# Guide de l'API REST

← Retour à la documentation principale

Ce guide fournit une documentation complète pour l'**API REST** et l'**interface Swagger** d'OmniSS7.

## Table des matières

1. Aperçu
2. Configuration du serveur HTTP
3. Swagger UI
4. Points de terminaison de l'API
5. Exemples pratiques
6. Authentification
7. Formats de réponse
8. Gestion des erreurs
9. Métriques (Prometheus)
10. Exemples de requêtes

---

## Aperçu

OmniSS7 expose une API REST sur le **port 9090** pour effectuer des opérations MAP (Mobile Application Part) et CAP (CAMEL Application Part) sur le réseau SS7, ainsi que l'injection brute SCCP/M3UA pour les tests. L'API définit **47 points de terminaison d'opération** `POST /api/*` ainsi que les points de terminaison de métriques et Swagger. Elle vous permet de :

- Déclencher des requêtes MAP (mobilité, SMS, authentification, données d'abonné, services supplémentaires, services de localisation, GPRS, équipement)
- Gérer les dialogues de contrôle d'appel CAP/CAMEL

- Injecter des charges utiles SCCP/M3UA brutes
- Surveiller les métriques système via Prometheus

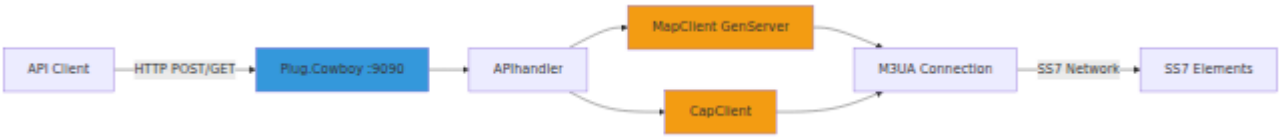
Pour les schémas de requête/réponse par opération et les opcodes MAP, ce guide fait référence au **Guide du client MAP**, qui documente chaque opération en détail.

## Ports et transports

OmniSS7 exécute **trois serveurs HTTP distincts**. Ce guide couvre le premier (:9090). Les autres sont documentés dans la Référence de configuration.

| Port | Transport | Serveur | Portée de ce guide |
|---|---|---|---|
| **9090** | HTTP (`Plug.Cowboy`) | `APIhandler` | **Oui.** MAP/CAP REST `/api/*`, `/metrics`, `/swagger`. |
| 8087 | HTTPS (`Bandit`, Phoenix) | `ControlPanelWeb.Endpoint` | Non : interface utilisateur du panneau de contrôle web. Voir Guide de l'interface Web. |
| 8445 | HTTPS / TLS | `:api_ex` contrôleurs REST | Non : état/routage/liens REST. Voir Référence de configuration. |

## Architecture de l'API




---

# Configuration du serveur HTTP

## Détails du serveur

Les points de terminaison `/api/*`, `/metrics`, et `/swagger` sont servis par un écouteur HTTP `Plug.Cowboy`.

| Paramètre | Valeur | Configurable |
|---|---|---|
| **Protocole** | HTTP | Non |
| **Adresse IP** | 0.0.0.0 (toutes les interfaces) | Via code uniquement |
| **Port** | 9090 | Oui : `config :omniss7, api_port: <port>` (par défaut `9090`) |
| **Transport** | `Plug.Cowboy` | Non |

**URL d'accès** : `http://[server-ip]:9090`

> Le panneau de contrôle (:8087, Bandit) et les contrôleurs REST `:api_ex` (:8445) sont des serveurs séparés. Voir Ports et transports.

## Activation/Désactivation du serveur HTTP

Contrôlez si le serveur HTTP :9090 démarre :

```
config :omniss7,
  start_http_server: true  # Mettre à false pour désactiver
```

**Par défaut** : `true` (activé)

**Lorsqu'il est désactivé** : Le serveur HTTP :9090 ne démarrera pas, et l'API REST / l'interface Swagger / le point de terminaison des métriques seront indisponibles. Le panneau de contrôle (:8087) et les serveurs `:api_ex` (:8445) ne sont pas affectés (ils sont contrôlés par `start_web_interfaces`).

---

# Swagger UI

L'API inclut une **Swagger UI** pour la documentation et les tests interactifs de l'API.

## Accéder à Swagger UI

**URL** : `http://[server-ip]:9090/swagger`

**Fonctionnalités** :

- Documentation interactive de l'API
- Fonctionnalité d'essai pour tester les points de terminaison
- Schémas de requête/réponse
- Charges utiles d'exemple

## Swagger JSON

La spécification OpenAPI est disponible à :

**URL** : `http://[server-ip]:9090/swagger.json`

**Cas d'utilisation** :

- Importer dans Postman ou d'autres clients API

- Générer des bibliothèques clientes
- Automatisation de la documentation de l'API

---

# Points de terminaison de l'API

Chaque point de terminaison d'opération suit le modèle `POST /api/{operation}` et accepte un corps de requête JSON. Les requêtes sont **synchrones** : le gestionnaire bloque jusqu'à ce que la réponse SS7 arrive ou que l'opération expire (délai par défaut **10 secondes** pour les opérations MAP attendues ; voir Gestion des erreurs).

Les tableaux ci-dessous listent tous les **47** points de terminaison d'opération regroupés par catégorie. Pour chaque champ de requête, schéma de réponse et opcode MAP, voir la section correspondante du **Guide du client MAP**, qui a été mis à jour en même temps que ce guide et documente chaque opération en entier.

> **Codes de réponse (tous les points de terminaison)** : `200` = succès, `400` = mauvaise requête/invalide (champs manquants ou de type incorrect), `504` = délai d'attente (pas de réponse SS7 dans le délai de l'opération), `500` = erreur d'opération (non-délai), `503` = fonctionnalité non activée (USSD), `404` = chemin inconnu. Voir Gestion des erreurs.

## Gestion de la mobilité

| Opération | Point de terminaison | Description |
|---|---|---|
| Mettre à jour la localisation | `POST /api/updateLocation` | Enregistrer le VLR de service d'un abonné avec le HLR. En mode HLR, cela déclenche la séquence InsertSubscriberData (ISD). |
| Annuler la localisation | `POST /api/cancelLocation` | Instruire un VLR de supprimer un enregistrement d'abonné. |
| Fournir un numéro d'itinérance | `POST /api/prn` | Obtenir un MSRN du MSC de service. |
| Purger MS | `POST /api/purgeMS` | Marquer un abonné comme purgé au HLR. |
| Envoyer l'identification | `POST /api/sendIdentification` | Récupérer les données IMSI/auth du VLR précédent (TMSI fourni en hexadécimal). |
| Réinitialiser | `POST /api/reset` | Réinitialisation initiée par le HLR vers un VLR (non confirmé : retourne `{"status":"sent"}`). |
| Restaurer les données | `POST /api/restoreData` | Le VLR demande la restauration des données d'abonné auprès du HLR. |

## SMS

| Opération | Point de terminaison | Description |
|---|---|---|
| Envoyer des informations de routage pour SM | `POST /api/sri-for-sm` | Interroger le HLR pour le nœud de service pour la livraison de SMS. |
| MT-Forward SM | `POST /api/MT-forwardSM` | Livrer un SM mobile terminé au MSC/SGSN de service (`smsPDU` en hexadécimal). |
| MO-Forward SM | `POST /api/forwardSM` | Soumettre un SM d'origine mobile (`smsPDU` en hexadécimal). |
| Envoyer SM | `POST /api/sendSM` | Construire et livrer un SMS-DELIVER à partir de texte GSM-7. |
| Construire PDU de livraison | `POST /api/deliverPDU` | Construire un PDU SMS-DELIVER et retourner l'hexadécimal (pas d'envoi SS7). |
| Signaler l'état de livraison de SM | `POST /api/reportSM-DeliveryStatus` | Signaler le résultat de la livraison de SM au HLR. |
| Prêt pour SM | `POST /api/readyForSM` | Notifier le HLR qu'un MS est joignable pour SMS. |
| Alerter le centre de services | `POST /api/alertServiceCentre` | Alerter un SMSC qu'un abonné est disponible. |

## Authentification

| Opération | Point de terminaison | Description |
| --- | --- | --- |
| Envoyer des informations d'authentification | `POST /api/send-auth-info` | Récupérer des vecteurs d'authentification auprès du HLR. |
| Envoyer IMSI | `POST /api/sendIMSI` | Résoudre un MSISDN à son IMSI au HLR. |

## Données d'abonné

| Opération | Point de terminaison | Descript |
|---|---|---|
| Envoyer des informations de routage | `POST /api/sri` | Interroger le HL les informations routage des ap vocaux. |
| Fournir des informations sur l'abonné | `POST /api/provideSubscriberInfo` | Demander l'état/localisatio l'abonné auprès |
| Supprimer les données d'abonné | `POST /api/deleteSubscriberData` | Supprimer les d d'abonné au VL (drapeaux `with` optionnels). |
| Interrogation à tout moment | `POST /api/anyTimeInterrogation` | gsmSCF interro informations/lo de l'abonné au |
| Interrogation d'abonnement à tout moment | `POST /api/anyTimeSubscriptionInterrogation` | Interroger les d d'abonnement |
| Modification à tout moment | `POST /api/anyTimeModification` | Modifier les dor d'abonnement |
| Noter que les données d'abonné ont été modifiées | `POST /api/noteSubscriberDataModified` | Le HLR notifie l gsmSCF des do modifiées. |
| Noter un événement | `POST /api/noteMM-Event` | Signaler un évé de gestion de n |

| Opération | Point de terminaison | Descript |
| --- | --- | --- |
| MM | | au gsmSCF. |

## Services supplémentaires

| Opération | Point de terminaison | Description |
| --- | --- | --- |
| Interroger SS | `POST /api/interrogateSS` | Interroger l'état d'un service supplémentaire. |
| Enregistrer SS | `POST /api/registerSS` | Enregistrer un service supplémentaire (par exemple, transfert d'appel / CFU). |
| Effacer SS | `POST /api/eraseSS` | Effacer un service supplémentaire. |
| Activer SS | `POST /api/activateSS` | Activer un service supplémentaire. |
| Désactiver SS | `POST /api/deactivateSS` | Désactiver un service supplémentaire. |
| Enregistrer le mot de passe | `POST /api/registerPassword` | Enregistrer un mot de passe SS (`ss_code` entier). |
| Obtenir le mot de passe | `POST /api/getPassword` | Récupérer/vérifier un mot de passe SS. |
| Vérification de l'indication SS de transfert | `POST /api/forwardCheckSS-Indication` | Vérification non confirmée vers un VLR (retourne `{"status":"sent"}`). |

Les cinq opérations SS principales (`interrogateSS`, `registerSS`, `eraseSS`, `activateSS`, `deactivateSS`) partagent une forme de requête : `ss_code` (entier) et `hlr_gt`, avec des `forwarded_to`, `no_reply_time` et `basic_service` optionnels.

## Services de localisation (LCS)

| Opération | Point de terminaison | Description |
|---|---|---|
| Envoyer des informations de routage pour LCS | `POST /api/sendRoutingInfoForLCS` | GMLC interroge le HLR pour le routage LCS. |
| Fournir la localisation de l'abonné | `POST /api/provideSubscriberLocation` | Demander la localisation d'un abonné au nœud de service. |
| Rapport de localisation de l'abonné | `POST /api/subscriberLocationReport` | Signaler la localisation d'un abonné au GMLC. |

## GPRS

| Opération | Point de terminaison | Description |
|---|---|---|
| Mettre à jour la localisation GPRS | `POST /api/updateGprsLocation` | Enregistrer le SGSN de service d'un abonné auprès du HLR. |
| Envoyer des informations de routage pour GPRS | `POST /api/sendRoutingInfoForGprs` | Interroger le HLR pour le routage GPRS (sélection GGSN). |

## Équipement

| Opération | Point de terminaison | Description |
|---|---|---|
| Vérifier l'IMEI | `POST /api/checkIMEI` | Interroger l'état de l'équipement au EIR. |

## USSD

| Opération | Point de terminaison | Description |
|---|---|---|
| Envoi USSD (d'origine réseau) | `POST /api/ussd/send` | Envoyer un message USSD à un abonné. Nécessite `ussd_gateway_enabled: true` (sinon `503`). Voir le Guide de la passerelle USSD. |

# CAP / CAMEL

| Opération | Point de terminaison | Description |
|---|---|---|
| CAP InitialDP | `POST /api/cap/initialDP` | Déclencher un dialogue CAMEL vers un gsmSCF. Retourne un `otid` pour les opérations de suivi. |
| CAP Connect | `POST /api/cap/connect` | Acheminer l'appel vers une destination (`destination_number` + `otid`). |
| CAP Continue | `POST /api/cap/continue` | Continuer le traitement de l'appel (`otid`). |
| CAP Release Call | `POST /api/cap/releaseCall` | Libérer l'appel avec une `cause` (`otid`). |
| CAP Apply Charging | `POST /api/cap/applyCharging` | Appliquer une fenêtre de facturation (`duration` + `otid`). |

Les opérations de suivi CAP portent le dialogue `otid` (hexadécimal, retourné par `initialDP`) et répondent avec `{"status":"sent","otid":"<hex>"}`. Voir le Guide de la passerelle CAMEL.

## Injection brute (tests / diagnostics)

| Opération | Point de terminaison | Description |
|---|---|---|
| Raw SCCP | `POST /api/raw/sccp` | Injecter une charge utile SCCP construite à la main (hexadécimal). Surcharges SCCP optionnelles (parti appelé/appelant, SSNs, OPC/DPC). |
| Raw M3UA | `POST /api/raw/m3ua` | Injecter une charge utile M3UA construite à la main (hexadécimal). Surcharges OPC/DPC/contexte de routage optionnelles. |

## Points de terminaison utilitaires

| Point de terminaison | Méthode | But |
|---|---|---|
| `/metrics` | GET | Métriques Prometheus (format texte). |
| `/swagger` | GET | Swagger UI. |
| `/swagger.json` | GET | Spécification OpenAPI. |

---

# Exemples pratiques

Les exemples suivants illustrent la forme de requête/réponse pour des opérations représentatives. Pour la référence complète des champs de chaque opération, voir le Guide du client MAP.

## SendRoutingInfo (SRI)

Récupérer des informations de routage pour établir un appel à un abonné mobile.

**Point de terminaison** : `POST /api/sri`

**Corps de la requête** :

```
{
  "msisdn": "1234567890",
  "gmsc": "5551234567"
}
```

| Champ | Type | Requis | Description |
|---|---|---|---|
| `msisdn` | String | Oui | MSISDN de la partie appelée |
| `gmsc` | String | Oui | Titre global du MSC de passerelle |

**Réponse** (200 OK) :

```
{
  "result": {
    "imsi": "001001234567890",
    "msrn": "5551234999",
    "vlr_number": "5551234800"
  }
}
```

**Exemple cURL** :

```
curl -X POST http://localhost:9090/api/sri \
  -H "Content-Type: application/json" \
  -d '{"msisdn": "1234567890", "gmsc": "5551234567"}'
```

---

## SendRoutingInfoForSM (SRI-for-SM)

Récupérer des informations de routage pour livrer un SMS à un abonné mobile.

**Point de terminaison** : `POST /api/sri-for-sm`

**Corps de la requête** :

```
{
  "msisdn": "1234567890",
  "service_center": "5551234567"
}
```

| Champ | Type | Requis | Description |
|---|---|---|---|
| `msisdn` | String | Oui | MSISDN de destination |
| `service_center` | String | Oui | Titre global du centre de services |

**Exemple cURL** :

```
curl -X POST http://localhost:9090/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{"msisdn": "1234567890", "service_center": "5551234567"}'
```

---

## UpdateLocation

Notifier le HLR d'un changement de localisation d'un abonné (enregistrement VLR). En mode HLR, cela déclenche la séquence InsertSubscriberData (ISD).

**Point de terminaison** : `POST /api/updateLocation`

**Corps de la requête** :

```
{
  "imsi": "001001234567890",
  "vlr": "5551234800"
}
```

| Champ | Type | Requis | Description |
|---|---|---|---|
| `imsi` | String | Oui | IMSI de l'abonné |
| `vlr` | String | Oui | Adresse du titre global VLR |

---

## USSD Send (d'origine réseau)

Envoyer un message USSD à un abonné. Nécessite `ussd_gateway_enabled: true`. Pour le protocole de rappel complet et le cycle de vie de la session, voir le Guide de la passerelle USSD.

**Point de terminaison** : `POST /api/ussd/send`

**Corps de la requête** :

```
{
  "msisdn": "+254712345678",
  "text": "Vous avez une facture en attente. Répondez 1 pour
payer.",
  "callback_url": "http://billing-app:9000/ussd"
}
```

| Champ | Type | Requis | Description |
|---|---|---|---|
| `msisdn` | String | Oui | MSISDN de l'abonné de destination |
| `text` | String | Oui | Texte USSD à afficher (GSM 7 bits) |
| `callback_url` | String | Oui | URL pour recevoir les réponses de l'abonné via HTTP POST |

**Réponse** (200 OK) :

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "status": "sent"
}
```

**Erreur** (503, passerelle désactivée) :

```
{
  "error": "Passerelle USSD non activée"
}
```

---

# Authentification

**État actuel** : L'API ne nécessite **pas d'authentification**.

**Considérations de sécurité** :

- L'API est destinée à un usage sur un réseau interne/de confiance
- Envisagez d'utiliser des règles de pare-feu pour restreindre l'accès
- Pour les déploiements en production, envisagez de mettre en œuvre un middleware d'authentification

---

# Formats de réponse

Toutes les réponses utilisent le format **JSON**.

## Réponse de succès

**Statut HTTP** : 200 OK

**Structure** :

```
{
  "result": {
    // Données de réponse spécifiques à l'opération
  }
}
```

## Réponse d'erreur

**Statut HTTP** :

- 400 Bad Request - Champs de requête invalides ou manquants
- 500 Internal Server Error - Échec de l'opération (erreur non-délai retournée par la pile SS7)
- 503 Service Unavailable - Fonctionnalité non activée (passerelle USSD)
- 504 Gateway Timeout - L'opération a dépassé son délai d'attente (10 secondes par défaut)
- 404 Not Found - Point de terminaison invalide

**Structure** :

```
{
  "error": "timeout"
}
```

ou

```
{
  "error": "invalid request"
}
```

# Gestion des erreurs

## Erreurs courantes

| Erreur | Code HTTP | Description | Solution |
|---|---|---|---|
| JSON invalide | 400 | Le corps de la requête n'est pas un JSON valide | Vérifiez la syntaxe JSON |
| Champs manquants/invalide | 400 | Champs requis manquants ou de type incorrect (les réponses incluent souvent une liste `"required"`) | Inclure tous les paramètres requis avec les types corrects |
| Erreur d'opération | 500 | La pile SS7 a retourné une erreur non-délai | Inspectez le message `"error"` ; vérifiez les journaux d'événements SS7 |
| Fonctionnalité désactivée | 503 | Passerelle USSD non activée | Définir `ussd_gateway_enabled: true` |
| Délai d'attente | 504 | L'opération a dépassé son délai d'attente (10s par défaut) | Vérifiez l'état de connectivité M3UA, la disponibilité HLR/VLR |

| Erreur | Code HTTP | Description | Solution |
|---|---|---|---|
| Non trouvé | 404 | Point de terminaison invalide | Vérifiez l'URL du point de terminaison |

## Comportement de délai d'attente

Les opérations attendues utilisent un **délai d'attente de 10 secondes** par défaut :

1. Requête envoyée au MapClient GenServer
2. Attend une réponse jusqu'à 10 secondes
3. Si aucune réponse → retourne 504 Gateway Timeout
4. Si réponse reçue → retourne 200 OK avec le résultat

**Dépannage des délais d'attente** :

- Vérifiez l'état de la connexion M3UA (Web UI → page M3UA)
- Vérifiez que l'élément réseau (HLR/VLR/MSC) est joignable
- Vérifiez la configuration de routage
- Consultez les journaux d'événements SS7 pour les erreurs

---

# Métriques (Prometheus)

L'API expose des métriques Prometheus pour la surveillance.

## Point de terminaison des métriques

**URL** : `http://[server-ip]:9090/metrics`

**Format** : format texte Prometheus

**Exemple de sortie** :

```
# HELP map_requests_total Total MAP requests
# TYPE map_requests_total counter
map_requests_total{operation="sri"} 42
map_requests_total{operation="sri_for_sm"} 158
map_requests_total{operation="updateLocation"} 23

# HELP cap_requests_total Total CAP requests
# TYPE cap_requests_total counter
cap_requests_total{operation="initialDP"} 87
cap_requests_total{operation="requestReportBCSMEvent"} 91

# HELP map_request_duration_milliseconds Duration of MAP
request/responses in ms
# TYPE map_request_duration_milliseconds histogram
map_request_duration_milliseconds_bucket{operation="sri",le="10"}
5
map_request_duration_milliseconds_bucket{operation="sri",le="50"}
12
map_request_duration_milliseconds_bucket{operation="sri",le="100"}
35
...

# HELP map_pending_requests Number of pending MAP TID waiters
# TYPE map_pending_requests gauge
map_pending_requests 3

# HELP omniss7_license_status Current license status (1 = valid, 0
= invalid)
# TYPE omniss7_license_status gauge
omniss7_license_status 1
```

## Métriques disponibles

Les métriques sont regroupées par couche de protocole / sous-système. Toutes sont déclarées au démarrage et émises avec des helpers protégés, donc une métrique non enregistrée ne perturbe jamais un chemin chaud de protocole.

#### Application (MAP / CAP / USSD / licence)

| Métrique | Type | Étiquettes | Des |
|---|---|---|---|
| `map_requests_total` | Compteur | `operation` | Nomb requê par ty d'opé |
| `map_request_errors_total` | Compteur | `operation` | Nomb d'erre requê par ty d'opé |
| `cap_requests_total` | Compteur | `operation` | Nomb requê par ty d'opé |
| `cap_request_errors_total` | Compteur | `operation` | Nomb d'erre requê par ty d'opé |
| `map_request_duration_milliseconds` | Histogramme | `operation` | Durée requê millise |
| `map_pending_requests` | Jauge | - | Nomb transa MAP e |
| `ussd_requests_total` | Compteur | `direction` | Nomb requê (entra |

| Métrique | Type | Étiquettes | Des |
|---|---|---|---|
| `ussd_active_sessions` | Jauge | - | Nomb sessio active |
| `omniss7_license_status` | Jauge | - | État d licenc (1 = v invali |

**SCCP**

| Métrique | Type | Étiquettes | |
|---|---|---|---|
| `sccp_messages_total` | Compteur | `message_type`, `direction` | Message (`udt`/`xu` et direct |
| `sccp_decode_errors_total` | Compteur | - | Message décoder |
| `sccp_segments_received_total` | Compteur | - | Segmen réassem |
| `sccp_reassembly_completed_total` | Compteur | - | Message réassem |
| `sccp_reassembly_timeouts_total` | Compteur | - | Tampons le minut |

**TCAP**

| Métrique | Type | Étiquettes | |
|---|---|---|---|
| `tcap_dialogues_total` | Compteur | `type` | Dialogues TCAP par |
| `tcap_aborts_total` | Compteur | `reason` | Messages d'abando `unrecognized_mess` (ou `p_abort_other`) |
| `tcap_decode_errors_total` | Compteur | - | Messages TCAP d'u |
| `tcap_active_dialogues` | Jauge | - | Transactions MAP a |

**SCTP**

| Métrique | Type | Étiquettes | |
|---|---|---|---|
| `sctp_association_state` | Jauge | `remote_ip`, `remote_port` | État |
| `sctp_association_state_changes_total` | Compteur | `event` | Tran (`com` |
| `sctp_send_errors_total` | Compteur | `reason` | Envo M2P |
| `sctp_heartbeat_sent_total` | Compteur | `type` | Batt (`bea` |
| `sctp_heartbeat_received_total` | Compteur | `type` | Batt (`bea` |

**SMS / SMSc**

| Métrique | Type | Étiquettes | Description |
| --- | --- | --- | --- |
| `sms_messages_total` | Compteur | `direction`, `operation` | SMS par directio (`mo`/`mt`) et opération (`submit`/`delive` |
| `sms_delivery_failures_total` | Compteur | `reason` | Échecs de livraison de SMS par raison |
| `sms_tracked_subscribers` | Jauge | - | Abonnés actuellement suivis par le SubscriberTracke |

**M2PA (MTP2 Peer-to-Peer Adaptation)**

| Métrique | Type | Étiquettes | Description |
|---|---|---|---|
| `m2pa_messages_sent_total` | Compteur | `link_name`, `point_code`, `adjacent_pc` | Messages MTP3 envoyés sur M2PA |
| `m2pa_messages_received_total` | Compteur | `link_name`, `point_code`, `adjacent_pc` | Messages MTP3 reçus sur M2PA |
| `m2pa_bytes_sent_total` | Compteur | `link_name`, `point_code`, `adjacent_pc` | Octets envoyés sur M2PA |
| `m2pa_bytes_received_total` | Compteur | `link_name`, `point_code`, `adjacent_pc` | Octets reçus sur M2PA |
| `m2pa_link_state_changes_total` | Compteur | `link_name`, `from_state`, `to_state` | Transitions d'état de lien M2PA |
| `m2pa_errors_total` | Compteur | `link_name`, `error_type` | Erreurs M2PA par type |

**M3UA STP**

| Métrique | Type | Étiquettes | Des |
|---|---|---|---|
| `m3ua_stp_messages_received_total` | Compteur | `peer_name`, `point_code` | Messages par pair |
| `m3ua_stp_messages_sent_total` | Compteur | `peer_name`, `point_code` | Messages envoyés p |
| `m3ua_stp_routing_failures_total` | Compteur | `reason` | Échecs de (`no_route` |

## Configuration Prometheus

Ajoutez à votre `prometheus.yml` :

```yaml
scrape_configs:
  - job_name: 'omniss7'
    static_configs:
      - targets: ['server-ip:9090']
    metrics_path: '/metrics'
    scrape_interval: 15s
```

---

# Exemples de requêtes

## Exemple Python

```python
import requests
import json

# Requête SRI-for-SM
url = "http://localhost:9090/api/sri-for-sm"
payload = {
    "msisdn": "1234567890",
    "service_center": "5551234567"
}

response = requests.post(url, json=payload, timeout=15)

if response.status_code == 200:
    result = response.json()
    print(f"Succès : {result}")
elif response.status_code == 504:
    print("Délai d'attente - pas de réponse du réseau")
else:
    print(f"Erreur : {response.status_code} - {response.text}")
```

## Exemple JavaScript

```
const axios = require('axios');

async function sendSRI() {
  try {
    const response = await
axios.post('http://localhost:9090/api/sri', {
      msisdn: '1234567890',
      gmsc: '5551234567'
    }, {
      timeout: 15000
    });

    console.log('Succès :', response.data);
  } catch (error) {
    if (error.code === 'ECONNABORTED') {
      console.error('Délai d'attente - pas de réponse du réseau');
    } else {
      console.error('Erreur :', error.response?.data ||
error.message);
    }
  }
}

sendSRI();
```

## Exemple Bash/cURL

```
#!/bin/bash

# Requête UpdateLocation
response=$(curl -s -w "\n%{http_code}" -X POST
http://localhost:9090/api/updateLocation \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vlr": "5551234800"
  }')

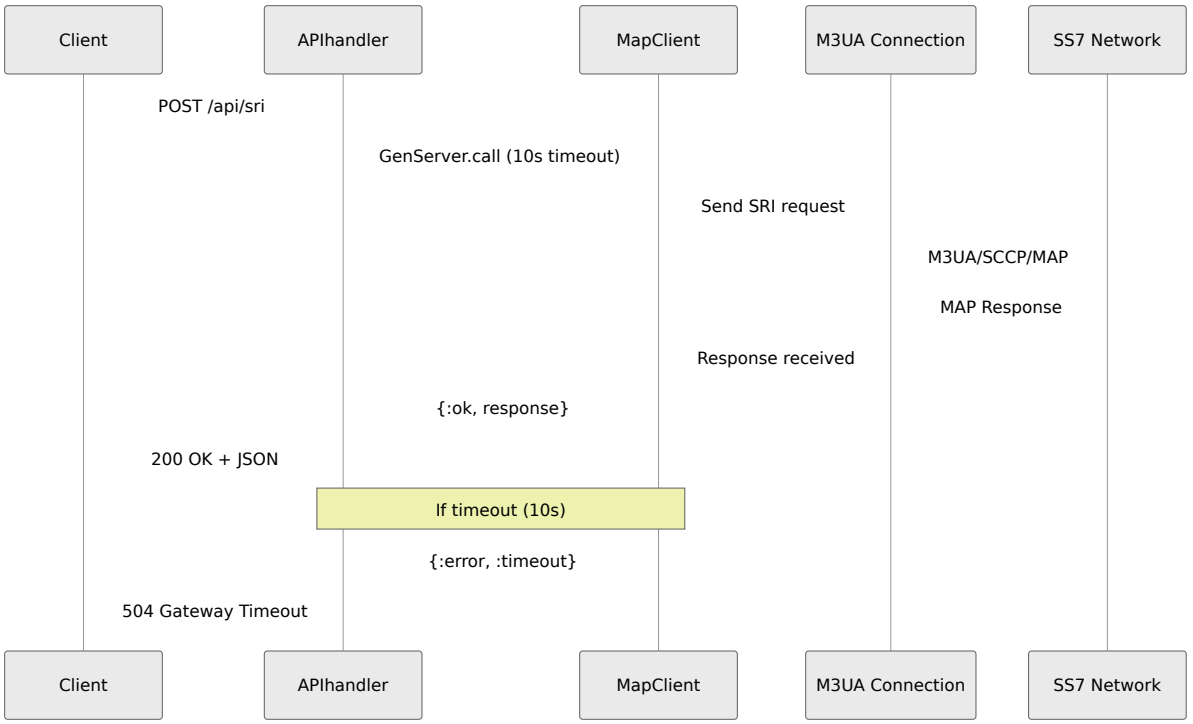
http_code=$(echo "$response" | tail -n 1)
body=$(echo "$response" | sed '$d')

