# M策略设计文档

## 程序启动

### 链表初始化



程序启动后，首先要查询数据库中的 plc\_table 表。如果为空，则此步骤完成。如果不为空，从数据库中逐条取出数据，并对取出的数据进行base64解密。解密后的数据包含[*处理消息*](#_处理消息结构体)*、*[*新增消息*](#_新增消息结构体)以及对应的*各类策略消息*。

从新增消息头中，可以获取到用户ID和规则ID，对用户ID和规则ID进行哈希，查找哈希表中是否存在。如果存在，则继续对下一条数据进行处理。如果不存在，则将数据库中取出的这条数据插入到哈希表中。

不断地重复上述步骤，直到数据库中的所有数据都取出。

### 规则全量更新

程序扫描哈希表g\_list\_head\_in\_mysql，对哈希表中的所有数据，根据策略类型的不同，执行各自的规则文件生成。具体可以参考全量更新

### 端口监听

程序启动后，会默认监听25321(配置中可配), 用于接收策略服务器下发的策略.

### 协议和应用XML文件解析

程序启动后，会调用*mxml*的接口取读取本地的协议和应用xml文件（在配置中可配文件路径）。将xml文件中的信息逐条读取，构建[协议和应用XML结构体](#_协议和应用XML)，然后将构建好的信息以app\_id值为key，插入到哈希表*g\_list\_head\_for\_pro\_app*中。

### HTTP关键字的读取

程序启动后，会从配置中加载HTTP扩展字段以及对应ID。并将其保存在内存中。

## 用户鉴别



策略服务器下发连接请求，此时会附带用户鉴别信息，消息格式如下表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字段名 | 长度 | 描述 |
| 消息头 | 消息标识 | 2字节 | 0x0111 |
| 消息类型 | 2字节 | 0x0101 |
| 消息体长度 | 4字节 | 48 |
| 消息体 | 用户名 | 16字节 | 登陆用户名信息，不足16字节时后面补0 |
| 口令 | 32字节 | 口令的MD5值 |

策略接收程序在接收到用户鉴别请求的时候，会对用户名进行验证，是否已经录入本地配置文件。如果未查询到，则直接回复ACK，错误码0x0007 (无效用户)。如果查询到，则对口令进行。

## 策略解析

### 新增消息

新增规则请求的消息体报文格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 用户ID | | | | | | | | 输出端口组号 | | | | | | | |
| 1 | 动作 | | 重启保存 | 中标统计 | 单包匹配 | 保留 | | | 运营商(默认:0x00) | | | | | | | |
| 2~3 | 规则ID | | | | | | | | | | | | | | | |
| 4~5 | 时间戳 | | | | | | | | | | | | | | | |
| 6 | not | 条件1规则类型 | | | | | | | not | 条件2规则类型 | | | | | | |
| 7 | not | 条件3规则类型 | | | | | | | 保留 | | | | | | | |
| 8~N | 实际规则内容 | | | | | | | | | | | | | | | |

新增规则请求的消息

|  |  |  |
| --- | --- | --- |
| 字段名 | 长度 | 描述 |
| 用户ID | 1字节 | 登陆用户ID信息 |
| 输出端口组号 | 1字节 | 输出端口组号 |
| 动作 | 2 bit | 0-丢弃; 1-转发 |
| 重启保存 | 1 bit | 0-不保存; 1-保存 |
| 中标统计 | 1 bit | 0-不统计; 1-统计 |
| 运营商 | 1字节 |  |
| 规则ID | 4字节 | 同一虚拟用户中的唯一规则标识 |
| 时间戳 | 4字节 | 从1970年1月1日0点0分0秒到当前的秒数 |
| 条件1规则类型 | 1字节 | 参见[7.1](#_规则类型)中表 |
| 条件2规则类型 | 1字节 | 组合规则下有效 |
| 条件3规则类型 | 1字节 | 组合规则下有效 |
| 实际规则内容 | 不定长 |  |

策略服务器在下发新增消息时, 会附带上述所示的的信息, 程序会使用用户*ID*和规则*ID*进行哈希表的查找和插入。

如果在哈希表中找到了对应的用户*ID*和规则*ID*的组合，则直接回复*ACK*规则已存在，[应答码](#_应答代码表)为0x0002。

如果哈希表中未找到，则根据条件1[消息块类型](#_规则类型)，分别按照对应的策略消息结构进行解析和校验。如果校验成功，按照[处理消息结构体](#_处理消息结构体)构建消息，并将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中.

