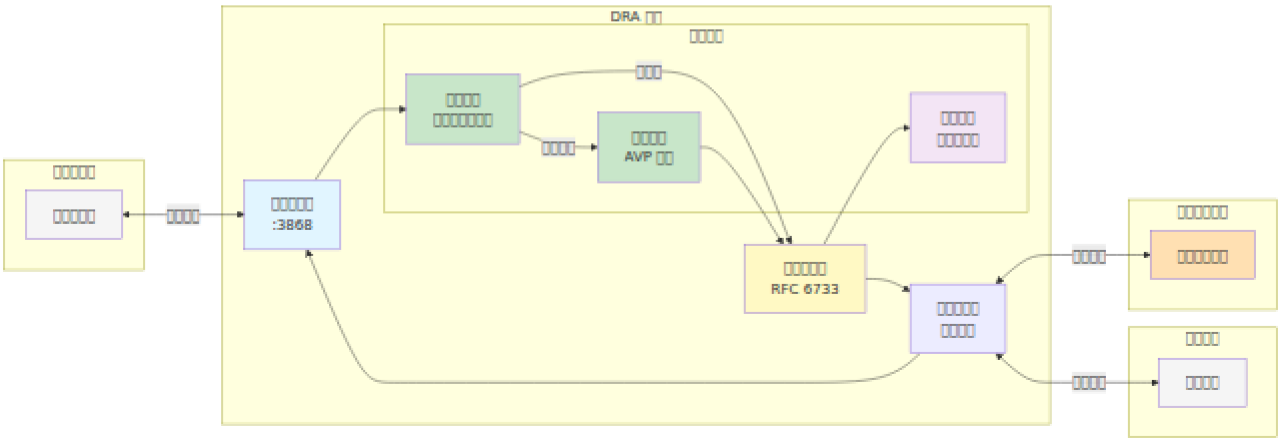
DRA □作指南

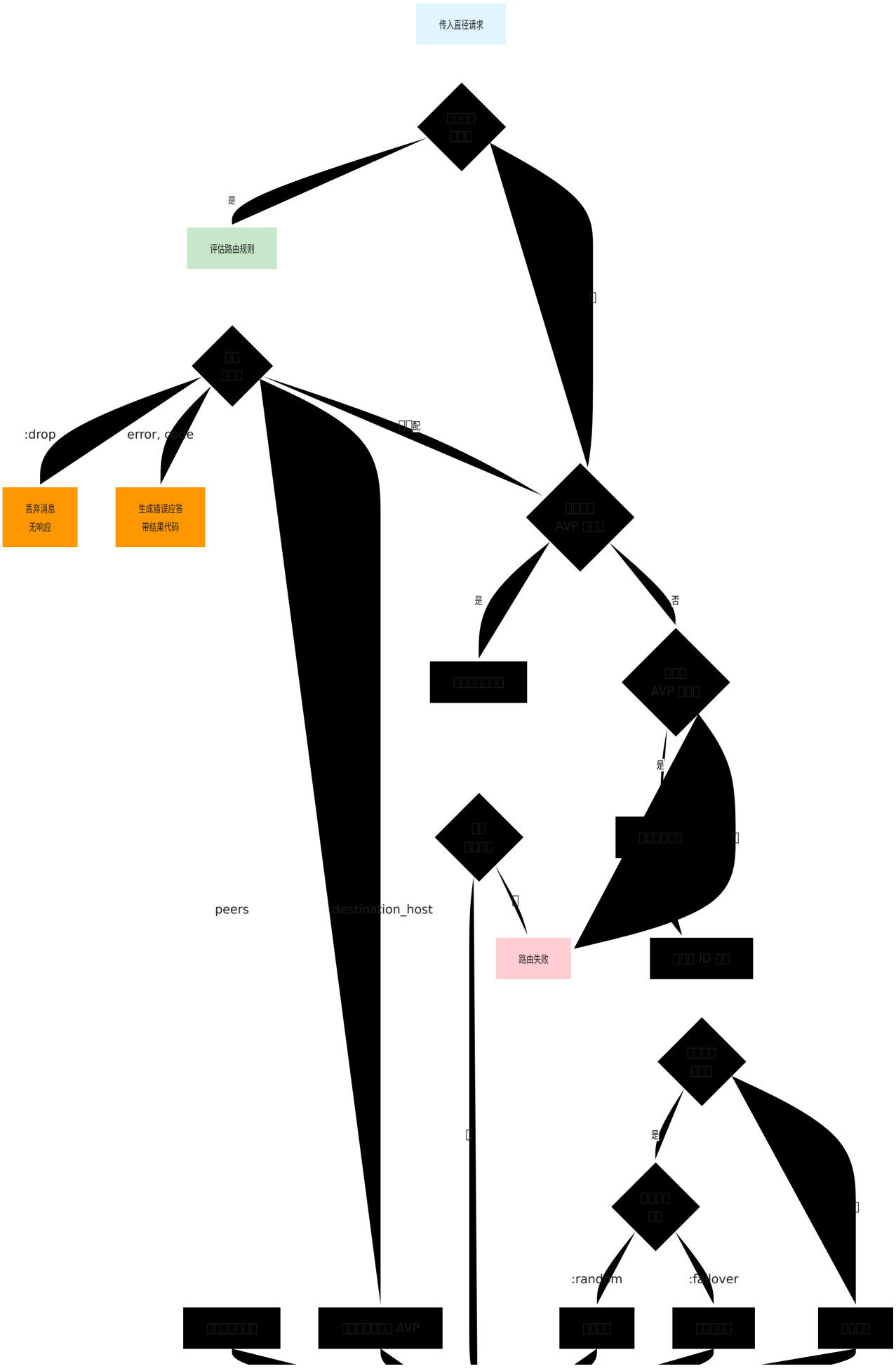
目录

DRA 架构概述



标准直径路由

在没有 高级路由 或 高级转换 模块的情况下，DRA 根据 直径基础协议 (RFC 6733) 执行□准直径路由：


传入直径请求
是
评估路由规则
:drop
error, code
丢弃消息
无响应
生成错误应答
带结果代码
是
否
是
peers
destination_host
路由失败
:random
:failover