if [ "$http_code" -eq 200 ]; then
  echo "Succès : $body"
elif [ "$http_code" -eq 504 ]; then
  echo "Délai d'attente - pas de réponse du réseau"
else
  echo "Erreur $http_code : $body"
fi
```

---

# Diagrammes de flux

## Flux de requête API




---

# Résumé

L'API REST d'OmniSS7 fournit :

- **47 points de terminaison d'opération** dans les domaines de la mobilité, des SMS, de l'authentification, des données d'abonné, des services supplémentaires, des services de localisation, du GPRS, de l'équipement, de l'USSD, du CAP/CAMEL et de l'injection brute.
- **Swagger UI** à `/swagger` pour la documentation et les tests interactifs.
- **Métriques Prometheus** à `/metrics` pour la surveillance et l'observabilité.
- **Délai d'attente par défaut de 10 secondes** pour les opérations attendues (`504` en cas de délai d'attente).
- Un serveur HTTP **Plug.Cowboy sur le port 9090**, activé par `start_http_server`. Ce serveur est séparé du panneau de contrôle (:8087) et des contrôleurs REST `:api_ex` (:8445).

Pour les schémas de requête/réponse par opération et les opcodes MAP, voir le Guide du client MAP.

Pour l'accès à l'interface Web, voir le Guide de l'interface Web.

Pour les détails de configuration, voir la Référence de configuration.

# Référence Technique (Annexe)

← Retour à la Documentation Principale

Référence technique pour les protocoles SS7 et l'implémentation d'OmniSS7.

## Pile de Protocoles SS7




---

## Codes d'Opération MAP

Les opcodes ci-dessous sont les codes d'opération `local` MAP selon 3GPP TS 29.002. Ce tableau couvre les opérations que génère et/ou gère OmniSS7 ; pour l'ensemble complet des opérations déclenchables par le client et leurs paramètres de demande, voir le **Guide Client MAP**.

| Opération | Opcode | But |
|---|---|---|
| updateLocation | 2 | Enregistrer la localisation de l'abonné (CS) |
| cancelLocation | 3 | Désenregistrer l'abonné du VLR/SGSN |
| provideRoamingNumber | 4 | Demander MSRN |
| noteSubscriberDataModified | 5 | Informer gsmSCF du changement de données de l'abonné |
| insertSubscriberData | 7 | Pousser le profil de l'abonné vers le VLR/SGSN |
| deleteSubscriberData | 8 | Retirer les données de l'abonné du VLR/SGSN |
| registerSS | 10 | Enregistrer un service supplémentaire (par ex. CFU) |
| eraseSS | 11 | Effacer un service supplémentaire |
| activateSS | 12 | Activer un service supplémentaire |
| deactivateSS | 13 | Désactiver un service supplémentaire |
| interrogateSS | 14 | Interroger l'état du service supplémentaire |

| Opération | Opcode | But |
|---|---|---|
| registerPassword | 17 | Enregistrer un mot de passe SS |
| getPassword | 18 | Demander un mot de passe SS à l'abonné |
| sendRoutingInfo | 22 | Interroger le routage des appels (SRI) |
| updateGprsLocation | 23 | Enregistrer la localisation par commutation de paquets (SGSN) |
| sendRoutingInfoForGprs | 24 | Interroger le routage GPRS vers le GGSN |
| reset | 37 | Réinitialisation HLR vers VLR/SGSN |
| forwardCheckSS-Indication | 38 | Indiquer que les données SS doivent être vérifiées à nouveau |
| checkIMEI | 43 | Interroger l'EIR pour l'état de l'équipement |
| mt-forwardSM | 44 | Livrer un SMS à l'abonné (MT) |
| sendRoutingInfoForSM | 45 | Interroger le routage SMS (SRI-for-SM) |
| mo-forwardSM | 46 | Transférer un SMS de l'abonné (MO) |

| Opération | Opcode | But |
|---|---|---|
| reportSM-DeliveryStatus | 47 | Rapporter le résultat de la livraison du SMS au HLR |
| sendIdentification | 55 | Récupérer IMSI/auth du VLR précédent |
| sendAuthenticationInfo | 56 | Demander des vecteurs d'authentification |
| restoreData | 57 | Restaurer les données de l'abonné après une défaillance du VLR |
| sendIMSI | 58 | Résoudre MSISDN en IMSI au HLR |
| processUnstructuredSS-Request | 59 | Demande USSD d'origine mobile (entrant) |
| unstructuredSS-Request | 60 | Demande USSD d'origine réseau (sortant) |
| unstructuredSS-Notify | 61 | Notification USSD d'origine réseau |
| anyTimeSubscriptionInterrogation | 62 | Interrogation gsmSCF des données d'abonnement |
| alertServiceCentre | 64 | Alerter le SMSc qu'un abonné est joignable |
| anyTimeModification | 65 | Modification gsmSCF des données de l'abonné |

| Opération | Opcode | But |
|---|---|---|
| readyForSM | 66 | Notifier le HLR que l'abonné est prêt pour le SMS |
| purgeMS | 67 | Purger l'enregistrement de l'abonné du VLR/SGSN |
| provideSubscriberInfo | 70 | Demander l'état/localisation de l'abonné |
| anyTimeInterrogation | 71 | Interrogation gsmSCF de l'état/localisation de l'abonné |
| provideSubscriberLocation | 83 | Demander une estimation de localisation du nœud de service (LCS) |
| sendRoutingInfoForLCS | 85 | Interrogation GMLC pour le nœud de service de l'abonné (LCS) |
| subscriberLocationReport | 86 | Le nœud de service rapporte une localisation au GMLC (LCS) |
| noteMM-Event | 89 | Rapporter un événement de gestion de mobilité au gsmSCF |

---

# Types de Messages TCAP

- **BEGIN** - Démarrer la transaction
- **CONTINUE** - Milieu de transaction

- **END** - Réponse finale
- **ABORT** - Annuler la transaction

---

# Adressage SCCP

## Formats de Titre Global

- **E.164** - Numéro de téléphone international (par ex., 447712345678)
- **E.212** - Format IMSI (par ex., 234509876543210)
- **E.214** - Format de code de point

## Numéros de Sous-système (SSN)

- **SSN 6**: HLR
- **SSN 7**: VLR
- **SSN 8**: MSC/SMSC
- **SSN 9**: GMLC
- **SSN 10**: SGSN

---

# SMS TPDU

## Types de Messages

- **SMS-DELIVER** (MT) - Réseau vers mobile
- **SMS-SUBMIT** (MO) - Mobile vers réseau
- **SMS-STATUS-REPORT** - État de livraison
- **SMS-COMMAND** - Commande à distance

## Encodages de Caractères

- **GSM7** - Alphabet GSM 7 bits (160 caractères par SMS)

- **UCS2** - Unicode 16 bits (70 caractères par SMS)
- **8-bit** - Données binaires (140 octets par SMS)

---

# États M3UA

- **DOWN** - Pas de connexion SCTP
- **CONNECTING** - Connexion SCTP en cours
- **ASPUP_SENT** - En attente de l'ACK ASPUP
- **INACTIVE** - ASP en ligne mais pas actif
- **ASPAC_SENT** - En attente de l'ACK ASPAC
- **ACTIVE** - Prêt pour le trafic

---

# Codes de Point SS7 Communs

Les codes de point sont généralement des valeurs de 14 bits (UIT) ou de 24 bits (ANSI).

**Format d'Exemple (UIT):**

- Réseau: 3 bits
- Cluster: 8 bits
- Membre: 3 bits

---

# Codes d'Erreur SCCP

- **0** - Pas de traduction pour l'adresse
- **1** - Pas de traduction pour une adresse spécifique
- **2** - Congestion du sous-système
- **3** - Échec du sous-système
- **4** - Utilisateur non équipé

- **5** - Échec MTP
- **6** - Congestion du réseau
- **7** - Non qualifié
- **8** - Erreur dans le transport de message

---

# Codes d'Erreur MAP

| Code | Erreur | Description |
|---|---|---|
| 1 | unknownSubscriber | Abonné non présent dans le HLR |
| 27 | absentSubscriber | Abonné non joignable |
| 34 | systemFailure | Échec du réseau |
| 35 | dataMissing | Données requises non disponibles |
| 36 | unexpectedDataValue | Valeur de paramètre invalide |

---

# Documentation Connexe


- ← Retour à la Documentation Principale
- Guide STP
- Guide Client MAP
- Guide Centre SMS
- Guide HLR
- Fonctionnalités Communes


---

# Guide de Configuration du CAMEL Gateway

## Aperçu

Le mode **CAMEL Gateway (CAMELGW)** transforme OmniSS7 en une plateforme de Réseau Intelligent (IN) qui fournit un contrôle d'appel en temps réel et une facturation en utilisant le protocole CAMEL Application Part (CAP). OmniSS7 agit en tant que **gsmSCF** (Service Control Function), recevant des déclencheurs du **gsmSSF** dans le MSC/VLR/GMSC et prenant des décisions de facturation via CGrateS.

### Qu'est-ce que CAMEL ?

**CAMEL** (Applications Personnalisées pour la Logique Améliorée des Réseaux Mobiles) est un ensemble de normes qui fonctionnent sur un réseau de cœur

GSM ou un réseau UMTS. Il permet aux opérateurs de fournir des services nécessitant un contrôle en temps réel des appels, tels que :

- **Appels prépayés** - Vérification et facturation du solde en temps réel
- **Services à tarif premium** - Facturation spéciale pour les services à valeur ajoutée
- **Contrôle de routage des appels** - Routage dynamique des destinations basé sur le temps/l'emplacement
- **Réseaux privés virtuels** - Plans de numérotation d'entreprise
- **Filtrage des appels** - Autoriser/bloquer les appels en fonction de critères

## Versions du Protocole CAP

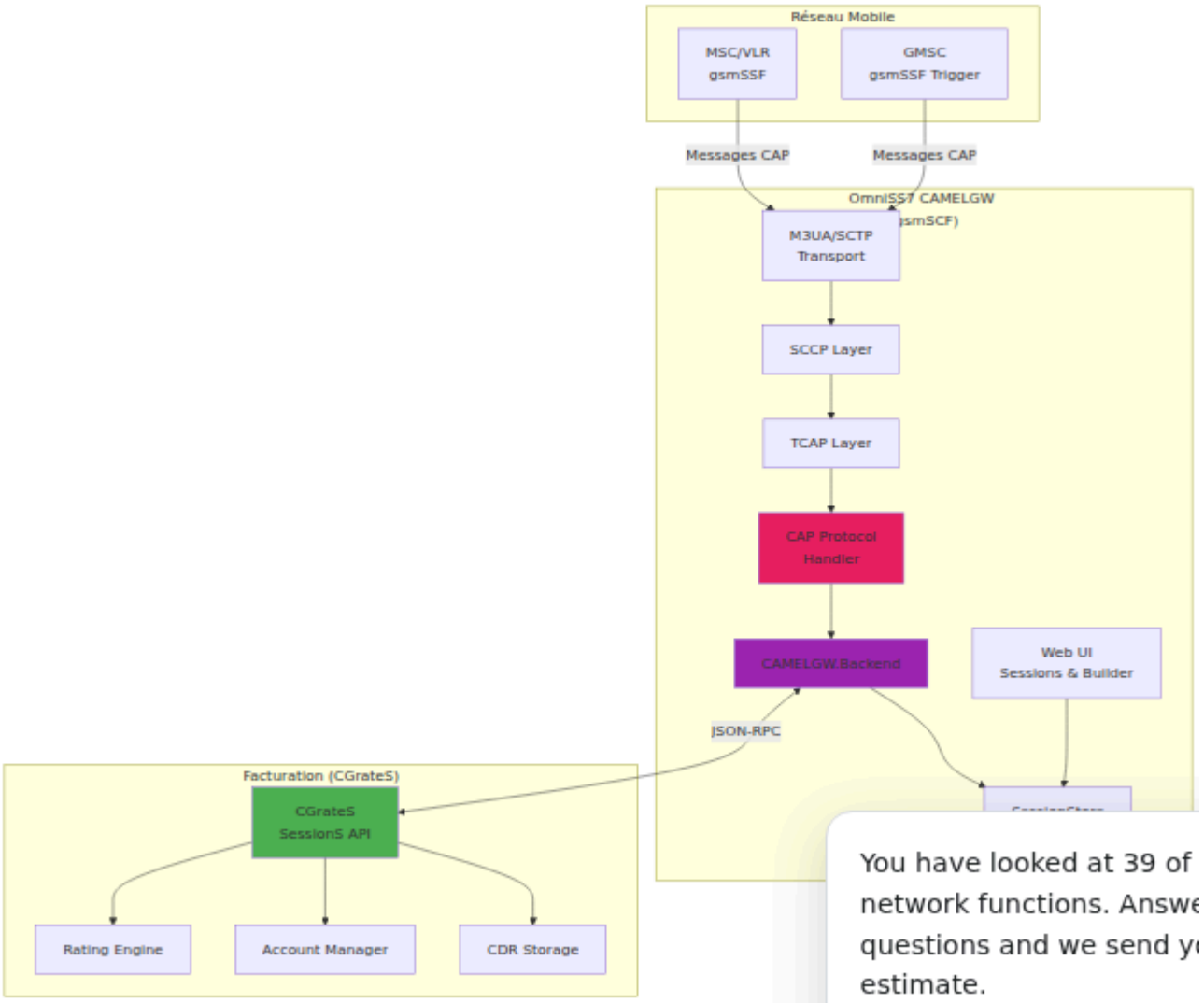
OmniSS7 CAMELGW prend en charge plusieurs versions de CAP. La version contrôle l'OID de contexte d'application transporté dans le dialogue TCAP :

| Version | Phase | OID de Contexte d'Application |
|---|---|---|
| **CAP v1** | Phase CAMEL 1 | `0.4.0.0.1.0.50.0` |
| **CAP v2** | Phase CAMEL 2 | `0.4.0.0.1.0.50.1` |
| **CAP v3** | Phase CAMEL 3 | `0.4.0.0.1.21.3.4` |
| **CAP v4** | Phase CAMEL 4 | `0.4.0.0.1.23.3.4` |

**Par défaut :** CAP v2 (le plus largement déployé). La version entrante est détectée automatiquement à partir de la portion de dialogue InitialDP et réutilisée lors de la génération de la réponse, de sorte que les réponses correspondent à la version CAP de chaque MSC.

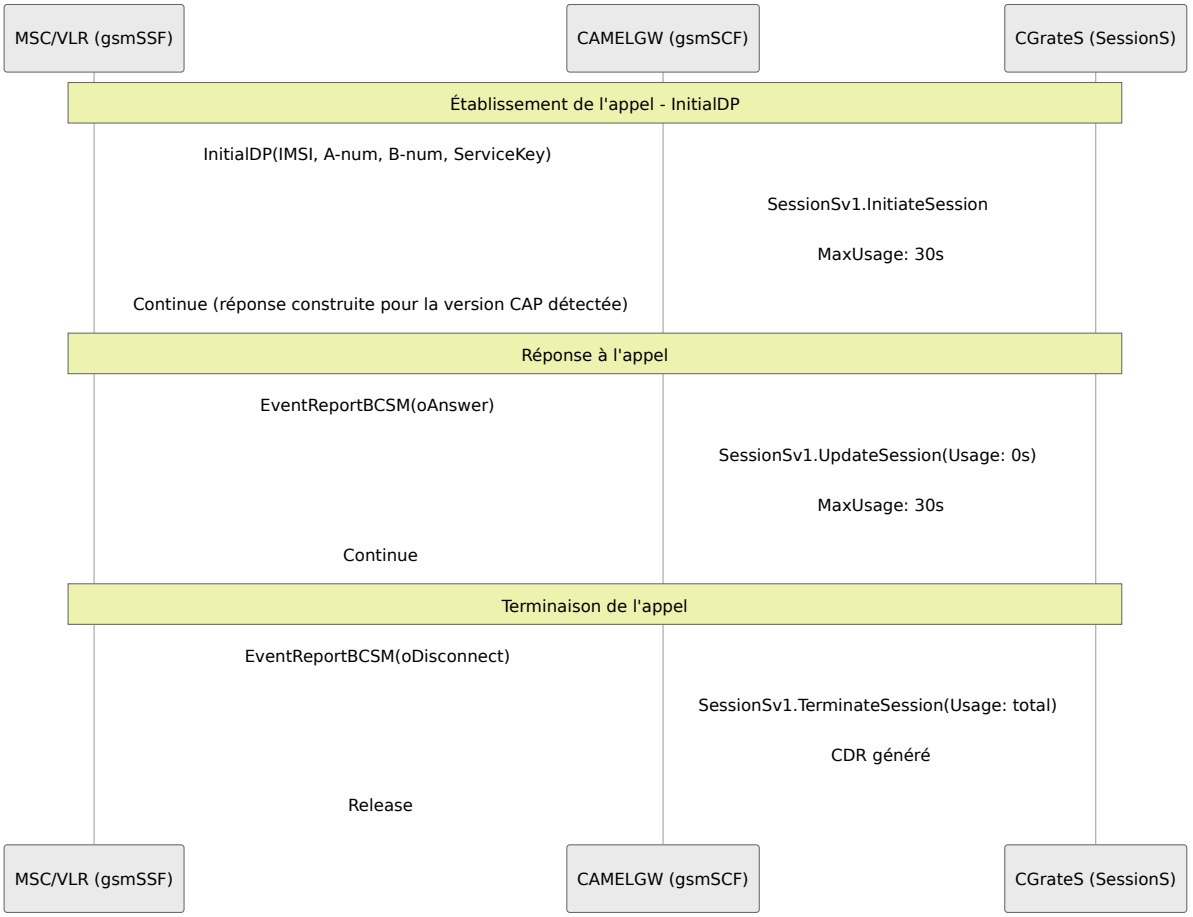
---

# Architecture




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

## Exemple de Flux d'Appel




---

# Configuration

## Prérequis

- OmniSS7 installé et en cours d'exécution
- Connectivité M3UA au MSC/GMSC (gsmSSF)
- Une instance de CGrateS accessible via JSON-RPC (facultatif, pour la facturation en temps réel)

## Activer le Mode CAMEL Gateway

Le traitement CAMEL est contrôlé par deux indicateurs indépendants. `cap_client_enabled` active le client CAP/M3UA et les composants de session

CAMEL ; `cgrates_enabled` active l'intégration de facturation CGrateS derrière le backend. Configurez-les dans `config/runtime.exs` :

```
config :omniss7,
  # Activer le client CAP/CAMEL (M3UA ASP, SessionStore, gestion
des requêtes)
  cap_client_enabled: true,

  # Autres modes indépendants (définir selon les besoins de votre
déploiement)
  map_client_enabled: false,
  smsc_mode_enabled: false,

  # Version CAP utilisée pour les requêtes sortantes / dialogue
(:v1 | :v2 | :v3 | :v4)
  cap_version: :v2,

  # Intégration de facturation CGrateS
  cgrates_enabled: true,

  # Connexion M3UA pour CAP (se connecter en tant qu'ASP vers le
MSC/GMSC gsmSSF)
  cap_client_m3ua: %{
    routing_context: 1,
    opc: 5013,
    dpc: 5011
  }
```

> **Remarque sur `cap_client_m3ua` :** le client CAMEL ne lit que `:routing_context`, `:opc` et `:dpc` à partir de cette carte. Toute clé supplémentaire est ignorée par le client CAP. Le nom du processus ASP M3UA est fixé en interne à `:camelgw_client_asp`.

> Il n'existe pas d'indicateur `camelgw_mode_enabled`, `hlr_mode_enabled` ou `smsc_mode_enabled`-for-CAMEL. Les seuls indicateurs qui régulent le comportement CAMEL sont `cap_client_enabled` et `cgrates_enabled`.

**Indicateurs de Mode**

| Clé | Par défaut | Objectif |
|---|---|---|
| `cap_client_enabled` | `false` | Démarre le client CAP/CAMEL (M3UA ASP, SessionStore, gestion des requêtes CAP) |
| `cgrates_enabled` | `false` | Active l'intégration de facturation CGrateS dans `CAMELGW.Backend` |
| `cap_version` | `:v2` | Version CAP par défaut / OID de dialogue pour les requêtes sortantes |
| `map_client_enabled` | `false` | Mode client MAP indépendant |
| `smsc_mode_enabled` | `false` | Mode SMSc indépendant (nécessite un client MAP) |

**Paramètres de Facturation CGrateS**

| Clé | Par défaut | C |
| --- | --- | --- |
| `cgrates_enabled` | `false` | Interrup principa l'intégr |
| `cgrates_url` | `http://localhost:2080/jsonrpc` | Point d JSON-R CGrate |
| `cgrates_tenant` | `cgrates.org` | Locatai chaque Sessior |
| `cgrates_request_type` | `*prepaid` | Type de CGrate exempl `*postp` |
| `cgrates_timeout` | `5000` | Délai d requête millisec |
| `cgrates_voice_usage_seconds` | *Requis* | Taille d vocal e chaque ApplyC réserve quantit l'Apply l'accord même l'interv intermé |

## Configurer les Pages de l'Interface Web

L'interface Web expose les pages CAMEL via le panneau de contrôle. Celles-ci sont configurées dans `config/config.exs` :

```
config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "Pairs"},
    {SS7.Web.CAMELSessionsLive, "/camel_sessions", "Sessions
CAMEL"},
    {SS7.Web.CAMELRequestLive, "/camel_request", "Constructeur de
Requêtes CAMEL"}
  ],
  page_order: [
    "/events", "/m3ua", "/camel_sessions", "/camel_request",
    "/application", "/configuration"
  ]
```

| Étiquette de Page | Route | Module |
|---|---|---|
| Pairs | `/m3ua` | `SS7.Web.M3UAStatusLive` |
| Sessions CAMEL | `/camel_sessions` | `SS7.Web.CAMELSessionsLive` |
| Constructeur de Requêtes CAMEL | `/camel_request` | `SS7.Web.CAMELRequestLive` |

---

# Opérations CAP Supportées

Les opérations CAP sont identifiées par nom et direction. Notez que certains numéros d'opcode CAP sont en interne incohérents (la carte des opcodes de journal d'événements et le générateur de requêtes ne s'accordent pas sur quelques opérations), donc les opérations ci-dessous sont décrites par **nom et point de terminaison**. Lorsqu'un opcode est cité, c'est celui que le générateur de requêtes émet réellement.

# Opérations Entrantes (gsmSSF → gsmSCF)

Dispatchées sur l'opcode de composant TCAP décodé :

| Opération | Remarques |
|---|---|
| **InitialDP** | Notification d'établissement d'appel ; déclenche CGrateS `InitiateSession`. Accepté à l'opcode 0, et aussi à l'opcode 60 (traité comme InitialDP/InitialDPSMS) |
| **AssistRequestInstructions** | Accusé de réception ; dispatché à l'opcode 1 dans un TCAP Begin |
| **EventReportBCSM** | Événements BCSM. oAnswer est enregistré (pas de réponse) ; oDisconnect / oAbandon / tDisconnect / tAbandon / occupé / pas de réponse terminent la session CGrateS et mettent fin au dialogue |
| **ApplyChargingReport** | Opcode 36. Appel actif : rapporte la période à CGrateS et accorde le prochain segment. Appel terminé : termine la session CGrateS avec l'utilisation totale |
| **Continue** | Accusé de réception |
| **ConnectToResource** | Accusé de réception |
| **RequestReportBCSMEvent** | Accusé de réception |
| **ApplyCharging** | Paramètres décodés en BER et accusés de réception |
| **Connect** | Paramètres décodés en BER et accusés de réception |
| **ReleaseCall** | Accusé de réception |

Les opcodes non gérés sont répondus par une erreur TCAP (Facility Not Supported / Unexpected Data Value).

# Opérations Sortantes (gsmSCF → gsmSSF)

Envoyées vers le gsmSSF. Les opcodes ci-dessous sont les valeurs émises par OmniSS7 :

| Opération | Opcode |
|---|---|
| **InitialDP** | 0 |
| **Connect** | 20 |
| **ReleaseCall** | 22 |
| **RequestReportBCSMEvent** | 23 |
| **Continue** | 31 |
| **ApplyCharging** | 35 |
| **InitialDPSMS** | 60 |
| **NoteMM-Event** | 89 (MAP) |

> **InitialDPSMS** (opcode 60) est pour le contrôle CAMEL MO-SMS : envoyé par le smsSSF au gsmSCF lorsqu'un SMS-CSI / mo-sms-CSI est armé, avant que le message court ne soit relayé au SMSC. L'argument est encodé avec le module CAP ASN.1 généré et le dialogue porte le contexte d'application `id-ac-cap3-sms-AC`.
>
> **NoteMM-Event** est une opération MAP (opcode 89, pas CAP) utilisée pour les notifications de mobilité CAMEL M-CSI du VLR au gsmSCF. Elle est encodée avec le module MAP ASN.1 (`:TCAPMessages.encode(:MapSpecificPDUs, …)`) et porte le contexte d'application `mm-EventReporting-v3`.

Tous les messages CAP TCAP sont encodés avec `:GenericTCAP.encode(:GenericPDUs, …)`. Seul le NoteMM-Event basé sur MAP utilise `:TCAPMessages`.

# Fonctionnalités de l'Interface Web

## Page des Sessions CAMEL

**Titre :** Sessions CAMEL · **Route :** `/camel_sessions` (HTTPS, port 8087)

Surveillance en temps réel des sessions d'appel CAMEL actives :

- **Liste des sessions en direct** avec actualisation automatique
- **Détails de la session** - OTID, ID d'appel, État, Durée
- **Version CAP** détectée à partir de l'InitialDP et stockée par session
- **Informations sur l'appel** - IMSI, A-number, B-number, Clé de service
- **Suivi de l'état** - `initié`, `répondu`, `terminé`

**Colonnes du tableau :** ID d'appel, État, Version, IMSI, Numéro appelant, Numéro appelé, Clé de service, Durée, Heure de début, OTID

Les mêmes données de session sont disponibles via l'API REST à `/camel-sessions` (voir Surveillance & Opérations).

## Constructeur de Requêtes CAMEL

**Titre :** Constructeur de Requêtes CAMEL · **Route :** `/camel_request` (HTTPS, port 8087)

Outil interactif pour construire et envoyer des requêtes CAP pour des tests.

**Types de requêtes proposés par le formulaire :** InitialDP, Connect, ReleaseCall, RequestReportBCSMEvent, ApplyCharging, Continue.

**Sélecteur de version CAP :** le formulaire dispose d'un menu déroulant de version CAP par requête (v1/v2/v3/v4 avec leurs OIDs). La sélection remplace temporairement la valeur par défaut configurée pour cette requête et persiste lors des changements de formulaire ; l'historique des requêtes montre la version utilisée dans une colonne **CAP Ver**.

**Options avancées SCCP/M3UA :**

| Champ | Par défaut | Remarques |
|---|---|---|
| GT de la Partie Appelée | `99988415011` | Titre Global de Destination |
| GT de la Partie Appelante | (configurable) | Titre Global d'Origine |
| SSN Appelé | `146` | gsmSSF |
| SSN Appelant | `146` | gsmSSF |
| OPC | `5013` | Code de Point d'Origine |
| DPC | `5011` | Code de Point de Destination |

---

# Intégration avec CGrateS

OmniSS7 facture les appels via l'API JSON-RPC **SessionS** de CGrateS. `CGrateS.SessionsClient` poste à l'URL `cgrates_url` configurée et mappe les opérations CAP aux méthodes SessionS :

| Déclencheur CAP | Méthode SessionS |
|---|---|
| InitialDP | `SessionSv1.InitiateSession` (réserve un segment) |
| ApplyChargingReport, appel actif | `SessionSv1.UpdateSession` (`LastUsed` = la période, réserve le prochain segment) |
| ApplyChargingReport, appel terminé ; EventReportBCSM déconnexion / abandon ; TCAP Fin ou Abandon du MSC | `SessionSv1.TerminateSession` (utilisation totale), puis `SessionSv1.ProcessCDR` |

La facturation vocale suit le modèle TAS Ro : le même locataire (`cgrates_tenant`), `ToR *voice`, `RequestType` `cgrates_request_type`, `Category call`, un segment fixe (`cgrates_voice_usage_seconds`), et `GetAttributes` à chaque requête afin que les profils AttributeS du locataire mappent l'abonné sur le compte de facturation. Le `Account` est l'**IMSI** de l'abonné servi à partir de l'InitialDP (un InitialDP sans IMSI est libéré). `Destination` est le numéro appelé. `OriginID` est `CAMEL-<hex OTID>`, `OriginHost` est `CAMELGW`.

# Cycle de Vie de l'Appel avec Facturation

### 1. Initiation de l'Appel (InitialDP)

Lorsque le MSC envoie InitialDP, CAMELGW :

1. **Détecte la version CAP** à partir de la portion de dialogue
2. **Décode l'InitialDP** (IMSI, numéro appelant/numéro appelé ISUP, CalledPartyBCDNumber, clé de service, type d'événement)
3. **Appelle CGrateS** `SessionSv1.InitiateSession` pour un segment
4. **Répond au MSC** dans un TCAP Continue avec RequestReportBCSMEvent (O-BCSM ou T-BCSM réponse, déconnexion des jambes 1 et 2, abandon ; tous notifyAndContinue), ApplyCharging et Continue

`MaxUsage` dimensionne l'ApplyCharging `maxCallPeriodDuration`. Un segment complet est accordé sans `releaseIfDurationExceeded` ; un reste plus court est accordé avec, de sorte que le MSC libère l'appel lorsque le dernier crédit est utilisé. Pas de crédit, ou une erreur CGrateS, répond avec **ReleaseCall** (cause 31) dans un TCAP Fin.

### 2. Réponse à l'Appel (EventReportBCSM - oAnswer / tAnswer)

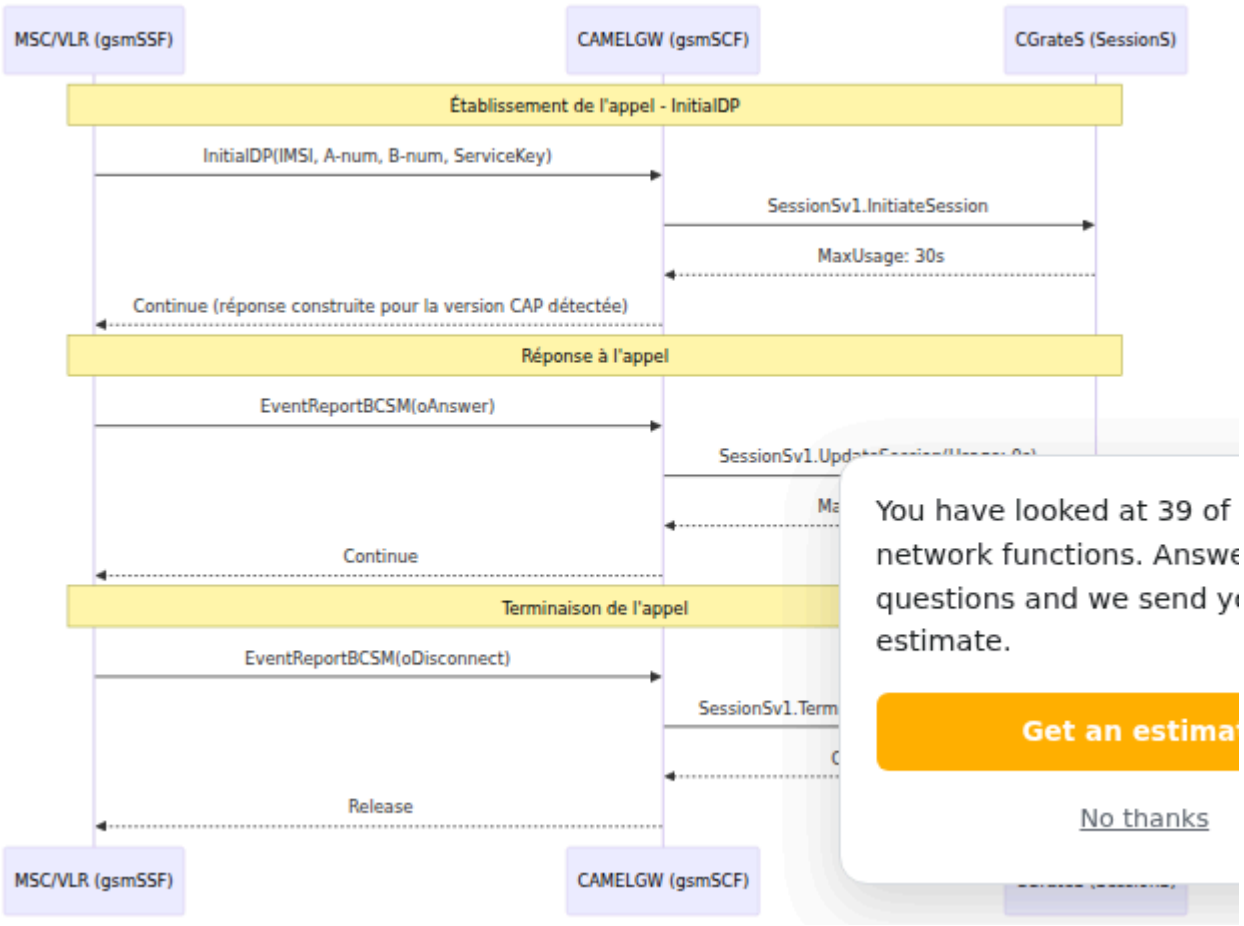
La session est marquée comme répondue. Rien n'est renvoyé ; le dialogue reste ouvert.

### 3. Expiration de la Période (ApplyChargingReport)

Chaque fois que la période accordée se termine, le MSC rapporte le temps utilisé. La passerelle le rapporte à CGrateS et accorde le prochain segment, ou envoie ReleaseCall lorsque le crédit est épuisé.

### 4. Terminaison de l'Appel

Le dernier ApplyChargingReport (appel plus actif), une déconnexion / abandon EventReportBCSM, ou la fermeture du dialogue par le MSC termine la session CGrateS avec l'utilisation totale, écrit le CDR (`ProcessCDR`) et supprime la session.




## Champs CDR de CAMEL

Les CDR sont produits par CGrateS et incluent généralement :

- `Account` - IMSI (ou numéro appelant en secours)
- `Destination` - Numéro de la partie appelée
- `OriginID` - Identifiant de l'appel (`CAMEL-<hex OTID>`)
- `Usage` - Durée totale de l'appel (secondes)
- `IMSI`, `CallingPartyNumber`, `CalledPartyNumber`
- `MSCAddress` - MSC servant
- `ServiceKey` - Clé de service CAMEL

---

# Tests

## Tests Manuels avec le Constructeur de Requêtes

1. Ouvrez `https://your-server:8087/camel_request`
2. Sélectionnez **InitialDP**, définissez la clé de service / numéros appelants / appelés, choisissez éventuellement une version CAP, et envoyez. Notez l'OTID généré.
3. Ouvrez `https://your-server:8087/camel_sessions` pour voir la session dans l'état `initié`.
4. Envoyez **EventReportBCSM** avec `oAnswer` pour déplacer la session à `répondu`.
5. Envoyez **ReleaseCall** avec cause `16` (Normal) pour terminer.

## Tests avec un MSC Réel

Configurez le service CAMEL sur votre MSC/VLR (Clé de service, Titre Global gsmSCF) afin qu'il déclenche InitialDP vers OmniSS7. Exemple pour un MSC Huawei :

```
ADD CAMELSERVICE:
  SERVICEID=1,
  SERVICEKEY=100,
  GSMSCFADDR="55512341234",   # Titre Global CAMELGW
  DEFAULTCALLHANDLING=CONTINUE;

ADD CAMELSUBSCRIBER:
  IMSI="310150123456789",
  SERVICEID=1,
  TRIGGERTYPE=TERMCALL;
```

#### Surveiller les Journaux

OmniSS7 émet des journaux JSON structurés (un objet par ligne via `OmniLogger.JsonFormatter`), vous pouvez donc filtrer sur les champs de métadonnées :

```
# Suivre et filtrer les événements CAP par le champ map_event
tail -f /var/log/omniss7/omniss7.log | jq 'select(.map_event |
tostring | startswith("CAP:"))'
```

---

# Surveillance & Opérations

## Métriques Prometheus

CAMELGW expose des métriques à `http://localhost:9090/metrics`.

**Métriques CAP :**

- `cap_requests_total{operation}` - Total des requêtes CAP par opération. Les valeurs d'étiquette sont en snake_case, par exemple `initial_dp`, `connect`, `release_call`, `continue`, `apply_charging`.
- `cap_request_errors_total{operation}` - Erreurs de génération/envoi de requêtes CAP par opération.

**Métriques MAP / API :**

- `map_requests_total{operation}` - Total des requêtes MAP par opération
- `map_pending_requests` - Nombre de MAP TID en attente

**Métriques STP M3UA** (si le mode STP est activé) :

- `m3ua_stp_routing_failures_total{reason}` - Échecs de routage par raison

```
# Tous les compteurs de requêtes CAP
curl http://localhost:9090/metrics | grep cap_requests_total

# InitialDP uniquement (notez l'étiquette en snake_case)
curl http://localhost:9090/metrics | grep
'cap_requests_total{operation="initial_dp"}'
```

## Vérifications de Santé

```
# Statut des pairs M3UA (API TLS sur le port 8445)
curl -k https://localhost:8445/m3ua-status

# Sessions CAMEL actives (retourne session_count + détail par
session)
curl -k https://localhost:8445/camel-sessions
```

Il n'existe pas d'endpoint `/api/ocs-status`, `/api/events`, ou `/api/camel/sessions/count`. Les comptes de session proviennent de `/camel-sessions`.

## Configuration des Journaux

Ajustez le niveau de journalisation dans `config/runtime.exs` :

```
config :logger, level: :info   # :debug | :info | :warning |
:error
```

Le formatteur est défini sur JSON structuré dans `config/config.exs` :

```
config :logger, :default_formatter,
  format: {OmniLogger.JsonFormatter, :format},
  metadata: :all,
  truncate: :infinity
```

---

# Dépannage

## Problème : Aucun message CAP reçu

**Symptômes :** Le Constructeur de Requêtes fonctionne, mais le MSC n'envoie jamais InitialDP.

**Vérifiez :**

1. Statut du pair M3UA : `curl -k https://localhost:8445/m3ua-status`
2. Configuration du service CAMEL du MSC (Clé de service, adresse gsmSCF)
3. Routage SCCP (le Titre Global gsmSCF doit router vers CAMELGW)
4. Règles de pare-feu (autoriser le port SCTP 2905)

# Problème : Erreurs CGrateS

**Symptômes :** `not_configured`, `http_error`, ou `cgrates_error` dans les journaux.

**Vérifiez :**

1. `cgrates_enabled: true` et `cgrates_url` pointent vers un point de terminaison JSON-RPC accessible
2. Le compte (IMSI ou numéro appelant) existe et a un solde dans CGrateS
3. `cgrates_tenant` et `cgrates_request_type` correspondent à votre configuration de tarification CGrateS
4. `cgrates_timeout` est suffisamment grand pour votre latence CGrateS

En cas d'échec de facturation pendant un appel, CAMELGW continue l'appel plutôt que de le couper ; l'erreur est enregistrée.

# Problème : Session non trouvée

**Symptômes :** EventReportBCSM enregistre "Session non trouvée".

**Cause :** Mismatch d'OTID ou la session a déjà été terminée.

**Vérifiez :** Vérifiez l'OTID dans les journaux et assurez-vous que le DTID/OTID correspondent à travers le dialogue.

# Problème : Erreurs de décodage

**Symptômes :** "Échec du décodage de l'InitialDP" dans les journaux.

**Cause :** Mismatch de version CAP ou message malformé.

**Vérifiez :** Confirmez que la configuration de version CAP correspond à celle du MSC, et capturez un PCAP pour analyse :

```
tcpdump -i eth0 -w cap_trace.pcap sctp port 2905
```

---

# Résumé

Le mode CAMEL Gateway permet à OmniSS7 d'agir en tant que gsmSCF avec :

- **Support du protocole CAP** (v1/v2/v3/v4) avec remplacement de version par requête
- **Facturation en temps réel** via l'API JSON-RPC SessionS de CGrateS
- **Opérations de contrôle d'appel** (Connect, Release, Continue, ApplyCharging)
- **Gestion des sessions** dans ETS, indexée par OTID
- **Tests interactifs** via le Constructeur de Requêtes de l'Interface Web
- **Surveillance en direct** des sessions actives (Interface Web et API `/camel-sessions`)
- **Génération de CDR** par CGrateS pour la facturation et l'analyse

Pour plus de détails :

- Référence Technique - Opérations CAP

---

**Produit :** OmniSS7 CAMEL Gateway
**Version de la Documentation :** 1.1

# Guide des Fonctionnalités Communes

← Retour à la Documentation Principale

Ce guide couvre les fonctionnalités communes à tous les modes de fonctionnement d'OmniSS7.

## Table des Matières

1. Aperçu de l'Interface Web
2. Documentation de l'API
3. Surveillance et Métriques
4. Meilleures Pratiques
5. Multihoming SCTP pour la Redondance Réseau

---

## Aperçu de l'Interface Web

L'interface Web est servie sur HTTPS sur le port `8087` (TLS est imposé via `force_ssl`), par exemple `https://localhost:8087/`.

## Navigation Principale

Le menu est construit à partir des pages activées dans `config/config.exs` et ordonné par `page_order`. Entrées communes (étiquette de menu en gras) :

- **Ressources** (`/application`) - État de l'application et informations d'exécution
- **Configuration** (`/configuration`) - Visualiseur de configuration système
- **Événements SS7** (`/events`) - Événements de signalisation SS7 en temps réel et journaux de messages
- **Journaux Système** (`/logs`) - Visualiseur de journaux d'application
- **Client SS7** (`/client`) - Client de test de requête MAP interactif
- **Pairs** (`/m3ua`) - Connexions de pairs M3UA/M2PA
- **Routage** (`/routing`) - Codes de point et routes de titre global
- **Test de Routage** (`/routing_test`) - Outil de test de résolution de route
- **Liens SMSc** (`/smsc_links`) - Liens de backend SMS (SMSc)
- **Abonnés SMSc** (`/smsc_subscribers`) - Abonnés SMS suivis (SMSc)
- **Liens HLR** (`/hlr_links`) - Liens de backend HLR (HLR)

- **Abonnés Actifs** (`/subscribers`) - Abonnés UpdateLocation suivis
- **Sessions CAMEL** (`/camel_sessions`) - Sessions d'appel CAMEL en direct (CAMEL GW)
- **Constructeur de Requêtes CAMEL** (`/camel_request`) - Constructeur de requêtes CAP interactif (CAMEL GW)
- **Test MSISDN/IMSI** (`/msisdn_imsi_test`) - Outil de test de traduction de numéro

# Accéder à l'Interface Web

1. Ouvrez votre navigateur web
2. Naviguez vers l'hôte configuré sur le port 8087 (par exemple, `https://localhost:8087/`)
3. Consultez le tableau de bord de l'état du système

# Documentation de l'API Swagger

La documentation interactive de l'API est servie par l'API d'opération MAP sur le port HTTP 9090 :

```
http://your-server:9090/swagger
```

# Configuration de l'Interface Web

Le point de terminaison de l'interface web est configuré sous `config :control_panel, ControlPanelWeb.Endpoint` dans `config/config.exs`, et le menu de navigation sous `config :control_panel` :

```
config :control_panel, ControlPanelWeb.Endpoint,
  force_ssl: [rewrite_on: [:x_forwarded_proto]],
  server: true,
  url: [host: "0.0.0.0", path: "/"],
  https: [
    port: 8087,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ]

config :control_panel,
  page_order: [
    "/application", "/configuration", "/events", "/logs",
"/client",
    "/m3ua", "/routing", "/routing_test", "/smsc_links",
"/smsc_subscribers",
    "/hlr_links", "/subscribers", "/camel_sessions",
"/camel_request",
    "/msisdn_imsi_test"
  ]
```

**Paramètres de Configuration :**

| Paramètre | Type | Par Défau |
| --- | --- | --- |
| `ControlPanelWeb.Endpoint.url.host` | String | `"0.0.0.0"` |
| `ControlPanelWeb.Endpoint.https.port` | Integer | `8087` |
| `ControlPanelWeb.Endpoint.https.keyfile` | String | `"priv/cert/omnit` |
| `ControlPanelWeb.Endpoint.https.certfile` | String | `"priv/cert/omnit` |
| `ControlPanelWeb.Endpoint.force_ssl` | Keyword | `[rewrite_on: [:x_forwarded_pro` |
| `page_order` | Liste de Strings | voir exemple ci-des |

> Voir la Référence de Configuration pour le schéma complet du Panneau de Contrôle.

# Configuration du Logger

Les journaux sont émis au format JSON structuré (un objet par ligne) par `OmniLogger.JsonFormatter`, qui inclut toutes les `Logger.metadata` afin que le contexte par processus (OPC/DPC/GTs de routage STP, passerelle SMS vers/depuis MSISDNs, transaction otid/opération) soit automatiquement pris en charge. Ce format est configuré dans `config/config.exs` :

