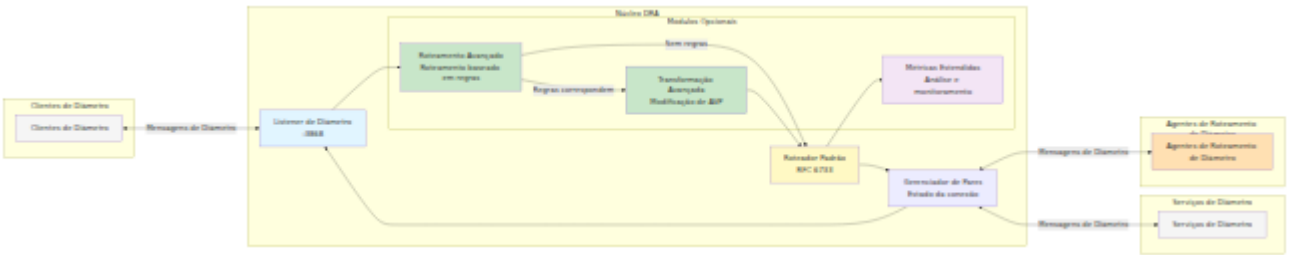
Guia de Operações DRA

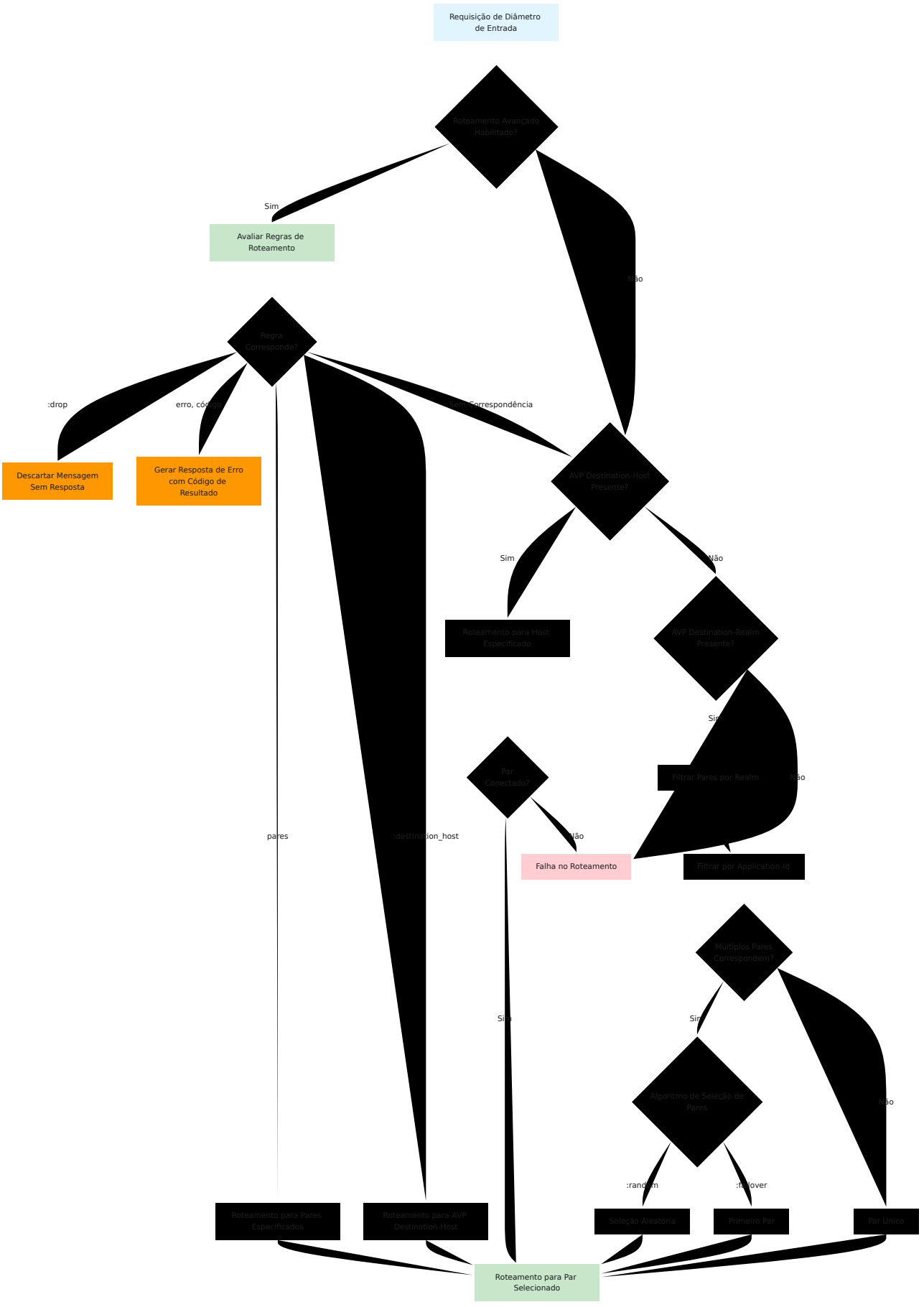
Índice

Visão Geral da Arquitetura DRA



Roteamento de Diâmetro Padrão

Sem os módulos Roteamento Avançado ou Transformação Avançada, o DRA realiza o roteamento padrão de Diâmetro com base no Protocolo Base de Diâmetro (RFC 6733):


Requisição de Diâmetro de Entrada
Roteamento Avançado Habilitado?
Sim
Avaliar Regras de Roteamento
Não
Regra Corresponde?
:drop
erro, código
Descartar Mensagem Sem Resposta
Gerar Resposta de Erro com Código de Resultado
Correspondência
AVP Destination-Host Presente?
Sim
Não
Roteamento para Host Especificado
AVP Destination-Realm Presente?
Sim
Não
Filtrar Pares por Realm
Filtrar por Application-Id
Par Conectado?
Não
Falha no Roteamento
pares
:destination_host
Sim
Múltiplos Pares Correspondem?
Sim
Não
Algoritmo de Seleção de Pares
:random
:failover
Seleção Aleatória
Primeiro Par
Par Único
Roteamento para Pares Especificados
Roteamento para AVP Destination-Host
Roteamento para Par Selecionado