#### 3.1.1 各类策略消息解析

##### IPv4五元组

IPv4五元组消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0~1 | 源IP | | | | | | | | | | | | | | | |
| 2~3 | 源IP掩码 | | | | | | | | | | | | | | | |
| 4~5 | 目的IP | | | | | | | | | | | | | | | |
| 6~7 | 目的IP掩码 | | | | | | | | | | | | | | | |
| 8 | 起始源端口 | | | | | | | | | | | | | | | |
| 9 | 结束源端口 | | | | | | | | | | | | | | | |
| 10 | 起始目的端口 | | | | | | | | | | | | | | | |
| 11 | 结束目的端口 | | | | | | | | | | | | | | | |
| 12 | 协议号 | | | | | | | | 协议号掩码 | | | | | | | |
| 13 | 内层/外层IP | | | | | | | |  |  |  |  |  |  |  |  |

IPv4五元组消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 源IP | 4字节 |  |
| 源IP掩码 | 4字节 |  |
| 目的IP | 4字节 |  |
| 目的IP掩码 | 4字节 |  |
| 起始源端口 | 2字节 |  |
| 结束源端口 | 2字节 |  |
| 起始目的端口 | 2字节 |  |
| 结束目的端口 | 2字节 |  |
| 协议号 | 1字节 | 五元组中的协议号，0xff表示全部 |
| 协议号掩码 | 1字节 | 暂时用不到 |
| 内层or外层IP | 1字节 | 0-内层，1-外层 |

策略模块在接收到新增五元组的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x01（IPv4五元组），则按照IPv4五元组消息块的格式对消息体进行解析。

在解析IPv4消息的过程中，如果读取信息失败，会回复ACK[未知错误](#_应答代码表)。否则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_ip\_port\_pro\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### IPv6五元组

IPv6五元组消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0~7 | 源IP | | | | | | | | | | | | | | | |
| 8~16 | 源IP掩码 | | | | | | | | | | | | | | | |
| 16~23 | 目的IP | | | | | | | | | | | | | | | |
| 24~31 | 目的IP掩码 | | | | | | | | | | | | | | | |
| 32 | 起始源端口 | | | | | | | | | | | | | | | |
| 33 | 结束源端口 | | | | | | | | | | | | | | | |
| 34 | 起始目的端口 | | | | | | | | | | | | | | | |
| 35 | 结束目的端口 | | | | | | | | | | | | | | | |
| 36 | 协议号 | | | | | | | | 协议号掩码 | | | | | | | |
| 37 | 内层/外层IP | | | | | | | |  |  |  |  |  |  |  |  |

IPv6五元组消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 源IP | 16字节 |  |
| 源IP掩码 | 16字节 |  |
| 目的IP | 16字节 |  |
| 目的IP掩码 | 16字节 |  |
| 起始源端口 | 2字节 |  |
| 结束源端口 | 2字节 |  |
| 起始目的端口 | 2字节 |  |
| 结束目的端口 | 2字节 |  |
| 协议号 | 1字节 | 五元组中的协议号，0xff表示全部 |
| 协议号掩码 | 1字节 | 暂时用不到 |
| 内层or外层IP | 1字节 | 0-内层，1-外层 |

策略模块在接收到新增五元组的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x02（IPv6五元组），则按照IPv6五元组消息块的格式对消息体进行解析。

在解析IPv6消息的过程中，如果读取信息失败，会回复ACK[未知错误](#_应答代码表)。否则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_ip\_port\_pro\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 特征码

特征码消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 浮动 | 保留 | 编码方式 | | 偏移类型 | | | | 保留 | | | | | | | |
| 1 | 偏移 | | | | | | | | | | | | | | | |
| 2~33 | 特征码内容 | | | | | | | | | | | | | | | |
| 34~65 | 特征码内容掩码 | | | | | | | | | | | | | | | |

特征码消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 浮动 | 1 bit | 0-固定偏移规则；1-浮动偏移规则 |
| 正则 | 1 bit | 保留 |
| 编码方式 | 2 bit | 0-ASCII码；1-16进制 |
| 偏移类型 | 4 bit | 0-IP；1-TCP；2-UDP |
| 偏移 | 2字节 | 偏移字节数 |
| 特征码内容 | 64字节 |  |
| 特征码内容掩码 | 64字节 | 掩码每个字节为0xff或者0x00 |

策略模块在接收到新增特征码的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x03（特征码），则按照特征码消息块的格式对消息体进行解析。

在解析特征码内容的时候，需要对内容和掩码进行校验。如果出现以下两种情况：

1. 掩码中不存在0xff；
2. 掩码的某个字节为0xff，但是对应的内容为0x0。

则返回解析特征码消息失败，会回复ACK[未知错误](#_应答代码表)。

如果解析成功，则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_cond\_code\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 账号

账号消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 账号类型 | | | | 保留 | | | | | | | | | | | |
| 1~32 | 账号 | | | | | | | | | | | | | | | |
| 33~64 | 账号掩码 | | | | | | | | | | | | | | | |

