-10	Lihai	添加和修改了 SLB 健康检查的部分公共请求参数和 HTTP、TCP、UDP 的个别参数
2022-2-10	Sushijie	其他个别参数不做详细记录

目录

修订记录	2
目录	3
总则	14
Authkey 认证密钥	14
URL API 使用的 URL	14
Method API 使用的 HTTP method	14
文件上传	14
必选参数	14
可选参数	14
响应内容	14
登陆和登出	15
登陆	15
登出	15
系统	16
系统信息	16
获取设备信息	16
获取设备许可信息	17
获取整机性能信息	18
获取应用加速统计信息	18
获取应用状态数量统计信息	19
获取设备物理接口分布信息	20
安全标识	22
获取登录安全标识	22
设置登录安全标识	22
系统配置	23
DNS 配置设置	23
DNS 配置获取	23
设置系统主机名	24
主机名 IPV4 地址获取	24
主机名 IPV4 地址设置	24
主机名 IPV6 地址获取	25
主机名 IPV6 地址设置	25
Hosts 列表获取	26
Hosts 增加	26
Hosts 删除	27
获取系统 banner 信息	27
时间配置设置	28
时间配置获取	30
网络管理 SNMP	30
添加 SNMP 团体字	30
获取 SNMP 团体字列表	31
删除 SNMP 团体字	31
设置 SNMP 服务配置	32

获取 SNMP 服务配置	32
添加 SNMP TRAP	33
获取 SNMP TRAP 列表	33
删除 SNMP TRAP	34
设置 SNMP TRAP 配置	34
获取 SNMP TRAP 配置	34
用户管理	35
添加用户	35
获取用户列表	36
获取指定用户	37
编辑指定用户	38
删除指定用户	38
设置用户锁定和密码配置	39
获取用户锁定配置和密码配置	39
解锁锁定用户	40
修改当前用户密码	40
获取管理员状态	41
禁用管理员用户	41
客户端锁定	42
设置客户端锁定配置	42
获取客户端锁定配置	42
获取锁定客户端列表	43
解锁锁定客户端	43
AAA 认证	44
AAA 全局配置设置	44
AAA 全局配置获取	45
Radius 认证配置设置	45
Radius 认证配置获取	46
TACACS+认证配置设置	47
TACACS+认证配置获取	48
许可管理	49
许可信息获取	49
许可文件上传	50
NTP 配置	50
添加 NTP 配置	50
获取 NTP 配置列表	51
获取指定 NTP 配置	52
编辑 NTP 配置	52
删除指定 NTP 配置	53
WEB 会话	53
设置 WEB 会话超时时间	53
获取 WEB 会话超时时间	53
设置 WEB 会话限制	54
获取 WEB 会话限制配置	54
WEB 证书管理	55
获取 web 证书列表	55

删除 web 证书	57
应用 web 证书	57
上传 web 私钥文件.....	58
下载 web 私钥文件.....	58
上传 web 证书文件.....	58
下载 web 证书文件.....	59
系统应用端口	59
设置系统应用端口配置	59
获取系统应用端口配置	60
系统操作	60
保存配置	60
系统重启	60
重新加载配置	61
系统关机	61
系统恢复默认配置	61
配置备份管理	62
添加启动配置文件备份	62
获取启动配置文件列表	63
指定启动配置文件	63
删除指定启动配置文件	64
配置导出	64
配置导入	64
自动备份管理	65
获取自动备份配置	65
设置自动备份配置	65
自动备份恢复	66
自动备份列表	67
自动备份删除	67
诊断信息	67
诊断信息导出	67
版本镜像管理	68
获取版本镜像列表	68
指定镜像	68
通过 FTP/TFTP 方式上传镜像	69
通过 HTTP/HTTPS 方式上传镜像	70
网络连通性测试	70
Ping 请求	70
Ping 结果	71
抓包	72
添加入接口过滤条件	72
获取入接口列表	72
删除入接口	72
添加出接口过滤条件	73
获取出接口列表	73
删除出接口	74
添加过滤规则	74

获取过滤规则列表	75
删除过滤规则	76
添加抓包文件	76
获取抓包文件列表	76
删除抓包文件	77
导出抓包文件	77
SMTP 配置	77
获取 SMTP 配置	77
编辑 SMTP 配置	78
日志配置	79
获取业务日志配置	79
编辑业务日志配置	79
获取审计日志配置	79
编辑审计日志配置	80
获取 NAT 日志配置	80
编辑 NAT 日志配置	81
日志获取	81
业务日志显示列表	81
审计日志显示列表	84
NAT 日志显示列表	87
业务日志清除	87
审计日志清除	87
NAT 日志清除	88
业务日志下载	88
审计日志下载	88
NAT 日志下载	88
诊断日志下载	89
系统日志下载	89
日志发送	89
添加日志发送配置	89
获取日志发送配置列表	90
获取指定日志发送配置	91
编辑日志发送配置	92
删除日志发送配置	93
日志告警	94
获取 Email 日志告警配置	94
编辑 Email 日志告警配置	94
获取短信日志告警配置	95
编辑短信日志告警配置	95
SLB 服务器负载均衡	96
SLB 全局混杂设置	96
获取 SLB 全局混杂配置	96
设置 SLB 全局混杂配置	96
SLB 全局路径保持	97
获取 SLB 全局路径保持	97
设置 SLB 全局路径保持	97

SLB 全局软关机设置.....	98
获取 SLB 全局软关机配置.....	98
设置 SLB 全局软关机配置.....	98
SLB 全局策略地址转换设置.....	99
获取 SLB 全局策略地址转换配置.....	99
设置 SLB 全局策略地址转换配置.....	99
虚拟服务.....	100
SLB 会话清除.....	100
节点.....	100
服务池.....	114
虚拟地址.....	126
虚拟服务.....	156
模板.....	182
健康检查.....	230
健康检查添加.....	230
健康检查列表.....	247
健康检查编辑.....	252
健康检查获取.....	252
健康检查删除.....	253
健康检查脚本列表.....	253
健康检查脚本上传.....	254
健康检查脚本删除.....	254
健康检查 POST 文件列表.....	255
健康检查 POST 文件上传.....	255
健康检查 POST 文件删除.....	256
被动健康检查.....	256
添加被动健康检查配置.....	256
获取被动健康检查配置列表.....	257
获取指定被动健康检查配置.....	257
编辑指定被动健康检查配置.....	258
删除指定被动健康检查配置.....	259
ERULE.....	259
erule 上传.....	259
erule 删除.....	260
erule 列表.....	260
erule 服务器文件上传.....	261
erule 服务器文件删除.....	261
erule 服务器文件列表.....	261
连接保持.....	262
Cookie 连接保持.....	262
源地址连接保持.....	266
目的地址连接保持.....	270
SSL 连接保持.....	273
加速.....	275
缓存.....	275
连接复用.....	281

SSL 卸载.....	283
策略.....	309
黑白名单.....	309
规则表.....	310
策略.....	314
Web 安全.....	322
WAF 模板.....	322
WAF 规则.....	328
GSLB 全局负载均衡.....	332
GSLB 侦听地址.....	332
添加 GSLB 侦听地址.....	332
获取 GSLB 侦听地址列表.....	334
获取 GSLB 指定侦听地址.....	335
编辑指定 GSLB 侦听地址.....	336
添加侦听端口.....	337
编辑侦听端口.....	339
删除侦听端口.....	340
删除 GSLB 侦听地址.....	340
GSLB 服务.....	341
添加 GSLB 服务.....	341
获取 GSLB 服务列表.....	342
获取指定 GSLB 服务.....	343
编辑指定 GSLB 服务.....	344
添加服务端口.....	346
编辑服务端口.....	346
删除服务端口.....	347
删除 GSLB 服务.....	348
GSLB 站点.....	348
添加 GSLB 站点.....	348
获取 GSLB 站点列表.....	349
获取 GSLB 指定站点.....	350
编辑指定 GSLB 站点.....	352
添加站点成员.....	353
编辑站点成员.....	354
删除站点成员.....	355
删除 GSLB 站点.....	355
GSLB 策略.....	355
添加 GSLB 策略.....	355
获取 GSLB 策略列表.....	358
获取 GSLB 指定策略.....	365
编辑指定 GSLB 策略.....	369
删除 GSLB 策略.....	374
GSLB 域名.....	374
添加 GSLB 域名.....	374
获取 GSLB 域名列表.....	376
获取 GSLB 指定域名.....	381

编辑指定 GSLB 域名	387
添加域服务	389
编辑域服务	392
删除域服务	396
删除 GSLB 域名	396
GSLB 地理位置	397
上传地理位置文件	397
增加地理位置条目	397
删除地理位置条目	397
GSLB 角色	398
GSLB 角色获取	398
GSLB 角色设置	398
GSLB 组	399
GSLB 组状态获取	399
GSLB 组状态设置	399
获取 GSLB 组邻居列表	400
添加 GSLB 组邻居	401
删除 GSLB 组邻居	401
网络	402
接口	402
网络模式	402
管理口配置	403
以太网接口配置	407
虚拟接口配置	416
汇聚接口配置	423
VLAN	428
VLAN 配置列表	428
VLAN 配置获取	430
VLAN 配置增加	431
VLAN 配置编辑	432
VLAN 配置删除	433
汇聚(TRUNK)	433
TRUNK 配置列表	433
TRUNK 配置获取	434
TRUNK 配置增加	435
TRUNK 配置编辑	436
TRUNK 配置删除	437
时间对象	437
时间对象配置列表	437
时间对象配置获取	439
时间对象配置增加	439
时间对象配置编辑	440
时间对象配置删除	441
IPv4 标准访问列表	441
IPv4 标准访问列表列表	441
IPv4 标准访问列表获取	442

IPv4 标准访问列表增加.....	442
IPv4 标准访问列表删除.....	443
IPv4 标准访问列表编辑.....	443
IPv4 标准访问列表描述设置.....	444
IPv4 扩展访问列表.....	444
IPv4 扩展访问列表列表.....	444
IPv4 扩展访问列表获取.....	445
IPv4 扩展访问列表增加.....	447
IPv4 扩展访问列表编辑.....	448
IPv4 扩展访问列表删除.....	449
IPv4 扩展访问列表描述设置.....	449
IPv6 访问列表.....	450
IPv6 访问列表列表.....	450
IPv6 访问列表获取.....	451
IPv6 访问列表增加.....	452
IPv6 访问列表删除.....	453
IPv6 访问列表编辑.....	454
IPv6 访问列表描述设置.....	455
NAT 地址转换	455
NAT 地址池	455
NAT 地址池统计信息	463
NAT 地址池组	466
静态 NAT(包括 PAT).....	469
网络 NAT	480
NAT 映射	483
地址转换策略	486
NAT 全局配置	489
地址解析	492
IPv4 ARP	492
IPv6 Neighbor	497
静态路由	502
IPv4 静态路由配置列表.....	502
IPv4 静态路由配置获取.....	503
IPv4 静态路由配置增加.....	503
IPv4 静态路由配置编辑.....	504
IPv4 静态路由配置删除.....	505
IPv6 静态路由配置列表.....	505
IPv6 静态路由配置获取.....	506
IPv6 静态路由配置增加.....	507
IPv6 静态路由配置编辑.....	507
IPv6 静态路由配置删除.....	508
静态管理路由	508
IPv4 静态管理路由配置列表.....	508
IPv4 静态管理路由配置获取.....	509
IPv4 静态管理路由配置增加.....	510
IPv4 静态管理路由配置编辑.....	510

IPv4 静态管理路由配置删除.....	511
IPv6 静态管理路由配置列表.....	511
IPv6 静态管理路由配置获取.....	512
IPv6 静态路管理由配置增加.....	513
IPv6 静态管理路由配置编辑.....	513
IPv6 静态管理路由配置删除.....	514
静态业务路由.....	515
IPv4 静态业务路由配置列表.....	515
IPv4 静态业务路由配置获取.....	515
IPv4 静态业务路由配置增加.....	516
IPv4 静态业务路由配置编辑.....	517
IPv4 静态路业务由配置删除.....	517
IPv6 静态业务路由配置列表.....	518
IPv6 静态业务路由配置获取.....	518
IPv6 静态业务路由配置增加.....	519
IPv6 静态业务路由配置编辑.....	520
IPv6 静态业务路由配置删除.....	520
OSPF.....	521
OSPF 网络列表.....	521
OSPF 网络增加.....	521
OSPF 网络删除.....	522
OSPF 状态获取.....	522
OSPF 状态设置.....	523
BGP.....	523
BGP 网络列表.....	523
BGP 网络增加.....	524
BGP 网络删除.....	524
BGP 邻居列表.....	524
BGP 邻居增加.....	525
BGP 邻居删除.....	525
BGP 状态获取.....	526
BGP 状态设置.....	526
网络安全 DDOS.....	527
获取网络安全 DDOS 配置.....	527
设置网络安全 DDOS 配置.....	527
流量控制.....	528
流量控制全局使能获取.....	528
流量控制全局使能设置.....	528
流量控制配置列表.....	529
流量控制配置获取.....	529
流量控制配置增加.....	530
流量控制配置编辑.....	530
流量控制配置删除.....	531
流量控制 RULE 配置列表.....	531
流量控制 RULE 配置获取.....	532
流量控制 RULE 配置增加.....	533

流量控制 RULE 配置编辑.....	533
流量控制 RULE 配置删除.....	534
VRRP 高可用.....	534
VRRP 全局配置.....	534
1.1 获取 VRRP 全局配置.....	534
1.2 设置 VRRP 全局配置.....	535
VRRP 组.....	536
添加 VRRP 组.....	536
获取 VRRP 组列表.....	536
获取指定 VRRP 组.....	537
编辑指定 VRRP 组.....	538
获取 VRRP 组状态列表.....	538
删除指定 VRRP 组.....	539
VRRP 心跳接口.....	540
添加以太网接口作为心跳接口.....	540
获取以太网心跳口配置列表.....	540
获取以太网心跳口统计信息列表.....	541
删除以太网心跳接口.....	541
编辑以太网心跳接口.....	542
添加汇聚接口作为心跳接口.....	542
获取汇聚心跳口配置列表.....	542
获取汇聚心跳口统计信息列表.....	543
删除汇聚心跳接口.....	543
编辑汇聚心跳接口.....	544
获取所有心跳口配置列表.....	544
获取所有心跳口统计信息列表.....	545
VRRP 浮动地址.....	546
添加 VRRP 浮动地址.....	546
获取 VRRP 浮动地址列表.....	546
删除 VRRP 浮动地址.....	547
VRRP 强制切换.....	547
获取 VRRP 强制备机信息.....	547
设置 VRRP 强制备机.....	548
设置 VRRP 强制主机.....	549
VRRP 网关监控.....	549
添加 VRRP 网关监控条件.....	549
获取 VRRP 网关监控条件列表.....	550
编辑 VRRP 网关监控条件.....	550
删除 VRRP 网关监控条件.....	551
VRRP 路由监控.....	551
添加 VRRP 路由监控条件.....	551
获取 VRRP 路由监控条件列表.....	552
编辑 VRRP 路由监控条件.....	552
删除 VRRP 路由监控条件.....	553
VRRP 以太网接口监控.....	553
添加 VRRP 以太网接口监控条件.....	553

获取 VRRP 以太网接口监控条件列表.....	554
编辑 VRRP 以太网接口监控条件.....	555
删除 VRRP 以太网接口监控条件.....	555
VRRP 汇聚接口监控.....	556
添加 VRRP 汇聚接口监控条件.....	556
获取 VRRP 汇聚接口监控条件列表.....	556
编辑 VRRP 汇聚接口监控条件.....	557
删除 VRRP 汇聚接口监控条件.....	557
VRRP 虚拟接口监控.....	558
添加 VRRP 虚拟接口监控条件.....	558
获取 VRRP 虚拟接口监控条件列表.....	558
编辑 VRRP 虚拟接口监控条件.....	559
删除 VRRP 虚拟接口监控条件.....	559
VRRP 配置同步对端.....	560
添加 VRRP 配置同步对端.....	560
获取 VRRP 配置同步对端列表.....	561
获取指定 VRRP 配置同步对端.....	561
获取 VRRP 配置同步模块.....	562
编辑 VRRP 配置同步模块.....	562
编辑指定 VRRP 配置同步对端.....	563
删除指定 VRRP 配置同步对端.....	563
触发配置同步.....	564
VRRP 自动同步.....	564
获取 VRRP 自动同步配置信息.....	564
设置 VRRP 自动同步配置.....	564
获取 VRRP 同步通知配置信息.....	565
设置 VRRP 同步通知配置.....	565
获取 VRRP 同步状态.....	565
VRRP 配置 L3peer.....	566
添加 VRRP L3peer 邻居.....	566
获取 VRRP L3peer 邻居.....	566
获取指定 VRRP L3peer 邻居.....	567
删除指定 VRRP L3peer 邻居.....	567
获取 VRRP L3peer 使能状态.....	568
编辑 VRRP L3peer 使能状态.....	568
VRRP 配置 MGMT peer.....	569
添加 VRRP mgmt peer 邻居.....	569
获取 VRRP mgmt peer 邻居列表.....	569
删除指定 VRRP mgmt peer 邻居.....	570
获取 VRRP mgmt peer 统计信息列表.....	570

总则

Authkey 认证密钥

除登陆 API 以外的 API,都需要使用登陆 API 获取的 authkey,authkey 默认超时时间为 10 分钟,当超时或者登出以后,authkey 立即失效。

URL API 使用的 URL

除特殊说明的情况之外,所有 API 的 URL 格式都为

`<protocol>://<host>:<port>/adcapi/v2.0/?authkey=<authkey>&action=<action>`

其中参数说明如下:

名称	含义
<code><protocol></code>	使用 API 的协议, HTTP 或者 HTTPS
<code><host></code>	目标设备的 IP 地址
<code><port></code>	目标设备的 IP 地址 HTTP/HTTPS 端口, 当为该协议默认端口 (HTTP:80, HTTPS:443) 可以省略
<code><authkey></code>	通过 login API 获取的 authkey
<code><action></code>	API 的 action 字段, 具体见各个 API

Method API 使用的 HTTP method

若未特殊说明,当无请求参数时使用 GET,有请求参数时使用 POST

文件上传

文件上传时,使用 form-data 的方式上传,此时如果有请求参数,需要在 URL 中携带参数,具体请查看对应 API

必选参数

执行 API 时,对于必选参数,必须明确指定该参数。

可选参数

执行 API 时,对于可选参数,不指定可选参数时,会使用缺省值。

当一个 API 全是可选参数时,至少需要指定其中一个参数。

响应内容

对于获取信息类的 API,包括所有以(.get/.list/.stats/.statis)结尾的 action 的 API,响应内容为标准 JSON 格式内容

具体内容请查看对于 API 说明

对于配置类的 API,包括所有以(.add/.edit/.del/.set/.apply)结尾的 action 的 API,响应内容为{"result":"success"},不再在每个 API 中进行描述

登出的 API(logout)的响应内容同配置类 API 的响应内容, 为{"result":"success"}

对于文件下载类的 API, 响应内容为文件内容 (application/octet-stream)

对于文件上传类的 API, 响应内容同配置类 API 的响应内容,为{"result":"success"}

其它所有未对响应内容做特殊说明的 API, 响应内容为{"result":"success"}

对于出错的情况, 响应内容为{"result":"error", "errcode":<errcode>, "errmsg":<errmsg>}

其中参数说明如下:

名称	含义
<errcode>	错误码, 为一个数字
<errmsg>	错误消息, 为一个字符串

登陆和登出

登陆

Action: login

请求参数:

名称	类型	范围	含义	必选	备注
username	字符串	长度 1-63	1 表示 VRRP 开启, 0 表示关闭	是	
password	字符串	长度 1-63	1 表示开启 AS_MODE, 0 表示关闭	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?action=login

Body: {"username":"admin","password":"admin"}

响应参数:

名称	类型	范围	含义
authkey	字符串		认证密钥, 后续其它所有 API 请求需要在 URL 中携带该值

响应举例:

```
{
  "authkey": "dbc121e55cc33c67911a99ce4829db"
}
```



Note

每次 login 会生成一个 authkey,设备能同时支持的 authkey 的数量是有限的,

因此在使用完成后需要使用 logout 使该 authkey 失效

登出

Action: logout

请求参数:无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=logout



Note

登出后该 authkey 不再有效.

系统

系统信息

获取设备信息

Action: **system.information.get**

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.information.get

响应参数:

名称	类型	范围	含义
hostname	字符串		主机名, 若为空字符串表示获取失败
product_model	字符串		产品型号, “N/A”表示获取失败
build_time	字符串		编译时间, “N/A”表示获取失败
software_version	字符串		软件版本, “N/A”表示获取失败
current_time	字符串		当前时间, “N/A”表示获取失败
save_time	字符串		上一次保存时间, “N/A”表示获取失败
running_time	字符串		本次运行时间, “N/A”表示获取失败
product_fc	整数	0-1	产品平台, 0 表示硬件设备, 1 表示虚拟设备
serial_number	字符串		设备序列号, “N/A”表示获取失败
api_version	字符串		API 版本号, “N/A”表示获取失败
mc_id	字符串		集中管理 ID, 空字符串表示没有连接集中管理, “N/A”表示获取失败
dp_cpu_usage	数组		业务 CPU 使用率, cpu 个数是数组的长度, 每一个 cpu 对应一组 cpu 使用率, 分别表示 cpu 在 1 秒内, 5 秒内和 60 秒内 cpu 平均使用率, 取值范围为 1-100, 比如值为 35, 表示使用率为 35%
mp_cpu_usage	数组		管理 CPU 使用率, cpu 个数是数组的长度, 每一个 cpu 对应一组 cpu 使用率, 分别表示 cpu 在 1 秒内, 5 秒内和 60 秒内 cpu 平均使用率, 取值范围为 1-100, 比如值为 35, 表示使用率为 35%
mem_free	整数	>=0	内容空闲数量, 单位 KB
mem_all	整数	>=0	内存总共数量, 单位 KB
cpu_temp	整数	1-100	CPU 温度, 单位℃
power_unit	字符串		电源单元统计, 表现为分数, 分母为电源总数, 分子为正常运行的电源个

			数
fan_count	字符串		风扇个数统计, 表现为分数, 分母为风扇总数, 分子为正常运行的风扇个数
disk_usage	字符串	>=0	硬盘使用率

响应举例:

```
{
  "hostname": "adc63",
  "product_model": "ADVIRT",
  "build_time": "10:08:00 CST Tue Jul 2 2019",
  "current_time": "18:09:56 CST Tue Jul 2 2019",
  "software_version": "V3.8R2B105",
  "running_time": "0 day, 7 hours, 23 minutes",
  "save_time": "10:44:57 CST Tue Jul 2 2019",
  "product_fc": 1,
  "api_version": "2.0",
  "serial_number": "8984a8ecce1ed15db1ae",
  "mc_id": "",
  "dp_cpu_usage": [ "2,2,1", "0,1,1" ],
  "mp_cpu_usage": [ "36,41,46" ],
  "mem_free": 1643933,
  "mem_all": 4049096
}
```

获取设备许可信息

Action: system.license.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.license.get

响应参数:

名称	类型	范围	含义
deadline	字符串		过期时间, 格式为 year-mon-day, "0"表示永久
device_id	字符串		设备 ID, 用于申请许可时提供
bandwidth	整数		许可带宽, 单位 Mb/s, 0 表示不限制带宽
license_exist	整数	0-1	是否有授权许可, 0 表示没有授权 license 1 表示授权 license
remain_days	整数	-1-2147483647	许可剩余天数, 单位天, -1 表示永久

响应举例:

```
{
  "deadline": "0",
  "device_id": "V6PkPdINbfkoP66.....",
  "bandwidth": 0,
```

```

    "license_exist": 1,
    "remain_days": -1
}

```

获取整机性能信息

Action: `system.performance.get`

请求参数:无

请求举例:

```

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.performance.get

```

响应参数:

名称	类型	范围	含义
throughput	整数	>=0	整机吞吐量, 单位 bps
curr_conn	整数	>=0	整机并发连接数,
all_conn_ps	整数	>=0	整机每秒新建连接数
l4_conn_ps	整数	>=0	整机每秒四层新建连接数
ssl_conn_ps	整数	>=0	整机每秒 SSL 新建连接数
nat_conn_ps	整数	>=0	整机每秒 NAT 新建连接数
l7_conn_ps	整数	>=0	整机每秒七层新建连接数
l7_trans_ps	整数	>=0	整机每秒七层新建事务数

响应举例:

```

{
  "throughput": 0,
  "curr_conn": 0,
  "all_conn_ps": 0,
  "l4_conn_ps": 0,
  "ssl_conn_ps": 0,
  "nat_conn_ps": 0,
  "l7_conn_ps": 0,
  "l7_trans_ps": 0
}

```

获取应用加速统计信息

Action: `slb.appacl.stat.get`

请求参数:无

请求举例:

```

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.appacl.stat.get

```

响应参数:

名称	类型	范围	含义
cache_hit_ratio	整数	0-100	缓存命中率, 单位%
compress_ratio	整数	0-100	压缩率, 单位%
connmulti_number	整数	>=0	连接复用数

响应举例:

```
{
  "cache_hit_ratio": 0,
  "compress_ratio": 0,
  "connmulti_number": 0
}
```

获取应用状态数量统计信息

Action: slb.app.stat.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.app.stat.get

响应参数:

名称	类型	范围	含义
node	对象		节点状态数量统计
normal	整数	>=0	节点正常数量
failure	整数	>=0	节点故障数量
disabled	整数	>=0	节点禁用数量
pool	对象		服务池状态数量统计
normal	整数	>=0	服务池正常数量
failure	整数	>=0	服务池故障数量
disabled	整数	>=0	服务池禁用数量

响应举例:

```
{
  "node": {
    "normal": 0,
    "failure": 0,
    "disabled": 0
  },
  "pool": {
    "normal": 0,
    "failure": 0,
    "disabled": 0
  }
}
```

获取设备物理接口分布信息

Action: system.interface.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.interface.get

响应参数:

名称	类型	范围	含义
slot_rows	整数	1-8	设备总槽位行数
slot_cols	整数	1-8	设备总槽位列数
slots	数组		由各槽位信息组成的槽位信息列表
interfaces	数组		由各接口信息组成的接口信息列表
slot_row	整数	1-8	槽位行序号
slot_col	整数	1-8	槽位列序号
port_row	整数	1-8	接口行序号
port_col	整数	1-8	接口列序号
port_rows	整数	1-8	接口总行数
port_cols	整数	1-8	接口总列数
port_num	整数	1-16	槽内接口号
all_port_num	整数	1-64	总排序接口号,仅物理接口有该状态
type	整数	0-2	接口类型,0:管理口,1:物理接口,2:串口
status	整数	0-2	接口状态,0:unknown,1:down,2:up
in_use	整数	0-1	接口使用中,仅管理口有该状态,0:未使用,1:使用中

响应举例:

```
{
  "slot_rows": 1,
  "slot_cols": 2,
  "slots": [ {
    "slot_row": 1,
    "slot_col": 1,
    "port_rows": 1,
    "port_cols": 6
  } ],
  "interfaces": [ {
    "port_num": 0,
    "slot_row": 1,
    "slot_col": 1,
    "port_row": 1,
    "port_col": 2,
    "status": 1,
    "all_port_num": 1,
    "type": 1
  } ]
}
```



```

}, {
  "port_num" : 1,
  "slot_row" : 1,
  "slot_col" : 1,
  "port_row" : 1,
  "port_col" : 3,
  "status" : 1,
  "all_port_num" : 2,
  "type" : 1
}, {
  "port_num" : 2,
  "slot_row" : 1,
  "slot_col" : 1,
  "port_row" : 1,
  "port_col" : 4,
  "status" : 1,
  "all_port_num" : 3,
  "type" : 1
}, {
  "port_num" : 3,
  "slot_row" : 1,
  "slot_col" : 1,
  "port_row" : 1,
  "port_col" : 5,
  "status" : 1,
  "all_port_num" : 4,
  "type" : 1
}, {
  "port_num" : 4,
  "slot_row" : 1,
  "slot_col" : 1,
  "port_row" : 1,
  "port_col" : 6,
  "status" : 1,
  "all_port_num" : 5,
  "type" : 1
}, {
  "port_num" : 1,
  "slot_row" : 1,
  "slot_col" : 1,
  "port_row" : 1,
  "port_col" : 1,
  "status" : 1,
  "in_use" : 1,
  "type" : 0
} ]

```

安全标识

获取登录安全标识

Action: system.security_banner.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.security_banner.get

响应参数:

名称	类型	范围	含义
login_banner	字符串	0-1019	登录前安全标识, 默认值 "Welcome To Login ADC System"
login_success_banner	字符串	0-1019	登录后安全标识, 默认值 "[type ? for help]"

响应举例:

```
{  
  "login_banner": "Welcome To Login ADC System",  
  "login_success_banner": "[type ? for help]"  
}
```

设置登录安全标识

Action: system.security_banner.set

请求参数:

名称	类型	范围	含义
login_banner	字符串	0-1019	登录前安全标识, 默认值 "Welcome To Login ADC System"
login_success_banner	字符串	0-1019	登录后安全标识, 默认值 "[type ? for help]"

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.security_banner.set

请求 body:

```
{  
  "login_banner": "Welcome To Login ADC System",  
  "login_success_banner": "[type ? for help]"  
}
```

}



1. 输入 “ ” 双引号须要用 “\” 转义字符，例： “\” ”

系统配置

DNS 配置设置

Action: system.dns.set

请求参数:

名称	类型	范围	含义	必选	备注
domain	字符串	长度 1-255	域名名称	否	未指定时表示不修改
primary_server	IPv4 地址		DNS 主服务器	否	未指定时表示不修改
sencondary_server	IPv4 地址		DNS 辅组服务器	否	未指定时表示不修改

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.dns.set

请求 body:

```
{
  "domain": "1234.com",
  "primary_server": "8.8.8.9",
  "secondary_server": "114.144.144.145"
}
```

DNS 配置获取

Action: system.dns.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.dns.get

响应参数:

名称	类型	范围	含义
domain	字符串	长度 1-255	域名名称
primary_server	IPv4 地址		DNS 主服务器
sencondary_server	IPv4 地址		DNS 辅组服务器

响应举例:

```
{
  "domain": "1234.com",
  "primary_server": "8.8.8.9",
  "secondary_server": "114.144.144.145"
}
```

}

设置系统主机名

Action: system.hostname.set

请求参数:

名称	类型	范围	含义	必选	备注
hostname	字符串	长度 1-31	主机名	是	支持字符 a-zA-Z0-9-

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.hostname.set

Body:

```
{  
  "hostname": "Horizon123"  
}
```

主机名 IPV4 地址获取

Action: system.host.ipv4.get

请求参数:

名称	类型	范围	含义	必选	备注
hosts 主机 IPV4 地址	字符串	长度 1-31	显示主机名解析 地址	否	IPV4 地址

请求举例:

GET
http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.host.ipv4.get

响应举例

```
{  
  "host_ipv4_address": "192.168.70.73"  
}
```

主机名 IPV4 地址设置

Action: system.host.ipv4.set

请求参数:

名称	类型	范围	含义	必选	备注
hosts 主机	字符串	长度 1-31	主机名指定解析	是	IPV4 地址

IPV4 地址			地址		
---------	--	--	----	--	--

请求举例:

POST
 http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.host.ipv4.set

Body:

```
{
  "host_ipv4_address": "192.168.70.73"
}
```

主机名 IPV6 地址获取

Action: system.host.ipv6.get

请求参数:

名称	类型	范围	含义	必选	备注
hosts 主机 IPV6 地址	字符串	长度 1-63	显示主机名解析地址	否	IPV6 地址

请求举例:

GET
 http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.host.ipv6.get

响应举例

```
{
  "host_ipv6_address": "2000::2000"
}
```

主机名 IPV6 地址设置

Action: system.host.ipv6.set

请求参数:

名称	类型	范围	含义	必选	备注
hosts 主机 IPV6 地址	字符串	长度 1-63	主机名指定解析地址	是	IPV6 地址

请求举例:

POST
 http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.host.ipv6.set

Body:

```
{
  "host_ipv6_address": "2000::2000"
}
```

Hosts 列表获取

Action: system.hosts.list

请求参数:

名称	类型	范围	含义	必选	备注
hosts 列表获取	字符串		显示所有 hosts	否	

请求举例:

GET
http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.hosts.list

响应举例

```
[
  {
    "hostname": "10.3.70.242-slb-ve-119",
    "host_ip_address": "2000:2000:2000:2000:2000:2000:2000:119"
  },
  {
    "hostname": "10.3.70.242-slb-ve-118",
    "host_ip_address": "192.168.0.118"
  },
  {
    "hostname": "10.3.70.242-slb-ve-1",
    "host_ip_address": "2000:2000:2000:2000:2000:2000:2000:1"
  }
]
```

Hosts 增加

Action: system.hosts.add

请求参数:

名称	类型	范围	含义	必选	备注
主机名称	字符串	1-31	指定 hosts 域名	是	
地址	字符串	1-63	Ipv4/ipv6 地址	是	

请求举例:

POST
http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.hosts.add
Body:

```
{
  "hostname": "10.3.70.242-slb-ve-1",
  "host_ip_address": "100.100.100.100"
}
```

```
}
```

Hosts 删除

Action: system.hosts.del

请求参数:

名称	类型	范围	含义	必选	备注
主机名称	字符串	1-31	指定 hosts 域名	是	
地址	字符串	1-63	Ipv4/ipv6 地址	是	

请求举例:

POST

http://192.168.70.73/adcapiv2/?authkey=e4f2ee52d8a18dc64e255278c47669&action=system.hosts.del

Body:

```
{
    "hostname": "10.3.70.242-slb-ve-1",
    "host_ip_address": "100.100.100.100"
}
```

获取系统 banner 信息

Action: system.banner.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.banner.get

响应参数:

名称	类型	范围	含义
hostname	字符串	长度 1-31	系统主机名
username	字符串		当前会话用户名
clientip	IPv4 地址		当前会话客户端 IPv4 地址
mgmtip	IPv4 地址		当前会话系统管理 IPv4 地址
ha	字符串		主备状态, standby 表示备机, active 表示主机, n/a 表示没有开启 VRRP
confdirty	整数	0, 1	是否存在未保存的配置, 0:不存在, 1:存在

响应举例:

```
{
    "hostname": "107",
    "username": "admin",
    "clientip": "192.168.50.107",
    "mgmtip": "192.168.70.73",
    "ha": "n/a",
    "confdirty": 1
}
```

}

时间配置设置

Action: system.time.set

请求参数:

名称	类型	范围	含义	必选	备注
date	字符串		日期,格式 月/日/年	否	未指定时表示不修改
time	字符串		时间,格式 HH:MM:SS	否	未指定时表示不修改
tzzone	字符串		时区,支持如下时区: "(GMT-11:00)Midway Island, Samoa" "(GMT-10:00)Hawaii" "(GMT-09:00)Alaska" "(GMT-08:00)Pacific Time - Tijuana" "(GMT-08:00)Pacific Time(US & Canada)" "(GMT-08:00)Pacific Time - west British Columbia" "(GMT-07:00)Arizona" "(GMT-07:00)Mountain Time(US & Canada)" "(GMT-06:00)Central Time(US & Canada)" "(GMT-06:00)Mexico City" "(GMT-06:00)Saskatchewan" "(GMT-06:00)Central America" "(GMT-05:00)Eastern Time(US & Canada)" "(GMT-05:00)Indiana(East)" "(GMT-05:00)Eastern Time - Ontario & Quebec - most locations" "(GMT-05:00)Eastern Time" "(GMT-05:00)Eastern Time - Toronto, Ontario" "(GMT-04:00)Caracas, La Paz" "(GMT-04:00)Atlantic Time(Canada)" "(GMT-04:00)Santiago" "(GMT-03:30)Newfoundland" "(GMT-03:00)Buenos Aires, Georgetown" "(GMT-03:00)Greenland" "(GMT-02:00)Mid-Atlantic" "(GMT-01:00)Azores" "(GMT-01:00)Cape Verde Is." "(GMT)Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London" "(GMT+01:00)West Central Africa" "(GMT+01:00)Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna" "(GMT+01:00)Belgrade, Bratislava, Budapest, Ljubljana, Prague"	否	未指定时表示不修改

		<p>"(GMT+01:00)Brussels, Copenhagen, Madrid, Paris"</p> <p>"(GMT+01:00)Sarajevo, Skopje, Sofija, Vilnius, Warsaw, Zagreb"</p> <p>"(GMT+02:00)Bucharest"</p> <p>"(GMT+02:00)Cairo"</p> <p>"(GMT+02:00)Athens, Istanbul, Minsk"</p> <p>"(GMT+02:00)Harare, Pretoria"</p> <p>"(GMT+02:00)Jerusalem"</p> <p>"(GMT+02:00)Helsinki, Riga, Tallinn"</p> <p>"(GMT+03:00)Nairobi"</p> <p>"(GMT+03:00)Baghdad"</p> <p>"(GMT+03:00)Kuwait, Riyadh"</p> <p>"(GMT+03:00)Moscow, St.Petersburg, Volgograd"</p> <p>"(GMT+03:30)Tehran"</p> <p>"(GMT+04:00)Baku, Tbilisi, Yerevan"</p> <p>"(GMT+04:00)Abu Dhabi, Muscat"</p> <p>"(GMT+04:30)Kabul"</p> <p>"(GMT+05:00)Islamabad, Karachi, Tashkent"</p> <p>"(GMT+05:00)Ekaterinburg"</p> <p>"(GMT+05:30)Calcutta, Chennai, Mumbai, New Delhi"</p> <p>"(GMT+05:45)Kathmandu"</p> <p>"(GMT+06:00)Almaty, Novosibirsk"</p> <p>"(GMT+06:00)Astana, Dhaka"</p> <p>"(GMT+06:00)Sri Jayawardenepura"</p> <p>"(GMT+06:30)Rangoon", "MMT" },</p> <p>"(GMT+07:00)Bangkok, Hanoi, Jakarta"</p> <p>"(GMT+07:00)Krasnoyarsk"</p> <p>"(GMT+08:00)Irkutsk, Ulaan Bataar"</p> <p>"(GMT+08:00)Kuala Lumpur, Singapore"</p> <p>"(GMT+08:00)Beijing, Chongqing, Hong Kong, Urumqi"</p> <p>"(GMT+08:00)Taipei"</p> <p>"(GMT+08:00)Perth"</p> <p>"(GMT+09:00)Seoul"</p> <p>"(GMT+09:00)Osaka, Sapporo, Tokyo"</p> <p>"(GMT+09:00)Yakutsk"</p> <p>"(GMT+09:30)Adelaide"</p> <p>"(GMT+09:30)Darwin"</p> <p>"(GMT+10:00)Hobart"</p> <p>"(GMT+10:00)Brisbane"</p> <p>"(GMT+10:00)Vladivostok"</p> <p>"(GMT+10:00)Canberra, Melbourne, Sydney"</p> <p>"(GMT+10:00)Guam, Port Moresby"</p> <p>"(GMT+11:00)Magadan, Solomon., New Caledonia"</p>	
--	--	--	--

			"(GMT+12:00)Auckland, Wellington"		
			"(GMT+12:00)Fiji, Kamchatka, Marshall Is."		
			"(GMT+12:00)Eniwetok, Kwajalein"		
			"(GMT+13:00)Nuku'alofa"		

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.time.set

Body:

```
{
  "date" : "07/20/2019",
  "time" : "13:28:35",
  "tzone" : "(GMT+08:00)Beijing, Chongqing, Hong Kong, Urumqi"
}
```

时间配置获取

Action: **system.time.get**

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.time.get

响应参数:

名称	类型	范围	含义
date	字符串		日期
time	字符串		时间
tzone	字符串		时区

响应举例:

```
{
  "date" : "07/20/2019",
  "time" : "13:28:35",
  "tzone" : "(GMT+08:00)Beijing, Chongqing, Hong Kong, Urumqi"
}
```

网络管理 SNMP

添加 SNMP 团体字

Action: **snmp.comm.item.add**

请求参数:

名称	类型	范围	含义	必选	备注
community	字符串	长度 1-31	团体字	是	
host	字符串	长度 1-127	允许使用该团体字的远端主机, 格式为 IPv4/v6 地址或主机名	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.comm.item.add

请求 body:

```
{
  "community": "public",
  "host": "192.168.1.100"
}
```

获取 SNMP 团体字列表

Action: snmp.comm.item.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.comm.item.list

响应参数:

名称	类型	范围	含义
community	字符串	长度 1-31	团体字
host	字符串	长度 1-127	允许使用该团体字的远端主机, 格式为 IPv4/v6 地址或主机名

响应举例:

```
[{
  "community": "public",
  "host": "192.168.1.100"
}]
```

删除 SNMP 团体字

Action: snmp.comm.item.del

请求参数:

名称	类型	范围	含义	必选	备注
community	字符串	长度 1-31	团体字	是	
host	字符串	长度 1-127	允许使用该团体字的远端主机, 格式为 IPv4/v6 地址或主机名	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.comm.item.del

请求 body:

```
{
  "community": "public",
  "host": "192.168.1.100"
}
```

设置 SNMP 服务配置

Action: snmp.server.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0, 1	是否开启 SNMP 服务	否	0 不开启, 1 开启, 缺省值: 不修改
port	整数	1-65535	SNMP 服务端口	否	缺省值: 不修改
contact	字符串	长度 0-63	联系方式	否	缺省值: 不修改
location	字符串	长度 0-63	位置信息	否	缺省值: 不修改

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.server.set

请求 body:

```
{
  "status":1,
  "port":161,
  "contact": "",
  "location": ""
}
```

获取 SNMP 服务配置

Action: snmp.server.get

请求参数: 无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.server.get

响应参数:

名称	类型	范围	含义
status	整数	0, 1	SNMP 服务是否开启, 0 未开启, 1 开启
port	整数	1-65535	SNMP 服务端口
contact	字符串	长度 0-127	联系方式
location	字符串	长度 0-127	位置信息
sys_name	字符串	长度 1-31	主机名

响应举例:

```
{
  "status":1,
  "port":161,
  "contact": "",
  "sys_name": "107",
  "location": ""
}
```

添加 SNMP TRAP

Action: snmp.trap.item.add

请求参数:

名称	类型	范围	含义	必选	备注
community	字符串	长度 1-31	发送 TRAP 使用的团体字	是	
host	字符串	长度 1-127	发送 TRAP 的目标主机, 格式为 IPv4/v6 地址或主机名	是	
port	整数	1-65535	发送 TRAP 的目标端口	否	缺省值:162
version	字符串	长度 1-3	发送 TRAP 消息的版本, 支持 v1 和 v2c	否	缺省值:v1

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.trap.item.add
请求 body:

```
{  
  "community": "public",  
  "host": "192.168.1.100",  
  "port": 162,  
  "version": "v2c"  
}
```

获取 SNMP TRAP 列表

Action: snmp.trap.item.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.trap.item.list

响应参数:

名称	类型	范围	含义
community	字符串	长度 1-31	发送 TRAP 使用的团体字
host	字符串	长度 1-127	发送 TRAP 的目标主机, 格式为 IPv4/v6 地址或主机名
port	整数	1-65535	发送 TRAP 的目标端口
version	字符串	长度 1-3	发送 TRAP 消息的版本, 支持 v1 和 v2c

响应举例:

```
[{  
  "community": "public",  
  "host": "192.168.1.100",  
  "port": 162,  
  "version": "v2c"  
}]
```

删除 SNMP TRAP

Action: snmp.trap.item.del

请求参数:

名称	类型	范围	含义	必选	备注
community	字符串	长度 1-31	发送 TRAP 使用的团体字	是	
host	字符串	长度 1-127	允许使用该 TRAP 的远端主机, 格式为 IPv4/v6 地址或主机名	是	
version	字符串	长度 1-3	发送 TRAP 消息的版本, 支持 v1 和 v2c	否	缺省值:v1

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.trap.item.del

请求 body:

```
{  
  "community": "public",  
  "host": "192.168.1.100",  
  "version": "v2c"  
}
```

设置 SNMP TRAP 配置

Action: snmp.trap.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0, 1	是否开启 SNMP TRAP 服务	否	0 不开启, 1 开启, 缺省值: 不修改

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.trap.set

请求 body:

```
{"status":1}
```

获取 SNMP TRAP 配置

Action: snmp.trap.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=snmp.trap.get

响应参数:

名称	类型	范围	含义
status	整数	0, 1	SNMP TRAP 服务是否开启, 0 未开启, 1 开启

响应举例:

```
{"status":1}
```

用户管理

添加用户

Action: admin.user.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-31	用户名	是	
password	字符串	长度 1-63	密码	是	若开启密码复杂度, 需要符合密码复杂度
privilege	整数	1, 2	用户权限	否	1: 只读权限, 2: 读写权限, 缺省值: 1
trust_host	IPv4 地址		信任地址	否	缺省值: 0.0.0.0
trust_acl	整数	1-199	信任 acl ID 号	否	缺省值: 0
trust_ipv6_acl	字符串	长度 1-191	信任 ipv6-acl 名称	否	缺省为空
trust_mask	掩码		信任掩码	否	缺省值: 0.0.0.0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.add

举例 1: 添加只读用户 readuser1 请求 body:

```
{
  "name": "readuser1",
  "password": "pWd@user1"
}
```

举例 2: 添加读写用户 admin2, 但是只能在主机 192.168.70.100 使用

请求 body:

```
{
  "name": "admin2",
  "password": "pWd@admin2",
  "privilege": 2,
  "trust_host": "192.168.70.100",
  "trust_mask": "255.255.255.255"
}
```

举例 3: 添加读写用户 admin3, 但是只能在 ipv4/v6 acl 地址过滤使用

请求 body:

```
{
  "name": "admin3",
  "password": "pWd@admin2",
  "privilege": 2,
  "trust_host": "0.0.0.0",
  "trust_mask": "0.0.0.0",
  "trust_acl": 2,
  "trust_ipv6_acl": "123"
}
```



1. 默认存在用户 admin,不能添加
2. Password 需要符合设置的密码复杂度和最小长度
3. 用户权限为只读时,只能获取配置和状态,不能修改
4. 只有用户信任地址/掩码范围内的主机可以使用该用户登陆

获取用户列表

Action: admin.user.list

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-31	用户名
privilege	整数	1, 2	用户权限, 1:只读权限, 2:读写权限
trust_host	IPv4 地址		信任地址
trust_mask	掩码		信任掩码
trust_acl	整数	1-199	信任 acl ID 号
trust_ipv6_acl	字符串	长度 1-191	信任 ipv6-acl 名称
locked	整数	0-1	用户是否被锁定, 1 已经锁定, 0, 未锁定

响应举例:

```
[
  {
    "name": "admin",
    "locked": 0,
    "trust_host": "0.0.0.0",
    "trust_mask": "0.0.0.0",
    "trust_acl": 0,
    "trust_ipv6_acl": "",
    "privilege": 2
  },
  {
    "name": "horizon",
    "locked": 0,
    "trust_host": "0.0.0.0",
    "trust_mask": "0.0.0.0",
    "trust_acl": 0,
    "trust_ipv6_acl": "",
    "privilege": 1
  }
]
```



```

    },
    {
        "name": "admin3",
        "locked": 0,
        "trust_host": "0.0.0.0",
        "trust_mask": "0.0.0.0",
        "trust_acl": 2,
        "trust_ipv6_acl": "123",
        "privilege": 1
    }
]

```

获取指定用户

Action: admin.user.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-31	用户名	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.get

请求 body:

```

{
    "name": "admin3"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-31	用户名
privilege	整数	1,2	用户权限, 1:只读权限, 2:读写权限
trust_host	IPv4 地址		信任地址
trust_mask	掩码		信任掩码
trust_acl	整数	1-199	信任 acl ID 号
trust_ipv6_acl	字符串	长度 1-191	信任 ipv6-acl 名称
locked	整数	0-1	用户是否被锁定, 1 已经锁定, 0, 未锁定

响应举例:

```

{
    "name": "admin3",
    "locked": 0,
    "trust_host": "0.0.0.0",
    "trust_mask": "0.0.0.0",
    "trust_acl": 2,
    "trust_ipv6_acl": "123",
    "privilege": 1
}

```

编辑指定用户

Action: admin.user.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-31	用户名	是	
password	字符串	长度 1-63	密码	否	缺省值:不修改
privilege	整数	1, 2	用户权限	否	1:只读权限, 2:读写权限, 缺省值: 不修改
trust_host	IPv4 地址		信任地址	否	缺省值: 0.0.0.0
trust_mask	掩码		信任掩码	否	缺省值: 0.0.0.0
trust_acl	整数	1-199	信任 acl ID 号	否	缺省值: 0
trust_ipv6_acl	字符串	长度 1-191	信任 ipv6-acl 名称	否	缺省为空

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.edit

请求 body:

```
{  
  "name": "admin2"  
}
```

该举例会编辑用户 admin2, 修改其信任地址和掩码为缺省值 0.0.0.0/0.0.0.0



2. 默认存在用户 admin 的权限不能修改

删除指定用户

Action: admin.user.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-31	用户名	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.del

请求 body:

```
{  
  "name": "admin2"  
}
```



1.默认存在用户 admin 不能删除

2.在线用户不能删除

设置用户锁定和密码配置

Action: admin.cfg.set

请求参数:

名称	类型	范围	含义	必选	备注
pwdtimeout	整数	0-129600	密码有效期	是	单位分钟,0表示永不过期,系统默认值0
pwdlength	整数	1-63	密码最小长度	是	系统默认值1
pwdcplex	整数	1-4	密码复杂度	是	表示密码至少需要大写字母\小写字母\数字\特殊字符4种中的几种字符 系统默认值1
pwdcpf1	整数	0,1	用户首次登陆需要修改密码	是	0表示不需要,1表示需要 系统默认值0
duration	整数	1-1440	锁定时间	是	单位秒,系统默认值10
resettime	整数	1-1440	检查间隔	是	单位秒,系统默认值10
threshold	整数	1-10	失败次数	是	系统默认值5
enable	整数	0,1	是否开启用户锁定功能	是	1表示开启,0表示关闭,系统默认值0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.cfg.set

请求 body:

```
{  
  "pwdtimeout":0,  
  "pwdlength":1,  
  "pwdcplex":1,  
  "pwdcpf1":0,  
  "duration":10,  
  "resettime":10,  
  "threshold":5,  
  "enable":1  
}
```



Note

- 1.开启用户锁定功能之后,同一个用户在 检查间隔(resettime)时间内,登陆失败次数达到设定失败次数(threshold)时,将会锁定该用户,使其不能登陆,锁定时间为(duration)。
- 2.当锁定时间倒计时结束时,用户会自动解锁
- 3.在锁定时间未结束时,可以通过用户解锁功能手动解除锁定的用户

获取用户锁定配置和密码配置

Action: admin.cfg.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.cfg.get

响应参数:

名称	类型	范围	含义
pwdtimeout	整数	0-129600	密码有效期
pwdlength	整数	1-63	密码最小长度
pwdcplex	整数	1-4	密码复杂度
pwdcpfl	整数	0, 1	用户首次登陆需要修改密码
duration	整数	1-1440	锁定时间
resettime	整数	1-1440	检查间隔
threshold	整数	1-10	失败次数
enable	整数	0, 1	是否开启用户锁定功能

响应举例:

```
{
  "pwdtimeout":0,
  "pwdlength":1,
  "pwdcplex":1,
  "pwdcpfl":0,
  "duration":10,
  "resettime":10,
  "threshold":5,
  "enable":1
}
```

解锁锁定用户

Action: admin.user.unlock

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-31	要解锁用户的用户名	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.user.unlock

请求 body:

```
{
  "name":"admin2"
}
```

修改当前用户密码

Action: system.password.set

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

old_password	字符串	长度 1-63	当前密码	是	
password	字符串	长度 1-63	新的密码	是	

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.password.set

请求 body:

```
{
  "old_password": "admin",
  "password": "password"
}
```

获取管理员状态

Action: admin.disable.get

请求参数: 无

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.disable.get

响应参数:

名称	类型	范围	含义
enable	整数	0, 1	0: 使用 admin 用户, 1: 禁用 admin 用户
name	字符串	长度 1-31	用户名
password	字符串	长度 1-63	密码

响应举例:

```
{
  "enable": 1,
  "name": "123",
  "password": "12345"
}
```

禁用管理员用户

Action: admin.disable.edit

请求参数:

名称	类型	范围	含义	必选	备注
enable	整数	0, 1	是否禁用	是	0: 使用 admin 用户, 1: 禁用 admin 用户
name	字符串	长度 1-31	用户名	是	
password	字符串	长度 1-63	密码	是	缺省值: 不修改

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=admin.disable.edit

请求 body:

```
{
  "enable": 1,
```

```

    "name": "123",
    "password": "12345"
}

```

客户端锁定

设置客户端锁定配置

Action: client.locked.set

请求参数:

名称	类型	范围	含义	必选	备注
interval	整数	30-180	检查间隔, 单位秒	是	系统默认值 60
maxnum	整数	3-16	失败次数,	是	系统默认值 8
locktime	整数	60-86400	锁定时间, 单位秒	是	系统默认值 300
enable	整数	0, 1	是否开启客户端锁定功能	是	0 表示关闭, 1 表示开启, 系统默认值 0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=client.locked.set
 请求 body:

```

{
  "interval": "180",
  "maxnum": 5,
  "locktime": "180",
  "enable": 1
}

```



Note

1. 开启客户端锁定功能之后, 同一个客户端在 检查间隔(interval)时间内, 登陆失败次数达到设定失败次数(maxnum)时, 将会锁定该客户端, 使其不能登陆, 锁定时间为(locktime)。
2. 当锁定时间倒计时结束时, 客户端会自动解锁
3. 在锁定时间未结束时, 可以通过客户端解锁功能手动解除锁定的客户端

获取客户端锁定配置

Action: client.locked.get

请求参数: 无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=client.locked.get

响应参数:

名称	类型	范围	含义
----	----	----	----

interval	整数	30-180	检查间隔, 单位秒
maxnum	整数	3-16	失败次数,
locktime	整数	60-86400	锁定时间, 单位秒
enable	整数	0, 1	客户端锁定功能是否开启 0 表示关闭, 1 表示开启,

响应举例:

```
{
  "interval": "180",
  "maxnum": 5,
  "locktime": "180",
  "enable": 1
}
```

获取锁定客户端列表

Action: client.locked.list

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=client.locked.list

响应参数:

名称	类型	范围	含义
srcip	IPv4/v6 地址		被锁定的客户端的 IPv4/v6 地址
remain	整数	0-86400	被锁定的客户端的剩余锁定时间,

响应举例:

```
[[
  {"srcip": "192.168.1.5",
    "remain": 105},
  {"srcip": "192.168.1.6",
    "remain": 100}
]]
```

解锁锁定客户端

Action: client.locked.unlock

请求参数:

名称	类型	范围	含义	必选	备注
srcip	IPv4/v6 地址		被锁定的客户端的 IPv4/v6 地址	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=client.locked.unlock

请求 body:

```
{  
"srcip": "192.168.1.5"  
}
```

AAA 认证

AAA 全局配置设置

Action: **aaa.general.set**

请求参数:

名称	类型	范围	含义	必选	备注
local_disabled	整数	0-1	本地认证去使能	否	1: 关闭本地认证 0: 开启本地认证 系统默认值 0, 缺省值 0
auth_order	整数	1-5	认证顺序	否	1: Local 本地认证 2: Local & Radius 本地认证/Radius 认证 3: Local & Tacacs+ 本地认证/TACACS+ 认证 4: Radius & Local Radius 认证/本地认证 5: Tacacs+ & Local TACACS+ 认证/ Radius 认证 系统默认值 1, 缺省值 1
auth_order_console	整数	0-5	console 认证顺序	否	0: None 不认证 1: Local 本地认证 2: Local & Radius 本地认证/Radius 认证 3: Local & Tacacs+ 本地认证/TACACS+ 认证 4: Radius & Local Radius 认证/本地认证 5: Tacacs+ & Local TACACS+ 认证/ Radius 认证 系统默认值 0, 缺省值 0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.general.set

请求 body:

```
{  
  "local_disabled": 0,  
  "auth_order": 1,  
  "auth_order_console": 1  
}
```


AAA 全局配置获取

Action: aaa.general.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.general.get

响应参数:

名称	类型	范围	含义
local_disabled	整数	0-1	本地认证去使能, 1: 关闭本地认证 0: 开启本地认证
auth_order	整数	1-5	认证顺序: 1: Local 本地认证 2: Local & Radius 本地认证/Radius 认证 3: Local & Tacacs+ 本地认证/TACACS+ 认证 4: Radius & Local Radius 认证/本地认证 5: Tacacs+ & Local TACACS+ 认证/ Radius 认证
auth_order_console	整数	0-5	console 认证顺序, 0: None 不认证 1: Local 本地认证 2: Local & Radius 本地认证/Radius 认证 3: Local & Tacacs+ 本地认证/TACACS+ 认证 4: Radius & Local Radius 认证/本地认证 5: Tacacs+ & Local TACACS+ 认证/ Radius 认证

响应举例:

```
{
  "local_disabled":0,
  "auth_order":1,
  "auth_order_console":0
}
```

Radius 认证配置设置

Action: aaa.radius.set

请求参数:

名称	类型	范围	含义	必选	备注
server_status	整数	0-1	主服务器状态,	否	1: 启用, 0: 不启用 系统默认值 0, 缺省值 0
server_hostname	字符串	长度 1-63	主服务器主机名 可以是主机名也可以是 IPv4/v6 地址	否	系统默认值 hostname 缺省值 hostname 当 server_status 为 1 时该 参数为必选参数
server_secret	整数	长度	主服务器共享密钥	否	系统默认值 secret

		1-127			缺省值 secret 当 server_status 为 1 时该参数为必选参数
server_authentication	整数	0-65535	主服务器认证端口	否	缺省值 1812, 系统默认值 1812
server_account	整数	0-65535	主服务器计费端口	否	缺省值 1813, 系统默认值 1813
server_retransmit	整数	0-5	主服务器重传次数	否	缺省值 3, 系统默认值 3
server_timeout	整数	1-15	主服务器超时时间,单位秒	否	缺省值 3, 系统默认值 3
server_status2	整数	0-1	从服务器状态,	否	1: 启用, 0: 不启用 系统默认值 0, 缺省值 0
server_hostname2	字符串	长度 1-63	从服务器主机名 可以是主机名也可以是 IPv4/v6 地址	否	系统默认值 hostname 缺省值 hostname 当 server_status2 为 1 时该参数为必选参数
server_secret2	整数	长度 1-127	从服务器共享密钥	否	系统默认值 secret 缺省值 secret 当 server_status2 为 1 时该参数为必选参数
server_authentication2	整数	0-65535	从服务器认证端口	否	缺省值 1812, 系统默认值 1812
server_account2	整数	0-65535	从服务器计费端口	否	缺省值 1813, 系统默认值 1813
server_retransmit2	整数	0-5	从服务器重传次数	否	缺省值 3, 系统默认值 3
server_timeout2	整数	1-15	从服务器超时时间,单位秒	否	缺省值 3, 系统默认值 3
def_priv_enabled	整数	0,1	认证成功后提供默认访问权限	否	系统默认值 0, 设置为 1 时, Radius 认证通过的用户具有读写权限, 设置为 0 时, Radius 认证通过的用户具有只读权限

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.radius.set

请求 body:

```
{
  "server_status": 1,
  "server_hostname": "192.168.70.250",
  "server_secret": "radius"
}
```

Radius 认证配置获取

Action: aaa.radius.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.radius.get

响应参数:

名称	类型	范围	含义
----	----	----	----

server_status	整数	0-1	主服务器状态, 1: 启用, 0: 不启用
server_hostname	字符串	长度 1-63	主服务器主机名 可以是主机名也可以是 IPv4 地址
server_secret	整数	长度 1-127	主服务器共享密钥
server_authentication	整数	0-65535	主服务器认证端口
server_account	整数	0-65535	主服务器计费端口
server_retransmit	整数	0-5	主服务器重传次数
server_timeout	整数	1-15	主服务器超时时间,单位秒
server_status2	整数	0-1	从服务器状态, 1: 启用, 0: 不启用
server_hostname2	字符串	长度 1-63	从服务器主机名 可以是主机名也可以是 IPv4 地址
server_secret2	整数	长度 1-127	从服务器共享密钥
server_authentication2	整数	0-65535	从服务器认证端口
server_account2	整数	0-65535	从服务器计费端口
server_retransmit2	整数	0-5	从服务器重传次数
server_timeout2	整数	1-15	从服务器超时时间,单位秒
def_priv_enabled	整数	0,1	认证成功后提供默认访问权限, 设置为 1 时, Radius 认证通过的用户具有读写权限, 设置为 0 时, Radius 认证通过的用户具有只读权限

响应举例:

```
{
  "server_status":1,
  "server_hostname":"192.168.70.250",
  "server_secret":"radius",
  "server_authentication":1812,
  "server_account":1813,
  "server_retransmit":3,
  "server_timeout":3,
  "server_status2":0,
  "server_hostname2":"hostname",
  "server_secret2":"secret",
  "server_authentication2":1812,
  "server_account2":1813,
  "server_retransmit2":3,
  "server_timeout2":3,
  "def_priv_enabled":0
}
```

TACACS+认证配置设置

Action: aaa.tacacs.set

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

server_status	整数	0-1	主服务器状态,	否	1: 启用,0: 不启用 系统默认值 0, 缺省值 0
server_hostname	字符串	长度 1-255	主服务器主机名 可以是主机名也可以是 IPv4/v6 地址	否	系统默认值 hostname 缺省值 hostname 当 server_status 为 1 时该 参数为必选参数
server_secret	整数	长度 1-127	主服务器共享密钥	否	系统默认值 secret 缺省值 secret 当 server_status 为 1 时该 参数为必选参数
server_authentication	整数	0-6553 5	主服务器认证端口	否	缺省值 49, 系统默认值 49
server_timeout	整数	1-12	主服务器超时时间,单位秒	否	缺省值 12, 系统默认值 12
server_status2	整数	0-1	从服务器状态,	否	1: 启用,0: 不启用 系统默认值 0, 缺省值 0
server_hostname2	字符串	长度 1-255	从服务器主机名 可以是主机名也可以是 IPv4/v6 地址	否	系统默认值 hostname 缺省值 hostname 当 server_status2 为 1 时该 参数为必选参数
server_secret2	整数	长度 1-127	从服务器共享密钥	否	系统默认值 secret 缺省值 secret 当 server_status2 为 1 时该 参数为必选参数
server_authentication2	整数	0-6553 5	从服务器认证端口	否	缺省值 49, 系统默认值 49
server_timeout2	整数	1-12	从服务器超时时间,单位秒	否	缺省值 12, 系统默认值 12

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.tacacs.set

请求 body:

```
{
  "server_status": 1,
  "server_hostname": "192.168.70.250",
  "server_secret": "tacacs"
}
```

TACACS+认证配置获取

Action: aaa.tacacs.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=aaa.tacacs.get

响应参数:

名称	类型	范围	含义
----	----	----	----

server_status	整数	0-1	主服务器状态, 1: 启用, 0: 不启用
server_hostname	字符串	长度 1-255	主服务器主机名 可以是主机名也可以是 IPv4 地址
server_secret	整数	长度 1-127	主服务器共享密钥
server_authentication	整数	0-65535	主服务器认证端口
server_account	整数	0-65535	主服务器计费端口
server_retransmit	整数	0-5	主服务器重传次数
server_timeout	整数	1-12	主服务器超时时间,单位秒
server_status2	整数	0-1	从服务器状态, 1: 启用, 0: 不启用
server_hostname2	字符串	长度 1-255	从服务器主机名 可以是主机名也可以是 IPv4 地址
server_secret2	整数	长度 1-127	从服务器共享密钥
server_authentication2	整数	0-65535	从服务器认证端口
server_account2	整数	0-65535	从服务器计费端口
server_retransmit2	整数	0-5	从服务器重传次数
server_timeout2	整数	1-12	从服务器超时时间,单位秒
def_priv_enabled	整数	0,1	认证成功后提供默认访问权限, 设置为 1 时, Radius 认证通过的用户具有读写权限, 设置为 0 时, Radius 认证通过的用户具有只读权限

响应举例:

```
{
  "server_status":1,
  "server_hostname":"192. 168. 70. 250",
  "server_secret":"tacacs",
  "server_authentication":49,
  "server_timeout":12,
  "server_status2":0,
  "server_hostname2":"hostname",
  "server_secret2":"secret",
  "server_authentication2":49,
  "server_timeout2":12
}
```

许可管理

许可信息获取

Action: license.info.get

请求参数:无

请求举例:

GET http://192. 168. 70. 73/adcap/v2. 0/?authkey=dbc121e55cc33c67911a99ce4829db&action=license. info. get

响应参数:

名称	类型	范围	含义
deadline	字符串		过期时间, 格式为 year-mon-day, “0”表示永久
device_id	字符串		设备 ID, 用于申请许可时提供
bandwidth	整数		许可带宽, 单位 Mb/s, 0 表示不限制带宽
lic_exist	整数	0-1	是否有授权许可, 0 表示没有授权 license 1 表示授权 license
rdays	整数	-1-2147483647	许可剩余天数, 单位天, -1 表示永久

响应举例:

```
{  
  "deadline": "0",  
  "device_id": "V6PkPdINbfkoP66.....",  
  "bandwidth": 0,  
  "lic_exist": 1,  
  "rdays": -1  
}
```



Note

此 API 与 API system.license.get 功能相同

许可文件上传

Action: license.file.upload

请求参数: 无

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=license.file.upload



Note

此 API 需要使用 form-data 的方式上传许可文件

NTP 配置

添加 NTP 配置

Action: system.ntp.add

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度仅指主机名
status	整数	0-1	NTP 开启状态, 0 表示关闭, 1 表示开启, 默认开启	否	
prefer	整数	0-1	NTP 优先选取, 0 表示关闭, 1 表示开启, 默认关闭	否	
minpoll	整数	4-14	NTP 同步时间指数最小值, 默认 6	否	
maxpoll	整数	4-14	NTP 同步时间指数最大值, 默认 10	否	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ntp.add

请求 body:

```
{
  "server": "12.2.2.2",
  "status": 0,
  "prefer": 1,
  "minpoll": 5,
  "maxpoll": 9
}
```

获取 NTP 配置列表

Action: system.ntp.list

请求参数: 无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ntp.list

响应参数:

名称	类型	范围	含义
server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址, 长度指主机名
status	整数	0-1	NTP 开启状态, 0 表示关闭, 1 表示开启
prefer	整数	0-1	NTP 优先选取, 0 表示关闭, 1 表示开启
minpoll	整数	4-14	NTP 同步时间指数最小值
maxpoll	整数	4-14	NTP 同步时间指数最大值

响应举例:

```
[{
  "server": "12.2.2.2",
  "status": 0,
  "prefer": 1,
  "minpoll": 7,
  "maxpoll": 10
}, {
  "server": "192.168.70.240",
  "status": 1,
  "prefer": 0,
  "minpoll": 6,
  "maxpoll": 10
}]
```

获取指定 NTP 配置

Action: system.ntp.get

请求参数:

名称	类型	范围	含义	必选	备注
server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ntp.get

请求 body:

```
{  
  "server": "12.2.2.2"  
}
```

响应参数:

名称	类型	范围	含义
server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址, 长度指主机名
status	整数	0-1	NTP 开启状态, 0 表示关闭, 1 表示开启
prefer	整数	0-1	NTP 优先选取, 0 表示关闭, 1 表示开启
minpoll	整数	4-14	NTP 同步时间指数最小值
maxpoll	整数	4-14	NTP 同步时间指数最大值

响应举例:

```
{  
  "server": "12.2.2.2",  
  "status": 0,  
  "prefer": 1,  
  "minpoll": 5,  
  "maxpoll": 9  
}
```

编辑 NTP 配置

Action: system.ntp.edit

请求参数:

名称	类型	范围	含义	必选	备注
server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
status	整数	0-1	NTP 开启状态, 0 表示关闭, 1 表示开启	否	
prefer	整数	0-1	NTP 优先选取, 0 表示关闭, 1 表示开启	否	
minpoll	整数	4-14	NTP 同步时间指数最小值, 默认 6	否	
maxpoll	整数	4-14	NTP 同步时间指数最大值, 默认 10	否	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ntp.edit

请求 body:

```
{  
  "server": "12.2.2.2",  
}
```



```

"status": 1,
"prefer": 1,
"minpoll": 5,
"maxpoll": 9
}

```

删除指定 NTP 配置

Action: system.ntp.del

请求参数:

名称	类型	范围	含义	必选	备注
server	字符串	长度 1-31	NTP 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ntp.del

请求 body:

```

{
  "server": "12.2.2.2"
}

```

WEB 会话

设置 WEB 会话超时时间

Action: system.web.set

请求参数:

名称	类型	范围	含义	必选	备注
timeout	整数	0-60	web 会话超时时间, 单位分钟, 其中 0 表示永不超时	是	系统默认 10 分钟

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.set

请求 body:

```

{
  "timeout":60
}

```



Note

设置 WEB 会话超时时间会同时设置 authkey 的超时时间

获取 WEB 会话超时时间

Action: system.web.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.get

响应参数:

名称	类型	范围	含义
timeout	整数	0-60	web 会话超时时间, 单位分钟, 其中 0 表示永不超时

响应举例:

```
{  
  "timeout":60  
}
```

设置 WEB 会话限制

Action: web.session.limit.set

请求参数:

名称	类型	范围	含义	必选	备注
global_limit	整数	1-128	全局 WEB 管理会话上限	是	系统默认 128
user_limit	整数	1-64	每用户 WEB 管理会话上限	是	系统默认 64

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=web.session.limit.set

请求 body:

```
{  
  "global_limit":20,  
  "user_limit":10  
}
```



API 登陆形成的会话也属于 WEB 管理会话

获取 WEB 会话限制配置

Action: web.session.limit.get

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=web.session.limit.get

响应参数:

名称	类型	范围	含义
global_limit	整数	1-128	全局 WEB 管理会话上限

user_limit	整数	1-64	每用户 WEB 管理会话上限
------------	----	------	----------------

响应举例:

```
{
  "global_limit":20,
  "user_limit":10
}
```

WEB 证书管理

获取 web 证书列表

Action: system.web.cert.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.cert.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	证书的名称。 运行证书固定为"web-cert.pem" 运行私钥固定为"web-cert.key" 缺省证书固定为"default-web-cert.pem" 缺省私钥固定为"default-web-cert.key" 自定义证书固定为"bak-web-cert.pem" 自定义私钥固定为"bak-web-cert.key"
type	字符串		类型: Certificate/key
issuer	字符串		发行者
common_name	字符串		通用名称
organization	字符串		组织
division	字符串		部门
locality	字符串		地区
state_province	字符串		省份
country	字符串		国家
email	字符串		电子邮箱
expiration	字符串		过期时间
key_type	字符串		公钥类型
version	字符串		版本
serial_number	字符串		序列号
fingerprint	字符串		指纹
key_size	数字		公钥长度

响应举例:

```
[{
  "name": "bak-web-cert.pem",
  "type": "certificate",
  "issuer": "Self",
  "common_name": "111111",
  "organization": "11111",
  "division": "11111",
  "locality": "11111",
  "state_province": "11111",
  "country": "CN",
  "email": "",
  "expiration": "Feb 11 07:32:35 2021 GMT",
  "key_type": "rsa",
  "version": "V1",
  "serial_number": "e52820eacdea09a3",
  "fingerprint":
    "SHA256/5A:C5:BE:25:69:03:31:6F:D7:B2:1C:A9:21:34:A3:4A:8E:23:19:C0:02:39:C3:26:86:32:CE:83:9D:
    5F:58:CF ",
  "key_size": 1024
}, {
  "name": "web-cert.pem",
  "type": "certificate",
  "issuer": "Self",
  "common_name": "222",
  "organization": "222",
  "division": "222",
  "locality": "222",
  "state_province": "222",
  "country": "CN",
  "email": "222@222.com",
  "expiration": "Jan 15 08:08:27 2021 GMT",
  "key_type": "rsa",
  "version": "V1",
  "serial_number": "b79dc75a480b33d1",
  "fingerprint":
    "SHA256/3E:9C:3E:B8:BA:A9:AE:A7:A6:62:45:47:03:0F:09:CA:8E:30:EF:E4:D5:7A:77:DA:53:BB:C7:25:5A:
    D6:96:28 ",
  "key_size": 1024
}, {
  "name": "default-web-cert.pem",
  "type": "certificate",
  "issuer": "Self",
  "common_name": "www.adc.com",
  "organization": "horizonadn",
  "division": "horizonadn",
```

```

    "locality": "horizonadn",
    "state_province": "Beijing",
    "country": "CN",
    "email": "",
    "expiration": "Oct 8 08:51:11 2027 GMT",
    "key_type": "rsa",
    "version": "V1",
    "serial_number": "aa4840194a01e9f2",
    "fingerprint":
"SHA256/B2:C4:C4:BB:CB:DB:78:07:48:A5:6B:CA:51:72:36:43:62:7A:0E:3A:33:D2:38:1D:AF:D4:4A:8A:A6:
88:B7:C9 ",
    "key_size": 2048
}, {
    "name": "bak-web-cert.key",
    "type": "key",
    "key_size": 1024
}, {
    "name": "web-cert.key",
    "type": "key",
    "key_size": 1024
}, {
    "name": "default-web-cert.key",
    "type": "key",
    "key_size": 2048
}]

```

删除 web 证书

Action: system.web.cert.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	要删除的证书或私钥名称	是	只能删除自定义证书 自定义证书固定为"bak-web-cert.pem" 自定义私钥固定为"bak-web-cert.key"

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.cert.del

请求 body: {"name": "bak-web-cert.key"}

应用 web 证书

Action: system.web.cert.apply

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.cert.app
ly

上传 web 私钥文件

Action: system.web.key.upload

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.key.uplo
ad

响应举例:

```
{"success":true,"msg":"Upload OK."}
```



Note

1. 使用 form-data 方式上传私钥文件,文件内容在 body 中
2. 上传之后的私钥会覆盖之前的私钥

下载 web 私钥文件

Action: system.web.key.download

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.key.down
load
请求 body: {"name": "bak-web-cert.key"}



Note

1. 此 API 会下载当前自定义 web 私钥文件,并且默认命名为 web-cert.key

上传 web 证书文件

Action: system.web.cert.upload


请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.cert.upload

响应举例:

```
{"success":true,"msg":"Upload OK."}
```

 Note	<ol style="list-style-type: none">1. 使用 form-data 方式上传证书文件,文件内容在 body 中2. 上传之后的证书会覆盖之前的证书
--	--

下载 web 证书文件

Action: system.web.cert.download

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.web.cert.download

请求 body: {"name": "bak-web-cert.pem"}

 Note	<ol style="list-style-type: none">1. 此 API 会下载当前自定义 web 证书文件,并且默认命名为 web-cert.pem
--	---

系统应用端口

设置系统应用端口配置

Action: system.ports.set

请求参数:

名称	类型	范围	含义	必选	备注
sshd_port	整数	1-65535	ssh 端口	否	系统默认 SSH 端口 22, 缺省表示不修改
telnet_port	整数	1-65535	telnet 端口	否	系统默认 TELNET 端口 23, 缺省表示不修改
http_port	整数	1-65535	http 端口	否	系统默认 HTTP 端口 80, 缺省表示不修改
https_port	整数	1-65535	https 端口	否	系统默认 HTTPS 端口 443, 缺省表示不修改

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ports.set

请求 body:
{
 "sshd_port": 220,

```

"telnet_port" : 230,
"http_port" : 80,
"https_port" : 4430
}

```



Note

SSH/TELNET/HTTP/HTTPS 的系统默认端口号只能自己使用,别的应用不能使用

例如 22 预留给 SSH, 即使 SSH 端口号已经修改为 22000,别的应用也不能用 22 端口

获取系统应用端口配置

Action: `system.ports.get`

请求参数:无

请求举例:

GET <http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ports.get>

响应参数:

名称	类型	范围	含义
ssh_port	整数	1-65535	ssh 端口
telnet_port	整数	1-65535	telnet 端口
http_port	整数	1-65535	http 端口
https_port	整数	1-65535	https 端口

响应举例:

```

{
  "ssh_port" : 220,
  "telnet_port" : 230,
  "http_port" : 80,
  "https_port" : 4430
}

```

系统操作

保存配置

Action: `save`

请求参数:无

请求举例:

GET <http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=save>

系统重启

Action: `system.action.reboot`

请求参数:

名称	类型	范围	含义	必选	备注
save	整数	0, 1	重启前是否保存配置, 0;不保存, 1:保存	否	缺省值: 0

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.action.reboot
 Body:
 {"save": 1}

重新加载配置

Action: system.action.reload

请求参数:

名称	类型	范围	含义	必选	备注
save	整数	0, 1	重新加载配置前是否保存配置, 0;不保存, 1:保存	否	缺省值: 0

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.action.reload
 Body:
 {"save": 1}

系统关机

Action: system.action.shutdown

请求参数:

名称	类型	范围	含义	必选	备注
save	整数	0, 1	系统关机前是否保存配置, 0;不保存, 1:保存	否	缺省值: 0

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.action.shutdown
 Body:
 {"save": 1}

系统恢复默认配置

Action: system.action.default

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

dhcp	整数	0, 1	恢复默认配置后, 管理口是否使用 dhcp 获取地址,	是	1: 是, 0: 否
ip	IPv4 地址		恢复默认配置后, 管理口使用的静态 IP 地址	否	当 dhcp 为 0 时必选
mask	掩码		恢复默认配置后, 管理口使用的掩码	否	当 dhcp 为 0 时必选
gw	IPv4 地址		恢复默认配置后, 管理口使用的网关 IP 地址	否	
ipv6	IPv6 地址		恢复默认配置后, 管理口使用的静态 IPv6 地址	否	
ipv6_prefix	IPv6 地址前缀		恢复默认配置后, 管理口使用的静态 IPv6 地址前缀	否	
ipv6_gw	IPv6 地址		恢复默认配置后, 管理口使用的网关 IPv6 地址	否	

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.action.default
Body:
{
  "dhcp": 0,
  "ip": "192.168.70.63",
  "mask": "255.255.255.0",
  "gw": "192.168.70.250",
  "ipv6": "2001:30::63",
  "ipv6_prefix": "64",
  "ipv6_gw": "2001:30::1"
}
```



1. 当 dhcp 为 1 时, 管理口使用 dhcp 获取 IP 地址, ip/mask/gw 参数无效
2. API 执行完成后, 需要不保存配置并重启才能生效

配置备份管理

添加启动配置文件备份

Action: system.config.add

请求参数:

名称	类型	范围	含义	必选	备注
file_name	字符串	长度 1-63	备份的启动配置文件的名称	是	
description	字符串	长度 1-63	备份的启动配置文件的描述信息	否	缺省值: 空字符串""
flag	整数	0, 1	要备份的配置文件类型, 0: 把当前运行配置进行备份, 1: 把当前启动配置文件进行备份	否	缺省值: 0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.add

请求 body:

```
{
  "file_name": "backup1",
  "description": "description of backup1",
  "flag": 1
}
```



- 1.仅备份了 startup 启动配置文件,不包括用户, erule 文件等其他文件
- 2.最多同时存在 3 个备份

获取启动配置文件列表

Action: system.config.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.list

响应参数:

名称	类型	范围	含义
file_name	字符串	长度 1-63	备份的启动配置文件的名称
description	字符串	长度 1-63	备份的启动配置文件的描述信息
time	字符串	长度 1-63	添加备份时的时间

响应举例:

```
[{
  "file_name": "config_beijing",
  "description": "bj",
  "time": "2019-09-06 14:25:30"
}, {
  "file_name": "config_nanjing",
  "description": "",
  "time": "2019-09-06 14:26:32"
}]
```

指定启动配置文件

Action: system.config.apply

请求参数:

名称	类型	范围	含义	必选	备注
file_name	字符串	长度 1-63	备份的启动配置文件的名称, 指定后, 下次系统重启或者下次加载配置时会使用指定的配置文件	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.apply
请求 body:
{
 "file_name": "config_beijing"
}

删除指定启动配置文件

Action: system.config.del

请求参数:

名称	类型	范围	含义	必选	备注
file_name	字符串	长度 1-63	要删除启动配置文件的名称	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.del
请求 body:
{
 "file_name": "config_beijing"
}

配置导出

Action: system.config.backup

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.backup



该 API 会下载一份全配置文件的 zip 压缩包, 包含所有的配置文件

配置导入

Action: system.config.restore

请求参数:无

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.config.restore

re



Note

1. 该 API 需要以 form-data 的方式上传之前导出的配置文件 zip 压缩包
2. 导入后需要重新加载配置或者系统重启,使恢复的配置生效

自动备份管理

获取自动备份配置

Action: `autobackup.config.get`

请求参数:无

请求举例:

GET

`http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=autobackup.config.get`

响应参数:

名称	类型	范围	含义
<code>ftp_host</code>	字符串	长度 1-191	ftp 服务器地址
<code>ftp_password</code>	字符串	长度 1-191	密码
<code>ftp_path</code>	字符串	长度 1-191	路径
<code>ftp_username</code>	字符串	长度 1-191	用户名
<code>timerange</code>	字符串	长度 1-63	时间对象
<code>max_backup</code>	整数	1-16	最大备份数 1-16
<code>ftp_port</code>	整数	1-65535	端口 1-65535

响应举例:

```
{
  "ftp_host": "",
  "ftp_password": "",
  "ftp_path": "",
  "ftp_username": "",
  "timerange": "",
  "max_backup": 3,
  "ftp_port": 21
}
```

设置自动备份配置

Action: `autobackup.config.set`

请求参数:

名称	类型	范围	含义	必选	备注
ftp_host	字符串	长度 1-191	ftp 服务器地址	是	
ftp_password	字符串	长度 1-191	密码	是	
ftp_path	字符串	长度 1-191	路径	是	
ftp_username	字符串	长度 1-191	用户名	是	
timerange	字符串	长度 1-63	时间对象	是	
max_backup	整数	1-16	最大备份数 1-16	是	
ftp_port	整数	1-65535	端口 1-65535	是	

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=autobackup.config.set

Body:

```
{
  "ftp_host": "",
  "ftp_password": "",
  "ftp_path": "",
  "ftp_username": "",
  "timerange": "day",
  "max_backup": 3,
  "ftp_port": 21
}
```



1. 需要提前配置时间对象，并填入相应的选项

自动备份恢复

Action: autobackup.restore

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	备份文件名	是	

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=autobackup.restore
 请求 body

```
{
  "name": "2.zip"
}
```

自动备份列表

Action: autobackup.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=autobackup.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	备份文件名

请求举例:

```
[  
  {  
    "name": "1.zip"  
  },  
  {  
    "name": "2.zip"  
  }  
]
```

自动备份删除

Action: autobackup.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	备份文件名	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=autobackup.del

请求 body

```
{  
  "name": "2.zip"  
}
```

诊断信息

诊断信息导出

Action: system.diaginfo.export

请求参数:无

请求举例:

GET

<http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.diaginfo.exp>
[ort](#)



该 API 会下载一份全配置文件的 zip 压缩包，包含所有的配置文件

版本镜像管理

获取版本镜像列表

Action: system.image.list

请求参数:无

请求举例:

GET

<http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.image.list>

响应参数:

名称	类型	范围	含义
file_name	字符串	长度 1-63	版本镜像文件的名称
bflag	字符串	长度 1-63	版本镜像是否为指定镜像,当版本镜像为指定镜像时才存在该关键字,并且取值固定为“V”
time	字符串	长度 1-63	版本镜像上传的时间

响应举例:

```
[{
  "file_name": "tst_ad_hongji_image_V3.8R2B410_2019-08-13_12h12m",
  "bflag": "V",
  "time": "2019-09-03 13:40:02"
}, {
  "file_name": "Current Image",
  "time": "2019-08-12 16:16:53"
}]
```

指定镜像

Action: system.image.apply

请求参数:

名称	类型	范围	含义	必选	备注
file_name	字符串	长度 1-63	指定下次启动使用的镜像文件的名称	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.image.apply

Body:

```
{"file_name":"tst_ad_hongji_image_V3.8R2B410_2019-08-13_12h12m"}
```

通过 FTP/TFTP 方式上传镜像

Action: system.image.upload

请求参数:

名称	类型	范围	含义	必选	备注
host	IPv4 地址	长度 1-31	FTP/TFTP 服务器的 IPv4 地址	是	
location	字符串	长度 1-63	镜像名称	是	镜像必须处于 FTP/TFTP 的根目录
protocol	字符串		使用的协议, 仅支持“FTP”和“TFTP”	是	
port	整数	1-65535	使用的端口号,	否	FTP 缺省 21, TFTP 缺省 69
username	字符串	长度 1-31	用户名	否	缺省值: 空字符串”
password	字符串	长度 1-31	密码	否	缺省值: 空字符串”
use_mgmt_port	整数	0, 1	使用管理口	否	缺省值:0,
reboot	整数	0, 1	上传完镜像后是否自动重启	否	1:是, 0: 否; 缺省值:1
save_config	整数	0, 1	自动重启时是否保存配置	否	1:是, 0: 否; 缺省值:0,

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.image.upload

举例 1:使用 FTP 上传镜像

Body1:

```
{  
  "host":"192.168.1.100",  
  "location":"tst_ad_hongji_image_V3.8R2B410_2019-08-13_12h12m",  
  "protocol":"FTP",  
  "username":"anonymous",  
  "password":"password",  
  "reboot":1,  
  "save_config":1  
}
```

举例 2:使用 TFTP 上传镜像

Body2:

```
{  
  "host":"192.168.1.100",  
  "location":"tst_ad_hongji_image_V3.8R2B410_2019-08-13_12h12m",  
}
```

```

"protocol": "TFTP",
"reboot": 1,
"save_config": 1
}

```



1. 上传之后的镜像会自动设为指定镜像
2. 上传的镜像总数始终为 2 个，上传后会删除上传前的非指定镜像

通过 HTTP/HTTPS 方式上传镜像

Action: system.image.upload

url: /webupload/adcap/v2.0/

请求参数:

名称	类型	范围	含义	必选	备注
webfilename	字符串	长度 1-63	要显示在镜像列表中的镜像名称	是	支持字符 0-9a-zA-Z._-
reboot	整数	0, 1	上传完镜像后是否自动重启	否	1:是, 0: 否; 缺省值:1
save_config	整数	0, 1	自动重启时是否保存配置	否	1:是, 0: 否; 缺省值:0,

请求举例:

POST
http://192.168.70.73/webupload/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.image.upload&webfilename=tst_ad_hongji_image_V3.8R2B410_2019-08-13_12h12m&reboot=1&save_config=1



3. 使用 form-data 方式上传镜像文件,文件内容在 body 中
4. 上传之后的镜像会自动设为指定镜像
5. 上传的镜像总数始终为 2 个，上传后会删除上传前的非指定镜像
6. 使用 HTTP/HTTPS 上传镜像的时候,URL 与别的 API 不一样
7. 使用 HTTP/HTTPS 上传镜像的时候,参数在 URL 中携带,而不是在 body 中
8. 根据上传的网速不同,需要耗费的时间不一样,从几十秒到几分钟不等.

