# Routage Avancé

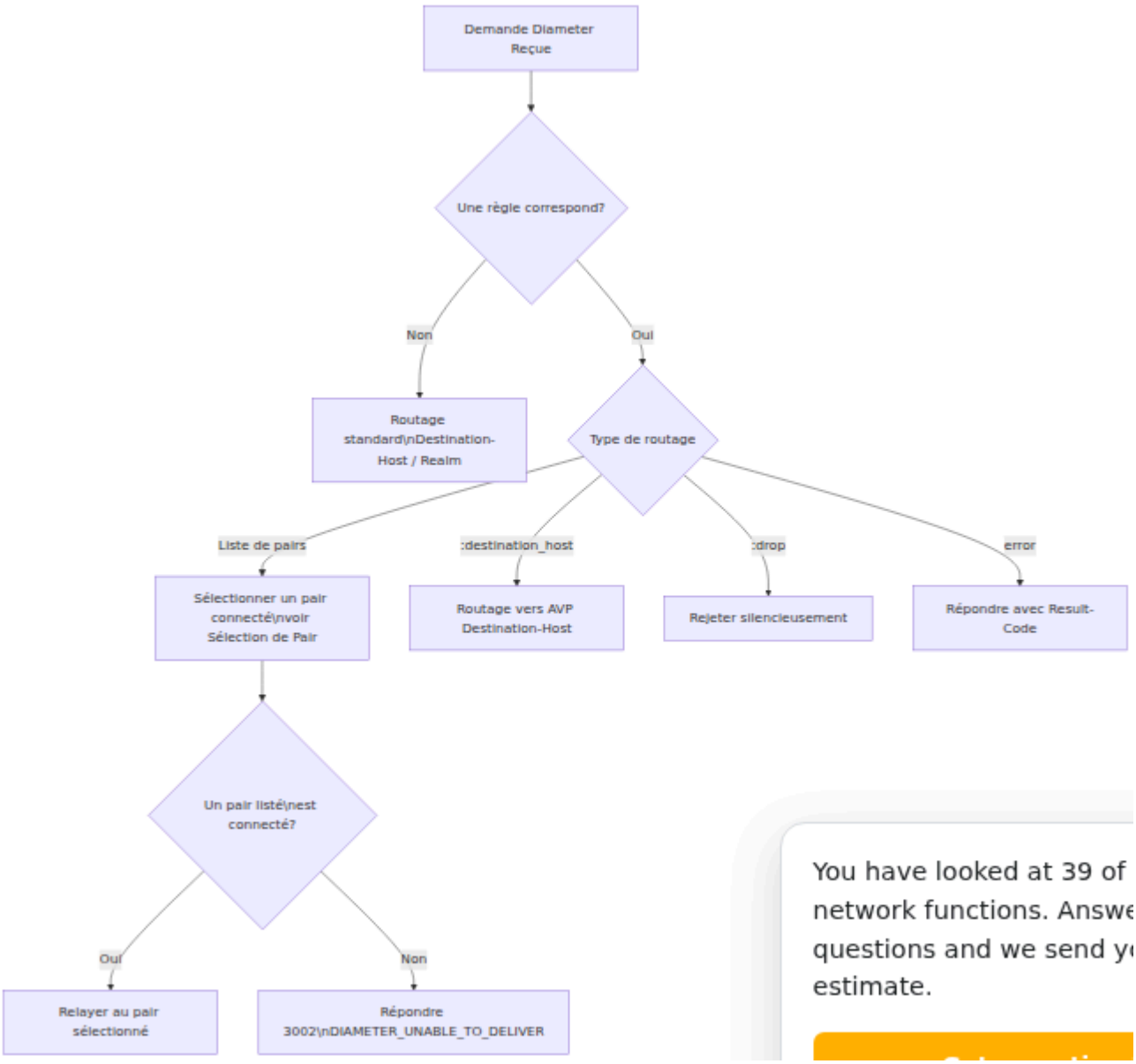
Le routage avancé permet au DRA de prendre des décisions de routage basées sur le contenu de chaque demande Diameter. Il n'utilise pas uniquement le **Destination-Realm**. Les **règles** correspondent aux propriétés du message, telles que le pair d'origine, l'Application-Id, le Command-Code et les valeurs AVP. Lorsqu'une règle correspond, elle dirige la demande vers un ensemble choisi de pairs ou une destination spéciale.

Une règle peut également distribuer le trafic entre plusieurs pairs pour **la redondance** et **l'équilibrage de charge**. Elle utilise la `priorité` et le `poids` par pair. Pour les applications avec état telles que Gx (PCRF) et Gy (OCS), le routage avancé préserve **l'affinité de session**. Les messages en cours de session retournent toujours au pair qui possède la session.

Le routage avancé évalue **uniquement les paquets de demande Diameter entrants**. Les paquets de réponse suivent le routage de session établi vers le pair d'origine. Le DRA ne les réévalue jamais par rapport à ces règles. Pour réécrire le contenu des messages, voir Transformation Avancée. Pour le routage basé sur le domaine/hôte qui s'exécute lorsque aucune règle ne correspond, voir Routage Standard.

# Comment ça fonctionne

Le DRA évalue chaque demande entrante par rapport aux règles configurées **dans l'ordre, de haut en bas**. La **première** règle dont les filtres correspondent l'emporte et décide du routage. Le DRA n'évalue pas les règles suivantes. Si aucune règle ne correspond, la demande continue avec le routage standard **Destination-Host** / **Destination-Realm**.




You have looked at 39 of
network functions. Answe
questions and we send y
estimate.

Une règle correspondante ne peut résoudre que vers des pairs qui ne sont **pas actuellement connectés**. Dans ce cas, le DRA répond avec **DIAMETER_UNABLE_TO_DELIVER (3002)**. Le DRA ne renvoie jamais la demande au routage standard. Le routage standard la livrerait au mauvais nœud et retournerait un résultat trompeur.

# Position dans le Pipeline

Le routage avancé s'exécute après le Roaming Steering et la Recherche d'Abonné. Sa décision remplace le pair que le routage standard sélectionnerait autrement.




# Traitement des Règles

Le DRA évalue les règles de haut en bas. La **première règle correspondante l'emporte**. Chaque règle combine ses filtres selon son mode de `match`. Elle applique son action de `route` dès qu'elle correspond. Si aucune règle ne correspond, le DRA revient au routage standard.



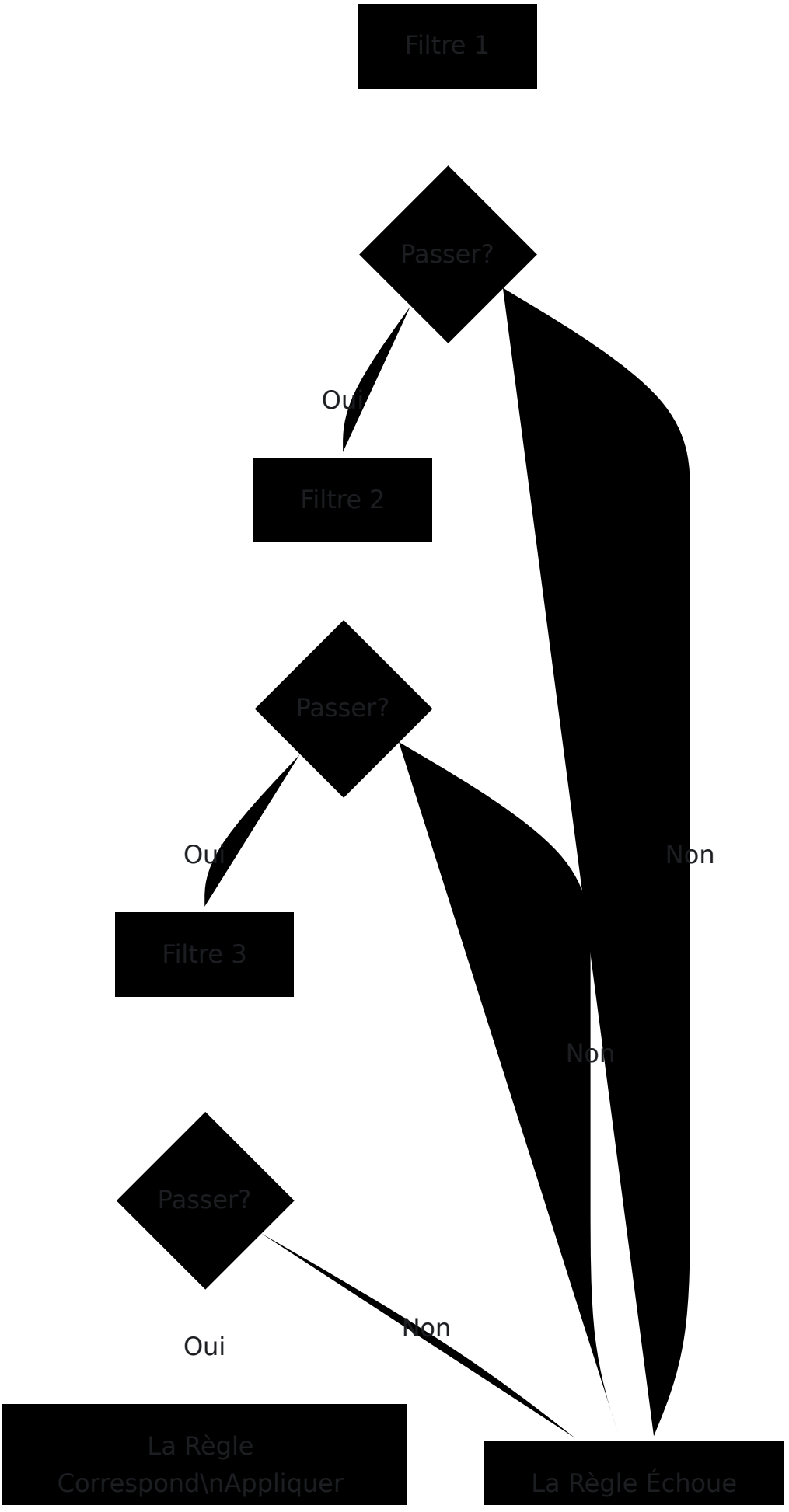

1. Le DRA évalue les règles **dans l'ordre de haut en bas**, comme défini dans la configuration.
2. Le DRA combine les filtres au sein d'une règle selon le mode de `match` de la règle (`:all`, `:any`, ou `:none`).
3. La **première règle correspondante l'emporte**. Le DRA n'évalue pas les règles suivantes.
4. Si aucune règle ne correspond, le DRA utilise le routage standard.

## Modes de Correspondance

Le paramètre `match` détermine comment le DRA combine les filtres d'une règle.

### `:all` (logique ET)

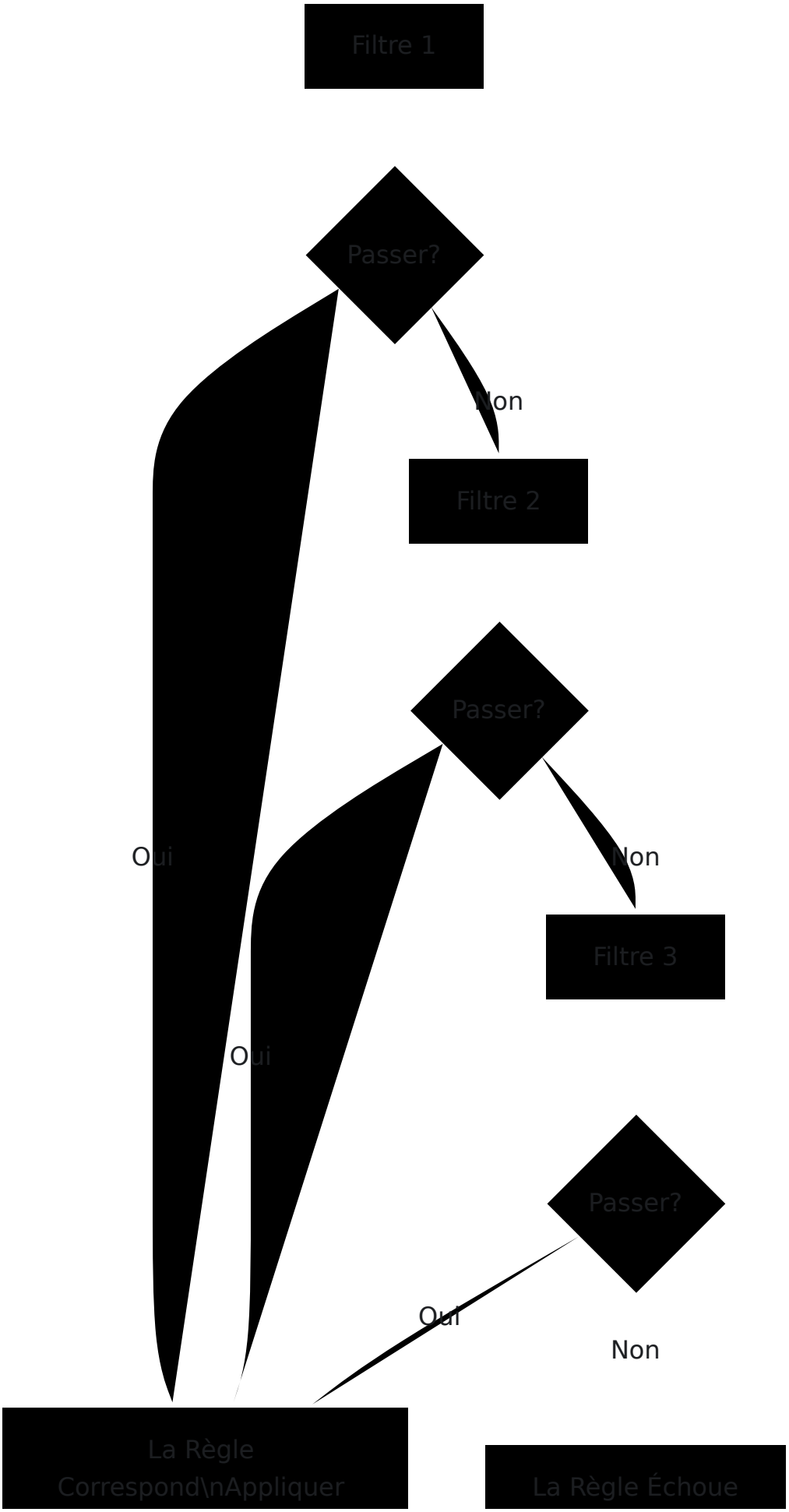
Chaque filtre doit correspondre pour que la règle réussisse. Avec trois filtres, `filter1 AND filter2 AND filter3` doit tous être vrai.


Filtre 1
Passer?
Oui
Filtre 2
Passer?
Oui
Non
Filtre 3
Non
Passer?
Non
Oui
La Règle Correspond\nAppliquer
La Règle Échoue

[redacted]

### `:any` (logique OU)

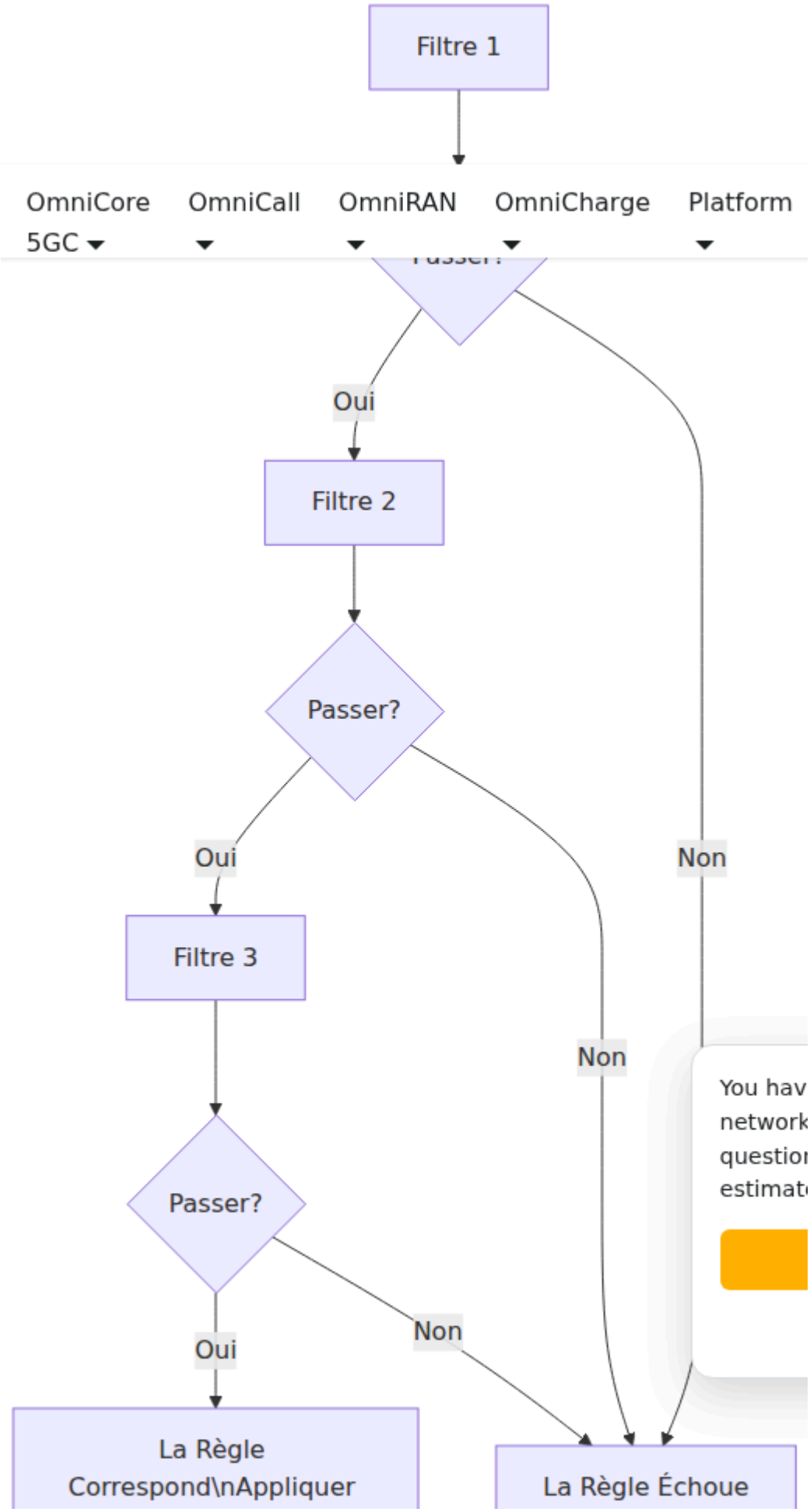
Au moins un filtre doit correspondre. Avec trois filtres, `filter1 OR filter2 OR filter3` réussit si l'un d'eux passe.


Filtre 1
Passer?
Non
Filtre 2
Passer?
Oui
Non
Filtre 3
Oui
Passer?
Oui
Non
La Règle Correspond\nAppliquer
La Règle Échoue

[redacted] [redacted]

### `:none` (logique NOR)

Aucun filtre ne doit correspondre. C'est une correspondance inverse. Avec trois filtres, `NOT filter1 AND NOT filter2 AND NOT filter3` doit tous tenir pour que la règle réussisse. Chaque filtre doit échouer.


Filtre 1
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Oui
Filtre 2
Passer?
Oui
Non
Filtre 3
Non
Passer?
Oui
Non
La Règle Correspond\nAppliquer
La Règle Échoue




# Sélection de Pair

Lorsqu'une règle route vers une liste de pairs, le DRA sélectionne uniquement parmi les pairs qui sont **actuellement connectés**. La façon dont vous écrivez les pairs détermine comment le DRA choisit entre eux.

## Noms d'hôtes simples

Une liste de noms d'hôtes simples suit l'algorithme de `peer_selection_algorithm` global (`:failover` ou `:random`) configuré sous `diameter`. C'est le comportement d'origine et ne change pas. Vous devez définir chaque pair dans la configuration de pair Diameter du DRA. Nommez chaque pair par son nom d'hôte **exact**, qui est sensible à la casse. Chaque pair doit être actuellement connecté. Le DRA ignore les pairs déconnectés.

```
route: %{peers: ["hss01.example.com", "hss02.example.com"]}
```

- `:failover` - toujours le premier pair connecté dans la liste. Cela donne un primaire déterministe, puis un secondaire.
- `:random` - un pair connecté aléatoire. Cela répartit la charge de manière uniforme.

## Priorité et poids

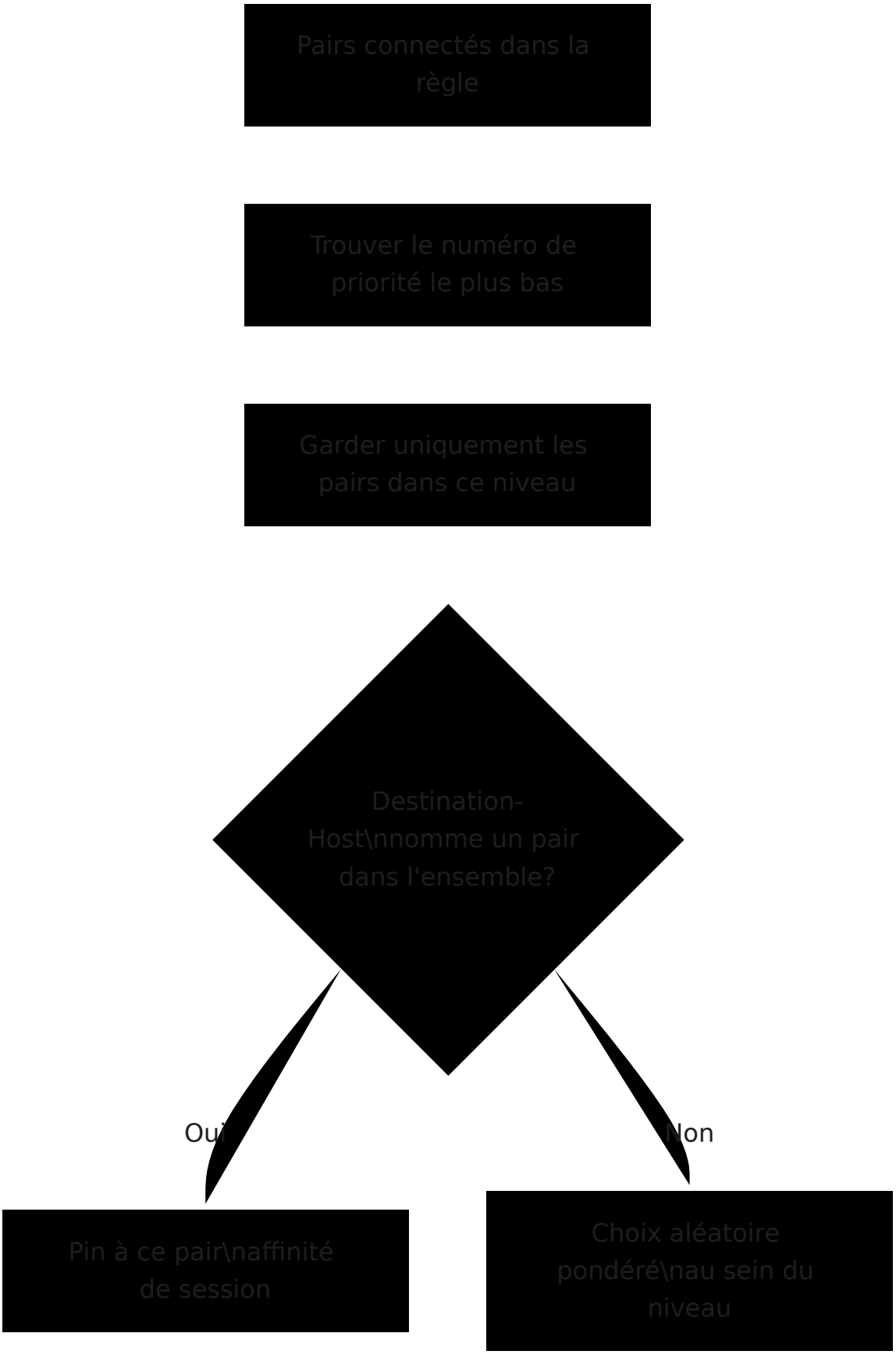
Pour la redondance par règle et l'équilibrage de charge, écrivez les pairs sous forme de cartes avec `priority` et `weight`. Cela est indépendant de l'algorithme global.

```
route: %{
  peers: [
    %{host: "pcrf01.example.com", priority: 10, weight: 50},
    %{host: "pcrf02.example.com", priority: 10, weight: 50},
    %{host: "pcrf-dr.example.com", priority: 20, weight: 100}
  ]
}
```

La sélection est **"la priorité choisit le niveau, le poids divise la charge"** :

1. Le DRA considère uniquement les pairs connectés.
2. Le **numéro de priorité le plus bas** présent l'emporte sur le niveau. Un numéro plus bas est préféré, comme avec les enregistrements DNS SRV. Les niveaux numérotés plus élevés agissent comme des sauvegardes. Le DRA utilise un niveau de sauvegarde uniquement après que chaque pair connecté dans un niveau inférieur est parti. Cela fournit **la redondance**.
3. Au sein du niveau gagnant, le DRA choisit un pair au hasard, **pondéré par le `weight`**. Des poids égaux répartissent la charge de manière uniforme. Des poids inégaux biaisent la répartition. Cela fournit **l'équilibrage de charge**.

Si vous omettez `priority` ou `weight`, chacun d'eux est par défaut à `100`.




Ce modèle unique exprime chaque topologie courante :

| Objectif | Comment l'exprimer |
| --- | --- |
| Pure redondance | Priorités distinctes décroissantes (par exemple 10, 20, 30) ; poids non pertinent |
| Équilibrage de charge pur | Une priorité pour tous ; poids égaux |
| Équilibrage de charge pondéré | Une priorité pour tous ; poids dans le ratio souhaité (par exemple 70 / 30) |
| Équilibrage de charge avec sauvegarde | Deux pairs à priorité 10, un troisième à priorité 20 |

# Affinité de Session (Gx / Gy)

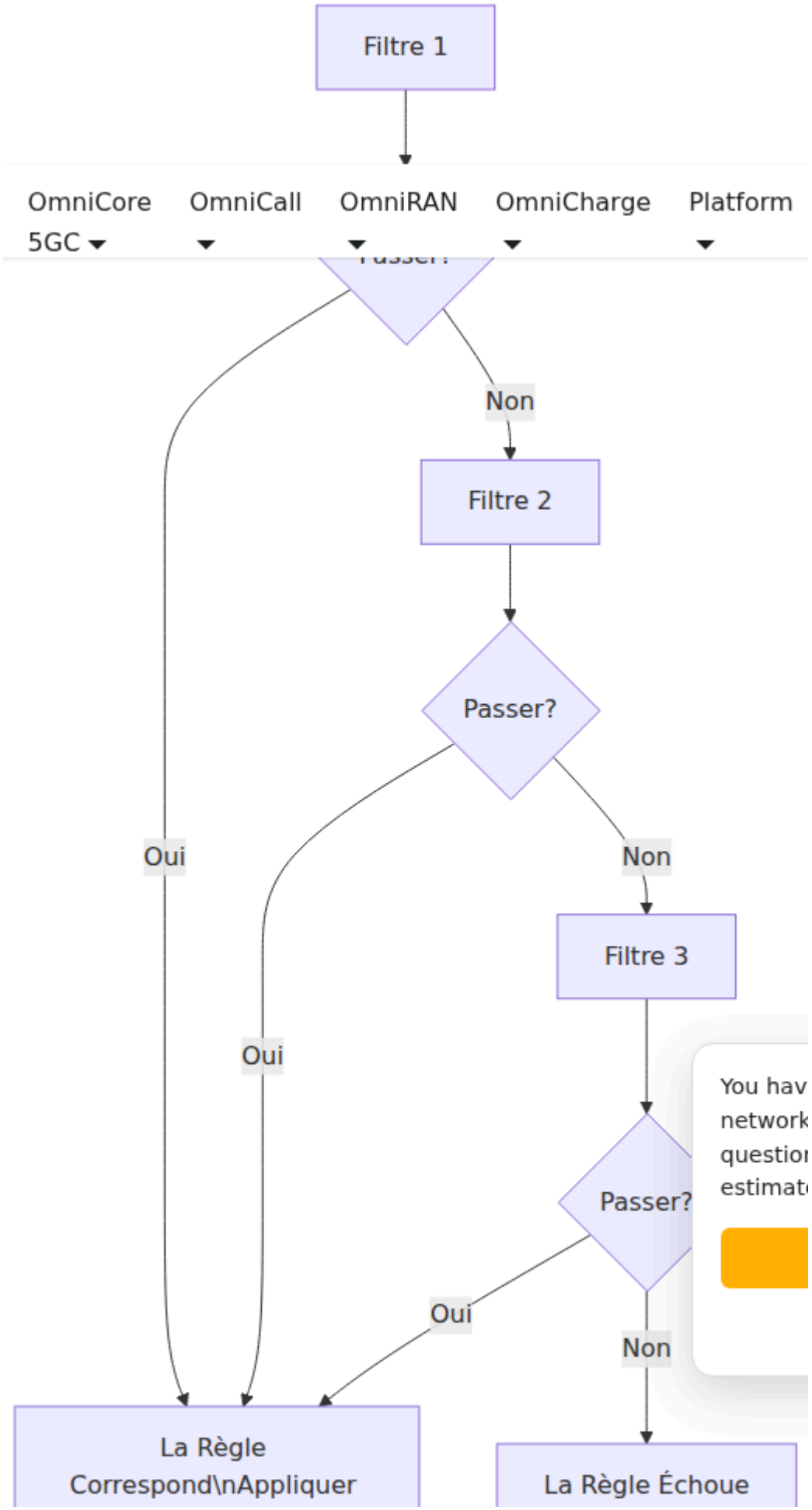
Les applications avec état portent un **Session-Id** pendant la durée d'une session. Les messages en cours de session doivent atteindre le **même** serveur qui a géré la demande d'initiation de session. Si le DRA rééquilibre un message en cours de session vers un pair différent, ce pair le rejette avec **DIAMETER_UNKNOWN_SESSION_ID (5002)**.

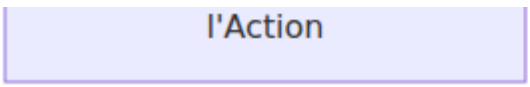
Le routage avancé préserve l'affinité avec le mécanisme Diameter standard, l'**AVP Destination-Host (293)**. Lorsqu'une règle utilise des pairs avec priorité/poids :

- Si la demande porte un Destination-Host qui nomme l'un des pairs connectés de la règle, le DRA **pince** la demande à ce pair. Cela remplace la priorité et le poids.
- S'il n'y a pas de tel Destination-Host, le DRA équilibre la charge de la demande normalement.

La demande d'initiation de session (par exemple Gx CCR-Initial) ne porte pas de Destination-Host. Le DRA l'équilibre à travers le niveau. La réponse retourne l'Origin-Host du serveur choisi. Le client place ensuite cette valeur dans Destination-Host sur toutes les demandes ultérieures (CCR-Update, CCR-

Terminate). Cela fixe automatiquement ces demandes à ce serveur. Dans Gx/Gy, l'**AVP CC-Request-Type (416)** distingue le type de demande.


Filtre 1
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Non
Filtre 2
Passer?
Oui
Non
Filtre 3
Oui
Passer?
Oui
Non
La Règle Correspond\nAppliquer
La Règle Échoue




Cette affinité s'applique aux listes de pairs avec priorité/poids. Les listes de noms d'hôtes simples suivent l'algorithme global. Ce comportement ne les affecte pas.

# Configuration

Configurez le routage avancé sous `module_advanced_routing` dans `config/runtime.exs`.

```
module_advanced_routing: %{
  # Activer ou désactiver le module
  enabled: false,

  # Les règles sont évaluées dans l'ordre ; la première
correspondance l'emporte
  rules: [
    %{
      rule_name: "route_s6a_air_to_hss",
      match: :all,
      filters: [
        {:application_id, 16_777_251},
        {:command_code, 318}
      ],
      route: %{
        peers: ["hss01.example.com"]
      }
    }
  ]
}
```

## Paramètres du Module

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `false` | Interrupteur principal pour le module. Lorsque `false`, le DRA n'évalue aucune règle et tout le trafic utilise le routage standard. |
| `rules` | Liste | Oui | `[]` | Liste ordonnée des règles de routage. Le DRA les évalue de haut en bas. La première correspondance l'emporte. Voir Paramètres de Règle. |

## Paramètres de Règle

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `rule_name` | Chaîne | Oui | - | Un nom descriptif et unique pour la règle. Le DRA l'affiche dans les journaux et l'interface de gestion. |
| `match` | Atome | Oui | - | Comment les filtres se combinent : `:all` (ET logique), `:any` (OU logique), ou `:none` (NOR logique). Voir Modes de Correspondance. |
| `filters` | Liste | Oui | - | Les conditions qu'un message doit remplir. Voir Filtres. |
| `route` | Carte ou Atome | Oui | - | Où le DRA envoie le trafic correspondant. C'est une liste de pairs ou une destination spéciale. Voir Destinations de Routage. |

## Filtres

Un filtre est un tuple de `{filter_type, value}`. Le routage avancé prend en charge ces filtres :

| Filtre | Exemple | Description |
|---|---|---|
| `:via_peer` | `{:via_peer, "mme01.example.com"}` | L'Origin-Host du pair qui a envoyé la demande (le dernier saut). |
| `:application_id` | `{:application_id, 16_777_251}` | L'Application-Id Diameter. Voir Identifiants d'Application. |
| `:command_code` | `{:command_code, 318}` | Le Command-Code Diameter. |
| `:avp` | `{:avp, {1, "999999000000001"}}` | Une valeur AVP, écrite sous la forme `{avp_code, value}`. |

**Valeurs de Filtre**

- **Valeurs de liste (OU).** Toute valeur de filtre peut être une liste. Le DRA traite la liste comme un OU logique sur ses membres. Par exemple, `{:command_code, [317, 318]}` correspond au code de commande 317 ou 318. `{:via_peer, ["dra01.example.com", "dra02.example.com"]}` correspond à l'un ou l'autre pair.
- **Valeurs Regex.** Les valeurs de filtre AVP et les membres de liste prennent en charge les expressions régulières via un sigil regex, par exemple `{:avp, {1, ~r"999001.*"}}`. Vous pouvez mélanger les membres, par exemple `{:avp, {1, ["505571234567", ~r"999001.*", ~r"505057.*"]}}`.
- **Vérification de présence (`:any`).** Une valeur de `:any` correspond lorsque l'attribut est présent avec n'importe quelle valeur. Par exemple, `{:avp, {264, :any}}` correspond chaque fois que l'AVP Origin-Host existe. `{:via_peer, :any}` correspond à n'importe quel pair d'origine.

**Limitations des filtres AVP**

Utilisez les filtres AVP pour des AVP **simples** uniquement, tels que User-Name, Origin-Host et Destination-Realm. **Les AVP groupés ne sont pas pris en**

**charge** et ne correspondent pas. Les valeurs binaires complexes ne sont pas prises en charge. Utilisez toujours la forme `{:avp, {code, value}}`.