账号消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 账号类型 | 4 bit | 0-ADSL账号；1-IMSI；2-MSISDN；3-IMEI |
| 账号 | 64字节 | 使用ASCII码描述 |
| 账号掩码 | 64字节 | 掩码每个字节为0xff或者0x00 |

策略模块在接收到新增账号的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x04（账号），则按照账号消息块的格式对消息体进行解析。

在解析账号内容的时候，需要对账号和掩码进行校验。如果出现以下两种情况：

1. 掩码中不存在0xff；
2. 掩码的某个字节为0xff，但是对应的内容为0x0。

则返回解析账号消息失败，会回复ACK[未知错误](#_应答代码表)。

如果解析成功，则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_account\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 协议和应用

协议和应用消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0~3 | 应用ID | | | | | | | | | | | | | | | |
| 4~11 | 协议名称 | | | | | | | | | | | | | | | |

协议和应用消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 应用ID | 8字节 | 应用ID和优先级有关 |
| 协议名称 | 16字节 | 参加下表 |

协议名称对应表

|  |  |
| --- | --- |
| 协议名称 | 协议编号 |
| ETH | 1001 |
| VLAN | 1002 |
| MPLS | 1003 |
| PPP | 1004 |
| PPP\_CCP | 1005 |
| PPP\_CHAP | 1006 |
| PPP\_COMP | 1007 |
| PPPOE\_PAP | 1008 |
| PPPOE\_DISCOVERY | 1009 |
| PPPOE\_SESSION | 1010 |
| ARP | 2001 |
| RARP | 2002 |
| IMCP | 2003 |
| IPV4 | 2004 |
| IPV6 | 2005 |
| TCP | 3001 |
| UDP | 3002 |
| SCTP | 3003 |
| GRE | 3004 |
| IP\_IPSEC | 4001 |
| IPSEC\_AH | 4002 |
| IS\_OSPF | 4003 |
| PTP | 4004 |
| CDMA\_A10 | 4005 |
| CDMA\_A11 | 4006 |
| GTP\_V0 | 4007 |
| GTP\_V1 | 4008 |
| GTP\_V2 | 4009 |
| GTP | 4010 |
| L2TP | 4011 |
| PPTP | 4012 |
| TEREDO | 4013 |
| RADIUS | 5001 |
| DNS | 5002 |
| RDP | 5003 |
| HTTPS | 5004 |
| HTTP | 5005 |
| HTTP\_PROXY | 5006 |
| FTP | 5007 |
| SIP | 5008 |
| MAIL\_IMAP | 5009 |
| MAIL\_POP3 | 5010 |
| MAIL\_SMTP | 5011 |
| MAIL\_IMAPS | 5012 |
| MAIL\_POP3S | 5013 |
| MAIL\_SMTPS | 5014 |
| BGP | 5015 |
| IAX | 5016 |
| IP\_OSPF | 5017 |
| REDIS | 5018 |
| RTMP | 5019 |
| RTSP | 5020 |
| MMS | 5021 |
| SSH | 5022 |
| TELNET | 5023 |
| MDNS | 5024 |
| NTP | 5025 |
| SNMP | 5026 |
| TFTP | 5027 |

策略模块在接收到新增协议和应用的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x05（协议和应用），则按照协议和应用消息块的格式对消息体进行解析。

在解析协议和应用消息的过程中，如果读取信息失败，会回复ACK[未知错误](#_应答代码表)。否则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_pro\_app\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 音频和视频

音视频消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 音/视频格式 | | | | | | | | A/V | 上下行 | | 保留 | | | | |

音视频消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 音/视频格式 | 1字节 | 参加下表，音频类型编码描述和视频类型编码描述 |
| A/V | 1 bit | 0-音频规则；1-视频规则。 |
| 上下行 | 2 bits | 0-all匹配；1-上行匹配；2-下行匹配 |

音视频类型规则描述

|  |  |  |  |
| --- | --- | --- | --- |
| 音频编码 | 音频类型 | 视频编码 | 视频类型 |
| 0x01 | MP3 | 0x80 | MP4 |
| 0x02 | AAC | 0x81 | M4V |
| 0x03 | WAV | 0x82 | MPG |
| 0x04 | WMA | 0x83 | MPEG |
| 0x05 | CDA | 0x84 | FLV |
| 0x06 | FLAC | 0x85 | F4V |
| 0x07 | M4A | 0x86 | SWF |
| 0x08 | MID | 0x87 | AVI |
| 0x09 | MKA | 0x88 | GIF |
| 0x0A | MP2 | 0x89 | WMV |
| 0x0B | MPA | 0x8A | RM |
| 0x0C | MPC | 0x8B | RMVB |
| 0x0D | APE | 0x8C | MOV |
| 0x0E | OFR | 0x8D | MTS |
| 0x0F | OGG | 0x8E | M2T |
| 0x10 | RA | 0x8F | WEBM |
| 0x11 | WV | 0x90 | OGG |
| 0x12 | TTA | 0x91 | OGM |
| 0x13 | AC3 | 0x92 | MKV |
| 0x14 | DTS | 0x93 | DIVX |
| 0x15~0x7F | 音频保留 | 0x94 | DAT |
|  |  | 0x95 | TS |
|  |  | 0x96 | TP |
|  |  | 0x97 | IFO |
|  |  | 0x98 | NSV |
|  |  | 0x99 | ASF |
|  |  | 0x9A | ASX |
|  |  | 其余 | 视频保留 |