网络连通性测试

Ping 请求

Action: system.ping.request

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPv4 地址		目的 IPv4 地址	是	
source	IPv4 地址		指定源 IPv4 地址, 不指定时根据路由自动选择源	否	缺省值: 不指定

			IP 地址		
count	整数	1-60	ping 的个数	否	缺省值: 5
interval	整数	1-5	ping 的间隔, 单位秒	否	缺省值: 0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ping.request

请求 body:

```
{
  "destination": "192.168.50.107",
  "source": "20.1.1.1",
  "count": 55,
  "interval": 3
}
```

Ping 结果

Action: system.ping.result

请求参数: 无

请求举例:


GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.ping.result

响应参数:

名称	类型	范围	含义
result	字符串		返回 ping 请求的结果

响应举例:

```
{
  "result": "PING 192.168.50.107 (192.168.50.107) from 192.168.70.73 : 56(84) bytes of data. \n64 bytes from 192.168.50.107: icmp_seq=1 ttl=127 time=3.54 ms\n64 bytes from 192.168.50.107: icmp_seq=2 ttl=127 time=0.442 ms\n64 bytes from 192.168.50.107: icmp_seq=3 ttl=127 time=0.405 ms\n64 bytes from 192.168.50.107: icmp_seq=4 ttl=127 time=0.405 ms\n64 bytes from 192.168.50.107: icmp_seq=5 ttl=127 time=0.518 ms\n64 bytes from 192.168.50.107: icmp_seq=6 ttl=127 time=3.14 ms\n64 bytes from 192.168.50.107: icmp_seq=7 ttl=127 time=0.593 ms\n\n--- 192.168.50.107 ping statistics ---\n7 packets transmitted, 7 received, 0% packet loss, time 30005ms\nrtt min/avg/max/mdev = 0.405/1.291/3.540/1.302 ms\n"}
}
```

 Note	只有在 Ping 请求执行完成后才能得到结果
--	------------------------

抓包

添加入接口过滤条件

Action: capture.rxif.add

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	物理接口的插槽号	是	
port	整数	0-28	物理接口的端口号	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rxif.add

请求 body:

```
{
  "slot":0,
  "port":2
}
```

获取入接口列表

Action: capture.rxif.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rxif.list

响应参数:

名称	类型	范围	含义
slot	整数	0-28	物理接口的插槽号
port	整数	0-28	物理接口的端口号

响应举例:

```
[{
  "slot":0,
  "port":2
}]
```

删除入接口

Action: capture.rxif.del

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	物理接口的插槽号	是	
port	整数	0-28	物理接口的端口号	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rxif.del

请求 body:

```
{
  "slot":0,
  "port":2
}
```

添加出接口过滤条件

Action: capture.txif.add

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	物理接口的插槽号	是	
port	整数	0-28	物理接口的端口号	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.txif.add

请求 body:

```
{
  "slot":0,
  "port":2
}
```

获取出接口列表

Action: capture.txif.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.txif.list

响应参数:

名称	类型	范围	含义
slot	整数	0-28	物理接口的插槽号
port	整数	0-28	物理接口的端口号

响应举例:

```
[{
  "slot":0,
  "port":2
}]
```

删除出接口

Action: capture.txif.del

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	物理接口的插槽号	是	
port	整数	0-28	物理接口的端口号	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.txif.del

请求 body:

```
{  
  "slot":0,  
  "port":2  
}
```

添加过滤规则

Action: capture.rule.add

请求参数:

名称	类型	范围	含义	必选	备注
ruleid	整数	1-255	过滤规则 ID	是	唯一,不能重复
srcport	整数	1-65535	源端口	否	缺省值:0,表示不检查源端口
dstport	整数	1-65535	目的端口	否	缺省值:0,表示不检查目的端口
l3p	字符串		L3 协议类型,支持 arp、ip 和 ipv6 三种	否	缺省值:空字符串"",表示不检查 L3 协议
l4p	字符串		L4 协议类型,支持 tcp、udp、icmp 和 icmp-v6 四种	否	缺省值:空字符串"",表示不检查 L4 协议
ipv6_addr	IPv6 地址		IPv6 地址	否	缺省值:空字符串"",表示不检查 IPv6 地址
srcaddr	IPv4 地址		源 IP 地址	否	缺省值:空字符串"",表示不检查源 IP 地址
dstaddr	IPv4 地址		目的 IP 地址	否	缺省值:空字符串"",表示不检查目的 IP 地址
srcmac	MAC 地址		源 MAC 地址	否	缺省值:空字符串"",表示不检查源 MAC 地址
dstmac	MAC 地址		目的 MAC 地址	否	缺省值:空字符串"",表示不检查目的 MAC 地址

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rule.add

请求 body:

```
{  
  "ruleid":1,  
  "srcport":65535,
```

```

"dstport":65535,
"l3p": "ip",
"l4p": "tcp",
"srcaddr":"20.1.1.1",
"dstaddr":"202.1.1.2",
"srcmac":"0050.56ad.11b3",
"dstmac":"0050.56ba.dad3"
}

```

获取过滤规则列表

Action: capture.rule.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rule.list

响应参数:

名称	类型	范围	含义
ruleid	整数	1-255	过滤规则 ID
srcport	整数	1-65535	源端口
dstport	整数	1-65535	目的端口
l3p	字符串		L3 协议类型, 支持 arp、ip 和 ipv6 三种
l4p	字符串		L4 协议类型, 支持 tcp、udp、icmp 和 icmp-v6 三种
ipv6_addr	IPv6 地址		IPv6 地址
srcaddr	IPv4 地址		源 IP 地址
dstaddr	IPv4 地址		目的 IP 地址
srcmac	MAC 地址		源 MAC 地址
dstmac	MAC 地址		目的 MAC 地址

响应举例:

```

[ {
  "ruleid": 1,
  "srcport": 0,
  "dstport": 0,
  "l3p": "ip",
  "l4p": "tcp",
  "ipv6_addr": "",
  "srcaddr": "1.1.1.1",
  "dstaddr": "",
  "srcmac": "",
  "dstmac": ""
} ]

```

删除过滤规则

Action: capture.rule.del

请求参数:

名称	类型	范围	含义	必选	备注
ruleid	整数	1-255	过滤规则 ID	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.rule.del

请求 body:

```
{"ruleid": 1}
```

添加抓包文件

Action: capture.file.add

请求参数:

名称	类型	范围	含义	必选	备注
filename	字符串	长度 1-127	抓包文件名称	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.file.add

请求 body:

```
{"filename": "123"}
```



Note

1. 添加抓包文件后,会立即开始按照抓包的过滤条件(包含入接口/出接口/规则)进行抓包
2. 默认抓包在达到最大抓包时间(300 秒)或者最大文件大小(10M)时会自动停止

获取抓包文件列表

Action: capture.file.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.file.list

响应参数:

名称	类型	范围	含义
filename	字符串	长度 1-127	抓包文件名称

响应举例:

```
[{"filename": "123"}]
```


删除抓包文件

Action: capture.file.del

请求参数:

名称	类型	范围	含义	必选	备注
filename	字符串	长度 1-127	抓包文件名称	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.file.del
请求 body:
{ "filename": "123" }

导出抓包文件

Action: capture.file.export

请求参数:

名称	类型	范围	含义	必选	备注
export_file	字符串	长度 1-255	抓包文件名称	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=capture.file.export
请求 body:
{ "export_file": "123" }



此 API 会下载一个抓包文件(pcap)的压缩包

SMTP 配置

获取 SMTP 配置

Action: system.smtp.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.smtp.get

响应参数:

名称	类型	范围	含义
server	字符串	长度 1-31	SMTP 服务器, 可以指定 IPv4 地址、IPv6 地址和主机名, 长度仅指主机名
port	整数	1-65535	SMTP 服务端口

emailto	字符串	长度 0-640	接收邮件列表，邮件地址之间用逗号隔开，每个邮件地址不超过 63 字节，不超过 10 个邮件
emailfrom	字符串	长度 0-63	发送者邮件地址
auth	整数	0-1	是否需要验证
user	字符串	1-31	用户名
password	字符串	1-31	密码

响应举例:

```
{
  "server": "202.110.78.20",
  "port": 35,
  "emailto": "pub@horizon.com,wangdong@263.com,ant@bigone.com.cn",
  "emailfrom": "lji@163.com",
  "auth": 1,
  "user": "lji",
  "password": "12345.com"
}
```

编辑 SMTP 配置

Action: system.smtp.set

请求参数:

名称	类型	范围	含义	必选	备注
server	字符串	长度 1-31	SMTP 服务器,可以指定 IPv4 地址、IPv6 地址和主机名	否	长度仅指主机名
port	整数	1-65535	SMTP 服务端口	否	默认 25
emailto	字符串	长度 0-640	接收邮件列表，邮件地址之间用逗号隔开，每个邮件地址不超过 63 字节，不超过 10 个邮件	否	
emailfrom	字符串	长度 0-63	发送者邮件地址	否	
auth	整数	0-1	是否需要验证	否	
user	字符串	1-31	用户名	否	
password	字符串	1-31	密码	否	

请求举例:

POST <http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.smtp.set>

Body:

```
{
  "server": "202.110.78.20",
  "port": 35,
  "emailto": "pub@horizon.com,wangdong@263.com,ant@bigone.com.cn",
  "emailfrom": "lji@163.com",
  "auth": 1,
}
```

```
"user" : "lji",
"password" : "12345.com"
}
```

日志配置

获取业务日志配置

Action: log.config.service.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action= log.config.service.get

响应参数:

名称	类型	范围	含义
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用
Size	整数	10000-60000	业务日志的最大保留个数。

响应举例:

```
{
  "disable": 0,
  "size": 10000
}
```

编辑业务日志配置

Action: log.config.service.set

请求参数:

名称	类型	范围	含义	必选	备注
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用	否	默认为 0, 不禁用
Size	整数	10000-60000	业务日志的最大保留个数。	否	默认 10000

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action= log.config.service.set

Body:

```
{
  "disable": 0,
  "size": 10000
}
```

获取审计日志配置

Action: log.config.audit.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.config.audit.get

响应参数:

名称	类型	范围	含义
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用
Size	整数	1000-30000	业务日志的最大保留个数。

响应举例:

```
{
  "disable": 0,
  "size": 10000
}
```

编辑审计日志配置

Action: log.config.audit.set

请求参数:

名称	类型	范围	含义	必选	备注
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用	否	默认为 0, 不禁用
Size	整数	1000-30000	业务日志的最大保留个数。	否	默认 20000

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.config.audit.set

Body:

```
{
  "disable": 0,
  "size": 10000
}
```

获取 NAT 日志配置

Action: log.config.nat.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.config.nat.get

响应参数:

名称	类型	范围	含义
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用
Size	整数	10000-60000	业务日志的最大保留个数。

响应举例:

```
{
  "disable": 0,
  "size": 10000
}
```

编辑 NAT 日志配置

Action: log.config.nat.set

请求参数:

名称	类型	范围	含义	必选	备注
disable	整数	0-1	禁止打印日志。0: 不禁用, 1: 禁用	否	默认为 0, 不禁用
Size	整数	10000-60000	业务日志的最大保留个数。	否	默认 10000

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.config.nat.set

Body:

```
{  
  "disable": 0,  
  "size": 10000  
}
```

日志获取

业务日志显示列表

Action: log.service.list

请求参数:

名称	类型	范围	含义	必选	备注
direct	整数	0-3	log 取页方向	否	0:first 1:prev 2:next 3:last 默认 0
index	整数	0-4294967294	日志索引	否	默认 0
limit	整数	1-1000	输出限制	否	默认 30
level	整数	0-7	输出其及以上级别	否	0: EMERG 1: ALERT 2: CRIT 3: ERR 4: WARNING 5: NOTICE 6: INFO 7: DEBUG 默认 7

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.service.list

请求 body:

```
{  
  "direct": 0,  
}
```

```

    "index": 0,
    "limit": 30,
    "level": 7
  }

```

响应参数:

名称	类型	范围	含义
item	字符串	1-1024	Log 内容
index	整数	0-4294967294	日志索引

响应举例:

```

  [
    {
      "index": 100,
      "item": "Sep 27 2019 15:18:21 local0.Notice      Horizon platform: admin has      opened a
web session 19 from 192.168.50.106."
    }, {
      "index": 99,
      "item": "Sep 27 2019 15:04:30 local0.Notice      Horizon platform: Session 18 is
closed successfully."
    }, {
      "index": 98,
      "item": "Sep 27 2019 15:04:30 local0.Notice      Horizon platform: admin session      18
timeout from 192.168.50.106."
    }, {
      "index": 97,
      "item": "Sep 27 2019 15:04:30 local0.Notice      Horizon platform: admin session      18
timeout from 192.168.50.106."
    }, {
      "index": 96,
      "item": "Sep 27 2019 14:53:39 local0.Notice      Horizon platform: admin has      opened a
web session 18 from 192.168.50.106."
    }, {
      "index": 95,
      "item": "Sep 27 2019 14:48:30 local0.Notice      Horizon platform: Session 17 is
closed successfully."
    }, {
      "index": 94,
      "item": "Sep 27 2019 14:48:30 local0.Notice      Horizon platform: admin session      17
timeout from 192.168.50.106."
    }, {
      "index": 93,
      "item": "Sep 27 2019 14:48:30 local0.Notice      Horizon platform: admin session      17
timeout from 192.168.50.106."
    }, {
      "index": 92,
      "item": "Sep 27 2019 14:37:47 local0.Notice      Horizon platform: admin has      opened a
web session 17 from 192.168.50.106."
    }
  ]

```

}, {	"index": 91,	"item": "Sep 27 2019 14:29:26 local0.Notice closed successfully."	Horizon platform: Session 16 is	
}, {	"index": 90,	"item": "Sep 27 2019 14:29:26 local0.Notice timeout from 192.168.50.106."	Horizon platform: admin session	16
}, {	"index": 89,	"item": "Sep 27 2019 14:19:16 local0.Notice shell session 16 from 192.168.50.106."	Horizon platform: admin has	opened a
}, {	"index": 88,	"item": "Sep 27 2019 14:13:18 local0.Notice closed successfully."	Horizon platform: Session 15 is	
}, {	"index": 87,	"item": "Sep 27 2019 14:13:18 local0.Notice timeout from 192.168.50.106."	Horizon platform: admin session	15
}, {	"index": 86,	"item": "Sep 27 2019 14:11:30 local0.Notice closed successfully."	Horizon platform: Session 14 is	
}, {	"index": 85,	"item": "Sep 27 2019 14:11:30 local0.Notice timeout from 192.168.50.106."	Horizon platform: admin session	14
}, {	"index": 84,	"item": "Sep 27 2019 14:11:30 local0.Notice timeout from 192.168.50.106."	Horizon platform: admin session	14
}, {	"index": 83,	"item": "Sep 27 2019 14:03:14 local0.Notice shell session 15 from 192.168.50.106."	Horizon platform: admin has	opened a
}, {	"index": 82,	"item": "Sep 27 2019 13:51:13 local0.Notice web session 14 from 192.168.50.106."	Horizon platform: admin has	opened a
}, {	"index": 81,	"item": "Sep 27 2019 13:40:49 local0.Notice web session 13 from 192.168.50.106."	Horizon platform: admin has	opened a
}, {	"index": 80,			

```

        "item": "Sep 27 2019 13:40:17 local0.Notice
closed successfully."
    }, {
        "index": 79,
        "item": "Sep 27 2019 13:40:17 local0.Notice
timeout from 192.168.50.106."
    }, {
        "index": 78,
        "item": "Sep 27 2019 13:40:17 local0.Notice
timeout from 192.168.50.106."
    }, {
        "index": 77,
        "item": "Sep 27 2019 13:31:18 local0.Notice
closed successfully."
    }, {
        "index": 76,
        "item": "Sep 27 2019 13:31:18 local0.Notice
timeout from 192.168.50.106."
    }, {
        "index": 75,
        "item": "Sep 27 2019 13:22:23 local0.Notice
web session 12 from 192.168.50.106."
    }, {
        "index": 74,
        "item": "Sep 27 2019 13:21:09 local0.Notice
shell session 11 from 192.168.50.106."
    }, {
        "index": 73,
        "item": "Sep 27 2019 13:19:19 local0.Info
down"
    }, {
        "index": 72,
        "item": "Sep 27 2019 13:17:50 local0.Notice
web session 10 from 192.168.50.106."
    }, {
        "index": 71,
        "item": "Sep 27 2019 11:11:46 local0.Notice
closed successfully."
    }
}

```

Horizon platform: Session 12 is

Horizon platform: admin session 12

Horizon platform: admin session 12

Horizon platform: Session 11 is

Horizon platform: admin session 11

Horizon platform: admin has opened a

Horizon platform: admin has opened a

Horizon os: Interface ve2 change status to

Horizon platform: admin has opened a

Horizon platform: Session 9 is

审计日志显示列表

Action: log.audit.list

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

direct	整数	0-3	log 取页方向	否	0:first 1:prev 2:next 3:last 默认 0
index	整数	0-4294967294	日志索引	否	默认 0
limit	整数	1-1000	输出限制	否	默认 30
start_time	整数	0-4294967294	日志产生时间过滤, 开始时间	否	时间单位为秒, 默认 0
time_range	整数	0-4294967294	日志产生时间过滤, 开始时间后持续时间	否	时间单位为秒, 0 表示持续到当前, 默认 0
user_name	字符串	1-31	操作用户	是	提取该操作用户的操作

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.audit.list

请求 body:

```
{
  "direct": 0,
  "index": 0,
  "limit": 20,
  "start_time": 0,
  "time_range": 0,
  "user_name": "admin"
}
```

响应参数:

名称	类型	范围	含义
item	字符串	1-1024	Log 内容
index	整数	0-4294967294	日志索引

响应举例:

```
[{
  "index": 277,
  "item": "Sep 29 2019 13:58:43 [@api]: (ADCAPIv2.0) [login] rspbody:
  {\"authkey\": \"84482cf5609bd6bc37664765937eda\"}"
}, {
  "index": 276,
  "item": "Sep 29 2019 13:58:43 [*]: User admin opened a web session 31 from
  192.168.50.106"
}, {
  "index": 275,
  "item": "Sep 29 2019 13:58:43 [@api]: (ADCAPIv2.0) [login] reqbody:
  {\"direct\":0}"
}, {
  "index": 274,
```

```

    "item": "Sep 29 2019 13:56:21  [*]: User admin opened a web session 30 from
192.168.50.106"
  }, {
    "index": 273,
    "item": "Sep 29 2019 13:56:20  [@api]: (ADCAPIv2.0) [login] username: admin"
  }, {
    "index": 272,
    "item": "Sep 29 2019 13:31:46  [*]: User admin opened a web session 29 from
192.168.50.106"
  }, {
    "index": 271,
    "item": "Sep 29 2019 13:31:46  [@api]: (ADCAPIv2.0) [login] username: admin"
  }, {
    "index": 270,
    "item": "Sep 29 2019 11:02:26  [*]: User admin opened a web session 28 from
192.168.50.106"
  }, {
    "index": 269,
    "item": "Sep 29 2019 10:38:14  [*]: Session 26 closed"
  }, {
    "index": 268,
    "item": "Sep 29 2019 10:30:42  [*]: User admin opened a web session 27 from
192.168.50.106"
  }, {
    "index": 267,
    "item": "Sep 29 2019 10:29:24  [admin@cli]: show running-config"
  }, {
    "index": 266,
    "item": "Sep 29 2019 10:29:24  [admin@cli]: enable"
  }, {
    "index": 265,
    "item": "Sep 29 2019 10:28:23  [*]: User admin opened a shell session 26 from
192.168.50.106"
  }, {
    "index": 264,
    "item": "Sep 29 2019 10:22:12  [*]: Session 24 closed"
  }, {
    "index": 263,
    "item": "Sep 29 2019 10:15:25  [*]: User admin opened a web session 25 from
192.168.50.106"
  }, {
    "index": 262,
    "item": "Sep 29 2019 10:12:11  [admin@cli]: show logging audit"
  }, {
    "index": 261,
    "item": "Sep 29 2019 10:11:21  [admin@cli]: configure"
  }

```

```

    }, {
      "index": 260,
      "item": "Sep 29 2019 10:11:20 [admin@cli]: enable"
    }, {
      "index": 259,
      "item": "Sep 29 2019 10:11:18 [*]: User admin opened a shell session 24 from
192.168.50.106"
    }, {
      "index": 258,
      "item": "Sep 29 2019 10:10:45 [*]: User admin opened a web session 23 from
192.168.50.106"
    }
  ]
}

```

NAT 日志显示列表

Action: log.nat.list

请求参数:

名称	类型	范围	含义	必选	备注
direct	整数	0-3	log 取页方向	是	0:first 1:prev 2:next 3:last 默认 0
index	整数	0-4294967294	日志索引	是	默认 0
limit	整数	1-1000	输出限制	是	默认 30

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.nat.list

响应参数:

名称	类型	范围	含义
item	字符串	1-1024	Log 内容
index	整数	0-4294967294	日志索引

业务日志清除

Action: log.service.clear

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.service.clear

审计日志清除

Action: log.audit.clear

请求参数: 无

请求举例:

GET <http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.audit.clear>

NAT 日志清除

Action: log.nat.clear

请求参数: 无

请求举例:

GET <http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.nat.clear>

业务日志下载

Action: log.service.download

请求参数: 无

请求举例:

GET

<http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.service.download>



此 API 会下载一个文件

审计日志下载

Action: log.audit.download

请求参数: 无

请求举例:

GET

<http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.audit.download>



此 API 会下载一个文件

NAT 日志下载

Action: log.nat.download

请求参数: 无

请求举例:

GET

<http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.nat.download>



Note

此 API 会下载一个文件

诊断日志下载

Action: log.coredump.download

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.coredump.download



Note

此 API 会下载一个文件

系统日志下载

Action: log.system.download

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.system.download



Note

此 API 会下载一个文件

日志发送

添加日志发送配置

Action: log.syslog.server.add

请求参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	
log_code	整数	0-1	code 编码开启状态, 0 表示关闭, 1 表示开启, 默认关闭, 开启时, 日志包含 code 编码	否	

facility	整数	0-7	Syslog 的设施号，默认 0	否	
nat_log	整数	0-1	发送 nat 日志开关，0 表示关闭，1 表示开启，默认开启	否	
audit_log	整数	0-1	发送审计日志开关，0 表示关闭，1 表示开启，默认开启	否	
service_log	整数	0-1	发送业务日志开关，0 表示关闭，1 表示开启，默认开启	否	
match_type	整数	0-2	过滤匹配，0：关闭过滤，1：level 过滤和 event 过滤是或操作，2：level 过滤和 event 过滤是与操作，默认 0	否	
level_filter	整数	0-7	日志级别发送过滤，默认 7	否	
event_list	整数 数组	1-15	1：接口状态；2：设备重启；3：登陆事件；4：健康检查故障；5：健康检查繁忙；9：虚拟机联动；11：GSLB 日志；12：erule 日志	否	

请求举例：

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.syslog.server.add

请求 body:

```
{
  "host" : "192.168.70.250",
  "port" : 514,
  "log_code" : 1,
  "facility" : 0,
  "nat_log" : 1,
  "audit_log" : 1,
  "service_log" : 1,
  "match_type" : 0,
  "level_filter" : 1,
  "event_list" : [ 3 ]
}
```

获取日志发送配置列表

Action: log.syslog.server.list

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.syslog.server.list

响应参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址，可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	
log_code	整数	0-1	code 编码开启状态，0 表示关闭，1 表示开启，默认关闭，开启时，日志包含 code 编码	否	
facility	整数	0-7	Syslog 的设施号，默认 0	否	
nat_log	整数	0-1	发送 nat 日志开关，0 表示关闭，1 表示开启，默认开启	否	
audit_log	整数	0-1	发送审计日志开关，0 表示关闭，1 表示开启，默认开启	否	
service_log	整数	0-1	发送业务日志开关，0 表示关闭，1 表示开启，默认开启	否	
match_type	整数	0-2	过滤匹配，0：关闭过滤，1：level 过滤和 event 过滤是	否	

			或操作, 2: level 过滤和 event 过滤是与操作, 默认 0		
level_filter	整数	0-7	日志级别发送过滤, 默认 7	否	
event_list	整数 数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动; 11: GSLB 日志; 12: erule 日志	否	

响应举例:

```
[ {
  "host" : "192.168.70.250",
  "port" : 514,
  "log_code" : 1,
  "facility" : 0,
  "nat_log" : 1,
  "audit_log" : 1,
  "service_log" : 1,
  "match_type" : 0,
  "level_filter" : 7,
  "event_list" : [ ]
}, {
  "host" : "192.168.70.251",
  "port" : 514,
  "log_code" : 1,
  "facility" : 0,
  "nat_log" : 1,
  "audit_log" : 1,
  "service_log" : 1,
  "match_type" : 0,
  "level_filter" : 2,
  "event_list" : [ 1, 4, 5 ]
} ]
```

获取指定日志发送配置

Action: log.syslog.server.get

请求参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	

请求举例:

POST <http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.syslog.server.get>

请求 body:

```
{
  "host" : "192.168.70.250",
  "port" : 514
}
```

}

响应参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	
log_code	整数	0-1	code 编码开启状态, 0 表示关闭, 1 表示开启, 默认关闭, 开启时, 日志包含 code 编码	否	
facility	整数	0-7	Syslog 的设施号, 默认 0	否	
nat_log	整数	0-1	发送 nat 日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	
audit_log	整数	0-1	发送审计日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	
service_log	整数	0-1	发送业务日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	
match_type	整数	0-2	过滤匹配, 0: 关闭过滤, 1: level 过滤和 event 过滤是或操作, 2: level 过滤和 event 过滤是与操作, 默认 0	否	
level_filter	整数	0-7	日志级别发送过滤, 默认 7	否	
event_list	整数 数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动; 11: GSLB 日志; 12: erule 日志	否	

响应举例:

```
{
  "host" : "192.168.70.250",
  "port" : 514,
  "log_code" : 1,
  "facility" : 0,
  "nat_log" : 1,
  "audit_log" : 1,
  "service_log" : 1,
  "match_type" : 0,
  "level_filter" : 7,
  "event_list" : [ ]
}
```

编辑日志发送配置

Action: log.syslog.server.edit

请求参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	
log_code	整数	0-1	code 编码开启状态, 0 表示关闭, 1 表示开启, 默认关闭, 开启时, 日志包含 code 编码	否	
facility	整数	0-7	Syslog 的设施号, 默认 0	否	
nat_log	整数	0-1	发送 nat 日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	

audit_log	整数	0-1	发送审计日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	
service_log	整数	0-1	发送业务日志开关, 0 表示关闭, 1 表示开启, 默认开启	否	
match_type	整数	0-2	过滤匹配, 0: 关闭过滤, 1: level 过滤和 event 过滤是或操作, 2: level 过滤和 event 过滤是与操作, 默认 0	否	
level_filter	整数	0-7	日志级别发送过滤, 默认 7	否	
event_list	整数数组	1-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动; 11: GSLB 日志; 12: erule 日志	否	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.syslog.server.edit

请求 body:

```
{
  "host" : "192.168.70.250",
  "port" : 514,
  "log_code" : 1,
  "facility" : 0,
  "nat_log" : 1,
  "audit_log" : 1,
  "service_log" : 1,
  "match_type" : 0,
  "level_filter" : 7,
  "event_list" : [3]
}
```

删除日志发送配置

Action: log.syslog.server.del

请求参数:

名称	类型	范围	含义	必选	备注
host	字符串	长度 1-31	syslog 服务器地址, 可以使用主机名/IPv4 地址/IPv6 地址	是	长度指主机名
port	整数	1-65535	Syslog 服务器端口	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.syslog.server.del

请求 body:

```
{
  "host" : "192.168.70.250",
  "port" : 514
}
```

日志告警

获取 Email 日志告警配置

Action: log.alarm.email.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.alarm.email.get

响应参数:

名称	类型	范围	含义
delay_send_buff	整数	长度 32-128	延迟发送触发的 buff 长度
delay_send_time	整数	10-1440	延迟发送触发的时间,单位分钟
send_event	整数数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动; 11: GSLB 日志; 12: erule 日志
send_level	整数	-1, 0, 1, 2, 5	日志级别发送过滤, -1 表示关闭级别过滤, 默认-1

响应举例:

```
{
  "delay_send_buff" : 120,
  "delay_send_time" : 110,
  "send_event" : [ 2, 4, 5 ],
  "send_level" : 5
}
```

编辑 Email 日志告警配置

Action: log.alarm.email.set

请求参数:

名称	类型	范围	含义	必选	备注
delay_send_buff	整数	长度 32-128	延迟发送触发的 buff 长度	否	默认 50
delay_send_time	整数	10-1440	延迟发送触发的时间,单位分钟	否	默认 10
send_event	整数数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动; 11: GSLB 日志; 12: erule 日志	否	
send_level	整数	-1, 0, 1, 2, 5	日志级别发送过滤, -1 表示关闭级别过滤, 默认-1	否	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.alarm.email.set

Body:

```
{
  "delay_send_buff" : 120,
  "delay_send_time" : 110,
  "send_event" : [ 2, 4, 5 ],
  "send_level" : 5
}
```

获取短信日志告警配置

Action: log.alarm.sms.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.alarm.sms.get

响应参数:

名称	类型	范围	含义
url	字符串	0-1023	发送短信平台
send_event	整数数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动;
send_level	整数	-1, 0, 1, 2, 5	日志级别发送过滤, -1 表示关闭级别过滤, 默认-1

响应举例:

```
{
  "send_event" : [ 1, 3 ],
  "url" : "h.163.com.cn",
  "send_level" : 2
}
```

编辑短信日志告警配置

Action: log.alarm.sms.set

请求参数:

名称	类型	范围	含义	必选	备注
url	字符串	0-1023	发送短信平台	否	
send_event	整数数组	0-15	1: 接口状态; 2: 设备重启; 3: 登陆事件; 4: 健康检查故障; 5: 健康检查繁忙; 9: 虚拟机联动;	否	
send_level	整数	-1, 0, 1, 2, 5	日志级别发送过滤, -1 表示关闭级别过滤, 默认-1	否	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=log.alarm.sms.set

Body:

```
{
  "send_event" : [ 1, 3 ],
  "url" : "h.163.com.cn",
}
```

```
"send_level" : 2
}
```

SLB 服务器负载均衡

SLB 全局混杂设置

获取 SLB 全局混杂配置

Action: slb.global.allow_promis_intf_vip.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.global.allow_promis_intf_vip.get

名称	类型	范围	含义
status	整数	0, 1	SLB 全局混杂设置, 0: 关闭, 1: 开启

响应参数:

响应举例:

```
{"status": 0}
```

设置 SLB 全局混杂配置

Action: slb.global.allow_promis_intf_vip.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0, 1	SLB 全局混杂设置	是	0: 关闭, 1: 开启

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.global.allow_promis_intf_vip.set

Body:

```
{"status": 1}
```



Note

当需要使用接口 IP 地址作为 VIP 地址时需要开启这个功能,具体步骤如下:

- 1.配置一个 ACL, 目的 IP 地址为接口 IP 地址, 目的端口为准备作为虚拟服务的端口
- 2.使用该 ACL 创建一个虚拟地址,
- 3.使用该虚拟地址创建虚拟服务

SLB 全局路径保持

获取 SLB 全局路径保持

Action: global.path_persist.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=global.path_persist.get

名称	类型	范围	含义
global_path_persist	整数	0, 1	SLB 全局路径保持, 0: 关闭, 1: 开启

响应参数:

响应举例:

```
{"global_path_persist":1}
```

设置 SLB 全局路径保持

Action: global.path_persist.set

请求参数:

名称	类型	范围	含义	必选	备注
global_path_persist	整数	0, 1	SLB 全局路径保持	是	0: 关闭, 1: 开启

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=global.path_persist.set

Body:

```
{"global_path_persist":1}
```

SLB 全局软关机设置

获取 SLB 全局软关机配置

Action: slb.graceful-shutdown.get

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.graceful-shutdown.get

响应参数:

名称	类型	范围	含义
time	整数	0, 65535	软关机超时时间, 0: 表示软关机状态不会超时强制结束
delete	整数	0, 1	删除 node 触发软关机, 0: 关闭, 1: 开启
disable	整数	0, 1	禁用 node 触发软关机, 0: 关闭, 1: 开启
persist	整数	0, 1	禁用 node 触发软关机后, 会话保持表继续有效, 0: 关闭, 1: 开启

响应举例:

```
{
  "graceful_shutdown_node" : {
    "time" : 10000,
    "delete" : 1,
    "disable" : 1,
    "persist" : 1
  }
}
```

设置 SLB 全局软关机配置

Action: slb.graceful-shutdown.set

请求参数:

名称	类型	范围	含义	必选	备注
time	整数	0, 65535	软关机超时时间	是	0: 表示软关机状态不会超时强制结束
delete	整数	0, 1	删除 node 触发软关机	是	0: 关闭, 1: 开启
disable	整数	0, 1	禁用 node 触发软关机	是	0: 关闭, 1: 开启
persist	整数	0, 1	禁用 node 触发软关机后, 会话保持表继续有效	是	0: 关闭, 1: 开启

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.graceful-shutdown.set

Body:

```
{
  "graceful_shutdown_node" : {
    "time" : 10000,
    "delete" : 1,
    "disable" : 1,
    "persist" : 1
  }
}
```

SLB 全局策略地址转换设置

获取 SLB 全局策略地址转换配置

Action: global.slb_snat_on_vip.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=global.slb_snat_on_vip.get

响应参数:

名称	类型	范围	含义
global_policy_snat	整数	0,1	0: 关闭, 1: 开启

响应举例:

```
{
  "global_policy_snat": 0
}
```

设置 SLB 全局策略地址转换配置

Action: global.slb_snat_on_vip.set

请求参数:

名称	类型	范围	含义
global_policy_snat	整数	0,1	0: 关闭, 1: 开启

请求举例:

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=global.slb_snat_on_vip.set

```

vip.set
POST
{
  "global_policy_snat": 1
}

```

虚拟服务

SLB 会话清除

Action: slb.session.clear

请求参数:无

请求举例:

GET

<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.session.clear>



Note

清除会话会导致当前会话中断,请谨慎操作

节点

节点(node)列表

Action: slb.node.list

请求参数:无

请求举例:

GET

<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.list>

响应参数:

名称	类型	范围	含义
tc_name	字符串	长度 1-191	流量控制模版名称
graceful_delete	整数	0, 1	删除节点触发软关机, 1 启用, 0 禁用,
graceful_disable	整数	0, 1	禁用节点触发软关机, 1 启用, 0 禁用,
graceful_persist	整数	0, 1	禁用节点触发软关机后会话保持表有效, 1 启用, 0 禁用,
graceful_time	整数	0-65535	软关机超时, 单位秒
name	字符串	长度 1-191	节点名称
host	字符串	长度 1-63	节点主机名或 IP 地址
weight	整数	1-100	节点权重,

healthcheck	字符串	长度 1-63	节点关联的健康检查名
status	整数	0, 1	节点使能状态, 1:启用, 0 禁用,
conn_limit	整数	1-6000000	节点连接限制, 6000000 表示不限制
desc_rserver	字符串	长度 1-191	描述
ports	数组		节点端口列表
port_number	整数	0-65534	节点端口的端口号
protocol	整数	0, 1	节点端口的协议, 0 tcp, 1 udp
status	整数	0, 1	节点端口使能状态, 1 启用, 0 禁用,
weight	整数	1-100	节点端口权重,
conn_limit	整数	1-6000000	节点端口连接限制, 8000000 表示不限制
phm_profile	字符串	长度 1-63	被动健康检查名称
healthcheck	字符串	长度 1-63	节点端口关联的健康检查名称
slow_start_type	整数	0, 1	暖启动类型; 0 表示连接限制, 1 表示连接速率限制
slow_start_recover	整数	0-3600	暖启动恢复时间
slow_start_rate	整数	0, 1	暖启动变化规则; 0 表示倍数增加, 1 表示线性增加
slow_start_from	整数	1-65535	暖启动初始量
slow_start_step	整数	2-10	暖启动增量
slow_start_interval	整数	1-60	暖启动间隔, 单位为秒
slow_start_interval_num	整数	1-100	暖启动间隔数
slow_start_tail	整数	1-65535	暖启动结束量

响应举例:

```
[
  {
    "tc_name": "",
    "graceful_time": 50,
    "graceful_delete": 1,
    "graceful_disable": 1,
    "graceful_persist": 1,
    "name": "192.168.1.100",
    "host": "192.168.1.100",
    "weight": 2,
    "healthcheck": "ping",
    "status": 1,
    "conn_limit": 123,
    "desc_rserver": "iamnode1",
    "ports": [
      {
        "port_number": 80,
        "protocol": 0,
        "status": 1,
        "weight": 12,
        "conn_limit": 8000000,
        "phm_profile": "default",
        "healthcheck": "ping"
      }
    ]
  }
]
```

```

    ]
  }
]

```

节点(node)获取

Action: slb.node.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	节点名称	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.get

请求 body

```

{
  "name": "192.168.1.100"
}

```

响应参数:

名称	类型	范围	含义
tc_name	字符串	长度 1-191	流量控制模版名称
graceful_delete	整数	0, 1	删除节点触发软关机, 1 启用, 0 禁用,
graceful_disable	整数	0, 1	禁用节点触发软关机, 1 启用, 0 禁用,
graceful_persist	整数	0, 1	禁用节点触发软关机后会话保持表有效, 1 启用, 0 禁用,
graceful_time	整数	0-65535	软关机超时, 单位秒
name	字符串	长度 1-191	节点名称
host	字符串	长度 1-63	节点主机名或 IP 地址
weight	整数	1-100	节点权重
healthcheck	字符串	长度 1-63	节点关联的健康检查名
status	整数	0, 1	节点使能状态, 1:启用, 0 禁用,
conn_limit	整数	1-6000000	节点连接限制, 6000000 表示不限制
desc_rserver	字符串	长度 1-191	描述
ports	数组		节点端口列表
port_number	整数	0-65534	节点端口的端口号
protocol	整数	0, 1	节点端口的协议, 0 tcp, 1 udp
status	整数	0, 1	节点端口使能状态, 1 启用, 0 禁用,
weight	整数	1-100	节点端口权重
conn_limit	整数	1-6000000	节点端口连接限制, 6000000 表示不限制
phm_profile	字符串	长度 1-63	被动健康检查名称
healthcheck	字符串	长度 1-63	节点端口关联的健康检查名称

slow_start_type	整数	0, 1	暖启动类型；0 表示连接限制，1 表示连接速率限制
slow_start_recover	整数	0-3600	暖启动恢复时间
slow_start_rate	整数	0, 1	暖启动变化规则；0 表示倍数增加，1 表示线性增加
slow_start_from	整数	1-65535	暖启动初始量
slow_start_step	整数	2-10	暖启动增量
slow_start_interval	整数	1-60	暖启动间隔，单位为秒
slow_start_interval_num	整数	1-100	暖启动间隔数
slow_start_tail	整数	1-65535	暖启动结束量

响应举例：

```
{
  "tc_name": "",
  "graceful_time": 50,
  "graceful_delete": 1,
  "graceful_disable": 1,
  "graceful_persist": 1,
  "name": "192.168.1.100",
  "host": "192.168.1.100",
  "weight": 1,
  "healthcheck": "",
  "status": 1,
  "conn_limit": 8000000,
  "desc_rserver": "",
  "ports": [{
    "port_number": 80,
    "protocol": 0,
    "status": 1,
    "weight": 1,
    "conn_limit": 8000000,
    "phm_profile": "",
    "healthcheck": ""
  }]
}
```

节点(node)添加

Action: slb.node.add

请求参数:

名称	类型	范围	含义	必选	备注
node	对象		节点对象	是	
tc_name	字符串	长度 1-191	流量控制模版名称	否	缺省值 空字符串"",表示没有流控
graceful_dele	整数	0, 1	删除节点触发软关机	否	1 启用, 0 禁用, 缺省值 0

te					
graceful_disable	整数	0, 1	禁用节点触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_persist	整数	0, 1	禁用节点触发软关机后会话保持表有效	否	1 启用, 0 禁用, 缺省值 0
graceful_time	整数	0-65535	软关机超时, 单位秒	否	缺省值 0, 0 表示永远不超时
name	字符串	长度 1-191	节点名称	是	唯一
host	字符串	长度 1-63	节点主机名或 IP 地址	是	
weight	整数	1-100	节点权重	否	缺省值 1
healthcheck	字符串	长度 1-63	节点关联的健康检查名	否	缺省值 空字符串 "", 表示没有检查
status	整数	0, 1	节点使能状态,	否	1: 启用, 0 禁用, 缺省值 1
conn_limit	整数	1-6000000	节点连接限制,	否	缺省值 6000000 表示不限制
desc_rserver	字符串	长度 1-191	描述	否	缺省值 空字符串 ""
slow_start_type	整数	0, 1	暖启动类型	否	0 表示连接限制, 1 表示连接速率限制; 缺省值: 0
slow_start_recover	整数	0-3600	暖启动恢复时间	否	缺省值: 15
slow_start_rate	整数	0, 1	暖启动变化规则	否	0 表示倍数增加, 1 表示线性增加; 缺省值: 0
slow_start_from	整数	1-65535	暖启动初始量	否	缺省值: 128
slow_start_step	整数	2-10	暖启动增量	否	缺省值: 2
slow_start_interval	整数	1-60	暖启动间隔, 单位为秒	否	缺省值: 10
slow_start_interval_num	整数	1-100	暖启动间隔数	否	缺省值: 6
slow_start_tail	整数	1-65535	暖启动结束量	否	缺省值: 4096

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.add

请求 body

```
{
  "node":{
    "tc_name": "",
    "graceful_time": 50,
    "graceful_delete": 1,
    "graceful_disable": 1,
    "graceful_persist": 1,
    "name": "192.168.1.101",
    "host": "192.168.1.101",
    "weight": 2,
    "healthcheck": "ping",
```

```

    "status": 1,
    "conn_limit": 123,
    "desc_rserver": "iamnode1"
  }
}

```

节点(node)编辑

Action: slb.node.edit

请求参数:

名称	类型	范围	含义	必选	备注
node	对象		节点对象	是	
tc_name	字符串	长度 1-191	流量控制模版名称	否	缺省值 空字符串"",表示没有流控
graceful_delete	整数	0, 1	删除节点触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_disable	整数	0, 1	禁用节点触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_persist	整数	0, 1	禁用节点触发软关机后会话保持表有效	否	1 启用, 0 禁用, 缺省值 0
graceful_time	整数	0-65535	软关机超时, 单位秒	否	缺省值 0, 0 表示永远不超时
name	字符串	长度 1-191	节点名称	是	节点必须存在
host	字符串	长度 1-63	节点主机名或 IP 地址	是	
weight	整数	1-100	节点权重	否	缺省值 1
healthcheck	字符串	长度 1-63	节点关联的健康检查名	否	缺省值 空字符串"",表示没有检查
status	整数	0, 1	节点使能状态,	否	1:启用, 0 禁用, 缺省值 1
conn_limit	整数	1-6000000	节点连接限制,	否	缺省值 6000000 表示不限制
desc_rserver	字符串	长度 1-191	描述	否	缺省值 空字符串""
slow_start_type	整数	0, 1	暖启动类型	否	0 表示连接限制, 1 表示连接速率限制; 缺省值: 0
slow_start_recover	整数	0-3600	暖启动恢复时间	否	缺省值: 15
slow_start_rate	整数	0, 1	暖启动变化规则	否	0 表示倍数增加, 1 表示线性增加; 缺省值: 0
slow_start_from	整数	1-65535	暖启动初始量	否	缺省值: 128
slow_start_step	整数	2-10	暖启动增量	否	缺省值: 2
slow_start_interval	整数	1-60	暖启动间隔, 单位为秒	否	缺省值: 10
slow_start_interval_num	整数	1-100	暖启动间隔数	否	缺省值: 6
slow_start_timeout	整数	1-65535	暖启动结束量	否	缺省值: 4096

il					
----	--	--	--	--	--

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.edit

请求 body

```
{
  "node":{
    "tc_name": "",
    "graceful_time": 50,
    "graceful_delete": 1,
    "graceful_disable": 1,
    "graceful_persist": 1,
    "name": "192.168.1.101",
    "host": "192.168.1.101",
    "weight": 2,
    "healthcheck": "ping",
    "status": 1,
    "conn_limit": 123,
    "desc_rserver": "iamnode1"
  }
}
```

节点(node)删除

Action: slb.node.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	节点名称	是	节点必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.del

请求 body

```
{
  "name": "192.168.1.123"
}
```

节点端口(node.port)添加

Action: slb.node.port.add

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	节点名称	是	节点必须存在
port	对象		节点对象	是	
port_number	整数	0-65534	节点端口的端口号	是	
protocol	整数	0, 1	节点端口的协议,	是	0 tcp, 1 udp
status	整数	0, 1	节点端口使能状态	否	1 启用, 0 禁用, 缺省值 1
weight	整数	1-100	节点端口权重,	否	缺省值 1
conn_limit	整数	1-6000000	节点端口连接限制	否	缺省值 6000000,表示不限制
graceful_delete	整数	0, 1	删除节点端口触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_disable	整数	0, 1	禁用节点端口触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_persist	整数	0, 1	禁用节点端口触发软关机后会话保持表有效	否	1 启用, 0 禁用, 缺省值 0
graceful_time	整数	0-65535	软关机超时, 单位秒	否	缺省值 0, 0 表示永远不超时
phm_profile	字符串	长度 1-63	被动健康检查名称	否	缺省值 空字符串"",表示没有检查
healthcheck	字符串	长度 1-63	节点端口关联的健康检查名称	否	缺省值 空字符串"",表示没有检查
nat_strategy	字符串	长度 1-191	策略名	否	缺省值 空字符串""

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.port.add

请求 body

```
{
  "name": "203.0.0.100",
  "port": {
    "port_number": 890,
    "protocol": 0,
    "status": 1,
    "weight": 1,
    "conn_limit": 8000000,
    "graceful_time": 0,
    "graceful_delete": 0,
    "graceful_disable": 0,
    "graceful_persist": 0,
    "phm_profile": "",
    "healthcheck": "",
    "nat_strategy": ""
  }
}
```

节点端口(node.port)编辑

Action: slb.node.port.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	节点名称	是	节点必须存在
port	对象		节点对象	是	
port_number	整数	0-65534	节点端口的端口号	是	
protocol	整数	0, 1	节点端口的协议,	是	0 tcp, 1 udp
status	整数	0, 1	节点端口使能状态	否	1 启用, 0 禁用, 缺省值 1
weight	整数	1-100	节点端口权重,	否	缺省值 1
conn_limit	整数	1-6000000	节点端口连接限制	否	缺省值 6000000,表示不限制
graceful_delete	整数	0, 1	删除节点端口触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_disable	整数	0, 1	禁用节点端口触发软关机	否	1 启用, 0 禁用, 缺省值 0
graceful_persist	整数	0, 1	禁用节点端口触发软关机后会话保持表有效	否	1 启用, 0 禁用, 缺省值 0
graceful_time	整数	0-65535	软关机超时, 单位秒	否	缺省值 0, 0 表示永远不超时
phm_profile	字符串	长度 1-63	被动健康检查名称	否	缺省值 空字符串"",表示没有检查
healthcheck	字符串	长度 1-63	节点端口关联的健康检查名称	否	缺省值 空字符串"",表示没有检查
nat_strategy	字符串	长度 1-191	策略名	否	缺省值 空字符串""

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=s1b.node.port.edit>
 请求 body

```
{
  "name": "203.0.0.100",
  "port": {
    "port_number": 890,
    "protocol": 0,
    "status": 1,
    "weight": 1,
    "conn_limit": 8000000,
    "graceful_time": 0,
    "graceful_delete": 0,
    "graceful_disable": 0,
    "graceful_persist": 0,
    "phm_profile": "",
    "healthcheck": "",
    "nat_strategy": ""
  }
}
```


节点端口(node.port)删除

Action: slb.node.port.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	节点名称	是	节点必须存在
port	对象		节点对象	是	
port_number	整数	0-65534	节点端口的端口号	是	
protocol	整数	0, 1	节点端口的协议,	是	0 tcp, 1 udp

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.port.del
请求 body

```
{  
  "name": "192.168.1.100",  
  "port": {  
    "port_number": 81,  
    "protocol": 0  
  }  
}
```

节点(node)状态列表

Action: slb.node.stat.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.stat.list
响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	节点名称
host	字符串	长度 1-63	节点主机名或 IP 地址
status	整数	0-4	节点状态: 0 禁用, 1 正常, 2 故障, 3 软关机, 4 暖启动
desc_rserver	字符串	长度 1-191	节点描述
current_conns	整数	>=0	节点当前并发连接数
total_conns	整数	>=0	节点累计连接连接数
send_packets	整数	>=0	节点发送包数
receive_packets	整数	>=0	节点接收包数
send_bytes	整数	>=0	节点发送字节数

receive_bytes	整数	>=0	节点接收字节数
send_rate	整数	>=0	节点发送速率 (bps)
receive_rate	整数	>=0	节点接收速率 (bps)
conn_rate	整数	>=0	节点新建连接速率 (conn/s)
current_request	整数	>=0	节点当前请求
total_request	整数	>=0	节点累计请求
total_request_succ	整数	>=0	节点累计成功请求
node_ports_stat	数组		由节点端口统计信息组成的数组
port_number	整数	0-65535	节点端口的端口号
protocol	整数	0, 1	节点端口的协议, 0 tcp, 1 udp
status	整数	0, 1	节点端口的状态, 1 启用, 0 禁用 p
current_conns	整数	>=0	节点端口当前并发连接数
total_conns	整数	>=0	节点端口累计连接连接数
send_packets	整数	>=0	节点端口发送包数
receive_packets	整数	>=0	节点端口接收包数
send_bytes	整数	>=0	节点端口发送字节数
receive_bytes	整数	>=0	节点端口接收字节数
send_rate	整数	>=0	节点端口发送速率 (bps)
receive_rate	整数	>=0	节点端口接收速率 (bps)
conn_rate	整数	>=0	节点端口新建连接速率 (conn/s)
current_request	整数	>=0	节点端口当前请求
total_request	整数	>=0	节点端口累计请求
total_request_succ	整数	>=0	节点端口累计成功请求

响应举例:

```

[[
  "name": "192.168.1.100",
  "host": "192.168.1.100",
  "status": 1,
  "desc_rserver": "",
  "current_conns": 0,
  "total_conns": 0,
  "send_packets": 0,
  "receive_packets": 0,
  "send_bytes": 0,
  "receive_bytes": 0,
  "send_rate": 0,
  "receive_rate": 0,
  "conn_rate": 0,
  "current_request": 0,
  "total_request": 0,
  "total_request_succ": 0,
  "node_ports_stat": [{
    "port_number": 80,
    "protocol": 0,
    "status": 1,

```

```

    "current_conns": 0,
    "total_conns": 0,
    "send_rate": 0,
    "receive_rate": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "current_request": 0,
    "total_request": 0,
    "total_request_succ": 0
  }}
},
{
  "name": "192.168.1.101",
  "host": "192.168.1.101",
  "status": 1,
  "desc_rserver": "",
  "current_conns": 0,
  "total_conns": 0,
  "send_packets": 0,
  "receive_packets": 0,
  "send_bytes": 0,
  "receive_bytes": 0,
  "send_rate": 0,
  "receive_rate": 0,
  "conn_rate": 0,
  "current_request": 0,
  "total_request": 0,
  "total_request_succ": 0,
  "node_ports_stat": [{
    "port_number": 80,
    "protocol": 0,
    "status": 1,
    "current_conns": 0,
    "total_conns": 0,
    "send_rate": 0,
    "receive_rate": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "current_request": 0,
    "total_request": 0,
    "total_request_succ": 0
  ]}
}

```

```
}  
]
```

节点(node)状态获取

Action: `slb.node.stat.get`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	节点名称	是	节点必须存在

请求举例:

POST

`http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.node.stat.get`

请求 body:

```
{  
  "name": "192.168.1.100"  
}
```

响应参数:

名称	类型	范围	含义
<code>name</code>	字符串	长度 1-191	节点名称
<code>host</code>	字符串	长度 1-63	节点主机名或 IP 地址
<code>status</code>	整数	0-4	节点状态: 0 禁用, 1 正常, 2 故障, 3 软关机, 4 暖启动
<code>desc_rserver</code>	字符串	长度 1-191	节点描述
<code>current_conns</code>	整数	>=0	节点当前并发连接数
<code>total_conns</code>	整数	>=0	节点累计连接连接数
<code>send_packets</code>	整数	>=0	节点发送包数
<code>receive_packets</code>	整数	>=0	节点接收包数
<code>send_bytes</code>	整数	>=0	节点发送字节数
<code>receive_bytes</code>	整数	>=0	节点接收字节数
<code>send_rate</code>	整数	>=0	节点发送速率 (bps)
<code>receive_rate</code>	整数	>=0	节点接收速率 (bps)
<code>conn_rate</code>	整数	>=0	节点新建连接速率 (conn/s)
<code>current_request</code>	整数	>=0	节点当前请求
<code>total_request</code>	整数	>=0	节点累计请求
<code>total_request_succ</code>	整数	>=0	节点累计成功请求
<code>node_ports_stat</code>	数组		由节点端口统计信息组成的数组
<code>port_number</code>	整数	0-65535	节点端口的端口号
<code>protocol</code>	整数	0, 1	节点端口的协议, 0 tcp, 1 udp
<code>status</code>	整数	0, 1	节点端口的状态, 1 启用, 0 禁用 p
<code>current_conns</code>	整数	>=0	节点端口当前并发连接数
<code>total_conns</code>	整数	>=0	节点端口累计连接连接数

send_packets	整数	>=0	节点端口发送包数
receive_packets	整数	>=0	节点端口接收包数
send_bytes	整数	>=0	节点端口发送字节数
receive_bytes	整数	>=0	节点端口接收字节数
send_rate	整数	>=0	节点端口发送速率 (bps)
receive_rate	整数	>=0	节点端口接收速率 (bps)
conn_rate	整数	>=0	节点端口新建连接速率 (conn/s)
current_request	整数	>=0	节点端口当前请求
total_request	整数	>=0	节点端口累计请求
total_request_succ	整数	>=0	节点端口累计成功请求

响应举例:

```
{
  "name": "192.168.1.100",
  "host": "192.168.1.100",
  "status": 1,
  "desc_rserver": "",
  "current_conns": 0,
  "total_conns": 0,
  "send_packets": 0,
  "receive_packets": 0,
  "send_bytes": 0,
  "receive_bytes": 0,
  "send_rate": 0,
  "receive_rate": 0,
  "conn_rate": 0,
  "current_request": 0,
  "total_request": 0,
  "total_request_succ": 0,
  "node_ports_stat": [{
    "port_number": 80,
    "protocol": 0,
    "status": 1,
    "current_conns": 0,
    "total_conns": 0,
    "send_rate": 0,
    "receive_rate": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "current_request": 0,
    "total_request": 0,
    "total_request_succ": 0
  ]
}
```

```
}  
}
```

节点状态清除

Action: slb.node.stat.clear

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.node.stat.clear



此 API 会清除所有节点的统计信息

服务池

服务池(pool)列表

Action: slb.pool.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务池名称
protocol	整数	0, 1	服务池关联协议, 0 tcp, 1 udp
lb_method	整数	0-18, 20, 22-24	服务池关联算法: 0: 轮询; 1: 加权轮询; 2: 节点最少连接; 3: 加权节点最少连接; 4: 服务最少连接 5: 加权服务最少连接; 6: 最快响应 7: 最少请求; 8: 精确轮询; 9: 无状态源 IP 端口哈希; 10: 无状态源 IP 哈希; 11: 无状态目的 IP 端口哈希; 12: 无状态源 IP 和目的 IP 端口哈希; 13: 无状态每包轮询; 14: 最小带宽; 15: 源地址哈希; 16: 源地址和端口哈希; 17: 目的地址哈希; 18: 目的地址和端口哈希; 20: 随机; 22: 基于 IPList; 23: 无状态加权每包轮询; 24: 严格加权轮询;
healthcheck	字符串	长度 1-63	服务池关联的健康检查名称
desc_pool	字符串	长度 1-191	描述
up-members-at-least	对象		优先级规则配置
status	整数	0, 1	num 大于 0 时为 1, num 等于 0 时为 0

num	整数	0-63	最少正常成员
type	整数	0-2	0 禁用, 1 仅使用低优先级成员, 2 动态使用低优先级成员
aux-node-log	整数	0, 1	1 开启优先级日志, 0 关闭
members	数组		服务池成员对象组成的数组
nodename	字符串	长度 1-191	服务池成员使用的节点名称
server	字符串	长度 1-63	服务池成员使用的节点主机名或者 IP 地址
port	整数	0-65534	服务池成员使用的节点端口
priority	整数	1-16	服务池成员的优先级
status	整数	0, 1	服务池成员是否启用

响应举例:

```
[
  {
    "name": "pool",
    "protocol": 0,
    "lb_method": 0,
    "healthcheck": "ping",
    "desc_pool": "iampool",
    "up-members-at-least": {
      "status": 1,
      "num": 32,
      "type": 2
    },
    "aux-node-log": 1,
    "members": [
      {
        "nodename": "192.168.1.100",
        "server": "192.168.1.100",
        "port": 80,
        "priority": 5,
        "status": 1
      }
    ]
  }
]
```

服务池(pool)获取

Action: slb.pool.get

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	服务池名称	是	服务池必须存在
-------------	-----	----------	-------	---	---------

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.get
请求 body

```
{
  "name": "pool"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务池名称
protocol	整数	0, 1	服务池的协议, 0 tcp, 1 udp
lb_method	整数	0-18, 20, 22-24	服务池关联算法: 0: 轮询; 1: 加权轮询; 2: 节点最少连接; 3: 加权节点最少连接; 4: 服务最少连接 5: 加权服务最少连接; 6: 最快响应 7: 最少请求; 8: 精确轮询; 9: 无状态源 IP 端口哈希; 10: 无状态源 IP 哈希; 11: 无状态目的 IP 端口哈希; 12: 无状态源 IP 和目的 IP 端口哈希; 13: 无状态每包轮询; 14: 最小带宽; 15: 源地址哈希; 16: 源地址和端口哈希; 17: 目的地址哈希; 18: 目的地址和端口哈希; 20: 随机; 22: 基于 IPList; 23: 无状态加权每包轮询; 24: 严格加权轮询;
healthcheck	字符串	长度 1-63	服务池关联的健康检查名称
desc_pool	字符串	长度 1-191	描述
up-members-at-least	对象		优先级规则配置
status	整数	0, 1	num 大于 0 时为 1, num 等于 0 时为 0
num	整数	1-63	最少正常成员
type	整数	0-2	0 禁用, 1 仅使用低优先级成员, 2 动态使用低优先级成员
aux-node-log	整数	0, 1	1 开启优先级日志, 0 关闭
members	数组		服务池成员对象组成的数组
nodename	字符串	长度 1-191	服务池成员使用的节点名称
server	字符串	长度 1-63	服务池成员使用的节点主机名或者 IP 地址
port	整数	0-65534	服务池成员使用的节点端口
priority	整数	1-16	服务池成员的优先级
status	整数	0, 1	服务池成员是否启用

响应举例:

```
{
  "name": "pool",
  "protocol": 0,
  "lb_method": 0,
  "healthcheck": "ping",
```



```

"desc_pool": "iampool",
"up-members-at-least": {
  "status": 1,
  "num": 32,
  "type": 2
},
"aux-node-log": 1,
"members": [
  {
    "nodename": "192.168.1.100",
    "server": "192.168.1.100",
    "port": 80,
    "priority": 5,
    "status": 1
  }
]
}

```

服务池(pool)添加

Action: slb.pool.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	唯一
protocol	整数	0, 1	服务池协议,	是	0 tcp, 1 udp, 缺省值: 0
lb_method	整数	0-18, 20, 22-24	服务池算法:	否	服务池关联算法: 0: 轮询; 1: 加权轮询; 2: 节点最少连接; 3: 加权节点最少连接; 4: 服务最少连接 5: 加权服务最少连接; 6: 最快响应 7: 最少请求; 8: 精确轮询; 9: 无状态源 IP 端口哈希; 10: 无状态源 IP 哈希; 11: 无状态目的 IP 端口哈希; 12: 无状态源 IP 和目的 IP 端口哈希; 13: 无状态每包轮询; 14: 最小带宽; 15: 源地址哈希; 16: 源地址和端口哈希; 17: 目的地址哈希; 18: 目的地址和端口哈希; 20: 随机; 22: 基于

					IPList; 23: 无状态加权每包轮询; 24: 严格加权轮询; 缺省值: 0
healthcheck	字符串	长度 1-63	健康检查名称	否	缺省值 空字符串"",表示没有检查
desc_pool	字符串	长度 1-191	描述	否	缺省值 空字符串""
up-members-at-least	对象		优先级规则配置	否	
status	整数	0, 1	num 大于 0 时为 1, num 等于 0 时为 0	否	缺省值: 0
num	整数	1-63	最少正常成员	否	缺省值: 1
type	整数	0-2	优先级	否	0 禁用, 1 仅使用低优先级成员, 2 动态使用低优先级成员 缺省值: 0
aux-node-log	整数	0, 1	1 开启优先级日志, 0 关闭	否	缺省值: 0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.add
请求 body

```
{
  "pool":{
    "name": "pool2",
    "protocol": 0,
    "lb_method": 7,
    "healthcheck": "ping",
    "desc_pool": "iampool",
    "up-members-at-least": {
      "status": 1,
      "num": 32,
      "type": 2
    },
    "aux-node-log": 1
  }
}
```

服务池(pool)编辑

Action: slb.pool.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在
protocol	整数	0, 1	服务池协议,	是	0 tcp, 1 udp, 缺省值: 0

lb_method	整数	0-18, 20, 22-24	服务池算法:	否	服务池关联算法: 0: 轮询; 1: 加权轮询; 2: 节点最少连接; 3: 加权节点最少连接; 4: 服务最少连接 5: 加权服务最少连接; 6: 最快响应 7: 最少请求; 8: 精确轮询; 9: 无状态源 IP 端口哈希; 10: 无状态源 IP 哈希; 11: 无状态目的 IP 端口哈希; 12: 无状态源 IP 和目的 IP 端口哈希; 13: 无状态每包轮询; 14: 最小带宽; 15: 源地址哈希; 16: 源地址和端口哈希; 17: 目的地址哈希; 18: 目的地址和端口哈希; 20: 随机; 22: 基于 IPList; 23: 无状态加权每包轮询; 24: 严格加权轮询; 缺省值: 0
healthcheck	字符串	长度 1-63	健康检查名称	否	缺省值 空字符串 "",表示没有检查
desc_pool	字符串	长度 1-191	描述	否	缺省值 空字符串 ""
up-members-at-least	对象		优先级规则配置	否	
status	整数	0, 1	num 大于 0 时为 1, num 等于 0 时为 0	否	缺省值: 0
num	整数	1-63	最少正常成员	否	缺省值: 0
type	整数	0-2	优先级	否	0 禁用, 1 仅使用低优先级成员, 2 动态使用低优先级成员 缺省值: 0
aux-node-log	整数	0, 1	1 开启优先级日志, 0 关闭	否	缺省值: 0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=s1b.pool.edit

请求 body

```
{
  "pool":{
    "name": "pool2",
    "protocol": 0,
    "lb_method": 7,
    "healthcheck": "ping",
    "desc_pool": "iampool",
    "up-members-at-least": {
```

```

        "status": 1,
        "num": 32,
        "type": 2
    },
    "aux-node-log": 1
}
}

```

服务池(pool)删除

Action: slb.pool.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.del

请求 body

```

{
    "name": "pool2"
}

```

服务池成员(pool.member)添加

Action: slb.pool.member.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在
member	对象		服务池成员对象	是	
server	字符串	长度 1-63	服务池成员使用的节点主机名或者 IP 地址	是	
port	整数	0-65534	服务池成员使用的节点端口	是	
priority	整数	1-16	服务池成员的优先级	否	缺省值:1
status	整数	0, 1	服务池成员是否启用	否	缺省值:1
cookie	字符串	长度 1-191	服务池名称	否	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.member.add

请求 body

```
{
  "name": "pool",
  "member":
  {
    "server": "192.168.1.100",
    "port": 81,
    "priority": 15,
    "status": 1
  }
}
```

服务池成员(pool.member)编辑

Action: slb.pool.member.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在
member	对象		服务池成员对象	是	
server	字符串	长度 1-63	服务池成员使用的节点主机名或者 IP 地址	是	
port	整数	0-65534	服务池成员使用的节点端口	是	
priority	整数	1-16	服务池成员的优先级	否	缺省值:1
status	整数	0, 1	服务池成员是否启用	否	缺省值:1
cookie	字符串	长度 1-191	服务池名称	否	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.member.edit

请求 body

```
{
  "name": "pool",
  "member":
  {
    "server": "192.168.1.100",
    "port": 81,
    "priority": 15,
    "status": 1
  }
}
```

服务池成员(pool.member)删除

Action: slb.pool.member.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在
member	对象		服务池成员对象	是	
server	字符串	长度 1-63	服务池成员使用的节点主机名或者 IP 地址	是	
port	整数	0-65534	服务池成员使用的节点端口	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.member.del

请求 body

```
{
  "name": "pool",
  "member":
  {
    "server": "192.168.1.100",
    "port": 81
  }
}
```

服务池(pool)状态列表

Action: slb.pool.stat.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.stat.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务池名称
protocol	整数	0, 1	服务池关联协议, 0 tcp, 1 udp
status	整数	0-4	服务池的状态: 0 禁用, 1 正常, 2 部分正常, 3 故障, 4 unknown
desc_pool	字符串	长度 1-191	描述
current_conns	整数	>=0	服务池当前并发连接数

total_conns	整数	>=0	服务池累计连接连接数
send_packets	整数	>=0	服务池发送包数
receive_packets	整数	>=0	服务池接收包数
send_bytes	整数	>=0	服务池发送字节数
receive_bytes	整数	>=0	服务池接收字节数
request_current	整数	>=0	服务池当前请求
request_total	整数	>=0	服务池累计请求
request_success	整数	>=0	服务池累计成功请求
members_stat	数组		由服务池成员状态统计信息对象组成的数组
server	字符串	长度 1-63	服务池成员使用的节点主机名或 IP 地址
mem_nodename	字符串	长度 1-191	服务池成员使用的节点名称
port	整数	0-65534	服务池成员使用的节点端口
status	整数	0-2	服务池成员状态：0 禁用，1 正常，2 故障
response_time_hc	整数	>=0	服务池成员健康检查响应时间
response_time_http	整数	>=0	服务池成员 http 协议响应时间
current_conns	整数	>=0	服务池成员当前并发连接数
total_conns	整数	>=0	服务池成员累计连接连接数
send_packets	整数	>=0	服务池成员发送包数
receive_packets	整数	>=0	服务池成员接收包数
send_bytes	整数	>=0	服务池成员发送字节数
receive_bytes	整数	>=0	服务池成员接收字节数
request_current	整数	>=0	服务池成员当前请求
request_total	整数	>=0	服务池成员累计请求
request_success	整数	>=0	服务池成员累计成功请求

响应举例：

```
[
  {
    "name": "pool",
    "protocol": 0,
    "status": 1,
    "desc_pool": "iampool",
    "current_conns": 0,
    "total_conns": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "request_current": 0,
    "request_total": 0,
    "request_success": 0,
    "members_stat": [
      {
        "server": "192.168.1.100",
        "port": 80,
        "status": 1,

```

```

    "current_conns": 0,
    "total_conns": 0,
    "mem_nodename": "192.168.1.100",
    "send_packets": 0,
    "receive_packets": 0,
    "response_time_hc": 112000,
    "response_time_http": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "request_current": 0,
    "request_total": 0,
    "request_success": 0
  }
]
}
]

```

服务池(pool)状态获取

Action: slb.pool.stat.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务池名称	是	服务池必须存在

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.pool.stat.get>
 请求 body

```

{
  "name": "pool"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务池名称
protocol	整数	0, 1	服务池关联协议, 0 tcp, 1 udp
status	整数	0-4	服务池的状态: 0 禁用, 1 正常, 2 部分正常, 3 故障, 4 unknown
desc_pool	字符串	长度 1-191	描述
current_conns	整数	>=0	服务池当前并发连接数
total_conns	整数	>=0	服务池累计连接连接数
send_packets	整数	>=0	服务池发送包数

receive_packets	整数	>=0	服务池接收包数
send_bytes	整数	>=0	服务池发送字节数
receive_bytes	整数	>=0	服务池接收字节数
request_current	整数	>=0	服务池当前请求
request_total	整数	>=0	服务池累计请求
request_success	整数	>=0	服务池累计成功请求
members_stat	数组		由服务池成员状态统计信息对象组成的数组
server	字符串	长度 1-63	服务池成员使用的节点主机名或 IP 地址
mem_nodename	字符串	长度 1-191	服务池成员使用的节点名称
port	整数	0-65534	服务池成员使用的节点端口
status	整数	0-2	服务池成员状态：0 禁用，1 正常，2 故障
response_time_hc	整数	>=0	服务池成员健康检查响应时间
response_time_http	整数	>=0	服务池成员 http 协议响应时间
current_conns	整数	>=0	服务池成员当前并发连接数
total_conns	整数	>=0	服务池成员累计连接连接数
send_packets	整数	>=0	服务池成员发送包数
receive_packets	整数	>=0	服务池成员接收包数
send_bytes	整数	>=0	服务池成员发送字节数
receive_bytes	整数	>=0	服务池成员接收字节数
request_current	整数	>=0	服务池成员当前请求
request_total	整数	>=0	服务池成员累计请求
request_success	整数	>=0	服务池成员累计成功请求

响应举例：

```
{
  "name": "pool",
  "protocol": 0,
  "status": 1,
  "desc_pool": "iampool",
  "current_conns": 0,
  "total_conns": 0,
  "send_packets": 0,
  "receive_packets": 0,
  "send_bytes": 0,
  "receive_bytes": 0,
  "request_current": 0,
  "request_total": 0,
  "request_success": 0,
  "members_stat": [
    {
      "server": "192.168.1.100",
      "port": 80,
      "status": 1,
      "current_conns": 0,
      "total_conns": 0,

```

```

    "mem_nodename": "192.168.1.100",
    "send_packets": 0,
    "receive_packets": 0,
    "response_time_hc": 112000,
    "response_time_http": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "request_current": 0,
    "request_total": 0,
    "request_success": 0
  }
]
}

```

服务池状态清除

Action: `slb.pool.stat.clear`

请求参数:无

请求举例:

GET

`http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.pool.stat.clear`



Note

此 API 会清除所有 pool 的统计信息

虚拟地址

虚拟地址列表

Action: `slb.va.list`

请求参数:无

请求举例:

GET

`http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.list`

响应参数:

名称	类型	范围	含义
<code>tc_name</code>	字符串	长度 1-191	虚拟地址关联的流量控制
<code>name</code>	字符串	长度 1-191	虚拟地址名称
<code>address</code>	字符串	长度 1-63	ipv4/v6 类型的虚拟地址才有此参数

subnet	对象		子网类型的虚拟地址才有此参数
address	IP		子网类型的虚拟地址的子网地址
mask_len	整数	16-32	子网类型的虚拟地址的子网掩码长度
acl_id	整数	1-199	IPv4 ACL 类型的虚拟地址才有此参数
acl_name	字符串	长度 1-191	IPv6 ACL 类型的虚拟地址才有此参数
status	整数	0, 1	虚拟地址状态:1:启用; 0:禁用
arp_status	整数	0, 1	ARP 状态 1:启用; 0:禁用
icmp_probe	整数	0-3	PING vIP 地址 0:always; 1:disable; 2:all; 3:any
vrid	整数	0-8	vrrp 组 id,
redistribution	整数	0, 1	路由联动:1:启用; 0:禁用
policy_profile	字符串	长度 1-191	策略模版
virtual_services	数组		该虚拟地址下的虚拟服务列表
name	字符串	长度 1-63	虚拟服务名称
protocol	整数		虚拟服务类型:含义: 2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务端口
pool	字符串	长度 1-63	虚拟服务的服务池
connection_limit	对象		虚拟服务的并发连接限制对象
status	整数	0, 1	虚拟服务的并发连接限制开关:1:启用; 0:禁用
connection_limit_number	整数	1-8000000	虚拟服务的并发连接限制数值, 仅当开启虚拟服务的并发连接限制开关为 1 时有效
vs_enable_intf	字符串	长度 1-63	使能接口号,ETH 开头, 逗号分隔
path_persist	整数	0, 1	路径保持:1:启用; 0:禁用
status	整数	0, 1	虚拟服务状态:1:启用; 0:禁用
desc_vport	字符串	长度 1-191	虚拟服务描述
snat_on_vip	整数	0, 1	使能全局 snat 映射:1:启用; 0:禁用
auto_snat	整数	0, 1	接口 snat:1:启用; 0:禁用

snat_port_preserve_enable	整数	0, 1	源端口保持:1:启用; 0:禁用
snat_port_preserve_type	整数	0, 1	源端口保持类型:1:强制; 0:尝试, 仅当 snat_port_preserve_enable 参数为 1 时有效
vs_acl_id	整数	0-199	访问列表,1-199 表示对应 ACL, 0 表示没有配置
aclnamev6	字符串	长度 1-191	ipv6 访问列表名称
erules	数组		关联的 erule 名称组成的列表
send_reset	整数	0, 1	选择节点失败发送 rst:1:启用; 0:禁用
source_nat	字符串	长度 1-63	源 snat 地址池
srcip_persist	字符串	长度 1-191	源地址保持模版,一个虚拟服务只能有一种连接保持模板
dstip_persist	字符串	长度 1-191	目的地址保持模版,一个虚拟服务只能有一种连接保持模板
cookie_persist	字符串	长度 1-191	cookie 保持模版,一个虚拟服务只能有一种连接保持模板
sslid_persist	字符串	长度 1-191	SSL 连接保持模版,一个虚拟服务只能有一种连接保持模板
policy_profile	字符串	长度 1-191	策略模版
aclsnats	数组		策略地址转换
acl_id	整数	1-199	策略地址转换-acl id,
nat_pool	字符串	长度 1-63	策略地址转换-snat 池
connection_mirror	整数	0, 1	连接镜像:1:启用; 0:禁用
no_dest_nat	整数	0, 1	直接转发:1:启用; 0:禁用
syncookie	整数	0, 1	syn cookie:1:启用; 0:禁用
udp_profile	字符串	长度 1-191	udp 模版
dns_profile	字符串	长度 1-191	dns 模版
tcp_profile	字符串	长度 1-191	tcp 模版
waf_profile	字符串	长度 1-191	waf 模版
http_profile	字符串	长度 1-191	http 模版
connmulti_profile	字符串	长度 1-191	连接复用模版
cache_profile	字符串	长度 1-191	缓存模版
tcpagent_profile	字符串	长度 1-191	快速四层模版
serverssl_profile	字符串	长度 1-191	服务器 ssl 模版
connmulti_profile	字符串	长度 1-191	连接复用模版
clientssl_profile	字符串	长度 1-191	客户端 ssl 模版
rtsp_profile	字符串	长度 1-191	rtsp 模版
smtp_profile	字符串	长度 1-191	smtp 模版
sip_profile	字符串	长度 1-191	sip 模版
udp_profile	字符串	长度 1-191	udp 模版
l4_profile_type	整数	2, 3	运输层协议, 2 tcp, 3 udp,

响应举例:

```
[
  {
    "tc_name": "",
    "name": "1.1.1.1_va",
```

```

"address": "1.1.1.1",
"status": 1,
"arp_status": 1,
"icmp_disable": 0,
"vrid": 0,
"redistribution": 0,
"policy_profile": "",

"virtual_services": [
  {
    "name": "dnsvs",
    "protocol": 22,
    "port": 53,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "ETH0/1,ETH0/3",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "snat_port_preserve_enable": 0,
    "snat_port_preserve_type": 0,
    "vs_acl_id": 101,
    "aclnamev6": "aclv6",
    "rules": [
      "erule-empty.txt"
    ],
    "send_reset": 0,
    "source_nat": "",
    "udp_profile": "",
    "srcip_persist": "",
    "dns_profile": "",
    "policy_profile": "",
    "aclsnats": [
      {
        "acl_id": 101,
        "nat_pool": "snatpool"
      }
    ]
  }
]
},

```

```

{
  "tc_name": "",
  "name": "2.2.2.2_va",
  "address": "2.2.2.2",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "ftpv",
      "protocol": 9,
      "port": 21,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "connection_mirror": 0,
      "no_dest_nat": 0,
      "syncookie": {
        "syncookie": 0
      },
      "source_nat": "",
      "tcp_profile": "",
      "srcip_persist": "",
      "policy_profile": "",
      "aclsnats": []
    }
  ]
},
{
  "tc_name": "",

