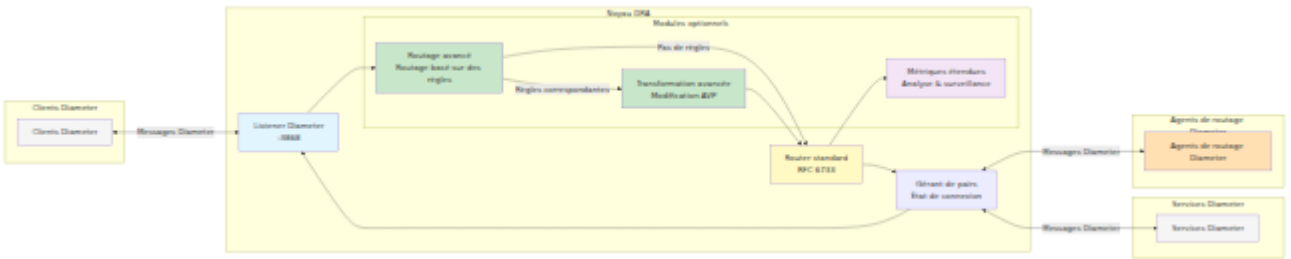
Guide des opérations DRA

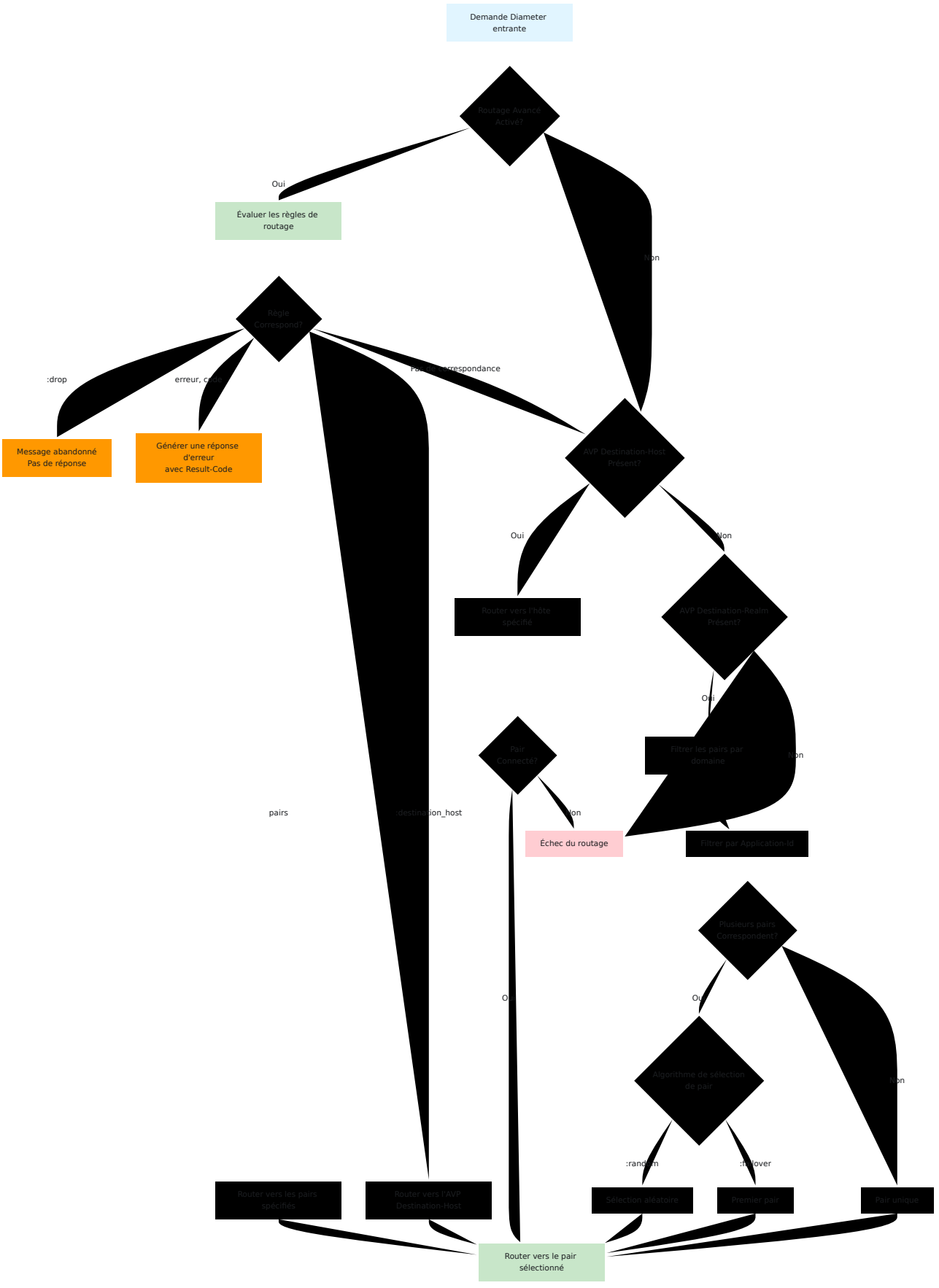
Table des matières

Vue d'ensemble de l'architecture DRA



Routage Diameter standard

Sans les modules Routage avancé ou Transformation avancée, le DRA effectue un routage Diameter standard basé sur le Protocole de base Diameter (RFC 6733):


Demande Diameter entrante
Routage Avancé Activé?
Oui
Évaluer les règles de routage
Règle Correspond?
:drop
Message abandonné Pas de réponse
Générer une réponse d'erreur avec Result-Code
AVP Destination-Host Présent?
Oui
Router vers l'hôte spécifié
AVP Destination-Realm Présent?
Oui
Filtrer les pairs par domaine
Filtrer par Application-Id
Pair Connecté?
Échec du routage
pairs
:destination_host
Plusieurs pairs Correspondent?
Algorithme de sélection de pair
:random
:failover
Sélection aléatoire
Premier pair
Pair unique
Router vers les pairs spécifiés
Router vers l'AVP Destination-Host
Router vers le pair sélectionné