```
config :logger, :default_formatter,
  format: {OmniLogger.JsonFormatter, :format},
  metadata: :all,
  truncate: :infinity
```

Le niveau de journalisation est défini séparément :

```
config :logger,
  level: :debug  # Options :debug, :info, :warning, :error
```

**Paramètres de Configuration :**

| Paramètre | Type | Par Défau |
| --- | --- | --- |
| `:logger.level` | Atom | `:debug` |
| `:logger.default_formatter.format` | Tuple | `{OmniLogger.JsonF` `:format}` |
| `:logger.default_formatter.metadata` | Atom/Liste | `:all` |
| `:logger.default_formatter.truncate` | Integer/Atom | `:infinity` |

---

# Documentation de l'API

OmniSS7 expose deux surfaces HTTP :

| Surface | Port | Protocole | Contenus |
| --- | --- | --- | --- |
| API d'opération MAP | `9090` | HTTP | Opérations MAP `/api/*`, `/swagger`, `/swagger.json`, `/metrics` |
| API d'état JSON (`api_ex`) | `8445` | HTTPS | État en lecture seule (`/api/status`, `/api/m3ua-status`, `/api/routing`, `/api/subscribers`, `/api/camel-sessions`, ...) |

## URL de Base de l'API MAP

```
http://your-server:9090/api
```

## Codes de Réponse

- **200** - Succès
- **400** - Mauvaise Requête
- **504** - Délai d'Attente de Passerelle

## Spécification OpenAPI

```
http://your-server:9090/swagger.json
```

---

# Surveillance et Métriques

## Point de Terminaison des Métriques Prometheus

```
http://your-server:9090/metrics
```

## Catégories de Métriques Clés

**Métriques M3UA/SCTP :**

- Changements d'état d'association SCTP
- Transitions d'état ASP M3UA
- Unités de données de protocole envoyées/reçues

**Métriques M2PA :**

- Transitions d'état de lien (DOWN → ALIGNMENT → PROVING → READY)
- Messages et octets envoyés/reçus par lien
- Erreurs spécifiques au lien (décodage, encodage, SCTP)

**Métriques STP :**

- Messages reçus/envoyés par pair
- Échecs de routage par raison
- Distribution du trafic entre les pairs

**Métriques Client MAP :**

- Requêtes MAP par type d'opération
- Histogrammes de durée de requête
- Jauge des transactions en attente

**Métriques CAP :**

- Requêtes CAP par type d'opération

- Opérations de passerelle CAMEL

**Métriques SMSc :**

- Profondeur de la file d'attente
- Taux de livraison
- Messages échoués

## Intégration Grafana

Les métriques d'OmniSS7 sont compatibles avec Prometheus et Grafana.

---

# Meilleures Pratiques

## Recommandations de Sécurité

1. **Isolation Réseau**
    - Déployer dans un VLAN dédié
    - Règles de pare-feu pour restreindre l'accès
    - Autoriser SCTP uniquement depuis des adresses connues
2. **Sécurité de l'Interface Web**
    - Activer TLS pour la production
    - Utiliser un proxy inverse avec authentification
    - Restreindre aux IPs de gestion
3. **Sécurité de l'API**
    - Mettre en œuvre une limitation de taux
    - Utiliser des clés API ou OAuth
    - Journaliser toutes les requêtes pour audit

## Optimisation des Performances

1. **Limites TPS**
    - Configurer un TPS approprié
    - Surveiller la charge système
    - Ajuster les tampons SCTP
2. **Optimisation du Backend SMS**
    - Garder l'API backend SMS réactive (les requêtes expirent après 5 secondes)
    - Ajuster `auto_flush_tps` et `auto_flush_interval` pour correspondre à la capacité du backend
    - Surveiller la profondeur de la file d'attente et les taux de livraison
3. **Optimisation M3UA**
    - Ajuster les intervalles de battement SCTP
    - Configurer les valeurs de délai d'attente
    - Utiliser plusieurs liens pour la redondance

---

# Multihoming SCTP pour la Redondance Réseau

## Qu'est-ce que le Multihoming SCTP ?

**Le Multihoming SCTP** est une fonctionnalité intégrée du protocole SCTP qui permet à une seule connexion M3UA de se lier à plusieurs adresses IP sur la même interface réseau ou à travers différentes interfaces réseau. Cela fournit une bascule automatique et une redondance au niveau de la couche de transport.

**Avantages Clés :**

- **Bascule Automatique** : Si un chemin réseau échoue, SCTP passe automatiquement à un chemin alternatif sans interrompre la connexion
- **Bascule sans Configuration** : Aucune logique au niveau de l'application n'est nécessaire - SCTP gère la surveillance des chemins et la bascule
- **Fiabilité Améliorée** : Survivre aux pannes réseau, aux pannes de commutateur ou aux pannes de NIC
- **Équilibrage de Charge** : SCTP peut distribuer le trafic sur plusieurs chemins (dépendant de l'implémentation)

# Comment Cela Fonctionne

Lorsque vous configurez plusieurs adresses IP pour une connexion M3UA, SCTP :

1. **Se Lie à Tous les IPs** : Le socket se lie à toutes les adresses IP configurées simultanément
2. **Surveille les Chemins** : SCTP envoie en continu des paquets de battement sur tous les chemins pour surveiller leur santé
3. **Détecte les Pannes** : Si les battements échouent sur le chemin principal, SCTP le marque comme injoignable
4. **Bascule Automatique** : Le trafic passe immédiatement à un chemin de secours sans intervention de l'application
5. **Récupération de Chemin** : Lorsque le chemin échoué se rétablit, SCTP le détecte et le marque à nouveau comme disponible

# Configuration

Le multihoming SCTP est configuré en fournissant une **liste d'adresses IP** au lieu d'un seul tuple IP.

### IP Unique (Traditionnelle)

```
# IP unique - pas de multihoming
local_ip: {10, 179, 4, 10}
```

### Plusieurs IPs (Multihoming Activé)

```
# Plusieurs IPs - multihoming activé
# La première IP est primaire, les IPs suivantes sont des chemins
de secours
local_ip: [{10, 179, 4, 10}, {10, 179, 4, 11}]
```

## Exemples de Configuration

> **Remarque Importante pour le Rôle Serveur (Connexions Entrantes) :**
>
> Lors de la configuration des pairs avec `role: :server` (acceptant des connexions entrantes de pairs multihomés), vous devez spécifier `remote_ip` comme un **unique tuple** - l'adresse IP que le pair distant utilise pour initier la connexion SCTP (SCTP INIT). Ne pas utiliser une liste.
>
> **Pourquoi ?** Le STP fait correspondre les connexions entrantes en fonction de l'IP source du paquet SCTP INIT. SCTP découvrira automatiquement les autres adresses IP multihomées du pair lors de la poignée de main d'association. Le format liste pour `remote_ip` n'est valide que pour `role: :client` (connexions sortantes).
>
> **Exemples :**
>
> ```
> # ✓ CORRECT - Rôle Serveur avec unique remote_ip
> %{
>   role: :server,
>   remote_ip: {10, 0, 2, 100},  # Unique tuple
>   # ...
> }
>
> # ✗ FAUX - Rôle Serveur avec liste ne fera PAS correspondre les
> connexions entrantes
> %{
>   role: :server,
>   remote_ip: [{10, 0, 2, 100}, {10, 0, 2,
> 101}],  # La liste ne fonctionnera pas
>   # ...
> }
> ```

**Exemple 1 : Pair STP avec Multihoming (Rôle Client - Sortant)**

```
# Configuration de pair en mode STP (connexion SORTANTE)
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "Partner_STP_Redundant",
      role: :client,  # Sortant - nous initiions la connexion
      # Multihoming : se lier à deux IPs locales pour redondance
      local_ip: [{213, 57, 23, 200}, {213, 57, 23, 201}],
      local_port: 0,
      # Le pair distant prend également en charge le multihoming -
la liste est OK pour le rôle client
      remote_ip: [{213, 57, 23, 100}, {213, 57, 23, 101}],
      remote_port: 2905,
      routing_context: 1,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

**Exemple 2 : Client MAP avec Multihoming**

```
# Mode client MAP avec multihoming
config :omniss7,
  map_client_enabled: true,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    # Multihoming : deux IPs locales pour la bascule
    local_ip: [{10, 0, 0, 100}, {10, 0, 0, 101}],
    local_port: 2905,
    # STP distant avec support multihoming
    remote_ip: [{10, 0, 0, 1}, {10, 0, 0, 2}],
    remote_port: 2905,
    routing_context: 1
  }
```

**Exemple 3 : Écouteur STP avec Multihoming**

```
# Serveur STP autonome avec multihoming
config :omniss7,
  sctp_handler: %{
    enabled: true,
    # Écouter sur plusieurs IPs pour les connexions entrantes
    local_ip: [{172, 16, 0, 10}, {172, 16, 0, 11}],
    local_port: 2905,
    point_code: 100
  }
```

**Exemple 4 : Rôle Serveur - Acceptation des Connexions Entrantes d'un Pair Multihomé**

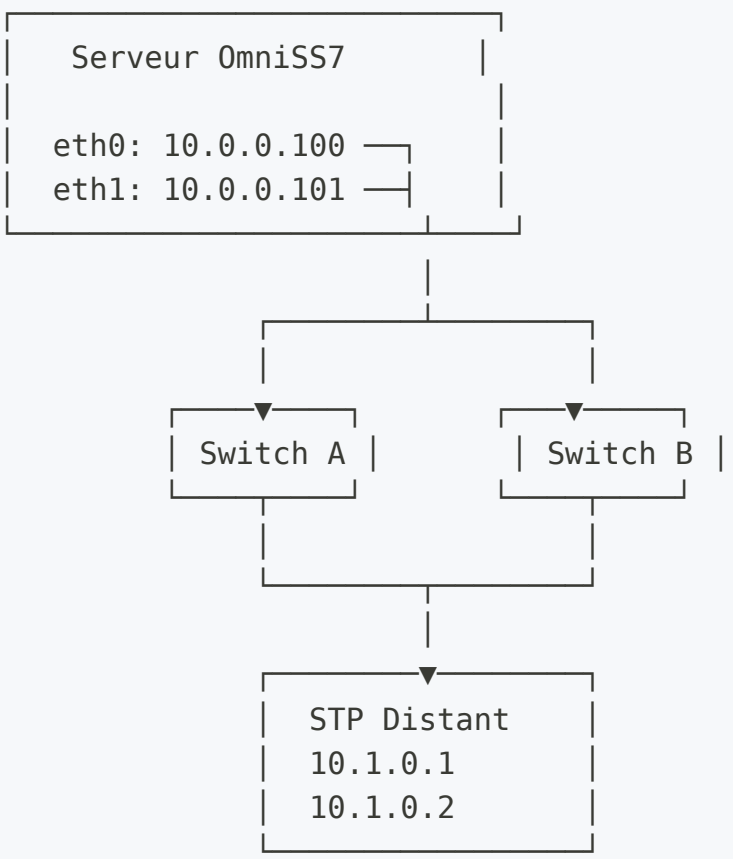
```
# Acceptation de la connexion entrante d'un HLR multihomé
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "HLR",
      role: :server,  # ENTRANT - HLR se connecte à nous
      # Multihoming : nos IPs locales (la liste est OK)
      local_ip: [{10, 0, 1, 10}, {10, 0, 1, 11}],
      local_port: 2905,
      # IMPORTANT : Unique tuple uniquement - l'IP que l'HLR
utilise pour initier SCTP INIT
      # SCTP découvrira automatiquement les autres IPs de l'HLR
lors de la poignée de main
      remote_ip: {10, 0, 2, 100},  # IP primaire uniquement (PAS
une liste !)
      remote_port: 0,  # Accepter de n'importe quel port source
      routing_context: 1,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

**Exemple 5 : Configuration Mixte (Rétrocompatible)**

```
# Mélange de pairs à IP unique et multihomés
config :omniss7,
  m3ua_peers: [
    # Pair hérité - IP unique
    %{
      peer_id: 1,
      name: "Legacy_STP",
      role: :client,
      local_ip: {10, 0, 0, 1},      # Tuple IP unique
      local_port: 0,
      remote_ip: {10, 0, 0, 10},
      remote_port: 2905,
      routing_context: 1,
      point_code: 100
    },
    # Nouveau pair - multihoming
    %{
      peer_id: 2,
      name: "Redundant_STP",
      role: :client,
      local_ip: [{10, 0, 0, 2}, {10, 0, 0, 3}],  # Liste d'IP
      local_port: 0,
      remote_ip: [{10, 0, 0, 20}, {10, 0, 0, 21}],
      remote_port: 2905,
      routing_context: 2,
      point_code: 200
    }
  ]
```

# Scénarios de Topologie Réseau

### Scénario 1 : Dual NICs (Déploiement Commun)



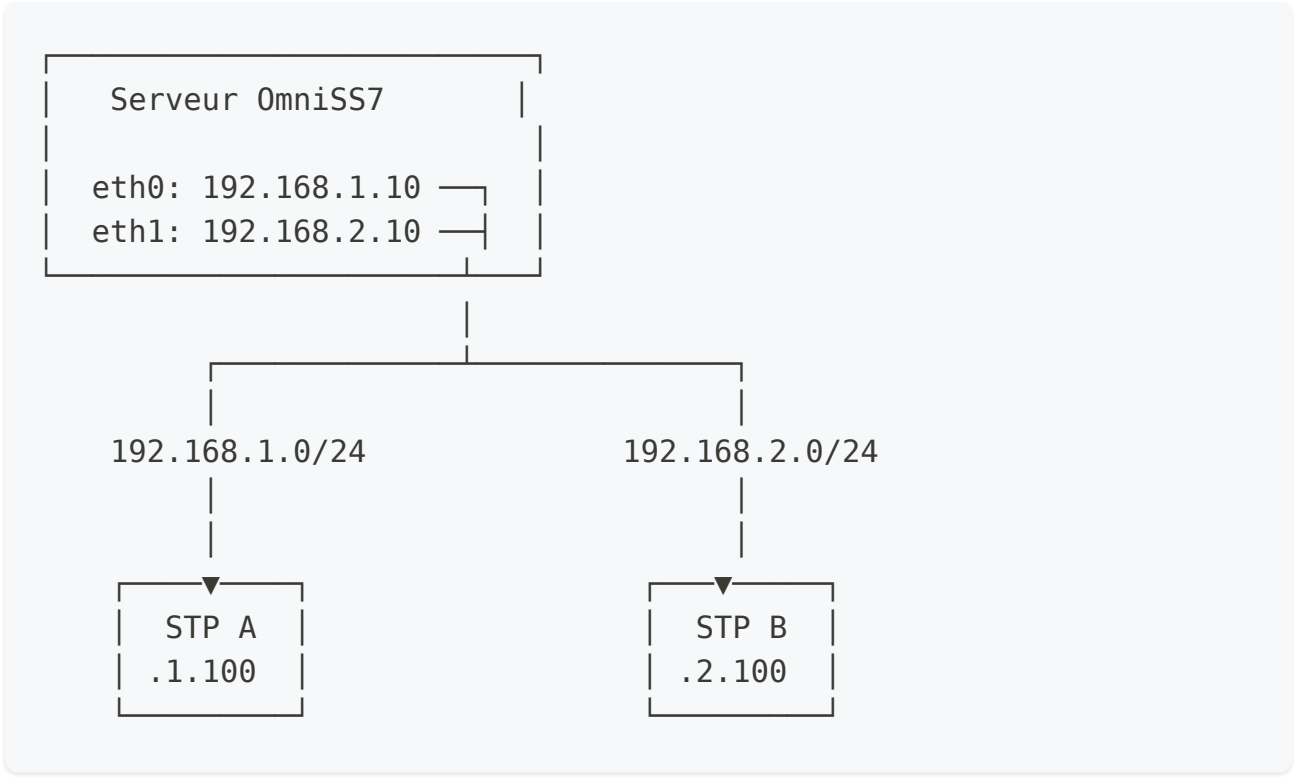

**Configuration :**

```
local_ip: [{10, 0, 0, 100}, {10, 0, 0, 101}]   # Les deux NICs
remote_ip: [{10, 1, 0, 1}, {10, 1, 0, 2}]      # Pair distant
```

**Avantages :**

- Survivent à la panne d'un NIC
- Survivent à la panne d'un commutateur
- Bascule automatique en <1 seconde

**Scénario 2 : Plusieurs Sous-Réseaux**




**Configuration :**

```
local_ip: [{192, 168, 1, 10}, {192, 168, 2, 10}]
remote_ip: [{192, 168, 1, 100}, {192, 168, 2, 100}]
```

**Avantages :**

- Survivent à la panne de sous-réseau
- Redondance géographique possible
- Chemins de routage indépendants

# Surveillance et Journalisation

Lorsque le multihoming est activé, vous verrez des messages de journal indiquant la configuration :

**Multihoming Réussi**

```
[info] Client SCTP multihoming : lié à 2 IPs locales
[info] Écouteur STP multihoming activé : 2 IPs locales liées
```

**Événements de Bascule de Chemin**

```
[warning] [MULTIHOMING] Chemin 10.0.0.100 est INJOIGNABLE pour le
pair Partner_STP (assoc_id=1)
[info] [MULTIHOMING] Chemin 10.0.0.101 est maintenant PRIMARY pour
le pair Partner_STP (assoc_id=1)
[info] [MULTIHOMING] Chemin 10.0.0.100 est maintenant DISPONIBLE
pour le pair Partner_STP (assoc_id=1)
```

# Affichage de l'Interface Web

L'interface Web affiche automatiquement les informations de multihoming :

**Page d'État M3UA :**

- **IP Unique** : Affiche comme `10.0.0.100`
- **Multiples IPs** : Affiche comme `10.0.0.100 (+1)` ou `10.0.0.100 (+2)`
- **Vue Détails** : Affiche toutes les IPs avec des étiquettes primaire/backup

# Meilleures Pratiques

**1. Conception Réseau**

- **Utiliser différents NICs** pour une redondance maximale
- **Différents commutateurs** pour survivre aux pannes de commutateur
- **Différents sous-réseaux** si possible pour la diversité de routage
- **Même datacenter initialement** - tester avant séparation géographique

**2. Planification des Adresses IP**

- **La première IP est primaire** - assurez-vous qu'elle est sur le chemin le plus fiable
- **L'ordre compte** - lister les IPs par ordre de préférence
- **Adressage cohérent** - utiliser des schémas d'adressage similaires pour le dépannage

**3. Tester la Bascule**

```
# Désactiver l'interface primaire pour tester la bascule
sudo ip link set eth0 down

# Surveiller les journaux pour la bascule
tail -f /var/log/omniss7.log | grep MULTIHOMING

# Réactiver l'interface
sudo ip link set eth0 up
```

**4. Les Deux Côtés Doivent Prendre en Charge le Multihoming**

- **Optimal** : Les deux côtés local et distant utilisent plusieurs IPs
- **Acceptable** : Un seul côté utilise le multihoming
- **Remarque** : La redondance est meilleure lorsque les deux points de terminaison la prennent en charge

**5. Configuration du Pare-feu**

```
# Autoriser SCTP sur toutes les IPs de multihoming
iptables -A INPUT -p sctp --dport 2905 -s 10.0.0.0/24 -j ACCEPT
iptables -A INPUT -p sctp --dport 2905 -s 10.1.0.0/24 -j ACCEPT
```

# Dépannage

**Problème : Multihoming Ne Fonctionne Pas**

**Symptômes** : Seule l'IP primaire est utilisée, pas de bascule

**Vérifications :**

1. Vérifiez le support SCTP d'Erlang : `erl -eval 'gen_sctp:open(9999, [binary, {ip, {127,0,0,1}}]).'`
2. Vérifiez le module SCTP du noyau : `lsmod | grep sctp`
3. Chargez SCTP si nécessaire : `sudo modprobe sctp`
4. Vérifiez que les deux IPs sont configurées sur le système : `ip addr show`

**Problème : Le Chemin Ne Basculé Pas**

**Symptômes** : Le chemin principal est marqué comme hors service mais le trafic ne passe pas

**Vérifications :**

1. Vérifiez les paramètres de battement SCTP
2. Vérifiez que la table de routage a des routes pour tous les chemins
3. Vérifiez que le pare-feu autorise SCTP sur toutes les IPs
4. Consultez les journaux de surveillance des chemins SCTP

**Problème : Fluctuation Fréquente des Chemins**

**Symptômes** : Les chemins passent constamment de UP à DOWN

**Vérifications :**

1. Instabilité réseau - vérifiez les liens physiques
2. Battement SCTP trop agressif - peut nécessiter un réglage
3. Pare-feu supprimant les battements SCTP
4. Problèmes de MTU sur un chemin

# Considérations de Performance

- **Surcharge minimale** : Les battements SCTP sont petits et peu fréquents
- **Aucun changement d'application** : Le multihoming est transparent pour la couche d'application
- **Bascule rapide** : Détection et bascule typiquement <1 seconde
- **Récupération automatique** : Aucune intervention manuelle nécessaire

# Compatibilité

- **Rétrocompatible** : Le format de tuple IP unique fonctionne toujours
- **Déploiements mixtes** : Peut mélanger des pairs à IP unique et multihomés
- **Tous les modes pris en charge** : Fonctionne dans les modes STP, HLR, SMSc et Client MAP
- **Exigence Erlang** : Nécessite Erlang avec le support SCTP compilé

## Surveillance et Alerte

**Métriques Clés :**

- État de la connexion M3UA
- Taux de réussite des requêtes MAP
- Temps de réponse de l'API
- Profondeur de la file de messages

**Seuils d'Alerte :**

- M3UA hors service > 1 minute
- Taux de délai d'attente MAP > 10%
- Profondeur de la file > 1000
- Taux d'erreur de l'API > 5%

---

# Référence de Configuration Complète

## Tous les Paramètres de Configuration

Cette section fournit une référence complète de tous les paramètres de configuration disponibles dans tous les modes de fonctionnement.

**Configuration du Logger (`:logger`)**

```
config :logger, :default_formatter,
  format: {OmniLogger.JsonFormatter, :format},
  metadata: :all,
  truncate: :infinity

config :logger,
  level: :debug  # :debug | :info | :warning | :error
```

| Paramètre | Type | Requis | |
|---|---|---|---|
| `:logger.level` | Atom | Non | `:debug` |
| `:logger.default_formatter.format` | Tuple | Non | `{OmniLog :format}` |
| `:logger.default_formatter.metadata` | Atom/Liste | Non | `:all` |
| `:logger.default_formatter.truncate` | Integer/Atom | Non | `:infinit` |

---

**Configuration de l'Interface Web (`:control_panel`)**

```
config :control_panel, ControlPanelWeb.Endpoint,
  force_ssl: [rewrite_on: [:x_forwarded_proto]],
  server: true,
  url: [host: "0.0.0.0", path: "/"],
  https: [
    port: 8087,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ]

config :control_panel,
  page_order: [
    "/application", "/configuration", "/events", "/logs",
"/client",
    "/m3ua", "/routing", "/routing_test", "/smsc_links",
"/smsc_subscribers",
    "/hlr_links", "/subscribers", "/camel_sessions",
"/camel_request",
    "/msisdn_imsi_test"
  ]
```

| Paramètre | Type | Requis | |
|---|---|---|---|
| `ControlPanelWeb.Endpoint.url.host` | String | Non | `"0.0.0.` |
| `ControlPanelWeb.Endpoint.https.port` | Integer | Non | `8087` |
| `ControlPanelWeb.Endpoint.https.keyfile` | String | Non | `"priv/c` |
| `ControlPanelWeb.Endpoint.https.certfile` | String | Non | `"priv/c` |
| `ControlPanelWeb.Endpoint.force_ssl` | Keyword | Non | `[rewrit` `[:x_for` |
| `page_order` | Liste de Strings | Non | voir exe |

---

**Configuration du SocketHandler SCTP (`:omniss7`)**

```
config :omniss7,
  sctp_handler: %{
    enabled: false,
    local_ip: {127, 0, 0, 1},
    local_port: 2905
  },
  enable_gt_routing: true,
  m3ua_peers: [...],
  m3ua_routes: [...],
  m3ua_gt_routes: [...]
```

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `sctp_handler.enabled` | Boolean | Oui | `false` | Activer le mode STP au démarrage |
| `sctp_handler.local_ip` | Tuple | Oui | `{127, 0, 0, 1}` | IP à lier pour M3UA entrant |
| `sctp_handler.local_port` | Integer | Oui | `2905` | Port SCTP pour M3UA |
| `enable_gt_routing` | Boolean | Non | `false` | Activer le routage de Titre Global |

**Paramètres des Pairs M3UA :**

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `peer_id` | Integer | Oui | Identifiant unique du pair |
| `name` | String | Oui | Nom descriptif du pair |
| `role` | Atom | Oui | `:client` ou `:server` |
| `local_ip` | Tuple ou Liste | Si `:client` | IP(s) locale(s) à lier. Unique : `{10, 0, 0, 1}` ou Liste : `[{10, 0, 0, 1}, {10, 0, 0, 2}]` |
| `local_port` | Integer | Si `:client` | Port local (0 pour dynamique) |
| `remote_ip` | Tuple ou Liste | Oui | IP(s) du pair distant. Unique : `{10, 0, 0, 10}` ou Liste : `[{10, 0, 0, 10}, {10, 0, 0, 11}]` |
| `remote_port` | Integer | Si `:client` | Port du pair distant |
| `routing_context` | Integer | Oui | Contexte de routage M3UA |
| `point_code` | Integer | Oui | Code de point SS7 |
| `network_indicator` | Atom | Non | `:international` ou `:national` |

**Paramètres de Route M3UA :**

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `dest_pc` | Integer | Oui | Code de point de destination |
| `peer_id` | Integer | Oui | Pair à travers lequel router |
| `priority` | Integer | Oui | Priorité de la route (plus bas = plus haute priorité) |
| `network_indicator` | Atom | Non | `:international` ou `:national` |

**Paramètres de Route GT M3UA :**

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `gt_prefix` | String | Oui | Préfixe de Titre Global à correspondre |
| `peer_id` | Integer | Oui | Pair de destination |
| `priority` | Integer | Oui | Priorité de la route |
| `description` | String | Non | Description de la route pour journalisation |
| `source_ssn` | Integer | Non | Correspondre uniquement si le SSN source correspond |
| `dest_ssn` | Integer | Non | Réécrire le SSN de destination à cette valeur |

---

**Configuration du Client MAP (`:omniss7`)**

```
config :omniss7,
  map_client_enabled: false,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :map_client_asp,
    local_ip: {10, 0, 0, 100},
    local_port: 2905,
    remote_ip: {10, 0, 0, 1},
    remote_port: 2905,
    routing_context: 1
  }
```

| Paramètre | Type | Requis | Par Défaut |
| --- | --- | --- | --- |
| `map_client_enabled` | Boolean | Oui | `false` |
| `map_client_m3ua.mode` | String | Oui | `"ASP"` |
| `map_client_m3ua.callback` | Tuple | Oui | `{MapClient, :handle_payload, []}` |
| `map_client_m3ua.process_name` | Atom | Oui | `:map_client_asp` |
| `map_client_m3ua.local_ip` | Tuple | Oui | - |
| `map_client_m3ua.local_port` | Integer | Oui | `2905` |
| `map_client_m3ua.remote_ip` | Tuple | Oui | - |
| `map_client_m3ua.remote_port` | Integer | Oui | `2905` |
| `map_client_m3ua.routing_context` | Integer | Oui | - |

### Configuration du Centre SMS (`:omniss7`)

```
config :omniss7,
  auto_flush_enabled: false,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: nil,
  auto_flush_tps: 10
```

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `auto_flush_enabled` | Boolean | Non | `false` | Activer le vidage automatique de la file d'attente SMS |
| `auto_flush_interval` | Integer | Non | `10000` | Intervalle de sondage de la file d'attente (millisecondes) |
| `auto_flush_dest_smsc` | String/nil | Non | `nil` | Filtrer par SMSC de destination (nil = tous) |
| `auto_flush_tps` | Integer | Non | `10` | Transactions maximales par seconde |

---

### Configuration de l'API HTTP (`:omniss7`)

Le backend SMS utilise désormais l'API HTTP au lieu de connexions directes à la base de données.

```
config :omniss7,
  smsc_api_base_url: "https://omnimessage.example.com:8443",
  smsc_name: "omni-smsc01"  # Optionnel : par défaut au nom dérivé
de l'hôte
```

**Paramètres de l'API :**

| Paramètre | Type | Requis | Par Défaut |
|---|---|---|---|
| `smsc_api_base_url` | String | Oui | `"https://omnimessage.example.co` |
| `smsc_name` | String | Non | dérivé de l'hôte |

**Points de Terminaison de l'API Utilisés :**

- `POST /api/frontends` - Enregistrer cette instance frontend avec le backend
- `POST /api/messages_raw` - Insérer de nouveaux messages SMS
- `GET /api/messages` - Récupérer la file de messages (avec l'en-tête `smsc`)
- `PATCH /api/messages/{id}` - Marquer le message comme livré
- `PUT /api/messages/{id}` - Mettre à jour l'état du message
- `POST /api/events` - Ajouter le suivi des événements
- `GET /api/status` - Point de terminaison de vérification de santé

**Notes de Configuration :**

- La vérification SSL est désactivée par défaut pour les certificats auto-signés
- Les requêtes HTTP expirent après 5 secondes
- Tous les horodatages sont au format ISO 8601

- L'API utilise JSON pour les corps de requête/réponse

---

# Documentation Connexe

- ← Retour à la Documentation Principale
- Guide STP
- Guide Client MAP
- Guide Centre SMS
- Guide HLR

---

**OmniSS7** par Omnitouch Network Services

# Référence de Configuration

← Retour à la Documentation Principale

Ce document fournit une référence complète pour tous les paramètres de configuration d'OmniSS7.

## Table des Matières

1. Aperçu
2. Où se Trouve la Configuration
3. Drapeaux de Mode Opérationnel
4. Paramètres de Mode HLR
5. Paramètres de Mode SMSc
6. Paramètres de Mode STP
7. Paramètres de Mode Passerelle CAMEL
8. Paramètres NAT de Titre Global
9. Paramètres de Connexion M3UA
10. État du Sous-système SCCP
11. Filtrage de Passerelle
12. Paramètres de Passerelle USSD
13. Drapeaux d'Exécution / Supervision
14. Paramètres d'Infrastructure
15. Paramètres de Base de Données
16. Valeurs Codées en Dur

---

# Aperçu

OmniSS7 prend en charge quatre modes opérationnels, sélectionnés entièrement par configuration :

- **Mode STP** - Point de Transfert de Signal pour le routage
- **Mode HLR** - Registre de Localisation Domiciliaire pour la gestion des abonnés
- **Mode SMSc** - Centre SMS pour la livraison de messages
- **Mode CAMEL GW** - Passerelle CAMEL pour le contrôle intelligent des appels

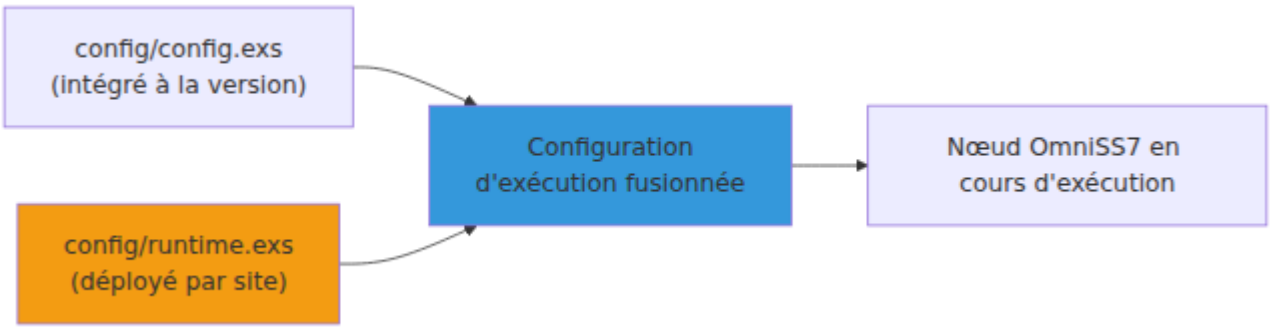
Un seul nœud peut combiner des rôles (par exemple HLR + SMSc) en activant les Drapeaux de Mode Opérationnel.

---

# Où se Trouve la Configuration

OmniSS7 lit sa configuration à partir de deux couches. Comprendre quel fichier possède une valeur détermine où vous devez la modifier sur un système déployé.

| Fichier | Rôle | Modifier lors du déploiement ? |
|---|---|---|
| `config/config.exs` | **Configuration par défaut / à la compilation** fournie avec la version. Définit les pages du panneau de contrôle, les contrôleurs REST `:api_ex`, le formatteur de journalisation, et tous les paramètres par défaut de base. | Non : fait partie de la construction. |
| `config/runtime.exs` (dans le dépôt) | Un **stub de test de 4 lignes** uniquement. Dans l'arbre source, il ne configure rien au-delà de l'environnement de test. | Non. |
| `config/runtime.exs` (déployé) | La **configuration d'exécution par déploiement**, déployée séparément sur chaque serveur (généralement rendue à partir d'un modèle Ansible/Jinja). C'est ici que les drapeaux de mode, les GT, les codes de point, les pairs M3UA et les backends API sont définis pour un site donné. | **Oui. C'est le fichier destiné à l'opérateur.** |



> **En résumé :** les exemples dans ce document utilisent `config :omniss7, ...`. Sur un nœud en direct, ceux-ci vont dans le **déployé** `config/runtime.exs`. Le `config/runtime.exs` engagé dans le dépôt est intentionnellement un stub utilisé uniquement par la suite de tests automatisés. Ne le confondez pas avec la configuration de production.

Tous les clés ci-dessous sont lues via `Application.get_env(:omniss7, ...)` (ou l'application pertinente) à l'exécution, donc les valeurs par défaut affichées sont les valeurs utilisées lorsqu'une clé est omise.

---

# Drapeaux de Mode Opérationnel

Contrôlez quelles fonctionnalités sont activées.

| Paramètre | Type | Par défaut | Description | Modes |
|---|---|---|---|---|
| `map_client_enabled` | Booléen | `false` | Activer le client MAP et la connectivité M3UA | Tous |
| `hlr_mode_enabled` | Booléen | `false` | Activer les fonctionnalités spécifiques au HLR | HLR |
| `smsc_mode_enabled` | Booléen | `false` | Activer les fonctionnalités spécifiques au SMSc | SMSc |
| `cap_client_enabled` | Booléen | `false` | Activer le client CAP pour les opérations CAMEL | CAMEL GW |
| `camelgw_mode_enabled` | Booléen | `false` | Activer les fonctionnalités de la Passerelle CAMEL | CAMEL GW |
| `ussd_gateway_enabled` | Booléen | `false` | Activer la Passerelle USSD (pont HTTP/JSON) | USSD GW |

**Exemple**:

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false
```

---

# Paramètres de Mode HLR

Configuration pour le mode HLR (Registre de Localisation Domiciliaire).

## Configuration API HLR

| Paramètre | Type | Par défaut | Requis | |
|---|---|---|---|---|
| `hlr_api_base_url` | Chaîne | - | **Oui** | U te H |
| `hlr_api_key` | Chaîne | - | **Oui** | C C d o c C c d ê l' a h S m v re é e h |
| `hlr_api_status_path` | Chaîne | `"/api/status"` | Non | C h p s li p **H** c / |

| Paramètre | Type | Par défaut | Requis | |
|---|---|---|---|---|
| `hlr_service_center_gt_address` | Chaîne | - | **Oui** | A G re re U |
| `smsc_service_center_gt_address` | Chaîne | - | **Oui** | A S d S |

> **Vérification SSL :** Il n'y a **pas** de clé `hlr_api_verify_ssl`. Elle n'est lue nulle part dans le code. La vérification des certificats TLS pour les appels API en arrière-plan est régie par le champ `verify_ssl` de la carte générique `:api` (voir Vérification SSL du Backend API).