```

```

"name": "3.3.3.3_24_va",
"subnet": {
  "address": "3.3.3.3",
  "mask_len": 24
},
"status": 1,
"arp_status": 1,
"icmp_disable": 0,
"icmp_disable": 0,
"vrid": 0,
"redistribution": 0,
"policy_profile": "",
"virtual_services": [
  {
    "name": "https",
    "protocol": 14,
    "port": 80,
    "pool": "pool",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "snat_port_preserve_enable": 0,
    "snat_port_preserve_type": 0,
    "aclnamev6": "",
    "erules": [
      "erule-empty.txt"
    ],
    "send_reset": 0,
    "source_nat": "",
    "no_dest_nat": 0,
    "waf_profile": "",
    "http_profile": "",
    "tcp_profile": "",
    "connmulti_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
]

```

```

},
{
  "tc_name": "",
  "name": "ACL_ID_101_va",
  "acl_id": 101,
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "httpenvs",
      "protocol": 12,
      "port": 80,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "1,2",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,
      "snat_port_preserve_enable": 0,
      "snat_port_preserve_type": 0,
      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "syncookie": {
        "syncookie": 0
      },
      "source_nat": "",
      "no_dest_nat": 0,
      "waf_profile": "",
      "http_profile": "",
      "cache_profile": "",
      "tcpagent_profile": "",
      "serverssl_profile": "",
      "connmulti_profile": "",
      "srcip_persist": "",
      "policy_profile": "",
      "aclsnats": []
    }
  ]
}

```



```

    }
  ]
},
{
  "tc_name": "",
  "name": "ACL_ID_aclv6_va",
  "acl_name": "aclv6",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "httpsvs",
      "protocol": 15,
      "port": 443,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "aclv6",
      "erules": [],
      "send_reset": 0,
      "syncookie": {
        "syncookie": 0
      },
      "source_nat": "",
      "no_dest_nat": 0,
      "waf_profile": "",
      "http_profile": "",
      "cache_profile": "",
      "tcpagent_profile": "",
      "clientssl_profile": "",
      "serverssl_profile": "",
      "connmulti_profile": "",

```

```

        "srcip_persist": "",
        "policy_profile": "",
        "aclsnats": []
    }
]
},
{
    "tc_name": "",
    "name": "0.0.0.0_va",
    "address": "0.0.0.0",
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": "",
    "virtual_services": [
        {
            "name": "rtspvs",
            "protocol": 8,
            "port": 554,
            "pool": "",
            "connection_limit": {
                "status": 0,
                "connection_limit_number": 8000000
            },
            "vs_enable_intf": "2",
            "path_persist": 1,
            "status": 1,
            "desc_vport": "",
            "snat_on_vip": 0,
            "auto_snat": 0,

            "aclnamev6": "",
            "erules": [],
            "send_reset": 0,
            "no_dest_nat": 0,
            "syncookie": {
                "syncookie": 0
            },
            "tcp_profile": "",
            "rtsp_profile": "",
            "srcip_persist": "",
            "policy_profile": "",
            "aclsnats": []
        }
    ]
}

```

```

    }
  ]
},
{
  "tc_name": "",
  "name": "6.6.6.6_va",
  "address": "6.6.6.6",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "smtpvs",
      "protocol": 17,
      "port": 25,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "syncookie": {
        "syncookie": 0
      },
      "source_nat": "",
      "tcpagent_profile": "",
      "clientsl_profile": "",
      "smtp_profile": "",
      "srcip_persist": "",
      "policy_profile": "",
      "aclsnats": []
    }
  ]
}
]

```

```

},
{
  "tc_name": "",
  "name": "7.7.7.7_va",
  "address": "7.7.7.7",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "sipvs",
      "protocol": 11,
      "port": 5060,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "connection_mirror": 0,
      "udp_profile": "",
      "srcip_persist": "",
      "sip_profile": "",
      "dns_profile": "",
      "policy_profile": "",
      "aclsnats": []
    }
  ]
},
{
  "tc_name": "",
  "name": "4.4.4.4_va",
  "address": "4.4.4.4",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,

```

```

"vrid": 0,
"redistribution": 0,
"policy_profile": "",
"virtual_services": [
  {
    "name": "siptcpvs",
    "protocol": 18,
    "port": 5060,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,

    "aclnamev6": "",
    "erules": [],
    "send_reset": 0,
    "syncookie": {
      "syncookie": 0
    },
    "source_nat": "",
    "serverssl_profile": "",
    "connmulti_profile": "",
    "tcpagent_profile": "",
    "srcip_persist": "",
    "sip_profile": "",
    "policy_profile": "",
    "aclsnats": []
  }
],
{
  "tc_name": "",
  "name": "5.5.5.5_va",
  "address": "5.5.5.5",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,

```

```

"redistribution": 0,
"policy_profile": "",
"virtual_services": [
  {
    "name": "sslagentvs",
    "protocol": 16,
    "port": 443,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,

    "aclnamev6": "",
    "erules": [],
    "send_reset": 0,
    "syncookie": {
      "syncookie": 0
    },
    "source_nat": "",
    "tcpagent_profile": "",
    "clientssl_profile": "",
    "serverssl_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
],
{
  "tc_name": "",
  "name": "8.8.8.8_va",
  "address": "8.8.8.8",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",

```

```

"virtual_services": [
  {
    "name": "tcpvs",
    "protocol": 2,
    "port": 80,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "snat_port_preserve_enable": 0,
    "snat_port_preserve_type": 0,
    "aclnamev6": "",
    "erules": [],
    "send_reset": 0,
    "connection_mirror": 0,
    "no_dest_nat": 0,
    "syncookie": {
      "syncookie": 0
    },
    "source_nat": "",
    "tcp_profile": "",
    "srcip_persist": "",
    "sslid_persis": ""
    "policy_profile": "",
    "aclsnats": []
  }
],
{
  "tc_name": "",
  "name": "9.9.9.9_va",
  "address": "9.9.9.9",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [

```

```

{
  "name": "tcpagentvs",
  "protocol":20,
  "port": 80,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "snat_port_preserve_enable": 0,
  "snat_port_preserve_type": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "fixup_ftp": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "tcpagent_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
],
{
  "tc_name": "",
  "name": "10.10.10.10_va",
  "address": "10.10.10.10",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "tcpexchangevs",
      "protocol": 25,

```



```

    "port": 80,
    "pool": "",
    "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,

    "aclnamev6": "",
    "erules": [],
    "source_nat": "",
    "connmulti_profile": "",
    "tcpagent_profile": "",
    "policy_profile": "",
    "aclsnats": []
}
]
},
{
    "tc_name": "",
    "name": "12.12.12.12_va",
    "address": "12.12.12.12",
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": "",
    "virtual_services": [
        {
            "name": "mblbvs",
            "protocol": 26,
            "port": 80,
            "pool": "",
            "connection_limit": {
                "status": 0,
                "connection_limit_number": 8000000
            },
            "vs_enable_intf": "",
            "path_persist": 1,

```

```

    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,

    "aclnamev6": "",
    "erules": [],
    "source_nat": "",
    "connmulti_profile": "",
    "tcpagent_profile": "",
    "policy_profile": "",
    "aclsnats": []
  }
]
},
{
  "tc_name": "",
  "name": "13.13.13.13_va",
  "address": "13.13.13.13",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "tftpvs",
      "protocol": 23,
      "port": 69,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "",
      "erules": [],

```

```

    "connection_mirror": 0,
    "no_dest_nat": 0,
    "source_nat": "",
    "udp_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
]
},
{
  "tc_name": "",
  "name": "14.14.14.14_va",
  "address": "14.14.14.14",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "radiusvs",
      "protocol": 28,
      "port": 1813,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "",
      "erules": [],
      "connection_mirror": 0,
      "no_dest_nat": 0,
      "source_nat": "",
      "udp_profile": "",
      "srcip_persist": "",
      "dns_profile": "",

```

```

        "policy_profile": "",
        "aclsnats": []
    }
]
},
{
    "tc_name": "",
    "name": "15.15.15.15_va",
    "address": "15.15.15.15",
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": "",
    "virtual_services": [
        {
            "name": "udpvs",
            "protocol": 3,
            "port": 8000,
            "pool": "",
            "connection_limit": {
                "status": 0,
                "connection_limit_number": 8000000
            },
            "vs_enable_intf": "",
            "path_persist": 1,
            "status": 1,
            "desc_vport": "",
            "snat_on_vip": 0,
            "auto_snat": 0,
            "snat_port_preserve_enable": 0,
            "snat_port_preserve_type": 0,
            "aclnamev6": "",
            "erules": [],
            "connection_mirror": 0,
            "no_dest_nat": 0,
            "source_nat": "",
            "udp_profile": "",
            "srcip_persist": "",
            "dns_profile": "",
            "policy_profile": "",
            "aclsnats": []
        }
    ]
},

```

```

{
  "tc_name": "",
  "name": "16.16.16.16_va",
  "address": "16.16.16.16",
  "status": 1,
  "arp_status": 1,
  "icmp_disable": 0,
  "vrid": 0,
  "redistribution": 0,
  "policy_profile": "",
  "virtual_services": [
    {
      "name": "othervs",
      "protocol": 5,
      "port": 0,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,

      "aclnamev6": "",
      "erules": [],
      "no_dest_nat": 0,
      "source_nat": "",
      "srcip_persist": "",
      "l4_profile_type": 3,
      "udp_profile": "",
      "policy_profile": "",
      "aclsnats": []
    }
  ]
}
]

```

虚拟地址获取

Action: slb.va.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	虚拟地址名称必须存在

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.get

请求 body

```
{  
  "name": "11.11.11.11_va"  
}
```

响应参数:

参考虚拟地址列表响应参数

响应举例:

```
{  
  "tc_name": "",  
  "name": "11.11.11.11_va",  
  "address": "11.11.11.11",  
  "status": 1,  
  "arp_status": 1,  
  "icmp_disable": 0,  
  "vrid": 0,  
  "redistribution": 0,  
  "policy_profile": "",  
  "virtual_services":  
  [{  
    "name":  
    "11.11.11.11",  
    "protocol": 9,  
    "port": 80,  
    "pool": "pool",  
    "connection_limit": {  
      "status": 0,  
      "connection_limit_number": 8000000 },  
    "vs_enable_intf": "",  
    "path_persist": 1,  
    "status": 1,  
    "desc_vport": "",  
    "snat_on_vip": 0,  
    "aclnamev6": "",  
    "erules": [],  
    "send_reset": 0,  
  }  
}
```

```

"source_nat": "",
"no_dest_nat": 0,
"auto_snat": 0,
"waf_profile": "12345",
"http_profile": "",
"tcp_profile": "",
"snat_port_preserve_enable": 0,
"snat_port_preserve_type": 0,
"connmulti_profile": "",
"srcip_persist": "",
"policy_profile": "",
"acls_nats": [] ]}
}

```

虚拟地址添加

虚拟地址有四种类型，分别为：IPv4/IPv6 地址类型、子网类型、IPv4 ACL 类型和 IPv6 ACL 类型。

添加 IPv4/IPv6 类型的虚拟地址

Action: slb.va.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	唯一
address	字符串	长度 1-63	ipv4/v6 类型的虚拟地址才有此参数	是	Address/network/acl 三选一，具体使用方法请参考 web 用户手册
status	整数	0, 1	虚拟地址状态	否	1:启用; 0:禁用;缺省值:1
arp_status	整数	0, 1	ARP 状态 1:启用; 0:禁用	否	缺省值:1
icmp_probe	整数	0-3	PING vIP 地址 0:always; 1:disable; 2:all; 3:any	否	缺省值:0
vrid	整数	0-8	vrrp 组 id,	否	缺省值:0
redistribution	整数	0, 1	路由联动:1:启用; 0:禁用	否	缺省值:0
policy_profile	字符串	长度 1-191	策略模版	否	缺省值:空

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.add

请求 body

```

{
  "virtual_address":
  {
    "tc_name": "",
    "name": "4.5.6.8_va",
    "address": "4.5.6.8",

```

```

    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": ""
  }
}

```

添加子网类型的虚拟地址

Action: slb.va.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	唯一
subnet	对象		子网对象	是	
address	IP		子网地址	是	
mask_len	整数	16-32	子网掩码长度	是	
status	整数	0, 1	虚拟地址状态	否	1:启用; 0:禁用;缺省值:1
arp_status	整数	0, 1	ARP 状态 1:启用; 0:禁用	否	缺省值:1
icmp_probe	整数	0-3	PING vIP 地址 0:always; 1:disable; 2:all; 3:any	否	缺省值:0
vrid	整数	0-8	vrrp 组 id,	否	缺省值:0
redistribution	整数	0, 1	路由联动:1:启用; 0:禁用	否	缺省值:0
policy_profile	字符串	长度 1-191	策略模版	否	缺省值:空

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.add

请求 body

```

{
  "virtual_address":
  {
    "tc_name": "",
    "name": "4.5.6.8_va",
    "subnet": {
      "address": "3.3.3.0",
      "mask_len": 24
    },
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": ""
  }
}

```


}

添加 IPv4 ACL 类型的虚拟地址

Action: slb.va.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	唯一
acl_id	整数	1-199	IPv4 ACL ID	是	
status	整数	0, 1	虚拟地址状态	否	1:启用; 0:禁用;缺省值:1
arp_status	整数	0, 1	ARP 状态 1:启用; 0:禁用	否	缺省值:1
icmp_probe	整数	0-3	PING vIP 地址 0:always; 1:disable; 2:all; 3:any	否	缺省值:0
vrid	整数	0-8	vrrp 组 id,	否	缺省值:0
redistribution	整数	0, 1	路由联动:1:启用; 0:禁用	否	缺省值:0
policy_profile	字符串	长度 1-191	策略模版	否	缺省值:空

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.add

请求 body

```
{
  "virtual_address":
  {
    "tc_name": "",
    "name": "4.5.6.8_va",
    "acl_id": 101,
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": ""
  }
}
```

添加 IPv6 ACL 类型的虚拟地址

Action: slb.va.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	唯一
acl_name	字符串	长度 1-191	IPv6 ACL 名称	是	
status	整数	0, 1	虚拟地址状态	否	1:启用; 0:禁用;缺省值:1
arp_status	整数	0, 1	ARP 状态 1:启用; 0:禁用	否	缺省值:1
icmp_probe	整数	0-3	PING vIP 地址 0:always; 1:disable; 2:all; 3:any	否	缺省值:0

vrid	整数	0-8	vrrp 组 id,	否	缺省值:0
redistribution	整数	0, 1	路由联动:1:启用; 0:禁用	否	缺省值:0
policy_profile	字符串	长度 1-191	策略模版	否	缺省值:空

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.add

请求 body

```
{
  "virtual_address":
  {
    "tc_name": "",
    "name": "4.5.6.8_va",
    "acl_name": "aclv6",
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": ""
  }
}
```

虚拟地址编辑

Action: slb.va.edit

请求参数:

虚拟地址编辑的参数与虚拟地址添加的请求参数完全相同

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.edit

```
{
  "virtual_address":
  {
    "tc_name": "",
    "name": "4.5.6.8_va",
    "address": "4.5.6.8",
    "status": 1,
    "arp_status": 1,
    "icmp_disable": 0,
    "vrid": 0,
    "redistribution": 0,
    "policy_profile": ""
  }
}
```



编辑和添加格式完全相同，只是 action 不同

虚拟地址删除

Action: slb.va.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	虚拟地址必须存在

请求举例:

```
POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.del
{
  "name": "4.5.6.7_va"
}
```

虚拟地址状态列表

Action: slb.va.stat.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.va.stat.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	虚拟地址名称
address	字符串	长度 1-63	虚拟地址的地址
status	整数	0-4	虚拟地址的状态: 0 禁用, 1 正常, 2 部分正常, 3 故障, 4 unknown
current_conns	整数	>=0	虚拟地址当前并发连接数
total_conns	整数	>=0	虚拟地址累计连接数
send_packets	整数	>=0	虚拟地址发送包数
receive_packets	整数	>=0	虚拟地址接收包数
send_bytes	整数	>=0	虚拟地址发送字节数
receive_bytes	整数	>=0	虚拟地址接收字节数
receive_rate	整数	>=0	虚拟地址接收接受速率, 单位 Bps
send_rate	整数	>=0	虚拟地址接发送速率, 单位 Bps
request_current	整数	>=0	虚拟地址当前请求
request_total	整数	>=0	虚拟地址累计请求
request_success	整数	>=0	虚拟地址累计成功请求
virtual_services_stat	数组		由虚拟服务状态统计信息对象组成的数组

name	字符串	长度 1-191	虚拟服务的名称
protocol	字符串	长度 1-63	虚拟服务的类型，2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务的端口
status	整数	0-2	虚拟服务状态：0 禁用，1 正常，2 故障
current_conns	整数	>=0	虚拟服务当前并发连接数
total_conns	整数	>=0	虚拟服务累计连接连接数
send_packets	整数	>=0	虚拟服务发送包数
receive_packets	整数	>=0	虚拟服务接收包数
send_bytes	整数	>=0	虚拟服务发送字节数
receive_bytes	整数	>=0	虚拟服务接收字节数
request_current	整数	>=0	虚拟服务当前请求
request_total	整数	>=0	虚拟服务累计请求
request_success	整数	>=0	虚拟服务累计成功请求

响应举例：

```
[
  {
    "name": "1.2.3.4_va",
    "address": "1.2.3.4",
    "status": 1,
    "current_conns": 0,
    "total_conns": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "receive_rate": 0,
```

```

"send_rate": 0,
"request_current": 0,
"request_total": 0,
"request_success": 0,
"virtual_services_stat": [
  {
    "name": "vs",
    "port": 443,
    "protocol": 15,
    "status": 1,
    "current_conns": 0,
    "total_conns": 0,
    "send_packets": 0,
    "receive_packets": 0,
    "send_bytes": 0,
    "receive_bytes": 0,
    "request_current": 0,
    "request_total": 0,
    "request_success": 0
  }
]
}
]

```

虚拟地址状态获取

Action: slb.va.stat.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	虚拟地址必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.va.stat.get

请求 body:

```

{
  "name": "1.2.3.4_va"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	虚拟地址名称
address	字符串	长度 1-63	虚拟地址的地址
status	整数	0-4	虚拟地址的状态: 0 禁用, 1 正常, 2 部分正常, 3 故障, 4 unknown
current_conns	整数	>=0	虚拟地址当前并发连接数
total_conns	整数	>=0	虚拟地址累计连接连接数

send_packets	整数	>=0	虚拟地址发送包数
receive_packets	整数	>=0	虚拟地址接收包数
send_bytes	整数	>=0	虚拟地址发送字节数
receive_bytes	整数	>=0	虚拟地址接收字节数
receive_rate	整数	>=0	虚拟地址接收接受速率，单位 Bps
send_rate	整数	>=0	虚拟地址接发送速率，单位 Bps
request_current	整数	>=0	虚拟地址当前请求
request_total	整数	>=0	虚拟地址累计请求
request_success	整数	>=0	虚拟地址累计成功请求
virtual_services_stat	数组		由虚拟服务状态统计信息对象组成的数组
name	字符串	长度 1-191	虚拟服务的名称
protocol	字符串	长度 1-63	虚拟服务的类型，2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务的端口
status	整数	0-2	虚拟服务状态：0 禁用，1 正常，2 故障
current_conns	整数	>=0	虚拟服务当前并发连接数
total_conns	整数	>=0	虚拟服务累计连接连接数
send_packets	整数	>=0	虚拟服务发送包数
receive_packets	整数	>=0	虚拟服务接收包数
send_bytes	整数	>=0	虚拟服务发送字节数
receive_bytes	整数	>=0	虚拟服务接收字节数
receive_rate	整数	>=0	虚拟地址接收接受速率，单位 Bps
send_rate	整数	>=0	虚拟地址接发送速率，单位 Bps
request_current	整数	>=0	虚拟服务当前请求
request_total	整数	>=0	虚拟服务累计请求
request_success	整数	>=0	虚拟服务累计成功请求

响应举例:

```
{
  "name": "1.2.3.4_va",
  "address": "1.2.3.4",
  "status": 1,
  "current_conns": 0,
  "total_conns": 0,
  "send_packets": 0,
  "receive_packets": 0,
  "send_bytes": 0,
  "receive_bytes": 0,
  "receive_rate": 0,
  "send_rate": 0,
  "request_current": 0,
  "request_total": 0,
  "request_success": 0,
  "virtual_services_stat": [
    {
      "name": "vs",
      "port": 443,
      "protocol": 15,
      "status": 1,
      "current_conns": 0,
      "total_conns": 0,
      "send_packets": 0,
      "receive_packets": 0,
      "send_bytes": 0,
      "receive_bytes": 0,
      "receive_rate": 0,
      "send_rate": 0,
      "request_current": 0,
      "request_total": 0,
      "request_success": 0
    }
  ]
}
```

虚拟地址状态清除

Action: slb.va.stat.clear

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.va.stat.clear



此 API 会清除所有虚拟地址的统计信息

虚拟服务

添加虚拟服务



添加虚拟服务之前需要先添加虚拟地址。

Action: slb.va.vs.add

公共请求参数:所有类型的虚拟服务都具有的参数,后续各种虚拟服务不再列举这些参数

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟服务使用过的虚拟地址名称	是	必须存在
virtual_service	对象		所有虚拟服务的参数都在此对象中	是	
name	字符串	长度 1-191	虚拟服务的名称	是	唯一
protocol	整数		虚拟服务的类型	是	2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务的端口	是	
vs_enable_intf	字符串	长度 1-63	使能接口号,ETH 开头,逗号分隔	否	缺省值:空
pool	字符串	长度 1-63	虚拟服务使用的服务池的名称	否	缺省值:空

connection_limit	对象		虚拟服务的并发连接限制对象	否	
status	整数	0, 1	虚拟服务的并发连接限制开关:	否	1: 启用; 0: 禁用; 缺省值: 0
connection_limit_number	整数	1-8000000	虚拟服务的并发连接限制数值, 仅当开启虚拟服务的并发连接限制开关是有效	否	缺省值: 8000000
syncookie	对象		虚拟服务启用 SYN Cookie 功能	否	
syncookie	整数	0, 1	syn cookie	否	1: 启用; 0: 禁用; 缺省值: 0
path_persist	整数	0, 1	路径保持:	否	1: 启用; 0: 禁用; 缺省值: 1
status	整数	0, 1	虚拟服务状态	否	1: 启用; 0: 禁用; 缺省值: 1
desc_vport	字符串	长度 1-191	虚拟服务描述	否	缺省值: 空
snat_on_vip	整数	0, 1	使能全局 snat 映射: 1: 启用; 0: 禁用	否	1: 启用; 0: 禁用; 缺省值: 0
auto_snat	整数	0, 1	接口 snat: 1: 启用; 0: 禁用	否	1: 启用; 0: 禁用; 缺省值: 0
snat_port_preserve_enable	整数	0, 1	源端口保持: 1: 启用; 0: 禁用	否	1: 启用; 0: 禁用; 缺省值: 0
snat_port_preserve_type	整数	0, 1	源端口保持类型: 1: 强制; 0: 尝试	否	1: 启用; 0: 禁用; 缺省值: 0
vs_acl_id	整数	0-199	访问列表, 1-199 表示对应 ACL, 0 表示没有配置	否	缺省值: 0
aclnamev6	字符串	长度 1-191	ipv6 访问列表名称	否	缺省值: 空
erules	数组		关联的 erule 名称组成的列表	否	缺省值: 空
send_reset	整数	0, 1	选择节点失败发送 rst: 1: 启用; 0: 禁用	否	缺省值: 0
source_nat	字符串	长度 1-63	源 snat 地址池	否	缺省值: 空
srcip_persist	字符串	长度 1-191	源地址保持模版, 一个虚拟服务只能有一种连接保持模板	否	缺省值: 空
dstip_persist	字符串	长度 1-191	目的地址保持模版, 一个虚拟服务只能有一种连接保持模板	否	缺省值: 空
cookie_persist	字符串	长度 1-191	cookie 保持模版, 一个虚拟服务只能有一种连接保持模板	否	缺省值: 空
sslid_persist	字符串	长度 1-191	SSL 连接保持模版, 一个虚拟服务只能有一种连接保持模板	否	缺省值: 空
policy_profile	字符串	长度 1-191	策略模版	否	缺省值: 0
acls_nats	数组		策略地址转换	否	缺省值: 空
acl_id	整数	1-199	策略地址转换-acl id,	否	缺省值: 空
acl_name	字符串	长度 1-191	策略地址转换-ipv6 acl name	否	缺省值: 空
nat_pool	字符串	长度 1-63	策略地址转换-snat 池	否	缺省值: 空
connection_mirror	整数	0, 1	连接镜像: 1: 启用; 0: 禁用	否	缺省值: 0
no_dest_nat	整数	0, 1	直接转发: 1: 启用; 0: 禁用	否	缺省值: 0
vimmediate_acti	整数	范围 0-2	服务 down 时的新建连接处理方式	是	0 代表 默认无,

on_on_service_d own					1 代表丢弃, 2 代表重置
vport_template_ name	整数	范围 1-191	虚拟服务模板	是	默认值为 default
ftp_profile	字符串	长度 1-191	ftp 模板	是	默认值为""

添加 DNS 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
udp_profile	字符串	长度 1-191	虚拟服务使用的 UDP 模板名称	否	必须存在
dns_profile	字符串	长度 1-191	虚拟服务使用的 DNS 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
  "name": "1.1.1.1_va",
  "virtual_service": {
    {
      "name": "dnsvs",
      "protocol": 22,
      "port": 53,
      "pool": "dnspool",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "aclnamev6": "",
    "erules": [
      ""
    ],
    "send_reset": 0,
    "source_nat": "",
    "udp_profile": "",
    "srcip_persist": "",
    "dstip_persist": "",
    "cookie_persist": "",
  }
}
```

```

"sslid_persis": "",
"dns_profile": "",
"policy_profile": "",
"aclsnats": [
  {
    "acl_id": 101,
    "nat_pool": "snatpool"
  }
]
}
}

```

添加 FTP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcp_profile	字符串	长度 1-191	虚拟服务使用的 TCP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "2.2.2.2_va",
  "virtual_service": {
    {
      "name": "ftpv",
      "protocol": 9,
      "port": 21,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,
      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "connection_mirror": 0,
      "no_dest_nat": 0,
      "syncookie": {

```

```

    "syncookie": 0
  },
  "source_nat": "",
  "tcp_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 HTTP 普通类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcp_profile	字符串	长度 1-191	虚拟服务使用的 TCP 模板名称	否	必须存在
http_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在
connmulti_profile	字符串	长度 1-191	虚拟服务使用的连接复用模板名称	否	必须存在
waf_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "3.3.3.3_24_va",
  "virtual_service":
  {
    "name": "https",
    "protocol": 12,
    "port": 80,
    "pool": "pool",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "snat_port_preserve_enable": 0,
    "snat_port_preserve_type": 0,
    "aclnamev6": "",
    "erules": [

```

```

    "erule-empty.txt"
  ],
  "send_reset": 0,
  "source_nat": "",
  "no_dest_nat": 0,
  "waf_profile": "",
  "http_profile": "",
  "tcp_profile": "",
  "connmulti_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 HTTP 增强类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
http_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在
connmulti_profile	字符串	长度 1-191	虚拟服务使用的连接复用模板名称	否	必须存在
waf_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在
cache_profile	字符串	长度 1-191	虚拟服务使用的缓存模板名称	否	必须存在
serverssl_profile	字符串	长度 1-191	虚拟服务使用的服务端 SSL 卸载模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "ACL_ID_101_va",
  "virtual_service":
  {
    "name": "httpenvs",
    "protocol": 14,
    "port": 80,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "1,2",
    "path_persist": 1,
    "status": 1,
  }
}

```

```

"desc_vport": "",
"snat_on_vip": 0,
"auto_snat": 0,
"snat_port_preserve_enable": 0,
"snat_port_preserve_type": 0,
"aclnamev6": "",
"erules": [],
"send_reset": 0,
"syncookie": {
  "syncookie": 0
},
"source_nat": "",
"no_dest_nat": 0,
"waf_profile": "",
"http_profile": "",
"cache_profile": "",
"tcpagent_profile": "",
"serverssl_profile": "",
"connmulti_profile": "",
"srcip_persist": "",
"policy_profile": "",
"aclsnats": []
}
}

```

添加 HTTPS 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
http_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在
connmulti_profile	字符串	长度 1-191	虚拟服务使用的连接复用模板名称	否	必须存在
waf_profile	字符串	长度 1-191	虚拟服务使用的 HTTP 模板名称	否	必须存在
cache_profile	字符串	长度 1-191	虚拟服务使用的缓存模板名称	否	必须存在
serverssl_profile	字符串	长度 1-191	虚拟服务使用的服务端 SSL 卸载模板名称	否	必须存在
clientsssl_profile	字符串	长度 1-191	虚拟服务使用的客户端 SSL 卸载模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "ACL_ID_aclv6_va",
  "virtual_service":
  {

```

```

"name": "httpsvs",
"protocol": 15,
"port": 443,
"pool": "",
"connection_limit": {
  "status": 0,
  "connection_limit_number": 8000000
},
"vs_enable_intf": "",
"path_persist": 1,
"status": 1,
"desc_vport": "",
"snat_on_vip": 0,
"auto_snat": 0,
"aclnamev6": "aclv6",
"erules": [],
"send_reset": 0,
"syncookie": {
  "syncookie": 0
},
"source_nat": "",
"no_dest_nat": 0,
"waf_profile": "",
"http_profile": "",
"cache_profile": "",
"tcpagent_profile": "",
"clientssl_profile": "",
"serverssl_profile": "",
"connmulti_profile": "",
"srcip_persist": "",
"policy_profile": "",
"aclsnats": []
}
}

```

添加 RTSP 类型的虚拟服务

Action: slb.va.vs.add

请求参数

名称	类型	范围	含义	必选	备注
tcp_profile	字符串	长度 1-191	虚拟服务使用的 TCP 模板名称	否	必须存在
rtsp_profile	字符串	长度 1-191	虚拟服务使用的 RTSP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
  "name": "0.0.0.0_va",
  "virtual_service": {
    {
      "name": "rtspvs",
      "protocol": 8,
      "port": 554,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "2",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "auto_snat": 0,
      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "no_dest_nat": 0,
      "syncookie": {
        "syncookie": 0
      },
      "tcp_profile": "",
      "rtsp_profile": "",
      "srcip_persist": "",
      "policy_profile": "",
      "aclsnats": []
    }
  }
}
```

添加 SMTP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
smtp_profile	字符串	长度 1-191	虚拟服务使用的 SMTP 模板名称	否	必须存在
clientssl_profile	字符串	长度 1-191	虚拟服务使用的客户端 SSL 卸载模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
  "name": "6.6.6.6_va",
  "virtual_service":
  {
    "name": "smtpvs",
    "protocol": 17,
    "port": 25,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "aclnamev6": "",
    "erules": [],
    "send_reset": 0,
    "syncookie": {
      "syncookie": 0
    },
    "source_nat": "",
    "tcpagent_profile": "",
    "clientssl_profile": "",
    "smtp_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
}
```

添加 SIP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
udp_profile	字符串	长度 1-191	虚拟服务使用的 UDP 模板名称	否	必须存在
dns_profile	字符串	长度 1-191	虚拟服务使用的 DNS 模板名称	否	必须存在
sip_profile	字符串	长度 1-191	虚拟服务使用的 SIP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
  "name": "7.7.7.7_va",
  "virtual_service": {
    {
      "name": "sipvs",
      "protocol": 11,
      "port": 5060,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
      "desc_vport": "",
      "snat_on_vip": 0,
      "aclnamev6": "",
      "erules": [],
      "send_reset": 0,
      "connection_mirror": 0,
      "udp_profile": "",
      "srcip_persist": "",
      "sip_profile": "",
      "dns_profile": "",
      "policy_profile": "",
      "aclsnats": []
    }
  }
}
```

添加 SIP-TCP 类型的虚拟服务

Action: slb.va.vs.add

请求参数

名称	类型	范围	含义	必选	备注
tcp_profile	字符串	长度 1-191	虚拟服务使用的 TCP 模板名称	否	必须存在
sip_profile	字符串	长度 1-191	虚拟服务使用的 SIP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
```

```

"name": "4.4.4.4_va",
"virtual_service":
{
  "name": "siptcpvs",
  "protocol": 18,
  "port": 5060,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "tcp_profile": "",
  "srcip_persist": "",
  "sip_profile": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 SSLAgent 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
serverssl_profile	字符串	长度 1-191	虚拟服务使用的服务端 SSL 卸载模板名称	否	必须存在
clientssl_profile	字符串	长度 1-191	虚拟服务使用的客户端 SSL 卸载模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "5.5.5.5_va",

```

```

"virtual_service":
{
  "name": "sslagentvs",
  "protocol": 16,
  "port": 443,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "snat_port_preserve_enable": 0,
  "snat_port_preserve_type": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "tcpagent_profile": "",
  "clientssl_profile": "",
  "serverssl_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 TCP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcp_profile	字符串	长度 1-191	虚拟服务使用的 TCP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
```

```

"name": "8.8.8.8_va",
"virtual_service":
{
  "name": "tcpvs",
  "protocol": 2,
  "port": 80,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "snat_port_preserve_enable": 0,
  "snat_port_preserve_type": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "connection_mirror": 0,
  "no_dest_nat": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "tcp_profile": "",
  "srcip_persist": "",
  "sslid_persist": ""
}
}

```

添加 TCPAgent 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
fixup_ftp	整数	0, 1	FTP 协议修正, 1:开启;0:关闭;	否	缺省值:0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "9.9.9.9_va",

```

```

"virtual_service":
{
  "name": "tcpagentvs",
  "protocol": 20,
  "port": 80,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "snat_port_preserve_enable": 0,
  "snat_port_preserve_type": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "fixup_ftp": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "tcpagent_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 TCPEXchange 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
connmulti_profile	字符串	长度 1-191	虚拟服务使用的连接复用模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```
{
```

```

"name": "10.10.10.10_va",
"virtual_service":
{
  "name": "tcpexchangevs",
  "protocol": 25,
  "port": 80,
  "pool": "",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "auto_snat": 0,
  "aclnamev6": "",
  "erules": [],
  "source_nat": "",
  "connmulti_profile": "",
  "tcpagent_profile": "",
  "policy_profile": "",
  "aclsnats": []
}
}

```

添加 MBLB 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
tcpagent_profile	字符串	长度 1-191	虚拟服务使用的快速四层模板名称	否	必须存在
connmulti_profile	字符串	长度 1-191	虚拟服务使用的连接复用模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "12.12.12.12_va",
  "virtual_service":
  {
    "name": "mblbvs",
    "protocol": 26,
    "port": 80,
    "pool": "",

```

```

"connection_limit": {
  "status": 0,
  "connection_limit_number": 8000000
},
"vs_enable_intf": "",
"path_persist": 1,
"status": 1,
"desc_vport": "",
"snat_on_vip": 0,
"auto_snat": 0,
"aclnamev6": "",
"erules": [],
"source_nat": "",
"connmulti_profile": "",
"tcpagent_profile": "",
"policy_profile": "",
"aclsnats": []
}
}

```

添加 TFTP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
udp_profile	字符串	长度 1-191	虚拟服务使用的 UDP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "13.13.13.13_va",
  "virtual_service": {
    {
      "name": "tftpvs",
      "protocol": 23,
      "port": 69,
      "pool": "",
      "connection_limit": {
        "status": 0,
        "connection_limit_number": 8000000
      },
      "vs_enable_intf": "",
      "path_persist": 1,
      "status": 1,
    }
  }
}

```



```

"desc_vport": "",
"snat_on_vip": 0,
"auto_snat": 0,
"aclnamev6": "",
"erules": [],
"connection_mirror": 0,
"no_dest_nat": 0,
"source_nat": "",
"udp_profile": "",
"srcip_persist": "",
"policy_profile": "",
"aclsnats": []
}
}

```

添加 RADIUS 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
udp_profile	字符串	长度 1-191	虚拟服务使用的 UDP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "14.14.14.14_va",
  "virtual_service": {
    "name": "radiusvs",
    "protocol": 28,
    "port": 1813,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "aclnamev6": "",
    "erules": [],
    "no_dest_nat": 0,

```

```

    "source_nat": "",
    "udp_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
}

```

添加 UDP 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
udp_profile	字符串	长度 1-191	虚拟服务使用的 UDP 模板名称	否	必须存在

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "15.15.15.15_va",
  "virtual_service":
  {
    "name": "udpvs",
    "protocol": 3,
    "port": 8000,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "snat_port_preserve_enable": 0,
    "snat_port_preserve_type": 0,
    "aclnamev6": "",
    "erules": [],
    "connection_mirror": 0,
    "no_dest_nat": 0,
    "source_nat": "",
    "udp_profile": "",
    "srcip_persist": "",
    "dns_profile": "",

```

```

    "policy_profile": "",
    "aclsnats": []
  }
}

```

添加 OTHER 类型的虚拟服务

Action: slb.va.vs.add

请求参数:

名称	类型	范围	含义	必选	备注
l4_profile_type	整数	2, 3	运输层协议	否	2:TCP;3:UDP;缺省值 2
tcp_profile	字符串	长度 1-191	虚拟服务使用的TCP模板名称	否	当 l4_profile_type 为 2 时有效
udp_profile	字符串	长度 1-191	虚拟服务使用的UDP模板名称	否	当 l4_profile_type 为 3 时有效

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.add

请求 body

```

{
  "name": "16.16.16.16_va",
  "virtual_service":
  {
    "name": "othervs",
    "protocol": 5,
    "port": 0,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "aclnamev6": "",
    "erules": [],
    "no_dest_nat": 0,
    "source_nat": "",
    "srcip_persist": "",
    "l4_profile_type": 3,
    "udp_profile": "",
    "policy_profile": "",
    "aclsnats": []
  }
}

```

}

}

虚拟服务获取

Action: slb.va.vs.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	必须存在
virtual_service	对象		虚拟服务对象	是	
protocol	整数		虚拟服务的类型	是	2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65535	端口服务	是	

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.get

请求 body

```
{
  "name": "1.2.3.4_va",
  "virtual_service":
  {
    "protocol": 12,
    "port": 444
  }
}
```

响应参数:slb.va.list 响应参数列表中一个虚拟地址中 **virtual_services** 中的一个

响应举例:

```
{
  "name": "vs2",
  "protocol": 12,
  "port": 444,
  "pool": "pool",
  "connection_limit": {
    "status": 0,
    "connection_limit_number": 8000000
  },
  "vs_enable_intf": "",
  "path_persist": 1,
  "status": 1,
  "desc_vport": "",
  "snat_on_vip": 0,
  "aclnamev6": "",
  "erules": [],
  "send_reset": 0,
  "syncookie": {
    "syncookie": 0
  },
  "source_nat": "",
  "no_dest_nat": 0,
  "auto_snat": 0,
  "waf_profile": "",
  "http_profile": "",
  "cache_profile": "",
  "tcpagent_profile": "",
  "clientssl_profile": "",
  "serverssl_profile": "",
  "connmulti_profile": "",
  "srcip_persist": "",
  "policy_profile": "",
  "aclsnats": []
}
```

虚拟服务编辑

Action: **slb.va.vs.edit**

请求参数:参考虚拟服务添加部分

请求举例:

POST <http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.edit>

请求 body

```

{
  "name": "2.2.2.2_va",
  "virtual_services":
  {
    "name": "ftpsv",
    "protocol": 9,
    "port": 21,
    "pool": "",
    "connection_limit": {
      "status": 0,
      "connection_limit_number": 8000000
    },
    "vs_enable_intf": "",
    "path_persist": 1,
    "status": 1,
    "desc_vport": "",
    "snat_on_vip": 0,
    "auto_snat": 0,
    "aclnamev6": "",
    "erules": [],
    "send_reset": 0,
    "connection_mirror": 0,
    "no_dest_nat": 0,
    "syncookie": {
      "syncookie": 0
    },
    "source_nat": "",
    "tcp_profile": "",
    "srcip_persist": "",
    "policy_profile": "",
    "aclsnats": []
  }
}

```



Note

虚拟服务编辑和添加请求 body 相同，action 不同，其他协议类型编辑参考添加

虚拟服务删除

Action: `slb.va.vs.del`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	必须存在
virtual_service	对象		虚拟服务对象	是	
protocol	整数		虚拟服务的类型	是	2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65535	端口服务	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.va.vs.del

请求 body

```
{
  "name": "1.2.3.4_va",
  "virtual_service":
  {
    "protocol": 12,
    "port": 444
  }
}
```

虚拟服务状态获取

Action: slb.va.vs.stat.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟地址名称	是	必须存在
virtual_service	对象		虚拟服务对象	是	

protocol	整数		虚拟服务的类型	是	2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent 17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务的端口	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.va.stat.get

请求 body

```
{
  "name": "1.2.3.4_va",
  "virtual_service":
  {
    "protocol": 12,
    "port": 444
  }
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	虚拟服务的名称
protocol	字符串	长度 1-63	虚拟服务的类型, 2: TCP 3: UDP 5: others 8: RTSP 9: FTP 11: SIP 12: fast_http (HTTP 普通) 14: http(HTTP 增强) 15: HTTPS 16: sslAgent

			17: SMTP 18: sip-TCP 20: tcpAgent 21: DIAMETER 22: DNS 23: TFTP 25: tcpExchange 26: MBLB 28: RADIUS
port	整数	0-65534	虚拟服务的端口
status	整数	0-2	虚拟服务状态: 0 禁用, 1 正常, 2 故障
current_conns	整数	>=0	虚拟服务当前并发连接数
total_conns	整数	>=0	虚拟服务累计连接连接数
send_packets	整数	>=0	虚拟服务发送包数
receive_packets	整数	>=0	虚拟服务接收包数
send_bytes	整数	>=0	虚拟服务发送字节数
receive_bytes	整数	>=0	虚拟服务接收字节数
receive_rate	整数	>=0	虚拟地址接收接受速率, 单位 Bps
send_rate	整数	>=0	虚拟地址接发送速率, 单位 Bps
request_current	整数	>=0	虚拟服务当前请求
request_total	整数	>=0	虚拟服务累计请求
request_success	整数	>=0	虚拟服务累计成功请求

响应举例:

```
{
  "name": "vs2",
  "port": 444,
  "protocol": 12,
  "status": 1,
  "current_conns": 0,
  "total_conns": 0,
  "receive_packets": 0,
  "send_packets": 0,
  "receive_bytes": 0,
  "send_bytes": 0,
  "receive_rate": 0,
  "send_rate": 0,
  "request_current": 0,
  "request_total": 0,
  "request_success": 0
}
```

模板

http 模板

http 模板列表

Action: slb.profile.http.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.http.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
fallback_url	字符串	长度 1-127	回退 url
force_reselect	整数	0-1	强制重选; 1:是;0:否
clientip_insert	字符串	长度 1-63	插入客户端 IP 头名称, 空表示不插入
clientip_insert_replace	整数	0-1	替换插入客户端 IP 头; 1:是;0:否
retry_503	整数	0-3	http 被动健康检查收到 5xx 次数
req_header_del	数组		请求头删除列表, 由多个请求头部名称组成
rsp_header_del	数组		响应头删除列表, 由多个响应头部名称组成
req_header_insert	数组		请求头插入数组
value	字符串	长度 1-63	插入请求头, key:value 格式
type	整数	0-2	插入模式:0:替换;1:强制;2:尝试;
rsp_header_insert	数组		响应头插入数组
value	字符串	长度 1-63	插入响应头, key:value 格式
type	整数	0-2	插入模式:0:替换;1:强制;2:尝试;
url_class	数组		http-url 分类数组--http-url 分类和 host 分类只能选一种
url	字符串	长度 1-63	http-url 分类--匹配 url
pool	字符串	长度 1-63	http-url 分类--匹配后发送到服务池名称
type	整数	0-3	http-url 分类--匹配方法:0:包含;1:头;2:尾;3:相同;
action	整数	1 或 32	url 匹配后行为: 1: 指定到服务池; 2: url 限速
limitaction	整数	2, 4, 8, 16	url 限速的操作: 2: 本地文件; 4: 重定向; 8: 丢弃; 16: 重置;
ruletable	字符串	已有规则表	url 限速的源 ip 匹配规则表 id, 使限速生效
ruletableid	整数	0-30	规则表的 id
reqrps	整数	0-8000000	Max-rps
path_or_url	字符串	长度 1-63	本地文件或重定向可以配置
host_class	数组		http-host 分类数组--http-url 分类和 host 分类

			只能选一种
host	字符串	长度 1-63	http-host 分类--匹配 host
pool	字符串	长度 1-63	http-host 分类--匹配后发送到服务池名称
type	整数	0-2	http-host 分类--匹配方法:0:包含;1:头;2:尾;
url_hash	整数	0-2	URL 哈希; 0:禁用;1:头部 HASH; 尾部 HASH
url_hash_len	整数	4-128	url 哈希字节长度
url_hash_offset	整数	0-255	url 哈希偏移量
redirect_modify	数组		重 URL 定向动作列表
match	字符串	长度 1-63	重定向--匹配 URL
to	字符串	长度 1-63	重定向--重定向目标
redirect_modify_https	整数	0-1	重定向更改 https;1:启用; 0:禁用
redirect_modify_https_port	整数	1-65535	重定向更改 https 到端口
compress	整数	0-1	压缩 1:启用; 0:禁用
compress_keep_header	整数	0-1	压缩时保持 accept 头部
compress_level	整数	1-9	压缩级别;1:最快;5:标准;9:最好;
compress_min_len	整数	0-2147483 647	最小内容长度
compress_content_type	数组		包含内容
compress_content_type_exclude	数组		排除内容
compress_url_exclude	数组		排除 uri
cookie_encrypt_name	字符串	1-63	Cookie 加密的名称
cookie_encrypt_password	字符串	1-63	Cookie 加密密码

响应举例:

```
[
  {
    "name": "p1",
    "fallback_url": "/asdas",
    "fallback_page": "",
    "force_reselect": 1,
    "clientip_insert": "X-Forwarded-For",
    "clientip_insert_replace": 0,
    "retry_503": 2,
    "cookie_encrypt_name": "",
    "cookie_encrypt_password": "",
    "conlimit_action": {
      "action": 0,
      "path_or_url": ""
    },
    "response_code_actions": [],
    "req_header_del": [
      "key"
    ],
    "rsp_header_del": [
      "key"
    ],
  },
]
```

```

"req_header_insert": [
    {
        "value": "key:value",
        "type": 0
    }
],
"rsp_header_insert": [
    {
        "value": "key:value",
        "type": 0
    }
],
"url_class_log_interval": 0,
"url_class": [
    {
        "url": "1",
        "pool": "poolv4-0",
        "type": 0,
        "action": 1
    },
    {
        "url": "1",
        "pool": "",
        "type": 0,
        "action": 32,
        "limitaction": 16,
        "ruletable": "1",
        "ruletableid": 1,
        "reqrps": 1,
        "path_or_url": ""
    }
],
"url_hash": 1,
"url_hash_len": 34,
"url_hash_offset": 12,
"redirect_modify": [
    {
        "match": "url",
        "to": "dest"
    }
],
"redirect_modify_https": 1,
"redirect_modify_https_port": 4433,
"cookie_select": 0,
"cookie_expire": 0,
"cookie_expire_enable": 0,

```

```

        "compress": 1,
        "compress_keep_header": 1,
        "compress_level": 9,
        "compress_min_len": 1283,
        "compress_content_type": [
            "content1",
            "content2"
        ],
        "compress_content_type_exclude": [
            "content1",
            "content2"
        ],
        "compress_url_exclude": [
            "url1",
            "url2"
        ]
    }
]

```

http 模板获取

Action: slb.profile.http.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.http.get

请求 Body:

```

{
    "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
fallback_url	字符串	长度 1-127	回退 url
force_reselect	整数	0-1	强制重选; 1:是;0:否
clientip_insert	字符串	长度 1-63	插入客户端 IP 头名称, 空表示不插入
clientip_insert_replace	整数	0-1	替换插入客户端 IP 头; 1:是;0:否
retry_503	整数	0-3	http 被动健康检查收到 5xx 次数
req_header_del	数组		请求头删除列表, 由多个请求头部名称组成
rsp_header_del	数组		响应头删除列表, 由多个响应头部名称组成
req_header_insert	数组		请求头插入数组

value	字符串	长度 1-63	插入请求头, key:value 格式
type	整数	0-2	插入模式:0:替换;1:强制;2:尝试;
rsp_header_insert	数组		响应头插入数组
value	字符串	长度 1-63	插入响应头, key:value 格式
type	整数	0-2	插入模式:0:替换;1:强制;2:尝试;
url_class	数组		http-url 分类数组--http-url 分类和 host 分类只能选一种
url	字符串	长度 1-63	http-url 分类--匹配 url
pool	字符串	长度 1-63	http-url 分类--匹配后发送到服务池名称
type	整数	0-3	http-url 分类--匹配方法:0:包含;1:头;2:尾;3:相同;
action	整数	1 或 32	url 匹配后行为: 1: 指定到服务池; 2: url 限速
limitaction	整数	2, 4, 8, 16	url 限速的操作: 2: 本地文件; 4: 重定向; 8: 丢弃; 16: 重置;
ruletable	字符串	已有规则表	url 限速的源 ip 匹配规则表 id, 使限速生效
ruletableid	整数	0-30	规则表的 id
reqrps	整数	0-8000000	Max-rps
path_or_url	字符串	长度 1-63	本地文件或重定向可以配置
host_class	数组		http-host 分类数组--http-url 分类和 host 分类只能选一种
host	字符串	长度 1-63	http-host 分类--匹配 host
pool	字符串	长度 1-63	http-host 分类--匹配后发送到服务池名称
type	整数	0-2	http-host 分类--匹配方法:0:包含;1:头;2:尾;
url_hash	整数	0-2	URL 哈希; 0:禁用;1:头部 HASH; 尾部 HASH
url_hash_len	整数	4-128	url 哈希字节长度
url_hash_offset	整数	0-255	url 哈希偏移量
redirect_modify	数组		重 URL 定向动作列表
match	字符串	长度 1-63	重定向--匹配 URL
to	字符串	长度 1-63	重定向--重定向目标
redirect_modify_https	整数	0-1	重定向更改 https;1:启用; 0:禁用
redirect_modify_https_port	整数	1-65535	重定向更改 https 到端口
compress	整数	0-1	压缩 1:启用; 0:禁用
compress_keep_header	整数	0-1	压缩时保持 accept 头部
compress_level	整数	1-9	压缩级别;1:最快;5:标准;9:最好;
compress_min_len	整数	0-2147483647	最小内容长度
compress_content_type	数组		包含内容
compress_content_type_exclude	数组		排除内容
compress_url_exclude	数组		排除 uri
cookie_encrypt_name	字符串	1-63	Cookie 加密的名称
cookie_encrypt_password	字符串	1-63	Cookie 加密密码

响应举例:

```
{
```

```

"name": "p1",
"fallback_url": "/asdas",
"fallback_page": "",
"force_reselect": 1,
"clientip_insert": "X-Forwarded-For",
"clientip_insert_replace": 0,
"retry_503": 2,
"cookie_encrypt_name": "",
"cookie_encrypt_password": "adccookiepassword",
"connlimit_action": {
    "action": 0,
    "path_or_url": ""
},
"response_code_actions": [],
"req_header_del": [
    "key"
],
"rsp_header_del": [
    "key"
],
"req_header_insert": [
    {
        "value": "key:value",
        "type": 0
    }
],
"rsp_header_insert": [
    {
        "value": "key:value",
        "type": 0
    }
],
"url_class_log_interval": 0,
"url_class": [
    {
        "url": "1",
        "pool": "poolv4-0",
        "type": 0,
        "action": 1
    },
    {
        "url": "1",
        "pool": "",
        "type": 0,
        "action": 32,
        "limitaction": 16,

```

```

        "ruletable": "1",
        "ruletableid": 1,
        "reqrps": 1,
        "path_or_url": ""
    }
],
"url_hash": 1,
"url_hash_len": 34,
"url_hash_offset": 12,
"redirect_modify": [
    {
        "match": "url",
        "to": "dest"
    }
],
"redirect_modify_https": 1,
"redirect_modify_https_port": 4433,
"cookie_select": 0,
"cookie_expire": 0,
"cookie_expire_enable": 0,
"compress": 1,
"compress_keep_header": 1,
"compress_level": 9,
"compress_min_len": 1283,
"compress_content_type": [
    "content1",
    "content2"
],
"compress_content_type_exclude": [
    "content1",
    "content2"
],
"compress_url_exclude": [
    "url1",
    "url2"
]
]
}

```

http 模板增加

Action: slb.profile.http.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
fallback_url	字符串	长度 1-127	回退 url	否	
force_reselect	整数	0-1	强制重选; 1:是;0:否	否	1:是;0:否, 缺省值:0

clientip_insert	字符串	长度 1-63	插入客户端 IP 头名称	否	空表示不插入
clientip_insert_replace	整数	0-1	替换插入客户端 IP 头; 1:是;0:否	否	1:是;0:否, 缺省值:0
retry_503	整数	0-3	http 被动健康检查收到 5xx 次数	否	缺省值:0, 0 表示禁用 被动健康检查
req_header_del	数组		请求头删除列表, 由多个 请求头部名称组成	否	缺省值:空
rsp_header_del	数组		响应头删除列表, 由多个 响应头部名称组成	否	缺省值:空
req_header_insert	数组		请求头插入数组	否	缺省值:空
value	字符串	长度 1-63	插入请求头, key:value 格式	是	
type	整数	0-2	插入模式:0: 替换;1: 强 制;2:尝试;	是	0 替换, 1 强制, 2 尝 试
rsp_header_insert	数组		响应头插入数组	否	缺省值:空
value	字符串	长度 1-63	插入响应头, key:value 格式	是	
type	整数	0-2	插入模式:0: 替换;1: 强 制;2:尝试;	是	0 替换, 1 强制, 2 尝 试
url_class	数组		http-url 分类数组 --http-url 分类和 host 分类只能选一种	是	缺省值:空
url	字符串	长度 1-63	http-url 分类--匹配 url	是	
pool	字符串	长度 1-63	http-url 分类--匹配后 发送到服务池名称	是	
type	整数	0-3	http-url 分类--匹配方 法 :0: 包含 ;1: 头 ;2: 尾;3:相同;	否	0: 包含 ;1: 头 ;2: 尾;3:相同;缺省值:0
action	整数	1 或 32	url 匹配后行为: 1: 指定 到服务池; 2: url 限速	是	
limitaction	整数	2, 4, 8, 16	url 限速的操作: 2: 本地 文件; 4: 重定向; 8: 丢 弃; 16: 重置;	是	
ruletable	字符串	已有规则表	url 限速的源 ip 匹配规则 表 id, 使限速生效	否	
ruletableid	整数	0-30	规则表的 id	否	
reqrps	整数	0-8000000	Max-rps	是	
path_or_url	字符串	长度 1-63	本地文件或重定向可以 配置	否	
host_class	数组		http-host 分类数组 --http-url 分类和 host 分类只能选一种	否	缺省值:空
host	字符串	长度 1-63	http-host 分类--配匹 host	是	
pool	字符串	长度 1-63	http-host 分类--匹配后	是	

			发送到服务池名称		
type	整数	0-2	http-host 分类--匹配方法:0:包含;1:头;2:尾;	否	0:包含;1:头;2:尾;缺省值:0
url_hash	整数	0-2	URL 哈希; 0:禁用;1:头部HASH; 尾部 HASH	否	0:禁用;1:头部HASH;尾部 HASH缺省值:0
url_hash_len	整数	4-128	url 哈希字节长度	否	缺省值:0
url_hash_offset	整数	0-255	url 哈希偏移量	否	缺省值:0
redirect_modify	数组		重 URL 定向动作列表	否	缺省值:空
match	字符串	长度 1-63	重定向--匹配 URL	是	
to	字符串	长度 1-63	重定向--重定向目标	是	
redirect_modify_https	整数	0-1	重定向更改 https;1:启用; 0:禁用	否	1:启用; 0:禁用; 缺省值:0
redirect_modify_https_port	整数	1-65535	重定向更改 https 到端口	否	缺省值:0
compress	整数	0-1	压缩 1:启用; 0:禁用	否	缺省值:0
compress_keep_header	整数	0-1	压缩时保持 accept 头部	否	缺省值:0
compress_level	整数	1-9	压缩级别;1:最快;5:标准;9:最好;	否	1:最快;5:标准;9:最好;缺省值:1
compress_min_len	整数	0-2147483647	最小内容长度	否	缺省值:0
compress_content_type	数组		包含内容	否	缺省值:空
compress_content_type_exclude	数组		排除内容	否	缺省值:空
compress_url_exclude	数组		排除 uri	否	缺省值:空
cookie_encrypt_name	字符串	1-63	Cookie 加密的名称	否	缺省值:空
cookie_encrypt_password	字符串	1-63	Cookie 加密密码	否	缺省值:空

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=s1b.profile.http.ad

请求 body:

```
{
  "name": "p1",
  "fallback_url": "/asdas",
  "fallback_page": "",
  "force_reselect": 1,
  "clientip_insert": "X-Forwarded-For",
  "clientip_insert_replace": 0,
  "retry_503": 2,
  "cookie_encrypt_name": "",
  "cookie_encrypt_password": "adccookiepassword",
  "connlimit_action": {
    "action": 0,

```

```

    "path_or_url": ""
  },
  "response_code_actions": [],
  "req_header_del": [
    "key"
  ],
  "rsp_header_del": [
    "key"
  ],
  "req_header_insert": [
    {
      "value": "key:value",
      "type": 0
    }
  ],
  "rsp_header_insert": [
    {
      "value": "key:value",
      "type": 0
    }
  ],
  "url_class_log_interval": 0,
  "url_class": [
    {
      "url": "1",
      "pool": "poolv4-0",
      "type": 0,
      "action": 1
    },
    {
      "url": "1",
      "pool": "",
      "type": 0,
      "action": 32,
      "limitaction": 16,
      "ruletable": "1",
      "ruletableid": 1,
      "reqrps": 1,
      "path_or_url": ""
    }
  ],
  "url_hash": 1,
  "url_hash_len": 34,
  "url_hash_offset": 12,
  "redirect_modify": [
    {

```

```

        "match": "url",
        "to": "dest"
    }
],
"redirect_modify_https": 1,
"redirect_modify_https_port": 4433,
"cookie_select": 0,
"cookie_expire": 0,
"cookie_expire_enable": 0,
"compress": 1,
"compress_keep_header": 1,
"compress_level": 9,
"compress_min_len": 1283,
"compress_content_type": [
    "content1",
    "content12"
],
"compress_content_type_exclude": [
    "content1",
    "content12"
],
"compress_url_exclude": [
    "uri1",
    "uri12"
]
}

```

http 模板编辑

Action: slb.profile.http.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
fallback_url	字符串	长度 1-127	回退 url	否	
force_reselect	整数	0-1	强制重选; 1:是;0:否	否	1:是;0:否, 缺省值:0
clientip_insert	字符串	长度 1-63	插入客户端 IP 头名称	否	空表示不插入
clientip_insert_replace	整数	0-1	替换插入客户端 IP 头; 1:是;0:否	否	1:是;0:否, 缺省值:0
retry_503	整数	0-3	http 被动健康检查收到 5xx 次数	否	缺省值:0, 0 表示禁用 被动健康检查
req_header_del	数组		请求头删除列表, 由多个 请求头部名称组成	否	缺省值:空
rsp_header_del	数组		响应头删除列表, 由多个 响应头部名称组成	否	缺省值:空
req_header_insert	数组		请求头插入数组	否	缺省值:空

value	字符串	长度 1-63	插入请求头, key:value 格式	是	
type	整数	0-2	插入模式:0: 替换;1: 强制;2: 尝试;	是	0 替换, 1 强制, 2 尝试
rsp_header_insert	数组		响应头插入数组	否	缺省值:空
value	字符串	长度 1-63	插入响应头, key:value 格式	是	
type	整数	0-2	插入模式:0: 替换;1: 强制;2: 尝试;	是	0 替换, 1 强制, 2 尝试
url_class	数组		http-url 分类数组 --http-url 分类和 host 分类只能选一种	是	缺省值:空
url	字符串	长度 1-63	http-url 分类--匹配 url	是	
pool	字符串	长度 1-63	http-url 分类--匹配后 发送到服务池名称	是	
type	整数	0-3	http-url 分类--匹配方法 :0: 包含;1: 头;2: 尾;3: 相同;	否	0: 包含;1: 头;2: 尾;3: 相同;缺省值:0
action	整数	1 或 32	url 匹配后行为: 1: 指定 到服务池; 2: url 限速	是	
limitaction	整数	2, 4, 8, 16	url 限速的操作: 2: 本地 文件; 4: 重定向; 8: 丢弃; 16: 重置;	是	
ruletable	字符串	已有规则表	url 限速的源 ip 匹配规则 表 id, 使限速生效	否	
ruletableid	整数	0-30	规则表的 id	否	
reqrps	整数	0-8000000	Max-rps	是	
path_or_url	字符串	长度 1-63	本地文件或重定向可以 配置	否	
host_class	数组		http-host 分类数组 --http-url 分类和 host 分类只能选一种	否	缺省值:空
host	字符串	长度 1-63	http-host 分类--匹配 host	是	
pool	字符串	长度 1-63	http-host 分类--匹配后 发送到服务池名称	是	
type	整数	0-2	http-host 分类--匹配方法:0: 包含;1: 头;2: 尾;	否	0: 包含;1: 头;2: 尾; 缺省值:0
url_hash	整数	0-2	URL 哈希; 0: 禁用;1: 头部 HASH; 尾部 HASH	否	0: 禁用;1: 头部 HASH; 尾部 HASH 缺省值:0
url_hash_len	整数	4-128	url 哈希字节长度	否	缺省值:0
url_hash_offset	整数	0-255	url 哈希偏移量	否	缺省值:0
redirect_modify	数组		重 URL 定向动作列表	否	缺省值:空
match	字符串	长度 1-63	重定向--匹配 URL	是	

to	字符串	长度 1-63	重定向--重定向目标	是	
redirect_modify_https	整数	0-1	重定向更改 https;1:启用; 0:禁用	否	1:启用; 0:禁用; 缺省值:0
redirect_modify_https_port	整数	1-65535	重定向更改 https 到端口	否	缺省值:0
compress	整数	0-1	压缩 1:启用; 0:禁用	否	缺省值:0
compress_keep_header	整数	0-1	压缩时保持 accept 头部	否	缺省值:0
compress_level	整数	1-9	压缩级别;1:最快;5:标准;9:最好;	否	1:最快;5:标准;9:最好;缺省值:1
compress_min_len	整数	0-2147483647	最小内容长度	否	缺省值:0
compress_content_type	数组		包含内容	否	缺省值:空
compress_content_type_exclude	数组		排除内容	否	缺省值:空
compress_url_exclude	数组		排除 uri	否	缺省值:空
cookie_encrypt_name	字符串	1-63	Cookie 加密的名称	否	缺省值:空
cookie_encrypt_password	字符串	1-63	Cookie 加密密码	否	缺省值:空

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=s1b.profile.http.ad

请求 body:

```
{
  "name": "p1",
  "fallback_url": "/asdas",
  "fallback_page": "",
  "force_reselect": 1,
  "clientip_insert": "X-Forwarded-For",
  "clientip_insert_replace": 0,
  "retry_503": 2,
  "cookie_encrypt_name": "",
  "cookie_encrypt_password": "adccookiepassword",
  "connlimit_action": {
    "action": 0,
    "path_or_url": ""
  },
  "response_code_actions": [],
  "req_header_del": [
    "key"
  ],
  "rsp_header_del": [
    "key"
  ],
}
```

```
"req_header_insert": [
  {
    "value": "key:value",
    "type": 0
  }
],
"rsp_header_insert": [
  {
    "value": "key:value",
    "type": 0
  }
],
"url_class_log_interval": 0,
"url_class": [
  {
    "url": "1",
    "pool": "poolv4-0",
    "type": 0,
    "action": 1
  },
  {
    "url": "1",
    "pool": "",
    "type": 0,
    "action": 32,
    "limitaction": 16,
    "ruletable": "1",
    "ruletableid": 1,
    "reqrps": 1,
    "path_or_url": ""
  }
],
"url_hash": 1,
"url_hash_len": 34,
"url_hash_offset": 12,
"redirect_modify": [
  {
    "match": "url",
    "to": "dest"
  }
],
"redirect_modify_https": 1,
"redirect_modify_https_port": 4433,
"cookie_select": 0,
"cookie_expire": 0,
"cookie_expire_enable": 0,
```

```

"compress": 1,
"compress_keep_header": 1,
"compress_level": 9,
"compress_min_len": 1283,
"compress_content_type": [
    "content1",
    "content12"
],
"compress_content_type_exclude": [
    "content1",
    "content12"
],
"compress_url_exclude": [
    "uri1",
    "uri12"
]
}

```

http 模板删除

Action: slb.profile.http.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.http.del

请求 body:

```

{
  "name": "p1"
}

```

RTSP 模板

rtsp 模板列表

Action: slb.profile.rtsp.list

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.rtsp.list

响应参数:

名称	类型	范围	含义
----	----	----	----

name	字符串	长度 1-191	模板名称
uri_switch	数组		策略列表
uri	字符串	长度 1-63	策略--匹配 URI
pool	字符串	长度 1-63	策略--匹配 URI 之后选择服务池名称

响应举例:

```
[
  {
    "name": "p1",
    "uri_switch": [
      {
        "uri": "/aaa",
        "pool": "pool0"
      }
    ]
  }
]
```

rtsp 模板获取

Action: slb.profile.rtsp.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.rtsp.get>

请求 body:

```
{
  "name": "p1"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
uri_switch	数组		策略列表
uri	字符串	长度 1-63	策略--匹配 URI
pool	字符串	长度 1-63	策略--匹配 URI 之后选择服务池名称

响应举例:

```
{
  "name": "p1",
  "uri_switch": [
    {
      "uri": "/aaa",
      "pool": "pool0"
    }
  ]
}
```

```

    }
  ]
}

```

rtsp 模板增加

Action: slb.profile.rtsp.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
uri_switch	数组		策略列表	否	缺省值:空
uri	字符串	长度 1-63	策略--匹配 URI	否	
pool	字符串	长度 1-63	策略--匹配 URI 之后选择服务池名称	否	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.rtsp.add>

请求 body:

```

{
  "name": "p1",
  "uri_switch": [
    {
      "uri": "/aaa",
      "pool": "pool0"
    }
  ]
}

```

rtsp 模板编辑

Action: slb.profile.rtsp.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
uri_switch	数组		策略列表	否	缺省值:空
uri	字符串	长度 1-63	策略--匹配 URI	否	
pool	字符串	长度 1-63	策略--匹配 URI 之后选择服务池名称	否	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.rtsp.edit>

请求 body:

```

{
  "name": "p1",

```

```

"uri_switch": [
  {
    "uri": "/aaa",
    "pool": "pool0"
  }
]
}

```

rtsp 模板删除

Action: **slb.profile.rtsp.del**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.rtsp.del>

请求 body:

```

{
  "name": "p1"
}

```

SMTP 模板

smtp 模板列表

Action: **slb.profile.smtp.list**

请求参数: 无

请求举例:

GET
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.smtp.list>

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
starttls	整数	0-2	starttls 需求; 0 禁止, 1 可选, 2 强制
expn	整数	0, 1	禁止命令 turn; 0 不禁止, 1 禁止
turn	整数	0, 1	禁止命令 expn; 0 不禁止, 1 禁止
vrfy	整数	0, 1	禁止命令 vrfy; 0 不禁止, 1 禁止
email_server_domain	字符串	长度 1-63	email 服务器域
server_ready_message	字符串	长度 1-63	服务器就绪消息
client_switch	数组		服务池选择
client_domain	字符串	长度 1-63	客户端域

pool	字符串	长度 1-63	服务池
match_method	整数	0-2	匹配方法:0:包含;1:头;2:尾;

响应举例:

```

[[
  "name": "p1",
  "starttls": 0,
  "expn": 1,
  "turn": 0,
  "vrfy": 0,
  "email_server_domain": "dom",
  "server_ready_message": "rdy",
  "client_switch": [
    {
      "client_domain": "client",
      "pool": "pool",
      "match_method": 0
    },
    {
      "client_domain": "client2",
      "pool": "pool",
      "match_method": 1
    },
    {
      "client_domain": "client3",
      "pool": "pool",
      "match_method": 2
    }
  ]
]]

```

smtp 模板获取

Action: slb.profile.smtp.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.smtp.get>

请求 body:

```

{
  "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
starttls	整数	0-2	starttls 需求; 0 禁止, 1 可选, 2 强制
expn	整数	0, 1	禁止命令 turn; 0 不禁止, 1 禁止
turn	整数	0, 1	禁止命令 expn; 0 不禁止, 1 禁止
vrfy	整数	0, 1	禁止命令 vrfy; 0 不禁止, 1 禁止
email_server_domain	字符串	长度 1-63	email 服务器域
server_ready_message	字符串	长度 1-63	服务器就绪消息
client_switch	数组		服务池选择
client_domain	字符串	长度 1-63	客户端域
pool	字符串	长度 1-63	服务池
match_method	整数	0-2	匹配方法:0:包含;1:头;2:尾;

响应举例:

```
{
  "name": "p1",
  "starttls": 0,
  "expn": 1,
  "turn": 0,
  "vrfy": 0,
  "email_server_domain": "dom",
  "server_ready_message": "rdy",
  "client_switch": [
    {
      "client_domain": "client",
      "pool": "pool",
      "match_method": 0
    },
    {
      "client_domain": "client2",
      "pool": "pool",
      "match_method": 1
    },
    {
      "client_domain": "client3",
      "pool": "pool",
      "match_method": 2
    }
  ]
}
```

smtp 模板增加

Action: slb.profile.smtp.add

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	模板名称	是	
starttls	整数	0-2	starttls 需求;	否	0 禁止;1 可选;2 强制;缺省值:0
expn	整数	0, 1	禁止命令 turn;	否	0 不禁止, 1 禁止; 缺省值:0
turn	整数	0, 1	禁止命令 expn;	否	0 不禁止, 1 禁止; 缺省值:0
vrfy	整数	0, 1	禁止命令 vrfy	否	0 不禁止, 1 禁止, 缺省值:0
email_server_domain	字符串	长度 1-63	email 服务器域	否	缺省值:空
server_ready_message	字符串	长度 1-63	服务器就绪消息	否	缺省值:空
client_switch	数组		服务池选择	否	缺省值:空
client_domain	字符串	长度 1-63	客户端域	否	
pool	字符串	长度 1-63	服务池	否	
match_method	整数	0-2	匹配方法:0:包含;1:头;2:尾;	否	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=s1b.profile.smtp.ad>
d

请求 body:

```
{
  "name": "p1",
  "starttls": 0,
  "expn": 1,
  "turn": 0,
  "vrfy": 0,
  "email_server_domain": "dom",
  "server_ready_message": "rdy",
  "client_switch": [
    {
      "client_domain": "client",
      "pool": "pool",
      "match_method": 0
    },
    {
      "client_domain": "client2",
      "pool": "pool",
      "match_method": 1
    },
    {
      "client_domain": "client3",
      "pool": "pool",
      "match_method": 2
    }
  ]
}
```

```
]
}
```

smtp 模板编辑

Action: slb.profile.smtp.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
starttls	整数	0-2	starttls 需求;	否	0 禁止;1 可选;2 强制;缺省值:0
expn	整数	0, 1	禁止命令 turn;	否	0 不禁止, 1 禁止; 缺省值:0
turn	整数	0, 1	禁止命令 expn;	否	0 不禁止, 1 禁止; 缺省值:0
vrfy	整数	0, 1	禁止命令 vrfy	否	0 不禁止, 1 禁止, 缺省值:0
email_server_domain	字符串	长度 1-63	email 服务器域	否	缺省值:空
server_ready_message	字符串	长度 1-63	服务器就绪消息	否	缺省值:空
client_switch	数组		服务池选择	否	缺省值:空
client_domain	字符串	长度 1-63	客户端域	否	
pool	字符串	长度 1-63	服务池	否	
match_method	整数	0-2	匹配方法:0:包含;1:头;2:尾;	否	

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.smtp.edit
```

请求 body:

```
{
  "name": "p1",
  "starttls": 0,
  "expn": 1,
  "turn": 0,
  "vrfy": 0,
  "email_server_domain": "dom",
  "server_ready_message": "rdy",
  "client_switch": [
    {
      "client_domain": "client",
      "pool": "pool",
      "match_method": 0
    },
    {
      "client_domain": "client2",
```

```

    "pool": "pool",
    "match_method": 1
  },
  {
    "client_domain": "client3",
    "pool": "pool",
    "match_method": 2
  }
]
}

```

smtp 模板删除

Action: slb.profile.smtp.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

```

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.smtp.del

```

请求 body:

```

{
  "name": "p1"
}

```

DNS 模板

dns 模板列表

Action: slb.profile.dns.list

请求参数: 无

请求举例:

```

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.dns.list

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
enabled	整数	0-1	状态; 1: 开启; 0: 禁用
drop_bad_req	整数	0-1	安全策略; 1: 开启; 0: 禁用
default_to_cache	整数	0-1	默认 cache; 1: 开启; 0: 禁用
cache_size	整数	0-2000000	最大 cache 数

响应举例:


```

[
  {
    "name": "p1",
    "enabled": 1,
    "drop_bad_req": 1,
    "default_to_cache": 1,
    "cache_size": 123
  }
]

```

dns 模板获取

Action: slb.profile.dns.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.dns.get

请求 body:

```

{
  "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	模板名称
enabled	整数	0-1	状态; 1: 开启; 0: 禁用
drop_bad_req	整数	0-1	安全策略; 1: 开启; 0: 禁用
default_to_cache	整数	0-1	默认 cache; 1: 开启; 0: 禁用
cache_size	整数	0-2000000	最大 cache 数

响应举例:

```

{
  "name": "p1",
  "enabled": 1,
  "drop_bad_req": 1,
  "default_to_cache": 1,
  "cache_size": 123
}

```

dns 模板增加

Action: slb.profile.dns.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
enabled	整数	0-1	状态	否	1 开启, 0 禁用, 缺省值:1

drop_bad_req	整数	0-1	安全策略	否	1 开启, 0 禁用, 缺省值:0
default_to_cache	整数	0-1	默认 cache	否	1 开启, 0 禁用, 缺省值:0
cache_size	整数	0-2000000	最大 cache 数	否	缺省值:0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.dns.add
 请求 body:

```
{
  "name": "p1",
  "enabled": 1,
  "drop_bad_req": 1,
  "default_to_cache": 1,
  "cache_size": 123
}
```

dns 模板编辑

Action: slb.profile.dns.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
enabled	整数	0-1	状态	否	1 开启, 0 禁用, 缺省值:1
drop_bad_req	整数	0-1	安全策略	否	1 开启, 0 禁用, 缺省值:0
default_to_cache	整数	0-1	默认 cache	否	1 开启, 0 禁用, 缺省值:0
cache_size	整数	0-2000000	最大 cache 数	否	缺省值:0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.dns.edi
 t
 请求 body:

```
{
  "name": "p1",
  "enabled": 1,
  "drop_bad_req": 1,
  "default_to_cache": 1,
  "cache_size": 123
}
```

dns 模板删除

Action: slb.profile.dns.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.dns.del

请求 body:

```
{"name": "p1"}
```

FTP 模板

ftp 模板列表

Action: slb.profile.ftp.list

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.ftp.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模板名称
active_mode_port	整数	1-65535	主动模式端口
disable_active_mode	整数	0-1	禁用主动模式: 0 不禁用, 1 禁用
disable_passive_mode	整数	0-1	禁用被动模式: 0 不禁用, 1 禁用

响应举例:

```
{
  "name": "p1",
  "active_mode_port": 20,
  "disable_active_mode": 0,
  "disable_passive_mode": 0
}
```

ftp 模板获取

Action: slb.profile.ftp.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.ftp.get

请求 body:

```
{
  "name": "p1"
}
```

响应参数:

名称	类型	范围	含义
----	----	----	----

name	字符串	长度 1-63	模板名称
active_mode_port	整数	1-65535	主动模式端口
disable_active_mode	整数	0-1	禁用主动模式：0 不禁用，1 禁用
disable_passive_mode	整数	0-1	禁用被动模式：0 不禁用，1 禁用

响应举例：

```
{
  "name": "p1",
  "active_mode_port": 20,
  "disable_active_mode": 0,
  "disable_passive_mode": 0
}
```

ftp 模板增加

Action: slb.profile.ftp.add

请求参数：

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
active_mode_port	整数	1-65535	主动模式端口	否	缺省值： 20
disable_active_mode	整数	0-1	禁用主动模式	否	0 不禁用，1 禁用，缺省值:0
disable_passive_mode	整数	0-1	禁用被动模式	否	0 不禁用，1 禁用，缺省值:0

请求举例：

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.ftp.add>
 请求 body:

```
{
  "name": "p1",
  "active_mode_port": 20,
  "disable_active_mode": 0,
  "disable_passive_mode": 0
}
```

ftp 模板编辑

Action: slb.profile.ftp.edit

请求参数：

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	
active_mode_port	整数	1-65535	主动模式端口	否	缺省值： : 不修改
disable_active_mode	整数	0-1	禁用主动模式	否	0 不禁用，1 禁用，缺省值:0

disable_passive_mode	整数	0-1	禁用被动模式	否	0 不禁用, 1 禁用, 缺省值:0
-----------------------------	----	-----	--------	---	--------------------

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.ftp.edit

请求 body:

```
{
  "name": "p1",
  "active_mode_port": 2000,
  "disable_active_mode": 0,
  "disable_passive_mode": 0
}
```

ftp 模板删除

Action: slb.profile.ftp.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.ftp.del

请求 body:

```
{"name": "p1"}
```

SIP 模板

sip 模板列表

Action: slb.profile.sip.list

请求参数: 无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.sip.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
header_del	字符串	长度 1-63	头删除
header_add	字符串	长度 1-63	头插入
header_mod_from	字符串	长度 1-63	头替换, 从
header_mod_to	字符串	长度 1-63	头替换, 到

registrar_pool	字符串	长度 1-63	注册服务池
timeout	整数	1-200	超时，单位分钟

响应举例：

```
[{
  "name": "p1",
  "header_del": "del",
  "header_add": "ins:ins",
  "header_mod_from": "aaa",
  "header_mod_to": "bbb",
  "registrar_pool": "",
  "timeout": 30
}]
```

sip 模板获取

Action: slb.profile.sip.get

请求参数：

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例：

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.sip.get>
 请求 body:

```
{
  "name": "p1"
}
```

响应参数：

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
header_del	字符串	长度 1-63	头删除
header_add	字符串	长度 1-63	头插入
header_mod_from	字符串	长度 1-63	头替换，从
header_mod_to	字符串	长度 1-63	头替换，到
registrar_pool	字符串	长度 1-63	注册服务池
timeout	整数	1-200	超时，单位分钟

响应举例：

```
[
  {
    "name": "p1",
    "header_del": "del",
    "header_add": "ins:ins",
    "header_mod_from": "aaa",
    "header_mod_to": "bbb",
    "registrar_pool": "",
    "timeout": 30
  }
]
```

```
}  
]
```

sip 模板增加

Action: slb.profile.sip.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
header_del	字符串	长度 1-63	头删除	否	缺省值:空
header_add	字符串	长度 1-63	头插入	否	缺省值:空
header_mod_from	字符串	长度 1-63	头替换, 从	否	缺省值:空
header_mod_to	字符串	长度 1-63	头替换, 到	否	缺省值:空
registrar_pool	字符串	长度 1-63	注册服务池	否	缺省值:空
timeout	整数	1-200	超时, 单位分钟	否	缺省值:30

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.sip.add>

请求 body:

```
{  
  "name": "p1",  
  "header_del": "del",  
  "header_add": "ins:ins",  
  "header_mod_from": "aaa",  
  "header_mod_to": "bbb",  
  "registrar_pool": "",  
  "timeout": 30  
}
```

sip 模板编辑

Action: slb.profile.sip.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
header_del	字符串	长度 1-63	头删除	否	缺省值:空
header_add	字符串	长度 1-63	头插入	否	缺省值:空
header_mod_from	字符串	长度 1-63	头替换, 从	否	缺省值:空
header_mod_to	字符串	长度 1-63	头替换, 到	否	缺省值:空
registrar_pool	字符串	长度 1-63	注册服务池	否	缺省值:空
timeout	整数	1-200	超时, 单位分钟	否	缺省值:30

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.sip.edit>

请求 body:

```
{
  "name": "p1",
  "header_del": "del",
  "header_add": "ins:ins",
  "header_mod_from": "aaa",
  "header_mod_to": "bbb",
  "registrar_pool": "",
  "timeout": 30
}
```

sip 模板删除

Action: slb.profile.sip.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.sip.del

请求 body:

```
{
  "name": "p1"
}
```

快速四层模板

快速四层模板列表

Action: slb.profile.fastl4.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.fastl4.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
fin_timeout	整数	1-60	fin 超时
timeout	整数	60-100000	空闲超时
reset_timeout	整数	0-31	超时重置
half_close_timeout	整数	0,60-15000	fin 等待
retransmit_number	整数	1-16	最大重传
syn_retransmit_number	整数	1-16	最大 syn 重传

time_wait	整数	0-600000	等待时间，单位为 ms
receive_buff	整数	1-2000000000	接收缓冲区
send_buffer	整数	1-2000000000	发送缓冲区
start_win_size	整数	0-65535	接收窗口
nagle	整数	0-1	nagle 算法;0 禁用, 1 使能
window_scale	整数	0-14	tcp 窗口比例
optts	整数	0-1	tcp 时间戳;0 禁用, 1 使能
tcp_accelerate_client	整数	0-1	tcp 加速;0 禁用, 1 使能
tcp_accelerate_client_cwnd	整数	1-10	网络冲突初始化窗口
tcp_accelerate_server	整数	0-1	服务器慢路径;0 禁用, 1 使能
tcp_accelerate_server_cwnd	整数	1-10	服务器冲突初始化窗口
mss	整数	0,128-4312	最大分段
rstnode	整数	0-1	服务器 rst;0 禁用, 1 使能
rstclient	整数	0-1	客户端 rst;0 禁用, 1 使能

