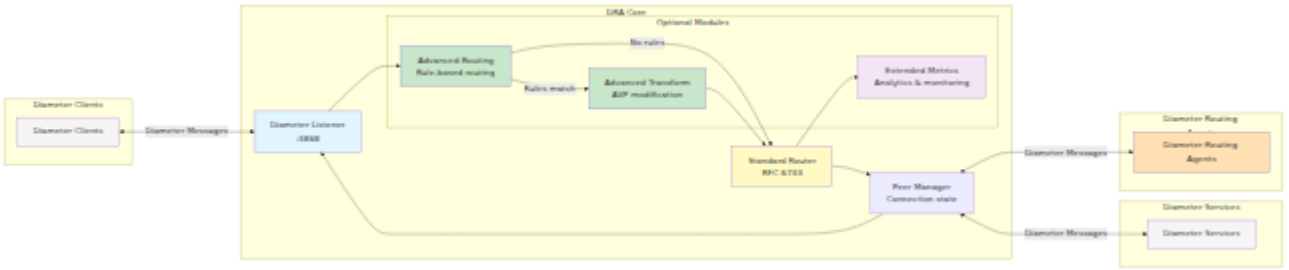
DRA Operations Guide

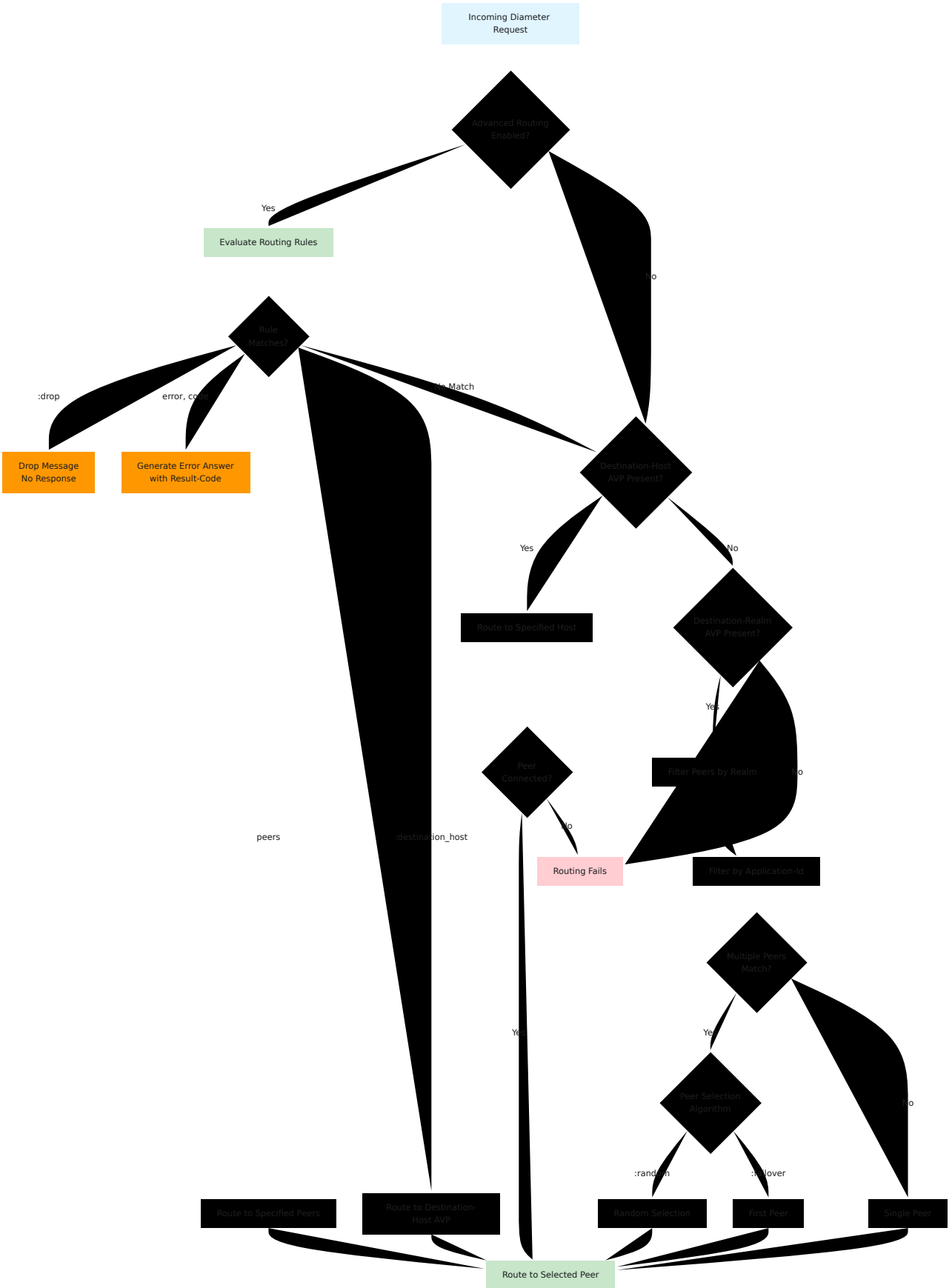
Table of Contents

DRA Architecture Overview



Standard Diameter Routing

Without the Advanced Routing or Advanced Transform modules, the DRA performs standard Diameter routing based on the Diameter Base Protocol (RFC 6733):


Incoming Diameter Request
Advanced Routing Enabled?
Yes
Evaluate Routing Rules
No
Rule Matches?
:drop
error, code
Drop Message No Response
Generate Error Answer with Result-Code
Destination-Host AVP Present?
Yes
No
Route to Specified Host
Destination-Realm AVP Present?
Yes
Filter Peers by Realm
No
Peer Connected?
No
Routing Fails
Filter by Application-Id
peers
:destination_host
Multiple Peers Match?
Yes
Peer Selection Algorithm
:random
:failover
Random Selection
First Peer
Single Peer
Route to Specified Peers
Route to Destination-Host AVP
Route to Selected Peer

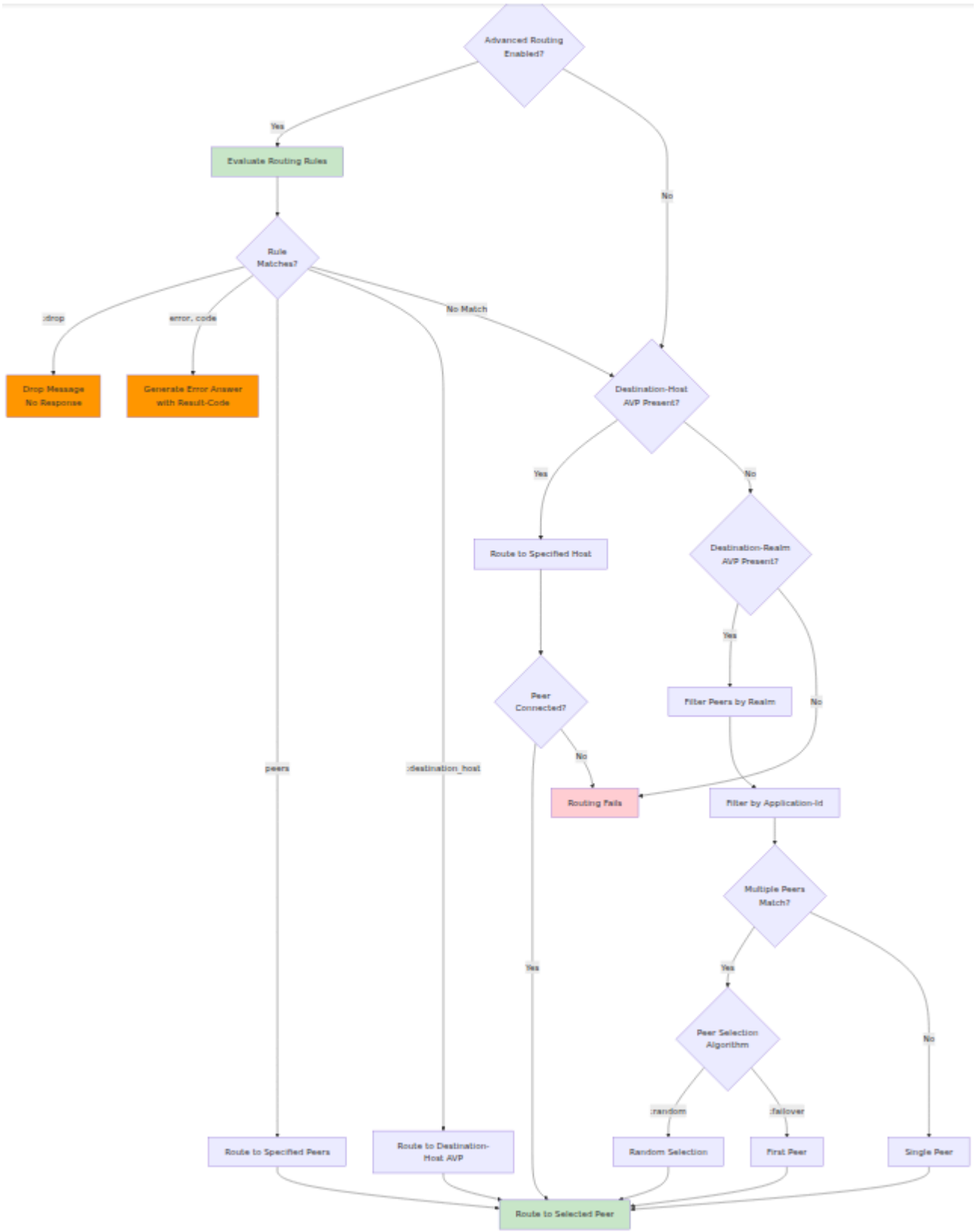
Request Routing

The DRA routes request messages using a priority-based mechanism defined in RFC 6733 Section 6.1:

1. **Destination-Host AVP (293)** - If present, the DRA routes directly to the specified peer
    - This is the highest priority routing mechanism
    - If the peer is not connected, routing fails
    - Provides explicit, host-level routing control
2. **Destination-Realm AVP (283)** - If Destination-Host is absent, routes based on realm
    - The DRA selects a connected peer that advertises support for the target realm
    - Load balancing is applied when multiple peers match the realm
    - Realm-based routing allows flexibility across multiple hosts
3. **Application-Id** - Peers are filtered by supported Diameter applications
    - Only peers advertising support for the message's Application-Id are considered
    - Based on Capabilities Exchange (CER/CEA) during peer connection establishment
    - See Common 3GPP Application IDs for reference

Answer Routing

Answer packets use a fundamentally different routing mechanism than requests:



- **Session-based routing**: Answer packets always follow the reverse path of the request

- **End-to-End ID preservation**: The End-to-End Identifier remains unchanged across all hops
- **Hop-by-Hop routing**: The DRA uses the Hop-by-Hop Identifier to maintain routing state (changes at each hop)
- **No rule evaluation**: The DRA does not evaluate routing rules or AVP contents for answers
- **Stateful correlation**: Internal routing tables track which peer sent each request

Why answers are not routed by advanced modules:

- Answer routing is deterministic and must return to the originating peer
- The Diameter protocol requires answers to follow the established request path
- Routing decisions for answers are made based on the original request context, not answer content
- This ensures proper session management and prevents routing loops

See RFC 6733 Section 6.2 for answer message routing details.

Peer Selection

When multiple peers match the routing criteria, the configured `peer_selection_algorithm` determines selection:

- `:random` - Randomly selects from available peers (default)
- `:failover` - Always selects the first peer in the list (priority-based)
- Peers must be in **connected state** to be selected
- Disconnected or down peers are automatically excluded

Limitations of Standard Routing

- No custom routing rules based on AVP values (e.g., IMSI patterns)
- No realm translation or AVP modification
- Cannot route based on originating peer
- Limited control over traffic distribution

The Advanced Routing and Advanced Transform modules extend this standard behavior with rule-based routing and packet manipulation capabilities.

Base DRA Configuration

The DRA requires base configuration defining its identity, network settings, and peer connections. This configuration establishes the foundation for all routing operations.

Configuration Structure

```
%{
  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: [
    # Peer configurations...
  ]
}
```

DRA Identity Parameters

Parameter	Type	Description
`host`	String	The DRA's Diameter Identity (fully qualified domain name)
`realm`	String	The DRA's Diameter realm
`product_name`	String	Product name advertised in CER/CEA messages
`vendor_id`	Integer	Vendor-ID as defined in RFC 6733 Section 5.3.3 (10415 = 3GPP)

Network Settings

Parameter	Type	Description
`listen_ip`	String or List	IP address(es) the DRA listens on. For SCTP multihoming, use a list of IP strings (see SCTP Multihoming)
`listen_port`	Integer	TCP/SCTP port for Diameter connections (standard: 3868)
`service_name`	Atom	Internal Erlang service identifier
`request_timeout`	Integer	Timeout in milliseconds for request/answer pairs (default: 5000)

Peer Selection Settings

Parameter	Type	Description
`peer_selection_algorithm`	Atom	Load balancing algorithm: `:random` (random selection) or `:failover` (first peer priority)
`allow_undefined_peers_to_connect`	Boolean	Allow connections from peers not in configuration (default: `false`)
`log_unauthorized_peer_connection_attempts`	Boolean	Log connection attempts from unauthorized peers

Peer Configuration

Each peer in the `peers` list defines a Diameter connection:

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

Peer Parameters

Parameter	Type	Description
`host`	String	Peer's Diameter Identity (FQDN) - must match exactly for routing
`realm`	String	Peer's Diameter realm
`ip`	String	Peer's primary IP address for connection (required)
`ips`	List	List of IP addresses for SCTP multihoming (optional, see SCTP Multihoming)
`port`	Integer	Peer's Diameter port (typically 3868)
`source_ip`	String	Local IP address to bind when connecting to this peer (optional, defaults to the service `listen_ip`)
`source_port`	Integer	Local port to bind when connecting to this peer (optional, OS assigns an ephemeral port if not set)
`transport`	Atom	Transport protocol: `:diameter_tcp` or `:diameter_sctp`
`tls`	Boolean	Enable TLS encryption (if `true`, typically use port 3869)
`initiate_connection`	Boolean	`true`: DRA connects to peer, `false`: DRA waits for peer to connect

Connection Modes

Initiate Connection (`initiate_connection: true`)

- DRA acts as Diameter client
- DRA initiates TCP/SCTP connection to peer
- Used for connecting to HSS, PCRF, or other backend systems
- DRA will retry connections if peer is unreachable

Accept Connection (`initiate_connection: false`)

- DRA acts as Diameter server
- DRA waits for peer to connect
- Used for MME, SGSN, P-GW connections
- Peer must be in configuration or `allow_undefined_peers_to_connect: true`
- Inbound peers are differentiated by their advertised Origin-Host (from the CER), so two Diameter nodes co-located on one host (same source IP) are matched independently as long as each presents a distinct `host`

Two Peers on One Host (Same IP)

When two Diameter nodes share a single host/IP — for example an XCAP server and a P-CSCF on the same VM — they are distinguished as follows:

- **Inbound (`initiate_connection: false`)** — each node sends its own Origin-Host in the CER. Configure one `peers` entry per node, each with the same `ip` but a distinct `host`/`realm`. The DRA authorises and registers them independently; no extra configuration is required.
- **Outbound (`initiate_connection: true`)** — when the DRA dials two nodes that share the same destination IP *and* port, set a distinct `source_port` (and optionally `source_ip`) on each peer so the two connections occupy separate local sockets and can be tracked independently.