**Exemple**:

```
config :omniss7,
  hlr_api_base_url: "https://omnihss.example.com:8443",
  hlr_api_key: "<Clé API OmniHSS avec le rôle hlr_interworking>",
  hlr_api_status_path: "/api/status",
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112"
```

## Configuration AlertServiceCenter

Lorsqu'un abonné effectue une mise à jour de localisation, le HLR envoie des messages alertServiceCenter aux GT SMSc configurés pour indiquer que l'abonné est maintenant joignable.

| Paramètre | Type | Par défaut | Requis | D |
|---|---|---|---|---|
| `hlr_smsc_alert_gts` | Liste de Chaînes | `[]` | Non | Liste Glob pour notifi alert |
| `hlr_alert_location_expiry_seconds` | Entier | `172800` | Non | Duré de la en se défa heur pour l'exp |

**Exemple**:

```
config :omniss7,
  hlr_smsc_alert_gts: [
    "15559876543",
    "15559876544"
  ],
  hlr_alert_location_expiry_seconds: 172800  # 48 heures
```

## Mapping MSISDN ↔ IMSI

Configuration pour la génération d'IMSI synthétiques à partir de MSISDN. Pour une explication technique détaillée de l'algorithme de mapping, voir Mapping MSISDN ↔ IMSI dans le Guide HLR.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `hlr_imsi_plmn_prefix` | Chaîne | `"50557"` | Non | Préfixe PLMN (MCC+MNC) pour la génération d'IMSI synthétiques |
| `hlr_msisdn_country_code` | Chaîne | `"61"` | Non | Préfixe de code pays pour le mapping inverse IMSI→MSISDN |
| `hlr_msisdn_nsn_offset` | Entier | `0` | Non | Décalage dans le MSISDN où commence le NSN (typiquement la longueur du code pays) |
| `hlr_msisdn_nsn_length` | Entier | `9` | Non | Longueur du Numéro d'Abonné National à extraire du MSISDN |

**Exemple** (code pays à 2 chiffres) :

```
config :omniss7,
  hlr_imsi_plmn_prefix: "50557",      # MCC 505 + MNC 57
  hlr_msisdn_country_code: "99",      # Exemple de code pays à 2
chiffres
  hlr_msisdn_nsn_offset: 2,           # Ignorer le code pays à 2
chiffres
  hlr_msisdn_nsn_length: 9            # Extraire le NSN à 9
chiffres
```

**Exemple** (code pays à 3 chiffres) :

```
config :omniss7,
  hlr_imsi_plmn_prefix: "50557",      # MCC 505 + MNC 57
  hlr_msisdn_country_code: "999",     # Exemple de code pays à 3
chiffres
  hlr_msisdn_nsn_offset: 3,           # Ignorer le code pays à 3
chiffres
  hlr_msisdn_nsn_length: 8            # Extraire le NSN à 8
chiffres
```

**Important** : Définissez `nsn_offset` à la longueur de votre code pays pour extraire correctement le NSN. Par exemple :

- Code pays "9" (1 chiffre) → `nsn_offset: 1`
- Code pays "99" (2 chiffres) → `nsn_offset: 2`
- Code pays "999" (3 chiffres) → `nsn_offset: 3`

## Configuration InsertSubscriberData (ISD)

Configuration pour les données de provisionnement des abonnés envoyées aux VLR lors de la mise à jour de localisation. Pour une explication détaillée de la séquence ISD et du flux de messages, voir Configuration InsertSubscriberData dans le Guide HLR.

| Paramètre | Type | Par défaut | Requis | |
|---|---|---|---|---|
| `isd_network_access_mode` | Atome | `:packetAndCircuit` | Non | Typ<br>rés<br>`:pa`<br>`:pa`<br>`:ci` |
| `isd_send_ss_data` | Booléen | `true` | Non | Env<br>ave<br>Ser<br>Sup |
| `isd_send_call_barring` | Booléen | `true` | Non | Env<br>ave<br>Bar |

**Exemple**:

```
config :omniss7,
  isd_network_access_mode: :packetAndCircuit,
  isd_send_ss_data: true,
  isd_send_call_barring: true
```

## Configuration CAMEL

Configuration pour le routage intelligent des appels basé sur CAMEL. Pour une explication détaillée de l'intégration CAMEL et des clés de service, voir Intégration CAMEL dans le Guide HLR.

| Paramètre | Type | Par défaut | Re |
|---|---|---|---|
| `camel_service_key` | Entier | `11_110` | No |
| `camel_trigger_detection_point` | Atome | `:termAttemptAuthorized` | No |
| `camel_gsmscf_gt_address` | Chaîne | (utilise le GT appelé) | No |

| `hlr_camel_subscriptions` | Liste | `[]` | Non | O-CSI / T-CSI par préfixe IMSI envoyés au VLR dans InsertSubscriberData (voir Abonnements CAMEL dans ISD) |

**Exemple**:

```
config :omniss7,
  camel_service_key: 11_110,
  camel_trigger_detection_point: :termAttemptAuthorized,
  hlr_camel_subscriptions: [
    %{
      imsi_prefix: "3133800000009",
      o_csi: %{service_key: 100, gsmscf_address: "14155550300",
default_call_handling: :release_call},
      t_csi: %{service_key: 101, gsmscf_address: "14155550300",
default_call_handling: :continue_call}
    }
  ]
```

## Préfixes VLR Domiciliaires

Configuration pour distinguer les abonnés domiciliés des abonnés en itinérance. Pour une explication détaillée de la détection domicile/itinérance et des opérations PRN, voir Gestion des Abonnés en Itinérance dans le Guide HLR.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `home_vlr_prefixes` | Liste | `["5551231"]` | Non | Préfixes GT VLR considérés comme réseau "domiciliaire" |

**Exemple**:

```
config :omniss7,
  home_vlr_prefixes: ["5551231", "5551234"]
```

---

# Paramètres de Mode SMSc

Configuration pour le mode Centre SMS.

## Configuration API SMSc

| Paramètre | Type | Par défaut |
|---|---|---|
| `smsc_api_base_url` | Chaîne | `"https://omnimessage.exampl` |
| `smsc_api_status_path` | Chaîne | `"/api/status"` |
| `smsc_name` | Chaîne | `"{hostname}_SMSc"` |
| `smsc_service_center_gt_address` | Chaîne | - |

> **Vérification SSL :** Il n'y a **pas** de clé `smsc_api_verify_ssl`. Elle n'est lue nulle part dans le code. La vérification des certificats TLS pour les appels API en arrière-plan est régie par le champ `verify_ssl` de la carte générique `:api` (voir Vérification SSL du Backend API).

**Exemple**:

```
config :omniss7,
  smsc_api_base_url: "https://omnimessage.example.com:8443",
  smsc_api_status_path: "/api/status",
  smsc_name: "ipsmgw",
  smsc_service_center_gt_address: "55512341112"
```

**Remarque** : L'enregistrement en frontend se produit toutes les **5 minutes** (codé en dur) via le module `SMS.FrontendRegistry`.

## Configuration d'Auto-Vidage

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `auto_flush_enabled` | Booléen | `true` | Non | Activer le traitemer automatique de la f d'attente SMS |
| `auto_flush_interval` | Entier | `10_000` | Non | Intervalle de traitement de la file d'attente en millisecondes |
| `auto_flush_dest_smsc` | Chaîne | - | **Oui** | Nom du SMSc de destination pour l'auto-vidage |
| `auto_flush_tps` | Entier | `10` | Non | Taux de traitement des messages (transactions/secon |

**Exemple**:

```
config :omniss7,
  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10
```

---

# Paramètres de Mode STP

Configuration pour le mode Point de Transfert de Signal M3UA. Pour une configuration de routage détaillée et des exemples, voir le Guide de Configuration STP.

## Serveur STP Autonome

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `sctp_handler.enabled` | Booléen | `false` | Non | Activer le serveur SocketHandler SCTP autonome |
| `sctp_handler.local_ip` | Tuple ou Liste | `{127, 0, 0, 1}` | Non | Adresse(s) IP à écouter pour les connexions. IP unique : `{10, 0, 0, 1}` ou plusieurs IP pour le multihoming SCTP : `[{10, 0, 0, 1}, {10, 0, 0, 2}]` |
| `sctp_handler.local_port` | Entier | `2905` | Non | Port à écouter |
| `sctp_handler.point_code` | Entier | - | **Oui** (si activé) | Code de point SS7 de ce STP |

**Exemple (IP Unique)**:

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    point_code: 100
  }
```

**Exemple (Multihoming SCTP)**:

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    # Plusieurs IP pour redondance
    local_ip: [{10, 179, 4, 10}, {10, 179, 4, 11}],
    local_port: 2905,
    point_code: 100
  }
```

> **Remarque** : Pour des informations détaillées sur la configuration et les avantages du multihoming SCTP, voir Multihoming SCTP dans le Guide Commun.

## Routage par Titre Global

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `enable_gt_routing` | Booléen | `false` | Non | Activer le routage GT en plus du routage PC |

**Exemple**:

```
config :omniss7,
  enable_gt_routing: true
```

# Configuration des Pairs M3UA/M2PA

Les pairs sont configurés via la liste `m3ua_peers` (prend en charge les protocoles M3UA et M2PA). Pour des exemples de configuration complets, voir le Guide de Configuration STP et Support du Protocole M2PA.

**Paramètres Communs aux Pairs :**

| Paramètre | Type | Par défaut | Requis | Descripti |
|---|---|---|---|---|
| `peer_id` | Entier | - | **Oui** | Identifiant unique du p |
| `name` | Chaîne | - | **Oui** | Nom descri du pair |
| `protocol` | Atome | `:m3ua` | Non | Type de protocole : `:m3ua` ou `:` |
| `role` | Atome | `:client` | Non | Rôle de connexion : `:client`, `:server`, `:` ou `:sgp` |
| `local_ip` | Tuple ou Liste | - | **Oui** | Adresse(s) l locale(s) po binding |
| `local_port` | Entier | - | **Oui** | Port SCTP lo (M3UA : 29( M2PA : 356 |
| `remote_ip` | Tuple ou Liste | - | **Oui** | Adresse(s) l distante(s) |
| `remote_port` | Entier | - | **Oui** | Port SCTP distant |
| `routing_context` | Entier | - | Non | Identifiant c contexte de routage M3 (M3UA uniquement |

| Paramètre | Type | Par défaut | Requis | Descripti |
|---|---|---|---|---|
| `point_code` | Entier | - | **Oui** | Code de po local |
| `network_indicator` | Atome | `:international` | Non | Indicateur d réseau : `:internati` ou `:nation` |
| `initiate_connection` | Booléen | `true` | Non | Indique si la connexion S doit être ini |

**Paramètres Spécifiques à M2PA :**

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `adjacent_point_code` | Entier | - | **Oui** (M2PA) | Code de point du pair adjacent |

**Gestion des Sockets :**

M2PA utilise automatiquement SCTP.SocketHandler pour la gestion des sockets partagées. Tous les pairs M2PA utilisent le socket partagé, ce qui permet à plusieurs pairs de partager efficacement le même port SCTP. Pour une configuration détaillée, voir Exigences de Socket M2PA.

**Exemple (Pair M3UA)**:

```
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "HLR_East",
      protocol: :m3ua,
      role: :sgp,
      local_ip: {10, 179, 4, 10},
      local_port: 2905,
      remote_ip: {10, 179, 4, 20},
      remote_port: 2905,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

**Exemple (Pair M2PA)**:

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,
    point_code: 100
  },
  m3ua_peers: [
    %{
      peer_id: 2,
      name: "M2PA_Link_STP_West",
      protocol: :m2pa,
      role: :client,
      local_ip: {10, 179, 4, 10},
      local_port: 3565,
      remote_ip: {10, 179, 4, 30},
      remote_port: 3565,
      point_code: 100,
      adjacent_point_code: 200
    }
  ]
```

# Routes de Code de Point M3UA

Configuration de routage par Code de Point. Les routes définissent quel pair utiliser pour atteindre des codes de point de destination spécifiques.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `m3ua_routes` | Liste de Cartes | `[]` | Non | Liste des routes de code de point. Si non spécifié, les routes sont générées automatiquement à partir des codes de point des pairs. |

**Format de Route :** Chaque route dans `m3ua_routes` doit être une carte avec :

- `dest_pc`: Code de point de destination (Entier)
- `peer_id`: ID du pair à travers lequel router (Entier)
- `priority`: Priorité de la route - valeur plus basse = priorité plus élevée (Entier)
- `network_indicator`: Indicateur de réseau (Atome) : `:international` ou `:national`

**Exemple :**

```
config :omniss7,
  m3ua_routes: [
    # Route vers PC 100 via pair 1 (priorité la plus élevée)
    %{dest_pc: 100, peer_id: 1, priority: 1, network_indicator:
:international},
    # Route vers PC 200 via pair 2
    %{dest_pc: 200, peer_id: 2, priority: 1, network_indicator:
:international},
    # Équilibrage de charge : même dest_pc avec différentes
priorités
    %{dest_pc: 300, peer_id: 3, priority: 1, network_indicator:
:international},
    %{dest_pc: 300, peer_id: 4, priority: 2, network_indicator:
:international}
  ]
```

## Routes de Titre Global M3UA

Routage basé sur le préfixe de Titre Global avec transformation avancée des paramètres SCCP. La correspondance du préfixe le plus long est utilisée en premier, puis la priorité.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `m3ua_gt_routes` | Liste de Cartes | `[]` | Non | Liste des règles de routage par Titre Global avec transformations optionnelles |

**Format de Route :** Chaque route dans `m3ua_gt_routes` doit être une carte avec :

**Paramètres de Base :**

- `gt_prefix`: Préfixe de Titre Global à correspondre (Chaîne) - chaîne vide correspond à tous

- `peer_id`: ID du pair à travers lequel router (Entier) - utiliser 0 pour DROP le trafic
- `priority`: Priorité de la route - valeur plus basse = priorité plus élevée (Entier)
- `description`: Description lisible par l'homme (Chaîne)

**Paramètres de Correspondance Optionnels :**

- `source_ssn`: Correspondre au Numéro de Sous-Système source (Entier)
- `source_tt`: Correspondre au Type de Traduction source (Entier)
- `source_npi`: Correspondre à l'Indicateur de Plan de Numérotation source (Entier)
- `source_nai`: Correspondre à l'Indicateur de Nature d'Adresse source (Entier)

**Paramètres de Transformation Optionnels :**

- `dest_ssn`: Transformer le SSN dans le message transféré (Entier)
- `dest_tt`: Transformer le Type de Traduction dans le message transféré (Entier)
- `dest_npi`: Transformer l'Indicateur de Plan de Numérotation dans le message transféré (Entier)
- `dest_nai`: Transformer l'Indicateur de Nature d'Adresse dans le message transféré (Entier)
- `route_on_ssn`: Inverser l'indicateur de routage de la Partie Appelée pour router sur SSN (Booléen, par défaut `false`). Laissez `false` pour le routage GT de transit afin que le nœud en aval effectue GTT (ITU-T Q.714 §2.4) ; définissez `true` **uniquement** lorsque ce STP est le traducteur final livrant à un point de terminaison uniquement SSN. Voir Guide STP → Route sur SSN Après GTT.

```
**Valeurs communes :**
- **Type de traduction (TT) :** 0=Inconnu, 1=International,
2=National, 3=Spécifique au réseau
- **Plan de numérotation (NPI) :** 0=Inconnu, 1=ISDN(E.164),
6=Mobile(E.212)
- **Nature de l'adresse (NAI) :** 0=Inconnu, 1=Abonné, 3=National,
4=International
- **Numéro de sous-système (SSN) :** 6=HLR, 7=VLR, 8=MSC, 9=EIR,
etc.

**Exemple :**
```elixir
config :omniss7,
  m3ua_gt_routes: [
    # Routage de préfixe de base
    %{gt_prefix: "1234", peer_id: 1, priority: 1,
description: "Numéros US"},
    %{gt_prefix: "44", peer_id: 2, priority: 1, description:
"Numéros UK"},

    # Transformation du type de traduction
    %{
      gt_prefix: "61",
      peer_id: 3,
      priority: 1,
      description: "Numéros australiens : transformation TT 0→1",
      source_tt: 0,  # Correspondre à TT=0 (Inconnu)
      dest_tt: 1     # Transformer en TT=1 (International)
    },

    # Transformation NPI
    %{
      gt_prefix: "49",
      peer_id: 1,
      priority: 1,
      description: "Numéros allemands : conversion Mobile→ISDN
NPI",
      source_npi: 6,  # Correspondre à NPI=6 (Mobile/E.212)
      dest_npi: 1     # Transformer en NPI=1 (ISDN/E.164)
    },

    # Transformation combinée avec routage SSN
    %{
```
```

```
      gt_prefix: "86",
      source_ssn: 8,   # Correspondre à SSN=8 (MSC)
      peer_id: 3,
      dest_ssn: 6,     # Réécrire en SSN=6 (HLR)
      priority: 1,
      description: "Trafic chinois : Normalisation complète",
      source_tt: 0,
      dest_tt: 2,
      source_npi: 6,
      dest_npi: 1,
      source_nai: 4,
      dest_nai: 3
    &#125;,

    # Route par défaut/de secours
    %&#123;
      gt_prefix: "",
      peer_id: 1,
      priority: 99,
      description: "Route de secours par défaut"
    &#125;
  ]
```

---

# Paramètres du mode passerelle CAMEL

Configuration pour le mode passerelle CAMEL (protocole CAP).

## Drapeaux du mode CAMEL

Activer les fonctionnalités CAMEL/CAP (définir `cap_client_enabled: true` et `camelgw_mode_enabled: true` dans Drapeaux de mode opérationnel).

# Configuration du protocole CAP

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `cap_version` | Atome | `:v2` | Non | Version du protocole CAP : `:v1`, `:v2`, `:v3` ou `:v4` |
| `camel_gsmscf_gt_address` | Chaîne | (utilise le GT appelé) | Non | Titre global gsmSCF par défaut pour les réponses CAMEL |

**Mapping de la version CAP :**

- `:v1` → OID du contexte d'application : 0.4.0.0.1.0.50.0
- `:v2` → OID du contexte d'application : 0.4.0.0.1.0.50.1 (par défaut - le plus largement supporté)
- `:v3` → OID du contexte d'application : 0.4.0.0.1.21.3.4
- `:v4` → OID du contexte d'application : 0.4.0.0.1.23.3.4

**Remarque :** Les requêtes entrantes sont détectées automatiquement à partir de leur OID de contexte d'application et les réponses correspondent à la version de la requête.

**Exemple :**

```
config :omniss7,
  cap_client_enabled: true,
  camelgw_mode_enabled: true,
  cap_version: :v2,
  camel_gsmscf_gt_address: "99988411553"
```

# Intégration CGrateS

Intégration de facturation en temps réel avec CGrateS pour la facturation prépayée/postpayée.

| Paramètre | Type | Par défaut | Requis | Descrip |
|---|---|---|---|---|
| `cgrates_enabled` | Booléen | `false` | Non | Activer l'intégration CGrateS |
| `cgrates_url` | Chaîne | - | **Oui** (si activé) | URL du poin terminaison RPC de CGr |
| `cgrates_tenant` | Chaîne | `"cgrates.org"` | Non | Identifiant d locataire CG |
| `cgrates_request_type` | Chaîne | `"*prepaid"` | Non | Type de facturation `"*prepaid"` `"*postpaid` `"*pseudopr` |
| `cgrates_timeout` | Entier | `5000` | Non | Délai d'atte la requête CGrateS en millisecond |

**Exemple :**

```
config :omniss7,
  cgrates_enabled: true,
  cgrates_url: "http://localhost:2080/jsonrpc",
  cgrates_tenant: "cgrates.org",
  cgrates_request_type: "*prepaid",
  cgrates_timeout: 5000
```

## Connexion CAP M3UA

La passerelle CAMEL utilise une connexion M3UA séparée pour les opérations CAP.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `cap_client_m3ua` | Carte | - | **Oui** | Configuration de connexion CAP M3UA (même structure que `map_client_m3ua`, y compris les paramètres `opc` et `dpc` optionnels) |

**Exemple :**

```
config :omniss7,
  cap_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;CapClient, :handle_payload, []&#125;,
    process_name: :camelgw_client_asp,
    local_ip: &#123;10, 5, 198, 200&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 4,
    opc: 5013,    # Code de point d'origine
    dpc: 5011     # Code de point de destination
  &#125;
```

---

# Paramètres de NAT de titre global

La traduction d'adresse réseau de titre global permet différents GT de réponse en fonction du préfixe de l'appelant, du préfixe du appelé, ou des deux. Les règles sont appariées par poids (plus bas = plus haute priorité), puis par

spécificité de préfixe (préfixe combiné plus long = plus spécifique). Pour une explication détaillée et des exemples, voir le Guide de NAT de titre global.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `gt_nat_enabled` | Booléen | `false` | Non | Activer/désactiver la fonctionnalité NAT GT |
| `gt_nat_rules` | Liste de cartes | `[]` | **Oui** (si activé) | Liste des règles NAT GT avec correspondance de préfixe |

**Format de règle :** Chaque règle dans `gt_nat_rules` doit être une carte avec :

- `calling_prefix` : Préfixe de chaîne à faire correspondre avec le GT appelant (optionnel)
- `called_prefix` : Préfixe de chaîne à faire correspondre avec le GT appelé (optionnel)
- `weight` : Priorité entière (plus bas = plus haute priorité) - par défaut : 100
- `response_gt` : Titre global à utiliser dans les réponses (requis)

**Priorité de correspondance :**

1. Les règles sont appariées par `weight` (valeur plus basse = priorité plus élevée)
2. Si les poids sont égaux, la longueur de préfixe combiné plus longue l'emporte
3. Les `calling_prefix` et `called_prefix` peuvent être utilisés ensemble pour un appariement précis

**Exemple :**

```
config :omniss7,
  gt_nat_enabled: true,
  gt_nat_rules: [
    # Haute priorité : Correspondre à la fois l'appelant de "8772"
ET le appelé à "555"
    %&#123;calling_prefix: "8772", called_prefix: "555", weight:
1, response_gt: "111111"&#125;,

    # Priorité moyenne : Correspondre uniquement l'appelant de
"8772"
    %&#123;calling_prefix: "8772", weight: 10, response_gt:
"99988411553"&#125;,

    # Priorité moyenne : Correspondre uniquement le appelé à "555"
    %&#123;called_prefix: "555", weight: 10, response_gt:
"99988411554"&#125;,

    # Correspondre uniquement l'appelant de "8773"
    %&#123;calling_prefix: "8773", weight: 10, response_gt:
"99988411554"&#125;,

    # Règle de secours générique (correspond à tout, poids le plus
élevé)
    %&#123;weight: 100, response_gt: "99988411555"&#125;
  ]
```

**Voir aussi** : Guide de NAT GT pour une utilisation détaillée et des exemples.

---

# Paramètres de connexion M3UA

Configuration de connexion M3UA pour le mode client MAP. Pour une utilisation détaillée et des exemples, voir le Guide du client MAP.

| Paramètre | Type | Par défaut | Requis | |
|---|---|---|---|---|
| `map_client_m3ua.mode` | Chaîne | - | **Oui** | |
| `map_client_m3ua.callback` | Tuple | - | **Oui** | |
| `map_client_m3ua.process_name` | Atome | - | **Oui** | |
| `map_client_m3ua.local_ip` | Tuple ou Liste | - | **Oui** | |
| `map_client_m3ua.local_port` | Entier | `2905` | **Oui** | |
| `map_client_m3ua.remote_ip` | Tuple ou Liste | - | **Oui** | |
| `map_client_m3ua.remote_port` | Entier | `2905` | **Oui** | |
| `map_client_m3ua.routing_context` | Entier | - | **Oui** | |
| `map_client_m3ua.opc` | Entier | `5013` | Non | |

| Paramètre | Type | Par défaut | Requis | |
|---|---|---|---|---|
| | | | | |
| `map_client_m3ua.dpc` | Entier | `5011` | Non | |
| `map_client_m3ua.receive_watchdog` | Booléen | `true` | Non | |
| `map_client_m3ua.receive_watchdog_idle` | Entier | `15` | Non | |

**Exemple (IP unique)** :

```
config :omniss7,
  map_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;MapClient, :handle_payload, []&#125;,
    process_name: :hlr_client_asp,
    local_ip: &#123;10, 179, 4, 11&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 1,
    opc: 5013,    # Code de point d'origine (2-114-5)
    dpc: 5011     # Code de point de destination (2-114-3)
  &#125;
```

> **Format du code de point** : Les codes de point au format `X-Y-Z` se convertissent en entiers comme : `(X * 2048) + (Y * 8) + Z`. Par exemple, `2-114-5` = `(2 * 2048) + (114 * 8) + 5` = `5013`.

**Exemple (multihoming SCTP)** :

```
config :omniss7,
  map_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;MapClient, :handle_payload, []&#125;,
    process_name: :hlr_client_asp,
    # Plusieurs IP locales pour la redondance
    local_ip: [&#123;10, 179, 4, 11&#125;, &#123;10, 179, 4,
12&#125;],
    local_port: 2905,
    # Plusieurs IP distantes pour la redondance STP
    remote_ip: [&#123;10, 179, 4, 10&#125;, &#123;10, 179, 4,
20&#125;],
    remote_port: 2905,
    routing_context: 1
  &#125;
```

> **Remarque** : Pour des informations détaillées sur la configuration et les avantages du multihoming SCTP, voir Multihoming SCTP dans le Guide

> commun.

## Receive Watchdog

Le **receive watchdog** surveille les connexions SCTP pour les sockets zombies. Un socket zombie est une association qui reste dans un état ÉTABLI au niveau du système d'exploitation, mais où l'extrémité distante a silencieusement cessé d'envoyer des données. Sans le watchdog, le nœud peut ne pas détecter une connexion morte avant que la prochaine tentative d'envoi échoue.

Toute charge utile SCTP reçue réinitialise le minuteur d'inactivité, y compris les réponses M3UA BEAT, les messages NOTIFY et les données d'application. Selon RFC 4666 §3.8, M3UA BEAT est optionnel. SCTP effectue déjà un heartbeat obligatoire au niveau de la couche de transport. Lorsque le SG distant n'envoie pas de trafic périodique de couche application (et n'envoie pas de M3UA BEATs), désactivez le watchdog pour éviter des cycles de reconnexion inutiles.

| Paramètre | Type | Par défaut | Description |
| --- | --- | --- | --- |
| `receive_watchdog` | Booléen | `true` | Activer ou désactiver le receive watchdog. Lorsque `false`, les connexions inactives ne sont jamais interrompues par le watchdog ; le heartbeat de la couche de transport SCTP fonctionne toujours normalement. |
| `receive_watchdog_idle` | Entier | `15` | Secondes d'inactivité avant que le watchdog n'interrompe la connexion et déclenche une reconnexion. N'a aucun effet lorsque `receive_watchdog: false`. |

**Exemple : désactiver le watchdog pour un client ASP :**

```
config :omniss7,
  map_client_m3ua: %&#123;
    mode: "ASP",
    local_ip: &#123;10, 179, 4, 11&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 1,
    receive_watchdog: false        # Désactiver - le SG distant
n'envoie pas de trafic périodique
  &#125;
```

**Exemple : personnaliser le seuil d'inactivité :**

```
config :omniss7,
  map_client_m3ua: %&#123;
    mode: "ASP",
    local_ip: &#123;10, 179, 4, 11&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 1,
    receive_watchdog: true,
    receive_watchdog_idle: 30      # Interrompre après 30 s de
silence (par défaut : 15 s)
  &#125;
```

---

# État du sous-système SCCP

OmniSS7 répond aux tests de statut de sous-système de gestion SCCP (SST) à partir du statut réel du sous-système local interrogé (ITU-T Q.714 §5.3.4). La clé `local_subsystems` déclare quels numéros de sous-système (SSN) ce nœud héberge réellement. Un SST pour un SSN répertorié est répondu par `Subsystem-Allowed` (SSA) ; tout autre SSN est répondu par `Subsystem-Prohibited` (SSP).

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `local_subsystems` | Liste d'entiers ou `:all` | `:all` | Non | SSN SCCP locaux en service utilisés pour répondre aux SST. `:all` (par défaut) considère chaque sous-système comme disponible. Cela convient à un relais pur qui n'héberge aucun sous-système. Les opérateurs qui terminent des SSN spécifiques doivent les lister afin que SST reflète la réalité. |

**SSN communs :** 6 = HLR, 7 = VLR, 8 = MSC, 9 = EIR, 146 = gsmSCF/CAMEL.

**Exemple :**

```
config :omniss7,
  local_subsystems: [6, 7, 8, 146]
```

---

# Filtrage de passerelle

Le filtrage de passerelle (le pare-feu SCCP) permet ou refuse le trafic de transit avant qu'il ne soit routé. Les règles correspondent à la source du pair, OPC, DPC, indicateur de service, préfixe GTA de l'appelant/du appelé SCCP et SSN ; la première règle activée correspondante (priorité la plus basse) décide de `allow`/`deny`. Lorsque aucune règle ne correspond, l'action par défaut ci-

dessous s'applique. Les messages refusés sont supprimés et comptés comme `m3ua_stp_routing_failures_total{reason="screened"}`. Voir le Guide STP pour le modèle complet.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `screening_default_action` | `:allow` ou `:deny` | `:allow` | Non | Action appliquée à un message de transit lorsque aucune règle de filtrage ne correspond. `:allow` (par défaut) est une posture de permis par défaut ; `:deny` est un refus par défaut (liste blanche). Définir sur `:deny` rend le filtrage actif même sans règles configurées. |

**Exemple :**

```
config :omniss7,
  # Refus par défaut : seul le trafic correspondant à une règle
d'autorisation explicite est transféré.
  screening_default_action: :deny
```

Les règles elles-mêmes sont conservées dans la table Mnesia `ss7_screen_rule` et gérées par programme ; seule l'action par défaut est définie via la configuration.

---

# Paramètres de passerelle USSD

La passerelle USSD relie l'USSD initié par le réseau et l'USSD initié par le mobile à un backend HTTP/JSON. Activez-la avec `ussd_gateway_enabled: true` (voir Drapeaux de mode opérationnel), puis configurez la carte `ussd_gateway`. Pour le protocole de rappel complet et le cycle de vie de la session, voir le Guide de la passerelle USSD.

| Paramètre | Type | Par défaut | Requis | Descr |
|---|---|---|---|---|
| `ussd_gateway.routes` | Liste de cartes | `[]` | **Oui** (si activé) | Règles routage shortco URL de Chaque est `%{pa` `"<prefi` `url: "` `<callba` `URL>"}`. **préfixe long ga** utilisez motifs c style `"*` comme solution secours |
| `ussd_gateway.session_timeout_ms` | Entier | `180_000` | Non | Durée d totale m de la se millisec minutes session terminé est dép |
| `ussd_gateway.turn_timeout_ms` | Entier | `30_000` | Non | Temps maximu à attend une rép d'abonn |

| Paramètre | Type | Par défaut | Requis | Descr |
|---|---|---|---|---|
| | | | | un dialo plusieur |
| `ussd_gateway.http_timeout_ms` | Entier | `5_000` | Non | Délai d' de la requête pour les à votre applicat (ms). |
| `ussd_gateway.max_text_length` | Entier | `182` | Non | Nombre maximu caractè une cha texte US (GSM 7 Les text longs so tronqué avertiss est enre |

**Routage :** Lorsqu'un code USSD arrive, la passerelle trie les routes configurées par longueur de `pattern` décroissante. La passerelle sélectionne ensuite la première route dont le `pattern` est un préfixe du code composé. Cela donne un appariement déterministe du préfixe le plus long.

Lexical error on line 2. Unrecognized text. ...x. *100*5#"] --> B&fl° °123¶ßCorrespondre ----------^

Réessayer

**Exemple :**

```
config :omniss7,
  ussd_gateway_enabled: true,
  ussd_gateway: %&#123;
    routes: [
      %&#123;pattern: "*100", url: "http://balance-
app:9000/ussd"&#125;,
      %&#123;pattern: "*200", url: "http://topup-
app:9000/ussd"&#125;,
      %&#123;pattern: "*",    url: "http://default-
app:9000/ussd"&#125;
    ],
    session_timeout_ms: 180_000,
    turn_timeout_ms: 30_000,
    http_timeout_ms: 5_000,
    max_text_length: 182
  &#125;
```

---

# Drapeaux d'exécution / de supervision

Ces drapeaux contrôlent combien de l'arbre de supervision démarre. Ils sont utiles pour exécuter OmniSS7 sans interface (comme une d��pendance de bibliothèque) ou pour supprimer la couche web dans des déploiements contraints.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `start_web_interfaces` | Booléen | `true` | Non | Démarrer l'interfac utilisateur du panr de contrôle, le serv REST `:api_ex` et l journal des événer SS7. Lorsque `fals` seul le journal des paquets (`SS7.Ever` démarre. Peut être remplacé au déma par la variable d'environnement `ENABLE_WEB_INTER` (`true`/`1` ou `false` |
| `start_http_server` | Booléen | `true` | Non | Démarrer le serveu HTTP Plug.Cowboy le port `api_port` ( `/api/*`, `/metrics` `/swagger`). Voir Se API REST. |
| `api_port` | Entier | `9090` | Non | Port TCP pour le se HTTP Plug.Cowboy (REST `/api/*`, `/metrics`, `/swagg` Mettez à jour la cib récupération Prometheus si vou changez. |
| `headless` | Booléen | `false` | Non | Supervision minim uniquement pour l bibliothèque. Lorsc |

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| | | | | `true`, OmniSS7 démarre **uniquem** les superviseurs d codec/transaction (registres MAP, CA USSD et supervise dynamiques) : pas d'interface web, pa client de licence, p gestionnaire SCTP, de serveur M3UA. Destiné à intégrer codecs OmniSS7 comme dépendan |
| `vlr_cache_ttl_seconds` | Entier | `3600` | Non | Durée de vie (seco pour les recherche mises en cache da chemin de livraiso SMSc. |

**Exemple (mode bibliothèque sans interface) :**

```
config :omniss7,
  headless: true
```

**Exemple (pas de couche web, uniquement API REST) :**

```
config :omniss7,
  start_web_interfaces: false,
  start_http_server: true
```

---

# Paramètres d'infrastructure

Configuration pour les composants d'infrastructure système, y compris la licence, l'interface web, le serveur API et la journalisation.

## Configuration de la licence

| Paramètre | Type | Par défaut | Requis |
|---|---|---|---|
| `license_client.license_server_api_urls` | Liste de chaînes | - | **Oui** |
| `license_client.licensee` | Chaîne | - | **Oui** |

**Exemple :**

```
config :license_client,
  license_server_api_urls: ["https://localhost:10443/api"],
  licensee: "Omnitouch Network Services Pty. Ltd."
```

## Interface web du panneau de contrôle

| Paramètre | Type | Pa |
|---|---|---|
| `control_panel.parent_application_readable_name` | Chaîne | `"OmniSS7 S` |
| `control_panel.use_additional_pages` | Liste de tuples | `[]` |
| `control_panel.page_order` | Liste de chaînes | `[]` |
| `ControlPanelWeb.Endpoint.url.host` | Chaîne | `"0.0.0.0"` |
| `ControlPanelWeb.Endpoint.https.port` | Entier | `8087` |
| `ControlPanelWeb.Endpoint.https.keyfile` | Chaîne | `"priv/cert` |
| `ControlPanelWeb.Endpoint.https.certfile` | Chaîne | `"priv/cert` |

**Exemple :**

```
config :control_panel, ControlPanelWeb.Endpoint,
  url: [host: "0.0.0.0", path: "/"],
  https: [
    port: 8087,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ],
  parent_application_readable_name: "OmniSS7 Stack STP"

config :control_panel,
  use_additional_pages: [
    &#123;SS7.Web.EventsLive, "/events", "Événements SS7"&#125;,
    &#123;SS7.Web.M3UAStatusLive, "/m3ua", "Pairs"&#125;
  ],
  page_order: ["/events", "/m3ua", "/application",
"/configuration"]
```

## Serveur API REST

| Paramètre | Type | Par défaut | Req |
|---|---|---|---|
| `start_http_server` | Booléen | `true` | Non |
| `api_ex.api.port` | Entier | `8445` | Non |
| `api_ex.api.listen_ip` | Chaîne | `"0.0.0.0"` | Non |
| `api_ex.api.product_name` | Chaîne | `"OmniSS7"` | Non |
| `api_ex.api.title` | Chaîne | `"API - OmniSS7"` | Non |
| `api_ex.api.hostname` | Chaîne | `"localhost"` | Non |
| `api_ex.api.enable_tls` | Booléen | `true` | Non |
| `api_ex.api.tls_cert_path` | Chaîne | `"priv/cert/omnitouch.crt"` | Non |
| `api_ex.api.tls_key_path` | Chaîne | `"priv/cert/omnitouch.pem"` | Non |

**Exemple :**

```
config :omniss7,
  start_http_server: true

config :api_ex,
  api: %&#123;
    port: 8445,
    listen_ip: "0.0.0.0",
    product_name: "OmniSS7",
    title: "API - OmniSS7",
    hostname: "localhost",
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  &#125;
```

**Ports et transports en un coup d'œil :**

| Port | Transport | Serveur | But |
|---|---|---|---|
| 9090 | HTTP (Plug.Cowboy) | `APIhandler` | MAP/CAP REST `/api`, `/metrics`, `/swagger.json`. Act par `start_http_ser` |
| 8087 | HTTPS (Bandit/Phoenix) | `ControlPanelWeb.Endpoint` | Interface utilisateur panneau de contrôl Web. Partie de la co Web (`start_web_interf` |
| 8445 | HTTPS / TLS | `:api_ex` contrôleurs REST | Statut, CRUD de rou liens, abonnés, vue CAMEL/SCTP/M3UA. Partie de la couche |

**Points de terminaison API :**

- API REST MAP/CAP + Swagger : `http://[server-ip]:9090/...` (voir Guide API)
- Contrôleurs REST `:api_ex` : `https://[server-ip]:8445/api/*`
- Panneau de contrôle Web : `https://[server-ip]:8087/`
- Métriques Prometheus : `http://[server-ip]:9090/metrics`

## Vérification SSL du Backend API

Les appels sortants vers les API HLR/SMSc du backend **n'utilisent pas** de clés `*_verify_ssl` par cible. La vérification des certificats est contrôlée par la carte générique `:api` :

| Paramètre | Type | Par défaut | Description |
|---|---|---|---|
| `api.base_url` | Chaîne | revient à `smsc_api_base_url` | URL de base pour le client API générique. |
| `api.timeout` | Entier | `5000` | Délai d'attente de la requête en millisecondes. |
| `api.verify_ssl` | Booléen | `false` | Indique si les certificats TLS du backend doivent être vérifiés. **C'est le seul interrupteur de vérification SSL.** Il n'y a pas de `hlr_api_verify_ssl` ou `smsc_api_verify_ssl`. |

**Exemple :**

```
config :omniss7,
  api: %&#123;
    base_url: "https://backend.example.com:8443",
    timeout: 5000,
    verify_ssl: false
  &#125;
```

## Configuration des journaux

| Paramètre | Type | Par défaut | Requ |
|---|---|---|---|
| `logger.level` | Atome | `:debug` | Non |
| `logger.backends` | Liste | `[:console]` | Non |
| `logger.default_formatter.format` | Chaîne | - | Non |
| `logger.default_formatter.metadata` | Liste | `[]` | Non |
| `logger.default_formatter.truncate` | Atome/Entier | `:infinity` | Non |

**Exemple :**

```
config :logger,
  level: :debug,
  backends: [:console, SS7.Web.LoggerBackend]

config :logger, :default_formatter,
  format: "[$date] [$time] [$level] $message\n",
  metadata: [:error_code, :file],
  truncate: :infinity
```

---

# Paramètres de base de données

Configuration pour la persistance de la base de données Mnesia.

| Paramètre | Type | Par défaut | Requis | Description |
|---|---|---|---|---|
| `mnesia_storage_type` | Atome | `:disc_copies` | Non | Type de stockage Mnesia : `:disc_copies` ou `:ram_copies` |

**Exemple** :

```
config :omniss7,
  mnesia_storage_type: :disc_copies  # Production
  # mnesia_storage_type: :ram_copies  # Test uniquement
```

**Types de stockage** :

- `:disc_copies` - Stockage persistant sur disque (survit aux redémarrages) - **Recommandé pour la production**
- `:ram_copies` - Seulement en mémoire (perdu lors du redémarrage) - Pour test uniquement

**Tables Mnesia** :

- `m3ua_peer` - Connexions de pairs M3UA
- `m3ua_route` - Routes de code de point
- `m3ua_gt_route` - Routes de titre global

**Emplacement** : répertoire `Mnesia.{node_name}/`

---

# Valeurs codées en dur

Les valeurs suivantes sont **codées en dur dans le code source** et ne peuvent pas être modifiées via la configuration.