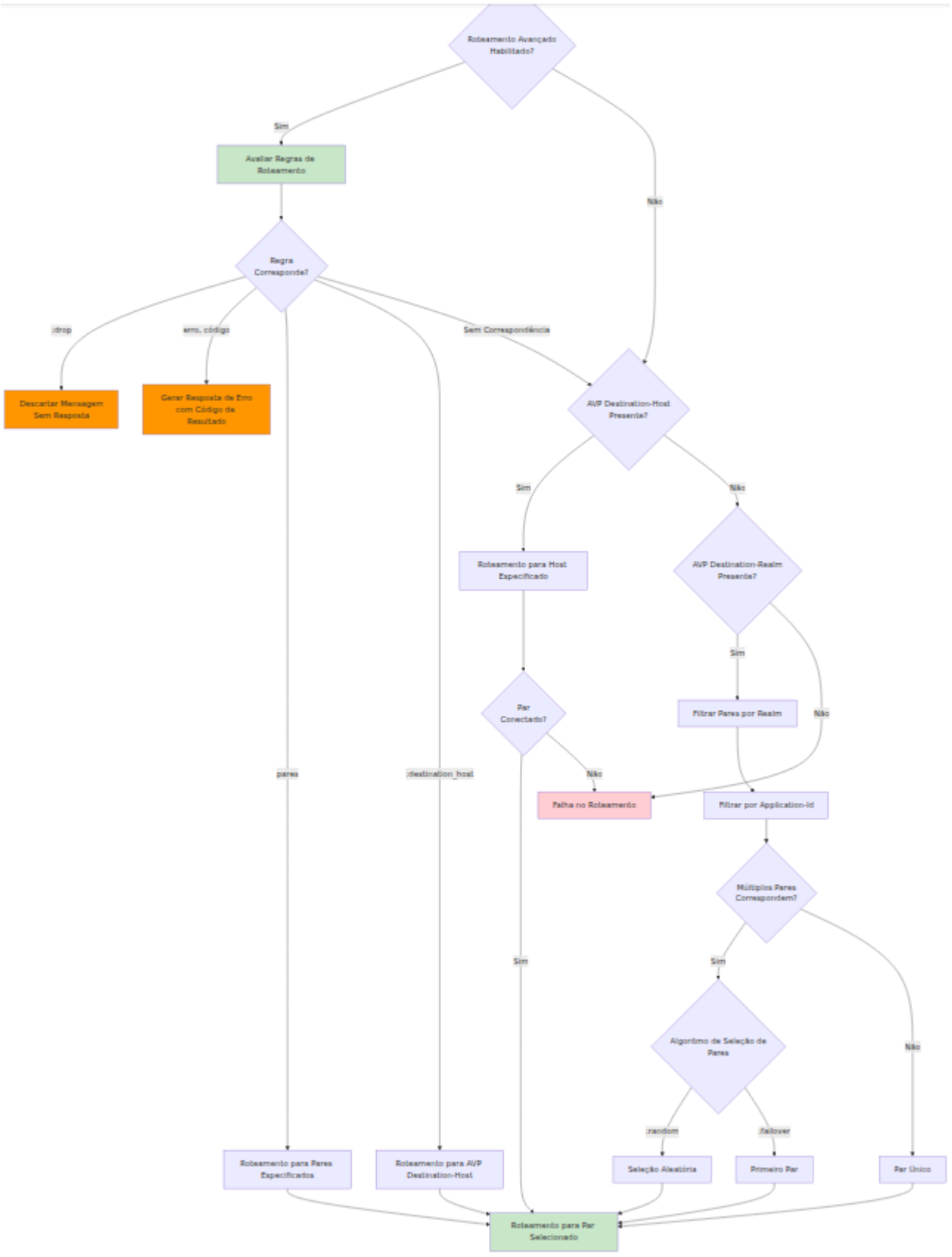
Roteamento de Requisições

O DRA roteia mensagens de requisição usando um mecanismo baseado em prioridade definido na Seção 6.1 da RFC 6733:

1. **AVP Destination-Host (293)** - Se presente, o DRA roteia diretamente para o par especificado
    - Este é o mecanismo de roteamento de mais alta prioridade
    - Se o par não estiver conectado, o roteamento falha
    - Fornece controle de roteamento explícito em nível de host
2. **AVP Destination-Realm (283)** - Se Destination-Host estiver ausente, roteia com base no realm
    - O DRA seleciona um par conectado que anuncia suporte para o realm alvo
    - O balanceamento de carga é aplicado quando múltiplos pares correspondem ao realm
    - O roteamento baseado em realm permite flexibilidade entre múltiplos hosts
3. **Application-Id** - Os pares são filtrados por aplicações de Diâmetro suportadas
    - Apenas pares que anunciam suporte para o Application-Id da mensagem são considerados
    - Baseado na Troca de Capacidades (CER/CEA) durante o estabelecimento da conexão do par
    - Veja IDs de Aplicação 3GPP Comuns para referência

Roteamento de Respostas

Pacotes de resposta usam um mecanismo de roteamento fundamentalmente diferente das requisições:



- **Roteamento baseado em sessão**: Pacotes de resposta sempre seguem o caminho reverso da requisição

- **Preservação do ID de Ponto a Ponto**: O Identificador de Ponto a Ponto permanece inalterado em todos os saltos
- **Roteamento Ponto a Ponto**: O DRA usa o Identificador Ponto a Ponto para manter o estado de roteamento (muda em cada salto)
- **Sem avaliação de regras**: O DRA não avalia regras de roteamento ou conteúdos de AVP para respostas
- **Correlação com estado**: Tabelas de roteamento internas rastreiam qual par enviou cada requisição

Por que as respostas não são roteadas por módulos avançados:

- O roteamento de respostas é determinístico e deve retornar ao par de origem
- O protocolo Diâmetro exige que as respostas sigam o caminho de requisição estabelecido
- As decisões de roteamento para respostas são feitas com base no contexto da requisição original, não no conteúdo da resposta
- Isso garante um gerenciamento de sessão adequado e previne loops de roteamento

Veja Seção 6.2 da RFC 6733 para detalhes sobre o roteamento de mensagens de resposta.

Seleção de Pares

Quando múltiplos pares correspondem aos critérios de roteamento, o `peer_selection_algorithm` configurado determina a seleção:

- `:random` - Seleciona aleatoriamente entre os pares disponíveis (padrão)
- `:failover` - Sempre seleciona o primeiro par da lista (baseado em prioridade)
- Os pares devem estar em **estado conectado** para serem selecionados
- Pares desconectados ou fora do ar são automaticamente excluídos

Limitações do Roteamento Padrão

- Sem regras de roteamento personalizadas baseadas em valores de AVP (por exemplo, padrões IMSI)
- Sem tradução de realm ou modificação de AVP
- Não é possível roteamento baseado no par de origem
- Controle limitado sobre a distribuição de tráfego

Os módulos Roteamento Avançado e Transformação Avançada estendem esse comportamento padrão com capacidades de roteamento baseadas em regras e manipulação de pacotes.

Configuração Base do DRA

O DRA requer uma configuração base que define sua identidade, configurações de rede e conexões de pares. Essa configuração estabelece a base para todas as operações de roteamento.

Estrutura de Configuração

```
%{
  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: [
    # Configurações de pares...
  ]
}
```

Parâmetros de Identidade do DRA

Parâmetro	Tipo	Descrição
`host`	String	A Identidade de Diâmetro do DRA (nome de domínio totalmente qualificado)
`realm`	String	O realm de Diâmetro do DRA
`product_name`	String	Nome do produto anunciado nas mensagens CER/CEA
`vendor_id`	Integer	Vendor-ID conforme definido na Seção 5.3.3 da RFC 6733 (10415 = 3GPP)

Configurações de Rede

Parâmetro	Tipo	Descrição
`listen_ip`	String ou Lista	Endereço(s) IP que o DRA escuta. Para multihoming SCTP, use uma lista de strings IP (veja Multihoming SCTP)
`listen_port`	Integer	Porta TCP/SCTP para conexões de Diâmetro (padrão: 3868)
`service_name`	Atom	Identificador de serviço interno do Erlang
`request_timeout`	Integer	Timeout em milissegundos para pares de requisição/resposta (padrão: 5000)

Configurações de Seleção de Pares

Parâmetro	Tipo	Descrição
`peer_selection_algorithm`	Atom	Algoritmo de balanceamento de carga: `:random` (seleção aleatória) ou `:failover` (prioridade do primeiro par)
`allow_undefined_peers_to_connect`	Boolean	Permitir conexões de pares não configurados (padrão: `false`)
`log_unauthorized_peer_connection_attempts`	Boolean	Registrar tentativas de conexão de pares não autorizados

Configuração de Pares

Cada par na lista `peers` define uma conexão de Diâmetro:

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

Parâmetros de Par

Parâmetro	Tipo	Descrição
`host`	String	A Identidade de Diâmetro do par (FQDN) - deve corresponder exatamente para roteamento
`realm`	String	O realm de Diâmetro do par
`ip`	String	Endereço IP primário do par para conexão (obrigatório)
`ips`	Lista	Lista de endereços IP para multihoming SCTP (opcional, veja Multihoming SCTP)
`port`	Integer	Porta de Diâmetro do par (tipicamente 3868)
`transport`	Atom	Protocolo de transporte: `:diameter_tcp` ou `:diameter_sctp`
`tls`	Boolean	Habilitar criptografia TLS (se `true`, tipicamente use a porta 3869)
`initiate_connection`	Boolean	`true`: DRA conecta ao par, `false`: DRA espera o par conectar