策略模块在接收到新增音频和视频的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x06（音频和视频），则按照音频和视频消息块的格式对消息体进行解析。

在解析音频和视频消息的过程中，如果读取信息失败（会对音频视频格式进行校验），会回复ACK[未知错误](#_应答代码表)。否则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_audio\_video\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### HTTP

HTTP消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | M | H | URL | 保留 | UA | CK | CT | 保留 | | | | | | | | |
| 1 | Method类型 | | | | | | | | 保留 | | | | | | | |
| 2~33 | HOST | | | | | | | | | | | | | | | |
| 34~65 | HOST掩码 | | | | | | | | | | | | | | | |
| 66~129 | URL | | | | | | | | | | | | | | | |
| 130~193 | URL掩码 | | | | | | | | | | | | | | | |
| 194~225 | User-Agent | | | | | | | | | | | | | | | |
| 226~289 | Cookie | | | | | | | | | | | | | | | |
| 290~305 | Content-Type | | | | | | | | | | | | | | | |
| 306~561 | 其他HTTP字段 | | | | | | | | | | | | | | | |

HTTP消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| M | 1 bit | 0-Method字段无效；1-Method字段有效。 |
| H | 1 bit | 0-HOST字段无效；1-HOST字段有效。 |
| URL | 1 bit | 0-URL字段无效；1-URL字段有效。 |
| U-R | 1 bit | 保留 |
| UA | 1 bit | 0-UserAgent字段无效；1-User-Agent字段有效 |
| CK | 1 bit | 0-Cookie字段无效；1-Cookie字段有效 |
| CT | 1 bit | 0-Content-Type字段无效；1-Content-type字段有效 |
| Method类型 | 1字节 | ASCII码 |
| HOST | 64字节 | HTTP HOST字段 |
| HOST掩码 | 64字节 | HTTP HOST字段掩码 |
| URL | 128字节 | HTTP URL字段（相对路径），即我们所使用的URI |
| URL掩码 | 128字节 | HTTP URL字段掩码 |
| User\_Agent | 64字节 | HTTP UA字段 |
| Cookie | 128字节 | HTTP Cookie字段 |
| Content-Type | 32字节 | HTTP Content-Type字段 |
| 其他HTTP字段 | 256字节 | HTTP其他字段，至少支持2个 |

策略模块在接收到新增HTTP的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x07（HTTP），则按照HTTP消息块的格式对消息体进行解析。

在解析HTTP消息的过程中，如果读取信息失败，会回复ACK[未知错误](#_应答代码表)。否则会对HTTP消息的格式进行校验。包括：

1. 是否所有的标记位都为空，如果都为空，则返回[未知错误](#_应答代码表)；
2. HOST和HOST掩码是否有效，如果无效，则返回[未知错误](#_应答代码表)；
3. URL和URL掩码是否有效，如果无效，则返回[未知错误](#_应答代码表)；
4. HTTP扩展字段类型校验。

按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add +* *t\_http\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### DNS

DNS消息块

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0~31 | DNS域名 | | | | | | | | | | | | | | | |
| 32~63 | DNS域名掩码 | | | | | | | | | | | | | | | |

DNS消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| DNS域名 | 64字节 |  |
| DNS域名掩码 | 64字节 |  |

策略模块在接收到新增DNS的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x08（DNS），则按照DNS消息块的格式对消息体进行解析。

在解析DNS内容的时候，需要对内容和掩码进行校验。如果出现以下两种情况：

1. 掩码中不存在0xff；
2. 掩码的某个字节为0xff，但是对应的内容为0x0。

则返回解析特征码消息失败，会回复ACK[未知错误](#_应答代码表)。

如果解析成功，则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_dns\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 模型

模型消息块各字段描述

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 模型类型数量 | 1字节 | 有多少种模型策略下发，目前只支持手机号和经纬度2种 |
| 类型 | 1字节 | 1-表示手机号；2表示经纬度 |
| 手机号 | 11字节 | 类型为1时有效 |
| 手机号掩码 | 11字节 | 类型为1时有效 |
| 起始经度 | 32字节 | 类型为2时有效 |
| 结束经度 | 32字节 | 类型为2时有效 |
| 起始纬度 | 32字节 | 类型为2时有效 |
| 结束纬度 | 32字节 | 类型为2时有效 |