## Destinations de Routage

Le paramètre `route` prend soit une carte de liste de pairs, soit l'un des atomes/tupples de destination spéciaux ci-dessous.

| Destination | Exemple | Description |
|---|---|---|
| Liste de pairs | `%{peers: [...]}` | Routage vers l'un des pairs listés. Les entrées peuvent être des noms d'hôtes simples ou des cartes de priorité/poids. Voir Sélection de Pair. |
| Destination-Host | `:destination_host` | Routage vers le pair nommé dans l'**AVP Destination-Host (293)** du message. Si l'AVP est absent, la règle ne s'applique pas et le DRA utilise le routage standard. |
| Rejeter | `:drop` | Jeter le message sans réponse au pair demandeur. Utilisez ceci pour bloquer le trafic indésirable, tel que des abonnés de test ou des plages non désirées. |
| Erreur | `{:error, 3004}` | Répondre avec une réponse d'erreur Diameter qui porte le Result-Code donné. Le DRA remplit automatiquement Origin-Host, Origin-Realm et Session-Id. Le DRA ne transmet pas le message à aucun pair. |

Pour une application de **relais**, seuls les codes de résultat 3xxx (protocole) sont strictement valides sur les réponses générées (`3002`, `3003`, `3004`, `3007`).

Certaines déploiements utilisent également `5012` (DIAMETER_UNABLE_TO_COMPLY) pour rejeter des commandes spécifiques. Voir les tables de référence pour les significations.

## Paramètres de Pair (forme priorité/poids)

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `host` | Chaîne | Oui | - | L'Origin-Host du pair. Il doit correspondre exactement à l'identité Diameter configurée du pair et est sensible à la casse. |
| `priority` | Entier | Non | `100` | Le niveau. Le DRA préfère les numéros plus bas. Il utilise les niveaux numérotés plus élevés uniquement lorsque tous les pairs connectés dans un niveau inférieur sont indisponibles. |
| `weight` | Entier | Non | `100` | La part de trafic au sein d'un niveau. Le DRA divise le trafic entre les pairs d'un niveau en proportion de leurs poids. Un poids de `0` exclut un pair, sauf si chaque pair dans le niveau est `0`. |

# Exemples de Configuration

## Redondance vers un Pair de Sauvegarde

Routage des demandes d'Information d'Authentification S6a vers un HSS principal. Revenir à un HSS de récupération en cas de panne du principal.

```
%{
  rule_name: "s6a_air_failover",
  match: :all,
  filters: [
    {:application_id, 16_777_251},
    {:command_code, 318}
  ],
  route: %{
    peers: [
      %{host: "hss01.example.com", priority: 10},
      %{host: "hss-dr.example.com", priority: 20}
    ]
  }
}
```

**Comment ça fonctionne :** Tant que `hss01` est connecté, chaque demande lui est envoyée. Si `hss01` se déconnecte, les demandes basculent vers `hss-dr`. Si les deux sont hors ligne, le DRA répond aux demandes avec 3002.

## Équilibrage de Charge Égal

Répartir le trafic Gx de manière égale entre deux PCRF.

```
%{
  rule_name: "gx_load_balance",
  match: :all,
  filters: [
    {:application_id, 16_777_238}
  ],
  route: %{
    peers: [
      %{host: "pcrf01.example.com", priority: 10, weight: 50},
      %{host: "pcrf02.example.com", priority: 10, weight: 50}
    ]
  }
}
```

**Comment ça fonctionne :** Le DRA divise les nouvelles sessions (CCR-Initial) à peu près 50/50 entre les deux PCRF. Le DRA fixe les messages en cours de session au PCRF qui a géré la session. Il utilise le Destination-Host que portent les messages.

## Équilibrage de Charge Pondéré avec une Sauvegarde

Envoyer plus de trafic à un PCRF plus grand et moins à un plus petit. Garder une sauvegarde pour quand les deux sont indisponibles.

```
%{
  rule_name: "gx_weighted_with_backup",
  match: :all,
  filters: [
    {:application_id, 16_777_238}
  ],
  route: %{
    peers: [
      %{host: "pcrf01.example.com", priority: 10, weight: 70},
      %{host: "pcrf02.example.com", priority: 10, weight: 30},
      %{host: "pcrf-dr.example.com", priority: 20, weight: 100}
    ]
  }
}
```

**Comment ça fonctionne :** Tant qu'un des PCRF à priorité 10 est connecté, le DRA divise les nouvelles sessions 70/30 entre eux. Le trafic passe à `pcrf-dr` uniquement lorsque **les deux** PCRF à priorité 10 sont hors ligne. Les sessions établies restent fixées à leur serveur pendant toute leur durée.

## Routage par Pair d'Origine

Routage du trafic S6a qui arrive de DRAs en amont spécifiques vers une paire de HSS locale. Cela est utile lors d'une migration ou d'un basculement.

```
%{
  rule_name: "s6a_via_peer_to_local_hss",
  match: :all,
  filters: [
    {:application_id, 16_777_251},
    {:via_peer, ["dra01.example.com", "dra02.example.com"]},
    {:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}}
  ],
  route: %{
    peers: ["hss01.example.com", "hss02.example.com"]
  }
}
```

**Comment ça fonctionne :** Le DRA route uniquement les demandes S6a vers les nœuds HSS locaux qui sont arrivées de `dra01` ou `dra02` et portent le Realm d'Origine correspondant (AVP 296). Tout le reste passe aux règles suivantes ou au routage standard.

## Roaming par Correspondance de Modèle par IMSI

Routage du trafic de roaming entrant basé sur les modèles IMSI (User-Name, AVP 1).

```
%{
  rule_name: "inbound_s6a_roaming",
  match: :all,
  filters: [
    {:application_id, 16_777_251},
    {:avp, {296, "epc.mnc001.mcc001.3gppnetwork.org"}},
    {:avp, {1, ["505571234567", ~r"999001.*"]}}
  ],
  route: %{
    peers: ["dra01.example.com", "dra02.example.com"]
  }
}
```

**Comment ça fonctionne :** La règle correspond aux messages S6a du Realm d'Origine donné dont le User-Name est exactement `505571234567` ou commence par `999001`. Le DRA route ces messages vers les pairs désignés. La valeur de liste applique une logique OU entre la chaîne exacte et l'expression régulière.

## Honorer l'AVP Destination-Host

Routage vers le pair que l'expéditeur a nommé, uniquement pour le trafic correspondant.

```
%{
  rule_name: "honour_destination_host",
  match: :all,
  filters: [
    {:avp, {1, [~r"90199.*"]}}
  ],
  route: :destination_host
}
```

**Comment ça fonctionne :** Lorsque l'IMSI correspond, le DRA envoie la demande au pair nommé dans son AVP Destination-Host (293). Si l'AVP Destination-Host est manquant, le DRA considère la correspondance comme un échec et applique le routage standard.

# Rejeter le Trafic Indésirable

Jeter silencieusement le trafic de plage de test.

```
%{
  rule_name: "drop_test_subscribers",
  match: :all,
  filters: [
    {:application_id, 16_777_251},
    {:avp, {1, [~r"999999.*"]}}
  ],
  route: :drop
}
```

**Comment ça fonctionne :** Le DRA rejette les messages S6a dont l'IMSI commence par `999999`. Il n'envoie aucune réponse. Utilisez ceci pour filtrer le trafic de test ou bloquer des plages d'abonnés spécifiques.

# Protection contre la Surcharge avec une Réponse d'Erreur

Retourner DIAMETER_TOO_BUSY à un pair spécifique en surcharge.

```
%{
  rule_name: "rate_limit_high_volume_peer",
  match: :all,
  filters: [
    {:via_peer, "mme-overloaded-01.example.com"},
    {:application_id, 16_777_251}
  ],
  route: {:error, 3004}
}
```

**Comment ça fonctionne :** Le DRA répond au trafic S6a du pair nommé avec **DIAMETER_TOO_BUSY (3004)**. Cela signale au pair de se retirer. Le DRA ne transmet rien en amont.

## Bloquer une Commande Spécifique

Rejeter un type de commande particulier avec une erreur explicative au lieu d'un rejet silencieux.

```
%{
  rule_name: "block_purge_requests",
  match: :all,
  filters: [
    {:application_id, 16_777_251},
    {:command_code, 321}
  ],
  route: {:error, 5012}
}
```

**Comment ça fonctionne :** Le DRA répond aux demandes S6a Purge-UE-Requests (commande 321) avec **DIAMETER_UNABLE_TO_COMPLY (5012)**. Cela désactive cette opération et laisse les autres commandes intactes.

# Lecture de la Configuration Active

Vous pouvez lire les règles actives via l'API de gestion à `GET /api/rules`. La réponse inclut les règles de routage, les règles de transformation et l'algorithme de `routing_algorithm` global.

Pour chaque règle qui route vers une liste de pairs, l'API rapporte comment la sélection se comporte :

- `route.selection` est `"priority_weight"` lorsque la règle utilise des cartes de priorité/poids. Sinon, c'est l'algorithme global (`"failover"` ou `"random"`).
- En mode `priority_weight`, l'API retourne chaque pair avec sa `priority` et son `weight` effectifs. Elle remplit les valeurs par défaut afin que la distribution soit explicite.

```
{
  "status": "success",
  "response": {
    "routing_algorithm": "random",
    "routing_rules": [
      {
        "rule_name": "gx_weighted_with_backup",
        "match": "all",
        "route": {
          "selection": "priority_weight",
          "peers": [
            { "host": "pcrf01.example.com", "priority": 10,
"weight": 70 },
            { "host": "pcrf02.example.com", "priority": 10,
"weight": 30 },
            { "host": "pcrf-dr.example.com", "priority": 20,
"weight": 100 }
          ]
        }
      }
    ],
    "transform_rules": []
  }
}
```


# Meilleures Pratiques

1. **Spécificité d'abord.** Ordre des règles de la plus spécifique à la plus générale. La première correspondance l'emporte.
2. **Correspondances courantes tôt.** Placez les correspondances à volume le plus élevé près du haut. Cela réduit la surcharge d'évaluation.
3. **Valider les regex.** Testez les motifs d'expressions régulières contre les messages capturés avant le déploiement.
4. **Nommer clairement.** Utilisez des valeurs descriptives pour `rule_name`. Les journaux et l'interface de gestion s'expliquent alors d'eux-mêmes.
5. **Préférer priorité/poids pour les applications avec état.** Utilisez des cartes de pairs priorité/poids pour Gx/Gy. Le DRA applique alors automatiquement l'affinité de session.

# Dépannage

## Sessions Rejetées avec UNKNOWN_SESSION_ID

**Symptômes** : Les sessions Gx ou Gy s'établissent correctement, mais le serveur rejette les mises à jour en cours de session avec le Result-Code 5002.

**Causes possibles** :

- Les clients (PCEF/OCF) ne définissent pas Destination-Host sur les demandes en cours de session, donc le DRA ne peut pas les fixer.
- Une règle route ce trafic avec des noms d'hôtes simples et l'algorithme global `:random`. Cet algorithme n'applique pas l'affinité de session.

**Résolution** :

1. Confirmez que les demandes en cours de session portent un Destination-Host qui nomme le serveur qui a répondu à la demande initiale.
2. Convertissez la liste de pairs de la règle en cartes de priorité/poids. Le DRA applique alors automatiquement l'affinité de session.

## Trafic Correspondant Répondu avec 3002

**Symptômes** : Une règle devrait router certaines demandes, mais le DRA y répond avec DIAMETER_UNABLE_TO_DELIVER.

**Causes possibles** :

- Aucun des pairs listés dans la règle correspondante n'est actuellement connecté.
- Les noms d'hôtes dans la règle ne correspondent pas exactement aux valeurs Origin-Host des pairs.

**Résolution** :

1. Vérifiez l'état de connexion des pairs dans l'interface de gestion ou les journaux.

2. Confirmez que les valeurs `host` dans la règle correspondent exactement aux Origin-Host des pairs. La correspondance est sensible à la casse.

## Une Règle Ne Prend Pas Effet

**Symptômes** : Le trafic qui devrait correspondre à une règle passe par le routage standard à la place.

**Causes possibles** :

- `enabled` est défini sur `false`.
- Une règle ant��rieure dans la liste correspond d'abord et remporte le routage.
- Une valeur de filtre ne correspond pas. Cela peut être un problème de casse sur une valeur AVP, ou un AVP groupé non pris en charge.

**Résolution** :

1. Confirmez que `enabled` est `true`. Redémarrez le DRA après le changement.
2. Passez en revue l'ordre des règles. La première règle correspondante l'emporte.
3. Confirmez les valeurs de filtre par rapport à un message capturé. N'oubliez pas que le DRA ne peut pas correspondre aux AVP groupés.

# Référence

## Codes de Résultat Diameter

| Code | Nom | Description | Référence |
|---|---|---|---|
| 3002 | DIAMETER_UNABLE_TO_DELIVER | Aucun pair connecté n'est disponible pour livrer la demande | RFC 6733 §7.1.3 |
| 3003 | DIAMETER_REALM_NOT_SERVED | Aucun routage configuré ne sert le domaine de la demande | RFC 6733 §7.1.3 |
| 3004 | DIAMETER_TOO_BUSY | Le nœud est surchargé et ne peut pas traiter la demande | RFC 6733 §7.1.3 |
| 5002 | DIAMETER_UNKNOWN_SESSION_ID | Le serveur ne reconnaît pas le Session-Id. Par exemple, un message en cours de session a atteint le mauvais serveur | RFC 6733 §7.1.5 |

| Code | Nom | Description | Référence |
|---|---|---|---|
| 5012 | DIAMETER_UNABLE_TO_COMPLY | Le DRA ne peut pas honorer la demande. Utilisez ceci pour rejeter des commandes spécifiques | RFC 6733 §7.1.5 |

## AVP Pertinents

| AVP | Code | Description | Référence |
|---|---|---|---|
| Destination-Host | 293 | Nomme le pair spécifique qui doit recevoir une demande. C'est la base de l'affinité de session | RFC 6733 §6.5 |
| Session-Id | 263 | Identifie une session pendant sa durée de vie | RFC 6733 §8.8 |
| CC-Request-Type | 416 | Distingue les demandes INITIAL, UPDATE et TERMINATION dans Gx/Gy | RFC 4006 §8.3 |

## Identifiants d'Application

| ID | Interface | Description | Référence |
|---|---|---|---|
| 16777238 | Gx | Contrôle de politique entre PCEF et PCRF | 3GPP TS 29.212 |
| 16777251 | S6a/S6d | Authentification MME/SGSN et gestion des abonnements vers HSS | 3GPP TS 29.272 |
| 4 | Gy | Chargement en ligne entre OCF et OCS | RFC 4006 |

# Documentation Connexe

- Routage Standard - Routage Destination-Host / Destination-Realm utilisé lorsque aucune règle ne correspond.
- Transformation Avancée - Réécriture de contenu pour les demandes et réponses, partageant le même moteur de filtre.
- Référence de Configuration - Référence complète des paramètres `config/runtime.exs`, y compris l'algorithme global `peer_selection_algorithm`.

# Transformation Avancée

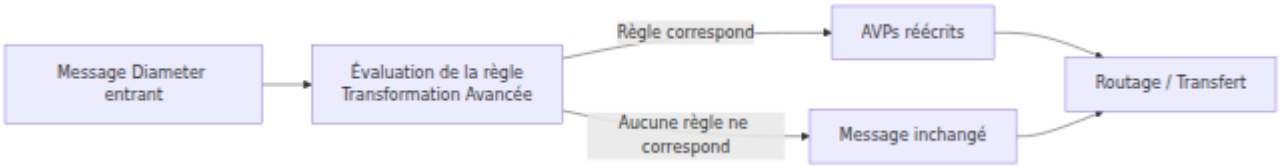
Le module **Transformation Avancée** réécrit les Paires Attribut-Valeur (**AVPs**) à l'intérieur des messages Diameter qui correspondent à des critères définis par l'opérateur. Utilisez-le pour corriger, traduire ou normaliser le contenu des messages au fur et à mesure qu'ils passent par OmniDRA. Vous n'avez pas besoin de modifier les éléments de réseau d'envoi ou de réception.

## Vue d'ensemble

La Transformation Avancée évalue une liste de **règles** pour chaque message Diameter. Une règle combine un ensemble de **filtres** (les critères de correspondance) avec une **transformation** (le changement à appliquer). Lorsqu'une règle correspond, le DRA applique sa transformation au message et arrête le traitement des règles.

La Transformation Avancée fournit trois actions de transformation. `:edit` change la valeur d'un AVP existant. `:remove` supprime un ou plusieurs AVPs par code. `:overwrite` réécrit un ou plusieurs AVPs par nom, y compris les AVPs groupés. Voir Actions de Transformation.

La Transformation Avancée partage son modèle de traitement des règles avec le moteur de routage. Voir Routage Avancé pour savoir comment le DRA prend des décisions de routage.

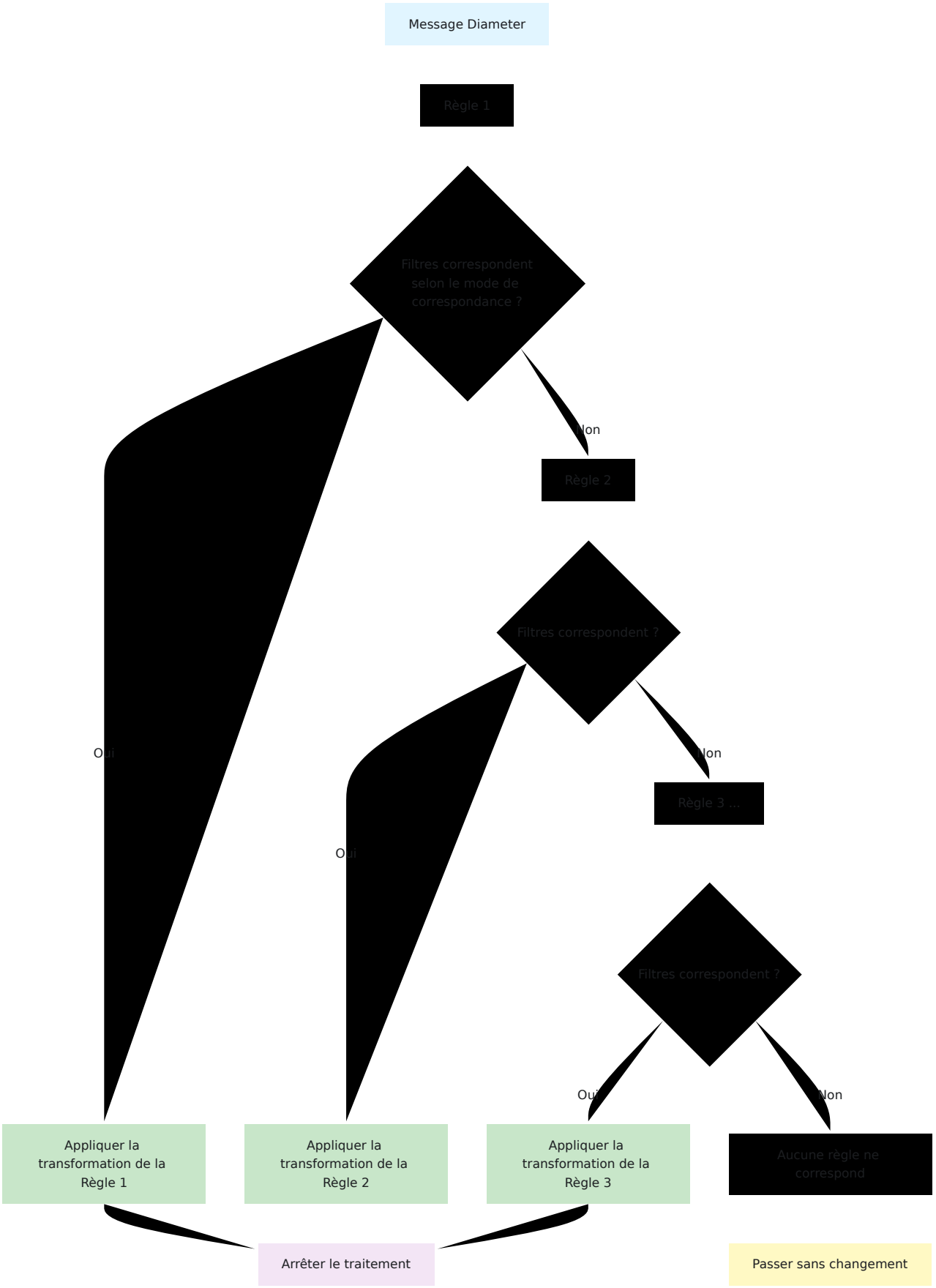


# Comment les Règles Sont Traitée

Le DRA évalue les règles **dans l'ordre, de haut en bas**, telles qu'elles apparaissent dans la configuration. La **première règle correspondante l'emporte**. Dès qu'un filtre de règle correspond, le DRA applique sa transformation et ne considère plus d'autres règles. Si aucune règle ne correspond, le message passe **transparente** sans modification.

Au sein d'une seule règle, le paramètre `match` contrôle comment les filtres individuels se combinent :

- `:all` - logique **ET**. Chaque filtre doit passer pour que la règle corresponde.
- `:any` - logique **OU**. Au moins un filtre doit passer pour que la règle corresponde.
- `:none` - logique **NOR**. Aucun filtre ne peut passer pour que la règle corresponde (correspondance inverse).

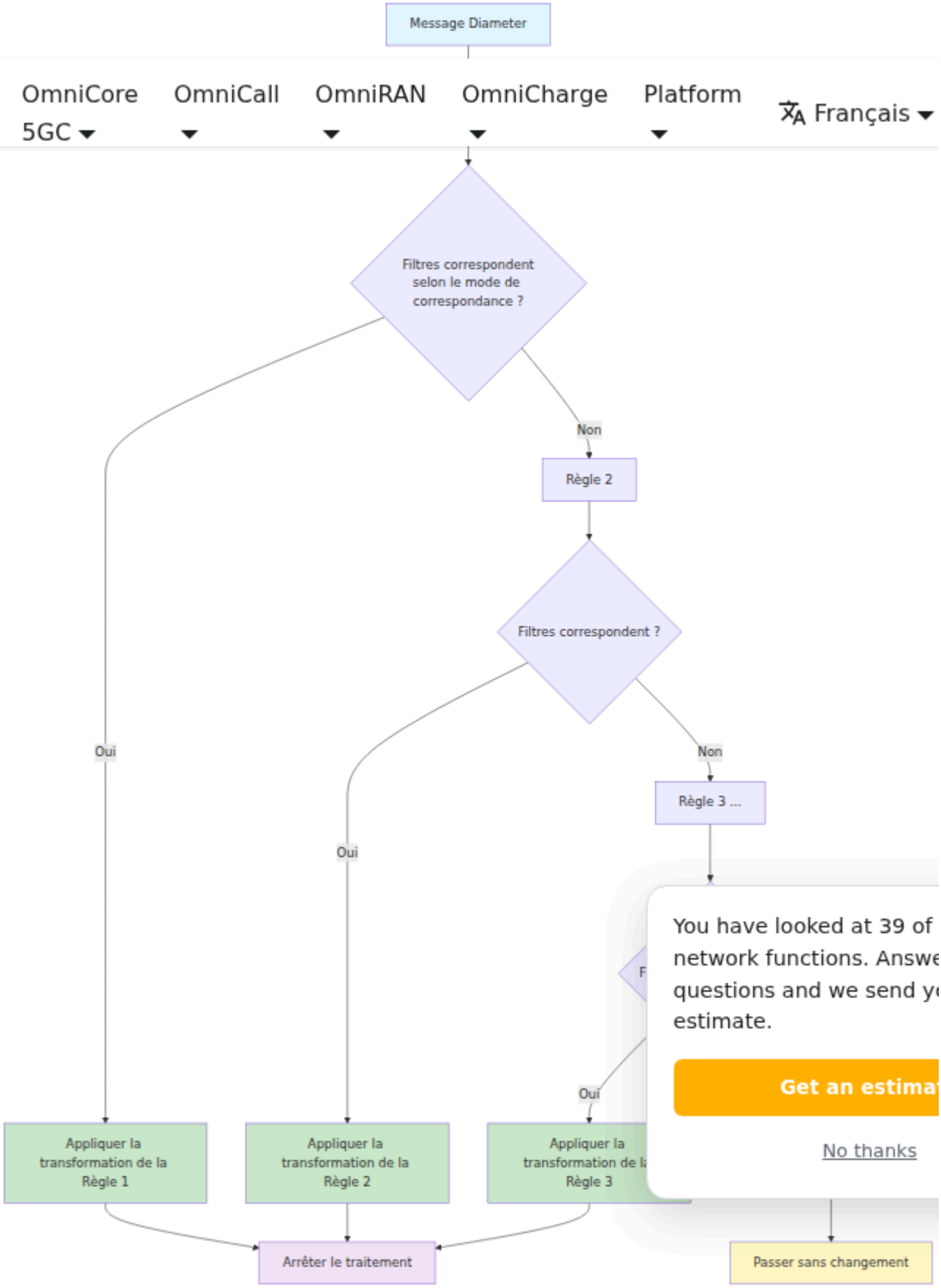



Les règles de transformation opèrent sur le contenu du message. Elles ne sélectionnent pas une destination. Une règle peut réécrire un AVP, par exemple

un domaine ou une identité. Le moteur de routage utilise ensuite cet AVP pour transférer le message.

## Évaluation du Mode de Correspondance

Les trois modes de correspondance évaluent la même liste de filtres différemment :




OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks

# Filtres

Les filtres sont les conditions qui décident si une règle s'applique à un message. La Transformation Avancée fournit six types de filtres.

| Filtre | Format | Description |
|---|---|---|
| `:packet_type` | `{:packet_type, :request}` | Correspond à la **direction** du message. Accepte uniquement `:request` ou `:answer` ; une liste de valeurs n'est **pas** acceptée. Voir Filtrage par Type de Paquet. |
| `:application_id` | `{:application_id, <id>}` | Correspond à l'Application-Id Diameter du message (par exemple `16777251` pour S6a/S6d). Voir Identifiants d'Application. |
| `:command_code` | `{:command_code, <code>}` | Correspond au Code de Commande Diameter du message (par exemple `318` pour AIR). Voir Codes de Commande. |
| `:avp` | `{:avp, {<code>, <value>}}` | Correspond à un AVP par son code numérique et sa valeur attendue. Voir Filtrage AVP. |
| `:to_peer` | `{:to_peer, "hss01.example.com"}` | Correspond au **pair de destination résolu** d'une demande. Il correspond uniquement aux paquets **de demande** et ne correspond jamais aux réponses. Accepte une liste de noms d'hôtes (OU). Voir Filtrage par Pair. |

| Filtre | Format | Description |
|---|---|---|
| `:from_peer` | `{:from_peer, "hss01.example.com"}` | Correspond au **pair qui a répondu** à un message. Il correspond uniquement aux paquets **de réponse** et ne correspond jamais aux demandes. Accepte une liste de noms d'hôtes (OU). Voir Filtrage par Pair. |

## Filtrage par Type de Paquet

Le filtre `:packet_type` restreint une règle à une direction de trafic. Il accepte exactement l'une des deux valeurs :

- `:request` - correspond uniquement aux messages Diameter **de demande**.
- `:answer` - correspond uniquement aux messages Diameter **de réponse**.

Contrairement à `:application_id`, `:command_code` et `:avp`, le filtre `:packet_type` **n'accepte pas** une liste de valeurs.

## Filtrage AVP

Le filtre `:avp` a la structure `{avp_id, avp_value}`. L'`avp_id` est le code AVP numérique. L'`avp_value` est la valeur à comparer, exprimée sous forme de chaîne.

Utilisez le filtre `:avp` pour des **AVPs simples** tels que `User-Name` et `Origin-Host`. **Les AVPs groupés ne sont pas pris en charge** et ne correspondent pas. Les valeurs binaires complexes ne correspondent pas.

Le filtre correspond aux valeurs de plusieurs manières :

- **Chaîne exacte** - `{:avp, {1, "999999000000001"}}` correspond uniquement à cette valeur exacte.

- **Liste de valeurs** - `{:avp, {1, ["50557...", "505057..."]}}` correspond si l'AVP est égal à **n'importe quelle** valeur de la liste. C'est une logique OU au sein de la liste.
- **Expression régulière** - `{:avp, {1, [~r"9999999.*"]}}` correspond aux valeurs contre un motif regex.

Utilisez la syntaxe `~r"pattern"` pour la correspondance par expression régulière :

| Motif | Correspond |
|---|---|
| `~r"999001.*"` | Valeurs `User-Name`/IMSI commençant par `999001` |
| `~r"^310[0-9]{3}.*"` | Valeurs avec des préfixes MCC/MNC spécifiques |
| `~r".*test$"` | Valeurs se terminant par `test` |

# Filtrage par Pair

Les filtres `:to_peer` et `:from_peer` correspondent à un **nom d'hôte de pair**, et non au contenu du message. Ils sont spécifiques à la direction et complémentaires :

- **`:to_peer`** correspond au **pair de destination résolu** auquel le DRA est sur le point de transférer une demande. Il correspond uniquement aux paquets **de demande**. Il ne correspond jamais à une réponse.
- **`:from_peer`** correspond au **pair qui a répondu** à un message. Il correspond uniquement aux paquets **de réponse**. Il ne correspond jamais à une demande.

Les deux filtres acceptent un seul nom d'hôte ou une **liste de noms d'hôtes**. Avec une liste, le filtre correspond si le pair est égal à **n'importe quel** nom d'hôte de la liste. C'est une logique OU.

| Filtre | Exemple | Correspond |
|---|---|---|
| `:to_peer` | `{:to_peer, "hss01.example.com"}` | Demandes routées vers le pair `hss01.example.com` |
| `:to_peer` | `{:to_peer, ["hss01.example.com", "hss02.example.com"]}` | Demandes routées vers **l'un ou l'autre** des pairs listés |
| `:from_peer` | `{:from_peer, "hss01.example.com"}` | Réponses reçues du pair `hss01.example.com` |

# Actions de Transformation

La carte `transform` décrit la transformation. Sa clé `action` sélectionne **une des trois** transformations. La **direction** du message détermine quelles actions sont autorisées :

| Action | Ce qu'elle fait | Demandes | Réponses |
|---|---|---|---|
| `:edit` | Réécrit la **valeur** d'un AVP existant. | Oui | Non |
| `:remove` | Supprime l'AVP(s) nommé(s) du message. | Oui | Oui |
| `:overwrite` | Réécrit l'AVP(s) par nom, y compris les AVPs **groupés**, en utilisant un dictionnaire de décodage. | Oui | Oui |

## L'Action `:edit`

```
transform: %{
  action: :edit,
  avps: [{:avp, {1, "999999000000002"}}]
}
```

`:edit` se comporte comme suit :

- Il **modifie la valeur** de l'AVP nommé dans le message Diameter à la valeur fournie.
- Il n'affecte que les AVPs qui **existent déjà** dans le message. Si l'AVP ciblé n'est pas présent, le DRA ne fait **aucun changement** et passe le message tel qu'il a été reçu.
- Chaque entrée dans `avps` est `{:avp, {<code>, <new_value>}}`. Cela définit cet AVP à la nouvelle valeur.
- `:edit` s'applique uniquement aux **demandes**. Pour réécrire des AVPs sur les réponses, ou pour atteindre des AVPs groupés, utilisez `:overwrite`.

## L'Action `:remove`

```
transform: %{
  action: :remove,
  avps: [{:avp, {264, :any}}]
}
```

`:remove` se comporte comme suit :

- Il **supprime** l'AVP(s) nommé(s) dans `avps` du message.
- Il correspond uniquement par **code AVP**. Il **ignore** la valeur dans chaque entrée `avps` pour la suppression. Utilisez `:any` comme valeur pour le rendre explicite.
- Il s'applique à la fois aux **demandes et aux réponses**.

## L'Action `:overwrite`

```
transform: %{
  action: :overwrite,
  dictionary: :diameter_gen_3gpp_s6a,
  avps: [{:avp, {1, "999999000000002"}}]
}
```

`:overwrite` se comporte comme suit :

- Il **réécrit** l'AVP(s) nommé(s) par nom. Il utilise le `dictionnaire` fourni pour décoder et réencoder le message.
- Il décode le message complet, donc `:overwrite` peut atteindre les **AVPs groupés / enregistrement** que `:edit` ne peut pas.
- Il **requiert** une clé `dictionary`. Cette clé nomme le dictionnaire Diameter qui décrit le message (par exemple `:diameter_gen_3gpp_s6a`).
- Il s'applique à la fois aux **demandes et aux réponses**.

## Paramètres de la Carte de Transformation

| Paramètre | Type | Obligatoire | Par défaut | Description |
|---|---|---|---|---|
| `action` | Atome | Oui | - | La transformation à appliquer : `:edit` (réécrire une valeur AVP existante - **demandes uniquement**), `:remove` (supprimer l'AVP(s) par code - demandes et réponses), ou `:overwri` (réécrire l'AVP(s) par no y compris les AVPs groupés - demandes et réponses). |
| `avps` | Liste | Oui | - | Liste des cibles AVP. Pou `:edit`/`:overwrite`, chaque entrée est `{:av` `{<code>, <new_value>}` et définit cet AVP à `<new_value>`. Pour `:remove`, le DRA utilise uniquement `<code>` et ignore la valeur (utilisez `:any`). |
| `dictionary` | Atome | Seulement pour `:overwrite` | `:not_used` | Le dictionnaire Diamete qui décode et réencode message afin que le DR puisse écraser les AVPs nommés (groupés), par exemple `:diameter_gen_3gpp_s6` Les actions `:edit` et |

| Paramètre | Type | Obligatoire | Par défaut | Description |
|---|---|---|---|---|
| | | | | `:remove` ignorent cette clé. |

# Configuration

Configurez la Transformation Avancée sous `module_advanced_transform` dans `config/runtime.exs`. Le module est désactivé par défaut. Il ne fait rien jusqu'à ce que vous définissiez `enabled` sur `true`.

```
module_advanced_transform: %{
  enabled: true,
  rules: [
    %{
      rule_name: "transform_user_name",
      match: :all,
      filters: [
        {:application_id, 16_777_251},
        {:command_code, 318},
        {:avp, {1, "999999000000001"}}
      ],
      transform: %{
        action: :edit,
        avps: [{:avp, {1, "999999000000002"}}]
      }
    }
  ]
}
```

## Paramètres du Module

| Paramètre | Type | Obligatoire | Par défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `false` | Active le module. Lorsque `false`, le DRA n'évalue aucune règle et tous les messages passent sans changement. |
| `rules` | Liste | Oui (lorsqu'il est activé) | - | Liste ordonnée des règles de transformation. Le DRA les évalue de haut en bas. La première règle correspondante l'emporte. Voir Paramètres de Règle. |

## Paramètres de Règle

Chaque entrée dans `rules` est une carte avec les clés suivantes.

| Paramètre | Type | Obligatoire | Par défaut | Description |
|---|---|---|---|---|
| `rule_name` | Chaîne | Oui | - | Identifiant unique et descriptif pour la règle. Il soutient la clarté opérationnelle et la surveillance. |
| `match` | Atome | Oui | - | Comment les filtres se combinent : `:all` (ET), `:any` (OU), ou `:none` (NOR). Voir Comment les Règles Sont Traitée. |
| `filters` | Liste | Oui | - | Liste des conditions de filtre qu'un message doit satisfaire selon le mode `match`. Voir Filtres. |
| `transform` | Carte | Oui | - | Le changement à appliquer lorsque la règle correspond. Voir Paramètres de la Carte de Transformation. |

# Exemples

## Exemple 1 : Réécrire un User-Name sur les Demandes S6a AIR

```
module_advanced_transform: %{
  enabled: true,
  rules: [
    %{
      rule_name: "transform_user_name",
      match: :all,
      filters: [
        {:application_id, 16_777_251},
        {:command_code, 318},
        {:avp, {1, "999999000000001"}}
      ],
      transform: %{
        action: :edit,
        avps: [{:avp, {1, "999999000000002"}}]
      }
    }
  ]
}
```

**Comment cela fonctionne :** Avec `match: :all`, chaque filtre doit passer. La règle ne correspond qu'aux messages S6a (`16777251`) de Demande d'Information d'Authentification (`318`) dont le `User-Name` (AVP `1`) est égal à `999999000000001`. Lors d'une correspondance, l'action `:edit` réécrit `User-Name` à `999999000000002`. Si un message ne contient pas d'AVP `User-Name`, le DRA ne fait aucun changement.