Modos de Conexão

Iniciar Conexão (`initiate_connection: true`)

- O DRA atua como cliente de Diâmetro
- O DRA inicia a conexão TCP/SCTP com o par
- Usado para conectar ao HSS, PCRF ou outros sistemas de backend
- O DRA tentará reconectar se o par estiver inacessível

Aceitar Conexão (`initiate_connection: false`)

- O DRA atua como servidor de Diâmetro
- O DRA espera o par conectar
- Usado para conexões MME, SGSN, P-GW
- O par deve estar na configuração ou `allow_undefined_peers_to_connect: true`

Exemplo de Configuração

```
%{
  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 - espera o MME conectar
    %{
      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 inicia a conexão
    %{
      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 com TLS - DRA inicia conexão segura
    %{
      host: "pcrf01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.22",
      port: 3869,
      transport: :diameter_tcp,
      tls: true,
```

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

Notas Importantes

- **Correspondência de Nome de Host**: Os nomes de host dos pares nas regras de Roteamento Avançado devem corresponder exatamente ao valor `host` configurado aqui (sensível a maiúsculas e minúsculas)
- **Troca de Capacidades**: Na conexão, os pares trocam aplicações suportadas via mensagens CER/CEA
- **Suporte a Aplicações**: O DRA anuncia todas as aplicações 3GPP suportadas (veja IDs de Aplicação 3GPP Comuns)
- **Vendor-ID 10415**: Valor padrão para aplicações 3GPP
- **Timeout de Requisição**: Afeta o TTL das Métricas Estendidas (timeout + 5 segundos)
- **Seleção de Pares**: Quando múltiplos pares correspondem aos critérios de roteamento, o `peer_selection_algorithm` determina qual é escolhido

Considerações de Segurança

- Defina `allow_undefined_peers_to_connect: false` em produção
- Habilite `log_unauthorized_peer_connection_attempts: true` para monitoramento de segurança
- Certifique-se de que as regras de firewall correspondam às configurações de `listen_ip` e `listen_port`
- Valide os certificados dos pares ao usar TLS

Multihoming SCTP

O multihoming SCTP fornece redundância de rede permitindo que os pontos finais se vinculem a múltiplos endereços IP. Se o caminho de rede primário

falhar, o SCTP automaticamente muda para um caminho alternativo sem interromper a sessão de Diâmetro.

Como Funciona

- Os batimentos cardíacos SCTP monitoram todos os caminhos de rede
- A mudança automática ocorre se o caminho primário se tornar inacessível
- Nenhuma interrupção da sessão de Diâmetro durante a troca de caminho
- O kernel lida automaticamente com a seleção de caminho

Configuração

Endereços de Escuta do DRA

Configure múltiplos endereços IP locais para o DRA se vincular:

```
%{
  # IP único (compatível com versões anteriores)
  listen_ip: "192.168.1.10",

  # Múltiplos IPs para multihoming SCTP
  listen_ip: ["192.168.1.10", "10.0.0.10"],

  listen_port: 3868,
  ...
}
```

Notas:

- O transporte TCP usa apenas o primeiro IP na lista
- O transporte SCTP se vincula a todos os IPs especificados
- O formato de string IP único continua totalmente suportado

Configuração de Par

Configure múltiplos endereços IP remotos para conexões de pares:

```
peers: [
  %{
    host: "hss01.example.com",
    realm: "example.com",
    ip: "192.168.1.20",                          # IP Primário
(obrigatório)
    additional_ips: ["192.168.1.20", "10.0.0.20"],        # Todos
os IPs para multihoming
    port: 3868,
    transport: :diameter_sctp,
    tls: false,
    initiate_connection: true
  }
]
```

Notas:

- O campo `ip` é obrigatório para compatibilidade com versões anteriores
- O campo `ips` é opcional; se omitido, apenas `ip` é usado
- Para SCTP, inclua o IP primário na lista `ips`
- Para TCP, apenas `ip` é usado (TCP não suporta multihoming)

Exemplo Completo

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: ["192.168.1.10", "10.0.0.10"],  # DRA multihomed
    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: [
      # Conexão HSS multihomed
      %{
        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
      },
      # MME com IP único (compatível com versões anteriores)
      %{
        host: "mme01.example.com",
        realm: "example.com",
        ip: "192.168.1.30",
        port: 3868,
        transport: :diameter_sctp,
        tls: false,
        initiate_connection: false
      }
    ]
  }
```

Requisitos

- O módulo do kernel SCTP deve estar carregado (pacote `lksctp-tools` no Linux)
- Todos os endereços IP devem ser roteáveis de/para o par
- As regras de firewall devem permitir tráfego SCTP em todos os IPs configurados
- Ambos os pontos finais devem ser configurados para multihoming para total redundância

Limitações

- O transporte TCP não suporta multihoming (usa apenas o IP primário)
- TLS sobre multihoming SCTP pode ter limitações de compatibilidade
- O tempo de falha do caminho depende dos parâmetros SCTP do kernel

---

Tabelas de Referência

IDs de Aplicação 3GPP Comuns

Application-Id	Interface	Descrição
16777251	S6a/S6d	MME/SGSN para HSS autenticação e dados de assinatura
16777252	S13/S13'	MME para verificação de identidade de equipamento EIR
16777238	Gx	PCEF para PCRF controle de política e cobrança
16777267	S9	Política de roaming do PCRF doméstico para o PCRF visitado
16777272	Sy	PCRF para OCS vinculação de sessão
16777216	Cx	I-CSCF/S-CSCF para registro IMS HSS
16777217	Sh	AS para dados de usuário IMS HSS
16777236	SLg	MME/SGSN para serviços de localização GMLC
16777291	SLh	GMLC para informações de assinante de localização HSS
16777302	S6m	MTC-IWF para HSS/HLR para dispositivos M2M
16777308	S6c	SMS-SC/IP-SM-GW para roteamento de SMS HSS

Application-Id	Interface	Descrição
16777343	S6t	SCEF para eventos de monitoramento HSS
16777334	Rx	AF para autorização de mídia PCRF

Códigos AVP Comuns

Código	Nome AVP	Tipo	Uso
1	User-Name	UTF8String	Identificador do assinante (IMSI em 3GPP)
264	Origin-Host	DiameterIdentity	Nome do host do par de origem
268	Result-Code	Unsigned32	Código de resultado padrão
283	Destination-Realm	DiameterIdentity	Realm alvo
293	Destination-Host	DiameterIdentity	Host alvo (opcional)
296	Origin-Realm	DiameterIdentity	Realm de origem
297	Experimental-Result	Grouped	Código de resultado específico do fornecedor

Códigos de Comando Comuns

Os códigos de comando fazem parte do cabeçalho da mensagem de Diâmetro, não dos AVPs:

Código	Nome do Comando	Descrição
257	CER/CEA	Solicitação/Resposta de Troca de Capacidades
258	RAR/RAA	Solicitação/Resposta de Reautenticação
274	ASR/ASA	Solicitação/Resposta de Abortamento de Sessão
275	STR/STA	Solicitação/Resposta de Término de Sessão
280	DWR/DWA	Solicitação/Resposta de Monitoramento de Dispositivo
282	DPR/DPA	Solicitação/Resposta de Desconexão de Par
316	ULR/ULA	Solicitação/Resposta de Atualização de Localização (S6a)
317	CLR/CLA	Solicitação/Resposta de Cancelamento de Localização (S6a)
318	AIR/AIA	Solicitação/Resposta de Informação de Autenticação (S6a)
321	PUR/PUA	Solicitação/Resposta de Purga de UE (S6a)

Módulo de Roteamento Avançado

O módulo de Roteamento Avançado fornece capacidades de roteamento de mensagens flexíveis e baseadas em regras com suporte para condições de correspondência complexas.

Importante: Este módulo avalia **apenas pacotes de requisição de Diâmetro de entrada** (não pacotes de resposta). Pacotes de resposta seguem o roteamento de sessão estabelecido de volta ao par de origem - veja Roteamento de Respostas para detalhes.

Configuração

Habilite o módulo e defina regras de roteamento em sua configuração:

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: <identificador_da_regra>
      match: <escopo_de_correspondência>
      filters: [<lista_de_filtros>]
      route:
        peers: [<lista_de_pares>]
```