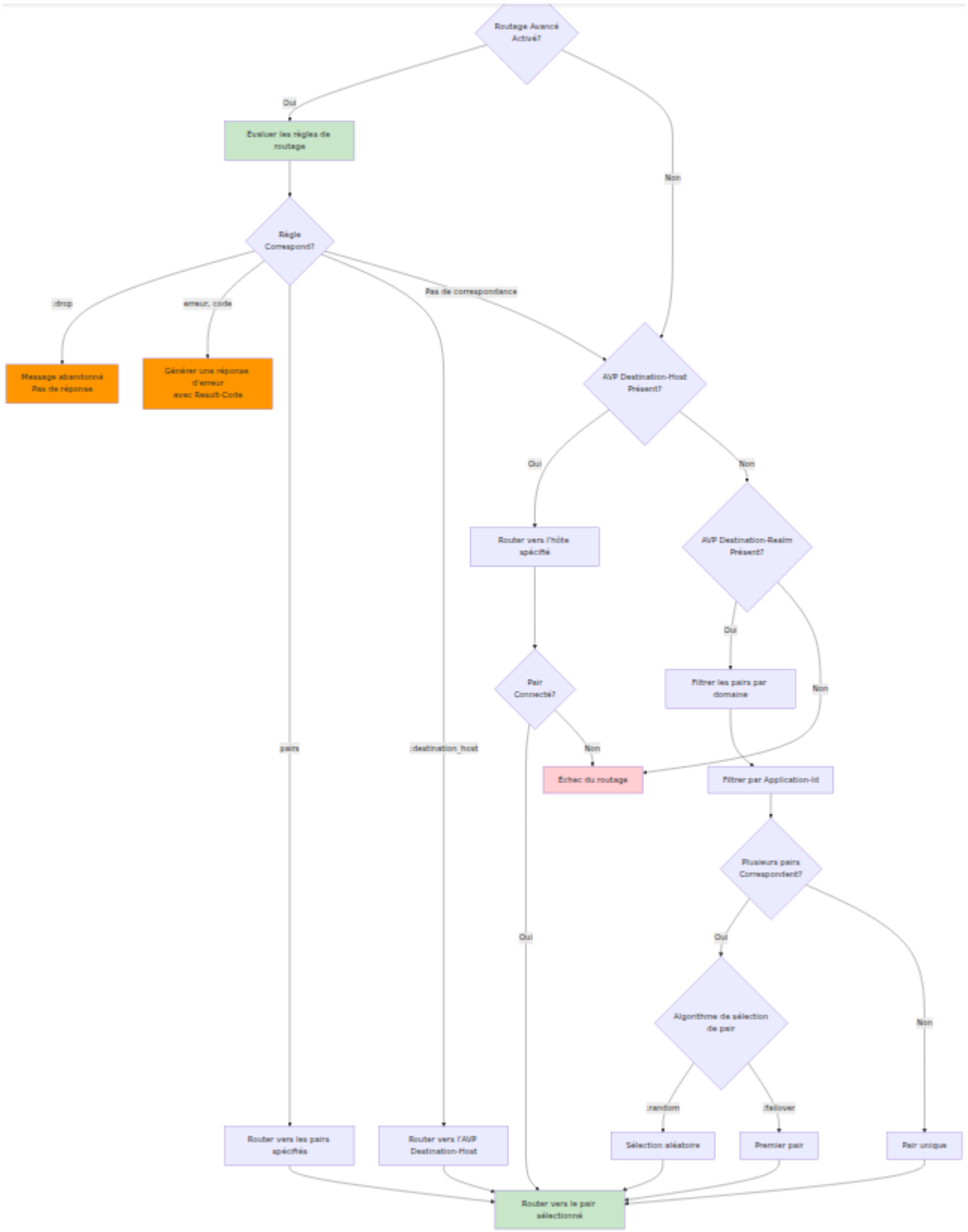
Routage des demandes

Le DRA route les messages de demande en utilisant un mécanisme basé sur la priorité défini dans RFC 6733 Section 6.1:

1. **AVP Destination-Host (293)** - S'il est présent, le DRA route directement vers le pair spécifié
    - C'est le mécanisme de routage de la plus haute priorité
    - Si le pair n'est pas connecté, le routage échoue
    - Fournit un contrôle explicite du routage au niveau de l'hôte
2. **AVP Destination-Realm (283)** - Si Destination-Host est absent, le routage se fait en fonction du domaine
    - Le DRA sélectionne un pair connecté qui annonce le support pour le domaine cible
    - L'équilibrage de charge est appliqué lorsque plusieurs pairs correspondent au domaine
    - Le routage basé sur le domaine permet une flexibilité entre plusieurs hôtes
3. **Application-Id** - Les pairs sont filtrés par les applications Diameter prises en charge
    - Seuls les pairs annonçant le support pour l'Application-Id du message sont pris en compte
    - Basé sur l'échange de capacités (CER/CEA) lors de l'établissement de la connexion du pair
    - Voir Identifiants d'application 3GPP courants pour référence

Routage des réponses

Les paquets de réponse utilisent un mécanisme de routage fondamentalement différent de celui des demandes :



- **Routage basé sur la session**: Les paquets de réponse suivent toujours le chemin inverse de la demande

- **Préservation de l'ID de bout en bout**: L'identifiant de bout en bout reste inchangé à travers tous les sauts
- **Routage par saut**: Le DRA utilise l'identifiant de saut pour maintenir l'état de routage (change à chaque saut)
- **Aucune évaluation de règle**: Le DRA n'évalue pas les règles de routage ou le contenu des AVP pour les réponses
- **Corrélation avec état**: Les tables de routage internes suivent quel pair a envoyé chaque demande

Pourquoi les réponses ne sont pas routées par des modules avancés :

- Le routage des réponses est déterministe et doit revenir au pair d'origine
- Le protocole Diameter exige que les réponses suivent le chemin de demande établi
- Les décisions de routage pour les réponses sont prises en fonction du contexte de la demande originale, pas du contenu de la réponse
- Cela garantit une gestion correcte des sessions et empêche les boucles de routage

Voir RFC 6733 Section 6.2 pour des détails sur le routage des messages de réponse.

Sélection de pairs

Lorsque plusieurs pairs correspondent aux critères de routage, l'algorithme de `peer_selection_algorithm` configuré détermine la sélection :

- `:random` - Sélectionne aléatoirement parmi les pairs disponibles (par défaut)
- `:failover` - Sélectionne toujours le premier pair de la liste (basé sur la priorité)
- Les pairs doivent être en **état connecté** pour être sélectionnés
- Les pairs déconnectés ou hors service sont automatiquement exclus

Limitations du routage standard

- Pas de règles de routage personnalisées basées sur les valeurs AVP (par exemple, motifs IMSI)
- Pas de traduction de domaine ou de modification d'AVP
- Impossible de router en fonction du pair d'origine
- Contrôle limité sur la distribution du trafic

Les modules Routage avancé et Transformation avancée étendent ce comportement standard avec des capacités de routage basées sur des règles et de manipulation de paquets.

Configuration de base du DRA

Le DRA nécessite une configuration de base définissant son identité, ses paramètres réseau et ses connexions de pair. Cette configuration établit la base de toutes les opérations de routage.

Structure de configuration

```
%{
  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: [
    # Configurations des pairs...
  ]
}
```

Paramètres d'identité du DRA

Paramètre	Type	Description
`host`	Chaîne	L'Identité Diameter du DRA (nom de domaine entièrement qualifié)
`realm`	Chaîne	Le domaine Diameter du DRA
`product_name`	Chaîne	Nom du produit annoncé dans les messages CER/CEA
`vendor_id`	Entier	Vendor-ID tel que défini dans RFC 6733 Section 5.3.3 (10415 = 3GPP)

Paramètres réseau

Paramètre	Type	Description
`listen_ip`	Chaîne ou Liste	Adresse(s) IP sur lesquelles le DRA écoute. Pour le multihoming SCTP, utilisez une liste de chaînes IP (voir Multihoming SCTP)
`listen_port`	Entier	Port TCP/SCTP pour les connexions Diameter (standard : 3868)
`service_name`	Atome	Identifiant de service interne Erlang
`request_timeout`	Entier	Délai d'attente en millisecondes pour les paires demande/réponse (par défaut : 5000)

Paramètres de sélection de pairs

Paramètre	Type	Description
`peer_selection_algorithm`	Atome	Algorithme d'équilibrage de charge : `:random` (sélection aléatoire) ou `:failover` (priorité au premier pair)
`allow_undefined_peers_to_connect`	Booléen	Autoriser les connexions de pairs non définis dans la configuration (par défaut : `false`)
`log_unauthorized_peer_connection_attempts`	Booléen	Journaliser les tentatives de connexion de pairs non autorisés

Configuration des pairs

Chaque pair dans la liste `peers` définit une connexion Diameter :

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

Paramètres des pairs

Paramètre	Type	Description
`host`	Chaîne	L'Identité Diameter du pair (FQDN) - doit correspondre exactement pour le routage
`realm`	Chaîne	Domaine Diameter du pair
`ip`	Chaîne	Adresse IP principale du pair pour la connexion (requise)
`ips`	Liste	Liste d'adresses IP pour le multihoming SCTP (optionnel, voir Multihoming SCTP)
`port`	Entier	Port Diameter du pair (typiquement 3868)
`transport`	Atome	Protocole de transport : `:diameter_tcp` ou `:diameter_sctp`
`tls`	Booléen	Activer le chiffrement TLS (si `true`, utiliser généralement le port 3869)
`initiate_connection`	Booléen	`true` : le DRA se connecte au pair, `false` : le DRA attend que le pair se connecte

Modes de connexion

Initier la connexion (`initiate_connection: true`)

- Le DRA agit en tant que client Diameter
- Le DRA initie la connexion TCP/SCTP au pair
- Utilisé pour se connecter à HSS, PCRF ou d'autres systèmes backend
- Le DRA réessaiera les connexions si le pair est injoignable

Accepter la connexion (`initiate_connection: false`)

- Le DRA agit en tant que serveur Diameter
- Le DRA attend que le pair se connecte
- Utilisé pour les connexions MME, SGSN, P-GW
- Le pair doit être dans la configuration ou `allow_undefined_peers_to_connect: true`