策略模块在接收到新增模型的消息时，会按照[新增消息格式](#_新增消息)对消息进行[解析验证](#_新增消息)，如果验证通过，且新增消息中的消息块类型为0x09（模型），则按照模型消息块的格式对消息体进行解析。

在解析模型内容的时候，如果为手机号，需要对手机号和掩码进行验证。其他只对信息读取是否成功做判断。

如果解析成功，则会按照[处理消息结构体](#_处理消息结构体)构建消息，消息格式为：*t\_handle\_msg\_info + t\_action\_add + t\_model\_info*。然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)。

##### 组合

组合规则相对比较复杂，会以上述的各类规则的1个或者多个（最多三个）组合组合在一起构成一条消息。

在构建组合规则的[处理消息结构体](#_处理消息结构体)消息时，会根据新增消息中的类型，分别依次顺序的对各类消息进行解析验证，并按照各类型的消息格式进行填充。

然后会将构建好的消息加入待入库的链表*g\_list\_head\_out\_mysql*中。

完成上述步骤后，会回复ACK[操作成功](#_应答代码表)

### 删除消息

删除规则请求的消息体报文格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 用户ID | | | | | | | | 运营商（默认： 0x00） | | | | | | | |
| 1~2 | 规则ID | | | | | | | | | | | | | | | |

删除规则请求的消息体各字段定义

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 用户ID | 1字节 | 登陆用户ID信息 |
| 运营商 | 1字节 |  |
| 规则ID | 4字节 | 同一虚拟用户中的唯一规则标识 |



当接收到删除消息时，程序会根据删除规则请求的消息体报文格式进行消息的解析。然后会验证解析得到的用户ID和认证的用户ID是否一致。如果不一致，回复ACK无效的用户。

此时已经获取到需要删除的用户ID和规则ID，以用户ID+规则ID作为key，进行哈希查找，如果未找到，则直接回复ACK规则不存在。

如果找到，则将数据从原来的链表中删除，并将其加入到待从数据库中删除的链表中。

此时，还需要按照[4.1.2节](#_4.1.2_删除消息)，增量生成删除策略的规则。

### 查询消息

查询规则请求的消息体报文格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 用户ID | | | | | | | | 保留 | | | | | | | |
| 1~2 | 规则ID | | | | | | | | | | | | | | | |

查询规则请求的消息体各字段定义

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段长度 | 字段描述 |
| 用户ID | 1字节 | 登陆用户ID信息 |
| 规则ID | 4字节 | 同一虚拟用户中的唯一规则标识 |



查询的流程和删除的流程很类似。

当接收到查询消息时，程序会根据查询规则请求的消息体报文格式进行消息的解析。然后会验证解析得到的用户ID和认证的用户ID是否一致。如果不一致，回复ACK无效的用户。

此时已经获取到需要查询的用户ID和规则ID，以用户ID+规则ID作为key，进行哈希查找，如果未找到，则直接回复ACK规则不存在。

如果找到，则向DPI发送查询请求，请求的内容为用户ID+规则ID，共5字节。然后接收到DPI的查询回复，获取对应用户和规则ID的命中次数，构建ACK消息，回复ACK。

## 规则生成

规则生成后通知DPI更新是通过TCP的方式。（后续如果需要，可以做成DPI的一个子模块）。

### 增量更新

增量更新指的是每收到一条策略，都在增量更新的规则文件后追加一条新的规则。通知DPI更新的情况有：

1. 当大于所设置的最大增量更新规则数（可在配置中配置），会通知DPI更新。
2. 当下发策略的连接断开时，如果有新增规则，且新增规则总数少于最大全量更新规则数，会通知DPI更新。

#### 新增消息

在新增消息时，如果是非规则，则在每条规则的每个匹配项前加一个 “!”。

新增消息中的action字段的opr字段为add

##### IPv4五元组

当收到一条新增IPv4五元组的消息时：

如果源IP掩码都为0，则不生成源IP相关规则，否则根据源IP和其掩码，构建源IP段。如果时内层IP，则生成规则“ip.src@{[s\_src\_ip,e\_src\_ip]}”，如果是外层IP，则生成规则“ext\_ip.src@{[s\_src\_ip,e\_src\_ip]}” 。

对于目的IP，如果目的IP掩码都为0，则不生成目的IP相关规则，否则根据目的IP和其掩码，构建目的IP段。如果时内层IP，则生成规则“ip.dst@{[s\_dst\_ip,e\_dst\_ip]}”，如果是外层IP，则生成规则“ext\_ip.dst@{[s\_dst\_ip,e\_dst\_ip]}” 。