## Délais d'attente

| Valeur | Impact | Contournement |
|---|---|---|
| Délai d'attente de la requête MAP : **10 secondes** | Toutes les opérations MAP expirent après 10s | Modifier le code source |
| Délai d'attente ISD : **10 secondes** | Chaque message ISD expire après 10s | Modifier le code source |

## Serveur HTTP

| Valeur | Impact | Contournement |
|---|---|---|
| IP HTTP : **0.0.0.0** | Le serveur de métriques/Swagger écoute sur toutes les interfaces | Modifier le code source |
| Port HTTP : **9090** | Le point de terminaison de métriques/Swagger fonctionne sur le port 9090 | Définir `config :omniss7, api_port: <port>` (par défaut `9090`) |

## Intervalles d'enregistrement

| Valeur | Impact | Contournement |
|---|---|---|
| Enregistrement frontal : **5 minutes** | SMSc s'enregistre avec le backend toutes les 5 min | Modifier le code source |

## Actualisation automatique de l'interface Web

| Page | Intervalle |
|---|---|
| Gestion du routage | 5 secondes |
| Abonnés actifs | 2 secondes |

---

# Exemples de configuration

## Configuration HLR minimale

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false,

  hlr_api_base_url: "https://omnihss.example.com:8443",
  hlr_api_key: "&lt;Clé API OmniHSS avec le rôle
hlr_interworking&gt;",
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112",

  map_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;MapClient, :handle_payload, []&#125;,
    process_name: :hlr_client_asp,
    local_ip: &#123;10, 179, 4, 11&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 1
  &#125;
```

## Configuration SMSc minimale

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: true,

  smsc_api_base_url: "https://omnimessage.example.com:8443",
  smsc_name: "ipsmgw",
  smsc_service_center_gt_address: "55512341112",

  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10,

  map_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;MapClient, :handle_payload, []&#125;,
    process_name: :stp_client_asp,
    local_ip: &#123;10, 179, 4, 12&#125;,
    local_port: 2905,
    remote_ip: &#123;10, 179, 4, 10&#125;,
    remote_port: 2905,
    routing_context: 1
  &#125;
```

## STP avec serveur autonome

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: false,

  enable_gt_routing: true,
  mnesia_storage_type: :disc_copies,

  sctp_handler: %&#123;
    enabled: true,
    local_ip: &#123;10, 179, 4, 10&#125;,
    local_port: 2905,
    point_code: 100
  &#125;,

  map_client_m3ua: %&#123;
    mode: "ASP",
    callback: &#123;MapClient, :handle_payload, []&#125;,
    process_name: :stp_client_asp,
    local_ip: &#123;10, 179, 4, 10&#125;,
    local_port: 2906,
    remote_ip: &#123;10, 179, 4, 11&#125;,
    remote_port: 2905,
    routing_context: 1
  &#125;
```

---

# Résumé

## Total des paramètres de configuration : 85+

**Par catégorie** :

- Mode opérationnel : 6 paramètres
- Mode HLR : 18 paramètres
- Mode SMSc : 8 paramètres

- Mode STP : 5+ paramètres (plus les listes m3ua_peers, m3ua_routes, m3ua_gt_routes)
- Mode CAMEL GW : 14 paramètres
- NAT de titre global : 2 paramètres
- Connexion M3UA : 8 paramètres
- État du sous-système SCCP : 1 paramètre
- Passerelle USSD : 5 paramètres
- Runtime / Supervision : 4 paramètres
- Infrastructure (Licence, Web, API, Journalisation) : 23 paramètres
- Base de données : 1 paramètre

**Paramètres requis par mode** :

**Mode HLR :**

- `hlr_api_base_url`
- `hlr_api_key`
- `hlr_service_center_gt_address`
- `smsc_service_center_gt_address`
- Tous les paramètres `map_client_m3ua.*` (8)

**Mode SMSc :**

- `smsc_api_base_url`
- `smsc_service_center_gt_address`
- `auto_flush_dest_smsc` (si auto-flush activé)
- Tous les paramètres `map_client_m3ua.*` (8)

**Mode STP :**

- `sctp_handler.point_code` (si le gestionnaire SCTP est activé)
- `sctp_handler.local_ip`
- `sctp_handler.local_port`

**Mode CAMEL GW :**

- `cgrates_url` (si CGrateS activé)

- Tous les paramètres `cap_client_m3ua.*` (8)

**Infrastructure :**

- `license_client.license_server_api_urls`
- `license_client.licensee`

---

# Documentation connexe

- **Guide HLR** - Configuration spécifique à HLR
- **Guide SMSc** - Configuration spécifique à SMSc
- **Guide STP** - Configuration de routage STP
- **Guide API** - Référence API REST
- **Guide Passerelle USSD** - Configuration de la passerelle USSD et protocole de rappel HTTP
- **Guide Interface Web** - Documentation de l'interface Web

# Guide de NAT de Titre Global

## Aperçu

La traduction d'adresse de titre global (GT NAT) est une fonctionnalité qui permet à OmniSS7 de répondre avec différentes adresses de titre global en fonction du préfixe GT de la partie appelante, du préfixe GT de la partie appelée, ou d'une combinaison des deux. Cela est essentiel lors de l'exploitation avec plusieurs titres globaux et en ayant besoin de s'assurer que les réponses utilisent le bon GT en fonction du réseau ou du pair qui appelle et/ou du GT qu'ils ont appelé.

## Quoi de Neuf (GT NAT Amélioré)

La fonctionnalité GT NAT a été améliorée avec de nouvelles capacités puissantes :

### Nouvelles Fonctionnalités

1. **Correspondance de Préfixe de Partie Appelée** : Les règles peuvent désormais correspondre sur `called_prefix` en plus de `calling_prefix`
2. **Correspondance Combinée** : Les règles peuvent correspondre à la fois sur les préfixes appelants ET appelés simultanément
3. **Priorisation Basée sur le Poids** : Les règles utilisent désormais un champ `weight` (plus bas = priorité plus élevée) au lieu de simplement la longueur du préfixe
4. **Correspondance Flexible** : Vous pouvez désormais créer des règles avec :
    - Seulement le préfixe appelant
    - Seulement le préfixe appelé
    - À la fois les préfixes appelants et appelés

- Aucun (règle de secours/wildcard)

## Nouveau Format de Règle

**Champs requis :**

- `weight` : Priorité entière (plus bas = priorité plus élevée)
- `response_gt` : Le GT à utiliser pour répondre

**Champs optionnels (au moins un recommandé pour une correspondance spécifique) :**

- `calling_prefix` : Correspondre sur le préfixe GT de la partie appelante
- `called_prefix` : Correspondre sur le préfixe GT de la partie appelée

**Exemple :**

```
gt_nat_rules: [
  # Règle spécifique avec les deux préfixes - priorité la plus
élevée
  %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

  # Règles spécifiques - priorité moyenne
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},
  %{called_prefix: "555", weight: 10, response_gt: "333333"},

  # Règle de secours wildcard - priorité la plus basse
  %{weight: 100, response_gt: "999999"}
]
```

# Cas d'Utilisation

## Opération Multi-Réseaux

Lorsque vous avez plusieurs réseaux pairs et que chacun attend des réponses d'un GT spécifique :

- **Réseau A** appelle votre GT `111111` et attend des réponses de `111111`
- **Réseau B** appelle votre GT `222222` et attend des réponses de `222222`

Sans GT NAT, vous auriez besoin d'instances séparées ou de routage complexe. Avec GT NAT, une seule instance OmniSS7 peut gérer cela intelligemment.

## Scénarios de Roaming

Lorsque vous opérez en tant qu'HLR ou SMSc avec des accords de roaming :

- Les abonnés du **réseau domestique** utilisent le GT `555000`
- Le **partenaire de roaming 1** utilise le GT `555001`
- Le **partenaire de roaming 2** utilise le GT `555002`

GT NAT garantit que chaque partenaire reçoit des réponses du bon GT auquel ils sont configurés pour router.

## Tests et Migration

Lors des migrations de réseau ou des tests :

- Migrer progressivement le trafic de l'ancien GT vers le nouveau GT
- Maintenir les deux GT pendant la période de transition
- Router les réponses en fonction du GT utilisé par l'appelant

# Comment Ça Marche

## Flux de Traduction d'Adresse

1. **Demande Entrante** : OmniSS7 reçoit un message SCCP avec :
    - GT de la Partie Appelée : `55512341112` (votre GT)
    - GT de la Partie Appelante : `877234567` (leur GT)
2. **Recherche GT NAT** : Le système vérifie le GT appelant `877234567` par rapport aux règles de préfixe configurées

3. **Correspondance de Préfixe** : Trouve le préfixe correspondant le plus long (par exemple, `8772` correspond à `877234567`)

4. **Sélection du GT de Réponse** : Utilise `response_gt` de la règle correspondante (par exemple, `55512341112`)

5. **Réponse Envoyée** : La réponse SCCP utilise :

    - GT de la Partie Appelée : `877234567` (inversé - leur GT)
    - GT de la Partie Appelante : `55512341112` (GT NAT)

## Types de Réponses Affectés

GT NAT s'applique à plusieurs couches de la pile SS7 :

#### Couche SCCP (Toutes les Réponses)

- Adresses GT appelées/appelantes SCCP dans tous les messages de réponse
- Accusés de réception ISD (InsertSubscriberData)
- Réponses UpdateLocation
- Réponses d'erreur

#### Couche MAP (Opération Spécifique)

- **Réponses SRI-for-SM** : `networkNode-Number` (adresse GT SMSc)
- **UpdateLocation** : `hlr-Number` dans les réponses
- **InsertSubscriberData** : GT HLR dans les messages ISD

# Configuration

## Configuration de Base

Ajoutez à `config/runtime.exs` :

```
config :omniss7,
  # Activer GT NAT
  gt_nat_enabled: true,

  # Définir les règles GT NAT
  gt_nat_rules: [
    # Règle 1 : Les appels du préfixe "8772" obtiennent une
réponse de "55512341112"
    %{calling_prefix: "8772", response_gt: "55512341112"},

    # Règle 2 : Les appels du préfixe "8773" obtiennent une
réponse de "55512341111"
    %{calling_prefix: "8773", response_gt: "55512341111"},

    # Règle par défaut (préfixe vide correspond à tout)
    %{calling_prefix: "", response_gt: "55512311555"}
  ]
```

## Paramètres de Configuration

Pour une référence complète de configuration, voir Paramètres de Titre Global NAT dans la Référence de Configuration.

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `gt_nat_enabled` | Boolean | Oui | Activer/désactiver la fonctionnalité GT NAT |
| `gt_nat_rules` | Liste de Maps | Oui (si activé) | Liste des règles de correspondance de préfixe |

## Format de Règle

Chaque règle est une map avec les clés suivantes :

```
%{
  calling_prefix: "8772",      # (Optionnel) Préfixe à
correspondre avec le GT appelant
  called_prefix: "555",        # (Optionnel) Préfixe à
correspondre avec le GT appelé
  weight: 10,                  # (Requis) Valeur de priorité (plus
bas = priorité plus élevée)
  response_gt: "55512341112"   # (Requis) GT à utiliser dans les
réponses
}
```

**Champs de Règle :**

- **`calling_prefix`** (Optionnel) : Préfixe de chaîne à correspondre avec le GT appelant entrant
  - La correspondance se fait comme une correspondance de préfixe de chaîne
  - Chaîne vide `""` ou `nil` agit comme un wildcard (correspond à tout GT appelant)
  - Peut être omis pour correspondre à tout GT appelant
- **`called_prefix`** (Optionnel) : Préfixe de chaîne à correspondre avec le GT appelé entrant
  - La correspondance se fait comme une correspondance de préfixe de chaîne
  - Chaîne vide `""` ou `nil` agit comme un wildcard (correspond à tout GT appelé)
  - Peut être omis pour correspondre à tout GT appelé
- **`weight`** (Requis) : Valeur de priorité entière
  - Poids plus bas = priorité plus élevée (traité en premier)
  - Doit être >= 0
  - Utilisé comme critère de tri principal pour les règles de correspondance
- **`response_gt`** (Requis) : L'adresse de Titre Global à utiliser dans les réponses

- Doit être une chaîne de numéro valide E.164
- Doit correspondre à l'un de vos GT configurés

**Au moins un des `calling_prefix` ou `called_prefix` doit être spécifié pour un routage spécifique. Les deux peuvent être omis pour une règle de secours/wildcard.**

# Logique de Correspondance de Règle

Les règles sont évaluées par **poids d'abord (croissant), puis par spécificité combinée de préfixe** :

**Algorithme de Correspondance :**

1. Filtrer les règles où tous les préfixes spécifiés correspondent
   - Si `calling_prefix` est défini, il doit correspondre au GT appelant
   - Si `called_prefix` est défini, il doit correspondre au GT appelé
   - Si les deux sont définis, les deux doivent correspondre
   - Si aucun n'est défini, la règle agit comme un wildcard
2. Trier les règles correspondantes par :
   - **Primaire** : Poids (croissant - valeurs plus basses en premier)
   - **Secondaire** : Longueur de préfixe combinée (décroissante - plus long = plus spécifique)
3. Retourner la première règle correspondante

**Exemples :**

```
# Exemples de règles
gt_nat_rules: [
  # Poids 1 : Priorité la plus élevée - correspond aux deux
préfixes
  %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

  # Poids 10 : Priorité moyenne - règles spécifiques
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},  #
Appelant seulement
  %{called_prefix: "555", weight: 10, response_gt: "333333"},    #
Appelé seulement

  # Poids 100 : Priorité la plus basse - wildcard fallback
  %{weight: 100, response_gt: "444444"}  # Correspond à tout
]

# Exemples de correspondance :
# Appelant : "877234567", Appelé : "555123" -> "111111" (poids 1,
les deux correspondent)
# Appelant : "877234567", Appelé : "999999" -> "222222" (poids 10,
appelant seulement)
# Appelant : "999999999", Appelé : "555123" -> "333333" (poids 10,
appelé seulement)
# Appelant : "999999999", Appelé : "888888" -> "444444" (poids
100, wildcard)
```

# Exemples

## Exemple 1 : Deux Partenaires Réseau

**Scénario** : Vous exploitez un SMSc avec deux partenaires réseau. Chacun attend des réponses d'un GT différent.

```
config :omniss7,
  gt_nat_enabled: true,

  # GT SMSc par défaut (utilisé lorsque GT NAT est désactivé ou
aucune règle ne correspond)
  smsc_service_center_gt_address: "5551000",

  # Règles GT NAT pour les partenaires
  gt_nat_rules: [
    # Partenaire A (préfixe 4412) attend des réponses du GT
5551001
    %{calling_prefix: "4412", weight: 10, response_gt: "5551001"},

    # Partenaire B (préfixe 4413) attend des réponses du GT
5551002
    %{calling_prefix: "4413", weight: 10, response_gt: "5551002"},

    # Par défaut : utiliser le GT SMSc standard (wildcard
fallback)
    %{weight: 100, response_gt: "5551000"}
  ]
```

**Flux de Trafic :**

```
SRI-for-SM entrant de 44121234567 :
  GT Appelé : 5551001 (votre GT que le Partenaire A utilise)
  GT Appelant : 44121234567 (GT du Partenaire A)

Recherche GT NAT :
  "44121234567" correspond au préfixe "4412"
  GT de réponse sélectionné : "5551001"

Réponse SRI-for-SM à 44121234567 :
  GT Appelé : 44121234567 (inversé)
  GT Appelant : 5551001 (NAT'd)
  networkNode-Number : 5551001 (dans la réponse MAP)
```

# Exemple 2 : HLR avec GTs Régionaux

**Scénario** : HLR national avec différents GT par région.

```
config :omniss7,
  gt_nat_enabled: true,
  hlr_service_center_gt_address: "555000",  # GT HLR par défaut

  gt_nat_rules: [
    # VLRs de la région nord (préfixe 5551)
    %{calling_prefix: "5551", weight: 10, response_gt: "555100"},

    # VLRs de la région sud (préfixe 5552)
    %{calling_prefix: "5552", weight: 10, response_gt: "555200"},

    # VLRs de la région ouest (préfixe 5553)
    %{calling_prefix: "5553", weight: 10, response_gt: "555300"},

    # Par défaut pour d'autres régions (wildcard)
    %{weight: 100, response_gt: "555000"}
  ]
```

## Exemple 3 : Scénario de Migration

**Scénario** : Migration progressive de l'ancien GT vers le nouveau GT.

```
config :omniss7,
  gt_nat_enabled: true,
  hlr_service_center_gt_address: "123456789",  # Ancien GT (par
défaut)

  gt_nat_rules: [
    # Réseaux migrés (ont déjà mis à jour leurs configurations)
    %{calling_prefix: "555", weight: 10, response_gt:
"987654321"},  # Nouveau GT
    %{calling_prefix: "666", weight: 10, response_gt:
"987654321"},  # Nouveau GT

    # Tout le monde utilise encore l'ancien GT (wildcard)
    %{weight: 100, response_gt: "123456789"}  # Ancien GT
  ]
```

# Exemple 4 : Correspondance de Préfixe de Partie Appelée (NOUVEAU)

**Scénario** : Vous avez plusieurs GT pour différents services, et vous souhaitez répondre avec le bon GT en fonction de celui qui a été appelé.

```
config :omniss7,
  gt_nat_enabled: true,

  gt_nat_rules: [
    # Lorsqu'ils appellent votre GT SMS (5551xxx), répondez avec
ce GT
    %{called_prefix: "5551", weight: 10, response_gt: "555100"},

    # Lorsqu'ils appellent votre GT Voix (5552xxx), répondez avec
ce GT
    %{called_prefix: "5552", weight: 10, response_gt: "555200"},

    # Lorsqu'ils appellent votre GT Données (5553xxx), répondez
avec ce GT
    %{called_prefix: "5553", weight: 10, response_gt: "555300"},

    # Règle de secours par défaut
    %{weight: 100, response_gt: "555000"}
  ]
```

**Flux de Trafic :**

```
Demande entrante au GT Appelé : 555100 (votre GT SMS)
GT Appelant : 441234567 (n'importe quel appelant)

Recherche GT NAT :
  GT Appelé "555100" correspond au préfixe "5551"
  GT de réponse sélectionné : "555100"

La réponse utilise le GT Appelant : 555100 (correspond à ce qu'ils
ont appelé)
```

# Exemple 5 : Correspondance Combinée de Préfixes Appelants + Appelés (AVANCÉE)

**Scénario** : Différents partenaires appellent différents GTs, et vous souhaitez un contrôle précis.

```
config :omniss7,
  gt_nat_enabled: true,

  gt_nat_rules: [
    # Partenaire A appelant votre GT SMS - priorité la plus élevée
(poids 1)
    %{calling_prefix: "4412", called_prefix: "5551", weight: 1,
response_gt: "555101"},

    # Partenaire B appelant votre GT SMS - priorité la plus élevée
(poids 1)
    %{calling_prefix: "4413", called_prefix: "5551", weight: 1,
response_gt: "555102"},

    # Quiconque appelant votre GT SMS - priorité moyenne (poids
10)
    %{called_prefix: "5551", weight: 10, response_gt: "555100"},

    # Partenaire A appelant n'importe quel GT - priorité moyenne
(poids 10)
    %{calling_prefix: "4412", weight: 10, response_gt: "555200"},

    # Règle de secours par défaut - faible priorité (poids 100)
    %{weight: 100, response_gt: "555000"}
  ]
```

**Exemples de Correspondance :**

```
# Partenaire A appelle le GT SMS
Appelant : "441234567", Appelé : "555100"
→ Correspond à la règle de poids 1 (les deux préfixes) → "555101"

# Partenaire A appelle le GT Voix
Appelant : "441234567", Appelé : "555200"
→ Correspond à la règle de poids 10 (appelant seulement) →
"555200"

# Appelant inconnu appelle le GT SMS
Appelant : "999999999", Appelé : "555100"
→ Correspond à la règle de poids 10 (appelé seulement) → "555100"

# Appelant inconnu appelle le GT Voix
Appelant : "999999999", Appelé : "555200"
→ Correspond à la règle de poids 100 wildcard → "555000"
```

# Modes Opérationnels

GT NAT fonctionne dans tous les modes opérationnels d'OmniSS7 :

## Mode HLR

GT NAT affecte :

- Réponses UpdateLocation (GT HLR dans la réponse)
- Messages InsertSubscriberData (GT HLR en tant que partie appelante)
- Réponses SendAuthenticationInfo
- Réponses Cancel Location

Pour plus d'informations sur les opérations HLR, voir le Guide de Configuration HLR.

**Configuration :**

```
config :omniss7,
  hlr_mode_enabled: true,
  hlr_service_center_gt_address: "5551234567",  # GT HLR par
défaut

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "331", weight: 10, response_gt:
"5551234568"},  # France
    %{calling_prefix: "44", weight: 10, response_gt:
"5551234569"},   # Royaume-Uni
    %{weight: 100, response_gt: "5551234567"}  # Wildcard par
défaut
  ]
```

## Mode SMSc

GT NAT affecte :

- Réponses SRI-for-SM (champ `networkNode-Number`) - voir Détails SRI-for-SM
- Accusés de réception MT-ForwardSM

Pour plus d'informations sur les opérations SMSc, voir le Guide de Configuration SMSc.

**Configuration :**

```
config :omniss7,
  smsc_mode_enabled: true,
  smsc_service_center_gt_address: "5559999",  # GT SMSc par défaut

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "1", weight: 10, response_gt: "5559991"},
# Amérique du Nord
    %{calling_prefix: "44", weight: 10, response_gt: "5559992"},
# Royaume-Uni
    %{calling_prefix: "86", weight: 10, response_gt: "5559993"},
# Chine
    %{weight: 100, response_gt: "5559999"}  # Wildcard par défaut
  ]
```

# Mode Passerelle CAMEL

GT NAT affecte :

- Toutes les réponses au niveau SCCP (GT gsmSCF en tant que Partie Appelante)
- Réponses d'opération CAMEL/CAP (InitialDP, EventReportBCSM, etc.)
- Accusés de réception RequestReportBCSMEvent
- Réponses ApplyCharging
- Réponses Continue

**Configuration :**

```
config :omniss7,
  camelgw_mode_enabled: true,
  camel_gsmscf_gt_address: "55512341112",  # GT gsmSCF par défaut

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "555", weight: 10, response_gt:
"55512341111"},  # Réseau A
    %{calling_prefix: "666", weight: 10, response_gt:
"55512311555"},  # Réseau B
    %{weight: 100, response_gt: "55512341112"}  # Wildcard par
défaut
  ]
```

**Cas d'Utilisation :** Lorsque vous opérez en tant que gsmSCF (Fonction de Contrôle de Service) pour plusieurs réseaux, chaque gsmSSF d'un réseau peut attendre des réponses d'un GT gsmSCF spécifique. GT NAT garantit que le bon GT est utilisé en fonction de quel gsmSSF appelle.

# Journalisation et Débogage

## Activer la Journalisation GT NAT

GT NAT inclut une journalisation automatique de toutes les traductions :

```
# Dans les journaux, vous verrez :
[info] GT NAT [réponse SRI-for-SM] : GT Appelant 877234567 -> GT
de Réponse 55512341112
[info] GT NAT [UpdateLocation ISD] : GT Appelant 331234567 -> GT
de Réponse 55512341111
[info] GT NAT [réponse MAP BEGIN] : GT Appelant 441234567 -> GT de
Réponse 55512311555
```

Le champ de contexte montre où le NAT a été appliqué :

- `"réponse SRI-for-SM"` - Dans le gestionnaire SRI-for-SM
- `"UpdateLocation ISD"` - Dans les messages InsertSubscriberData

- `"UpdateLocation END"` - Dans la réponse UpdateLocation END
- `"réponse MAP BEGIN"` - Réponses MAP BEGIN génériques
- `"ISD ACK"` - Accusé de réception ISD
- `"réponse d'erreur HLR"` - Réponse d'erreur de l'HLR
- `"réponse CAMEL"` - Réponses d'opération CAMEL/CAP (gsmSCF)

## Validation

Le système valide la configuration GT NAT au démarrage :

```
# Vérifier la config GT NAT
iex> GtNat.validate_config()
{:ok, [
  %{calling_prefix: "8772", weight: 10, response_gt:
"55512341112"},
  %{calling_prefix: "8773", weight: 10, response_gt:
"55512341111"}
]}

# Vérifier si activé
iex> GtNat.enabled?()
true

# Obtenir toutes les règles
iex> GtNat.get_rules()
[
  %{calling_prefix: "8772", weight: 10, response_gt:
"55512341112"},
  %{calling_prefix: "8773", weight: 10, response_gt:
"55512341111"}
]
```

## Tester GT NAT

Testez la logique GT NAT de manière programmatique :

```
# Tester la traduction avec seulement le GT appelant (called_gt
est nil)
iex> GtNat.translate_response_gt("877234567", nil, "default_gt")
"55512341112"

# Tester la traduction avec les GT appelant et appelé
iex> GtNat.translate_response_gt("877234567", "555123",
"default_gt")
"55512341112"

# Tester avec journalisation (GT appelé nil)
iex> GtNat.translate_response_gt_with_logging("877234567", nil,
"default_gt", "test")
# Journaux : GT NAT [test] : GT Appelant 877234567 -> GT de
Réponse 55512341112
"55512341112"

# Tester avec journalisation (les deux GT)
iex> GtNat.translate_response_gt_with_logging("877234567",
"555123", "default_gt", "test")
# Journaux : GT NAT [test] : GT Appelant 877234567, GT Appelé
555123 -> GT de Réponse 55512341112
"55512341112"

# Tester sans correspondance (retourne par défaut)
iex> GtNat.translate_response_gt("999999999", "888888",
"default_gt")
"default_gt"
```

# Dépannage

## Problème : GT NAT Ne Fonctionne Pas

#### Vérification 1 : Est-il activé ?

```
iex> Application.get_env(:omniss7, :gt_nat_enabled)
true  # Doit être vrai
```

**Vérification 2 : Les règles sont-elles configurées ?**

```
iex> Application.get_env(:omniss7, :gt_nat_rules)
[%{calling_prefix: "8772", response_gt: "55512341112"}, ...]  #
Doit retourner une liste
```

**Vérification 3 : Vérifiez les journaux** Recherchez "GT NAT" dans les journaux pour voir si des traductions se produisent.

# Problème : Mauvais GT dans les Réponses

**Symptôme** : Les réponses utilisent une adresse GT inattendue

**Cause** : La correspondance de préfixe de règle pourrait être trop large ou la règle par défaut capte le trafic

**Solution** : Révisez les poids et préfixes des règles :

```
# MAUVAIS : Wildcard avec un poids bas (attrape tout en premier)
gt_nat_rules: [
  %{weight: 1, response_gt: "111111"},              # Cela
correspond à tout en premier !
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"}  #
Jamais atteint
]

# BON : Règles spécifiques avec un poids plus bas, wildcard avec
un poids plus élevé
gt_nat_rules: [
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},  #
Spécifique, poids bas
  %{weight: 100, response_gt: "111111"}  # Wildcard, poids élevé
(fallback)
]
```

## Problème : GT NAT Non Appliqué à un Type de Message Spécifique

**Symptôme** : Certaines réponses utilisent un GT NAT'd, d'autres non

**Couverture Actuelle :**

- □ GT SCCP Appelant (toutes les réponses)
- □ Réponses SRI-for-SM (networkNode-Number)
- □ Messages ISD UpdateLocation (GT HLR)
- □ Réponses UpdateLocation END
- □ Accusés de réception ISD
- □ Réponses MAP BEGIN

Si un type de message spécifique n'utilise pas GT NAT, il se peut qu'il ne soit pas encore implémenté. Vérifiez le code source ou contactez le support.

# Considérations de Performance

## Performance de Recherche

GT NAT utilise une correspondance de préfixe simple avec une complexité O(n) où n est le nombre de règles.

**Conseils de performance :**

- Gardez le nombre de règles en dessous de 100 pour de meilleures performances
- Utilisez des préfixes spécifiques pour réduire le nombre de règles
- La règle par défaut (préfixe vide) doit être la dernière

**Benchmark (système typique) :**

- 10 règles : < 1µs par recherche
- 50 règles : < 5µs par recherche
- 100 règles : < 10µs par recherche

## Utilisation de Mémoire

Chaque règle nécessite environ 100 octets de mémoire :

- 10 règles ≈ 1 Ko
- 100 règles ≈ 10 Ko

# Meilleures Pratiques

## 1. Inclure Toujours une Règle de Secours Wildcard

```
gt_nat_rules: [
  %{calling_prefix: "8772", weight: 10, response_gt: "111111"},
  %{calling_prefix: "8773", weight: 10, response_gt: "222222"},
  %{weight: 100, response_gt: "default_gt"}  # Avoir toujours un
wildcard avec un poids élevé
]
```

## 2. Utiliser des Préfixes et Poids Significatifs

```
# BON : Préfixes clairs et spécifiques avec des poids appropriés
%{calling_prefix: "331", weight: 10, response_gt: "..."}  # France
%{calling_prefix: "44", weight: 10, response_gt: "..."}   #
Royaume-Uni

# MAUVAIS : Préfixes trop larges ou poids déroutants
%{calling_prefix: "3", weight: 5, response_gt: "..."}     # Trop
de pays
%{calling_prefix: "331", weight: 100, response_gt: "..."}  # Le
poids devrait être plus bas pour les règles spécifiques
```

## 3. Documenter Vos Règles

```
gt_nat_rules: [
  # Partenaire XYZ - réseau du Royaume-Uni (plage GT :
4412xxxxxxx)
  # Poids 10 : Priorité standard du partenaire
  %{calling_prefix: "4412", weight: 10, response_gt: "5551001"},

  # Partenaire ABC - réseau de France (plage GT : 33123xxxxxx)
  # Poids 10 : Priorité standard du partenaire
  %{calling_prefix: "33123", weight: 10, response_gt: "5551002"}
]
```

## 4. Tester Avant Déploiement

```
# Tester dans iex avant de déployer
iex> GtNat.translate_response_gt("44121234567", nil, "default")
"5551001"  # Résultat attendu

# Tester avec GT appelé
iex> GtNat.translate_response_gt("44121234567", "555123",
"default")
"5551001"  # Résultat attendu
```

## 5. Surveiller les Journaux

Activez la journalisation au niveau INFO pour voir toutes les traductions GT NAT en production.

# Intégration avec D'autres Fonctionnalités

## Mode STP

GT NAT fonctionne indépendamment du routage STP. STP route en fonction des codes de point et des GT de destination, tandis que GT NAT gère l'adressage

des réponses.

Pour plus d'informations sur le routage STP, voir le Guide de Configuration STP.

# Intégration CAMEL

GT NAT est **entièrement intégré** aux opérations CAMEL/CAP :

**Couche SCCP :**

- GT de la Partie Appelante dans toutes les réponses CAMEL
- Appliqué automatiquement en fonction du GT gsmSSF entrant

**Configuration :**

- `camel_gsmscf_gt_address` - GT gsmSCF par défaut (optionnel)
- Si non configuré, utilise le GT de la Partie Appelée de la demande entrante
- Les règles GT NAT remplacent le défaut en fonction du préfixe de la partie appelante

**Exemple :**

```
# Lorsque gsmSSF 555123456 appelle votre gsmSCF
# Entrant : Appelé=55512341112, Appelant=555123456
# Recherche GT NAT : "555" -> response_gt="55512341111"
# Réponse : Appelé=555123456, Appelant=55512341111
```

# Équilibrage de Charge

GT NAT peut être combiné avec l'équilibrage de charge M3UA pour une gestion avancée du trafic.

# Guide de Migration

## Activer GT NAT sur un Système Existant

1. **Préparer la Configuration**

```
# Ajouter à runtime.exs (garder désactivé au début)
config :omniss7,
  gt_nat_enabled: false,  # Commencer désactivé
  gt_nat_rules: [
    # Vos règles ici avec des poids
    %{calling_prefix: "877", weight: 10, response_gt:
"111111"},
    %{weight: 100, response_gt: "999999"}  # Wildcard fallback
  ]
```

2. **Tester la Configuration**

```
# Valider que la config compile
mix compile

# Tester dans iex
iex -S mix
iex> GtNat.validate_config()
```

3. **Activer en Staging**

```
gt_nat_enabled: true  # Changer à vrai
```

4. **Surveiller les Journaux**

```
tail -f log/omniss7.log | grep "GT NAT"
```

5. **Déployer en Production**

    - Déployer pendant la fenêtre de maintenance

- Surveiller les 24 premières heures de près
- Avoir un plan de retour prêt (mettre `gt_nat_enabled: false`)

# Support

Pour des problèmes ou des questions :

- Vérifiez les journaux pour les messages "GT NAT"
- Validez la config avec `GtNat.validate_config()`
- Consultez la section de dépannage de ce guide
- Contactez le support OmniSS7 avec des extraits de journaux

# Voir Aussi

- Guide HLR - Configuration du mode HLR
- Guide SMSC - Configuration du mode SMSc
- Guide STP - Configuration du routage STP
- Référence de Configuration - Référence complète de config

# Guide de Configuration HLR

← Retour à la Documentation Principale

Ce guide fournit la configuration pour utiliser OmniSS7 comme un **Home Location Register (HLR/HSS)** avec **OmniHSS** comme base de données des abonnés en backend.

# Intégration d'OmniHSS

**Le mode HLR d'OmniSS7 fonctionne comme un frontend de signalisation SS7** qui s'interface avec **OmniHSS**, un serveur d'abonnés à domicile (HSS) complet. Cette architecture sépare les préoccupations :

- **OmniSS7 (Frontend HLR)** : Gère toute la signalisation du protocole SS7/MAP, le routage SCCP et la communication réseau
- **OmniHSS (Backend HSS)** : Gère les données des abonnés, l'authentification, la configuration et les fonctionnalités avancées

## Pourquoi OmniHSS ?

OmniHSS fournit une gestion des abonnés de qualité opérateur avec des fonctionnalités incluant :

- **Support Multi-IMSI** : Chaque abonné peut avoir plusieurs IMSI associés à un seul MSISDN pour l'itinérance internationale, le changement de réseau et la configuration d'eSIM
- **Authentification Flexible** : Support pour les algorithmes d'authentification Milenage (3G/4G/5G) et COMP128 (2G)
- **Suivi des Sessions Circuits & Paquets** : Suivi indépendant des enregistrements réseau CS (circuit commuté) et PS (paquet commuté)
- **Provisionnement Avancé** : Profils de service personnalisables, services supplémentaires et données d'abonnement CAMEL

- **Conception API-First** : API HTTP RESTful pour l'intégration avec les systèmes de facturation, CRM et de provisionnement
- **Mises à Jour en Temps Réel** : Suivi de localisation, gestion de session et génération de vecteurs d'authentification

Toutes les données des abonnés, les informations d'authentification et les configurations de service sont stockées et gérées dans OmniHSS. OmniSS7 interroge OmniHSS via des appels API HTTPS pour répondre aux opérations MAP telles que UpdateLocation, SendAuthenticationInfo et SendRoutingInfo.

**Important** : Le mode HLR d'OmniSS7 est un **frontend de signalisation uniquement**. Toute la logique de gestion des abonnés, les algorithmes d'authentification, les règles de provisionnement et les opérations de base de données sont gérées par OmniHSS. Ce guide couvre la configuration du protocole SS7/MAP dans OmniSS7. Pour des informations sur le provisionnement des abonnés, la configuration de l'authentification, les profils de service et les opérations administratives, **reportez-vous à la documentation d'OmniHSS**.

# Support Multi-IMSI

**OmniHSS prend en charge nativement les configurations Multi-IMSI**, permettant à un seul abonné (identifié par MSISDN) d'avoir plusieurs IMSI. Cela permet :

- **Profils d'Itinérance Internationale** : Différents IMSI pour différentes régions afin de réduire les coûts d'itinérance
- **eSIM Multi-Profile** : Plusieurs profils réseau sur un seul appareil compatible eSIM
- **Changement de Réseau** : Changement sans couture entre les réseaux sans changer le MSISDN
- **Coordination Dual SIM** : Coordination entre plusieurs SIM physiques ou virtuelles
- **Tests & Développement** : Plusieurs IMSI de test pointant vers le même abonné

**Comment ça fonctionne :**

- Chaque IMSI a ses propres informations d'authentification (Ki, OPc, algorithme)
- Chaque IMSI peut avoir des enregistrements de session circuits et paquets indépendants
- Les services et profils des abonnés peuvent être partagés ou personnalisés par IMSI
- OmniSS7 interroge OmniHSS par IMSI, et OmniHSS renvoie les données appropriées de l'abonné
- Les systèmes de facturation peuvent suivre l'utilisation par IMSI tout en associant tous les IMSI à un seul compte

**Exemple de scénario Multi-IMSI :**