Exemple de configuration

```
%{
  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 - attend que le MME se connecte
    %{
      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 - le DRA initie la connexion
    %{
      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 avec TLS - le DRA initie une connexion sécurisée
    %{
      host: "pcrf01.mvno.example.com",
      realm: "mvno.example.com",
      ip: "10.100.3.22",
      port: 3869,
      transport: :diameter_tcp,
      tls: true,
```

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

Notes importantes

- **Correspondance des noms d'hôtes** : Les noms d'hôtes des pairs dans les règles de Routage avancé doivent correspondre exactement à la valeur `host` configurée ici (sensible à la casse)
- **Échange de capacités** : Lors de la connexion, les pairs échangent les applications prises en charge via des messages CER/CEA
- **Support des applications** : Le DRA annonce toutes les applications 3GPP prises en charge (voir Identifiants d'application 3GPP courants)
- **Vendor-ID 10415** : Valeur standard pour les applications 3GPP
- **Délai d'attente des demandes** : Affecte le TTL des Métriques étendues (délai d'attente + 5 secondes)
- **Sélection de pairs** : Lorsque plusieurs pairs correspondent aux critères de routage, `peer_selection_algorithm` détermine lequel est choisi

Considérations de sécurité

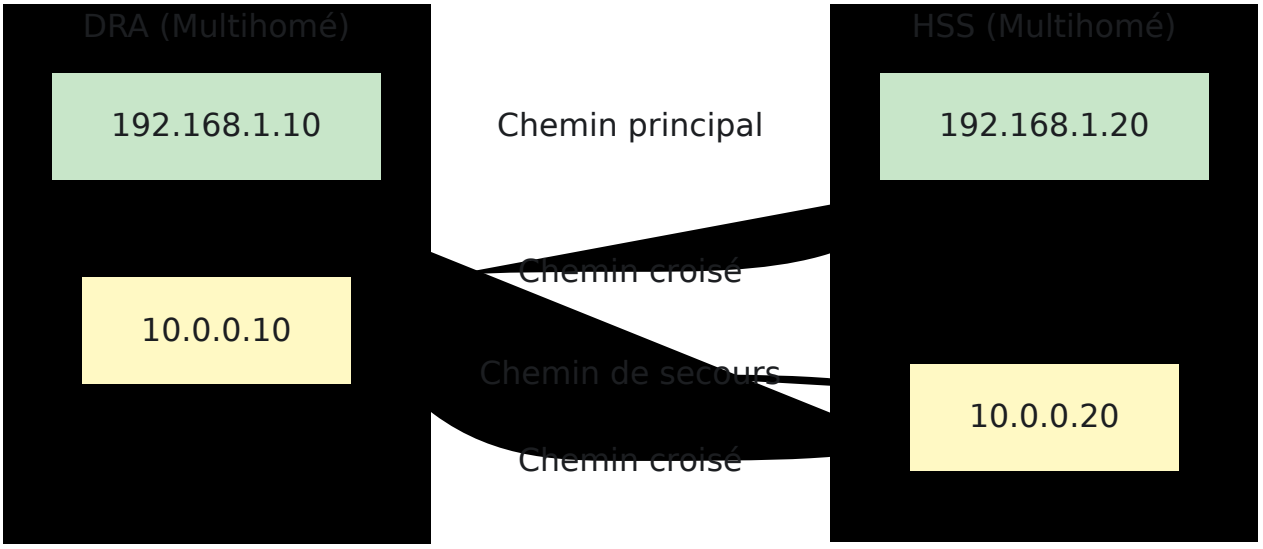
- Définir `allow_undefined_peers_to_connect: false` en production
- Activer `log_unauthorized_peer_connection_attempts: true` pour la surveillance de la sécurité
- S'assurer que les règles de pare-feu correspondent aux paramètres `listen_ip` et `listen_port`
- Valider les certificats des pairs lors de l'utilisation de TLS

Multihoming SCTP

Le multihoming SCTP fournit une redondance réseau en permettant aux points de terminaison de se lier à plusieurs adresses IP. Si le chemin réseau principal

échoue, le SCTP bascule automatiquement vers un chemin alternatif sans perturber la session Diameter.

Comment ça fonctionne



- Les cœurs SCTP surveillent tous les chemins réseau
- La bascule automatique se produit si le chemin principal devient injoignable
- Aucune perturbation de la session Diameter pendant le changement de chemin
- Le noyau gère automatiquement la sélection de chemin

Configuration

Adresses d'écoute DRA

Configurez plusieurs adresses IP locales pour que le DRA se lie à :

```
%{
  # IP unique (compatible avec les versions antérieures)
  listen_ip: "192.168.1.10",

  # Plusieurs IP pour le multihoming SCTP
  listen_ip: ["192.168.1.10", "10.0.0.10"],

  listen_port: 3868,
  ...
}
```

Remarques :

- Le transport TCP utilise uniquement la première IP de la liste
- Le transport SCTP se lie à toutes les IP spécifiées
- Le format de chaîne IP unique reste entièrement pris en charge

Configuration des pairs

Configurez plusieurs adresses IP distantes pour les connexions de pairs :

```
peers: [
  %{
    host: "hss01.example.com",
    realm: "example.com",
    ip: "192.168.1.20",                         # IP principale
(requise)
    additional_ips: ["192.168.1.20", "10.0.0.20"],        # Toutes
les IP pour le multihoming
    port: 3868,
    transport: :diameter_sctp,
    tls: false,
    initiate_connection: true
  }
]
```

Remarques :

- Le champ `ip` est requis pour la compatibilité avec les versions antérieures

- Le champ `ips` est optionnel ; s'il est omis, seul `ip` est utilisé
- Pour SCTP, incluez l'IP principale dans la liste `ips`
- Pour TCP, seul `ip` est utilisé (TCP ne prend pas en charge le multihoming)

Exemple complet

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: ["192.168.1.10", "10.0.0.10"],  # DRA multihomé
    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: [
      # Connexion HSS multihomée
      %{
        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 à domicile unique (compatible avec les versions
antérieures)
      %{
        host: "mme01.example.com",
        realm: "example.com",
        ip: "192.168.1.30",
        port: 3868,
        transport: :diameter_sctp,
        tls: false,
        initiate_connection: false
      }
    ]
  }
```

Exigences

- Le module noyau SCTP doit être chargé (package `lksctp-tools` sur Linux)
- Toutes les adresses IP doivent être routables depuis/vers le pair
- Les règles de pare-feu doivent autoriser le trafic SCTP sur toutes les IP configurées
- Les deux points de terminaison doivent être configurés pour le multihoming pour une redondance complète

Limitations

- Le transport TCP ne prend pas en charge le multihoming (n'utilise que l'IP principale)
- Le TLS sur le multihoming SCTP peut avoir des limitations de compatibilité
- Le timing de basculement de chemin dépend des paramètres SCTP du noyau

---

Tables de référence

Identifiants d'application 3GPP courants

Application-Id	Interface	Description
16777251	S6a/S6d	Authentification MME/SGSN et données d'abonnement vers HSS
16777252	S13/S13'	Vérification de l'identité de l'équipement MME vers EIR
16777238	Gx	Contrôle de la politique et de la facturation PCEF vers PCRF
16777267	S9	Politique de roaming PCRF domicile vers PCRF visité
16777272	Sy	Liaison de session PCRF vers OCS
16777216	Cx	Enregistrement IMS I-CSCF/S-CSCF vers HSS
16777217	Sh	Données utilisateur IMS AS vers HSS
16777236	SLg	Services de localisation MME/SGSN vers GMLC
16777291	SLh	Informations sur l'abonné de localisation GMLC vers HSS
16777302	S6m	MTC-IWF vers HSS/HLR pour dispositifs M2M
16777308	S6c	Routage SMS HSS vers SMS-SC/IP-SM-GW
16777343	S6t	Événements de surveillance SCEF vers HSS

Application-Id	Interface	Description
16777334	Rx	Autorisation média AF vers PCRF

Codes AVP courants

Code	Nom AVP	Type	Utilisation
1	User-Name	UTF8String	Identifiant de l'abonné (IMSI dans 3GPP)
264	Origin-Host	DiameterIdentity	Nom d'hôte du pair d'origine
268	Result-Code	Unsigned32	Code de résultat standard
283	Destination-Realm	DiameterIdentity	Domaine cible
293	Destination-Host	DiameterIdentity	Hôte cible (optionnel)
296	Origin-Realm	DiameterIdentity	Domaine source
297	Experimental-Result	Grouped	Code de résultat spécifique au fournisseur

Codes de commande courants

Les codes de commande font partie de l'en-tête du message Diameter, pas des AVP :

Code	Nom de commande	Description
257	CER/CEA	Demande/Réponse d'échange de capacités
258	RAR/RAA	Demande/Réponse de ré-authentification
274	ASR/ASA	Demande/Réponse d'annulation de session
275	STR/STA	Demande/Réponse de terminaison de session
280	DWR/DWA	Demande/Réponse de surveillance de dispositif
282	DPR/DPA	Demande/Réponse de déconnexion de pair
316	ULR/ULA	Demande/Réponse de mise à jour de localisation (S6a)
317	CLR/CLA	Demande/Réponse d'annulation de localisation (S6a)
318	AIR/AIA	Demande/Réponse d'information d'authentification (S6a)
321	PUR/PUA	Demande/Réponse de purge d'UE (S6a)

Module de routage avancé

Le module de routage avancé fournit des capacités de routage de message flexibles basées sur des règles avec support pour des conditions de correspondance complexes.

Important : Ce module évalue **uniquement les paquets de demande Diameter entrants** (pas les paquets de réponse). Les paquets de réponse

suivent le routage de session établi vers le pair d'origine - voir Routage des réponses pour plus de détails.

Configuration

Activez le module et définissez les règles de routage dans votre configuration :