**Cas d'utilisation :** Corriger ou normaliser une identité d'abonné spécifique alors qu'elle transite par le DRA. Par exemple, mapper un IMSI hérité à sa valeur actuelle lors d'une migration.

# Exemple 2 : Réécriture de Domaine Basée sur la Plage IMSI

```
module_advanced_transform: %{
  enabled: true,
  rules: [
    %{
      rule_name: "rewrite_destination_realm_roaming_partner",
      match: :all,
      filters: [
        {:avp, {296, "epc.mnc057.mcc505.3gppnetwork.org"}},
        {:avp, {1, [~r"50557.*"]}}
      ],
      transform: %{
        action: :edit,
        avps: [{:avp, {283, "epc.mnc030.mcc310.3gppnetwork.org"}}]
      }
    }
  ]
}
```

**Comment cela fonctionne :** La règle correspond lorsque le `Origin-Realm` (AVP `296`) est égal au domaine du partenaire en itinérance **et** que le `User-Name`/IMSI (AVP `1`) commence par `50557`. Elle réécrit ensuite le `Destination-Realm` (AVP `283`). Le moteur de routage transfère ensuite le message vers le bon réseau partenaire. La première règle correspondante l'emporte, donc placez les plages IMSI plus spécifiques au-dessus des plus larges.

**Cas d'utilisation :** Diriger le trafic des abonnés en itinérance ou MVNO vers le bon réseau central hébergé. Le DRA sélectionne le réseau par la plage IMSI et le domaine d'origine.

## Exemple 3 : Supprimer un AVP des Réponses d'un Pair Spécifique

```
module_advanced_transform: %{
  enabled: true,
  rules: [
    %{
      rule_name: "strip_origin_host_from_partner_answers",
      match: :all,
      filters: [
        {:packet_type, :answer},
        {:from_peer, "hss01.partner.example.com"},
        {:command_code, 318}
      ],
      transform: %{
        action: :remove,
        avps: [{:avp, {264, :any}}]
      }
    }
  ]
}
```

**Comment cela fonctionne :** Le filtre `:packet_type` limite la règle aux **réponses**. Le filtre `:from_peer` la limite aux réponses de `hss01.partner.example.com`. Le filtre `:command_code` la limite aux AIA (`318`). Lorsque les trois correspondent, l'action `:remove` supprime l'`Origin-Host` (AVP `264`) quelle que soit sa valeur. La suppression correspond uniquement par code AVP.

**Cas d'utilisation :** Supprimer un AVP des réponses d'un réseau partenaire. Utilisez cela lorsque le partenaire remplit l'AVP de manière incorrecte ou lorsque l'AVP ne doit pas être transmis en aval.

# Exemple 4 : Écraser un AVP Groupé sur les Demandes S6a

```
module_advanced_transform: %{
  enabled: true,
  rules: [
    %{
      rule_name: "overwrite_user_name_s6a",
      match: :all,
      filters: [
        {:packet_type, :request},
        {:application_id, 16_777_251},
        {:command_code, 318}
      ],
      transform: %{
        action: :overwrite,
        dictionary: :diameter_gen_3gpp_s6a,
        avps: [{:avp, {1, "999999000000002"}}]
      }
    }
  ]
}
```

**Comment cela fonctionne :** La règle correspond aux **demandes** S6a (`16777251`) AIR (`318`). L'action `:overwrite` décode le message avec le dictionnaire `:diameter_gen_3gpp_s6a`, réécrit les AVP(s) nommés, et réencode le message. Elle décode le message complet, donc `:overwrite` peut atteindre les AVPs imbriqués à l'intérieur de structures groupées/enregistrement que `:edit` ne peut pas toucher.

**Cas d'utilisation :** Corriger un AVP qui se trouve à l'intérieur d'un AVP groupé, par exemple un champ au sein d'un abonnement ou d'un groupe spécifique au fournisseur. Un simple `:edit` ne correspondrait pas à un tel AVP.

# Tables de Référence

## Identifiants d'Application

| ID | Interface | Description | Référence |
|---|---|---|---|
| 16777251 | S6a/S6d | Authentification et abonnement MME/SGSN ↔ HSS | 3GPP TS 29.272 |

## Codes de Commande

| Code | Commande | Interface | Référence |
|---|---|---|---|
| 318 | Demande/Réponse d'Information d'Authentification (AIR/AIA) | S6a/S6d | 3GPP TS 29.272 §7.2.5 |
| 316 | Demande/Réponse de Mise à Jour de Localisation (ULR/ULA) | S6a/S6d | 3GPP TS 29.272 §7.2.3 |

### Codes AVP Communs

| Code | AVP | Description | Référence |
|---|---|---|---|
| 1 | User-Name | Identité de l'abonné (IMSI sur S6a). | 3GPP TS 29.272 / RFC 6733 §8.14 |
| 264 | Origin-Host | Identité Diameter de l'initiateur du message. | RFC 6733 §6.3 |
| 283 | Destination-Realm | Le domaine auquel le message est destiné. Le DRA l'utilise pour le routage. | RFC 6733 §6.6 |
| 296 | Origin-Realm | Domaine de l'initiateur du message. | RFC 6733 §6.4 |

# Meilleures Pratiques

1. **Spécificité** - Ordre des règles de la plus spécifique à la plus générale. La première règle correspondante l'emporte.
2. **Performance** - Placez les règles les plus souvent correspondantes en premier. Cela réduit la surcharge d'évaluation.
3. **Test** - Validez les motifs d'expressions régulières avant le déploiement.
4. **Nommage** - Utilisez des valeurs descriptives pour `rule_name` pour la clarté opérationnelle et la surveillance.
5. **Surveillance** - Suivez les taux de correspondance des règles. Cela confirme que les règles se comportent comme prévu.

# Documentation Connexe

- Routage Avancé - Sélection de destination utilisant le même modèle de traitement des règles.

# Vue d'ensemble de l'architecture DRA

OmniDRA est un **Diameter Routing Agent (DRA)**. Il se situe au centre d'un réseau de signalisation Diameter. Il relaie les messages de demande et de réponse entre des clients tels que les nœuds MME/SGSN et des services en arrière-plan tels que le HSS, le PCRF et l'OCS. OmniDRA est construit en Elixir/Erlang sur la bibliothèque OTP `:diameter`. Il fonctionne comme un **relai Diameter** et transmet les messages entre pairs. Il ne termine pas les applications que les messages transportent.

## Table des matières


1. Ce qu'est le DRA
2. Contexte réseau
3. Pipeline de traitement des demandes
4. Registre des pairs
5. Relais de messages
6. Tables de référence
7. Lectures complémentaires


---

## Ce qu'est le DRA

Un **relai Diameter** reçoit des messages Diameter sur un socket d'écoute. Il décide quel pair connecté doit gérer chaque message. Il transmet ensuite le message sans changement, à l'exception des AVPs de routage que le relai gère. OmniDRA fonctionne en tant que relai, il n'implémente donc pas la logique applicative des interfaces qu'il transporte. Il route **S6a**, **Gx**, **Gy**, et toute autre application Diameter de manière transparente. Les pairs doivent annoncer leur support pour l'application.

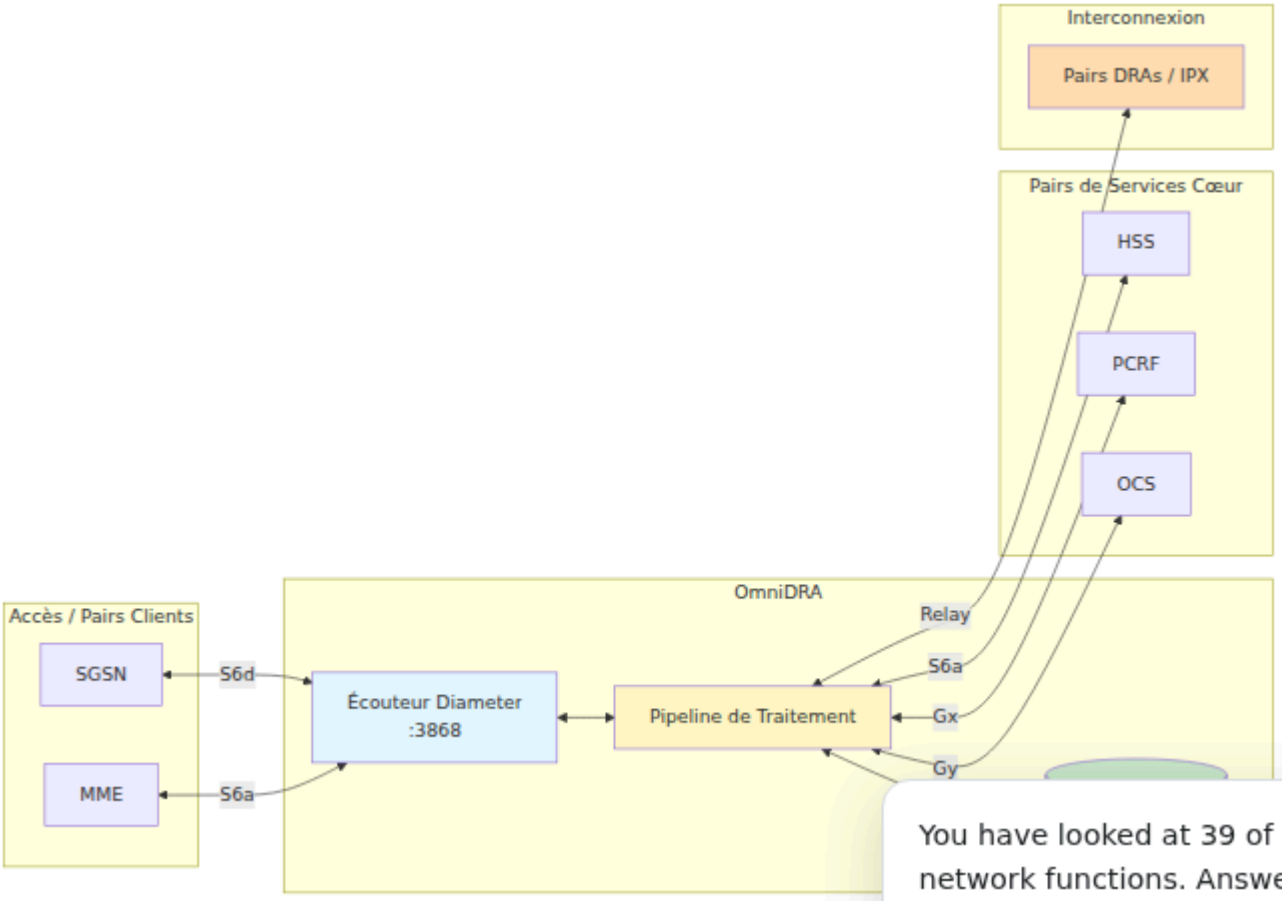
Caractéristiques clés :

- **Routage basé sur des normes** - Le routage des demandes suit le mécanisme de priorité du Diameter Base Protocol (RFC 6733). Il sélectionne un pair avec le **Destination-Host**, **Destination-Realm**, et **Application-Id**.
- **Corrélation des réponses avec état** - Les réponses retournent toujours le long du chemin inverse de leur demande. Le DRA corrèle chaque réponse par l'**Identifiant Hop-by-Hop**.
- **Modules de traitement optionnels** - Le pilotage de l'itinérance, la recherche d'abonnés, le routage avancé et les modules de transformation étendent le comportement standard. Ils ne le remplacent pas.
- **Pipeline de sécurité** - La limitation de débit, le pare-feu Diameter, la désinfection des AVP et le masquage de topologie protègent le réseau à la frontière de signalisation.

---

# Contexte réseau

D��ployez le DRA entre les nœuds de signalisation **côté accès** (tels que MME et SGSN) et les services **côté cœur** (tels que HSS, PCRF et OCS). Chaque pair établit une connexion Diameter au DRA. Le DRA n'initie jamais de trafic applicatif de sa propre initiative.

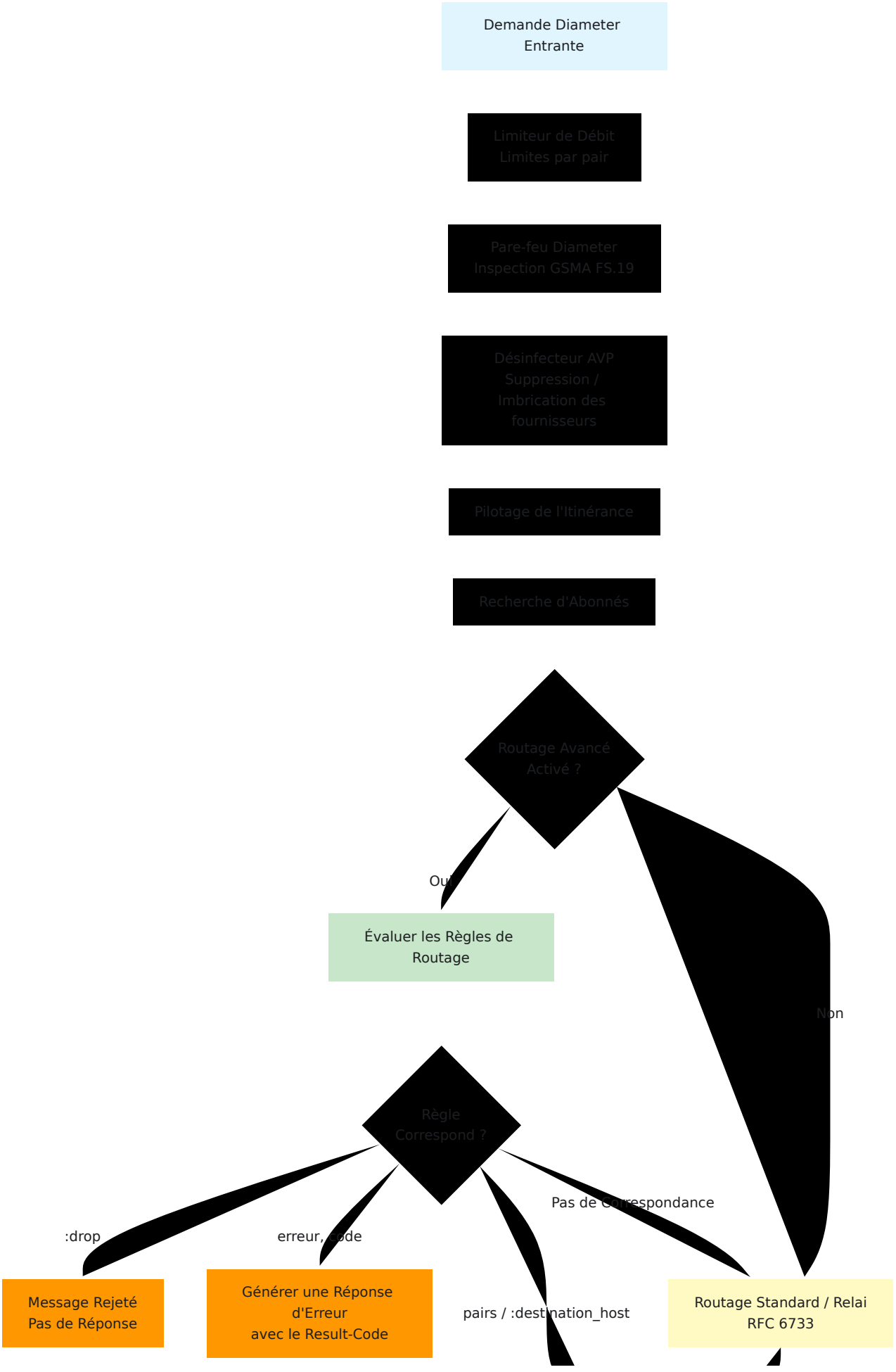



Le DRA écoute les connexions Diameter entrantes sur le port TCP/SCTP `3868`. C'est le port standard défini par RFC 6733. Chaque pair annonce les domaines et les applications qu'il supporte lors de l'**Échange de Capacités** (CER/CEA). Le DRA enregistre cela dans son registre des pairs pour l'utiliser dans les décisions de routage.

---

# Pipeline de traitement des demandes

Chaque demande entrante passe par un pipeline ordonné. Les étapes de sécurité et de pilotage s'exécutent en premier. Les étapes de routage s'exécutent ensuite. Les réponses contournent ce pipeline. Voir Relais de messages.


Demande Diameter Entrante
Limiteur de Débit Limites par pair
Pare-feu Diameter Inspection GSMA FS.19
Désinfecteur AVP Suppression / Imbrication des fournisseurs
Pilotage de l'Itinérance
Recherche d'Abonnés
Routage Avancé Activé ?
Oui
Évaluer les Règles de Routage
Non
Règle Correspond ?
:drop
erreur, code
Pas de Correspondance
pairs / :destination_host
Message Rejeté Pas de Réponse
Générer une Réponse d'Erreur avec le Result-Code
Routage Standard / Relai RFC 6733




## Étapes du pipeline

| Étape | But |
|---|---|
| **Limiteur de Débit** | Applique des limites de taux de demande par pair pour protéger les services en aval contre la surcharge. |
| **Pare-feu Diameter** | Inspecte les demandes selon les vérifications de couche basse et de catégorie 1/2/3 de GSMA FS.19. Voir Sécurité. |
| **Désinfecteur AVP** | Supprime les AVP de fournisseur non autorisés et impose des limites de profondeur d'imbrication des AVP. |
| **Pilotage de l'Itinérance** | Applique un pilotage défini par l'opérateur pour influencer le réseau vers lequel un abonné en itinérance est dirigé. |
| **Recherche d'Abonnés** | Résout le contexte de routage spécifique à l'abonné avant l'évaluation des règles. |
| **Routage Avancé** | Évalue le routage basé sur des règles qui peut rejeter, générer une erreur ou diriger une demande vers des pairs spécifiques. Voir Routage Avancé. |

| Étape | But |
| --- | --- |
| **Routage Standard / Relai** | Se rabat sur le routage prioritaire RFC 6733 lorsque aucune règle avancée ne s'applique. Voir Routage Standard. |
| **Liage de Session Rx** | Redirige une demande Rx vers le PCRF qui détient sa session. C'est la seule étape qui s'exécute *après* le Routage Avancé. Une demande `Session-Termination-Request` Rx ne transporte rien d'autre sur lequel router. Voir Liage de Session Rx. |
| **Masquage de Topologie** | Réécrit ou supprime les AVP identifiants (Route-Record, Origin-Host/Realm) sur les messages sortants. Voir Sécurité. |

Lorsque le Routage Avancé produit un résultat `:drop`, le DRA rejette la demande sans réponse. Un résultat `error` génère une réponse Diameter qui transporte le **Result-Code** configuré. Sinon, le DRA transmet les demandes correspondantes aux pairs spécifiés ou au pair **Destination-Host**. Les demandes non correspondantes passent au routage standard.

Le Liage de Session Rx est la seule exception au fait que le Routage Avancé ait le dernier mot. Son champ d'application est étroit. Il ne redirige qu'une demande qui allait déjà être relayée, et uniquement pour Rx. Il laisse un résultat `:drop`, un résultat `error`, et un `Destination-Host` explicite intouchés.

## Priorité du Routage Standard

Lorsque une demande atteint le routage standard, le DRA sélectionne un pair en utilisant le mécanisme de priorité de RFC 6733 §6.1:

1. **Destination-Host AVP (293)** - Si la demande contient cet AVP, le DRA route directement vers le pair nommé. C'est le mécanisme de priorité le plus élevé. Si ce pair n'est pas connecté, le routage échoue.
2. **Destination-Realm AVP (283)** - Si le Destination-Host est absent, le DRA sélectionne un pair connecté qui annonce le domaine cible. Le DRA

équilibre la charge entre les pairs correspondants.

3. **Application-Id** - Le DRA filtre les pairs candidats pour ceux qui annoncent un support pour l'application du message. Les pairs annoncent ce support lors de l'Échange de Capacités.

Le comportement complet du routage standard, y compris l'algorithme de sélection des pairs, est documenté dans Routage Standard.

---

# Registre des pairs

Le DRA maintient un **registre des pairs** en mémoire. Le registre suit chaque connexion de pair et les capacités que le pair a annoncées lors de l'Échange de Capacités. Le registre est la source autorisée pour les décisions de routage. Seuls les pairs dans l'état **connect��** sont éligibles pour recevoir du trafic. Le DRA exclut automatiquement les pairs déconnectés ou hors service.

Pour chaque pair connecté, le registre enregistre :

- L'**Identité Diameter** du pair (Origin-Host / FQDN).
- Les **domaines** que le pair annonce.
- Les **Application-Ids** que le pair supporte.
- L'**état de connexion** actuel.

Le DRA apprend le support des domaines et des applications de manière dynamique à partir de CER/CEA. Pour ajouter de la capacité à une interface, connectez un pair supplémentaire qui annonce le domaine et l'application pertinents. Le DRA commence à équilibrer la charge vers le nouveau pair automatiquement.

---

# Relais de messages

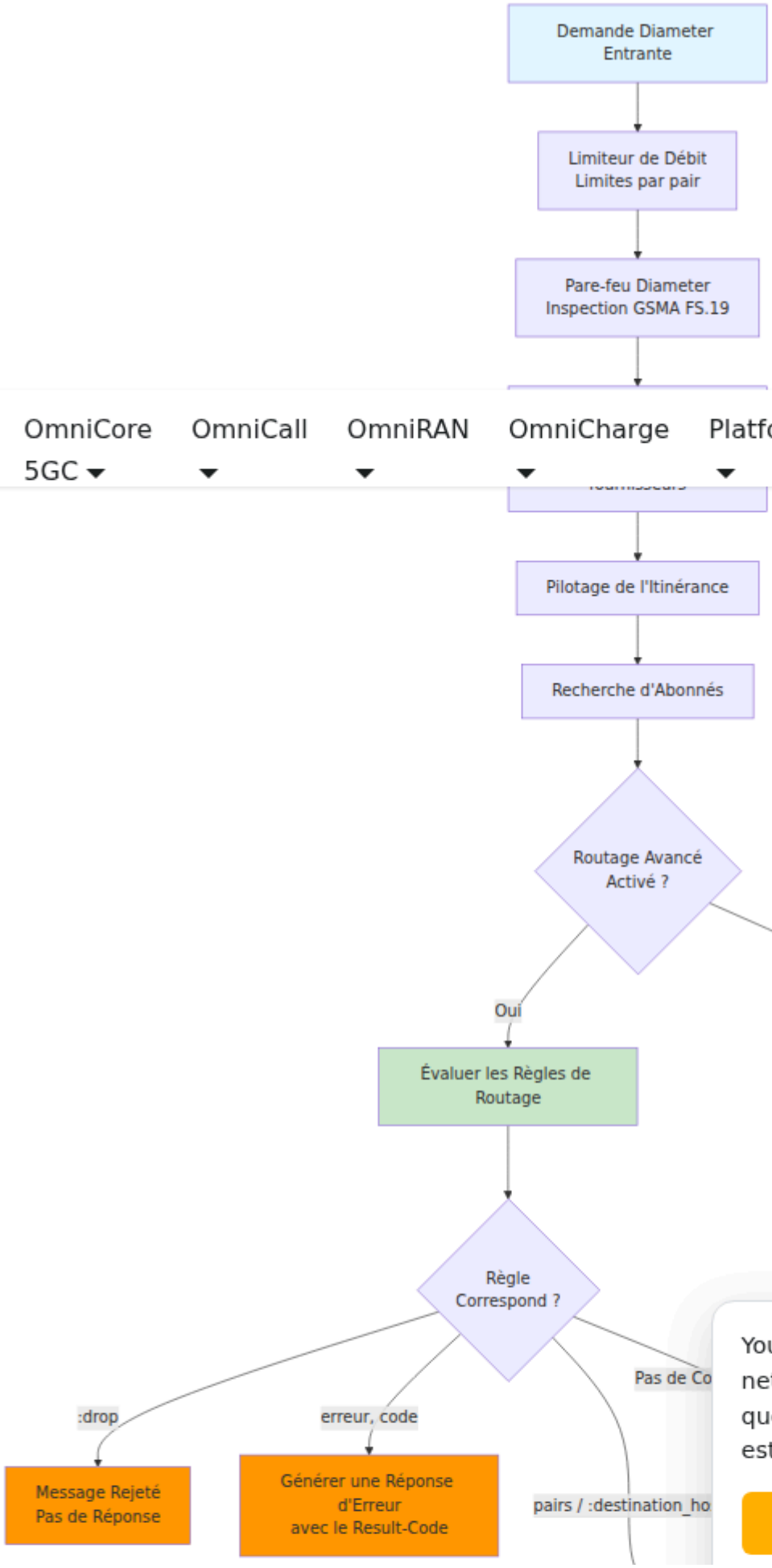
Le DRA relaie deux classes de messages fondamentalement différentes et les traite différemment.

## Demandes

Les demandes traversent l'**pipeline de traitement** complet. Le DRA sélectionne un pair de destination, met à jour les AVPs de routage qu'il gère, et transmet le message. Le DRA est un relai, donc il passe la charge utile de l'application sans changement.

## Réponses

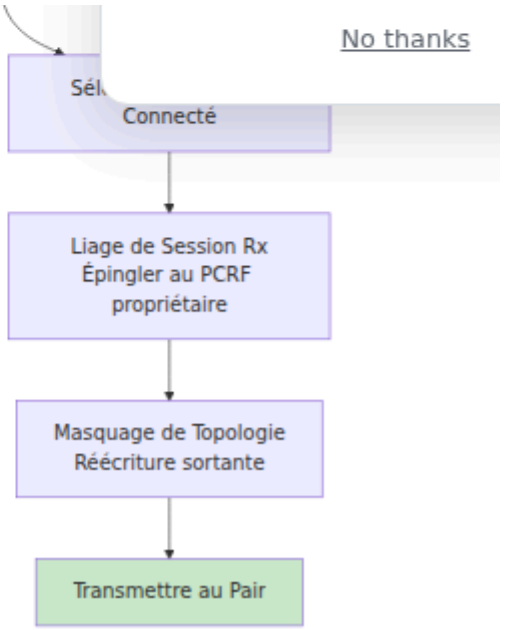
Les réponses ne réintègrent jamais le pipeline de routage. Au lieu de cela, le DRA corrèle chaque réponse à sa demande d'origine. Le DRA retourne ensuite la réponse le long du chemin inverse.




OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français

You have looked at 39 of
network functions. Answe
questions and we send y
estimate.

Get an estima




Les règles de relais pour les réponses sont :

- **Routage basé sur la session** - Les réponses suivent toujours le chemin inverse de leur demande.
- **Préservation de l'ID de bout en bout** - L'**Identifiant de Bout en Bout** reste inchangé à chaque saut.
- **Routage Hop-by-Hop** - L'**Identifiant Hop-by-Hop** change à chaque saut. Le DRA l'utilise pour maintenir l'état de routage et pour faire correspondre une réponse au pair qui a envoyé la demande.
- **Pas d'évaluation des règles** - Le DRA n'évalue pas les règles de routage ni n'inspecte le contenu des AVP pour les réponses.
- **Corrélation avec état** - Les tables de routage internes suivent quel pair a envoyé chaque demande. Le DRA nettoie cet état après avoir relayé la réponse.

Le routage des réponses est déterministe. Le protocole Diameter exige qu'une réponse retourne au pair d'origine le long du chemin de demande établi. Cela préserve la gestion correcte des sessions et empêche les boucles de routage. Voir RFC 6733 §6.2 pour les détails du routage des messages de réponse.

---

# Tables de référence

## Identifiants d'application 3GPP courants

| ID | Interface | Description | Référence |
|---|---|---|---|
| 16777251 | S6a/S6d | Authentification et abonnement MME/SGSN ↔ HSS | 3GPP TS 29.272 |
| 16777238 | Gx | Contrôle de politique et de facturation PCEF ↔ PCRF | 3GPP TS 29.212 |
| 4 | Gy | Facturation en ligne (contrôle de crédit) ↔ OCS | 3GPP TS 32.299 |

## AVPs de routage courants

| Code | Nom | Description | Référence |
|---|---|---|---|
| 293 | Destination-Host | Cible de routage explicite au niveau de l'hôte (priorité la plus élevée) | RFC 6733 §6.5 |
| 283 | Destination-Realm | Domaine utilisé pour sélectionner un pair connecté | RFC 6733 §6.4 |
| 282 | Route-Record | Enregistre l'identité des relais qu'un message a traversés | RFC 6733 §6.7.1 |

---

# Lectures complémentaires

- **Routage Standard** - Routage prioritaire RFC 6733, sélection des pairs et routage des réponses en détail.

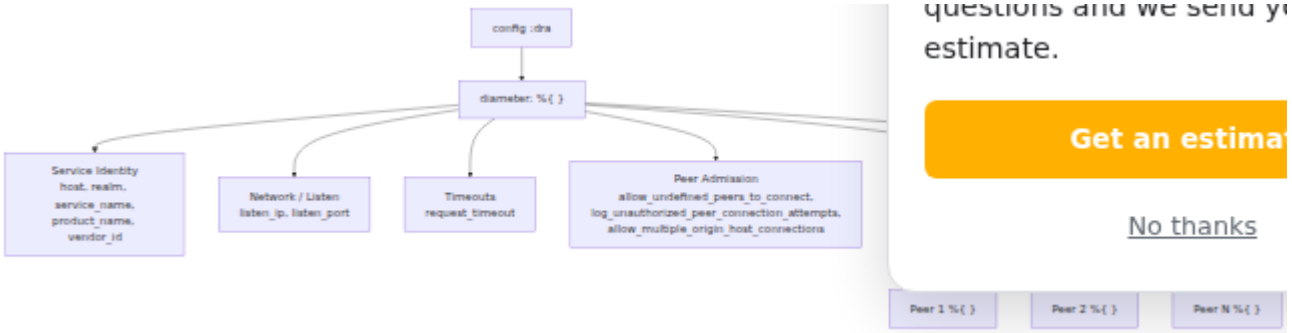
- **Routage Avancé** - Routage basé sur des règles, résultats de rejet/d'erreur, et ciblage des pairs.
- **Configuration** - Configuration de base du DRA, transports et paramètres des modules.
- **Sécurité** - Limitation de débit, pare-feu Diameter GSMA FS.19, désinfection des AVP et masquage de topologie.

# Configuration de Base DRA

La **Configuration de Base DRA** définit l'identité de l'OmniDRA, les écouteurs réseau, les délais d'attente, la politique d'admission des pairs et la liste des pairs Diameter auxquels il se connecte. Ce bloc est la fondation de chaque décision de routage que l'agent prend. Déclarez-le sous `config :dra, diameter: %{...}` dans `config/runtime.exs`.

## Aperçu

Tous les paramètres de base se trouvent dans une seule carte `diameter`. La carte contient un ensemble de clés de premier niveau qui décrivent le DRA lui-même. Elle contient également une liste `peers`. Chaque entrée de la liste décrit un pair Diameter.



# Structure de Configuration

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: "127.0.0.2",
    listen_port: 3868,
    host: "dra01",
    realm: "production.omnitouch",
    product_name: "OmniDRA",
    vendor_id: 10415,
    request_timeout: 5000,
    allow_undefined_peers_to_connect: false,
    log_unauthorized_peer_connection_attempts: true,
    allow_multiple_origin_host_connections: false,
    peer_selection_algorithm: :random,
    peers: [
      # Configurations des pairs...
    ]
  }
```

# Paramètres de Premier Niveau

Ces clés configurent l'identité propre du DRA, son écouteur réseau, les délais d'attente, ainsi que sa politique d'admission et de sélection des pairs. Chaque clé apparaît une fois dans la carte `diameter`.

| Paramètre | Type | Requis | Par défau |
| --- | --- | --- | --- |
| `service_name` | Atom | Oui | - |
| `listen_ip` | String ou List | Oui | - |
| `listen_port` | Integer | Oui | - |
| `host` | String | Oui | - |
| `realm` | String | Oui | - |
| `product_name` | String | Oui | - |

| Paramètre | Type | Requis | Par défau |
| --- | --- | --- | --- |
| `vendor_id` | Integer | Oui | - |
| `request_timeout` | Integer | Non | `5000` |
| `allow_undefined_peers_to_connect` | Boolean | Oui | `false` |
| `log_unauthorized_peer_connection_attempts` | Boolean | Oui | `true` |
| `allow_multiple_origin_host_connections` | Boolean | Oui | `false` |

| Paramètre | Type | Requis | Par défau |
| --- | --- | --- | --- |
| | | | |
| `peer_selection_algorithm` | Atom | Oui | - |
| `peer_settle_time_ms` | Integer | Non | `0` |
| `watchdog_timer_ms` | Integer | Non | `30000` |
| `peers` | List | Oui | - |

# Taille du Tampon de Socket SCTP

La taille du tampon de socket est une clé **de premier niveau** `:dra`. Configurez-la avec `config :dra, diameter_socket_buffer_size:`. Ce n'est **pas** un membre de la carte `diameter:`.

```
config :dra, diameter_socket_buffer_size: 4_194_304

config :dra,
  diameter: %{
    # ... configuration de base ...
  }
```

| Paramètre | Type | Requis | Par défaut | D |
|---|---|---|---|---|
| `diameter_socket_buffer_size` | Integer | Non | `4194304` (4 Mo) | Taille du tar réception/e en octets. U **niveau** `:dr` `diameter_s` pas un mer `diameter:` . pairs `:diam` Multihomin |

# Sélection de Pairs

Lorsque plus d'un pair configuré correspond aux critères de routage pour une requête, `peer_selection_algorithm` détermine quel pair la reçoit. Ce paramètre affecte la manière dont le DRA distribue le trafic et son comportement lors des pannes de pairs. Voir Routage Standard pour le flux complet de décision de routage. Voir Routage Avancé pour les remplacements basés sur des règles.

| Valeur | Comportement |
| --- | --- |
| `:random` | Le DRA choisit un pair correspondant au hasard pour chaque requête. Cela répartit la charge entre tous les pairs éligibles. |
| `:failover` | Le DRA préfère le premier pair correspondant. Il utilise les pairs suivants uniquement lorsque les pairs de plus haute priorité ne sont pas disponibles. |

# Configuration des Pairs

Chaque entrée de la liste `peers` est une carte qui décrit un seul pair Diameter. La carte définit son identité, son adresse réseau, le transport qu'il utilise, et si le DRA compose le pair ou attend d'être composé.