响应举例:

```

[[
  "name": "p1",
  "fin_timeout": 8,
  "timeout": 300,
  "reset_timeout": 0,
  "half_close_timeout": 345,
  "retransmit_number": 3,
  "syn_retransmit_number": 3,
  "time_wait": 0,
  "receive_buff": 85000,
  "send_buffer": 15000,
  "start_win_size": 4096,
  "nagle": 1,
  "window_scale": 2,
  "optts": 1,
  "optsack": 1,
  "tcp_accelerate_client": 1,
  "tcp_accelerate_client_cwnd": 4,
  "tcp_accelerate_server": 0,
  "tcp_accelerate_server_cwnd": 4,
  "mss": 432,
  "rstnode": 1,
  "rstclient": 1
]]

```

快速四层模板获取

Action: `slb.profile.fastl4.get`

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	模板名称	是	
------	-----	----------	------	---	--

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.fast14.
 get

请求 body:

```
{
  "name": "p1"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
fin_timeout	整数	1-60	fin 超时
timeout	整数	60-100000	空闲超时
reset_timeout	整数	0-31	超时重置
half_close_timeout	整数	0,60-15000	fin 等待
retransmit_number	整数	1-16	最大重传
syn_retransmit_number	整数	1-16	最大 syn 重传
time_wait	整数	0-600000	等待时间, 单位为 ms
receive_buff	整数	1-2000000000	接收缓冲区
send_buffer	整数	1-2000000000	发送缓冲区
start_win_size	整数	0-65535	接收窗口
nagle	整数	0-1	nagle 算法;0 禁用, 1 使能
window_scale	整数	0-14	tcp 窗口比例
optts	整数	0-1	tcp 时间戳;0 禁用, 1 使能
tcp_accelerate_client	整数	0-1	tcp 加速;0 禁用, 1 使能
tcp_accelerate_client_cnwd	整数	1-10	网络冲突初始化窗口
tcp_accelerate_server	整数	0-1	服务器慢路径;0 禁用, 1 使能
tcp_accelerate_server_cnwd	整数	1-10	服务器冲突初始化窗口
mss	整数	0,128-4312	最大分段
rstnode	整数	0-1	服务器 rst;0 禁用, 1 使能
rstclient	整数	0-1	客户端 rst;0 禁用, 1 使能

响应举例:

```
{
  "name": "p1",
  "fin_timeout": 8,
  "timeout": 300,
  "reset_timeout": 0,
  "half_close_timeout": 345,
  "retransmit_number": 3,
  "syn_retransmit_number": 3,
  "time_wait": 0,
  "receive_buff": 85000,
}
```

```
"send_buffer": 15000,
"start_win_size": 4096,
"nagle": 1,
>window_scale": 2,
"optts": 1,
"optsack": 1,
"tcp_accelerate_client": 1,
"tcp_accelerate_client_cwnd": 4,
"tcp_accelerate_server": 0,
"tcp_accelerate_server_cwnd": 4,
"mss": 432,
"rstnode": 1,
"rstclient": 1 }
```

快速四层模板增加

Action: slb.profile.fastl4.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
fin_timeout	整数	1-60	fin 超时	否	缺省值:5
timeout	整数	60-100000	空闲超时	否	缺省值:600
reset_timeout	整数	0-30	超时重置	否	缺省值:0
half_close_timeout	整数	0,60-15000	fin 等待	否	缺省值:0
retransmit_number	整数	1-16	最大重传	否	缺省值:3
syn_retransmit_number	整数	1-16	最大 syn 重传	否	缺省值:5
time_wait	整数	0-600000	等待时间, 单位为 ms	否	缺省值:0
receive_buff	整数	1-2000000000	接收缓冲区	否	缺省值:51200
send_buffer	整数	1-2000000000	发送缓冲区	否	缺省值:51200
start_win_size	整数	0-65535	接收窗口	否	缺省值:0
nagle	整数	0-1	nagle 算法	否	0 禁用, 1 使能, 缺省值:0
window_scale	整数	0-12	tcp 窗口比例	否	缺省值:0
optts	整数	0-1	tcp 时间戳	否	0 禁用, 1 使能, 缺省值:1
tcp_accelerate_client	整数	0-1	tcp 加速	否	0 禁用, 1 使能, 缺省值:1
tcp_accelerate_client_cwnd	整数	1-10	网络冲突初始化窗口	否	缺省值:4
tcp_accelerate_server	整数	0-1	服务器慢路径	否	0 禁用, 1 使能, 缺省值:0
tcp_accelerate_server_cwnd	整数	1-10	服务器冲突初始化窗口	否	缺省值:4
mss	整数	0,128-4312	最大分段	否	缺省值:1460
rstnode	整数	0-1	服务器 rst	否	0 禁用, 1 使能, 缺省值:0
rstclient	整数	0-1	客户端 rst	否	0 禁用, 1 使能, 缺省值:0

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.fastl4.add
```

请求 body:

```

{
  "name": "pl",
  "fin_timeout": 8,
  "timeout": 300,
  "reset_timeout": 0,
  "half_close_timeout": 345,
  "retransmit_number": 3,
  "syn_retransmit_number": 3,
  "time_wait": 3,
  "receive_buff": 85000,
  "send_buffer": 15000,
  "start_win_size": 4096,
  "nagle": 1,
  "window_scale": 2,
  "optts": 1,
  "tcp_accelerate_client": 1,
  "tcp_accelerate_client_cwnd": 4,
  "tcp_accelerate_server": 0,
  "tcp_accelerate_server_cwnd": 4,
  "mss": 432,
  "rstnode": 1,
  "rstclient": 1
}

```

快速四层模板编辑

Action: slb.profile.fastl4.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
fin_timeout	整数	1-60	fin 超时	否	缺省值:不修改
timeout	整数	60-100000	空闲超时	否	缺省值:不修改
reset_timeout	整数	0-31	超时重置	否	缺省值:不修改
half_close_timeout	整数	0,60-15000	fin 等待	否	缺省值:不修改
retransmit_number	整数	1-16	最大重传	否	缺省值:不修改
syn_retransmit_number	整数	1-16	最大 syn 重传	否	缺省值:不修改
time_wait	整数	0-600000	等待时间, 单位为 ms	否	缺省值:不修改
receive_buff	整数	1-2000000000	接收缓冲区	否	缺省值:不修改
send_buffer	整数	1-2000000000	发送缓冲区	否	缺省值:不修改
start_win_size	整数	0-65535	接收窗口	否	缺省值:不修改
nagle	整数	0-1	nagle 算法	否	0 禁用, 1 使能, 缺省值:不修改
window_scale	整数	0-12	tcp 窗口比例	否	缺省值:不修改
optts	整数	0-1	tcp 时间戳	否	0 禁用, 1 使能, 缺省值:不修改

tcp_accelerate_client	整数	0-1	tcp 加速	否	0 禁用, 1 使能, 缺省值: 不修改
tcp_accelerate_client_cwnd	整数	1-10	网络冲突初始化窗口	否	缺省值:不修改
tcp_accelerate_server	整数	0-1	服务器慢路径	否	0 禁用, 1 使能, 缺省值: 不修改
tcp_accelerate_server_cwnd	整数	1-10	服务器冲突初始化窗口	否	缺省值:不修改
mss	整数	0,128-4312	最大分段	否	缺省值:不修改
rstnode	整数	0-1	服务器 rst	否	0 禁用, 1 使能, 缺省值: 不修改
rstclient	整数	0-1	客户端 rst	否	0 禁用, 1 使能, 缺省值: 不修改

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.fastl4.edit

请求 body:

```
{
  "name": "pl",
  "fin_timeout": 8,
  "timeout": 300,
  "reset_timeout": 0,
  "half_close_timeout": 345,
  "retransmit_number": 3,
  "syn_retransmit_number": 3,
  "time_wait": 3,
  "receive_buff": 85000,
  "send_buffer": 15000,
  "start_win_size": 4096,
  "nagle": 1,
  "window_scale": 2,
  "optts": 1,
  "tcp_accelerate_client": 1,
  "tcp_accelerate_client_cwnd": 4,
  "tcp_accelerate_server": 0,
  "tcp_accelerate_server_cwnd": 4,
  "mss": 432,
  "rstnode": 1,
  "rstclient": 1
}
```

快速四层模板删除

Action: slb.profile.fastl4.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.fast14.del
 请求 body:
 {
 "name": "p1"
 }

TCP 模板

tcp 模板列表

Action: slb.profile.tcp.list

请求参数: 无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.tcp.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
timeout	整数	60-100000	空闲超时
reset_timeout	整数	0-31	超时重置
half_close_timeout	整数	0,60-15000	fin 等待
insertcip	整数	0-1	插入客户端 ip;0 禁用, 1 使能
rstnode	整数	0-1	服务器 rst;0 禁用, 1 使能
rstclient	整数	0-1	客户端 rst;0 禁用, 1 使能
start_win_size	整数	0-65535	接收窗口
time_wait	整数	0-600000	等待时间, 单位为 ms

响应举例:

```
[[
  "name": "p1",
  "timeout": 3600,
  "reset_timeout": 23,
  "start_win_size": 45,
  "half_close_timeout": 675,
  "insertcip": 1,
  "rstnode": 1,
  "rstclient": 1,
  "time_wait": 0
]]
```

```
}]
```

tcp 模板获取

Action: slb.profile.tcp.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.tcp.get

请求 body:

```
{  
  "name": "p1"  
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
timeout	整数	60-100000	空闲超时
reset_timeout	整数	0-31	超时重置
half_close_timeout	整数	0,60-12000	fin 等待
insertcip	整数	0-1	插入客户端 ip;0 禁用, 1 使能
rstnode	整数	0-1	服务器 rst;0 禁用, 1 使能
rstclient	整数	0-1	客户端 rst;0 禁用, 1 使能
start_win_size	整数	0-65535	接收窗口
time_wait	整数	0-600000	等待时间, 单位为 ms

响应举例:

```
{  
  "name": "p1",  
  "timeout": 3600,  
  "reset_timeout": 23,  
  "start_win_size": 45,  
  "half_close_timeout": 675,  
  "insertcip": 1,  
  "rstnode": 1,  
  "rstclient": 1,  
  "time_wait": 0  
}
```

tcp 模板增加

Action: slb.profile.tcp.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
timeout	整数	60-100000	空闲超时	否	默认 120
reset_timeout	整数	0-31	超时重置	否	默认 0
half_close_timeout	整数	0,60-12000	fin 等待	否	默认 0
insertcip	整数	0-1	插入客户端 ip	否	1 使能, 0 禁用, 默认 0
rstnode	整数	0-1	服务器 rst	否	1 使能, 0 禁用, 默认 0
rstclient	整数	0-1	客户端 rst	否	1 使能, 0 禁用, 默认 0
start_win_size	整数	0-65535	接收窗口	否	默认 0
time_wait	整数	0-600000	等待时间, 单位为 ms	否	默认 0

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.tcp.add>
 请求 body:

```
{
  "name": "p1",
  "timeout": 3600,
  "reset_timeout": 23,
  "start_win_size": 45,
  "half_close_timeout": 675,
  "insertcip": 1,
  "rstnode": 1,
  "rstclient": 1,
  "time_wait": 0
}
```

tcp 模板编辑

Action: slb.profile.tcp.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
timeout	整数	60-100000	空闲超时	否	缺省值:不修改
reset_timeout	整数	0-31	超时重置	否	缺省值:不修改
half_close_timeout	整数	0,60-12000	fin 等待	否	缺省值:不修改
insertcip	整数	0-1	插入客户端 ip	否	1 使能, 0 禁用, 缺省值:不修改
rstnode	整数	0-1	服务器 rst	否	1 使能, 0 禁用, 缺省值:不修改
rstclient	整数	0-1	客户端 rst	否	1 使能, 0 禁用, 缺省值:不修改
start_win_size	整数	0-65535	接收窗口	否	缺省值:不修改
time_wait	整数	0-600000	等待时间, 单位为 ms	否	默认 0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.tcp.edi
t

请求 body:

```
{  
  "name": "p1",  
  "timeout": 3600,  
  "reset_timeout": 23,  
  "start_win_size": 45,  
  "half_close_timeout": 675,  
  "insertcip": 1,  
  "rstnode": 1,  
  "rstclient": 1,  
  "time_wait": 0  
}
```

tcp 模板删除

Action: slb.profile.tcp.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.tcp.del
请求 body:

```
{  
  "name": "p1"  
}
```

UDP 模板

udp 模板列表

Action: slb.profile.udp.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.udp.lis
t

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
timeout	整数	60-120000	空闲超时

aging	字符串	长度 1-63	快速老化选项: fast 立即老化, delayed 延迟老化 空表示不快速老化
delayed_timeout	整数	1-31	快速老化超时:只有在 aging 参数为 delayed 时生效
node_reselect	整数	0-1	服务器重选; 1 使能, 0 禁用

响应举例:

```
[[
  "name": "p1",
  "timeout": 120,
  "aging": "delayed",
  "delayed_timeout": 23,
  "node_reselect": 1
]]
```

udp 模板获取

Action: slb.profile.udp.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.udp.get>

请求 body:

```
{
  "name": "p1"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	模版名称
timeout	整数	60-120000	空闲超时
aging	字符串	长度 1-63	快速老化选项: fast 立即老化, delayed 延迟老化 空表示不快速老化
delayed_timeout	整数	1-31	快速老化超时:只有在 aging 参数为 delayed 时生效
node_reselect	整数	0-1	服务器重选; 1 使能, 0 禁用

响应举例:

```
{
  "name": "p1",
  "timeout": 120,
  "aging": "delayed",
  "delayed_timeout": 23,
  "node_reselect": 1
}
```

udp 模板增加

Action: slb.profile.udp.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
timeout	整数	60-120000	空闲超时	否	缺省值:120
aging	字符串	长度 1-63	快速老化选项	否	fast 立即老化, delayed 延迟老化, 空表示不快速老化 缺省值:空
delayed_timeout	整数	1-31	快速老化超时:	否	只有在 aging 参数为 delayed 时生效 缺省值:空
node_reselect	整数	0-1	服务器重选	否	0 禁用, 1 使能, 缺省值:0

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.udp.add

请求 body:

```
{
  "name": "p1",
  "timeout": 120,
  "aging": "delayed",
  "delayed_timeout": 23,
  "node_reselect": 1
}
```

udp 模板编辑

Action: slb.profile.udp.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	
timeout	整数	60-120000	空闲超时	否	缺省值:120
aging	字符串	长度 1-63	快速老化选项	否	fast 立即老化, delayed 延迟老化, 空表示不快速老化 缺省值:空
delayed_timeout	整数	1-31	快速老化超时:	否	只有在 aging 参数为 delayed 时生效 缺省值:空
node_reselect	整数	0-1	服务器重选	否	0 禁用, 1 使能, 缺省值:0

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.udp.edi

t
请求 body:

```
{
  "name": "p1",
```

```

"timeout": 120,
"aging": "delayed",
"delayed_timeout": 23,
"node_reselect": 1
}

```

udp 模板删除

Action: slb.profile.udp.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模板名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.profile.udp.del>
 请求 body:

```

{
  "name": "p1"
}

```

虚拟服务模板

虚拟服务模板列表

Action: slb.profile.vs.list

请求参数:无

请求举例:

Get
<http://10.2.120.21/adcap/v2.0/?authkey=1021a0c521314f602f77749a662a0b&action=slb.profile.vs.list>

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	虚拟服务模板名称
ignored_tcp_msl	整数	0-1	忽略 MSL; 0 关闭, 1 开启
reset_unknown_conn	整数	0-1	忽略非法连接; 0 关闭, 1 开启
reset_l7_on_failover	整数	0-1	节点选择失败重置; 0 关闭, 1 开启
syn_otherflags	整数	0-1	非 Syn 新建连接; 0 关闭, 1 开启
conn_limit_switch	整数	0-1	连接限制; 0 关闭, 1 开启
conn_limit	整数	1-8000000	连接数
conn_over_limit_action	整数	0-1	连接数超限重置; 0 关闭, 1 开启

log_conn_limit_exceed	整数	0-1	连接数超限日志; 0 关闭, 1 开启
conn_rate_limit_switch	整数	0-1	连接速率限制; 0 关闭, 1 开启
conn_rate_limit	整数	1-1048575	连接速率值
conn_rate_over_limit_action	整数	0-1	连接速率超限重置; 0 关闭, 1 开启
conn_rate_unit	整数	0-1	连接速率计算间隔; 0 代表 second, 1 代表 100ms
log_conn_rate_limit_exceed	整数	0-1	连接速率超限日志; 0 关闭, 1 开启

响应举例:

```
[
{
  "name": "p1",
  "ignored_tcp_msl": 1,
  "reset_unknown_conn": 1,
  "reset_l7_on_failover": 1,
  "syn_otherflags": 1,
  "conn_limit_switch": 1,
  "conn_limit": 8000000,
  "conn_over_limit_action": 1,
  "log_conn_limit_exceed": 1,
  "conn_rate_limit_switch": 1,
  "conn_rate_limit": 1048575,
  "conn_rate_over_limit_action": 1,
  "conn_rate_unit": 0,
  "log_conn_rate_limit_exceed": 1
}
]
```

虚拟服务模板获取

Action: slb.profile.vs.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟服务模板名称	是	

请求举例:

POST
<http://10.2.120.21/adcap/v2.0/?authkey=1021a0c521314f602f77749a662a0b&action=slb.profile.vs.get>

请求 Body:

```
{
  "name": "p1"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	虚拟服务模板名称
ignored_tcp_msl	整数	0-1	忽略 MSL；0 关闭，1 开启
reset_unknown_conn	整数	0-1	忽略非法连接；0 关闭，1 开启
reset_l7_on_failover	整数	0-1	节点选择失败重置；0 关闭，1 开启
syn_otherflags	整数	0-1	非 Syn 新建连接；0 关闭，1 开启
conn_limit_switch	整数	0-1	连接限制；0 关闭，1 开启
conn_limit	整数	1-8000000	连接数
conn_over_limit_action	整数	0-1	连接数超限重置；0 关闭，1 开启
log_conn_limit_exceed	整数	0-1	连接数超限日志；0 关闭，1 开启
conn_rate_limit_switch	整数	0-1	连接速率限制；0 关闭，1 开启
conn_rate_limit	整数	1-1048575	连接速率值
conn_rate_over_limit_action	整数	0-1	连接速率超限重置；0 关闭，1 开启
conn_rate_unit	整数	0-1	连接速率计算间隔；0 代表 second，1 代表 100ms
log_conn_rate_limit_exceed	整数	0-1	连接速率超限日志；0 关闭，1 开启

响应举例：

```
{
  "name": "p1",
  "ignored_tcp_msl": 1,
  "reset_unknown_conn": 1,
  "reset_l7_on_failover": 1,
  "syn_otherflags": 1,
  "conn_limit_switch": 1,
  "conn_limit": 8000000,
  "conn_over_limit_action": 1,
  "log_conn_limit_exceed": 1,
  "conn_rate_limit_switch": 1,
  "conn_rate_limit": 1048575,
  "conn_rate_over_limit_action": 1,
  "conn_rate_unit": 0,
  "log_conn_rate_limit_exceed": 1
}
```

虚拟服务模板增加

Action: **slb.profile.vs.add**

请求参数

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟服务模板 名称	是	
ignored_tcp_msl	整数	0-1	忽略 MSL; 0 关 闭, 1 开启	是	0 关闭, 1 开启
reset_unknown_c onn	整数	0-1	忽略非法连接; 0 关闭, 1 开启	是	0 关闭, 1 开启
reset_l7_on_fai lover	整数	0-1	节点选择失败 重置; 0 关闭, 1 开启	是	0 关闭, 1 开启
syn_otherflags	整数	0-1	非 Syn 新建连 接; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_limit_swit ch	整数	0-1	连接限制; 0 关 闭, 1 开启	是	0 关闭, 1 开启
conn_limit	整数	1-80000 00	连接数	是	
conn_over_limit _action	整数	0-1	连接数超限重 置; 0 关闭, 1 开启	是	0 关闭, 1 开启
log_conn_limit_ exceed	整数	0-1	连接数超限日 志; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_rate_limit _switch	整数	0-1	连接速率限制; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_rate_limit	整数	1-10485 75	连接速率值	是	
conn_rate_over_ limit_action	整数	0-1	连接速率超限 重置; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_rate_unit	整数	0-1	连接速率计算 间隔; 0 代表 second, 1 代表 100ms	是	0 代表 second, 1 代表 100ms
log_conn_rate_l imit_exceed	整数	0-1	连接速率超限 日志; 0 关闭, 1 开启	是	0 关闭, 1 开启

请求举例

POST
<http://10.2.120.21/adcap/v2.0/?authkey=1021a0c521314f602f77749a662a0b&action=slb.profile.vs.add>
 请求 body:

```
{
    "name": "p1",
    "ignored_tcp_msl": 1,
```

```

"reset_unknown_conn": 1,
"reset_l7_on_failover": 1,
"syn_otherflags": 1,
"conn_limit_switch": 1,
"conn_limit": 8000000,
"conn_over_limit_action": 1,
"log_conn_limit_exceed": 1,
"conn_rate_limit_switch": 1,
"conn_rate_limit": 1048575,
"conn_rate_over_limit_action": 1,
"conn_rate_unit": 1,
"log_conn_rate_limit_exceed": 1
}

```

虚拟服务模板编辑

Action: slb.profile.vs.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟服务模板名称	是	
ignored_tcp_msl	整数	0-1	忽略MSL; 0 关闭, 1 开启	是	0 关闭, 1 开启
reset_unknown_conn	整数	0-1	忽略非法连接; 0 关闭, 1 开启	是	0 关闭, 1 开启
reset_l7_on_failover	整数	0-1	节点选择失败重置; 0 关闭, 1 开启	是	0 关闭, 1 开启
syn_otherflags	整数	0-1	非 Syn 新建连接; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_limit_switch	整数	0-1	连接限制; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_limit	整数	1-8000000	连接数	是	
conn_over_limit_action	整数	0-1	连接数超限重置; 0 关闭, 1 开启	是	0 关闭, 1 开启
log_conn_limit_exceed	整数	0-1	连接数超限日志; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_rate_limit_switch	整数	0-1	连接速率限制; 0 关闭, 1 开启	是	0 关闭, 1 开启
conn_rate_limit	整数	1-1048575	连接速率值	是	
conn_rate_over_	整数	0-1	连接速率超限	是	0 关闭, 1 开启

limit_action			重置：0 关闭， 1 开启		
conn_rate_unit	整数	0-1	连接速率计算 间隔：0 代表 second, 1 代表 100ms	是	0 代表 second, 1 代表 100ms
log_conn_rate_l imit_exceed	整数	0-1	连接速率超限 日志：0 关闭， 1 开启	是	0 关闭, 1 开启

请求举例：

POST

<http://10.2.120.21/adcap/v2.0/?authkey=1021a0c521314f602f77749a662a0b&action=slb.profile.vs.edit>

请求 body:

```
{
    "name": "p1",
    "ignored_tcp_msl": 1,
    "reset_unknown_conn": 1,
    "reset_l7_on_failover": 1,
    "syn_otherflags": 1,
    "conn_limit_switch": 1,
    "conn_limit": 6000000,
    "conn_over_limit_action": 1,
    "log_conn_limit_exceed": 1,
    "conn_rate_limit_switch": 1,
    "conn_rate_limit": 1048575,
    "conn_rate_over_limit_action": 1,
    "conn_rate_unit": 1,
    "log_conn_rate_limit_exceed": 1
}
```

虚拟服务模板删除

Action: slb.profile.vs.del

请求参数：

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	虚拟服务模板名 称	是	

请求举例：

POST

<http://10.2.120.21/adcap/v2.0/?authkey=1021a0c521314f602f77749a662a0b&action=slb.profile.vs.del>

请求 body:

```
{
    "name": "p1"
}
```

健康检查

健康检查添加

Action: slb.healthcheck.add

公共请求参数:所有种类的健康检查都具有的参数,后续各种健康检查不再列举这些参数

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	健康检查名称	是	唯一
retry	整数	一般健康检查: 1-5 Pro 类型: 0	重试次数	否	缺省值: 3 Pro 类型必须为 0
interval	整数	1-180	间隔时间,单位秒	否	缺省值: 5
timeout	整数	一般健康检查: 1-12 Pro 类型: 1-1800	超时时间,单位秒	否	缺省值: 5 Pro 类型的值必须大于 interval
type	字符串		健康检查类型,	否	缺省值: icmp 支持字段: icmp:ICMP 健康检查 http:HTTP 健康检查 http-pro:HTTP-PRO 健康检查 arp:ARP 健康检查 database:数据库健康检查 dns:DNS 健康检查 ftp:FTP 健康检查 https:HTTPS 健康检查 imap:IMAP 健康检查 ldap:LDAP 健康检查 ntp:NTP 健康检查 pop3:POP3 健康检查 radius:RADIUS 健康检查 rtsp:RTSP 健康检查 sip:SIP 健康检查 smtp:SMTP 健康检查 snmp:SNMP 健康检查 tcp:TCP 健康检查 tcp-pro:TCP-PRO 健康检查 udp:UDP 健康检查 udp-pro:UDP-PRO 健康检查 script:自定义脚本健康检查 combo:组合健康检查
auto_disable	整数	0,1	自动禁用。	否	未配置此项时,获取配置也无此项; 0 不开启, 1 开启。缺省值: 0
alias_ipv4_src	ipv4		源地址,指定该参数后,	否	未配置此项时,获取配置也无

	地址		检查时以该地址作为源地址		此项; 缺省值: 无
interface	字符串		源接口,指定该参数后,检查时以该接口作为出接口	否	未配置此项时,获取配置也无此项; 缺省值: 无
alias_ipv4	ipv4地址		IPv4 地址别名,指定该参数后,检查时以该地址作为目的地址	否	未配置此项时,获取配置也无此项; 缺省值: 无
alias_ipv6	IPv6地址		IPv6 地址别名,指定该参数后,检查时以该地址作为目的地址	否	未配置此项时,获取配置也无此项; 缺省值: 无
alias_port	整数	1-65535	端口别名, 指定该参数后,检查时以该端口作为目的端口	否	未配置此项时,获取配置也无此项; 仅对 TCP/UDP 协议或基于 TCP/UDP 之上的协议有效 缺省值: 无
port	整数	1-65535	在关联到节点时,使用该端口作为目的端口检查,其他情况使用关联的端口做检查	否	仅对 TCP/UDP 协议或基于 TCP/UDP 之上的协议有效 缺省值:各应用协议的默认端口
up_check_cnt	整数	1-10	对 应 命 令 行 的 up-check-cnt, 表示检测对象 up 前最少检测成功的个数	否	默认参数为 1, 即只要有一次健康检查成功, 就认为检查对象是 up 的
wait_all_retry	整数	0,1	对 应 命 令 行 的 wait-all-retry, 表示等待所有的 retry 尝试次数都失败了, 才会将检测对象标记为 down		默认为 0: 表示不开启, 1: 表示开启

由于各种类型的健康检查支持的参数不一样,下面分类型说明:

ICMP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	icmp	对于 ICMP 健康检查类型, 固定为 icmp	是	
mode	字符串	transparent	表明设备是属于透明模式下	否	仅仅是为了表明模式, 可以不填
icmp_alias_addr	ipv4/ipv6 地址		透明模式下, icmp 的透明地址	否	空的话就表明不是透明模式

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add

请求 body

```
{
  "name": "ping",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "icmp"
}
```

HTTP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	http	对于 HTTP 健康检查类型, 固定为 http	是	
host	字符串	长度 1-63	HTTP 头部 HOST 字段	否	缺省值:自动
url	字符串	长度 1-127	请求方法和 URL	否	请求方法支持 GET/POST 缺省值: GET /
post_data	字符串	长度 1-255	发送 body, 当 url 中方法为 POST 时有效, 和 post_data 两者只能配置一个	否	缺省值:空
post_file	字符串	长度 1-31	发送 body 文件名, 当 url 中方法为 POST 时有效, 和 post_data 两者只能配置一个	否	缺省值:空
username	字符串	长度 1-31	认证用户名,当 HTTP 需要登录认证时使用	否	缺省值:空
password	字符串	长度 1-31	认证密码,当 HTTP 需要登录认证时使用	否	缺省值:空
code	字符串	长度 1-31	HTTP 返回码,code 和 pattern 只能使用一个	否	格式为 100-899 之间的数字组成的字符串,可以使用逗号(,)或者连接符(-)连接; 缺省值:200
pattern	字符串	长度 1-255	接收字符串,code 和 pattern 只能使用一个	否	缺省值:空
pattern_disable_str	字符串	长度 1-256	接收禁用字符串, 没有收到 pattern 字符串但是收到该字符串之后, 对健康检查目标执行软关机操作。必须配合 pattern 参数使用	否	缺省值:空
server_fail_code	字符串	长度 1-31	进入维护模式的响应码	否	格式为 100-899 之间的数字组成的字符串,可以使用逗号(,)或者连接符(-)连接; 添加时缺省为空, 编辑时缺省不修改

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add

请求 body

```
{
  "name": "http",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "auto_disable": 0,
  "alias_ipv4_src": "192.168.1.1",
  "interface": "Ethernet 0/1",
  "alias_ipv4": "1.2.3.4",
  "alias_port": 889,
  "alias_ipv6": "2001::1",
  "type": "http",
  "port": 80,
  "host": "ahost",
  "url": "POST /index",
  "post_data": "mybodystring",
  "username": "username",
  "password": "pass",
  "code": "202"
}
```

HTTP-PRO 健康检查添加

Action: slb.healthcheck-pro.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	http-pro	对于 HTTP 健康检查类型, 固定为 http-pro	是	
retry	整数	Pro 类型: 0	重试次数	是	Pro 类型必须设为 0
interval	整数	1-180	间隔时间,单位秒	是	缺省值: 5
timeout	整数	Pro 类型: 1-1800	超时时间,单位秒	是	缺省值: 5 Pro 类型的值必须大于 interval
host	字符串	长度 1-63	HTTP 头部 HOST 字段	否	缺省值: 自动
url	字符串	长度 1-127	请求方法和 URL	否	请求方法支持 GET/POST 缺省值: GET /
post_data	字符串	长度 1-255	发送 body, 当 url 中方法为 POST 时有效, 和 post_data 两者只能配置一个	否	缺省值: 空
post_file	字符串	长度 1-31	发送 body 文件名, 当 url 中方法为 POST 时有效, 和 post_data 两者只能配置一个	否	缺省值: 空
code	字符串	长度 1-31	HTTP 返回码,code 和 pattern 只能使用一个	否	格式为 100-899 之间的数字组成的字符

					串,可以使用逗号(,)或者连接符(-)连接;缺省值:200
pattern	字符串	长度 1-255	接收字符串, code 和 pattern 只能使用一个	否	缺省值:空
pattern_disable_str	字符串	长度 1-256	接收禁用字符串, 没有收到 pattern 字符串但是收到该字符串之后, 对健康检查目标执行软关机操作。必须配合 pattern 参数使用	否	缺省值:空
server_fail_code	字符串	长度 1-31	进入维护模式的响应码	否	格式为 100-899 之间的数字组成的字符串,可以使用逗号(,)或者连接符(-)连接;添加时缺省为空,编辑时缺省不修改

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
 请求 body

```
{
  "name": "http-pro",
  "retry": 0,
  "interval": 1,
  "timeout": 4,
  "auto_disable": 0,
  "alias_ipv4_src": "192.168.1.1",
  "interface": "Ethernet 1/0",
  "alias_ipv4": "1.2.3.4",
  "alias_port": 889,
  "alias_ipv6": "2001::1",
  "type": "http-pro",
  "port": 80,
  "host": "ahost",
  "url": "POST /index",
  "post_data": "mybodystring",
  "code": "202"
}
```

ARP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	arp	对于 ARP 健康检查类型, 固定为 arp	是	

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
{
  "name": "arp",
  "type": "arp"
}
```

数据库健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	arp	检查类型	是	对于 ARP 健康检查类型, 固定为 arp
database_name	字符串	长度 1-63	数据库名称	是	
username	字符串	长度 1-31	认证用户名	是	
password	字符串	长度 1-31	认证密码	是	
query	字符串		发送字符串	否	缺省值:空
response_str	字符串		接收字符串	否	缺省值:空
row	整数	1-10	接收行	否	缺省值:1
column	整数	1-10	接收列	否	缺省值:1
database_type	整数	1-4	数据库类型:	否	1:mssql;2: mysql;3:oracle;4:postgre;缺省值:2

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
{
  "name": "database",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "database",
  "database_name": "db",
  "username": "aaa",
  "password": "sss",
  "query": "",
  "response_str": "",
  "row": 1,
  "column": 1,
  "database_type": 4
}
```

DNS 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	dns	检查类型	是	对于 DNS 健康检查类型, 固定为 dns
domain	字符串	长度 1-63	请求域名	否	缺省值:www.baidu.com
record_type	字符串	长度 1-63	记录类型	否	A/CNAME/SOA/PTR/MX/TXT/AAAA 缺省值:A
wantreturn	字符串		返回 code	否	可以接收多种 code, code 范围 0-15 例如 "0, 1, 3-5"; 缺省值:空
want_ipv4	ipv4 地址		期望返回 IPv4 地址	否	IPv4 地址格式字符串,缺省值:空
want_ipv6	Ipv6 地址		期望返回 IPv6 地址	否	IPv6 地址格式字符串,缺省值:空
ip_addr	ipv4 地址		反向解析域名的 ip 地址	否	与 domain 互斥, 二者必选其一

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
```

```
{
  "name": "dns",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "dns",
  "port": 53,
  "domain": "www.baidu.com",
  "record_type": "A",
  "wantreturn": "1,2,3",
  "want_ipv4": "4.3.2.1"
}
```

FTP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	ftp	检查类型	是	对于 FTP 健康检查类型, 固定为 ftp
username	字符串	1-31	用户名	否	缺省值:空
password	字符串	1-31	密码	否	缺省值:空

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add

请求 body

```
{
  "name": "ftp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "ftp",
  "port": 21,
  "username": "",
  "password": ""
}
```

HTTPS 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	https	对于 HTTP 健康检查类型, 固定为 https	是	
port	整数	1-65534	端口,在关联到节点时,使用该端口作为目的端口检查,其他情况使用关联的端口做检查	否	缺省值:80
host	字符串	长度 1-63	HTTP 头部 HOST 字段	否	缺省值:自动
url	字符串	长度 1-127	请求方法和 URL	否	请求方法支持 GET/POST/HEAD 缺省值: GET /index
post_data	字符串	长度 1-255	发送 body, 当 url 中方法为 POST 时有效	否	缺省值:空
username	字符串	长度 1-31	认证用户名,当 HTTP 需要登录认证时使用	否	缺省值:空
password	字符串	长度 1-31	认证密码,当 HTTP 需要登录认证时使用	否	缺省值:空
code	字符串	长度 1-31	HTTP 返回码,code 和 pattern 只能使用一个	否	格式为 100-899 之间的数字组成的字符串,可以使用逗号(,)或者连接符(-)连接; 缺省值:200
pattern	字符串	长度 1-255	接收字符串,code 和 pattern 只能使用一个	否	缺省值:空
pattern_disable_str	字符串	长度 1-256	接收禁用字符串, 没有收到 pattern 字符串但是收到该字符串之后, 对健康检查目标执行软关机操作。必须配合 pattern 参数使用	否	缺省值:空
server_fail_code	字符串	长度 1-31	进入维护模式的响应码	否	格式为 100-899 之间的数字组成的字符串,可以使用逗号(,)或者连接符(-)连接; 添加时缺省为空,

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
 请求 body

```
{
  "name": "https",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "https",
  "port": 80,
  "host": "ahost",
  "url": "GET /index",
  "post_data": "mybodystring",
  "username": "username",
  "password": "pass",
  "code": "202"
}
```

IMAP 健康检查添加**Action: slb.healthcheck.add****请求参数:**

名称	类型	范围	含义	必选	备注
type	字符串	Imap	检查类型	是	对于 IMAP 健康检查类型, 固定为 imap
username	字符串	1-31	用户名	否	缺省值:空
password	字符串	1-31	密码	否	缺省值:空
plain_text	整数	0,1		否	0 禁用, 1 使能, 缺省值:0
login	整数	0,1		否	0 禁用, 1 使能, 缺省值:0
cram_md5	整数	0,1		否	0 禁用, 1 使能, 缺省值:0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
 请求 body

```
{
  "name": "imap",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "imap",
  "port": 143,
  "plain_text": 0,
  "login": 0,
}
```

```

    "cram_md5": 0,
    "username": "",
    "password": ""
}

```

LDAP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	ldap	检查类型	是	对于 LDAP 健康检查类型，固定为 ldap
username	字符串	1-31	用户名	否	缺省值:空
password	字符串	1-31	密码	否	缺省值:空
ssl	整数	0,1	使用 SSL	否	1:是;0:否; 缺省值:0

请求举例:

```

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```

```

{
    "name": "ldap",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "ldap",
    "port": 389,
    "ssl": 0,
    "password": ""
}

```

NTP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	ntp	检查类型	是	对于 NTP 健康检查类型，固定为 ntp

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
{
    "name": "ntp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
}

```

```

"type": "ntp",
"port": 123
}

```

POP3 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	pop3	检查类型	是	对于 POP3 健康检查类型, 固定为 pop3
username	字符串	1-31	用户名	否	缺省值:空
password	字符串	1-31	密码	否	缺省值:空

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```

```

{
  "name": "pop3",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "pop3",
  "port": 110,
  "username": "aas",
  "password": "dww"
}

```

RADIUS 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	radius	检查类型	是	对于 RADIUS 健康检查类型, 固定为 radius
username	字符串	1-31	用户名	否	缺省值:空
password	字符串	1-31	密码	否	缺省值:空
secret	字符串	1-31	共享密码	否	缺省值:adc

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```

```

{
  "name": "radius",
  "retry": 3,

```

```

"interval": 5,
"timeout": 5,
"type": "radius",
"port": 1812,
"username": "",
"password": "",
"secret": "secret"
}

```

RTSP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	rtsp	检查类型	是	对于 RTSP 健康检查类型, 固定为 rtsp
url	字符串	长度 1-63	URL	否	缺省值:/music.mp3

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
{
  "name": "rtsp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "rtsp",
  "port": 554,
  "url": "/music.mp3"
}

```

SIP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	sip	检查类型	是	对于 SIP 健康检查类型, 固定为 sip
register	整数	0,1	使用 register	否	1:是;0:否;缺省值:0

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body
{
  "name": "sip",
  "retry": 3,

```

```

"interval": 5,
"timeout": 5,
"type": "sip",
"port": 5060,
"register": 0
}

```

SMTP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	smtp	检查类型	是	对于 SMTP 健康检查类型，固定为 smtp
domain	字符串	长度 1-63	域名	否	缺省值:ADC

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add>
 请求 body

```

{
  "name": "smtp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "smtp",
  "port": 25,
  "domain": "mail"
}

```

SNMP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	snmp	检查类型	是	对于 SNMP 健康检查类型，固定为 snmp
operation	字符串	长度 1-63	操作	否	get/get_next; 缺省值:get
oid	字符串	长度 1-63	OID, 会自动加上前缀 1.3.6.1	否	缺省值:. 2.1.1.1.0
community	字符串	长度 1-63	团体字	否	缺省值:public

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add>
 请求 body

```
{
  "name": "snmp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "snmp",
  "port": 161,
  "operation": "get",
  "oid": "2.1.1.1.0",
  "community": "public"
}
```

TCP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	tcp	检查类型	是	对于 TCP 健康检查类型，固定为 tcp
send_rst	整数	0, 1	发送 reset	否	1:是;0:否;缺省值:0
send_str	字符串	长度 1-256	发送字符串	否	缺省值:空
response_str	字符串	长度 1-256	接收字符串	否	缺省值:空
response_disable_str	字符串	长度 1-256	接收禁用字符串，没有收到 response 字符串但是收到该字符串之后，对健康检查目标执行软关机操作。必须配合 response 参数使用	否	缺省值:空

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add>
 请求 body

```
{
  "name": "tcp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "tcp",
  "port": 80,
  "send_rst": 0,
  "send_str": "aaaaa",
  "response_str": "bbbbbb"
}
```

TCP-PRO 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	tcp-pro	检查类型	是	对于 TCP-PRO 健康检查类型, 固定为 tcp-pro
retry	整数	Pro 类型: 0	重试次数	是	Pro 类型必须设为 0
interval	整数	1-180	间隔时间,单位秒	是	缺省值: 5
timeout	整数	Pro 类型: 1-1800	超时时间,单位秒	是	缺省值: 5 Pro 类型的值必须大于 interval
send_rst	整数	0, 1	发送 reset	否	1:是;0:否;缺省值:0
send_str	字符串	长度 1-256	发送字符串	否	缺省值:空
response_str	字符串	长度 1-256	接收字符串	否	缺省值:空
response_disable_str	字符串	长度 1-256	接收禁用字符串, 没有收到 response 字符串但是收到该字符串之后, 对健康检查目标执行软关机操作。必须配合 response 参数使用	否	缺省值:空

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```
{  
  "name": "tcp-pro",  
  "retry": 0,  
  "interval": 3,  
  "timeout": 5,  
  "type": "tcp-pro",  
  "port": 80,  
  "send_rst": 0,  
  "send_str": "aaaaa",  
  "response_str": "bbbbbb"  
}
```

UDP 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

type	字符串	udp	检查类型	是	对于 UDP 健康检查类型，固定为 udp
send_str	字符串	长度 1-256	发送字符串	否	缺省值:空
response_str	字符串	长度 1-256	接收字符串	否	缺省值:空
response_disable_str	字符串	长度 1-256	接收禁用字符串，没有收到 response 字符串但是收到该字符串之后，对健康检查目标执行软关机操作。必须配合 response 参数使用	否	缺省值:空

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
 请求 body

```
{
  "name": "udp",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "udp",
  "port": 8000
}
```

UDP-PRO 健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	udp-pro	检查类型	是	对于 UDP-PRO 健康检查类型，固定为 udp-pro
retry	整数	Pro 类型: 0	重试次数	是	Pro 类型必须设为 0
interval	整数	1-180	间隔时间,单位秒	是	缺省值: 5
timeout	整数	Pro 类型: 1-1800	超时时间,单位秒	是	缺省值: 5 Pro 类型的值必须大于 interval
send_str	字符串	长度 1-256	发送字符串	否	缺省值:空
response_str	字符串	长度 1-256	接收字符串	否	缺省值:空
response_disable_str	字符串	长度 1-256	接收禁用字符串，没有收到 response 字符串但是收到该字符串之后，对健康检查目标执行软关机操作。必须配合 response 参数使用	否	缺省值:空

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```
{  
  "name": "udp-pro",  
  "retry": 0,  
  "interval": 3,  
  "timeout": 5,  
  "type": "udp-pro",  
  "port": 8000  
}
```

自定义脚本健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	script	检查类型	是	对于自定义脚本健康检查类型，固定为 script
script	字符串	长度 1-191	脚本名称	是	需要先上传脚本

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
请求 body

```
{  
  "name": "script",  
  "retry": 3,  
  "interval": 5,  
  "timeout": 5,  
  "type": "script",  
  "script": "checkbody.py",  
  "port": 80  
}
```

组合健康检查添加

Action: slb.healthcheck.add

请求参数:

名称	类型	范围	含义	必选	备注
type	字符串	combo	检查类型	是	对于组合健康检查类型，固定为 combo
combo	字符串	长度 1-63	组合健康检查表达式	是	hck 关键字表示健康检查名称

					and 表示与 or 表示 或 not 表示非 使用后缀表达式
--	--	--	--	--	--

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.add
 请求 body

```
{
  "name": "combo",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "combo",
  "combo": "hck ping not "
}
```

健康检查列表

Action: slb.healthcheck.list

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.list

响应参数:

响应参数参考健康检查添加中各种类型健康检查的相应参数

响应举例:

```
[
  {
    "name": "ping",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "icmp"
  },
  {
    "name": "http",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "auto_disable": 0,

    "alias_ipv4_src": "192.168.1.1",
```

```

"interface": "Ethernet 0/1",

"alias_ipv4": "1.2.3.4",
"alias_port": 889,
"alias_ipv6": "2001::1",
"type": "http"
"port": 80,
"host": "ahost",
"url": "GET /index",
"post_data": "mybodystring",
"username": "username",
"password": "pass",
"code": "202",
"pattern": "rvstring"
},
{
"name": "arp",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "arp"
},
{
"name": "database",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "database",
"database_name": "db",
"username": "aaa",
"password": "sss",
"query": "",
"response_str": "",
"row": 1,
"column": 1,
"database_type": 4
},
{
"name": "dns",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "dns",
"port": 53,
"domain": "www.baidu.com",
"record_type": "A",

```

```

    "wantreturn": "1,2,3",
    "want_ipv4": "4.3.2.1"
},
{
    "name": "ftp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "ftp",
    "port": 21,
    "username": "",
    "password": ""
},
{
    "name": "https",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "https",
    "port": 80,
    "host": "ahost",
    "url": "GET /index",
    "post_data": "mybodystring",
    "username": "username",
    "password": "pass",
    "code": "202",
    "pattern": "rvstring"
},
{
    "name": "imap",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "imap",
    "port": 143,
    "plain_text": 0,
    "login": 0,
    "cram_md5": 0,
    "username": "",
    "password": ""
},
{
    "name": "ldap",
    "retry": 3,
    "interval": 5,
    "timeout": 5,

```

```

"type": "ldap",
"port": 389,
"ssl": 0,
"distinguished_name": "",
"password": ""
},
{
"name": "ntp",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "ntp",
"port": 123
},
{
"name": "pop3",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "pop3",
"port": 110,
"username": "aas",
"password": "dww"
},
{
"name": "radius",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "radius",
"port": 1812,
"username": "",
"password": "",
"secret": "secret"
},
{
"name": "rtsp",
"retry": 3,
"interval": 5,
"timeout": 5,
"type": "rtsp",
"port": 554,
"url": "/music.mp3"
},
{
"name": "sip",

```

```

    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "sip",
    "port": 5060,
    "register": 0
  },
  {
    "name": "smtp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "smtp",
    "port": 25,
    "domain": "mail"
  },
  {
    "name": "snmp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "snmp",
    "port": 161,
    "operation": "get",
    "oid": "2.1.1.1.0",
    "community": "public"
  },
  {
    "name": "tcp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "tcp",
    "port": 80,
    "send_rst": 0,
    "send_str": "aaaaa",
    "response_str": "bbbbbb"
  },
  {
    "name": "udp",
    "retry": 3,
    "interval": 5,
    "timeout": 5,
    "type": "udp",
    "port": 8000
  },

```

```

{
  "name": "script",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "script",
  "script": "checkbody.py",
  "port": 80
},
{
  "name": "combo",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "type": "combo",
  "combo": "hck ping not "
}
]

```

健康检查编辑

Action: slb.healthcheck.edit

请求参数:

请求参数请参考 [健康检查添加](#) 中各种类型健康检查的响应参数

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.edi
t
{
  "name": "ping",
  "retry": 1,
  "interval": 4,
  "timeout": 2,
  "type": "icmp"
}

```

健康检查获取

Action: slb.healthcheck.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	健康检查名称	是	必须存在

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.get
请求 body

```
{
  "name": "ping"
}
```

响应参数:

响应参数参考健康检查添加中各种类型健康检查的相应参数

响应举例:

```
{
  name: "ping",
  "retry": 3,
  "interval": 5,
  "timeout": 5,
  "auto_disable": 0,
  "type": "icmp"
}
```

健康检查删除

Action: slb.healthcheck.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	健康检查名称	是	必须存在

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.del
请求 body

```
{
  "name": "ping"
}
```

健康检查脚本列表

Action: slb.healthcheck.script.list

请求参数:

无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.script.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-50	脚本名称

响应举例:

```
[
  {
    "name": "checkbody.py"
  },
  {
    "name": "checkbody1.py"
  }
]
```

健康检查脚本上传

Action: slb.healthcheck.script.upload

请求参数:

无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.script.upload



Note

此 API 需要使用 form-data 的方式上传健康检查脚本文件

健康检查脚本删除

Action: slb.healthcheck.script.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-50	脚本名称	是	唯一

请求举例:

POST

```
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.scrip
t.del
请求 body
{
  "name": "health_check_script1.py"
}
```

健康检查 POST 文件列表

Action: slb.healthcheck.postfile.list

请求参数:

无

请求举例:

GET

```
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.postfile.list
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-50	POST 文件名称

响应举例:

```
[
  {
    "name": "checkbody.html"
  },
  {
    "name": "checkbody1.html"
  }
]
```

健康检查 POST 文件上传

Action: slb.healthcheck.postfile.upload

请求参数:

无

请求举例:

POST

```
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.postfile.upload
```



此 API 需要使用 form-data 的方式上传健康检查 POST 文件

健康检查 POST 文件删除

Action: slb.healthcheck.postfile.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-50	脚本名称	是	唯一

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.healthcheck.postfile.del
请求 body
{
  "name": "health_check_post.html"
}
```

被动健康检查

添加被动健康检查配置

Action: slb.passive-health-check.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	被动健康检查名称	是	唯一
inband_check	整数	0, 1	带内检查	否	1: 开启, 0: 关闭, 缺省值 0
inband_retry	整数	1-3	新建重试次数	否	当 inband_check 为 1 时有效, 缺省值 2
inband_reassign	整数	1-127	新建失败次数	否	当 inband_check 为 1 时有效, 缺省值 25
rst_check	整数	0, 1	RST 检查	否	1: 开启, 0: 关闭, 缺省值 0
rst_num	整数	1-127	RST 统计	否	当 rst_check 为 1 时有效, 缺省值 1
zerowin_check	整数	0, 1	零窗口检查	否	1: 开启, 0: 关闭, 缺省值 0
zerowin_num	整数	1-65535	零窗口统计	否	当 zerowin_check 为 1 时有效, 缺省值 65535

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.passive-health-check.add
请求 body:
{
  "name": "pcheck1",
```

```

    "inband_check":1,
    "inband_retry":2,
    "inband_reassign":2,
    "rst_check":1,
    "rst_num":11,
    "zerowin_check":1,
    "zerowin_num":10
}

```

获取被动健康检查配置列表

Action: slb.passive-health-check.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.passive-health-check.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	被动健康检查名称
inband_check	整数	0,1	带内检查, 1: 开启, 0: 关闭
inband_retry	整数	1-3	新建重试次数
inband_reassign	整数	1-127	新建失败次数
rst_check	整数	0,1	RST 检查, 1: 开启, 0: 关闭
rst_num	整数	1-127	RST 统计
zerowin_check	整数	0,1	零窗口检查, 1: 开启, 0: 关闭
zerowin_num	整数	1-65535	零窗口统计

响应举例:

```

[ {
    "name": "pcheck1",
    "inband_check": 1,
    "inband_retry": 2,
    "inband_reassign": 2,
    "rst_check": 1,
    "rst_num": 11,
    "zerowin_check": 1,
    "zerowin_num": 10
} ]

```

获取指定被动健康检查配置

Action: slb.passive-health-check.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	被动健康检查名称	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.passive-health-check.get

请求 body:

```
{"name": "pcheck1"}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	被动健康检查名称
inband_check	整数	0, 1	带内检查, 1: 开启, 0: 关闭
inband_retry	整数	1-3	新建重试次数
inband_reassign	整数	1-127	新建失败次数
rst_check	整数	0, 1	RST 检查, 1: 开启, 0: 关闭
rst_num	整数	1-127	RST 统计
zerowin_check	整数	0, 1	零窗口检查, 1: 开启, 0: 关闭
zerowin_num	整数	1-65535	零窗口统计

响应举例:

```
{
  "name": "pcheck1",
  "inband_check": 1,
  "inband_retry": 2,
  "inband_reassign": 2,
  "rst_check": 1,
  "rst_num": 11,
  "zerowin_check": 1,
  "zerowin_num": 10
}
```

编辑指定被动健康检查配置

Action: **slb.passive-health-check.edit**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	被动健康检查名称	是	必须存在
inband_check	整数	0, 1	带内检查	否	1: 开启, 0: 关闭, 缺省值 0
inband_retry	整数	1-3	新建重试次数	否	当 inband_check 为 1 时有效, 缺省值 2
inband_reassign	整数	1-127	新建失败次数	否	当 inband_check 为 1 时有效, 缺省值 25
rst_check	整数	0, 1	RST 检查	否	1: 开启, 0: 关闭, 缺省值 0
rst_num	整数	1-127	RST 统计	否	当 rst_check 为 1 时有效, 缺省值 1
zerowin_check	整数	0, 1	零窗口检查	否	1: 开启, 0: 关闭, 缺省值 0
zerowin_num	整数	1-65535	零窗口统计	否	当 zerowin_check 为 1 时有效, 缺省值 65535

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.passive-health-check.edit

请求 body:

```
{
  "name": "pcheck1",
  "inband_check": 1,
  "inband_retry": 2,
  "inband_reassign": 2,
  "rst_check": 1,
  "rst_num": 11,
  "zerowin_check": 1,
  "zerowin_num": 10
}
```

删除指定被动健康检查配置

Action: slb.passive-health-check.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	被动健康检查名称	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.passive-health-check.del

请求 body:

```
{"name": "pcheck1"}
```

ERULE

erule 上传

Action: slb.erule.upload

请求参数:无

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erule.upload



Note

此 API 使用 form-data 方式上传一个 erule 脚本文件

erule 删除

Action: slb.erule.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	erule 文件名称	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erule.del

```
{  
  "name": "erule-empty.txt"  
}
```

erule 列表

Action: slb.erule.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erule.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	erule 文件名称
syntax_ok	整数	0-1	erule 文件语法检查结果:1:通过;0:失败

响应举例:

```
[  
  {  
    "name": "erule-empty.txt",  
    "syntax_ok": 1  
  }  
]
```


erule 服务器文件上传

Action: slb.erulefiles.upload

请求参数:无

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erulefiles.upload



此 API 使用 form-data 方式上传一个 erule 服务器文件

erule 服务器文件删除

Action: slb.erulefiles.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	erule 服务器文件名称	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erulefiles.del

请求 body

```
{
  "name": "erulefiletest.txt"
}
```

erule 服务器文件列表

Action: slb.erulefiles.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.erulefiles.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	erule 服务器文件列表名称

响应举例:

```
[
  {
```

```

    "name": "bwlist1.txt"
  },
  {
    "name": "llbdomainroute.txt"
  }
]

```

连接保持

Cookie 连接保持

cookie 保持列表

Action: slb.persist.cookie.list

请求参数: 无

请求举例:

```

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.cookie.list

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	cookie 连接保持名称
expire_enable	整数	0, 1	expire 是否使能;1: 是; 0: 否
expire	整数	0-30000000	超时时间, 仅当 expire_enable 为 1 时生效
method	整数	0, 1, 2	cookie 保持方式,0 插入,1 重写, 2 被动; 缺省值: 0
encrypt	整数	0, 1, 2	cookie 加密,0 不加密,1 加密服务池名称,2 全加密
encrypt_password	字符串	长度 1-63	加密的密码
http_only	整数	0, 1	使用 http_only 字段, 1:是,0:否
secure	整数	0, 1	使用 secure 字段, 1:是,0:否
cookie_name	字符串	长度 1-63	cookie 名称
domain	字符串	长度 1-31	域名
path	字符串	长度 1-31	路径
type	整数	0, 1, 2	匹配方法: 0 端口, 1 节点, 2 服务池节点, 3 服务池, 缺省值 0
insert	整数	0, 1	cookie 插入; 1: 是; 0: 否
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否

响应举例:

```

[[
  "name": "p1",
  "expire_enable": 1,
  "expire": 123,
  "method": 1,

```

```

"encrypt": 1,
"http_only": 1,
"secure": 1,
"cookie_name": "name",
"domain": "domain",
"path": "path",
"type": 2,
"insert": 1,
"ignore_connlimit": 1
}]

```

cookie 保持获取

Action: slb.persist.cookie.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	cookie 连接保持名称	是	

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.cookie.get

```

请求 body:

```

{
  "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	cookie 连接保持名称
expire_enable	整数	0, 1	expire 是否使能; 1: 是; 0: 否
method	整数	0, 1, 2	cookie 保持方式, 0 插入, 1 重写, 2 被动; 缺省值: 0
encrypt	整数	0, 1, 2	cookie 加密, 0 不加密, 1 加密服务池名称, 2 全加密
encrypt_password	字符串	长度 1-63	加密的密码
http_only	整数	0, 1	使用 http_only 字段, 1:是,0:否
secure	整数	0, 1	使用 secure 字段, 1:是,0:否
expire	整数	0-30000000	cookie 超时时间
cookie_name	字符串	长度 1-63	cookie 名称
domain	字符串	长度 1-31	域名
path	字符串	长度 1-31	路径
type	整数	0, 1, 2	匹配方法: 0 端口, 1 节点, 2 服务池节点, 3 服务池, 缺省值 0
insert	整数	0, 1	cookie 插入; 1: 是; 0: 否
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否

响应举例:

```

{

```

```

"name": "p1",
"expire_enable": 1,
"expire": 123,
"method": 1,
"encrypt": 1,
"http_only": 1,
"secure": 1,
"cookie_name": "name",
"domain": "domain",
"path": "path",
"type": 2,
"insert": 1,
"ignore_connlimit": 1
}

```

cookie 保持增加

Action: slb.persist.cookie.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	cookie 连接保持名称	是	唯一
expire_enable	整数	0, 1	expire 是否使能;	否	1: 是; 0: 否;缺省值 0
expire	整数	0-30000000	超时时间	否	expire_enable 为 1 时有效,缺省值 0
method	整数	0, 1, 2	cookie 保持方式	否	cookie 保持方式, 0 插入, 1 重写, 2 被动; 缺省值: 0
encrypt	整数	0, 1, 2	cookie 加密	否	,0 不加密,1 加密服务池名称,2 全加密, 缺省值 0
encrypt_password	字符串	长度 1-63	加密的密码	否	缺省的话为空, 不显示
http_only	整数	0, 1	使用 http_only 字段,	否	1:是,0:否,缺省值 0
secure	整数	0, 1	使用 secure 字段	否	1:是,0:否,缺省值 0
cookie_name	字符串	长度 1-63	cookie 名称	否	缺省值:空字符串
domain	字符串	长度 1-31	域名	否	缺省值:空字符串
path	字符串	长度 1-31	路径	否	缺省值:空字符串
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池节点, 3 服务池, 缺省值 0
insert	整数	0, 1	开启 cookie 插入	否	1 使能, 0 禁用, 缺省值 0
ignore_connlimit	整数	0, 1	覆盖连接限制	否	1 使能, 0 禁用, 缺省值 0

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.cookie.add

```

请求 body:

```

{
  "name": "p1",

```

```

"expire_enable": 1,
"expire": 123,
"method": 1,
"encrypt": 1,
"http_only": 1,
"secure": 1,
"cookie_name": "name",
"domain": "domain",
"path": "path",
"type": 2,
"insert": 1,
"ignore_connlimit": 1
}

```

cookie 保持编辑

Action: slb.persist.cookie.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	cookie 连接保持名称	是	必须存在
expire_enable	整数	0, 1	expire 是否使能;	否	1: 是; 0: 否;缺省值:不修改
expire	整数	0-30000000	超时时间	否	expire_enable 为 1 时有效, 缺省值:不修改
method	整数	0, 1, 2	cookie 保持方式	否	cookie 保持方式, 0 插入, 1 重写, 2 被动
encrypt	整数	0, 1, 2	cookie 加密	否	0 不加密,1 加密服务池名称,2 全加密, 缺省值:不修改
encrypt_password	字符串	长度 1-63	加密的密码	否	缺省的话为空, 不显示
http_only	整数	0, 1	使用 http_only 字段,	否	1:是,0:否, 缺省值:不修改
secure	整数	0, 1	使用 secure 字段	否	1:是,0:否, 缺省值:不修改
cookie_name	字符串	长度 1-63	cookie 名称	否	缺省值:不修改
domain	字符串	长度 1-31	域名	否	缺省值:不修改
path	字符串	长度 1-31	路径	否	缺省值:不修改
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池节点, 3 服务池, 缺省值 0
insert	整数	0, 1	开启 cookie 插入	否	1 使能, 0 禁用, 缺省值:不修改
ignore_connlimit	整数	0, 1	覆盖连接限制	否	1 使能, 0 禁用, 缺省值:不修改

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.cookie.edit

```

请求 body:

```
{
```

```

"name": "p1",
"expire_enable": 1,
"expire": 123,
"method": 1,
"encrypt": 1,
"http_only": 1,
"secure": 1,
"cookie_name": "name",
"domain": "domain",
"path": "path",
"type": 2,
"insert": 1,
"ignore_connlimit": 1
}

```

cookie 保持删除

Action: slb.persist.cookie.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	cookie 连接保持名称	是	必须存在

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.cookie.del

请求 body:

```

{
  "name": "p1"
}

```

源地址连接保持

获取源 IP 地址保持模板列表

Action: slb.persist.srcip.list

请求参数: 无

请求举例:

GET
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.srcip.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	源 IP 地址保持模版名称
type	整数	0, 1, 2	匹配方法, 0 端口, 1 节点, 2 服务池
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否
with_port	整数	0, 1	包含端口; 1: 是; 0: 否
netmask	掩码		子网掩码
ipv6masklen	整数	0-128	ipv6 前缀长度

响应举例:

```

[[
  "name": "p1",
  "type": 1,
  "timeout": 1012,
  "ignore_connlimit": 1,
  "with_port": 0,
  "netmask": "255.255.255.0",
  "ipv6masklen": 23
]]

```

获取指定源 IP 地址保持模板

Action: slb.persist.srcip.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	源 IP 地址保持模版名称	是	必须存在

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.srcip.get>

请求 body:

```

{
  "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	源 IP 地址保持模版名称
type	整数	0, 1, 2	匹配方法, 0 端口, 1 节点, 2 服务池
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否
with_port	整数	0, 1	包含端口; 1: 是; 0: 否
netmask	掩码		子网掩码
ipv6masklen	整数	0-128	ipv6 前缀长度

响应举例:

```
{
  "name": "p1",
  "type": 1,
  "timeout": 1012,
  "ignore_connlimit": 1,
  "with_port": 0,
  "netmask": "255.255.255.0",
  "ipv6masklen": 23
}
```

增加源 IP 地址保持模板

Action: slb.persist.srcip.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	源 IP 地址保持模版名称	是	唯一
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池; 缺省值 0
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0
with_port	整数	0, 1	包含端口;	否	1: 是; 0: 否; 缺省值 0
netmask	掩码		子网掩码	否	缺省值 255.255.255.255
ipv6masklen	整数	0-128	ipv6 前缀长度	否	缺省值 128

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.srcip.add>

请求 body:

```
{
  "name": "p1",
  "type": 1,
  "timeout": 1012,
  "ignore_connlimit": 1,
  "with_port": 0,
  "netmask": "255.255.255.0",
  "ipv6masklen": 23
}
```

编辑源 IP 地址保持模板

Action: slb.persist.srcip.edit

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	源 IP 地址保持模版名称	是	必须存在
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池; 缺省值 0
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0
with_port	整数	0, 1	包含端口;	否	1: 是; 0: 否; 缺省值 0
netmask	掩码		子网掩码	否	缺省值 255.255.255.255
ipv6masklen	整数	0-128	ipv6 前缀长度	否	缺省值 128

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.srcip.edit

请求 body:

```
{
  "name": "p1",
  "type": 1,
  "timeout": 1012,
  "ignore_connlimit": 1,
  "with_port": 0,
  "netmask": "255.255.255.0",
  "ipv6masklen": 23
}
```

删除源 IP 地址保持模板

Action: slb.persist.srcip.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	源 IP 地址保持模版名称	是	必须存在

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.srcip.del

请求 body:

```
{
  "name": "p1"
}
```

目的地址连接保持

获取目的 IP 地址保持模板列表

Action: slb.persist.dstip.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.dstip.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	目的 IP 地址保持模版名称
type	整数	0, 1, 2	匹配方法, 0 端口, 1 节点, 2 服务池
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否
netmask	掩码		子网掩码
ipv6masklen	整数	0-128	ipv6 前缀长度

响应举例:

```
[[
  "name": "p1",
  "type": 2,
  "timeout": 34,
  "ignore_connlimit": 1,
  "netmask": "255.255.255.0",
  "ipv6masklen": 12
]]
```

获取指定目的 IP 地址保持模板

Action: slb.persist.dstip.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	目的 IP 地址保持模版名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.dstip.get
请求 body:
{
 "name": "p1"

```
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	目的 IP 地址保持模版名称
type	整数	0, 1, 2	匹配方法, 0 端口, 1 节点, 2 服务池
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否
netmask	掩码		子网掩码
ipv6masklen	整数	0-128	ipv6 前缀长度

响应举例:

```
{
  "name": "p1",
  "type": 2,
  "timeout": 34,
  "ignore_connlimit": 1,
  "netmask": "255.255.255.0",
  "ipv6masklen": 12
}
```

增加目的 IP 地址保持模板

Action: slb.persist.dstip.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	目的 IP 地址保持模版名称	是	唯一
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池; 缺省值 0
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0
netmask	掩码		子网掩码	否	缺省值 255.255.255.255
ipv6masklen	整数	0-128	ipv6 前缀长度	否	缺省值 128

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.dstip.add>

请求 body:

```
{
  "name": "p1",
  "type": 2,
  "timeout": 34,
  "ignore_connlimit": 1,
  "netmask": "255.255.255.0",
```

```
"ipv6masklen": 12
}
```

编辑目的 IP 地址保持模板

Action: slb.persist.dstip.edit

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	目的 IP 地址保持模版名称	是	必须存在
type	整数	0, 1, 2	匹配方法	否	0 端口, 1 节点, 2 服务池; 缺省值 0
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0
netmask	掩码		子网掩码	否	缺省值 255.255.255.255
ipv6masklen	整数	0-128	ipv6 前缀长度	否	缺省值 128

请求参数:

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.dstip.edit
```

请求 body:

```
{
  "name": "p1",
  "type": 2,
  "timeout": 34,
  "ignore_connlimit": 1,
  "netmask": "255.255.255.0",
  "ipv6masklen": 12
}
```

删除目的 IP 地址保持模板

Action: slb.persist.dstip.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	目的 IP 地址保持模版名称	是	必须存在

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.dstip.del
```

请求 body:

```
{
  "name": "p1"
}
```

}

SSL 连接保持

获取 SSL 连接保持模板列表

Action: slb.persist.sslid.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.sslid.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	SSL 连接保持模版名称
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否

响应举例:

```
[[  
  "name": "p1",  
  "timeout": 1001,  
  "ignore_connlimit": 0  
]]
```

获取指定 SSL 连接保持模板

Action: slb.persist.sslid.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	SSL 连接保持模版名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.sslid.get
请求 body:
{
 "name": "p1"
}

响应参数:

名称	类型	范围	含义
----	----	----	----

name	字符串	长度 1-191	SSL 连接保持模版名称
timeout	整数	1-1800	超时,
ignore_connlimit	整数	0, 1	覆盖连接限制; 1: 是; 0: 否

响应举例:

```
{
  "name": "p1",
  "timeout": 1001,
  "ignore_connlimit": 0
}
```

增加 SSL 连接保持模板

Action: slb.persist.sslid.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	目的 IP 地址保持模版名称	是	唯一
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.sslid.add>

请求 body:

```
{
  "name": "p1",
  "timeout": 1001,
  "ignore_connlimit": 0
}
```

编辑 SSL 连接保持模板

Action: slb.persist.sslid.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	SSL 连接保持模版名称	是	必须存在
timeout	整数	1-1800	超时时间, 单位分钟	否	缺省值 10
ignore_connlimit	整数	0, 1	覆盖连接限制;	否	1: 是; 0: 否; 缺省值 0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.sslid.e
dit

请求 body:

```
{  
  "name": "p1",  
  "timeout": 1001,  
  "ignore_connlimit": 0  
}
```

删除 SSL 连接保持模板

Action: slb.persist.sslid.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	SSL 连接保持模版名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.persist.sslid.d
el

请求 body:

```
{  
  "name": "p1"  
}
```

加速

缓存

获取缓存模板列表

Action: slb.cache.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.cache.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	缓存模版名称
age	整数	1-999999	最长保存时间
size_max	整数	1-16384	最大缓存数量

object_min	整数	1-268435455	最小对象大小,单位字节
object_max	整数	1-268435455	最大对象大小,单位字节
handle_reload_req	整数	0, 1	缓存控制头; 0 禁用 1 使能
cache_host	整数	0, 1	缓存主机名
no_cache	整数	0, 1	默认不缓存
no_age_header	整数	0, 1	不插入 age 头
no_via_header	整数	0, 1	不插入 via 头
cache_policy	数组		缓存策略对象组成的数组,根据 URL 设定缓存
url	字符串	长度 1-63	缓存策略-URL, 匹配 URL 后使用本策略
match	字符串	长度 1-63	缓存策略-匹配字符串,匹配字符串刷新存活时间
age	整数	1-900000	缓存策略-缓存保存时间
action	整数	0, 1, 2	缓存策略-算法: 0 缓存; 1 禁用缓存; 2 刷新

响应举例:

```

[[
  "name": "p1",
  "age": 3001,
  "size_max": 101,
  "object_min": 501,
  "object_max": 50001,
  "handle_reload_req": 1,
  "cache_host": 1,
  "no_cache": 1,
  "no_age_header": 1,
  "no_via_header": 1,
  "cache_policy": [
    {
      "url": "/url",
      "match": "",
      "age": 0,
      "action": 1
    },
    {
      "url": "/url2",
      "match": "asdsa",
      "age": 0,
      "action": 2
    }
  ]
]]

```

获取指定缓存模板

Action: slb.cache.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	缓存模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=s1b.cache.get

请求 body:

```
{
  "name": "p0"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	缓存模版名称
age	整数	1-999999	最长保存时间
size_max	整数	1-16384	最大缓存数量
object_min	整数	1-268435455	最小对象大小,单位字节
object_max	整数	1-268435455	最大对象大小,单位字节
handle_reload_req	整数	0, 1	缓存控制头; 0 禁用 1 使能
cache_host	整数	0, 1	缓存主机名
no_cache	整数	0, 1	默认不缓存
no_age_header	整数	0, 1	不插入 age 头
no_via_header	整数	0, 1	不插入 via 头
cache_policy	数组		缓存策略对象组成的数组,根据 URL 设定缓存
url	字符串	长度 1-63	缓存策略-URL, 匹配 URL 后使用本策略
match	字符串	长度 1-63	缓存策略-匹配字符串,匹配字符串刷新存活时间
age	整数	1-900000	缓存策略-缓存保存时间
action	整数	0, 1, 2	缓存策略-算法: 0 缓存; 1 禁用缓存; 2 刷新

响应举例:

```
{
  "name": "p1",
  "age": 3001,
  "size_max": 101,
  "object_min": 501,
  "object_max": 50001,
  "handle_reload_req": 1,
  "cache_host": 1,
  "no_cache": 1,
  "no_age_header": 1,
  "no_via_header": 1,
  "cache_policy": [
    {
      "url": "/url",
      "match": "",
      "age": 0,
      "action": 1
    }
  ]
}
```

```

    },
    {
      "url": "/url2",
      "match": "asdsa",
      "age": 0,
      "action": 2
    }
  ]
}

```

增加缓存模板

Action: slb.cache.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	缓存模版名称	是	唯一
age	整数	1-999999	最长保存时间	否	缺省值:3600
size_max	整数	1-16384	最大缓存数量	否	缺省值:80
object_min	整数	1-268435455	最小对象大小,单位字节	否	缺省值:512
object_max	整数	1-268435455	最大对象大小,单位字节	否	缺省值:81920
handle_reload_req	整数	0, 1	缓存控制头	否	0 禁用 1 使能, 缺省值:0
cache_host	整数	0, 1	缓存主机名	否	0 禁用 1 使能, 缺省值:0
no_cache	整数	0, 1	默认不缓存	否	0 禁用 1 使能, 缺省值:0
no_age_header	整数	0, 1	不插入 age 头	否	0 禁用 1 使能, 缺省值:0
no_via_header	整数	0, 1	不插入 via 头	否	0 禁用 1 使能, 缺省值:0
cache_policy	数组		缓存策略对象组成的数组,根据 URL 设定缓存	否	缺省值:[]无缓存策略
url	字符串	长度 1-63	缓存策略-URL, 匹配 URL 后使用本策略	是	
match	字符串	长度 1-63	缓存策略-匹配字符串, 匹配字符串刷新存活时间	是	
age	整数	1-900000	缓存策略-缓存保存时间	是	
action	整数	0, 1, 2	缓存策略-算法;	否	0 缓存, 1 禁用缓存, 2 刷新; 缺省值:0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.cache.add

请求 body:

```

{
  "name": "p1",
  "age": 3001,
  "size_max": 101,

```

```

"object_min": 501,
"object_max": 50001,
"handle_reload_req": 1,
"cache_host": 1,
"no_cache": 1,
"no_age_header": 1,
"no_via_header": 1,
"cache_policy": [
  {
    "url": "/url",
    "match": "",
    "age": 0,
    "action": 1
  },
  {
    "url": "/url2",
    "match": "asdsa",
    "age": 0,
    "action": 2
  }
]
}

```

编辑缓存模板

Action: slb.cache.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	缓存模版名称	是	必须存在
age	整数	1-999999	最长保存时间	否	缺省值:3600
size_max	整数	1-16384	最大缓存数量	否	缺省值:80
object_min	整数	1-268435455	最小对象大小,单位字节	否	缺省值:512
object_max	整数	1-268435455	最大对象大小,单位字节	否	缺省值:81920
handle_reload_req	整数	0, 1	缓存控制头	否	0 禁用 1 使能, 缺省值:0
cache_host	整数	0, 1	缓存主机名	否	0 禁用 1 使能, 缺省值:0
no_cache	整数	0, 1	默认不缓存	否	0 禁用 1 使能, 缺省值:0
no_age_header	整数	0, 1	不插入 age 头	否	0 禁用 1 使能, 缺省值:0
no_via_header	整数	0, 1	不插入 via 头	否	0 禁用 1 使能, 缺省值:0
cache_policy	数组		缓存策略对象组成的数组,根据 URL 设定缓存	否	缺省值:[]无缓存策略
url	字符串	长度 1-63	缓存策略-URL, 匹配 URL 后使用本策略	是	
match	字符串	长度 1-63	缓存策略-匹配字符串,	是	

			匹配字符串刷新存活时间		
age	整数	1-900000	缓存策略-缓存保存时间	是	
action	整数	0, 1, 2	缓存策略-算法:	否	0 缓存, 1 禁用缓存, 2 刷新; 缺省值:0

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.cache.edit

请求 body:

```
{
  "name": "p1",
  "age": 3001,
  "size_max": 101,
  "object_min": 501,
  "object_max": 50001,
  "handle_reload_req": 1,
  "cache_host": 1,
  "no_cache": 1,
  "no_age_header": 1,
  "no_via_header": 1,
  "cache_policy": [
    {
      "url": "/url",
      "match": "",
      "age": 0,
      "action": 1
    },
    {
      "url": "/url2",
      "match": "asdsa",
      "age": 0,
      "action": 2
    }
  ]
}
```

删除缓存模板

Action: slb.cache.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	缓存模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.cache.del

请求 body:

```
{
  "name": "p0"
}
```

连接复用

获取连接复用模板列表

Action: **slb.connmulti.list**

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.connmulti.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	连接复用模版名称
limit	整数	0-65535	最大重用数量
timeout	整数	1-3600	最大重用数量

响应举例:

```
[[
  "name": "p1",
  "limit": 1025,
  "timeout": 1801
]]
```

获取指定连接复用模板

Action: **slb.connmulti.get**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	连接复用模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.connmulti.get

请求 body:

```
{
  "name": "p0"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	连接复用模版名称
limit	整数	0-65535	最大重用数量
timeout	整数	1-3600	最大重用数量

响应举例:

```
{
  "name": "p1",
  "limit": 1025,
  "timeout": 1801
}
```

增加连接复用模板

Action: slb.connmulti.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	连接复用模版名称	是	唯一
limit	整数	0-65535	最大重用数量	否	缺省值:1024
timeout	整数	1-3600	最大重用数量	否	缺省值:1800

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.connmulti.add>
 请求 body:

```
{
  "name": "p1",
  "limit": 1025,
  "timeout": 1801
}
```

编辑连接复用模板

Action: slb.connmulti.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	连接复用模版名称	是	必须存在
limit	整数	0-65535	最大重用数量	否	缺省值:1024
timeout	整数	1-3600	最大重用数量	否	缺省值:1800

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.connmulti.edit>

请求 body:

```
{
  "name": "p1",
  "limit": 1025,
  "timeout": 1801
}
```

删除连接复用模板

Action: slb.connmulti.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	连接复用模版名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.connmulti.del

请求 body:

```
{
  "name": "p0"
}
```

SSL 卸载

证书管理

证书上传

Action: slb.ssl.certificate.upload

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.certificate.upload



Note

此 API 需要使用 form-data 的方式上传证书文件

私钥上传

Action: slb.ssl.key.upload

请求参数:无

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.key.upload



此 API 需要使用 form-data 的方式上传私钥文件

证书吊销列表上传

Action: slb.ssl.crl.upload

请求参数:无

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.crl.upload



此 API 需要使用 form-data 的方式上传证书吊销列表文件

PFX 证书上传

Action: slb.ssl.pfx.upload

请求参数:

名称	类型	范围	含义	必选	备注
password	字符串	长度 1-63	证书密码	否	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.pfx.upload&password=1234.com



- 1.此 API 需要使用 form-data 的方式上传 pfx 证书文件
- 2.此 API 参数在 URL 中指定而不是在 BODY 中
- 3.上传 pfx 证书会产生一个证书和一个私钥, 如要删除需分别删除。

证书删除

Action: slb.ssl.certificate.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-255	证书名称	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.certificate.del

请求 body

```
{  
  "name": "csr2"  
}
```

私钥删除

Action: slb.ssl.key.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	私钥名称	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.key.del

```
{  
  "name": "csr2"  
}
```

证书吊销列表删除

Action: slb.ssl.crl.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-255	证书吊销列表名称	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.crl.del

```
{  
  "name": "csr2"  
}
```

```
}
```

PFX 证书删除

Action: slb.ssl.pfx.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	PFX 证书名称	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.pfx.del

请求 body

```
{  
  "name": "csr2"  
}
```

获取 SSL 证书列表

Action: slb.ssl.certificate.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.certificate.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-255	名称
type	字符串	长度 1-63	类型 类型字段说明: certificate 证书 key 私钥 certificate-key 证书和私钥 crl 吊销列表

响应举例:

```
[  
  {  
    "name": "mysert1",  
    "type": "certificate-key"  
  }  
]
```

生成 ssl 自签名证书

Action: slb.ssl.certificate.add

请求参数:

名称	类型	范围	含义	必选	备注
issuer	字符串	self	固定写 self 表示是自签名	是	
name	字符串	长度 1-255	证书名称	是	唯一
common_name	字符串	长度 1-63	通用名称	是	
division	字符串	长度 1-63	部门	否	缺省值:"" 空字符串
organization	字符串	长度 1-63	组织	否	缺省值:"" 空字符串
locality	字符串	长度 1-63	位置 (城市)	否	缺省值:"" 空字符串
state	字符串	长度 1-63	州/省	否	缺省值:"" 空字符串
country	字符串	长度 1-63	国家	否	缺省值:CN
email	字符串	长度 1-63	email	否	缺省值:"" 空字符串
password	字符串	长度 1-63	密码	否	缺省值:"" 空字符串
key_size	整数	512/1024/2048	密钥长度	否	缺省值:1024
days	整数	1-3650	证书有效天数	否	缺省值:365

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.ssl.certificate.add

请求 body

```
{
  "issuer":"self",
  "name":"mysert1",
  "common_name":"adc.com",
  "division":"division",
  "organization":"organization",
  "locality":"locality",
  "state":"Beijing",
  "country":"CN",
  "email":"a@b.c",
  "password":"123456",
  "key_size":2048,
  "days":345
}
```

生成 CSR

Action: slb.ssl.certificate.add

请求参数:

名称	类型	范围	含义	必选	备注
issuer	字符串	ca	固定写 ca 表示 CA 签名	是	
name	字符串	长度 1-255	证书名称	是	唯一

common_name	字符串	长度 1-63	通用名称	是	
division	字符串	长度 1-63	部门	否	缺省值:"" 空字符串
organization	字符串	长度 1-63	组织	否	缺省值:"" 空字符串
locality	字符串	长度 1-63	位置 (城市)	否	缺省值:"" 空字符串
state	字符串	长度 1-63	州/省	否	缺省值:"" 空字符串
country	字符串	长度 1-63	国家	否	缺省值:CN
email	字符串	长度 1-63	email	否	缺省值:"" 空字符串
password	字符串	长度 1-63	密码	否	缺省值:"" 空字符串
key_size	整数	512/1024/2048	密钥长度	否	缺省值:1024
days	整数	1-3650	证书有效天数	否	缺省值:365

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=s1b.ssl.certificate.add

请求 body

```
{
  "issuer": "ca",
  "name": "csr2",
  "common_name": "adc.com",
  "division": "division",
  "organization": "organization",
  "locality": "locality",
  "state": "Beijing",
  "country": "CN",
  "email": "a@b.c",
  "password": "123456",
  "key_size": 2048,
  "days": 345
}
```

响应参数:

名称	类型	范围	含义
csr	字符串		响应为生成的 csr 请求内容,使用该 CSR 再去向 CA 申请证书

响应举例:

```
{
  "csr": "-----BEGIN CERTIFICATE REQUEST-----\nMIIC4TCCAckCAQAwYQxCZAJBgNVBAYTAkNOMRAwDgYDVQQIEwdCZWlqaW5nMREw\nDwYDVQQHEwhsb2Nhbg10eTEVMBMGA1UEChMMb3JnYW5pemF0aW9uMREwDwYDVQQQLnEwhkaXZpc2lvbjEQMA4GA1UEAxMHYWRjLmNvbTEUUMBIGCSqGSIb3DQEJARYFYUBi\nLmMwggEiMA0GCSqGSIb3DQEBBQUAA4IBDwAwggEKAoIBAQCsyTwXID5RRVEpvIks\nana8+3S/nR2XPnsORfDD6HZAWY9GOOy+53/KJD4n6j3oKdD40EbvGhZfZ9iwupmXSc\nnJhao04n05TFE2DxmQQSCNO2OTO5Suiq0o1PMJq2Tq/KQIS2NPN1Xqzf1Fcj23muh\nn+bmLkihEVITS5Ka8QBNfrsiGPCdpgsTX84Oto+IEWC6X/7ohABZ150HzB5S/PwL\nnYyvedSxVc1vOjui4gT9Z4ZckBQOLFULLCUNWOC4FwYAHkh5gddB79vdoqU/Xpk9x\nnhbyDm/p1obqnSek6zLogtC+2KP5Nm83djkh5gQdvvYV4ph3J6Y6pDuq+FVmkXKQ\nnm6ZtAgMBAAGgFzAVBgqhkiG9w0BCQcxCBMGMTIzNDU2MA0GCSqGSIb3DQEB\nCwUA\nnA4IBAQCEqefmIYUFlaIDxc2rjVRF2YDXXerEc7EHkFNo3fdFRylzxcG2yI8MB55L\n\nnwj8eifWEw7q/TCwVgjDbwdEPGhV1KvCbbS5s
```

```
w0LIpP/Z26PhJaGPVthNnJU7GKg6\nuGl3YpXa2+QCgLP1+Lag0nBpd0gQHURz5Qv1sGoOw0zVMsrPSiSMgNVsDu8i
wev\nVsSCS0aK0qRuIUMvx1WMadfnDANZ0vL82q5c8MII+hD663Bbxqdm3ncn2PJRIrc/\nFZV36+VZ84BMeLOTiz8N
XHFeKo6uUeNVMM8HyaKusB1sgmn44T3+jdVpCNgOW778\nMpQZ/M/EuJrE8KVTlgYX6Mbdhd79\n-----END
CERTIFICATE REQUEST-----\n"
}
```

服务器 SSL 卸载

获取服务端 SSL 卸载模板列表

Action: slb.sslserver.list

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.sslserver.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务端 SSL 卸载模版名称
cert	字符串		证书名称
key	字符串		私钥名称
password	字符串	长度 1-46	密码
ca_cert	字符串		CA 证书名称
ssl_version	整数	0, 1, 2, 3	版本; 0: SSL v3.0 ; 1: TLS v1.0 ; 2: TLS v1.1, 3: TLS v1.2
cipher_suite	数组		加密算法列表, 支持以下算法: "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA", "TLS1_RSA_EXPORT1024_RC4_56_MD5", "TLS1_RSA_EXPORT1024_RC4_56_SHA", "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", ,

			"TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", , "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"
--	--	--	---

响应举例:

```
[
  {
    "name": "server",
    "cert": "server.crt",
    "key": "server.key",
    "password": "",
    "ca_cert": "",
    "ssl_version": 3,
    "cipher_suite": [
      "TLS1_RSA_AES_128_SHA",
      "TLS1_RSA_AES_256_SHA",
      "TLS1_RSA_EXPORT1024_RC4_56_MD5",
      "TLS1_RSA_EXPORT1024_RC4_56_SHA",
      "TLS1_RSA_AES_128_SHA256",
      "TLS1_RSA_AES_256_SHA256",
      "SSL3_RSA_DES_40_CBC_SHA",
      "SSL3_RSA_DES_64_CBC_SHA",
      "SSL3_RSA_DES_192_CBC3_SHA",
      "SSL3_RSA_RC4_40_MD5",
      "SSL3_RSA_RC4_128_MD5",
      "SSL3_RSA_RC4_128_SHA",
      "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
      "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
      "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
      "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
      "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
      "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
      "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"
    ]
  }
]
```

获取指定服务端 SSL 卸载模板获取

Action: slb.sslserver.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务端 SSL 卸载模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.sslserver.get

请求 body:

```
{  
  "name": "server"  
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务端 SSL 卸载模版名称
cert	字符串		证书名称
key	字符串		私钥名称
password	字符串	长度 1-46	密码
ca_cert	字符串		CA 证书名称
ssl_version	整数	0, 1, 2, 3	版本; 0: SSL v3.0 ; 1: TLS v1.0 ; 2: TLS v1.1, 3: TLS v1.2
cipher_suite	数组		加密算法列表, 支持以下算法: "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA", "TLS1_RSA_EXPORT1024_RC4_56_MD5", "TLS1_RSA_EXPORT1024_RC4_56_SHA", "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", , "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", , "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"

响应举例:

```
{
```

```

"name": "server",
"cert": "server.crt",
"key": "server.key",
"password": "",
"ca_cert": "",
"ssl_version": 3,
"cipher_suite": [
  "TLS1_RSA_AES_128_SHA",
  "TLS1_RSA_AES_256_SHA",
  "TLS1_RSA_EXPORT1024_RC4_56_MD5",
  "TLS1_RSA_EXPORT1024_RC4_56_SHA",
  "TLS1_RSA_AES_128_SHA256",
  "TLS1_RSA_AES_256_SHA256",
  "SSL3_RSA_DES_40_CBC_SHA",
  "SSL3_RSA_DES_64_CBC_SHA",
  "SSL3_RSA_DES_192_CBC3_SHA",
  "SSL3_RSA_RC4_40_MD5",
  "SSL3_RSA_RC4_128_MD5",
  "SSL3_RSA_RC4_128_SHA",
  "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
  "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
  "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
  "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
  "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
  "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
  "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"
]
}

```

增加服务端 SSL 卸载模板

Action: slb.sslserver.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	唯一
cert	字符串		证书名称	否	必须存在
key	字符串		私钥名称	否	必须存在
password	字符串	长度 1-46	密码	否	缺省值:"" 空字符串
ca_cert	字符串		CA 证书名称	否	必须存在; 缺省值:"" 空字符串
ssl_version	整数	0, 1, 2, 3	SSL 协议版本	否	版本; 0: SSL v3.0 ; 1: TLS v1.0 ; 2: TLS v1.1, 3: TLS v1.2。缺省值:0
cipher_suite	数组		加密算法列表	否	支持以下算法: "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA",

				"TLS1_RSA_EXPORT1024_RC4_56_MD5", "TLS1_RSA_EXPORT1024_RC4_56_SHA", "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA" 缺省值:支持的全部算法
--	--	--	--	--

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=3dc96e30a551175be4f021fb6325d7&action=slb.sslserver.add

请求 body:

```
{
  "name": "server",
  "cert": "server.crt",
  "key": "server.crt",
  "password": "",
  "ca_cert": "",
  "ssl_version": 3,
  "cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
```

```

"SSL3_RSA_DES_64_CBC_SHA",
"SSL3_RSA_DES_192_CBC3_SHA",
"SSL3_RSA_RC4_40_MD5",
"SSL3_RSA_RC4_128_MD5",
"SSL3_RSA_RC4_128_SHA",
"TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
"TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
"TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
"TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
"TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
"TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
"TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"
]
}

```

服务端 SSL 卸载模板编辑

Action: slb.sslserver.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	唯一
cert	字符串		证书名称	否	必须存在
key	字符串		私钥名称	否	必须存在
password	字符串	长度 1-46	密码	否	缺省值: "" 空字符串
ca_cert	字符串		CA 证书名称	否	必须存在; 缺省值: "" 空字符串
ssl_version	整数	0, 1, 2, 3	SSL 协议版本	否	版本; 0: SSL v3.0 ; 1: TLS v1.0 ; 2: TLS v1.1, 3: TLS v1.2。缺省值: 0
cipher_suite	数组		加密算法列表	否	支持以下算法: "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA", "TLS1_RSA_EXPORT1024_RC4_56_MD5", "TLS1_RSA_EXPORT1024_RC4_56_SHA", "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA",

				"TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA" 缺省值:支持的全部算法
--	--	--	--	--

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=3dc96e30a551175be4f021fb6325d7&action=s1b.sslserver.edit
 请求 body:

```
{
  "name": "server",
  "cert": "server.crt",
  "key": "server.crt",
  "password": "",
  "ca_cert": "",
  "ssl_version": 3,
  "cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
    "SSL3_RSA_DES_64_CBC_SHA",
    "SSL3_RSA_DES_192_CBC3_SHA",
    "SSL3_RSA_RC4_40_MD5",
    "SSL3_RSA_RC4_128_MD5",
    "SSL3_RSA_RC4_128_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
```

```
"TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
"TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA"
```

```
]
}
```

服务端 SSL 卸载模板删除

Action: slb.sslserver.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务端 SSL 卸载模版名称	是	必须存在

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=3dc96e30a551175be4f021fb6325d7&action=slb.sslserver.del
请求 body:
```

```
{
  "name": "sever"
}
```

客户端 SSL 卸载

客户端 SSL 卸载模板列表

Action: slb.sslclient.list

请求参数: 无

请求举例:

```
GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.sslclient.list
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	客户端 SSL 卸载模版名称
cert	字符串		第一证书名称 (可选 RSA,ECC,国密签名证书)
chain_cert	字符串		第一证书链名称 (可选 RSA,ECC,国密签名证书)
key	字符串		第一证书私钥名称
password	字符串	长度 1-46	第一证书密码
dcert	字符串		第二证书名称
dchain_cert	字符串		第二证书链名称
dkey	字符串		第二证书私钥名称
dpassword	字符串	长度 1-46	第二证书密码
ecert	字符串		国密加密证书名称

ekey	字符串			国密加密证书私钥
cache_num	整数	0-131072		会话缓存大小
cache_timeout	整数	0-10000000		会话缓存超时，单位秒
disable_ssl30	整数	0/1/		SSL3.0 协议版本开关：0 开启，1 关闭
disable_tls10	整数	0/1/		TLS1.0 协议版本开关：0 开启，1 关闭
disable_tls11	整数	0/1/		TLS1.1 协议版本开关：0 开启，1 关闭
disable_tls12	整数	0/1/		TLS1.2 协议版本开关：0 开启，1 关闭
client_cert_req	整数	0/1/2		检查客户端证书：0 必需；1 可选；2 忽略
revoke	字符串	长度 1-63		吊销证书列表名称
ca_cert	字符串			客户端 CA 证书名称
cipher_suite	数组			加密算法列表，支持以下算法： "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA", "TLS1_RSA_EXPORT1024_RC4_56_MD5", ", "TLS1_RSA_EXPORT1024_RC4_56_SHA" , "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA", "GM1_ECC_SM4_SM3", "GM1_ECDHE_SM4_SM3"
sni_list	数组	domain	长度 1-191	SNI 证书主机名
		cert		SNI 证书名称

		key		SNI 证书私钥
		password	长度 1-63	SNI 证书密码

响应举例:

```
{
  "name": "test",
  "cert": "4096mycert.pem",
  "chain_cert": "",
  "key": "4096mykey.pem",
  "password": "*****",
  "dcert": "www.test.com.cer",
  "dchain_cert": "",
  "dkey": "www.test.com.key",
  "dpassword": "*****",
  "ecert": "www.test.com.enc.cer",
  "ekey": "www.test.com.enc.key",
  "epassword": "*****",
  "cache_num": 0,
  "cache_timeout": 0,
  "disable_ssl30": 1,
  "disable_tls10": 1,
  "disable_tls11": 0,
  "disable_tls12": 0,
  "client_cert_req": 2,
  "revoke": "client.crl",
  "ca_cert": [
    "rsa.crt"
  ],
}
```

```

"cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
    "SSL3_RSA_DES_64_CBC_SHA",
    "SSL3_RSA_DES_192_CBC3_SHA",
    "SSL3_RSA_RC4_40_MD5",
    "SSL3_RSA_RC4_128_MD5",
    "SSL3_RSA_RC4_128_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA",
    "GM1_ECC_SM4_SM3",
    "GM1_ECDHE_SM4_SM3"
],
"sni_list": [
    {
        "domain": "www.abc.com",
        "cert": "www.abc.com.cer",
        "key": "www.abc.com.key",
        "password": ""
    }
]
}

```

客户端 SSL 卸载模板获取

Action: slb.sslclient.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	客户端 SSL 卸载模版名称	是	必须存在

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=4f1fdc5dca55b80fd8fbd743add066&action=slb.sslclient.get>
 请求 body:

```
{
```

```
"name": "sslc"
```

```
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	客户端 SSL 卸载模版名称
cert	字符串		第一证书名称 (可选 RSA,ECC,国密签名证书)
chain_cert	字符串		第一证书链名称 (可选 RSA,ECC,国密签名证书)
key	字符串		第一证书私钥名称
password	字符串	长度 1-46	第一证书密码
dcert	字符串		第二证书名称
dchain_cert	字符串		第二证书链名称
dkey	字符串		第二证书私钥名称
dpassword	字符串	长度 1-46	第二证书密码
ecert	字符串		国密加密证书名称
ekey	字符串		国密加密证书私钥
cache_num	整数	0-131072	会话缓存大小
cache_timeout	整数	0-10000000	会话缓存超时, 单位秒
disable_ssl30	整数	0/1/	SSL3.0 协议版本开关: 0 开启, 1 关闭
disable_tls10	整数	0/1/	TLS1.0 协议版本开关: 0 开启, 1 关闭
disable_tls11	整数	0/1/	TLS1.1 协议版本开关: 0 开启, 1 关闭
disable_tls12	整数	0/1/	TLS1.2 协议版本开关: 0 开启, 1 关闭
client_cert_req	整数	0/1/2	检查客户端证书: 0 必需; 1 可选; 2 忽略
revoke	字符串	长度 1-63	吊销证书列表名称
ca_cert	字符串		客户端 CA 证书名称
cipher_suite	数组		加密算法列表, 支持以下算法: "TLS1_RSA_AES_128_SHA", "TLS1_RSA_AES_256_SHA", "TLS1_RSA_EXPORT1024_RC4_56_MD5", ", "TLS1_RSA_EXPORT1024_RC4_56_SHA", , "TLS1_RSA_AES_128_SHA256", "TLS1_RSA_AES_256_SHA256", "SSL3_RSA_DES_40_CBC_SHA", "SSL3_RSA_DES_64_CBC_SHA", "SSL3_RSA_DES_192_CBC3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5", "SSL3_RSA_RC4_128_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",

				"TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA", "GM1_ECC_SM4_SM3", "GM1_ECDHE_SM4_SM3"
sni_list	数组	domain	长度 1-191	SNI 证书主机名
		cert		SNI 证书名称
		key		SNI 证书私钥
		password	长度 1-63	SNI 证书密码

响应举例:

```

{
  "name": "test",
  "cert": "4096mycert.pem",
  "chain_cert": "",
  "key": "4096mykey.pem",
  "password": "*****",
  "dcert": "www.test.com.cer",
  "dchain_cert": "",
  "dkey": "www.test.com.key",
  "dpassword": "*****",
  "ecert": "www.test.com.enc.cer",
  "ekey": "www.test.com.enc.key",
  "epassword": "*****",
  "cache_num": 0,
  "cache_timeout": 0,
  "disable_ssl30": 1,
  "disable_tls10": 1,
  "disable_tls11": 0,
  "disable_tls12": 0,
  "client_cert_req": 2,
  "revoke": "client.crl",
  "ca_cert": [

```

```

    "rsa.crt"
  ],
  "cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
    "SSL3_RSA_DES_64_CBC_SHA",
    "SSL3_RSA_DES_192_CBC3_SHA",
    "SSL3_RSA_RC4_40_MD5",
    "SSL3_RSA_RC4_128_MD5",
    "SSL3_RSA_RC4_128_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA",
    "GM1_ECC_SM4_SM3",
    "GM1_ECDHE_SM4_SM3"
  ],
  "sni_list": [
    {
      "domain": "www.abc.com",
      "cert": "www.abc.com.cer",
      "key": "www.abc.com.key",
      "password": ""
    }
  ]
}

```

客户端 SSL 卸载模板增加

Action: `slb.sslclient.add`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	唯一
cert	字符串		第一证书名称	否	
chain_cert	字符串		第一证书链名称	否	
key	字符串		第一证书私钥名称	否	
password	字符串		第一证书密码	否	缺省值: "" 空字符串
dcert	字符串		第二证书名称	否	
dchain_cert	字符串		第二证书链名称	否	
dkey	字符串		第二证书私钥名称	否	
dpassword	字符串		第二证书密码	否	缺省值: "" 空字符串
ecert	字符串		国密加密证书		依据第一第二证书
ekey	字符串		国密加密证书私钥		
cache_num	整数	0-131072	会话缓存大小	否	默认为 0
cache_timeout	整数	0-10000000	会话缓存超时，单位秒	否	默认为 0
client_cert_req	整数	0/1/2	检查客户端证书	否	0 必需；1 可选；2 忽略缺省值:0
revoke	字符串	长度 1-63	吊销证书列表名称	否	缺省值: "" 空字符串
ca_cert	数组		客户端 CA 证书名称列表	否	缺省值: "" 空字符串
cipher_suite	数组		加密算法列表	否	支持以下算法： "TLS1_RSA_AES_128_SHA" , "TLS1_RSA_AES_256_SHA" , "TLS1_RSA_EXPORT1024_RC4_56_MD5", "TLS1_RSA_EXPORT1024_RC4_56_SHA", "TLS1_RSA_AES_128_SHA"

					256", "TLS1_RSA_AES_256_SHA 256", "SSL3_RSA_DES_40_CBC_ SHA", "SSL3_RSA_DES_64_CBC_ SHA", "SSL3_RSA_DES_192_CBC 3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5 ", "SSL3_RSA_RC4_128_SHA" , "TLS1_ECDHE_ECDSA_WI TH_AES_128_GCM_SHA25 6", "TLS1_ECDHE_ECDSA_WI TH_AES_256_GCM_SHA38 4", "TLS1_ECDHE_ECDSA_WI TH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WI TH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH _AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH _AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH _AES_128_CBC_SHA",
sni_list	数组	doma in	长 度 1-1 91	SNI 证书主机 名	否
		cert		SNI 证书名称	否
		key		SNI 证书私钥	否
		pass word	长 度 1-6 3	SNI 证书密码	否

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=s1b.sslclient.add

请求 body:

```
{
  "name": "ssl",
  "cert": "aes256.crt",
  "chain_cert": "",
  "key": "aes256.key",
  "password": "secret",
  "dcert": "eccprime256v1.cert.pem",
  "dchain_cert": "",
  "dkey": "eccprime256v1.key.pem",
  "dpassword": "*****",
  "ecert": "",
  "ekey": "",
  "epassword": "*****",
  "cache_num": 131072,
  "cache_timeout": 10000000,
  "disable_ssl30": 0,
  "disable_tls10": 0,
  "disable_tls11": 0,
  "disable_tls12": 0,
  "client_cert_req": 0,
  "revoke": "rsa_Cin_4096.crl",
  "ca_cert": [
    "ecc_Cin_ca.pem"
  ],
  "cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
    "SSL3_RSA_DES_64_CBC_SHA",
    "SSL3_RSA_DES_192_CBC3_SHA",
    "SSL3_RSA_RC4_40_MD5",
    "SSL3_RSA_RC4_128_MD5",
    "SSL3_RSA_RC4_128_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA",
```

```

    "GM1_ECC_SM4_SM3",
    "GM1_ECDHE_SM4_SM3"
  ],
  "sni_list": [
    {
      "domain": "domain114ecc",
      "cert": "ecc.pem",
      "key": "ecc.key",
      "password": ""
    }
  ]
}

```

客户端 SSL 卸载模板编辑

Action: slb.sslclient.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	模版名称	是	唯一
cert	字符串		第一证书名称	否	
chain_cert	字符串		第一证书链名称	否	
key	字符串		第一证书私钥名称	否	
password	字符串		第一证书密码	否	缺省值: "" 空字符串
dcert	字符串		第二证书名称	否	
dchain_cert	字符串		第二证书链名称	否	
dkey	字符串		第二证书私钥名称	否	
dpassword	字符串		第二证书密码	否	缺省值: "" 空字符串
ecert	字符串		国密加密证书		依据第一第二证书
ekey	字符串		国密加密证书私钥		
cache_num	整数	0-131072	会话缓存大小	否	默认为 0
cache_timeout	整数	0-10000000	会话缓存超时, 单位秒	否	默认为 0
client_cert_req	整数	0/1/2	检查客户端证书	否	0 必需; 1 可选; 2 忽略缺省值: 0
revoke	字符串	长度 1-63	吊销证书列表名称	否	缺省值: "" 空字符串
ca_cert	数组		客户端 CA 证书名称列表	否	缺省值: "" 空字符串
cipher_suite	数组		加密算法列表	否	支持以下算法: "TLS1_RSA_AES_128_SHA" , "TLS1_RSA_AES_256_SHA"

					<pre> ", "TLS1_RSA_EXPORT1024_ RC4_56_MD5", "TLS1_RSA_EXPORT1024_ RC4_56_SHA", "TLS1_RSA_AES_128_SHA 256", "TLS1_RSA_AES_256_SHA 256", "SSL3_RSA_DES_40_CBC_ SHA", "SSL3_RSA_DES_64_CBC_ SHA", "SSL3_RSA_DES_192_CBC 3_SHA", "SSL3_RSA_RC4_40_MD5", "SSL3_RSA_RC4_128_MD5 ", "SSL3_RSA_RC4_128_SHA" , "TLS1_ECDHE_ECDSA_WI TH_AES_128_GCM_SHA25 6", "TLS1_ECDHE_ECDSA_WI TH_AES_256_GCM_SHA38 4", "TLS1_ECDHE_ECDSA_WI TH_AES_128_CBC_SHA", "TLS1_ECDHE_ECDSA_WI TH_AES_256_CBC_SHA", "TLS1_ECDHE_RSA_WITH _AES_128_GCM_SHA256", "TLS1_ECDHE_RSA_WITH _AES_256_GCM_SHA384", "TLS1_ECDHE_RSA_WITH _AES_128_CBC_SHA", </pre>
sni_list	数组	doma in	长 度 1-1 91	SNI 证书主机 名	否
		cert		SNI 证书名称	否
		key		SNI 证书私钥	否
		pass	长	SNI 证书密码	否

		word	度			
			1-6			
			3			

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.sslclient.edit

请求 body:

```
{
  "name": "ssl",
  "cert": "aes256.crt",
  "chain_cert": "",
  "key": "aes256.key",
  "password": "secret",
  "dcert": "eccprime256v1.cert.pem",
  "dchain_cert": "",
  "dkey": "eccprime256v1.key.pem",
  "dpassword": "*****",
  "ecert": "",
  "ekey": "",
  "epassword": "*****",
  "cache_num": 131072,
  "cache_timeout": 10000000,
  "disable_ssl30": 0,
  "disable_tls10": 0,
  "disable_tls11": 0,
  "disable_tls12": 0,
  "client_cert_req": 0,
  "revoke": "rsa_Cin_4096.crl",
  "ca_cert": [
    "ecc_Cin_ca.pem"
  ],
  "cipher_suite": [
    "TLS1_RSA_AES_128_SHA",
    "TLS1_RSA_AES_256_SHA",
    "TLS1_RSA_EXPORT1024_RC4_56_MD5",
    "TLS1_RSA_EXPORT1024_RC4_56_SHA",
    "TLS1_RSA_AES_128_SHA256",
    "TLS1_RSA_AES_256_SHA256",
    "SSL3_RSA_DES_40_CBC_SHA",
    "SSL3_RSA_DES_64_CBC_SHA",
    "SSL3_RSA_DES_192_CBC3_SHA",
    "SSL3_RSA_RC4_40_MD5",
    "SSL3_RSA_RC4_128_MD5",
    "SSL3_RSA_RC4_128_SHA",
  ]
}
```



```

    "TLS1_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
    "TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA",
    "TLS1_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "TLS1_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA",
    "GM1_ECC_SM4_SM3",
    "GM1_ECDHE_SM4_SM3"
  ],
  "sni_list": [
    {
      "domain": "domain114ecc",
      "cert": "ecc.pem",
      "key": "ecc.key",
      "password": ""
    }
  ]
}

```

策略

黑白名单

黑白名单上传

Action: slb.bwlist.upload

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	黑白名单名称	是	唯一

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.bwlist.upload&name=bw1>



Note

此 API 需要使用 form-data 的方式上传黑白名单文件

参数 name 在 URL 中不在 Body 中

黑白名单删除

Action: slb.bwlist.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	黑白名单名称	是	必须存在

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.bwlist.del
{ "name": "myname" }

黑白名单列表

Action: slb.bwlist.list

请求参数:无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=slb.bwlist.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	黑白名单名称

响应举例:

```
[  
  {  
    "name": "bw_list"  
  },  
  {  
    "name": "myname"  
  }  
]
```



Note

黑白名单用于策略模板中调用使用

规则表

获取规则表列表

Action: slb.ruletable.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	规则表名称
entrys	数组		规则条目组成的数组
ip	字符串	IP 地址	IP 地址掩码, 格式例如 1.1.1.1/32;2.2.2.0/24
id	整数	1-30	id
age	整数	0-1800	生存时间,单位秒, 0 表示永久

响应举例:

```

[[
  {
    "name": "rt1",
    "entrys": [
      {
        "ip": "1.2.3.4/32",
        "id": 1,
        "age": 0
      },
      {
        "ip": "1.2.3.5/32",
        "id": 2,
        "age": 123
      }
    ]
  }
]]

```

获取指定规则表

Action: slb.ruletable.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	规则表名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.get

请求 body:

```

{
  "name": "rt1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	规则表名称
entrys	数组		规则条目组成的数组

ip	字符串	IP 地址	IP 地址掩码, 格式例如 1.1.1.1/32;2.2.2.0/24
id	整数	1-30	id
age	整数	0-1800	生存时间,单位秒, 0 表示永久

响应举例:

```
{
  "name": "rt1",
  "entrys": [
    {
      "ip": "1.2.3.4/32",
      "id": 1,
      "age": 0
    },
    {
      "ip": "1.2.3.5/32",
      "id": 2,
      "age": 123
    }
  ]
}
```

增加规则表

Action: slb.ruletable.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	规则表名称	是	唯一

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.add

请求 body:

```
{
  "ruletable":
  {
    "name": "rt1"
  }
}
```

删除规则表

Action: slb.ruletable.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	规则表名称	是	必须存在

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.del
请求 body:
{
  "name": "rt1"
}
```

增加规则表条目

Action: slb.ruletable.entry.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	规则表名称	是	必须存在
entrys	数组		规则条目组成的数组	是	
ip	字符串		IP 地址掩码	是	格式例如 1.1.1.1/32;2.2.2.0/24
id	整数	1-30	id	否	缺省值:1
age	整数	0-1800	生存时间,单位秒	否	0 表示永久; 缺省值:0

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.entry.add
请求 body:
{
  "name": "rt1",
  "entrys": [
    {
      "ip": "1.2.3.4/32",
      "id": 1,
      "age": 0
    },
    {
      "ip": "1.2.3.5/32",
      "id": 2,
      "age": 123
    }
  ]
}
```

删除规则表条目

Action: slb.ruletable.entry.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	规则表名称	是	必须存在
entrys	数组		规则条目组成的数组	是	
ip	字符串	IP 地址	IP 地址掩码	是	格式例如 1.1.1.1/32;2.2.2.0/24

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.ruletable.entry.del

请求 body:

```
{
  "name": "rt1",
  "entrys": [
    {
      "ip": "1.2.3.4/32"
    }
  ]
}
```

策略

策略模板列表

Action: slb.policy.list

请求参数: 无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.policy.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	策略模版名称
match_dst_ip	整数	0-1	匹配目的 IP 地址;1:是; 0:否
match_overlap	整数	0-1	覆盖匹配;1:是; 0:否
bwlist_name	字符串	长度 1-63	黑白名单名称
ruletable_name	字符串	长度 1-63	规则表名称
match_client	整数	0-2	客户端匹配方式; 0:源 IP 匹配; 1:目的 IP 匹配; 2:头名称匹配
header_name	字符串	长度 1-191	头名称
bwlists	数组		基于黑白名单的策略规则列表

type	字符串	长度 1-63	基于黑白名单的策略规则-算法： DROP 丢弃 RESET 重置 其它字符串 服务池名称
id	整数	0-31	基于黑白名单的策略规则-黑白名单条目 id
ruletables	数组		基于规则表的策略规则列表
id	整数	0-1023	基于规则表的策略规则-规则表条目 id
conn_limit	整数	0-1048575	基于规则表的策略规则-并发连接限制
conn_rate_limit	整数	0-2147483647	基于规则表的策略规则-连接速率限制
req_limit	整数	0-1048575	基于规则表的策略规则-并发请求限制
req_rate_limit	整数	0-4294967295	基于规则表的策略规则-请求速率限制
type	整数	0-31	基于规则表的策略规则-算法;1:丢弃; 0:重置
lock_time	整数	0-1023	基于规则表的策略规则-锁定时间,单位秒
pool_policys	数组		基于 ACL 的策略规则列表
acl_id	整数	2-198	基于 ACL 的策略规则-ACL ID
pool	字符串	长度 1-63	基于 ACL 的策略规则-服务池名称

响应举例:

```
[{
  "name": "p1",
  "match_dst_ip": 1,
  "match_overlap": 1,
  "bwlist_name": "bwl",
  "ruletable_name": "rt0",
  "match_client": 2,
  "header_name": "head",
  "bwlists": [
    {
      "type": "pool",
      "id": 1
    },
    {
      "type": "RESET",
      "id": 2
    },
    {
      "type": "DROP",
      "id": 3
    }
  ],
  "ruletables": [
    {
      "id": 1,
      "conn_limit": 123,
      "conn_rate_limit": 456,
      "req_limit": 789,
```

```

    "req_rate_limit": 555,
    "type": 1,
    "lock_time": 666
  }
],
"pool_policys": [
  {
    "acl_id": 101,
    "pool": "pool"
  }
]
}]

```

策略模板获取

Action: slb.policy.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.policy.get

请求 body:

```

{
  "name": "p1"
}

```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	策略模版名称
match_dst_ip	整数	0-1	匹配目的 IP 地址;1:是; 0:否
match_overlap	整数	0-1	覆盖匹配;1:是; 0:否
bwlist_name	字符串	长度 1-63	黑白名单名称
ruletable_name	字符串	长度 1-63	规则表名称
match_client	整数	0-2	客户端匹配方式; 0:源 IP 匹配; 1:目的 IP 匹配; 2:头名称匹配
header_name	字符串	长度 1-63	头名称
bwlists	数组		基于黑白名单的策略规则列表
type	字符串	长度 1-63	基于黑白名单的策略规则-算法: DROP 丢弃 RESET 重置 其它字符串 服务池名称

id	整数	0-31	基于黑白名单的策略规则-黑白名单条目 id
ruletables	数组		基于规则表的策略规则列表
id	整数	0-1023	基于规则表的策略规则-规则表条目 id
conn_limit	整数	0-1048575	基于规则表的策略规则-并发连接限制
conn_rate_limit	整数	0-2147483647	基于规则表的策略规则-连接速率限制
req_limit	整数	0-1048575	基于规则表的策略规则-并发请求限制
req_rate_limit	整数	0-4294967295	基于规则表的策略规则-请求速率限制
type	整数	0-31	基于规则表的策略规则-算法;1:丢弃; 0:重置
lock_time	整数	0-1023	基于规则表的策略规则-锁定时间,单位秒
pool_policys	数组		基于 ACL 的策略规则列表
acl_id	整数	2-198	基于 ACL 的策略规则-ACL ID
pool	字符串	长度 1-63	基于 ACL 的策略规则-服务池名称

响应举例:

```
{
  "name": "p1",
  "match_dst_ip": 1,
  "match_overlap": 1,
  "bwlist_name": "bwl",
  "ruletable_name": "rt0",
  "match_client": 2,
  "header_name": "head",
  "bwlists": [
    {
      "type": "pool",
      "id": 1
    },
    {
      "type": "RESET",
      "id": 2
    },
    {
      "type": "DROP",
      "id": 3
    }
  ],
  "ruletables": [
    {
      "id": 1,
      "conn_limit": 123,
      "conn_rate_limit": 456,
      "req_limit": 789,
      "req_rate_limit": 555,
      "type": 1,
      "lock_time": 666
    }
  ]
}
```

```

    }
  ],
  "pool_policys": [
    {
      "acl_id": 101,
      "pool": "pool"
    }
  ]
}

```

策略模板增加

Action: slb.policy.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略模版名称	是	唯一
match_dst_ip	整数	0-1	匹配目的 IP 地址;	否	1:是; 0:否; 缺省值:0
match_overlap	整数	0-1	覆盖匹配;1:是; 0:否	否	1:是; 0:否; 缺省值:0
bwlist_name	字符串	长度 1-63	黑白名单名称	否	缺省值:"" 空字符
ruletable_name	字符串	长度 1-63	规则表名称	否	缺省值:"" 空字符
match_client	整数	0-2	客户端匹配方式;	否	0:源 IP 匹配; 1:目的 IP 匹配; 2:头名称匹配; 缺省值:0
header_name	字符串	长度 1-63	头名称	否	缺省值:"" 空字符
bwlists	数组		基于黑白名单的策略规则列表	否	缺省值:空
type	字符串	长度 1-63	基于黑白名单的策略规则-算法:	否	DROP 丢弃 RESET 重置 其它字符串 服务池名称
id	整数	0-31	基于黑白名单的策略规则-黑白名单条目 id	否	
ruletables	数组		基于规则表的策略规则列表	否	缺省值:空
id	整数	0-1023	基于规则表的策略规则-规则表条目 id	否	
conn_limit	整数	0-1048575	基于规则表的策略规则-并发连接限制	否	
conn_rate_limit	整数	0-2147483647	基于规则表的策略规则-连接速率限制	否	
req_limit	整数	0-1048575	基于规则表的策略规则-并发请求限制	否	
req_rate_limit	整数	0-4294967295	基于规则表的策略规则-请求速率限制	否	
type	整数	0-31	基于规则表的策略规则-算法;	否	1:丢弃; 0:重置;
lock_time	整数	0-1023	基于规则表的策略规则-锁定时间, 单位秒	否	

pool_policys	数组		基于 ACL 的策略规则列表	否	缺省值:空
acl_id	整数	2-198	基于 ACL 的策略规则-ACL ID	否	
pool	字符串	长度 1-63	基于 ACL 的策略规则-服务池名称	否	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.policy.add

请求 body:

```
{
  "name": "p1",
  "match_dst_ip": 1,
  "match_overlap": 1,
  "bwlist_name": "bwl",
  "ruletable_name": "rt0",
  "match_client": 2,
  "header_name": "head",
  "bwlists": [
    {
      "type": "pool",
      "id": 1
    },
    {
      "type": "RESET",
      "id": 2
    },
    {
      "type": "DROP",
      "id": 3
    }
  ],
  "ruletables": [
    {
      "id": 1,
      "conn_limit": 123,
      "conn_rate_limit": 456,
      "req_limit": 789,
      "req_rate_limit": 555,
      "type": 1,
      "lock_time": 666
    }
  ],
  "pool_policys": [
    {
      "acl_id": 101,
      "pool": "pool"
    }
  ]
}
```

```
]
}
```

策略模板编辑

Action: slb.policy.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略模版名称	是	必须存在
match_dst_ip	整数	0-1	匹配目的 IP 地址;	否	1:是; 0:否; 缺省值:0
match_overlap	整数	0-1	覆盖匹配;1:是; 0:否	否	1:是; 0:否; 缺省值:0
bwlist_name	字符串	长度 1-63	黑白名单名称	否	缺省值:"" 空字符
ruletable_name	字符串	长度 1-63	规则表名称	否	缺省值:"" 空字符
match_client	整数	0-2	客户端匹配方式;	否	0:源 IP 匹配; 1:目的 IP 匹配; 2:头名称匹配; 缺省值:0
header_name	字符串	长度 1-63	头名称	否	缺省值:"" 空字符
bwlists	数组		基于黑白名单的策略规则列表	否	缺省值:空
type	字符串	长度 1-63	基于黑白名单的策略规则-算法;	否	DROP 丢弃 RESET 重置 其它字符串 服务池名称
id	整数	0-31	基于黑白名单的策略规则-黑白名单条目 id	否	
ruletables	数组		基于规则表的策略规则列表	否	缺省值:空
id	整数	0-1023	基于规则表的策略规则-规则表条目 id	否	
conn_limit	整数	0-1048575	基于规则表的策略规则-并发连接限制	否	
conn_rate_limit	整数	0-2147483647	基于规则表的策略规则-连接速率限制	否	
req_limit	整数	0-1048575	基于规则表的策略规则-并发请求限制	否	
req_rate_limit	整数	0-4294967295	基于规则表的策略规则-请求速率限制	否	
type	整数	0-31	基于规则表的策略规则-算法;	否	1:丢弃; 0:重置;
lock_time	整数	0-1023	基于规则表的策略规则-锁定时间, 单位秒	否	
pool_policys	数组		基于 ACL 的策略规则列表	否	缺省值:空
acl_id	整数	2-198	基于 ACL 的策略规则-ACL ID	否	
pool	字符串	长度 1-63	基于 ACL 的策略规则-服务池名称	否	

请求举例:

POST

<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.policy.edit>

请求 body:

```
{
  "name": "p1",
  "match_dst_ip": 1,
  "match_overlap": 1,
  "bwlis_name": "bwl",
  "ruletable_name": "rt0",
  "match_client": 2,
  "header_name": "head",
  "bwlists": [
    {
      "type": "pool",
      "id": 1
    },
    {
      "type": "RESET",
      "id": 2
    },
    {
      "type": "DROP",
      "id": 3
    }
  ],
  "ruletables": [
    {
      "id": 1,
      "conn_limit": 123,
      "conn_rate_limit": 456,
      "req_limit": 789,
      "req_rate_limit": 555,
      "type": 1,
      "lock_time": 666
    }
  ],
  "pool_policys": [
    {
      "acl_id": 101,
      "pool": "pool"
    }
  ]
}
```

策略模板删除

Action: slb.policy.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略模版名称	是	必须存在

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=slb.policy.del

请求 body:

```
{
  "name": "p1"
}
```

Web 安全

WAF 模板

获取 WAF 模板列表

Action: waf.profile.list

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.profile.list

响应参数:

名称	类型	范围	含义
name	字符串	1-191	WAF 模板名称
rule_name	字符串	0-63	WAF 规则表名称, 默认空串
white_url_list	字符串列表	最多 32 个, 每个长度 1-127	url 白名单列表: 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回
black_url_list	字符串列表	最多 8 个, 每个长度 1-127	url 黑名单列表: 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回
uri_stick_check	布尔	0-1	url 严格检查开关: 0 关闭 1 开启
mode	整数	0-2	WAF 模式: 0: normal 流量识别后进行流量阻断 1: forward 只进行流量识别操作, 不阻断 2: learning 学习模式, 自动学习流量规则
behavior	整数	0-1	阻断行为: 0: block 数据流 1: drop 数据报
enable	布尔	0-1	使能: 表示模版是否工作, 0 关闭, 1 开启
logging	布尔	0-1	表示是否开启 log 日志: 0 关闭, 1 开启
injection_sql	布尔	0-1	表示是否开始 sql 注入: 0 关闭, 1 开启

injection_xss	布尔	0-1	表示是否开始 xss 注入：0 关闭，1 开启
disable_len_check	布尔	0-1	http 字段长度检查：表示是否关闭 http 字段长度检查
headers_mlen	整数	1-65535	头部允许最大长度
url_mlen"	整数	1-65535	URL 允许最大长度
cookie_mlen	整数	1-65535	COOKIE 允许最大长度

响应举例：

```

[[
  {
    "name": "wafprofile2",
    "rule_name": "",
    "white_url_list": ["^/abcb", "^/aa"],
    "black_url_list": ["^/123"],
    "uri_stick_check": 0,
    "mode": 0,
    "behavior": 1,
    "enable": 1,
    "logging": 1,
    "injection_sql": 1,
    "injection_xss": 1,
    "disable_len_check": 0,
    "headers_mlen": 8192,
    "url_mlen": 2803,
    "cookie_mlen": 4079
  }, {
    "name": "wafprofile1",
    "rule_name": "",
    "white_url_list": ["^/abc", "^/aaa"],
    "black_url_list": ["^/123"],
    "uri_stick_check": 0,
    "mode": 1,
    "behavior": 1,
    "enable": 1,
    "logging": 1,
    "injection_sql": 1,
    "injection_xss": 1,
    "disable_len_check": 1,
    "headers_mlen": 8192,
    "url_mlen": 2803,
    "cookie_mlen": 4079
  }
]]

```

获取指定 WAF 模板

Action: waf.profile.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	Waf 模板名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.profile.get

请求 body:

```
{  
  "name": "wafprofile1"  
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-63	WAF 模板名称
rule_name	字符串	0-63	WAF 规则表名称, 默认空串
white_url_list	字符串列表	最多 32 个, 每个长度 1-127	url 白名单列表: 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回
black_url_list	字符串列表	最多 8 个, 每个长度 1-127	url 白名单列表: 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回
uri_stick_check	布尔	0-1	url 严格检查开关: 0 关闭 1 开启
mode	整数	0-2	WAF 模式: 0: normal 流量识别后进行流量阻断 1: forward 只进行流量识别操作, 不阻断 2: learning 学习模式, 自动学习流量规则
behavior	整数	0-1	阻断行为: 0: block 数据流 1: drop 数据报
enable	布尔	0-1	使能: 表示模版是否工作, 0 关闭, 1 开启动
logging	布尔	0-1	表示是否开启 log 日志: 0 关闭, 1 开启
injection_sql	布尔	0-1	表示是否开始 sql 注入: 0 关闭, 1 开启
injection_xss	布尔	0-1	表示是否开始 xss 注入: 0 关闭, 1 开启
disable_len_check	布尔	0-1	http 字段长度检查: 表示是否关闭 http 字段长度检查
headers_mlen	整数	1-65535	头部允许最大长度
url_mlen	整数	1-65535	URL 允许最大长度
cookie_mlen	整数	1-65535	COOKIE 允许最大长度

响应举例:

```
{  
  "name": "wafprofile1",  
  "rule_name": "naxsi_core.rules",  
  "white_url_list": ["^/abcb", "^/aa"],  
  "black_url_list": ["^/123"],  
  "uri_stick_check": 0,  
  "mode": 0,  
  "behavior": 1,  
  "enable": 1,  
  "logging": 1,  
  "injection_sql": 1,
```



```

"injection_xss": 1,
"disable_len_check": 0,
"headers_mlen": 8192,
"url_mlen": 2803,
"cookie_mlen": 4079
}

```

WAF 模板增加

Action: waf.profile.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	WAF 模板名称	是	唯一
rule_name	字符串	0-63	WAF 规则表名称	否	默认空串
white_url_list	字符串列表	最多 32 个, 每个长度 1-127	url 白名单列表(默认[]): 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回	否	
black_url_list	字符串列表	最多 8 个, 每个长度 1-127	url 黑名单列表(默认[]): 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回	否	
uri_stick_check	布尔	0-1	url 严格检查开关: 0 关闭 1 开启, 默认 0	否	
mode	整数	0-2	WAF 模式(默认 0): 0: normal 流量识别后进行流量阻断 1: forward 只进行流量识别操作, 不阻断 2: learning 学习模式, 自动学习流量规则	否	
behavior	整数	0-1	阻断行为(默认 0): 0: block 数据流 1: drop 数据报	否	
enable	布尔	0-1	使能: 表示模版是否工作, 0 关闭, 1 开启动, 默认 0	否	
logging	布尔	0-1	表示是否开启 log 日志: 0 关闭, 1 开启, 默认 0	否	
injection_sql	布尔	0-1	表示是否开始 sql 注入: 0 关闭, 1 开启, 默认 0	否	
injection_xss	布尔	0-1	表示是否开始 xss 注入: 0 关闭, 1 开启, 默认 0	否	
disable_len_check	布尔	0-1	http 字段长度检查: 表示是否关闭 http 字段长度检查, 默认 0	否	0 时表示不关闭检查, 1 表示关闭检查
headers_mlen	整数	1-65535	头部允许最大长度, 默认 8192	否	

url_mlen	整数	1-65535	URL 允许最大长度, 默认 2803	否	
cookie_mlen	整数	1-65535	COOKIE 允许最大长度, 默认 4079	否	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.profile.add
 请求 body:

```
{
  "name": "112",
  "rule_name": "main_core.rules",
  "white_url_list": ["^/abcdefg1234567"],
  "black_url_list": ["^/aaabbbccc111222"],
  "uri_stick_check": 1,
  "mode": 1,
  "behavior": 1,
  "enable": 1,
  "logging": 1,
  "injection_sql": 1,
  "injection_xss": 1,
  "disable_len_check": 0,
  "headers_mlen": 33333,
  "url_mlen": 123,
  "cookie_mlen": 111
}
```



rule_name 是上传的 WAF 规则的名称

WAF 模板编辑

Action: waf.profile.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	WAF 模板名称	是	必须存在
rule_name	字符串	0-63	WAF 规则表名称	否	默认空串
white_url_list	字符串列表	最多 32 个, 每个长度 1-127	url 白名单列表(默认[]): 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不对会返回	否	
black_url_list	字符串列表	最多 8 个, 每个长度 1-127	url 白名单列表(默认[]): 最多 32 个正则表达式, 每个表达式长度 1-127 正则表达式后台检查合法性, 不	否	

			对会返回		
uri_stick_check	布尔	0-1	url 严格检查开关: 0 关闭 1 开启, 默认 0	否	
mode	整数	0-2	WAF 模式(默认 0): 0: normal 流量识别后进行流量阻断 1: forward 只进行流量识别操作, 不阻断 2: learning 学习模式, 自动学习流量规则	否	
behavior	整数	0-1	阻断行为(默认 0): 0: block 数据流 1: drop 数据报	否	
enable	布尔	0-1	使能: 表示模版是否工作, 0 关闭, 1 开启动, 默认 0	否	
logging	布尔	0-1	表示是否开启 log 日志: 0 关闭, 1 开启, 默认 0	否	
injection_sql	布尔	0-1	表示是否开始 sql 注入: 0 关闭, 1 开启, 默认 0	否	
injection_xss	布尔	0-1	表示是否开始 xss 注入: 0 关闭, 1 开启, 默认 0	否	
disable_len_check	布尔	0-1	http 字段长度检查: 表示是否关闭 http 字段长度检查, 默认 0	否	0 时表示不关闭检查, 1 表示关闭检查
headers_mlen	整数	1-65535	头部允许最大长度, 默认 8192	否	
url_mlen"	整数	1-65535	URL 允许最大长度, 默认 2803	否	
cookie_mlen	整数	1-65535	COOKIE 允许最大长度, 默认 4079	否	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.profile.edit

请求 body:

```
{
  "name": "112",
  "rule_name": "main_core.rules",
  "white_url_list": ["^/abcdefg12345678"],
  "black_url_list": ["^/aaabbbccc1112232"],
  "uri_stick_check": 0,
  "mode": 0,
  "behavior": 1,
  "enable": 1,
  "logging": 1,
  "injection_sql": 1,
  "injection_xss": 1,
  "disable_len_check": 0,
  "headers_mlen": 33333,
  "url_mlen": 123,
  "cookie_mlen": 111
}
```

}

WAF 模板删除

Action: waf.profile.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	Waf 模板名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.profile.del

请求 body:

```
{  
"name": "wafprofile3"  
}
```

WAF 规则

获取 WAF 规则脚本列表

Action: waf.rule.list

请求参数:无

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.rule.list

响应参数:

名称	类型	范围	含义
name	字符串	1-63	Waf 规则文件名
check	布尔	0-1	语法是否检查 1 表示语法合法, 0 表示语法不合法或没有检查语法
ref_count	整数	0-65535	当前使用数量

响应举例:

```
[[  
  {"name": "naxsi_core.rules",  
    "check": 1,  
    "ref_count": 2  
  }, {  
    "name": "main_core.rules",  
    "check": 1,  
    "ref_count": 0  
  }  
]]
```

获取指定 WAF 规则脚本

Action: waf.rule.get

请求参数:

名称	类型	范围	含义
name	字符串	1-63	Waf 规则文件名

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.rule.get

请求 body:

```
{
  "name": "naxsi_core.rules"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-63	Waf 规则文件名
check	布尔	0-1	语法是否检查 1 表示语法合法， 0 表示语法不合法或没有检查语法
ref_count	整数	0-65535	当前使用数量

响应举例:

```
{
  "name": "naxsi_core.rules",
  "check": 1,
  "ref_count": 2
}
```

WAF 规则脚本删除

Action: waf.rule.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-63	Waf 规则文件名	是	必须存在

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.rule.del

请求 body:

```
{
  "name": "naxsi_core.rules"
}
```

WAF 规则脚本下载

Action: waf.rule.download

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-63	Waf 规则文件名	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.rule.download
请求 body:

```
{  
  "name": "naxsi_core.rules"  
}
```



此 API 会下载一个 naxsi_core.rules 的文件

WAF 规则脚本上传

Action: waf.rule.upload

请求参数:无

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.rule.upload



此 API 使用 form-data 的方式上传一个 WAF 规则脚本文件

WAF 统计

WAF 统计获取

Action: waf.vserver.statis

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.vserver.statis

响应参数:

名称	类型	范围	含义
name	字符串	1-63	Waf 模板名称

req_check_total	整数	>=0	请求报文总数
req_check_matched	整数	>=0	请求报文匹配数量
req_check_block	整数	>=0	请求报文阻挡数量
uri_buf_check_total	整数	>=0	uri 超长报文总数
uri_buf_check_matched	整数	>=0	uri 超长报文匹配数量
uri_buf_check_block	整数	>=0	uri 超长报阻挡数量
hdr_buf_check_total	整数	>=0	Headers 超长检查报文总数
hdr_buf_check_matched	整数	>=0	Headers 超长检查匹配数量
hdr_buf_check_block	整数	>=0	Headers 超常检查阻挡数量
cookie_buf_check_total	整数	>=0	Cookie 超长检查报文总数
cookie_buf_check_matched	整数	>=0	Cookie 超长报文检查匹配数量
cookie_buf_check_block	整数	>=0	Cookie 超长报文检查阻挡数量
uri_wlist_check_total	整数	>=0	Uri 白名单检查报文总数
uri_wlist_check_matched	整数	>=0	Uri 白名单检查报文匹配数量
uri_wlist_check_block	整数	>=0	Uri 白名单检查报文阻挡数量
uri_blist_check_total	整数	>=0	Uri 黑名单检查报文总数
uri_blist_check_matched	整数	>=0	Uri 黑名单检查报文匹配数量
uri_blist_check_block	整数	>=0	Uri 黑名单检查报文阻挡数量
sql_injection_check_total	整数	>=0	Sql 注入检查总数
sql_injection_check_matched	整数	>=0	Sql 注入检查匹配数量
sql_injection_check_block	整数	>=0	Sql 注入检查阻挡数量
xss_injection_check_total	整数	>=0	Xss 注入检查总数
xss_injection_check_matched	整数	>=0	Xss 注入检查匹配数量
xss_injection_check_block	整数	>=0	Xss 注入检查阻挡数量
rule_check_total	整数	>=0	Waf 规则检查总数
rule_check_matched	整数	>=0	Waf 规则匹配数
rule_check_block	整数	>=0	Wad 规则阻挡数量

响应举例:

```

{{
  "name": "wafhttp",
  "req_check_total": 0,
  "req_check_matched": 0,
  "req_check_block":0,
  "uri_buf_check_total": 0,
  "uri_buf_check_matched": 0,
  "uri_buf_check_block": 0,
  "hdr_buf_check_total": 0,
  "hdr_buf_check_matched": 0,
  "hdr_buf_check_block":0,
  "cookie_buf_check_total": 0,
  "cookie_buf_check_matched": 0,
  "cookie_buf_check_block": 0,
  "uri_wlist_check_total":0,
  "uri_wlist_check_matched": 0,
  "uri_wlist_check_block": 0,

```

```

    "uri_blist_check_total": 0,
    "uri_blist_check_matched": 0,
    "uri_blist_check_block":0,
    "sql_injection_check_total": 0,
    "sql_injection_check_matched": 0,
    "sql_injection_check_block":0,
    "xss_injection_check_total": 0,
    "xss_injection_check_matched": 0,
    "xss_injection_check_block":0,
    "rule_check_total": 0,
    "rule_check_matched": 0,
    "rule_check_block": 0
}

```

WAF 统计清除

Action: waf.vserver.clear

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dfa0ebc0b62922154bee2943520b24&action=waf.vserver.clear

GSLB 全局负载均衡

GSLB 侦听地址

添加 GSLB 侦听地址

Action: gslb.dns_listener.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	侦听地址名称	是	唯一
ip	IPv4 地址		侦听地址	否	缺省值: 0.0.0.0
acl_id	整数	2-198	acl ID, 仅当 ip 为 0.0.0.0 时生效	否	缺省值: 无
enabled	整数	0, 1	侦听地址状态	否	1:启用;0:禁用;缺省值:1
link_service	整数	0, 1	服务联动	否	1:启用;0:禁用;缺省值:1
gslb_vport_list	数组		由侦听地址的侦听端口对象组成的数组	是	至少需要 1 个侦听端口对象, 最多 65535 个
port_name	字符串	长度 1-191	侦听端口对象名称	是	唯一

port	整数	0-65534	侦听端口对象—侦听端口	是	每侦听地址唯一
pool	字符串	长度 1-63	侦听端口对象—服务池	否	缺省值:空
enabled	整数	0,1	侦听端口对象—侦听端口状态	否	1:启用;0:禁用;缺省值:1
path_persistence	整数	0,1	侦听端口对象—路径保持	否	1:启用;0:禁用;缺省值:0
conn_limit	整数	1-8000000	侦听端口对象—连接限制	否	缺省值:8000000,表示不限制
snat_pool	字符串	长度 1-31	侦听端口对象—源地址转换地址池	否	缺省值:空
intf_enabled	字符串	长度 1-63	侦听端口对象—启用该侦听端口的接口名称,多个接口用逗号隔开,例如"ETH0/1,ETH0/4"	否	仅当侦听地址 ip 为 0.0.0.0 时可设置 缺省值:空
direct_forward	整数	0,1	侦听端口对象—直接转发	否	1:启用;0:禁用;缺省值:0
udp_profile	字符串	长度 1-191	侦听端口对象—UDP 模板	否	缺省值:空
snat_policy_list	数组		侦听端口对象—策略源地址转换列表	否	成员个数 0-32 个,缺省值:空
acl_id	整数	1-199	源地址转换策略—ACL ID	是	acl 必须存在
nat_pool	字符串	长度 1-31	源地址转换策略—NAT 地址池	是	nat_pool 必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.add

请求 body:

```
{
  "name" : "192.168.108.53_va",
  "ip" : "192.168.108.53",
  "enabled" : 1,
  "link_service" : 0,
  "gslb_vport_list" : [ {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
    "udp_profile" : "udp_profile",
    "snat_policy_list" : [ {
      "acl_id" : 100,
      "nat_pool" : "snat_pool"
    } ]
  } ]
}
```

获取 GSLB 侦听地址列表

Action: `gslb.dns_listener.list`

请求参数:无

请求举例:

POST `http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.list`

响应参数:

名称	类型	范围	含义
<code>name</code>	字符串	长度 1-191	侦听地址名称
<code>ip</code>	IPv4 地址		侦听地址
<code>acl_id</code>	整数	2-198	acl ID, 仅当 ip 为 0.0.0.0 时生效
<code>enabled</code>	整数	0, 1	侦听地址状态 1:启用;0:禁用;
<code>link_service</code>	整数	0, 1	服务联动 1:启用;0:禁用;
<code>gslb_vport_list</code>	数组		由侦听地址的侦听端口对象组成的数组
<code>port_name</code>	字符串	长度 1-191	侦听端口对象—名称
<code>port</code>	整数	0-65534	侦听端口对象—侦听端口
<code>pool</code>	字符串	长度 1-63	侦听端口对象—服务池
<code>enabled</code>	整数	0, 1	侦听端口对象—侦听端口状态 1:启用;0:禁用;
<code>path_persistence</code>	整数	0, 1	侦听端口对象—路径保持 1:启用;0:禁用;
<code>conn_limit</code>	整数	1-8000000	侦听端口对象—连接限制 8000000, 表示不限制
<code>snat_pool</code>	字符串	长度 1-31	侦听端口对象—源地址转换地址池
<code>intf_enabled</code>	字符串	长度 1-63	侦听端口对象—启用该侦听端口的接口名称, 多个接口用逗号隔开, 例如"ETH0/1, ETH0/4"
<code>direct_forward</code>	整数	0, 1	侦听端口对象—直接转发 1:启用;0:禁用;
<code>udp_profile</code>	字符串	长度 1-191	侦听端口对象—UDP 模板
<code>snat_policy_list</code>	数组		侦听端口对象—策略源地址转换列表
<code>acl_id</code>	整数	1-199	源地址转换策略—ACL ID
<code>nat_pool</code>	字符串	长度 1-31	源地址转换策略—NAT 地址池

响应举例:

```
[{
  "name" : "192.168.108.53_va",
  "ip" : "192.168.108.53",
  "enabled" : 1,
  "link_service" : 0,
  "gslb_vport_list" : [ {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
    "udp_profile" : "udp_profile",
```

```

    "snat_policy_list" : [ {
        "acl_id" : 100,
        "nat_pool" : "snat_pool"
    } ]
} ]
} ]
}}

```

获取 GSLB 指定侦听地址

Action: `gslb.dns_listener.get`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	侦听地址名称	是	唯一

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.get

Body: {"name": "192.168.108.53_va"}

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	侦听地址名称
ip	IPv4 地址		侦听地址
acl_id	整数	2-198	acl ID, 仅当 ip 为 0.0.0.0 时生效
enabled	整数	0, 1	侦听地址状态 1: 启用; 0: 禁用;
link_service	整数	0, 1	服务联动 1: 启用; 0: 禁用;
gslb_vport_list	数组		由侦听地址的侦听端口对象组成的数组
port_name	字符串	长度 1-191	侦听端口对象--名称
port	整数	0-65534	侦听端口对象--侦听端口
pool	字符串	长度 1-63	侦听端口对象--服务池
enabled	整数	0, 1	侦听端口对象--侦听端口状态 1: 启用; 0: 禁用;
path_persistence	整数	0, 1	侦听端口对象--路径保持 1: 启用; 0: 禁用;
conn_limit	整数	1-8000000	侦听端口对象--连接限制 8000000, 表示不限制
snat_pool	字符串	长度 1-31	侦听端口对象--源地址转换地址池
intf_enabled	字符串	长度 1-63	侦听端口对象--启用该侦听端口的接口名称, 多个接口用逗号隔开, 例如"ETH0/1, ETH0/4"
direct_forward	整数	0, 1	侦听端口对象--直接转发 1: 启用; 0: 禁用;
udp_profile	字符串	长度 1-191	侦听端口对象--UDP 模板
snat_policy_list	数组		侦听端口对象--策略源地址转换列表
acl_id	整数	1-199	源地址转换策略--ACL ID
nat_pool	字符串	长度 1-31	源地址转换策略--NAT 地址池

响应举例:

```
{
```

```

"name" : "192.168.108.53_va",
"ip" : "192.168.108.53",
"enabled" : 1,
"link_service" : 0,
"gslib_vport_list" : [ {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
    "udp_profile" : "udp_profile",
    "snat_policy_list" : [ {
        "acl_id" : 100,
        "nat_pool" : "snat_pool"
    } ]
} ]
} ]
}

```

编辑指定 GSLB 侦听地址

Action: `gslib.dns_listener.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	侦听地址名称	是	必须存在
<code>enabled</code>	整数	0, 1	侦听地址状态	否	1: 启用; 0: 禁用; 缺省值: 不修改
<code>link_service</code>	整数	0, 1	服务联动	否	1: 启用; 0: 禁用; 缺省值: 不修改
<code>gslib_vport_list</code>	数组		由侦听地址的侦听端口对象组成的数组	是	至少需要 1 个侦听端口对象, 最多 65535 个
<code>port_name</code>	字符串	长度 1-191	侦听端口对象--名称	是	唯一
<code>port</code>	整数	0-65534	侦听端口对象--侦听端口	是	每侦听地址唯一
<code>pool</code>	字符串	长度 1-63	侦听端口对象--服务池	否	缺省值: 不修改
<code>enabled</code>	整数	0, 1	侦听端口对象--侦听端口状态	否	1: 启用; 0: 禁用; 缺省值: 不修改
<code>path_persistence</code>	整数	0, 1	侦听端口对象--路径保持	否	1: 启用; 0: 禁用; 缺省值: 不修改
<code>conn_limit</code>	整数	1-8000000	侦听端口对象--连接限制	否	缺省值: 不修改, 表示不限制
<code>snat_pool</code>	字符串	长度 1-31	侦听端口对象--源地址转换地址池	否	缺省值: 不修改

intf_enable	字符串	长度 1-63	侦听端口对象—启用该侦听端口的接口名称, 多个接口用逗号隔开, 例如"ETH0/1, ETH0/4"	否	仅当侦听地址 ip 为 0.0.0.0 时可设置 缺省值: 不修改
direct_forward	整数	0,1	侦听端口对象—直接转发	否	1: 启用; 0: 禁用; 缺省值: 不修改
udp_profile	字符串	长度 1-191	侦听端口对象—UDP 模板	否	缺省值: 不修改
snat_policy_list	数组		侦听端口对象—策略源地址转换列表	否	成员个数 0-32 个, 缺省值: 不修改
acl_id	整数	1-199	源地址转换策略—ACL ID	是	acl 必须存在
nat_pool	字符串	长度 1-31	源地址转换策略—NAT 地址池	是	nat_pool 必须存在

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.edit

请求 body:

```
{
  "name" : "192.168.108.53_va",
  "enabled" : 1,
  "link_service" : 0,
  "gslb_vport_list" : [ {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
    "udp_profile" : "udp_profile",
    "snat_policy_list" : [ {
      "acl_id" : 100,
      "nat_pool" : "snat_pool"
    } ]
  } ]
}
```

添加侦听端口

Action: gslb.dns_listener.port.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	侦听地址名称	是	必须存在

gslb_vport	对象		侦听端口对象	是	
port_name	字符串	长度 1-191	侦听端口对象--名称	是	唯一
port	整数	0-65534	侦听端口对象--侦听端口	是	每侦听地址唯一
pool	字符串	长度 1-63	侦听端口对象--服务池	否	缺省值:空
enabled	整数	0, 1	侦听端口对象--侦听端口状态	否	1:启用;0:禁用;缺省值:1
path_persistence	整数	0, 1	侦听端口对象--路径保持	否	1:启用;0:禁用;缺省值:0
conn_limit	整数	1-8000000	侦听端口对象--连接限制	否	缺省值:8000000,表示不限制
snat_pool	字符串	长度 1-31	侦听端口对象--源地址转换地址池	否	缺省值:空
intf_enabled	字符串	长度 1-63	侦听端口对象--启用该侦听端口的接口名称,多个接口用逗号隔开,例如"ETH0/1,ETH0/4"	否	仅当侦听地址 ip 为 0.0.0.0 时可设置 缺省值:空
direct_forward	整数	0, 1	侦听端口对象--直接转发	否	1:启用;0:禁用;缺省值:0
udp_profile	字符串	长度 1-191	侦听端口对象--UDP 模板	否	缺省值:空
snat_policy_list	数组		侦听端口对象--策略源地址转换列表	否	成员个数 0-32 个,缺省值:空
acl_id	整数	1-199	源地址转换策略--ACL ID	是	
nat_pool	字符串	长度 1-191	源地址转换策略--NAT 地址池	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.port.add

请求 body:

```
{
  "name" : "192.168.108.53_va",
  "gslb_vport" : {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
    "udp_profile" : "udp_profile",
    "snat_policy_list" : [ {
      "acl_id" : 100,
      "nat_pool" : "snat_pool"
    } ]
  }
}
```

编辑侦听端口

Action: `gslb.dns_listener.port.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	侦听地址名称	是	必须存在
<code>gslb_vport</code>	对象		侦听端口对象	是	
<code>port_name</code>	字符串	长度 1-191	侦听端口对象--名称	是	必须存在
<code>port</code>	整数	0-65534	侦听端口对象--侦听端口	是	必须存在
<code>pool</code>	字符串	长度 1-63	侦听端口对象--服务池	否	缺省值:空
<code>enabled</code>	整数	0,1	侦听端口对象--侦听端口状态	否	1:启用;0:禁用;缺省值:不修改
<code>path_persistence</code>	整数	0,1	侦听端口对象--路径保持	否	1:启用;0:禁用;缺省值:不修改
<code>conn_limit</code>	整数	1-8000000	侦听端口对象--连接限制	否	缺省值:不修改,表示不限制
<code>snat_pool</code>	字符串	长度 1-31	侦听端口对象--源地址转换地址池	否	缺省值:不修改
<code>intf_enabled</code>	字符串	长度 1-63	侦听端口对象--启用该侦听端口的接口名称,多个接口用逗号隔开,例如"ETH0/1,ETH0/4"	否	仅当侦听地址 ip 为 0.0.0.0 时可设置 缺省值:不修改
<code>direct_forward</code>	整数	0,1	侦听端口对象--直接转发	否	1:启用;0:禁用;缺省值:不修改
<code>udp_profile</code>	字符串	长度 1-191	侦听端口对象--UDP 模板	否	缺省值:不修改
<code>snat_policy_list</code>	数组		侦听端口对象--策略源地址转换列表	否	成员个数 0-32 个,缺省值:不修改
<code>acl_id</code>	整数	1-199	源地址转换策略--ACL ID	是	
<code>nat_pool</code>	字符串	长度 1-191	源地址转换策略--NAT 地址池	是	

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.port.edit
```

请求 body:

```
{
  "name" : "192.168.108.53_va",
  "gslb_vport" : {
    "port_name" : "dns53",
    "port" : 53,
    "pool" : "udp_pool",
    "enabled" : 1,
    "path_persistence" : 1,
    "conn_limit" : 8000000,
    "snat_pool" : "snat_pool",
    "direct_forward" : 0,
```

```

    "udp_profile" : "udp_profile",
    "snat_policy_list" : [ {
        "acl_id" : 100,
        "nat_pool" : "snat_pool"
    } ]
}
}
}

```

删除侦听端口

Action: `gslb.dns_listener.port.del`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	侦听地址名称	是	必须存在
gslb_vport	对象		侦听端口对象	是	
port_name	字符串	长度 1-191	侦听端口对象--名称	是	必须存在
port	整数	0-65534	侦听端口对象--侦听端口	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.port.del

请求 body:

```

{
    "name" : "192.168.108.53_va",
    "gslb_vport" : {
        "port_name" : "dns53",
        "port" : 53
    }
}
}

```

删除 GSLB 侦听地址

Action: `gslb.dns_listener.del`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	侦听地址名称	是	必须存在

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.dns_listener.del

请求 body:

```
{
  "name": "192.168.108.53_va"
}
```

GSLB 服务

添加 GSLB 服务

Action: **gslb.service.add**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	唯一
ip	IP 地址		服务的 IPv4/IPv6 地址	是	
description	字符串	长度 0-191	描述信息	否	缺省值: 空
enabled	整数	0, 1	服务状态	否	1: 启用; 0: 禁用; 缺省值: 1
health_check	字符串	长度 0-50	健康检查名称	否	缺省值: 空
gmp	整数	0, 1	协议检查	否	1: 启用; 0: 禁用; 缺省值: 1
port_list	数组		由服务端口对象组成的数组	是	至少需要 1 个服务端口对象, 最多 65535 个
port	整数	0-65534	服务端口对象—服务端口	是	每服务唯一
protocol	整数	2-3	服务端口对象—服务协议	是	2: TCP; 3: UDP;
enabled	整数	0, 1	服务端口对象—服务端口状态	否	1: 启用; 0: 禁用; 缺省值: 1
gmp	整数	0, 1	服务端口对象—协议检查	否	1: 启用; 0: 禁用; 缺省值: 1
port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称	否	缺省值: 空

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.add
请求 body:

```
{
  "name": "service_test1",
  "ip": "100.20.30.1",
  "description": "desc of service_test1",
  "health_check": "http",
  "gmp": 1,
  "enabled": 1,
  "port_list": [
    {
      "port": 1,
      "protocol": 3,
      "port_health_check": "ping",
      "gmp": 1,
    }
  ]
}
```

```

        "enabled": 1
    },
    {
        "port": 65534,
        "protocol": 2,
        "port_health_check": "",
        "gmmp": 0,
        "enabled": 0
    }
]
}

```

获取 GSLB 服务列表

Action: `gslb.service.list`

请求参数:无

请求举例:

POST <http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.list>

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	服务名称
ip	IP 地址		服务的 IPv4/IPv6 地址
description	字符串	长度 0-191	描述信息
enabled	整数	0, 1	服务状态; 1:启用;0:禁用;
health_check	字符串	长度 0-50	健康检查名称
state	整数	3, 5	虚拟服务的状态; 3:故障;5:正常
gmmp	整数	0, 1	协议检查; 1:启用;0:禁用;
port_list	数组		由服务端口对象组成的数组
port	整数	0-65534	服务端口对象—服务端口
protocol	整数	2-3	服务端口对象—服务协议; 2:TCP;3:UDP
enabled	整数	0, 1	服务端口对象—服务端口状态; 1:启用;0:禁用;
gmmp	整数	0, 1	服务端口对象—协议检查; 1:启用;0:禁用;
port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称

响应举例:

```

[ {
    "name": "service_test1",
    "description": "desc of service_test1",
    "ip": "100.20.30.1",
    "health_check": "http",
    "gmmp": 1,
    "enabled": 1,
    "state": 3,

```

```

    "port_list": [
      {
        "port": 65534,
        "protocol": 2,
        "port_health_check": "",
        "gmmp": 0,
        "enabled": 0,
        "state": 3
      },
      {
        "port": 1,
        "protocol": 3,
        "port_health_check": "ping",
        "gmmp": 1,
        "enabled": 1,
        "state": 3
      }
    ]
  }
]

```

获取指定 GSLB 服务

Action: `gslb.service.get`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	服务名称	是	必须存在

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.get>

Body: {"name": "service_test1"}

响应参数:

名称	类型	范围	含义
<code>name</code>	字符串	长度 1-191	服务名称
<code>ip</code>	IP 地址		服务的 IPv4/IPv6 地址
<code>description</code>	字符串	长度 0-191	描述信息
<code>enabled</code>	整数	0, 1	服务状态; 1:启用;0:禁用;
<code>health_check</code>	字符串	长度 0-50	健康检查名称
<code>gmmp</code>	整数	0, 1	协议检查; 1:启用;0:禁用;
<code>state</code>	整数	3, 5	虚拟服务的状态; 3:故障;5:正常
<code>port_list</code>	数组		由服务端口对象组成的数组
<code>port</code>	整数	0-65534	服务端口对象—服务端口

protocol	整数	2-3	服务端口对象—服务协议; 2:TCP;3:UDP
enabled	整数	0, 1	服务端口对象—服务端口状态; 1:启用;0:禁用;
gmmp	整数	0, 1	服务端口对象—协议检查; 1:启用;0:禁用;
port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称

响应举例:

```
{
  "name": "service_test1",
  "description": "desc of service_test1",
  "ip": "100.20.30.1",
  "health_check": "http",
  "gmmp": 1,
  "enabled": 1,
  "state": 3,
  "port_list": [
    {
      "port": 65534,
      "protocol": 2,
      "port_health_check": "",
      "gmmp": 0,
      "enabled": 0,
      "state": 3
    },
    {
      "port": 1,
      "protocol": 3,
      "port_health_check": "ping",
      "gmmp": 1,
      "enabled": 1,
      "state": 3
    }
  ]
}
```

编辑指定 GSLB 服务

Action: `gslb.service.edit`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	唯一
ip	IP 地址		服务的 IPv4/IPv6 地址	是	必须与原来的相同

description	字符串	长度 0-191	描述信息	否	缺省值: 不修改
enabled	整数	0, 1	服务状态	否	1: 启用; 0: 禁用; 缺省值: 不修改
health_check	字符串	长度 0-50	健康检查名称	否	缺省值: 不修改
gmmp	整数	0, 1	协议检查	否	1: 启用; 0: 禁用; 缺省值: 1
port_list	数组		由服务端口对象组成的数组	是	至少需要 1 个服务端口对象, 最多 65535 个
port	整数	0-65534	服务端口对象—服务端口	是	每服务唯一
protocol	整数	2-3	服务端口对象—服务协议	是	2: TCP; 3: UDP;
enabled	整数	0, 1	服务端口对象—服务端口状态	否	1: 启用; 0: 禁用; 缺省值: 不修改
gmmp	整数	0, 1	服务端口对象—协议检查	否	1: 启用; 0: 禁用; 缺省值: 不修改
port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称	否	缺省值: 空

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gs1b.service.edit

请求 body:

```
{
  "name": "service_test1",
  "description": "desc of service_test1 modify",
  "ip": "100.20.30.1",
  "health_check": "tcp",
  "gmmp": 1,
  "enabled": 1,
  "port_list": [
    {
      "port": 1000,
      "protocol": 2,
      "port_health_check": "udp",
      "gmmp": 0,
      "enabled": 0
    },
    {
      "port": 2000,
      "protocol": 3,
      "port_health_check": "",
      "gmmp": 1,
      "enabled": 1
    }
  ]
}
```

添加服务端口

Action: **gslb.service.port.add**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	唯一
port	对象		服务端口对象	是	
port	整数	0-65534	服务端口对象—服务端口	是	每服务唯一
protocol	整数	2-3	服务端口对象—服务协议	是	2:TCP;3:UDP;
enabled	整数	0,1	服务端口对象—服务端口状态	否	1:启用;0:禁用;缺省值:1
gmmp	整数	0,1	服务端口对象—协议检查	否	1:启用;0:禁用;缺省值:1
port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称	否	缺省值:空

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.port.add

请求 body:

```
{
  "name": "service_test1",
  "port": {
    "port": 3000,
    "protocol": 3,
    "health_check": "ping",
    "gmmp": 1,
    "enabled": 1
  }
}
```

编辑服务端口

Action: **gslb.service.port.edit**

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	唯一
port	对象		服务端口对象	是	
port	整数	0-65534	服务端口对象—服务端口	是	每服务唯一
protocol	整数	2-3	服务端口对象—服务协议	是	2:TCP;3:UDP;
enabled	整数	0,1	服务端口对象—服务端口状态	否	1:启用;0:禁用;缺省值:不修改
gmmp	整数	0,1	服务端口对象—协议检查	否	1:启用;0:禁用;缺省值:不修改

port_health_check	字符串	长度 0-50	服务端口对象—健康检查名称	否	缺省值:不修改
-------------------	-----	---------	---------------	---	---------

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.port.edit

请求 body:

```
{
  "name": "service_test1",
  "port": {
    "port": 3000,
    "protocol": 2,
    "port_health_check": "udp",
    "gmmp": 1,
    "enabled": 1
  }
}
```

删除服务端口

Action: gslb.service.port.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	必须存在
port	对象		服务端口对象	是	
port	整数	0-65534	服务端口对象—服务端口	是	必须存在
protocol	整数	2-3	服务端口对象—服务协议	是	2:TCP;3:UDP;

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.port.del

请求 body:

```
{
  "name": "service_test1",
  "port": {
    "port": 65534,
    "protocol": 2
  }
}
```

删除 GSLB 服务

Action: `gslb.service.del`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	服务名称	是	必须存在

请求举例:

POST

`http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.service.del`

请求 body:

```
{"name": "service_test1"}
```

GSLB 站点

添加 GSLB 站点

Action: `gslb.site.add`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	站点名称	是	唯一
weight	整数	1-100	站点权重	否	缺省值:1
status	整数	0,1	站点状态	否	1:启用;0:禁用;缺省值:1
gslb_geography_info	字符串	长度 0-127	地理位置信息,		格式如 "liantong asia.china.shanxi.taiyuan",多个地理位置信息用 隔开,不同级别用.隔开 缺省值:空
site_member_list	数组		由站点成员对象组成的数组	否	缺省值:空
name	字符串	长度 1-191	站点成员对象—名称	是	唯一
ip	IPv4 地址		站点成员对象—站点成员 IP 地址	是	每站点唯一
preference	整数	0-255	站点成员对象—优先级	否	缺省值:100
gateway	IPv4 地址		站点成员对象—网关	否	缺省值:0.0.0.0
service_list	数组		站点成员对象—服务列表	否	缺省值:空
name	字符串	长度 1-191	站点成员服务—服务名称	是	

请求举例:

POST `http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.add`

请求 body:

```
{
```



```

"name": "site3",
"weight": 1,
"status": 1,
"gslb_geography_info": "liantong|asia.china.shanxi.taiyuan",
"site_member_list": [
  {
    "name": "sm1",
    "ip": "111.111.111.111",
    "preference": 1,
    "gateway": "0.0.0.0",
    "service_list": [
      {"name": "service1"}
    ]
  },
  {
    "name": "sm2",
    "ip": "222.222.222.222",
    "preference": 2,
    "gateway": "0.0.0.0",
    "service_list": [
      {"name": "service2"}
    ]
  }
]
}

```

获取 GSLB 站点列表

Action: gslb.site.list

请求参数:无

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	站点名称
weight	整数	1-100	站点权重
status	整数	0, 1	站点状态
gslb_geography_info	字符串	长度 0-127	地理位置信息,
site_member_list	数组		由站点成员对象组成的数组
name	字符串	长度 1-191	站点成员对象一名称
ip	IPv4 地址		站点成员对象一站点成员 IP 地址

preference	整数	0-255	站点成员对象—优先级
gateway	IPv4 地址		站点成员对象—网关
service_list	数组		站点成员对象—服务列表
name	字符串	长度 1-191	站点成员服务—服务名称

响应举例:

```
[
  {
    "name": "site3",
    "weight": 1,
    "status": 1,
    "gslb_geography_info": "liantong|asia.china.shanxi.taiyuan",
    "site_member_list": [
      {
        "name": "sm1",
        "ip": "111.111.111.111",
        "preference": 1,
        "gateway": "0.0.0.0",
        "service_list": [
          {
            "name": "service1"
          }
        ]
      },
      {
        "name": "sm2",
        "ip": "222.222.222.222",
        "preference": 2,
        "gateway": "0.0.0.0",
        "service_list": [
          {
            "name": "service2"
          }
        ]
      }
    ]
  }
]
```

获取 GSLB 指定站点

Action: gslb.site.get

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-191	站点名称	是	必须存在
------	-----	----------	------	---	------

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.get
Body: {"name": "site3"}

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	站点名称
weight	整数	1-100	站点权重
status	整数	0, 1	站点状态
gslb_geography_info	字符串	长度 0-127	地理位置信息,
site_member_list	数组		由站点成员对象组成的数组
name	字符串	长度 1-191	站点成员对象--名称
ip	IPv4 地址		站点成员对象--站点成员 IP 地址
preference	整数	0-255	站点成员对象--优先级
gateway	IPv4 地址		站点成员对象--网关
service_list	数组		站点成员对象--服务列表
name	字符串	长度 1-191	站点成员服务--服务名称

响应举例:

```
{
  "name": "site3",
  "weight": 1,
  "status": 1,
  "gslb_geography_info": "liantong|asia.china.shanxi.taiyuan",
  "site_member_list": [
    {
      "name": "sm1",
      "ip": "111.111.111.111",
      "preference": 1,
      "gateway": "0.0.0.0",
      "service_list": [
        {
          "name": "service1"
        }
      ]
    },
    {
      "name": "sm2",
      "ip": "222.222.222.222",
      "preference": 2,
      "gateway": "0.0.0.0",
      "service_list": [
        {
```

```

        "name": "service2"
    }
}
]
}
}

```

编辑指定 GSLB 站点

Action: `gslb.site.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	站点名称	是	必须存在
<code>weight</code>	整数	1-100	站点权重	否	缺省值: 不修改
<code>status</code>	整数	0, 1	站点状态	否	1: 启用; 0: 禁用; 缺省值: 不修改
<code>gslb_geography_info</code>	字符串	长度 0-127	地理位置信息,		格式如 "liantong asia.china.shanxi.taiyuan", 多个地理位置信息用 隔开, 不同级别用. 隔开 缺省值: 不修改
<code>site_member_list</code>	数组		由站点成员对象组成的数组	否	缺省值: 空
<code>name</code>	字符串	长度 1-191	站点成员对象—名称	是	唯一
<code>ip</code>	IPv4 地址		站点成员对象—站点成员 IP 地址	是	每站点唯一
<code>preference</code>	整数	0-255	站点成员对象—优先级	否	缺省值: 不修改
<code>gateway</code>	IPv4 地址		站点成员对象—网关	否	缺省值: 不修改
<code>service_list</code>	数组		站点成员对象—服务列表	否	缺省值: 不修改
<code>name</code>	字符串	长度 1-191	站点成员服务—服务名称	是	

请求举例:

POST `http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.edit`

请求 body:

```

{
    "name": "site3",
    "weight": 1,
    "status": 1,
    "gslb_geography_info": "liantong|asia.china.shanxi.taiyuan",
    "site_member_list": [
        {
            "name": "sml",
            "ip": "111.111.111.111",
            "preference": 1,

```

```

    "gateway": "0.0.0.0",
    "service_list": [
        {"name": "service1"}
    ]
},
{
    "name": "sm2",
    "ip": "222.222.222.222",
    "preference": 2,
    "gateway": "0.0.0.0",
    "service_list": [
        {"name": "service2"}
    ]
}
]
}

```

添加站点成员

Action: gslb.site.member.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	站点名称	是	必须存在
site_member	对象		站点成员对象	是	
name	字符串	长度 1-191	站点成员对象--名称	是	唯一
ip	IPv4 地址		站点成员对象--站点成员 IP 地址	是	每站点唯一
preference	整数	0-255	站点成员对象--优先级	否	缺省值:100
gateway	IPv4 地址		站点成员对象--网关	否	缺省值:0.0.0.0
service_list	数组		站点成员对象--服务列表	否	缺省值:空
name	字符串	长度 1-191	站点成员服务--服务名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.member.add>