Parâmetros

Parâmetro	Descrição
`enabled`	Defina como `True` para ativar o módulo
`rule_name`	Identificador único para a regra de roteamento
`match`	Como os filtros são combinados: `:all` (lógica AND - todos os filtros devem corresponder), `:any` (lógica OR - pelo menos um filtro deve corresponder), `:none` (lógica NOR - nenhum filtro pode corresponder)
`filters`	Lista de condições de filtro (veja Filtros Disponíveis)
`route`	Ação de roteamento (veja Ações de Roteamento abaixo)

Ações de Roteamento

O parâmetro `route` suporta múltiplas ações:

Roteamento para Pares

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

Roteia para nomes de host de pares especificados. Os pares devem ser:

- Definidos na configuração de pares de Diâmetro do DRA
- O nome do host exato conforme configurado (sensível a maiúsculas e minúsculas)
- Atualmente conectados para que o roteamento tenha sucesso (pares desconectados são ignorados)

Roteamento para AVP Destination-Host

```
route: :destination_host
```

Roteia para o par especificado no AVP Destination-Host (293) da mensagem. Se o AVP Destination-Host estiver ausente, o roteamento volta ao comportamento normal.

Descartar Tráfego

```
route: :drop
```

Descarta silenciosamente a mensagem sem enviar qualquer resposta. Use para:

- Filtragem de tráfego e blackholing
- Bloqueio de requisições indesejadas
- Limitação de taxa descartando tráfego excessivo

Comportamento:

- A mensagem é descartada no DRA (não é encaminhada)
- Nenhuma mensagem de resposta é enviada ao par solicitante
- Implementa o comportamento `:discard` do Diâmetro Erlang
- Métrica: `diameter_advanced_routing_drop_count_total` (veja Métricas Prometheus)

Gerar Resposta de Erro

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

Gera uma resposta de erro de Diâmetro com o código de resultado especificado e a envia de volta ao par solicitante. Códigos de resultado comuns:

- `3002` - DIAMETER_UNABLE_TO_DELIVER (roteamento indisponível)
- `3003` - DIAMETER_REALM_NOT_SERVED (realm não suportado)
- `3004` - DIAMETER_TOO_BUSY (proteção contra sobrecarga, limitação de taxa)
- `5012` - DIAMETER_UNABLE_TO_COMPLY (rejeição geral)

Comportamento:

- O DRA gera resposta de erro com o código de resultado especificado
- A resposta inclui Origin-Host, Origin-Realm, Session-Id (auto-preenchido pelo Diâmetro)
- A mensagem NÃO é encaminhada para nenhum par
- Implementa `{:protocol_error, code}` do Diâmetro Erlang (equivalente a `{:answer_message, code}`)
- Métrica: `diameter_advanced_routing_error_count_total` (veja Métricas Prometheus)

Filtros Disponíveis

Filtros Padrão

Disponíveis tanto no Roteamento Avançado quanto na Transformação Avançada:

- `:application_id` - Correspondência do ID de aplicação de Diâmetro (veja referência de ID de Aplicação)
  - Valor único: `{:application_id, 16777251}` (S6a/S6d)
  - Múltiplos valores: `{:application_id, [16777251, 16777252]}` (S6a ou S6b)
- `:command_code` - Correspondência do código de comando de Diâmetro
  - Valor único: `{:command_code, 318}` (requisição AIR)
  - Múltiplos valores: `{:command_code, [317, 318]}` (ULR ou AIR)
- `:avp` - Correspondência do valor de AVP (veja referência de código AVP)
  - Correspondência exata: `{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}`
  - Correspondência regex: `{:avp, {1, ~r"999001.*"}}`
  - Múltiplos padrões: `{:avp, {1, ["505057001313606", ~r"999001.*", ~r"505057.*"]}}`
  - Qualquer valor (verificação de presença): `{:avp, {264, :any}}`

Filtro Específico de Roteamento

Disponível apenas no Roteamento Avançado:

- `:via_peer` - Correspondência do par de onde a requisição foi recebida
  - Par único: `{:via_peer, "omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org"}`
  - Múltiplos pares: `{:via_peer, ["omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org", "omnitouch-lab-dra02.epc.mnc001.mcc001.3gppnetwork.org"]}`
  - Qualquer par: `{:via_peer, :any}`

Filtros Específicos de Transformação

Disponíveis apenas na Transformação Avançada:

- `:to_peer` - Correspondência em um par de destino predeterminado (apenas pacotes de requisição)
  - Par único: `{:to_peer, "dra01.omnitouch.com.au"}`
  - Múltiplos pares: `{:to_peer, ["dra01.omnitouch.com.au", "dra02.omnitouch.com.au"]}`
- `:from_peer` - Correspondência do par que enviou a resposta (apenas pacotes de resposta)
  - Par único: `{:from_peer, "hss-01.example.com"}`
  - Múltiplos pares: `{:from_peer, ["hss-01.example.com", "hss-02.example.com"]}`
- `:packet_type` - Correspondência da direção do pacote
  - Requisição: `{:packet_type, :request}`
  - Resposta: `{:packet_type, :answer}`

Notas Importantes sobre Filtros

- **Filtros AVP**: Recomendados apenas para AVPs simples (User-Name, Origin-Host, Destination-Realm, etc.)
  - AVPs agrupados **não são suportados** e não corresponderão
  - Valores binários complexos **não são suportados**
  - Use o formato: `{:avp, {code, value}}`
- **Operadores de Lista**: Suportados para todos os valores de filtro, exceto `:packet_type`
  - Quando uma lista é usada, aplica-se **lógica OR** dentro da lista
  - Exemplo: `{:command_code, [317, 318]}` corresponde ao código de comando 317 **OU** 318
- **Valores Especiais**:
  - `:any` - Corresponde a qualquer valor (verifica a presença de AVP)
  - Exemplo: `{:avp, {264, :any}}` corresponde se o AVP Origin-Host existir com qualquer valor

Exemplos de Roteamento

Exemplo 1: Roteamento Via Par

Roteie mensagens com base em qual DRA elas chegaram:

```
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]
```

Como funciona: Roteia tráfego S6a que chega via pares DRA específicos para nós HSS locais.

Exemplo 2: Roaming de Entrada com Correspondência de Padrão

Roteie tráfego de roaming com base em padrões IMSI:

```
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]
```

Como funciona: Roteia mensagens S6a de um Realm de Origem específico com padrões IMSI correspondentes para pares DRA designados.