对于源端口，则生成规则 “pkt.srcport@{[s\_src\_port,e\_src\_port]}”。

对于目的端口，则生成规则 “pkt.dstport@{[s\_src\_port,e\_src\_port]}”。

##### IPv6五元组

Ipv6五元组生成规则时和IPv4类似。

当收到一条新增IPv6五元组的消息时：

如果源IP掩码都为0，则不生成源IP相关规则，否则根据源IP和其掩码，构建源IP段。如果时内层IP，则生成规则“ip.srcv6@{[s\_src\_ip,e\_src\_ip]}”，如果是外层IP，则生成规则“ext\_ip.srcv6@{[s\_src\_ip,e\_src\_ip]}” 。

对于目的IP，如果目的IP掩码都为0，则不生成目的IP相关规则，否则根据目的IP和其掩码，构建目的IP段。如果时内层IP，则生成规则“ip.dstv6@{[s\_dst\_ip,e\_dst\_ip]}”，如果是外层IP，则生成规则“ext\_ip.dstv6@{[s\_dst\_ip,e\_dst\_ip]}” 。

对于源端口，则生成规则 “pkt.srcport@{[s\_src\_port,e\_src\_port]}”。

对于目的端口，则生成规则 “pkt.dstport@{[s\_src\_port,e\_src\_port]}”。

##### 特征码

如果是固定偏移，则生成规则 pkt.payload~"^特征码正则"；如果是浮动偏移，则生成规则 pkt.payload~"特征码正则"。

##### 账号

账号根据账号类型，生成规则pkt.desc~ " acc\_账号正则"。

其他的账号类型，账号的正则格式为：\x\*\*\x\*\*\x\*\*\x\*\*\x\*\*\x\*\*\x\*\*\x\*\*，占8字节，为十六进制字符串，如果不足，则补ff。 生成规则pkt.desc~"ims\_账号正则"，pkt.desc~"msi\_账号正则"，pkt.desc~"ime\_账号正则".

##### 协议和应用

在生成协议和应用规则的时候，首先会根据appid，以appid作为key，查找哈希表g\_list\_head\_for\_pro\_app，如果找到对应的appid，则遍历对应的链表，逐条生成规则，

1. 如果包含URL，即（HOST+ URI），格式为 pkt.payload~"^(GET |POST )[ -~]\*URI[ -~]\*\r\nHost: [ -~]\*host[ -~]\*\r\n"
2. 如果只包含host，格式为 pkt.payload~"Host: [ -~]\*host[ -~]\*\r\n"
3. 如果只包含URI，格式为 pkt.payload~"^(GET |POST )[ -~]\*URI[ -~]\*\r\n "
4. 如果只包含User-Agent， 格式为pkt.payload~" User-Agent: [ -~]\*user-agent[ -~]\*\r\n"

如果未找到，则根据协议名去比对，找到对应的协议ID，生成规则 pkt.desc~"pro\_\x\*\*\x\*\*\x\*\*\x\*\*"

##### 音频和视频

如果是上行，则生成规则 pkt.payload~" Content-Type: [ -~]\*\.音视频\r\n"

如果是下行，则生成规则 pkt.payload~" Content-Type: [ -~]\*\.音视频\r\n"

如果是全部，则上述两类规则都生成

##### HTTP

当method标记位有效时，根据method值的不同，填充的正则的起始部分不同，可能出现的值为"GET", "HEAD", "POST", "PUT", "DELETE", "CONNECT", "OPTIONS", "TRACE"。

当method、host或者url标记位中某一位有效时，生成规则pkt.payload~"method uri host",引号中的host和uri都是正则。

当UA位有效时，生成规则pkt.payload~" User-Agent: [ -~]\*user-agent[ -~]\*\r\n"

当CK位有效时，生成规则pkt.payload~" Cookie: [ -~]\*user-agent[ -~]\*\r\n"

如果有扩展字段，则对于扩展字段中的每个，生成规则pkt.payload~" 扩展字段关键字: [ -~]\*扩展字段内容[ -~]\*\r\n"

上述几类规则的关系是与关系，如果payload的数量大于2，且开启了生成流模式规则的开关时，将pkt.payload修改为flow.payload，并将这类规则写入到专有的规则文件中。

如果CT位有效时，生成规则pkt.payload~" Content-Type: [ -~]\*user-agent[ -~]\*\r\n"。这条规则和上面其他规则的关系是或关系。属于下行规则

##### DNS

生成规则 pkt. desc~" DNS正则"

##### 模型

如果有手机号，生成规则pkt.desc~"pho"

如果有经纬度，生成规则gps.long@{[起始经度,结束经度]}&& gps.lat@{[起始纬度,结束纬度]}

##### 组合

组合规则中，如果payload的匹配项数量大于2，且开启了流模式开关，则生成flow.payload和pkt.payload规则两类，否则只有pkt.payload。