Configuration Example

```
%{
  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 - waits for MME to connect
    %{
      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 initiates connection with an explicit local source
port
    %{
      host: "hss01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.141",
      port: 3868,
      source_port: 13868,
      transport: :diameter_tcp,
      tls: false,
      initiate_connection: true
    },
    # PCRF with TLS - DRA initiates secure connection
    %{
      host: "pcrf01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.22",
      port: 3869,
```

```
      transport: :diameter_tcp,
      tls: true,
      initiate_connection: true
    }
  ]
}
```

Important Notes

- **Hostname Matching**: Peer hostnames in Advanced Routing rules must exactly match the `host` value configured here (case-sensitive)
- **Capabilities Exchange**: On connection, peers exchange supported applications via CER/CEA messages
- **Application Support**: The DRA advertises all supported 3GPP applications (see Common 3GPP Application IDs)
- **Vendor-ID 10415**: Standard value for 3GPP applications
- **Request Timeout**: Affects Extended Metrics TTL (timeout + 5 seconds)
- **Peer Selection**: When multiple peers match routing criteria, `peer_selection_algorithm` determines which is chosen

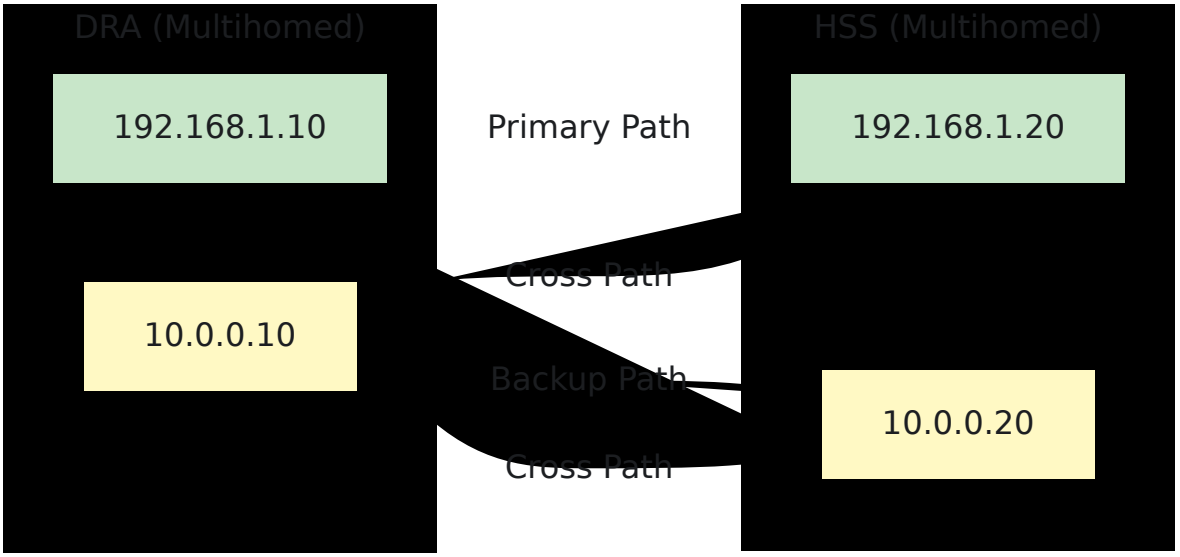
Security Considerations

- Set `allow_undefined_peers_to_connect: false` in production
- Enable `log_unauthorized_peer_connection_attempts: true` for security monitoring
- Ensure firewall rules match `listen_ip` and `listen_port` settings
- Validate peer certificates when using TLS

SCTP Multihoming

SCTP multihoming provides network redundancy by allowing endpoints to bind to multiple IP addresses. If the primary network path fails, SCTP automatically fails over to an alternative path without disrupting the Diameter session.

How It Works



- SCTP heartbeats monitor all network paths
- Automatic failover occurs if the primary path becomes unreachable
- No Diameter session disruption during path switchover
- The kernel handles path selection automatically

Configuration

DRA Listen Addresses

Configure multiple local IP addresses for the DRA to bind to:

```
%{
  # Single IP (backward compatible)
  listen_ip: "192.168.1.10",

  # Multiple IPs for SCTP multihoming
  listen_ip: ["192.168.1.10", "10.0.0.10"],

  listen_port: 3868,
  ...
}
```

Notes:

- TCP transport uses only the first IP in the list
- SCTP transport binds to all specified IPs
- Single IP string format remains fully supported

Peer Configuration

Configure multiple remote IP addresses for peer connections:

```
peers: [
  %{
    host: "hss01.example.com",
    realm: "example.com",
    ip: "192.168.1.20",                          # Primary IP
(required)
    additional_ips: ["192.168.1.20", "10.0.0.20"],        # All
IPs for multihoming
    port: 3868,
    transport: :diameter_sctp,
    tls: false,
    initiate_connection: true
  }
]
```

Notes:

- `ip` field is required for backward compatibility
- `ips` field is optional; if omitted, only `ip` is used
- For SCTP, include the primary IP in the `ips` list
- For TCP, only `ip` is used (TCP does not support multihoming)

Complete Example

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: ["192.168.1.10", "10.0.0.10"],  # Multihomed 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: [
      # Multihomed HSS connection
      %{
        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
      },
      # Single-homed MME (backward compatible)
      %{
        host: "mme01.example.com",
        realm: "example.com",
        ip: "192.168.1.30",
        port: 3868,
        transport: :diameter_sctp,
        tls: false,
        initiate_connection: false
      }
    ]
  }
```

Requirements

- SCTP kernel module must be loaded (`lksctp-tools` package on Linux)
- All IP addresses must be routable from/to the peer
- Firewall rules must allow SCTP traffic on all configured IPs
- Both endpoints should be configured for multihoming for full redundancy

Limitations

- TCP transport does not support multihoming (only uses primary IP)
- TLS over SCTP multihoming may have compatibility limitations
- Path failover timing depends on kernel SCTP parameters

Reference Tables

Common 3GPP Application IDs

Application-Id	Interface	Description
16777251	S6a/S6d	MME/SGSN to HSS authentication and subscription data
16777252	S13/S13'	MME to EIR equipment identity check
16777238	Gx	PCEF to PCRF policy and charging control
16777267	S9	Home PCRF to Visited PCRF roaming policy
16777272	Sy	PCRF to OCS session binding
16777216	Cx	I-CSCF/S-CSCF to HSS IMS registration
16777217	Sh	AS to HSS IMS user data
16777236	SLg	MME/SGSN to GMLC location services
16777291	SLh	GMLC to HSS location subscriber info
16777302	S6m	MTC-IWF to HSS/HLR for M2M devices
16777308	S6c	SMS-SC/IP-SM-GW to HSS SMS routing
16777343	S6t	SCEF to HSS monitoring events
16777334	Rx	AF to PCRF media authorization

Common AVP Codes

Code	AVP Name	Type	Usage
1	User-Name	UTF8String	Subscriber identifier (IMSI in 3GPP)
264	Origin-Host	DiameterIdentity	Originating peer hostname
268	Result-Code	Unsigned32	Standard result code
283	Destination-Realm	DiameterIdentity	Target realm
293	Destination-Host	DiameterIdentity	Target host (optional)
296	Origin-Realm	DiameterIdentity	Source realm
297	Experimental-Result	Grouped	Vendor-specific result code

Common Command Codes

Command codes are part of the Diameter message header, not AVPs:

Code	Command Name	Description
257	CER/CEA	Capabilities-Exchange-Request/Answer
258	RAR/RAA	Re-Auth-Request/Answer
274	ASR/ASA	Abort-Session-Request/Answer
275	STR/STA	Session-Termination-Request/Answer
280	DWR/DWA	Device-Watchdog-Request/Answer
282	DPR/DPA	Disconnect-Peer-Request/Answer
316	ULR/ULA	Update-Location-Request/Answer (S6a)
317	CLR/CLA	Cancel-Location-Request/Answer (S6a)
318	AIR/AIA	Authentication-Information-Request/Answer (S6a)
321	PUR/PUA	Purge-UE-Request/Answer (S6a)

Advanced Routing Module

The Advanced Routing module provides flexible, rule-based message routing capabilities with support for complex matching conditions.

Important: This module evaluates **inbound Diameter request packets only** (not answer packets). Answer packets follow the established session routing back to the originating peer - see Answer Routing for details.

Configuration

Enable the module and define routing rules in your configuration:

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: <rule_identifier>
      match: <match_scope>
      filters: [<filter_list>]
      route:
        peers: [<peer_list>]
```

Parameters

Parameter	Description
`enabled`	Set to `True` to activate the module
`rule_name`	Unique identifier for the routing rule
`match`	How filters are combined: `:all` (AND logic - all filters must match), `:any` (OR logic - at least one filter must match), `:none` (NOR logic - no filters can match)
`filters`	List of filter conditions (see Available Filters)
`route`	Routing action (see Route Actions below)

Route Actions

The `route` parameter supports multiple actions:

Route to Peers

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

Routes to specified peer hostnames. Peers must be:

- Defined in the DRA's Diameter peer configuration
- The exact hostname as configured (case-sensitive)
- Currently connected for routing to succeed (disconnected peers are skipped)

Route to Destination-Host AVP

```
route: :destination_host
```

Routes to the peer specified in the message's Destination-Host AVP (293). If Destination-Host AVP is missing, routing falls back to normal behavior.

Drop Traffic

```
route: :drop
```

Silently discards the message without sending any response. Use for:

- Traffic filtering and blackholing
- Blocking unwanted requests
- Rate limiting by dropping excess traffic

Behavior:

- Message is dropped at DRA (not forwarded)
- No answer message is sent to requesting peer
- Implements Erlang Diameter `:discard` behavior
- Metric: `diameter_advanced_routing_drop_count_total` (see Prometheus Metrics)

Generate Error Response

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

Generates a Diameter error answer with the specified Result-Code and sends it back to the requesting peer. Common result codes:

- `3002` - DIAMETER_UNABLE_TO_DELIVER (routing unavailable)
- `3003` - DIAMETER_REALM_NOT_SERVED (realm not supported)
- `3004` - DIAMETER_TOO_BUSY (overload protection, rate limiting)
- `5012` - DIAMETER_UNABLE_TO_COMPLY (general rejection)

Behavior:

- DRA generates error answer with specified Result-Code
- Answer includes Origin-Host, Origin-Realm, Session-Id (auto-populated by Diameter)
- Message is NOT forwarded to any peer
- Implements Erlang Diameter `{:protocol_error, code}` (equivalent to `{:answer_message, code}`)
- Metric: `diameter_advanced_routing_error_count_total` (see Prometheus Metrics)

Available Filters

Standard Filters

Available in both Advanced Routing and Advanced Transform:

- **`:application_id`** - Match Diameter application ID (see Application ID reference)
  - Single value: `{:application_id, 16777251}` (S6a/S6d)
  - Multiple values: `{:application_id, [16777251, 16777252]}` (S6a or S6b)
- **`:command_code`** - Match Diameter command code
  - Single value: `{:command_code, 318}` (AIR request)

    - Multiple values: `{:command_code, [317, 318]}` (ULR or AIR)
- **`:avp`** - Match AVP value (see AVP code reference)
    - Exact match: `{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}`
    - Regex match: `{:avp, {1, ~r"999001.*"}}`
    - Multiple patterns: `{:avp, {1, ["505057001313606", ~r"999001.*", ~r"505057.*"]}}`
    - Any value (presence check): `{:avp, {264, :any}}`

Routing-Specific Filter

Only available in Advanced Routing:

- **`:via_peer`** - Match the peer where the request was received from
    - Single peer: `{:via_peer, "omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org"}`
    - Multiple peers: `{:via_peer, ["omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org", "omnitouch-lab-dra02.epc.mnc001.mcc001.3gppnetwork.org"]}`
    - Any peer: `{:via_peer, :any}`

Transform-Specific Filters

Only available in Advanced Transform:

- **`:to_peer`** - Match on predetermined destination peer (request packets only)
    - Single peer: `{:to_peer, "dra01.omnitouch.com.au"}`
    - Multiple peers: `{:to_peer, ["dra01.omnitouch.com.au", "dra02.omnitouch.com.au"]}`
- **`:from_peer`** - Match peer who sent the answer (answer packets only)
    - Single peer: `{:from_peer, "hss-01.example.com"}`
    - Multiple peers: `{:from_peer, ["hss-01.example.com", "hss-02.example.com"]}`
- **`:packet_type`** - Match packet direction

- Request: `{:packet_type, :request}`
- Answer: `{:packet_type, :answer}`

Important Filter Notes

- **AVP Filters**: Recommended for simple AVPs only (User-Name, Origin-Host, Destination-Realm, etc.)
  - Grouped AVPs are **not supported** and will not match
  - Complex binary values are **not supported**
  - Use format: `{:avp, {code, value}}`
- **List Operators**: Supported for all filter values except `:packet_type`
  - When a list is used, it applies **OR logic** within the list
  - Example: `{:command_code, [317, 318]}` matches command code 317 **OR** 318
- **Special Values**:
  - `:any` - Matches any value (checks for AVP presence)
  - Example: `{:avp, {264, :any}}` matches if Origin-Host AVP exists with any value

Routing Examples

Example 1: Via Peer Routing

Route messages based on which DRA they arrived from:

```
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]
```

How it works: Routes S6a traffic that arrives via specific DRA peers to local HSS nodes.

Example 2: Inbound Roaming with Pattern Matching

Route roaming traffic based on IMSI patterns:

```
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]
```

How it works: Routes S6a messages from specific Origin-Realm with matching IMSI patterns to designated DRA peers.

Example 3: Dynamic Routing with :destination_host

Route to the Destination-Host AVP value in the message:

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

How it works:

- When filters match, routes to the peer specified in the Destination-Host AVP (293)
- If Destination-Host AVP is missing, the match is considered a failure and falls back to normal routing
- Useful for honor routing when the sender specifies the exact destination

Example 4: Drop Unwanted Traffic

Drop traffic from specific IMSI ranges:

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

How it works:

- Matches S6a messages with IMSI starting with 999999
- Silently drops the message without sending any response
- Useful for filtering test traffic or blocking specific subscriber ranges

- See Prometheus Metrics for monitoring dropped traffic

Example 5: Rate Limiting with Error Responses

Return DIAMETER_TOO_BUSY for specific traffic patterns:

```
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}
```

How it works:

- Matches S6a traffic from specific overloaded peer
- Returns DIAMETER_TOO_BUSY (3004) error response
- Requesting peer receives error and should back off
- Useful for overload protection and rate limiting
- See Prometheus Metrics for monitoring error responses

Example 6: Conditional Error Responses by Command

Block specific command types with appropriate error codes:

```
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}
```

How it works:

- Matches S6a Purge-UE-Request messages
- Returns DIAMETER_UNABLE_TO_COMPLY (5012) error
- Blocks specific operations without dropping traffic silently
- Useful for selectively disabling certain Diameter commands

Advanced Transform Module

The Advanced Transform module enables dynamic modification of Diameter message AVPs based on matching criteria. See Rule Processing for details on how rules are evaluated.