```
dra_module_advanced_routing:
  enabled: True
  rules:
    - rule_name: <identifiant_de_règle>
      match: <portée_de_correspondance>
      filters: [<liste_de_filtres>]
      route:
        peers: [<liste_de_pairs>]
```

Paramètres

Paramètre	Description
`enabled`	Définir sur `True` pour activer le module
`rule_name`	Identifiant unique pour la règle de routage
`match`	Comment les filtres sont combinés : `:all` (logique ET - tous les filtres doivent correspondre), `:any` (logique OU - au moins un filtre doit correspondre), `:none` (logique NOR - aucun filtre ne peut correspondre)
`filters`	Liste des conditions de filtre (voir Filtres disponibles)
`route`	Action de routage (voir Actions de routage ci-dessous)

Actions de routage

Le paramètre `route` prend en charge plusieurs actions :

Router vers des pairs

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

Routage vers les noms d'hôtes de pairs spécifiés. Les pairs doivent être :

- Définis dans la configuration des pairs Diameter du DRA
- Le nom d'hôte exact tel que configuré (sensible à la casse)
- Actuellement connectés pour que le routage réussisse (les pairs déconnectés sont ignorés)

Router vers l'AVP Destination-Host

```
route: :destination_host
```

Routage vers le pair spécifié dans l'AVP Destination-Host (293). Si l'AVP Destination-Host est manquant, le routage revient au comportement normal.

Abandonner le trafic

```
route: :drop
```

Élimine silencieusement le message sans envoyer de réponse. Utilisé pour :

- Filtrage de trafic et blackholing
- Blocage de demandes non désirées
- Limitation de débit en abandonnant le trafic excessif

Comportement :

- Le message est abandonné au DRA (non transmis)
- Aucun message de réponse n'est envoyé au pair demandeur
- Implemente le comportement `:discard` de Diameter Erlang
- Métrique : `diameter_advanced_routing_drop_count_total` (voir Métriques Prometheus)

Générer une réponse d'erreur

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

Génère une réponse d'erreur Diameter avec le Result-Code spécifié et l'envoie au pair demandeur. Codes de résultat courants :

- `3002` - DIAMETER_UNABLE_TO_DELIVER (routage indisponible)
- `3003` - DIAMETER_REALM_NOT_SERVED (domaine non pris en charge)
- `3004` - DIAMETER_TOO_BUSY (protection contre la surcharge, limitation de débit)
- `5012` - DIAMETER_UNABLE_TO_COMPLY (rejet général)

Comportement :

- Le DRA génère une réponse d'erreur avec le Result-Code spécifié
- La réponse inclut Origin-Host, Origin-Realm, Session-Id (auto-rempli par Diameter)
- Le message n'est PAS transmis à aucun pair
- Implemente `{:protocol_error, code}` de Diameter Erlang (équivalent à `{:answer_message, code}`)
- Métrique : `diameter_advanced_routing_error_count_total` (voir Métriques Prometheus)

Filtres disponibles

Filtres standard

Disponibles à la fois dans Routage avancé et Transformation avancée :

- **`:application_id`** - Correspondre à l'identifiant d'application Diameter (voir référence d'ID d'application)
  - Valeur unique : `{:application_id, 16777251}` (S6a/S6d)
  - Valeurs multiples : `{:application_id, [16777251, 16777252]}` (S6a ou S6b)

- `:command_code` - Correspondre au code de commande Diameter
  - Valeur unique : `{:command_code, 318}` (demande AIR)
  - Valeurs multiples : `{:command_code, [317, 318]}` (ULR ou AIR)
- `:avp` - Correspondre à la valeur AVP (voir référence de code AVP)
  - Correspondance exacte : `{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}`
  - Correspondance regex : `{:avp, {1, ~r"999001.*"}}`
  - Plusieurs motifs : `{:avp, {1, ["505057001313606", ~r"999001.*", ~r"505057.*"]}}`
  - Toute valeur (vérification de présence) : `{:avp, {264, :any}}`

Filtre spécifique au routage

Uniquement disponible dans Routage avancé :

- `:via_peer` - Correspond au pair d'où la demande a été reçue
  - Pair unique : `{:via_peer, "omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org"}`
  - Plusieurs pairs : `{:via_peer, ["omnitouch-lab-dra01.epc.mnc001.mcc001.3gppnetwork.org", "omnitouch-lab-dra02.epc.mnc001.mcc001.3gppnetwork.org"]}`
  - Tout pair : `{:via_peer, :any}`

Filtres spécifiques à la transformation

Uniquement disponibles dans Transformation avancée :

- `:to_peer` - Correspond à un pair de destination prédéterminé (uniquement pour les paquets de demande)
  - Pair unique : `{:to_peer, "dra01.omnitouch.com.au"}`
  - Plusieurs pairs : `{:to_peer, ["dra01.omnitouch.com.au", "dra02.omnitouch.com.au"]}`
- `:from_peer` - Correspond au pair qui a envoyé la réponse (uniquement pour les paquets de réponse)

    - Pair unique : `{:from_peer, "hss-01.example.com"}`
    - Plusieurs pairs : `{:from_peer, ["hss-01.example.com", "hss-02.example.com"]}`
- `:packet_type` - Correspond à la direction du paquet
    - Demande : `{:packet_type, :request}`
    - Réponse : `{:packet_type, :answer}`

Remarques importantes sur les filtres

- **Filtres AVP** : Recommandés uniquement pour les AVP simples (User-Name, Origin-Host, Destination-Realm, etc.)
    - Les AVP groupés ne sont **pas pris en charge** et ne correspondront pas
    - Les valeurs binaires complexes ne sont **pas prises en charge**
    - Utilisez le format : `{:avp, {code, value}}`
- **Opérateurs de liste** : Pris en charge pour toutes les valeurs de filtre sauf `:packet_type`
    - Lorsqu'une liste est utilisée, elle applique une logique **OU** à l'intérieur de la liste
    - Exemple : `{:command_code, [317, 318]}` correspond au code de commande 317 **OU** 318
- **Valeurs spéciales** :
    - `:any` - Correspond à toute valeur (vérifie la présence de l'AVP)
    - Exemple : `{:avp, {264, :any}}` correspond si l'AVP Origin-Host existe avec n'importe quelle valeur

Exemples de routage

Exemple 1 : Routage via le pair

Routez les messages en fonction du DRA d'où ils sont arrivés :

```
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]
```

Comment ça fonctionne : Route le trafic S6a qui arrive via des pairs DRA spécifiques vers des nœuds HSS locaux.

Exemple 2 : Roaming entrant avec correspondance de motifs

Routez le trafic de roaming en fonction des motifs 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]
```

Comment ça fonctionne : Route les messages S6a d'un domaine d'origine spécifique avec des motifs IMSI correspondants vers des pairs DRA désignés.