Exemplo 3: Roteamento Dinâmico com :destination_host

Roteie para o valor do AVP Destination-Host na mensagem:

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

Como funciona:

- Quando os filtros correspondem, roteia para o par especificado no AVP Destination-Host (293)
- Se o AVP Destination-Host estiver ausente, a correspondência é considerada uma falha e volta ao roteamento normal
- Útil para honrar o roteamento quando o remetente especifica o destino exato

Exemplo 4: Descartar Tráfego Indesejado

Descarte tráfego de intervalos IMSI específicos:

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

Como funciona:

- Corresponde a mensagens S6a com IMSI começando com 999999
- Silenciosamente descarta a mensagem sem enviar qualquer resposta
- Útil para filtrar tráfego de teste ou bloquear intervalos de assinantes específicos
- Veja Métricas Prometheus para monitorar tráfego descartado

Exemplo 5: Limitação de Taxa com Respostas de Erro

Retorne DIAMETER_TOO_BUSY para padrões de tráfego específicos:

```
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}
```

Como funciona:

- Corresponde ao tráfego S6a de um par sobrecarregado específico
- Retorna a resposta de erro DIAMETER_TOO_BUSY (3004)
- O par solicitante recebe o erro e deve recuar
- Útil para proteção contra sobrecarga e limitação de taxa
- Veja Métricas Prometheus para monitorar respostas de erro

Exemplo 6: Respostas de Erro Condicionais por Comando

Bloqueie tipos de comando específicos com códigos de erro apropriados:

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

Como funciona:

- Corresponde a mensagens de requisição S6a Purge-UE-Request
- Retorna DIAMETER_UNABLE_TO_COMPLY (5012) como erro
- Bloqueia operações específicas sem descartar o tráfego silenciosamente
- Útil para desabilitar seletivamente certos comandos de Diâmetro

---

Módulo de Transformação Avançada

O módulo de Transformação Avançada permite a modificação dinâmica de AVPs de mensagens de Diâmetro com base em critérios de correspondência. Veja Processamento de Regras para detalhes sobre como as regras são avaliadas.

Configuração

Habilite o módulo e defina regras de transformação:

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: <identificador_da_regra>
      match: <escopo_de_correspondência>
      filters: [<lista_de_filtros>]
      transform:
        action: <ação_de_transformação>
        avps: [<modificações_de_avp>]
```

Parâmetros

Parâmetro	Descrição
`enabled`	Defina como `True` para ativar o módulo
`rule_name`	Identificador único para a regra de transformação
`match`	Como os filtros são combinados: `:all` (lógica AND), `:any` (lógica OR), `:none` (lógica NOR) - veja Lógica de Filtro
`filters`	Lista de condições de filtro (veja Filtros Disponíveis)
`transform.action`	Tipo de transformação (`:edit`, `:remove` ou `:overwrite`)
`transform.avps`	Lista de modificações de AVP a serem aplicadas (veja referência de código AVP)

Ações de Transformação

Pacotes de Requisição (Requisições de Diâmetro)

- **`:edit`** - Modificar valores de AVP existentes
  - Apenas modifica AVPs que existem na mensagem

    - Se o AVP não existir, nenhuma alteração é feita
- `:remove` - Remover AVPs da mensagem
- `:overwrite` - Substituir estruturas inteiras de AVP
    - Requer o parâmetro `dictionary` especificando o dicionário de Diâmetro (por exemplo, `:diameter_gen_3gpp_s6a`)

Pacotes de Resposta (Respostas de Diâmetro)

- `:remove` - Remover AVPs da mensagem
- `:overwrite` - Substituir estruturas inteiras de AVP
    - Requer o parâmetro `dictionary`

Importante: Se nenhuma regra corresponder, o pacote é passado através de forma transparente sem quaisquer transformações.

Sintaxe de Modificação de AVP

Modificação padrão:

- `{:avp, {<código>, <novo_valor>}}` - Define AVP para novo valor

Removendo AVPs:

- `{:avp, {<código>, :any}}` - Remove AVP pelo ID (remove independentemente do valor atual)
- Nota: Remover com base no avp_id é suportado; remover com base no conteúdo de AVP não é suportado

Sobrescrever com dicionário:

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

Exemplos de Transformação

Exemplo 1: Reescrita de Realm de Destino Baseada em Par

Reescreva o Destination-Realm com base em onde a mensagem está sendo roteada:

```
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"}}'
```

Como funciona: Quando requisições S6a são roteadas para pares DRA espec��ficos e correspondem ao padrão IMSI, reescreve o Destination-Realm para a rede do Operador X.

Exemplo 2: Roteamento de Múltiplos Operadores com Transformações

```
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"}}'
```

Como funciona: Roteia diferentes intervalos de assinantes IMSI para os realms de rede apropriados com base em padrões IMSI. A primeira regra correspondente vence (veja Ordem de Execução).

Exemplo 3: Reescrita de Realm de MVNO

```
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"]}}'  # Correspondência
exata de IMSI
      transform:
        action: ":edit"
        avps:
          - '{:avp, {283, "epc.mnc001.mcc001.3gppnetwork.org"}}'
```

Como funciona: Transforma o Destination-Realm para um assinante específico de MVNO para sua rede central hospedada.

Exemplo 4: Transformação Apenas de Requisição com Filtro de Tipo de Pacote

Transforme apenas pacotes de requisição (não respostas):

```
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 deve existir com
qualquer valor
      transform:
        action: ":edit"
        avps:
          - '{:avp, {1, "999999000000002"}}'
```

Como funciona:

- Corresponde apenas pacotes de requisição S6a AIR **requisições** (não pacotes de resposta)
- Verifica se o User-Name (AVP 1) é igual a "999999000000001"
- Verifica se o Origin-Host (AVP 264) existe com qualquer valor
- Reescreve o User-Name para "999999000000002"
- Se o AVP não existir, nenhuma alteração é feita

Exemplo 5: Remover AVP

Remova um AVP específico das mensagens:

```
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}}'  # Remove User-Name
independentemente do valor
```

Como funciona: Remove o AVP User-Name (código 1) de todas as mensagens S6a, independentemente de seu valor atual.

Exemplo 6: Sobrescrever AVP Agrupado em Pacotes de Resposta

Modifique AVPs agrupados complexos em pacotes de resposta usando a ação `:overwrite` com suporte a dicionário:

```
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 (Resposta de
Atualização de Localização)
        - '{:packet_type, :answer}'           # Apenas pacotes de
resposta
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'  #
Realm de Origem
      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, [], [],
```

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

Como funciona:

- Corresponde a pacotes S6a de Resposta de Atualização de Localização (ULA) de um Realm de Origem específico
- Usa a ação `:overwrite` para substituir todo o AVP agrupado APN-Configuration-Profile
- **Requer o parâmetro `dictionary`** para codificar corretamente estruturas de AVP agrupadas complexas
- Adiciona três configurações de APN: "internet" (contexto 1), "ims" (contexto 2) e "sos" (contexto 3)
- Cada APN inclui perfis de QoS, limites de largura de banda (AMBR) e configurações de tipo PDN
- A transformação garante que serviços de emergência (SOS) APN sejam provisionados para todos os assinantes desse realm

Quando usar `:overwrite` com dicionário:

- Modificando AVPs agrupados com estruturas aninhadas (como APN-Configuration-Profile)
- Adicionando ou reestruturando dados de assinatura complexos 3GPP
- Quando a ação `:edit` não pode lidar com a complexidade do AVP
- O dicionário deve corresponder à aplicação de Diâmetro (`:diameter_gen_3gpp_s6a` para S6a, etc.)

Notas importantes:

- `:overwrite` substitui todo o AVP, não apenas campos individuais
- A estrutura do AVP deve corresponder exatamente à definição do dicionário
- Estruturas incorretas causarão falhas de codificação e pacotes descartados
- Este é um recurso avançado - valide minuciosamente em ambiente de teste primeiro

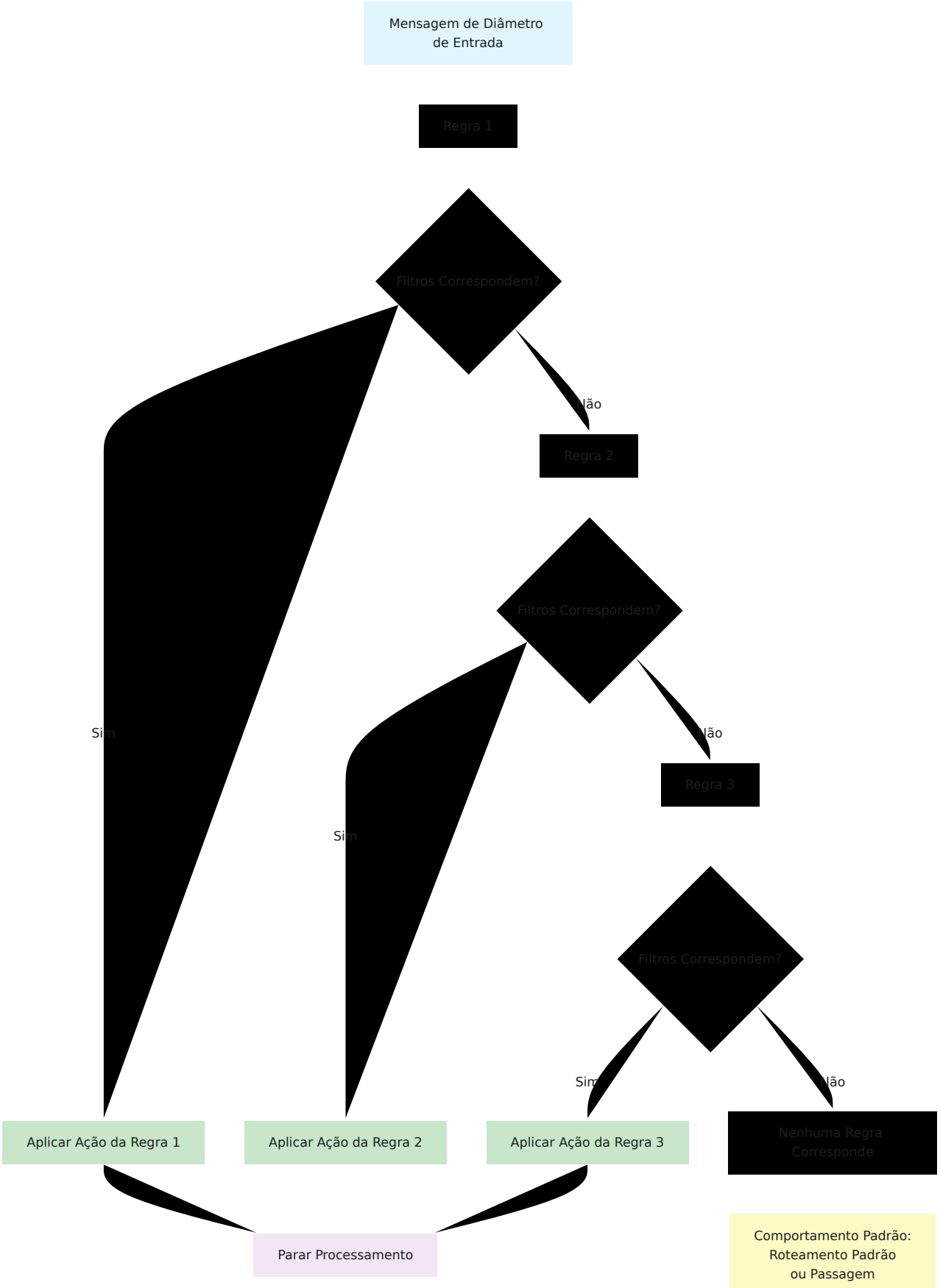
Casos de Uso

- **Suporte a MVNO**: Roteie tráfego de operadores virtuais para redes centrais hospedadas
- **Migração de Rede**: Redirecione gradualmente assinantes para nova infraestrutura
- **Tradução de Realm**: Converta entre diferentes esquemas de nomenclatura para parceiros de roaming
- **Multi-tenancy**: Isolar populações de assinantes por realm
- **Roteamento de Operadores**: Direcione tráfego para redes de operadoras corretas com base em intervalos IMSI

Processamento de Regras

Aplica-se tanto aos módulos Roteamento Avançado quanto à Transformação Avançada.

Ordem de Execução



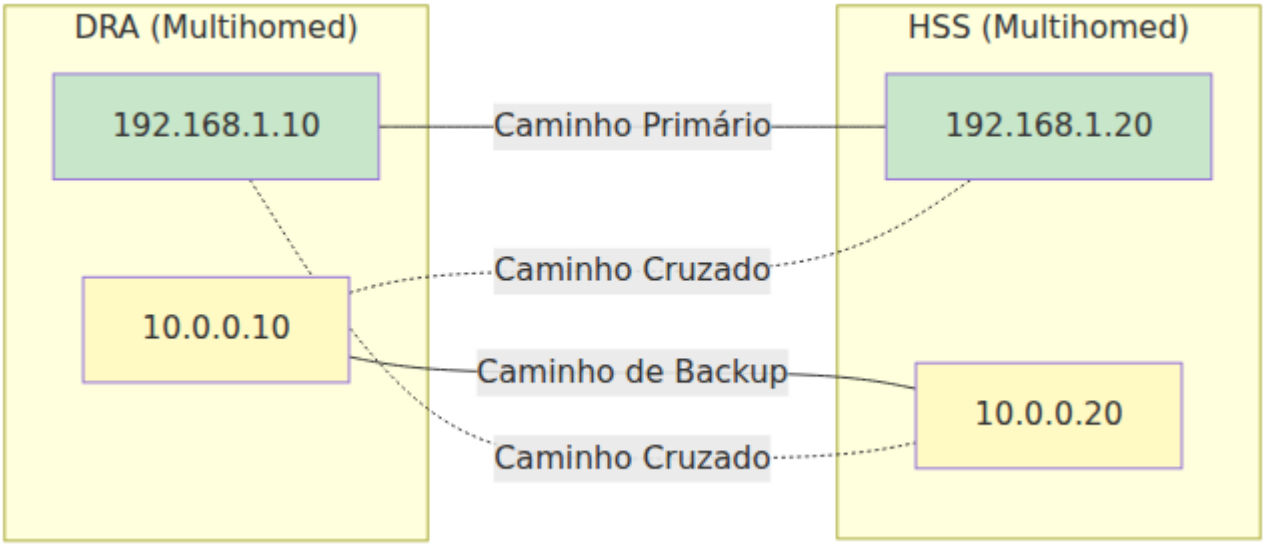
1. As regras são avaliadas **na ordem de cima para baixo** conforme definido na configuração
2. Os filtros dentro de uma regra são avaliados com base no parâmetro `match` (`:all`, `:any` ou `:none`)
3. **A primeira regra correspondente vence** - regras subsequentes não são avaliadas
4. Se nenhuma regra corresponder, o comportamento padrão de roteamento/passthrough é usado

Lógica de Filtro

O parâmetro `match` determina como os filtros são combinados:

match: :all (Lógica AND)

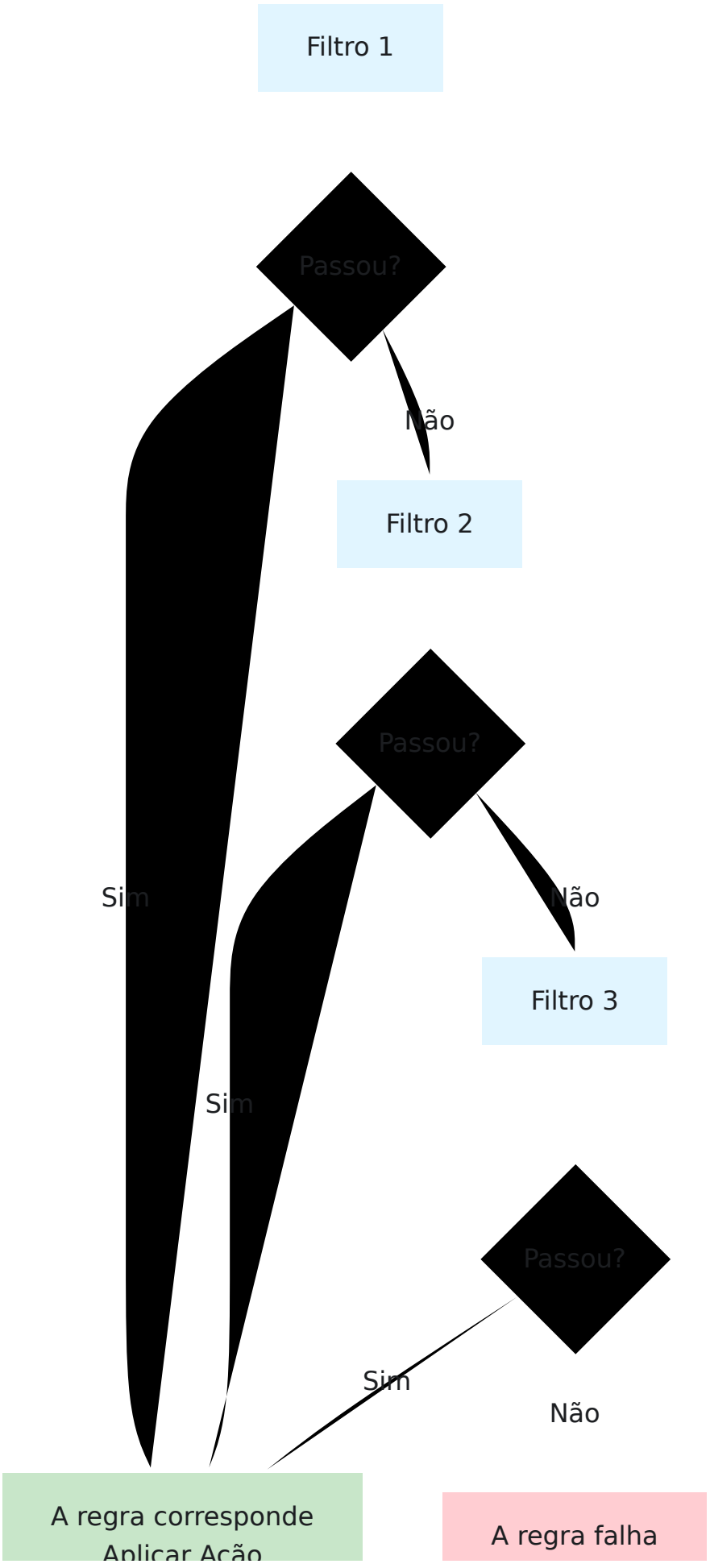
Todos os filtros devem corresponder para que a regra tenha sucesso.



Exemplo: Com 3 filtros, `filtro1 AND filtro2 AND filtro3` devem ser todos verdadeiros.

match: :any (Lógica OR)

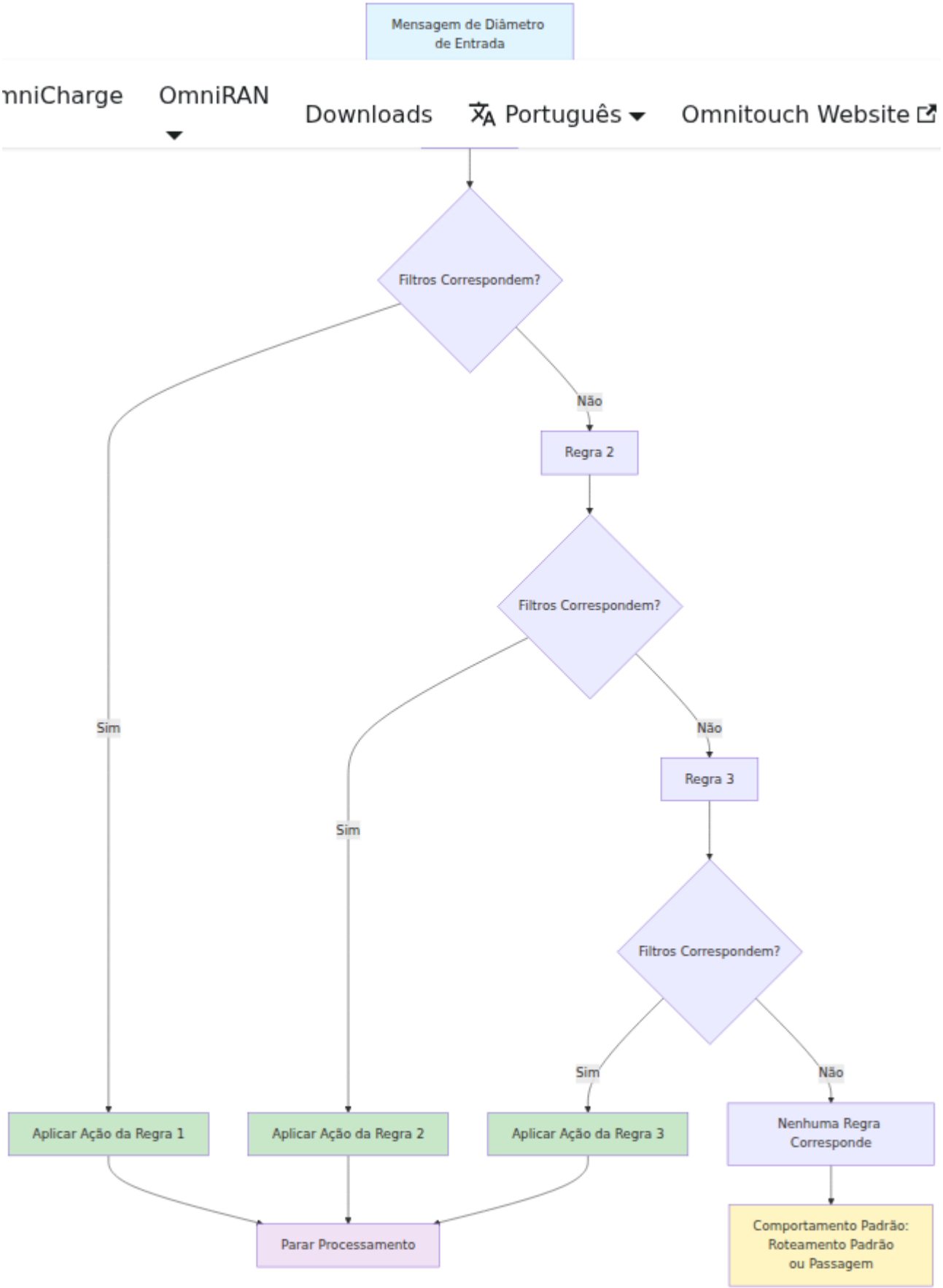
Pelo menos um filtro deve corresponder para que a regra tenha sucesso.


Filtro 1
Passou?
Não
Filtro 2
Passou?
Sim
Não
Filtro 3
Sim
Passou?
Sim
Não
A regra corresponde
Aplicar Ação
A regra falha

Exemplo: Com 3 filtros, `filtro1 OR filtro2 OR filtro3` (qualquer um passa).

match: :none (Lógica NOR)

Nenhum filtro pode corresponder para que a regra tenha sucesso (correspondência inversa).



Exemplo: Com 3 filtros, `NOT filtro1 AND NOT filtro2 AND NOT filtro3` (todos devem falhar).

Notas Adicionais:

Ao usar operadores de lista dentro de um valor de filtro (por exemplo, `{:avp, {1, ["valor1", "valor2"]}}`), os valores usam lógica **OR** (qualquer um pode corresponder).

Padrões de Expressão Regular

Use a sintaxe `~r"padrão"` para correspondência regex:

- `~r"999001.*"` - Corresponde a IMSI começando com 999001