Configuration

Enable the module and define transformation rules:

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: <rule_identifier>
      match: <match_scope>
      filters: [<filter_list>]
      transform:
        action: <transform_action>
        avps: [<avp_modifications>]
```

Parameters

Parameter	Description
`enabled`	Set to `True` to activate the module
`rule_name`	Unique identifier for the transform rule
`match`	How filters are combined: `:all` (AND logic), `:any` (OR logic), `:none` (NOR logic) - see Filter Logic
`filters`	List of filter conditions (see Available Filters)
`transform.action`	Type of transformation (`:edit`, `:remove`, or `:overwrite`)
`transform.avps`	List of AVP modifications to apply (see AVP code reference)

Transform Actions

Request Packets (Diameter Requests)

- **`:edit`** - Modify existing AVP values
  - Only modifies AVPs that exist in the message
  - If the AVP doesn't exist, no change is made
- **`:remove`** - Remove AVPs from the message
- **`:overwrite`** - Replace entire AVP structures
  - Requires `dictionary` parameter specifying the Diameter dictionary (e.g., `:diameter_gen_3gpp_s6a`)

Answer Packets (Diameter Answers)

- **`:remove`** - Remove AVPs from the message
- **`:overwrite`** - Replace entire AVP structures
  - Requires `dictionary` parameter

Important: If no rules match, the packet is passed through transparently without any transformations.

AVP Modification Syntax

Standard modification:

- `{:avp, {<code>, <new_value>}}` - Set AVP to new value

Removing AVPs:

- `{:avp, {<code>, :any}}` - Remove AVP by ID (removes regardless of current value)
- Note: Removing based on avp_id is supported; removing based on AVP contents is not supported

Overwrite with dictionary:

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

Transform Examples

Example 1: To-Peer Based Realm Rewriting

Rewrite Destination-Realm based on where the message is being routed:

```
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"}}'
```

How it works: When S6a requests are routed to specific DRA peers and match the IMSI pattern, rewrites the Destination-Realm for Operator X network.

Example 2: Multiple Carrier Routing with Transforms

```
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"}}'
```

How it works: Routes different IMSI subscriber ranges to appropriate network realms based on IMSI patterns. First matching rule wins (see Execution Order).

Example 3: MVNO Realm Rewriting

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

How it works: Transforms Destination-Realm for specific MVNO subscriber to their hosted core network.

Example 4: Request-Only Transform with Packet Type Filter

Transform only request packets (not answers):

```
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 must exist with any
value
      transform:
        action: ":edit"
        avps:
          - '{:avp, {1, "999999000000002"}}'
```

How it works:

- Matches only S6a AIR **request** packets (not answer packets)
- Checks User-Name (AVP 1) equals "999999000000001"
- Verifies Origin-Host (AVP 264) exists with any value
- Rewrites User-Name to "999999000000002"
- If AVP doesn't exist, no change is made

Example 5: Remove AVP

Remove specific AVP from messages:

```
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 regardless of
value
```

How it works: Removes User-Name AVP (code 1) from all S6a messages, regardless of its current value.

Example 6: Overwrite Grouped AVP on Answer Packets

Modify complex grouped AVPs in answer packets using the `:overwrite` action with dictionary support:

```
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 (Update
Location Answer)
        - '{:packet_type, :answer}'           # Answer packets
only
        - '{: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, [], [],
```

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

How it works:

- Matches S6a Update Location Answer (ULA) packets from a specific Origin-Realm
- Uses `:overwrite` action to replace the entire APN-Configuration-Profile grouped AVP
- **Requires `dictionary` parameter** to properly encode complex grouped AVP structures
- Adds three APN configurations: "internet" (context 1), "ims" (context 2), and "sos" (context 3)
- Each APN includes QoS profiles, bandwidth limits (AMBR), and PDN type settings
- The transformation ensures emergency services (SOS) APN is provisioned for all subscribers from this realm

When to use `:overwrite` with dictionary:

- Modifying grouped AVPs with nested structures (like APN-Configuration-Profile)
- Adding or restructuring complex 3GPP subscription data
- When `:edit` action cannot handle the AVP complexity
- Dictionary must match the Diameter application (`:diameter_gen_3gpp_s6a` for S6a, etc.)

Important notes:

- `:overwrite` replaces the entire AVP, not just individual fields
- The AVP structure must match the dictionary definition exactly
- Incorrect structure will cause encoding failures and dropped packets
- This is an advanced feature - validate thoroughly in test environment first

Use Cases

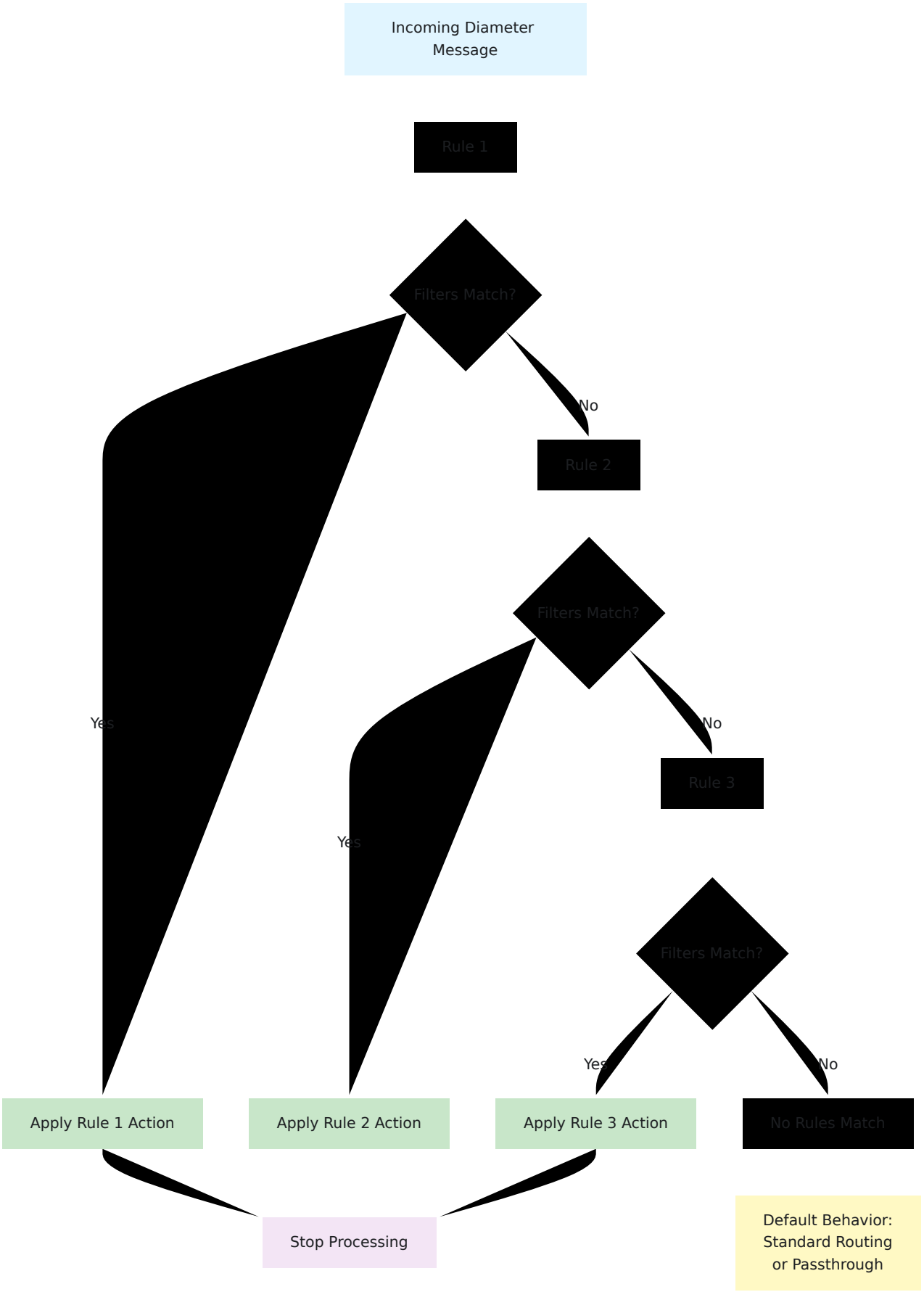
- **MVNO Support**: Route virtual operator traffic to hosted core networks
- **Network Migration**: Gradually redirect subscribers to new infrastructure
- **Realm Translation**: Convert between different naming schemes for roaming partners
- **Multi-tenancy**: Isolate subscriber populations by realm
- **Carrier Routing**: Direct traffic to correct carrier networks based on IMSI ranges

---

Rule Processing

Applies to both Advanced Routing and Advanced Transform modules.

Execution Order



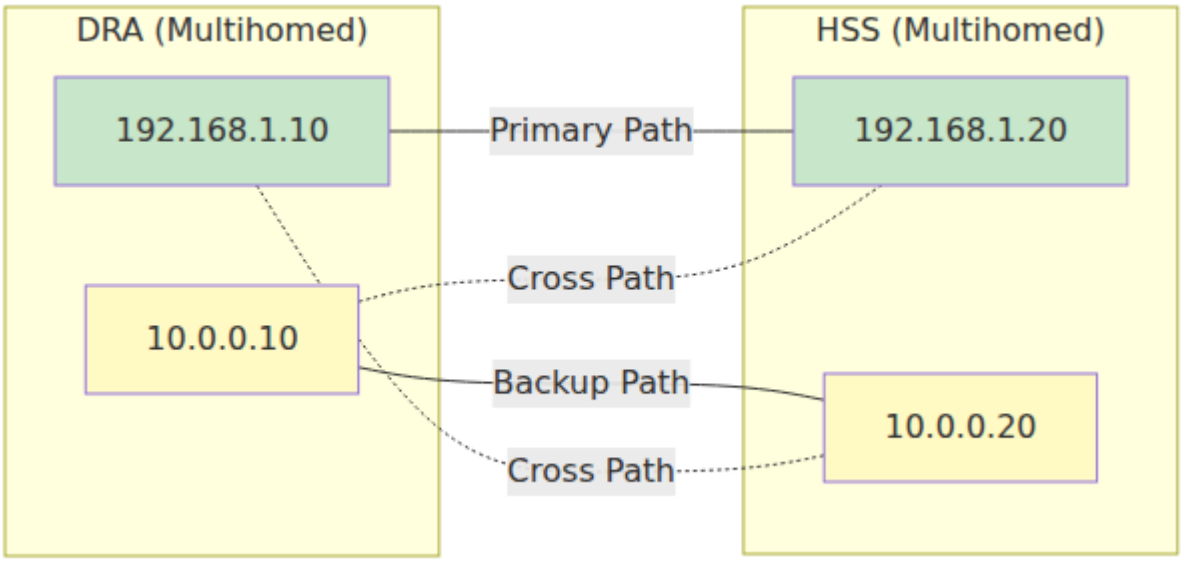
1. Rules are evaluated **in order from top to bottom** as defined in configuration
2. Filters within a rule are evaluated based on the `match` parameter (`:all`, `:any`, or `:none`)
3. **First matching rule wins** - subsequent rules are not evaluated
4. If no rules match, default routing/passthrough behavior is used

Filter Logic

The `match` parameter determines how filters are combined:

match: :all (AND Logic)

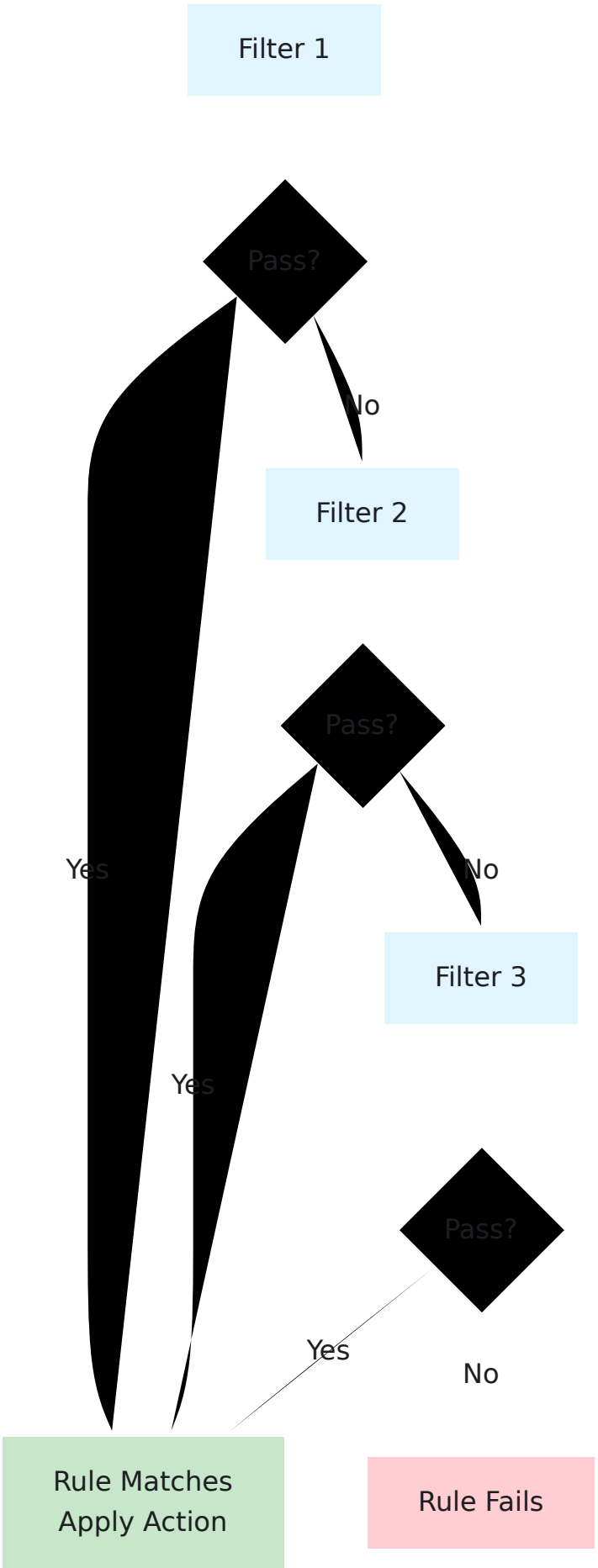
All filters must match for the rule to succeed.



Example: With 3 filters, `filter1 AND filter2 AND filter3` must all be true.

match: :any (OR Logic)

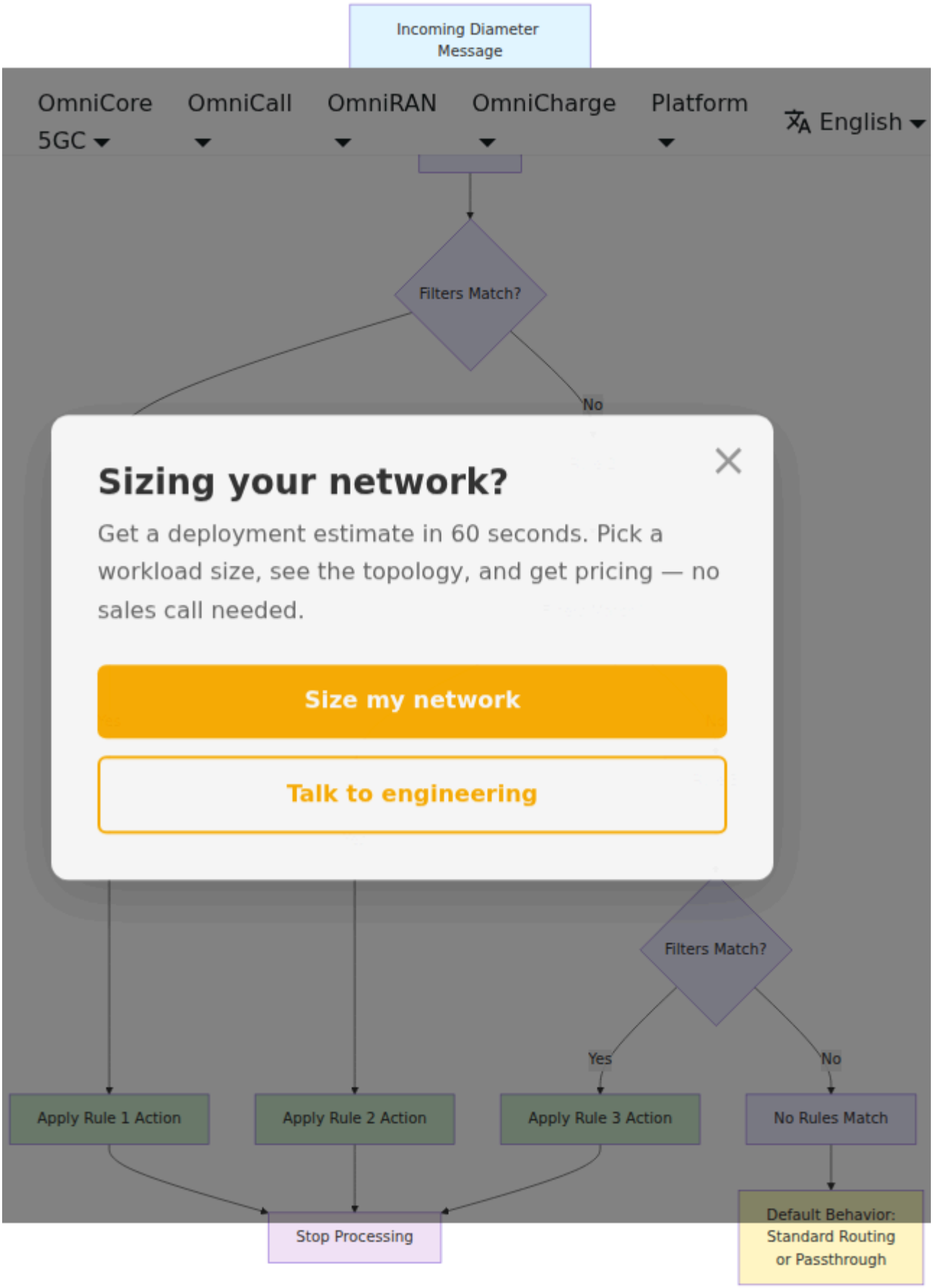
At least one filter must match for the rule to succeed.


Filter 1
Pass?
No
Filter 2
Pass?
Yes
No
Filter 3
Yes
Pass?
Yes
No
Rule Matches
Apply Action
Rule Fails

Example: With 3 filters, `filter1 OR filter2 OR filter3` (any one passes).

match: :none (NOR Logic)

No filters can match for the rule to succeed (inverse matching).



Example: With 3 filters, `NOT filter1 AND NOT filter2 AND NOT filter3` (all must fail).

Additional Notes:

When using list operators within a filter value (e.g., `{:avp, {1, ["value1", "value2"]}}`), the values use **OR** logic (any can match).

Regular Expression Patterns

Use `~r"pattern"` syntax for regex matching:

- `~r"999001.*"` - Matches IMSI starting with 999001
- `~r"^310[0-9]{3}.*"` - Matches IMSI with specific MNC patterns
- `~r".*test$"` - Matches values ending with "test"

Best Practices

1. **Specificity**: Order rules from most specific to most general
2. **Performance**: Place most common matches first to reduce processing overhead
3. **Testing**: Validate regex patterns before deployment
4. **Documentation**: Use descriptive `rule_name` values for operational clarity
5. **Monitoring**: Track rule match rates to verify expected behavior

---

Extended Metrics Module

The Extended Metrics module provides advanced telemetry and analytics capabilities for analyzing Diameter traffic patterns beyond the standard metrics.

Configuration

Enable the module and configure specific metric types:

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

Parameters

Parameter	Description
`enabled`	Set to `true` to activate the extended metrics module
`attach_attempt_reporting_enabled`	Enable tracking and reporting of LTE attach attempts (S6a AIR/AIA)

Available Metrics

Attach Attempt Tracking

Tracks LTE subscriber attach attempts by monitoring Authentication Information Request (AIR) and Answer (AIA) message pairs:

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'

Try again

Measurement: `attach_attempt_count`

Fields:

- `imsi` - The subscriber IMSI (from User-Name AVP - see AVP codes)

Tags:

- `origin_host` - The peer that originated the attach request
- `result_code` - The Diameter result code from the HSS response

How it works:

1. When an AIR (command code 318, S6a application 16777251 - see Application IDs) is received, the module extracts:
   - End-to-End-ID for request/response correlation
   - IMSI (User-Name AVP code 1)
   - Origin-Host (AVP code 264)
2. Request metadata is stored in ETS with TTL
3. When the matching AIA is received, the module:
   - Correlates using End-to-End-ID
   - Extracts the result code (AVP 268 or experimental result code AVP 297)
   - Emits the metric with IMSI, origin host, and result code

Use Cases

- **Attach Success Rate Analysis** - Track successful vs failed attach attempts by result code
- **IMSI-Level Troubleshooting** - Identify subscribers experiencing attach failures
- **Network Performance Monitoring** - Monitor attach attempt patterns by origin (MME/SGSN)
- **Roaming Analytics** - Analyze inbound roaming attach success rates

Integration

Extended metrics are exported via InfluxDB integration:

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

Result codes are standard Diameter codes:

- `2001` - Success (DIAMETER_SUCCESS)
- `5001` - Authentication failure (DIAMETER_AUTHENTICATION_REJECTED)
- `5004` - Diameter AVP unsupported
- See RFC 6733 for complete result code list

Important Notes

- Attach attempt metrics only track S6a AIR/AIA pairs (Application-Id 16777251, Command-Code 318)
- Request metadata expires based on configured request timeout + 5 seconds
- Metric processing is asynchronous (spawned process) to avoid blocking message flow
- The module operates independently from routing and transform modules

---

Security & Steering Modules

The DRA ships with three self-contained, independently configurable modules for interconnect security and roaming control. Each has its own dedicated operations guide:

- **Diameter Security** — GSMA FS.19/FS.21 aligned protection: the Diameter Firewall (low-layer format checks plus Category 1/2/3 filtering), Topology Hiding (Route-Record stripping and Origin rewriting), per-peer Rate Limiting, and AVP Sanitization.
- **Roaming Steering (SoR)** — Steers roaming subscribers towards preferred VPLMNs by rejecting Update-Location-Requests originating from non-preferred networks.
- **Subscriber Lookup (SLF)** — Dynamically learns subscriber-to-serving-node bindings from Diameter traffic and routes SLg/SLh requests to the correct serving node.

Prometheus Metrics

The DRA exposes comprehensive Prometheus metrics for monitoring Diameter traffic, peer health, and module operations. All metrics are available at the `/metrics` endpoint.

Core Diameter Metrics

Peer Status

Metric: `diameter_peer_status` **Type**: Gauge **Description**: Whether the peer is connected (1) or not (0) **Tags**:

- `origin_host` - Peer's Diameter Identity
- `ip` - Peer's IP address

Example:

```
# Check if specific peer is connected
diameter_peer_status{origin_host="hss01.example.com"}

# Count disconnected peers
count(diameter_peer_status == 0)
```

Message Count

Metric: `diameter_peer_message_count_total` **Type**: Counter **Description**: Total number of Diameter messages exchanged with peers **Tags**:

- `origin_host` - Peer's Diameter Identity
- `received_from` - Peer the message was received from
- `application_id` - Diameter Application-Id (see Application ID reference)
- `cmd_code` - Diameter Command-Code (see Common Command Codes)
- `application_name` - Human-readable application name (e.g., "3GPP_S6a")
- `cmd_name` - Human-readable command name (e.g., "AIR")
- `direction` - "request" or "response"

Example:

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

# Total message rate by application
sum by (application_name)
(rate(diameter_peer_message_count_total[5m]))
```

Response Result Codes

Metric: `diameter_peer_message_result_code_count_total` **Type**: Counter **Description**: Total number of Diameter responses by result code **Tags**:

- `origin_host` - Original requester
- `routed_to` - Peer that sent the answer
- `application_id` - Diameter Application-Id
- `cmd_code` - Diameter Command-Code
- `application_name` - Application name
- `cmd_name` - Command name
- `result_code` - Diameter Result-Code or Experimental-Result-Code

Example:

```
# Success rate for S6a AIR requests
rate(diameter_peer_message_result_code_count_total{
  cmd_code="318",
  result_code="2001"
}[5m])

# Error rate by result code
sum by (result_code) (
  rate(diameter_peer_message_result_code_count_total{
    result_code!="2001"
  }[5m])
)
```

Common Result Codes:

- `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

Response Delay

Metric: `diameter_peer_last_response_delay` **Type**: Gauge **Description**: Most recent response delay in milliseconds (DRA → Peer → DRA) **Tags**:

- `origin_host` - Original requester
- `routed_to` - Peer that sent the answer
- `application_name` - Application name
- `cmd_name` - Command name

Example:

```
# Average response time from HSS
avg(diameter_peer_last_response_delay{routed_to="hss01.example.com"})

# P95 response time for S6a
histogram_quantile(0.95,
  rate(diameter_peer_last_response_delay{application_name="3GPP_S6a"}
[5m])
)
```

Unanswered Requests

Metric: `diameter_peer_unanswered_request_count_total` **Type**: Counter **Description**: Requests sent but not answered within timeout period **Tags**:

- `origin_host` - Original requester
- `routed_to` - Peer that didn't answer
- `application_id` - Diameter Application-Id
- `cmd_code` - Diameter Command-Code
- `application_name` - Application name
- `cmd_name` - Command name

Example:

```
# Unanswered request rate
rate(diameter_peer_unanswered_request_count_total[5m])

# Identify problematic peers
topk(5, sum by (routed_to) (
  rate(diameter_peer_unanswered_request_count_total[5m])
))
```

Unauthorized Connection Attempts

Metric: `diameter_peer_unauthorized_connection_count_total` **Type**: Counter **Description**: Connection attempts from unauthorized peers **Tags**:

- `origin_host` - Unauthorized peer's claimed identity
- `supported_applications` - Applications advertised by peer

- `peer_ip` - IP address of connection attempt

Example:

```
# Unauthorized connection attempts
rate(diameter_peer_unauthorized_connection_count_total[5m])

# Alert on unauthorized access
diameter_peer_unauthorized_connection_count_total > 0
```

Advanced Routing Module Metrics

Dropped Traffic

Metric: `diameter_advanced_routing_drop_count_total` **Type**: Counter **Description**: Requests dropped by advanced routing `:drop` action (no response sent) **Tags**:

- `application_id` - Diameter Application-Id
- `cmd_code` - Diameter Command-Code
- `application_name` - Application name
- `cmd_name` - Command name

Example:

```
# Drop rate by command
sum by (cmd_name) (
  rate(diameter_advanced_routing_drop_count_total[5m])
)

# Total dropped traffic
sum(rate(diameter_advanced_routing_drop_count_total[5m]))
```

When traffic is dropped:

- Advanced routing rule matches with `route: :drop`
- Message is silently discarded per Erlang Diameter `:discard` behavior
- No answer message is sent to the requesting peer

- See Advanced Routing Module for configuration

Error Responses

Metric: `diameter_advanced_routing_error_count_total` **Type**: Counter **Description**: Error responses generated by advanced routing `{:error, result_code}` action **Tags**:

- `result_code` - Diameter Result-Code sent in error response
- `application_id` - Diameter Application-Id
- `cmd_code` - Diameter Command-Code
- `application_name` - Application name
- `cmd_name` - Command name

Example:

```
# Error responses by result code
sum by (result_code) (
  rate(diameter_advanced_routing_error_count_total[5m])
)

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

# Rate limiting or overload detection
diameter_advanced_routing_error_count_total{
  result_code="3004"
} > 100
```

When error responses are generated:

- Advanced routing rule matches with `route: {:error, result_code}`
- DRA generates Diameter answer with specified Result-Code
- Answer is sent back to requesting peer (not forwarded)
- Per Erlang Diameter: `{:protocol_error, code}` equivalent to `{:answer_message, code}`
- See Advanced Routing Module for configuration

Common Error Codes Used:

- `3002` - DIAMETER_UNABLE_TO_DELIVER (routing failure)
- `3003` - DIAMETER_REALM_NOT_SERVED (realm not supported)
- `3004` - DIAMETER_TOO_BUSY (overload protection)
- `5012` - DIAMETER_UNABLE_TO_COMPLY (general error)

Important Notes

- All metrics are available at the `/metrics` endpoint in Prometheus format
- **Extended Metrics Module** provides additional S6a-specific metrics (see Extended Metrics Module)
- **BEAM/Erlang metrics** are also exposed but not documented here
- All counters are cumulative and should be queried with `rate()` for per-second rates
- High cardinality tags (like IMSI) are only used in Extended Metrics module to avoid metric explosion

---

Troubleshooting

Rule Not Matching

- Verify all filter conditions are correct
- Check AVP codes match your Diameter application (see AVP code reference)
- Test regex patterns independently (see Regular Expression Patterns)
- Ensure message type matches `match` scope (see Filter Logic)
- Review Available Filters to ensure you're using the correct filter type for your module

Unexpected Routing

- Review rule ordering - first match wins

- Verify peer names are correct and reachable
- Check for conflicting rules with overlapping filters
- Confirm Standard Diameter Routing behavior when no rules match

Transform Not Applied

- Confirm AVP codes are correct for your use case (see AVP code reference)
- For `:edit` action: Verify the AVP exists in the message (edit won't create new AVPs)
- Check that filters match the intended message flow
- Verify packet type filter if used (`:request` vs `:answer`)
- Ensure action is supported for packet type (`:edit` only works on requests - see Transform Actions)
- Review Rule Processing for execution order

Roaming Steering (SoR)

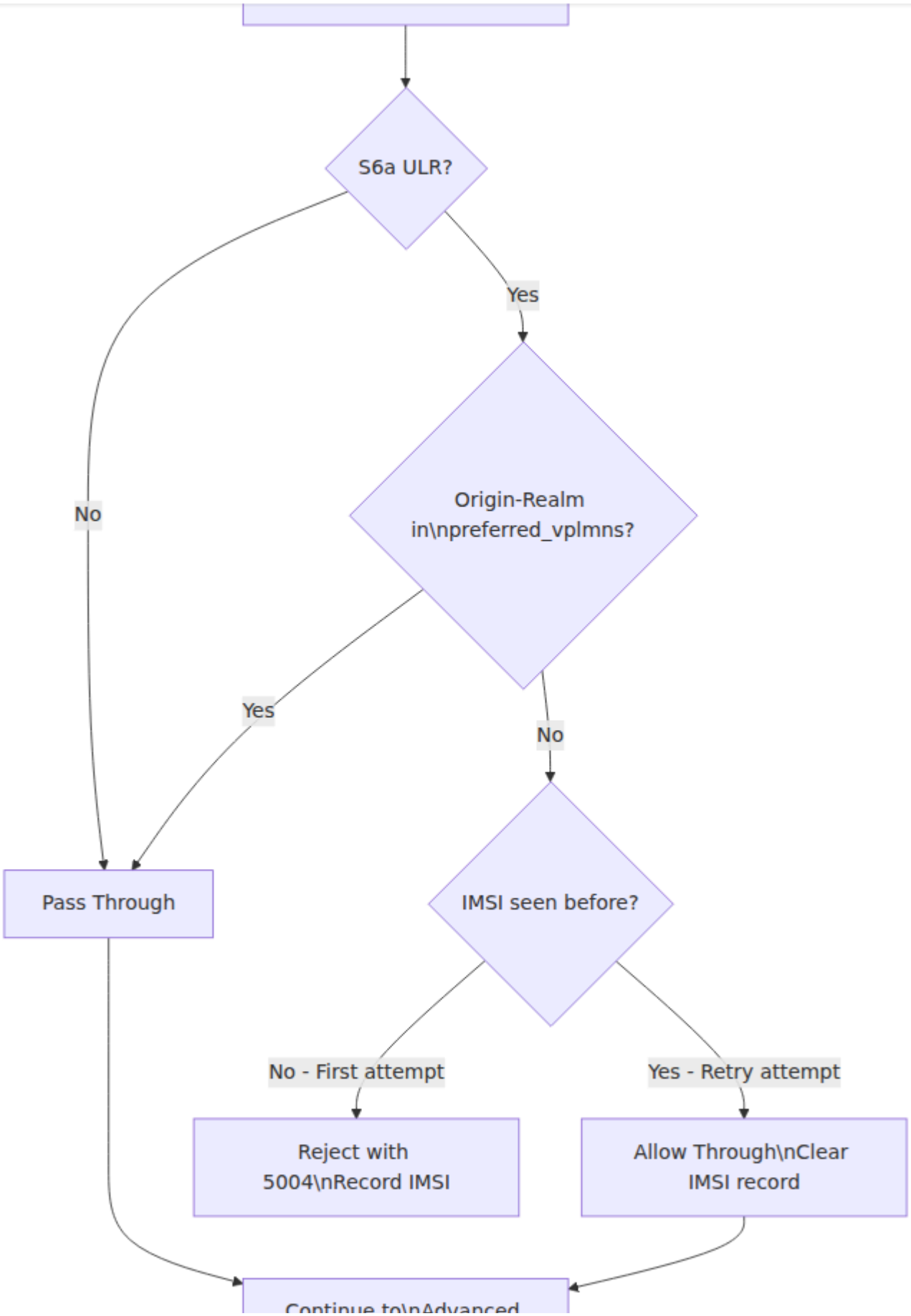
Steering of Roaming (SoR) allows the HPLMN to influence which VPLMN a roaming subscriber attaches to. When a subscriber attempts to register via a non-preferred VPLMN, the DRA intercepts the S6a Update-Location-Request (ULR) and rejects it with **DIAMETER_ERROR_ROAMING_NOT_ALLOWED (5004)**. This causes the UE to detach and re-attach, ideally selecting a preferred VPLMN.

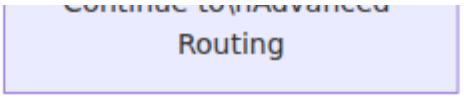
If preferred coverage is unavailable and the subscriber returns to the same non-preferred VPLMN, the module allows the request through on the subsequent attempt.