具体的规则生成，参考本节中的其他类型规则生成。

#### 删除消息

删除消息的增量更新规则可以参考[新增规则的增量更新](#_新增消息_1)，其中，删除消息action中的opr字段值为del

### 全量更新

全量更新时，会将增量更新文件清空，同时遍历g\_list\_head\_in\_mysql和g\_list\_head\_out\_mysql两个哈希表，根据类型，生成规则文件，参考[新增消息规则生成](#_新增消息_1)

全量更新会有三种情况：

1. 程序启动时，全量更新一次规则文件；
2. 程序会开启一个线程，当超过所设置的时间（参考配置中的full\_freq值），如果检测到有增量更新的文件，会执行全量更新；
3. 当接收到的规则数量大于等于所设置的全量更新规则数（参考配置中的max\_full\_num值）时，会执行全量更新。

## 数据入库和数据库删除

### 入库

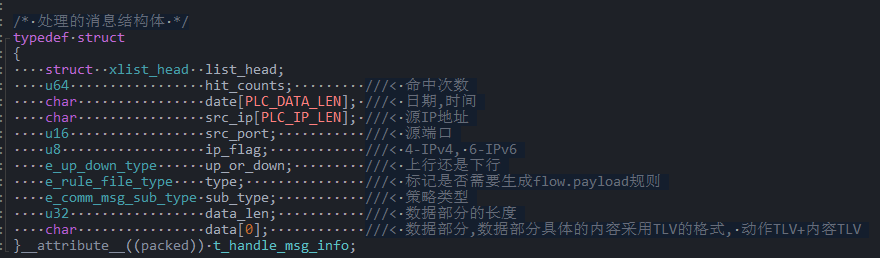
遍历g\_list\_head\_out\_mysql，将每条数据进行base64加密，将加密后的数据，插入数据库；同时将原始数据从g\_list\_head\_out\_mysql中删除，并将其加入到g\_list\_head\_in\_mysql中。

### 删除

遍历g\_list\_del\_for\_mysql，获取规则ID，根据规则ID，将对应的数据从数据库中删除。

## 常用数据结构

#### 处理消息结构体



### 消息块类型

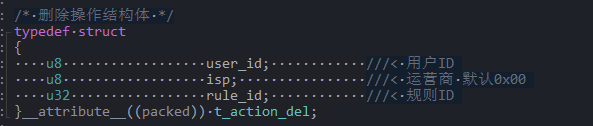
|  |  |
| --- | --- |
| 规则类型 | 编码 |
| IPv4五元组 | 0x01 |
| IPv6五元组 | 0x02 |
| 特征码 | 0x03 |
| 宽带账号 | 0x04 |
| 协议和应用 | 0x05 |
| 音频和视频 | 0x06 |
| HTTP | 0x07 |
| DNS域名 | 0x08 |
| 模型 | 0x09 |