```
%{
  host: "diameter-benchmark-server.production.omnitouch",
  realm: "production.omnitouch",
  ip: "127.0.0.3",
  port: 3868,
  transport: :diameter_tcp,
  initiate_connection: true
}
```

## Paramètres Par Pair

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `host` | String | Oui | - | L'Identité Diameter du pair (FQDN). Cette valeur doi correspondre exactement et est sensible à la casse. Les règles de routage et de Routage Avancé nécessitent une correspondance exacte. |
| `realm` | String | Oui | - | Le realm Diameter du pair. |
| `ip` | String | Oui | - | L'adresse IP principale du pair utilisée pou la connexion. |
| `additional_ips` | List | Non | - | Liste d'adresses IP pour une association SCTP multi-homée. Lorsqu'elle n'est pas définie, le DRA utilise `ip`. Cette clé |

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| | | | | s'applique uniquement avec `transport: :diameter_sctp`. Voir SCTP Multihoming. |
| `port` | Integer | Oui | - | Le port Diameter du pair (typiquement `3868`). |
| `transport` | Atom | Oui | - | Protocole de transport pour la connexion : `:diameter_tcp` pour TCP, ou `:diameter_sctp` pour SCTP. Voir la référence Transports ci-dessous. |
| `source_ip` | String | Non | `listen_ip` | Adresse IP locale à lier lorsque le DRA compose ce pair (connexions sortantes uniquement). Par défaut, cela correspond à l'`listen_ip` du service. |

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| `source_port` | Integer | Non | - | Port local à lier lorsque le DRA compose ce pair (connexions sortantes uniquement). Lorsqu'il n'est pas défini, le système d'exploitation attribue un port éphémère. Si deux pairs partagent la même adresse IP et le même port de destination, définissez une valeur explicite. Chaque connexion utilise alors un socket local distinct. |
| `initiate_connection` | Boolean | Oui | - | Lorsque `true`, le DRA se connecte au pair (rôle client). Lorsque `false`, le DRA attend que le pair se connecte (rôle serveur). Voir |

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| | | | | Modes de Connexion. |
| `origin_host` | String | Non | service `host.realm` | L'`Origin-Host` que le DRA présente **à** ce pair. Cette clé est distincte de `host` ci-dessus, qui est l'identité propre de ce pair. Définissez cette clé lorsqu'un pair s'attend à ce que le DRA utilise un hôte spécifique (par exemple un realm partenaire). Le DRA applique le remplacement à la `CER`/`CEA` et aux messages de protocole de base sur cette connexion. Voir Identité Par Pair. |
| `origin_realm` | String | Non | service `realm` | L'`Origin-Realm` que le DRA présente **à** ce pair. Cette clé est distincte de `realm` ci-dessus |

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| | | | | qui est le realm propre de ce pair. Définissez cette clé avec `origin_host` ou seule. Voir Identité Par Pair. |

> **SCTP multihoming :** un pair qui utilise `:diameter_sctp` peut lier plusieurs adresses locales et distantes pour la résilience. SCTP Multihoming décrit les paramètres de multi-homing et les listes d'adresses par pair.

## Transports

| Valeur de transport | Protocole | Référence |
| --- | --- | --- |
| `:diameter_tcp` | TCP | RFC 6733 (Diameter Base) |
| `:diameter_sctp` | SCTP | RFC 4960 (SCTP) - voir SCTP Multihoming |

## Modes de Connexion

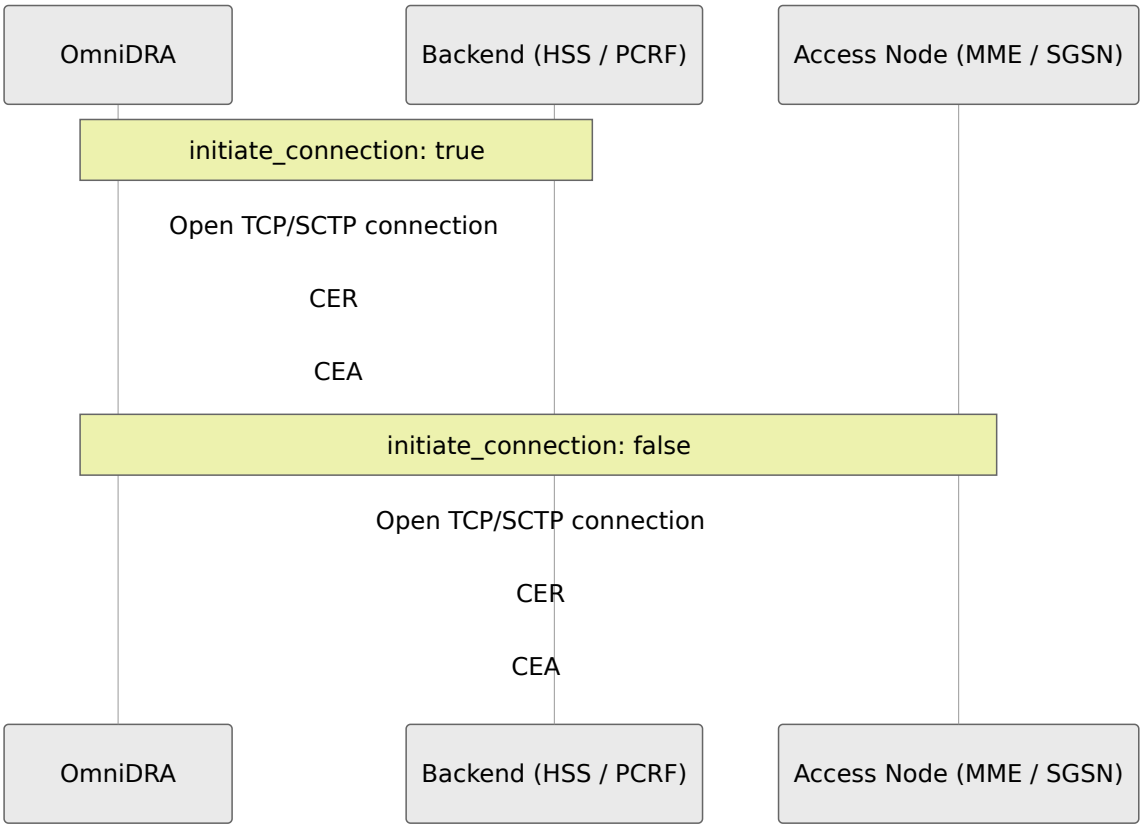
La clé `initiate_connection` contrôle si le DRA compose un pair ou attend d'être composé.

### Initier la Connexion (`initiate_connection: true`)

- Le DRA agit en tant que client Diameter et initie la connexion TCP/SCTP au pair.
- Utilisez ce mode pour des systèmes backend tels qu'un HSS ou PCRF.
- Si le pair est inaccessible, le DRA réessaie la connexion.

**Accepter la Connexion (`initiate_connection: false`)**

- Le DRA agit en tant que serveur Diameter et attend que le pair se connecte.
- Utilisez ce mode pour des nœuds côté accès tels qu'un MME, SGSN ou P-GW.
- Le pair doit être présent dans `peers`, sauf si vous définissez `allow_undefined_peers_to_connect` sur `true`.
- Le DRA différencie les pairs entrants par l'`Origin-Host` qu'ils annoncent dans le CER. Deux nœuds Diameter peuvent partager une même adresse IP source. Le DRA les associe indépendamment lorsque chacun présente un `host` distinct.




# Identité Par Pair

Le DRA a une identité unique pour l'ensemble du service. Il combine `host` et `realm` en l'`Origin-Host` (par exemple `dra01.production.omnitouch`) et annonce `realm` comme l'`Origin-Realm`. Par défaut, le DRA présente cette identité à chaque pair.

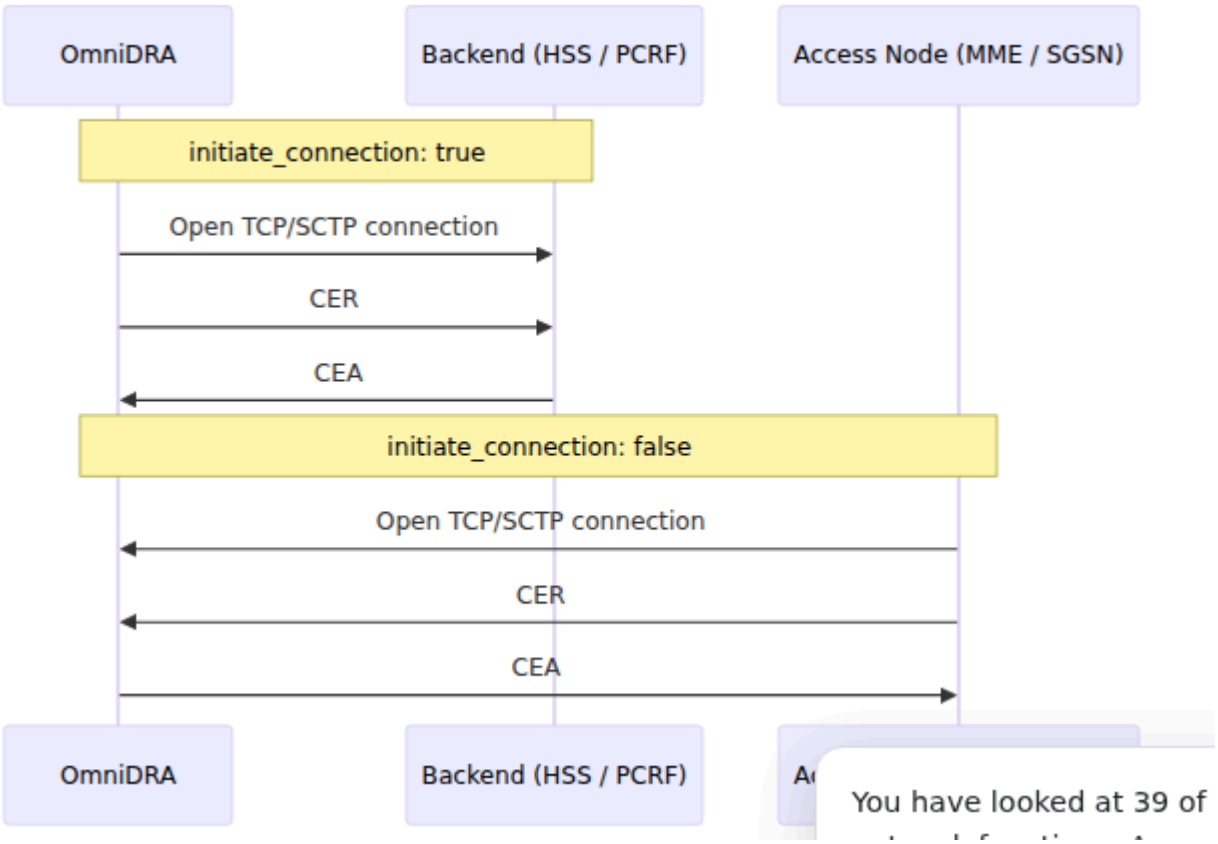
Certains pairs s'attendent à ce que le DRA présente une identité différente. Un partenaire d'interconnexion ou de roaming exige souvent que le DRA utilise un hôte et un realm provenant du schéma de nommage du partenaire. Définissez `origin_host` et `origin_realm` sur ce pair pour changer l'identité que le DRA présente sur cette connexion.

Ces deux clés sont l'identité que le DRA présente **au** pair. Elles sont distinctes de l'`host` et du `realm` propres au pair, qui sont l'identité que le pair annonce **au** DRA. Le DRA applique le remplacement uniquement à cette connexion. Chaque autre pair continue de voir l'identité pour l'ensemble du service.

```
peers: [
  %{
    # Identité propre de ce pair (ce qu'il annonce au DRA) :
    host: "hss.partner.example",
    realm: "partner.example",
    ip: "10.0.0.9",
    port: 3868,
    transport: :diameter_sctp,
    initiate_connection: true,
    # L'identité que le DRA présente À ce pair :
    origin_host: "dra-partner.epc.mnc001.mcc001.3gppnetwork.org",
    origin_realm: "epc.mnc001.mcc001.3gppnetwork.org"
  }
]
```

Voir Paramètres Par Pair pour les définitions des paramètres `origin_host` et `origin_realm`.

Le diagramme ci-dessous montre le DRA présentant son identité par défaut à un pair et une identité de remplacement à un pair partenaire.




## Comportement

Le remplacement change l'identité dans l'Échange de Capacités (`CER` / `CEA`) sur cette connexion. Il change également l'identité dans les messages de protocole de base sur cette connexion : les messages de surveillance (`DWR` / `DWA`) et les messages de déconnexion (`DPR` / `DPA`).

Vous pouvez définir l'une ou l'autre clé seule. Une clé non définie revient à la valeur par défaut du service. Si vous définissez uniquement `origin_realm`, le DRA change uniquement le realm et conserve l'`Origin-Host` pour l'ensemble du service.

Le DRA utilise l'`Origin-Host` de remplacement pour la détection de boucle sur cette connexion. Si une requête porte déjà un AVP `Route-Record` qui correspond au remplacement, le DRA détecte toujours la boucle selon RFC 6733 §6.1.8.

Le DRA fixe l'identité lorsqu'il envoie le `CER`. Si vous changez un remplacement et rechargez la configuration à l'exécution, le DRA rétablit cette connexion afin

que la nouvelle identité prenne effet. Les connexions à d'autres pairs ne sont pas affectées.

# Exemple Complet

L'exemple suivant configure un DRA. Le DRA accepte les connexions d'un MME via SCTP. Il compose un HSS et un PCRF via TCP.

```
config :dra,
  diameter: %{
    service_name: :mvno_dra,
    listen_ip: "10.100.1.10",
    listen_port: 3868,
    host: "dra01",
    realm: "mvno.example.com",
    product_name: "OmniDRA",
    vendor_id: 10415,
    request_timeout: 5000,
    allow_undefined_peers_to_connect: false,
    log_unauthorized_peer_connection_attempts: true,
    allow_multiple_origin_host_connections: false,
    peer_selection_algorithm: :random,
    peers: [
      # MME - DRA 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,
        initiate_connection: false
      },
      # HSS - DRA initie la connexion à partir d'un port source
local explicite
      %{
        host: "hss01.mvno.example.com",
        realm: "mvno.example.com",
        ip: "10.100.3.141",
        port: 3868,
        source_port: 13868,
        transport: :diameter_tcp,
        initiate_connection: true
      },
      # PCRF - DRA initie la connexion
      %{
        host: "pcrf01.mvno.example.com",
        realm: "mvno.example.com",
        ip: "10.100.3.22",
        port: 3868,
        transport: :diameter_tcp,
        initiate_connection: true
```

```
      }
    ]
  }
```

# Notes Opérationnelles

- **Correspondance des Noms d'Hôte** - Le DRA correspond exactement et de manière sensible à la casse les valeurs `host` des pairs. Les règles dans Routage Avancé doivent faire référence à la même chaîne `host` que vous configurez ici.
- **Échange de Capacités** - Lors de la connexion, le DRA et chaque pair échangent les applications et identités prises en charge via CER/CEA. Le DRA annonce les valeurs `host`, `realm`, `product_name`, et `vendor_id` à ce stade.
- **Vendor-ID 10415** - C'est le `Vendor-Id` standard 3GPP. C'est la valeur attendue pour les interfaces Diameter 3GPP.
- **Sélection de Pairs** - Lorsque plusieurs pairs correspondent à une requête, `peer_selection_algorithm` décide lequel le DRA utilise. Voir Routage Standard.

# Considérations de Sécurité

- Définissez `allow_undefined_peers_to_connect: false` en production afin que seuls les pairs connus puissent se connecter.
- Activez `log_unauthorized_peer_connection_attempts: true` pour enregistrer les tentatives de connexion rejetées pour le suivi de la sécurité.
- Assurez-vous que les règles de pare-feu correspondent à l'`listen_ip` et au `listen_port` configurés.

# Documentation Connexe

- SCTP Multihoming - Transport SCTP multi-homé et configuration des adresses par pair.

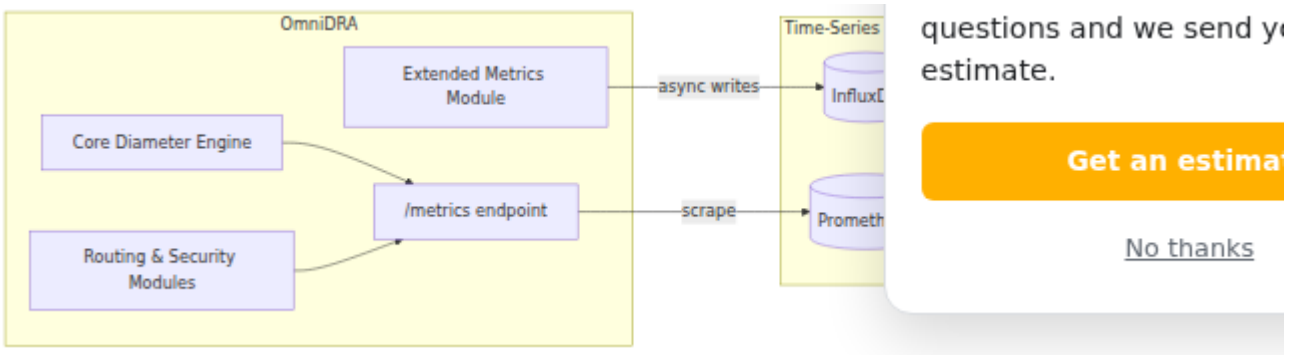
- Routage Standard - Comment le DRA sélectionne les pairs et route les requêtes, y compris le comportement de `peer_selection_algorithm`.
- Routage Avancé - Remplacements de routage basés sur des règles clés sur les valeurs `host` des pairs.

# Métriques & Surveillance

OmniDRA expose des métriques **Prometheus** complètes. Elles couvrent la connectivité des pairs, les taux de messages, les résultats de succès et d'erreur, la latence de réponse, le trafic abandonné et les tentatives de connexion non autorisées. Cette page documente chaque métrique, ses étiquettes et des requêtes PromQL prêtes à l'emploi. Utilisez-les pour créer des tableaux de bord et des alertes.

## Flux d'Observabilité

OmniDRA publie toutes les métriques au format d'exposition Prometheus à l'endpoint `/metrics`. Prometheus interroge cet endpoint à intervalles réguliers et stocke les échantillons. Les tableaux de bord ou les outils d'alerte interrogent ensuite la série temporelle résultante.




Prometheus interroge les **métriques standard de Prometheus** à partir de `/metrics`. Le **Module de Métriques Étendues** publie des télémetries de niveau abonné à haute cardinalité directement dans **InfluxDB**. Cela maintient le nombre de séries Prometheus gérable (voir Module de Métriques Étendues).

> OmniDRA expose également des **métriques d'exécution BEAM/Erlang** sur `/metrics`. Cette page ne les documente pas. Tous les compteurs sont cumulés. Interrogez-les avec `rate()` pour obtenir des taux par seconde.

---

# Métriques de Base Diameter

Ces métriques sont toujours disponibles. Elles décrivent la santé des pairs et le flux de messages à travers le DRA.

## Connectivité des Pairs

#### Statut des Pairs

- **Métrique**: `diameter_peer_status`
- **Type**: Gauge
- **Description**: Indique si le pair est connecté (`1`) ou non (`0`).

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `origin_host` | Identité Diameter du pair (FQDN). |
| `ip` | Adresse IP du pair. |

**Exemples de requêtes**:

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

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

Pour diagnostiquer les pairs qui fluctuent ou qui n'atteignent jamais un état connecté, voir Dépannage.

---

#### Applications Supportées

- **Métrique**: `diameter_peer_supported_applications`
- **Type**: Gauge

- **Description**: Les applications qu'un pair a annoncées lors de l'échange de capacités (CER/CEA), lorsqu'elles sont connues.

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `origin_host` | Identité Diameter du pair (FQDN). |
| `supported_applications` | Les applications que le pair a annoncées comme supportées. |

**Exemples de requêtes**:

```
# Applications annoncées par un pair spécifique
diameter_peer_supported_applications{origin_host="hss01.example.com"}
```

---

# Taux de Messages

### Compte de Messages

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

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `origin_host` | Identité Diameter du pair. |
| `received_from` | Pair qui a envoyé le message. |
| `application_id` | Identifiant d'application Diameter. Voir Identifiants d'Application. |
| `cmd_code` | Code de commande Diameter. Voir Codes de Commande. |
| `application_name` | Nom de l'application lisible par l'homme (ex. `3GPP_S6a`). |
| `cmd_name` | Nom de la commande lisible par l'homme (ex. `AIR`). |
| `direction` | `request` ou `response`. |

**Exemples de requêtes**:

```
# Taux de requêtes 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]))
```

---

# Taux de Succès & d'Erreur

### Codes de Résultat de Réponse

- **Métrique**: `diameter_peer_message_result_code_count_total`

- **Type**: Counter
- **Description**: Nombre total de réponses Diameter par code de résultat.

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `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. |

**Exemples de requêtes**:

```
# Taux de succès pour les requêtes 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 Communs**:

| Code | Nom | Référence |
|---|---|---|
| `2001` | DIAMETER_SUCCESS | RFC 6733 §7.1.1 |
| `3002` | DIAMETER_UNABLE_TO_DELIVER | RFC 6733 §7.1.3 |
| `3003` | DIAMETER_REALM_NOT_SERVED | RFC 6733 §7.1.3 |
| `3004` | DIAMETER_TOO_BUSY | RFC 6733 §7.1.3 |
| `5001` | DIAMETER_AUTHENTICATION_REJECTED | RFC 6733 §7.1.5 |
| `5004` | DIAMETER_INVALID_AVP_VALUE | RFC 6733 §7.1.5 |
| `5012` | DIAMETER_UNABLE_TO_COMPLY | RFC 6733 §7.1.5 |

---

# Latence

### Délai de Réponse

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

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `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. |

**Exemples de requêtes**:

```
# Délai de réponse le plus récent depuis HSS
diameter_peer_last_response_delay{routed_to="hss01.example.com"}

# Délai de réponse le plus élevé récemment parmi les pairs S6a
max(diameter_peer_last_response_delay{application_name="3GPP_S6a"})
```

> Ceci est un gauge qui conserve le dernier délai par pair. Ce n'est pas un histogramme, donc les fonctions de quantile telles que `histogram_quantile` ne s'appliquent pas.

---

# Requêtes Non Répondues

### Compte de Requêtes Non Répondues

- **Métrique**: `diameter_peer_unanswered_request_count_total`
- **Type**: Counter
- **Description**: Requêtes que le DRA a envoyées mais que le pair n'a pas répondu dans le délai imparti.

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `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. |

**Exemples de requêtes**:

```
# Taux de requêtes 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])
))
```

---

# Connexions Non Autorisées

### Tentatives de Connexion Non Autorisées

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

**Étiquettes**:

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

**Exemples de requêtes**:

```
# 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
```

Pour des détails plus approfondis sur les métriques du module de sécurité (pare-feu, limitation de taux, masquage de topologie, assainissement AVP), voir Sécurité.

---

# Métriques du Module de Routage Avancé

Ces métriques décrivent les résultats que produisent les règles de routage avancées. Voir la documentation du module de sécurité dans Sécurité pour la télémetrie de contrôle de trafic associée.

## Trafic Abandonné

### Compte d'Abandon

- **Métrique**: `diameter_advanced_routing_drop_count_total`
- **Type**: Counter

- **Description**: Requêtes que l'action `:drop` du routage avancé a abandonnées. Le DRA n'envoie pas de réponse.

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `application_id` | Identifiant d'application Diameter. |
| `cmd_code` | Code de commande Diameter. |
| `application_name` | Nom de l'application. |
| `cmd_name` | Nom de la commande. |

**Exemples de requêtes**:

```
# Taux d'abandon par commande
sum by (cmd_name) (
  rate(diameter_advanced_routing_drop_count_total[5m])
)

# Trafic total abandonné
sum(rate(diameter_advanced_routing_drop_count_total[5m]))
```

**Lorsque le DRA abandonne le trafic**:

- Une règle de routage avancée correspond avec une action **drop**.
- Le DRA rejette le message.
- Le DRA n'envoie aucun message de réponse au pair demandeur.

---

# Limitation de Taux & Surcharge

### Réponses d'Erreur

- **Métrique**: `diameter_advanced_routing_error_count_total`

- **Type**: Counter
- **Description**: Réponses d'erreur générées par l'action d'erreur du routage avancé.

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `result_code` | Code de résultat Diameter envoyé dans la réponse d'erreur. |
| `application_id` | Identifiant d'application Diameter. |
| `cmd_code` | Code de commande Diameter. |
| `application_name` | Nom de l'application. |
| `cmd_name` | Nom de la commande. |

**Exemples de requêtes**:

```
# Réponses d'erreur par code de résultat
sum by (result_code) (
  rate(diameter_advanced_routing_error_count_total[5m])
)

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

# Détection de limitation de taux ou de surcharge
diameter_advanced_routing_error_count_total{
  result_code="3004"
} > 100
```

**Lorsque le DRA génère des réponses d'erreur**:

- Une règle de routage avancée correspond avec une action **error** pour un Code de Résultat spécifié.
- Le DRA génère une réponse Diameter avec le Code de Résultat spécifié.
- Le DRA renvoie la réponse au pair demandeur. Il ne transmet pas la réponse.

**Codes d'Erreur Communs Utilisés**:

| Code | Nom | Signification |
|---|---|---|
| `3002` | DIAMETER_UNABLE_TO_DELIVER | Échec de routage. |
| `3003` | DIAMETER_REALM_NOT_SERVED | Domaine non supporté. |
| `3004` | DIAMETER_TOO_BUSY | Protection contre la surcharge. |
| `5012` | DIAMETER_UNABLE_TO_COMPLY | Erreur générale. |

---

# Module de Métriques Étendues

Le **Module de Métriques Étendues** fournit une télémetrie et des analyses avancées. Utilisez-le pour analyser les modèles de trafic Diameter au-delà des métriques standard. Il suit actuellement les tentatives de connexion des abonnés LTE. Il corrèle les paires de messages S6a Authentication Information Request (AIR) et Answer (AIA).

Ces métriques portent des champs à **haute cardinalité** tels que l'IMSI. Le module les écrit donc de manière asynchrone dans **InfluxDB** et ne les expose pas sur l'endpoint `/metrics` de Prometheus. Cela maintient le nombre de séries Prometheus limité et évite l'explosion des métriques.

## Configuration

Le module est désactivé par défaut. Configurez-le sous `config/runtime.exs`:

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

| Paramètre | Type | Requis | Par défaut | Descr |
|---|---|---|---|---|
| `enabled` | Boolean | Non | `false` | Défini `true` active Modul Métric Étend Lorsqu `false` DRA contou modul |
| `attach_attempt_reporting_enabled` | Boolean | Non | `false` | Active suivi e report des tentat de conne LTE (S AIR/AI |

## Connexion InfluxDB

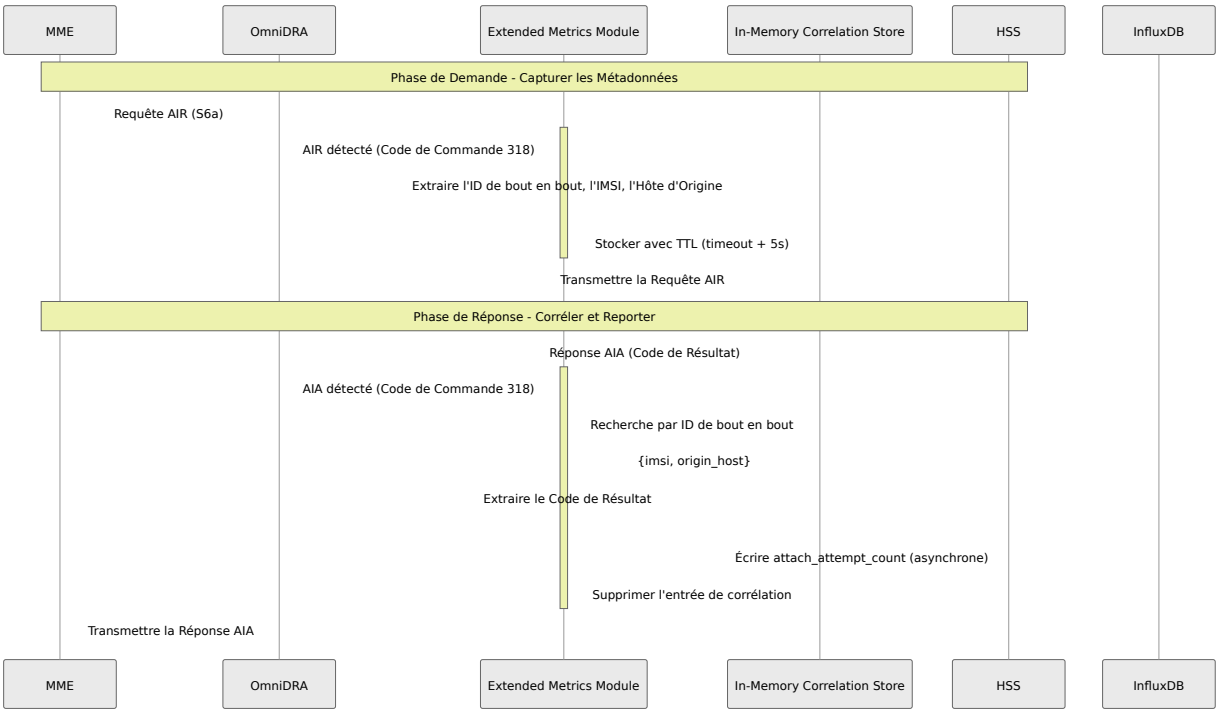
Le Module de Métriques Étendues écrit de manière asynchrone dans InfluxDB via une connexion Instream (Instream). Configurez la cible sous `config :dra, DRA.Metrics.InfluxDB`.

```
config :dra, DRA.Metrics.InfluxDB,
  auth: [username: "example_user", password: "example_password"],
  database: "dra",
  host: "127.0.0.15",
  port_udp: 8089,
  writer: Instream.Writer.Line
```

| Paramètre | Type | Requis | Par défaut | Descriptio |
|---|---|---|---|---|
| `auth` | Liste de mots-clés | Non | - | Identifiants Influ sous forme de `[username: ... password: ...]` |
| `database` | String | Oui | - | Base de donnée InfluxDB cible qu reçoit les mesur métriques. |
| `host` | String | Oui | - | Nom d'hôte ou a IP du serveur Inf |
| `port_udp` | Integer | Non | `8089` | Port UDP qui écr données au forn ligne dans Influx Utilisez le port d'écoute UDP d'InfluxDB (`8089` défaut), pas le p l'API HTTP (`8086` |
| `writer` | Module | Non | `Instream.Writer.Line` | Module d'écritur Instream. `Instream.Write` écrit avec le pro de ligne. |

# Suivi des Tentatives de Connexion

Le module surveille les paires S6a AIR/AIA. Il corrèle chaque requête avec sa réponse via l'ID de bout en bout Diameter. Il émet ensuite une métrique de niveau abonné. La métrique porte le pair d'origine et le code de résultat HSS.




**Mesure**: `attach_attempt_count`

**Champs**:

| Champ | Description |
|---|---|
| `imsi` | L'IMSI de l'abonné, pris de l'AVP User-Name. |

**Étiquettes**:

| Étiquette | Description |
|---|---|
| `origin_host` | Le pair qui a originairement demandé la connexion. |
| `result_code` | Le code de résultat Diameter de la réponse HSS. |

**Codes de résultat** écrits à cette mesure sont des codes Diameter standard :

| Code | Nom |
|---|---|
| `2001` | DIAMETER_SUCCESS |
| `5001` | DIAMETER_AUTHENTICATION_REJECTED |
| `5004` | Diameter AVP unsupported |

Voir RFC 6733 pour la liste complète des codes 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 individuels ayant des échecs de connexion.
- **Surveillance de la Performance 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.

## Notes Opérationnelles

- Les métriques de tentative de connexion ne suivent que les paires S6a AIR/AIA (Identifiant d'Application `16777251`, Code de Commande `318`).
- Les métadonnées de la requête expirent après le délai de requête configuré plus 5 secondes.
- Le traitement des métriques est asynchrone. Il ne bloque pas le flux de messages.
- Le module fonctionne indépendamment des modules de routage et de transformation.

---

# Tables de Référence

## Référence des Identifiants d'Application

| ID | Application | Description | Référence |
|---|---|---|---|
| `16777251` | S6a/S6d | MME/SGSN ↔ HSS authentification et abonnement | 3GPP TS 29.272 |
| `16777238` | Gx | PCEF ↔ PCRF contrôle de politique et de facturation | 3GPP TS 29.212 |
| `16777216` | Cx | I-CSCF/S-CSCF ↔ HSS IMS enregistrement | 3GPP TS 29.229 |

## Référence des Codes de Commande

| Code | Commande | Application | Référence |
|---|---|---|---|
| `318` | AIR/AIA (Information d'Authentification) | S6a | 3GPP TS 29.272 |

---

# Notes

- Toutes les métriques standard sont disponibles à l'endpoint `/metrics` au format Prometheus.
- Le **Module de Métriques Étendues** fournit des métriques supplémentaires spécifiques à S6a via InfluxDB.
- OmniDRA expose également des **métriques d'exécution BEAM/Erlang**. Cette page ne les documente pas.

- Tous les compteurs sont cumulés. Interrogez-les avec `rate()` pour obtenir des taux par seconde.
- Seul le Module de Métriques Étendues utilise des étiquettes à haute cardinalité telles que l'IMSI. Cela évite l'explosion des métriques.

# Documentation Connexe

- Dépannage - Diagnostiquer les problèmes de connectivité, de routage et de livraison.
- Sécurité - Détails sur les métriques du module de sécurité (pare-feu, limitation de taux, masquage de topologie, assainissement AVP).

# Connectivité des pairs & Temps de stabilisation

Ce module contrôle la rapidité avec laquelle un pair Diameter devient routable après sa connexion. Il empêche également le trafic vers les pairs qui flappent. Un pair flappant se connecte et se déconnecte de manière répétée.

## Contexte

Avant qu'OmniDRA ne route vers un pair, le pair doit compléter l'échange de capacités (CER/CEA) et être promu au service. Par défaut, le RFC 3539 maintient un pair *reconnecté* dans l'état REOPEN. Le pair y reste jusqu'à ce qu'il passe des échanges de surveillance successifs (DWR/DWA). Le minuteur de surveillance **Tw** rythme ces échanges, environ 30 secondes avec la valeur par défaut standard de 30000 ms. Ce maintien s'applique même si le CER/CEA a déjà réussi.

OmniDRA supprime ce délai implicite. La surveillance promeut immédiatement les pairs reconnectés. OmniDRA remplace le délai par un **temps de stabilisation**. Le temps de stabilisation est un maintien explicite et configurable qui s'applique après que le pair se soit connecté. Pendant le maintien, le pair est connecté mais pas routable. S'il se déconnecte dans la fenêtre, OmniDRA annule le maintien et n'envoie aucun trafic vers lui.

## Configuration

Les deux clés sont optionnelles et se trouvent dans la carte de configuration `:diameter`. Si vous ne définissez aucune clé, les pairs sont routés dès qu'ils se connectent.

```
config :dra,
  diameter: %{
    # ...paramètres de service...
    peer_settle_time_ms: 5000,
    watchdog_timer_ms: 30_000
  }
```

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `peer_settle_time_ms` | Entier | Non | `0` | Durée (ms) pendant laquelle un pair nouvellement connecté doit rester actif avant que le DRA ne route le trafic vers lui. `0` route immédiatement. Une valeur positive maintient le pair pendant cette durée. Elle annule le maintien si le pair se déconnecte pendant la fenêtre. |
| `watchdog_timer_ms` | Entier | Non | `30000` | Remplacement pour le minuteur de surveillance RFC 3539 **Tw**, l'intervalle de vivacité DWR/DWA. Diminuez-le pour une détection plus rapide des pairs morts. Cela coûte plus de trafic de surveillance. |

Définissez le temps de stabilisation pas plus haut que l'intervalle de flapping dans le pire des cas que vous devez supporter. Une valeur plus longue augmente la protection contre le flapping. Elle retarde également le retour d'un pair sain après un redémarrage légitime.