请求 body:

```

{
    "name": "site3",
    "site_member": {
        "name": "sm1",
        "ip": "111.111.111.111",
        "preference": 1,

```

```

    "gateway": "0.0.0.0",
    "service_list": [
      {"name": "service1"}
    ]
  }
}

```

编辑站点成员

Action: `gslb.site.member.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	站点名称	是	必须存在
<code>site_member</code>	对象		站点成员对象	是	
<code>name</code>	字符串	长度 1-191	站点成员对象--名称	是	必须存在
<code>ip</code>	IPv4 地址		站点成员对象--站点成员 IP 地址	是	
<code>preference</code>	整数	0-255	站点成员对象--优先级	否	缺省值:不修改
<code>gateway</code>	IPv4 地址		站点成员对象--网关	否	缺省值:不修改
<code>service_list</code>	数组		站点成员对象--服务列表	否	缺省值:不修改
<code>name</code>	字符串	长度 1-191	站点成员服务--服务名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.member.edit>

请求 body:

```

{
  "name": "site3",
  "site_member": {
    "name": "sml",
    "ip": "111.111.111.111",
    "preference": 1,
    "gateway": "0.0.0.0",
    "service_list": [
      {"name": "service1"}
    ]
  }
}

```

删除站点成员

Action: `gslb.site.member.del`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	站点名称	是	必须存在
<code>site_member</code>	对象		站点成员对象	是	
<code>name</code>	字符串	长度 1-191	站点成员对象--名称	是	必须存在
<code>ip</code>	IPv4 地址		站点成员对象--站点成员 IP 地址	是	必须存在

请求举例:

POST
`http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.member.del`

请求 body:

```
{
  "name": "site3",
  "site_member": {
    "name": "sml",
    "ip": "111.111.111.111"
  }
}
```

删除 GSLB 站点

Action: `gslb.site.del`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	站点名称	是	必须存在

请求举例:

POST `http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.site.del`

请求 body:

```
{"name": "site3"}
```

GSLB 策略

添加 GSLB 策略

Action: `gslb.policy.add`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略名称	是	唯一
metric	对象		策略算法, 包含各种算法	否	
session_usage	对象		session_usage(连接数使用量)算法	否	
seq_num	整数	0-17	session_usage-算法顺序(越小越优先)	否	缺省值:4
enabled	整数	0-1	session_usage-算法使能:1:是;0:否;	否	缺省值:0
threshold	整数	0-100	session_usage-连接数百分比, 单位%	否	缺省值:0
rtt-active	对象		rtt-active(最快时间)算法	否	
seq_num	整数	0-17	rtt-active-算法顺序(越小越优先)	否	缺省值:7
enabled	整数	0-1	rtt-active-算法使能:1:是;0:否;	否	缺省值:0
samples	整数	1-8	rtt-active-样本个数	否	缺省值:5
tolerance	整数	0-100	rtt-active-差异度,	否	缺省值:10
available_connection	对象		available_connection(有效连接数)算法	否	
seq_num	整数	0-17	available_connection-算法顺序(越小越优先)	否	缺省值:8
enabled	整数	0-1	available_connection-算法使能:1:是;0:否;	否	缺省值:0
limit	整数	0-999999999	available_connection-新建连接上限;0表示不限制	否	缺省值:0
samples_num	整数	1-8	available_connection-样本数量	否	缺省值:5
samples_interval	整数	1-60	available_connection-采样间隔, 单位秒	否	缺省值:5
available_sessions	对象		available_sessions(会话)算法	否	
seq_num	整数	0-17	available_sessions-算法顺序(越小越优先)	否	缺省值:9
enabled	整数	0-1	available_sessions-算法使能:1:是;0:否;	否	缺省值:0
tolerance	整数	0-100	available_sessions-差异度,	否	缺省值:0
geography	对象		geography(地理位置)算法	否	
seq_num	整数	0-17	geography-算法顺序(越小越优先)	否	缺省值:1
enabled	整数	0-1	geography-算法使能:1:是;0:否;	否	缺省值:1
ip_order	对象		ip_order(IP顺序)算法	否	
seq_num	整数	0-17	ip_order-算法顺序(越小越优先)	否	缺省值:13
enabled	整数	0-1	ip_order-算法使能:1:是;0:否;	否	缺省值:0
site_weight	对象		site_weight(站点权重)算法	否	
seq_num	整数	0-17	site_weight-算法顺序(越小越优先)	否	缺省值:3

enabled	整数	0-1	site_weight-算法使能:1:是;0:否;	否	缺省值:0
available_nodes	对象		available_nodes(有效节点数量)算法	否	
seq_num	整数	0-17	available_nodes-算法顺序(越小越优先)	否	缺省值:5
enabled	整数	0-1	available_nodes-算法使能:1:是;0:否;	否	缺省值:0
service_weight	对象		service_weight(服务权重)算法	否	
seq_num	整数	0-17	service_weight-算法顺序(越小越优先)	否	缺省值:2
enabled	整数	0-1	service_weight-算法使能:1:是;0:否;	否	缺省值:0
bandwidth	对象		bandwidth(带宽)算法	否	
seq_num	整数	0-17	bandwidth-算法顺序(越小越优先)	否	缺省值:11
enabled	整数	0-1	bandwidth-算法使能:1:是;0:否;	否	缺省值:0
health_check	对象		health_check(健康检查)算法	否	
seq_num	整数	0-17	health_check-算法顺序(越小越优先)	否	缺省值:0
enabled	整数	0-1	health_check-算法使能:1:是;0:否;	否	缺省值:1
preference	对象		preference(优先级)算法	否	
seq_num	整数	0-17	preference-算法顺序(越小越优先)	否	缺省值:10
enabled	整数	0-1	preference-算法使能:1:是;0:否;	否	缺省值:0
least_hits	对象		least_hits(最少命中)算法	否	
seq_num	整数	0-17	least_hits-算法顺序(越小越优先)	否	缺省值:12
enabled	整数	0-1	least_hits-算法使能:1:是;0:否;	否	缺省值:0
round_robin	对象		round_robin(轮循)算法	否	
seq_num	整数	0-17	round_robin-算法顺序(越小越优先)-轮循算法固定为17	否	缺省值:17
enabled	整数	0-1	round_robin-算法使能:1:是;0:否;	否	缺省值:1
dns_options	对象		DNS 选项	否	
action	整数	0-1	是否执行域服务的动作:1:是;0:否;	是	
active_address	整数	0-1	激活地址:1:开启;0:关闭;	否	缺省值:0
best_addresses	整数	0-1	最佳地址:1:开启;0:关闭;	否	缺省值:0
cache_mode	对象		缓存模式	否	
enabled	整数	0-1	缓存模式-是否开启:1:是;0:否;	否	缺省值:0

dns_cache_aging_time	整数	0-100000000	缓存模式-缓存老化时间, 单位秒	否	缺省值:0
cname-record	整数	0-1	提供 CNAME 记录: 1:是;0:否;	否	缺省值:0
outside-ip	整数	0-1	提供外部地址: 1:是;0:否;	否	缺省值:0
address_replace	整数	0-1	地址替换: 1:是;0:否;	否	缺省值:0
geography-alias	整数	0-1	提供别名记录: 1:是;0:否;	否	缺省值:0
additional_mx	整数	0-1	提供附加 MX: 1:是;0:否;	否	缺省值:0
server_mode	对象		服务器模式	否	
enabled	整数	0-1	服务器模式-是否开启:1:是;0:否;	否	缺省值:0
author	整数	0-1	服务器模式-作为授权服务器:1:是;0:否;	否	缺省值:0
all-a-records	整数	0-1	服务器模式-提供所有 A 记录:1:是;0:否;	否	缺省值:0
mx_record	整数	0-1	服务器模式-提供 MX 记录:1:是;0:否;	否	缺省值:0
additional-mx	整数	0-1	服务器模式-附加 MX:1:是;0:否;	否	缺省值:0
ns_record	整数	0-1	服务器模式-提供 NS 记录:1:是;0:否;	否	缺省值:0
ns_record_auto	整数	0-1	服务器模式-转换 NS 记录为 A 记录:1:是;0:否;	否	缺省值:0
ptr_record	整数	0-1	服务器模式-提供 PTR 记录:1:是;0:否;	否	缺省值:0
ptr_record_auto	整数	0-1	服务器模式-转换 PTR 记录为 A 记录:1:是;0:否;	否	缺省值:0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.policy.add

请求 body:

```
{"name": "policy1"}
```



- 1.各算法的算法顺序(seq_num)在后台会重新计算取值
- 2.轮循算法(round_robin)的算法顺序(seq_num)固定为 17

获取 GSLB 策略列表

Action: gslb.policy.list

请求参数:无

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.policy.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	策略名称
metric	对象		策略算法, 包含各种算法
session_usage	对象		session_usage(连接数使用量)算法
seq_num	整数	0-17	session_usage-算法顺序(越小越优先)
enabled	整数	0-1	session_usage-算法使能:1:是;0:否;
threshold	整数	0-100	session_usage-连接数百分比, 单位%
rtt-active	对象		rtt-active(最快时间)算法
seq_num	整数	0-17	rtt-active-算法顺序(越小越优先)
enabled	整数	0-1	rtt-active-算法使能:1:是;0:否;
samples	整数	1-8	rtt-active-样本个数
tolerance	整数	0-100	rtt-active-差异度,
available_connection	对象		available_connection(有效连接数)算法
seq_num	整数	0-17	available_connection-算法顺序(越小越优先)
enabled	整数	0-1	available_connection-算法使能:1:是;0:否;
limit	整数	0-999999999	available_connection-新建连接上限;0 表示不限制
samples_num	整数	1-8	available_connection-样本数量
samples_interval	整数	1-60	available_connection-采样间隔, 单位秒
available_sessions	对象		available_sessions(会话)算法
seq_num	整数	0-17	available_sessions-算法顺序(越小越优先)
enabled	整数	0-1	available_sessions-算法使能:1:是;0:否;
tolerance	整数	0-100	available_sessions-差异度,
geography	对象		geography(地理位置)算法
seq_num	整数	0-17	geography-算法顺序(越小越优先)
enabled	整数	0-1	geography-算法使能:1:是;0:否;
ip_order	对象		ip_order(IP 顺序)算法
seq_num	整数	0-17	ip_order-算法顺序(越小越优先)
enabled	整数	0-1	ip_order-算法使能:1:是;0:否;
site_weight	对象		site_weight(站点权重)算法
seq_num	整数	0-17	site_weight-算法顺序(越小越优先)
enabled	整数	0-1	site_weight-算法使能:1:是;0:否;
available_nodes	对象		available_nodes(有效节点数量)算法
seq_num	整数	0-17	available_nodes-算法顺序(越小越优先)
enabled	整数	0-1	available_nodes-算法使能:1:是;0:否;
service_weight	对象		service_weight(服务权重)算法
seq_num	整数	0-17	service_weight-算法顺序(越小越优先)
enabled	整数	0-1	service_weight-算法使能:1:是;0:否;
bandwidth	对象		bandwidth(带宽)算法
seq_num	整数	0-17	bandwidth-算法顺序(越小越优先)
enabled	整数	0-1	bandwidth-算法使能:1:是;0:否;

health_check	对象		health_check(健康检查)算法
seq_num	整数	0-17	health_check-算法顺序(越小越优先)
enabled	整数	0-1	health_check-算法使能:1:是;0:否;
preference	对象		preference(优先级)算法
seq_num	整数	0-17	preference-算法顺序(越小越优先)
enabled	整数	0-1	preference-算法使能:1:是;0:否;
least_hits	对象		least_hits(最少命中)算法
seq_num	整数	0-17	least_hits-算法顺序(越小越优先)
enabled	整数	0-1	least_hits-算法使能:1:是;0:否;
round_robin	对象		round_robin(轮循)算法
seq_num	整数	0-17	round_robin-算法顺序(越小越优先)-轮循算法固定为 17
enabled	整数	0-1	round_robin-算法使能:1:是;0:否;
dns_option	对象		DNS 选项
action	整数	0-1	是否执行域服务的动作:1:是;0:否;
active_address	整数	0-1	激活地址: 1:开启;0:关闭;
best_address	整数	0-1	最佳地址: 1:开启;0:关闭;
cache_mode	对象		缓存模式
enabled	整数	0-1	缓存模式-是否开启:1:是;0:否;
dns_cache_aging_time	整数	0-1000000000	缓存模式-缓存老化时间, 单位秒
cname-record	整数	0-1	提供 CNAME 记录: 1:是;0:否;
outside-ip	整数	0-1	提供外部地址: 1:是;0:否;
address_replace	整数	0-1	地址替换: 1:是;0:否;
geography-alias	整数	0-1	提供别名记录: 1:是;0:否;
additional_mx	整数	0-1	提供附加 MX: 1:是;0:否;
server_mode	对象		服务器模式
enabled	整数	0-1	服务器模式-是否开启:1:是;0:否;
author	整数	0-1	服务器模式-作为授权服务器:1:是;0:否;
all-a-records	整数	0-1	服务器模式-提供所有 A 记录:1:是;0:否;
mx_record	整数	0-1	服务器模式-提供 MX 记录:1:是;0:否;
additional-mx	整数	0-1	服务器模式-附加 MX:1:是;0:否;
ns_record	整数	0-1	服务器模式-提供 NS 记录:1:是;0:否;
ns_record_auto	整数	0-1	服务器模式-转换 NS 记录为 A 记录:1:是;0:否;
ptr_record	整数	0-1	服务器模式-提供 PTR 记录:1:是;0:否;
ptr_record_auto	整数	0-1	服务器模式-转换 PTR 记录为 A 记录:1:是;0:否;

响应举例:

```
[
  {
    "name": "default",
    "metric": {
      "session_usage": {
        "seq_num": 3,
        "enabled": 0,
        "threshold": 90
      }
    }
  }
]
```

```

},
"rtt-active": {
  "seq_num": 6,
  "enabled": 0,
  "samples": 5,
  "tolerance": 10
},
"available_connection": {
  "seq_num": 8,
  "enabled": 0,
  "limit": 0,
  "samples_num": 5,
  "samples_interval": 5
},
"available_sessions": {
  "seq_num": 9,
  "enabled": 0,
  "tolerance": 10
},
"geography": {
  "seq_num": 7,
  "enabled": 1
},
"ip_order": {
  "seq_num": 16,
  "enabled": 0
},
"site_weight": {
  "seq_num": 2,
  "enabled": 0
},
"available_nodes": {
  "seq_num": 4,
  "enabled": 0
},
"service_weight": {
  "seq_num": 1,
  "enabled": 0
},
"bandwidth": {
  "seq_num": 13,
  "enabled": 0
},
"health_check": {
  "seq_num": 0,
  "enabled": 1
}

```

```

    },
    "preference": {
        "seq_num": 11,
        "enabled": 0
    },
    "least_hits": {
        "seq_num": 15,
        "enabled": 0
    },
    "round_robin": {
        "seq_num": 17,
        "enabled": 1
    }
},
"dns_options": {
    "action": 0,
    "active_address": 0,
    "best_address": 0,
    "cache_mode": {
        "enabled": 0,
        "dns_cache_aging_time": 0
    },
    "cname-record": 1,
    "outside-ip": 1,
    "address_replace": 0,
    "geography-alias": 0,
    "additional_mx": 0,
    "server_mode": {
        "enabled": 0,
        "author": 0,
        "all-a-records": 0,
        "mx_record": 0,
        "additional-mx": 0,
        "ns_record": 0,
        "ns_record_auto": 0,
        "ptr_record": 0,
        "ptr_record_auto": 0
    }
},
{
    "name": "policy1",
    "metric": {
        "session_usage": {
            "seq_num": 4,
            "enabled": 1,

```

```

    "threshold": 100
  },
  "rtt-active": {
    "seq_num": 7,
    "enabled": 1,
    "samples": 8,
    "tolerance": 100
  },
  "available_connection": {
    "seq_num": 8,
    "enabled": 0,
    "limit": 999999999,
    "samples_num": 8,
    "samples_interval": 60
  },
  "available_sessions": {
    "seq_num": 9,
    "enabled": 1,
    "tolerance": 100
  },
  "geography": {
    "seq_num": 1,
    "enabled": 1
  },
  "ip_order": {
    "seq_num": 13,
    "enabled": 1
  },
  "site_weight": {
    "seq_num": 3,
    "enabled": 1
  },
  "available_nodes": {
    "seq_num": 5,
    "enabled": 1
  },
  "service_weight": {
    "seq_num": 2,
    "enabled": 1
  },
  "bandwidth": {
    "seq_num": 11,
    "enabled": 1
  },
  "health_check": {
    "seq_num": 0,

```

```

        "enabled": 1
    },
    "preference": {
        "seq_num": 10,
        "enabled": 1
    },
    "least_hits": {
        "seq_num": 12,
        "enabled": 1
    },
    "round_robin": {
        "seq_num": 17,
        "enabled": 1
    }
},
"dns_options": {
    "action": 1,
    "active_address": 1,
    "best_address": 1,
    "cache_mode": {
        "enabled": 1,
        "dns_cache_aging_time": 1000000000
    },
    "cname-record": 1,
    "outside-ip": 1,
    "address_replace": 1,
    "geography-alias": 1,
    "additional_mx": 1,
    "server_mode": {
        "enabled": 1,
        "author": 1,
        "all-a-records": 1,
        "mx_record": 1,
        "additional-mx": 1,
        "ns_record": 1,
        "ns_record_auto": 1,
        "ptr_record": 1,
        "ptr_record_auto": 1
    }
}
}
]

```


获取 GSLB 指定策略

Action: `gslb.policy.get`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略名称	是	必须存在

请求举例:

POST `http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.policy.get`

Body:

```
{"name": "policy1"}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-191	策略名称
metric	对象		策略算法, 包含各种算法
session_usage	对象		session_usage (连接数使用量) 算法
seq_num	整数	0-17	session_usage-算法顺序(越小越优先)
enabled	整数	0-1	session_usage-算法使能:1:是;0:否;
threshold	整数	0-100	session_usage-连接数百分比, 单位%
rtt-active	对象		rtt-active (最快时间) 算法
seq_num	整数	0-17	rtt-active-算法顺序(越小越优先)
enabled	整数	0-1	rtt-active-算法使能:1:是;0:否;
samples	整数	1-8	rtt-active-样本个数
tolerance	整数	0-100	rtt-active-差异度,
available_connection	对象		available_connection (有效连接数) 算法
seq_num	整数	0-17	available_connection-算法顺序(越小越优先)
enabled	整数	0-1	available_connection-算法使能:1:是;0:否;
limit	整数	0-999999999	available_connection-新建连接上限;0 表示不限制
samples_num	整数	1-8	available_connection-样本数量
samples_interval	整数	1-60	available_connection-采样间隔, 单位秒
available_sessions	对象		available_sessions (会话) 算法
seq_num	整数	0-17	available_sessions-算法顺序(越小越优先)
enabled	整数	0-1	available_sessions-算法使能:1:是;0:否;
tolerance	整数	0-100	available_sessions-差异度,
geography	对象		geography (地理位置) 算法
seq_num	整数	0-17	geography-算法顺序(越小越优先)
enabled	整数	0-1	geography-算法使能:1:是;0:否;
ip_order	对象		ip_order (IP 顺序) 算法
seq_num	整数	0-17	ip_order-算法顺序(越小越优先)
enabled	整数	0-1	ip_order-算法使能:1:是;0:否;
site_weight	对象		site_weight (站点权重) 算法

seq_num	整数	0-17	site_weight-算法顺序(越小越优先)
enabled	整数	0-1	site_weight-算法使能:1:是;0:否;
available_nodes	对象		available_nodes(有效节点数量)算法
seq_num	整数	0-17	available_nodes-算法顺序(越小越优先)
enabled	整数	0-1	available_nodes-算法使能:1:是;0:否;
service_weight	对象		service_weight(服务权重)算法
seq_num	整数	0-17	service_weight-算法顺序(越小越优先)
enabled	整数	0-1	service_weight-算法使能:1:是;0:否;
bandwidth	对象		bandwidth(带宽)算法
seq_num	整数	0-17	bandwidth-算法顺序(越小越优先)
enabled	整数	0-1	bandwidth-算法使能:1:是;0:否;
health_check	对象		health_check(健康检查)算法
seq_num	整数	0-17	health_check-算法顺序(越小越优先)
enabled	整数	0-1	health_check-算法使能:1:是;0:否;
preference	对象		preference(优先级)算法
seq_num	整数	0-17	preference-算法顺序(越小越优先)
enabled	整数	0-1	preference-算法使能:1:是;0:否;
least_hits	对象		least_hits(最少命中)算法
seq_num	整数	0-17	least_hits-算法顺序(越小越优先)
enabled	整数	0-1	least_hits-算法使能:1:是;0:否;
round_robin	对象		round_robin(轮循)算法
seq_num	整数	0-17	round_robin-算法顺序(越小越优先)-轮循算法固定为 17
enabled	整数	0-1	round_robin-算法使能:1:是;0:否;
dns_option	对象		DNS 选项
action	整数	0-1	是否执行域服务的动作:1:是;0:否;
active_address	整数	0-1	激活地址: 1:开启;0:关闭;
best_address	整数	0-1	最佳地址: 1:开启;0:关闭;
cache_mode	对象		缓存模式
enabled	整数	0-1	缓存模式-是否开启:1:是;0:否;
dns_cache_aging_time	整数	0-1000000000	缓存模式-缓存老化时间, 单位秒
cname-record	整数	0-1	提供 CNAME 记录: 1:是;0:否;
outside-ip	整数	0-1	提供外部地址: 1:是;0:否;
address_replace	整数	0-1	地址替换: 1:是;0:否;
geography-alias	整数	0-1	提供别名记录: 1:是;0:否;
additional_mx	整数	0-1	提供附加 MX: 1:是;0:否;
server_mode	对象		服务器模式
enabled	整数	0-1	服务器模式-是否开启:1:是;0:否;
author	整数	0-1	服务器模式-作为授权服务器:1:是;0:否;
all-a-records	整数	0-1	服务器模式-提供所有 A 记录:1:是;0:否;
mx_record	整数	0-1	服务器模式-提供 MX 记录:1:是;0:否;
additional-mx	整数	0-1	服务器模式-附加 MX:1:是;0:否;
ns_record	整数	0-1	服务器模式-提供 NS 记录:1:是;0:否;
ns_record_auto	整数	0-1	服务器模式-转换 NS 记录为 A 记录:1:是;0:否;

ptr_record	整数	0-1	服务器模式-提供 PTR 记录:1:是;0:否;
ptr_record_auto	整数	0-1	服务器模式-转换 PTR 记录为 A 记录:1:是;0:否;

响应举例:

```
{
  "name": "policy1",
  "metric": {
    "session_usage": {
      "seq_num": 3,
      "enabled": 0,
      "threshold": 90
    },
    "rtt-active": {
      "seq_num": 6,
      "enabled": 0,
      "samples": 5,
      "tolerance": 10
    },
    "available_connection": {
      "seq_num": 8,
      "enabled": 0,
      "limit": 0,
      "samples_num": 5,
      "samples_interval": 5
    },
    "available_sessions": {
      "seq_num": 9,
      "enabled": 0,
      "tolerance": 10
    },
    "geography": {
      "seq_num": 7,
      "enabled": 1
    },
    "ip_order": {
      "seq_num": 16,
      "enabled": 0
    },
    "site_weight": {
      "seq_num": 2,
      "enabled": 0
    },
    "available_nodes": {
      "seq_num": 4,
      "enabled": 0
    }
  }
}
```

```

},
"service_weight": {
    "seq_num": 1,
    "enabled": 0
},
"bandwidth": {
    "seq_num": 13,
    "enabled": 0
},
"health_check": {
    "seq_num": 0,
    "enabled": 1
},
"preference": {
    "seq_num": 11,
    "enabled": 0
},
"least_hits": {
    "seq_num": 15,
    "enabled": 0
},
"round_robin": {
    "seq_num": 17,
    "enabled": 1
}
},
"dns_options": {
    "action": 0,
    "active_address": 0,
    "best_address": 0,
    "cache_mode": {
        "enabled": 0,
        "dns_cache_aging_time": 0
    },
    "cname-record": 1,
    "outside-ip": 1,
    "address_replace": 0,
    "geography-alias": 0,
    "additional_mx": 0,
    "server_mode": {
        "enabled": 0,
        "author": 0,
        "all-a-records": 0,
        "mx_record": 0,
        "additional-mx": 0,
        "ns_record": 0,

```

```

    "ns_record_auto": 0,
    "ptr_record": 0,
    "ptr_record_auto": 0
  }
}
}

```

编辑指定 GSLB 策略

Action: `gslb.policy.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-191	策略名称	是	必须存在
<code>metric</code>	对象		策略算法, 包含各种算法	否	缺省值: 不修改
<code>session_usage</code>	对象		<code>session_usage</code> (连接数使用量) 算法	否	缺省值: 不修改
<code>seq_num</code>	整数	0-17	<code>session_usage</code> -算法顺序 (越小越优先)	否	缺省值: 不修改
<code>enabled</code>	整数	0-1	<code>session_usage</code> -算法使能: 1: 是; 0: 否;	否	缺省值: 不修改
<code>threshold</code>	整数	0-100	<code>session_usage</code> -连接数百分比, 单位%	否	缺省值: 不修改
<code>rtt-active</code>	对象		<code>rtt-active</code> (最快时间) 算法	否	缺省值: 不修改
<code>seq_num</code>	整数	0-17	<code>rtt-active</code> -算法顺序 (越小越优先)	否	缺省值: 不修改
<code>enabled</code>	整数	0-1	<code>rtt-active</code> -算法使能: 1: 是; 0: 否;	否	缺省值: 不修改
<code>samples</code>	整数	1-8	<code>rtt-active</code> -样本个数	否	缺省值: 不修改
<code>tolerance</code>	整数	0-100	<code>rtt-active</code> -差异度,	否	缺省值: 不修改
<code>available_connection</code>	对象		<code>available_connection</code> (有效连接数) 算法	否	缺省值: 不修改
<code>seq_num</code>	整数	0-17	<code>available_connection</code> -算法顺序 (越小越优先)	否	缺省值: 不修改
<code>enabled</code>	整数	0-1	<code>available_connection</code> -算法使能: 1: 是; 0: 否;	否	缺省值: 不修改
<code>limit</code>	整数	0-999999999	<code>available_connection</code> -新建连接上限; 0 表示不限制	否	缺省值: 不修改
<code>samples_num</code>	整数	1-8	<code>available_connection</code> -样本数量	否	缺省值: 不修改
<code>samples_interval</code>	整数	1-60	<code>available_connection</code> -采样间隔, 单位秒	否	缺省值: 不修改
<code>available_sessions</code>	对象		<code>available_sessions</code> (会话) 算法	否	缺省值: 不修改
<code>seq_num</code>	整数	0-17	<code>available_sessions</code> -算法顺序 (越小越优先)	否	缺省值: 不修改

enabled	整数	0-1	available_sessions-算法使能:1:是;0:否;	否	缺省值:不修改
tolerance	整数	0-100	available_sessions-差异度,	否	缺省值:不修改
geography	对象		geography(地理位置)算法	否	缺省值:不修改
seq_num	整数	0-17	geography-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	geography-算法使能:1:是;0:否;	否	缺省值:不修改
ip_order	对象		ip_order(IP顺序)算法	否	缺省值:不修改
seq_num	整数	0-17	ip_order-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	ip_order-算法使能:1:是;0:否;	否	缺省值:不修改
site_weight	对象		site_weight(站点权重)算法	否	缺省值:不修改
seq_num	整数	0-17	site_weight-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	site_weight-算法使能:1:是;0:否;	否	缺省值:不修改
available_nodes	对象		available_nodes(有效节点数量)算法	否	缺省值:不修改
seq_num	整数	0-17	available_nodes-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	available_nodes-算法使能:1:是;0:否;	否	缺省值:不修改
service_weight	对象		service_weight(服务权重)算法	否	缺省值:不修改
seq_num	整数	0-17	service_weight-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	service_weight-算法使能:1:是;0:否;	否	缺省值:不修改
bandwidth	对象		bandwidth(带宽)算法	否	缺省值:不修改
seq_num	整数	0-17	bandwidth-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	bandwidth-算法使能:1:是;0:否;	否	缺省值:不修改
health_check	对象		health_check(健康检查)算法	否	缺省值:不修改
seq_num	整数	0-17	health_check-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	health_check-算法使能:1:是;0:否;	否	缺省值:不修改
preference	对象		preference(优先级)算法	否	缺省值:不修改
seq_num	整数	0-17	preference-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	preference-算法使能:1:是;0:否;	否	缺省值:不修改
least_hits	对象		least_hits(最少命中)算法	否	缺省值:不修改
seq_num	整数	0-17	least_hits-算法顺序(越小越优先)	否	缺省值:不修改
enabled	整数	0-1	least_hits-算法使能:1:是;0:否;	否	缺省值:不修改
round_robin	对象		round_robin(轮循)算法	否	缺省值:不修改

seq_num	整数	0-17	round_robin-算法顺序(越小越优先)-轮循算法固定为17	否	缺省值:不修改
enabled	整数	0-1	round_robin-算法使能:1:是;0:否;	否	缺省值:不修改
dns_options	对象		DNS 选项	否	缺省值:不修改
action	整数	0-1	是否执行域服务的动作:1:是;0:否;	是	
active_address	整数	0-1	激活地址:1:开启;0:关闭;	否	缺省值:不修改
best_addresses	整数	0-1	最佳地址:1:开启;0:关闭;	否	缺省值:不修改
cache_mode	对象		缓存模式	否	缺省值:不修改
enabled	整数	0-1	缓存模式-是否开启:1:是;0:否;	否	缺省值:不修改
dns_cache_aging_time	整数	0-100000000	缓存模式-缓存老化时间,单位秒	否	缺省值:不修改
cname-record	整数	0-1	提供 CNAME 记录:1:是;0:否;	否	缺省值:不修改
outside-ip	整数	0-1	提供外部地址:1:是;0:否;	否	缺省值:不修改
address_replace	整数	0-1	地址替换:1:是;0:否;	否	缺省值:不修改
geography-alias	整数	0-1	提供别名记录:1:是;0:否;	否	缺省值:不修改
additional_mx	整数	0-1	提供附加 MX:1:是;0:否;	否	缺省值:不修改
server_mode	对象		服务器模式	否	缺省值:不修改
enabled	整数	0-1	服务器模式-是否开启:1:是;0:否;	否	缺省值:不修改
author	整数	0-1	服务器模式-作为授权服务器:1:是;0:否;	否	缺省值:不修改
all-a-records	整数	0-1	服务器模式-提供所有 A 记录:1:是;0:否;	否	缺省值:不修改
mx_record	整数	0-1	服务器模式-提供 MX 记录:1:是;0:否;	否	缺省值:不修改
additional-mx	整数	0-1	服务器模式-附加 MX:1:是;0:否;	否	缺省值:不修改
ns_record	整数	0-1	服务器模式-提供 NS 记录:1:是;0:否;	否	缺省值:不修改
ns_record_auto	整数	0-1	服务器模式-转换 NS 记录为 A 记录:1:是;0:否;	否	缺省值:不修改
ptr_record	整数	0-1	服务器模式-提供 PTR 记录:1:是;0:否;	否	缺省值:不修改
ptr_record_auto	整数	0-1	服务器模式-转换 PTR 记录为 A 记录:1:是;0:否;	否	缺省值:不修改

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.policy.edit

请求 body:

```
{
  "name": "policy1",
  "metric": {
    "session_usage": {
      "seq_num": 4,
      "enabled": 1,
      "threshold": 100
    },
    "rtt-active": {
      "seq_num": 7,
      "enabled": 1,
      "samples": 8,
      "tolerance": 100
    },
    "available_connection": {
      "seq_num": 8,
      "enabled": 0,
      "limit": 999999999,
      "samples_num": 8,
      "samples_interval": 60
    },
    "available_sessions": {
      "seq_num": 9,
      "enabled": 1,
      "tolerance": 100
    },
    "geography": {
      "seq_num": 1,
      "enabled": 1
    },
    "ip_order": {
      "seq_num": 13,
      "enabled": 1
    },
    "site_weight": {
      "seq_num": 3,
      "enabled": 1
    },
    "available_nodes": {
      "seq_num": 5,
      "enabled": 1
    },
    "service_weight": {
```



```

        "seq_num": 2,
        "enabled": 1
    },
    "bandwidth": {
        "seq_num": 11,
        "enabled": 1
    },
    "health_check": {
        "seq_num": 0,
        "enabled": 1
    },
    "preference": {
        "seq_num": 10,
        "enabled": 1
    },
    "least_hits": {
        "seq_num": 12,
        "enabled": 1
    },
    "round_robin": {
        "seq_num": 17,
        "enabled": 1
    }
},
"dns_options": {
    "action": 1,
    "active_address": 1,
    "best_address": 1,
    "cache_mode": {
        "enabled": 1,
        "dns_cache_aging_time": 1000000000
    },
    "cname-record": 1,
    "outside-ip": 1,
    "address_replace": 1,
    "geography-alias": 1,
    "additional_mx": 1,
    "server_mode": {
        "enabled": 1,
        "author": 1,
        "all-a-records": 1,
        "mx_record": 1,
        "additional-mx": 1,
        "ns_record": 1,
        "ns_record_auto": 1,
        "ptr_record": 1,

```

```

    "ptr_record_auto": 1
  }
}
}

```



- 1.各算法的算法顺序(seq_num)在后台会重新计算取值
- 2.轮循算法(round_robin)的算法顺序(seq_num)固定为 17

删除 GSLB 策略

Action: gslb.policy.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	策略名称	是	必须存在

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.policy.del
 请求 body:
 {"name": "policy1"}

GSLB 域名

添加 GSLB 域名

Action: gslb.domain.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-127	域名名称, 例如 www.baidu.com 的域名名称应该为 baidu.com	是	唯一
ttd	整数	0-100000000	TTL	否	缺省值:10
policy	字符串	长度 1-191	域名使用的 GSLB 策略名称, 必须存在	否	缺省值:default
status	整数	0-1	域名状态;1:启用;0:禁用	否	缺省值:0
soastatus	整数	0-1	主 SOA 状态;1:启用;0:禁用	否	缺省值:0
soans	字符串	长度 1-127	主 SOA 服务器	否	缺省值:n/a
soamaibox	字符串	长度 1-127	主 SOA 负责人	否	缺省值:n/a
soaserial	整数	0-2147483647	主 SOA 序列号	否	缺省值:101010
soarefresh	整数	0-2147483647	主 SOA 刷新闻隔, 单位秒	否	缺省值:3600

		47			
soaretry	整数	0-21474836 47	主 SOA 重试间隔	否	缺省值:900
soaexpire	整数	0-21474836 47	主 SOA 过期间隔	否	缺省值:1209600
soattl	整数	0-21474836 47	主 SOA 最小 TTL	否	缺省值:1209600
soastatus2	整数	0-1	备 SOA 状态;1:启用;0:禁用	否	缺省值:0
soans2	字符串	长度 1-127	备 SOA 服务器	否	缺省值:n/a
soamaibox2	字符串	长度 1-127	备 SOA 负责人	否	缺省值:n/a
soaserial2	整数	0-21474836 47	备 SOA 序列号	否	缺省值:101010
soarefresh2	整数	0-21474836 47	备 SOA 刷新闻隔, 单位秒	否	缺省值:3600
soaretry2	整数	0-21474836 47	备 SOA 重试间隔	否	缺省值:900
soaexpire2	整数	0-21474836 47	备 SOA 过期间隔	否	缺省值:1209600
soattl2	整数	0-21474836 47	备 SOA 最小 TTL	否	缺省值:1209600
domain_reco rd_list	对象		包 含 mx_record_list 和 ns_record_list	否	
mx_record_l ist	数组		MX 记录列表	否	缺省值:空
name	字符串	长度 1-127	MX 记录-名称	是	
priority	整数	0-65535	MX 记录-优先级	是	
ns_record_l ist	数组		NS 记录列表	否	缺省值:空
name	字符串	长度 1-127	NS 记录-名称	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gs1b.domain.add

请求 body:

```
{
  "name": "apitest2.domain",
  "ttl": 1000000000,
  "policy": "policy_1",
  "status": 1,
  "soastatus": 1,
  "soans": "soans1",
  "soamaibox": "soamailbox1",
  "soaserial": 2147483647,
  "soarefresh": 2147483647,
  "soaretry": 2147483647,
  "soaexpire": 2147483647,
```

```

"soattl": 2147483647,
"soastatus2": 1,
"soans2": "2147483647",
"soamaibox2": "2147483647",
"soaserial2": 2147483647,
"soarefresh2": 2147483647,
"soaretry2": 2147483647,
"soaexpire2": 2147483647,
"soattl2": 2147483647,
"domain_record_list": {
  "mx_record_list": [
    {
      "name": "mx1",
      "priority": 123
    }
  ],
  "ns_record_list": [
    {
      "name": "ns1"
    },
    {
      "name": "ns2"
    }
  ]
}
}

```

获取 GSLB 域名列表

Action: `gslb.domain.list`

请求参数:无

请求举例:

POST

<http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.list>

响应参数:

名称	类型	范围	含义
<code>name</code>	字符串	长度 1-127	域名名称, 例如 <code>www.baidu.com</code> 的域名名称应该为 <code>baidu.com</code>
<code>ttd</code>	整数	0-10000000 00	TTL
<code>policy</code>	字符串	长度 1-191	域名使用的 GSLB 策略名称, 必须存在
<code>status</code>	整数	0-1	域名状态;1:启用;0:禁用

soastatus	整数	0-1	主 SOA 状态;1:启用;0:禁用
soans	字符串	长度 1-127	主 SOA 服务器
soamaibox	字符串	长度 1-127	主 SOA 负责人
soaserial	整数	0-2147483647	主 SOA 序列号
soarefresh	整数	0-2147483647	主 SOA 刷新间隔,单位秒
soaretry	整数	0-2147483647	主 SOA 重试间隔
soaexpire	整数	0-2147483647	主 SOA 过期间隔
soattl	整数	0-2147483647	主 SOA 最小 TTL
soastatus2	整数	0-1	备 SOA 状态;1:启用;0:禁用
soans2	字符串	长度 1-127	备 SOA 服务器
soamaibox2	字符串	长度 1-127	备 SOA 负责人
soaserial2	整数	0-2147483647	备 SOA 序列号
soarefresh2	整数	0-2147483647	备 SOA 刷新间隔,单位秒
soaretry2	整数	0-2147483647	备 SOA 重试间隔
soaexpire2	整数	0-2147483647	备 SOA 过期间隔
soattl2	整数	0-2147483647	备 SOA 最小 TTL
domain_record_list	对象		包含 mx_record_list 和 ns_record_list
mx_record_list	数组		MX 记录列表
name	字符串	长度 1-127	MX 记录-名称
priority	整数	0-65535	MX 记录-优先级
ns_record_list	数组		NS 记录列表
name	字符串	长度 1-127	NS 记录-名称
domain_service_list	数组		域服务列表
name	字符串	长度 1-63	域服务对象--域服务名称
port	整数	1-65534	域服务对象--域服务端口
query_cnt	整数	>=0	域服务对象--域服务接收请求计数
resp_cnt	整数	>=0	域服务对象--域服务发送响应计数
service_policy	字符串	长度 1-63	域服务对象--域服务 GSLB 策略
service_action	整数	0-5	域服务对象--域服务行为; 0:通过;1:转发响应;2:转发;3:转发请求;4:丢弃;5:拒绝;
status	整数	0-1	域服务对象--域服务状态;1:启用;0:禁用

a_record_list	数组		域服务对象—A 记录列表
service_name	字符串	长度 1-63	A 记录—GSLB 服务名称
replace	整数	0-1	A 记录—替换;1:是;0:否;
static	整数	0-1	A 记录—静态;1:是;0:否;
ttl	整数	0-2147483647	A 记录—TTL
weight	整数	0-100	A 记录—权重
mx_record_list	数组		域服务对象—MX 记录列表
name	字符串	长度 1-127	MX 记录—名称
priority	整数	0-65535	MX 记录—优先级;
cname_record_list	数组		域服务对象—CNAME 记录列表
name	字符串	长度 1-127	CNAME 记录—名称
ns_record_list	数组		域服务对象—NS 记录列表
name	字符串	长度 1-127	NS 记录—名称
ptr_record_list	数组		域服务对象—PTR 记录列表
name	字符串	长度 1-127	PTR 记录—名称
geography_list	数组		域服务对象—地理位置列表
geography	字符串	长度 1-127	地理位置—地理位置名称
alias	字符串	长度 1-127	地理位置—别名名称
srv_record_list	数组		域服务对象—SRV 记录列表
name	字符串	长度 1-127	SRV 记录—名称
priority	整数	0-65535	SRV 记录—优先级;
weight	整数	0-65535	SRV 记录—权重
port_flag	整数	0-1	SRV 记录—端口标记;1:是;0:否;
port	整数	0-65534	SRV 记录—端口
dname_record_list	数组		域服务对象—DNAME 记录列表
name	字符串	长度 1-127	DNAME 记录—名称
aaaa_record_list	数组		域服务对象—AAAA 记录列表
name	字符串	长度 1-63	AAAA 记录—GSLB 服务名称
hinfo_record_list	数组		域服务对象—HINFO 记录列表
name	字符串	长度 1-127	HINFO 记录—名称
txt_record_list	数组		域服务对象—TXT 记录列表
name	字符串	长度 1-127	TXT 记录—名称

响应举例:

```
[
  {
    "name": "apitest2.domain",
    "ttl": 1000000000,
    "policy": "policy_1",
    "status": 1,
    "soastatus": 1,
    "soans": "soans1",
```

```

"soamaibox": "soamailbox1",
"soaserial": 2147483647,
"soarefresh": 2147483647,
"soaretry": 2147483647,
"soaexpire": 2147483647,
"soattl": 2147483647,
"soastatus2": 1,
"soans2": "2147483647",
"soamaibox2": "2147483647",
"soaserial2": 2147483647,
"soarefresh2": 2147483647,
"soaretry2": 2147483647,
"soaexpire2": 2147483647,
"soattl2": 2147483647,
"domain_record_list": {
  "mx_record_list": [
    {
      "name": "mx1",
      "priority": 123
    }
  ],
  "ns_record_list": [
    {
      "name": "ns1"
    },
    {
      "name": "ns2"
    }
  ]
},
"domain_service_list": [
  {
    "name": "www",
    "port": 80,
    "query_cnt": 0,
    "resp_cnt": 0,
    "service_policy": "default",
    "service_action": 0,
    "status": 1,
    "a_record_list": [
      {
        "service_name": "service_test1",
        "replace": 1,
        "static": 1,
        "ttl": 2147483647,
        "weight": 0
      }
    ]
  }
]

```

```

    },
    {
      "service_name": "service_test2",
      "replace": 0,
      "static": 0,
      "ttl": 0,
      "weight": 0
    }
  ],
  "mx_record_list": [
    {
      "name": "mx-2.record.com",
      "priority": 123
    },
    {
      "name": "mx1.record.com",
      "priority": 65535
    }
  ],
  "cname_record_list": [
    {
      "name": "cname_record1"
    }
  ],
  "ns_record_list": [
    {
      "name": "ns1.record.com"
    }
  ],
  "ptr_record_list": [
    {
      "name": "arpa.1.2.0.1"
    }
  ],
  "geography_list": [
    {
      "geography": "beijing",
      "alias": "cname_record1"
    },
    {
      "geography": "nanjing",
      "alias": "cname_record1"
    }
  ],
  "srv_record_list": [
    {

```



```

        "name": "serv_record_2",
        "priority": 2,
        "weight": 10,
        "port_flag": 0
    },
    {
        "name": "serv_record_1",
        "priority": 65535,
        "weight": 65535,
        "port_flag": 1,
        "port": 65534
    }
],
"dname_record_list": [
    {
        "name": "dname.record.com"
    }
],
"aaaa_record_list": [
    {
        "name": "aaaa.record.com"
    }
],
"hinfo_record_list": [
    {
        "name": "hinfo.record.com"
    }
],
"txt_record_list": [
    {
        "name": "text_record"
    }
]
}
]
}
]

```

获取 GSLB 指定域名

Action: `gslb.domain.get`

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	长度 1-127	域名名称	是	必须存在
------	-----	----------	------	---	------

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.get

Body:

```
{
  "name": "apitest2.domain"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-127	域名名称, 例如 www.baidu.com 的域名名称应该为 baidu.com
ttd	整数	0-10000000 00	TTL
policy	字符串	长度 1-191	域名使用的 GSLB 策略名称, 必须存在
status	整数	0-1	域名状态;1:启用;0:禁用
soastatus	整数	0-1	主 SOA 状态;1:启用;0:禁用
soans	字符串	长度 1-127	主 SOA 服务器
soamaibox	字符串	长度 1-127	主 SOA 负责人
soaserial	整数	0-21474836 47	主 SOA 序列号
soarefresh	整数	0-21474836 47	主 SOA 刷新间隔, 单位秒
soaretry	整数	0-21474836 47	主 SOA 重试间隔
soaexpire	整数	0-21474836 47	主 SOA 过期间隔
soattl	整数	0-21474836 47	主 SOA 最小 TTL
soastatus2	整数	0-1	备 SOA 状态;1:启用;0:禁用
soans2	字符串	长度 1-127	备 SOA 服务器
soamaibox2	字符串	长度 1-127	备 SOA 负责人
soaserial2	整数	0-21474836 47	备 SOA 序列号
soarefresh2	整数	0-21474836 47	备 SOA 刷新间隔, 单位秒
soaretry2	整数	0-21474836 47	备 SOA 重试间隔
soaexpire2	整数	0-21474836 47	备 SOA 过期间隔
soattl2	整数	0-21474836 47	备 SOA 最小 TTL
domain_record_list	对象		包含 mx_record_list 和 ns_record_list
mx_record_list	数组		MX 记录列表

name	字符串	长度 1-127	MX 记录-名称
priority	整数	0-65535	MX 记录-优先级
ns_record_list	数组		NS 记录列表
name	字符串	长度 1-127	NS 记录-名称
domain_service_list	数组		域服务列表
name	字符串	长度 1-63	域服务对象-域服务名称
port	整数	1-65534	域服务对象-域服务端口
service_policy	字符串	长度 1-63	域服务对象-域服务 GSLB 策略
query_cnt	整数	>=0	域服务对象-域服务接收请求计数
resp_cnt	整数	>=0	域服务对象-域服务发送响应计数
service_action	整数	0-5	域服务对象-域服务行为; 0:通过;1:转发响应;2:转发;3:转发请求;4:丢弃;5:拒绝;
status	整数	0-1	域服务对象-域服务状态;1:启用;0:禁用
a_record_list	数组		域服务对象-A 记录列表
service_name	字符串	长度 1-63	A 记录-GSLB 服务名称
replace	整数	0-1	A 记录-替换;1:是;0:否;
static	整数	0-1	A 记录-静态;1:是;0:否;
ttl	整数	0-2147483647	A 记录-TTL
weight	整数	0-100	A 记录-权重
mx_record_list	数组		域服务对象-MX 记录列表
name	字符串	长度 1-127	MX 记录-名称
priority	整数	0-65535	MX 记录-优先级;
cname_record_list	数组		域服务对象-CNAME 记录列表
name	字符串	长度 1-127	CNAME 记录-名称
ns_record_list	数组		域服务对象-NS 记录列表
name	字符串	长度 1-127	NS 记录-名称
ptr_record_list	数组		域服务对象-PTR 记录列表
name	字符串	长度 1-127	PTR 记录-名称
geography_list	数组		域服务对象-地理位置列表
geography	字符串	长度 1-127	地理位置-地理位置名称
alias	字符串	长度 1-127	地理位置-别名名称
srv_record_list	数组		域服务对象-SRV 记录列表
name	字符串	长度 1-127	SRV 记录-名称
priority	整数	0-65535	SRV 记录-优先级;
weight	整数	0-65535	SRV 记录-权重
port_flag	整数	0-1	SRV 记录-端口标记;1:是;0:否;
port	整数	0-65534	SRV 记录-端口
dname_record_list	数组		域服务对象-DNAME 记录列表
name	字符串	长度 1-127	DNAME 记录-名称
aaaa_record_list	数组		域服务对象-AAAA 记录列表

name	字符串	长度 1-63	AAAA 记录—GSLB 服务名称
hinfo_record_list	数组		域服务对象—HINFO 记录列表
name	字符串	长度 1-127	HINFO 记录—名称
txt_record_list	数组		域服务对象—TXT 记录列表
name	字符串	长度 1-127	TXT 记录—名称

响应举例：

```
{
  "name": "apitest2.domain",
  "ttl": 1000000000,
  "policy": "policy_1",
  "status": 1,
  "soastatus": 1,
  "soans": "soans1",
  "soamaibox": "soamailbox1",
  "soaserial": 2147483647,
  "soarefresh": 2147483647,
  "soaretry": 2147483647,
  "soaexpire": 2147483647,
  "soattl": 2147483647,
  "soastatus2": 1,
  "soans2": "2147483647",
  "soamaibox2": "2147483647",
  "soaserial2": 2147483647,
  "soarefresh2": 2147483647,
  "soaretry2": 2147483647,
  "soaexpire2": 2147483647,
  "soattl2": 2147483647,
  "domain_record_list": {
    "mx_record_list": [
      {
        "name": "mx1",
        "priority": 123
      }
    ],
    "ns_record_list": [
      {
        "name": "ns1"
      },
      {
        "name": "ns2"
      }
    ]
  }
},
```

```

"domain_service_list": [
  {
    "name": "www",
    "port": 80,
    "query_cnt": 0,
    "resp_cnt": 0,
    "service_policy": "default",
    "service_action": 0,
    "status": 1,
    "a_record_list": [
      {
        "service_name": "service_test1",
        "replace": 1,
        "static": 1,
        "ttl": 2147483647,
        "weight": 0
      },
      {
        "service_name": "service_test2",
        "replace": 0,
        "static": 0,
        "ttl": 0,
        "weight": 0
      }
    ],
    "mx_record_list": [
      {
        "name": "mx-2.record.com",
        "priority": 123
      },
      {
        "name": "mx1.record.com",
        "priority": 65535
      }
    ],
    "cname_record_list": [
      {
        "name": "cname_record1"
      }
    ],
    "ns_record_list": [
      {
        "name": "ns1.record.com"
      }
    ],
    "ptr_record_list": [

```

```

    {
      "name": "arpa.1.2.0.1"
    }
  ],
  "geography_list": [
    {
      "geography": "beijing",
      "alias": "cname_record1"
    },
    {
      "geography": "nanjing",
      "alias": "cname_record1"
    }
  ],
  "srv_record_list": [
    {
      "name": "serv_record_2",
      "priority": 2,
      "weight": 10,
      "port_flag": 0
    },
    {
      "name": "serv_record_1",
      "priority": 65535,
      "weight": 65535,
      "port_flag": 1,
      "port": 65534
    }
  ],
  "dname_record_list": [
    {
      "name": "dname.record.com"
    }
  ],
  "aaaa_record_list": [
    {
      "name": "aaaa.record.com"
    }
  ],
  "hinfo_record_list": [
    {
      "name": "hinfo.record.com"
    }
  ],
  "txt_record_list": [
    {

```

```

        "name": "text_record"
    }
}
]
}
}
}

```

编辑指定 GSLB 域名

Action: `gslb.domain.edit`

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-127	域名名称, 例如 www.baidu.com 的域名名称应该为 baidu.com	是	唯一
ttd	整数	0-100000 0000	TTL	否	缺省值: 不修改
policy	字符串	长度 1-191	域名使用的 GSLB 策略名称, 必须存在	否	缺省值: 不修改
status	整数	0-1	域名状态; 1: 启用; 0: 禁用	否	缺省值: 不修改
soastatus	整数	0-1	主 SOA 状态; 1: 启用; 0: 禁用	否	缺省值: 不修改
soans	字符串	长度 1-127	主 SOA 服务器	否	缺省值: 不修改
soamaibox	字符串	长度 1-127	主 SOA 负责人	否	缺省值: 不修改
soaserial	整数	0-214748 3647	主 SOA 序列号	否	缺省值: 不修改
soarefresh	整数	0-214748 3647	主 SOA 刷新间隔, 单位秒	否	缺省值: 不修改
soaretry	整数	0-214748 3647	主 SOA 重试间隔	否	缺省值: 不修改
soaexpire	整数	0-214748 3647	主 SOA 过期间隔	否	缺省值: 不修改
soattl	整数	0-214748 3647	主 SOA 最小 TTL	否	缺省值: 不修改
soastatus2	整数	0-1	备 SOA 状态; 1: 启用; 0: 禁用	否	缺省值: 不修改
soans2	字符串	长度 1-127	备 SOA 服务器	否	缺省值: 不修改
soamaibox2	字符串	长度 1-127	备 SOA 负责人	否	缺省值: 不修改
soaserial2	整数	0-214748 3647	备 SOA 序列号	否	缺省值: 不修改
soarefresh2	整数	0-214748 3647	备 SOA 刷新间隔, 单位秒	否	缺省值: 不修改

soaretry2	整数	0-214748 3647	备 SOA 重试间隔	否	缺省值:不修改
soaexpire2	整数	0-214748 3647	备 SOA 过期间隔	否	缺省值:不修改
soattl2	整数	0-214748 3647	备 SOA 最小 TTL	否	缺省值:不修改
domain_reco rd_list	对象		包 含 mx_record_list 和 ns_record_list	否	
mx_record_l ist	数组		MX 记录列表	否	缺省值:不修改
name	字符串	长 度 1-127	MX 记录-名称	是	
priority	整数	0-65535	MX 记录-优先级	是	
ns_record_l ist	数组		NS 记录列表	否	缺省值:不修改
name	字符串	长 度 1-127	NS 记录-名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.edit>

请求 body:

```
{
  "name": "apitest2.domain",
  "ttl": 1000000000,
  "policy": "policy_1",
  "status": 1,
  "soastatus": 1,
  "soans": "soans1",
  "soamaibox": "soamailbox1",
  "soaserial": 2147483647,
  "soarefresh": 2147483647,
  "soaretry": 2147483647,
  "soaexpire": 2147483647,
  "soattl": 2147483647,
  "soastatus2": 1,
  "soans2": "2147483647",
  "soamaibox2": "2147483647",
  "soaserial2": 2147483647,
  "soarefresh2": 2147483647,
  "soaretry2": 2147483647,
  "soaexpire2": 2147483647,
  "soattl2": 2147483647,
  "domain_record_list": {
    "mx_record_list": [{
```



```

        "name": "mx1",
        "priority": 123
    }],
    "ns_record_list": [
        {"name": "ns1"}, {"name": "ns2"}
    ]
}
}

```

添加域服务

Action: gslb.domain.service.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-127	域名名称	是	必须存在
domain_service	对象		域服务对象	是	
name	字符串	长度 1-63	域服务对象—域服务名称	是	唯一
port	整数	1-65534	域服务对象—域服务端口	是	每域名唯一
service_policy	字符串	长度 1-191	域服务对象—域服务 GSLB 策略	否	必须存在; 缺省值: default
service_action	整数	0-5	域服务对象—域服务行为	否	0: 通过 1: 转发响应 2: 转发 3: 转发请求 4: 丢弃 5: 拒绝 缺省值: 0
status	整数	0-1	域服务对象—域服务状态	否	1: 开启; 0: 禁用; 缺省值: 1
a_record_list	数组		域服务对象—A 记录列表	否	缺省值: 空, 最多支持 65535 个
service_name	字符串	长度 1-191	A 记录—GSLB 服务名称	是	必须存在
replace	整数	0-1	A 记录—替换; 1: 是; 0: 否;	否	缺省值: 0
static	整数	0-1	A 记录—静态; 1: 是; 0: 否;	否	缺省值: 0
ttl	整数	0-2147483647	A 记录—TTL	否	缺省值: 0
weight	整数	0-100	A 记录—权重	否	缺省值: 1
mx_record_list	数组		域服务对象—MX 记录列表	否	缺省值: 空, 最多支持 65535 个
name	字符串	长度 1-127	MX 记录—名称	是	
priority	整数	0-65535	MX 记录—优先级;	是	
cname_record_list	数组		域服务对象—CNAME 记录列表	否	缺省值: 空, 最多支持 65535 个
name	字符串	长度 1-127	CNAME 记录—名称	是	

ns_record_list	数组		域服务对象—NS 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	NS 记录—名称	是	
ptr_record_list	数组		域服务对象—PTR 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	PTR 记录—名称	是	
geography_list	数组		域服务对象—地理位置列表	否	缺省值:空, 最多支持 65535 个
geography	字符串	长度 1-127	地理位置—地理位置名称	是	
alias	字符串	长度 1-127	地理位置—别名名称	是	必须在 CNAME 中存在
srv_record_list	数组		域服务对象—SRV 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	SRV 记录—名称	是	
priority	整数	0-65535	SRV 记录—优先级;	是	
weight	整数	0-65535	SRV 记录—权重	否	缺省值:10
port_flag	整数	0-1	SRV 记录—端口标记;1:是;0: 否;	否	缺省值:0
port	整数	0-65534	SRV 记录—端口	否	缺省值:0
cname_record_list	数组		域服务对象—CNAME 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	CNAME 记录—名称	是	
aaaa_record_list	数组		域服务对象—AAAA 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-63	AAAA 记录—GSLB 服务名称	是	必须存在
hinfo_record_list	数组		域服务对象—HINFO 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	HINFO 记录—名称	是	
txt_record_list	数组		域服务对象—TXT 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	TXT 记录—名称	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.service
.add

请求 body:

```
{
  "name": "apitest2.domain",
  "domain_service":{
    "name":"www",
    "port":80,
    "service_policy":"default",
    "service_action":0,
    "status":1,
    "a_record_list":[
```

```

    {
      "service_name":"service_test1",
      "replace":1,
      "static":1,
      "ttl":2147483647
    },
    {
      "service_name":"service_test2"
    }
  ],
  "mx_record_list":[
    {
      "name":"mx1.record.com",
      "priority":65535
    },
    {
      "name":"mx-2.record.com",
      "priority":123
    }
  ],
  "cname_record_list":[
    {
      "name":"cname_record1"
    }
  ],
  "ns_record_list":[
    {
      "name":"ns1.record.com"
    }
  ],
  "ptr_record_list":[
    {
      "name":"arpa.1.2.0.1"
    }
  ],
  "geography_list":[
    {
      "geography":"beijing",
      "alias":"cname_record1"
    },
    {
      "geography":"nanjing",
      "alias":"cname_record1"
    }
  ],
  "srv_record_list":[

```

```

    {
      "name":"serv_record_1",
      "priority":65535,
      "weight":65535,
      "port_flag":1,
      "port":65534
    },
    {
      "name":"serv_record_2",
      "priority":2
    }
  ],
  "dname_record_list":[
    {
      "name":"dname.record.com"
    }
  ],
  "aaaa_record_list":[
    {
      "name":"aaaa.record.com"
    }
  ],
  "hinfo_record_list":[
    {
      "name":"hinfo.record.com"
    }
  ],
  "txt_record_list":[
    {
      "name":"text_record"
    }
  ]
}
}

```

编辑域服务

Action: `gslb.domain.service.edit`

请求参数:

名称	类型	范围	含义	必选	备注
<code>name</code>	字符串	长度 1-127	域名名称	是	必须存在
<code>domain_service</code>	对象		域服务对象	是	
<code>name</code>	字符串	长度 1-63	域服务对象--域服务名称	是	必须存在

port	整数	1-65534	域服务对象—域服务端口	是	每域名唯一
service_policy	字符串	长度 1-191	域服务对象—域服务 GSLB 策略	否	必须存在; 缺省值:不修改
service_action	整数	0-5	域服务对象—域服务行为	否	0:通过 1:转发响应 2:转发 3:转发请求 4:丢弃 5:拒绝 缺省值:不修改
status	整数	0-1	域服务对象—域服务状态	否	1:开启;0:禁用; 缺省值:不修改
a_record_list	数组		域服务对象—A 记录列表	否	缺省值:不修改
service_name	字符串	长度 1-191	A 记录—GSLB 服务名称	是	必须存在
replace	整数	0-1	A 记录—替换;1:是;0:否;	否	缺省值:0
static	整数	0-1	A 记录—静态;1:是;0:否;	否	缺省值:0
ttl	整数	0-2147483647	A 记录—TTL	否	缺省值:0
weight	整数	0-100	A 记录—权重	否	缺省值:1
mx_record_list	数组		域服务对象—MX 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	MX 记录—名称	是	
priority	整数	0-65535	MX 记录—优先级;	是	
cname_record_list	数组		域服务对象—CNAME 记录列表	否	缺省值:不修改
name	字符串	长度 1-127	CNAME 记录—名称	是	
ns_record_list	数组		域服务对象—NS 记录列表	否	缺省值:不修改
name	字符串	长度 1-127	NS 记录—名称	是	
ptr_record_list	数组		域服务对象—PTR 记录列表	否	缺省值:不修改
name	字符串	长度 1-127	PTR 记录—名称	是	
geography_list	数组		域服务对象—地理位置列表	否	缺省值:不修改
geography	字符串	长度 1-127	地理位置—地理位置名称	是	
alias	字符串	长度 1-127	地理位置—别名名称	是	必须在 CNAME 中存在
srv_record_list	数组		域服务对象—SRV 记录列表	否	缺省值:不修改
name	字符串	长度 1-127	SRV 记录—名称	是	
priority	整数	0-65535	SRV 记录—优先级;	是	
weight	整数	0-65535	SRV 记录—权重	否	缺省值:10
port_flag	整数	0-1	SRV 记录—端口标记;1:是;0:否;	否	缺省值:0
port	整数	0-65534	SRV 记录—端口	否	缺省值:0
dname_record_list	数组		域服务对象—DNAME 记录列表	否	缺省值:空, 最多支持 65535 个
name	字符串	长度 1-127	DNAME 记录—名称	是	
aaaa_record_list	数组		域服务对象—AAAA 记录列表	否	缺省值:不修改
name	字符串	长度 1-63	AAAA 记录—GSLB 服务名称	是	必须存在
hinfo_record_list	数组		域服务对象—HINFO 记录列表	否	缺省值:不修改

name	字符串	长度 1-127	HINFO 记录一名称	是	
txt_record_list	数组		域服务对象—TXT 记录列表	否	缺省值:不修改
name	字符串	长度 1-127	TXT 记录一名称	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.service.edit

请求 body:

```
{
  "name": "apitest2.domain",
  "domain_service": {
    "name": "www",
    "port": 80,
    "service_policy": "default",
    "service_action": 0,
    "status": 1,
    "a_record_list": [
      {
        "service_name": "service_test1",
        "replace": 1,
        "static": 1,
        "ttl": 2147483647
      },
      {
        "service_name": "service_test2"
      }
    ],
    "mx_record_list": [
      {
        "name": "mx1.record.com",
        "priority": 65535
      },
      {
        "name": "mx-2.record.com",
        "priority": 123
      }
    ],
    "cname_record_list": [
      {
        "name": "cname_record1"
      }
    ],
    "ns_record_list": [
      {
```

```
        "name": "ns1.record.com"
    }
],
"ptr_record_list": [
    {
        "name": "arpa.1.2.0.1"
    }
],
"geography_list": [
    {
        "geography": "beijing",
        "alias": "cname_record1"
    },
    {
        "geography": "nanjing",
        "alias": "cname_record1"
    }
],
"srv_record_list": [
    {
        "name": "serv_record_1",
        "priority": 65535,
        "weight": 65535,
        "port_flag": 1,
        "port": 65534
    },
    {
        "name": "serv_record_2",
        "priority": 2
    }
],
"dname_record_list": [
    {
        "name": "dname.record.com"
    }
],
"aaaa_record_list": [
    {
        "name": "aaaa.record.com"
    }
],
"hinfo_record_list": [
    {
        "name": "hinfo.record.com"
    }
],
```

```

    "txt_record_list":[
      {
        "name":"text_record"
      }
    ]
  }
}

```

删除域服务

Action: gslb.domain.service.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-127	域名名称	是	必须存在
domain_service	对象		域服务对象	是	
name	字符串	长度 1-63	域服务对象--域服务名称	是	必须存在

请求举例:

POST
 http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.service.del

请求 body:

```

{
  "name": "apitest2.domain",
  "domain_service": {
    "name": "www"
  }
}

```

删除 GSLB 域名

Action: gslb.domain.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-127	域名名称	是	必须存在

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.domain.del

请求 body:

```

{"name": "apitest2.domain"}

```


GSLB 地理位置

上传地理位置文件

Action: `gslb.geo.file.upload`

请求参数:无

请求举例:

POST

`http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.geo.file.upload`



此 API 使用 form-data 方式上传一个 GEO 地理位置文件

增加地理位置条目

Action: `gslb.geo.item.add`

请求参数:

名称	类型	范围	含义	必选	备注
<code>location</code>	字符串	长度 1-127	地理位置	是	
<code>ipstart</code>	IPv4 地址		起始地址	是	
<code>ipend</code>	IPv4 地址		结束地址	是	

请求举例:

POST

`http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.geo.item.add`

Body:

```
{
  "location": "Asia.China.Beijing",
  "ipstart": "111.111.111.111",
  "ipend": "111.111.111.112"
}
```

删除地理位置条目

Action: `gslb.geo.item.del`

请求参数:

名称	类型	范围	含义	必选	备注
<code>location</code>	字符串	长度 1-127	地理位置	是	
<code>ipstart</code>	IPv4 地址		起始地址	是	

ipend	IPv4 地址		结束地址	是	
-------	---------	--	------	---	--

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.geo.item.del
 Body:

```
{
  "location": "Asia.China.Beijing",
  "ipstart": "111.111.111.111",
  "ipend": "111.111.111.112"
}
```

GSLB 角色

GSLB 角色获取

Action: **gslb.global.role.get**

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.global.role.get

响应参数:

名称	类型	范围	含义
role_controller	整数	0,1	控制器角色:1:是;2:否
role_member	整数	0,1	成员角色:1:是;2:否

响应举例:

```
{
  "role_member": 1,
  "role_controller": 0
}
```

GSLB 角色设置

Action: **gslb.global.role.set**

请求参数:

名称	类型	范围	含义	必选	备注
role_controller	整数	0,1	控制器角色:1:是;2:否	是	系统默认值为0
role_member	整数	0,1	成员角色:1:是;2:否	是	系统默认值为0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.global.role.set

```
t
Body:
{
  "role_member": 1,
  "role_controller": 0
}
```

GSLB 组

GSLB 组状态获取

Action: `gslb.group.status.get`

请求参数:无

请求举例:

```
GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.group.status.get
```

响应参数:

名称	类型	范围	含义
groupid	整数	0-255	组 ID, 0 表示 delete 当前组所有配置
status	整数	0, 1	组状态:1:开启;2:禁用
priority	整数	1-255	优先级
autolearn	整数	0, 1	自动学习:1:是;2:否
saveconfig	整数	0, 1	保存配置:1:是;2:否
anyoneconfig	整数	0, 1	任何成员:1:是;2:否

响应举例:

```
{
  "groupid":1,
  "status":1,
  "priority":100,
  "autolearn":1,
  "saveconfig":1,
  "anyoneconfig":1
}
```

GSLB 组状态设置

Action: `gslb.group.status.set`

请求参数:

名称	类型	范围	含义	必选	备注
groupid	整数	0-255	组 ID, 0 表示 delete 当前组所有配置	是	系统默认值为 0
status	整数	0, 1	组状态:1:开启;2:禁用	是	系统默认值为 0

priority	整数	1-255	优先级	是	系统默认值为 0
autolearn	整数	0, 1	自动学习:1:是;2:否	是	系统默认值为 0
saveconfig	整数	0, 1	保存配置:1:是;2:否	是	系统默认值为 0
anyoneconfig	整数	0, 1	任何成员:1:是;2:否	是	系统默认值为 0

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.group.status.set

Body:

```
{
  "groupid":1,
  "status":1,
  "priority":100,
  "autolearn":1,
  "saveconfig":1,
  "anyoneconfig":1
}
```



改变组状态的 group_id 设置时,会删除所有 GSLB 组邻居

获取 GSLB 组邻居列表

Action: gslb.group.neighbor.list

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.group.neighbor.list

响应参数:

名称	类型	范围	含义
groupid	整数	1-255	组 ID
srcaddr	IPv4 地址		组邻居地址

响应举例:

```
[{
  "groupid":1,
  "srcaddr":"100.10.10.1"
},{
  "groupid":1,
  "srcaddr":"100.20.20.2"
}]
```

添加 GSLB 组邻居

Action: gslb.group.neighbor.add

请求参数:

名称	类型	范围	含义	必选	备注
groupid	整数	1-255	组 ID	是	
srcaddr	IPv4 地址		组邻居地址	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.group.neighbor.add

Body:

```
{  
  "groupid":1,  
  "srcaddr":"100.30.30.3"  
}
```



Note

添加 GSLB 组邻居时,邻居的组 ID 必须与组状态的 group_id 相同

删除 GSLB 组邻居

Action: gslb.group.neighbor.del

请求参数:

名称	类型	范围	含义	必选	备注
groupid	整数	1-255	组 ID	是	
srcaddr	IPv4 地址		组邻居地址	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=gslb.group.neighbor.del

Body:

```
{  
  "groupid":1,  
  "srcaddr":"100.30.30.3"  
}
```

网络

接口

网络模式

网络模式配置获取

Action: network.mode.get

请求参数: 无

请求举例:

GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.mode.get

响应参数:

名称	类型	范围	含义
network_mode	整数	0-1	0: 网关模式 1: 桥模式 网关模式下全局 IP 地址配置必须为空, 桥模式下全局 IP 地址必须配置
ipv4_addr	Ipv4 address		全局 Ipv4 地址
ipv4_mask	Ipv4 address		全局 Ipv4 掩码
ipv4_gw	Ipv4 address		全局 Ipv4 网关
ipv6_addr	Ipv6 address		全局 Ipv6 地址
ipv6_prefix	整数	1-128	全局 Ipv6 前缀

响应举例:

```
{
  "network_mode": 0,
  "ipv4_addr": "0.0.0.0",
  "ipv4_mask": "0.0.0.0",
  "ipv4_gw": "0.0.0.0",
  "ipv6_addr": "::",
  "ipv6_prefix": 0,
  "ipv6_gw": "::"
}
```

网络模式配置设置

Action: network.mode.set

请求参数:

名称	类型	范围	含义	必选	备注
network_mode	整数	0-1	网络模式	是	0: 网关模式 1: 桥模式 网关模式下全局 IP 地址配置必须为空, 桥模式下全局 IP 地址必须配置

ipv4_addr	Ipv4 address		全局 Ipv4 地址	是	
ipv4_mask	Ipv4 address		全局 Ipv4 掩码	是	
ipv4_gw	Ipv4 address		全局 Ipv4 网关	是	
ipv6_addr	Ipv6 address		全局 Ipv6 地址	否	
ipv6_prefix	整数	1-128	全局 Ipv6 前缀	否	
ipv6_gw	Ipv6 address		全局 Ipv6 网关	否	

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.mode.set

请求 body:

```
{
  "network_mode": 1,
  "ipv4_addr": "100.0.0.10",
  "ipv4_mask": "255.0.0.0",
  "ipv4_gw": "0.0.0.0",
  "ipv6_addr": "::",
  "ipv6_prefix": 0,
  "ipv6_gw": "::"
}
```



Note

网络模式由网关模式改为桥模式时，所有接口需要清除所有配置

管理口配置

管理接口配置获取

Action: interface.mgmt.get

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.mgmt.get

响应参数:

名称	类型	范围	含义
status	整数	0-1	接口状态
mac_addr	字符串	Mac 地址	MAC 地址
dhcp_clien	整数	0-1	启动 dhcp
management_services	数组	0-7	管理服务配置数组
service_type	整数	0-6	服务类型
service_ipv4_acl	整数	2-198, 0	0 表示没有配置 ipv4 的 acl
service_ipv6_acl	字符串	0-191	空字符串表示没有配置 ipv6 的 acl

ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向
ipv4_acl	整数	2-198	Ipv4 访问列表
ipv6_acl	字符串	0-191	Ipv6 访问列表
ipv4_addr	IP 地址		Ipv4 地址
ipv4_mask	IP 地址		Ipv4 掩码
ipv4_gw	IP 地址		Ipv4 网关
ipv6_addr	IPV6 地址		Ipv6 地址
ipv6_prefix	整数	1-128	Ipv6 前缀
ipv6_gw	IPV6 地址		Ipv6 网关
speed	整数	0-4	速度
duplexity	整数	0-2	模式
flow_control	整数	0-1	硬件流控控制

响应举例:

```
{
  "status": 1,
  "mac_addr": "00:0D:48:4F:03:58",
  "dhcp_client": 0,
  "ipv4_addr": "192.168.70.150",
  "ipv4_mask": "255.255.255.0",
  "ipv4_gw": "192.168.70.250",
  "ipv4_acl": 0,
  "ipv6_addr": "2000:192:168:70::150",
  "ipv6_prefix": 64,
  "ipv6_gw": "",
  "ipv6_acl": "",
  "management_services": [
    {
      "service_type": 0
    },
    {
      "service_type": 1
    },
    {
      "service_type": 2
    },
    {
      "service_type": 3
    },
    {
      "service_type": 4
    },
    {
      "service_type": 5
    }
  ],
}
```



```

"speed": 0,
"duplexity": 2,
"flow_control": 1
}

```

管理接口配置设置

Action: interface.mgmt.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0-1	接口状态	否	0:disable;1:enable;缺省值:不修改
dhcp_client	整数	0-1	启动 dhcp	否	0:disable;1:enable;缺省值:不修改
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http 1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh 5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置
ipv6_acl	字符串	0-191	Ipv6 访问列表	否	“”表示没有配置
ipv4_addr	IP 地址		Ipv4 地址	否	空表示没有配置, 缺省值:不修改
ipv4_mask	IP 地址		Ipv4 掩码	否	空表示没有配置, 缺省值:不修改
ipv4_gw	IP 地址		Ipv4 网关	否	空表示没有配置, 缺省值:不修改
ipv6_addr	IPV6 地址		Ipv6 地址	否	空表示没有配置, 缺省值:不修改
ipv6_prefix	整数	1-128	Ipv6 前缀	否	空表示没有配置, 缺省值:不修改
ipv6_gw	IPV6 地址		Ipv6 网关	否	空表示没有配置, 缺省值:不修改
speed	整数	0-4	速度	否	0: 10Mbit/s 1: 100Mbit/s 2: 1000Mbit/s 3: 10000Mbit/s 4: auto 缺省值:不修改

duplexity	整数	0-2	模式	否	0: full 1: half 2: auto 缺省值: 不修改
flow_control	整数	0-1	硬件流控控制	否	0:disable;1:enable;缺省值: 不修改

请求举例:

POST

http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.mgmt.set

请求 body:

```
{
  "status": 1,
  "mac_addr": "00:0D:48:4F:03:58",
  "dhcp_client": 0,
  "ipv4_addr": "192.168.70.150",
  "ipv4_mask": "255.255.255.0",
  "ipv4_gw": "192.168.70.250",
  "ipv4_acl": 0,
  "ipv6_addr": "2000:192:168:70::150",
  "ipv6_prefix": 64,
  "ipv6_gw": "",
  "ipv6_acl": "",
  "management_services": [
    {
      "service_type": 0
    },
    {
      "service_type": 1
    },
    {
      "service_type": 2
    },
    {
      "service_type": 3
    },
    {
      "service_type": 4
    },
    {
      "service_type": 5
    }
  ],
  "speed": 0,
  "duplexity": 2,
  "flow_control": 1
}
```

以太网接口配置

以太接口配置列表

Action: interface.ethernet.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ethernet.list

响应参数:

名称	类型	范围	含义	必选	说明
slot	整数	0-255	槽位	是	
port	整数	0-255	端口	是	
status	整数	0-1	接口状态	否	0 表示 disable, 1 表示 enable
description	字符串	0-191	接口描述	否	
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http 1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh 5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection
mac_addr	字符串	Mac 地址	MAC 地址	否	
ipv4_addr	IP 地址		Ipv4 地址	否	
ipv4_mask	IP 地址		Ipv4 掩码	否	
dhcp_client	整数	0-1	DHCP 启动	否	0 表示 disable, 1 表示 enable
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置
ipv4_list	数组	0-16	Ipv4 地址列表	否	
Ipv6_list	数组	0-16	Ipv6 地址列表	否	

ipv6_addr	IPv6 地址		Ipv6 地址	否	空表示没有配置
ipv6_prefix	整数	1-128	Ipv6 前缀	否	0 表示没有配置
ipv6_anycast	整数	0-1	Ipv6 多播	否	0 表示 disable, 1 表示 enable
ipv6_local_auto	整数	0-1	Ipv6 自动配置	否	0 表示 disable, 1 表示 enable
ipv6_local_addr	IPv6 地址		Ipv6 本地地址	否	
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀	否	0 表示没有配置
ipv6_local_anycast	整数		Ipv6 本地多播	否	0 表示 disable, 1 表示 enable
ipv6_acl	字符串	0-15	Ipv6 访问列表	否	0 表示没有配置
speed	整数	0-4	速度	否	0: 10Mbit/s 1: 100Mbit/s 2: 1000Mbit/s 3: 10000Mbit/s 4: auto
duplexity	整数	0-2	模式	否	0: full 1: half 2: auto
flow_control	整数	0-1	硬件流控控制	否	0 表示 disable, 1 表示 enable
permit_wildcard	整数	0-1	混杂模式	否	0 表示 disable, 1 表示 enable
grat_arp	整数	0-1	免费 ARP	否	0 表示 disable, 1 表示 enable
nat_dir	整数	0-3	Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection
no_vlan_forward	整数	0-1	禁止 VLAN 转发	否	0 表示关闭, 1 表示开启

响应举例:

```

[[
  {
    "slot": 1,
    "port": 0,
    "description": "",
    "status": 0,
    "mac_addr": "00:0D:48:4E:46:BB",
    "duplexity": 2,
    "speed": 0,
    "flow_control": 0,
    "permit_wildcard": 0,
    "grat_arp": 0,
    "ipv4_acl": 0,
    "nat_dir": 0,
  }
]]

```

```

    "dhcp_client": 0,
    "management_services": [
      {
        "service_type": 2
      }
    ],
    "ipv4_list": [],
    "ipv6_list": [],
    "ipv6_local_auto": 0,
    "ipv6_acl": ""
  },
  {
    "slot": 1,
    "port": 1,
    "description": "",
    "status": 1,
    "mac_addr": "00:0D:48:4E:46:B9",
    "duplexity": 2,
    "speed": 0,
    "flow_control": 0,
    "permit_wildcard": 0,
    "grat_arp": 0,
    "ipv4_acl": 0,
    "nat_dir": 0,
    "dhcp_client": 0,
    "management_services": [
      {
        "service_type": 2
      }
    ],
    "ipv4_list": [],
    "ipv6_list": [],
    "ipv6_local_auto": 0,
    "ipv6_acl": ""
  },
  {
    "slot": 3,
    "port": 3,
    "description": "",
    "status": 1,
    "mac_addr": "00:0D:48:6D:36:7C",
    "duplexity": 2,
    "speed": 0,
    "flow_control": 0,
    "permit_wildcard": 0,
    "grat_arp": 0,

```

```

    "ipv4_acl": 0,
    "nat_dir": 0,
    "dhcp_client": 0,
    "management_services": [
        {
            "service_type": 2
        }
    ],
    "ipv4_list": [],
    "ipv6_list": [],
    "ipv6_local_auto": 0,
    "ipv6_acl": ""
}
]

```

以太接口配置获取

Action: interface.ethernet.get

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-255	槽位	是	
port	整数	0-255	端口	是	

请求举例:

```

POST
http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ethernet.get

```

请求 body:

```

{
    "slot": 1,
    "port": 0
}

```

响应参数:

名称	类型	范围	含义	必选	说明
slot	整数	0-255	槽位	是	
port	整数	0-255	端口	是	
status	整数	0-1	接口状态	否	0 表示 disable, 1 表示 enable
description	字符串	0-191	接口描述	否	
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http 1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh

					5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection
mac_addr	字符串	Mac 地址	MAC 地址	否	
ipv4_addr	IP 地址		Ipv4 地址	否	
ipv4_mask	IP 地址		Ipv4 掩码	否	
dhcp_client	整数	0-1	DHCP 启动	否	0 表示 disable, 1 表示 enable
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置
ipv4_list	数组	0-16	Ipv4 地址列表	否	
Ipv6_list	数组	0-16	Ipv6 地址列表	否	
ipv6_addr	IPV6 地址		Ipv6 地址	否	空表示没有配置
ipv6_prefix	整数	1-128	Ipv6 前缀	否	0 表示没有配置
ipv6_anycast	整数	0-1	Ipv6 多播	否	0 表示 disable, 1 表示 enable
ipv6_local_auto	整数	0-1	Ipv6 自动配置	否	0 表示 disable, 1 表示 enable
ipv6_local_address	IPV6 地址		Ipv6 本地地址	否	
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀	否	0 表示没有配置
ipv6_local_anycast	整数		Ipv6 本地多播	否	0 表示 disable, 1 表示 enable
ipv6_acl	字符串	0-15	Ipv6 访问列表	否	0 表示没有配置
speed	整数	0-4	速度	否	0: 10Mbit/s 1: 100Mbit/s 2: 1000Mbit/s 3: 10000Mbit/s 4: auto
duplexity	整数	0-2	模式	否	0: full 1: half 2: auto
flow_control	整数	0-1	硬件流控控制	否	0 表示 disable, 1 表示 enable
permit_wildcard	整数	0-1	混杂模式	否	0 表示 disable, 1 表示 enable
grat_arp	整数	0-1	免费 ARP	否	0 表示 disable, 1 表示 enable

nat_dir	整数	0-3	Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection
no_vlan_forward	整数	0-1	禁止 VLAN 转发	否	0 表示关闭, 1 表示开启