Exemple 3 : Routage dynamique avec :destination_host

Routez vers la valeur de l'AVP Destination-Host dans le message :

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

Comment ça fonctionne :

- Lorsque les filtres correspondent, route vers le pair spécifié dans l'AVP Destination-Host (293)
- Si l'AVP Destination-Host est manquant, la correspondance est considérée comme un échec et revient au routage normal
- Utile pour honorer le routage lorsque l'expéditeur spécifie la destination exacte

Exemple 4 : Abandonner le trafic indésirable

Abandonnez le trafic de plages IMSI spécifiques :

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

Comment ça fonctionne :

- Correspond aux messages S6a avec un IMSI commençant par 999999
- Abandonne silencieusement le message sans envoyer de réponse

- Utile pour filtrer le trafic de test ou bloquer des plages d'abonnés spécifiques
- Voir Métriques Prometheus pour surveiller le trafic abandonné

Exemple 5 : Limitation de débit avec des réponses d'erreur

Retournez DIAMETER_TOO_BUSY pour des motifs de trafic spécifiques :

```
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}
```

Comment ça fonctionne :

- Correspond au trafic S6a d'un pair surchargé spécifique
- Retourne une réponse d'erreur DIAMETER_TOO_BUSY (3004)
- Le pair demandeur reçoit une erreur et doit se retirer
- Utile pour la protection contre la surcharge et la limitation de débit
- Voir Métriques Prometheus pour surveiller les réponses d'erreur

Exemple 6 : Réponses d'erreur conditionnelles par commande

Bloquez des types de commandes spécifiques avec des codes d'erreur appropriés :

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

Comment ça fonctionne :

- Correspond aux messages de demande de purge S6a
- Retourne une erreur DIAMETER_UNABLE_TO_COMPLY (5012)
- Bloque certaines opérations sans abandonner silencieusement le trafic
- Utile pour désactiver sélectivement certaines commandes Diameter

---

Module de transformation avancée

Le module de transformation avancée permet la modification dynamique des AVP de message Diameter en fonction de critères de correspondance. Voir Traitement des règles pour des détails sur la façon dont les règles sont évaluées.

Configuration

Activez le module et définissez les règles de transformation :

```
dra_module_advanced_transform:
  enabled: True
  rules:
    - rule_name: <identifiant_de_règle>
      match: <portée_de_correspondance>
      filters: [<liste_de_filtres>]
      transform:
        action: <action_de_transformation>
        avps: [<modifications_avp>]
```

Paramètres

Paramètre	Description
`enabled`	Définir sur `True` pour activer le module
`rule_name`	Identifiant unique pour la règle de transformation
`match`	Comment les filtres sont combinés : `:all` (logique ET), `:any` (logique OU), `:none` (logique NOR) - voir Logique de filtre
`filters`	Liste des conditions de filtre (voir Filtres disponibles)
`transform.action`	Type de transformation (`:edit`, `:remove` ou `:overwrite`)
`transform.avps`	Liste des modifications d'AVP à appliquer (voir référence de code AVP)

Actions de transformation

Paquets de demande (Demandes Diameter)

- **`:edit`** - Modifier les valeurs AVP existantes
  - Modifie uniquement les AVP qui existent dans le message

    - Si l'AVP n'existe pas, aucun changement n'est effectué
- `:remove` - Supprimer les AVP du message
- `:overwrite` - Remplacer des structures AVP entières
    - Nécessite le paramètre `dictionary` spécifiant le dictionnaire Diameter (par exemple, `:diameter_gen_3gpp_s6a`)

Paquets de réponse (Réponses Diameter)

- `:remove` - Supprimer les AVP du message
- `:overwrite` - Remplacer des structures AVP entières
    - Nécessite le paramètre `dictionary`

Important : Si aucune règle ne correspond, le paquet est transmis de manière transparente sans aucune transformation.

Syntaxe de modification AVP

Modification standard :

- `{:avp, {<code>, <new_value>}}` - Définir l'AVP à une nouvelle valeur

Suppression d'AVP :

- `{:avp, {<code>, :any}}` - Supprimer l'AVP par ID (supprime indépendamment de la valeur actuelle)
- Remarque : La suppression basée sur avp_id est prise en charge ; la suppression basée sur le contenu de l'AVP n'est pas prise en charge

Écraser avec dictionnaire :

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

Exemples de transformation

Exemple 1 : Réécriture de domaine de destination basée sur le pair

Réécrire le domaine de destination en fonction de l'endroit où le message est routé :

```
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"}}'
```

Comment ça fonctionne : Lorsque les demandes S6a sont routées vers des pairs DRA spécifiques et correspondent au motif IMSI, réécrit le domaine de destination pour le réseau d'Operator X.

Exemple 2 : Routage de plusieurs opérateurs avec transformations

```
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"}}'
```

Comment ça fonctionne : Route différentes plages d'abonnés IMSI vers les domaines de réseau appropriés en fonction des motifs IMSI. La première règle correspondante l'emporte (voir Ordre d'exécution).

Exemple 3 : Ré��criture de domaine pour 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"]}}'  # Correspondance
exacte de l'IMSI
      transform:
        action: ":edit"
        avps:
          - '{:avp, {283, "epc.mnc001.mcc001.3gppnetwork.org"}}'
```

Comment ça fonctionne : Transforme le domaine de destination pour un abonné MVNO spécifique vers leur réseau central hébergé.

Exemple 4 : Transformation uniquement pour les demandes avec filtre de type de paquet

Transformez uniquement les paquets de demande (pas les réponses) :

```
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 doit exister avec
n'importe quelle valeur
      transform:
        action: ":edit"
        avps:
          - '{:avp, {1, "999999000000002"}}'
```

Comment ça fonctionne :

- Correspond uniquement aux paquets **demande** S6a (pas aux paquets de réponse)
- Vérifie que User-Name (AVP 1) est égal à "999999000000001"
- Vérifie que Origin-Host (AVP 264) existe avec n'importe quelle valeur
- Réécrit User-Name en "999999000000002"
- Si l'AVP n'existe pas, aucun changement n'est effectué

Exemple 5 : Supprimer un AVP

Supprimer un AVP spécifique des 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}}'  # Supprimer User-Name
indépendamment de la valeur
```

Comment ça fonctionne : Supprime l'AVP User-Name (code 1) de tous les messages S6a, indépendamment de sa valeur actuelle.

Exemple 6 : Écraser un AVP groupé sur les paquets de réponse

Modifier des AVP groupés complexes dans les paquets de réponse en utilisant l'action `:overwrite` avec le support du dictionnaire :

```
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 (Réponse de
mise à jour de localisation)
        - '{:packet_type, :answer}'           # Paquets de réponse
uniquement
        - '{:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}'  #
Domaine d'origine
      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, [], [],
```

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

Comment ça fonctionne :

- Correspond aux paquets S6a de réponse de mise à jour de localisation (ULA) d'un domaine d'origine spécifique
- Utilise l'action `:overwrite` pour remplacer l'ensemble de l'AVP groupé APN-Configuration-Profile
- **Nécessite le paramètre `dictionary`** pour encoder correctement les structures AVP groupées complexes
- Ajoute trois configurations APN : "internet" (contexte 1), "ims" (contexte 2) et "sos" (contexte 3)
- Chaque APN inclut des profils QoS, des limites de bande passante (AMBR) et des paramètres de type PDN
- La transformation garantit que les services d'urgence (SOS) APN sont provisionnés pour tous les abonnés de ce domaine

Quand utiliser `:overwrite` avec dictionnaire :

- Modifier des AVP groupés avec des structures imbriquées (comme APN-Configuration-Profile)
- Ajouter ou restructurer des données d'abonnement complexes 3GPP
- Lorsque l'action `:edit` ne peut pas gérer la complexité de l'AVP
- Le dictionnaire doit correspondre à l'application Diameter (`:diameter_gen_3gpp_s6a` pour S6a, etc.)