```
Abonné MSISDN : +1-555-123-4567
├─ IMSI 1 : 310260123456789 (Réseau Domicile US - authentification
Milenage)
├─ IMSI 2 : 208011234567890 (Profil d'Itinérance France -
authentification Milenage)
└─ IMSI 3 : 440201234567891 (Profil d'Itinérance UK -
authentification COMP128)
```

Tous les trois IMSI peuvent être utilisés indépendamment pour l'enregistrement réseau, mais ils appartiennent tous au même compte d'abonné. OmniHSS gère le mappage IMSI-abonné et assure une authentification et un provisionnement appropriés pour chaque IMSI.

# Table des Matières


1. Intégration d'OmniHSS
2. Support Multi-IMSI
3. Qu'est-ce que le Mode HLR ?
4. Activation du Mode HLR
5. Base de Données des Abonnés
6. Vecteurs d'Authentification
7. Mises à Jour de Localisation
8. Intégration CAMEL
9. Gestion des Abonnés en Itinérance
10. Opérations HLR

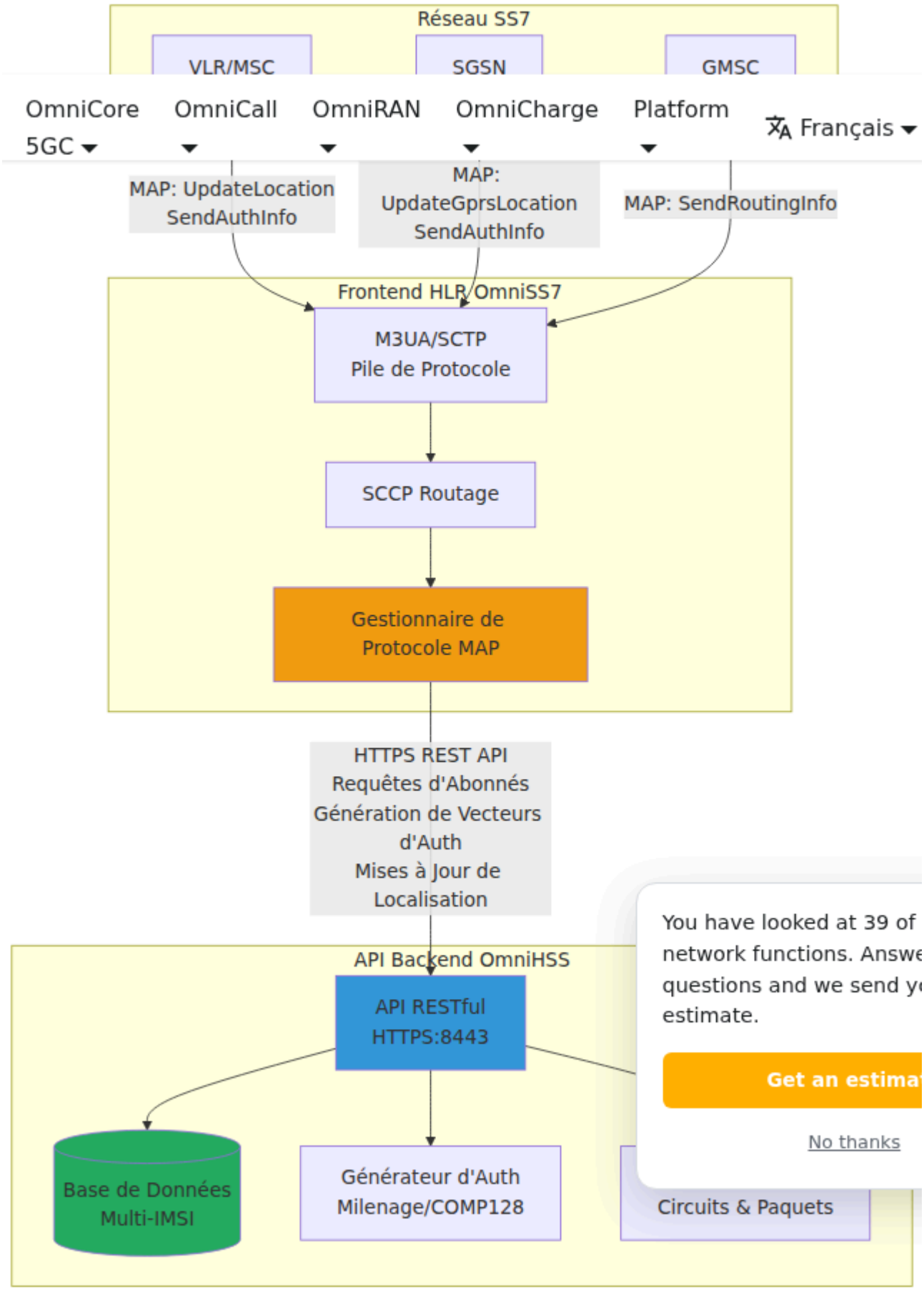
- Mapping des Champs de Réponse
  - SendRoutingInfo (SRI)
  - UpdateLocation / ISD
  - SendRoutingInfoForSM
- Résumé des Sources de Champs

# Qu'est-ce que le Mode HLR ?

**Le Mode HLR** permet à OmniSS7 de fonctionner comme un Home Location Register pour :

- **Gestion des Abonnés** : Stocker et gérer les données des abonnés
- **Authentification** : Générer des vecteurs d'authentification pour l'accès au réseau
- **Suivi de Localisation** : Traiter les mises à jour de localisation des VLR
- **Informations de Routage** : Fournir des informations de routage pour les appels et les SMS

# Architecture HLR



# Activation du Mode HLR

OmniSS7 peut fonctionner en différents modes (STP, HLR, SMSc). Le mode est sélectionné via un ensemble de drapeaux de mode et la configuration associée dans **`config/config.exs`**, le fichier de configuration par défaut de l'application.

## Changement vers le Mode HLR

Pour exécuter OmniSS7 en tant que HLR, définissez les drapeaux de mode afin que les fonctionnalités HLR soient activées et que les fonctionnalités SMSc soient désactivées, puis ajoutez la configuration spécifique au HLR :

1. **Ouvrir** `config/config.exs`
2. **Définir** les drapeaux de mode sous `config :omniss7` :
   - `map_client_enabled: true` : requis pour envoyer et recevoir des opérations MAP
   - `hlr_mode_enabled: true` : active le traitement spécifique au HLR
   - `smsc_mode_enabled: false` : désactive les fonctionnalités SMSc
3. **Ajouter** les paramètres de configuration HLR (point de terminaison API, adresses GT, paramètres ISD/CAMEL, connexion M3UA) montrés ci-dessous
4. **Personnaliser** les paramètres pour votre déploiement
5. **Redémarrer** l'application pour que les changements prennent effet

> **Remarque** : `config/runtime.exs` est un stub minimal utilisé uniquement pour les tests automatisés. Il ne contient **pas** de définitions de mode opérationnel. Toute la configuration de mode et HLR se trouve dans `config/config.exs` (avec des remplacements spécifiques à l'environnement dans des fichiers tels que `config/test.exs`). Les déploiements en production fournissent leur propre `config/config.exs`.

## Configuration du Mode HLR

La configuration complète du HLR (dans `config/config.exs`) ressemble à ceci :

```
config :omniss7,
  # Drapeaux de mode - Activer uniquement les fonctionnalités HLR
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false,

  # Configuration de l'API Backend OmniHSS
  hlr_api_base_url: "https://omnihss.example.com:8443",
  hlr_api_key: "<Clé API OmniHSS avec le rôle hlr_interworking>",

  # Adresse GT du Centre de Service HLR pour les opérations SMS
  hlr_service_center_gt_address: "1234567890",

  # Configuration de Mappage MSISDN ↔ IMSI
  # Voir : section Mappage MSISDN ↔ IMSI pour les détails
  hlr_imsi_plmn_prefix: "50557",
  hlr_msisdn_country_code: "61",
  hlr_msisdn_nsn_offset: 0,
  hlr_msisdn_nsn_length: 9,

  # Configuration InsertSubscriberData
  # Mode d'Accès Réseau : :packetAndCircuit, :packetOnly, ou
:circuitOnly
  isd_network_access_mode: :packetAndCircuit,

  # Envoyer ISD #2 (données des Services Supplémentaires)
  isd_send_ss_data: true,

  # Envoyer ISD #3 (données de Barrage d'Appels)
  isd_send_call_barring: true,

  # Configuration CAMEL (pour les réponses SendRoutingInfo)
  # Clé de Service pour l'initiation de service CAMEL
  camel_service_key: 11_110,

  # Point de Détection de Déclenchement CAMEL
  # Options : :termAttemptAuthorized, :tBusy, :tNoAnswer, :tAnswer
  camel_trigger_detection_point: :termAttemptAuthorized,

  # Préfixes VLR Domicile
  # Liste des préfixes d'adresse VLR considérés comme réseau
"domicile"
  # Si le VLR de l'abonné commence par l'un de ces préfixes,
```

```
utilisez la réponse SRI standard
  # Sinon, l'abonné est en itinérance et nous devons envoyer PRN
pour obtenir MSRN
  # Par défaut [] (pas de préfixes = tous les VLR traités comme
itinérants) si omis
  home_vlr_prefixes: ["555123"],

  # Configuration de Connexion M3UA
  # Se connecter en tant qu'ASP pour recevoir des opérations MAP
(UpdateLocation, SendAuthInfo, etc.)
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    # Point de terminaison local (système HLR)
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    # Point de terminaison STP distant
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

## Paramètres de Configuration à Personnaliser

Pour une référence complète de tous les paramètres de configuration, voir la Référence de Configuration.

| Paramètre | Type | Par Défaut |
| --- | --- | --- |
| `hlr_api_base_url` | String | *Requis* |
| `hlr_api_key` | String | *Requis* |
| `hlr_service_center_gt_address` | String | *Requis* |
| `smsc_service_center_gt_address` | String | *Requis* |
| `hlr_smsc_alert_gts` | Liste | `[]` |
| `hlr_alert_location_expiry_seconds` | Entier | `172800` |

| Paramètre | Type | Par Défaut |
| --- | --- | --- |
| `hlr_imsi_plmn_prefix` | String | `"50557"` |
| `hlr_msisdn_country_code` | String | `"61"` |
| `hlr_msisdn_nsn_offset` | Entier | `0` |
| `hlr_msisdn_nsn_length` | Entier | `9` |
| `isd_network_access_mode` | Atome | `:packetAndCircuit` |
| `isd_send_ss_data` | Booléen | `true` |

| Paramètre | Type | Par Défaut |
|---|---|---|
| `isd_send_call_barring` | Booléen | `true` |
| `camel_service_key` | Entier | `11_110` |
| `camel_trigger_detection_point` | Atome | `:termAttemptAuthorized` |
| `hlr_camel_subscriptions` | Liste | `[]` |
| `home_vlr_prefixes` | Liste | `[]` |
| `local_ip` | Tuple | *Requis* |
| `local_port` | Entier | `2905` |
| `remote_ip` | Tuple | *Requis* |

| Paramètre | Type | Par Défaut |
|---|---|---|
| `remote_port` | Entier | `2905` |
| `routing_context` | Entier | `1` |

## Que se Passe-t-il Lorsque le Mode HLR est Activé

Lorsque `hlr_mode_enabled: true`, l'interface web affichera :

- **Événements SS7** - Journalisation des événements
- **Client SS7** - Test des opérations MAP
- **Pairs** - État de la connexion (pairs M3UA/SCTP)
- **Liens HLR** - État de l'API HLR + gestion des abonnés ← *spécifique au HLR*
- **Ressources** - Surveillance du système
- **Configuration** - Visualiseur de configuration

Les onglets **Routage**, **Test de Routage** et **Liens SMSc** seront cachés.

## Remarques Importantes

- **Configuration Requise** : Le paramètre `hlr_service_center_gt_address` est **obligatoire**. L'application échouera à démarrer s'il n'est pas configuré.
- **Backend OmniHSS** : L'API OmniHSS doit être accessible à l'URL `hlr_api_base_url` configurée
- **Clé API OmniHSS** : `hlr_api_key` est obligatoire. OmniSS7 l'envoie dans l'en-tête `x-omnihss-key` sur chaque requête OmniHSS, et également lors de la vérification de l'état des Liens HLR. Si la clé est manquante ou vide, chaque requête OmniHSS échoue avec une erreur nommant `hlr_api_key`. OmniSS7 ne fait pas de requête sans la clé. Si la clé est incorrecte, OmniHSS renvoie HTTP 401.
- **Délai d'Expiration de la Requête API** : Toutes les requêtes API OmniHSS ont un **délai d'expiration codé en dur de 5 secondes**

- **Délai d'Expiration de la Requête MAP** : Toutes les requêtes MAP (SRI, UpdateLocation, SendAuthInfo, etc.) ont un **délai d'expiration codé en dur de 10 secondes**
- **Délai ISD** : Chaque message InsertSubscriberData (ISD) dans une séquence UpdateLocation a un **délai d'expiration codé en dur de 10 secondes**
- La connexion M3UA au STP est requise pour recevoir des opérations MAP
- Après avoir changé de mode, vous devez redémarrer l'application pour que les changements prennent effet
- **Interface Web** : Voir le Guide de l'Interface Web pour des informations sur l'utilisation de l'interface web
- **Accès API** : Voir le Guide API pour la documentation de l'API REST et l'accès à Swagger UI

---

# Base de Données des Abonnés

**OmniHSS gère toutes les données des abonnés** y compris les identités, les informations d'authentification, les profils de service et les informations de localisation. OmniSS7 récupère ces données via des appels API RESTful.

## Modèle d'Abonné OmniHSS

OmniHSS stocke des informations complètes sur les abonnés :

- **Plusieurs IMSI par abonné** : Support pour les configurations Multi-IMSI (eSIM, profils d'itinérance, changement de réseau)
- **Informations d'authentification** : Ki, OPc et sélection d'algorithme (Milenage ou COMP128)
- **Profils de service** : Catégorie d'abonné, services autorisés, paramètres QoS
- **Suivi de localisation** : Suivi indépendant de l'emplacement actuel VLR/MSC (session circuit) et SGSN/GGSN (session paquet)
- **Données d'abonnement CAMEL** : Clés de service, points de déclenchement et adresses gsmSCF

- **Services supplémentaires** : Transfert d'appels, barrages, attente, configurations CLIP/CLIR
- **État administratif** : Activé/désactivé, restrictions de service, dates d'expiration

---

# Vecteurs d'Authentification

## Générer des Vecteurs d'Auth

**OmniHSS génère des vecteurs d'authentification** en utilisant les algorithmes Milenage ou COMP128 en fonction de la méthode d'authentification configurée pour chaque abonné. Lorsque OmniSS7 reçoit des requêtes MAP **sendAuthenticationInfo** :

1. OmniSS7 extrait l'IMSI de la requête MAP
2. OmniSS7 appelle l'API OmniHSS pour générer des vecteurs d'authentification
3. OmniHSS récupère les informations d'authentification Ki et OPc de l'abonné
4. OmniHSS génère le nombre demandé de vecteurs (RAND, XRES, CK, IK, AUTN)
5. OmniSS7 encode les vecteurs au format MAP et les renvoie au VLR/SGSN demandeur

## Intégration de l'API OmniHSS

OmniSS7 communique avec OmniHSS via l'API REST HTTPS pour récupérer les informations des abonnés, mettre à jour les données de localisation et générer des vecteurs d'authentification :

```
config :omniss7,
  hlr_api_base_url: "https://omnihss-server:8443",
  hlr_api_key: "<Clé API OmniHSS avec le rôle hlr_interworking>"
```

Chaque requête à OmniHSS contient l'en-tête `x-omnihss-key` avec la valeur de `hlr_api_key`. OmniHSS utilise la clé pour trouver le rôle du client. La clé HLR doit avoir le rôle `hlr_interworking`. Ce rôle donne accès à la recherche d'abonnés, à l'ensemble de clés, aux vecteurs d'authentification et aux points de terminaison de session circuit que le HLR utilise.

Lorsque OmniSS7 reçoit des opérations MAP du réseau SS7, il interroge OmniHSS pour :

- **Récupérer les données de l'abonné** par IMSI ou MSISDN
- **Générer des vecteurs d'authentification** en utilisant les informations Ki/OPc stockées
- **Mettre à jour la localisation de la session circuit** lorsque les abonnés effectuent UpdateLocation
- **Vérifier l'état de l'abonné** et les droits de service

---

# Mises à Jour de Localisation

## Traitement de la Mise à Jour de Localisation

Lors de la réception des requêtes MAP **updateLocation**, OmniSS7 coordonne avec OmniHSS pour enregistrer l'abonné à un nouveau VLR :

1. **Extraire les informations de localisation** de la requête UpdateLocation (IMSI, nouveau GT VLR, nouveau GT MSC)
2. **Interroger OmniHSS** pour vérifier que l'abonné existe et est activé
3. **Mettre à jour la session circuit** dans OmniHSS avec la nouvelle localisation VLR/MSC
4. **Envoyer des messages InsertSubscriberData (ISD)** pour provisionner l'abonné au nouveau VLR
5. **Retourner la réponse UpdateLocation** au VLR (inclut le GT HLR de `hlr_service_center_gt_address`)
6. **Envoyer alertServiceCenter** aux GT SMSc configurés (si `hlr_smsc_alert_gts` est peuplé)

**Remarque** : Le paramètre de configuration `hlr_service_center_gt_address` spécifie le Titre Global du HLR qui est renvoyé dans les réponses UpdateLocation. Cela permet au VLR/MSC d'identifier et de router les messages de retour vers ce HLR.

## Intégration du Centre de Service d'Alerte

Après une mise à jour de localisation réussie, le HLR peut automatiquement notifier les systèmes SMSc qu'un abonné est désormais joignable en envoyant des messages **alertServiceCenter** (opcode MAP 64). Pour des informations sur la façon dont le SMSc gère ces alertes, voir Gestion du Centre de Service d'Alerte dans le Guide SMSc.

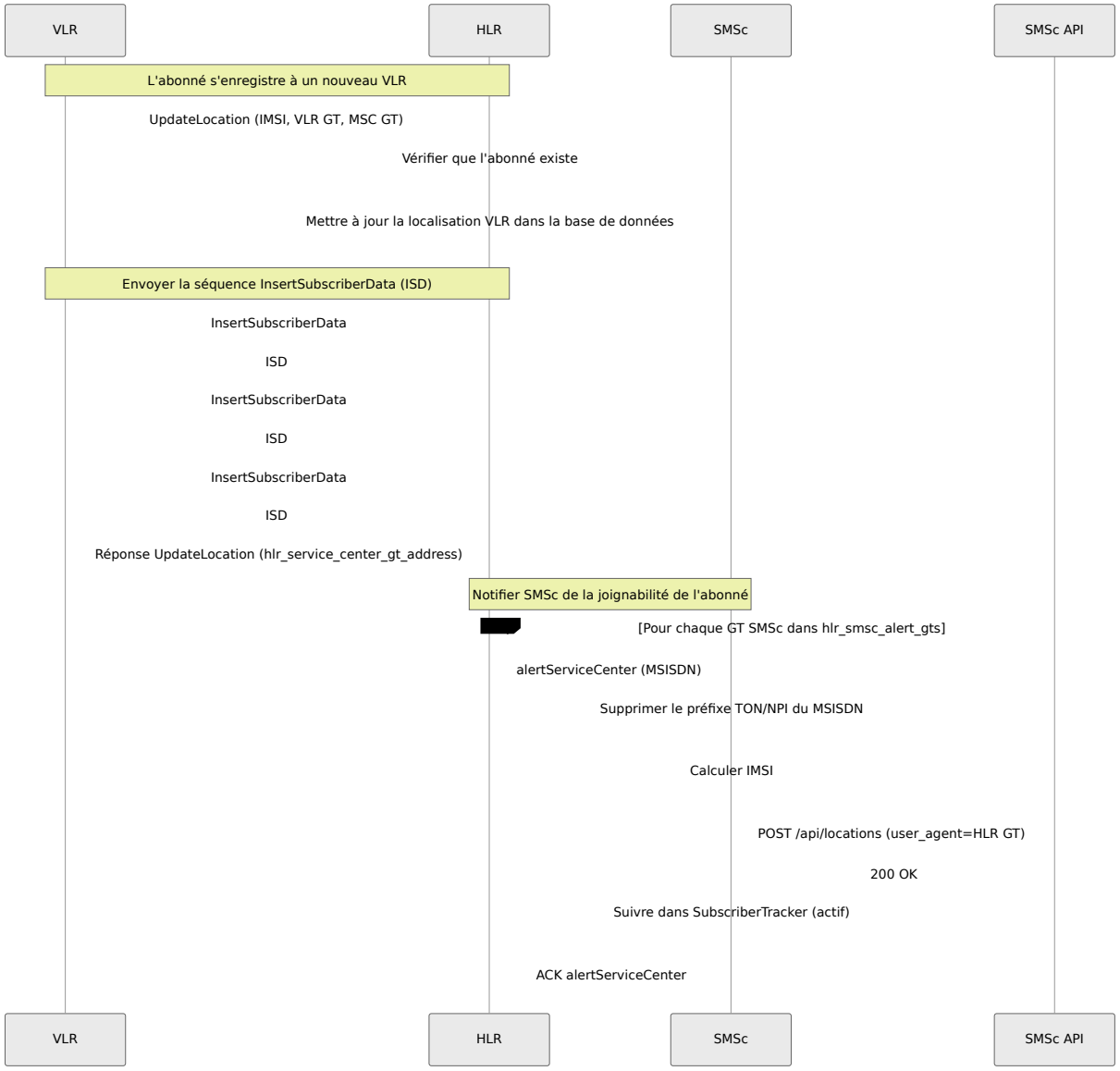
### Configuration

Configurer la liste des Titres Globaux SMSc à notifier :

```
config :omniss7,
  # Liste des GT SMSc pour envoyer alertServiceCenter après
UpdateLocation
  hlr_smsc_alert_gts: [
    "15559876543",
    "15559876544"
  ],

  # Temps d'expiration de la localisation lorsque le SMSc reçoit
alertServiceCenter (par défaut : 48 heures)
  hlr_alert_location_expiry_seconds: 172800
```

### Diagramme de Flux




## Comportement

Lorsqu'un abonné effectue une mise à jour de localisation :

1. Le HLR envoie alertServiceCenter à **chaque** GT SMSc dans la liste `hlr_smsc_alert_gts`
2. Le message inclut le MSISDN de l'abonné
3. Le HLR utilise `hlr_service_center_gt_address` comme GT de l'appelant
4. Adressage SCCP : SSN appelant=6 (HLR), SSN appelé=8 (SMSc)

Le SMSc reçoit l'alerte et :

- **Supprime le préfixe TON/NPI** du MSISDN (par exemple, "19123123213" → "123123213")

- Marque l'abonné comme joignable dans sa base de données de localisation (via POST à /api/locations)
- **Définit le champ `user_agent`** sur le GT HLR lors de l'appel à l'API (pour suivre quel HLR a envoyé l'alerte)
- Définit le temps d'expiration de la localisation basé sur `hlr_alert_location_expiry_seconds`
- Suit l'abonné dans le Tracker d'Abonnés du SMSc pour la surveillance

#### Tests

Utilisez la page **Abonnés Actifs** dans l'interface Web pour envoyer manuellement des messages alertServiceCenter pour les tests :

1. Accédez à l'onglet "Abonnés Actifs"
2. Trouvez la section "Tester le Service d'Alerte"
3. Entrez MSISDN, SMSc GT et HLR GT (les valeurs par défaut sont pré-remplies à partir de la configuration)
   - SMSc GT par défaut est la première entrée dans `hlr_smsc_alert_gts`
   - HLR GT par défaut est `hlr_service_center_gt_address`
4. Cliquez sur "Envoyer alertServiceCenter"

Ceci est utile pour tester le traitement des alertes SMSc sans nécessiter un flux complet UpdateLocation. Le formulaire utilise la validation `phx-blur` pour éviter d'afficher des erreurs lors de la saisie.

## Configuration InsertSubscriberData (ISD)

Après un UpdateLocation réussi, le HLR envoie des données de provisionnement d'abonnés au VLR en utilisant des messages **InsertSubscriberData** (ISD). La configuration ISD vous permet de personnaliser les données envoyées et comment.

Pour référence des paramètres de configuration, voir Configuration ISD dans la Référence de Configuration.

#### Séquence ISD

Le HLR peut envoyer jusqu'à 3 messages ISD séquentiels :

1. **ISD #1** (Toujours envoyé) - Données de base de l'abonné :
    - IMSI
    - MSISDN
    - Catégorie d'abonné
    - Statut de l'abonné (serviceGranted)
    - Liste des services de transport
    - Liste des services télématiques
    - Mode d'accès au réseau
2. **ISD #2** (Optionnel) - Données des Services Supplémentaires (SS) :
    - Paramètres de renvoi d'appel (inconditionnel, occupé, sans réponse, non joignable)
    - Attente d'appel
    - Mise en attente d'appel
    - Service multi-parties
    - Statut et fonctionnalités des services supplémentaires
3. **ISD #3** (Optionnel) - Données de Blocage d'Appels :
    - Blocage de tous les appels sortants (BAOC)
    - Blocage des appels internationaux sortants (BOIC)
    - Données de restriction d'accès

**Options de Configuration**

```
# Configuration InsertSubscriberData
# Mode d'accès au réseau : :packetAndCircuit, :packetOnly, ou
:circuitOnly
isd_network_access_mode: :packetAndCircuit,

# Envoyer ISD #2 (données des Services Supplémentaires)
isd_send_ss_data: true,

# Envoyer ISD #3 (données de Blocage d'Appels)
isd_send_call_barring: true,
```

### Mode d'Accès au Réseau

Le paramètre `isd_network_access_mode` contrôle quel type d'accès au réseau l'abonné est autorisé :

| Valeur | Description | Cas d'utilisation |
|---|---|---|
| `:packetAndCircuit` | À la fois commuté par paquets (GPRS/LTE) et commuté par circuits (voix) | Par défaut - Abonnés à service complet |
| `:packetOnly` | Commuté par paquets uniquement (données/LTE) | Cartes SIM uniquement de données, dispositifs IoT |
| `:circuitOnly` | Commuté par circuits uniquement (voix/SMS) | Dispositifs anciens, forfaits voix uniquement |

### Contrôle des Messages ISD

Vous pouvez contrôler quels messages ISD sont envoyés en fonction de vos exigences réseau :

**Envoyer tous les ISD** (Par défaut - Ensemble complet de fonctionnalités) :

```
isd_send_ss_data: true,
isd_send_call_barring: true,
```

**Envoyer uniquement les données de base de l'abonné** (Provisionnement minimal) :

```
isd_send_ss_data: false,
isd_send_call_barring: false,
```

**Envoyer de base + services supplémentaires** (Pas de blocage d'appels) :

```
isd_send_ss_data: true,
isd_send_call_barring: false,
```

**Exemple de Flux ISD**

Lorsque UpdateLocation est reçu :

```
VLR → HLR: UpdateLocation (DÉBUT)
HLR → VLR: InsertSubscriberData #1 (CONTINUER) - Données de base
VLR → HLR: ISD #1 ACK (CONTINUER)
HLR → VLR: InsertSubscriberData #2 (CONTINUER) - Données SS [si
activé]
VLR → HLR: ISD #2 ACK (CONTINUER)
HLR → VLR: InsertSubscriberData #3 (CONTINUER) - Blocage d'appels
[si activé]
VLR → HLR: ISD #3 ACK (CONTINUER)
HLR → VLR: Réponse UpdateLocation (FIN)
```

Si `isd_send_ss_data` ou `isd_send_call_barring` sont réglés sur `false`, ces messages ISD sont ignorés, et la FIN UpdateLocation est envoyée plus tôt.

**Meilleures Pratiques**

- **Configuration par Défaut** : Utilisez `:packetAndCircuit` et activez tous les ISD pour une compatibilité maximale
- **IoT/M2M** : Utilisez `:packetOnly` et désactivez les données SS/le blocage d'appels pour les dispositifs uniquement de données
- **Interopérabilité** : Certains VLR plus anciens peuvent ne pas prendre en charge tous les services supplémentaires - désactivez `isd_send_ss_data` si des problèmes surviennent
- **Performance** : Désactiver les ISD inutilisés réduit la surcharge des messages et accélère les mises à jour de localisation

---

# Intégration CAMEL

## Configuration CAMEL pour SendRoutingInfo

Lors de la réponse aux demandes **SendRoutingInfo** (SRI) d'un GMSC (Gateway MSC), le HLR peut instruire le GMSC d'invoquer des services CAMEL pour le routage intelligent des appels et le contrôle des services.

Pour référence des paramètres de configuration, voir Configuration CAMEL dans la Référence de Configuration.

### Qu'est-ce que CAMEL ?

**CAMEL** (Applications Personnalisées pour la Logique Améliorée des Réseaux Mobiles) est un protocole qui permet des services réseau intelligents dans les réseaux GSM/UMTS. Il permet aux opérateurs de réseau de mettre en œuvre des services à valeur ajoutée tels que :

- Facturation prépayée
- Filtrage et blocage d'appels
- Réseaux Privés Virtuels (VPN)
- Services à tarif premium
- Renvoi d'appels avec logique personnalisée
- Services basés sur la localisation

### Options de Configuration

```
# Configuration CAMEL (pour les réponses SendRoutingInfo)
# Clé de service pour l'initiation du service CAMEL
camel_service_key: 11_110,

# Point de Détection du Déclencheur CAMEL
# Options : :termAttemptAuthorized, :tBusy, :tNoAnswer, :tAnswer
camel_trigger_detection_point: :termAttemptAuthorized,
```

### Clé de Service

La `camel_service_key` identifie quel service CAMEL doit être invoqué au gsmSCF (Service Control Function). Il s'agit d'un identifiant numérique configuré dans votre réseau :

| Clé de Service | Cas d'utilisation typique |
|---|---|
| `11_110` | Contrôle des appels terminants prépayés (par défaut) |
| `100` | Service prépayé d'origine |
| `200` | Renvoi d'appels avec logique personnalisée |
| `300` | Réseau Privé Virtuel (VPN) |
| Personnalisé | Services spécifiques à l'opérateur |

**Exemple de Configuration :**

```
# Pour le contrôle des appels terminants prépayés
camel_service_key: 11_110,

# Pour le service VPN
camel_service_key: 300,
```

**Point de Détection du Déclencheur**

Le `camel_trigger_detection_point` spécifie quand le service CAMEL doit être déclenché lors de la configuration de l'appel :

| Point de Détection | Description | Quand Déclenché |
|---|---|---|
| `:termAttemptAuthorized` | Tentative d'appel autorisée (par défaut) | Avant que l'appel ne soit routé vers l'abonné |
| `:tBusy` | Occupé terminant | Lorsque l'abonné est occupé |
| `:tNoAnswer` | Pas de réponse terminante | Lorsque l'abonné ne répond pas |
| `:tAnswer` | Réponse terminante | Lorsque l'abonné répond à l'appel |

**Exemples de Configuration :**

**Contrôle prépayé standard** (déclenchement avant le routage) :

```
camel_trigger_detection_point: :termAttemptAuthorized,
```

**Gestion personnalisée des appels occupés** (déclenchement lorsque occupé) :

```
camel_trigger_detection_point: :tBusy,
```

**Facturation basée sur la réponse** (déclenchement à la réponse) :

```
camel_trigger_detection_point: :tAnswer,
```

**Réponse SRI avec CAMEL**

Lorsqu'il est configuré, les réponses SendRoutingInfo incluent des informations d'abonnement CAMEL :

```
GMSC → HLR: SendRoutingInfo (DÉBUT)
HLR → GMSC: Réponse SRI (FIN) avec :
  - IMSI
  - Numéro VLR
  - État de l'abonné
  - Informations de routage CAMEL :
    * Clé de service : 11_110
    * Adresse gsmSCF : <adresse configurée>
    * Point de détection du déclencheur : termAttemptAuthorized
    * Gestion des appels par défaut : continueCall

GMSC contacte gsmSCF au point de déclenchement pour exécuter le
service CAMEL
```

**Meilleures Pratiques**

- **Réseaux de Production** : Utilisez des clés de service standardisées convenues avec votre fournisseur gsmSCF
- **Tests** : Utilisez `:termAttemptAuthorized` pour des tests les plus complets
- **Services Prépayés** : La clé de service `11_110` est une norme industrielle courante pour les appels terminants prépayés
- **Gestion de Fallback** : `defaultCallHandling: :continueCall` garantit que les appels se poursuivent si le gsmSCF est inaccessible

# Abonnements CAMEL dans InsertSubscriberData

Lors de l'UpdateLocation, le HLR envoie un InsertSubscriberData supplémentaire contenant `vlrCamelSubscriptionInfo` lorsque l'IMSI de l'abonné correspond à une règle dans `hlr_camel_subscriptions`. La première règle correspondante l'emporte. Le MSC utilise l'O-CSI pour les appels MO (DP collectedInfo) et le VT-CSI pour les appels MT (DP termAttemptAuthorized), et ouvre le dialogue CAP vers le titre global `gsmscf_address`, de sorte que le gsmSCF est choisi par le HLR, et non par la configuration du MSC.

```
config :omniss7,
  hlr_camel_subscriptions: [
    %{
      imsi_prefix: "3133800000009",
      o_csi: %{service_key: 100, gsmscf_address: "14155550300",
default_call_handling: :release_call},
      t_csi: %{service_key: 101, gsmscf_address: "14155550300",
default_call_handling: :continue_call}
    }
  ]
```

| Clé | Description |
|---|---|
| `imsi_prefix` | Préfixe IMSI auquel la règle s'applique |
| `o_csi` / `t_csi` | Au moins un. `service_key` (InitialDP serviceKey), `gsmscf_address` (gsmSCF E.164 GT), `default_call_handling` (`:release_call` ou `:continue_call`, appliqué par le MSC lorsque le gsmSCF ne répond pas) |

Les CSIs sont envoyés avec la gestion des capacités CAMEL 2 (CAP v2). Une règle avec une clé manquante génère une erreur nommant la clé.

---

# Gestion des Abonnés en Roaming

## Détection du VLR Domicile vs VLR en Roaming

Lorsque le HLR reçoit une demande **SendRoutingInfo** (SRI), il doit déterminer si l'abonné est sur un VLR "domicile" (dans votre réseau) ou sur un VLR en roaming (visite d'un autre réseau). Le comportement diffère en fonction de cette détermination :

Pour référence des paramètres de configuration, voir Préfixes VLR Domicile dans la Référence de Configuration.