# Surveillance

Un pair dans sa fenêtre de stabilisation rapporte **hors ligne** dans le compteur `diameter_peer_status` (`0` = hors ligne/pas encore routable, `1` = routable). Il reste hors ligne jusqu'à ce qu'il devienne routable. La métrique reflète donc ce que le DRA peut router.

Le démarrage produit ces lignes de journal :

| Message de journal | Quand |
|---|---|
| `peer connected, settling before routable` | Pair connecté, fenêtre de stabilisation commencée (inclut `settle_time_ms`). |
| `peer disconnected during settle window; never became routable` | Pair déconnecté avant de se stabiliser. Le DRA n'a routé aucun trafic. |
| `destination peer is settling; request not yet routable...` | Une demande est arrivée pour un pair qui est encore en stabilisation. Le DRA a répondu `DIAMETER_UNABLE_TO_DELIVER (3002)`. |
| `no suitable peer to route request...` | Le DRA ne peut pas router la demande vers une destination qui **n'est pas** en stabilisation (hors ligne, inconnu ou mal configuré). |

Les deux dernières lignes distinguent un maintien transitoire attendu d'une destination réellement manquante.

# Référence

- RFC 3539 - Machine d'état de surveillance Diameter et le minuteur de surveillance (Tw).
- RFC 6733 - Protocole de base Diameter (CER/CEA, DWR/DWA).

# Direction de Roaming (SoR)

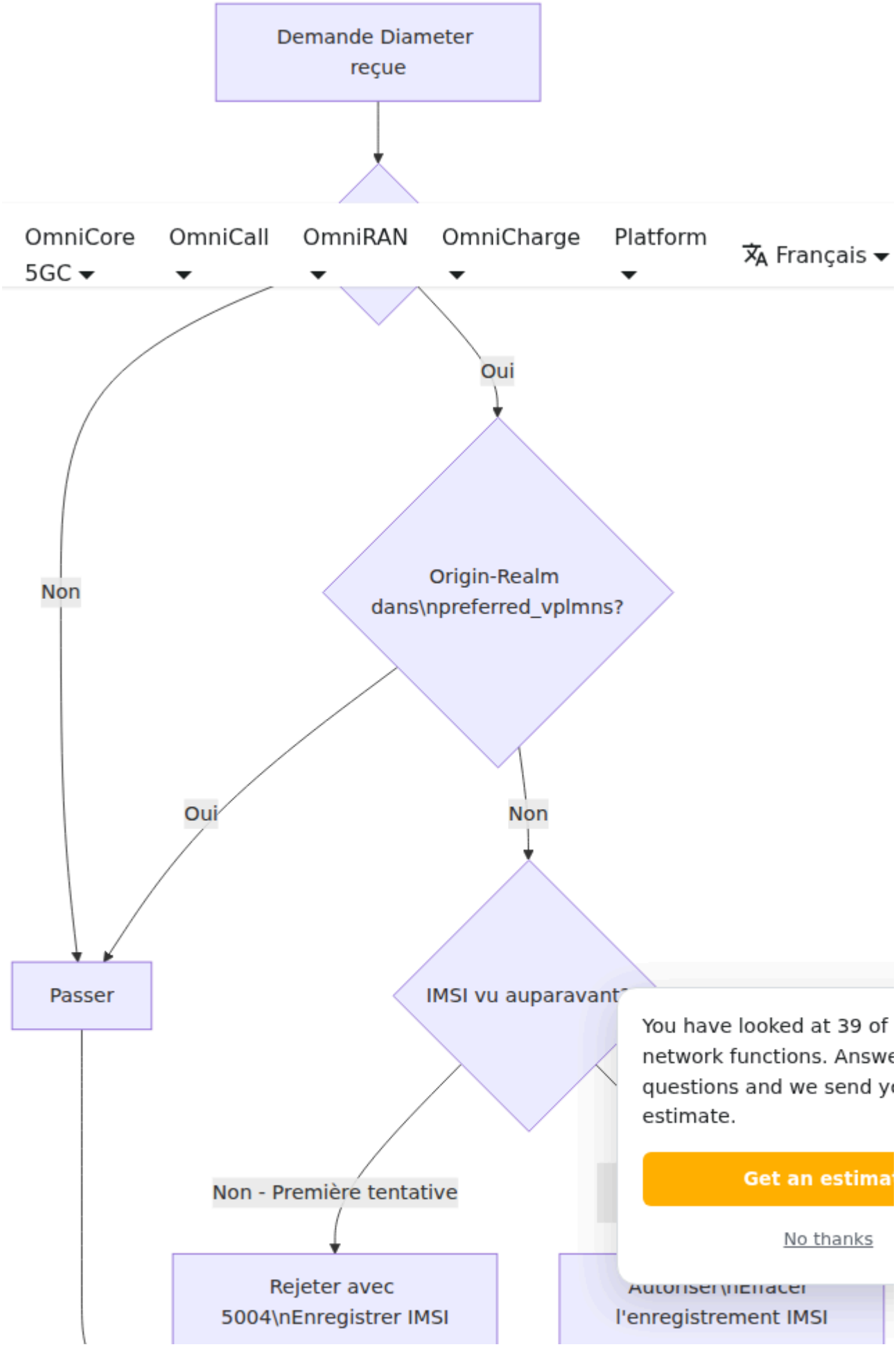
La direction de roaming (SoR) permet au HPLMN d'influencer quel VPLMN un abonné en roaming se connecte. Lorsqu'un abonné essaie de s'enregistrer via un VPLMN non préféré, le DRA intercepte la demande S6a Update-Location-Request (ULR). Il rejette l'ULR avec **DIAMETER_ERROR_ROAMING_NOT_ALLOWED (5004)**. Cela provoque le détachement et le rattachement de l'UE. L'UE sélectionne ensuite un VPLMN préféré s'il est disponible.

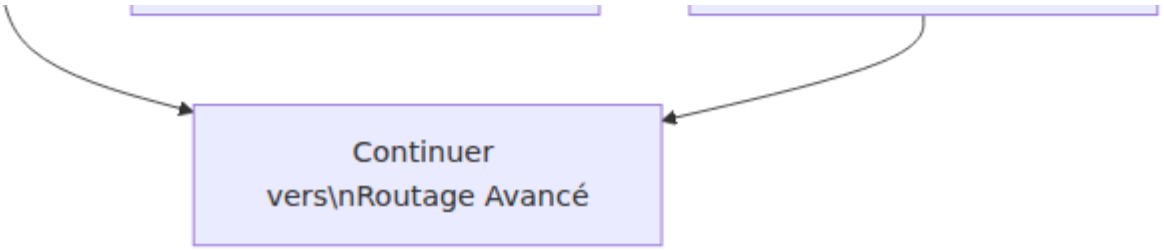
Si la couverture préférée n'est pas disponible et que l'abonné revient au même VPLMN non préféré, le module permet la demande lors de la prochaine tentative.

3GPP TS 29.272 Section 5.2.1.1 et 3GPP TS 23.122 définissent le SoR.

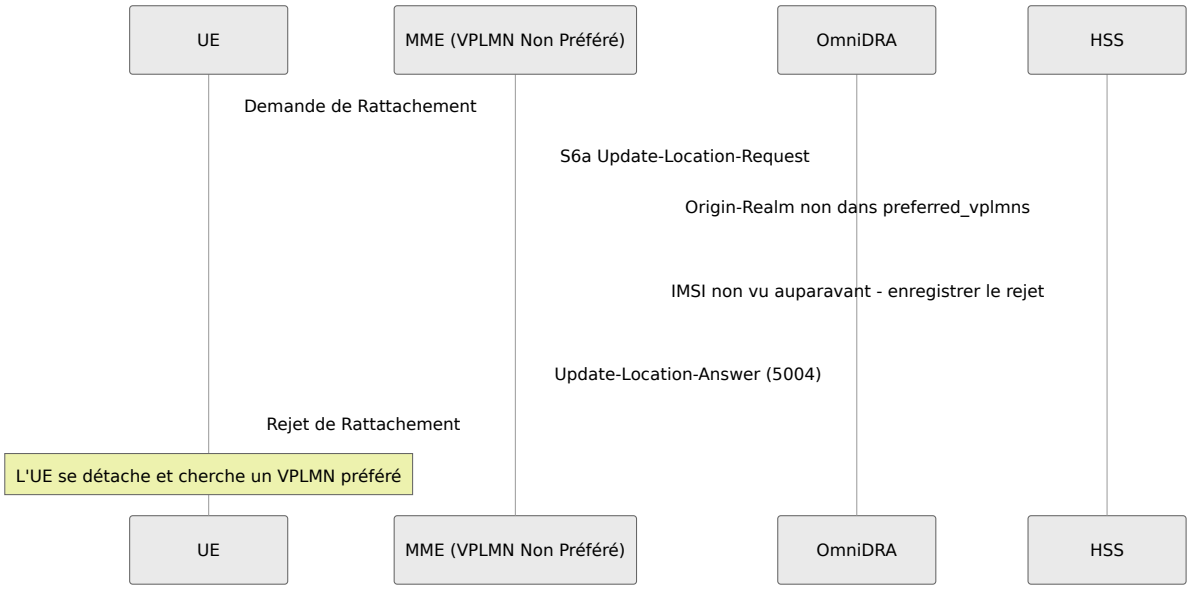
## Comment ça fonctionne

Le module fonctionne dans le pipeline de traitement des demandes DRA. Il s'exécute **avant** le routage avancé. Le module évalue uniquement les S6a Update-Location-Requests (Application-Id `16777251`, Command-Code `316`). Tous les autres messages Diameter passent sans modification.

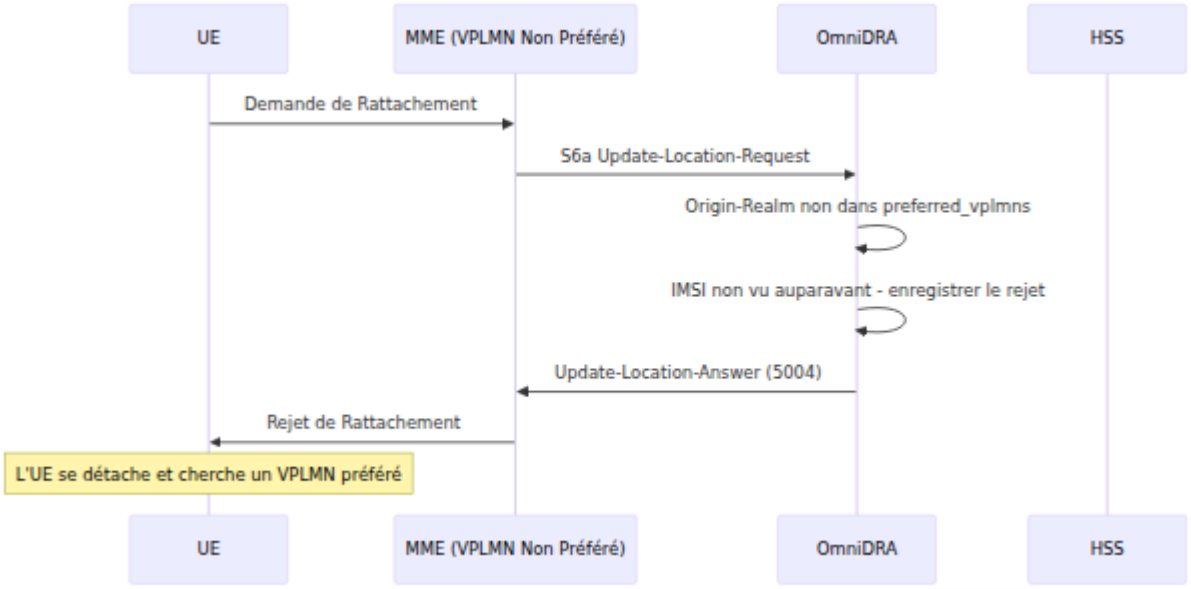

Demande Diameter reçue
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
Oui
Non
Origin-Realm dans\npreferred_vplmns?
Oui
Non
Passer
IMSI vu auparavant?
Non - Première tentative
Rejeter avec 5004\nEnregistrer IMSI
l'enregistrement IMSI
You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks




# Séquence : Première Tentative (Rejetée)




# Séquence : Deuxième Tentative (Autorisée)



# Position dans le Pipeline

La direction de roaming s'exécute **avant** le routage avancé dans le pipeline de traitement des demandes. Si le SoR rejette une demande, ce message n'atteint jamais le routage avancé.




# Suivi des Abonnés

Le module suit chaque abonné rejeté par IMSI. Chaque enregistrement comprend un compteur de rejets et un horodatage. Le module nettoie automatiquement les enregistrements après l'expiration du TTL configuré.

Lorsqu'un IMSI suivi atteint le seuil `max_rejections` et envoie un autre ULR, le module permet la demande. Il supprime ensuite l'enregistrement de suivi.

Si aucun autre ULR n'arrive dans la fenêtre TTL, l'enregistrement expire. Le prochain cycle de nettoyage le supprime. Le module traite alors un ULR ultérieur du même IMSI comme une première tentative et le rejette à nouveau.

# Configuration

Configurez le module sous `module_roaming_steering` dans `config/runtime.exs`.

```
module_roaming_steering: %{
  # Activer ou désactiver le module
  enabled: false,

  # Nombre de rejets avant d'autoriser
  max_rejections: 1,

  # Temps en secondes avant qu'un enregistrement IMSI suivi
n'expire
  rejection_ttl_seconds: 300,

  # Code de résultat Diameter renvoyé dans la réponse de rejet
  rejection_result_code: 5004,

  # Liste des valeurs Origin-Realm considérées comme VPLMNs
préférés
  # Les ULR de ces royaumes sont toujours autorisés
  preferred_vplmns: [
    "epc.mnc001.mcc001.3gppnetwork.org",
    "epc.mnc002.mcc001.3gppnetwork.org"
  ]
}
```

## Paramètres

| Paramètre | Type | Requis | Par défaut | De |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activer ou désactiv de roaming. Lorsqu demandes passent |
| `children` | Liste | Non | `[]` | Processus que le s lorsqu'il est activé. `[DRA.Router.Modu` afin que le process |
| `max_rejections` | Entier | Non | `1` | Nombre de fois à r que le module ne l par défaut de `1`, le premier ULR et aut |
| `rejection_ttl_seconds` | Entier | Non | `300` | Temps en seconde enregistrement IM nettoyage le suppr ne réessaie pas da module traite la pr une première tenta contrôle égalemen nettoyage. |
| `rejection_result_code` | Entier | Non | `5004` | Code de résultat D de rejet. `5004` corr DIAMETER_ERROR_ selon 3GPP TS 29.2 |
| `preferred_vplmns` | Liste | Non | `[]` | Liste de chaînes O les VPLMNs préfére toujours un ULR d' Realm correspond |

# Exemples de Configuration

### Rejeter Une Fois, Puis Autoriser

C'est le comportement par défaut. Le module rejette un abonné sur un VPLMN non préféré une fois. Si l'abonné revient, le module transfère l'ULR au 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"
  ]
}
```

**Comment ça fonctionne :** Un ULR arrive d'un MME dans `epc.mnc003.mcc310.3gppnetwork.org`, qui n'est pas dans la liste préférée. Le module le rejette avec le Code de Résultat 5004. L'UE se détache et essaie de trouver un réseau préféré. Si elle revient au même VPLMN non préféré dans les 300 secondes, le module transfère l'ULR normalement.

**Cas d'utilisation :** Direction de roaming standard. L'opérateur a des accords de roaming préférés avec des réseaux partenaires spécifiques.

### Rejets Multiples Avant Autorisation

Utilisez ceci lorsque l'opérateur souhaite que l'UE effectue plusieurs tentatives avant de revenir à un VPLMN non préféré.

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

**Comment ça fonctionne :** Le module rejette l'abonné trois fois. Il autorise ensuite l'abonné lors de la quatrième tentative. Le TTL s'étend à 600 secondes pour les cycles de réessai supplémentaires.

**Cas d'utilisation :** Zones urbaines denses où une couverture préférée existe. L'UE peut avoir besoin de plusieurs tentatives de rattachement pour l'acquérir.

# Métriques

## Rejets

**Métrique**: `diameter.roaming_steering.reject.count` **Type**: Compteur **Description**: Le DRA incrémente ce compteur chaque fois que le module SoR rejette un ULR. **Étiquettes**:

- `origin_realm` - Origin-Realm du VPLMN non préféré
- `result_code` - Code de résultat Diameter renvoyé dans le rejet
- `imsi` - IMSI de l'abonné rejeté

## Autorisé (VPLMN Préféré)

**Métrique**: `diameter.roaming_steering.allow.count` **Type**: Compteur **Description**: Le DRA incrémente ce compteur lorsque le module autorise un ULR. **Étiquettes**:

- `origin_realm` - Origin-Realm du VPLMN d'origine

- `reason` - Pourquoi le module a autorisé la demande : `preferred_vplmn` ou `max_rejections_reached`
- `imsi` - IMSI de l'abonné. Présent uniquement lorsque la raison est `max_rejections_reached`

## Réponses d'Erreur

**Métrique**: `diameter.roaming_steering.error.count` **Type**: Compteur **Description**: Le DRA incrémente ce compteur lorsque le processeur renvoie un message de réponse d'erreur pour un rejet SoR. **Étiquettes**:

- `result_code` - Code de résultat Diameter
- `application_id` - Application-Id Diameter (numérique)
- `cmd_code` - Command-Code Diameter (numérique)
- `application_name` - Nom d'application lisible par l'homme
- `cmd_name` - Nom de commande lisible par l'homme

# Dépannage

## Abonnés Toujours Rejetés (Jamais Autorisés)

**Symptômes**: Le module rejette systématiquement les abonnés sur des VPLMNs non préférés. Ils ne s'attachent jamais avec succès.

**Causes possibles**:

- `rejection_ttl_seconds` est trop court. L'enregistrement de suivi expire avant que l'abonné ne réessaie.
- `max_rejections` est supérieur au nombre de réessais que l'UE effectue avant d'abandonner.

**Résolution**:

1. Augmentez `rejection_ttl_seconds` pour s'adapter au timing de réessai de l'UE.

2. Confirmez que `max_rejections` est une valeur que l'UE peut atteindre dans son cycle de réessai.

## Abonnés sur VPLMNs Préférés Rejetés

**Symptômes**: Le module rejette les ULR des réseaux partenaires préférés avec 5004.

**Causes possibles**:

- L'Origin-Realm du VPLMN préféré n'est pas dans `preferred_vplmns`.
- La chaîne Origin-Realm ne correspond pas exactement. La correspondance est sensible à la casse.

**Résolution**:

1. Vérifiez l'Origin-Realm du pair dans les journaux DRA au moment de la connexion.
2. Confirmez que la chaîne exacte de l'Origin-Realm est dans la liste `preferred_vplmns`.

## Module Ne Prend Pas Effet

**Symptômes**: Le module transfère les ULR des VPLMNs non préférés sans rejet.

**Causes possibles**:

- `enabled` est réglé sur `false`.
- Le processus du module n'est pas en cours d'exécution. Vérifiez le superviseur.

**Résolution**:

1. Confirmez `enabled: true` dans la configuration.
2. Confirmez que le DRA a redémarré après le changement de configuration.

# Référence

## Codes Diameter

| Code | Nom | Description | Référ |
|---|---|---|---|
| 5004 | DIAMETER_ERROR_ROAMING_NOT_ALLOWED | L'abonné ne peut pas roaming dans ce VPLMN | 3GPP 29.27 Sectio 7.4.4 |

## Commandes S6a

| Command-Code | Nom | Description | Référence |
|---|---|---|---|
| 316 | Update-Location-Request/Answer (ULR/ULA) | Le MME envoie ceci au HSS lors du rattachement pour mettre à jour le nœud de service de l'abonné | 3GPP TS 29.272 Section 7.2.3 |

## Identifiants d'Application

| ID | Interface | Description | Référence |
|---|---|---|---|
| 16777251 | S6a/S6d | Authentification MME/SGSN et gestion des abonnements au HSS | 3GPP TS 29.272 |

# Liaison de Session Rx

**Liaison de Session Rx** conserve chaque message d'une session Rx sur le PCRF qui la possède. Elle existe parce que Rx est la seule interface dont la requête de terminaison de session ne contient rien sur quoi router.

## Le Problème

3GPP TS 29.214 §5.6.5 rend `Destination-Host` **optionnel** dans la `Session-Termination-Request` Rx. Elle ne définit **aucun** `Subscription-Id` et **aucune** `Framed-IP-Address` dans le message. Ce second point est structurel. Ces AVPs ne sont pas simplement absents en pratique. Ils ne figurent pas dans la définition du message, donc aucun AF conforme ne peut les fournir.

Que le STR contienne un `Destination-Host` dépend donc de l'AF. Certaines implémentations d'AF n'en envoient pas. Une capture représentative montre ces AVPs :

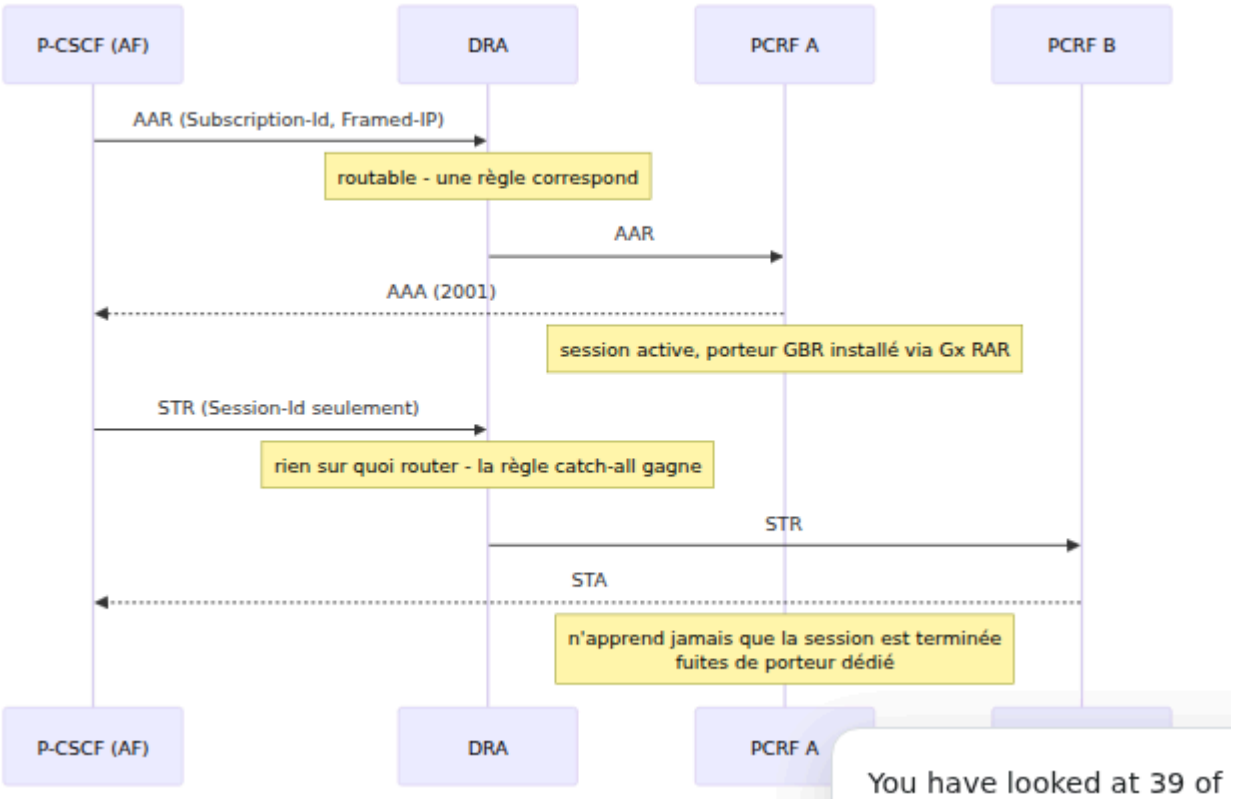
`Session-Id`, `Origin-Host`, `Origin-Realm`, `Destination-Realm`, `Auth-Application-Id`, `Vendor-Specific-Application-Id`, `AF-Application-Identifier`, `Termination-Cause`.

Le `Session-Id` est le propre identifiant opaque de l'AF. Selon RFC 6733 §8.8, il est globalement unique et stable pendant la durée de la session. Il n'identifie ni l'abonné ni le serveur.

Comparez cela avec la `AA-Request` qui a ouvert la session. Cette requête contient `Subscription-Id` et `Framed-IP-Address`, donc une règle Advanced Routing peut la router. Comparez-le également avec **Gx**. La ABNF de `CC-Request` (TS 29.212 §5.6.2) inclut `[ Destination-Host ]`, `*[ Subscription-Id ]` **et** `[ Framed-IP-Address ]`. C'est exactement pourquoi Gx n'a pas besoin d'équivalent de ce module.

TS 29.213 §7.3.5 impose la charge au DRA de toute façon. Un client "doit être capable d'envoyer chaque message d'une session au DRA". Un contournement est seulement quelque chose qui "peut être configuré".

Avec un seul PCRF, cette asymétrie est invisible. Avec plus d'un PCRF derrière le DRA, ce n'est pas le cas :




**Cet échec est silencieux.** RFC 6733 §8.4.2 ne prescrit aucun code de résultat pour un STR qui nomme une session inconnue. La machine d'état §8.1 gère "STR reçu" depuis *n'importe quel* état avec "envoyer STA, nettoyer". Un PCRF qui répond `2001` pour une session qu'il n'a jamais eue est donc conforme, et de vraies implémentations font cela. Ne vous attendez pas à ce que les codes de résultat révèlent cela. (`5002 DIAMETER_UNKNOWN_SESSION_ID` est prescrit pour l'**ASA**, §8.5.2, pas pour le STA.)

Il en va de même, moins visiblement, pour un *pool* de PCRFs interchangeables. Une règle qui route vers `peers: [a, b]` sélectionne par message. Un AAR et son STR peuvent donc atterrir sur des membres différents du même pool.

# Quand Vous En Avez Besoin

- Migration entre PCRFs, ou exécution de deux en parallèle pour un essai par abonné.
- Toute règle d'Advanced Routing qui envoie Rx à plus d'un pair.

Vous **n'en avez pas besoin** avec un seul PCRF. Il est désactivé par défaut.

# Comment Cela Fonctionne

| Message | Action |
|---|---|
| `AA-Answer` (265, `Result-Code` 2001) | Lier `Session-Id` → le PCRF qui a répondu |
| `AA-Request` (265) | Router vers le PCRF lié |
| `Session-Termination-Request` (275) | Router vers le PCRF lié |
| `Session-Termination-Answer` (275) | Libérer la liaison, **uniquement** si elle provient du PCRF lié et que son `Result-Code` est final (2xxx ou 5xxx) |

## Les Liaisons Suivent les Faits, Pas les Intentions

Le DRA lie sur la **réponse**, jamais sur la requête. Une requête ne dit que où le DRA *a essayé* d'envoyer quelque chose. La réponse prouve qu'une session existe sur ce nœud. Cela a trois conséquences :

- Un AAR répondu avec une erreur, ou jamais répondu, ne laisse aucune liaison.
- Le DRA libère une session lorsque son STA arrive, pas lorsqu'il route le STR. RFC 6733 §5.5.4 sanctionne explicitement la retransmission après un basculement. Une libération anticipée fausserait la nouvelle tentative, l'échec même que ce module empêche.
- Une requête ne réécrit jamais une liaison. Une panne transitoire d'un pair ne peut donc pas migrer silencieusement une session active vers un PCRF différent.

La libération nécessite **les deux** conditions. Le STA doit provenir du pair qui possède la liaison, et son `Result-Code` doit être final. RFC 6733 §8.4.2 stipule que le serveur libère ses ressources lorsqu'il envoie le STA, donc 2xxx et 5xxx sont également finaux. Un 3xxx/4xxx est transitoire et l'AF peut retransmettre, donc la liaison survit pour la nouvelle tentative.

La condition de propriété n'est pas une propreté défensive. Sans elle, le module échoue dans le scénario exact pour lequel il a été construit. Le propriétaire devient injoignable. Le STR passe à un autre PCRF. Ce PCRF répond `2001` pour une session qu'il n'a jamais eue. Le DRA détruit alors la liaison au propriétaire pendant que son porteur est toujours actif. Cela vous laisse dans une situation pire que sans module, car la destruction semble maintenant *fixée*. La condition de propriété est le miroir de `insert_new` du côté de la liaison. Un second PCRF ne peut ni voler une liaison ni en détruire une.

## Ce Qu'il Ne Sera Pas Surcharge

Le module ne redirige que le routage loin d'une décision de relais. Tout le reste est un résultat délibéré, et le module le laisse tranquille :

| Résultat | Source | Honoré |
|---|---|---|
| `:discard` | une règle `route: :drop` | oui |
| `{:answer_message, code}` | une règle `route: {:error, code}` | oui |
| `{:answer_message, 3002}` | Advanced Routing échouant fermé sans pair connecté | oui |
| `Destination-Host` explicite | l'AF nommant son serveur (RFC 6733 §6.1.5) | oui |

Le coût est que le DRA ne peut pas détruire une session tant que chaque pair vers lequel ses règles pointent est injoignable. C'est le bon compromis. Un opérateur qui vide un nœud, ou échoue délibérément, ne demande pas à être contourné.

# Ordonnancement

C'est la seule étape qui s'exécute **après** Advanced Routing. Elle doit s'exécuter là. Un STR ne peut correspondre qu'à une règle catch-all, et la liaison doit pouvoir la battre.

L'observateur du chemin de réponse s'exécute **avant** Advanced Transform. Advanced Transform peut réécrire ou supprimer `Session-Id` et `Result-Code` sur les réponses. La liaison doit se baser sur le `Session-Id` que l'AF a réellement utilisé. Le module est délibérément insensible à l'occultation de topologie. Il lit le pair répondant à partir de ses capacités Diameter, pas d'un AVP `Origin-Host`. Une réécriture d'AVP ne peut donc pas le tromper.

# Configuration

```
config :dra,
  module_rx_session_binding: %{
    enabled: true,
    children: [DRA.Router.Modules.RxSessionBinding],
    ttl_seconds: 43_200
  }
```

| Clé | Objectif |
|---|---|
| `enabled` | Désactivé par défaut. Indépendant de `module_subscriber_lookup`. Les deux résolvent des problèmes différents, et vous pouvez vouloir l'un ou l'autre seul. |
| `ttl_seconds` | Garde-fou pour les sessions dont le STA n'arrive jamais. Ce n'est **pas** un minuteur de session. |

Taille `ttl_seconds` **au-dessus** de la `Authorization-Lifetime` de l'AF. Le DRA ne délie donc jamais un appel long en cours de session. Un P-CSCF qui annonce `Authorization-Lifetime: 36000` a besoin de plus de 10 heures. La valeur par défaut de 12 heures lui convient. Des balayages s'exécutent toutes les

`ttl_seconds / 2` (minimum 60s). Le DRA récupère une liaison perdue dans un délai de 1,5 × TTL.

# Observabilité

Trois événements de télémétrie, tous portant `session_id` :

| Événement | Métadonnées |
|---|---|
| `[:diameter, :rx_session_binding, :bind]` | `serving_host` |
| `[:diameter, :rx_session_binding, :route]` | `serving_host` |
| `[:diameter, :rx_session_binding, :release]` | - |

Un quatrième événement, `[:diameter, :rx_session_binding, :reap]`, porte un `count`. Le DRA l'émet uniquement lorsque le balayage TTL trouve des liaisons expirées. Un compte de récolte non nul signifie que des STAs disparaissent. Alertez sur ce compte.

Le DRA expose un compte de liaisons actives et peut les lister sous la forme `{session_id, origin_host, inserted_at}`.

# Limitations

- **Les liaisons sont en mémoire et locales au nœud.** Un redémarrage du DRA, ou un plantage d'un frère sous le superviseur `:one_for_all`, les supprime. Les sessions en cours retombent alors sur ce que les règles disent pour le reste de l'appel. Le DRA ne partage pas les liaisons entre les DRA. Un AF doit donc atteindre le même DRA pour les deux moitiés d'une session.
- **Le DRA lie le pair adjacent.** En mode relais, le DRA peut ajouter un Route-Record mais pas insérer un `Destination-Host` (RFC 6733 §6.1.9). Le DRA ne peut donc pas fixer un PCRF qui se trouve derrière un autre agent. Seul cet agent peut.

- **Une réponse AA-Answer tardive peut ressusciter une session libérée.** Si le DRA traite la réponse d'une mise à jour AAR après le STA, `insert_new` recrée la ligne. La session est terminée, donc c'est une ligne perdue, pas un mauvais routage. Les Session-Ids ne sont pas réutilisés (RFC 6733 §8.8). La valeur `ttl_seconds` limite la ligne, et elle apparaît dans le compteur `:reap`.
- **Le DRA ne peut pas libérer une session tant qu'Advanced Routing échoue fermé** pour Rx. Le résultat `{:answer_message, 3002}` est indiscernable d'un `route: {:error, 3002}` délibéré de l'opérateur. Gardez chaque PCRF qui détient des sessions actives dans la liste des pairs de la règle. Ce cas ne se présente donc pas.
- **`children` est requis dans la configuration.** `DRA.Supervisor` démarre les processus d'un module uniquement lorsque la configuration contient une liste `children`. Avec `enabled: true` et sans `children`, le GenServer ne démarre jamais. Chaque point d'entrée se dégrade en "pas de liaison", et rien ne vous avertit.
- **Le DRA lie uniquement la direction AF → PCRF.** Les `RAR` et `ASR` initiés par le PCRF portent un `Destination-Host` obligatoire (RFC 6733 §8.3.1, §8.5.1) et routent sur cela. Ils n'ont pas besoin de liaison.
- **Le DRA ne lie que l'identifiant d'application Rx de Rel-7+ (16777236).** Il ne lie pas les sessions sur l'identifiant Rx/Gq de Rel-6 hérité (16777229).
- **Le DRA émet de la télémétrie mais ne l'exporte pas.** Les quatre événements ci-dessus n'ont pas de définitions `Telemetry.Metrics`, donc rien n'atteint Prometheus aujourd'hui. C'est le même écart que les événements `subscriber_lookup` existants. Utilisez le compte de liaisons actives depuis un shell distant jusqu'à ce que cela soit connecté.

# Multihoming SCTP

**Le multihoming SCTP** fournit une redondance réseau pour le transport Diameter. Il permet à chaque point de terminaison de se lier à plusieurs adresses IP au sein d'une seule association. Si le chemin réseau principal échoue, SCTP bascule automatiquement vers un chemin alternatif. Cela ne perturbe pas la session Diameter.

## Pourquoi le Multihoming est Important

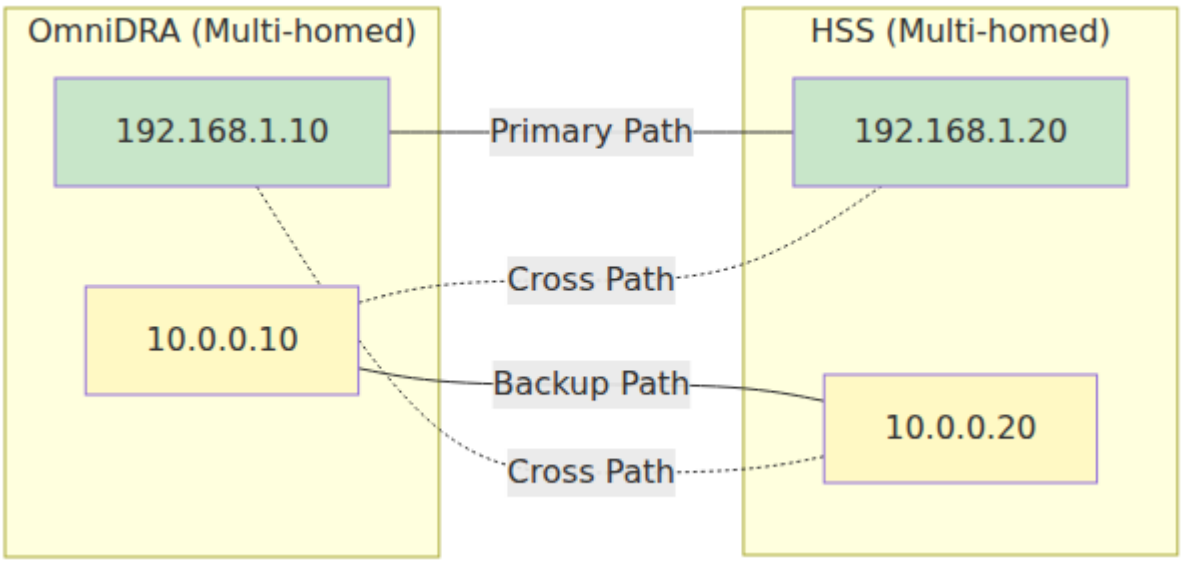
RFC 6733 définit Diameter sur SCTP. Le comportement de multihoming décrit ici est une capacité native de **SCTP** lui-même. RFC 4960 le spécifie. TCP lie une association à une seule paire d'adresses IP source/destination. SCTP permet à une association de s'étendre sur plusieurs adresses IP de chaque côté. Cela donne à OmniDRA une résilience au niveau de la couche de transport contre la perte d'un chemin réseau individuel, d'une carte réseau, d'un commutateur ou d'un segment routé. La connexion Diameter ne se dégrade pas et ne se r��tablit pas.

Caractéristiques clés :

- Les **cœurs de SCTP** surveillent tous les chemins réseau dans l'association en continu.
- **La bascule automatique** se produit au niveau du noyau si le chemin principal devient inaccessible.
- Le changement de chemin ne perturbe pas la session Diameter. L'association et son état Diameter persistent.
- Le noyau du système d'exploitation sélectionne le chemin automatiquement.

# Comment Cela Fonctionne

OmniDRA et un pair établissent une association multi-homée. Chaque point de terminaison annonce plusieurs adresses IP. Le noyau sélectionne un chemin principal pour la transmission de données. Il traite les chemins restants comme des routes de secours. Si le chemin principal échoue, le noyau promeut automatiquement une route de secours.




Dans le diagramme ci-dessus, le **Chemin Principal** (vert) transporte le trafic dans des conditions normales. Le **Chemin de Secours** (jaune) et les **Chemins Croisés** (pointillés) font tous partie de la même association SCTP. Ils restent disponibles pour une bascule immédiate.

# Configuration

Configurez le multihoming dans `config/runtime.exs` à deux endroits. Le premier est les adresses d'écoute locales auxquelles OmniDRA se lie. Le second est les adresses distantes de chaque pair multi-homé. Voir la Référence de Configuration pour la structure de configuration complète.

## Adresses d'Écoute DRA

Le champ `listen_ip` contrôle quelles adresses IP locales OmniDRA se lie pour les connexions entrantes. Il accepte une seule chaîne IP (compatible avec les

versions antérieures) ou une liste de chaînes IP. Utilisez une liste pour activer le multihoming.