- `~r"^310[0-9]{3}.*"` - Corresponde a IMSI com padrões MNC específicos
- `~r".*teste$"` - Corresponde a valores terminando com "teste"

Melhores Práticas

1. **Especificidade**: Ordene regras da mais específica para a mais geral
2. **Desempenho**: Coloque as correspondências mais comuns primeiro para reduzir a sobrecarga de processamento
3. **Teste**: Valide padrões regex antes da implantação
4. **Documentação**: Use valores descritivos para `rule_name` para clareza operacional
5. **Monitoramento**: Acompanhe taxas de correspondência de regras para verificar o comportamento esperado

---

Módulo de Métricas Estendidas

O módulo de Métricas Estendidas fornece telemetria avançada e capacidades analíticas para analisar padrões de tráfego de Diâmetro além das métricas padrão.

Configuração

Habilite o módulo e configure tipos específicos de métricas:

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

Parâmetros

Parâmetro	Descrição
`enabled`	Defina como `true` para ativar o módulo de métricas estendidas
`attach_attempt_reporting_enabled`	Habilitar rastreamento e relatório de tentativas de anexação LTE (S6a AIR/AIA)

Métricas Disponíveis

Rastreamento de Tentativas de Anexação

Rastreia tentativas de anexação de assinantes LTE monitorando pares de mensagens de Solicitação de Informação de Autenticação (AIR) e Resposta (AIA):

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'

Tentar novamente

Medição: `attach_attempt_count`

Campos:

- `imsi` - O IMSI do assinante (do AVP User-Name - veja códigos AVP)

Tags:

- `origin_host` - O par que originou a solicitação de anexação
- `result_code` - O código de resultado de Diâmetro da resposta HSS

Como funciona:

1. Quando uma AIR (código de comando 318, aplicação S6a 16777251 - veja IDs de Aplicação) é recebida, o módulo extrai:
   - ID de Ponto a Ponto para correlação de requisição/resposta
   - IMSI (AVP User-Name código 1)
   - Origin-Host (AVP código 264)
2. Os metadados da requisição são armazenados no ETS com TTL
3. Quando a AIA correspondente é recebida, o módulo:
   - Correlaciona usando ID de Ponto a Ponto
   - Extrai o código de resultado (AVP 268 ou AVP de resultado experimental AVP 297)
   - Emite a métrica com IMSI, host de origem e código de resultado

Casos de Uso

- **Análise da Taxa de Sucesso de Anexação** - Acompanhe tentativas de anexação bem-sucedidas vs falhadas por código de resultado
- **Solução de Problemas em Nível de IMSI** - Identifique assinantes que estão enfrentando falhas de anexação
- **Monitoramento de Desempenho da Rede** - Monitore padrões de tentativas de anexação por origem (MME/SGSN)
- **Análise de Roaming** - Analise taxas de sucesso de anexação de roaming de entrada

Integração

As métricas estendidas são exportadas via integração com InfluxDB:

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

Os códigos de resultado são códigos padrão de Diâmetro:

- `2001` - Sucesso (DIAMETER_SUCCESS)
- `5001` - Falha de autenticação (DIAMETER_AUTHENTICATION_REJECTED)
- `5004` - AVP de Diâmetro não suportado
- Veja a RFC 6733 para a lista completa de códigos de resultado

Notas Importantes

- As métricas de tentativas de anexação rastreiam apenas pares AIR/AIA de S6a (Application-Id 16777251, Command-Code 318)
- Os metadados da requisição expiram com base no timeout de requisição configurado + 5 segundos
- O processamento de métricas é assíncrono (processo gerado) para evitar bloquear o fluxo de mensagens
- O módulo opera de forma independente dos módulos de roteamento e transformação

---

Métricas Prometheus

O DRA expõe métricas abrangentes do Prometheus para monitorar tráfego de Diâmetro, saúde de pares e operações de módulos. Todas as métricas estão disponíveis no endpoint `/metrics`.

Métricas de Diâmetro do Núcleo

Status do Par

Métrica: `diameter_peer_status` **Tipo**: Gauge **Descrição**: Se o par está conectado (1) ou não (0) **Tags**:

- `origin_host` - Identidade de Diâmetro do par
- `ip` - Endereço IP do par

Exemplo:

```
# Verifique se um par específico está conectado
diameter_peer_status{origin_host="hss01.example.com"}

# Conte pares desconectados
count(diameter_peer_status == 0)
```

Contagem de Mensagens

Métrica: `diameter_peer_message_count_total` **Tipo**: Counter **Descrição**: Total de mensagens de Diâmetro trocadas com pares **Tags**:

- `origin_host` - Identidade de Diâmetro do par
- `received_from` - Par de onde a mensagem foi recebida
- `application_id` - Application-Id de Diâmetro (veja referência de ID de Aplicação)
- `cmd_code` - Código de Comando de Diâmetro (veja Códigos de Comando Comuns)
- `application_name` - Nome da aplicação legível por humanos (por exemplo, "3GPP_S6a")
- `cmd_name` - Nome do comando legível por humanos (por exemplo, "AIR")
- `direction` - "request" ou "response"

Exemplo:

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

# Taxa total de mensagens por aplicação
sum by (application_name)
(rate(diameter_peer_message_count_total[5m]))
```

Códigos de Resultado de Resposta

Métrica: `diameter_peer_message_result_code_count_total` **Tipo**: Counter **Descrição**: Total de respostas de Diâmetro por código de resultado **Tags**:

- `origin_host` - Solicitante original
- `routed_to` - Par que enviou a resposta
- `application_id` - Application-Id de Diâmetro
- `cmd_code` - Código de Comando de Diâmetro
- `application_name` - Nome da aplicação
- `cmd_name` - Nome do comando
- `result_code` - Código de Resultado de Diâmetro ou Código de Resultado Experimental

Exemplo:

```
# Taxa de sucesso para requisições AIR de S6a
rate(diameter_peer_message_result_code_count_total{
  cmd_code="318",
  result_code="2001"
}[5m])

# Taxa de erro por código de resultado
sum by (result_code) (
  rate(diameter_peer_message_result_code_count_total{
    result_code!="2001"
  }[5m])
)
```

Códigos de Resultado Comuns:

- `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

Atraso de Resposta

Métrica: `diameter_peer_last_response_delay` **Tipo**: Gauge **Descrição**: Atraso mais recente de resposta em milissegundos (DRA → Par → DRA) **Tags**:

- `origin_host` - Solicitante original
- `routed_to` - Par que enviou a resposta
- `application_name` - Nome da aplicação
- `cmd_name` - Nome do comando

Exemplo:

```
# Tempo médio de resposta do HSS
avg(diameter_peer_last_response_delay{routed_to="hss01.example.com"})

# P95 do tempo de resposta para S6a
histogram_quantile(0.95,
  rate(diameter_peer_last_response_delay{application_name="3GPP_S6a"}
[5m])
)
```

Requisições Não Respondidas

Métrica: `diameter_peer_unanswered_request_count_total` **Tipo**: Counter **Descrição**: Requisições enviadas mas não respondidas dentro do período de timeout **Tags**:

- `origin_host` - Solicitante original
- `routed_to` - Par que não respondeu
- `application_id` - Application-Id de Diâmetro
- `cmd_code` - Código de Comando de Diâmetro
- `application_name` - Nome da aplicação
- `cmd_name` - Nome do comando

Exemplo:

```
# Taxa de requisições não respondidas
rate(diameter_peer_unanswered_request_count_total[5m])

# Identificar pares problemáticos
topk(5, sum by (routed_to) (
  rate(diameter_peer_unanswered_request_count_total[5m])
))
```

Tentativas de Conexão Não Autorizadas

Métrica: `diameter_peer_una