SoR is defined in 3GPP TS 29.272 Section 5.2.1.1 and 3GPP TS 23.122.

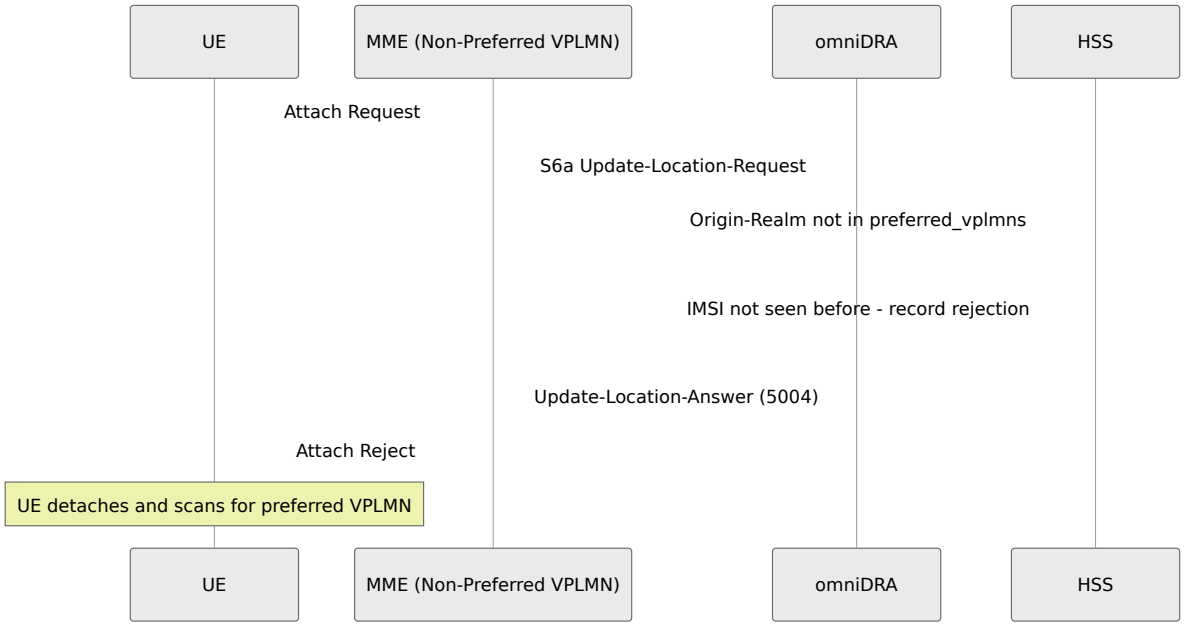
How It Works

The module operates within the DRA request processing pipeline, executing **before** Advanced Routing. Only S6a Update-Location-Requests (Application-Id `16777251`, Command-Code `316`) are evaluated. All other Diameter messages pass through unmodified.

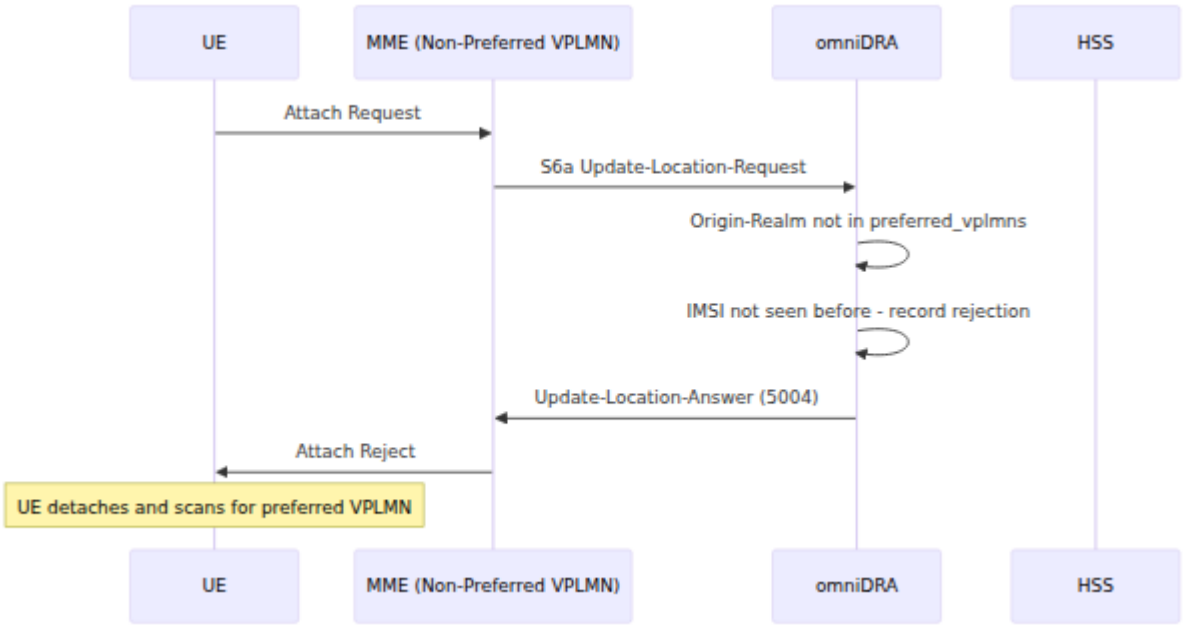

S6a ULR?
Yes
No
Origin-Realm in\npreferred_vplmns?
Yes
No
Pass Through
IMSI seen before?
No - First attempt
Yes - Retry attempt
Reject with 5004\nRecord IMSI
Allow Through\nClear IMSI record
Continue to\nAdvanced



Sequence: First Attempt (Rejected)



Sequence: Second Attempt (Allowed)



Pipeline Position

Roaming Steering runs **before** Advanced Routing in the request processing pipeline. If SoR rejects a request, Advanced Routing is never reached for that message.

Subscriber Tracking

The module tracks each rejected subscriber by IMSI. Each record includes a rejection count and a timestamp. Records are automatically cleaned up after the configured TTL expires.

When a tracked IMSI has reached the `max_rejections` threshold and sends another ULR, the request is allowed through and the tracking record is deleted.

If no further ULR arrives within the TTL window, the record expires and is removed during the next cleanup cycle. A subsequent ULR from the same IMSI will be treated as a first attempt and rejected again.

Configuration

The module is configured under `module_roaming_steering` in `config/runtime.exs`.

```
module_roaming_steering: %{
  # Enable or disable the module
  enabled: false,

  # Number of rejections before allowing through
  max_rejections: 1,

  # Time in seconds before a tracked IMSI record expires
  rejection_ttl_seconds: 300,

  # Diameter Result-Code returned in the rejection answer
  rejection_result_code: 5004,

  # List of Origin-Realm values considered preferred VPLMNs
  # ULRs from these realms are always allowed through
  preferred_vplmns: [
    "epc.mnc001.mcc001.3gppnetwork.org",
    "epc.mnc002.mcc001.3gppnetwork.org"
  ]
}
```

Parameters

Parameter	Type	Required	Default	
`enabled`	Boolean	Yes	`false`	Enable or disab module. When through unmoc
`max_rejections`	Integer	No	`1`	Number of time before allowing `1`, the first ULR allowed.
`rejection_ttl_seconds`	Integer	No	`300`	Time in second expires and is does not retry attempt is trea controls the cle
`rejection_result_code`	Integer	No	`5004`	Diameter Resu rejection answe DIAMETER_ERR per 3GPP TS 29
`preferred_vplmns`	List	No	`[]`	List of Origin-R preferred VPLM peers with a m always allowed

Configuration Examples

Reject Once, Then Allow

The default behaviour. Subscribers hitting a non-preferred VPLMN are rejected once. If they return, the ULR is forwarded to the HSS.

```
module_roaming_steering: %{
  enabled: true,
  max_rejections: 1,
  rejection_ttl_seconds: 300,
  rejection_result_code: 5004,
  preferred_vplmns: [
    "epc.mnc001.mcc310.3gppnetwork.org",
    "epc.mnc002.mcc310.3gppnetwork.org"
  ]
}
```

How it works: A ULR arriving from an MME in `epc.mnc003.mcc310.3gppnetwork.org` (not in the preferred list) is rejected with Result-Code 5004. The UE detaches and attempts to find a preferred network. If it returns to the same non-preferred VPLMN within 300 seconds, the ULR is forwarded normally.

Use case: Standard roaming steering where the operator has preferred roaming agreements with specific partner networks.

Multiple Rejections Before Allow

For scenarios where the operator wants the UE to make multiple attempts before falling back to a non-preferred VPLMN.

```
module_roaming_steering: %{
  enabled: true,
  max_rejections: 3,
  rejection_ttl_seconds: 600,
  rejection_result_code: 5004,
  preferred_vplmns: [
    "epc.mnc001.mcc310.3gppnetwork.org"
  ]
}
```

How it works: The subscriber is rejected three times before being allowed through on the fourth attempt. The TTL is extended to 600 seconds to accommodate the additional retry cycles.

Use case: Dense urban areas where preferred coverage exists but may require multiple attach attempts to acquire.

Metrics

Rejections

Metric: `diameter.roaming_steering.reject.count` **Type**: Counter
Description: Incremented each time a ULR is rejected by the SoR module.
Labels:

- `origin_realm` - Origin-Realm of the non-preferred VPLMN
- `result_code` - Diameter Result-Code returned in the rejection
- `imsi` - IMSI of the rejected subscriber

Allowed (Preferred VPLMN)

Metric: `diameter.roaming_steering.allow.count` **Type**: Counter
Description: Incremented when a ULR is allowed through. **Labels**:

- `origin_realm` - Origin-Realm of the originating VPLMN
- `reason` - Why the request was allowed: `preferred_vplmn` or `max_rejections_reached`
- `imsi` - IMSI of the subscriber (present only when reason is `max_rejections_reached`)

Error Responses

Metric: `diameter.roaming_steering.error.count` **Type**: Counter
Description: Incremented when the processor returns an error answer message due to a SoR rejection. **Labels**:

- `result_code` - Diameter Result-Code
- `application_id` - Diameter Application-Id (numeric)
- `cmd_code` - Diameter Command-Code (numeric)
- `application_name` - Human-readable application name

- `cmd_name` - Human-readable command name

Troubleshooting

Subscribers Always Rejected (Never Allowed Through)

Symptoms: Subscribers on non-preferred VPLMNs are repeatedly rejected and never successfully attach.

Possible causes:

- `rejection_ttl_seconds` is too short, causing the tracking record to expire before the subscriber retries
- `max_rejections` is set higher than the number of retries the UE performs before giving up

Resolution:

1. Increase `rejection_ttl_seconds` to accommodate the UE's retry timing
2. Verify `max_rejections` is set to a value the UE can realistically reach within its retry cycle

Subscribers on Preferred VPLMNs Being Rejected

Symptoms: ULRs from preferred partner networks are being rejected with 5004.

Possible causes:

- The Origin-Realm of the preferred VPLMN is not listed in `preferred_vplmns`
- The Origin-Realm string does not exactly match (case-sensitive)

Resolution:

1. Check the peer's Origin-Realm in the DRA logs at connection time

2. Ensure the exact Origin-Realm string is included in the `preferred_vplmns` list

Module Not Taking Effect

Symptoms: ULRs from non-preferred VPLMNs are being forwarded without rejection.

Possible causes:

- `enabled` is set to `false`
- The module process is not running (check supervisor)

Resolution:

1. Verify `enabled: true` in the configuration
2. Confirm the DRA was restarted after the configuration change

Reference

Diameter Codes

Code	Name	Description	Refer
5004	DIAMETER_ERROR_ROAMING_NOT_ALLOWED	Subscriber not permitted to roam in this VPLMN	3GPP 29.27 Sectic 7.4.4

S6a Commands

Command-Code	Name	Description	Reference
316	Update-Location-Request/Answer (ULR/ULA)	Sent by the MME to the HSS during attach to update the subscriber's serving node	3GPP TS 29.272 Section 7.2.3

Application IDs

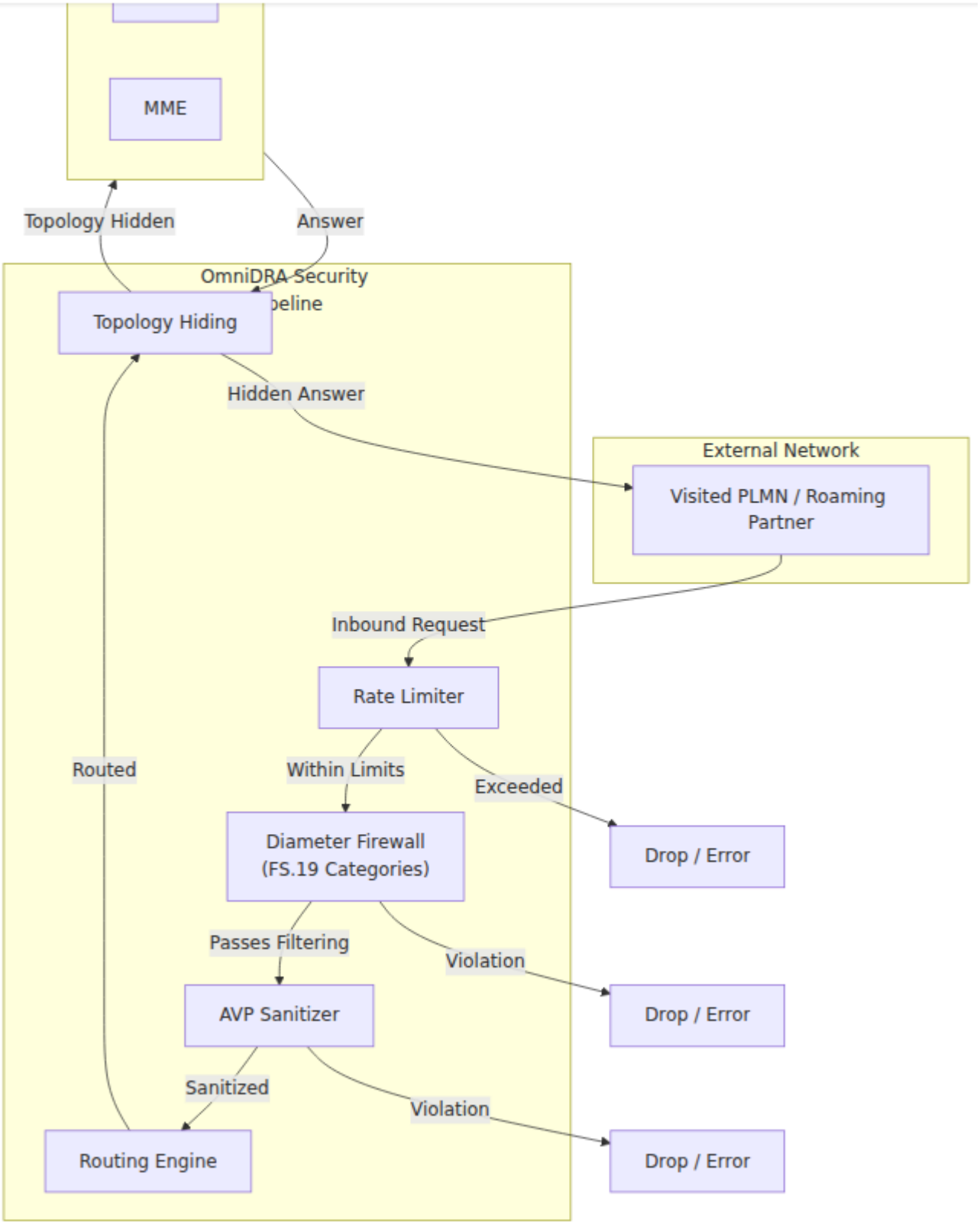
ID	Interface	Description	Reference
16777251	S6a/S6d	MME/SGSN to HSS authentication and subscription management	3GPP TS 29.272

Diameter Security

OmniDRA provides a comprehensive suite of Diameter security modules aligned with **GSMA FS.19** (Diameter Interconnect Security, v10.0) and **GSMA FS.21** (Interconnect Signalling Security Recommendations, v12.0). These modules protect the network against signalling-level attacks on Diameter interconnect interfaces.