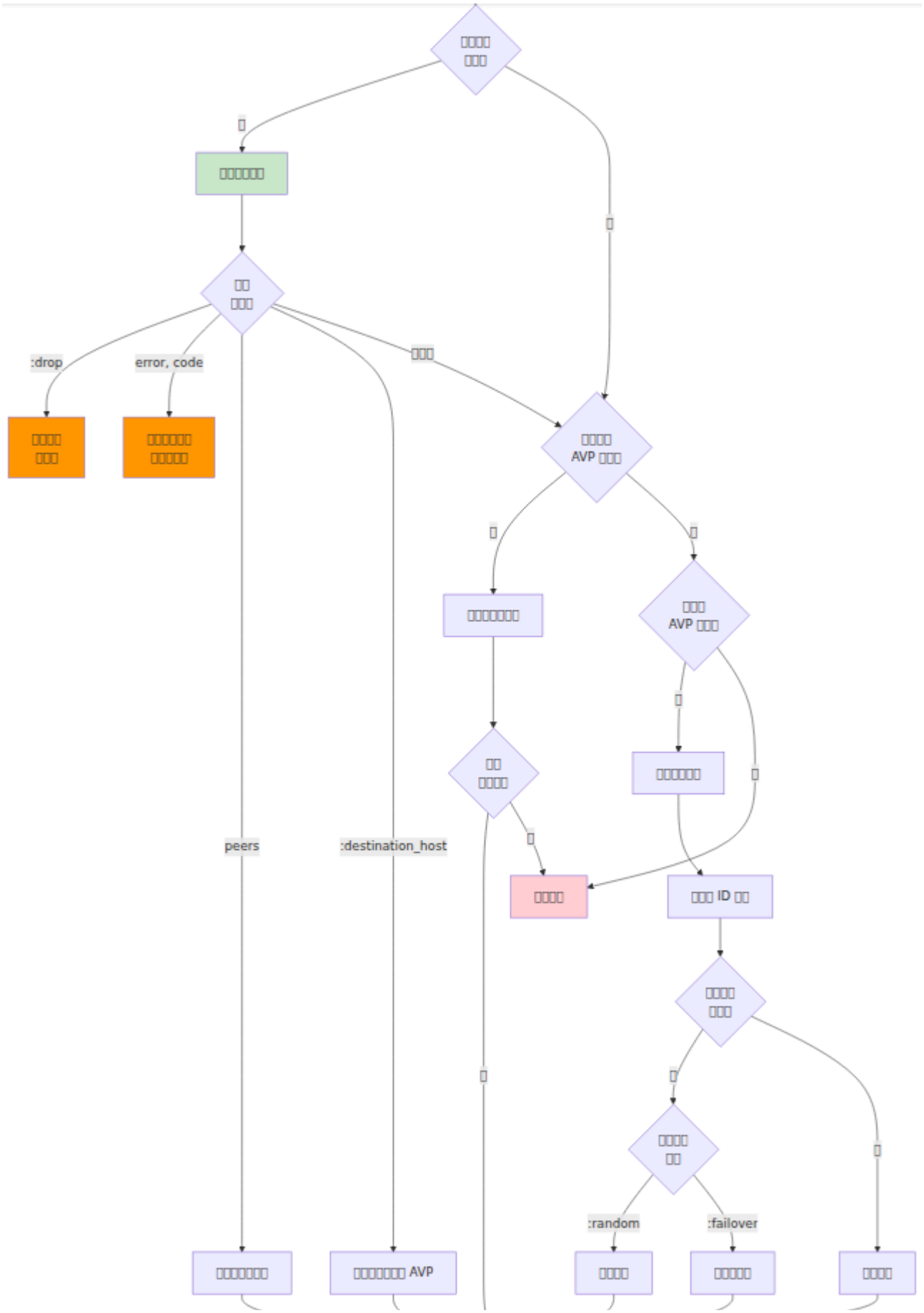
请求路由

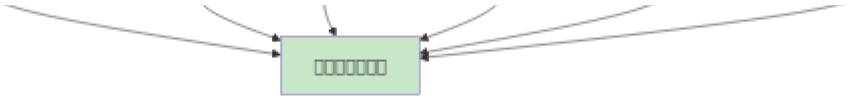
DRA 使用 RFC 6733 第 6.1 节中定义的基于优先级的机制路由请求消息：

1. 目标主机 **AVP (293)** - 如果存在，DRA 直接路由到指定对等
    - 这是最高优先级的路由机制
    - 如果对等未连接，则路由失败
    - 提供明确的主机级路由控制
2. 目标域 **AVP (283)** - 如果目□主机缺失，则根据域路由
    - DRA 选择一个连接的对等，该对等声明支持目□域
    - 当多个对等匹配域时应用负载均衡
    - 基于域的路由允许在多个主机之间灵活性
3. 应用 **ID** - 对等根据支持的直径应用进行过滤
    - 仅考虑声明支持消息的应用 ID 的对等
    - 基于对等连接建立期间的能力交换 (CER/CEA)
    - 参见 常见 3GPP 应用 IDs 以供参考

应答路由

应答数据包使用与请求根本不同的路由机制：


□□□□
□□□
□
□□□□□□□
□
□□
□□□
:drop
error, code
□□□
□□□□
□□□□□□□
□□□□□
AVP □□□
□
□
□□□□□□□□
□□□
AVP □□□
□
□□
□□□□□
□□□□□□□
□
peers
:destination_host
□□□□
□□□ ID □□
□□□□
□□□
□
□□□□
□□
:random
:failover
□□□□□□□□
□□□□□□□□ AVP
□□□□
□□□□□□
□□□□



- 基于会话的路由：应答数据包始终沿请求的反向路径
- 端到端 ID 保留：端到端□识符在所有跳跃中保持不变
- 逐跳路由：DRA 使用逐跳□识符来维护路由状态（在每个跳跃处更改）
- 无规则评估：DRA 不会评估应答的路由规则或 AVP 内容
- 有状态关联：内部路由表跟踪每个请求是由哪个对等发送的

为什么应答不通过高级模块路由：

- 应答路由是确定性的，必须返回到原始对等
- 直径协议要求应答遵循已建立的请求路径
- 应答的路由决策基于原始请求上下文，而不是应答内容
- 这确保了适当的会话管理并防止路由循环

有关应答消息路由详细信息，请参见 RFC 6733 第 6.2 节。

对等选择

当多个对等匹配路由□准时，配置的 `peer_selection_algorithm` 决定选择：

- `:random` - 从可用对等中随机选择（默认）
- `:failover` - 始终选择列表中的第一个对等（基于优先级）
- 对等必须处于 连接状态 才能被选择
- 断开或关闭的对等会被自动排除

标准路由的限制

- 无法根据 AVP 值（例如，IMSI 模式）自定义路由规则
- 无域转换或 AVP 修改
- 无法根据原始对等路由
- 对流量分配的控制有限

高级路由 和 高级转换 模块通过基于规则的路由和数据包操作能力扩展了此□准行为。

□□ DRA □□

DRA □□

□□□□

```
%{
  host: "dra01.example.com",
  realm: "example.com",
  listen_ip: "192.168.1.10",
  listen_port: 3868,
  service_name: :example_dra,
  product_name: "OmniDRA",
  vendor_id: 10415,
  request_timeout: 5000,
  peer_selection_algorithm: :random,
  allow_undefined_peers_to_connect: false,
  log_unauthorized_peer_connection_attempts: true,
  peers: [
    # □□□□...
  ]
}
```

DRA □□□□

□□	□□	□□
`host`	□□□	DRA □ □□□□□□□□□□□
`realm`	□□□	DRA □ □□□
`product_name`	□□□	□ CER/CEA □□□□□□□□□□
`vendor_id`	□□	□□ RFC 6733 □ 5.3.3 □ □□□□□□ ID□10415 = 3GPP□

网络设置

参数	类型	描述
`listen_ip`	字符串或列表	DRA 监听的 IP 地址。对于 SCTP 多宿主，使用 IP 字符串列表（参见 SCTP 多宿主）
`listen_port`	整数	直径连接的 TCP/SCTP 端口（准：3868）
`service_name`	原子	内部 Erlang 服务识符
`request_timeout`	整数	请求/应答对的超时时间（毫秒）（默认：5000）

对等选择设置

参数	类型	描述
`peer_selection_algorithm`	原子	负载均衡算法：`:random`（随机选择）或 `:failover`（第一个对等优先）
`allow_undefined_peers_to_connect`	布尔	允许来自未在配置中的对等的连接（默认：`false`）
`log_unauthorized_peer_connection_attempts`	布尔	记录来自未授权对等的连接尝试

对等配置

`peers` 列表中的每个对等定义一个直径连接：

```
%{
  host: "mme01.operator.com",
  realm: "operator.com",
  ip: "192.168.1.20",
  port: 3868,
  transport: :diameter_tcp,
  tls: false,
  initiate_connection: false
}
```

□□□□

□□	□□	□□
`host`	□□□	□□□ □□□□□□FQDN□ - □□□□□□□□□□□□□□□
`realm`	□□□	□□□□□□□
`ip`	□□□	□□□□□□ IP □□□□□□□
`ips`	□□	SCTP □□□□□ IP □□□□□□□□□□□□□□□ SCTP □□□□□
`port`	□□	□□□□□□□□□□□□□□ 3868□
`transport`	□□	□□□□□□ `:diameter_tcp` □ `:diameter_sctp`
`tls`	□□	□□ TLS □□□□□□□□ `true` □□□□□□□□□ 3869□
`initiate_connection`	□□	`true` □DRA □□□□□□□□ `false` □DRA □□□□□□□□

□□□□

□□□□□ (`initiate_connection: true`)

- DRA □□□□□□□□□
- DRA □□□□□□ TCP/SCTP □□
- □□□□□ HSS□PCRF □□□□□□□□□

- 如果对等不可达，DRA 将重试连接

接受连接 (`initiate_connection: false`)

- DRA 作为直径服务器
- DRA 等待对等连接
- 用于 MME、SGSN、P-GW 连接
- 对等必须在配置中或 `allow_undefined_peers_to_connect: true`

□□□□

```
%{
  host: "dra01.mvno.example.com",
  realm: "mvno.example.com",
  listen_ip: "10.100.1.10",
  listen_port: 3868,
  service_name: :mvno_dra,
  product_name: "OmniDRA",
  vendor_id: 10415,
  request_timeout: 5000,
  peer_selection_algorithm: :random,
  allow_undefined_peers_to_connect: false,
  log_unauthorized_peer_connection_attempts: true,
  peers: [
    # MME - □□ MME □□
    %{
      host: "mme01.operator.example.com",
      realm: "operator.example.com",
      ip: "10.100.2.15",
      port: 3868,
      transport: :diameter_sctp,
      tls: false,
      initiate_connection: false
    },
    # HSS - DRA □□□□
    %{
      host: "hss01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.141",
      port: 3868,
      transport: :diameter_tcp,
      tls: false,
      initiate_connection: true
    },
    # PCRF □ TLS - DRA □□□□□□
    %{
      host: "pcrf01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.22",
      port: 3869,
      transport: :diameter_tcp,
      tls: true,
```

```
      initiate_connection: true
    }
  ]
}
```