- **VLR Domicile** : Retourner la réponse SRI standard avec des informations de routage CAMEL
- **VLR en Roaming** : Envoyer une demande Provide Roaming Number (PRN) pour obtenir un MSRN, puis le retourner dans la réponse SRI

# Configuration

```
# Préfixes VLR Domicile
# Liste des préfixes d'adresse VLR considérés comme réseau
"domicile"
# Si l'adresse VLR de l'abonné commence par l'un de ces préfixes,
utiliser la réponse SRI standard
# Sinon, l'abonné est en roaming et nous devons envoyer PRN pour
obtenir MSRN
home_vlr_prefixes: ["555123"],
```

**Exemple de Configuration** :

```
# Réseau domicile unique
home_vlr_prefixes: ["555123"],

# Réseaux domicile multiples (par exemple, différentes régions ou
filiales)
home_vlr_prefixes: ["555123", "555124", "555125"],
```

# Comment Cela Fonctionne

### 1. Flux Abonné Domicile (Standard)

Lorsque l'adresse VLR de l'abonné commence par un préfixe domicile configuré :

```
GMSC → HLR: SendRoutingInfo (MSISDN: "1234567890")
HLR interroge l'API backend pour les données de l'abonné
HLR vérifie l'adresse VLR : "5551234567"
HLR détermine : VLR commence par "555123" → Réseau domicile
HLR → GMSC: Réponse SRI avec informations de routage CAMEL :
  - IMSI
  - Numéro VLR : "5551234567"
  - Adresse gsmSCF (MSC) : "5551234501"
  - Clé de service CAMEL : 11_110
  - Point de détection du déclencheur : termAttemptAuthorized
```

### 2. Flux Abonné en Roaming (PRN Requis)

Lorsque l'adresse VLR de l'abonné NE correspond PAS à un préfixe domicile :

```
GMSC → HLR: SendRoutingInfo (MSISDN: "1234567890")
HLR interroge l'API backend pour les données de l'abonné
HLR vérifie l'adresse VLR : "49170123456"
HLR détermine : VLR ne commence pas par "555123" → Roaming
HLR → MSC: ProvideRoamingNumber (PRN) :
  - MSISDN : "1234567890"
  - IMSI : "999999876543210"
  - Numéro MSC : "49170123456"
  - Adresse GMSC : "5551234501"
MSC → HLR: Réponse PRN avec MSRN : "49170999888777"
HLR → GMSC: Réponse SRI avec informations de routage :
  - IMSI
  - Numéro VLR : "49170123456"
  - Numéro de Roaming (MSRN) : "49170999888777"
```

## Différences de Structure de Réponse

#### Réponse SRI Abonné Domicile

```
%{
  imsi: "999999876543210",
  extendedRoutingInfo: {
    :camelRoutingInfo, %{
      gmscCamelSubscriptionInfo: %{
        "t-CSI": %{
          serviceKey: 11_110,
          "gsmSCF-Address": "5551234501",
          defaultCallHandling: :continueCall,
          "t-BcsmTriggerDetectionPoint": :termAttemptAuthorized
        }
      }
    }
  },
  subscriberInfo: %{
    locationInformation: %{"vlr-number": "5551234567"},
    subscriberState: {:notProvidedFromVLR, :NULL}
  }
}
```

### Réponse SRI Abonné en Roaming

```
%{
  imsi: "999999876543210",
  extendedRoutingInfo: {
    :routingInfo, %{
      roamingNumber: "49170999888777"  # MSRN de PRN
    }
  },
  subscriberInfo: %{
    locationInformation: %{"vlr-number": "49170123456"},
    subscriberState: {:notProvidedFromVLR, :NULL}
  }
}
```

# Opération Provide Roaming Number (PRN)

### Structure de Demande PRN

La demande PRN envoyée au MSC/VLR contient :

| Champ | Source | Description |
|---|---|---|
| **MSISDN** | Demande SRI | Numéro de téléphone de l'abonné |
| **IMSI** | API HLR | IMSI de l'abonné |
| **Numéro MSC** | API HLR | MSC servant l'abonné en roaming (`serving_msc`) |
| **Adresse GMSC** | Demande SRI | GMSC effectuant la demande SRI originale |
| **Numéro de Référence d'Appel** | Statique | Identifiant de référence d'appel |
| **Phases CAMEL Supportées** | Statique | Phases CAMEL supportées par GMSC |

**Gestion de la Réponse PRN**

Le HLR s'attend à une réponse PRN contenant :

- **MSRN** (Numéro de Roaming de la Station Mobile) : Un numéro temporaire alloué par le réseau visité pour le routage de l'appel

**Gestion des Erreurs** :

- Si le PRN expire → Retourne l'erreur 27 (Abonné Absent) dans la réponse SRI
- Si le PRN échoue → Retourne l'erreur 27 (Abonné Absent) dans la réponse SRI
- Si le MSRN ne peut pas être extrait → Retourne l'erreur 27 (Abonné Absent) dans la réponse SRI

# Exemples de Configuration

### Opérateur avec Réseau Domicile Unique

```
# Tous les adresses VLR commençant par "555123" sont considérées
comme domicile
home_vlr_prefixes: ["555123"],
```

- VLR `5551234567` → Domicile (réponse CAMEL)
- VLR `5551235001` → Domicile (réponse CAMEL)
- VLR `49170123456` → Roaming (PRN + réponse MSRN)

### Opérateur Multi-Régions

```
# Plusieurs réseaux domicile à travers différentes régions
home_vlr_prefixes: ["555123", "555124", "555125"],
```

- VLR `5551234567` → Domicile (région 1)
- VLR `5552341234` → Domicile (région 2)
- VLR `5553411111` → Domicile (région 3)
- VLR `44201234567` → Roaming (international)

### Configuration de Test

Pour tester la fonctionnalité PRN, définissez une liste vide pour traiter tous les VLR comme en roaming :

```
# Tous les VLR sont traités comme en roaming (pour tester le flux
PRN)
home_vlr_prefixes: [],
```

# Meilleures Pratiques

- **Sélection de Préfixe** : Utilisez le préfixe unique le plus court qui identifie les VLR de votre réseau (par exemple, code pays + code réseau)

- **Préfixes Multiples** : Incluez tous les préfixes VLR de votre réseau, y compris différentes régions et filiales
- **Accords de Roaming** : Assurez-vous que le PRN est correctement pris en charge par les réseaux partenaires en roaming
- **Tests** : Testez à la fois les scénarios domicile et roaming de manière approfondie avant le déploiement en production
- **Surveillance** : Surveillez les taux de timeout PRN pour identifier les problèmes de connectivité avec les partenaires en roaming

# Dépannage

**Symptôme** : Tous les abonnés traités comme en roaming

- **Cause** : `home_vlr_prefixes` non configuré ou préfixes ne correspondent pas aux adresses VLR
- **Solution** : Vérifiez les adresses VLR dans votre base de données et mettez à jour les préfixes en conséquence

**Symptôme** : Les demandes PRN expirent

- **Cause** : Problèmes de connectivité réseau vers le MSC/VLR partenaire en roaming
- **Solution** : Vérifiez le routage M3UA/SCCP vers les adresses MSC distantes

**Symptôme** : MSRN invalide dans la réponse SRI

- **Cause** : Format de réponse PRN du partenaire en roaming ne correspond pas à la structure attendue
- **Solution** : Examinez les journaux de réponse PRN et ajustez `extract_msrn_from_prn/1` si nécessaire

---

# Opérations HLR

## Opérations MAP Supportées

OmniSS7 dispatches des opérations MAP entrantes par opcode (indépendamment de la phase TCAP). Les opérations entrantes suivantes sont gérées :

| Opcode | Opération | But |
|---|---|---|
| 2 | `updateLocation` | Enregistrer la localisation VLR (déclenche la séquence ISD + alertServiceCenter optionnel) |
| 23 | `updateGprsLocation` | Enregistrer la localisation SGSN (commuté par paquets) |
| 3 | `cancelLocation` | Désinscrire de l'ancien VLR |
| 7 | `insertSubscriberData` | Recevoir le profil de l'abonné (par exemple, lors de l'action en tant que côté VLR) |
| 22 | `sendRoutingInfo` (SRI) | Fournir MSRN/routage CAMEL pour les appels (domicile vs roaming) |
| 45 | `sendRoutingInfoForSM` (SRI-for-SM) | Fournir IMSI + SMSC GT pour le routage SMS |
| 44 | `MT-Forward-SM` | Livraison de SMS mobile-terminé |
| 46 | `MO-Forward-SM` | Réception de SMS mobile-initié |
| 47 | `reportSM-DeliveryStatus` | Rapport de statut de livraison SMS-GMSC |
| 64 | `alertServiceCenter` | Alerte d'abonné-joignable au SMSc |
| 66 | `readyForSM` | MSC/SGSN signale que l'abonné est prêt pour SMS |
| 56 | `sendAuthenticationInfo` | Générer des vecteurs d'authentification |

| Opcode | Opération | But |
|---|---|---|
| 57 | `restoreData` | Restauration des données d'abonné VLR |
| 67 | `purgeMS` | Purger un enregistrement d'abonné au HLR |
| 59 | `processUnstructuredSS-Request` | USSD (géré via la passerelle USSD lorsqu'il est activé) |
| 71 | `anyTimeInterrogation` (ATI) | Interrogation de la localisation/ état de l'abonné |
| 10 | `registerSS` | Enregistrement de service supplémentaire (par exemple, Renvoi d'Appel Inconditionnel) |

Les opcodes non gérés sont répondus avec une erreur `facilityNotSupported` (21).

# Mappage des Champs de Réponse

Cette section détaille d'où provient chaque champ dans les réponses HLR.

#### Réponse SendRoutingInfo (SRI)

**But** : Fournit des informations de routage pour les appels entrants à un abonné.

Le HLR fournit deux types de réponse différents en fonction de si l'abonné est sur un VLR domicile ou en roaming :

##### Réponse Abonné Domicile (Routage CAMEL)

Utilisé lorsque l'adresse VLR de l'abonné commence par une valeur configurée dans `home_vlr_prefixes`.

**Structure de Réponse** :

| Champ | Source | Description |
|---|---|---|
| **IMSI** | API OmniHSS | IMSI de l'abonné provenant de la base de données OmniHSS |
| **Numéro VLR** | API OmniHSS | VLR actuel servant l'abonné (`circuit_session.assigned_vlr`) |
| **État de l'Abonné** | Statique | Toujours `notProvidedFromVLR` |
| **extendedRoutingInfo** | - | Type : `camelRoutingInfo` |
| **Adresse gsmSCF** | API OmniHSS | MSC servant l'abonné (`circuit_session.assigned_msc`) |
| **Clé de Service** | config.exs | Identifiant de service CAMEL (`camel_service_key`) |
| **Point de Détection du Déclencheur** | config.exs | Quand déclencher CAMEL (`camel_trigger_detection_point`) |
| **Gestion des Capacités CAMEL** | Statique | Niveau de support de phase CAMEL |
| **Gestion des Appels par Défaut** | Statique | Fallback si gsmSCF inaccessible |

#### Réponse Abonné en Roaming (Routage MSRN)

Utilisé lorsque l'adresse VLR de l'abonné NE correspond PAS à une valeur configurée dans `home_vlr_prefixes`.

**Structure de Réponse** :

| Champ | Source | Description | |
|---|---|---|---|
| **IMSI** | API OmniHSS | IMSI de l'abonné provenant de la base de données OmniHSS | `"9` |
| **Numéro VLR** | API OmniHSS | VLR actuel servant l'abonné (`circuit_session.assigned_vlr`) | `"4` |
| **État de l'Abonné** | Statique | Toujours `notProvidedFromVLR` | `:n` |
| **extendedRoutingInfo** | - | Type : `routingInfo` | - |
| **Numéro de Roaming (MSRN)** | Réponse PRN | MSRN obtenu de la demande ProvideRoamingNumber | `"4` |

**Logique de Décision de Routage** :

```
1. OmniSS7 reçoit la demande SendRoutingInfo
2. OmniSS7 interroge les données de l'abonné depuis l'API OmniHSS
3. OmniSS7 vérifie l'adresse VLR par rapport aux home_vlr_prefixes
:

   Si VLR commence par un préfixe domicile :
     → Retourner les informations de routage CAMEL (flux abonné
domicile)

   Si VLR ne correspond PAS à un préfixe domicile :
     → Envoyer ProvideRoamingNumber (PRN) au MSC
     → Extraire MSRN de la réponse PRN
     → Retourner les informations de routage avec MSRN (flux
abonné en roaming)
```

**Flux de Données** :

- OmniSS7 interroge OmniHSS pour les informations de l'abonné
- OmniHSS retourne IMSI, localisation actuelle VLR/MSC, et état de l'abonné
- OmniSS7 utilise ces données pour construire la réponse MAP

**Exigences de Configuration** :

```
# Dans config/config.exs
home_vlr_prefixes: ["555123"],  # Liste des préfixes VLR domicile
```

**Réponses d'Erreur** :

- Si `serving_vlr` et `serving_msc` sont `null` : Retourne l'erreur 27 (Abonné Absent)
- Si l'abonné n'est pas trouvé : Retourne l'erreur 1 (Abonné Inconnu)
- Si la demande PRN expire (cas de roaming) : Retourne l'erreur 27 (Abonné Absent)
- Si la réponse PRN est invalide (cas de roaming) : Retourne l'erreur 27 (Abonné Absent)

### Réponse UpdateLocation avec InsertSubscriberData

**Objectif** : Enregistre l'abonné au nouveau VLR et provisionne les données de l'abonné.

#### Réponse UpdateLocation END

| Champ | Source | Description | Exemple |
|---|---|---|---|
| **Numéro HLR** | config.exs | Le titre global de ce HLR (`hlr_service_center_gt_address`) | `"5551234568"` |
| **Type de message TCAP** | Statique | Réponse finale après tous les ISD | `END` |

#### InsertSubscriberData #1 (Données de base de l'abonné)

| Champ | Source | Description | Exemple |
|---|---|---|---|
| **IMSI** | Demande | Provenant de la demande UpdateLocation | `"99999987654321` |
| **MSISDN** | API OmniHSS | Numéro de téléphone de l'abonné provenant d'OmniHSS | `"555123456"` |
| **Catégorie** | Statique | Catégorie de l'abonné | `"\n"` (0x0A) |
| **Statut de l'abonné** | Statique | Statut du service | `:serviceGranted` |
| **Liste des services de support** | Statique | Services de support pris en charge | `[<<31>>]` |
| **Liste des téléservices** | Statique | Téléservices pris en charge | `[<<17>>, "!", "\""]` |
| **Mode d'accès au réseau** | config.exs | Accès paquet/circuit (`isd_network_access_mode`) | `:packetAndCircu` |

**InsertSubscriberData #2 (Services supplémentaires) - Optionnel**

| Champ | Source | Description | Contrôlé par |
|---|---|---|---|
| **SS provisionné** | Statique | Données des services supplémentaires | `isd_send_ss_data: true` |
| **Renvoi d'appel** | Statique | Configurations de renvoi (inconditionnel, occupé, pas de réponse, non joignable) | Configuration activée |
| **Attente d'appel** | Statique | Statut du service d'attente d'appel | Configuration activée |
| **Service multi-parties** | Statique | Support des appels en conférence | Configuration activée |

**ISD #2 inclut** :

- Renvoi d'appel inconditionnel (code SS 21)
- Renvoi d'appel en cas d'occupation (code SS 41)
- Renvoi d'appel en cas de non réponse (code SS 42)
- Renvoi d'appel en cas de non joignable (code SS 62)
- Attente d'appel (code SS 43)
- Service multi-parties (code SS 51)
- Services CLIP/CLIR

**InsertSubscriberData #3 (Interdiction d'appel) - Optionnel**

| Champ | Source | Description | Contrôlé par |
|---|---|---|---|
| **Informations sur l'interdiction d'appel** | Statique | Configurations d'interdiction d'appel | `isd_send_call_barring: true` |
| **BAOC** | Statique | Interdiction de tous les appels sortants (code SS 146) | Configuration activée |
| **BOIC** | Statique | Interdiction des appels internationaux sortants (code SS 147) | Configuration activée |
| **Données de restriction d'accès** | Statique | Restrictions d'accès au réseau | Configuration activée |

**Contrôle de séquence ISD** :

- ISD #1 : **Toujours envoyé** - Contient des données essentielles de l'abonné
- ISD #2 : Envoyé uniquement si `isd_send_ss_data: true` dans config/config.exs
- ISD #3 : Envoyé uniquement si `isd_send_call_barring: true` dans config/config.exs

---

#### Réponse SendRoutingInfoForSM (SRI-for-SM)

**Objectif** : Fournit des informations de routage MSC/SMSC pour la livraison de SMS. Lorsqu'un SMSc doit livrer un SMS à un abonné, il envoie une demande SRI-for-SM au HLR pour déterminer où router le message.

Le HLR répond dans l'un des deux modes, décidé par le **GT de la partie appelante SCCP** de la demande :

**1. Routage à domicile (par défaut : chaque initiateur externe).** Ce mode renvoie un IMSI synthétique plus le GT propre du IP-SM-GW comme numéro de nœud réseau. Le HLR n'expose jamais de vrais IMSI et nœuds de service aux opérateurs externes. Le MT-ForwardSM revient au IP-SM-GW, qui gère ensuite la livraison. C'est le comportement correct pour le trafic d'interconnexion et d'itin��rance.

| Champ | Source | Description | Comment généré |
|---|---|---|---|
| **IMSI** | Calculé | IMSI synthétique dérivé du MSISDN | `PLMN_PREFIX + zero_padded_MSISDN` |
| **Numéro de nœud réseau** | config.exs | Adresse GT SMSC pour le routage SMS | `smsc_service_center_gt_address` |

**2. Routage direct sur le réseau (uniquement lorsque la demande provient de l'un de nos propres nœuds SMSc / IP-SM-GW).** Lorsque le GT appelant correspond à la liste configurée `home_smsc_gts`, le HLR recherche l'abonné dans son backend et renvoie le **vrai IMSI** et le **vrai MSC de service** comme numéro de nœud réseau, afin que notre SMSc puisse envoyer le MT-ForwardSM directement au MSC. Si aucun MSC de service n'est disponible pour cet abonné, il revient au routage à domicile.

| Champ | Source | Description | Exemple |
|---|---|---|---|
| **IMSI** | Backend HLR | Vrai IMSI de l'abonné | `"505570123456789"` |
| **Numéro de nœud réseau** | Backend HLR | Vrai `serving_msc` de la dernière updateLocation | `"5559876543"` |

**Paramètres de configuration** (à partir de `config/config.exs`) :

```
# Adresse GT du centre de service (retournée dans les réponses
SRI-for-SM routées à domicile)
# Cela indique au SMSc demandeur où envoyer les messages MT-
ForwardSM
smsc_service_center_gt_address: "5551234567",  # Requis

# Routage direct sur le réseau : GT SCCP des nœuds SMSc / IP-SM-
GW.
# Une SRI-for-SM de l'un de ceux-ci obtient le VRAI MSC de service
au lieu du routage à domicile.
# Tous les autres sont routés à domicile. Laissez vide ([]) pour
routage à domicile de tout.
home_smsc_gts: ["5551234567"],                # Optionnel (par
défaut [])

# Configuration de mappage MSISDN ↔ IMSI
# Préfixe PLMN : MCC (001 = Réseau de test) + MNC (01 = Opérateur
de test)
hlr_imsi_plmn_prefix: "001001",                 # Seul paramètre
de configuration nécessaire !
```

> **Remarque :** Le routage direct sur le réseau ne fonctionne que si le HLR détient effectivement le `serving_msc` de l'abonné. C'est-à-dire que le MSC de service a effectué un MAP `updateLocation` vers ce HLR. Si le MSC s'enregistre via un autre mécanisme (par exemple, GSUP vers un HLR séparé), le backend n'a pas l'emplacement et la requête est routée à domicile.

---

# Mappage MSISDN ↔ IMSI

**Paramètres de configuration** :

Ces paramètres contrôlent comment OmniSS7 génère des IMSI synthétiques à partir des MSISDN pour les réponses SRI-for-SM :

- `hlr_imsi_plmn_prefix` : Le préfixe MCC+MNC à utiliser lors de la construction des IMSI synthétiques (par exemple, "50557" pour MCC=505, MNC=57)
- `hlr_msisdn_country_code` : Code de pays à préfixer lors de la réalisation du mappage inverse IMSI→MSISDN (par exemple, "61" pour l'Australie, "1" pour les États-Unis/Canada)
- `hlr_msisdn_nsn_offset` : Position du caractère où le Numéro d'Abonné National (NSN) commence dans le MSISDN (typiquement 0 si le MSISDN n'inclut pas le code de pays, ou la longueur du code de pays s'il l'inclut)
- `hlr_msisdn_nsn_length` : Nombre de chiffres à extraire du MSISDN comme NSN

Pour des détails de configuration supplémentaires, voir Mappage MSISDN ↔ IMSI dans la Référence de Configuration.

**Pourquoi le mappage MSISDN vers IMSI est-il nécessaire ?**

Le protocole MAP pour **SendRoutingInfoForSM** (SRI-for-SM) exige que le HLR renvoie un **IMSI** (Identité d'Abonné Mobile Internationale) dans sa réponse. Cependant, le SMSc demandeur ne connaît que le **MSISDN** de l'abonné (numéro de téléphone).

Dans un réseau traditionnel :

- Le SMSc envoie SRI-for-SM avec le MSISDN de destination (par exemple, "5551234567")
- Le HLR doit rechercher l'abonné dans sa base de données pour trouver son IMSI
- Le HLR renvoie l'IMSI dans la réponse SRI-for-SM
- Le SMSc utilise ensuite cet IMSI lors de l'envoi du MT-ForwardSM au MSC/VLR

**Approche d'OmniSS7 - IMSIs Synthétiques** :

Au lieu de maintenir une base de données complète des abonnés avec des mappages MSISDN vers IMSI, OmniSS7 utilise un schéma d'encodage simple pour **calculer** des IMSI synthétiques directement à partir du MSISDN. Cette approche offre deux avantages clés :

1. **Confidentialité** : Les vrais IMSI des abonnés stockés dans la base de données HLR ne sont jamais exposés dans les réponses SRI-for-SM envoyées sur le réseau SS7
2. **Simplicité** : Pas besoin de requêter la base de données HLR pour des recherches IMSI lors des opérations SRI-for-SM - l'IMSI est calculé à la volée à partir du MSISDN

**Comment ça fonctionne** :

Les MSISDN sont directement encodés dans la partie abonné de l'IMSI (les chiffres après MCC+MNC) :

```
IMSI = PLMN_PREFIX + zero_padded_MSISDN
```

Où :

- **PLMN_PREFIX** : MCC + MNC (par exemple, "001001" pour le Réseau de Test)
- **MSISDN** : Tous les chiffres numériques du numéro de téléphone
- **Zero Padding** : Complété par des zéros à gauche pour remplir l'IMSI à exactement 15 chiffres

**Exemple étape par étape** :

```
# Configuration
plmn_prefix = "001001"  # MCC 001 + MNC 01

# Entrée : MSISDN de la demande SRI-for-SM (décodé TBCD)
msisdn = "555123456"    # 9 chiffres

# Étape 1 : Calculer l'espace disponible pour le numéro d'abonné
subscriber_digits = 15 - String.length("001001")  # = 9 chiffres

# Étape 2 : Compléter le MSISDN avec des zéros pour remplir la
portion abonnée
padded_msisdn = String.pad_leading("555123456", 9, "0")  # =
"555123456" (aucun remplissage nécessaire)

# Étape 3 : Concaténer le préfixe PLMN + MSISDN complété
imsi = "001001" <> "555123456"  # = "001001555123456" (exactement
15 chiffres)
```

**Exemples complets** :

| MSISDN d'entrée | Préfixe PLMN | Chiffres d'abonné disponibles | MSISDN complété | IMSI final |
|---|---|---|---|---|
| `"555123456"` | `"001001"` (6) | 9 | `"555123456"` | `"001001555123` |
| `"99"` | `"001001"` (6) | 9 | `"000000099"` | `"001001000000` |
| `"999999999"` | `"001001"` (6) | 9 | `"999999999"` | `"001001999999` |
| `"91123456789"` | `"001001"` (6) | 9 | `"555123456"` | `"001001555123` |

**Gestion des cas limites** :

- **MSISDN courts** : Complétés par des zéros (par exemple, `"99"` → `"000000099"`)
- **MSISDN longs** : Les chiffres de droite sont conservés, les chiffres de gauche sont tronqués (par exemple, `"91123456789"` → `"555123456"`)
- **Longueur de l'IMSI** : Toujours exactement 15 chiffres

**Mappage inverse (IMSI → MSISDN)** :

Le SMSc peut inverser ce mappage pour convertir les IMSIs en MSISDN :

```
# Entrée : IMSI de la réponse SRI-for-SM
imsi = "001001555123456"

# Étape 1 : Supprimer le préfixe PLMN
plmn_prefix = "001001"
subscriber_portion = String.slice(imsi, 6, 9)  # = "555123456"

# Étape 2 : Supprimer les zéros à gauche pour obtenir le MSISDN
réel
msisdn = String.replace_leading(subscriber_portion, "0", "")  # =
"555123456"
```

**Exemples de mappage inverse** :

| IMSI d'entrée | Préfixe PLMN | Portion d'abonné | Supprimer les zéros à gauche | MSISDI final |
|---|---|---|---|---|
| `"001001555123456"` | `"001001"` | `"555123456"` | `"555123456"` | `"5551234` |
| `"001001000000099"` | `"001001"` | `"000000099"` | `"99"` | `"99"` |
| `"001001999999999"` | `"001001"` | `"999999999"` | `"999999999"` | `"9999999` |

**Propriétés de ce mappage** :

- **Déterministe** : Le même MSISDN produit toujours le même IMSI
- **Réciproque** : Peut convertir de l'IMSI au MSISDN
- **Configuration minimale** : Nécessite seulement `hlr_imsi_plmn_prefix`
- **Préservation de la vie privée** : Les vrais IMSI ne sont jamais exposés
- **Pas de recherche dans la base de données** : Calcul rapide, aucun appel API nécessaire
- **Toujours 15 chiffres** : L'IMSI est toujours exactement de 15 chiffres

**Gestion des entrées MSISDN** :

Lorsque le HLR reçoit une demande SRI-for-SM, le MSISDN subit un décodage TBCD :

1. **Décodage TBCD** : Convertir le TBCD binaire en chaîne (peut inclure un préfixe TON/NPI comme "91")
2. **Extraction des chiffres** : Conserver uniquement les chiffres numériques, supprimer tout caractère non numérique
3. **Normalisation** : Si plus long que l'espace disponible, prendre les chiffres de droite ; si plus court, compléter par des zéros
4. **Encodage** : Concaténer le préfixe PLMN + MSISDN normalisé

**Considérations de sécurité** :

Les IMSIs synthétiques retournés dans les réponses SRI-for-SM sont uniquement destinés à des fins de routage. Ce ne sont PAS les vrais IMSIs stockés dans la base de données des abonnés HLR. Cela fournit une couche supplémentaire de protection de la vie privée, car les vrais IMSIs des abonnés ne sont exposés que lorsque cela est absolument nécessaire (par exemple, lors des opérations UpdateLocation ou SendAuthenticationInfo qui nécessitent de vrais vecteurs d'authentification).

**Flux de réponse** :

```
1. SMSc → HLR : Demande SRI-for-SM
   - MSISDN (TBCD) : "91123456789" (inclut TON/NPI)

2. Traitement HLR :
   - Décodage TBCD : "91123456789"
   - Extraction des chiffres : "91123456789" (11 chiffres)
   - Ajustement à 9 chiffres : "555123456" (9 de droite)
   - Ajout de PLMN : "001001" + "555123456" = "001001555123456"
   - Obtenir le GT SMSC à partir de la config : "5551234567"

3. HLR → SMSc : Réponse SRI-for-SM
   - IMSI : "001001555123456" (synthétique, toujours 15 chiffres)
   - Numéro de nœud réseau : "5551234567" (où envoyer MT-
ForwardSM)

4. SMSc envoie MT-ForwardSM à "5551234567" avec IMSI
"001001555123456"
```

**Configuration** :

Les paramètres suivants sont utilisés dans `config/config.exs` :

```
# Préfixe PLMN : MCC (001 = Réseau de test) + MNC (01 = Opérateur
de test)
hlr_imsi_plmn_prefix: "001001",

# Extraction NSN (si les MSISDN incluent le code de pays)
hlr_msisdn_country_code: "1",      # Utilisé pour le mappage
inverse (IMSI→MSISDN)
hlr_msisdn_nsn_offset: 1,          # Ignorer le code de pays d'une
seule chiffre
hlr_msisdn_nsn_length: 10          # Extraire le NSN de 10
chiffres
```

**Configuration d'extraction NSN** :

Si vos MSISDN incluent le code de pays (par exemple, `"99988000088"` au lieu de juste `"88000088"`), vous devez configurer l'extraction du NSN :

- **`hlr_msisdn_nsn_offset`** : Position où commence le NSN (typiquement la longueur de votre code de pays)
- **`hlr_msisdn_nsn_length`** : Nombre de chiffres dans le NSN

**Exemples** :

| Exemple | Code pays | Exemple MSISDN | nsn_offset | nsn_length | NSN ex |
|---|---|---|---|---|---|
| CC à 1 chiffre | `"9"` | `"95551234567"` | `1` | `10` | `"555123` |
| CC à 2 chiffres | `"99"` | `"99412345678"` | `2` | `9` | `"412345` |
| CC à 3 chiffres | `"999"` | `"99988000088"` | `3` | `8` | `"880000` |

**Comment ça fonctionne** :

1. **MSISDN → IMSI** : Extraire le NSN du MSISDN, compléter par des zéros à gauche, concaténer avec le préfixe PLMN

```
MSISDN : "99988000088"
NSN : String.slice("99988000088", 3, 8) = "88000088"
NSN complété : "088000088" (9 chiffres)
IMSI : "547050" + "088000088" = "547050088000088"
```

2. **IMSI → MSISDN** : Supprimer le préfixe PLMN, enlever les zéros à gauche, préfixer le code pays

```
IMSI : "547050088000088"
Portion d'abonné : "088000088"
Supprimer les zéros : "88000088"
MSISDN : "+999" + "88000088" = "+99988000088"
```

# Exigences API : Aucune - SRI-for-SM utilise uniquement des valeurs calculées et la configuration.

# Aucun appel API backend n'est requis.

## Résumé des sources de champ

| Type de source | Description | Exemples |
|---|---|---|
| **API OmniHSS** | Données dynamiques de la base de données des abonnés OmniHSS | IMSI, MSISDN, VLR/MSC de service à partir de circuit_session |
| **config.exs** | Paramètres de configuration d'OmniSS7 | `smsc_service_center_gt_address`, `camel_service_key`, `isd_network_access_mode` |
| **Statique** | Valeurs codées en dur dans le générateur de réponse | Statut de l'abonné, services de support, codes SS |
| **Demande** | Champs extraits de la demande MAP entrante | IMSI de UpdateLocation, MSISDN de SRI |
| **Calculé** | Valeurs dérivées en utilisant la logique | IMSI synthétique dans SRI-for-SM (hlr_imsi_prefix + NSN) |

## Dépendances de configuration

**Requis dans config/config.exs** :

- `hlr_service_center_gt_address` - Utilisé dans les réponses UpdateLocation
- `smsc_service_center_gt_address` - Utilisé dans les réponses SRI-for-SM (où le MT-ForwardSM doit être routé)

**Optionnel dans config/config.exs** (avec valeurs par défaut) :

- `camel_service_key` - Par défaut : `11_110`
- `camel_trigger_detection_point` - Par défaut : `:termAttemptAuthorized`
- `isd_network_access_mode` - Par défaut : `:packetAndCircuit`
- `isd_send_ss_data` - Par défaut : `true`
- `isd_send_call_barring` - Par défaut : `true`
- `hlr_imsi_plmn_prefix` - Par défaut : `"001001"` (préfixe PLMN pour le mappage MSISDN↔IMSI)

**Requis d'OmniHSS** :

OmniHSS doit fournir des points de terminaison API REST pour :

- Recherche d'abonnés par IMSI et MSISDN
- Mises à jour de localisation de session de circuit (assignation VLR/MSC)
- Génération de vecteurs d'authentification
- Requêtes de statut d'abonné et de profil de service

---

# Documentation Connexe

**Documentation OmniSS7 :**


- ← Retour à la Documentation Principale
- Guide des Fonctionnalités Communes
- Guide du Client MAP
- Référence Technique
- Référence de Configuration

**Documentation OmniHSS :** Pour la gestion des abonnés, la provision, la configuration d'authentification et les opérations administratives, référez-vous à la **documentation produit OmniHSS**. OmniHSS contient toute la logique de base de données des abonnés, les algorithmes d'authentification, les règles de provisionnement de services et les capacités de gestion Multi-IMSI.

---

# Guide de Configuration du Client MAP

← Retour à la Documentation Principale

Ce guide couvre comment faire fonctionner OmniSS7 en tant que **Client MAP**. Dans ce rôle, OmniSS7 se connecte à un réseau SS7 en tant que Processus Serveur d'Application (ASP). Il émet des opérations MAP (Mobile Application Part) vers les HLR, MSC/VLR, SGSN, GMLC et d'autres éléments du réseau central. Le comportement du protocole suit 3GPP TS 29.002.

## Table des Matières

1. Qu'est-ce que le Mode Client MAP ?
2. Activation du Mode Client MAP
3. Référence des Opérations MAP
4. Exemples d'Opérations Courantes

5. Envoi de Requêtes via l'API
6. Métriques et Surveillance
7. Dépannage

---

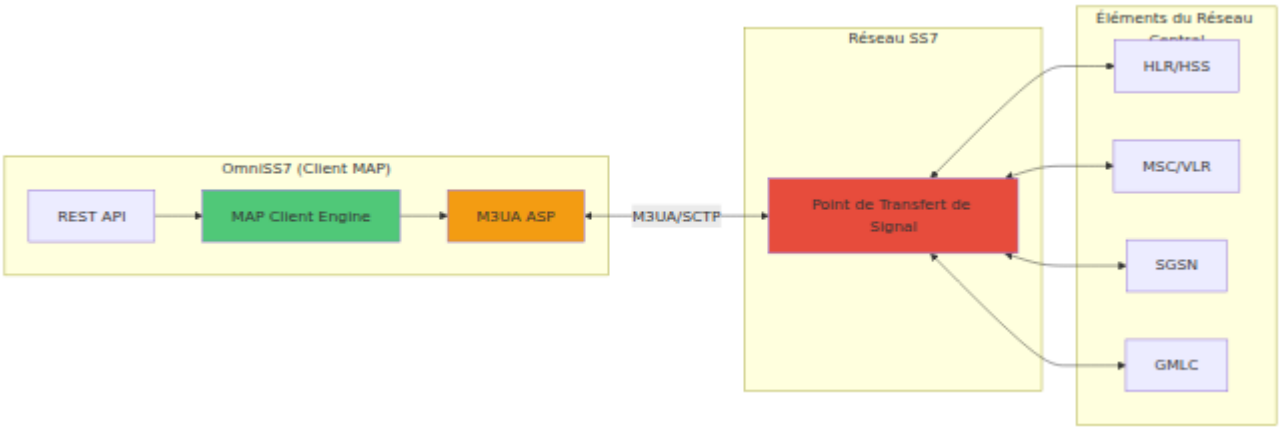
# Qu'est-ce que le Mode Client MAP ?

Le **Mode Client MAP** permet à OmniSS7 de se connecter en tant que **Processus Serveur d'Application (ASP)** à un pair M3UA (STP ou SGP) via SCTP et d'échanger des messages **MAP**. Chaque opération est exposée comme un point de terminaison HTTP. L'API encode une requête comme un dialogue TCAP/MAP et la transporte via SCCP et M3UA vers le réseau SS7. OmniSS7 décode ensuite la réponse du réseau et la renvoie de manière synchrone à l'appelant.

Les utilisations typiques incluent :

- **Gestion de la mobilité** : Mise à jour de la localisation, Annulation de la localisation, Purge MS, Fournir un numéro de roaming
- **Gestion des appels** : Envoyer des informations de routage, Fournir des informations sur l'abonné
- **SMS** : SRI-for-SM, MO/MT-ForwardSM, rapport de statut de livraison
- **Authentification** : Envoyer des informations d'authentification
- **Données sur l'abonné** : Insérer/Supprimer des données sur l'abonné, Interrogation à tout moment
- **Services supplémentaires** : enregistrer/effacer/activer/interroger SS, gestion des mots de passe
- **Services de localisation (LCS)** et routage **GPRS**

## Architecture Réseau




---

# Activation du Mode Client MAP

Le mode Client MAP est configuré dans `config/runtime.exs`. Pour la référence complète des connexions M3UA, voir Paramètres de Connexion M3UA.

```
config :omniss7,
  # Activer le mode Client MAP
  map_client_enabled: true,

  # Connexion M3UA pour le Client MAP (se connecte en tant qu'ASP
à un STP/SGP distant)
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :map_client_asp,
    local_ip: {10, 0, 0, 100},
    local_port: 2905,
    remote_ip: {10, 0, 0, 1},
    remote_port: 2905,
    routing_context: 1
  }
```

## Paramètres `map_client_m3ua`

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `mode` | String | Oui | - | Rôle M3UA. `"ASP"` pour le mode client (se connecte à un STP/SGP). |
| `callback` | Tuple | Oui | - | `{Module, Function, Args}` invoqué pour chaque charge utile M3UA entrante. Pour le Client MAP, c'est `{MapClient, :handle_payload, []}`. |
| `process_name` | Atom | Oui | - | Nom enregistré du processus M3UA ASP. |
| `local_ip` | Tuple | Oui | - | Adresse de liaison SCTP locale sous forme de tuple IP, par exemple `{10, 0, 0, 100}`. |
| `local_port` | Integer | Non | 2905 | Port SCTP local. 2905 est le port M3UA enregistré par l'IANA. |
| `remote_ip` | Tuple | Oui | - | Adresse IP du STP/SGP distant sous forme de tuple. |
| `remote_port` | Integer | Non | 2905 | Port SCTP du STP/SGP distant. |

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| `routing_context` | Integer | Non | - | Valeur du Contexte de Routage M3UA, lorsque le pair en nécessite un. |

> Clés de niveau supérieur : `map_client_enabled` (Boolean, par défaut `false`) active le Client MAP ; `map_client_m3ua` (Map) contient la connexion ci-dessus.

Vous configurez l'adressage SCCP pour les opérations sortantes (Titres Globaux des Parties Appelantes/Appelées, SSNs, codes de point) séparément. Voir la Référence de Configuration.

---

# Référence des Opérations MAP

Chaque opération ci-dessous est accessible via `POST /api/<endpoint>`. Les opcodes sont les codes d'opération MAP provenant de 3GPP TS 29.002. Les requêtes sont synchrones : l'API bloque jusqu'à ce que la réponse SS7 arrive ou que la requête expire (voir Codes de Réponse).