Each module is independently configurable, can be enabled or disabled without affecting the others, and emits its own telemetry events for monitoring and alerting.

Architecture Overview



Processing Order

Inbound Diameter requests pass through the security modules in the following order. A message rejected at any stage is never forwarded to subsequent stages.

Order	Module	Direction	Purpose
1	Rate Limiter	Inbound	Volumetric flood protection
2	Diameter Firewall	Inbound	FS.19 protocol and content filtering
3	AVP Sanitizer	Inbound	AVP validation and stripping
4	Routing Engine	-	Standard Diameter routing
5	Topology Hiding	Outbound	Network topology concealment

Actions

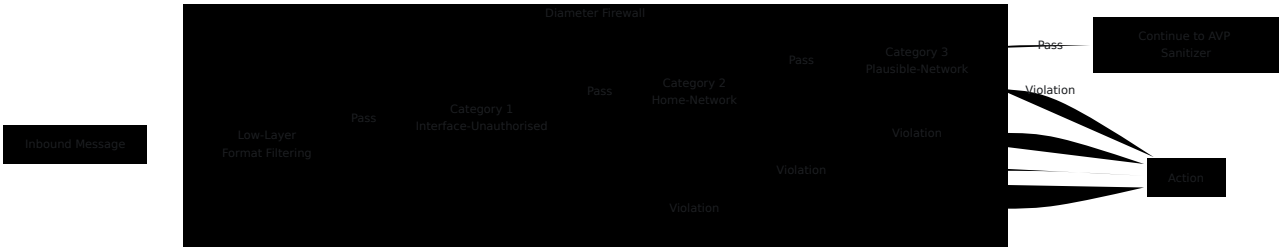
All security modules support configurable actions when a violation is detected. Actions are configured per-module (and per-category for the Diameter Firewall).

Action	Behaviour	Diameter Result
`{:error, 3002}`	Respond with a Diameter error answer	DIAMETER_UNABLE_TO_DELIVER
`{:error, 3004}`	Respond with a Diameter error answer	DIAMETER_TOO_BUSY
`{:error, 3007}`	Respond with a Diameter error answer	DIAMETER_APPLICATION_UNSUPPORTED
`:drop`	Silently discard the message	No response sent
`:log_only`	Log the violation but allow the message through	Message continues

---

Diameter Firewall

The Diameter Firewall implements the four filtering layers defined in **GSMA FS.19 Section 3.3** and the protocol-agnostic packet categorisation from **FS.21 Section 7**. Messages are evaluated through each layer in order; a violation at any layer stops further processing.



Low-Layer Format Filtering

GSMA Reference: FS.19 Section 3.3.4, FS.21 Section 7.3.1

Low-layer filtering detects protocol-level violations and basic anti-spoofing attempts without needing to understand upper-layer application semantics. This layer catches malformed messages before they reach deeper inspection.

Checks performed:

Check	FS.19 Reference	Description
AVP Duplication	Section 3.3.4, 4.8.1	Detects duplicate instances of AVPs that must appear at most once (e.g. Origin-Host, Origin-Realm). Prevents AVP doubling evasion attacks.
Session-Id Ordering	Section 3.3.4	Validates that Session-Id (AVP 263) is the first AVP in the message, per RFC 6733 Section 8.8.
Destination-Host in Answers	Section 3.3.4	Rejects answer messages that contain a Destination-Host AVP, which is a protocol violation that may indicate filter evasion.
Mandatory AVP Validation	Section 3.3.5	Validates that required AVPs are present for each command code (e.g. ULR must contain User-Name and Visited-PLMN-Id per 3GPP TS 29.272 Section 7.2.3).

```
low_layer: %{
  enabled: true,
  action: {:error, 3002},
  # AVP codes that must not appear more than once (FS.19 Section
3.3.4)
  # 264 = Origin-Host, 296 = Origin-Realm, 283 = Destination-
Realm, 293 = Destination-Host
  single_instance_avps: [264, 296, 283, 293],
  # Session-Id must be the first AVP (RFC 6733 Section 8.8)
  enforce_session_id_first: true,
  # Answer messages must not contain Destination-Host (FS.19
Section 3.3.4)
  reject_destination_host_in_answers: true,
  # Mandatory AVPs per command code (3GPP TS 29.272)
  # 1 = User-Name (IMSI), 1407 = Visited-PLMN-Id, 264 = Origin-
Host, 296 = Origin-Realm
  mandatory_avps: %{
    316 => [1, 1407, 264, 296],
    318 => [1, 1407, 264, 296]
  }
}
```

Category 1 — Interface-Unauthorised Packet Filtering

GSMA Reference: FS.19 Section 3.3.5, FS.21 Section 7.2.1

Category 1 filtering ensures that only authorised Diameter Application IDs and Command Codes are accepted on each interconnect interface. This prevents external access to internal-only interfaces (e.g. blocking Sh commands over an S6a interface) and enforces that roaming partners only send message types covered by their roaming agreements.

The whitelist approach follows the FS.19 recommendation: **block all Diameter messages except those explicitly required for a given interface**.

Whitelists can be configured globally (applying to all peers) or per-peer, allowing different roaming partners to have different permitted message sets.

```
category_1: %{
  enabled: true,
  action: {:error, 3007},
  whitelists: %{
    # Per-peer whitelist — restrict this partner to ULR and AIR
only (FS.19 Section 3.3.5)
    "restricted-partner.roaming.com" => %{
      16_777_251 => [316, 318]
    },
    # Default whitelist — standard S6a commands permitted for all
other peers
    all: %{
      # S6a/S6d (FS.19 Section 3.3.5, Table 2)
      16_777_251 => [316, 317, 318, 319, 320, 321, 323],
      # S13 — ME Identity Check (FS.19 Section 3.3.5)
      16_777_252 => [324],
      # S6c — SMS via HSS (FS.19 Section 3.3.5.1)
      16_777_312 => [8388647, 8388648],
      # SGd — SMS via MME (FS.19 Section 3.3.5.2)
      16_777_313 => [8388645, 8388646]
    }
  }
}
```

Common S6a/S6d Whitelist:

Command Code	Name	Direction	Reference
316	Update-Location-Request/Answer	MME → HSS	3GPP TS 29.272 §7.2.3
317	Cancel-Location-Request/Answer	HSS → MME	3GPP TS 29.272 §7.2.7
318	Authentication-Information-Request/Answer	MME → HSS	3GPP TS 29.272 §7.2.5
319	Insert-Subscriber-Data-Request/Answer	HSS → MME	3GPP TS 29.272 §7.2.9
320	Delete-Subscriber-Data-Request/Answer	HSS → MME	3GPP TS 29.272 §7.2.11
321	Purge-UE-Request/Answer	MME → HSS	3GPP TS 29.272 §7.2.13
323	Notify-Request/Answer	MME → HSS	3GPP TS 29.272 §7.2.15

Common Application IDs for roaming interconnect (FS.19 Section 3.3.5):

Application ID	Interface	Reference
16777251	S6a/S6d	3GPP TS 29.272
16777252	S13	3GPP TS 29.272
16777312	S6c	3GPP TS 29.338
16777313	SGd	3GPP TS 29.338
16777255	SLg	3GPP TS 29.172
16777267	S9	3GPP TS 29.215

Category 2 — Home-Network Packet Filtering

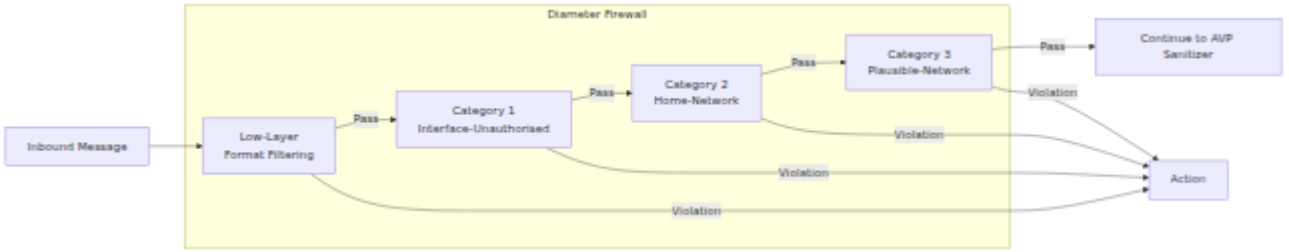
GSMA Reference: FS.19 Section 3.3.6, FS.21 Section 7.2.2

Category 2 filtering protects home subscribers from being targeted by messages arriving from the interconnect. Messages on protected command codes are inspected for subscriber identity (IMSI in the User-Name AVP, MSISDN in AVP 701). If the identity matches a home subscriber prefix, the message is rejected — legitimate traffic for home subscribers should originate from within the home network, not from external peers.

This prevents attacks where an external entity sends location updates, subscriber data requests, or authentication queries targeting the operator's own subscribers.

```
# Top-level: define home subscriber prefixes
home_imsi_prefixes: ["31338", "31339"],
home_msisdn_prefixes: ["+1313"],

category_2: %{
  enabled: true,
  action: {:error, 3002},
  # S6a command codes that carry subscriber identity (FS.19
Section 3.3.6, Table 12)
  protected_command_codes: [316, 317, 318, 319, 320, 321, 323]
}
```



Category 3 — Plausible-Network Packet Filtering

GSMA Reference: FS.19 Section 3.3.7, FS.21 Section 7.2.3

Category 3 filtering detects implausible location changes by tracking the last-seen network for each subscriber (IMSI). When an Update-Location-Request arrives from a different visited network than the previous one, the time elapsed is compared against a configurable threshold. A realm change occurring faster than physically possible indicates a potential attack (e.g. spoofed location updates).

This module maintains stateful per-IMSI tracking using an ETS table with automatic cleanup of stale entries (entries older than 24 hours are removed periodically).

Checks performed:

Check	FS.19 Reference	Description
Previous Location Check	Section 3.3.7.1	Compares the Origin-Realm of the current ULR with the last-seen Origin-Realm for that IMSI
Velocity/Time Check	Section 3.3.7.2	Flags realm changes that occur within a configurable minimum time window

```
category_3: %{
  enabled: true,
  # Start with log_only to observe patterns before enforcing
(FS.19 Section 3.3.7)
  action: :log_only,
  # Maximum plausible velocity in km/h (FS.19 Section 3.3.7.2)
  max_velocity_kmh: 1200,
  # Minimum seconds between ULRs from different realms for the
same IMSI
  min_time_between_updates_seconds: 2
}
```

Configuration

The Diameter Firewall is enabled at the top level, with each filtering layer configured independently. The config snippets shown above within each category section are combined under a single `module_diameter_firewall` key:

```
config :dra,
  module_diameter_firewall: %{
    enabled: true,
    home_imsi_prefixes: ["31338", "31339"],
    home_msisdn_prefixes: ["+1313"],
    low_layer: %{ ... },      # See Low-Layer Format Filtering
above
    category_1: %{ ... },     # See Category 1 above
    category_2: %{ ... },     # See Category 2 above
    category_3: %{ ... }      # See Category 3 above
  }
```

Top-Level Parameters

Parameter	Type	Required	Default	Description
`enabled`	Boolean	Yes	`false`	Enable or disable the entire Diameter Firewall module. When `false`, all messages pass through without inspection.
`home_imsi_prefixes`	List	No	`[]`	List of IMSI prefix strings identifying home subscribers. Used by Category 2 filtering. Example: `["31338", "31339"]`.
`home_msisdn_prefixes`	List	No	`[]`	List of MSISDN prefix strings identifying home subscribers. Used by Category 2 filtering.

Parameter	Type	Required	Default	Description
				Example: `["+1313"]`.

Low-Layer Parameters

Parameter	Type	Required	Default
`enabled`	Boolean	No	`true`
`action`	Action	No	`{:error, 3002}`
`single_instance_avps`	List	No	`[264, 296, 283, 293]`
`enforce_session_id_first`	Boolean	No	`true`

Parameter	Type	Required	Default
`reject_destination_host_in_answers`	Boolean	No	`true`
`mandatory_avps`	Map	No	`%{}`

Category 1 Parameters

Parameter	Type	Required	Default	Description
`enabled`	Boolean	No	`true`	Enable or disable Category
`action`	Action	No	`{:error, 3007}`	Action to take when a Cate violation is detected. Defa with DIAMETER_APPLICATION_U
`whitelists`	Map	Yes	-	Map of peer hostname (or permitted Application ID / Code combinations. The `:` provides a default whitelis specific entries take priorit

Category 2 Parameters

Parameter	Type	Required	Default	Descriptio
`enabled`	Boolean	No	`true`	Enable or disable Category 2 filtering.
`action`	Action	No	`{:error, 3002}`	Action to take when a home subscriber i targeted from interconnec
`protected_command_codes`	List	No	`[]`	Command codes that trigger hom subscriber identity checks. Typically the S6a command codes that carry subscriber identity.

Category 3 Parameters

Parameter	Type	Required	Default
`enabled`	Boolean	No	`false`
`action`	Action	No	`:log_only`
`max_velocity_kmh`	Integer	No	`1200`

Parameter	Type	Required	Default
`min_time_between_updates_seconds`	Integer	No	`2`

Telemetry Events

Event	Description
`[:diameter, :firewall, :low_layer, :block]`	Low-layer format violation detected and blocked
`[:diameter, :firewall, :category_1, :block]`	Category 1 violation detected and blocked
`[:diameter, :firewall, :category_2, :block]`	Category 2 violation detected and blocked
`[:diameter, :firewall, :category_3, :block]`	Category 3 violation detected and blocked
`[:diameter, :firewall, :pass, :count]`	Message passed all firewall checks

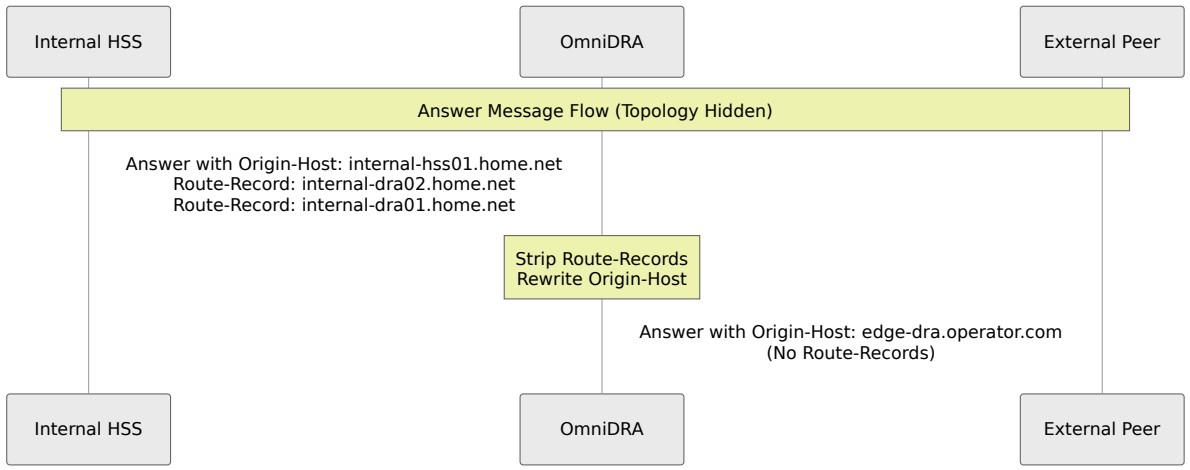
All events include metadata: `origin_host`, `application_id`, `command_code`, `application_name`, `command_name`, and `reason`.

Topology Hiding

GSMA Reference: FS.19 Section 2.4, Section 3.4; FS.21 Section 3.6

The Topology Hiding module prevents internal network topology from being exposed to external peers. When Diameter messages traverse multiple internal nodes, they accumulate Route-Record AVPs and carry Origin-Host/Origin-Realm values that reveal internal hostnames, network structure, and node counts. This information can be used by attackers to map the internal network for targeted attacks.

Topology hiding operates on the **outbound path** — after routing decisions have been made, before messages are forwarded to the destination peer.