重要说明

- 主机名匹配：在 高级路由 规则中的对等主机名必须与此处配置的 `host` 值完全匹配（区分大小写）
- 能力交换：在连接时，对等通过 CER/CEA 消息交换支持的应用
- 应用支持：DRA 声明所有支持的 3GPP 应用（参见 常见 3GPP 应用 IDs）
- 供应商 **ID 10415**：3GPP 应用的准值
- 请求超时：影响 扩展指 TTL（超时 + 5 秒）
- 对等选择：当多个对等匹配路由准时，`peer_selection_algorithm` 决定选择哪个

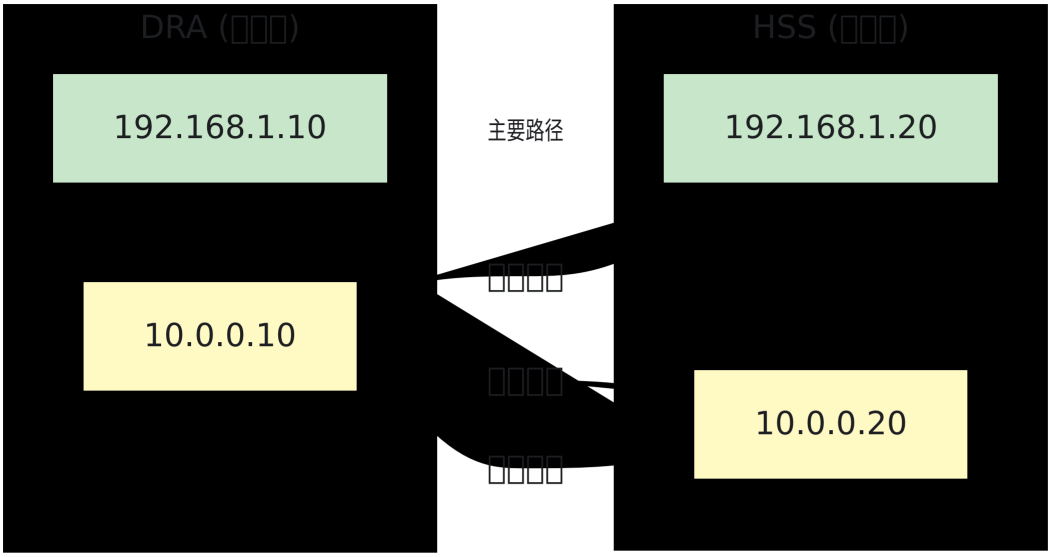
安全考虑

- 在生产环境中设置 `allow_undefined_peers_to_connect: false`
- 启用 `log_unauthorized_peer_connection_attempts: true` 以进行安全监控
- 确保防火墙规则与 `listen_ip` 和 `listen_port` 设置匹配
- 使用 TLS 时验证对等证书

SCTP 多宿主

SCTP 多宿主通过允许端点绑定到多个 IP 地址提供网络冗余。如果主要网络路径失败，SCTP 会自动切换到替代路径，而不会中断直径会话。

工作原理



- SCTP 心跳监控所有网络路径
- 如果主要路径变得不可达，则自动切换
- 在路径切换期间不会中断直径会话
- 内核自动处理路径选择

配置

DRA 监听地址

配置多个本地 IP 地址供 DRA 绑定:

```
%{
  # 单个 IP (向后兼容)
  listen_ip: "192.168.1.10",

  # SCTP 多宿主的多个 IP
  listen_ip: ["192.168.1.10", "10.0.0.10"],

  listen_port: 3868,
  ...
}
```

说明:

- TCP 传输仅使用列表中的第一个 IP
- SCTP 传输绑定到所有指定的 IP
- 单个 IP 字符串格式仍然完全支持

对等配置

配置多个远程 IP 地址以进行对等连接：

```
peers: [
  %{
    host: "hss01.example.com",
    realm: "example.com",
    ip: "192.168.1.20",                       # 主要 IP（必需）
    additional_ips: ["192.168.1.20", "10.0.0.20"],       # 多宿主的所有 IP
    port: 3868,
    transport: :diameter_sctp,
    tls: false,
    initiate_connection: true
  }
]
```

说明：

- `ip` 字段是向后兼容所必需的
- `ips` 字段是可选的；如果省略，则仅使用 `ip`
- 对于 SCTP，在 `ips` 列表中包含主要 IP
- 对于 TCP，仅使用 `ip`（TCP 不支持多宿主）

完整示例

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: ["192.168.1.10", "10.0.0.10"],  # 多宿主 DRA
    listen_port: 3868,
    host: "dra01",
    realm: "example.com",
    product_name: "OmniDRA",
    vendor_id: 10415,
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: false,
    peers: [
      # 多宿主 HSS 连接
      %{
        host: "hss01.example.com",
        realm: "example.com",
        ip: "192.168.1.20",
        additional_ips: ["192.168.1.20", "10.0.0.20"],
        port: 3868,
        transport: :diameter_sctp,
        tls: false,
        initiate_connection: true
      },
      # 单宿主 MME（向后兼容）
      %{
        host: "mme01.example.com",
        realm: "example.com",
        ip: "192.168.1.30",
        port: 3868,
        transport: :diameter_sctp,
        tls: false,
        initiate_connection: false
      }
    ]
  }
```

要求

- 必须加载 SCTP 内核模块（Linux 上的 `lksctp-tools` 包）
- 所有 IP 地址必须可路由到/从对等
- 防火墙规则必须允许所有配置的 IP 上的 SCTP 流量
- 两个端点都应配置为多宿主以实现完全冗余

限制

- TCP 传输不支持多宿主（仅使用主要 IP）
- SCTP 多宿主上的 TLS 可能存在兼容性限制
- 路径切换的时机取决于内核 SCTP 参数

参考表

常见 **3GPP** 应用 **IDs**

应用 **ID**	接口	描述
16777251	S6a/S6d	MME/SGSN 到 HSS 身份验证和订阅数据
16777252	S13/S13'	MME 到 EIR 设备身份检查
16777238	Gx	PCEF 到 PCRF 策略和计费控制
16777267	S9	主 PCRF 到被访 PCRF 漫游策略
16777272	Sy	PCRF 到 OCS 会话绑定
16777216	Cx	I-CSCF/S-CSCF 到 HSS IMS 注册
16777217	Sh	AS 到 HSS IMS 用户数据
16777236	SLg	MME/SGSN 到 GMLC 定位服务
16777291	SLh	GMLC 到 HSS 定位订户信息
16777302	S6m	MTC-IWF 到 HSS/HLR 用于 M2M 设备
16777308	S6c	SMS-SC/IP-SM-GW 到 HSS SMS 路由
16777343	S6t	SCEF 到 HSS 监控事件
16777334	Rx	AF 到 PCRF 媒体授权

常见 **AVP** 代码

代码	**AVP** 名称	类型	用法
1	用户名	UTF8String	订阅者□识符（IMSI 在 3GPP 中）
264	Origin-Host	DiameterIdentity	发起对等主机名
268	Result-Code	Unsigned32	□准结果代码
283	Destination-Realm	DiameterIdentity	目□域
293	Destination-Host	DiameterIdentity	目□主机（可选）
296	Origin-Realm	DiameterIdentity	源域
297	Experimental-Result	Grouped	供应商特定结果代码

常见命令代码

命令代码是直径消息头的一部分，而不是 AVP：

代码	命令名称	描述
257	CER/CEA	能力交换请求/应答
258	RAR/RAA	重新认证请求/应答
274	ASR/ASA	中止会话请求/应答
275	STR/STA	会话终止请求/应答
280	DWR/DWA	设备看门狗请求/应答
282	DPR/DPA	断开对等请求/应答
316	ULR/ULA	更新位置请求/应答 (S6a)
317	CLR/CLA	取消位置请求/应答 (S6a)
318	AIR/AIA	身份验证信息请求/应答 (S6a)
321	PUR/PUA	清除 UE 请求/应答 (S6a)

高级路由模块

高级路由模块提供灵活的基于规则的消息路由能力，支持复杂的匹配条件。

重要：此模块仅评估 传入的直径请求数据包（不包括应答数据包）。应答数据包遵循已建立的会话路由返回到原始对等 - 有关详细信息，请参见 应答路由。

配置

启用模块并在配置中定义路由规则：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: <规则标识符>
      match: <匹配范围>
      filters: [<过滤器列表>]
      route:
        peers: [<□等列表>]
```

参数

参数	描述
`enabled`	设置为 `True` 以激活模块
`rule_name`	路由规则的唯一□识符
`match`	过滤器如何组���: `:all`（AND 逻辑 - 所有过滤器必须匹配），`:any`（OR 逻辑 - 至少一个过滤器必须匹配），`:none`（NOR 逻辑 - 不能有过滤器匹配）
`filters`	过滤条件列表（参见 可用过滤器）
`route`	路由操作（参见 路由操作）

路由□作

`route` 参数支持多种操作:

路由到对等

```
route:
  peers: [peer01.example.com, peer02.example.com]
```

路由到指定的对等主机名。对等必须:

- 在 DRA 的直径对等配置中定义

- 与配置的完全主机名（区分大小写）匹配
- 当前连接以成功路由（断开连接的对等将被跳过）

路由到目标主机 **AVP**

```
route: :destination_host
```

路由到消息中的 目□主机 AVP (293) 指定的对等。如果目□主机 AVP 缺失，则路由回退到正常行为。

丢弃流量

```
route: :drop
```

静默丢弃消息而不发送任何响应。用于：

- 流量过滤和黑洞处理
- 阻止不需要的请求
- 通过丢弃过量流量进行速率限制

行为：

- 消息在 DRA 中被丢弃（不转发）
- 不向请求对等发送应答消息
- 实现 Erlang Diameter `:discard` 行为
- 指□: `diameter_advanced_routing_drop_count_total` (参见 Prometheus 指□)

生成错误响应

```
route: {:error, 3004}
```

生成带有指定结果代码的直径错误应答并将其发送回请求对等。常见结果代码：

- `3002` - DIAMETER_UNABLE_TO_DELIVER（路由不可用）
- `3003` - DIAMETER_REALM_NOT_SERVED（域不支持）
- `3004` - DIAMETER_TOO_BUSY（过载保护，速率限制）
- `5012` - DIAMETER_UNABLE_TO_COMPLY（一般拒绝）

行为：

- DRA 生成带有指定结果代码的错误应答
- 应答包括 Origin-Host、Origin-Realm、Session-Id（由直径自动填充）
- 消息不会转发到任何对等
- 实现 Erlang Diameter `{:protocol_error, code}`（等同于 `{:answer_message, code}`）
- 指□: `diameter_advanced_routing_error_count_total`（参见 Prometheus 指□）

可用过滤器

标准过滤器

在 高级路由 和 高级转换 中均可用：

- **`:application_id`** - 匹配直径应用 ID（参见 应用 ID 参考）
  - 单个值: `{:application_id, 16777251}`（S6a/S6d）
  - 多个值: `{:application_id, [16777251, 16777252]}`（S6a 或 S6b）
- **`:command_code`** - 匹配直径命令代码
  - 单个值: `{:command_code, 318}`（AIR 请求）
  - ���个值: `{:command_code, [317, 318]}`（ULR 或 AIR）
- **`:avp`** - 匹配 AVP 值（参见 AVP 代码参考）
  - 精确匹配: `{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}`
  - 正则匹配: `{:avp, {1, ~r"999001.*"}}`
  - 多个模式: `{:avp, {1, ["505057001313606", ~r"999001.*", ~r"505057.*"]}}`
  - 任何值（存在性检查）: `{:avp, {264, :any}}`

路由特定过滤器

仅在 高级路由 中可用：

- **`:via_peer`** - 匹配请求来自的对等

- 单个对等：`{:via_peer, "omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org"}`
- 多个对等：`{:via_peer, ["omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org", "omnitouch-lab-dra02.epc.mnc001.mcc001.3gppnetwork.org"]}`
- 任何对等：`{:via_peer, :any}`

转换特定过滤器

仅在 高级转换 中可用：

- **`:to_peer`** - 匹配预定的目□对等（仅请求数据包）
  - 单个对等：`{:to_peer, "dra01.omnitouch.com.au"}`
  - 多个对等：`{:to_peer, ["dra01.omnitouch.com.au", "dra02.omnitouch.com.au"]}`
- **`:from_peer`** - 匹配发送应答的对等（仅应答数据包）
  - 单个对等：`{:from_peer, "hss-01.example.com"}`
  - 多个对等：`{:from_peer, ["hss-01.example.com", "hss-02.example.com"]}`
- **`:packet_type`** - 匹配数据包方向
  - 请求：`{:packet_type, :request}`
  - 应答：`{:packet_type, :answer}`

重要过滤器说明

- **AVP** 过滤器：推荐仅用于简单 AVP（用户名、Origin-Host、Destination-Realm 等）
  - 分组 AVP 不支持，将无法匹配
  - 复杂的二进制值 不支持
  - 使用格式：`{:avp, {code, value}}`
- 列表运算符：对所有过滤器值（除了 `:packet_type`）支持
  - 当使用列表时，应用 **OR** 逻辑 在列表内
  - 示例：`{:command_code, [317, 318]}` 匹配命令代码 317 或 318

- 特殊值：
  - `:any` - 匹配任何值（检查 AVP 存在性）
  - 示例：`{:avp, {264, :any}}` 匹配如果存在任何值的 Origin-Host AVP

路由示例

示例 **1**：通过对等路由

根据消息到达的 DRA 路由消息：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: temporary_until_cutover_s6a_via_to_local_hss
      match: ":all"
      filters:
        - '{:application_id, 16777251}'
        - '{:via_peer, ["omnitouch-lab-
dra01.epc.mnc001.mcc001.3gppnetwork.org", "omnitouch-lab-
dra02.epc.mnc001.mcc001.3gppnetwork.org"]}'
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'
      route:
        peers: [omnitouch-lab-
hss01.epc.mnc001.mcc001.3gppnetwork.org, omnitouch-lab-
hss02.epc.mnc001.mcc001.3gppnetwork.org]
```

工作原理：将通过特定 DRA 对等到达的 S6a 流量路由到本地 HSS 节点。

示例 **2**：入境漫游与模式匹配

根据 IMSI 模式路由漫游流量：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: inbound_s6a_roaming_to_dcc
      match: ":all"
      filters:
        - '{:application_id, 16777251}'
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'
        - '{:avp, {1, ["505571234567", ~r"999001.*"]}}'
      route:
        peers: [dra01.omnitouch.com.au, dra02.omnitouch.com.au]
```

工作原理：将来自特定 Origin-Realm 的 S6a 消息与匹配的 IMSI 模式路由到指定的 DRA 对等。

示例 **3**：使用 **:destination_host** 的动态路由

路由到消息中的目□主机 AVP 值：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: route_to_specified_destination_host
      match: ":all"
      filters:
        - '{:avp, {1, [~r"90199.*"]}}'  # 匹配 IMSI 模式
      route: :destination_host
```

工作原理：

- 当过滤器匹配时，路由到目□主机 AVP (293) 指定的对等
- 如果目□主机 AVP 缺失，则匹配被视为失败并回退到正常路由
- 在��送方指定确切目□时有助于荣誉路由

示例 **4**：丢弃不需要的流量

丢弃来自特定 IMSI 范围的流量：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: drop_test_subscribers
      match: ":all"
      filters:
        - '{:application_id, 16777251}'  # S6a
        - '{:avp, {1, [~r"999999.*"]}}'  # 测试 IMSI 范围
      route: :drop
```

工作原理:

- 匹配以 999999 开头的 S6a 消息的 IMSI
- 静默丢弃消息而不发送任何响应
- 有助于过滤测试流量或阻止特定订阅者范围
- 参见 Prometheus 指[] 以监控丢弃的流量

示例 **5**: 通过错误响应进行速率限制

对于特定流量模式返回 DIAMETER_TOO_BUSY:

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: rate_limit_high_volume_peer
      match: ":all"
      filters:
        - '{:via_peer, "mme-overloaded-01.example.com"}'
        - '{:application_id, 16777251}'
      route: {:error, 3004}
```

工作原理:

- 匹配来自特定过载对等的 S6a 流量
- 返回 DIAMETER_TOO_BUSY (3004) 错误响应
- 请求对等收到错误并应当退避
- 有助于过载保护和速率限制
- 参见 Prometheus 指[] 以监控错误响应

示 6：按命令阻止特定错误响应

使用适当的错误代码阻止特定命令类型：

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: block_purge_requests
      match: ":all"
      filters:
        - '{:application_id, 16777251}'  # S6a
        - '{:command_code, 321}'         # PUR (清除 UE 请求)
      route: {:error, 5012}
```

工作原理：

- 匹配 S6a 清除 UE 请求消息
- 返回 DIAMETER_UNABLE_TO_COMPLY (5012) 错误
- 阻止特定操作而不静默丢弃流量
- 有助于选择性禁用某些直径命令