Remarques importantes :

- `:overwrite` remplace l'ensemble de l'AVP, pas seulement des champs individuels
- La structure de l'AVP doit correspondre exactement à la définition du dictionnaire

- Une structure incorrecte entraînera des échecs d'encodage et des paquets abandonnés
- Il s'agit d'une fonctionnalité avancée - validez soigneusement dans un environnement de test d'abord

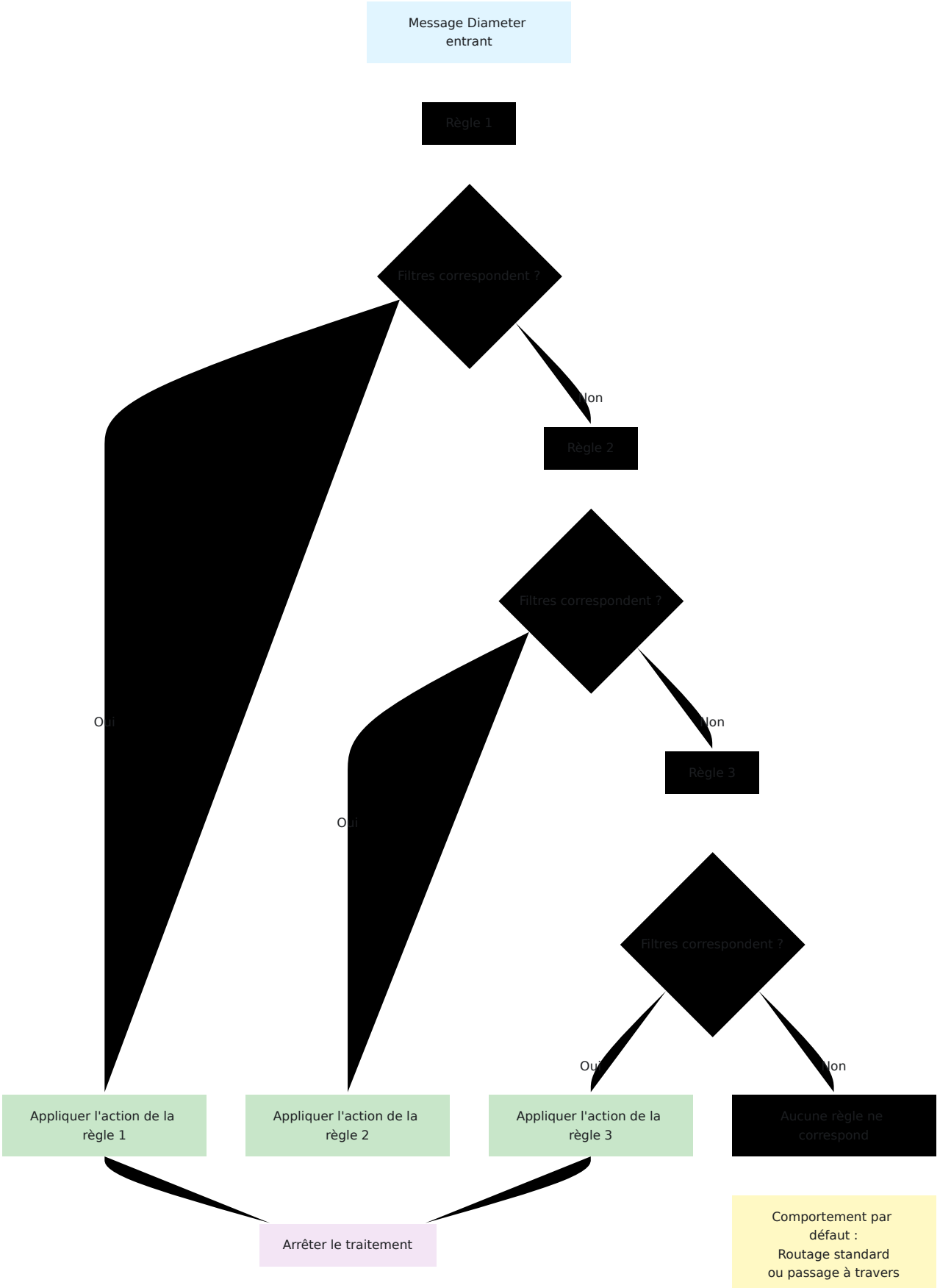
Cas d'utilisation

- **Support MVNO** : Router le trafic des opérateurs virtuels vers des réseaux centraux hébergés
- **Migration de réseau** : Rediriger progressivement les abonnés vers une nouvelle infrastructure
- **Traduction de domaine** : Convertir entre différents schémas de nommage pour les partenaires de roaming
- **Multi-location** : Isoler les populations d'abonnés par domaine
- **Routage des transporteurs** : Diriger le trafic vers les réseaux de transporteurs corrects en fonction des plages IMSI

Traitement des règles

S'applique aux modules Routage avancé et Transformation avancée.

Ordre d'exécution



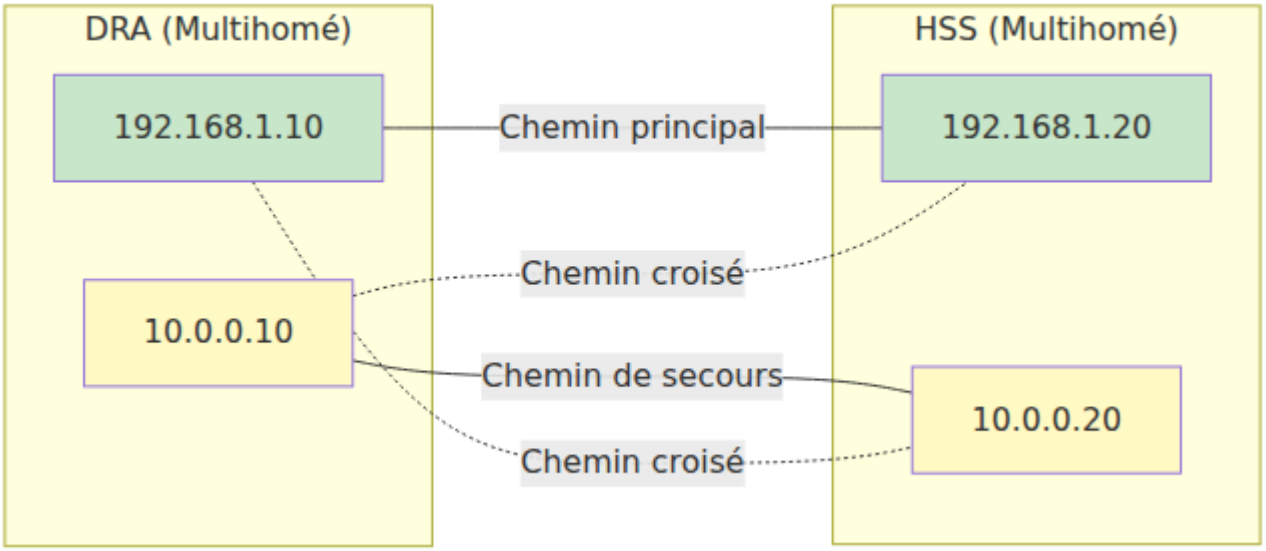
1. Les règles sont évaluées **dans l'ordre de haut en bas** tel que défini dans la configuration
2. Les filtres au sein d'une règle sont évalués en fonction du paramètre `match` (`:all`, `:any` ou `:none`)
3. **La première règle correspondante l'emporte** - les règles suivantes ne sont pas évaluées
4. Si aucune règle ne correspond, le comportement de routage/passage à travers par défaut est utilisé

Logique de filtre

Le paramètre `match` détermine comment les filtres sont combinés :

match: :all (Logique ET)