Features

Feature	FS.19 Reference	Description
Route-Record Stripping	Section 2.4	Removes all Route-Record AVPs (282) that reveal internal routing paths and node hostnames
Origin-Host Rewriting	Section 3.4	Replaces the Origin-Host AVP with the DRA's own identity (or a custom value) on answer messages
Origin-Realm Rewriting	Section 3.4	Optionally replaces the Origin-Realm AVP to hide internal realm structure
Per-Peer Control	-	Apply topology hiding selectively — only to external peers, or to all peers

```
module_topology_hiding: %{
  enabled: true,
  # Strip Route-Record AVPs revealing internal paths (FS.19
Section 2.4)
  strip_route_records: true,
  # Rewrite Origin-Host on answers to hide internal node names
(FS.19 Section 3.4)
  rewrite_origin_host: %{enabled: true, replacement: :self},
  # Optionally hide internal realm structure
  rewrite_origin_realm: %{enabled: false, replacement: :self},
  # Apply to all external peers, or list specific hostnames
  external_peers: :all
}
```

Parameters

Parameter	Type	Required	Default	Description
`enabled`	Boolean	Yes	`false`	Enable or disable the Topology Hiding module.
`strip_route_records`	Boolean	No	`true`	Remove all Route-Record AVPs (code 282) from messages before forwarding to external peers.
`rewrite_origin_host`	Map	No	See below	Controls Origin-Host rewriting on answer messages.
`rewrite_origin_realm`	Map	No	See below	Controls Origin-Realm rewriting on answer messages.
`external_peers`	`:all` or List	No	`:all`	Peers considered external. Topology hiding is only

Parameter	Type	Required	Default	Description
				applied to messages destined for these peers. Use `:all` for interconnect deployments. Use a list of hostnames for selective application.

Rewrite Parameters (apply to both `rewrite_origin_host` and `rewrite_origin_realm`):

Parameter	Type	Required	Default	Description
`enabled`	Boolean	No	varies	Enable rewriting for this AVP. Default is `true` for Origin-Host, `false` for Origin-Realm.
`replacement`	`:self` or String	No	`:self`	Replacement value. `:self` uses the DRA's own identity from the `diameter` config (`host.realm`). A string value is used as-is.

Telemetry Events

Event	Description
`[:diameter, :topology_hiding, :route_record, :stripped]`	Route-Record AVPs removed. Measurement includes `count` of AVPs stripped.
`[:diameter, :topology_hiding, :origin_host, :rewritten]`	Origin-Host AVP rewritten
`[:diameter, :topology_hiding, :origin_realm, :rewritten]`	Origin-Realm AVP rewritten

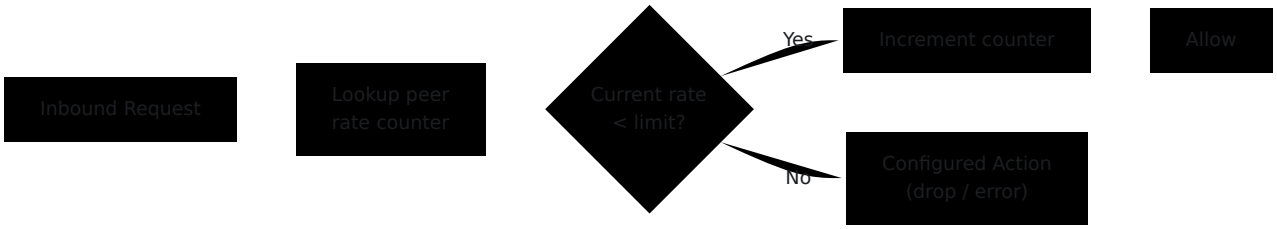
Rate Limiter

GSMA Reference: FS.19 Section 3.4 (Availability / DoS Protection)

The Rate Limiter enforces per-peer message rate limits to protect against volumetric attacks and message flooding. It operates as the first security check in the pipeline — before any message parsing or content inspection — to shed excess load as early as possible.

Rate limits are tracked per-peer using a sliding window counter. Each peer has an independent counter that resets every second.

```
module_rate_limiter: %{
  enabled: true,
  # Default limit for all peers (FS.19 Section 3.4)
  default_max_requests_per_second: 1000,
  default_action: {:error, 3004},
  # Per-peer overrides
  peer_limits: %{
    "high-volume-partner.roaming.com" => %
{max_requests_per_second: 5000, action: {:error, 3004}},
    "restricted-peer.roaming.com" => %{max_requests_per_second:
100, action: :drop}
  }
}
```



Parameters

Parameter	Type	Required	Default	
`enabled`	Boolean	Yes	`false`	En Ra
`default_max_requests_per_second`	Integer	No	`1000`	De re all sir
`default_action`	Action	No	`{:error, 3004}`	De pe lim `30` DI
`peer_limits`	Map	No	`%{}`	Ma ho co no de

Per-Peer Override Parameters:

Parameter	Type	Required	Default	Description
`max_requests_per_second`	Integer	No	Inherits default	Maximum requests per second for this specific peer.
`action`	Action	No	Inherits default	Action when this peer exceeds its rate limit.

Telemetry Events

Event	Description
`[:diameter, :rate_limiter, :throttled]`	A message was rate limited. Measurements include `current_rate` and `limit`.
`[:diameter, :rate_limiter, :allowed]`	A message was within rate limits.

AVP Sanitizer

GSMA Reference: FS.19 Section 3.3.4, 4.8.1, 4.8.2; FS.21 Section 3.6

The AVP Sanitizer validates and sanitizes Diameter AVPs at the interconnect boundary. It addresses the FS.21 Section 3.6 recommendations for handling protocol-level manipulation attacks that operate below the level of the FS.19 filtering categories.

Features

Feature	GSMA Reference	Description
Unknown Vendor AVP Stripping	FS.21 Section 3.6	Removes AVPs from non-whitelisted vendors. Prevents injection of proprietary AVPs that may trigger unintended behaviour in backend nodes.
Grouped AVP Nesting Depth	FS.21 Section 3.6	Enforces a maximum nesting depth for grouped AVPs. Prevents stack overflow attacks using deeply nested AVP structures.

```
module_avp_sanitizer: %{
  enabled: true,
  action: {:error, 3002},
  # Only allow standard vendor AVPs on interconnect (FS.21 Section
3.6)
  # 0 = IETF, 10415 = 3GPP, 13019 = ETSI, 5535 = 3GPP2
  allowed_vendor_ids: [0, 10415, 13019, 5535],
  strip_unknown_vendor_avps: true,
  # Prevent stack overflow via deeply nested grouped AVPs (FS.21
Section 3.6)
  max_avp_nesting_depth: 10
}
```

Parameters

Parameter	Type	Required	Default	Desc
`enabled`	Boolean	Yes	`false`	Enable or AVP Sanit module.
`action`	Action	No	`{:error, 3002}`	Action to a nesting violation Vendor st silently re offending without b message.
`allowed_vendor_ids`	List	No	`[0, 10415, 13019, 5535]`	List of ve permitted interconn from othe are stripp
`strip_unknown_vendor_avps`	Boolean	No	`true`	Enable st AVPs from not in the `allowed_` list.
`max_avp_nesting_depth`	Integer	No	`10`	Maximum depth of AVP nesti Messages this are s the config `action`.

Common Vendor IDs:

Vendor ID	Organisation	Notes
0	IETF	Standard Diameter AVPs defined in RFCs
10415	3GPP	All 3GPP-defined AVPs (S6a, Gx, Rx, etc.)
13019	ETSI	ETSI-defined AVPs
5535	3GPP2	3GPP2-defined AVPs (CDMA interworking)

Telemetry Events

Event	Description
`[:diameter, :avp_sanitizer, :unknown_vendor, :stripped]`	One or more AVPs from unknown vendors were stripped
`[:diameter, :avp_sanitizer, :nesting_depth, :violation]`	A message exceeded the maximum AVP nesting depth
`[:diameter, :avp_sanitizer, :pass, :count]`	Message passed all sanitization checks

Deployment Recommendations

Recommended Enablement Order

When deploying the security modules for the first time, enable them incrementally to avoid disrupting live traffic:

1. **Rate Limiter** — Start with generous limits and monitor traffic patterns. Tighten limits once baseline rates are understood.

2. **AVP Sanitizer** — Low risk of false positives. Enable stripping and nesting checks.
3. **Diameter Firewall (Category 1)** — Define whitelists based on roaming agreements. Start with known-good Application ID / Command Code combinations.
4. **Diameter Firewall (Low-Layer)** — Enable protocol conformance checks.
5. **Diameter Firewall (Category 2)** — Configure home IMSI/MSISDN prefixes.
6. **Topology Hiding** — Enable Route-Record stripping first, then Origin-Host rewriting.
7. **Diameter Firewall (Category 3)** — Enable with `:log_only` action to observe location patterns before enforcing.

Defence in Depth

These modules implement the **defence in depth** principle described in FS.19 Section 3.4. Each layer addresses a different class of attack:

Attack Class	Primary Defence	Secondary Defence
Volumetric DoS	Rate Limiter	Diameter Firewall (all categories)
Interface Abuse	Category 1	AVP Sanitizer
Home Subscriber Targeting	Category 2	Category 1 (interface restriction)
Location Spoofing	Category 3	Category 2 (home sub check)
Topology Discovery	Topology Hiding	-
AVP Injection	AVP Sanitizer	Low-Layer Filtering
Protocol Evasion (AVP Doubling)	Low-Layer Filtering	AVP Sanitizer

GSMA Document Cross-Reference

Module / Feature	FS.19 v10.0	FS.21 v12.0
Low-Layer Format Filtering	Section 3.3.4	Section 7.3.1
Category 1 Filtering	Section 3.3.5, Annex B.3.3	Section 7.2.1
Category 2 Filtering	Section 3.3.6, Annex B.3.4	Section 7.2.2, Section 16
Category 3 Filtering	Section 3.3.7, Annex B.3.5	Section 7.2.3
Topology Hiding	Section 2.4, Section 3.4	Section 3.6
Rate Limiting	Section 3.4	-
AVP Sanitization	Section 4.8.1, 4.8.2	Section 3.6
Defence in Depth	Section 3.4	Section 3.15
Filtering Categories (Protocol-Agnostic)	Annex A	Section 7

Subscriber Lookup Function (SLF)

The Subscriber Lookup Function (SLF) dynamically learns which network element is serving each subscriber by observing Diameter signaling traffic passing through the DRA. It uses these learned bindings to route subsequent requests — such as location service queries — directly to the correct serving node without requiring static routing rules.

This is particularly useful for SLg/SLh location service requests, where the GMLC needs to reach the serving MME for a given subscriber but has no prior knowledge of which MME that is.

The SLF concept is described in 3GPP TS 29.172 and 3GPP TS 29.173 for location services, with subscriber registration defined in 3GPP TS 29.272.

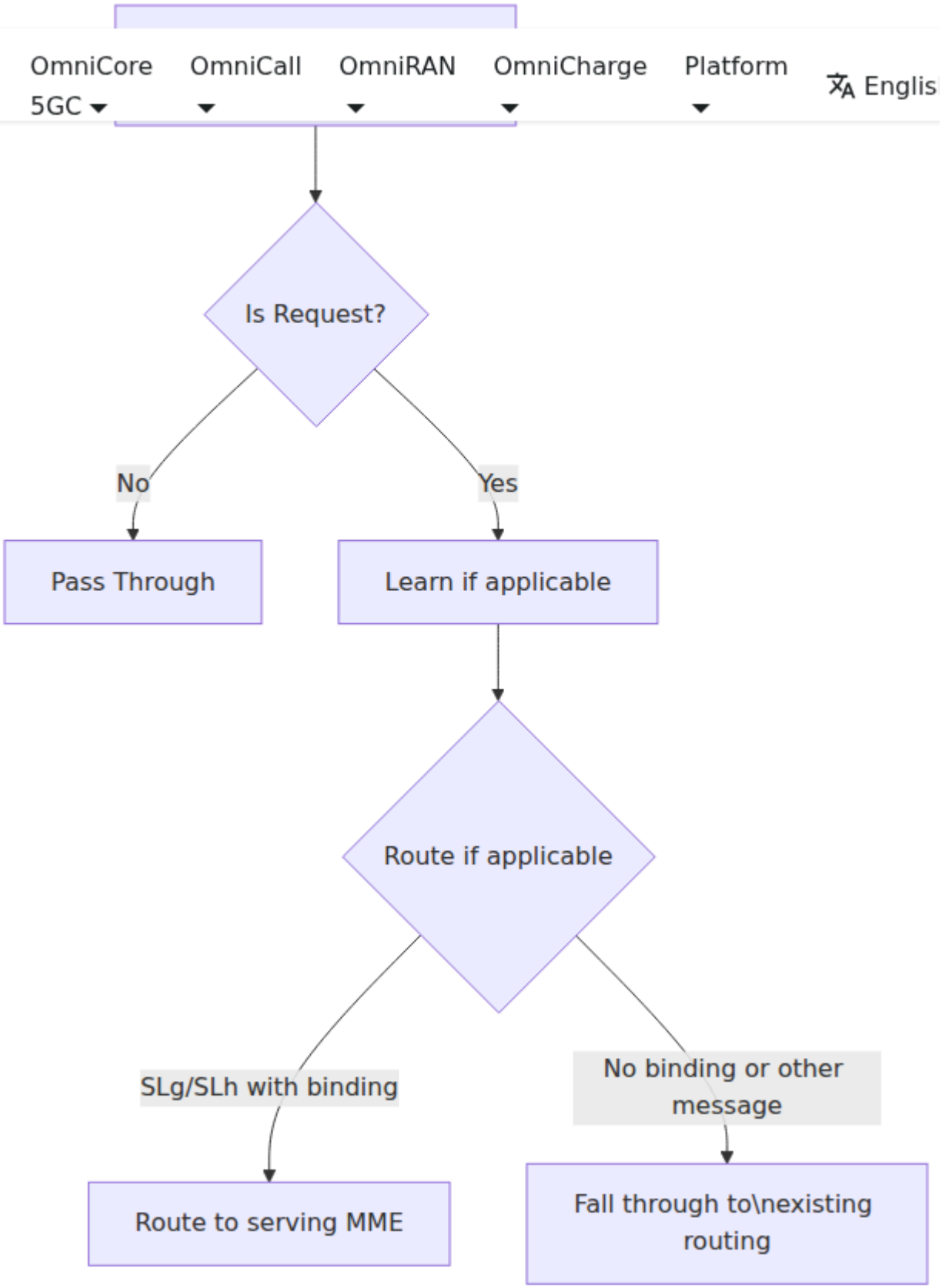
How It Works

The module operates in two phases: **learning** and **routing**.

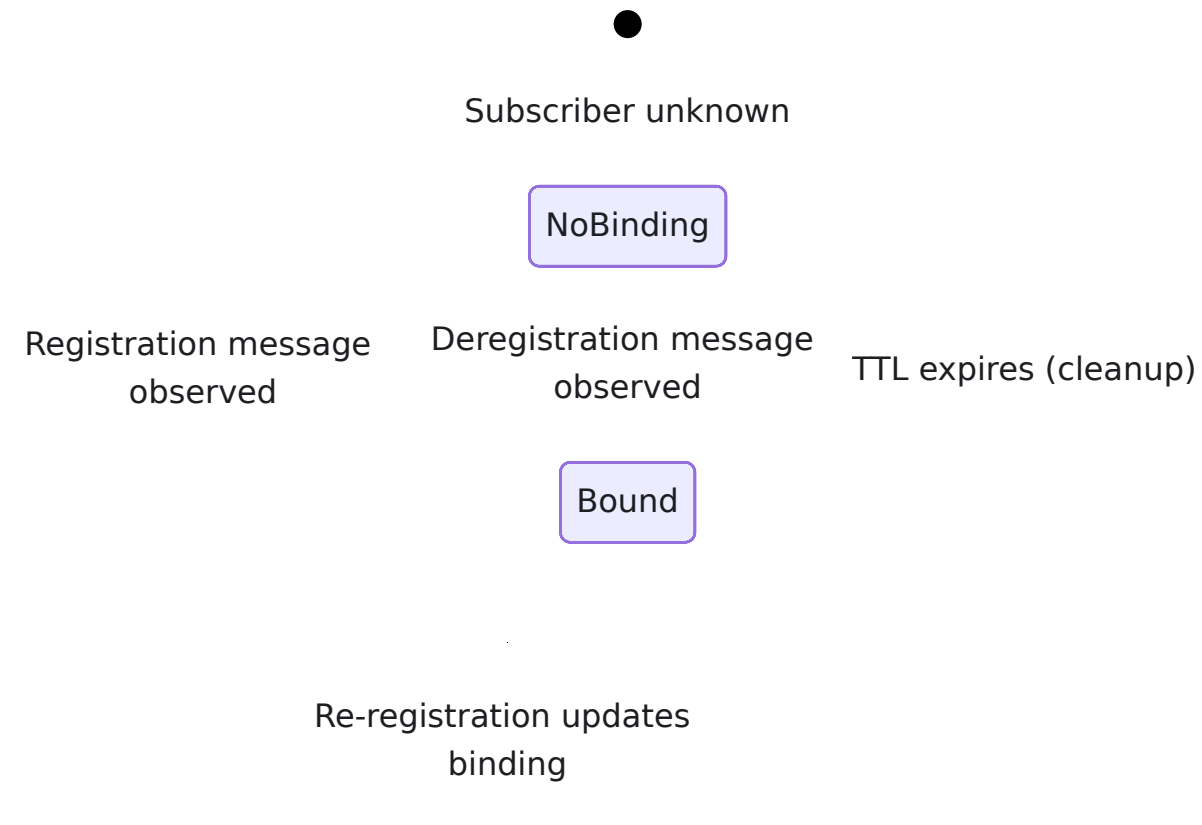
During **learning**, the module passively observes registration and deregistration messages flowing through the DRA. When a subscriber registers (e.g. via an S6a Update-Location-Request), the module records which network element is now serving that subscriber.