响应举例:

```
{
  "slot": 1,
  "port": 0,
  "description": "",
  "status": 0,
  "mac_addr": "00:0D:48:4E:46:BB",
  "duplexity": 2,
  "speed": 0,
  "flow_control": 0,
  "permit_wildcard": 0,
  "grat_arp": 0,
  "ipv4_acl": 0,
  "nat_dir": 0,
  "dhcp_client": 0,
  "management_services": [
    {
      "service_type": 2
    }
  ],
  "ipv4_list": [],
  "ipv6_list": [],
  "ipv6_local_auto": 0,
  "ipv6_acl": ""
}
```

以太接口配置编辑

Action: interface.ethernet.edit

请求参数:

名称	类型	范围	含义	必选	说明
slot	整数	0-255	槽位	是	
port	整数	0-255	端口	是	
status	整数	0-1	接口状态	否	0:disable;1:enable;缺省值:不修改
description	字符串	0-191	接口描述	否	空表示没有配置;缺省值:不修改
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http

					1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh 5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置; 缺省值: 不修改 1 表示 inside 2 表示 outside 3 表示 bidirection
ipv4_addr	Ipv4 address		Ipv4 地址	否	空表示没有配置;缺省值:不修改
ipv4_mask	Ipv4 address		Ipv4 掩码	否	空表示没有配置;缺省值:不修改
dhcp_client	整数	0-1	DHCP 启动	否	0:disable;1:enable;缺省值:不修改
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置;缺省值:不修改
ipv4_list	数组	0-16	Ipv4 地址列表	否	空表示没有配置;缺省值:不修改
Ipv6_list	数组	0-16	Ipv6 地址列表	否	空表示没有配置;缺省值:不修改
ipv6_addr	Ipv6 address		Ipv6 地址	否	空表示没有配置;缺省值:不修改
ipv6_prefix	整数	1-128	Ipv6 前缀	否	0 表示没有配置;缺省值:不修改
ipv6_anycast	整数	0-1	Ipv6 多播	否	0:disable;1:enable;缺省值:不修改
ipv6_local_auto	整数	0-1	Ipv6 自动配置	否	0:disable;1:enable;缺省值:0
ipv6_local_address	Ipv6 address		Ipv6 本地地址	否	空表示没有配置;缺省值: 空
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀	否	0 表示没有配置;缺省值:不修改
ipv6_local_anycast	整数		Ipv6 本地多播	否	0:disable;1:enable;缺省值:不修改
ipv6_acl	字符串	0-15	Ipv6 访问列表	否	0 表示没有配置;缺省值:不修改
speed	整数	0-4	速度	否	0: 10Mbit/s 1: 100Mbit/s 2: 1000Mbit/s 3: 10000Mbit/s 4: auto ;缺省值:不修改
duplexity	整数	0-2	模式	否	0: full 1: half 2: auto ;缺省值:不修改
flow_control	整数	0-1	硬件流控控制	否	0:disable;1:enable;缺省值:不修改
permit_wildcar	整数	0-1	混杂模式	否	0:disable;1:enable;缺省值:不修改

d					
grat_arp	整数	0-1	免费 ARP	否	0:disable;1:enable;缺省值:不修改
nat_dir	整数	0-3	Nat 方向	否	0 表示没有配置;缺省值:不修改 1 表示 inside 2 表示 outside 3 表示 bidirection
no_vlan_forward	整数	0-1	禁止 VLAN 转发	否	0 表示关闭, 1 表示开启

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ethernet.edit>

请求 body:

```
{
  "slot": 1,
  "port": 0,
  "description": "",
  "status": 0,
  "mac_addr": "00:0D:48:4E:46:BB",
  "duplexity": 2,
  "speed": 0,
  "flow_control": 0,
  "permit_wildcard": 0,
  "grat_arp": 0,
  "ipv4_acl": 0,
  "nat_dir": 0,
  "dhcp_client": 0,
  "management_services": [
    {
      "service_type": 2
    }
  ],
  "ipv4_list": [],
  "ipv6_list": [],
  "ipv6_local_auto": 0,
  "ipv6_acl": ""
}
```

获取接口统计信息

Action: interface.ethernet.statis

请求参数: 无

请求举例:

GET
<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ethernet.statis>

响应参数:

名称	类型	范围	含义
slot	整数	0-127	接口插槽号
port	整数	0-127	接口端口号
mtu	整数	1-1500	
speed	字符串		"10,000,000,000", 10G "1,000,000,000", 1G "100,000,000", 100M "10,000,000", 10M "auto" 自动
description	字符串		接口描述信息
ipaddr	字符串		接口 IP 地址
status	整数	0-2	0: unknown;1:down;2:up
in_unicast_pkts	字符串		接收的单播报文数
out_unicast_pkts	字符串		发送的单播报文数
in_bytes	字符串		接收的字节数
out_bytes	字符串		发送的字节数
in_error_pkts	字符串		接收的错误报文数
out_error_pkts	字符串		发送的错误报文数
in_discard_pkts	字符串		接收后丢弃的报文数
out_discard_pkts	字符串		发送后丢弃的报文数
in_bit_rate	字符串		接收速率(bps)
out_bit_rate	字符串		发送速率(bps)
in_pkt_rate	字符串		接收报文速率(pps)
out_pkt_rate	字符串		发送报文速率(pps)

响应举例:

```

{{
    "slot": 0,
    "port": 2,
    "mtu": 1500,
    "speed": "1,000,000,000",
    "ipaddr": "2.2.2.2/24",
    "description": "",
    "status": 2,
    "in_unicast_pkts": "1",
    "in_bytes": "0",
    "in_error_pkts": "0",
    "in_discard_pkts": "0",
    "in_bit_rate": "0",
    "in_pkt_rate": "0",
    "out_unicast_pkts": "347",
    "out_bytes": "0",
    "out_error_pkts": "0",
    "out_discard_pkts": "0",
    "out_bit_rate": "0",
    "out_pkt_rate": "0"
}}

```

```

}, {
  "slot": 0,
  "port": 3,
  "mtu": 1500,
  "speed": "1,000,000,000",
  "ipaddr": "2001::1/24",
  "description": "",
  "status": 2,
  "in_unicast_pkts": "0",
  "in_bytes": "0",
  "in_error_pkts": "0",
  "in_discard_pkts": "0",
  "in_bit_rate": "0",
  "in_pkt_rate": "0",
  "out_unicast_pkts": "3",
  "out_bytes": "0",
  "out_error_pkts": "0",
  "out_discard_pkts": "0",
  "out_bit_rate": "0",
  "out_pkt_rate": "0"
}
}

```

虚拟接口配置

虚拟接口配置列表

Action: interface.ve.list

请求参数: 无

请求举例:

GET

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ve.list>

响应参数:

名称	类型	范围	含义
port_num	整数	2-4094	端口
status	整数	0-1	接口状态
description	字符串	0-191	接口描述
management_services	数组	0-7	管理服务配置数组
service_type	整数	0-6	服务类型
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向
mac_addr	字符串		MAC 地址

ipv4_addr	Ipv4 address		Ipv4 地址
ipv4_mask	Ipv4 address		Ipv4 掩码
ipv4_acl	整数	2-198	Ipv4 访问列表
ipv4_list	数组	0-16	Ipv4 地址列表
Ipv6_list	数组	0-16	Ipv6 地址列表
ipv6_addr	Ipv6 address		Ipv6 地址
ipv6_prefix	整数	1-128	Ipv6 前缀
ipv6_anycast	整数	0-1	Ipv6 多播
ipv6_local_auto	整数	0-1	Ipv6 自动配置
ipv6_local_addr	Ipv6 address		Ipv6 本地地址
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀
ipv6_local_anycast	整数		Ipv6 本地多播
ipv6_acl	字符串	0-15	Ipv6 访问列表
nat_dir	整数	0-3	Nat 方向
permit_wildcard	整数	0-1	混杂模式
grat_arp	整数	0-1	免费 ARP
no_vlan_forward	整数	0-1	禁止 VLAN 转发

响应举例:

```
[
  {
    "port_num": 100,
    "description": "",
    "status": 1,
    "mac_addr": "000D.486D.367D",
    "permit_wildcard": 0,
    "grat_arp": 0,
    "nat_dir": 0,
    "ipv6_nat_dir": 0,
    "ipv4_acl": 0,
    "management_services": [
      {
        "service_type": 2
      }
    ],
    "ipv4_list": [
      {
        "ipv4_addr": "192.168.10.1",
        "ipv4_mask": "255.255.255.0"
      }
    ],
    "ipv6_list": [],
    "ipv6_local_auto": 0,
    "ipv6_acl": ""
  },
  {
```

```

"port_num": 101,
"description": "",
"status": 1,
"mac_addr": "000D.486D.367C",
"permit_wildcard": 0,
"grat_arp": 0,
"nat_dir": 0,
"ipv6_nat_dir": 0,
"ipv4_acl": 0,
"management_services": [
  {
    "service_type": 2
  }
],
"ipv4_list": [
  {
    "ipv4_addr": "101.0.0.193",
    "ipv4_mask": "255.255.255.0"
  },
  {
    "ipv4_addr": "190.168.10.193",
    "ipv4_mask": "255.255.255.0"
  },
  {
    "ipv4_addr": "190.168.20.193",
    "ipv4_mask": "255.255.255.0"
  }
],
"ipv6_list": [
  {
    "ipv6_addr": "2000:10::193",
    "ipv6_prefix": 64,
    "ipv6_anycast": 0
  },
  {
    "ipv6_addr": "2000:20::193",
    "ipv6_prefix": 64,
    "ipv6_anycast": 0
  }
],
"ipv6_local_auto": 1,
"ipv6_acl": ""
},
{
"port_num": 200,
"description": "",

```

```

    "status": 1,
    "mac_addr": "000D.486D.367D",
    "permit_wildcard": 0,
    "grat_arp": 0,
    "nat_dir": 0,
    "ipv6_nat_dir": 0,
    "ipv4_acl": 0,
    "management_services": [
      {
        "service_type": 2
      }
    ],
    "ipv4_list": [
      {
        "ipv4_addr": "192.168.20.1",
        "ipv4_mask": "255.255.255.0"
      }
    ],
    "ipv6_list": [],
    "ipv6_local_auto": 0,
    "ipv6_acl": ""
  }
]

```



Note

只有创建虚拟接口才能对虚拟接口操作

虚拟接口配置获取

Action: interface.ve.get

请求参数:

名称	类型	范围	含义	必选	备注
port_num	整数	2-4094	端口	是	

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ve.get

请求 body:

```

{
  "port_num": 100
}

```

响应参数:

名称	类型	范围	含义
----	----	----	----

port_num	整数	2-4094	端口
status	整数	0-1	接口状态
description	字符串	0-191	接口描述
management_services	数组	0-7	管理服务配置数组
service_type	整数	0-6	服务类型
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向
mac_addr	字符串		MAC 地址
ipv4_addr	Ipv4 address		Ipv4 地址
ipv4_mask	Ipv4 address		Ipv4 掩码
ipv4_acl	整数	2-198	Ipv4 访问列表
ipv4_list	数组	0-16	Ipv4 地址列表
Ipv6_list	数组	0-16	Ipv6 地址列表
ipv6_addr	Ipv6 address		Ipv6 地址
ipv6_prefix	整数	1-128	Ipv6 前缀
ipv6_anycast	整数	0-1	Ipv6 多播
ipv6_local_auto	整数	0-1	Ipv6 自动配置
ipv6_local_addr	Ipv6 address		Ipv6 本地地址
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀
ipv6_local_anycast	整数		Ipv6 本地多播
ipv6_acl	字符串	0-15	Ipv6 访问列表
nat_dir	整数	0-3	Nat 方向
permit_wildcard	整数	0-1	混杂模式
grat_arp	整数	0-1	免费 ARP
no_vlan_forward	整数	0-1	禁止 VLAN 转发

响应举例:

```
{
  "port_num": 100,
  "description": "",
  "status": 1,
  "mac_addr": "000D.486D.367D",
  "permit_wildcard": 0,
  "grat_arp": 0,
  "nat_dir": 0,
  "ipv6_nat_dir": 0,
  "ipv4_acl": 0,
  "management_services": [
    {
      "service_type": 2
    }
  ]
}
```



```

],
"ipv4_list": [
  {
    "ipv4_addr": "192.168.10.1",
    "ipv4_mask": "255.255.255.0"
  }
],
"ipv6_list": [],
"ipv6_local_auto": 0,
"ipv6_acl": ""
}

```

虚拟接口配置编辑

Action: interface.ve.edit

请求参数:

名称	类型	范围	含义	必选	说明
port_num	整数	2-4094	端口	是	对应 ve 接口序号
status	整数	0-1	接口状态	否	0:disable;1:enable;缺省值:不修改
description	字符串	0-191	接口描述	否	缺省值:不修改
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http 1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh 5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置; 缺省值: 不修改 1 表示 inside 2 表示 outside 3 表示 bidirection
ipv4_addr	Ipv4 address		Ipv4 地址	是	
ipv4_mask	Ipv4 address		Ipv4 掩码	是	
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置;缺省值:不修改
ipv4_list	数组	0-16	Ipv4 地址列表	否	缺省值:不修改
Ipv6_list	数组	0-16	Ipv6 地址列表	否	缺省值:不修改
ipv6_addr	Ipv6 address		Ipv6 地址	是	空表示没有配置;缺省值:不修改

ipv6_prefix	整数	1-128	Ipv6 前缀	是	0 表示没有配置;缺省值:不修改
ipv6_anycast	整数	0-1	Ipv6 多播	是	0:disable;1:enable;缺省值:不修改
ipv6_local_auto	整数	0-1	Ipv6 自动配置	否	0:disable;1:enable;缺省值:不修改
ipv6_local_addr	Ipv6 address		Ipv6 本地地址	否	默认值空字符串, 配置该项 ipv6_local_auto 自动变成 disable;缺省值:0
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀	否	0 表示没有配置;缺省值:不修改
ipv6_local_anycast	整数		Ipv6 本地多播	否	0:disable;1:enable;缺省值:不修改
ipv6_acl	字符串	0-15	Ipv6 访问列表	否	0 表示没有配置;缺省值:不修改
nat_dir	整数	0-3	Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection ;缺省值:不修改
permit_wildcard	整数	0-1	混杂模式	否	0:disable;1:enable;缺省值:不修改
grat_arp	整数	0-1	免费 ARP	否	0:disable;1:enable;缺省值:不修改
no_vlan_forward	整数	0-1	禁止 VLAN 转发	否	0 表示关闭, 1 表示开启

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.ve.edit>

请求 body:

```
{
  "port_num": 100,
  "description": "",
  "status": 1,
  "mac_addr": "000D.486D.367D",
  "permit_wildcard": 0,
  "grat_arp": 0,
  "nat_dir": 0,
  "ipv6_nat_dir": 0,
  "ipv4_acl": 0,
  "management_services": [
    {
      "service_type": 2
    }
  ],
  "ipv4_list": [
    {
      "ipv4_addr": "192.168.10.1",
      "ipv4_mask": "255.255.255.0"
    }
  ],
  "ipv6_list": [],
  "ipv6_local_auto": 0,
```

```
"ipv6_acl": ""
}
```

汇聚接口配置

汇聚接口配置列表

Action: interface.trunk.list

请求参数: 无

请求举例:

GET <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.trunk.list>

响应参数:

名称	类型	范围	含义
port_num	整数	1-8	端口
status	整数	0-2	接口状态
description	字符串	0-191	接口描述
management_services	数组	0-7	管理服务配置数组
service_type	整数	0-6	服务类型
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向
mac_addr	字符串		MAC 地址
ipv4_addr	Ipv4 address		Ipv4 地址
ipv4_mask	Ipv4 address		Ipv4 掩码
ipv4_acl	整数	2-198	Ipv4 访问列表
ipv4_list	数组	0-16	Ipv4 地址列表
Ipv6_list	数组	0-16	Ipv6 地址列表
ipv6_addr	Ipv6 address		Ipv6 地址
ipv6_prefix	整数	1-128	Ipv6 前缀
ipv6_anycast	整数	0-1	Ipv6 多播
ipv6_local_auto	整数	0-1	Ipv6 自动配置
ipv6_local_addr	Ipv6 address		Ipv6 本地地址
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀
ipv6_local_anycast	整数	0-1	Ipv6 本地多播
ipv6_acl	字符串	0-15	Ipv6 访问列表
nat_dir	整数	0-3	Nat 方向
permit_wildcard	整数	0-1	混杂模式
grat_arp	整数	0-1	免费 ARP
no_vlan_forward	整数	0-1	禁止 VLAN 转发

响应举例:

```
[ {
  "port_num": 8,
  "description": "",
  "status": 0,
  "mac_addr": "000D.484E.46BA",
  "permit_wildcard": 0,
  "grat_arp": 0,
  "nat_dir": 0,
  "ipv6_nat_dir": 0,
  "ipv4_acl": 0,
  "management_services": [
    {
      "service_type": 2
    }
  ],
  "ipv4_list": [
    {
      "ipv4_addr": "20.6.1.1",
      "ipv4_mask": "255.255.255.0"
    }
  ],
  "ipv6_list": [
    {
      "ipv6_addr": "2000:20:6:1::1",
      "ipv6_prefix": 64,
      "ipv6_anycast": 0
    }
  ],
  "ipv6_local_auto": 1,
  "ipv6_acl": ""
}
]
```



Note

只有创建汇聚接口才能对汇聚接口操作

汇聚接口配置获取

Action: interface.trunk.get

请求参数:

名称	类型	范围	含义	必选	备注
port_num	整数	1-8	端口	是	对应 trunk 接口序号

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.trunk.get
{
  "port_num": 8
}
```

响应参数:

名称	类型	范围	含义
port_num	整数	2-4094	端口
status	整数	0-1	接口状态
description	字符串	0-191	接口描述
management_services	数组	0-7	管理服务配置数组
service_type	整数	0-6	服务类型
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl
service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向
mac_addr	字符串		MAC 地址
ipv4_addr	Ipv4 address		Ipv4 地址
ipv4_mask	Ipv4 address		Ipv4 掩码
ipv4_acl	整数	2-198	Ipv4 访问列表
ipv4_list	数组	0-16	Ipv4 地址列表
Ipv6_list	数组	0-16	Ipv6 地址列表
ipv6_addr	Ipv6 address		Ipv6 地址
ipv6_prefix	整数	1-128	Ipv6 前缀
ipv6_anycast	整数	0-1	Ipv6 多播
ipv6_local_auto	整数	0-1	Ipv6 自动配置
ipv6_local_addr	Ipv6 address		Ipv6 本地地址
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀
ipv6_local_anycast	整数	0-1	Ipv6 本地多播
ipv6_acl	字符串	0-15	Ipv6 访问列表
nat_dir	整数	0-3	Nat 方向
permit_wildcard	整数	0-1	混杂模式
grat_arp	整数	0-1	免费 ARP
no_vlan_forward	整数	0-1	禁止 VLAN 转发

响应举例:

```
{
  "port_num": 8,
  "description": "",
  "status": 0,
  "mac_addr": "000D.484E.46BA",
  "permit_wildcard": 0,
  "grat_arp": 0,
```

```

"nat_dir": 0,
"ipv6_nat_dir": 0,
"ipv4_acl": 0,
"management_services": [
  {
    "service_type": 2
  }
],
"ipv4_list": [
  {
    "ipv4_addr": "20.6.1.1",
    "ipv4_mask": "255.255.255.0"
  }
],
"ipv6_list": [
  {
    "ipv6_addr": "2000:20:6:1::1",
    "ipv6_prefix": 64,
    "ipv6_anycast": 0
  }
],
"ipv6_local_auto": 1,
"ipv6_acl": ""
}

```

汇聚接口配置编辑

Action: interface.trunk.edit

请求参数:

名称	类型	范围	含义	必选	说明
port_num	整数	1-8	端口	是	对应 trunk 接口序号
status	整数	0-1	接口状态	否	0:disable;1:enable;缺省值:不修改
description	字符串	0-63	接口描述	否	
management_services	数组	0-7	管理服务配置数组	否	
service_type	整数	0-6	服务类型	是	0 表示 http 1 表示 http 2 表示 ping 3 表示 snmp 4 表示 ssh 5 表示 telnet 6 表示 acl
service_ipv4_acl	整数	2-198, 0	0 表示取消配置 ipv4 的 acl	否	

service_ipv6_acl	字符串	0-128	空字符串表示取消配置 ipv6 的 acl	否	
ipv6_nat_dir	整数	0-3	IPv6 的 Nat 方向	否	0 表示没有配置;缺省值:不修改 1 表示 inside 2 表示 outside 3 表示 bidirection
ipv4_addr	Ipv4 address		Ipv4 地址	是	
ipv4_mask	Ipv4 address		Ipv4 掩码	是	
ipv4_acl	整数	2-198	Ipv4 访问列表	否	0 表示没有配置;缺省值:不修改
ipv4_list	数组	0-16	Ipv4 地址列表	否	空表示没有配置;缺省值:不修改
Ipv6_list	数组	0-16	Ipv6 地址列表	否	空表示没有配置;缺省值:不修改
ipv6_addr	Ipv6 address		Ipv6 地址	是	空表示没有配置;缺省值:不修改
ipv6_prefix	整数	1-128	Ipv6 前缀	是	0 表示没有配置;缺省值:不修改
ipv6_anycast	整数	0-1	Ipv6 多播	是	0:disable;1:enable;缺省值:不修改
ipv6_local_auto	整数	0-1	Ipv6 自动配置	否	0:disable;1:enable;缺省值:不修改
ipv6_local_addr	Ipv6 address		Ipv6 本地地址	否	空表示没有配置;缺省值:不修改
ipv6_local_prefix	整数	1-128	Ipv6 本地前缀	否	0 表示没有配置;缺省值:不修改
ipv6_local_anycast	整数		Ipv6 本地多播	否	0:disable;1:enable;缺省值:不修改
ipv6_acl	字符串	0-15	Ipv6 访问列表	否	0 表示没有配置;缺省值:不修改
nat_dir	整数	0-3	Nat 方向	否	0 表示没有配置 1 表示 inside 2 表示 outside 3 表示 bidirection ;缺省值:不修改
permit_wildcard	整数	0-1	混杂模式	否	0:disable;1:enable;缺省值:不修改
grat_arp	整数	0-1	免费 ARP	否	0:disable;1:enable;缺省值:不修改
no_vlan_forward	整数	0-1	禁止 VLAN 转发	否	0 表示关闭, 1 表示开启

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=interface.trunk.edit>

请求 body:

```
{
  "port_num": 8,
  "description": "",
  "status": 0,
```

```

"mac_addr": "000D.484E.46BA",
"permit_wildcard": 0,
"grat_arp": 0,
"nat_dir": 0,
"ipv6_nat_dir": 0,
"ipv4_acl": 0,
"management_services": [
  {
    "service_type": 2
  }
],
"ipv4_list": [
  {
    "ipv4_addr": "20.6.1.1",
    "ipv4_mask": "255.255.255.0"
  }
],
"ipv6_list": [
  {
    "ipv6_addr": "2000:20:6:1::1",
    "ipv6_prefix": 64,
    "ipv6_anycast": 0
  }
],
"ipv6_local_auto": 1,
"ipv6_acl": ""
}

```

VLAN

VLAN 配置列表

Action: network.vlan.list

请求参数: 无

请求举例:

GET <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.vlan.list>

响应参数:

名称	范围	类型	含义
id	1-4094	整数	VLANID
description	0-63	字符串	描述
l2_fwd_disable	0-1	整数	二层转发开关
path_persist	0-1	整数	路径保持
mcast_limit	0-65535	整数	多播转发速率

ve_if	0-1	整数	是否建立 ve 接口
slot	0-255	整数	槽位
port	0-255	整数	端口
tagged	0-1	整数	vlan 标签

响应举例:

```

{{
    "id": 1,
    "description": "",
    "l2_fwd_disable": 1,
    "path_persist": 0,
    "mcast_limit": 4000,
    "ve_if": 0,
    "interface_list": [{
        "slot": 0,
        "port": 4,
        "tagged": 0
    }]
}, {
    "id": 100,
    "description": "test",
    "l2_fwd_disable": 1,
    "path_persist": 0,
    "mcast_limit": 4000,
    "ve_if": 1,
    "interface_list": [{
        "slot": 0,
        "port": 0,
        "tagged": 1
    }, {
        "slot": 0,
        "port": 1,
        "tagged": 1
    }]
}, {
    "id": 200,
    "description": "ve200",
    "l2_fwd_disable": 0,
    "path_persist": 0,
    "mcast_limit": 4000,
    "ve_if": 0,
    "interface_list": [{
        "slot": 0,
        "port": 2,
        "tagged": 0
    }, {

```

```

        "slot": 0,
        "port": 3,
        "tagged": 0
    }
}

```

VLAN 配置获取

Action: network.vlan.get

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-4094	VLANID	是	

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.vlan.get
```

请求 body:

```

{
  "id": 100
}

```

响应参数:

名称	范围	类型	含义
id	1-4094	整数	VLANID
description	0-63	字符串	描述
l2_fwd_disable	0-1	整数	二层转发开关
path_persist	0-1	整数	路径保持
mcast_limit	0-65535	整数	多播转发速率
ve_if	0-1	整数	是否建立 ve 接口
slot	0-255	整数	槽位
port	0-255	整数	端口
tagged	0-1	整数	vlan 标签

响应举例:

```

{
  "id": 100,
  "description": "test",
  "l2_fwd_disable": 1,
  "path_persist": 0,
  "mcast_limit": 4000,
  "ve_if": 1,
  "interface_list": [
    {
      "slot": 0,
      "port": 0,
      "tagged": 1
    },
    {
      "slot": 0,
      "port": 1,

```

```

        "tagged": 1
    }
}

```

VLAN 配置增加

Action: network.vlan.add

请求参数:

名称	范围	类型	含义	必选	备注
id	1-4094	整数	VLANID	是	
description	0-63	字符串	描述	否	
l2_fwd_disable	0-1	整数	二层转发开关	否	1 : disable 0 : enable
path_persist	0-1	整数	路径保持	否	0 : disable 1 : enable
mcast_limit	0-65535	整数	多播转发速率	否	
ve_if	0-1	整数	是否建立 ve 接口	否	1 表示建立
slot	0-255	整数	槽位	是	
port	0-255	整数	端口	是	
tagged	0-1	整数	vlan 标签	是	1 表示打 vlan 标签

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.vlan.add>

请求 body:

```

{
    "id": 100,
    "description": "test",
    "l2_fwd_disable": 1,
    "path_persist": 0,
    "mcast_limit": 5000,
    "ve_if": 1,
    "interface_list": [
        {
            "slot": 0,
            "port": 0,
            "tagged": 1
        },
        {
            "slot": 0,
            "port": 1,
            "tagged": 1
        }
    ]
}

```

VLAN 配置编辑

Action: network.vlan.edit

请求参数:

名称	范围	类型	含义	必选	备注
id	1-4094	整数	VLANID	是	
description	0-63	字符串	描述	否	缺省值:不修改
l2_fwd_disable	0-1	整数	二层转发开关	否	1:disable 0:enable 缺省值:不修改
path_persist	0-1	整数	路径保持	否	0 : disable 1 : enable 缺省值:0
mcast_limit	0-65535	整数	多播转发速率	否	缺省值:5000
ve_if	0-1	整数	是否建立 ve 接口	否	1 表示建立 缺省值:不修改
slot	0-255	整数	槽位	是	
port	0-255	整数	端口	是	
tagged	0-1	整数	vlan 标签	否	1 表示打 vlan 标签 缺省值:不修改

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.vlan.edit>

请求 body:

```
{
  "id": 100,
  "description": "test",
  "l2_fwd_disable": 1,
  "path_persist": 0,
  "mcast_limit": 5000,
  "ve_if": 1,
  "interface_list": [{
    "slot": 0,
    "port": 0,
    "tagged": 1
  }, {
    "slot": 0,
    "port": 1,
    "tagged": 1
  }]
}
```

VLAN 配置删除

Action: network.vlan.del

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-4094	VLANID	是	

请求举例:

```
POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.vlan.del
```

请求 body:

```
{
  "id": 100
}
```

汇聚(TRUNK)

TRUNK 配置列表

Action: network.trunk.list

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.trunk.list
```

响应参数:

名称	类型	范围	含义
id	整数	1-8	TRUNK id
type	整数	0-1	TRUNK 类型
status	整数	0-1	状态
slot	整数	1-255	接口插槽
port	整数	1-255	接口序号
enable	整数	0-1	是否加入 trunk
mode	整数	0-1	接口模式
timeout	整数	0-1	超时模式
priority	整数	1-65535	接口优先级

响应举例:

```
[[
  "id": 1,
  "type": 0,
  "status": 1,
  "interface_list": [{
    "slot": 0,
    "port": 2,
    "enable": 0
  }
]
```

```

    }, {
      "slot": 0,
      "port": 3,
      "enable": 1
    }
  ], {
    "id": 2,
    "type": 1,
    "status": 2,
    "interface_list": [
      {
        "slot": 0,
        "port": 0,
        "enable": 1,
        "mode": 0,
        "timeout": 0,
        "priority": 32767
      }, {
        "slot": 0,
        "port": 1,
        "enable": 1,
        "mode": 1,
        "timeout": 1,
        "priority": 32768
      }
    ]
  }
}

```

TRUNK 配置获取

Action: network.trunk.get

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	1-8	TRUNK id	是	

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.trunk.get

请求 body:

```

{
  "id": 2
}

```

响应参数:

名称	类型	范围	含义
id	整数	1-8	TRUNK id
type	整数	0-1	TRUNK 类型
status	整数	0-1	状态
slot	整数	1-255	接口插槽

port	整数	1-255	接口序号
enable	整数	0-1	是否加入 trunk
mode	整数	0-1	接口模式
timeout	整数	0-1	超时模式
priority	整数	1-655 35	接口优先级

响应举例:

```
{
  "id": 2,
  "type": 1,
  "status": 1,
  "interface_list": [{
    "slot": 0,
    "port": 0,
    "enable": 1,
    "mode": 0,
    "timeout": 0,
    "priority": 32767
  }, {
    "slot": 0,
    "port": 1,
    "enable": 1,
    "mode": 1,
    "timeout": 1,
    "priority": 32768
  }]
}
```

TRUNK 配置增加

Action: network.trunk.add

请求参数:

名称	类型	范围	含义	必选	说明
id	整数	1-8	TRUNK id	是	
type	整数	0-1	TRUNK 类型	是	0 静态 trunk, 1 动态 trunk (lacp)
status	整数	0-1	状态	否	0: down 1:up
slot	整数	1-255	接口插槽	否	
port	整数	1-255	接口序号	否	
enable	整数	0-1	是否加入 trunk	否	0: 不加入 trunk 1:加入 trunk
mode	整数	0-1	接口模式	否	0: passive 1: active
timeout	整数	0-1	超时模式	否	0: short 模式 1: long 模式
priority	整数	1-65535	接口优先级	否	默认 32768

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.trunk.add>

请求 body:

```
{
  "id": 2,
  "type": 1,
  "interface_list": [{
    "slot": 0,
    "port": 0,
    "enable": 1,
    "mode": 0,
    "timeout": 0,
    "priority": 32766
  }, {
    "slot": 0,
    "port": 1,
    "enable": 1,
    "mode": 0,
    "timeout": 0,
    "priority": 32765
  }
]
```

TRUNK 配置编辑

Action: network.trunk.edit

请求参数:

名称	类型	范围	含义	必选	说明
id	整数	1-8	TRUNK id	是	
type	整数	0-1	TRUNK 类型	是	0 静态 trunk, 1 动态 trunk (lacp)
status	整数	0-1	状态	否	0: down 1:up;缺省值:不修改
interface_list	数组		trunk 成员接口列表	是	
slot	整数	1-255	接口插槽	是	
port	整数	1-255	接口序号	是	
enable	整数	0-1	是否加入 trunk	否	0: 不加入 trunk 1:加入 trunk
mode	整数	0-1	接口模式	否	0: passive 1: active
timeout	整数	0-1	超时模式	否	0: short 模式 1: long 模式
priority	整数	1-65535	接口优先级	否	默认 32768

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.trunk.edit>

请求 body:

```
{
  "id": 2,
  "type": 1,
  "interface_list": [{
```



```

        "slot": 0,
        "port": 0,
        "enable": 1,
        "mode": 0,
        "timeout": 0,
        "priority": 32766
    }, {
        "slot": 0,
        "port": 1,
        "enable": 1,
        "mode": 0,
        "timeout": 0,
        "priority": 32765
    }
]
}

```

TRUNK 配置删除

Action: network.trunk.del

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	1-8	TRUNK id	是	

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.trunk.del>

请求 body:

```

{
    "id": 2
}

```

时间对象

时间对象配置列表

Action: system.timerange.list

请求参数: 无

请求举例:

GET

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.timerange.list>

响应参数:

名称	类型	范围	含义
name	字符	1-63	时间对象名字

	串		
type	整数	0-2	时间类型
day_list	整数	0-6 1-31	天列表
time_list			时间列表
start	整数	0-86400	开始时间
end	整数	0-86400	结束时间

响应举例:

```

[[
  {
    "name": "daytest",
    "type": 0,
    "time_list": [
      {
        "start": 0,
        "end": 10800
      }, {
        "start": 14400,
        "end": 14400
      }, {
        "start": 18000,
        "end": 21600
      }, {
        "start": 43200,
        "end": 86400
      }
    ]
  }, {
    "name": "weektest",
    "type": 1,
    "day_list": [0, 2, 3, 4, 5, 6],
    "time_list": [
      {
        "start": 3600,
        "end": 10800
      }, {
        "start": 10800,
        "end": 18000
      }
    ]
  }, {
    "name": "montest",
    "type": 2,
    "day_list": [1, 3, 4, 5, 6, 7, 8, 9, 10, 23, 24, 25, 26, 27, 28, 29, 30, 31],
    "time_list": [
      {
        "start": 3600,
        "end": 10800
      }, {
        "start": 18000,
        "end": 28800
      }
    ]
  }
]

```

```
    }  
  }  
}
```

时间对象配置获取

Action: system.timerange.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-63	时间对象名字	是	key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.timerange.get

请求 body:

```
{  
  "name": "weektest"  
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-63	时间对象名字
type	整数	0-2	时间类型
day_list	整数	0-6 1-31	天列表
time_list			时间列表
start	整数	0-86400	开始时间
end	整数	0-86400	结束时间

响应举例:

```
{  
  "name": "weektest",  
  "type": 1,  
  "day_list": [0, 2, 3, 4, 5, 6],  
  "time_list": [{  
    "start": 3600,  
    "end": 10800  
  }, {  
    "start": 10800,  
    "end": 18000  
  }]  
}
```

时间对象配置增加

Action: system.timerange.add

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

name	字符串	1-63	时间对象名字	是	key
type	整数	0-2	时间类型	是	0: 每天 1: 每周 2: 每月
day_list	整数	0-6 1-31	天列表	是	如果 type=1, 列表值 0-6, 表示周日到周六 如果 type=2, 列表值 1-31, 表示 1 号到 31 号
time_list			时间列表		一天的时间段
start	整数	0-86400	开始时间	是	必须 3600 整数倍
end	整数	0-86400	结束时间	是	必须 3600 整数倍

请求举例:

POST

<http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.timerange.add>

请求 body:

```
{
  "name": "weektest",
  "type": 1,
  "day_list": [0, 3, 4, 6],
  "time_list": [{
    "start": 3600,
    "end": 10800
  }, {
    "start": 10800,
    "end": 18000
  }]
}
```

时间对象配置编辑

Action: system.timerange.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-63	时间对象名字	是	key
type	整数	0-2	时间类型	是	0: 每天 1: 每周 2: 每月
day_list	整数	0-6 1-31	天列表	是	如果 type=1, 列表值 0-6, 表示周日到周六 如果 type=2, 列表值 1-31, 表示 1 号到 31 号
time_list			时间列表		一天的时间段
start	整数	0-86400	开始时间	是	必须 3600 整数倍
end	整数	0-86400	结束时间	是	必须 3600 整数倍

请求举例:

POST <http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.timerange.edit>

请求 body:

```
{
  "name": "weektest",
  "type": 1,
  "day_list": [0, 3, 4, 6],
  "time_list": [{
```

```

        "start": 3600,
        "end": 10800
    }, {
        "start": 10800,
        "end": 18000
    }
}

```

时间对象配置删除

Action: system.timerange.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-63	时间对象名字	是	key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=system.timerange.del

请求 body:

```

{
    "name": "weektest"
}

```

IPv4 标准访问列表

IPv4 标准访问列表列表

Action: acl.ipv4.std.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.list

响应参数:

名称	类型	范围	含义
description	字符串	0-191	访问列表描述
id	整数	2-99	访问列表号
item_list	列表		多个访问列表规则组成

访问列表规则:

名称	类型	范围	含义
sequence	整数	1-2000	访问列表号
acl_action	整数	0-2	匹配后动作 0: deny; 1: permit; 2: no-vlan-forward

sec_ip	字符串	Ipv4 地址	源网络地址
sec_mask	字符串	Ipv4 掩码	源网络掩码
hits	字符串		命中数

响应举例:

```
[{
  "description": "",
  "id": 3,
  "item_list": [{
    "sequence": 10,
    "acl_action": 0,
    "src_ip": "0.0.0.0",
    "src_mask": "255.255.255.255",
    "hits": "0"
  }]
}]
```

IPv4 标准访问列表获取

Action: acl.ipv4.std.get

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-99	访问列表号	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.get>
 请求 Body:

```
{
  "id": 2
}
```

IPv4 标准访问列表增加

Action: acl.ipv4.std.item.add

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-99	访问列表号	是	
sequence	整数	1-2000	访问列表号	是	
al_action	整数	0-2	动作	否	缺省值: 0
src_ip	字符串	ipv4 地址	源网络地址	否	缺省值: 0.0.0.0
src_mask	字符串	Ipv4 掩码	源网络掩码	否	缺省值: 255.255.255.255

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.item.a>

dd

请求 body:

```
{
  "id": 3,
  "sequence": 10,
  "acl_action": 0,
  "src_ip": "0.0.0.0",
  "src_mask": "255.255.255.255"
}
```

IPv4 标准访问列表删除

Action: acl.ipv4.std.item.del

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-99	访问列表号	是	
sequence	整数	1-2000	访问列表号	否	如果"sequence"缺失, 删除所有表项

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.item.del

请求 body:

```
{
  "id": 2,
  "sequence": 2
}
```

IPv4 标准访问列表编辑

Action: acl.ipv4.std.item.edit

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-99	访问列表号	是	
sequence	整数	1-2000	访问列表号	是	
acl_action	整数	0-2	动作	否	缺省值: 0
src_ip	字符串	Ipv4 地址	源网络地址	否	缺省值: 不修改
src_mask	字符串	Ipv4 掩码	源网络掩码	否	缺省值: 不修改

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.item.edit

请求 body:

```
{
  "id": 3,
```

```

"sequence": 10,
"acl_action": 0,
"src_ip": "0.0.0.0",
"src_mask": "255.255.255.255"
}

```

IPv4 标准访问列表描述设置

Action: acl.ipv4.std.desc.set

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	2-99	访问列表号	是	
description	字符串	0-191	访问列表描述	是	

请求举例:

```

post
http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.std.desc.set

```

请求 body:

```

{
  "id": "2",
  "description": "adc"
}

```

IPv4 扩展访问列表

IPv4 扩展访问列表列表

Action: acl.ipv4.ext.list

请求参数:无

请求举例:

```

GET
http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.list

```

响应参数:

名称	类型	范围	含义
description	字符串	0-191	访问列表描述
id	整数	100-198	访问列表号
Item_list	列表		访问控制列表

访问控制列表:

名称	类型	范围	含义
sequence	整数	1-2000	访问列表号
acl_action	整数	0-2	动作 0: deny; 1: permit; 2: no-vlan-forward
protocol	整数	0-17	协议
src_ip	字符串	Ipv4 地址	源网络地址

icmp_type	整数	0-254	Icmp 报文类型
icmp_code	整数	0-254	Icmp code 码
src_port_min	整数	1-65535	源端口最小值
src_port_max	整数	1-65535	源端口最大值
src_mask	字符串	Ipv4 掩码	源网络掩码
dst_ip	字符串	Ipv4 地址	目的网络地址
dst_mask	字符串	Ipv4 掩码	目的网络掩码
dst_port_min	整数	1-65535	目的端口最小值
dst_port_max	整数	1-65535	目的端口最大值
ip_fragments	整数	0-1	IP 分片
vlan_id	整数	1-4094	vlan id
dscp	整数	1-63	DSCP 值
tcp_established	整数	0-1	TCP 会话建立
description	整数	1-191	访问列表描述
timerange	字符串	1-191	时间对象
hits	整数		命中数

响应举例:

```
[{
  "description": "ip",
  "id": 100,
  "item_list": [{
    "sequence": 20,
    "acl_action": 1,
    "protocol": 0,
    "src_ip": "2.0.0.0",
    "src_mask": "0.255.255.255",
    "dst_ip": "0.0.0.0",
    "dst_mask": "255.255.255.255",
    "ip_fragments": 1,
    "vlan_id": 2,
    "dscp": 1,
    "hits": "0"
  }]
}]
```

IPv4 扩展访问列表获取

Action: acl.ipv4.ext.get

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	100-198	访问列表号	是	

请求举例:

POST

```

http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.get
{
  "id": 100
}

```

响应参数:

名称	类型	范围	含义
description	字符串	0-191	访问列表描述
id	整数	100-198	访问列表号
Item_list	列表		访问控制列表

访问控制列表:

名称	类型	范围	含义
sequence	整数	1-2000	访问列表号
acl_action	整数	0-2	动作 0: deny; 1: permit; 2: no-vlan-forward
protocol	整数	0-17	协议
src_ip	字符串	Ipv4 地址	源网络地址
icmp_type	整数	0-254	Icmp 报文类型
icmp_code	整数	0-254	Icmp code 码
src_port_min	整数	1-65535	源端口最小值
src_port_max	整数	1-65535	源端口最大值
src_mask	字符串	Ipv4 掩码	源网络掩码
dst_ip	字符串	Ipv4 地址	目的网络地址
dst_mask	字符串	Ipv4 掩码	目的网络掩码
dst_port_min	整数	1-65535	目的端口最小值
dst_port_max	整数	1-65535	目的端口最大值
ip_fragments	整数	0-1	IP 分片
vlan_id	整数	1-4094	vlan id
dscp	整数	1-63	DSCP 值
tcp_established	整数	0-1	TCP 会话建立
description	整数	1-191	访问列表描述
timerange	字符串	1-191	时间对象
hits	整数		命中数

响应举例:

```

{
  "description": "ip",
  "id": 100,
  "item_list": [{
    "sequence": 20,
    "acl_action": 1,
    "protocol": 0,
    "src_ip": "2.0.0.0",
    "src_mask": "0.255.255.255",
    "dst_ip": "0.0.0.0",
    "dst_mask": "255.255.255.255",
    "ip_fragments": 1,

```

```

        "vlan_id": 2,
        "dscp": 1,
        "hits": "0"
    }]
}

```

IPv4 扩展访问列表增加

Action: acl.ipv4.ext.item.add

请求参数:

名称	类型	范围	含义	必选	备注
description	字符串	1-191	访问列表描述	否	
id	整数	100-198	访问列表号	是	
sequence	整数	1-2000	访问列表号	是	
acl_action	整数	0-2	动作	否	动作 0: deny; 1: permit; 2: no-vlan-forward
protocol	整数	0/1/6/17	协议	否	0: ip; 1: icmp; 6: tcp; 17:udp
src_ip	字符串	Ipv4 地址	源网络地址	否	缺省值: "0.0.0.0" 全 0 表示匹配任何地址
src_mask	字符串	Ipv4 地址	源网络掩码	否	缺省值: "255.255.255.255" 反掩码: 全 1 表示匹配任何地址, 全 0 表示主机地址
icmp_type	整数	0-254	Icmp 报文类型	否	协议 protocol 是 ICMP 的时候, 才有效
icmp_code	整数	0-254	Icmp code 码	否	协议 protocol 是 ICMP 的时候, 才有效
src_port_min	整数	1-65535	源端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
src_port_max	整数	1-65535	源端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_ip	字符串	Ipv4 地址	目的网络地址	否	缺省值: "0.0.0.0" 全 0 表示匹配任何地址
dst_mask	字符串	Ipv4 掩码	目的网络掩码	否	缺省值: "255.255.255.255" 反掩码: 全 1 表示匹配任何地址, 全 0 表示主机地址
dst_port_min	整数	1-65535	目的端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_port_max	整数	1-65535	目的端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
ip_fragments	整数	0-1	IP 分片	否	
vlan_id	整数	1-4094	vlan id	否	
dscp	整数	1-63	DSCP 值	否	
tcp_established	整数	0-1	TCP 会话建立	否	协议 protocol 是 TCP 的时候, 才有效
description	整数	1-191	访问列表描述	否	
timerange	字符串	1-191	时间对象	否	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.item.add

请求 body:

```
{
```

```

"description": "ip",
"id": 100,
"sequence": 20,
"acl_action": 1,
"protocol": 0,
"src_ip": "2.0.0.0",
"src_mask": "0.255.255.255",
"dst_ip": "0.0.0.0",
"dst_mask": "255.255.255.255",
"ip_fragments": 1,
"vlan_id": 2,
"dscp": 1
}

```

IPv4 扩展访问列表编辑

Action: acl.ipv4.ext.item.edit

请求参数:

名称	类型	范围	含义	必选	备注
description	字符串	1-191	访问列表描述	否	
id	整数	100-198	访问列表号	是	
sequence	整数	1-2000	访问列表号	是	
acl_action	整数	0-2	动作	否	动作 0: deny; 1: permit; 2: no-vlan-forward
protocol	整数	0-17	协议	否	
src_ip	字符串	Ipv4 地址	源网络地址	否	缺省值: "0.0.0.0" 全 0 表示匹配任何地址
src_mask	字符串	Ipv4 地址	源网络掩码	否	缺省值: "255.255.255.255" 反掩码: 全 1 表示匹配任何地址, 全 0 表示主机地址
icmp_type	整数	0-254	Icmp 报文类型	否	协议 protocol 是 ICMP 的时候, 才有效
icmp_code	整数	0-254	Icmp code 码	否	协议 protocol 是 ICMP 的时候, 才有效
src_port_min	整数	1-65535	源端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
src_port_max	整数	1-65535	源端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_ip	字符串	Ipv4 地址	目的网络地址	否	缺省值: "0.0.0.0" 全 0 表示匹配任何地址
dst_mask	字符串	Ipv4 掩码	目的网络掩码	否	缺省值: "255.255.255.255" 反掩码: 全 1 表示匹配任何地址, 全 0 表示主机地址
dst_port_min	整数	1-65535	目的端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_port_max	整数	1-65535	目的端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
ip_fragments	整数	0-1	IP 分片	否	
vlan_id	整数	1-4094	vlan id	否	
dscp	整数	1-63	DSCP 值	否	

tcp_established	整数	0-1	TCP 会话建立	否	协议 protocol 是 TCP 的时候，才有效
description	整数	1-191	访问列表描述	否	
timerange	字符串	1-191	时间对象	否	

请求举例：

POST
 http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.item.e
 dit

请求 body：

```
{
  "description": "ip",
  "id": 100,
  "sequence": 20,
  "acl_action": 1,
  "protocol": 0,
  "src_ip": "2.0.0.0",
  "src_mask": "0.255.255.255",
  "dst_ip": "0.0.0.0",
  "dst_mask": "255.255.255.255",
  "ip_fragments": 1,
  "vlan_id": 2,
  "dscp": 1
}
```

IPv4 扩展访问列表删除

Action: acl.ipv4.ext.item.del

请求参数：

名称	类型	范围	含义	必选	备注
id	整数	100-198	访问列表号	是	
sequence	整数	1-2000	访问列表号	是	如果"sequence"缺失，删除所有表项；

请求举例：

POST
 http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.item.d
 el

请求 body：

```
{
  "id": 100,
  "sequence": 20
}
```

IPv4 扩展访问列表描述设置

Action: acl.ipv4.ext.desc.set

请求参数:

名称	类型	范围	含义	必选	备注
id	整数	100-198	访问列表号	是	
description	字符串	0-191	访问列表描述	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv4.ext.desc.s
et

请求 body:

```
{  
  "id": 100,  
  "description": "adc"  
}
```

IPv6 访问列表

IPv6 访问列表列表

Action: **acl.ipv6.ext.list**

请求参数:无

请求举例:

GET http://10.2.20.35/adcapiv2.0/?authkey=a209da70aa9e6dcdca81bb93c59cf0&action=acl.ipv6.ext.list

响应参数:

名称	类型	范围	含义
description	字符串	0-191	IPV6 访问列表描述
name	字符串	长度 1-191	IPV6 访问列表名
item_list	列表		多个访问列表规则组成

访问列表规则:

名称	类型	范围	含义
sequence	整数	1-2000	访问列表号
acl_action	整数	0-2	匹配后动作 0: deny; 1: permit; 2: no-vlan-forward
src_ip	字符串	IPv6 地址	源网络地址
src_mask	字符串	IPv6 掩码	源网络掩码
hits	字符串		命中数
protocol	整数		协议类型, UDP:17; IPV6:41; ICMP:58; TCP:6

ip_fragments	整数	0-1	IP 分片; 0: 关闭; 1: 开启
dst_ip	字符串	IPv6 地址	目的网络地址
dst_mask	字符串	IPv6 掩码	目的网络掩码
src_port_min	整数	1-65535	源端口最小值
src_port_max	整数	1-65535	源端口最大值
dst_port_min	整数	1-65535	目的端口最小值
dst_port_max	整数	1-65535	目的端口最大值

响应举例:

```
[
  {
    "description": "",
    "name": "hhhh",
    "item_list": [
      {
        "sequence": 222,
        "acl_action": 0,
        "protocol": 6,
        "src_ip": ":::",
        "src_mask": 0,
        "src_port_min": 1,
        "src_port_max": 65535,
        "dst_ip": ":::",
        "dst_mask": 0,
        "dst_port_min": 1,
        "dst_port_max": 65535,
        "ip_fragments": 1,
        "tcp_established": 0,
        "hits": "0"
      }
    ]
  }
]
```

IPv6 访问列表获取

Action: acl.ipv6.ext.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	IPV6 访问 列表名	是	

请求举例:

POST http://10.2.20.35/adcap/v2.0/?authkey={{authkey}}&action=acl.ipv6.ext.get

请求 Body:

```
{
  "name": "hhhh"
}
```

IPv6 访问列表增加

Action: acl.ipv6.ext.item.add

请求参数:

名称	类型	范围	含义	必选	备注
description	字符串	0-191	IPV6 访问列表描述	否	
name	字符串	长度 1-191	IPV6 访问列表名	是	
sequence	整数	1-2000	访问列表号	是	
acl_action	整数	0-2	匹配后动作	否	动作 0: deny; 1: permit ; 2 : no-vlan-forward
src_ip	字符串	IPv6 地址	源网络地址	否	
src_mask	字符串	IPv6 掩码	源网络掩码	否	
protocol	整数		协议类型	否	UDP:17; IPV6:41; ICMP:58; TCP:6
ip_fragments	整数	0-1	IP 分片; 0: 关闭; 1: 开启	否	0: 关闭; 1: 开启
dst_ip	字符串	IPv6 地址	目的网络地址	否	
dst_mask	字符串	IPv6 掩码	目的网络掩码	否	
src_port_min	整数	1-65535	源端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
src_port_max	整数	1-65535	源端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_port_min	整数	1-65535	目的端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效

dst_port_max	整数	1-65535	目的端口最大值	否	协议 protocol 是 TCP/UDP 的时候,才有效
--------------	----	---------	---------	---	-------------------------------

请求举例:

POST http://10.2.20.35/adcapiv2.0/?authkey={{authkey}}&action=acl.ipv6.ext.item.add

请求 body:

```
{
    "description": "123",
    "name": "hhhh",
    "sequence": 222,
    "acl_action": 0,
    "protocol": "tcp",
    "src_ip": "::",
    "src_mask": 0,
    "src_port_min": 1,
    "src_port_max": 65535,
    "dst_ip": "::",
    "dst_mask": 0,
    "dst_port_min": 1,
    "dst_port_max": 65535,
    "ip_fragments": 0,
    "hits": "0"
}
```

IPv6 访问列表删除

Action: acl.ipv6.ext.item.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	IPV6 访问列表名	是	
sequence	整数	1-2000	访问列表号	是	

请求举例:

POST
http://10.2.20.35/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv6.ext.item.del

请求 body:

```
{
    "name": "2",
    "sequence": 2
}
```

IPv6 访问列表编辑

Action: acl.ipv6.ext.item.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	IPv6 访问列表名	是	
description	字符串	0-191	访问列表描述	是	
sequence	整数	1-2000	访问列表号	是	
acl_action	整数	0-2	匹配后动作	否	动作 0: deny; 1: permit ; 2 : no-vlan-forward
src_ip	字符串	IPv6 地址	源网络地址	否	
src_mask	字符串	IPv6 掩码	源网络掩码	否	
protocol	整数		协议类型	否	UDP:17; IPv6:41; ICMP:58; TCP:6
ip_fragments	整数	0-1	IP 分片; 0: 关闭; 1: 开启	否	0: 关闭; 1: 开启
dst_ip	字符串	IPv6 地址	目的网络地址	否	
dst_mask	字符串	IPv6 掩码	目的网络掩码	否	
src_port_min	整数	1-65535	源端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
src_port_max	整数	1-65535	源端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_port_min	整数	1-65535	目的端口最小值	否	协议 protocol 是 TCP/UDP 的时候, 才有效
dst_port_max	整数	1-65535	目的端口最大值	否	协议 protocol 是 TCP/UDP 的时候, 才有效

请求举例:

POST

http://10.2.20.35/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv6.ext.item.edit

请求 body:

```
{  
  "description": "",  
  "name": "hhhh",
```

```

"sequence": 222,
"acl_action": 0,
"protocol": "tcp",
"src_ip": "::",
"src_mask": 0,
"src_port_min": 1,
"src_port_max": 65535,
"dst_ip": "::",
"dst_mask": 0,
"dst_port_min": 1,
"dst_port_max": 65535,
"ip_fragments": 0
}

```

IPv6 访问列表描述设置

Action: acl.ipv6.ext.desc.set

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-191	IPv6 访问列表名	是	
description	字符串	0-191	访问列表描述	是	

请求举例:

```

post
http://10.2.20.35/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=acl.ipv6.ext.desc.set

```

请求 body:

```

{
  "name": "hhhh"
  "description": "5555"
}

```

NAT 地址转换

NAT 地址池

NAT 地址池列表

Action: nat.pool.list

请求参数: 无

请求举例:

GEThttp://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.list

响应参数:

名称	类型	范围	含义
name	字符串	1-191	地址池名字
ip_type	整数	0-1	0:IPv4, 1:IPv6
global_gateway	IPv4/IPv6		网关 IP 地址
vrid	整数	0-8	VRRP 的 ID
ip_rr	整数	0-1	IP 选择轮询开关

响应举例:

```
[
  {
    "name": "nat",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0
  },
  {
    "name": "pop3_nat",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0
  },
  {
    "name": "dx",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 8,
    "ip_rr": 1
  },
  {
    "name": "lt",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 8,
    "ip_rr": 0
  },
  {
    "name": "test",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0
  },
]
```

```
{
  "name": "test_new_add_1",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat21",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-200",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-201",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-202",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-203",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-204",
  "ip_type": 0,
  "global_gateway": "",
```

```
"vrid": 0,
"ip_rr": 0
},
{
  "name": "nat-vip-205",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-206",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-207",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-208",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-209",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
  "name": "nat-vip-210",
  "ip_type": 0,
  "global_gateway": "",
  "vrid": 0,
  "ip_rr": 0
},
{
```

```

    "name": "nat-vip-211",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0
  },
  {
    "name": "dx-v6",
    "ip_type": 1,
    "global_gateway": "",
    "vrid": 8,
    "ip_rr": 0
  },
  {
    "name": "lt-v6",
    "ip_type": 1,
    "global_gateway": "2018::1",
    "vrid": 8,
    "ip_rr": 0
  },
  {
    "name": "6t",
    "ip_type": 1,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0
  },
  {
    "name": "te",
    "ip_type": 1,
    "global_gateway": "4::4",
    "vrid": 1,
    "ip_rr": 0
  }
]

```

NAT 地址池获取

Action: nat.pool.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	地址池名字	是	

请求举例:

GEThttp://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.get

请求 body:

```
{
  "name": "ipv4pool"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-191	地址池名字
ip_type	整数	0-1	0:IPv4, 1:IPv6
global_gateway	IPv4/IPv6		地址池全局网关 IP 地址
vrid	整数	0-8	VRRP 的 ID
ip_rr	整数	0-1	IP 选择轮询开关
member_list	数组		地址池成员列表
start_ip_addr	IPv4/IPv6		地址池成员起始 IP 地址
end_ip_addr	IPv4/IPv6		地址池成员结束 IP 地址
netmask	IPv4 掩码/IPv6 前缀长度		地址池成员掩码(IPv4)或前缀长度(IPv6)
gateway	IPv4/IPv6		地址池成员网关 IP 地址

响应举例:

```
{
  "pool": {
    "name": "ipv4_snat_pool_api_test_20",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0,
    "member_list": [
      {
        "start_ip_addr": "192.168.112.9",
        "end_ip_addr": "192.168.112.9",
        "netmask": "255.255.255.0",
        "gateway": "192.168.112.9"
      },
      {
        "start_ip_addr": "192.168.112.10",
        "end_ip_addr": "192.168.112.15",
        "netmask": "255.255.255.0",
        "gateway": ""
      }
    ]
  }
}
```

NAT 地址池增加

Action: nat.pool.add

请求参数:

名称	类型	范围	含义	必选	备注
pool	object		地址池对象	是	
name	字符串	1-191	地址池名字	是	地址池名字
ip_type	整数	0-1	0:IPv4, 1:IPv6	否	缺省为 0
global_gateway	IPv4/IPv6		地址池全局网关 IP 地址	否	缺省为空
vrid	整数	0-8	VRRP 的 ID	否	缺省为 0
ip_rr	整数	0-1	IP 选择轮询开关	否	0 表示关, 1 表示开, 缺省为 0
member_list	数组		地址池成员列表	否	缺省为空
start_ip_addr	IPv4/IPv6		地址池成员起始 IP 地址	是	
end_ip_addr	IPv4/IPv6		地址池成员结束 IP 地址	是	
netmask	IPv4 掩码/IPv6 前缀长度		地址池成员掩码 (IPv4)或前缀长 度(IPv6)	是	
gateway	IPv4/IPv6		地址池成员网关 IP 地址	否	缺省为空

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.add>

请求 body:

```
{
  "pool":{
    "name": "ipv4_snat_pool_test1",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0,
    "member_list": [
      {
        "start_ip_addr": "192.168.112.9",
        "end_ip_addr": "192.168.112.9",
        "netmask": "255.255.255.0",
        "gateway": "192.168.112.9"
      },
      {
        "start_ip_addr": "192.168.112.10",
        "end_ip_addr": "192.168.112.15",
        "netmask": "255.255.255.0",
        "gateway": ""
      }
    ]
  }
}
```

}

NAT 地址池编辑

Action: nat.pool.edit

请求参数:

名称	类型	范围	含义	必选	备注
pool	object		地址池对象	是	
name	字符串	1-191	地址池名字	是	地址池名字
ip_type	整数	0-1	0:IPv4, 1:IPv6	否	缺省为不改变
global_gateway	IPv4/IPv6		地址池全局网关 IP 地址	否	缺省为不改变
vrid	整数	0-8	VRRP 的 ID	否	缺省为不改变
ip_rr	整数	0-1	IP 选择轮询开关	否	0 表示关, 1 表示开,缺省为不改变
member_list	数组		地址池成员列表	否	缺省为不改变
start_ip_addr	IPv4/IPv6		地址池成员起始 IP 地址	是	
end_ip_addr	IPv4/IPv6		地址池成员结束 IP 地址	是	
netmask	IPv4 掩码/IPv6 前缀长度		地址池成员掩码(IPv4)或前缀长度(IPv6)	是	
gateway	IPv4/IPv6		地址池成员网关 IP 地址	否	缺省为空

请求举例:

POST <http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.edit>

请求 body:

```
{
  "pool":{
    "name": "ipv4_snat_pool_test1",
    "ip_type": 0,
    "global_gateway": "",
    "vrid": 0,
    "ip_rr": 0,
    "member_list": [
      {
        "start_ip_addr": "192.168.112.9",
        "end_ip_addr": "192.168.112.9",
        "netmask": "255.255.255.0",
        "gateway": "192.168.112.9"
      }
    ]
  },
}
```

```

    {
      "start_ip_addr": "192.168.112.8",
      "end_ip_addr": "192.168.112.16",
      "netmask": "255.255.255.0",
      "gateway": "192.168.112.9"
    },
    {
      "start_ip_addr": "192.168.112.117",
      "end_ip_addr": "192.168.112.117",
      "netmask": "255.255.255.255",
      "gateway": ""
    }
  ]
}

```

NAT 地址池删除

Action: nat.pool.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	地址池名字	是	地址池名字

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.del

请求 body:

```

{
  "name": "ipv4_snat_pool_test1"
}

```

NAT 地址池统计信息

NAT 地址池统计信息列表

Action: nat.pool.stats.list

请求参数:

名称	类型	范围	含义	必选	备注
ip_type	整数	0-1	0:IPv4, 1:IPv6	否	缺省为 0
start	整数	0-99999	开始返回数据的编号	否	缺省为 0
limit	整数	0-99999	最多返回数据的条数	否	缺省为 30; 0 表示 unlimited
searchbk	字符串	0-127	过滤条件表达式	否	可选, 默认""

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.stats.list&ip_type=0&start=1&limit=3

响应参数:

名称	类型	范围	含义
pool_num	整数	0-99999	地址池的数量 (总数)
match_num	整数	0-99999	匹配过滤条件的数量
limit_num	整数	0-99999	限制返回的数量
total	整数	0-99999	地址池的数量 (总数)
ip_type	整数	0-1	0:IPv4, 1:IPv6
data	数组		地址池统计信息数组
name	字符串	1-191	地址池名字
port	整数		地址池当前连接数量
used	整数		地址池累计连接数量
freed	整数		地址池累计释放连接数量

响应举例:

```
{
  "pool_num": 19,
  "match_num": 0,
  "limit_num": 3,
  "total": 19,
  "ip_type": 0,
  "data": [
    {
      "name": "pop3_nat",
      "port": 0,
      "used": 0,
      "freed": 0
    },
    {
      "name": "dx",
      "port": 0,
      "used": 1828408,
      "freed": 1828408
    },
    {
      "name": "lt",
      "port": 0,
      "used": 788268,
      "freed": 788268
    }
  ]
}
```

NAT 地址池统计信息获取

Action: nat.pool.stats.list

请求参数:

名称	类型	范围	含义	必选	备注
ip_type	整数	0-1	0:IPv4, 1:IPv6	否	缺省为 0
name	字符串	1-191	地址池名字	是	
start	整数	0-99999	开始返回数据的编号	否	缺省为 0
limit	整数	0-99999	最多返回数据的条数	否	缺省为 30; 0 表示 unlimited
searchbk	字符串	0-127	过滤条件表达式	否	可选, 默认""

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.stats.list&ip_type=0&start=1&limit=3&name=dx

响应参数:

名称	类型	范围	含义
addr_num	整数	0-99999	地址的数量 (总数)
match_num	整数	0-99999	匹配的数量
limit_num	整数	0-99999	限制返回的数量
total	整数	0-99999	地址的数量 (总数)
ip_type	整数	0-1	0:IPv4, 1:IPv6
data	数组		地址统计信息数组
addr	IPv4/IPv6 地址	1-191	IPv4/IPv6 地址
port	整数		地址当前连接数量
used	整数		地址累计连接数量
freed	整数		地址累计释放连接数量

响应举例:

```
{
  "addr_num": 6,
  "match_num": 0,
  "limit_num": 3,
  "total": 6,
  "ip_type": 0,
  "data": [
    {
      "addr": "192.168.17.2",
      "port": 0,
      "used": 305126,
      "freed": 305126
    }
  ]
}
```

```

    },
    {
      "addr": "192.168.17.3",
      "port": 0,
      "used": 304953,
      "freed": 304953
    },
    {
      "addr": "192.168.17.4",
      "port": 0,
      "used": 304803,
      "freed": 304803
    }
  ]
}

```

NAT 地址池统计信息清除

Action: nat.pool.stats.clear

请求参数:

名称	类型	范围	含义	必选	备注
ip_type	整数	0-1	0:IPv4, 1:IPv6	否	缺省为 1
name	字符串	1-191	地址池名字	否	指定名称时清除指定地址池的统计信息, 为指定时清除指定类型的所有地址池的统计信息

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool.stats.clear&ip_type=0&name=dx

响应参数: 无

NAT 地址池组

NAT 地址池组列表

Action: nat.pool_group.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool_group.list

响应参数:

名称	类型	范围	含义
name	字符串	1-191	地址池组名字
vrrp_id	整数	0-8	VRRP 的 ID
pool_list	字符串数组	1-25	地址池数组

响应举例:

```
[ {
  "name" : "ipv6group",
  "vrrp_id" : 0,
  "pool_list" : [ "ipv6pool" ]
}, {
  "name" : "ipv4group",
  "vrrp_id" : 1,
  "pool_list" : [ "ipv4pool", "ipv4pool2" ]
}]
```

NAT 地址池组获取

Action: nat.pool_group.get

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool_group.get

响应参数:

名称	类型	范围	含义
name	字符串	1-191	地址池组名字
vrrp_id	整数	0-8	VRRP 的 ID
pool_list	字符串数组	1-25	地址池数组

响应举例:

```
{
  "name" : "ipv4group",
  "vrrp_id" : 1,
  "pool_list" : [ "ipv4pool", "ipv4pool2" ]
}
```

NAT 地址池组增加

Action: nat.pool_group.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	地址池组名字	是	

vrrp_id	整数	0-8	VRRP 的 ID	否	
pool_list	字符串数组	1-25	地址池数组	否	
pool_list 成员	字符串	1-31	地址池名字	是	

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool_group.add

请求 body:

```
{
  "name": "ipv4group",
  "vrrp_id": 1,
  "pool_list": [ "ipv4pool", "ipv4pool2" ]
}
```

NAT 地址池组编辑

Action: nat.pool_group.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	地址池组名字	是	
vrrp_id	整数	0-8	VRRP 的 ID	否	
pool_list	字符串数组	1-25	地址池数组	否	
pool_list 成员	字符串	1-31	地址池名字	是	

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool_group.edit

请求 body:

```
{
  "name": "ipv4group",
  "vrrp_id": 1,
  "pool_list": [ "ipv4pool", "ipv4pool2" ]
}
```

NAT 地址池组删除

Action: nat.pool_group.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	地址池组名字	是	

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.pool_group.del

请求 body:

```
{
  "name": "ipv4group"
}
```


}

静态 NAT(包括 PAT)

静态 NAT 列表

Action: nat.static.list

请求参数:

名称	类型	范围	含义	必选	备注
type	整数	0-2	类型	否	可选, 默认 0 0: all 1: nat 2: pat
limit	整数	0-1000	读取数	否	可选, 默认 0 0 表示读取所有配置
index	整数	0-9999	序号	否	可选, 默认 0
searchbk	字符串	0-127	表达式	否	可选, 默认 ""

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.list
```

请求 body:

```
{  
  "type": 0,  
  "limit": 0,  
  "index": 30,  
  "searchbk": ""  
}
```

响应参数:

名称	类型	范围	含义
origin_ip	IPv4		内网源地址
nat_ip	IPv4		外网全局地址
protocol	枚举	0, 1, 6, 17	协议
origin_port	整数	1-65535	内网源端口
nat_port	整数	1-65535	外网端口
port_num	整数	1-65535	映射端口数
vrrp_id	整数	0-8	VRRP 的 ID
application	整数	0-5	PAT 中表明应用协议
priority	整数	0-1	优先级
if_type	枚举	0-3	接口类型
if_slot	整数	0-127	ETH 插槽
if_port	整数	0-127	ETH 接口
if_ve	整数	2-4094	VE 接口
if_trunk	整数	1-8	Trunk 接口

disable	bool	0-1	使能
description	字符串	0-191	描述
index	整数	0-9999	序号

响应举例:

```
[ {
  "origin_ip" : "10.0.0.1",
  "nat_ip" : "20.0.0.100",
  "protocol" : 6,
  "origin_port" : 100,
  "nat_port" : 200,
  "port_num" : 1,
  "vrrp_id" : 0,
  "application" : 2,
  "if_slot" : 0,
  "if_port" : 1,
  "if_type" : 1,
  "disable" : 0,
  "description" : "pattest",
  "index" : 0
}, {
  "origin_ip" : "30.1.1.1",
  "nat_ip" : "40.1.1.1",
  "protocol" : 17,
  "origin_port" : 233,
  "nat_port" : 233,
  "port_num" : 2,
  "vrrp_id" : 0,
  "application" : 1,
  "if_type" : 0,
  "disable" : 1,
  "description" : "nattest",
  "index" : 1
}, {
  "origin_ip" : "203.1.1.1",
  "nat_ip" : "203.1.1.2",
  "protocol" : 1,
  "port_num" : 1,
  "vrrp_id" : 0,
  "if_ve" : 2,
  "if_type" : 2,
  "disable" : 0,
  "description" : "paticmp",
  "index" : 2
}, {
  "origin_ip" : "102.1.1.1",
```

```

"nat_ip" : "103.1.1.1",
"protocol" : 0,
"port_num" : 1,
"vrrp_id" : 0,
"priority" : 0,
"if_slot" : 0,
"if_port" : 2,
"if_type" : 1,
"disable" : 0,
"description" : "natstest",
"index" : 3
}, {
"origin_ip" : "104.1.1.1",
"nat_ip" : "105.1.1.1",
"protocol" : 0,
"port_num" : 1,
"vrrp_id" : 0,
"priority" : 1,
"if_ve" : 2,
"if_type" : 2,
"disable" : 0,
"description" : "vetestnat",
"index" : 4
}]

```

静态 NAT 获取

Action: nat.static.get

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535
port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0
application	整数	0-5	PAT 中表明应用协议	否	PAT 中有效, 1 表示 FTP; 2 表示 RTSP; 3 表示 PPTP; 4 表示 MMS; 5 表示 SIP, 默认 0
priority	整数	0-1	优先级	否	NAT 中有效, 0 低优先级; 1 高优

					优先级，默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能 1: ETH 2: VE 3: TRUNK 默认 0
if_slot	整数	0-127	ETH 插槽	否	if_type 为 1 有效，默认 0
if_port	整数	0-127	ETH 接口	否	if_type 为 1 有效，默认 0
if_ve	整数	2-4094	VE 接口	否	if_type 为 2 有效
if_trunk	整数	1-8	Trunk 接口	否	if_type 为 3 有效
disable	bool	0-1	使能	否	默认 0
description	字符串	0-191	描述	否	默认空串
index	整数	0-9999	序号	否	翻页使用，默认 0

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.get

静态 NAT 获取，当 protocol 为 0

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0

请求 body:

```
{
  "origin_ip": "102.1.1.1",
  "nat_ip": "103.1.1.1",
  "protocol": 0
}
```

响应参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能 1: ETH 2: VE 3: TRUNK 默认 0
if_slot	整数	0-127	ETH 插槽	否	if_type 为 1 有效，默认 0
if_port	整数	0-127	ETH 接口	否	if_type 为 1 有效，默认 0

description	字符串	0-191	描述	否	默认空串
disable	bool	0-1	使能	否	默认 0
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0
priority	整数	0-1	优先级	否	NAT 中有效，0 低优先级；1 高优先级，默认 0

响应举例：

```
{
  "origin_ip" : "102.1.1.1",
  "nat_ip" : "103.1.1.1",
  "protocol" : 0,
  "vrrp_id" : 0,
  "priority" : 0,
  "if_slot" : 0,
  "if_port" : 2,
  "if_type" : 1,
  "disable" : 0,
  "description" : "nattest"
}
```

静态 NAT 获取，当 **protocol** 为 1

请求参数：

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT，非 0 表示 PAT，PAT 中：1 表示 icmp 转换，6 表示 tcp 转换，17 表示 udp 转换，默认 0

请求 body：

```
{
  "origin_ip" : "203.1.1.1",
  "nat_ip" : "203.1.1.2",
  "protocol" : 1
}
```

响应参数：

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT，非 0 表示 PAT，PAT 中：1 表示 icmp 转换，6 表示 tcp 转换，17 表示 udp 转换，默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能 1: ETH 2: VE 3: TRUNK

					默认 0
description	字符串	0-191	描述	否	默认空串
disable	bool	0-1	使能	否	默认 0
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0

响应举例:

```
{
  "origin_ip" : "203.1.1.1",
  "nat_ip" : "203.1.1.2",
  "protocol" : 1,
  "vrrp_id" : 0,
  "if_ve" : 2,
  "if_type" : 2,
  "disable" : 0,
  "description" : "paticmp"
}
```

静态 NAT 获取, 当 protocol 为 6 或 17

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535
port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535

请求 body:

```
{
  "origin_ip" : "30.1.1.1",
  "nat_ip" : "40.1.1.1",
  "protocol" : 17,
  "origin_port" : 233,
  "nat_port" : 233,
  "port_num" : 2
}
```

响应参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能

					1: ETH 2: VE 3: TRUNK 默认 0
description	字符串	0-191	描述	否	默认空串
disable	bool	0-1	使能	否	默认 0
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535
port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535

响应举例:

```
{
  "origin_ip" : "30.1.1.1",
  "nat_ip" : "40.1.1.1",
  "protocol" : 17,
  "origin_port" : 233,
  "nat_port" : 233,
  "port_num" : 2,
  "vrrp_id" : 0,
  "application" : 1,
  "if_type" : 0,
  "disable" : 1,
  "description" : "natstest"
}
```

静态 NAT 增加

Action: nat.static.add

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535
port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0
application	整数	0-5	PAT 中表明应用协议	否	PAT 中有效, 1 表示 FTP; 2 表示 RTSP; 3 表示 PPTP; 4 表示 MMS; 5 表示 SIP, 默认 0
priority	整数	0-1	优先级	否	NAT 中有效, 0 低优先级; 1 高

					优先级，默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能 1: ETH 2: VE 3: TRUNK 默认 0
if_slot	整数	0-127	ETH 插槽	否	if_type 为 1 有效，默认 0
if_port	整数	0-127	ETH 接口	否	if_type 为 1 有效，默认 0
if_ve	整数	2-4094	VE 接口	否	if_type 为 2 有效
if_trunk	整数	1-8	Trunk 接口	否	if_type 为 3 有效
disable	bool	0-1	使能	否	默认 0
description	字符串	0-191	描述	否	默认空串
index	整数	0-9999	序号	否	翻页使用，默认 0

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.add

请求 body:

```
{
  "origin_ip": "30.1.1.1",
  "nat_ip": "40.1.1.1",
  "protocol": 17,
  "origin_port": 233,
  "nat_port": 233,
  "port_num": 2,
  "vrrp_id": 0,
  "application": 1,
  "if_type": 0,
  "disable": 1,
  "description": "natstest"
}
```

请求参数参考静态 NAT 获取

静态 NAT 编辑

Action: nat.static.edit

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	
protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT, PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535

port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535
vrrp_id	整数	0-8	VRRP 的 ID	否	默认 0
application	整数	0-5	PAT 中表明应用协议	否	PAT 中有效, 1 表示 FTP; 2 表示 RTSP; 3 表示 PPTP; 4 表示 MMS; 5 表示 SIP, 默认 0
priority	整数	0-1	优先级	否	NAT 中有效, 0 低优先级; 1 高优先级, 默认 0
if_type	枚举	0-3	接口类型	否	0: 不启动接口选择功能 1: ETH 2: VE 3: TRUNK 默认 0
if_slot	整数	0-127	ETH 插槽	否	if_type 为 1 有效, 默认 0
if_port	整数	0-127	ETH 接口	否	if_type 为 1 有效, 默认 0
if_ve	整数	2-4094	VE 接口	否	if_type 为 2 有效
if_trunk	整数	1-8	Trunk 接口	否	if_type 为 3 有效
disable	bool	0-1	使能	否	默认 0
description	字符串	0-191	描述	否	默认空串
index	整数	0-9999	序号	否	翻页使用, 默认 0

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.edit

请求 body:

```
{
  "origin_ip": "30.1.1.1",
  "nat_ip": "40.1.1.1",
  "protocol": 17,
  "origin_port": 233,
  "nat_port": 233,
  "port_num": 2,
  "vrrp_id": 0,
  "application": 1,
  "if_type": 0,
  "disable": 1,
  "description": "natstest"
}
```

静态 NAT 删除

Action: nat.static.del

请求参数:

名称	类型	范围	含义	必选	备注
origin_ip	IPv4		内网源地址	是	
nat_ip	IPv4		外网全局地址	是	

protocol	枚举	0, 1, 6, 17	协议	是	0 表示 NAT, 非 0 表示 PAT ,PAT 中: 1 表示 icmp 转换, 6 表示 tcp 转换, 17 表示 udp 转换, 默认 0
origin_port	整数	1-65535	内网源端口	是(PAT)	PAT 中有效, 取值 1-65535
nat_port	整数	1-65535	外网端口	是(PAT)	PAT 中有效, 取值 1-65535
port_num	整数	1-65535	映射端口数	是(PAT)	PAT 中有效, 取值 1-65535

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.del

1) protocol 为 0

请求 body:

```
{
  "origin_ip": "102.1.1.1",
  "nat_ip": "103.1.1.1",
  "protocol": 0
}
```

2) protocol 为 1

请求 body:

```
{
  "origin_ip": "203.1.1.1",
  "nat_ip": "203.1.1.2",
  "protocol": 1
}
```

1) protocol 为 6 或 17

请求 body:

```
{
  "origin_ip": "30.1.1.1",
  "nat_ip": "40.1.1.1",
  "protocol": 17,
  "origin_port": 233,
  "nat_port": 233,
  "port_num": 2
}
```

静态 NAT 统计信息列表

Action: nat.static.statis

请求参数:

名称	类型	范围	含义	必选	备注
type	整数	0-2	类型	否	可选, 默认 0 0: all 1: nat 2: pat
limit	整数	0-1000	读取数	否	可选, 默认 0 0 表示读取所有配置
index	整数	0-9999	序号	否	可选, 默认 0
searchbk	字符串	0-127	表达式	否	可选, 默认 ""

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.statis

请求 body:

```
{
  "type": 0,
  "limit": 1,
  "index": 30,
  "searchbk": "1"
}
```

响应参数:

名称	类型	范围	含义
origin_ip	IPv4		内网源地址
nat_ip	IPv4		外网全局地址
protocol	枚举	0, 1, 6, 17	协议
origin_port	整数	1-65535	内网源端口
nat_port	整数	1-65535	外网端口
port_num	整数	1-65535	映射端口数
port_used	整数	0- 4294967295	端口使用量
port_total_used	整数	0- 4294967295	端口使用累计量
port_total_freed	整数	0- 4294967295	端口释放累计量
port_total_hits	整数	0- 4294967295	端口命中累计量

响应举例:

```
{
  "static_nat_list": [ {
    "origin_ip": "1.1.1.12",
    "nat_ip": "2.2.2.22",
    "protocol": 0,
    "port_used": 0,
    "port_total_used": 0,
    "port_total_freed": 0,
    "port_total_hits": 0
  }, {
    "origin_ip": "100.1.1.1",
    "nat_ip": "200.1.1.1",
    "protocol": 6,
    "origin_port": 22,
    "nat_port": 22,
    "port_num": 3,
    "port_used": 0,
    "port_total_used": 0,
    "port_total_freed": 0,
    "port_total_hits": 0
  } ],
  "total": 2
}
```

静态 NAT 统计信息清除

Action: nat.static.clear

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.static.clear

网络 NAT

网络 NAT 列表

Action: nat.network.list

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.network.list

响应参数:

名称	类型	范围	含义
name	字符串	1-191	网络 NAT 名称
origin_ip	IPv4		内网源地址
origin_mask	IPv4		内网源地址掩码
nat_ip	IPv4		外网全局地址
nat_mask	IPv4		外网全局地址掩码
number	整数	1-180000	转换个数
vrrp_id	整数	0-8	VRRP 的 ID

响应举例:

```
[ {
  "name": "netnat",
  "origin_ip": "10.0.1.0",
  "origin_mask": "255.255.255.0",
  "nat_ip": "220.22.22.0",
  "nat_mask": "255.255.255.0",
  "number": 10,
  "vrrp_id": 1
}, {
  "name": "net2",
  "origin_ip": "20.1.1.1",
  "origin_mask": "255.255.255.0",
  "nat_ip": "30.1.1.1",
  "nat_mask": "255.255.255.0",
  "number": 10,
  "vrrp_id": 1
} ]
```

网络 NAT 获取

Action: nat.network.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	网络 NAT 名称	是	

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.network.get
```

请求 body:

```
{
  "name": "netnat"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-191	网络 NAT 名称
origin_ip	IPv4		内网源地址
origin_mask	IPv4		内网源地址掩码
nat_ip	IPv4		外网全局地址
nat_mask	IPv4		外网全局地址掩码
number	整数	1-180000	转换个数
vrrp_id	整数	0-8	VRRP 的 ID

响应举例:

```
{
  "name": "netnat",
  "origin_ip": "10.0.1.0",
  "origin_mask": "255.255.255.0",
  "nat_ip": "220.22.22.0",
  "nat_mask": "255.255.255.0",
  "number": 10,
  "vrrp_id": 1
}
```

网络 NAT 增加

Action: nat.network.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	网络 NAT 名称	是	必选
origin_ip	IPv4		内网源地址	是	必选
origin_mask	IPv4		内网源地址掩码	是	必选

nat_ip	IPv4		外网全局地址	是	必选
nat_mask	IPv4		外网全局地址掩码	是	必选
number	整数	1-180000	转换个数	是	必选，默认 1
vrrp_id	整数	0-8	VRRP 的 ID	否	可选，默认 0