### 消息动作头部

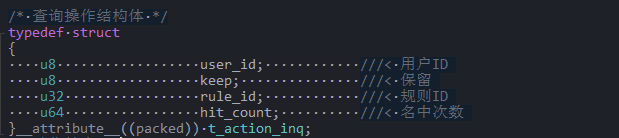
#### 8.2.1 新增消息结构体



#### 8.2.2 删除消息结构体

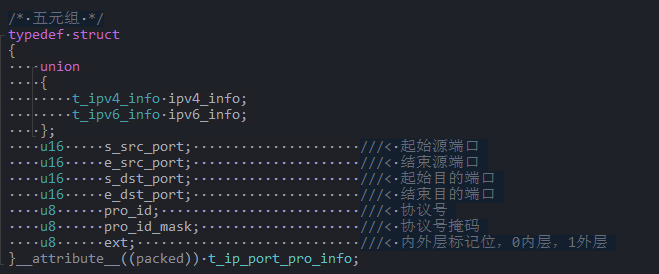


#### 8.2.3 查询消息结构体

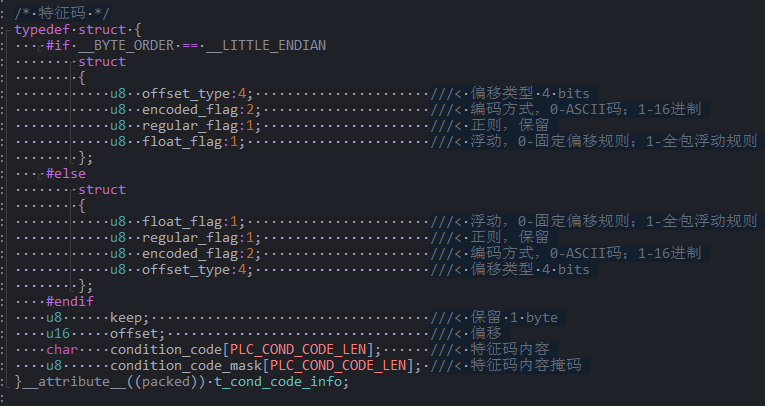


### 各类策略消息数据结构

#### 8.3.1 五元组



#### 8.3.2 特征码



#### 8.3.3 账号



#### 8.3.4 HTTP

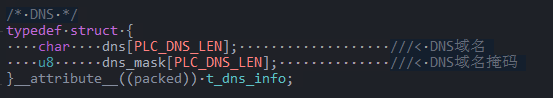


在http消息的扩展字段中，第一个字节为0xff或者0x00。如果为0xff表示有扩展字段信息，如果为0x00表示无扩展字段信息。

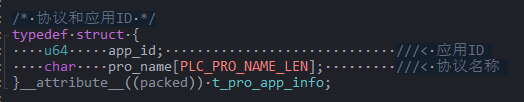
第二个字节表示的是有扩展字段中包含HTTP关键字的个数。

后续的内容则为TLV格式，分别是T表示类型，对应[1.5节](#_HTTP关键字的读取)中的关键字对应ID，L表示的是接下来该类型的有效负载的长度，V表示的是该类型的有效负载内容。

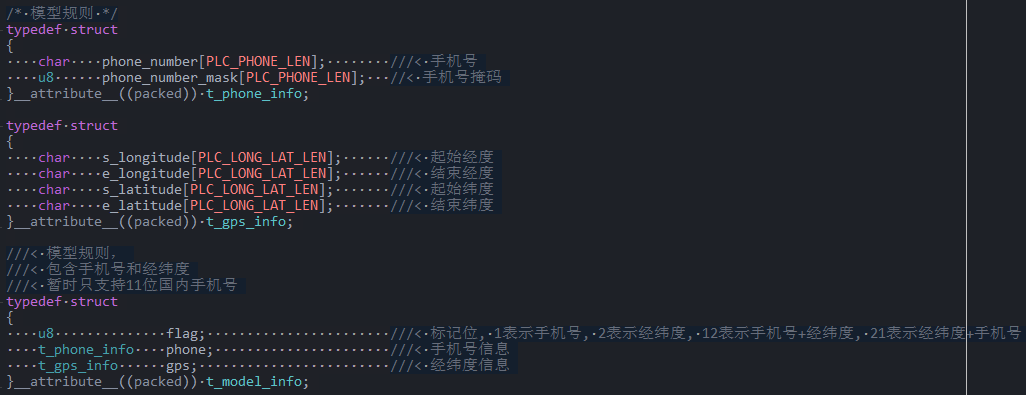
#### 8.3.5 DNS



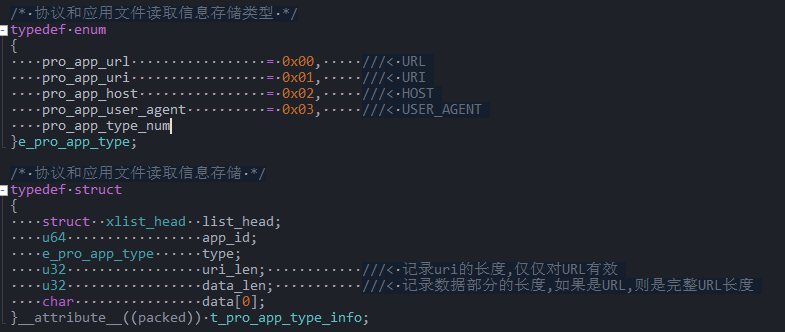
#### 8.3.6 协议和应用



#### 8.3.7 模型



### 协议和应用XML



这里记录了协议和应用的XML文件的本地存储结构。

### 应答代码表

|  |  |  |
| --- | --- | --- |
| 应答代码 | 长度 | 描述 |
| 0x0000 | 2字节 | 操作成功 |
| 0x0001 | 2字节 | 规则已满 |
| 0x0002 | 2字节 | 规则已存在 |
| 0x0003 | 2字节 | 规则不存在 |
| 0x0004 | 2字节 | 无效规则ID |
| 0x0005 | 2字节 | 无效的指令参数 |
| 0x0006 | 2字节 | 无效的消息类型 |
| 0x0007 | 2字节 | 无效用户 |
| 0x0008 | 2字节 | 无效口令 |
| 0x0009 | 2字节 | 当前请求未完成 |
| 0x000a | 2字节 | 当前保存未完成 |
| 0x000b | 2字节 | 服务器忙 |
| 0x000c | 2字节 | 未知错误 |

## 日志

当每收到一次操作且成功时，会记录一条日志到日志文件（日志文件路径可配，参考配置中的log\_file值）中。日志的格式为：

1. 新增消息日志：动作（add），日期时间，源IP地址，源端口，用户id，规则id，规则内容的base64编码；
2. 删除消息日志：动作（del），日期时间，源IP地址，源端口，用户id，规则id；
3. 查询消息日志：动作（inq），日期时间，源IP地址，源端口，用户id，规则id