During **routing**, when a request arrives that needs to reach a subscriber's serving node (e.g. an SLg Provide-Location-Request), the module looks up the binding and routes directly to the correct peer.

If no binding exists for a subscriber, the request falls through to the existing routing logic (including Advanced Routing).

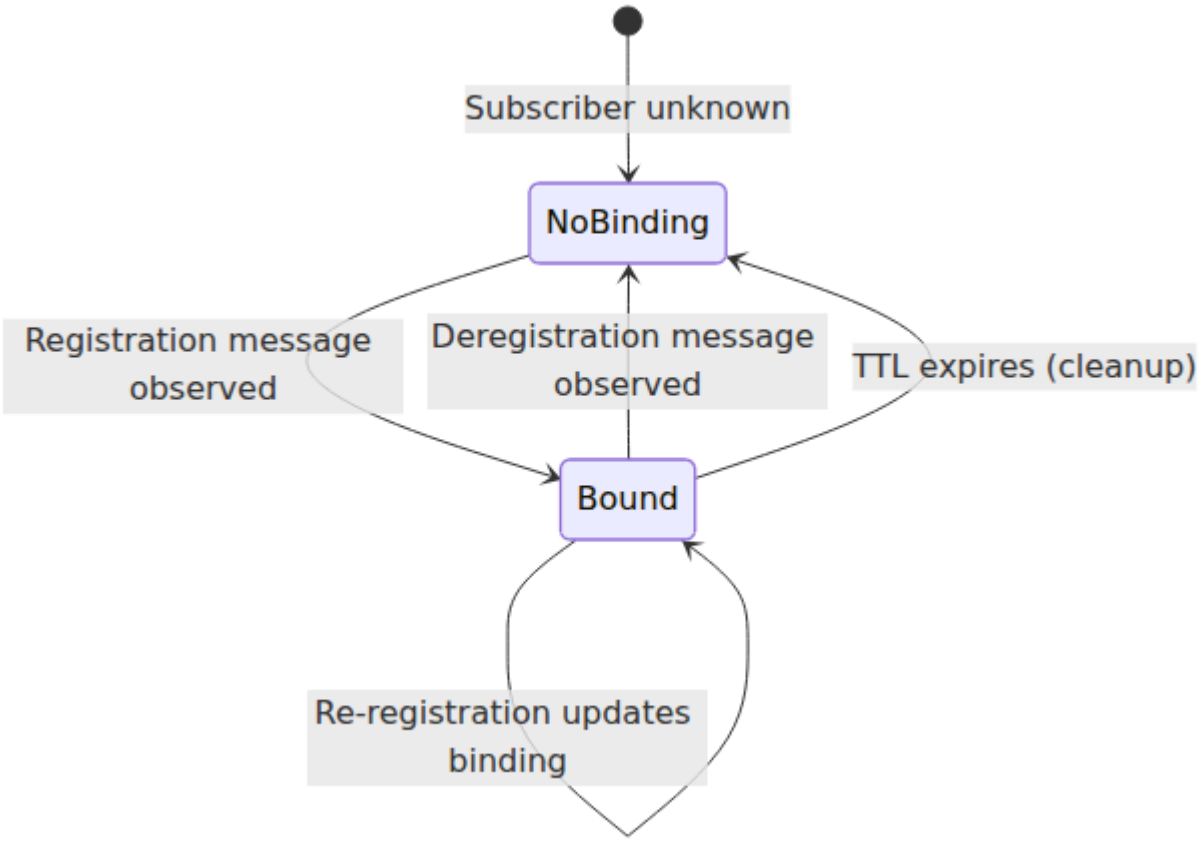

Is Request?
No
Yes
Pass Through
Learn if applicable
Route if applicable
SLg/SLh with binding
No binding or other message
Route to serving MME
Fall through to\nexisting routing

Binding Lifecycle



Learning: Messages That Create Bindings

The module learns subscriber-to-serving-node bindings from the following Diameter messages:



Interface	Message	Command-Code	Binding Created	IMSI Source
S6a	Update-Location-Request (ULR)	316	`serving_mme`	User-Name AVP (1)
Gx	Credit-Control-Request Initial (CCR-I)	272 (CC-Request-Type=1)	`serving_pgw`	Subscription-Id AVP (443)
Cx	Server-Assignment-Request (SAR)	301 (Server-Assignment-Type=1)	`serving_cscf`	Public-Identity AVP (601)

Learning: Messages That Remove Bindings

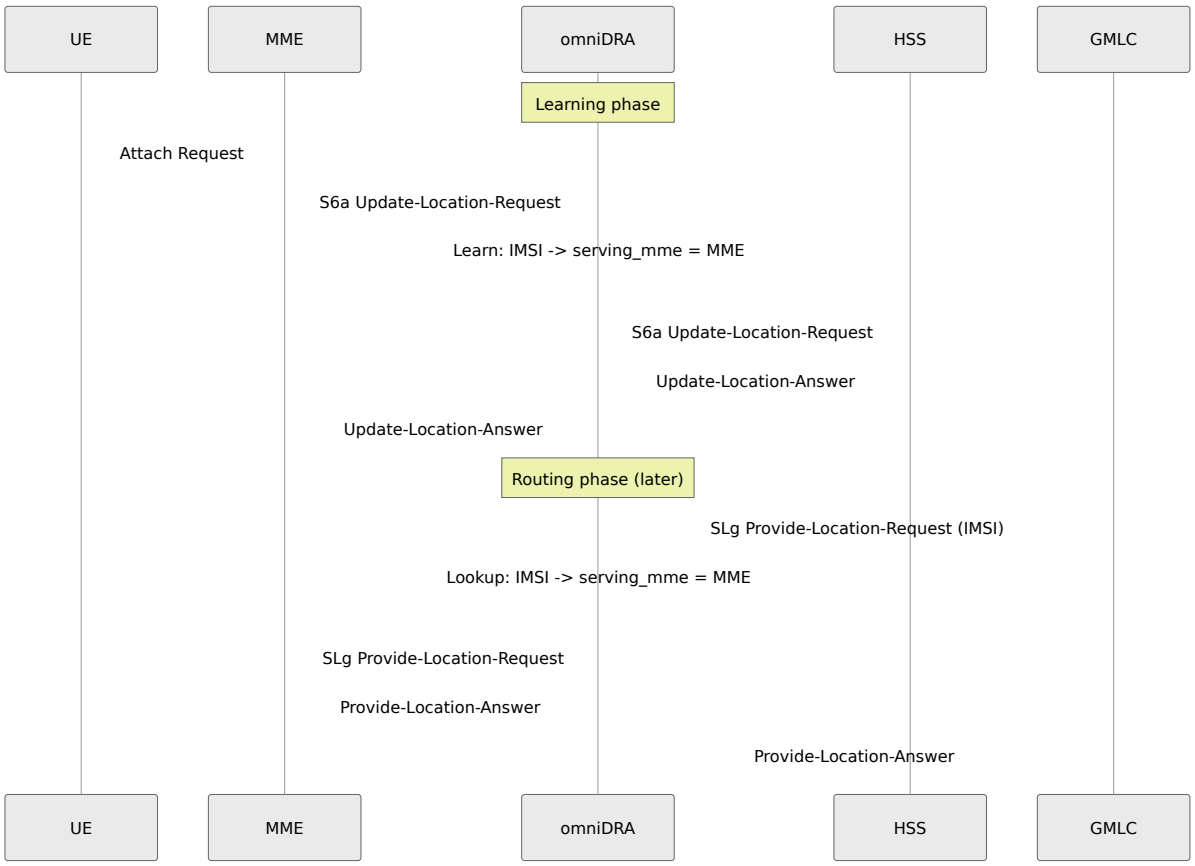
Interface	Message	Command-Code	Binding Removed
S6a	Cancel-Location-Request (CLR)	317	`serving_mme`
S6a	Purge-UE-Request (PUR)	321	`serving_mme`
Gx	Credit-Control-Request Termination (CCR-T)	272 (CC-Request-Type=3)	`serving_pgw`
Cx	Server-Assignment-Request (SAR)	301 (Server-Assignment-Type=4)	`serving_cscf`

When the last binding for a subscriber is removed, the entire subscriber record is deleted from the table.

Routing: Messages That Use Bindings

Interface	Message	Command-Code	Binding Used
SLg	Provide-Location-Request (PLR)	8388620	`serving_mme`
SLh	LCS-Routing-Info-Request (RIR)	8388622	`serving_mme`

Example Flow: Location Service Request



Pipeline Position

Subscriber Lookup runs **before** Advanced Routing in the request processing pipeline. If SLF finds a binding and overrides the route, Advanced Routing still runs but the SLF route takes precedence.

Configuration

The module is configured under `module_subscriber_lookup` in `config/runtime.exs`.

```
module_subscriber_lookup: %{
  # Enable or disable the module
  enabled: false,

  # How long to keep bindings before expiring (seconds)
  binding_ttl_seconds: 86400
}
```

Parameters

Parameter	Type	Required	Default	Description
`enabled`	Boolean	Yes	`false`	Enable or disable the Subscriber Lookup module. When `false`, all requests pass through unmodified and no bindings are learned.
`binding_ttl_seconds`	Integer	No	`86400`	Time in seconds before a subscriber binding expires and is removed during cleanup. Default is 24 hours. Cleanup runs at half the TTL interval, with a minimum of 60 seconds.

Configuration Examples

Standard Deployment

Suitable for most networks where subscribers re-register within 24 hours.

```
module_subscriber_lookup: %{
  enabled: true,
  binding_ttl_seconds: 86400
}
```

How it works: The module learns MME/PGW/CSCF bindings from passing traffic and holds them for 24 hours. SLg and SLh requests are automatically routed to the correct serving MME. Bindings are refreshed whenever a new registration message is observed for the same subscriber.

Use case: Networks with location service (LCS) requirements where the GMLC needs to reach the serving MME without static subscriber-to-MME mappings.

High-Churn Environment

For networks with frequent subscriber mobility where stale bindings should expire quickly.

```
module_subscriber_lookup: %{
  enabled: true,
  binding_ttl_seconds: 3600
}
```

How it works: Bindings expire after 1 hour. This is appropriate when subscribers frequently move between MMEs and stale bindings would cause misdirected location requests. The cleanup interval runs every 30 minutes.

Use case: Dense urban networks or events with high subscriber mobility.

Metrics

Binding Updates

Metric: `diameter.subscriber_lookup.binding.update` **Type**: Counter **Description**: Incremented each time a subscriber binding is created or updated. **Labels**:

- `imsi` - IMSI of the subscriber
- `binding_type` - Type of binding: `serving_mme`, `serving_pgw`, or `serving_cscf`
- `serving_host` - Origin-Host of the serving network element

Binding Deletions

Metric: `diameter.subscriber_lookup.binding.delete` **Type**: Counter **Description**: Incremented each time a subscriber binding is explicitly removed via a deregistration message. **Labels**:

- `imsi` - IMSI of the subscriber
- `binding_type` - Type of binding removed

Routed Requests

Metric: `diameter.subscriber_lookup.route.count` **Type**: Counter **Description**: Incremented each time an SLg/SLh request is successfully routed using a learned binding. **Labels**:

- `imsi` - IMSI of the subscriber
- `binding_type` - Binding type used for routing (currently always `serving_mme`)
- `serving_host` - Origin-Host of the peer the request was routed to

Troubleshooting

Location Requests Not Being Routed to Serving MME

Symptoms: SLg Provide-Location-Requests or SLh LCS-Routing-Info-Requests are not being routed to the expected MME, falling through to default routing instead.

Possible causes:

- The subscriber has not registered through this DRA, so no binding exists
- The binding has expired (TTL exceeded)
- The serving MME peer is not connected to this DRA
- `enabled` is set to `false`

Resolution:

1. Verify the module is enabled in the configuration
2. Check that S6a ULR traffic for the subscriber flows through this DRA (bindings are only learned from observed traffic)
3. Verify `binding_ttl_seconds` is long enough to cover the gap between registration and location request
4. Confirm the serving MME is connected as a peer

Bindings Not Being Learned

Symptoms: The binding table remains empty despite S6a/Gx/Cx traffic passing through the DRA.

Possible causes:

- The module is not enabled
- The module process is not running (check supervisor)
- Messages do not contain a valid IMSI in the expected AVP

Resolution:

1. Verify `enabled: true` in the configuration
2. Confirm the DRA was restarted after the configuration change
3. Check DRA debug logs for `SubscriberLookup: Learned` entries to confirm binding activity

Stale Bindings Causing Misrouted Requests

Symptoms: Location requests are routed to an MME that is no longer serving the subscriber.

Possible causes:

- The Cancel-Location-Request (CLR) or Purge-UE-Request (PUR) did not pass through this DRA
- `binding_ttl_seconds` is set too high for the network's mobility pattern

Resolution:

1. Reduce `binding_ttl_seconds` to match the expected subscriber re-registration interval
2. Ensure all S6a deregistration traffic flows through this DRA

Reference

Application IDs

ID	Interface	Description	Reference
16777251	S6a/S6d	MME/SGSN to HSS authentication and subscription management	3GPP TS 29.272
16777238	Gx	PCEF to PCRF policy and charging control	3GPP TS 29.212
16777216	Cx	I-CSCF/S-CSCF to HSS IMS registration	3GPP TS 29.229
16777255	SLg	GMLC to MME location services	3GPP TS 29.172
16777291	SLh	GMLC to HSS/DRA LCS routing info	3GPP TS 29.173

Command Codes

Code	Name	Interface	Description
272	Credit-Control-Request/Answer (CCR/CCA)	Gx	Session-level policy and charging control
301	Server-Assignment-Request/Answer (SAR/SAA)	Cx	IMS registration and deregistration
316	Update-Location-Request/Answer (ULR/ULA)	S6a	Subscriber location update at MME
317	Cancel-Location-Request/Answer (CLR/CLA)	S6a	HSS-initiated location cancellation
321	Purge-UE-Request/Answer (PUR/PUA)	S6a	MME-initiated UE purge
8388620	Provide-Location-Request/Answer (PLR/PLA)	SLg	Location service request to serving MME
8388622	LCS-Routing-Info-Request/Answer (RIR/RIA)	SLh	LCS routing info lookup

Binding Types

Binding Type	Learned From	Used By	Description
`serving_mme`	S6a ULR	SLg PLR, SLh RIR	The MME currently serving the subscriber
`serving_pgw`	Gx CCR-I	—	The PGW handling the subscriber's session (reserved for future routing)
`serving_cscf`	Cx SAR (Registration)	—	The S-CSCF serving the subscriber's IMS registration (reserved for future routing)