高级转换模块

高级转换模块使根据匹配准动态修改直径消息 AVP 成为可能。有关规则如何评估的详细信息，请参见 规则处理。

配置

启用模块并定义转换规则：

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: <规则标识符>
      match: <匹配范围>
      filters: [<过滤器列表>]
      transform:
        action: <转换操作>
        avps: [<avp 修改>]
```

参数

参数	描述
`enabled`	设置为 `True` 以激活模块
`rule_name`	转换规则的唯一□识符
`match`	过滤器如何组合：`:all`（AND 逻辑），`:any`（OR 逻辑），`:none`（NOR 逻辑） - 参见 过滤器逻辑
`filters`	过滤条件列表（参见 可用过滤器）
`transform.action`	转换类型（`:edit`、`:remove` 或 `:overwrite`）
`transform.avps`	要应用的 AVP 修改列表（参见 AVP 代码参考）

转换□作

请求数据包（直径请求）

- **`:edit`** - 修改现有 AVP 值
  - 仅修改消息中存在的 AVP
  - 如果 AVP 不存在，则不进行更改
- **`:remove`** - 从消息中删除 AVP
- **`:overwrite`** - 替换整个 AVP 结构

    - 需要 `dictionary` 参数指定直径字典（例如，`:diameter_gen_3gpp_s6a`）

应答数据包（直径应答）

- `:remove` - 从消息中删除 AVP
- `:overwrite` - 替换整个 AVP 结构
    - 需要 `dictionary` 参数

重要：如果没有规则匹配，数据包将透明地通过，而不进行任何转换。

AVP 修改语法

标准修改：

- `{:avp, {<code>, <new_value>}}` - 将 AVP 设置为新值

删除 **AVP**：

- `{:avp, {<code>, :any}}` - 按 ID 删除 AVP（无论当前值如何均删除）
- 注意：基于 avp_id 删除是支持的；基于 AVP 内容删除是不支持的

使用字典进行覆盖：

```
transform: %{
  action: :overwrite,
  dictionary: :diameter_gen_3gpp_s6a,
  avps: [{:avp, {:"s6a_Supported-Features", {:"s6a_Supported-
Features", 10415, 1, 3221225470, []}}}]
}
```

转换示例

示例 **1**：基于对等的域重写

根据消息被路由的地方重写目□域：

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: rewrite_s6a_destination_realm_for_Operator_X
      match: ":all"
      filters:
        - '{:to_peer, ["dra01.omnitouch.com.au",
"dra02.omnitouch.com.au"]}'
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'
        - '{:avp, {1, [~r"9999999.*"]}}'
      transform:
        action: ":edit"
        avps:
          - '{:avp, {283, "epc.mnc999.mcc999.3gppnetwork.org"}}'
```

工作原理：当 S6a 请求被路由到特定 DRA 对等并匹配 IMSI 模式时，重写目域为运营商 X 网络。

示例 **2**：多运营商路由与转换

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name:
rewrite_s6a_destination_realm_for_roaming_partner_ausie
      match: ":all"
      filters:
        - '{:to_peer, ["dra01.omnitouch.com.au",
"dra02.omnitouch.com.au"]}'
        - '{:avp, {296, "epc.mnc057.mcc505.3gppnetwork.org"}}'
        - '{:avp, {1, [~r"50557.*"]}}'
      transform:
        action: ":edit"
        avps:
          - '{:avp, {283, "epc.mnc030.mcc310.3gppnetwork.org"}}'
```

工作原理：将不同 IMSI 订阅者范围路由到适当的网络域，基于 IMSI 模式。第一个匹配规则胜出（参见 执行顺序）。

示例 **3**：**MVNO** 域重写

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: rewrite_s6a_destination_realm_for_single_sub
      match: ":all"
      filters:
        - '{:to_peer, ["dra01.omnitouch.com.au",
"dra02.omnitouch.com.au"]}'
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'
        - '{:avp, {1, ["505057000003606"]}}'  # 精确 IMSI 匹配
      transform:
        action: ":edit"
        avps:
          - '{:avp, {283, "epc.mnc001.mcc001.3gppnetwork.org"}}'
```

工作原理：为特定 MVNO 订阅者转换目域到其托管核心网络。

示例 **4**：仅请求的转换与数据包类型过滤

仅转换请求数据包（不应答数据包）：

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: Tutorial_Rule_AIR
      match: ":all"
      filters:
        - '{:application_id, 16777251}'
        - '{:command_code, 318}'
        - '{:packet_type, :request}'
        - '{:avp, {1, "999999000000001"}}'
        - '{:avp, {264, :any}}'  # Origin-Host 必须存在且值可以是任何
      transform:
        action: ":edit"
        avps:
          - '{:avp, {1, "999999000000002"}}'
```

工作原理：

- 仅匹配 S6a AIR 请求 数据包（不应答数据包）

- 检查用户名称（AVP 1）等于"999999000000001"
- 验证Origin-Host（AVP 264）存在且值可以是任何
- 将用户名称重写为"999999000000002"
- 如果AVP不存在，则不进行更改

示例5：删除AVP

从消息中删除特定AVP：

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: remove_user_name_avp
      match: ":all"
      filters:
        - '{:application_id, 16777251}'
      transform:
        action: ":remove"
        avps:
          - '{:avp, {1, :any}}'  # 无论值如何删除用户名
```

工作原理：从所有S6a消息中删除用户名AVP（代码1），无论其当前值如何。

示例6：在应答数据包上覆盖分组AVP

使用字典支持修改应答数据包中的复杂分组AVP：

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: add_sos_apn_to_ula
      match: ":all"
      filters:
        - '{:application_id, 16777251}'       # S6a/S6d
        - '{:command_code, 316}'              # ULA (□□□□□□)
        - '{:packet_type, :answer}'           # □□□□□□
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'  #
Origin-Realm
      transform:
        action: ":overwrite"
        dictionary: ":diameter_gen_3gpp_s6a"
        avps:
          - '{:avp, {:"s6a_APN-Configuration-Profile",
              {:"s6a_APN-Configuration-Profile", 1, 0, [
                {:"s6a_APN-Configuration", 1, 0, "internet", [],
                  [{:"s6a_EPS-Subscribed-QoS-Profile", 9,
                    {:"s6a_Allocation-Retention-Priority", 1, [0],
[0], []}, []}],
                  [1], [], [], [1], ["0800"],
                  [{:s6a_AMBR, 4200000000, 4200000000, [], [],
[]}],
                  [], [], [], [], [], [], [], [], [], [], [], [],
[], [], []},
                {:"s6a_APN-Configuration", 2, 0, "ims", [],
                  [{:"s6a_EPS-Subscribed-QoS-Profile", 5,
                    {:"s6a_Allocation-Retention-Priority", 1, [0],
[1], []}, []}],
                  [0], [], [], [1], ["0800"],
                  [{:s6a_AMBR, 4200000000, 4200000000, [], [],
[]}],
                  [], [], [], [], [], [], [], [], [], [], [], [],
[], [], []},
                {:"s6a_APN-Configuration", 3, 0, "sos", [],
                  [{:"s6a_EPS-Subscribed-QoS-Profile", 5,
                    {:"s6a_Allocation-Retention-Priority", 1, [0],
[1], []}, []}],
                  [1], [], [], [1], ["0800"],
                  [{:s6a_AMBR, 4200000000, 4200000000, [], [],
[]}],
                  [], [], [], [], [], [], [], [], [], [], [], [],
```