```
%{
  # 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
}
```

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `listen_ip` | Chaîne ou Liste de Chaînes | Oui | - | Adresse(s) IP locale(s) auxquelles OmniDRA se lie pour les connexions entrantes. Une seule chaîne lie une adresse. Une liste de chaînes active le multihoming SCTP sur toutes les adresses listées. Pour le **transport TCP**, OmniDRA utilise uniquement la première IP de la liste. Pour le **transport SCTP**, OmniDRA lie toutes les IP listées à l'association. |
| `listen_port` | Entier | Oui | - | Port local pour les connexions Diameter entrantes. Le port standard est 3868 selon RFC 6733. |

**Compatibilité avec les versions antérieures :** Une seule chaîne IP reste entièrement supportée. Les configurations multi-homées existantes continuent de fonctionner sans changement. Pour opter pour le multihoming, fournissez une liste.

## Configuration des Pairs

Chaque entrée de pair déclare l'adresse distante du pair. Le champ `ip` contient l'adresse principale et est toujours requis. Le champ optionnel `additional_ips` liste toutes les adresses pour l'association multi-homée.

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

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `host` | Chaîne | Oui | - | Identité Diameter du pair (FQDN). Elle doit correspondre exactement à l'identité configurée du pair. |
| `realm` | Chaîne | Oui | - | Royaume Diameter du pair. Le DRA l'utilise pour les décisions de routage. |
| `ip` | Chaîne | Oui | - | Adresse IP principale du pair. Toujours requise, y compris pour la compatibilité avec les configurations mono-homées. |
| `additional_ips` | Liste de Chaînes | Non | - | Toutes les adresses IP pour l'association SCTP multi-homée. Ce champ est optionnel. Si vous l'omettez, OmniDRA utilise uniquement `ip`. Pour SCTP, incluez l'IP principale dans cette liste. Pour |

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| | | | | TCP, OmniDRA ignore ce champ, car TCP ne supporte pas le multihoming. |
| `port` | Entier | Oui | - | Port Diameter du pair. Le port standard est 3868 selon RFC 6733. |
| `transport` | Atome | Oui | - | Protocole de transport pour la connexion au pair : `:diameter_sctp` ou `:diameter_tcp`. Le multihoming nécessite `:diameter_sctp`. |
| `initiate_connection` | Booléen | Oui | - | Lorsque `true`, OmniDRA initie la connexion au pair. Lorsque `false`, OmniDRA attend que le pair se connecte. |

**Compatibilité avec les versions antérieures :** Le champ `ip` seul décrit un pair mono-homé. Si vous omettez `additional_ips`, le pair reste mono-homé. Par conséquent, les configurations de pair existantes n'ont pas besoin de changements.

# Exemple Complet

La configuration suivante lie OmniDRA à deux adresses locales. Elle se connecte à un HSS multi-homé et à un MME mono-homé.

```
config :dra,
  diameter: %{
    service_name: :omnitouch_dra,
    listen_ip: ["192.168.1.10", "10.0.0.10"],  # DRA multi-homé
    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 multi-homée
      %{
        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,
        initiate_connection: true
      },
      # MME mono-homé (compatible avec les versions antérieures)
      %{
        host: "mme01.example.com",
        realm: "example.com",
        ip: "192.168.1.30",
        port: 3868,
        transport: :diameter_sctp,
        initiate_connection: false
      }
    ]
  }
```

Les tableaux ci-dessus documentent `listen_ip`, `listen_port` et tous les paramètres des pairs. Les autres paramètres de niveau supérieur contrôlent l'identité Diameter d'OmniDRA et le comportement de sélection des pairs. La Référence de Configuration les documente :

| Paramètre | Type | Requis | Par Défaut | Desc |
|---|---|---|---|---|
| `service_name` | Atome | Oui | - | Nom i du se Diame d'Omi |
| `host` | Chaîne | Oui | - | Identi Diame propre d'Omi (FQDN |
| `realm` | Chaîne | Oui | - | Royau Diame propre d'Omi |
| `product_name` | Chaîne | Oui | - | Nom d produ annon pairs l'écha capac |
| `vendor_id` | Entier | Oui | - | ID du fourni annon pairs. est l'ident du fourni 3GPP. |

| Paramètre | Type | Requis | Par Défaut | Desc |
|---|---|---|---|---|
| `request_timeout` | Entier | Non | `5000` | Temps maxir millise à atte pour u répon pair a d'expi dema |
| `peer_selection_algorithm` | Atome | Oui | - | Algori qui sélect parmi pairs lors d routag |
| `allow_undefined_peers_to_connect` | Booléen | Oui | `false` | Lorsq `false` les pa liste `p` peuve conne Lorsq `true`, accep égale pairs listés. |
| `peers` | Liste | Oui | - | Liste config de co |

| Paramètre | Type | Requis | Par Défaut | Desc |
| --- | --- | --- | --- | --- |
| | | | | des pa<br>Config<br>des Pa |

# Tampons de Socket SCTP

OmniDRA définit un tampon de réception et d'envoi de socket explicite sur chaque transport SCTP. Il utilise la clé de niveau supérieur `diameter_socket_buffer_size` (par défaut 4 Mo). Contrairement à `listen_ip` et `peers`, cette clé se trouve au **niveau supérieur** de la configuration de l'application `:dra`. Elle n'est **pas** à l'intérieur de la carte `diameter:`.

Sans un tampon explicite, le socket SCTP hérite d'un petit `SO_RCVBUF` par défaut. Cela fixe la fenêtre de réception SCTP à environ une MTU et provoque des blocages de fenêtre nulle sous charge. Le noyau Linux limite `SO_RCVBUF` à `net.core.rmem_max`. Il limite le tampon d'envoi à `net.core.wmem_max`. Pour utiliser des tailles de tampon supérieures à cette limite, augmentez ces sysctls sur l'hôte.

```
config :dra,
  # Clé de niveau supérieur - PAS à l'intérieur de la carte
diameter: . Appliquée à chaque transport SCTP.
  diameter_socket_buffer_size: 4 * 1024 * 1024,  # 4 Mo
(recbuf/sndbuf)
  diameter: %{
    # ... configuration du service et des pairs ...
  }
```

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| `diameter_socket_buffer_size` | Entier | Non | `4194304` (4 Mo) | Taille du tam de socket en transport SC `recbuf` et `s` pairs `:diame` pour TCP. De `net.core.rm` nécessitent sysctls du no |

# Exigences

Le multihoming SCTP nécessite ces conditions sur les deux points de terminaison :

- Le **module noyau SCTP** doit être chargé sur l'hôte (le paquet `lksctp-tools` sur Linux).
- Toutes les adresses IP configurées doivent être routables vers et depuis le pair.
- Les règles de pare-feu doivent permettre le trafic SCTP sur chaque adresse IP configurée.
- Configurez les deux points de terminaison pour le multihoming afin d'atteindre une redondance complète. Un OmniDRA multi-homé connecté à un pair mono-homé obtient une redondance de chemin uniquement du côté d'OmniDRA.

# Limitations

- **TCP ne supporte pas le multihoming.** Avec le transport `:diameter_tcp`, OmniDRA utilise uniquement l'IP principale. `listen_ip` utilise la première IP de la liste, et OmniDRA ignore `additional_ips` du pair.

- **Le timing de bascule de chemin** dépend des paramètres SCTP du noyau, tels que l'intervalle de cœur et les seuils de retransmission. Vous devez les ajuster en dehors d'OmniDRA au niveau du système d'exploitation.

# Références

- RFC 4960 - Stream Control Transmission Protocol (SCTP)
- RFC 6733 - Diameter Base Protocol
- Référence de Configuration

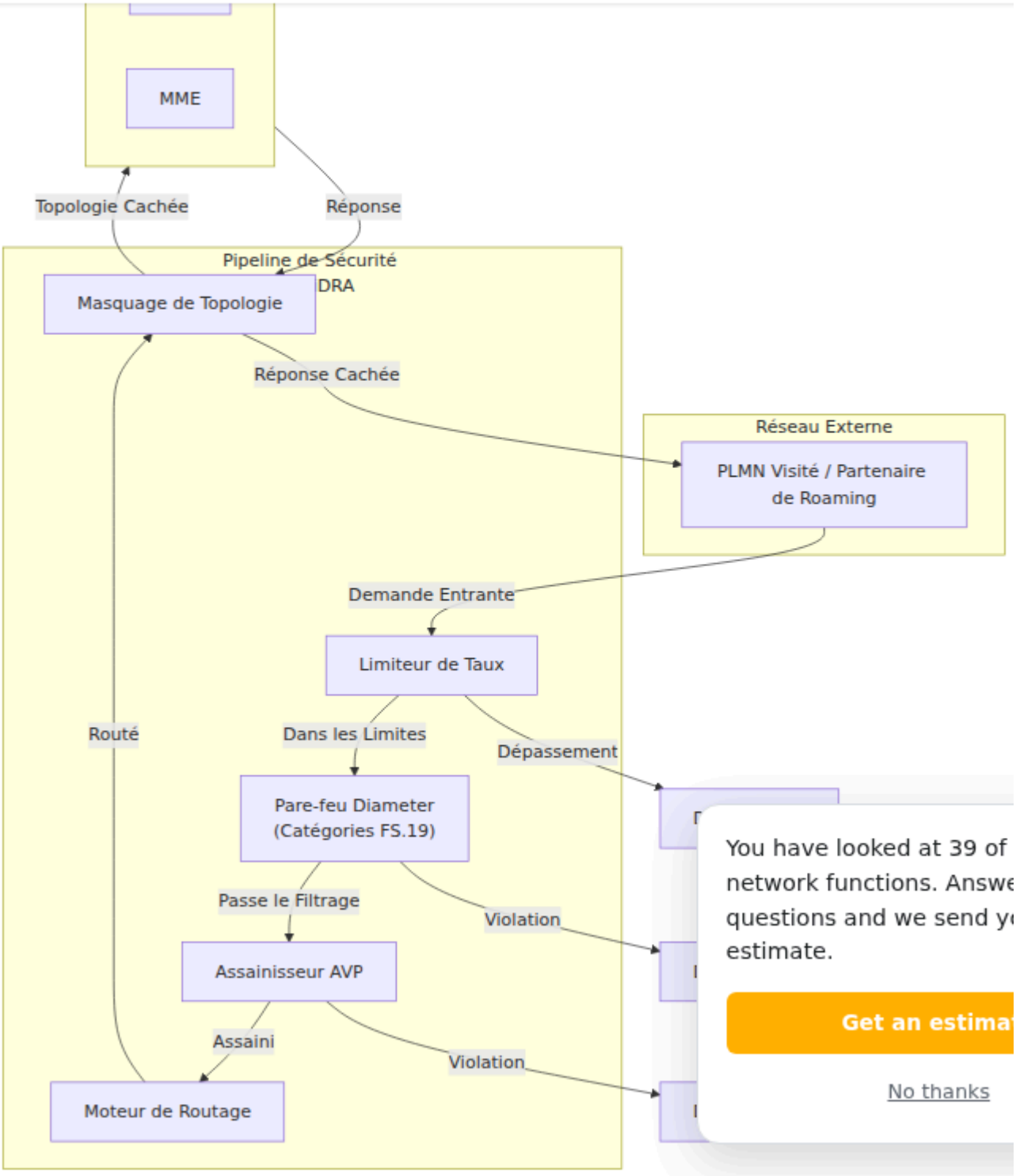
# Sécurité Diameter

OmniDRA fournit un ensemble complet de modules de sécurité Diameter. Ils sont conformes à **GSMA FS.19** (Sécurité de l'interconnexion Diameter, v10.0) et **GSMA FS.21** (Recommandations de sécurité des signaux d'interconnexion, v12.0). Ces modules protègent le réseau contre les attaques au niveau des signaux sur les interfaces d'interconnexion Diameter.

Vous configurez chaque module indépendamment. Vous pouvez activer ou désactiver chaque module sans effet sur les autres. Chaque module émet ses propres événements de télémétrie pour la surveillance et l'alerte.

# Vue d'ensemble de l'architecture






You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks

# Ordre de Traitement

Les demandes Diameter entrantes passent par les modules de sécurité dans l'ordre suivant. Si une étape rejette un message, le DRA ne le transmet jamais aux étapes suivantes.

| Ordre | Module | Direction | But |
|---|---|---|---|
| 1 | Limiteur de Taux | Entrant | Protection contre les inondations volumétriques |
| 2 | Pare-feu Diameter | Entrant | Filtrage de protocole et de contenu FS.19 |
| 3 | Assainisseur AVP | Entrant | Validation et suppression des AVP |
| 4 | Moteur de Routage | - | Routage Diameter standard |
| 5 | Masquage de Topologie | Sortant | Dissimulation de la topologie du réseau |

# Actions

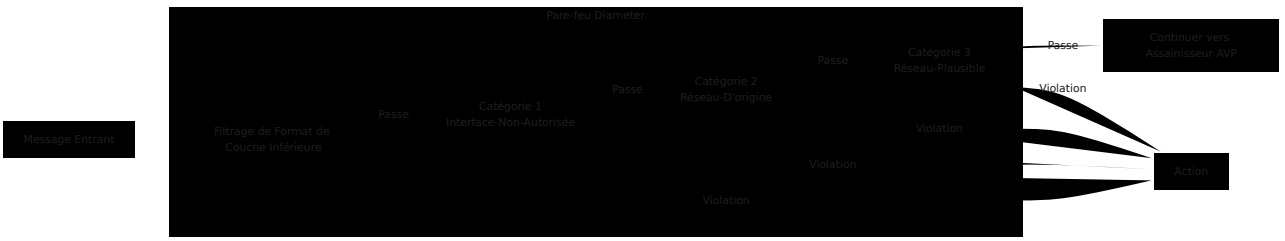
Tous les modules de sécurité prennent en charge des actions configurables lorsqu'ils détectent une violation. Vous configurez les actions par module et par catégorie pour le Pare-feu Diameter.

| Action | Comportement | Résultat Diameter |
|---|---|---|
| `{:error, 3002}` | Répondre avec une réponse d'erreur Diameter | DIAMETER_UNABLE_TO_DELIVER |
| `{:error, 3004}` | Répondre avec une réponse d'erreur Diameter | DIAMETER_TOO_BUSY |
| `{:error, 3007}` | Répondre avec une réponse d'erreur Diameter | DIAMETER_APPLICATION_UNSUPPORTED |
| `:drop` | Écarter le message | Aucune réponse envoyée |
| `:log_only` | Journaliser la violation mais laisser passer le message | Le message continue |

---

# Pare-feu Diameter

Le Pare-feu Diameter met en œuvre les quatre couches de filtrage définies dans **GSMA FS.19 Section 3.3**. Il met également en œuvre la catégorisation des paquets indépendante du protocole de **FS.21 Section 7**. Le pare-feu évalue les messages à travers chaque couche dans l'ordre. Une violation à n'importe quelle couche arrête le traitement ultérieur.



# Filtrage de Format de Couche Inférieure

**Référence GSMA :** FS.19 Section 3.3.4, FS.21 Section 7.3.1

Le filtrage de couche inférieure détecte les violations au niveau du protocole et les tentatives de spoofing de base. Il n'a pas besoin de comprendre la sémantique des applications de couche supérieure. Cette couche attrape les messages mal formés avant qu'ils n'atteignent une inspection plus approfondie.

**Vérifications effectuées :**

| Vérification | Référence FS.19 | Description |
|---|---|---|
| Duplication d'AVP | Section 3.3.4, 4.8.1 | Détecte les instances dupliquées d'AVP qui doivent apparaître au maximum une fois (par exemple, Origin-Host, Origin-Realm). Empêche les attaques d'évasion par doublement d'AVP. |
| Ordre de Session-Id | Section 3.3.4 | Confirme que le Session-Id (AVP 263) est le premier AVP dans le message, conformément à RFC 6733 Section 8.8. |
| Destination-Host dans les Réponses | Section 3.3.4 | Rejette un message de réponse contenant un AVP Destination-Host. Il s'agit d'une violation de protocole qui peut indiquer une évasion de filtre. |
| Validation des AVP Obligatoires | Section 3.3.5 | Confirme que les AVP requis sont présents pour chaque code de commande. Par exemple, un ULR doit contenir User-Name et Visited-PLMN-Id conformément à 3GPP TS 29.272 Section 7.2.3. |

```
low_layer: %{
  enabled: true,
  action: {:error, 3002},
  # Codes AVP qui ne doivent pas apparaître plus d'une fois (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 doit être le premier AVP (RFC 6733 Section 8.8)
  enforce_session_id_first: true,
  # Les messages de réponse ne doivent pas contenir Destination-
Host (FS.19 Section 3.3.4)
  reject_destination_host_in_answers: true,
  # AVP obligatoires par code de commande (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]
  }
}
```

## Catégorie 1 - Filtrage de Paquets Interface-Non-Autorisée

**Référence GSMA :** FS.19 Section 3.3.5, FS.21 Section 7.2.1

Le filtrage de la catégorie 1 n'accepte que les IDs d'application Diameter et les codes de commande autorisés sur chaque interface d'interconnexion. Cela empêche l'accès externe aux interfaces réservées à l'interne. Par exemple, il bloque les commandes Sh sur une interface S6a. Il impose également qu'un partenaire de roaming n'envoie que les types de messages couverts par son accord de roaming.

L'approche de liste blanche suit la recommandation FS.19 : **bloquer tous les messages Diameter sauf ceux qu'une interface donnée nécessite**.

Vous pouvez configurer des listes blanches globalement, ce qui les applique à tous les pairs, ou par pair. Une liste blanche par pair permet à différents partenaires de roaming d'avoir différents ensembles de messages autorisés.

```
category_1: %{
  enabled: true,
  action: {:error, 3007},
  whitelists: %{
    # Liste blanche par pair - restreindre ce partenaire à ULR et
AIR uniquement (FS.19 Section 3.3.5)
    "restricted-partner.roaming.com" => %{
      16_777_251 => [316, 318]
    },
    # Liste blanche par défaut - commandes S6a standard autorisées
pour tous les autres pairs
    all: %{
      # S6a/S6d (FS.19 Section 3.3.5, Tableau 2)
      16_777_251 => [316, 317, 318, 319, 320, 321, 323],
      # S13 - Vérification d'Identité ME (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]
    }
  }
}
```

**Liste Blanche Commune S6a/S6d :**

| Code de Commande | Nom | Direction | Référence |
|---|---|---|---|
| 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 |

**IDs d'Application Communs pour l'Interconnexion de Roaming** (FS.19 Section 3.3.5) :

| ID d'Application | Interface | Référence |
|---|---|---|
| 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 |

## Catégorie 2 - Filtrage de Paquets Réseau-D'origine

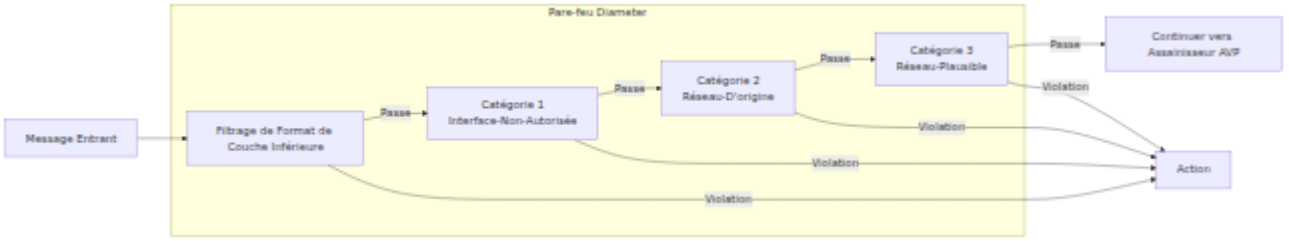
**Référence GSMA :** FS.19 Section 3.3.6, FS.21 Section 7.2.2

Le filtrage de la catégorie 2 protège les abonnés d'origine contre les messages qui les ciblent depuis l'interconnexion. Le pare-feu inspecte les messages sur les codes de commande protégés pour l'identité de l'abonné (IMSI dans l'AVP User-Name, MSISDN dans l'AVP 701). Si l'identité correspond à un préfixe d'abonné d'origine, le pare-feu rejette le message. Le trafic légitime pour les abonnés d'origine provient de l'intérieur du réseau d'origine, et non de pairs externes.

Cela empêche les attaques où une entité externe envoie des mises à jour de localisation, des demandes de données d'abonné ou des requêtes d'authentification ciblant les propres abonnés de l'opérateur.

```
# Niveau supérieur : définir les préfixes d'abonnés d'origine
home_imsi_prefixes: ["31338", "31339"],
home_msisdn_prefixes: ["+1313"],

category_2: %{
  enabled: true,
  action: {:error, 3002},
  # Codes de commande S6a qui portent l'identité de l'abonné
(FS.19 Section 3.3.6, Tableau 12)
  protected_command_codes: [316, 317, 318, 319, 320, 321, 323]
}
```




# Catégorie 3 - Filtrage de Paquets Réseau-Plausible

**Référence GSMA :** FS.19 Section 3.3.7, FS.21 Section 7.2.3

Le filtrage de la catégorie 3 détecte les changements de localisation peu plausibles. Il suit le dernier réseau vu pour chaque abonné (IMSI). Lorsqu'une demande de mise à jour de localisation arrive d'un réseau visité différent de celui précédent, le pare-feu compare le temps écoulé à un seuil configurable. Un changement de domaine qui se produit plus rapidement que ce qui est physiquement possible indique une attaque potentielle, telle qu'une mise à jour de localisation falsifiée.

Ce module maintient un suivi d'état par IMSI dans une table ETS. Il nettoie automatiquement les entrées obsolètes. Il supprime périodiquement les entrées de plus de 24 heures.

**Vérifications effectuées :**

| Vérification | Référence FS.19 | Description |
| --- | --- | --- |
| Vérification de Localisation Précédente | Section 3.3.7.1 | Compare l'Origin-Realm du ULR actuel avec le dernier Origin-Realm vu pour cet IMSI |
| Vérification de Vitesse/Temps | Section 3.3.7.2 | Signale un changement de domaine qui se produit dans une fenêtre de temps minimale configurable |

```
category_3: %{
  enabled: true,
  # Commencer avec log_only pour observer les motifs avant
d'appliquer (FS.19 Section 3.3.7)
  action: :log_only,
  # Vitesse maximale plausible en km/h (FS.19 Section 3.3.7.2)
  max_velocity_kmh: 1200,
  # Temps minimum entre ULRs de différents domaines pour le même
IMSI
  min_time_between_updates_seconds: 2
}
```

## Configuration

Activez le Pare-feu Diameter au niveau supérieur. Configurez chaque couche de filtrage indépendamment. Combinez les extraits de configuration de chaque section de catégorie ci-dessus sous une seule clé `module_diameter_firewall` :

```
config :dra,
  module_diameter_firewall: %{
    enabled: true,
    home_imsi_prefixes: ["31338", "31339"],
    home_msisdn_prefixes: ["+1313"],
    low_layer: %{ ... },      # Voir Filtrage de Format de Couche
Inférieure ci-dessus
    category_1: %{ ... },     # Voir Catégorie 1 ci-dessus
    category_2: %{ ... },     # Voir Catégorie 2 ci-dessus
    category_3: %{ ... }      # Voir Catégorie 3 ci-dessus
  }
```

## Paramètres de Niveau Supérieur

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activer ou désactiv Diameter. Lorsque inspection. |
| `children` | Liste | Non | `[]` | Processus que le su est activé. Pour le p `[DRA.Router.Modul` Le suivi d'état (vite |
| `home_imsi_prefixes` | Liste | Non | `[]` | Liste de chaînes de d'origine. Le filtrage `["31338", "31339"` |
| `home_msisdn_prefixes` | Liste | Non | `[]` | Liste de chaînes de abonnés d'origine. Exemple : `["+1313"` |

## Paramètres de Couche Inférieure

| Paramètre | Type | Requis | Par Défaut | D |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `true` | A d fi fc c in |
| `action` | Action | Non | `{:error, 3002}` | A e lo p d vi c in |
| `single_instance_avps` | Liste | Non | `[264, 296, 283, 293]` | C q d a pl fc m d c O (2 O (2 D R et |

| Paramètre | Type | Requis | Par Défaut | D |
|---|---|---|---|---|
| | | | | D H |
| `enforce_session_id_first` | Booléen | Non | `true` | R m le n' pr |
| `reject_destination_host_in_answers` | Booléen | Non | `true` | R m ré co ur D H |
| `mandatory_avps` | Carte | Non | `%{}` | C co co lis co re E {3 14 29 no U Vi Pl O et R le |

| Paramètre | Type | Requis | Par Défaut | D |
|---|---|---|---|---|
| | | | | m U |

## Paramètres de Catégorie 1

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `true` | Activer ou désactiver le filtra Catégorie 1. |
| `action` | Action | Non | `{:error, 3007}` | Action à entreprendre lorsqu feu détecte une violation de Catégorie 1. Le défaut répon DIAMETER_APPLICATION_UNS |
| `whitelists` | Carte | Oui | - | Carte du nom d'hôte du pair aux combinaisons d'ID d'app code de commande autorisé `:all` fournit une liste blanch défaut. Les entrées spécifiqu ont la priorité. |

## Paramètres de Catégorie 2

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `true` | Activer ou désactiver le filtrage de la Catégorie 2. |
| `action` | Action | Non | `{:error, 3002}` | Action à entreprendre lorsqu'un message cible un abonné d'origine depu l'interconnexio |
| `protected_command_codes` | Liste | Non | `[]` | Codes de commande qu déclenchent d vérifications d'identité d'abonné d'origine. Typiquement l codes de commande S6 qui portent l'identité d'abonné. |

## Paramètres de Catégorie 3

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| `enabled` | Booléen | Non | `false` | Ac dé filt Ca né pr Lo Le dé pr au lor es |
| `action` | Action | Non | `:log_only` | Ac en lor co pla loc éc Co `:l` aju av d'a |
| `max_velocity_kmh` | Entier | Non | `1200` | Vit pla km po ca |

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| | | | | dis<br>gé |
| `min_time_between_updates_seconds` | Entier | Non | `2` | Te<br>en<br>en<br>jou<br>loc<br>dif<br>do<br>mé<br>pa<br>un<br>de<br>es<br>qu |

## Événements de Télémétrie

| Événement | Description |
|---|---|
| `[:diameter, :firewall, :low_layer, :block]` | Le pare-feu a détecté et bloqué une violation de format de couche inférieure |
| `[:diameter, :firewall, :category_1, :block]` | Le pare-feu a détecté et bloqué une violation de la Catégorie 1 |
| `[:diameter, :firewall, :category_2, :block]` | Le pare-feu a détecté et bloqué une violation de la Catégorie 2 |
| `[:diameter, :firewall, :category_3, :block]` | Le pare-feu a détecté et bloqué une violation de la Catégorie 3 |
| `[:diameter, :firewall, :pass, :count]` | Le message a passé tous les contrôles du pare-feu |

Tous les événements incluent ces métadonnées : `origin_host`, `application_id`, `command_code`, `application_name`, `command_name`, et `reason`.

---

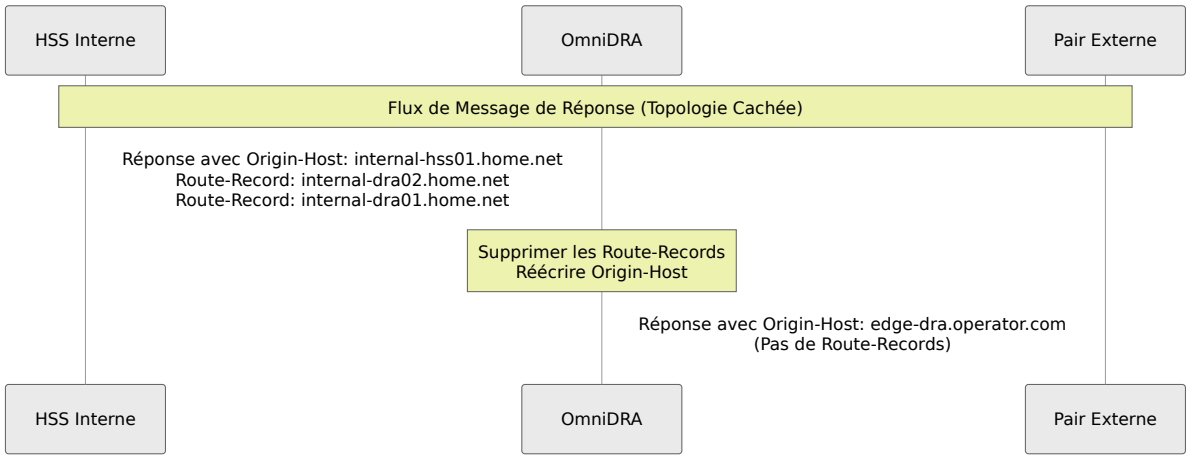
# Masquage de Topologie

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

Le module de Masquage de Topologie empêche l'exposition de la topologie interne du réseau aux pairs externes. Lorsque les messages Diameter traversent plusieurs nœuds internes, ils accumulent des AVP Route-Record. Ils portent également des valeurs Origin-Host/Origin-Realm qui révèlent des noms d'hôtes internes, la structure du réseau et le nombre de nœuds. Un attaquant peut utiliser ces informations pour cartographier le réseau interne pour des attaques ciblées.

Le masquage de topologie fonctionne sur le **chemin sortant**. Il s'exécute après que le DRA a pris des décisions de routage et avant qu'il ne transmette

des messages au pair de destination.




# Fonctionnalités

| Fonctionnalité | Référence FS.19 | Description |
|---|---|---|
| Suppression de Route-Record | Section 2.4 | Supprime tous les AVP Route-Record (282) qui révèlent des chemins de routage internes et des noms d'hôtes de nœuds |
| Réécriture de Origin-Host | Section 3.4 | Remplace l'AVP Origin-Host par l'identité propre du DRA (ou une valeur personnalisée) sur les messages de réponse |
| Réécriture de Origin-Realm | Section 3.4 | Remplace optionnellement l'AVP Origin-Realm pour cacher la structure de domaine interne |
| Contrôle par Pair | - | Appliquer le masquage de topologie de manière sélective, uniquement aux pairs externes ou à tous les pairs |

```
module_topology_hiding: %{
  enabled: true,
  # Supprimer les AVP Route-Record révélant des chemins internes
(FS.19 Section 2.4)
  strip_route_records: true,
  # Réécrire Origin-Host sur les réponses pour cacher les noms de
nœuds internes (FS.19 Section 3.4)
  rewrite_origin_host: %{enabled: true, replacement: :self},
  # Optionnellement cacher la structure de domaine interne
  rewrite_origin_realm: %{enabled: false, replacement: :self},
  # Appliquer à tous les pairs externes, ou lister des noms
d'hôtes spécifiques
  external_peers: :all
}
```

## Paramètres

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activer ou désactiver le module de Masquage de Topologie. |
| `strip_route_records` | Booléen | Non | `true` | Supprimer tous les AVP Route-Record (code 282) des messages avant que le DRA ne les transmette aux pairs externes. |
| `rewrite_origin_host` | Carte | Non | Voir ci-dessous | Contrôle la réécriture de Origin-Host sur les messages de réponse. |
| `rewrite_origin_realm` | Carte | Non | Voir ci-dessous | Contrôle la réécriture de Origin-Realm sur les messages de réponse. |
| `external_peers` | `:all` ou Liste | Non | `:all` | Pairs que le module considère comme |

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| | | | | externes. Le module applique le masquage de topologie uniquement aux messages destinés à ces pairs. Utilisez `:all` pour les déploiements d'interconnexion. Utilisez une liste de noms d'hôtes pour une application sélective. |

**Paramètres de Réécriture** (s'appliquent à la fois à `rewrite_origin_host` et `rewrite_origin_realm`) :

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `enabled` | Booléen | Non | varie | Activer la réécriture pour cet AVP. Par défaut, c'est `true` pour Origin-Host, `false` pour Origin-Realm. |
| `replacement` | `:self` ou Chaîne | Non | `:self` | Valeur de remplacement. `:self` utilise l'identité propre du DRA à partir de la configuration `diameter` (`host.realm`). Le module utilise une valeur de chaîne telle quelle. |

## Événements de Télémétrie

| Événement | Description |
|---|---|
| `[:diameter, :topology_hiding, :route_record, :stripped]` | Le module a supprimé des AVP Route-Record. La mesure inclut le `count` des AVP supprimés. |
| `[:diameter, :topology_hiding, :origin_host, :rewritten]` | Le module a réécrit l'AVP Origin-Host |
| `[:diameter, :topology_hiding, :origin_realm, :rewritten]` | Le module a réécrit l'AVP Origin-Realm |

---

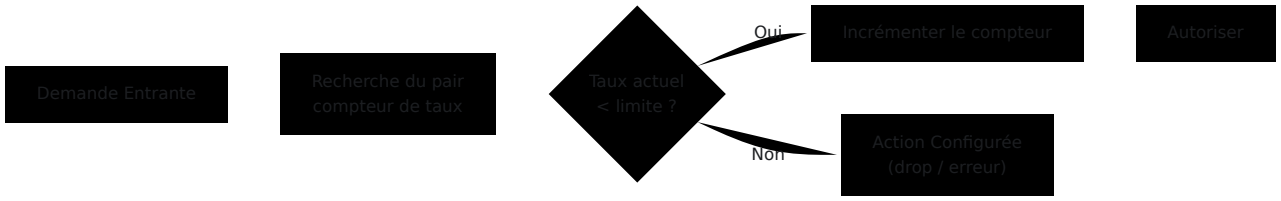
# Limiteur de Taux

**Référence GSMA :** FS.19 Section 3.4 (Disponibilité / Protection contre les DoS)

Le Limiteur de Taux impose des limites de taux de message par pair. Il protège contre les attaques volumétriques et les inondations de messages. Il fonctionne comme le premier contrôle de sécurité dans le pipeline, avant tout parsing de message ou inspection de contenu. Cela réduit la charge excessive le plus tôt possible.

Le Limiteur de Taux suit les limites de taux par pair avec un compteur à fenêtre glissante. Chaque pair a un compteur indépendant qui se réinitialise chaque seconde.