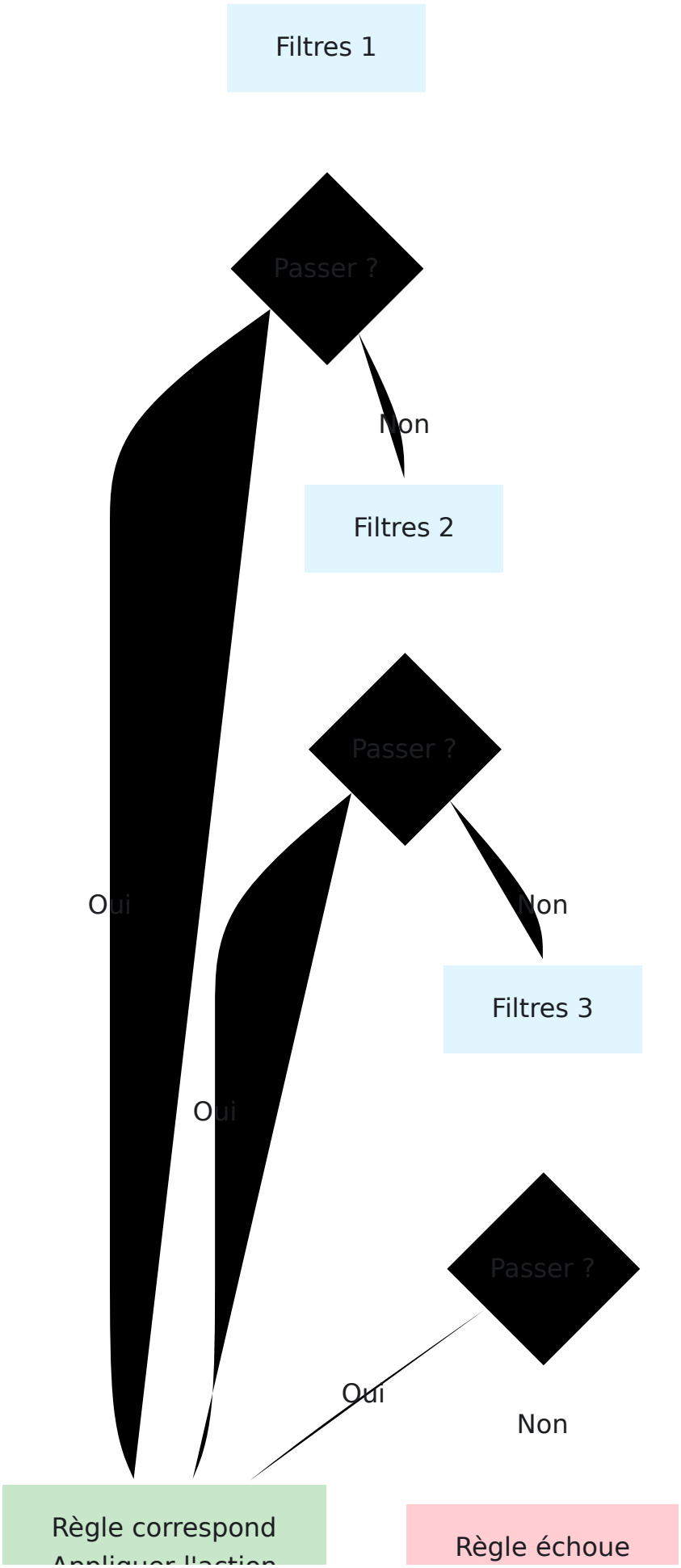
Tous les filtres doivent correspondre pour que la règle réussisse.



Exemple : Avec 3 filtres, `filter1 ET filter2 ET filter3` doivent tous être vrais.

match: :any (Logique OU)

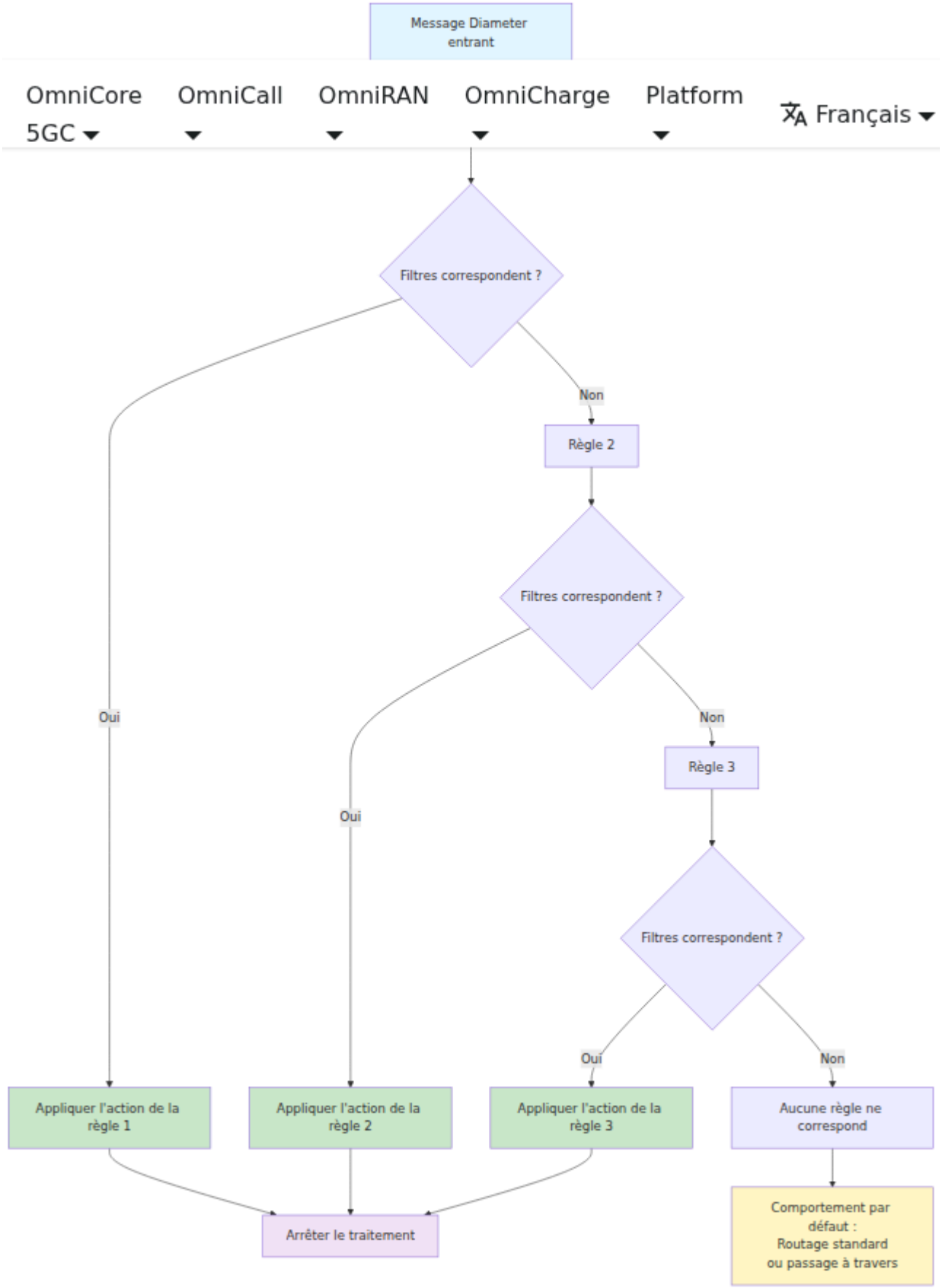
Au moins un filtre doit correspondre pour que la règle réussisse.


Filtres 1
Passer ?
Non
Filtres 2
Passer ?
Oui
Non
Filtres 3
Oui
Passer ?
Oui
Non
Règle correspond
Règle échoue

Exemple : Avec 3 filtres, `filter1 OU filter2 OU filter3` (au moins un passe).

match: :none (Logique NOR)

Aucun filtre ne peut correspondre pour que la règle réussisse (correspondance inverse).



Exemple : Avec 3 filtres, `PAS filter1 ET PAS filter2 ET PAS filter3` (tous doivent échouer).

Remarques supplémentaires :

Lors de l'utilisation d'opérateurs de liste au sein d'une valeur de filtre (par exemple, `{:avp, {1, ["value1", "value2"]}}`), les valeurs utilisent la logique **OU** (n'importe quelle peut correspondre).