```
[], [], []}
              ], []}
            }}'
```

工作原理：

- 匹配来自特定 Origin-Realm 的 S6a 更新位置应答 (ULA) 数据包
- 使用 `:overwrite` 操作替换整个 APN 配置文件分组 AVP
- 需要 **`dictionary`** 参数以正确编码复杂的分组 AVP 结构
- 添加三个 APN 配置："internet"（上下文 1）、"ims"（上下文 2）和 "sos"（上下文 3）
- 每个 APN 包括 QoS 配置文件、带宽限制（AMBR）和 PDN 类型设置
- 转换确保紧急服务（SOS）APN 为来自该域的所有订阅者提供

何时使用 **`:overwrite`** 与字典：

- 修改具有嵌套结构的分组 AVP（如 APN 配置文件）
- 添加或重组复杂的 3GPP 订阅数据
- 当 `:edit` 操作无法处理 AVP 复杂性时
- 字典必须与直径应用匹配（S6a 的 `:diameter_gen_3gpp_s6a` 等）

重要说明：

- `:overwrite` 替换整个 AVP，而不仅仅是单个字段
- AVP 结构必须与字典定义完全匹配
- 不正确的结构将导致编码失败和丢弃数据包
- 这是一个高级功能 - 在测试环境中首先彻底验证

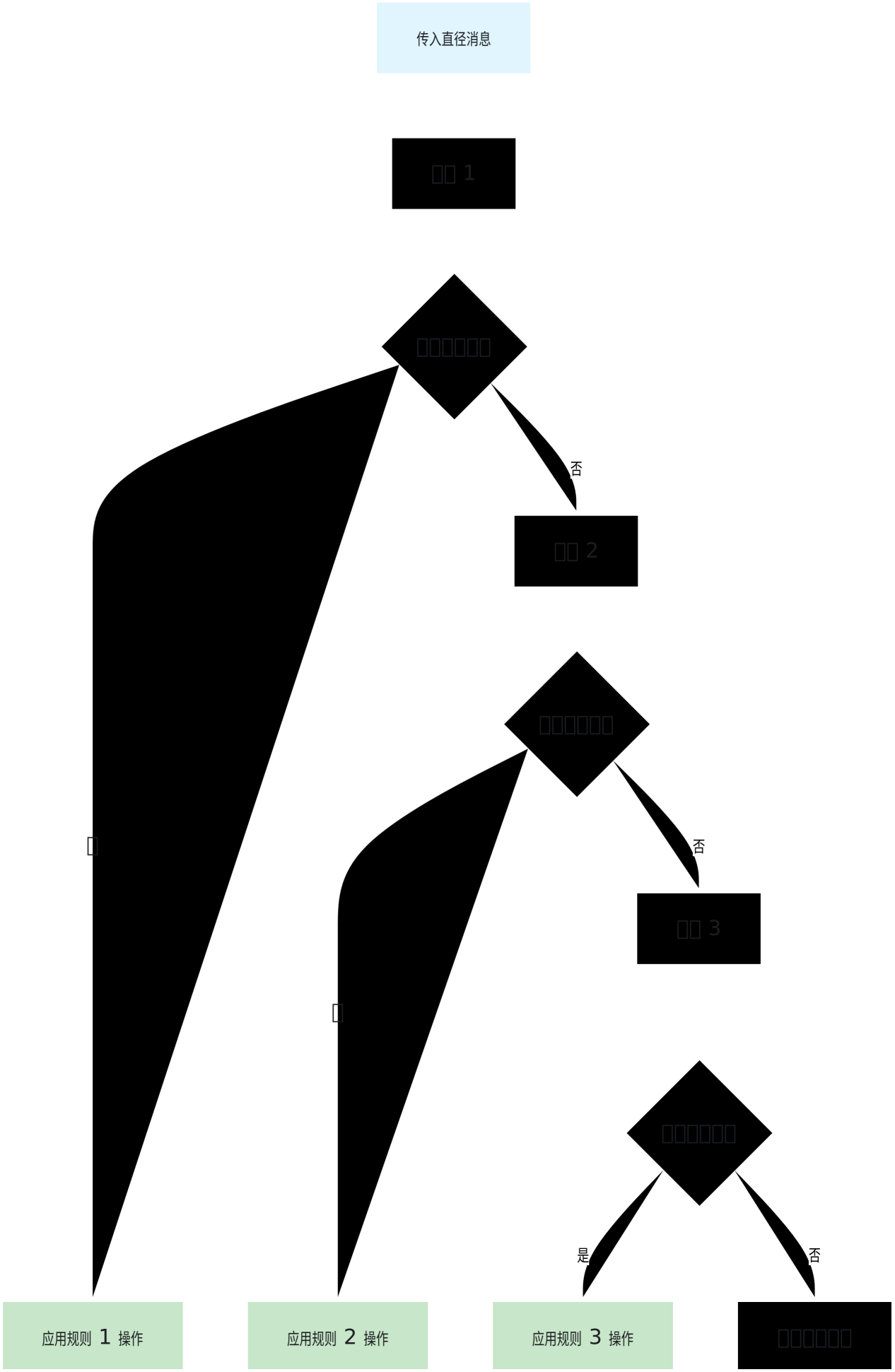
用例

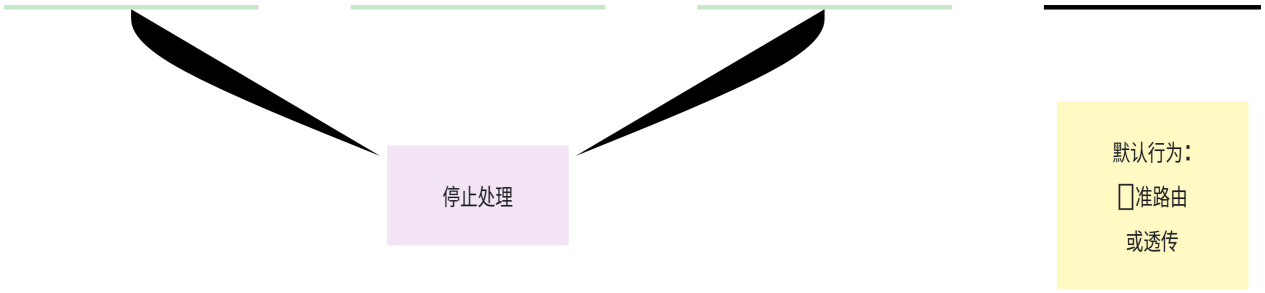
- **MVNO** 支持：将虚拟运营商流量路由到托管核心网络
- 网络迁移：逐步将订阅者重定向到新基础设施
- 域翻译：在漫游合作伙伴之间转换不同的命名方案
- 多租户：按域隔离订阅者群体
- 运营商路由：根据 IMSI 范围将流量直接路由到正确的运营商网络

规则处理

适用于 高级路由 和 高级转换 模块。

□□□□


传入直径消息
□□ 1
□□□□□□□
否
□□ 2
□□□□□□□
否
□□ 3
□□□□□□□
是
否
应用规则 1 操作
应用规则 2 操作
应用规则 3 操作
□□□□□□□



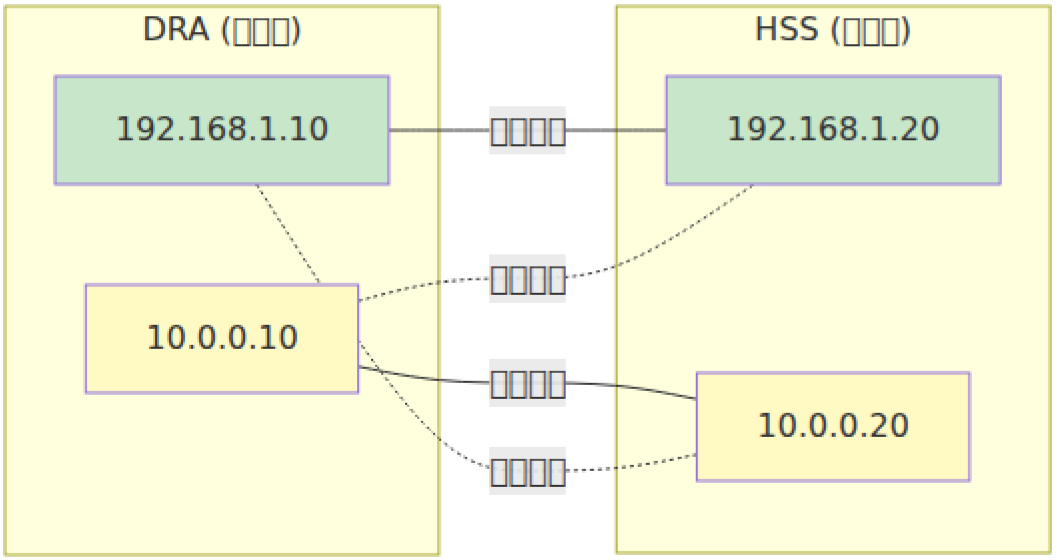
1. 规则按配置中定义的 从上到下 的顺序进行评估
2. 规则内的过滤器根据 `match` 参数进行评估（`:all`、`:any` 或 `:none`）
3. 第一个匹配规则胜出 - 后续规则不再评估
4. 如果没有规则匹配，则使用默认路由/透传行为

过滤器逻辑

`match` 参数决定了过滤器如何组合：

match: :all (AND 逻辑)

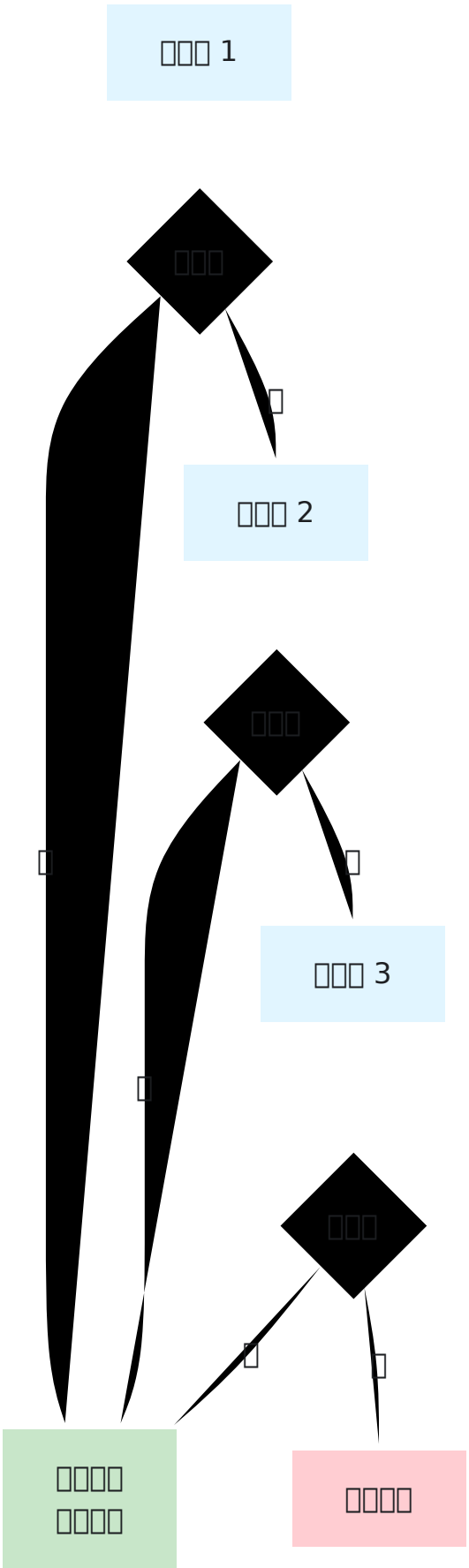
所有过滤器必须匹配才能使规则成功。



示例：对于 3 个过滤器，`filter1 AND filter2 AND filter3` 必须全部为真。

match: :any (OR 逻辑)

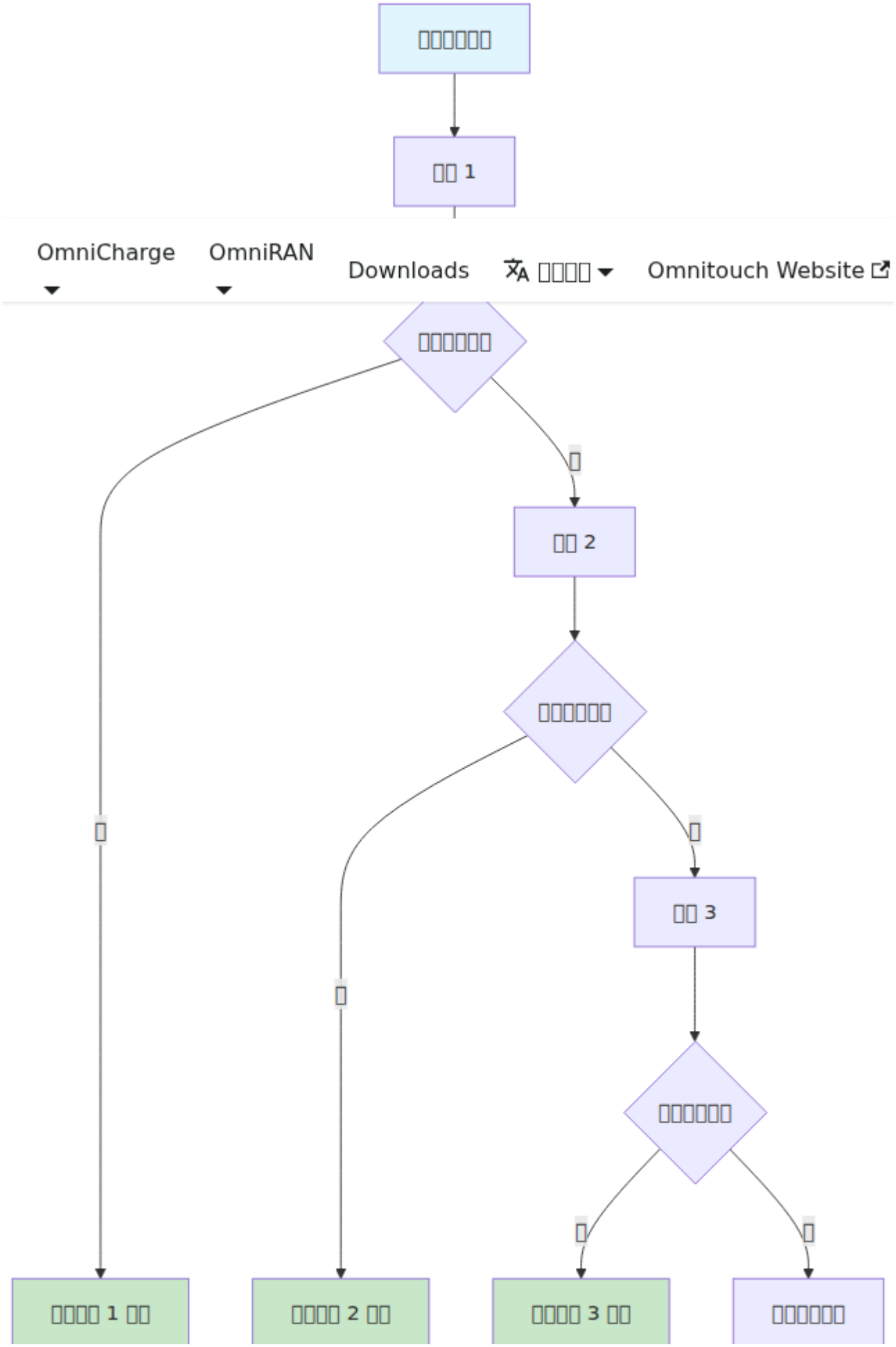
至少一个过滤器必须匹配才能使规则成功。

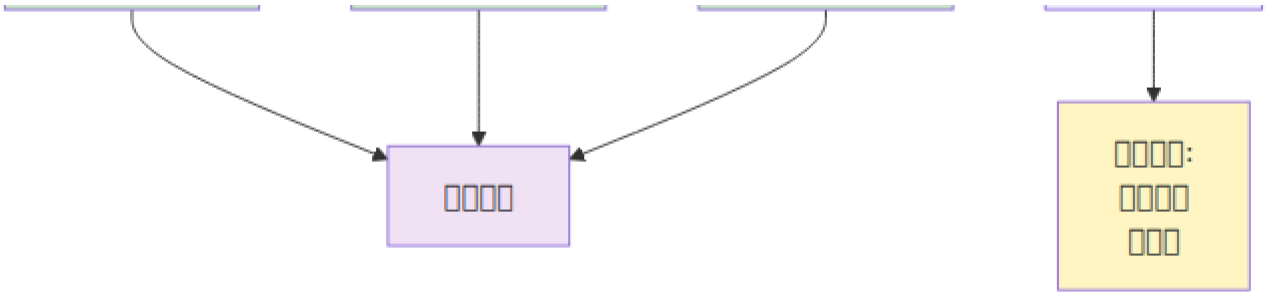


□□□□□ 3 □□□□□□ `filter1 OR filter2 OR filter3` □□□□□□□□□□□

match: :none (NOR 逻辑)

规则成功的前提是没有过滤器可以匹配（反向匹配）。


□□□□□□□
□□ 1
OmniCharge
OmniRAN
Downloads
□□□□
Omnitouch Website
□□□□□□□
□
□□ 2
□□□□□□□
□
□
□□ 3
□□□□□□□
□
□
□□□□ 1 □□
□□□□ 2 □□
□□□□ 3 □□
□□□□□□□



示例：对于 3 个过滤器，`NOT filter1 AND NOT filter2 AND NOT filter3`（全部必须失败）。

附加说明：

在使用过滤器值中的列表运算符时（例如，`{:avp, {1, ["value1", "value2"]}}`），值使用 **OR** 逻辑（任何一个可以匹配）。

正则表达式模式

使用 `~r"pattern"` 语法进行正则匹配：

- `~r"999001.*"` - 匹配以 999001 开头的 IMSI
- `~r"^310[0-9]{3}.*"` - 匹配具有特定 MNC 模式的 IMSI
- `~r".*test$"` - 匹配以 "test" 结尾的值

最佳实践

1. 特异性：将规则按最��定到最一般的顺序排列
2. 性能：将最常见的匹配放在前面以减少处理开销
3. 测试：在部署之前验证正则模式
4. 文档：使用描述性的 `rule_name` 值以便于操作清晰
5. 监控：跟踪规则匹配率以验证预期行为

扩展指标模块

扩展指□模块提供高级遥测和分析能力，用于分析超出□准指□的直径流量模式。

配置

启用模块并配置特定指□类型：