```
module_rate_limiter: %{
  enabled: true,
  # Limite par défaut pour tous les pairs (FS.19 Section 3.4)
  default_max_requests_per_second: 1000,
  default_action: {:error, 3004},
  # Remplacements par pair
  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}
  }
}
```



## Paramètres

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activ Limit |
| `children` | Liste | Non | `[]` | Proce déma activ `[DRA` afin d de ta |
| `default_max_requests_per_second` | Entier | Non | `1000` | Nomb secon |
| `default_action` | Action | Non | `{:error, 3004}` | Actio dépa `3004` DIAM |
| `peer_limits` | Carte | Non | `%{}` | Carte remp non l défau |

**Paramètres de Remplacement par Pair :**

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `max_requests_per_second` | Entier | Non | Hérite du défaut | Nombre maximum de demandes par seconde pour ce pair spécifique. |
| `action` | Action | Non | Hérite du défaut | Action lorsque ce pair dépasse sa limite de taux. |

## Événements de Télémétrie

| Événement | Description |
|---|---|
| `[:diameter, :rate_limiter, :throttled]` | Le module a limité le taux d'un message. Les mesures incluent `current_rate` et `limit`. |
| `[:diameter, :rate_limiter, :allowed]` | Un message était dans les limites de taux. |

---

# Assainisseur AVP

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

L'Assainisseur AVP valide et assainit les AVP Diameter à la frontière de l'interconnexion. Il répond aux recommandations de la Section 3.6 de FS.21. Ces recommandations traitent des attaques de manipulation au niveau du protocole qui opèrent en dessous du niveau des catégories de filtrage FS.19.

## Fonctionnalités

| Fonctionnalité | Référence GSMA | Description |
|---|---|---|
| Suppression d'AVP de Vendeur Inconnu | FS.21 Section 3.6 | Supprime les AVP de vendeurs non listés. Empêche l'injection d'AVP propriétaires qui peuvent déclencher un comportement inattendu dans les nœuds backend. |
| Profondeur de Nesting d'AVP Groupés | FS.21 Section 3.6 | Imposer une profondeur de nesting maximale pour les AVP groupés. Empêche les attaques de débordement de pile qui utilisent des structures AVP profondément imbriquées. |

```
module_avp_sanitizer: %{
  enabled: true,
  action: {:error, 3002},
  # Autoriser uniquement les AVP de vendeur standard sur
l'interconnexion (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,
  # Prévenir le débordement de pile via des AVP groupés
profondément imbriqués (FS.21 Section 3.6)
  max_avp_nesting_depth: 10
}
```

## Paramètres

| Paramètre | Type | Requis | Par Défaut | Descri |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activer ou d le module Assainisseu |
| `action` | Action | Non | `{:error, 3002}` | Action à entreprendr le module d une violatio profondeur nesting. La suppression vendeur enl AVP en caus bloque pas message. |
| `allowed_vendor_ids` | Liste | Non | `[0, 10415, 13019, 5535]` | Liste des ID vendeur aut sur l'interco Le module s les AVP d'au vendeurs. |
| `strip_unknown_vendor_avps` | Booléen | Non | `true` | Activer la suppression de vendeurs figurent pas liste `allowed_ve` |
| `max_avp_nesting_depth` | Entier | Non | `10` | Profondeur autorisée de |

| Paramètre | Type | Requis | Par Défaut | Descri |
|---|---|---|---|---|
| | | | | d'AVP group<br>L'action con<br>s'applique à<br>message qu<br>cette profor |

**IDs de Vendeur Communs :**

| ID de Vendeur | Organisation | Remarques |
|---|---|---|
| 0 | IETF | AVP Diameter standard définis dans les RFC |
| 10415 | 3GPP | Tous les AVP définis par la 3GPP (S6a, Gx, Rx, etc.) |
| 13019 | ETSI | AVP définis par l'ETSI |
| 5535 | 3GPP2 | AVP définis par la 3GPP2 (interopérabilité CDMA) |

## Événements de Télémétrie

| Événement | Description |
|---|---|
| `[:diameter, :avp_sanitizer, :unknown_vendor, :stripped]` | Le module a supprimé un ou plusieurs AVP de vendeurs inconnus |
| `[:diameter, :avp_sanitizer, :nesting_depth, :violation]` | Un message a dépassé la profondeur maximale de nesting d'AVP |
| `[:diameter, :avp_sanitizer, :pass, :count]` | Le message a passé tous les contrôles d'assainissement |

---

# Recommandations de Déploiement

## Ordre de Activation Recommandé

Lorsque vous déployez les modules de sécurité pour la première fois, activez-les un par un. Cela évite de perturber le trafic en direct :

1. **Limiteur de Taux** - Commencez avec des limites généreuses et surveillez les motifs de trafic. Resserrez les limites après avoir compris les taux de base.
2. **Assainisseur AVP** - Risque faible de faux positifs. Activez les vérifications de suppression et de nesting.
3. **Pare-feu Diameter (Catégorie 1)** - Définissez des listes blanches basées sur les accords de roaming. Commencez avec des combinaisons d'ID d'application / code de commande connues et bonnes.
4. **Pare-feu Diameter (Couche Inférieure)** - Activez les vérifications de conformité au protocole.
5. **Pare-feu Diameter (Catégorie 2)** - Configurez les préfixes IMSI/MSISDN d'origine.

6. **Masquage de Topologie** - Activez d'abord la suppression des Route-Records, puis la réécriture de Origin-Host.
7. **Pare-feu Diameter (Catégorie 3)** - Activez-le avec l'action `:log_only`. Observez les motifs de localisation avant d'appliquer.

## Défense en Profondeur

Ces modules mettent en œuvre le principe de **défense en profondeur** de la Section 3.4 de FS.19. Chaque couche aborde une classe différente d'attaque :

| Classe d'Attaque | Défense Principale | Défense Secondaire |
|---|---|---|
| DoS Volumétrique | Limiteur de Taux | Pare-feu Diameter (toutes catégories) |
| Abus d'Interface | Catégorie 1 | Assainisseur AVP |
| Ciblage d'Abonnés d'origine | Catégorie 2 | Catégorie 1 (restriction d'interface) |
| Spoofing de Localisation | Catégorie 3 | Catégorie 2 (vérification d'abonné d'origine) |
| Découverte de Topologie | Masquage de Topologie | - |
| Injection d'AVP | Assainisseur AVP | Filtrage de Couche Inférieure |
| Évasion de Protocole (Doublement d'AVP) | Filtrage de Couche Inférieure | Assainisseur AVP |

# Référence Croisée des Documents GSMA

| Module / Fonctionnalité | FS.19 v10.0 | FS.21 v12.0 |
|---|---|---|
| Filtrage de Format de Couche Inférieure | Section 3.3.4 | Section 7.3.1 |
| Filtrage de Catégorie 1 | Section 3.3.5, Annexe B.3.3 | Section 7.2.1 |
| Filtrage de Catégorie 2 | Section 3.3.6, Annexe B.3.4 | Section 7.2.2, Section 16 |
| Filtrage de Catégorie 3 | Section 3.3.7, Annexe B.3.5 | Section 7.2.3 |
| Masquage de Topologie | Section 2.4, Section 3.4 | Section 3.6 |
| Limitation de Taux | Section 3.4 | - |
| Assainissement d'AVP | Section 4.8.1, 4.8.2 | Section 3.6 |
| Défense en Profondeur | Section 3.4 | Section 3.15 |
| Catégories de Filtrage (Indépendant du Protocole) | Annexe A | Section 7 |

# Routage de Diamètre Standard

**Routage de Diamètre Standard** est le comportement par défaut de transfert de requêtes du DRA. Le DRA l'applique lorsque aucune règle de Routage Avancé ne correspond à une requête entrante. Il suit le mécanisme de routage basé sur la priorité défini dans le Protocole de Base Diamètre (RFC 6733). Il sélectionne une destination à partir du contenu des AVP et de l'ensemble des pairs connectés du DRA.

## Quand le Routage Standard S'applique

Le DRA tente toujours le routage standard comme son comportement de base. Si le module de Routage Avancé est activé et qu'une de ses règles correspond à la requête, cette règle décide du résultat à la place. La règle peut router vers des pairs nommés, router par `Destination-Host`, générer une réponse d'erreur ou abandonner la requête. Lorsque le Routage Avancé est désactivé, ou qu'aucune règle ne correspond, la requête passe au niveau de priorité de routage standard décrit ici.

Une exception étroite s'applique après les deux étapes. Lorsqu'il est activé, le Liaison de Session Rx redirige une requête Rx vers le PCRF qui détient sa session. Cela ne remplace qu'une décision de relais. Un résultat `:drop`, une réponse d'erreur, et un `Destination-Host` explicite restent valables.

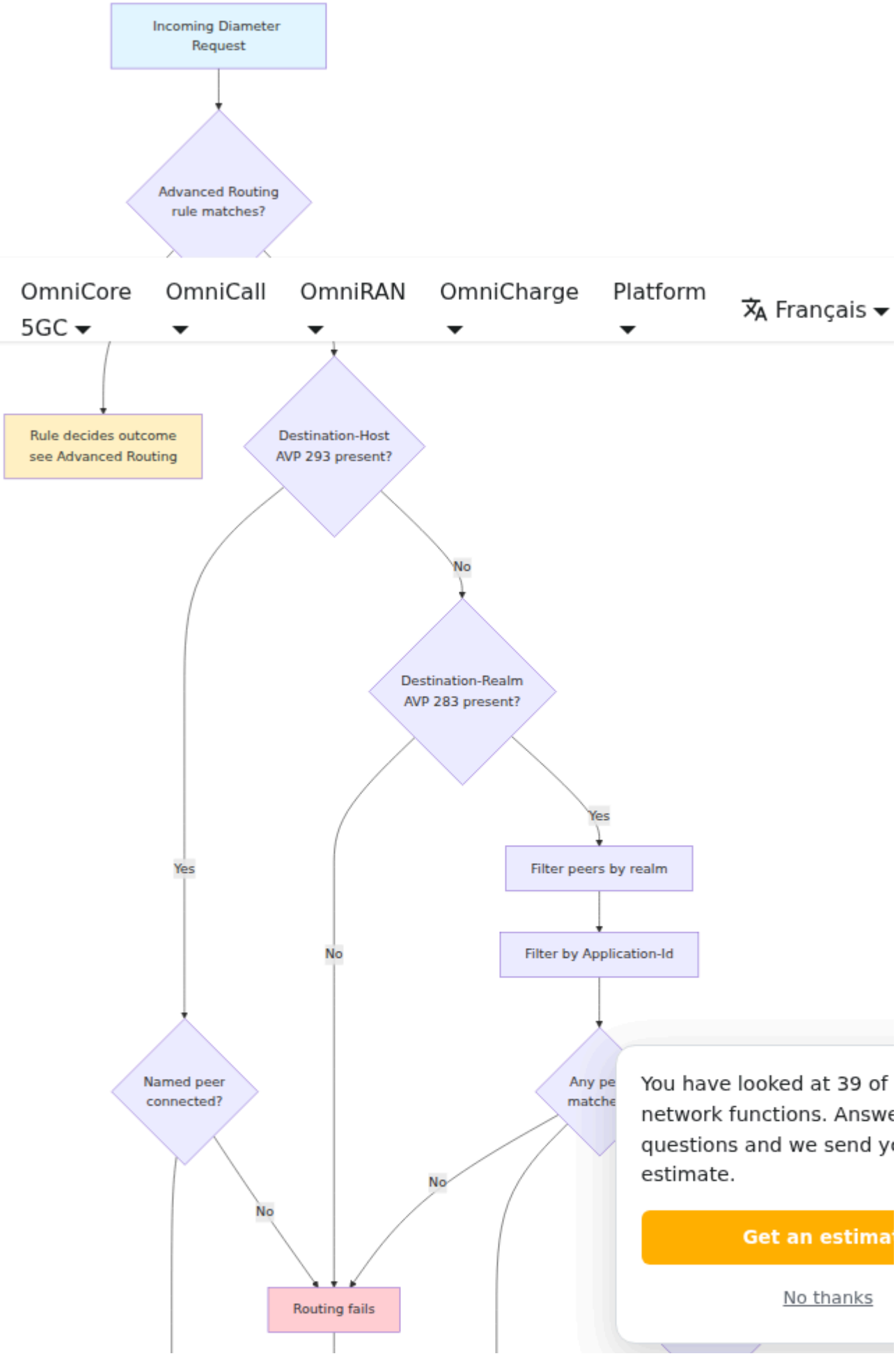
Le Routage Avancé décrit les remplacements basés sur le contenu. Ceux-ci remplacent le routage sur les valeurs AVP telles que les modèles IMSI, la traduction de domaine, et les décisions de pair d'origine. Cette page détaille également **l'affinité de session** via l'AVP `Destination-Host`.

# Précédence de Décision de Routage

Le routage standard évalue chaque requête selon un ordre de précédence fixe. Le premier mécanisme applicable l'emporte.

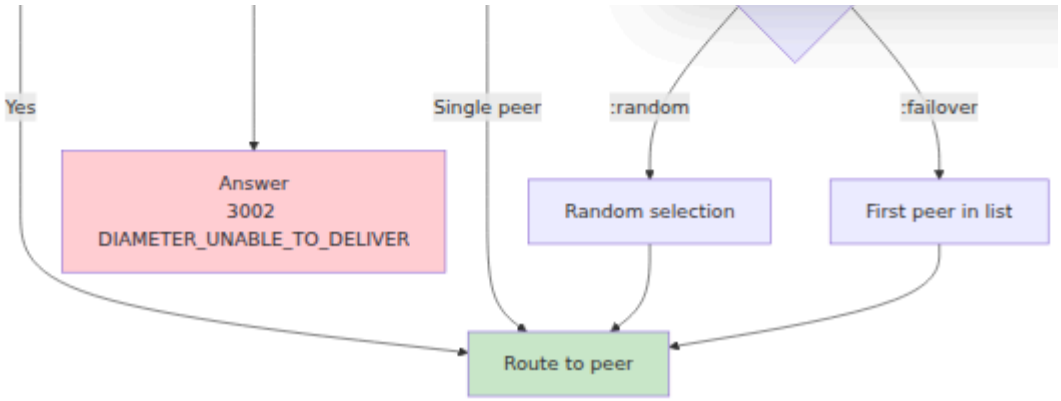
1. **AVP `Destination-Host` (293)** - C'est le mécanisme de la plus haute priorité. Si la requête contient un AVP `Destination-Host`, le DRA route directement vers cet hôte. Cela donne un contrôle explicite du routage au niveau de l'hôte. Si le pair nommé n'est pas connecté, le routage échoue.
2. **AVP `Destination-Realm` (283) + `Application-Id`** - Si le `Destination-Host` est absent, le DRA sélectionne un pair connecté qui annonce le support pour le **domaine** cible. Le DRA filtre ensuite les pairs candidats par l'**Application-Id** du message. Seuls les pairs qui annoncent le support pour cette application restent. Les pairs annoncent ce support lors de l'Échange de Capacités. Lorsque plus d'un pair correspond, l'algorithme de sélection de pair configuré choisit entre eux.
3. **Échec de relais / routage** - Si ni un `Destination-Host` accessible ni un pair correspondant domaine-plus-application n'est disponible, le DRA ne peut pas livrer la requête. Le DRA renvoie une réponse `3002 DIAMETER_UNABLE_TO_DELIVER`.

Le diagramme de flux suivant montre le chemin de décision complet. Il inclut le point où le Routage Avancé peut intercepter la requête avant que le routage standard ne s'exécute.






You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks




Seuls les pairs dans l'état **connecté** sont éligibles à chaque étape. Le DRA exclut automatiquement les pairs déconnectés ou hors service du filtrage de domaine et d'application. Une requête `Destination-Host` visant un pair déconnecté échoue immédiatement.

# Sélection de Pair

Lorsque plusieurs pairs connectés correspondent aux critères de domaine et d'Application-Id, le paramètre global `peer_selection_algorithm` détermine quel pair reçoit la requête. C'est un paramètre à l'échelle du service. Vous ne le configurez pas par domaine ou par pair.

```
config :dra, :service, %{
  # ... DRA identity and network settings ...
  peer_selection_algorithm: :random
}
```

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| `peer_selection_algorithm` | Atom | Non | `:random` | Stratégie d'équilibrage de charge lorsque plusieurs pairs connectés correspondent à une requête. `:random` sélectionne un pair aléatoire dans l'ensemble correspondant. `:failover` sélectionne toujours le premier pair dans la liste configurée. Il utilise les pairs restants uniquement lorsque le premier n'est pas disponible. |

- **`:random`** - Répartit la charge de manière uniforme entre tous les pairs correspondants. Utilisez ceci lorsque les pairs sont interchangeables et de coût égal.
- **`:failover`** - Privilégie un seul pair principal. Il traite la liste de pairs ordonnée comme une séquence de priorité. Utilisez ceci pour des conceptions actives/de secours. Un pair spécifique transporte le trafic tant qu'il reste connecté.

Voir Configuration pour le bloc de configuration complet du service et les définitions de pairs.

# Lorsque Aucun Pair n'est Disponible

L'évaluation de la précédence peut échouer à identifier une destination accessible. Cela se produit lorsque le pair `Destination-Host` est déconnecté, lorsque aucun pair connecté n'annonce le domaine cible, lorsque aucun pair correspondant ne prend en charge l'Application-Id, ou lorsque la requête manque des AVP nécessaires pour prendre une décision. Dans chacun de ces cas, le DRA ne peut pas transférer la requête.

Le DRA renvoie alors une réponse avec `Result-Code` défini sur `3002 DIAMETER_UNABLE_TO_DELIVER`, comme défini dans RFC 6733 §7.1.5. Cela informe le pair d'origine que le DRA a reçu la requête mais n'a pas pu la livrer à un prochain saut approprié. L'initiateur peut alors appliquer sa propre logique de réessai ou de secours.

# Référence

## Codes de Résultat

| Code | Nom | Description | Référence |
|---|---|---|---|
| 2001 | DIAMETER_SUCCESS | Le DRA a complété la requête avec succès | RFC 6733 §7.1.1 |
| 3002 | DIAMETER_UNABLE_TO_DELIVER | Le DRA n'a pas pu livrer la requête à un prochain saut approprié. Aucun pair correspondant et connecté n'était disponible | RFC 6733 §7.1.3 |

## Codes AVP Communs

| Code | Nom AVP | Type | Utilisation | Référence |
|---|---|---|---|---|
| 1 | User-Name | UTF8String | Identifiant de l'abonné (IMSI dans 3GPP) | RFC 6733 §8.14 |
| 264 | Origin-Host | DiameterIdentity | Nom d'hôte du pair d'origine | RFC 6733 §6.3 |
| 268 | Result-Code | Unsigned32 | Code de résultat standard | RFC 6733 §7.1 |
| 283 | Destination-Realm | DiameterIdentity | Domaine cible | RFC 6733 §6.6 |
| 293 | Destination-Host | DiameterIdentity | Hôte cible (optionnel) | RFC 6733 §6.5 |
| 296 | Origin-Realm | DiameterIdentity | Domaine source | RFC 6733 §6.4 |
| 297 | Experimental-Result | Grouped | Code de résultat spécifique au fournisseur | RFC 6733 §7.6 |

## Codes de Commande Communs

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

| Code | Nom de Commande | Description | Référence |
|---|---|---|---|
| 257 | CER/CEA | Demande/Réponse d'Échange de Capacités | RFC 6733 §5.3 |
| 258 | RAR/RAA | Demande/Réponse de Ré-authentification | RFC 6733 §8.3 |
| 274 | ASR/ASA | Demande/Réponse d'Interruption de Session | RFC 6733 §8.5 |
| 275 | STR/STA | Demande/Réponse de Terminaison de Session | RFC 6733 §8.4 |
| 280 | DWR/DWA | Demande/Réponse de Surveillance de Dispositif | RFC 6733 §5.5 |
| 282 | DPR/DPA | Demande/Réponse de Déconnexion de Pair | RFC 6733 §5.4 |
| 316 | ULR/ULA | Demande/Réponse de Mise à Jour de Localisation (S6a) | 3GPP TS 29.272 |
| 317 | CLR/CLA | Demande/Réponse d'Annulation de Localisation (S6a) | 3GPP TS 29.272 |
| 318 | AIR/AIA | Demande/Réponse d'Information d'Authentification (S6a) | 3GPP TS 29.272 |
| 321 | PUR/PUA | Demande/Réponse de Purge d'UE (S6a) | 3GPP TS 29.272 |

## Identifiants d'Application 3GPP Communs

Les pairs annoncent leurs Application-Ids supportés lors de l'Échange de Capacités. Le routage standard filtre les pairs candidats par l'Application-Id de

la requête.

| Application-Id | Interface | Description | Référence |
|---|---|---|---|
| 16777251 | S6a/S6d | Authentification et données d'abonnement MME/SGSN ↔ HSS | 3GPP TS 29.272 |
| 16777252 | S13/S13' | Vérification d'identité d'équipement MME ↔ EIR | 3GPP TS 29.272 |
| 16777238 | Gx | Contrôle de politique et de facturation PCEF ↔ PCRF | 3GPP TS 29.212 |
| 16777267 | S9 | Politique de roaming PCRF domicile ↔ PCRF visité | 3GPP TS 29.215 |
| 16777272 | S6b | PDN-GW ↔ 3GPP AAA pour accès non-3GPP | 3GPP TS 29.273 |
| 16777216 | Cx | I-CSCF/S-CSCF ↔ HSS enregistrement IMS | 3GPP TS 29.229 |
| 16777217 | Sh | AS ↔ HSS données utilisateur IMS | 3GPP TS 29.329 |
| 16777255 | SLg | Services de localisation MME/SGSN ↔ GMLC | 3GPP TS 29.172 |
| 16777291 | SLh | GMLC ↔ HSS informations d'abonné de localisation | 3GPP TS 29.173 |
| 16777302 | Sy | Liaison de session PCRF ↔ OCS | 3GPP TS 29.219 |
| 16777310 | S6m | MTC-IWF ↔ HSS/HLR pour dispositifs M2M | 3GPP TS 29.336 |

| Application-Id | Interface | Description | Référence |
|---|---|---|---|
| 16777312 | S6c | SMS-SC/IP-SM-GW ↔ HSS routage SMS | 3GPP TS 29.338 |
| 16777345 | S6t | SCEF ↔ HSS événements de surveillance | 3GPP TS 29.336 |
| 16777236 | Rx | AF ↔ PCRF autorisation média | 3GPP TS 29.214 |

Les applications de facturation (Gy/Ro) suivent 3GPP TS 32.299.

# Documentation Connexe

- Configuration - Référence complète de configuration du service et des pairs.
- Routage Avancé - Règles de routage basées sur le contenu, affinité de session via `Destination-Host`, et remplacements pilotés par des règles qui prennent la priorité sur le routage standard.

# Fonction de Recherche d'Abonnés (SLF)

La Fonction de Recherche d'Abonnés (SLF) apprend dynamiquement quel élément du réseau sert chaque abonné. Elle observe le trafic de signalisation Diameter qui passe par le DRA. Elle utilise les liaisons apprises pour acheminer les demandes ultérieures, telles que les requêtes de service de localisation, directement vers le nœud de service correct. Elle n'a pas besoin de règles de routage statiques.

Ceci est utile pour les demandes de service de localisation SLg/SLh. Le GMLC doit atteindre le MME de service pour un abonné donné mais ne sait pas quel MME c'est.

3GPP TS 29.172 et 3GPP TS 29.173 décrivent le concept de SLF pour les services de localisation. 3GPP TS 29.272 définit l'enregistrement des abonnés.

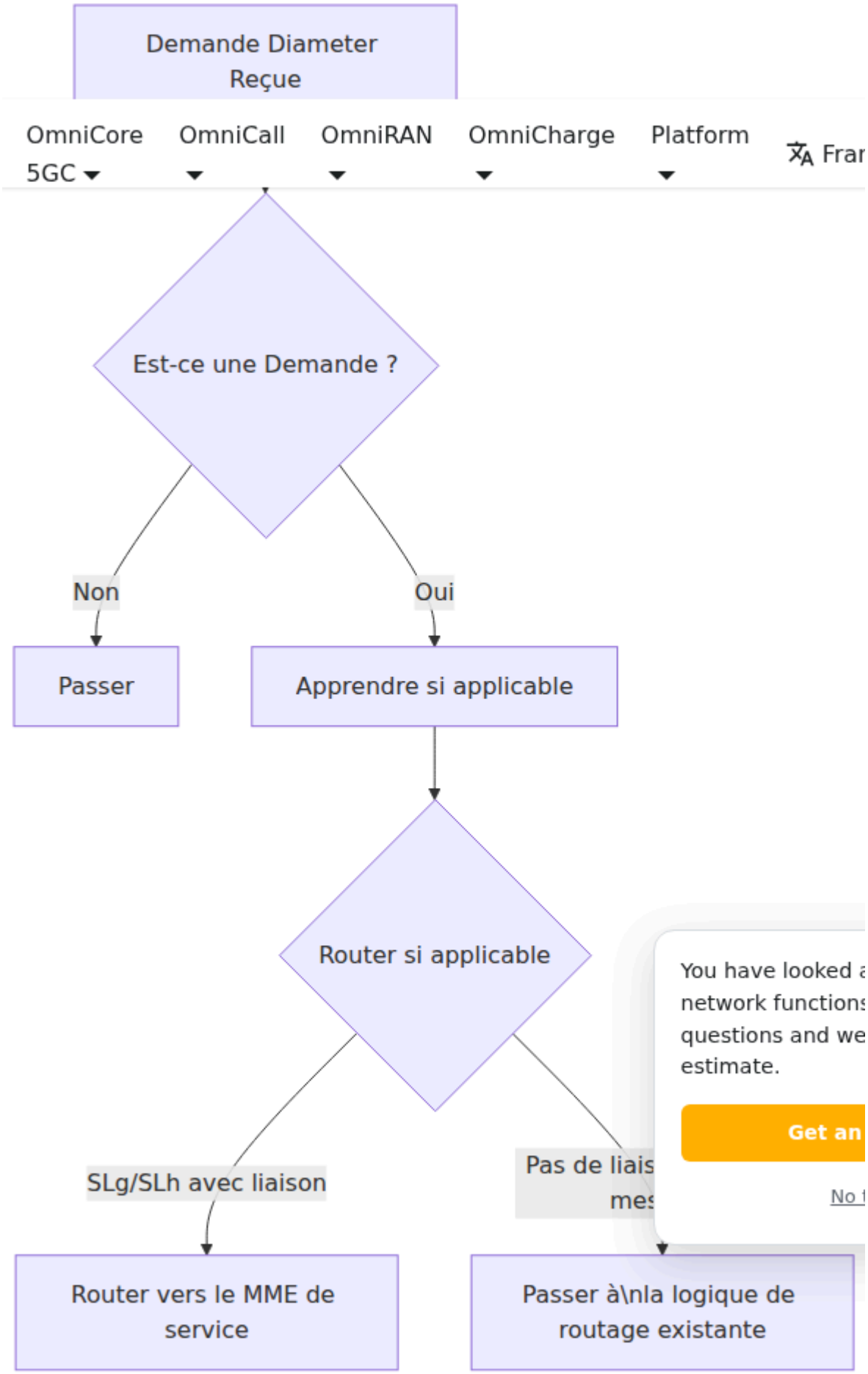
## Comment ça fonctionne

Le module fonctionne en deux phases : **apprentissage** et **routage**.

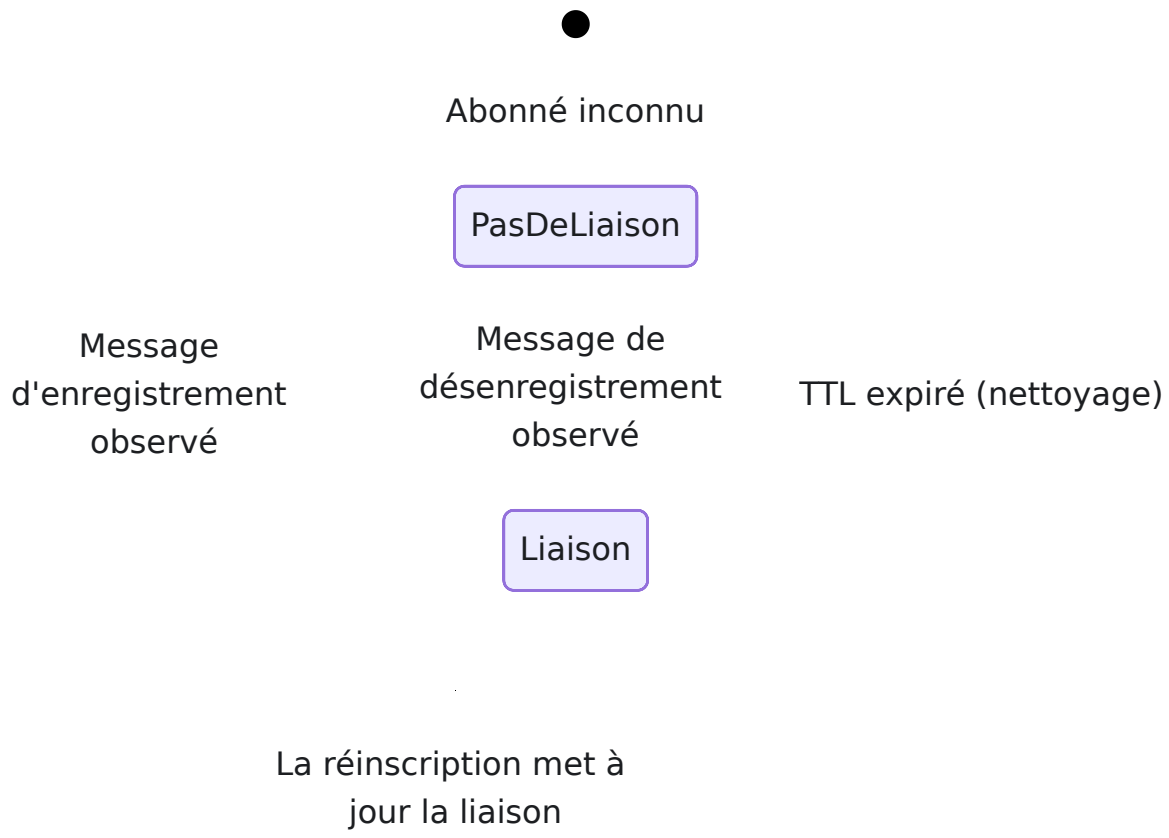
Pendant l'**apprentissage**, le module observe les messages d'enregistrement et de désenregistrement qui circulent à travers le DRA. Lorsqu'un abonné s'enregistre (par exemple via une demande de mise à jour de localisation S6a), le module enregistre quel élément du réseau sert maintenant cet abonné.

Pendant le **routage**, une demande peut arriver qui doit atteindre le nœud de service d'un abonné (par exemple une demande de fourniture de localisation SLg). Le module recherche la liaison et route directement vers le pair correct.

Si aucune liaison n'existe pour un abonné, la demande passe à la logique de routage existante, y compris le routage avancé.

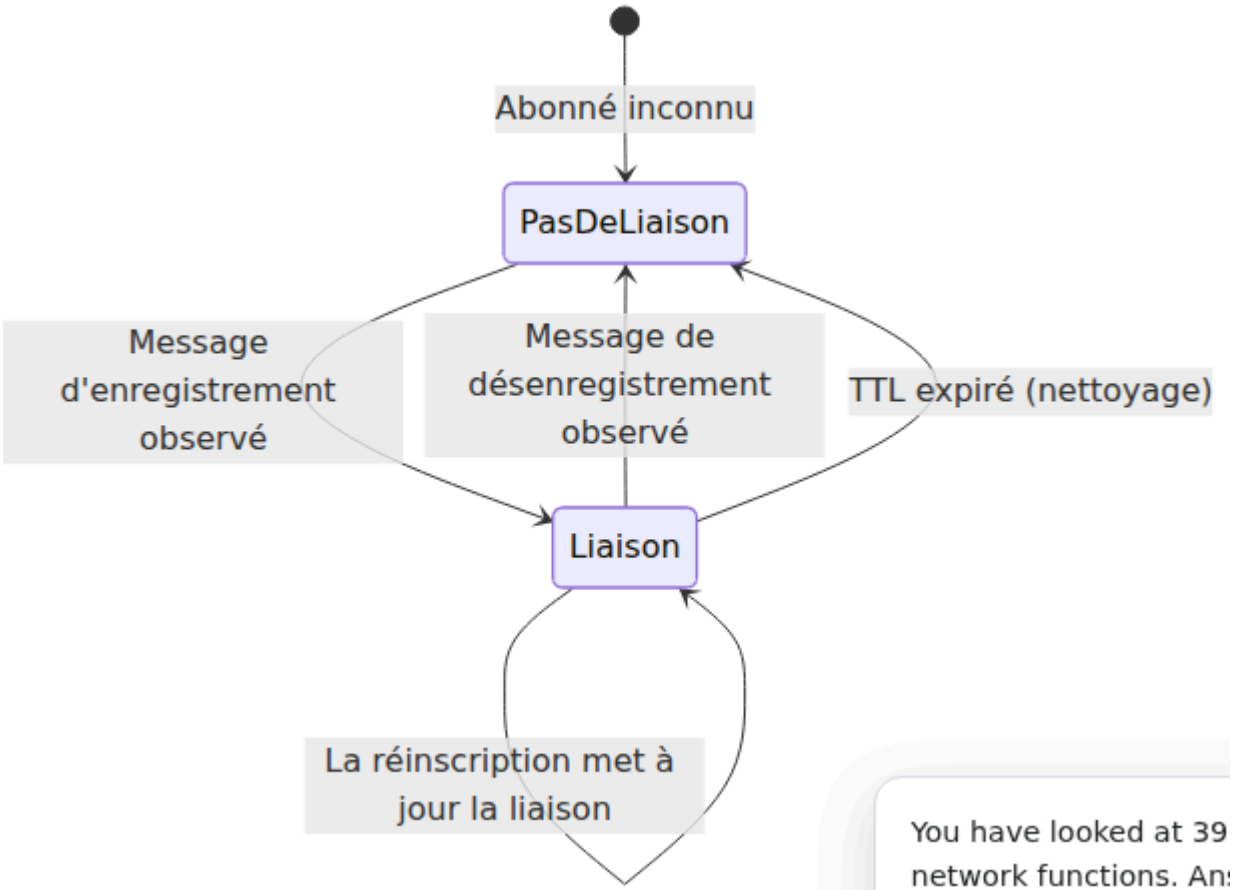

Demande Diameter Reçue
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Fran
Est-ce une Demande ?
Non
Oui
Passer
Apprendre si applicable
Router si applicable
SLg/SLh avec liaison
Pas de liais
mes
Router vers le MME de service
Passer à\nla logique de routage existante
You have looked a
network functions
questions and we
estimate.
Get an
No t

# Cycle de Vie des Liaisons




## Apprentissage : Messages qui créent des liaisons

Le module apprend les liaisons abonné-nœud de service à partir des messages Diameter suivants :




You have looked at 39
network functions. An

| Interface | Message | Code de Commande | Liaison Créée | Source IMSI |
|---|---|---|---|---|
| S6a | Demande de Mise à jour de Localisation (ULR) | 316 | `serving_mme` | AVP User-Name (1) |
| Gx | Demande de Contrôle de Crédit Initiale (CCR-I) | 272 (CC-Request-Type=1) | `serving_pgw` | AVP Subscription-Id (443) |
| Cx | Demande d'Attribution de Serveur (SAR) | 301 (Server-Assignment-Type=1) | `serving_cscf` | AVP Public-Identity (601) |

## Apprentissage : Messages qui suppriment des liaisons

| Interface | Message | Code de Commande | Liaison Supprimée |
|---|---|---|---|
| S6a | Demande d'Annulation de Localisation (CLR) | 317 | `serving_mme` |
| S6a | Demande de Purge d'UE (PUR) | 321 | `serving_mme` |
| Gx | Demande de Contrôle de Crédit de Fin (CCR-T) | 272 (CC-Request-Type=3) | `serving_pgw` |
| Cx | Demande d'Attribution de Serveur (SAR) | 301 (Server-Assignment-Type=4) | `serving_cscf` |

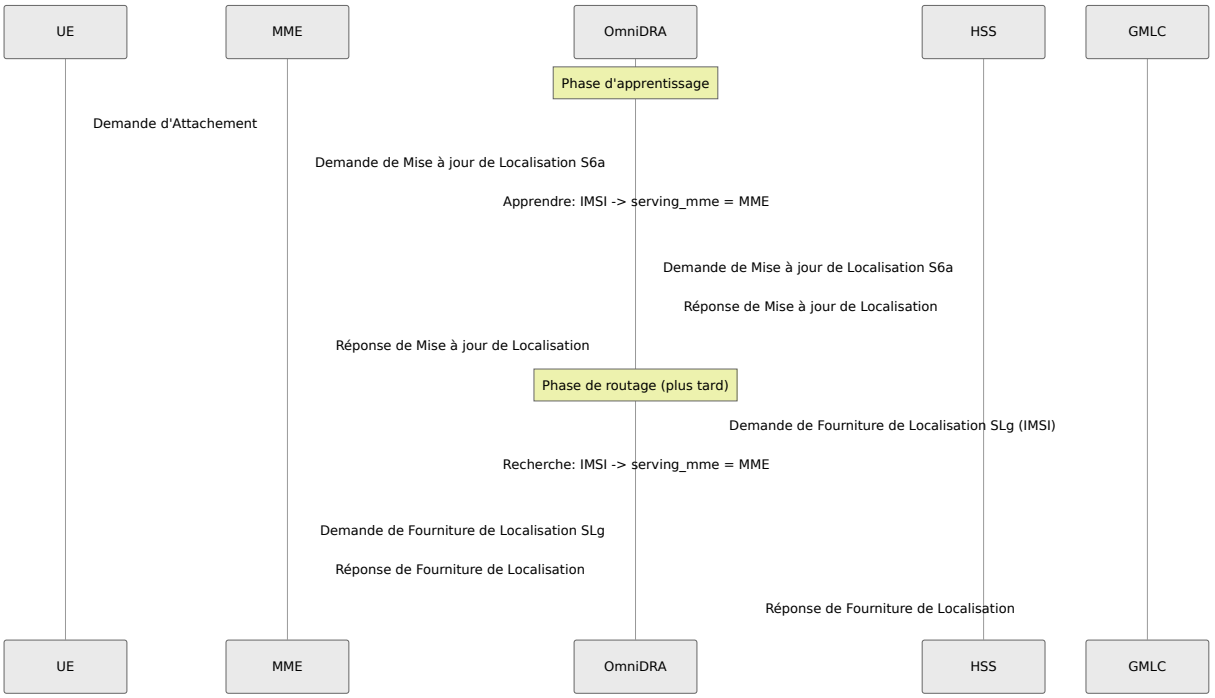
Lorsque le module supprime la dernière liaison pour un abonné, il supprime l'ensemble de l'enregistrement de l'abonné de la table.

## Routage : Messages qui utilisent des liaisons

| Interface | Message | Code de Commande | Liaison Utilisée |
|---|---|---|---|
| SLg | Demande de Fourniture de Localisation (PLR) | 8388620 | `serving_mme` |
| SLh | Demande d'Info de Routage LCS (RIR) | 8388622 | `serving_mme` |

Le DRA relaie une demande routée par une liaison apprise au nœud de service. Il utilise le délai de relais de base provenant du paramètre `diameter.request_timeout` (voir Connectivité des Pairs ou Configuration). Le module ne définit pas son propre délai pour ces demandes routées.

# Exemple de Flux : Demande de Service de Localisation




# Position dans le Pipeline

La recherche d'abonnés s'exécute **avant** le routage avancé dans le pipeline de traitement des demandes. Si le SLF trouve une liaison et remplace la route, le routage avancé s'exécute toujours. La route SLF a la priorité.



# Configuration

Configurez le module sous `module_subscriber_lookup` dans `config/runtime.exs`.