Modèles d'expressions régulières

Utilisez la syntaxe `~r"pattern"` pour la correspondance regex :

- `~r"999001.*"` - Correspond aux IMSI commençant par 999001
- `~r"^310[0-9]{3}.*"` - Correspond aux IMSI avec des motifs MNC spécifiques
- `~r".*test$"` - Correspond aux valeurs se terminant par "test"

Meilleures pratiques

1. **Spécificité** : Ordre des règles de la plus spécifique à la plus générale
2. **Performance** : Placez les correspondances les plus courantes en premier pour réduire la surcharge de traitement
3. **Tests** : Validez les motifs regex avant le déploiement
4. **Documentation** : Utilisez des valeurs descriptives pour `rule_name` pour une clarté opérationnelle
5. **Surveillance** : Suivez les taux de correspondance des règles pour vérifier le comportement attendu

---

Module de métriques étendues

Le module de métriques étendues fournit des capacités de télémétrie et d'analyse avancées pour analyser les modèles de trafic Diameter au-delà des métriques standard.

Configuration

Activez le module et configurez des types de métriques spécifiques :

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

Paramètres

Paramètre	Description
`enabled`	Définir sur `true` pour activer le module de métriques étendues
`attach_attempt_reporting_enabled`	Activer le suivi et le rapport des tentatives de connexion LTE (S6a AIR/AIA)

Métriques disponibles

Suivi des tentatives de connexion

Suit les tentatives de connexion des abonnés LTE en surveillant les paires de messages de demande d'information d'authentification (AIR) et de réponse (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'

Réessayer

Mesure : `attach_attempt_count`

Champs :

- `imsi` - L'IMSI de l'abonné (à partir de l'AVP User-Name - voir codes AVP)

Tags :

- `origin_host` - Le pair qui a originairement fait la demande de connexion
- `result_code` - Le code de résultat Diameter de la réponse HSS

Comment ça fonctionne :

1. Lorsqu'une AIR (code de commande 318, application S6a 16777251 - voir Identifiants d'application) est reçue, le module extrait :
   - ID de bout en bout pour la corrélation demande/réponse
   - IMSI (AVP User-Name code 1)
   - Origin-Host (AVP code 264)
2. Les métadonnées de la demande sont stockées dans ETS avec un TTL
3. Lorsque l'AIA correspondante est reçue, le module :
   - Corrèle en utilisant l'ID de bout en bout
   - Extrait le code de résultat (AVP 268 ou AVP de résultat expérimental 297)
   - Émet la métrique avec l'IMSI, l'hôte d'origine et le code de résultat

Cas d'utilisation

- **Analyse du taux de succès des connexions** - Suivre les tentatives de connexion réussies par rapport aux échecs par code de résultat
- **Dépannage au niveau de l'IMSI** - Identifier les abonnés rencontrant des échecs de connexion
- **Surveillance des performances du réseau** - Surveiller les modèles de tentatives de connexion par origine (MME/SGSN)
- **Analyse du roaming** - Analyser les taux de succès des connexions de roaming entrant

Intégration

Les métriques étendues sont exportées via l'intégration InfluxDB :

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

Les codes de résultat sont des codes Diameter standard :

- `2001` - Succès (DIAMETER_SUCCESS)
- `5001` - Échec d'authentification (DIAMETER_AUTHENTICATION_REJECTED)
- `5004` - AVP Diameter non pris en charge
- Voir RFC 6733 pour la liste complète des codes de résultat

Notes importantes

- Les métriques des tentatives de connexion ne suivent que les paires AIR/AIA S6a (Application-Id 16777251, Command-Code 318)
- Les métadonnées de la demande expirent en fonction du délai d'attente de la demande configuré + 5 secondes
- Le traitement des métriques est asynchrone (processus lancé) pour éviter de bloquer le flux de messages
- Le module fonctionne indépendamment des modules de routage et de transformation

---

Métriques Prometheus

Le DRA expose des métriques Prometheus complètes pour surveiller le trafic Diameter, la santé des pairs et les opérations des modules. Toutes les métriques sont disponibles à l'endpoint `/metrics`.

Métriques de base Diameter

État des pairs

Métrique : `diameter_peer_status` **Type** : Gauge **Description** : Si le pair est connecté (1) ou non (0) **Tags** :

- `origin_host` - Identité Diameter du pair
- `ip` - Adresse IP du pair

Exemple :

```
# Vérifier si un pair spécifique est connecté
diameter_peer_status{origin_host="hss01.example.com"}

# Compter les pairs déconnectés
count(diameter_peer_status == 0)
```

Compte de messages

Métrique : `diameter_peer_message_count_total` **Type** : Counter **Description** : Nombre total de messages Diameter échangés avec les pairs **Tags** :

- `origin_host` - Identité Diameter du pair
- `received_from` - Pair d'où le message a été reçu
- `application_id` - Identifiant d'application Diameter (voir référence d'ID d'application)
- `cmd_code` - Code de commande Diameter (voir Codes de commande courants)
- `application_name` - Nom d'application lisible par l'homme (par exemple, "3GPP_S6a")
- `cmd_name` - Nom de commande lisible par l'homme (par exemple, "AIR")
- `direction` - "request" ou "response"

Exemple :

```
# Taux de demande AIR S6a d'un MME spécifique
rate(diameter_peer_message_count_total{
  cmd_code="318",
  direction="request",
  origin_host="mme01.example.com"
}[5m])

# Taux total de messages par application
sum by (application_name)
(rate(diameter_peer_message_count_total[5m]))
```

Codes de résultat des réponses

Métrique : `diameter_peer_message_result_code_count_total` **Type** : Counter **Description** : Nombre total de réponses Diameter par code de résultat **Tags** :

- `origin_host` - Demandeur d'origine
- `routed_to` - Pair qui a envoyé la réponse
- `application_id` - Identifiant d'application Diameter
- `cmd_code` - Code de commande Diameter
- `application_name` - Nom de l'application
- `cmd_name` - Nom de la commande
- `result_code` - Code de résultat Diameter ou code de résultat expérimental

Exemple :

```
# Taux de succès pour les demandes AIR S6a
rate(diameter_peer_message_result_code_count_total{
  cmd_code="318",
  result_code="2001"
}[5m])

# Taux d'erreur par code de résultat
sum by (result_code) (
  rate(diameter_peer_message_result_code_count_total{
    result_code!="2001"
  }[5m])
)
```

Codes de résultat courants :

- `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

Délai de réponse

Métrique : `diameter_peer_last_response_delay` **Type** : Gauge **Description** : Délai de réponse le plus récent en millisecondes (DRA → Pair → DRA) **Tags** :

- `origin_host` - Demandeur d'origine
- `routed_to` - Pair qui a envoyé la réponse
- `application_name` - Nom de l'application
- `cmd_name` - Nom de la commande

Exemple :

```
# Temps de réponse moyen de HSS
avg(diameter_peer_last_response_delay{routed_to="hss01.example.com"})

# Temps de réponse P95 pour S6a
histogram_quantile(0.95,
  rate(diameter_peer_last_response_delay{application_name="3GPP_S6a"}
[5m])
)
```

Demandes non répondues

Métrique : `diameter_peer_unanswered_request_count_total` **Type** : Counter **Description** : Demandes envoyées mais non répondues dans le délai d'attente **Tags** :

- `origin_host` - Demandeur d'origine

- `routed_to` - Pair qui n'a pas répondu
- `application_id` - Identifiant d'application Diameter
- `cmd_code` - Code de commande Diameter
- `application_name` - Nom de l'application
- `cmd_name` - Nom de la commande

Exemple :

```
# Taux de demandes non répondues
rate(diameter_peer_unanswered_request_count_total[5m])

# Identifier les pairs problématiques
topk(5, sum by (routed_to) (
  rate(diameter_peer_unanswered_request_count_total[5m])
))
```

Tentatives de connexion non autorisées

Métrique : `diameter_peer_unauthorized_connection_count_total` **Type** : Counter **Description** : Tentatives de connexion de pairs non autorisés **Tags** :

- `origin_host` - Identité revendiquée du pair non autorisé
- `supported_applications` - Applications annoncées par le pair
- `peer_ip` - Adresse IP de la tentative de connexion

Exemple :

```
# Tentatives de connexion non autorisées
rate(diameter_peer_unauthorized_connection_count_total[5m])

# Alerte sur un accès non autorisé
diameter_peer_unauthorized_connection_count_total > 0
```

Métriques du module de routage avancé

Trafic abandonné

Métrique : `diameter_advanced_routing_drop_count_total` **Type** : Counter
Description : Demandes abandonnées par l'action