```
module_extended_metrics:
  enabled: true
  attach_attempt_reporting_enabled: true
```

参数

参数	描述
`enabled`	设置为 `true` 以激活扩展指□模块
`attach_attempt_reporting_enabled`	启用跟踪和报告 LTE 附加尝试（S6a AIR/AIA）

可用指标

附加尝试跟踪

通过监控身份验证信息请求 (AIR) 和应答 (AIA) 消息对来跟踪 LTE 订阅者附加尝试：

Parse error on line 36: ... style Metrics fill:#f3e5f5 style E ----------------------^ Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW', 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW', 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

□试

测量：`attach_attempt_count`

字段：

- `imsi` - 订阅者 IMSI（来自用户名 AVP - 参见 AVP 代码）

标签：

- `origin_host` - 发起附加请求的对等
- `result_code` - HSS 响应中的直径结果代码

工作原理：

1. 当接收到 AIR（命令代码 318，S6a 应用 16777251 - 参见 应用 ID）时，模块提取：
   - 用于请求/应答关联的端到端 ID
   - IMSI（用户名 AVP 代码 1）
   - Origin-Host（AVP 代码 264）
2. 请求元数据存储在 ETS 中，带 TTL
3. 当接收到匹配的 AIA 时，模块：
   - 使用端到端 ID 进行关联
   - 提取结果代码（AVP 268 或实验结果代码 AVP 297）
   - 发出带有 IMSI、origin host 和 result code 的指

用例

- 附加成功率分析 - 跟踪成功与失败的附加尝试及其结果代码
- **IMSI** 级故障排除 - 确定经历附加失败的订阅者
- 网络性能监控 - 按来源（MME/SGSN）监控附加尝试模式
- 漫游分析 - 分析入境漫游附加成功率

集成

扩展指通过 InfluxDB 集成导出：