```
module_subscriber_lookup: %{
  # Activer ou désactiver le module
  enabled: false,

  # Durée de conservation des liaisons avant expiration (secondes)
  binding_ttl_seconds: 86400
}
```

## Paramètres

| Paramètre | Type | Requis | Par Défaut | Desc |
|---|---|---|---|---|
| `enabled` | Booléen | Oui | `false` | Activer ou désactive recherche d'abonnés toutes les demandes modification et le mo aucune liaison. |
| `children` | Liste | Non | `[]` | Processus que le sup démarre lorsqu'il est sur `[DRA.Router.Module` afin que le processus s'exécute. |
| `binding_ttl_seconds` | Entier | Non | `86400` | Temps en secondes a d'abonné n'expire. L supprime ensuite. La de 24 heures. Le net moitié de l'intervalle minimum de 60 secc |

## Exemples de Configuration

### Déploiement Standard

Utilisez ceci pour la plupart des réseaux où les abonnés se réenregistrent dans les 24 heures.

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

**Comment ça fonctionne :** Le module apprend les liaisons MME/PGW/CSCF à partir du trafic passant et les conserve pendant 24 heures. Le DRA route automatiquement les demandes SLg et SLh vers le MME de service correct. Le module rafraîchit une liaison chaque fois qu'il observe un nouveau message d'enregistrement pour le même abonné.

**Cas d'utilisation :** Réseaux avec des exigences de service de localisation (LCS). Le GMLC doit atteindre le MME de service sans mappages statiques abonné-MME.

**Environnement à Forte Mobilité**

Utilisez ceci pour les réseaux avec une mobilité fréquente des abonnés. Les liaisons obsolètes doivent expirer rapidement.

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

**Comment ça fonctionne :** Les liaisons expirent après 1 heure. Utilisez ceci lorsque les abonnés se déplacent souvent entre les MMEs. Les liaisons obsolètes pourraient alors causer des demandes de localisation mal orientées. L'intervalle de nettoyage s'exécute toutes les 30 minutes.

**Cas d'utilisation :** Réseaux urbains denses ou événements avec une forte mobilité des abonnés.

# Métriques

## Mises à Jour des Liaisons

**Métrique** : `diameter.subscriber_lookup.binding.update` **Type** : Compteur **Description** : Le DRA incrémente ce compteur chaque fois qu'il crée ou met à jour une liaison d'abonné. **Étiquettes** :

- `imsi` - IMSI de l'abonné

- `binding_type` - Type de liaison : `serving_mme`, `serving_pgw`, ou `serving_cscf`
- `serving_host` - Origin-Host de l'élément de réseau de service

## Suppressions de Liaisons

**Métrique** : `diameter.subscriber_lookup.binding.delete` **Type** : Compteur **Description** : Le DRA incrémente ce compteur chaque fois qu'un message de désenregistrement supprime une liaison d'abonné. **Étiquettes** :

- `imsi` - IMSI de l'abonné
- `binding_type` - Type de liaison supprimée

## Demandes Routées

**Métrique** : `diameter.subscriber_lookup.route.count` **Type** : Compteur **Description** : Le DRA incrémente ce compteur chaque fois qu'il route une demande SLg/SLh avec une liaison apprise. **Étiquettes** :

- `imsi` - IMSI de l'abonné
- `binding_type` - Type de liaison que le DRA a utilisé pour le routage (actuellement toujours `serving_mme`)
- `serving_host` - Origin-Host du pair auquel le DRA a routé la demande

# Dépannage

## Demandes de Localisation Non Routées vers le MME de Service

**Symptômes** : Le DRA ne route pas les demandes de fourniture de localisation SLg ou les demandes d'info de routage LCS SLh vers le MME attendu. Elles passent à la routage par défaut à la place.

**Causes possibles** :

- L'abonné ne s'est pas enregistré via ce DRA, donc aucune liaison n'existe.

- La liaison a expiré. Le TTL est dépassé.
- Le pair MME de service n'est pas connecté à ce DRA.
- `enabled` est réglé sur `false`.

**Résolution** :

1. Confirmez que le module est activé dans la configuration.
2. Vérifiez que le trafic ULR S6a pour l'abonné passe par ce DRA. Le module apprend les liaisons uniquement à partir du trafic observé.
3. Confirmez que `binding_ttl_seconds` est suffisamment long pour couvrir l'écart entre l'enregistrement et la demande de localisation.
4. Confirmez que le MME de service est connecté en tant que pair.

## Liaisons Non Apprises

**Symptômes** : La table de liaison reste vide malgré le passage du trafic S6a/Gx/Cx à travers le DRA.

**Causes possibles** :

- Le module n'est pas activé.
- Le processus du module n'est pas en cours d'exécution. Vérifiez le superviseur.
- Les messages ne contiennent pas un IMSI valide dans l'AVP attendu.

**Résolution** :

1. Confirmez `enabled: true` dans la configuration.
2. Confirmez que le DRA a redémarré après le changement de configuration.
3. Vérifiez les journaux de débogage du DRA pour les entrées `SubscriberLookup: Learned`. Celles-ci confirment l'activité de liaison.

## Liaisons Obsolètes Causant des Demandes Mal Routées

**Symptômes** : Le DRA route les demandes de localisation vers un MME qui ne sert plus l'abonné.

**Causes possibles** :

- La demande d'annulation de localisation (CLR) ou la demande de purge d'UE (PUR) n'est pas passée par ce DRA.
- `binding_ttl_seconds` est réglé trop haut pour le modèle de mobilité du réseau.

**Résolution** :

1. Réduisez `binding_ttl_seconds` pour correspondre à l'intervalle de réenregistrement attendu des abonnés.
2. Confirmez que tout le trafic de désenregistrement S6a passe par ce DRA.

# Référence

## Identifiants d'Application

| ID | Interface | Description | Référence |
|---|---|---|---|
| 16777251 | S6a/S6d | Authentification MME/SGSN et gestion des abonnements vers HSS | 3GPP TS 29.272 |
| 16777238 | Gx | Contrôle de politique et de facturation PCEF vers PCRF | 3GPP TS 29.212 |
| 16777216 | Cx | Enregistrement IMS vers HSS I-CSCF/S-CSCF | 3GPP TS 29.229 |
| 16777255 | SLg | Services de localisation GMLC vers MME | 3GPP TS 29.172 |
| 16777291 | SLh | Info de routage LCS GMLC vers HSS/DRA | 3GPP TS 29.173 |

# Codes de Commande

| Code | Nom | Interface | Description |
|---|---|---|---|
| 272 | Demande/Réponse de Contrôle de Crédit (CCR/CCA) | Gx | Contrôle de politique et de facturation au niveau de la session |
| 301 | Demande/Réponse d'Attribution de Serveur (SAR/SAA) | Cx | Enregistrement et désenregistrement IMS |
| 316 | Demande/Réponse de Mise à jour de Localisation (ULR/ULA) | S6a | Mise à jour de la localisation de l'abonné au MME |
| 317 | Demande/Réponse d'Annulation de Localisation (CLR/CLA) | S6a | Annulation de localisation initiée par HSS |
| 321 | Demande/Réponse de Purge d'UE (PUR/PUA) | S6a | Purge d'UE initiée par MME |
| 8388620 | Demande/Réponse de Fourniture de Localisation (PLR/PLA) | SLg | Demande de service de localisation vers le MME de service |
| 8388622 | Demande/Réponse d'Info de Routage LCS (RIR/RIA) | SLh | Recherche d'info de routage LCS |

## Types de Liaisons

| Type de Liaison | Appris de | Utilisé par | Description |
|---|---|---|---|
| `serving_mme` | S6a ULR | SLg PLR, SLh RIR | Le MME servant actuellement l'abonné |
| `serving_pgw` | Gx CCR-I | - | Le PGW gérant la session de l'abonné (réservé pour le routage futur) |
| `serving_cscf` | Cx SAR (Enregistrement) | - | Le S-CSCF servant l'enregistrement IMS de l'abonné (réservé pour le routage futur) |

# Dépannage

Ce guide couvre les problèmes les plus courants dans le fonctionnement d'OmniDRA. Chaque entrée liste les symptômes observés par un opérateur, les causes probables et un ensemble ordonné d'étapes. Utilisez les étapes pour isoler et résoudre le problème. Pour les noms de métriques dans ce guide, voir Métriques.

## Arbre de Décision de Diagnostic

Utilisez ce flux pour trouver où se situe un problème avant de passer à une section spécifique.




You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks

# Règles de Routage

## Règle Non Correspondante

**Symptômes** : Une règle de routage avancée configurée ne se déclenche jamais. Le trafic passe par le routage standard ou par une règle ultérieure à la place.

**Causes possibles** :

- Une ou plusieurs conditions de filtre ne correspondent pas au contenu réel du message.
- Le code AVP dans le filtre ne correspond pas à l'AVP utilisé par l'application Diameter en question.
- Un motif d'expression régulière est incorrect ou mal ancré.
- Le type de message est en dehors du champ `match` de la règle. Par exemple, un filtre uniquement pour les requêtes s'applique à une réponse.
- La règle utilise le mauvais type de filtre pour le module.

**Résolution** :

1. Vérifiez chaque condition de filtre par rapport à un message capturé. Toutes les conditions d'une règle doivent correspondre pour que la règle se déclenche.
2. Confirmez que les codes AVP correspondent à l'application Diameter. Voir la référence de code AVP dans Routage Avancé.
3. Testez les motifs d'expressions régulières par rapport à de vraies valeurs AVP avant de les déployer.
4. Confirmez que le type de message est dans le champ `match` de la règle.
5. Passez en revue les types de filtres disponibles dans Routage Avancé. Confirmez que vous utilisez le bon pour le module.

## Routage Inattendu

**Symptômes** : Le DRA transmet des requêtes, mais à un pair différent de celui prévu. Alternativement, le DRA utilise le routage standard alors que vous vous

attendiez à ce qu'une règle s'applique.

**Causes possibles** :

- Ordre des règles. Le routage avancé utilise le premier match gagne, donc une règle antérieure et plus large peut masquer une règle ultérieure et plus spécifique.
- Le nom du pair prévu est incorrect, ou le pair n'est pas accessible.
- Deux règles ont des filtres qui se chevauchent, et la mauvaise l'emporte.
- Aucune règle ne correspond, donc le Routage Standard prend le relais.

**Résolution** :

1. Passez en revue l'ordre des règles. Placez les règles plus spécifiques avant celles plus larges. La première règle correspondante gagne. Voir Routage Avancé.
2. Confirmez que les noms des pairs correspondent exactement à la configuration et que le pair cible est connecté (`diameter_peer_status` rapporte `1`).
3. Vérifiez les règles conflictuelles avec des filtres qui se chevauchent. Réorganisez-les ou resserrez-les.
4. Confirmez que le comportement de passage dans Routage Standard correspond à ce que vous observez lorsque aucune règle ne s'applique.

## Transformation Non Appliquée

**Symptômes** : Une règle de routage correspond, mais une transformation d'ajout/modification/suppression d'AVP n'a pas d'effet sur le message.

**Causes possibles** :

- La transformation cible un code AVP qui est incorrect pour le cas d'utilisation.
- Une action `edit` a été exécutée sur un AVP qui n'existe pas dans le message. `edit` modifie un AVP existant mais ne crée jamais un nouvel AVP.
- Les filtres de la règle ne correspondent pas au flux de message prévu, donc le message n'atteint jamais la transformation.

- Le filtre de type de paquet (`:request` vs `:answer`) exclut le message.
- L'action ne prend pas en charge le type de paquet. `edit` s'applique uniquement aux requêtes.

**Résolution** :

1. Confirmez que les codes AVP sont corrects pour votre cas d'utilisation. Voir la référence de code AVP dans Routage Avancé.
2. Pour une action `edit`, confirmez que l'AVP existe déjà dans le message. Utilisez `add` si l'AVP peut être absent.
3. Confirmez que les filtres de la règle correspondent au flux de message prévu.
4. Confirmez que tout filtre de type de paquet (`:request` vs `:answer`) est correct pour la direction dans laquelle vous transformez.
5. Confirmez que l'action prend en charge le type de paquet. `edit` fonctionne uniquement sur les requêtes.

---

# Connectivité des Pairs

## Échecs de Connexion de Pair

**Symptômes** : Un pair configuré n'atteint jamais l'état connecté. `diameter_peer_status{origin_host="..."}` reste `0`, et le routage vers ce pair échoue.

**Causes possibles** :

- Un pare-feu bloque le `listen_port` configuré (par défaut `3868`) ou le port du pair.
- La configuration du pair a un `ip`, `port` ou `transport` incorrect.
- Transport non correspondant. Un côté est configuré pour `:diameter_tcp` et l'autre pour `:diameter_sctp`.
- L'Origin-Host annoncé par le pair ne correspond pas à l'`host` configuré. Les identités doivent correspondre exactement.

- `initiate_connection` est défini de la même manière aux deux extrémités, donc aucun côté ne compose, ou les deux composent.
- Le pair n'annonce pas une application prise en charge lors de l'Échange de Capacités (CER/CEA).
- Pour un pair sortant, le service du pair n'est pas en cours d'exécution ou n'écoute pas.

**Résolution** :

1. Confirmez que les règles de pare-feu permettent le trafic sur `listen_port` et le port du pair pour le transport configuré. SCTP nécessite le module SCTP du noyau et des règles de pare-feu qui autorisent SCTP, pas seulement TCP.
2. Confirmez que les valeurs `ip`, `port` et `transport` du pair correspondent exactement à l'extrémité réelle du pair.
3. Confirmez que les deux extrémités utilisent le même protocole de transport.
4. Confirmez que l'Origin-Host annoncé par le pair correspond à l'`host` configuré. Les identités Diameter doivent correspondre exactement pour le routage.
5. Vérifiez les rôles `initiate_connection`. Exactement un côté doit initier. Définissez `initiate_connection: true` sur le côté qui compose (généralement le DRA vers un HSS ou PCRF). Définissez `false` sur le côté qui attend (généralement vers un MME, SGSN ou P-GW).
6. Confirmez que le pair annonce au moins une application que le DRA sert lors de CER/CEA. Un pair sans application commune n'est pas utilisable pour le routage.
7. Pour un pair sortant, confirmez que le service distant est en cours d'exécution et écoute. Le DRA réessaie automatiquement les pairs inaccessibles.

# Flapping de Pair / Trafic Envoyé Trop Tôt Après Reconnexion

**Symptômes** : Après qu'un pair se soit reconnecté, le DRA envoie des requêtes à celui-ci, et elles échouent immédiatement ou expirent pendant que le pair est

encore en cours de stabilisation. Alternativement, un pair qui s'est récemment rétabli semble utilisable mais perd du trafic.

**Causes possibles** :

- Le pair est instable au niveau du réseau et se reconnecte de manière répétée (flapping).
- Le temps de stabilisation au niveau de l'application est trop court pour ce pair, donc le DRA route le trafic avant que le pair soit prêt.
- Un temps de stabilisation excessivement long retarde la récupération légitime et prolonge la fenêtre de panne.

**Résolution** :

1. Enquêtez d'abord sur l'instabilité du transport sous-jacent (chemin réseau, basculement de chemin SCTP, ou redémarrages côté pair). Un pair en flapping est un symptôme, pas un problème de configuration de routage.

2. Ajustez le temps de stabilisation afin qu'un pair fraîchement connecté doive rester actif pendant un intervalle défini avant que le DRA ne route le trafic vers celui-ci. Un pair qui se déconnecte à l'intérieur de cette fenêtre n'entre jamais dans l'ensemble routable. Cela maintient le trafic hors des pairs en flapping.

```
config :dra, :gateway,
  # Maintenez un pair fraîchement connecté pendant ce nombre de
ms avant de router le trafic vers lui.
  # 0 = router immédiatement une fois que CER/CEA est terminé.
  peer_settle_time_ms: 2_000,

  # Remplacement optionnel (ms) pour le minuteur de
surveillance RFC 3539 Tw.
  # watchdog_timer_ms: 30_000
```

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `peer_settle_time_ms` | Entier | Non | `0` | Millisecondes qu'un pair nouvellement connecté doit rester actif avant que le DRA ne route le trafic vers lui. `0` route immédiatement apr��s que CER/CEA soit terminé. Si le pair se déconnecte pendant la fenêtre, le DRA ne le marque jamais comme routable. Cela maintient le trafic hors des pairs en flapping. |
| `watchdog_timer_ms` | Entier | Non | `30000` | Remplacement optionnel pour le minuteur de surveillance RFC 3539 Tw, l'intervalle de vivacité du Dispositif de Surveillance (DWR/DWA). |

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| | | | | Laissez-le non défini à moins que vous n'ayez une raison spécifique de changer la cadence de vivacité. |

3. `diameter_peer_status` reflète la routabilité réelle. Un pair à l'intérieur de sa fenêtre de stabilisation continue de signaler comme non encore routable dans les métriques. Utilisez cela pour confirmer que le comportement de stabilisation fonctionne comme vous l'attendez.

4. Augmentez `peer_settle_time_ms` pour les pairs en flapping chroniques afin de supprimer les envois prématurés. Gardez-le bas (ou `0`) pour les pairs stables afin de minimiser le temps de récupération.

> Le DRA promeut un pair en reconnection à routable immédiatement après que le temps de stabilisation au niveau de l'application s'est écoulé. Il n'attend pas la période de délai implicite de reconnexion du minuteur de surveillance RFC 3539. Le temps de stabilisation ci-dessus est le contrôle unique et explicite sur la durée pendant laquelle le DRA maintient un pair en montée.

## Connexions de Pair Non Autorisées

**Symptômes** : Le DRA rejette la connexion d'un pair. Alternativement, `diameter_peer_unauthorized_connection_count_total` s'incrémente avec un `origin_host` / `peer_ip` inattendu.

**Causes possibles** :

- Le pair se connectant n'est pas dans la configuration `peers`, et `allow_undefined_peers_to_connect` est `false`.

- L'Origin-Host annoncé par le pair ne correspond à aucun `host` configuré.
- Un nœud réellement non autorisé ou mal configuré essaie de se connecter.

**Résolution** :

1. Confirmez si vous devez autoriser le pair. Si oui, ajoutez-le à la liste `peers` avec son `host`, `realm`, `ip`, `port` et `transport` exacts.

2. Confirmez que l'Origin-Host annoncé par le pair correspond exactement à l'`host` configuré. Le DRA considère un écart comme un pair inconnu.

3. Gardez `allow_undefined_peers_to_connect: false` en production et ajoutez les pairs explicitement. Activez-le uniquement dans des environnements contrôlés où tout pair peut se connecter.

4. Gardez `log_unauthorized_peer_connection_attempts: true` afin que le DRA enregistre les tentatives rejetées pour un examen de sécurité.

5. Enquêtez sur les étiquettes `peer_ip` et `origin_host` revendiquées sur `diameter_peer_unauthorized_connection_count_total`. Un taux soutenu d'une source inconnue justifie une enquête de sécurité.

| Paramètre | Type | Requis | dé |
| --- | --- | --- | --- |
| `allow_undefined_peers_to_connect` | Booléen | Non | `fa` |
| `log_unauthorized_peer_connection_attempts` | Booléen | Non | - |

## Problèmes d'Association SCTP

**Symptômes** : Un pair SCTP échoue à établir une association. Alternativement, une association établie se déconnecte ou échoue de manière inattendue. Les pairs TCP sur le même hôte se connectent normalement.

**Causes possibles** :

- Le module SCTP du noyau n'est pas chargé sur l'hôte.
- Les règles de pare-feu permettent TCP mais pas SCTP sur les adresses configurées.
- Une ou plusieurs adresses dans l'ensemble de multihoming ne sont pas routables depuis le pair.

- L'IP principale n'est pas dans la liste `additional_ips` du pair.
- Un seul point de terminaison est configuré pour le multihoming, donc le DRA n'atteint pas la redondance complète.
- Les paramètres SCTP du noyau régissent le timing de basculement de chemin, et cela diffère des attentes.

**Résolution** :

1. Confirmez que le module SCTP du noyau est chargé (le paquet `lksctp-tools` sur Linux).
2. Confirmez que les règles de pare-feu autorisent le trafic SCTP sur chaque IP configurée, pas seulement TCP.
3. Confirmez que toutes les adresses IP dans l'ensemble de multihoming sont routables dans les deux sens entre le DRA et le pair.
4. Pour un pair SCTP, incluez l'IP principale dans la liste `additional_ips`. Pour les pairs TCP, le DRA utilise uniquement `ip`, car TCP ne prend pas en charge le multihoming.
5. Configurez les deux points de terminaison pour le multihoming pour une redondance complète du chemin.
6. Si le timing de basculement n'est pas comme prévu, passez en revue les paramètres SCTP du noyau. Le timing de basculement de chemin dépend d'eux, pas de la configuration du DRA.
7. Voir Multihoming SCTP pour la configuration complète du transport et ses limitations.

---

# Livraison de Messages

## Messages Non Routés / 3002 UNABLE_TO_DELIVER

**Symptômes** : Les requêtes arrivent au DRA, mais le DRA y répond avec le Code de Résultat `3002` (DIAMETER_UNABLE_TO_DELIVER). Alternativement, le DRA ne les transmet pas du tout à la destination attendue.

**Causes possibles** :

- Aucun pair connecté ne correspond à l'Destination-Host ou Destination-Realm de la requête.
- Le pair cible est déconnecté, ou il est à l'intérieur de sa fenêtre de stabilisation et pas encore routable.
- Le pair cible n'annonce pas le support de l'application dans la requête.
- Une règle de routage avancée a sélectionné un pair qui n'est pas accessible.
- Aucun pair ne sert le domaine demandé. Cela donne `3003` (DIAMETER_REALM_NOT_SERVED), pas `3002`.

**Résolution** :

1. Confirmez qu'au moins un pair qui correspond à l'Destination-Host ou Destination-Realm est connecté. `diameter_peer_status` pour ce pair doit lire `1`.

2. Vérifiez si le pair cible est à l'intérieur de sa fenêtre de stabilisation. Un pair qui vient de se reconnecter n'est délibérément pas routable jusqu'à ce que `peer_settle_time_ms` s'écoule. Voir Flapping de Pair.

3. Confirmez que le pair annonce l'application dans la requête lors de CER/CEA. Le DRA exclut les pairs qui n'annoncent pas l'application de la sélection.

4. Si une règle de routage avancée cible un pair spécifique, confirmez que ce pair est connecté. Le DRA saute les pairs déconnectés. Voir Routage Avancé.

5. Distinguez `3002` de `3003`. `3002` signifie que le DRA n'a pas pu sélectionner un pair accessible. `3003` signifie qu'aucun pair ne sert le domaine demandé. Corrigez la configuration du domaine si vous observez `3003`.

6. Suivez le taux de réponses `3002` pour repérer des échecs de livraison systémiques :

   ```
   rate(diameter_peer_message_result_code_count_total{result_code="3
   [5m])
   ```

# Requêtes Sans Réponse (Délai d'Attente)

**Symptômes** : Le DRA transmet des requêtes à un pair, mais aucune réponse ne revient dans le délai d'attente.
`diameter_peer_unanswered_request_count_total` s'incrémente, et `diameter_peer_last_response_delay` augmente.

**Causes possibles** :

- Le pair a reçu la requête mais n'a pas répondu dans le délai `request_timeout` (par défaut `5000` ms).
- Le pair est surchargé ou lent.
- Un chemin réseau est dégradé. Cela est pertinent pour le multihoming SCTP lors du basculement.
- Le pair a traité la requête, mais la réponse a été perdue en transit.

**Résolution** :

1. Identifiez les pairs lents ou silencieux :

   ```
   topk(5, sum by (routed_to) (
     rate(diameter_peer_unanswered_request_count_total[5m])
   ))
   ```

2. Comparez avec `diameter_peer_last_response_delay` pour le même pair. Cela montre s'il est lent plutôt que silencieux.

3. Passez en revue la charge et la santé du pair. Des délais d'attente soutenus indiquent généralement un problème en aval, pas un problème de DRA.

4. Pour les pairs SCTP, vérifiez les événements de basculement de chemin qui coïncident avec les délais d'attente. Voir Multihoming SCTP.

5. Ajustez `request_timeout` uniquement si le temps de traitement légitime du pair dépasse le défaut. Une valeur plus élevée prolonge également le TTL des Métriques Étendues (`request_timeout` + 5 secondes).

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `request_timeout` | Entier | Non | `5000` | Millisecondes à attendre pour une réponse avant que le DRA considère la paire requête/réponse comme expirée. |

## UNKNOWN_SESSION_ID sur Gx / Gy Stateful

**Symptômes** : Les sessions stateful (contrôle de politique Gx, facturation en ligne Gy) échouent avec le Code de Résultat `5002` (DIAMETER_UNKNOWN_SESSION_ID). Typiquement, le pair rejette une mise à jour (CCR-U) ou une terminaison (CCR-T) même si la première (CCR-I) a réussi.

**Causes possibles** :

- Le DRA a routé une requête en milieu de session à un pair différent de celui qui détient l'état de session.
- Le pair qui détient la session a redémarré et a perdu sa table de sessions.
- Le DRA a routé vers un pair qui s'est reconnecté avant de restaurer son état. C'est une interaction de fenêtre de stabilisation.
- L'équilibrage de charge a réparti les requêtes pour un Session-Id sur plusieurs pairs.

**Résolution** :

1. Confirmez que toutes les requêtes avec le même Session-Id atteignent le même pair stateful. Pour Gx et Gy, l'affinité de session est essentielle. Les messages en milieu de session doivent atterrir sur le pair qui a répondu à CCR-I. Préférez le routage basé sur `Destination-Host`, ou une règle avancée qui fixe une famille de Session-Id à un pair, plutôt qu'un équilibrage de charge basé sur le domaine. Voir Routage Standard et Routage Avancé.

2. Passez en revue l'`peer_selection_algorithm`. `:random` distribue entre les pairs correspondants et peut briser l'affinité de session pour les applications stateful. `:failover` envoie au premier pair de la liste. Pour les interfaces stateful, routez sur Destination-Host. L'affinité est alors explicite et ne dépend pas de l'équilibrage de charge.

3. Vérifiez si le pair cible a redémarré autour du moment où les réponses `5002` ont commencé. Un pair qui a perdu sa table de sessions rejette les sessions en vol. Le client doit rétablir ces sessions.

4. Si un pair s'est récemment reconnecté, confirmez que `peer_settle_time_ms` est suffisamment grand pour que le DRA route le trafic uniquement après que le pair soit prêt. Voir Flapping de Pair.

5. Suivez le taux de réponses `5002` pour les corréler avec des événements de pair :

```
rate(diameter_peer_message_result_code_count_total{result_code="5
[5m])
```

---

# Référence : Codes de Résultat

# Communs

| Code | Nom | Signification | Référence |
|---|---|---|---|
| 2001 | DIAMETER_SUCCESS | Le DRA a complété la requête avec succès. | RFC 6733 §7.1.1 |
| 3002 | DIAMETER_UNABLE_TO_DELIVER | Le DRA n'a pas pu sélectionner un pair accessible pour livrer la requête. | RFC 6733 §7.1.3 |
| 3003 | DIAMETER_REALM_NOT_SERVED | Aucun pair ne sert le domaine demandé. | RFC 6733 §7.1.3 |
| 3004 | DIAMETER_TOO_BUSY | Le pair sélectionné est surchargé. | RFC 6733 §7.1.3 |
| 5002 | DIAMETER_UNKNOWN_SESSION_ID | Le pair ne reconnaît pas le Session-Id. C'est une session stateful perdue ou mal routée. | RFC 6733 §7.1.5 |

| Code | Nom | Signification | Référence |
| --- | --- | --- | --- |
| 5012 | DIAMETER_UNABLE_TO_COMPLY | Le DRA ne peut pas traiter la requête telle que présentée. | RFC 6733 §7.1.5 |

# Documentation Connexe

- Métriques - Métriques Prometheus et exemples de requêtes référencées ci-dessus.
- Routage Standard - Routage prioritaire RFC 6733, sélection de pairs et routage de réponses.
- Routage Avancé - Routage basé sur des règles, filtres, transformations et ciblage de pairs.
- Multihoming SCTP - Transport SCTP, multihoming et basculement de chemin.