请求举例：

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.network.add

请求 body：

```
{
  "name" : "netnat3",
  "origin_ip" : "50.0.1.0",
  "origin_mask" : "255.255.255.0",
  "nat_ip" : "215.22.22.0",
  "nat_mask" : "255.255.255.0",
  "number" : 20,
  "vrrp_id" : 1
}
```

网络 NAT 编辑

Action: nat.network.edit

请求参数：

名称	类型	范围	含义	必选	备注
name	字符串	1-191	网络 NAT 名称	是	必选
origin_ip	IPv4		内网源地址	是	必选
origin_mask	IPv4		内网源地址掩码	是	必选
nat_ip	IPv4		外网全局地址	是	必选
nat_mask	IPv4		外网全局地址掩码	是	必选
number	整数	1-180000	转换个数	是	必选，默认 1
vrrp_id	整数	0-8	VRRP 的 ID	否	可选，默认 0

请求举例：

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.network.edit

请求 body：

```
{
  "name" : "netnat3",
  "origin_ip" : "50.0.1.0",
  "origin_mask" : "255.255.255.0",
  "nat_ip" : "215.22.22.0",
  "nat_mask" : "255.255.255.0",
}
```

```
"number" : 20,
"vrrp_id" : 1
}
```

网络 NAT 删除

Action: nat.network.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	网络 NAT 名称	是	

请求举例:

```
POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.network.del
```

请求 body:

```
{
  "name" : "netnat3"
}
```

NAT 映射

NAT 映射列表

Action: nat.map.list

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.map.list
```

响应参数:

名称	类型	范围	含义
acl	整数	2-99	Ipv4 访问列表
nat_pool	字符串	1-191	Nat 地址池

响应举例:

```
[ {
  "acl" : 2,
  "nat_pool" : "ipv4pool2"
}, {
  "acl" : 3,
  "nat_pool" : "ipv4pool"
} ]
```

NAT 映射增加

Action: nat.map.add

请求参数:

名称	类型	范围	含义	必选	备注
acl	整数	2-99	Ipv4 访问列表	是	key
nat_pool	字符串	1-191	Nat 地址池	是	key

请求举例:

```
POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.map.add
```

请求 body:

```
{
  "acl" : 3,
  "nat_pool" : "ipv4pool"
}
```

NAT 映射删除

Action: nat.map.del

请求参数:

名称	类型	范围	含义	必选	备注
acl	整数	2-99	Ipv4 访问列表	是	key
nat_pool	字符串	1-191	Nat 地址池	是	key

请求举例:

```
POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.map.del
```

请求 body:

```
{
  "acl" : 3,
  "nat_pool" : "ipv4pool"
}
```

IPv6 NAT 映射列表

Action: nat.ipv6.map.list

请求参数: 无

请求举例:

```
GET
```

```
http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.ipv6.map.list
```

响应参数:

名称	类型	范围	含义
acl_name	字符串	1-191	Ipv6 访问列表

nat_pool	字符串	1-191	Nat 地址池
----------	-----	-------	---------

响应举例:

```
[{
  "acl_name" : "mgmtacl",
  "nat_pool" : "aclipv6"
}]
```

IPv6 NAT 映射增加

Action: nat.ipv6.map.add

请求参数:

名称	类型	范围	含义	必选	备注
acl_name	字符串	1-191	Ipv6 访问列表	是	key
nat_pool	字符串	1-191	Nat 地址池	是	key

请求举例:

POST

http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.ipv6.map.add

请求 body:

```
{
  "acl_name" : "mgmtacl",
  "nat_pool" : "aclipv6"
}
```

IPv6 NAT 映射删除

Action: nat.ipv6.map.del

请求参数:

名称	类型	范围	含义	必选	备注
acl_name	字符串	1-191	Ipv6 访问列表	是	key
nat_pool	字符串	1-191	Nat 地址池	是	key

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.ipv6.map.del

请求 body:

```
{
  "acl_name" : "mgmtacl",
  "nat_pool" : "aclipv6"
}
```

地址转换策略

地址转换策略列表

Action: nat.policy.list

请求参数: 无

请求举例:

GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.policy.list

响应参数:

名称	类型	范围	含义
name	字符串	1-191	策略名
acl	整数	2-198	访问列表
nat_pool	字符串	1-191	地址池
pool_bind_list	数组	0-64	地址池绑定列表
member_acl	整数	2-198	访问列表
member_nat_pool	字符串	1-191	地址池

响应举例:

```
[[
  {
    "name": "123",
    "acl": 2,
    "nat_pool": "pool1",
    "pool_bind_list": [{
      "member_acl": 3,
      "member_nat_pool": "pool2"
    }, {
      "member_acl": 4,
      "member_nat_pool": "pool3"
    }
  ]
}, {
  "name": "test",
  "acl": 2,
  "nat_pool": "pool2",
  "pool_bind_list": [{
    "member_acl": 3,
    "member_nat_pool": "pool3"
  }
}]
```

地址转换策略获取

Action: nat.policy.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	策略名	是	

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.policy.get

请求 body:

```
{
  "name": "123"
}
```

响应参数:

名称	类型	范围	含义
name	字符串	1-191	策略名
acl	整数	2-198	访问列表
nat_pool	字符串	1-191	地址池
pool_bind_list	数组	0-64	地址池绑定列表
member_acl	整数	2-198	访问列表
member_nat_pool	字符串	1-191	地址池

响应举例:

```
{
  "name": "123",
  "acl": 2,
  "nat_pool": "pool1",
  "pool_bind_list": [{
    "member_acl": 3,
    "member_nat_pool": "pool2"
  }, {
    "member_acl": 4,
    "member_nat_pool": "pool3"
  }]
}
```

地址转换策略增加

Action: nat.policy.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	策略名	是	
acl	整数	2-198	访问列表	否	0 表示没有配置
nat_pool	字符串	1-191	地址池	否	空字符串表示没有配置
pool_bind_list	数组	0-64	地址池绑定列表	否	
member_acl	整数	2-198	访问列表	是	
member_nat_pool	字符串	1-191	地址池	是	

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.policy.add

请求 body:

```
{
  "name": "new",
  "acl": 2,
  "nat_pool": "pool1",
  "pool_bind_list": [{
    "member_acl": 3,
    "member_nat_pool": "pool2"
  }, {
    "member_acl": 4,
    "member_nat_pool": "pool3"
  }]
}
```

地址转换策略编辑

Action: nat.policy.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	策略名	是	
acl	整数	2-198	访问列表	否	0 表示没有配置
nat_pool	字符串	1-191	地址池	否	空字符串表示没有配置
pool_bind_list	数组	0-64	地址池绑定列表	否	
member_acl	整数	2-198	访问列表	是	
member_nat_pool	字符串	1-191	地址池	是	

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.policy.edit

请求 body:

```
{
  "name": "new",
  "acl": 2,
  "nat_pool": "pool1",
  "pool_bind_list": [{
    "member_acl": 3,
    "member_nat_pool": "pool2"
  }, {
    "member_acl": 4,
    "member_nat_pool": "pool3"
  }]
}
```

地址转换策略删除

Action: nat.policy.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	1-191	策略名	是	

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.policy.del
```

请求 body:

```
{
  "name": "new"
}
```

NAT 全局配置

NAT 超时参数获取

Action: nat.timeout.get

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.timeout.get
```

响应参数:

名称	类型	范围	含义
syn	整数	60-300 (秒)	半连接会话超时删除时间
tcp	整数	60-18000 (秒)	tcp 连接会话超时删除时间
udp	整数	60-18000 (秒)	udp 连接会话超时删除时间
icmp	整数	60-18000 (秒)	icmp 连接会话超时删除时间
service_list	LIST		服务超时列表
protocol	整数	0-1	协议
port	整数	1-65535	端口
timeout	整数	0-18000 (秒)	连接会话超时删除时间

响应举例:

```
{
  "syn": 180,
  "tcp": 180,
  "udp": 180,
  "icmp": 120,
  "service_list": [ {
    "protocol": 1,
```

```

    "port" : 53,
    "timeout" : 120
  },
  {
    "protocol" : 1,
    "port" : 54,
    "timeout" : 180
  }
}

```

NAT 超时参数设置

Action: nat.timeout.set

请求参数:

名称	类型	范围	含义	必选	备注
syn	整数	60-300 (秒)	半连接会话超时删除时间	否	可选, 只能配置 60 的整倍数, 缺省值:60
tcp	整数	60-18000 (秒)	tcp 连接会话超时删除时间	否	可选, 只能配置 60 的整倍数, 缺省值:300
udp	整数	60-18000 (秒)	udp 连接会话超时删除时间	否	可选, 只能配置 60 的整倍数, 缺省值:300
icmp	整数	60-18000 (秒)	icmp 连接会话超时删除时间	否	可选, 只能配置 60 的整倍数, 缺省值:60
service_list	LIST		服务超时列表	否	缺省值:不可修改
protocol	整数	0-1	协议	是	必选, 1 表示 udp 0 表示 tcp
port	整数	1-65535	端口	是	必选, 协议端口
timeout	整数	0-18000 (秒)	连接会话超时删除时间	是	必选, 只能配置 60 的整倍数

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.timeout.set
```

请求 body:

```

{
  "syn" : 180,
  "tcp" : 180,
  "udp" : 180,
  "icmp" : 120,
  "service_list" : [ {
    "protocol" : 1,
    "port" : 53,
    "timeout" : 120
  },
  {
    "protocol" : 1,
    "port" : 54,

```

```

        "timeout" : 180
    }
}

```

NAT 全局参数获取

Action: nat.global.get

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.global.get
```

响应参数:

名称	类型	范围	含义
pptp_alg	布尔型	0-1	开启 PPTP 算法
port_enable	布尔型	0-1	端口启动 NAT
gw_enable	布尔型	0-1	启动网管
nat_pool_log	布尔型	0-1	开启 snat 资源池使用率告警日志, 0 关闭, 1 开启
nat_pool_log_warn_th	整数	1-100	告警门限, 单位为百分比
nat_pool_log_warn_cth	整数	0-100	恢复告警门限日志, 单位为百分比, 0 为不恢复

响应举例:

```

{
  "pptp_alg" : 1,
  "port_enable" : 0,
  "gw_enable" : 1,
  "nat_pool_log": 1,
  "nat_pool_log_warn_th": 80,
  "nat_pool_log_warn_cth": 1
}

```

NAT 全局参数设置

Action: nat.global.set

请求参数:

名称	类型	范围	含义	必选	备注
pptp_alg	布尔型	0-1	开启 PPTP 算法		启动该功能, pptp 才能透传 nat, 默认 1
port_enable	布尔型	0-1	端口启动 NAT		该参数为 1, 接口 nat 命令才能工作, 默认 0
gw_enable	布尔型	0-1	启动网管		默认 0
nat_pool_log	布尔型	0-1	开启 snat 资源池使用率告警		0 关闭, 1 开启, 默认 0

			日志		
nat_pool_log_warn_th	整数	1-100	告警门限，单位为百分比		默认 80
nat_pool_log_warn_cth	整数	0-100	恢复告警门限日志，单位为百分比，0 为不恢复		默认 0

请求举例：

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=nat.global.set

请求 body:

```
{
  "pptp_alg": 0,
  "port_enable": 1,
  "gw_enable": 0,
  "nat_pool_log": 1,
  "nat_pool_log_warn_th": 80,
  "nat_pool_log_warn_cth": 1
}
```

地址解析

IPv4 ARP

增加静态 ARP 配置

Action: arp.ipv4.add

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv4 地址		IP 地址	是	
mac_addr	MAC 地址		MAC 地址	是	
port_type	整数	0/1/3	接口类型	否	0:表示不指定接口(默认同网段的接口) 1: 以太口,当 ARP 所在接口为 VE 接口时必须使用该参数,其它情况不需要 3:trunk 接口 缺省值:0
slot_num	整数	0-127	接口插槽号	否	当指定 port_type 为 1 时需要指定该参数 缺省值:无
port_num	整数	0-127	接口号	否	当指定 port_type 为 1 或者 3 时需要指定该参数 当 port_type 为 1 时表示以太网接口端口号 当 port_type 为 3 时表示 trunk 接口 ID

					缺省值:无
--	--	--	--	--	-------

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.add

举例 1:添加非 VE 接口的 ARP

请求 body:

```
{
  "ip_addr": "192.168.70.100",
  "mac_addr": "0000.0000.0001"
}
```

举例 2: 添加 VE 接口中 (ETH0/0)的 ARP

```
{
  "ip_addr": "192.168.10.100",
  "mac_addr": "0000.0000.0001",
  "port_type": 1,
  "slot_num": 0,
  "port_num": 0
}
```



Note

1. 添加静态 ARP 时,IP 地址必须属于设备直连路由可达的范围
2. 静态 ARP 对应直连路由属于 VE 接口时,必须指定其具体的以太网接口

获取静态 ARP 配置列表

Action: arp.ipv4.list

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.list

响应参数:

名称	类型	范围	含义
ip_addr	IPv4 地址		IP 地址
mac_addr	MAC 地址		MAC 地址
port_type	整数	1,3	接口类型:1 以太网接口 3 trunk 接口
slot_num	整数	0-127	接口插槽号
port_num	整数	0-127	当 port_type 为 1 时表示以太网接口端口号 当 port_type 为 3 时表示 trunk 接口 ID

响应举例:

```
[[
  "ip_addr": "192.168.220.229",
  "mac_addr": "0000.0000.0001"
}, {
  "ip_addr": "10.0.3.100",
  "mac_addr": "0000.0003.0001"
}, {
```

```

    "ip_addr": "10.0.2.100",
    "mac_addr": "0000.0002.0001"
  }, {
    "ip_addr": "10.0.0.100",
    "mac_addr": "0000.0001.0001",
    "port_type": 1,
    "slot_num": 0,
    "port_num": 0
  }
}

```

获取指定静态 ARP 配置

Action: arp.ipv4.get

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv4 地址		IP 地址	是	

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.get

请求 body:

```

{"ip_addr":"10.0.3.100"}

```

响应参数:

名称	类型	范围	含义
ip_addr	IPv4 地址		IP 地址
mac_addr	MAC 地址		MAC 地址
port_type	整数	1	接口类型:1 以太网接口, 3 trunk 接口
slot_num	整数	0-127	接口插槽号
port_num	整数	0-127	接口号

响应举例:

```

{
  "ip_addr": "10.0.3.100",
  "mac_addr": "0000.0003.0001"
}

```

编辑指定静态 ARP 配置

Action: arp.ipv4.edit

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv4 地址		IP 地址	是	
mac_addr	MAC 地址		MAC 地址	否	缺省值:不修改

port_type	整数	0/1/3	接口类型	是	0:表示不指定接口 1: 以太口,当 ARP 所在接口为 VE 接口时必须使用该参数,其它情况不需要 3:trunk 接口 缺省值:0
slot_num	整数	0-127	接口插槽号	否	当指定 port_type 为 1 时需要指定该参数 缺省值:无
port_num	整数	0-127	接口号	否	当指定 port_type 为 1 或者 3 时需要指定该参数 当 port_type 为 1 时表示以太网接口端口号 当 port_type 为 3 时表示 trunk 接口 ID 缺省值:无

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.edit

请求 body:

```
{
  "ip_addr":"192.168.70.100",
  "mac_addr":"0000.0070.0001"
}
```

删除指定静态 ARP 配置

Action: arp.ipv4.del

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv4 地址		IP 地址	是	

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.del
{"ip_addr":"10.0.3.100"}

获取 ARP 状态信息列表

Action: arp.ipv4.statis

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.statis

响应参数:

名称	类型	范围	含义
ip_addr	IPv4 地址		IP 地址
mac_addr	MAC 地址		MAC 地址

port_type	整数	1-4	接口类型:1 以太网接口, 3 trunk 接口, 4: 管理口
slot_num	整数	0-127	接口插槽号
port_num	整数	0-127	当 port_type 为 1 时表示 以太网接口号 当 port_type 为 3 时表示 trunk 接口号
type	字符串		ARP 类型: Static : 静态 ARP , Dynamic : 动态 ARP
age	整数	0-300	占用时间,单位秒
state	字符串		状态: "Incomplete", 未完成的 "Reachable", 可达的 "Stale", 陈旧的/过期的 "Delay", 延迟状态 "Probe", 探测状态 "Failed", 失败状态 "No ARP", 不需要解析 "Permanent", 永久的 "None" 初始状态

响应举例:

```

[[
  {
    "ip_addr": "10.0.0.100",
    "mac_addr": "0000.0001.0001",
    "port_type": 1,
    "slot_num": 0,
    "port_num": 0,
    "type": "Static",
    "age": 0,
    "state": "Permanent",
    "vlan_id": 100
  }, {
    "ip_addr": "10.0.2.100",
    "mac_addr": "0000.0003.0001",
    "port_type": 3,
    "port_num": 1,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }, {
    "ip_addr": "10.0.2.123",
    "mac_addr": "000c.29c9.678e",
    "port_type": 3,
    "port_num": 1,
    "type": "Dynamic",
    "age": 223,
    "state": "Stale"
  }, {

```

```

    "ip_addr": "10.0.3.100",
    "mac_addr": "0000.0003.0001",
    "port_type": 1,
    "slot_num": 0,
    "port_num": 2,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }, {
    "ip_addr": "192.168.220.1",
    "mac_addr": "24b8.d200.840e",
    "port_type": 4,
    "type": "Dynamic",
    "age": 74,
    "state": "Reachable"
  }
}

```

清除动态 ARP

Action: arp.ipv4.clear

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv4.clear
```

IPv6 Neighbor

增加静态 IPv6 邻居(Neighbor)

Action: arp.ipv6.add

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv6 地址		IPv6 地址	是	
mac_addr	MAC 地址		MAC 地址	是	
port_type	整数	1, 3	接口类型	是	1: 以太口,3: trunk 口
slot_num	整数	0-127	接口插槽号	否	当指定 port_type 为 1 时需要指定该参数 缺省值:无
port_num	整数	0-127	接口号	是	当指定 port_type 为 1 时表示以太网接口号 当指定 port_type 为 3 时表示 trunk 号

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.add
```

举例 1:添加以太网接口的 Neighbor

请求 body:

```
{
  "ip_addr": "2000::3",
  "mac_addr": "0000.0000.0003",
  "port_type": 1,
  "slot_num": 0,
  "port_num": 1
}
```

举例 2: 添加 TRUNK 口的 Neighbor

```
{
  "ip_addr": "2002::2",
  "mac_addr": "0000.0002.0002",
  "port_type": 3,
  "port_num": 1
}
```



1. 添加静态 IPv6 Neighbor 时,IPv6 地址必须属于设备 IPv6 直连路由可达的范围
2. 静态 IPv6 Neighbor 对应直连路由属于 VE 接口时,必须指定其具体的以太网接口

获取静态 IPv6 邻居(Neighbor)列表

Action: arp.ipv6.list

请求参数: 无

请求举例:

GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.list

响应参数:

名称	类型	范围	含义
ip_addr	IPv6 地址		IPv6 地址
mac_addr	MAC 地址		MAC 地址
port_type	整数	1, 3	接口类型:1 以太网接口, 3 trunk 接口
slot_num	整数	0-127	接口插槽号
port_num	整数	0-127	接口号

响应举例:

```
[{
  "ip_addr": "2000::2",
  "mac_addr": "0021.0000.0002",
  "port_type": 1,
  "slot_num": 0,
  "port_num": 0
}, {
  "ip_addr": "2001::2",
  "mac_addr": "0021.0000.0002",
  "port_type": 1,
```

```

        "slot_num": 0,
        "port_num": 2
    }, {
        "ip_addr": "2004::2",
        "mac_addr": "0024.0000.0002",
        "port_type": 1,
        "slot_num": 0,
        "port_num": 4
    }
}

```

获取指定静态 IPv6 邻居(Neighbor)

Action: arp.ipv6.get

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv6 地址		IPv6 地址	是	

请求举例:

```

POST      http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.get
{"ip_addr": "2000::2"}

```

响应参数:

名称	类型	范围	含义
ip_addr	IPv6 地址		IPv6 地址
mac_addr	MAC 地址		MAC 地址
port_type	整数	1, 3	接口类型: 1 以太网接口, 3 trunk 接口
slot_num	整数	0-127	接口插槽号
port_num	整数	0-127	接口号

响应举例:

```

{
    "ip_addr": "2000::2",
    "mac_addr": "0021.0000.0002",
    "port_type": 1,
    "slot_num": 0,
    "port_num": 0
}

```

编辑指定静态 IPv6 邻居(Neighbor)

Action: arp.ipv6.edit

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv6 地址		IPv6 地址	是	

mac_addr	MAC 地址		MAC 地址	否	
port_type	整数	1, 3	接口类型	是	1: 以太口,3: trunk 口
slot_num	整数	0-127	接口插槽号	否	当指定 port_type 为 1 时需要指定该参数 缺省值:无
port_num	整数	0-127	接口号	是	当指定 port_type 为 1 时表示以太网接口号 当指定 port_type 为 3 时表示 trunk 号

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.edit

请求 body:

```
{
    "ip_addr": "2000::2",
    "mac_addr": "0000.2000.0002",
    "port_type": 1,
    "slot_num": 0,
    "port_num": 0
}
```

删除指定静态 IPv6 邻居(Neighbor)

Action: arp.ipv6.del

请求参数:

名称	类型	范围	含义	必选	备注
ip_addr	IPv6 地址		IPv6 地址	是	

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.del
{"ip_addr": "2000::2"}

获取 IPV6 邻居状态信息列表

Action: arp.ipv6.statis

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.statis

响应参数:

名称	类型	范围	含义
ip_addr	IPv6 地址		IPv6 地址
mac_addr	MAC 地址		MAC 地址
port_type	整数	1, 3, 4	接口类型:1 以太网接口, 2: VE 接口, 3 trunk 接口
slot_num	整数	0-127	接口插槽号

port_num	整数	0-127	当 port_type 为 1 时表示 以太网接口号 当 port_type 为 3 时表示 trunk 接口号
type	字符串		ARP 类型: Static: 静态 IPv6 邻居(Neighbor) , Dynamic: 动态 ARP
age	整数	0-300	占用时间,单位秒
state	字符串		状态: "Incomplete", 未完成的 "Reachable", 可达的 "Stale", 陈旧的/过期的 "Delay",延迟状态 "Probe",探测状态 "Failed",失败状态 "No ARP",不需要解析 "Permanent",永久的 "None"初始状态

响应举例:

```

[
  {
    "ip_addr": "2001::2",
    "mac_addr": "0021.0000.0002",
    "port_type": 2,
    "port_num": 100,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }, {
    "ip_addr": "2004::3",
    "mac_addr": "0004.0000.0003",
    "port_type": 3,
    "port_num": 1,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }, {
    "ip_addr": "2004::2",
    "mac_addr": "0024.0000.0002",
    "port_type": 3,
    "port_num": 1,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }, {
    "ip_addr": "2000::2",
    "mac_addr": "0021.0000.0002",
    "port_type": 1,
  }
]

```

```

    "slot_num": 0,
    "port_num": 0,
    "type": "Static",
    "age": 0,
    "state": "Permanent"
  }
}

```

清除动态 IPv6 邻居(Neighbor)

Action: arp.ipv6.clear

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=arp.ipv6.clear
```

静态路由

IPv4 静态路由配置列表

Action: route.static.ipv4.list

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv4.list
```

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	0-255	管理距离, 默认为 0
description	字符串	0-191	描述

响应举例:

```

[
  {
    "destination": "192.168.0.0",
    "netmask": "255.255.0.0",
    "gateway": "192.168.70.250",
    "distance": 0,
    "description": ""
  },
  {
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 1,
  }
]

```

```
"description": ""
```

```
}]
```



配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv4 静态路由配置获取

Action: route.static.ipv4.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key
gateway	IPV4	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv4.get

请求 body:

```
{
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	0-255	管理距离, 默认为 0
description	字符串	0-191	描述

响应举例:

```
{
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 1,
    "description": ""
}
```

IPv4 静态路由配置增加

Action: route.static.ipv4.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	0-255	管理距离	否	默认值 0
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv4.add>

请求 body:

```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

IPv4 静态路由配置编辑

Action: route.static.ipv4.edit

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	0-255	管理距离	否	默认值 0
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv4.edit>

请求 body:

```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 2,
    "description": "test"
}
```

IPv4 静态路由配置删除

Action: route.static.ipv4.del

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key
gateway	IPV4	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv4.del

请求 body:

```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

IPv6 静态路由配置列表

Action: route.static.ipv6.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv6.list

响应参数:

名称	类型	范围	含义
destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关
distance	整数	0-255	管理距离

响应举例:

```
[{
    "destination": "2002::",
    "prefix_len": "64",
    "gateway": "2001::62",
    "distance": 1
}, {
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62",
```

```
"distance": 2
}]
```



配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv6 静态路由配置获取

Action: route.static.ipv6.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 0

请求举例:

POST

```
http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv6.get
{
  "destination": "2003::",
  "prefix_len": "65",
  "gateway": "2001::62"
}
```

响应参数:

名称	类型	范围	含义
destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关
distance	整数	0-255	管理距离

响应举例: {

```
"destination": "2003::",
"prefix_len": "65",
"gateway": "2001::62",
"distance": 1
}
```

IPv6 静态路由配置增加

Action: route.static.ipv6.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 0

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv6.add>

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62",
    "distance": 1
}
```

IPv6 静态路由配置编辑

Action: route.static.ipv6.edit

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 0

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv6.edit>

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62",
    "distance": 2
}
```

IPv6 静态路由配置删除

Action: route.static.ipv6.del

请求参数:

名称	类型	范围	含义	必选	说明
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ipv6.del

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62"
}
```

静态管理路由

IPv4 静态管理路由配置列表

Action: route.static.mgmt.ipv4.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv4.

list

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	1-255	管理距离
description	字符串	0-191	描述

响应举例:

```
[[
    "destination": "192.168.0.0",
    "netmask": "255.255.0.0",
    "gateway": "192.168.70.250",
```



```

    "distance": 0,
    "description": ""
  }, {
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 1,
    "description": ""
  }
}

```



配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv4 静态管理路由配置获取

Action: route.static.mgmt.ipv4.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key
gateway	IPV4	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv4.

get

请求 body:

```

{
  "destination": "201.0.21.0",
  "netmask": "255.255.255.0",
  "gateway": "30.0.0.62"
}

```

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	1-255	管理距离
description	字符串	0-191	描述

响应举例:

```

{
  "destination": "201.0.21.0",
  "netmask": "255.255.255.0",
  "gateway": "30.0.0.62",

```

```

    "distance": 1,
    "description": ""
}

```

IPv4 静态管理路由配置增加

Action: route.static.mgmt.ipv4.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	1-255	管理距离	否	默认值 1
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv4.add>

add

请求 body:

```

{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}

```

IPv4 静态管理路由配置编辑

Action: route.static.mgmt.ipv4.edit

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	1-255	管理距离	否	默认值 1
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv4.edit>

```

edit
请求 body:
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 2,
    "description": "test"
}

```

IPv4 静态管理路由配置删除

Action: route.static.mgmt.ipv4.del

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key
gateway	IPV4	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv4.

del

请求 body:

```

{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}

```

IPv6 静态管理路由配置列表

Action: route.static.mgmt.ipv6.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv6.

list

响应参数:

名称	类型	范围	含义
destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关

distance	整数	1-255	管理距离
-----------------	----	-------	------

响应举例:

```

[[
  {
    "destination": "2002::",
    "prefix_len": "64",
    "gateway": "2001::62",
    "distance": 1
  }, {
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62",
    "distance": 2
  }
]]

```



Note

配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv6 静态管理路由配置获取

Action: route.static.mgmt.ipv6.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	1-255	管理距离	否	默认值 1

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv6.

get

```

{
  "destination": "2003::",
  "prefix_len": "65",
  "gateway": "2001::62"
}

```

响应参数:

名称	类型	范围	含义
----	----	----	----

destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关
distance	整数	1-255	管理距离

响应举例: {

```
"destination": "2003::",
"prefix_len": "65",
"gateway": "2001::62",
"distance": 1
```

}

IPv6 静态路由管理由配置增加

Action: route.static.mgmt.ipv6.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	1-255	管理距离	否	默认值 1

请求举例:

POST

<http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv6.add>

add

请求 body:

```
{
  "destination": "2003::",
  "prefix_len": "65",
  "gateway": "2001::62",
  "distance": 1
}
```

IPv6 静态管理路由配置编辑

Action: route.static.mgmt.ipv6.edit

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	1-255	管理距离	否	默认值 1

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv6.

edit

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62",
    "distance": 2
}
```

IPv6 静态管理路由配置删除

Action: route.static.mgmt.ipv6.del

请求参数:

名称	类型	范围	含义	必选	说明
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.mgmt.ipv6.

del

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62"
}
```

静态业务路由

IPv4 静态业务路由配置列表

Action: route.static.ctrl.ipv4.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv4.list

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	0-255	管理距离
description	字符串	0-191	描述

响应举例:

```
{
  "destination": "192.168.0.0",
  "netmask": "255.255.0.0",
  "gateway": "192.168.70.250",
  "distance": 0,
  "description": ""
}, {
  "destination": "201.0.21.0",
  "netmask": "255.255.255.0",
  "gateway": "30.0.0.62",
  "distance": 1,
  "description": ""
}
```



配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv4 静态业务路由配置获取

Action: route.static.ctrl.ipv4.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key

gateway	IPV4	-	网关	是	Key
---------	------	---	----	---	-----

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv4.get

请求 body:

```
{
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

响应参数:

名称	类型	范围	含义
destination	IPV4	-	目的网段
netmask	IPV4 掩码	-	掩码
gateway	IPV4	-	网关
distance	整数	0-255	管理距离
description	字符串	0-191	描述

响应举例:

```
{
    "destination": "201.0.21.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 1,
    "description": ""
}
```

IPv4 静态业务路由配置增加

Action: route.static.ctrl.ipv4.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	0-255	管理距离	否	默认值 1
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv4.ad

d

请求 body:


```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

IPv4 静态业务路由配置编辑

Action: route.static.ctrl.ipv4.edit

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	
netmask	IPV4 掩码	-	掩码	是	
gateway	IPV4	-	网关	是	
distance	整数	0-255	管理距离	否	默认值 1
description	字符串	0-191	描述	否	默认空串

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv4.edi

t

请求 body:

```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62",
    "distance": 2,
    "description": "test"
}
```

IPv4 静态路业务由配置删除

Action: route.static.ctrl.ipv4.del

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV4	-	目的网段	是	Key
netmask	IPV4 掩码	-	掩码	是	Key
gateway	IPV4	-	网关	是	Key

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv4.del

请求 body:

```
{
    "destination": "201.0.22.0",
    "netmask": "255.255.255.0",
    "gateway": "30.0.0.62"
}
```

IPv6 静态业务路由配置列表

Action: route.static.ctrl.ipv6.list

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv6.list

响应参数:

名称	类型	范围	含义
destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关
distance	整数	0-255	管理距离

响应举例:

```
[[
    {
        "destination": "2002::",
        "prefix_len": "64",
        "gateway": "2001::62",
        "distance": 1
    }, {
        "destination": "2003::",
        "prefix_len": "65",
        "gateway": "2001::62",
        "distance": 2
    }
]]
```



Note

配置路由的时候本机接口的网络地址需要跟路由的下一跳在相同网段

IPv6 静态业务路由配置获取

Action: route.static.ctrl.ipv6.get

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 1

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv6.get

```
{
  "destination": "2003::",
  "prefix_len": "65",
  "gateway": "2001::62"
}
```

响应参数:

名称	类型	范围	含义
destination	IPV6	-	目的网段
prefix_len	掩码前缀	1-128	掩码
gateway	IPV6	-	网关
distance	整数	1-255	管理距离

响应举例: {

```
"destination": "2003::",
"prefix_len": "65",
"gateway": "2001::62",
"distance": 1
}
```

IPv6 静态业务路由配置增加

Action: route.static.ctrl.ipv6.add

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 1

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv6.ad
d

请求 body:

```
{  
    "destination": "2003::",  
    "prefix_len": "65",  
    "gateway": "2001::62",  
    "distance": 1  
}
```

IPv6 静态业务路由配置编辑

Action: route.static.ctrl.ipv6.edit

请求参数:

名称	类型	范围	含义	必选	备注
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key
gateway	IPV6	-	网关	是	Key
distance	整数	0-255	管理距离	否	默认值 1

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv6.edi
t

请求 body:

```
{  
    "destination": "2003::",  
    "prefix_len": "65",  
    "gateway": "2001::62",  
    "distance": 2  
}
```

IPv6 静态业务路由配置删除

Action: route.static.ctrl.ipv6.del

请求参数:

名称	类型	范围	含义	必选	说明
destination	IPV6	-	目的网段	是	Key
prefix_len	掩码前缀	1-128	掩码	是	Key

gateway	IPV6	-	网关	是	Key
---------	------	---	----	---	-----

请求举例:

POST

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=route.static.ctrl.ipv6.del

请求 body:

```
{
    "destination": "2003::",
    "prefix_len": "65",
    "gateway": "2001::62"
}
```

OSPF

OSPF 网络列表

Action: ospf.network.list

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=ospf.network.list

响应参数:

名称	类型	范围	含义
srcaddr	字符串	Ipv4 地址	网络 ip
netmask	字符串	Ipv4 掩码	网络掩码
areaid	整数	0-4294967295	区域 id

响应举例:

```
[{
    "srcaddr": "120.120.120.0",
    "netmask": "255.255.255.0",
    "areaid": "1"
}]
```

OSPF 网络增加

Action: ospf.network.add

请求参数:

名称	类型	范围	含义	必选	备注
srcaddr	字符串	Ipv4 地址	网络 ip	是	
netmask	字符串	Ipv4 掩码	网络掩码	是	
areaid	整数	0-4294967295	区域 id	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=ospf.network.add

请求 body:

```
{
  "srcaddr": "120.120.120.0",
  "netmask": "255.255.255.0",
  "areaid": "1"
}
```

OSPF 网络删除

Action: ospf.network.del

请求参数:

名称	类型	范围	含义	必选	备注
srcaddr	字符串	Ipv4 地址	网络 ip	是	
netmask	字符串	Ipv4 掩码	网络掩码	是	
areaid	整数	0-4294967295	区域 id	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=ospf.network.del
请求 body:

```
{
  "srcaddr": "122.121.121.0",
  "netmask": "255.255.255.0",
  "areaid": "1"
}
```

OSPF 状态获取

Action: ospf.status.get

请求参数: 无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=ospf.status.get

响应参数:

名称	类型	范围	含义
status	整数	0-1	状态
redistribute_static	整数	0-1	静态路由重分发
redistribute_connected	整数	0-1	直连路由重分发
redistribute_vip	整数	0-1	SLB 虚拟服务地址重分发
redistribute_default	整数	0-1	缺省路由重分发
metric_standby	整数	0-65535	备机路由度量

响应举例:

```
{
  "status": 1,
  "redistribute_static": 1,
  "redistribute_connected": 0,
}
```

```

"redistribute_vip": 1,
"redistribute_default": 0,
"metric_standby": 100
}

```

OSPF 状态设置

Action: ospf.status.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0-1	状态	是	
redistribute_static	整数	0-1	静态路由重分发	否	默认 0
redistribute_connected	整数	0-1	直连路由重分发	否	默认 0
redistribute_vip	整数	0-1	SLB 虚拟服务地址重分发	否	默认 0
redistribute_default	整数	0-1	缺省路由重分发	否	默认 0
metric_standby	整数	0-65535	备机路由度量	否	默认 0, 0: 关闭鼓掌倒换触发功能; 1-65535: 开启鼓掌倒换触发功能后的备机路由度量值

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=ospf.status.set

请求 body:

```

{
  "status": 1,
  "redistribute_static": 1,
  "redistribute_connected": 0,
  "redistribute_vip": 1,
  "redistribute_default": 0,
  "metric_standby": 100
}

```

BGP

BGP 网络列表

Action: bgp.network.list

请求参数: 无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.network.list

响应参数:

名称	类型	范围	含义
srcaddr	字符串	Ipv4 地址	网络 ip
netmask	字符串	Ipv4 掩码	网络掩码

响应举例:

```
[{
  "srcaddr": "120.120.120.0",
  "netmask": "255.255.255.0",
}]
```

BGP 网络增加

Action: bgp.network.add

请求参数:

名称	类型	范围	含义	必选	备注
srcaddr	字符串	Ipv4 地址	网络 ip	是	
netmask	字符串	Ipv4 掩码	网络掩码	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.network.add

请求 body:

```
{
  "srcaddr": "120.120.120.0",
  "netmask": "255.255.255.0"
}
```

BGP 网络删除

Action: bgp.network.del

请求参数:

名称	类型	范围	含义	必选	备注
srcaddr	字符串	Ipv4 地址	网络 ip	是	
netmask	字符串	Ipv4 掩码	网络掩码	是	

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.network.del

请求 body:

```
{
  "srcaddr": "122.121.121.0",
  "netmask": "255.255.255.0"
}
```

BGP 邻居列表

Action: bgp.neighbor.list

请求参数: 无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.neighbor.list

响应参数:

名称	类型	范围	含义
neighbor_id	字符串	Ipv4 地址	邻居
remote_as_id	整数	1-4294967295	邻居 id

响应举例:

```
{
  "neighbor_id": "13.14.15.16",
  "remote_as_id": 2
}
```

BGP 邻居增加

Action: bgp.neighbor.add

请求参数:

名称	类型	范围	含义	必选	备注
neighbor_id	字符串	Ipv4 地址	邻居	是	
remote_as_id	整数	1-4294967295	邻居 id	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.neighbor.add

请求 body:

```
{
  "neighbor_id": "13.14.15.16",
  "remote_as_id": 2
}
```

BGP 邻居删除

Action: bgp.neighbor.del

请求参数:

名称	类型	范围	含义	必选	备注
neighbor_id	字符串	Ipv4 地址	邻居	是	
remote_as_id	整数	1-4294967295	邻居 id	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=436406305dc37d81c09a75740d4253&action=bgp.neighbor.del

请求 body:

```
{
  "neighbor_id": "13.14.15.16",
  "remote_as_id": 2
}
```

BGP 状态获取

Action: bgp.status.get

请求参数: 无

请求举例:

GET http://192.168.70.73/adcapiv2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=bgp.status.get

响应参数:

名称	类型	范围	含义
status	整数	0-1	状态
as-id	整数	1-4294967295	AS-ID
redistribute_static	整数	0-1	静态路由重分发
redistribute_connected	整数	0-1	直连路由重分发
redistribute_vip	整数	0-1	SLB 虚拟服务地址重分发
redistribute_ospf	整数	0-1	OSPF 路由重分发

响应举例:

```
{
  "status": 1,
  "as-id": 1,
  "redistribute_static": 1,
  "redistribute_connected": 0,
  "redistribute_ospf": 0,
  "redistribute_vip": 1
}
```

BGP 状态设置

Action: bgp.status.set

请求参数:

名称	类型	范围	含义	必选	备注
status	整数	0-1	状态	是	
as-id	整数	1-4294967295	AS-ID	是	
redistribute_static	整数	0-1	静态路由重分发	否	缺省值:0
redistribute_connected	整数	0-1	直连路由重分发	否	缺省值:0
redistribute_vip	整数	0-1	SLB 虚拟服务地址重分发	否	缺省值:0
redistribute_ospf	整数	0-1	OSPF 路由重分发	否	缺省值:0

请求举例:

POST http://192.168.70.73/adcapiv2.0/?authkey=a5c149123abf65d3f6407bd95f88dd&action=bgp.status.set

请求 body:

```
{
  "status": 1,
  "as-id": 1,
  "redistribute_static": 1,
  "redistribute_connected": 0,
}
```

```

"redistribute_ospf": 0,
"redistribute_vip": 1
}

```

网络安全 DDOS

获取网络安全 DDOS 配置

Action: network.ddos.get

请求参数: 无

请求举例:

```
GET http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.ddos.get
```

响应参数:

名称	类型	范围	含义
ip_option	整数	0-1	阻断含有 ip 的 option 报文
land_attack	整数	0-1	Land 攻击
dead_ping	整数	0-1	死 ping
fragment	整数	0-1	阻断分片报文
tcp_no_flag	整数	0-1	阻断没有 flag 的 tcp 报文
tcp_fault_flag	整数	0-1	阻断错误 flag 的 tcp 报文
tcp_syn_frag	整数	0-1	阻断对 tcp 的 syn 报文分了片的报文

响应举例:

```

{
  "ip_option" : 0,
  "land_attack" : 1,
  "dead_ping" : 0,
  "fragment" : 1,
  "tcp_no_flag" : 1,
  "tcp_fault_flag" : 1,
  "tcp_syn_frag" : 1
}

```



Note

Tcp 攻击包含三种攻击方式 tcp_syn_frag, tcp_fault_flag, tcp_no_flag

设置网络安全 DDOS 配置

Action: network.ddos.set

请求参数:

名称	类型	范围	含义	必选	备注
ip_option	整数	0-1	阻断含有 ip 的 option 报文	否	缺省值:0
land_attack	整数	0-1	Land 攻击	否	缺省值:0
dead_ping	整数	0-1	死 ping	否	缺省值:0
fragment	整数	0-1	阻断分片报文	否	缺省值:0
tcp_no_flag	整数	0-1	阻断没有 flag 的 tcp 报文	否	缺省值:0
tcp_fault_flag	整数	0-1	阻断错误 flag 的 tcp 报文	否	缺省值:0
tcp_syn_frag	整数	0-1	阻断对 tcp 的 syn 报文分了片的报文	否	缺省值:0

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.ddos.set

请求 body:

```
{
  "ip_option" : 0,
  "land_attack" : 1,
  "dead_ping" : 0,
  "fragment" : 1,
  "tcp_no_flag" : 1,
  "tcp_fault_flag" : 1,
  "tcp_syn_frag" : 1
}
```

流量控制

流量控制全局使能获取

Action: network.tc.global.get

请求参数: 无

请求举例:

GET

http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.global.get

响应参数:

名称	类型	范围	含义
enable	整数	0-1	流控使能

响应举例:

```
{
  "enable": 0
}
```

流量控制全局使能设置

Action: network.tc.global.set

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

enable	整数	0-1	流控使能	是	Key, 0 表示全局关闭流控, 1 表示全局开启流控;缺省值:0
---------------	----	-----	------	---	-----------------------------------

请求举例:

POST

http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.global.set

请求 body:

```
{
  "enable": 0
}
```

流量控制配置列表

Action: network.tc.list

请求参数: 无

请求举例:

GET http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.list

响应参数:

名称	类型	范围	含义
tc_name	字符串	1-191	流控名字
fw_bandwidth	整数	1-100000 000	上行带宽
rev_bandwidth	整数	1-100000 000	下行带宽

响应举例:

```
[[
  {
    "tc_name": "bwall",
    "fw_bandwidth": 100000,
    "rev_bandwidth": 120000
  }, {
    "tc_name": "bwone",
    "fw_bandwidth": 400000,
    "rev_bandwidth": 500000
  }
]]
```

流量控制配置获取

Action: network.tc.get

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.get

请求 body:

```
{
```

```
"tc_name": "bwall"
```

```
}
```

响应参数:

名称	类型	范围	含义
tc_name	字符串	1-191	流控名字
fw_bandwidth	整数	1-100000000	上行带宽
rev_bandwidth	整数	1-100000000	下行带宽

响应举例:

```
{  
    "tc_name": "bwall",  
    "fw_bandwidth": 100000,  
    "rev_bandwidth": 120000  
}
```

流量控制配置增加

Action: network.tc.add

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
fw_bandwidth	整数	1-100000000	上行带宽	是	单位 Kbps
rev_bandwidth	整数	1-100000000	下行带宽	是	单位 Kbps

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.add
```

请求 body:

```
{  
    "tc_name": "addbwall",  
    "fw_bandwidth": 100000,  
    "rev_bandwidth": 120000  
}
```

流量控制配置编辑

Action: network.tc.edit

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
fw_bandwidth	整数	1-100000000	上行带宽	是	单位 Kbps
rev_bandwidth	整数	1-100000000	下行带宽	是	单位 Kbps

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.edit
```

请求 body:

```
{
  "tc_name": "addbwall",
  "fw_bandwidth": 200000,
  "rev_bandwidth": 220000
}
```

流量控制配置删除

Action: network.tc.del

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.del

请求 body:

```
{
  "tc_name": "addbwall"
}
```

流量控制 RULE 配置列表

Action: network.tc.rule.list

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.rule.list

请求 body:

```
{
  "tc_name": "addbwall"
}
```

响应参数:

名称	类型	范围	含义
tc_name	字符串	1-191	流控名字
fw_bandwidth	整数	1-100000000	规则上行带宽
rev_bandwidth	整数	1-100000000	规则下行带宽
rule_name	字符串	1-191	规则名字
acl	整数	2-198	访问列表

响应举例:

```
[[
  "rule_name": "manager",
```

```

        "fw_bandwidth": 50000,
        "rev_bandwidth": 50000,
        "acl": 2
    }, {
        "rule_name": "employee",
        "fw_bandwidth": 20000,
        "rev_bandwidth": 30000,
        "acl": 3
    }
}

```



Note

创建 rule 规则时需要在已创建的流控中创建

流量控制 RULE 配置获取

Action: network.tc.rule.get

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
rule_name	字符串	1-191	规则名字	是	

请求举例:

POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.rule.get

请求 body:

```

{
    "tc_name": "bwall",
    "rule_name": "manager"
}

```

响应参数:

名称	类型	范围	含义
tc_name	字符串	1-191	流控名字
fw_bandwidth	整数	1-100000000	规则上行带宽
rev_bandwidth	整数	1-100000000	规则下行带宽
rule_name	字符串	1-191	规则名字
acl	整数	2-198	访问列表

响应举例:

```

{
    "rule_name": "manager",
    "fw_bandwidth": 50000,
    "rev_bandwidth": 50000,
    "acl": 2
}

```


}

流量控制 RULE 配置增加

Action: network.tc.rule.add

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
fw_bandwidth	整数	1-100000000	规则上行带宽	是	单位 Kbps
rev_bandwidth	整数	1-100000000	规则下行带宽	是	单位 Kbps
rule_name	字符串	1-191	规则名字	是	Key
acl	整数	2-198	访问列表	是	必须存在
basename	字符串	1-191	规则名	否	若不为空,必须存在
pos	整数	0-1	插入位置	否	如果 basename 不为空, pos 为 0, 表示该规则插入 basename 之前, 如果 pos 为 1, 表示插入 basename 之后

请求举例:

```
POST http://192.168.70.63/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.rule.add
```

请求 body:

```
{
    "tc_name": "bwall",
    "rule_name": "inszero",
    "fw_bandwidth": 600000,
    "rev_bandwidth": 600000,
    "acl": 2,
    "basename": "manager",
    "pos": 0
}
```

流量控制 RULE 配置编辑

Action: network.tc.rule.edit

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
fw_bandwidth	整数	1-100000000	规则上行带宽	是	单位 Kbps
rev_bandwidth	整数	1-100000000	规则下行带宽	是	单位 Kbps
rule_name	字符串	1-191	规则名字	是	Key
acl	整数	2-198	访问列表	是	

basename	字符串	1-191	规则名	否	
pos	整数	0-1	插入位置	否	如果 basename 不为空, pos 为 0, 表示该规则插入 basename 之前, 如果 pos 为 1, 表示插入 basename 之后

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.rule.edit

请求 body:

```
{
  "tc_name": "bwall",
  "rule_name": "inszero",
  "fw_bandwidth": 600000,
  "rev_bandwidth": 600000,
  "acl": 2,
  "basename": "manager",
  "pos": 1
}
```

流量控制 RULE 配置删除

Action: network.tc.rule.del

请求参数:

名称	类型	范围	含义	必选	备注
tc_name	字符串	1-191	流控名字	是	Key
rule_name	字符串	1-191	规则名字	是	Key

请求举例:

POST http://192.168.70.63/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=network.tc.rule.del

请求 body:

```
{
  "tc_name": "bwall",
  "rule_name": "inszero"
}
```

VRRP 高可用

VRRP 全局配置

1.1 获取 VRRP 全局配置

Action: vrrp.global.get

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.global.get

响应参数:

名称	类型	范围	含义
enabled	整数	0,1	1 表示 VRRP 开启, 0 表示关闭
as_model	整数	0,1	1 表示开启 AS_MODE, 0 表示关闭
unit_id	整数	0-8	设备 ID, 用于区分不同的设备; 0 表示关闭
interval	整数	1-255	心跳报文发送间隔, 单位 100 毫秒, 例如: 10 表示 1 秒
retry	整数	2-200	心跳重试次数, 当连续 retry 数量的心跳未接收到时, 认为对应设备故障
delay	整数	1-100	失效间隔, 当发生接口 UP/DOWN 事件后, 延迟 delay 时间之后再改变 vrrp 状态, 单位 100 毫秒
cluster_id	整数	0-7	集群 ID, 用于集群再分类; 0 表示关闭

响应举例:

```
{
  "enabled": 1,
  "as_mode": 0,
  "unit_id": 8,
  "cluster_id": 6,
  "interval": 166,
  "retry": 6,
  "delay": 40
}
```

1.2 设置 VRRP 全局配置

Action: vrrp.global.set

请求参数:

名称	类型	范围	含义	必选	备注
enabled	整数	0,1	1 表示 VRRP 开启, 0 表示关闭	否	未指定时表示不修改, 系统默认值为 0
as_model	整数	0,1	1 表示开启 AS_MODE, 0 表示关闭	否	未指定时表示不修改, 系统默认值为 0
unit_id	整数	0-8	设备 ID, 用于区分不同的设备	否	未指定时表示不修改, 系统默认值为 0
interval	整数	1-255	心跳报文发送间隔, 单位 100 毫秒, 例如: 10 表示 1 秒	否	未指定时表示不修改, 系统默认值为 2
retry	整数	2-200	心跳重试次数, 当连续 retry 数量的心跳未接收到时, 认为对应设备故障	否	未指定时表示不修改, 系统默认值为 5
delay	整数	1-100	失效间隔, 当发生接口 UP/DOWN 事件后, 延迟 delay 时间之后再改变 vrrp 状态, 单位 100 毫秒	否	未指定时表示不修改, 系统默认值为 30
cluster_id	整数	0-7	集群 ID, 用于集群再分类; 0 表示关闭	否	未指定时表示不修改, 系统默认值为 0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.global.set

请求 Body:

```
{
  "enabled": 1,
  "as_mode": 0,
  "unit_id": 8,
  "cluster_id": 6,
  "interval": 166,
  "retry": 6,
  "delay": 40
}
```



Note

1.as_mode 模式下只能有 group_id 0

2.unit_id 为 0 时不能开启 VRRP

VRRP 组

添加 VRRP 组

Action: vrrp.group.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	1-8	组序号	是	
priority	整数	1-200	优先级, 优先级高的优先	否	缺省值: 150
preempt_th	整数	0-100	抢占阈值	否	缺省值: 0
preempt_dis	整数	0, 1	1 表示禁止抢占, 0 表示允许抢占	否	缺省值: 0

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.add

请求 body:

```
{
  "group_id":2,
  "priority":150,
  "preempt_th":0,
  "preempt_dis":0
}
```

获取 VRRP 组列表

Action: vrrp.group.list

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	优先级
preempt_th	整数	0-100	抢占阈值
preempt_dis	整数	0, 1	1 表示禁止抢占, 0 表示允许抢占
manually_standby	整数	0, 1	1 表示处于强制备机状态, 0 表示不处于强制备机状态

响应举例:

```
[{
  "group_id":0,
  "priority":150,
  "preempt_th":0,
  "preempt_dis":0,
  "manually_standby":0
}, {
  "group_id":2,
  "priority":150,
  "preempt_th":0,
  "preempt_dis":0,
  "manually_standby":0
}]
```

获取指定 VRRP 组

Action: vrrp.group.get

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.get

请求 body:

```
{
  "group_id":0
}
```

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	优先级
preempt_th	整数	0-100	抢占阈值
preempt_dis	整数	0, 1	1 表示禁止抢占, 0 表示允许抢占
manually_standby	整数	0, 1	1 表示处于强制备机状态, 0 表示不处于强制备机状态

响应举例:

```
{
```

```

"group_id":0,
"priority":150,
"preempt_th":0,
"preempt_dis":0,
"manually_standby":0
}

```

编辑指定 VRRP 组

Action: vrrp.group.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	优先级	否	缺省值: 不修改
preempt_th	整数	0-100	抢占阈值	否	缺省值: 不修改
preempt_dis	整数	0, 1	1 表示禁止抢占, 0 表示允许抢占	否	缺省值: 不修改

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.edit

请求 body:

```

{
  "group_id":2,
  "priority":120
}

```

获取 VRRP 组状态列表

Action: vrrp.group.stats

请求参数:无

请求举例:

GET http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.stats

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
unit_id	整数	0-8	设备 ID
priority	整数	1-200	配置优先级
priority_cur	整数	1-200	实际优先级
state	整数	0, 1, 2	1 表示主, 0 表示备, 2 表示强制备
peer_list	数组		由多个 PEER 结构组成的数组, 表示 peer 的信息和状态

PEER 结构参数:

名称	类型	范围	含义
state	整数	0, 1, 2	表示 peer 的主备状态, 1 表示主, 0 表示备, 2 表示强制备

priority_cur	整数	1-200	表示 peer 的当前优先级
unit_id	整数	1-8	表示 peer 的设备 ID

响应举例:

```
[{
  "group_id":0,
  "unit_id":1,
  "priority":150,
  "priority_cur":150,
  "state":0,
  "peer_list":[{"state":1,
  "priority_cur":50,
  "unit_id":2
  }]
},{
  "group_id":1,
  "unit_id":1,
  "priority":100,
  "priority_cur":100,
  "state":1,
  "peer_list":[{"state":0,
  "priority_cur":150,
  "unit_id":2
  }]
}]
```

删除指定 VRRP 组

Action: vrrp.group.del

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	1-8	组序号	是	

请求举例:

POST http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.group.del

请求 body:

```
{
  "group_id":1
}
```

 Note	group_id 0 不能删除
--	-----------------

VRRP 心跳接口

添加以太网接口作为心跳接口

Action: vrrp.heart_eth.add

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	以太网接口插槽号	是	ETH0/1 对应 slot 为 0
port	整数	0-28	以太网接口端口号	是	ETH0/1 对应 port 为 1
vlan_id	整数	0-4094	VLAN ID	是	默认 0 表示不带 vlan tag

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_eth.add

请求 body:

```
{
  "slot":0,
  "port":1,
  "vlan_id":1
}
```

获取以太网心跳口配置列表

Action: vrrp.heart_eth.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_eth.list

响应参数:

名称	类型	范围	含义
slot	整数	0-28	以太网接口插槽号
port	整数	0-28	以太网接口端口号
vlan_id	整数	0-4094	VLAN ID

响应举例:

```
[{
  "slot":0,
  "port":1,
  "vlan_id":1
}]
```


获取以太网心跳口统计信息列表

Action: vrrp.heart_eth.statis

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_eth.statis

响应参数:

名称	类型	范围	含义
slot	整数	0-28	以太网接口插槽号
port	整数	0-28	以太网接口端口号
vlan_id	整数	0-4094	VLAN ID; 默认 0 表示不带 vlan tag 标签
packets_rcvd	整数	>=0	接收的报文数
packets_send	整数	>=0	发送的报文数

响应举例:

```
[{
  "slot":0,
  "port":1,
  "vlan_id":1,
  "packets_rcvd":802,
  "packets_send":805
}]
```

删除以太网心跳接口

Action: vrrp.heart_eth.del

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	以太网接口插槽号	是	ETH0/1 对应 slot 为 0
port	整数	0-28	以太网接口端口号	是	ETH0/1 对应 port 为 1
vlan_id	整数	0-4094	VLAN ID	是	默认 0 表示不带 vlan tag

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_eth.del

请求 body:

```
{
  "slot":0,
  "port":1,
  "vlan_id":1
}
```

编辑以太网心跳接口

Action: vrrp.heart_eth.edit

请求参数:

名称	类型	范围	含义	必选	备注
slot	整数	0-28	以太网接口插槽号	是	ETH0/1 对应 slot 为 0
port	整数	0-28	以太网接口端口号	是	ETH0/1 对应 port 为 1
vlan_id	整数	0-4094	VLAN ID	是	仅有 vlan_id 可编辑

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_eth.edit
请求 body:

```
{  
  "slot":0,  
  "port":1,  
  "vlan_id":1  
}
```

添加汇聚接口作为心跳接口

Action: vrrp.heart_trunk.add

请求参数:

名称	类型	范围	含义	必选	备注
trunk_id	整数	1-8	汇聚口端口号	是	
vlan_id	整数	0-4094	VLAN ID	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_trunk.add
请求 body:

```
{  
  "trunk_id":1,  
  "vlan_id":1  
}
```

获取汇聚心跳口配置列表

Action: vrrp.heart_trunk.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_trunk.list

响应参数:

名称	类型	范围	含义
trunk_id	整数	1-8	汇聚口端口号
vlan_id	整数	0-4094	VLAN ID

响应举例:

```
[{
  "trunk_id":1,
  "vlan_id":1
}]
```

获取汇聚心跳口统计信息列表

Action: vrrp.heart_trunk.statis

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_trunk.statis

响应参数:

名称	类型	范围	含义
trunk_id	整数	1-8	汇聚口端口号
vlan_id	整数	0-4094	VLAN ID
packets_rcvd	整数	>=0	接收的报文数
packets_send	整数	>=0	发送的报文数

响应举例:

```
[
{
  "trunk_id":1,
  "vlan_id":1,
  "packets_rcvd":802,
  "packets_send":805
}]
```

删除汇聚心跳接口

Action: vrrp.heart_trunk.del

请求参数:

名称	类型	范围	含义	必选	备注
----	----	----	----	----	----

trunk_id	整数	1-8	汇聚口端口号	是	
vlan_id	整数	0-4094	抢占阈值	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_trunk.delete

请求 body:

```
{
  "trunk_id":1,
  "vlan_id":1
}
```

编辑汇聚心跳接口

Action: vrrp.heart_trunk.edit

请求参数:

名称	类型	范围	含义	必选	备注
trunk_id	整数	1-8	汇聚口端口号	是	
vlan_id	整数	0-4094	抢占阈值	是	仅 vlan_id 可编辑

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_trunk.edit

请求 body:

```
{
  "trunk_id":1,
  "vlan_id":1
}
```

获取所有心跳口配置列表

Action: vrrp.heart_all.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_all.list

响应参数:

名称	类型	范围	含义
slot	整数	0-28	以太网接口插槽号
port	整数	0-28	以太网接口端口号
trunk_id	整数	1-8	汇聚口端口号
vlan_id	整数	0-4094	VLAN ID

响应举例:

[

```
{
  "slot":0,
  "port":1,
  "vlan_id":1
},
{
  "trunk_id":1,
  "vlan_id":1
}
]
```

获取所有心跳口统计信息列表

Action: vrrp.heart_all.stat

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.heart_all.stat

响应参数:

名称	类型	范围	含义
slot	整数	0-28	以太网接口插槽号
port	整数	0-28	以太网接口端口号
trunk_id	整数	1-8	汇聚口端口号
vlan_id	整数	0-4094	VLAN ID
packets_rcvd	整数	>=0	接收的报文数
packets_send	整数	>=0	发送的报文数

响应举例:

```
[{
  "slot":0,
  "port":1,
  "vlan_id":1,
  "packets_rcvd":802,
  "packets_send":805
},
{
  "trunk_id":1,
  "vlan_id":1,
  "packets_rcvd":803,
  "packets_send":806
}
]
```

VRRP 浮动地址

添加 VRRP 浮动地址

Action: vrrp.floating_ip.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
floating_ip	IP 地址		浮动 IP 地址, 支持 IPv4 地址和 IPv6 地址	是	添加浮动 IP 时 group 必须存在

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.floating_ip.add

举例 1: 添加 IPv4 浮动 IP

请求 body1:

```
{
  "group_id":0,
  "floating_ip":"1.1.1.1"
}
```

举例 2: 添加 IPv6 浮动 IP

请求 body2:

```
{
  "group_id":0,
  "floating_ip":"2000::1"
}
```

获取 VRRP 浮动地址列表

Action: vrrp.floating_ip.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.floating_ip.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
floating_ip	IP 地址		浮动 IP 地址

响应举例:

```
[{
  "group_id":0,
  "floating_ip":"1.1.1.1"
},{
  "group_id":0,
  "floating_ip":"2000::1"
}]
```

删除 VRRP 浮动地址

Action: vrrp.floating_ip.del

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
floating_ip	IP 地址		浮动 IP 地址, 支持 IPv4 地址和 IPv6 地址	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.floating_ip.de
l

举例 1: 删除 IPv4 浮动 IP

请求 body1:

```
{
  "group_id":0,
  "floating_ip":"1.1.1.1"
}
```

举例 2: 删除 IPv6 浮动 IP

请求 body2:

```
{
  "group_id":0,
  "floating_ip":"2000::1"
}
```

VRRP 强制切换

获取 VRRP 强制备机信息

Action: vrrp.manually_standby.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.manually_standby.get

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	优先级
preempt_th	整数	0-100	抢占阈值
preempt_dis	整数	0, 1	1 表示禁止抢占, 0 表示允许抢占
manually_standby	整数	0, 1	1 表示处于强制备机状态, 0 表示不处于强制备机状态

响应举例:

```
[{
  "group_id":0,
  "priority":150,
  "preempt_th":0,
  "preempt_dis":0,
  "manually_standby":0
}, {
  "group_id":2,
  "priority":150,
  "preempt_th":0,
  "preempt_dis":0,
  "manually_standby":0
}]
```

设置 VRRP 强制备机

Action: vrrp.manually_standby.set

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	否	未指定该参数时表示所有组
manually_standby	整数	0, 1	1 表示处于强制备机状态, 0 表示不处于强制备机状态	否	缺省值: 1

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.manually_standby.set

举例 1: 设置所有组为强制备机:

{ } 或者 { "manually_standby":1 }

举例 2: 设置 group 2 为强制备机:

{ "group_id ":2,"manually_standby":1 }

举例 3: 取消 group3 强制备机设置

{ "group_id ":3,"manually_standby":0 }

举例 4: 取消所有组强制备机设置

{ "manually_standby":0 }

设置 VRRP 强制主机

Action: vrrp.manually_active.set

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
unit_id	整数	1-8	设备 ID	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.manually_active.set

请求 body:

```
{  
  "group_id":2,  
  "unit_id":2  
}
```

VRRP 网关监控

添加 VRRP 网关监控条件

Action: vrrp.track.gateway.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	
ip	IPv4/IPv6 地址		要监控的网关地址	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.gateway.add

请求 body:

```
{  
  "group_id":2,  
  "priority":50,  
  "ip": "10.0.0.1"  
}
```



添加网关监控条件时,group 必须存在

获取 VRRP 网关监控条件列表

Action: vrrp.track.gateway.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.gateway.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	代价值, 检查失败时减少的优先级
ip	IPv4/IPv6 地址		要监控的网关地址

响应举例:

```
[{
  "group_id":2,
  "priority":50,
  "ip":"10.0.0.1"
}]
```

编辑 VRRP 网关监控条件

Action: vrrp.track.gateway.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	可编辑项
ip	IPv4/IPv6 地址		要监控的网关地址	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.gateway.edit

请求 body:

```
{
  "group_id":2,
```

```

    "priority":120,
    "ip":"10.0.0.1"
}

```

删除 VRRP 网关监控条件

Action: vrrp.track.gateway.del

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	否	删除时不检查
ip	IPv4/IPv6 地址		要监控的网关地址	是	

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.gateway.del

```

请求 body:

```

{
    "group_id":2,
    "ip":"10.0.0.1"
}

```

VRRP 路由监控

添加 VRRP 路由监控条件

Action: vrrp.track.route.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	
ip	IPv4/IPv6 地址		要监控的路由地址	是	
netmask	IPv4 掩码 /IPv6 前缀		要监控的 IPv4 路由掩码或 IPv6 前缀 (XXX.XXX.XXX.XXX/1-128)	是	

请求举例:

```

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.route.add

```

请求 body:

```
{
  "group_id":2,
  "priority":50,
  "ip":"10.0.0.0",
  "netmask":"255.255.255.0"
}
```



添加路由监控条件时,group 必须存在

获取 VRRP 路由监控条件列表

Action: vrrp.track.route.list

请求参数:无

请求举例:

GET
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.route.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	代价值, 检查失败时减少的优先级
ip	IPv4/IPv6 地址		要监控的路由地址
netmask	IPv4 掩码 /IPv6 前缀		要监控的 IPv4 路由掩码或 IPv6 前缀 (XXX.XXX.XXX.XXX/1-128)

响应举例:

```
[{
  "group_id":2,
  "priority":50,
  "ip":"10.0.0.0",
  "netmask":"255.255.255.0"
}]
```

编辑 VRRP 路由监控条件

Action: vrrp.track.route.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	可编辑项
ip	IPv4/IPv6		要监控的路由地址	是	

	地址				
netmask	IPv4 掩码 /IPv6 前缀		要监控的 IPv4 路由掩码或 IPv6 前缀 (XXX.XXX.XXX.XXX/1-128)	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.route.ed
 it

请求 body:

```
{
  "group_id":2,
  "priority":120,
  "ip":"10.0.0.0",
  "netmask":"255.255.255.0"
}
```

删除 VRRP 路由监控条件

Action: **vrrp.track.route.del**

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	否	删除时不检查
ip	IPv4/IPv6 地址		要监控的路由地址	是	
netmask	IPv4 掩码 /IPv6 前缀		要监控的 IPv4 路由掩码或 IPv6 前缀 (XXX.XXX.XXX.XXX/1-128)	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.route.de
 l

请求 body:

```
{
  "group_id":2,
  "ip":"10.0.0.0",
  "netmask":"255.255.255.0"
}
```

VRRP 以太网接口监控

添加 VRRP 以太网接口监控条件

Action: **vrrp.track.ethernet.add**

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	
slot	整数	0-28	要监控的以太网接口插槽号	是	
port	整数	0-28	要监控的以太网接口端口号	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.ethernet.add

请求 body:

```
{  
  "group_id":2,  
  "priority":50,  
  "slot":0,  
  "port":1  
}
```



添加以太网接口监控条件时.group 必须存在

获取 VRRP 以太网接口监控条件列表

Action: vrrp.track.ethernet.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.ethernet.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	代价值, 检查失败时减少的优先级
slot	整数	0-28	要监控的以太网接口插槽号
port	整数	0-28	要监控的以太网接口端口号

响应举例:

```
[{  
  "group_id":2,  
  "priority":50,  
  "slot":0,  
  "port":1  
}]
```

编辑 VRRP 以太网接口监控条件

Action: vrrp.track.ethernet.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	是	可编辑项
slot	整数	0-28	要监控的以太网接口插槽号	是	
port	整数	0-28	要监控的以太网接口端口号	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.ethernet.edit

请求 body:

```
{  
  "group_id":2,  
  "priority":120,  
  "slot":0,  
  "port":1  
}
```

删除 VRRP 以太网接口监控条件

Action: vrrp.track.ethernet.del

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 检查失败时减少的优先级	否	删除时不检查
slot	整数	0-28	要监控的以太网接口插槽号	是	
port	整数	0-28	要监控的以太网接口端口号	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.ethernet.del

请求 body:

```
{  
  "group_id":2,  
  "slot":0,  
  "port":1  
}
```

VRRP 汇聚接口监控

添加 VRRP 汇聚接口监控条件

Action: vrrp.track.trunk.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 汇聚口检查失败时减少的优先级	是	
trunk_id	整数	1-8	汇聚接口端口号	是	
member_priority	整数	0-200	成员端口代价值, 汇聚子接口失效时的代价值	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.trunk.add

请求 body:

```
{  
  "group_id":2,  
  "priority":100,  
  "trunk_id":1,  
  "member_priority":50  
}
```



添加汇聚接口监控条件时,group 必须存在

获取 VRRP 汇聚接口监控条件列表

Action: vrrp.track.trunk.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.trunk.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	代价值, 汇聚口检查失败时减少的优先级
trunk_id	整数	1-8	汇聚接口端口号

member_priority	整数	0-200	成员端口代价值, 汇聚子接口失效时的代价值
-----------------	----	-------	-----------------------

响应举例:

```
[{
  "group_id":2,
  "priority":100,
  "trunk_id":1,
  "member_priority":50
}]
```

编辑 VRRP 汇聚接口监控条件

Action: vrrp.track.trunk.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 汇聚口检查失败时减少的优先级	是	
trunk_id	整数	1-8	汇聚接口端口号	是	
member_priority	整数	0-200	成员端口代价值, 汇聚子接口失效时的代价值	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.trunk.edit>

请求 body:

```
{
  "group_id":2,
  "priority":100,
  "trunk_id":1,
  "member_priority":50
}
```

删除 VRRP 汇聚接口监控条件

Action: vrrp.track.trunk.del

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 汇聚口检查失败时减少的优先级	否	删除时不检查
trunk_id	整数	1-8	汇聚接口端口号	是	
member_priority	整数	0-200	成员端口代价值, 汇聚子接口失效时的代价值	否	删除时不检查

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.trunk.de
l

请求 body:

```
{  
  "group_id":2,  
  "trunk_id":1  
}
```

VRRP 虚拟接口监控

添加 VRRP 虚拟接口监控条件

Action: vrrp.track.vlan.add

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 虚拟接口检查失败时减少的优先级	是	
vlan_id	整数	2-4094	VLAN 接口端口号	是	
timeout	整数	2-600	失效判定的时长	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.vlan.add
请求 body:

```
{  
  "group_id":2,  
  "priority":100,  
  "vlan_id":2,  
  "timeout":50  
}
```



Note

添加虚拟接口监控条件时,group 必须存在

添加虚拟接口监控条件时,虚拟接口必须有 IP 地址

获取 VRRP 虚拟接口监控条件列表

Action: vrrp.track.vlan.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.vlan.list

响应参数:

名称	类型	范围	含义
group_id	整数	0-8	组序号
priority	整数	1-200	代价值, 虚拟接口检查失败时减少的优先级
vlan_id	整数	2-4094	VLAN 接口端口号
timeout	整数	2-600	失效判定的时长

响应举例:

```
[{
  "group_id":2,
  "priority":100,
  "vlan_id":2,
  "timeout":50
}]
```

编辑 VRRP 虚拟接口监控条件

Action: vrrp.track.vlan.edit

请求参数:

名称	类型	范围	含义	必选	备注
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 虚拟接口检查失败时减少的优先级	是	
vlan_id	整数	2-4094	VLAN 接口端口号	是	
timeout	整数	2-600	失效判定的时长	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.vlan.edit

请求 body:

```
{
  "group_id":2,
  "priority":100,
  "vlan_id":2,
  "timeout":50
}
```

删除 VRRP 虚拟接口监控条件

Action: vrrp.track.vlan.del

请求参数:

名称	类型	范围	含义	必	备注
----	----	----	----	---	----

				选	
group_id	整数	0-8	组序号	是	
priority	整数	1-200	代价值, 虚拟接口检查失败时减少的优先级	否	删除时不检查
vlan_id	整数	2-4094	VLAN 接口端口号	是	
timeout	整数	2-600	失效判定的时长	是	删除时不检查

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.track.vlan.del

请求 body:

```
{
  "group_id":2,
  "vlan_id":2,
  "timeout":10
}
```

VRRP 配置同步对端

添加 VRRP 配置同步对端

Action: vrrp.cfgsync.peer.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	配置同步对端名称	是	唯一
ip	IPv4/v6 地址		配置同步对端 ip 地址	是	
port	整数	1-65535	配置同步对端 ssh 端口号	是	
user	字符串	长度 1-31	配置同步对端用户名	是	
password	字符串	长度 1-30	配置同步对端密码	是	
save	整数	0, 1	1 表示同步时自动保存配置, 0 表示不保存	是	

请求举例:

POST
 http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.peer.add

请求 body:

```
{
  "name": "peer1",
  "ip": "1.1.1.1",
  "port": 22,
  "user": "user1",
  "password": "user1_password",
  "save": 0
}
```

获取 VRRP 配置同步对端列表

Action: vrrp.cfgsync.peer.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.peer.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	配置同步对端名称
ip	IPv4/v6 地址		配置同步对端 ip 地址
port	整数	1-65535	配置同步对端 ssh 端口号
user	字符串	长度 1-31	配置同步对端用户名
save	整数	0, 1	1 表示同步时自动保存配置, 0 表示不保存
status	字符串	"Succeeded", "None", "Syncing", "Failed", "Itself"	"Succeeded"表示同步成功, "None"表示初始状态, "Syncing"表示该同步对端同步中, "Failed"表示该同步对端同步失败, "Itself"表示该同步对端就是自己

响应举例:

```
[{  
  "name": "peer1",  
  "ip": "1.1.1.1",  
  "port": 22,  
  "user": "user1",  
  "status": "None",  
  "save": 0  
}]
```

获取指定 VRRP 配置同步对端

Action: vrrp.cfgsync.peer.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	配置同步对端名称	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.peer.get

请求 body:

```
{"name": "peer1"}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	配置同步对端名称
ip	IPv4/v6 地址		配置同步对端 ip 地址
port	整数	1-65535	配置同步对端 ssh 端口号
user	字符串	长度 1-31	配置同步对端用户名
save	整数	0, 1	1 表示同步时自动保存配置, 0 表示不保存
status	字符串	"Succeeded", "None", "Syncing", "Failed", "Itself"	"Succeeded"表示同步成功成功, "None"表示初始状态, "Syncing"表示该同步对端同步中, "Failed"表示该同步对端同步失败, "Itself"表示该同步对端就是自己

响应举例:

```
{  
  "name": "peer1",  
  "ip": "1.1.1.1",  
  "port": 22,  
  "user": "user1",  
  "status": "None",  
  "save": 0  
}
```

获取 VRRP 配置同步模块

Action: vrrp.sync.module.get

请求参数:

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.sync.module.get

请求 body:

响应参数:

名称	类型	范围	含义
module	字符串	"route-static", "route-proto"	vrrp 模式下定制的需要同步的路由模块 "route-static" 表示同步静态路由 "route-proto" 表示同步动态路由

响应举例:

```
{  
  "modules": "route-proto,route-static"  
}
```

编辑 VRRP 配置同步模块

Action: vrrp.sync.module.set

请求参数:

名称	类型	范围	含义
module	字符串	“route-static”, “route-proto”	vrrp 模式下定制的需要同步的路由模块 “route-static” 表示同步静态路由 “route-proto” 表示同步动态路由

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.sync.module.set
请求 body:
{
  "modules": "route-proto,route-static"
}
```

编辑指定 VRRP 配置同步对端

Action: vrrp.cfgsync.peer.edit

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	配置同步对端名称	是	
user	字符串	长度 1-31	配置同步对端用户名	是	
password	字符串	长度 1-30	配置同步对端密码	是	
save	整数	0, 1	1 表示同步时自动保存配置, 0 表示不保存	是	

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.peer.edit
请求 body:
{
  "name": "peer1",
  "user": "user2",
  "password": "user2_password",
  "save": 1
}
```

删除指定 VRRP 配置同步对端

Action: vrrp.cfgsync.peer.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	配置同步对端名称	是	

请求举例:

```
POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.peer.d
```

e1

请求 body:

```
{"name": "peer1"}
```

触发配置同步

Action: vrrp.cfgsync.action.sync

请求参数:无

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.action.sync

VRRP 自动同步

获取 VRRP 自动同步配置信息

Action: vrrp.cfgsync.auto.get

请求参数:无

请求举例:

GET

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.auto.get

响应参数:

名称	类型	范围	含义
enabled	整数	0, 1	1 表示开启了自动同步, 0 表示关闭了自动同步

响应举例:

```
{"enabled": 0}
```

设置 VRRP 自动同步配置

Action: vrrp.cfgsync.auto.set

请求参数:

名称	类型	范围	含义	必选	备注
enabled	整数	0, 1	1 表示开启了自动同步, 0 表示关闭了自动同步	是	

请求举例:

POST

http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.auto.set

Body:

```
{"enabled": 1}
```


获取 VRRP 同步通知配置信息

Action: vrrp.cfgsync.notify.get

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.notify.get

响应参数:

名称	类型	范围	含义
notify	整数	0, 1	1 表示开启了同步通知, 0 表示关闭了同步通知

响应举例:

```
{"notify":0}
```

设置 VRRP 同步通知配置

Action: vrrp.cfgsync.notify.set

请求参数:

名称	类型	范围	含义	必选	备注
notify	整数	0, 1	1 表示开启了同步通知, 0 表示关闭了同步通知	是	

请求举例:

POST
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.notify.set
Body:
{ "notify":1 }

获取 VRRP 同步状态

Action: vrrp.cfgsync.flag.get

请求参数:无

请求举例:

GET
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.cfgsync.flag.get

响应参数:

名称	类型	范围	含义
syncflag	整数	0, 1	1 表示有需要同步的配置未同步, 0 表示没有需要同步的配置

响应举例:

```
{"syncflag":0}
```



开启同步通知功能后,若有需要同步的配置未同步,则获取到的 VRRP 同步状态为 1

VRRP 配置 L3peer

添加 VRRP L3peer 邻居

Action: vrrp.l3.peer.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	L3peer 邻居名称	是	唯一
ip	IPv4/v6 地址		L3peer 邻居的 ip/ipv6 地址	是	

请求举例:

POST

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3.peer.add

请求 body:

```
{
  "name": "1234567890123456",
  "ip": "10.1.1.14",
  "status": "Peer"
}
```

获取 VRRP L3peer 邻居

Action: vrrp.l3.peer.list

请求参数:无

请求举例:

GET

http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3.peer.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	L3peer 邻居名称
ip	IPv4/v6 地址		L3peer 邻居的 ip/ipv6 地址
status	字符串	"Peer", "Itself"	"Peer"表示同步的是对端, "Itself"表示该同步对端就是自己

响应举例:

```
[
  {
    "name": "123456789012345678901234567",
    "ip": "10.1.1.14",
    "status": "Peer"
  }
]
```

```

    },
    {
        "name": "1234567890123456789012345",
        "ip": "10.1.1.11",
        "status": "Itself"
    }
]

```

获取指定 VRRP L3peer 邻居

Action: vrrp.l3.peer.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	L3peer 邻居名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3.peer.get>

请求 body:

```
{"name": "peer1"}
```

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	L3peer 邻居名称
ip	IPv4/v6 地址		L3peer 邻居的 ip/ipv6 地址
status	字符串	"Peer", "Itself"	"Peer"表示同步的是对端, "Itself"表示该同步对端就是自己

响应举例:

```

{
    "name": "1234567890123456",
    "ip": "10.1.1.14",
    "status": "Peer"
}

```

删除指定 VRRP L3peer 邻居

Action: vrrp.l3.peer.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	L3peer 邻居名称	是	

请求举例:

POST
<http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3.peer.del>

请求 body:

```
{
  "name": "1234567890123456"
}
```

获取 VRRP L3peer 使能状态

Action: vrrp.l3peer.enable.get

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	L3peer 邻居名称	是	

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3peer.enable.get
```

请求 body:

```
{
  "name": "1234567890123456"
}
```

响应参数:

名称	类型	范围	含义
enable	整数	0, 1	0: 不使能三层同步 1: 使能三层同步

响应举例:

```
{
  "enabled": 0
}
```

编辑 VRRP L3peer 使能状态

Action: vrrp.l3peer.enable.set

请求参数:

名称	类型	范围	含义
enable	整数	0, 1	0: 不使能三层同步 1: 使能三层同步

请求举例:

```
POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.l3peer.enable.set
```

请求 body:

```
{
  "enabled": 0
}
```

VRRP 配置 MGMT peer

添加 VRRP mgmt peer 邻居

Action: vrrp.mgmt.peer.add

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	VRRP 邻居名称	是	唯一
ip	IPv4/v6 地址		VRRP 邻居 ip/ipv6 地址	是	

请求举例:

POST
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.mgmt.peer.add
请求 body:

```
{  
  "name": "1234567890123456",  
  "ip": "10.1.1.14",  
  "status": "Peer"  
}
```

获取 VRRP mgmt peer 邻居列表

Action: vrrp.mgmt.peer.list

请求参数:无

请求举例:

GET
http://192.168.70.73/adcap/v2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.mgmt.peer.list

响应参数:

名称	类型	范围	含义
name	字符串	长度 1-63	VRRP 邻居名称
ip	IPv4/v6 地址		VRRP 邻居 ip/ipv6 地址
status	字符串	"Peer", "Itself"	"Peer"表示同步的是对端, "Itself"表示该同步对端就是自己

响应举例:

```
[  
  {  
    "name": "123456789012345678901234567",  
    "ip": "10.1.1.14",  
    "status": "Peer"  
  },  
  {  
    "name": "1234567890123456789012345",  
    "ip": "10.1.1.11",  
    "status": "Itself"  
  }  
]
```

```
}  
]
```

删除指定 VRRP mgmt peer 邻居

Action: vrrp.mgmt.peer.del

请求参数:

名称	类型	范围	含义	必选	备注
name	字符串	长度 1-63	VRRP 邻居名称	是	

请求举例:

```
POST  
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.mgmt.peer.del  
请求 body:  
{  
  "name": "1234567890123456"  
}
```

获取 VRRP mgmt peer 统计信息列表

Action: vrrp.heart_mgmt.statis

请求参数:无

请求举例:

```
GET  
http://192.168.70.73/adcapiv2.0/?authkey=dbc121e55cc33c67911a99ce4829db&action=vrrp.mgmt.peer.list
```

响应参数:

名称	类型	范围	含义
type	整数	0-28	管理口
packets_rcvd	整数	>=0	接收的报文数
packets_send	整数	>=0	发送的报文数

响应举例:

```
[  
  {  
    "type": "mgmt",  
    "packets_rcvd": 1368615,  
    "packets_send": 1368615  
  }  
]
```