```
DRA.Metrics.InfluxDB.write(%{
  measurement: "attach_attempt_count",
  fields: %{imsi: "505057000000001"},
  tags: %{origin_host: "mme-01.example.com", result_code: 2001}
})
```

结果代码是准直径代码：

- `2001` - 成功（DIAMETER_SUCCESS）
- `5001` - 身份验证失败（DIAMETER_AUTHENTICATION_REJECTED）

- `5004` - 直径 AVP 不支持
- 参见 RFC 6733 以获取完整的结果代码列表

重要说明

- 附加尝试指□仅跟踪 S6a AIR/AIA 对（应用 ID 16777251，命令代码 318）
- 请求元数据根据配置的请求超时 + 5 秒过期
- 指□处理是异步的（生成的进程），以避免阻塞消息流
- 该模块独立于路由和转换模块操作

Prometheus 指标

DRA 暴露全面的 Prometheus 指□，用于监控直径流量、对等健康和模块操作。所有指□都可以在 `/metrics` 端点获取。

核心直径指标

对等状态

指标：`diameter_peer_status` 类型：Gauge 描述：对等是否连接（1）或未连接（0） 标签：

- `origin_host` - 对等的直径身份
- `ip` - 对等的 IP 地址

示例：

```
# 检查特定□等是否连接
diameter_peer_status{origin_host="hss01.example.com"}

# 计算断开连接的□等
count(diameter_peer_status == 0)
```

消息计数

指标：`diameter_peer_message_count_total` 类型：Counter 描述：与对等交换的直径消息总数 标签：

- `origin_host` - 对等的直径身份

- `received_from` - □□□□□□□□
- `application_id` - □□□□ ID□□□ □□ ID □□□
- `cmd_code` - □□□□□□□□□□ □□□□□□□□□
- `application_name` - □□□□□□□□□□□□“3GPP_S6a”□
- `cmd_name` - □□□□□□□□□□□□“AIR”□
- `direction` - "request" □ "response"

□□□

```
# □□□□□ MME □ S6a AIR □□□□
rate(diameter_peer_message_count_total{
  cmd_code="318",
  direction="request",
  origin_host="mme01.example.com"
}[5m])

# □□□□□□□□□□
sum by (application_name)
(rate(diameter_peer_message_count_total[5m]))
```

□□□□□□

□□□`diameter_peer_message_result_code_count_total` □□□Counter □□□□□□□□□□ □□□□□□□□□ □□□

- `origin_host` - □□□□□
- `routed_to` - □□□□□□□
- `application_id` - □□□□ ID
- `cmd_code` - □□□□□□
- `application_name` - □□□□
- `cmd_name` - □□□□
- `result_code` - □□□□□□□□□□□□□□□

□□□

```
# S6a AIR 请求的成功率
rate(diameter_peer_message_result_code_count_total{
  cmd_code="318",
  result_code="2001"
}[5m])

# 按结果代码的错误率
sum by (result_code) (
  rate(diameter_peer_message_result_code_count_total{
    result_code!="2001"
  }[5m])
)
```

常见结果代码:

- `2001` - DIAMETER_SUCCESS
- `3002` - DIAMETER_UNABLE_TO_DELIVER
- `3003` - DIAMETER_REALM_NOT_SERVED
- `3004` - DIAMETER_TOO_BUSY
- `5001` - DIAMETER_AUTHENTICATION_REJECTED
- `5004` - DIAMETER_INVALID_AVP_VALUE
- `5012` - DIAMETER_UNABLE_TO_COMPLY

响应延迟

指标: `diameter_peer_last_response_delay` 类型: Gauge 描述: 最近的响应延迟（毫秒）（DRA → 对等 → DRA）标签:

- `origin_host` - 原始请求者
- `routed_to` - 发送应答的对等
- `application_name` - 应用名称
- `cmd_name` - 命令名称

示例:

```
# HSS 的平均响应时间
avg(diameter_peer_last_response_delay{routed_to="hss01.example.com"})

# S6a 的 P95 响应时间
histogram_quantile(0.95,
  rate(diameter_peer_last_response_delay{application_name="3GPP_S6a"}
[5m])
)
```

未应答请求

指标：`diameter_peer_unanswered_request_count_total` 类型：Counter 描述：发送但在超时期间未应答的请求 标签：

- `origin_host` - 原始请求者
- `routed_to` - 未应答的对等
- `application_id` - 直径应用 ID
- `cmd_code` - 直径命令代码
- `application_name` - 应用名称
- `cmd_name` - 命令名称

示例：

```
# 未应答请求速率
rate(diameter_peer_unanswered_request_count_total[5m])

# 确定有问题的□等
topk(5, sum by (routed_to) (
  rate(diameter_peer_unanswered_request_count_total[5m])
))
```

未授权连接尝试

指标：`diameter_peer_unauthorized_connection_count_total` 类型：Counter 描述：来自未授权对等的连接尝试 标签：

- `origin_host` - 未授权对等声称的身份
- `supported_applications` - 对等声明的应用

- `peer_ip` - 连接尝试的 IP 地址

示例：

```
# 未授权连接尝试
rate(diameter_peer_unauthorized_connection_count_total[5m])

# 未授权访问发出警报
diameter_peer_unauthorized_connection_count_total > 0
```

高级路由模块指标

丢弃的流量

指标：`diameter_advanced_routing_drop_count_total` 类型：Counter 描述：通过高级路由 `:drop` 操作丢弃的请求（未发送响应） 标签：

- `application_id` - 直径应用 ID
- `cmd_code` - 直径命令代码
- `application_name` - 应用名称
- `cmd_name` - 命令名称

示例：

```
# 按命令丢弃速率
sum by (cmd_name) (
  rate(diameter_advanced_routing_drop_count_total[5m])
)

# 总丢弃流量
sum(rate(diameter_advanced_routing_drop_count_total[5m]))
```

何时丢弃流量：

- 高级路由规则匹配 `route: :drop`
- 消息静默丢弃，未转发
- 不向请求对等发送应答消息

- 实现 Erlang Diameter `:discard` 行为
- 参见 高级路由模块 以获取配置

错误响应

指标：`diameter_advanced_routing_error_count_total` 类型：Counter 描述：通过高级路由 `{:error, result_code}` 操作生成的错误响应 标签：

- `result_code` - 发送的直径结果代码
- `application_id` - 直径应用 ID
- `cmd_code` - 直径命令代码
- `application_name` - 应用名称
- `cmd_name` - 命令名称

示例：

```
# 按结果代码的错误响应
sum by (result_code) (
  rate(diameter_advanced_routing_error_count_total[5m])
)

# DIAMETER_TOO_BUSY 响应
rate(diameter_advanced_routing_error_count_total{
  result_code="3004"
}[5m])

# 速率限制或过载检测
diameter_advanced_routing_error_count_total{
  result_code="3004"
} > 100
```

何时生成错误响应：

- 高级路由规则匹配 `route: {:error, result_code}`
- DRA 生成带有指定结果代码的直径应答
- 应答发送回请求对等（不转发）
- 根据 Erlang Diameter：`{:protocol_error, code}` 等同于 `{:answer_message, code}`

- 参见 高级路由模块 以获取配置

常用错误代码：

- `3002` - DIAMETER_UNABLE_TO_DELIVER（路由失败）
- `3003` - DIAMETER_REALM_NOT_SERVED（域不支持）
- `3004` - DIAMETER_TOO_BUSY（过载保护）
- `5012` - DIAMETER_UNABLE_TO_COMPLY（一般错误）

重要说明

- 所有指□都可以在 `/metrics` 端点以 Prometheus 格式获取
- 扩展指标模块 提供额外的 S6a 特定指□（参见 扩展指□模块）
- **BEAM/Erlang** 指标 也被暴露，但未在此文档中记录
- 所有计数器都是累积的，应使用 `rate()` 查询每秒速率
- 高基数□签（如 IMSI）仅在扩展指□模块中使用，以避免指□爆炸

故障排除

规则未匹配

- 验证所有过滤条件是否正确
- 检查 AVP 代码是否与您的直径应用匹配（参见 [AVP 代码