## Gestion de la Mobilité

| Opération | Point de Terminaison | Opcode | Description |
|---|---|---|---|
| Mise à jour de la localisation | `/api/updateLocation` | 2 | Enregistrer le VLR de service d'un abonné auprès du HLR. |
| Annuler la localisation | `/api/cancelLocation` | 3 | Instruire un VLR à supprimer un enregistrement d'abonné. |
| Fournir un numéro de roaming | `/api/prn` | 4 | Obtenir un MSRN du MSC de service. |
| Purger MS | `/api/purgeMS` | 67 | Marquer un abonné comme purgé au HLR. |
| Envoyer une identification | `/api/sendIdentification` | 55 | Récupérer les données IMSI/authentification du VLR précédent. |
| Réinitialiser | `/api/reset` | 37 | Réinitialisation initiée par le HLR vers un VLR. |
| Restaurer les données | `/api/restoreData` | 57 | Le VLR demande la restauration des données de l'abonné auprès du HLR. |

## Gestion des Appels & Données sur l'Abonné

| Opération | Point de Terminaison | Opcode | |
|---|---|---|---|
| Envoyer des informations de routage | `/api/sri` | 22 | Inte des rou |
| Fournir des informations sur l'abonné | `/api/provideSubscriberInfo` | 70 | Der l'éta l'ab |
| Supprimer les données de l'abonné | `/api/deleteSubscriberData` | 8 | Sup de l |
| Interrogation à tout moment | `/api/anyTimeInterrogation` | 71 | gsm info de l |
| Interrogation des données d'abonnement à tout moment | `/api/anyTimeSubscriptionInterrogation` | 62 | Inte d'al |
| Modification à tout moment | `/api/anyTimeModification` | 65 | Mod d'al |
| Noter les données de l'abonné modifiées | `/api/noteSubscriberDataModified` | 5 | Le H gsm mo |
| Noter l'événement MM | `/api/noteMM-Event` | 89 | Sig de mo |

## SMS

| Opération | Point de Terminaison | Opcode | Description |
|---|---|---|---|
| Envoyer des informations de routage pour SM | `/api/sri-for-sm` | 45 | Interroger le HLR pour le nœud de service pour la livraison de SMS. |
| MT-Forward SM | `/api/MT-forwardSM` | 44 | Livrer un SM mobile terminé au MSC/SGSN de service. |
| MO-Forward SM | `/api/forwardSM` | 46 | Soumettre un SM d'origine mobile au SMSC. |
| Envoyer SM (PDU) | `/api/sendSM`, `/api/deliverPDU` | 44 | Construire et livrer un PDU SMS-DELIVER. |
| Signaler le statut de livraison de SM | `/api/reportSM-DeliveryStatus` | 47 | Signaler le résultat de la livraison de SM au HLR. |
| Prêt pour SM | `/api/readyForSM` | 66 | Notifier le HLR qu'un MS est joignable pour SMS. |
| Alerter le Centre de Service | `/api/alertServiceCentre` | 64 | Alerter un SMSC qu'un abonné est disponible. |

## Authentification & Équipement

| Opération | Point de Terminaison | Opcode | Description |
|---|---|---|---|
| Envoyer des informations d'authentification | `/api/send-auth-info` | 56 | Récupérer des vecteurs d'authentification auprès du HLR. |
| Envoyer IMSI | `/api/sendIMSI` | 58 | Résoudre un MSISDN à son IMSI auprès du HLR. |
| Vérifier IMEI | `/api/checkIMEI` | 43 | Interroger l'état de l'équipement auprès de l'EIR. |

## Services Supplémentaires

| Opération | Point de Terminaison | Opcode | Description |
|---|---|---|---|
| Enregistrer SS | `/api/registerSS` | 10 | Enregistrer un service supplémentaire (par exemple, transfert d'appel / CFU). |
| Effacer SS | `/api/eraseSS` | 11 | Effacer un service supplémentaire. |
| Activer SS | `/api/activateSS` | 12 | Activer un service supplémentaire. |
| Désactiver SS | `/api/deactivateSS` | 13 | Désactiver un service supplémentaire. |
| Interroger SS | `/api/interrogateSS` | 14 | Interroger l'état d'un service supplémentaire. |
| Enregistrer le mot de passe | `/api/registerPassword` | 17 | Enregistrer un mot de passe SS. |
| Obtenir le mot de passe | `/api/getPassword` | 18 | Récupérer/vérifier un mot de passe SS. |
| Indication de Vérification de SS Transmise | `/api/forwardCheckSS-Indication` | 38 | Vérification non confirmée vers un VLR. |

## Services de Localisation (LCS) & GPRS

| Opération | Point de Terminaison | Opcode | Description |
|---|---|---|---|
| Envoyer des informations de routage pour LCS | `/api/sendRoutingInfoForLCS` | 85 | GMLC interroge le HLR pour le routage LCS. |
| Fournir la localisation de l'abonné | `/api/provideSubscriberLocation` | 83 | Demander la localisation d'un abonné au nœud de service. |
| Rapport de localisation de l'abonné | `/api/subscriberLocationReport` | 86 | Signaler la localisation d'un abonné au GMLC. |
| Envoyer des informations de routage pour GPRS | `/api/sendRoutingInfoForGprs` | 24 | Interroger le HLR pour le routage GPRS (sélection de GGSN). |
| Mettre à jour la localisation GPRS | `/api/updateGprsLocation` | 23 | Enregistrer le SGSN de service d'un abonné auprès du HLR. |

> **Portée LCS :** les opérations de Services de Localisation sont **uniquement sortantes (demandeur)**. Le Client MAP les déclenche vers un HLR, un

> nœud de service ou un GMLC. Le nœud n'agit pas en tant que *serveur* LCS (il ne répondra pas à un ProvideSubscriberLocation entrant ni ne traitera un SubscriberLocationReport entrant). Les encodeurs transportent les éléments d'information obligatoires ainsi que les options de base notées ci-dessus. L'ensemble de paramètres LCS plus large (QoS, priorité, formes GAD, mot de code, numéro de référence) n'est pas encore exposé.

## CAP / CAMEL & Injection Brute

| Opération | Point de Terminaison | Description |
|---|---|---|
| CAP InitialDP | `/api/cap/initialDP` | Déclencher un dialogue CAMEL vers un gsmSCF. |
| CAP Connect / Continue / ReleaseCall / ApplyCharging | `/api/cap/connect`, `/api/cap/continue`, `/api/cap/releaseCall`, `/api/cap/applyCharging` | Opérations de contrôle d'appel CAMEL. Voir le Guide du Passerelle CAMEL. |
| Raw SCCP | `/api/raw/sccp` | Injecter une charge utile SCCP construite à la main (test/diagnostics). |
| Raw M3UA | `/api/raw/m3ua` | Injecter une charge utile M3UA construite à la main (test/diagnostics). |

---

# Exemples d'Opérations Courantes

## Envoyer des Informations de Routage pour SM (SRI-for-SM)

Interroge le HLR pour le nœud de service utilisé pour livrer un SMS. Pour savoir comment le côté HLR traite cela, voir SRI-for-SM dans le Guide HLR.

**Adressage SCCP (chemin de livraison SMS SMSc).** Par défaut, le SRI-for-SM est envoyé **route-on-GT adressé au MSISDN de destination** (appelé SSN 6/HLR), ce qui nécessite que le STP détienne une entrée GTT pour cette plage de numéros. Pour les abonnés **sur le réseau**, le SMSc adresse directement le SRI au **HLR GT** configuré. Il route ensuite sur l'adresse de pair (déjà routable) du HLR, donc aucune GTT par plage n'est nécessaire, et le HLR retourne le véritable MSC de service. Cela est sélectionné par destination :

| Config (`:omniss7`) | But | Exemple |
|---|---|---|
| `hlr_gt_address` | GT du HLR domicile à adresser aux SRI sur le réseau | `"61455501111"` |
| `home_msisdn_prefixes` | Préfixes MSISDN sur le réseau ; une correspondance adresse le HLR GT | `["6777"]` |

Une destination correspondant à une entrée `home_msisdn_prefixes` (avec `hlr_gt_address` défini) est envoyée au HLR GT avec le SSN appelé 6 ; tout le reste conserve la valeur par défaut route-on-GT-by-MSISDN. La requête SRI-for-SM accepte `sccp_opts` (`:called_party`, `:called_ssn`) pour remplacer la partie appelée SCCP pour tout appelant.

**Point de Terminaison :** `POST /api/sri-for-sm`

**Requête :**

```
{
  "msisdn": "447712345678",
  "serviceCenter": "447999123456"
}
```

**Réponse :**

```
{
  "result": {
    "imsi": "234509876543210",
    "locationInfoWithLMSI": {
      "networkNode-Number": "447999555111"
    }
  }
}
```

**cURL :**

```
curl -X POST http://localhost/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{"msisdn": "447712345678", "serviceCenter": "447999123456"}'
```

# Envoyer des Informations de Routage (SRI)

**Point de Terminaison :** `POST /api/sri`

**Requête :**

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456"
}
```

# Fournir un Numéro de Roaming (PRN)

**Point de Terminaison :** `POST /api/prn`

**Requête :**

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456",
  "msc_number": "447999555111",
  "imsi": "234509876543210"
}
```

## Envoyer des Informations d'Authentification

**Point de Terminaison :** `POST /api/send-auth-info`

**Requête :**

```
{
  "imsi": "234509876543210",
  "vectors": 5
}
```

**Réponse :**

```
{
  "result": {
    "authenticationSetList": [
      {
        "rand": "0123456789ABCDEF0123456789ABCDEF",
        "xres": "ABCDEF0123456789",
        "ck": "0123456789ABCDEF0123456789ABCDEF",
        "ik": "FEDCBA9876543210FEDCBA9876543210",
        "autn": "0123456789ABCDEF0123456789ABCDEF"
      }
    ]
  }
}
```

## Mettre à Jour la Localisation

Enregistre le VLR de service d'un abonné auprès du HLR, qui répond avec la séquence d'Insertion des Données de l'Abonné. Voir Mises à Jour de Localisation dans le Guide HLR.

**Point de Terminaison :** `POST /api/updateLocation`

**Requête :**

```
{
  "imsi": "234509876543210",
  "vlr": "447999555111"
}
```

## Interrogation à Tout Moment (ATI)

**Point de Terminaison :** `POST /api/anyTimeInterrogation`

**Requête :**

```
{
  "hlr_gt": "447999123456",
  "msisdn": "447712345678"
}
```

## Catégories d'Opérations



# Envoi de Requêtes via l'API

## Utilisation de Swagger UI

L'interface Swagger UI fournit une interface interactive pour chaque point de terminaison.

1. Naviguez vers `http://your-server/swagger`.
2. Sélectionnez un point de terminaison (par exemple, `/api/sri-for-sm`) et cliquez sur **Essayer**.
3. Remplissez le corps de la requête et cliquez sur **Exécuter**.
4. La réponse SS7 décodée est affichée ci-dessous.

Le document OpenAPI brut est disponible à `http://your-server/swagger.json`.

## Codes de Réponse de l'API

| Code | Signification |
|---|---|
| 200 | Succès : le résultat décodé est dans le corps de la réponse. |
| 400 | Mauvaise requête : paramètres invalides ou manquants. |
| 504 | Délai d'attente de la passerelle : pas de réponse SS7 dans le délai de la requête (par défaut 10 secondes). |

Certaines opérations sont non confirmées (fire-and-forget) et retournent immédiatement une fois le message envoyé.

---

# Métriques du Client MAP

Toutes les métriques sont exportées sur le point de terminaison Prometheus (`/metrics`).

**Métrique :** `map_requests_total` **Type :** Compteur **Description :** Total des requêtes MAP envoyées. **Étiquettes :** `operation` : par exemple `sri`, `sri_for_sm`, `prn`, `authentication_info`, `update_location`.

**Métrique :** `map_request_errors_total` **Type :** Compteur **Description :** Total des requêtes MAP qui ont entraîné une erreur ou un délai d'attente. **Étiquettes :** `operation`.

**Métrique :** `map_request_duration_milliseconds` **Type :** Histogramme **Description :** Distribution des temps de réponse des requêtes MAP. **Étiquettes :** `operation`.

**Métrique :** `map_pending_requests` **Type :** Jauge **Description :** Requêtes MAP actuellement en attente d'une réponse.

## Exemples de Requêtes Prometheus

```
# Requêtes SRI-for-SM dans la dernière heure
increase(map_requests_total{operation="sri_for_sm"}[1h])

# Temps de réponse moyen SRI
rate(map_request_duration_milliseconds_sum{operation="sri"}[5m])
  / rate(map_request_duration_milliseconds_count{operation="sri"}
[5m])

# Taux d'erreur par opération
sum(rate(map_request_errors_total[5m])) by (operation)

# Requêtes en attente actuelles
map_pending_requests
```

---

# Dépannage du Client MAP

## Délai d'Attente des Requêtes (504)

**Symptômes :** L'API retourne 504 ; pas de réponse du HLR/MSC.

**Causes possibles :**

- L'association M3UA au STP/SGP n'est pas ACTIVE.
- Le STP n'a pas de route pour le Titre Global de destination.
- Adressage SCCP incorrect (GT des Parties Appelées/Appelantes, SSN) ou Contexte de Routage.

**Résolution :**

1. Confirmer l'état de l'association M3UA sur le tableau de bord ou via les vues d'état du Guide STP.
2. Vérifier la connectivité réseau (SCTP) vers le STP.
3. Confirmer que le Titre Global de destination est routable et que le sous-système de destination (par exemple HLR SSN 6) est en service.
4. Examiner les journaux pour les messages de retour SCCP (voir ci-dessous).

# Messages de Retour / Erreur SCCP

**Symptômes :** Les journaux montrent des messages de retour du Service Unitdata SCCP (UDTS).

**Causes de retour courantes :**

- **Pas de Traduction pour l'Adresse** : le Titre Global n'est pas dans la table de routage du STP.
- **Échec de Sous-système** : le sous-système de destination (par exemple HLR SSN 6) est indisponible.
- **Échec / Congestion Réseau** : problème réseau transitoire.

**Résolution :**

- Confirmer la configuration de routage/traduction du STP pour le GT de destination.
- Vérifier que le sous-système de destination est opérationnel.
- Réessayer après que la congestion se soit dissipée.

---

# Documentation Connexe

- ← Retour à la Documentation Principale
- Référence de Configuration : tous les paramètres de configuration
- Guide des Fonctionnalités Courantes : Interface Web, API, surveillance
- Guide STP : routage et Traduction de Titre Global
- Guide HLR : traitement des opérations côté HLR
- Guide du Centre SMS : livraison de SMS
- Guide du Passerelle CAMEL : opérations CAP/CAMEL

---

**OmniSS7** par Omnitouch Network Services

# Guide de Configuration du Centre SMS (SMSc)

← Retour à la Documentation Principale

Ce guide fournit une configuration détaillée pour utiliser OmniSS7 comme un **Centre SMS (SMSc)** en frontend avec **OmniMessage** comme plateforme de stockage et de livraison des messages en backend.

# Intégration d'OmniMessage

**Le mode SMSc d'OmniSS7 fonctionne comme un frontend de signalisation SS7** qui s'interface avec **OmniMessage**, une plateforme SMS de niveau opérateur. Cette architecture sépare les préoccupations :

- **OmniSS7 (Frontend SMSc)** : Gère toute la signalisation des protocoles SS7/MAP, le routage SCCP et la communication réseau
- **OmniMessage (Backend SMS)** : Gère le stockage des messages, la mise en file d'attente, la logique de réessai, le suivi de livraison et les décisions de routage

## Pourquoi OmniMessage ?

OmniMessage fournit des capacités de messagerie SMS de niveau opérateur avec des fonctionnalités incluant :

- **Gestion de la File d'Attente des Messages** : Stockage persistant avec une logique de réessai configurable et une mise en file d'attente par priorité
- **Suivi de Livraison** : Statut de livraison en temps réel, rapports de livraison (DLR) et suivi des raisons d'échec
- **Support Multi-SMSc** : Plusieurs instances de frontend peuvent se connecter à un seul backend OmniMessage pour l'équilibrage de charge et la redondance

- **Intelligence de Routage** : Règles de routage avancées basées sur la destination, l'expéditeur, le contenu du message et l'heure de la journée
- **Limitation de Taux** : Contrôles TPS (transactions par seconde) par route pour prévenir la congestion du réseau
- **Conception API-First** : API HTTP RESTful pour l'intégration avec les systèmes de facturation, les portails clients et les applications tierces
- **Analytique & Reporting** : Statistiques sur le volume des messages, taux de succès de livraison et métriques de performance

Toutes les données de message, l'état de livraison et les configurations de routage sont stockées et gérées dans OmniMessage. OmniSS7 interroge OmniMessage via des appels API HTTPS pour récupérer les messages en attente, mettre à jour l'état de livraison et s'enregistrer en tant que frontend actif.

**Important** : Le mode SMSc d'OmniSS7 est un **frontend de signalisation uniquement**. Toute la logique de routage des messages, la gestion des files d'attente, les algorithmes de réessai, le suivi de livraison et les règles commerciales sont gérés par OmniMessage. Ce guide couvre la configuration du protocole SS7/MAP dans OmniSS7. Pour des informations sur le routage des messages, la configuration des files d'attente, les rapports de livraison, la limitation de taux et l'analytique, **reportez-vous à la documentation d'OmniMessage**.

# Table des Matières


1. Intégration d'OmniMessage
2. Qu'est-ce que le Mode Centre SMS ?
3. Activation du Mode SMSc
4. Configuration de l'API HTTP
5. Flux de Messages SMS
6. Gestion du Service Center d'Alerte
7. Prévention de Boucle
8. Suivi des Abonnés SMSc
9. Mise en Cache de l'Adresse VLR

10. Configuration de Flush Automatique
11. Métriques et Surveillance
12. Dépannage

---

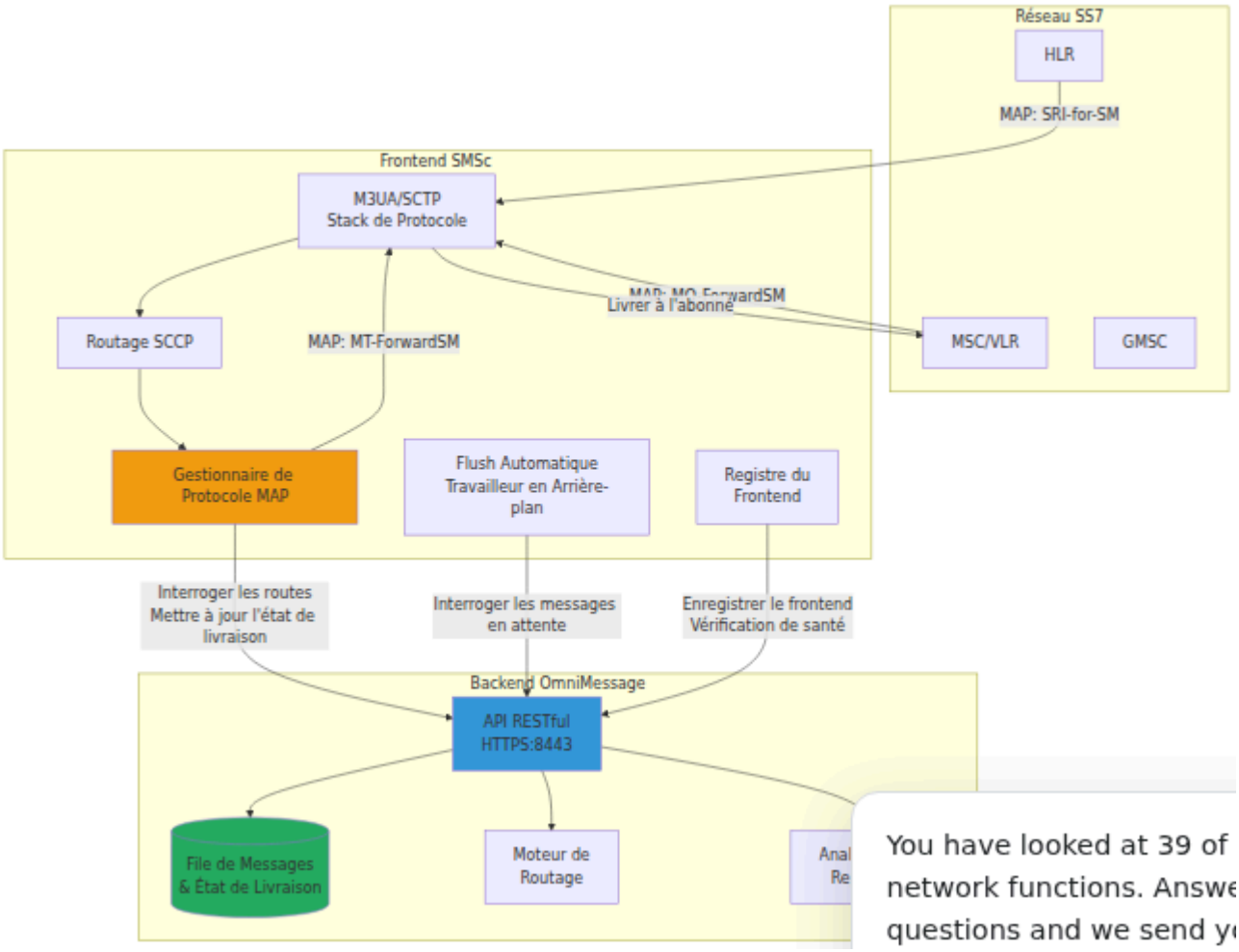
# Qu'est-ce que le Mode Centre SMS ?

**Remarque** : Cette section couvre uniquement la configuration de signalisation SS7 d'OmniSS7. Pour les règles de routage des messages, la gestion des files d'attente, le suivi de livraison et la configuration de la logique commerciale, consultez la **documentation produit d'OmniMessage**.

**Le Mode Centre SMS** permet à OmniSS7 de fonctionner comme un SMSc pour :

- **Livraison MT-SMS** : Livraison de SMS terminés sur mobile aux abonnés
- **Gestion MO-SMS** : Réception et routage de SMS d'origine mobile
- **Mise en File d'Attente des Messages** : File d'attente de messages soutenue par une base de données avec logique de réessai
- **Flush Automatique** : Livraison automatique de SMS depuis la file d'attente
- **Rapports de Livraison** : Suivi de l'état de livraison des messages

## Architecture du Centre SMS




---

# Activation du Mode SMSc

OmniSS7 peut fonctionner en différents modes (STP, HLR, SMSc). Le mode est sélectionné via un ensemble de drapeaux de mode et la configuration associée dans **`config/config.exs`**, le fichier de configuration par défaut de l'application.

## Changement en Mode SMSc

Pour exécuter OmniSS7 en tant que SMSc, définissez les drapeaux de mode afin que les fonctionnalités SMSc soient activées, puis ajoutez la configuration spécifique au SMSc :

1. **Ouvrir** `config/config.exs`

2. **Définir** les drapeaux de mode sous `config :omniss7` :
   - `map_client_enabled: true` : requis pour les capacités de routage/MAP
   - `smsc_mode_enabled: true` : active le traitement spécifique au SMSc
   - `hlr_mode_enabled: false` : désactive les fonctionnalités HLR
3. **Ajouter** les paramètres de configuration SMSc (point de terminaison API, nom du SMSc, adresse GT, paramètres de flush automatique, connexion M3UA) montrés ci-dessous
4. **Personnaliser** les paramètres pour votre déploiement
5. **Redémarrer** l'application pour que les modifications prennent effet

> **Remarque** : `config/runtime.exs` est un stub minimal utilisé uniquement pour les tests automatisés. Il ne contient **pas** de définitions de mode opérationnel. Toute la configuration de mode et de SMSc se trouve dans `config/config.exs` (avec des remplacements spécifiques à l'environnement dans des fichiers tels que `config/test.exs`). Les déploiements en production fournissent leur propre `config/config.exs`.

## Configuration du Mode SMSc

La configuration complète du SMSc (dans `config/config.exs`) ressemble à ceci :

```
config :omniss7,
  # Drapeaux de mode - Activer STP + fonctionnalités SMSc
  # Remarque : map_client_enabled est vrai car le SMSc a besoin de
capacités de routage
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: true,

  # Configuration de l'API Backend OmniMessage
  smsc_api_base_url: "https://omnimessage.example.com:8443",
  # Identification du SMSc pour l'enregistrement avec le backend
  smsc_name: "ipsmgw",
  # Adresse GT du Centre de Service pour les opérations SMS
  smsc_service_center_gt_address: "5551234567",

  # Configuration de Flush Automatique (traitement de la file
d'attente SMS en arrière-plan)
  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10,

  # Configuration de Connexion M3UA
  # Se connecter en tant qu'ASP pour envoyer/recevoir des
opérations MAP SMS
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    # Point de terminaison local (système SMSc)
    local_ip: {10, 179, 4, 12},
    local_port: 2905,
    # Point de terminaison STP distant
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

Les pages de l'interface web sont enregistrées séparément sous `config :control_panel` dans `config/config.exs`. La liste par défaut `use_additional_pages` enregistre toutes les pages d'OmniSS7 (~16 entrées) ;

le runtime filtre les pages qui ne s'appliquent pas au mode actif. Notez que la page de statut des pairs est étiquetée **"Pairs"** :

```
config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.TestClientLive, "/client", "Client SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "Pairs"},
    {SS7.Web.SCTPConnectionsLive, "/sctp", "Connexions SCTP"},
    {SS7.Web.RoutingLive, "/routing", "Routage"},
    {SS7.Web.RoutingTestLive, "/routing_test", "Test de Routage"},
    {SS7.Web.SmscLinksLive, "/smsc_links", "Liens SMSc"},
    {SS7.Web.SmscSubscribersLive, "/smsc_subscribers", "Abonnés
SMSc"},
    {SS7.Web.HlrLinksLive, "/hlr_links", "Liens HLR"},
    # ... pages supplémentaires (Abonnés Actifs, CAMEL, Journaux,
Test MSISDN/IMSI, etc.)
  ],
  page_order: [
    "/application", "/configuration", "/events", "/logs",
"/client", "/m3ua",
    "/routing", "/routing_test", "/smsc_links",
"/smsc_subscribers", "/hlr_links",
    # ... chemins restants ordonnés
  ]
```

## Paramètres de Configuration à Personnaliser

Pour une référence complète de tous les paramètres de configuration, consultez la Référence de Configuration.

| Paramètre | Type | Par Défaut | De |
|---|---|---|---|
| `smsc_api_base_url` | Chaîne | *Requis* | Point de API du b **OmniM** |
| `smsc_name` | Chaîne | `"{hostname}_SMSc"` | Votre id pour l'e |
| `smsc_service_center_gt_address` | Chaîne | *Requis* | Titre Gl de Serv |
| `auto_flush_enabled` | Booléen | `true` | Activer automa d'attent |
| `auto_flush_interval` | Entier | `10_000` | Interval traiteme d'attent millisec |
| `auto_flush_dest_smsc` | Chaîne | `nil` | Filtre de de desti flush au (`nil` = messag |
| `auto_flush_tps` | Entier | `10` | Taux de des mes (transac |
| `local_ip` | Tuple | *Requis* | Adresse système |
| `local_port` | Entier | `2905` | Port SC |

| Paramètre | Type | Par Défaut | De |
|---|---|---|---|
| `remote_ip` | Tuple | *Requis* | Adresse connect |
| `remote_port` | Entier | `2905` | Port SC |
| `routing_context` | Entier | `1` | ID de co routage |

## Que Se Passe-t-il Lorsque le Mode SMSc est Activé

Lorsque `smsc_mode_enabled: true` et `map_client_enabled: true`, l'interface web affichera :

- **Événements SS7** - Journalisation des événements
- **Client SS7** - Test des opérations MAP
- **Pairs** - État de connexion (pairs M3UA/SCTP)
- **Routage** - Gestion de la table de routage (STP activé)
- **Test de Routage** - Test de routage (STP activé)
- **Liens SMSc** - État de l'API SMSc + gestion de la file d'attente SMS ← *Spécifique au SMSc*
- **Ressources** - Surveillance du système
- **Configuration** - Visionneuse de configuration

L'onglet **Liens HLR** sera caché.

## Remarques Importantes

- Le mode SMSc nécessite `map_client_enabled: true` pour les capacités de routage
- **Backend OmniMessage** : L'API OmniMessage doit être accessible à l'URL `smsc_api_base_url` configurée

- **Enregistrement du Frontend** : Le système s'enregistre automatiquement auprès d'OmniMessage toutes les **~85 secondes** (`@registration_interval` = 85 000 ms) via le module `SMS.FrontendRegistry`
- **Délai d'Expiration des Requêtes API** : Toutes les requêtes API OmniMessage ont un **délai d'expiration codé en dur de 5 secondes**
- **Délai d'Expiration des Requêtes MAP** : SRI-for-SM a un délai d'expiration de dialogue de 10 secondes. MT-ForwardSM a un **délai d'expiration de dialogue de 30 secondes** pour accommoder les réponses lentes du VLR sur les réseaux 2G/CS
- Le flush automatique traite automatiquement la file d'attente SMS en arrière-plan
- La connexion M3UA au STP est requise pour envoyer/recevoir des opérations MAP SMS
- Après avoir changé de modes, vous devez redémarrer l'application pour que les changements prennent effet
- **Interface Web** : Consultez le Guide de l'Interface Web pour des informations sur l'utilisation de l'interface web
- **Accès API** : Consultez le Guide API pour la documentation de l'API REST et l'accès à Swagger UI

---

# Configuration de l'API HTTP

## Configuration du Backend OmniMessage

OmniSS7 communique avec OmniMessage via l'API REST HTTPS pour gérer la livraison des messages, suivre l'état des abonnés et s'enregistrer comme un frontend actif :

```
config :omniss7,
  # URL de base de l'API OmniMessage
  smsc_api_base_url: "https://omnimessage.example.com:8443",
  # Identifiant de nom de SMSc pour l'enregistrement (par défaut à
hostname_SMSc si vide)
  smsc_name: "omni-smsc01",
  # Adresse GT du Centre de Service pour les opérations SMS
  smsc_service_center_gt_address: "5551234567"
```

**Paramètres de Configuration :**

| Paramètre | Type | Requis | Par Défaut |
| --- | --- | --- | --- |
| `smsc_api_base_url` | Chaîne | Oui | `"https://localhos` |
| `smsc_name` | Chaîne | Non | `""` (utilise `"{hostname}_SMSc"`) |
| `smsc_service_center_gt_address` | Chaîne | Non | `"5551234567"` |

## Enregistrement du Frontend

Le système s'enregistre automatiquement auprès d'OmniMessage au démarrage et **se réenregistre toutes les ~85 secondes** via le module `SMS.FrontendRegistry`. Cela permet à OmniMessage de :

- Suivre les frontends actifs pour l'équilibrage de charge
- Surveiller le temps de fonctionnement et l'état de santé

- Collecter des informations de configuration
- Gérer le routage SMS distribué à travers plusieurs frontends

**Détails de Mise en Œuvre :**

- **Intervalle d'Enregistrement** : ~85 secondes (`@registration_interval 85_000`, codé en dur)
- **Processus** : Démarré automatiquement lorsque `smsc_mode_enabled: true`

**Payload d'Enregistrement :**

```
{
  "frontend_name": "omni-smsc01",
  "configuration": "{...}",
  "frontend_type": "SS7",
  "hostname": "smsc-server01",
  "uptime_seconds": 12345
}
```

**Remarque** : Le nom du frontend est pris du paramètre de configuration `smsc_name`. S'il n'est pas défini, il par défaut à `"{hostname}_SMSc"`.

## Communication API d'OmniMessage

Lorsque OmniSS7 reçoit des opérations MAP du réseau SS7 ou traite la file d'attente de messages, il communique avec OmniMessage pour :

- **S'enregistrer comme un frontend actif** et rapporter l'état de santé
- **Soumettre des messages d'origine mobile (MO)** reçus des abonnés
- **Récupérer des messages terminés sur mobile (MT)** de la file d'attente pour livraison
- **Mettre à jour l'état de livraison** avec des rapports de succès/échec
- **Interroger des informations de routage** pour le transfert de messages

| Endpoint | Méthode | But | Corps de l |
|---|---|---|---|
| `/api/frontends` | POST | Enregistrer l'instance du frontend | `{"frontend_n` `"configuratic` `"frontend_typ` `"hostname": "` `"uptime_secor` |
| `/api/messages_raw` | POST | Insérer un nouveau message SMS (TPDU brut) | `{"rp_origina` `"...",` `"rp_destinat` `"...", "direc` `"message_body` `"send_time":` `"smsc_node_na` `"source_smsc"` `"source_type"` |
| `/api/messages` | GET | Obtenir la file d'attente de messages | En-tête : `smsc` `<smsc_name>` |
| `/api/messages/{id}` | PATCH | Marquer le message comme livré | `{"deliver_ti` `"dest_smsc":` |
| `/api/messages/{id}` | PUT | Mettre à jour l'état du message | `{"dest_smsc"` |
| `/api/locations` | POST | Insérer/mettre à jour la localisation de l'abonné | `{"msisdn": "` `"...", "locat` `"ims_capable"` `"csfb": false` `"...", "user_` `"...", "ran_` `"...", "imei"` `"registered"` |

| Endpoint | Méthode | But | Corps de l |
|---|---|---|---|
| `/api/events` | POST | Ajouter le suivi des événements | `{"message_id` `"name": "..."` `"description"` |
| `/api/status` | GET | Vérification de santé (le chemin est configurable via `smsc_api_status_path`, par défaut `/api/status`; interrogé par la page Liens SMSc, pas par `SMSc.APIClient`) | - |

# Format de Réponse API

Toutes les réponses API utilisent le format JSON avec les conventions suivantes :

- **Réponses de succès** : HTTP 200-201 avec un corps JSON contenant les données de résultat
- **Réponses d'erreur** : HTTP 4xx/5xx avec des détails d'erreur dans le corps de la réponse
- **Horodatages** : format ISO 8601 (par exemple, `"2025-10-21T12:34:56Z"`)
- **IDs de message** : identifiants entiers ou chaînes

# Modules Client API

Le système SMS se compose de trois modules principaux :

### 1. SMSc.APIClient

Module principal du client API fournissant toute la communication HTTP API avec OmniMessage :

- `frontend_register/4` - Enregistrer le frontend auprès d'OmniMessage
- `insert_message/1` - Insérer un message SMS brut (prend une seule carte de message ; poste à `/api/messages_raw`)
- `insert_location/9` - Insérer/mettre à jour les données de localisation de l'abonné
- `get_message_queue/2` - Récupérer les messages en attente de la file d'attente
- `mark_dest_smsc/3` - Marquer le message comme livré ou échoué
- `add_event/3` - Ajouter le suivi des événements pour les messages
- `flush_queue/2` - Traiter les messages en attente (SRI-for-SM + MT-forwardSM)
- `auto_flush/2` - Boucle de traitement continu de la file d'attente

**2. SMS.FrontendRegistry**

Gère l'enregistrement périodique du frontend avec le backend :

- S'enregistre automatiquement au démarrage
- Se réenregistre toutes les ~85 secondes
- Utilise `smsc_name` de la config (retombe sur le nom d'hôte)
- Collecte des informations de configuration et de temps de fonctionnement du système

**3. SMS.Utils**

Fonctions utilitaires pour les opérations SMS :

- `generate_tp_scts/0` - Générer un horodatage SMS au format TPDU

---

# Flux de Messages SMS

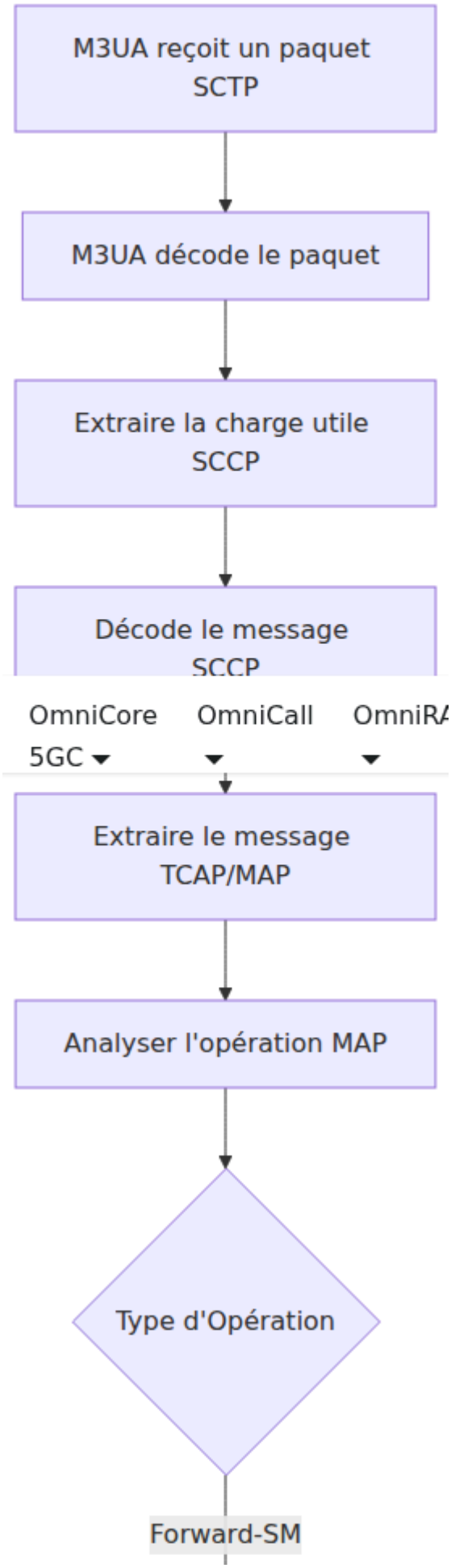
## Flux SMS Entrants (D'origine Mobile)

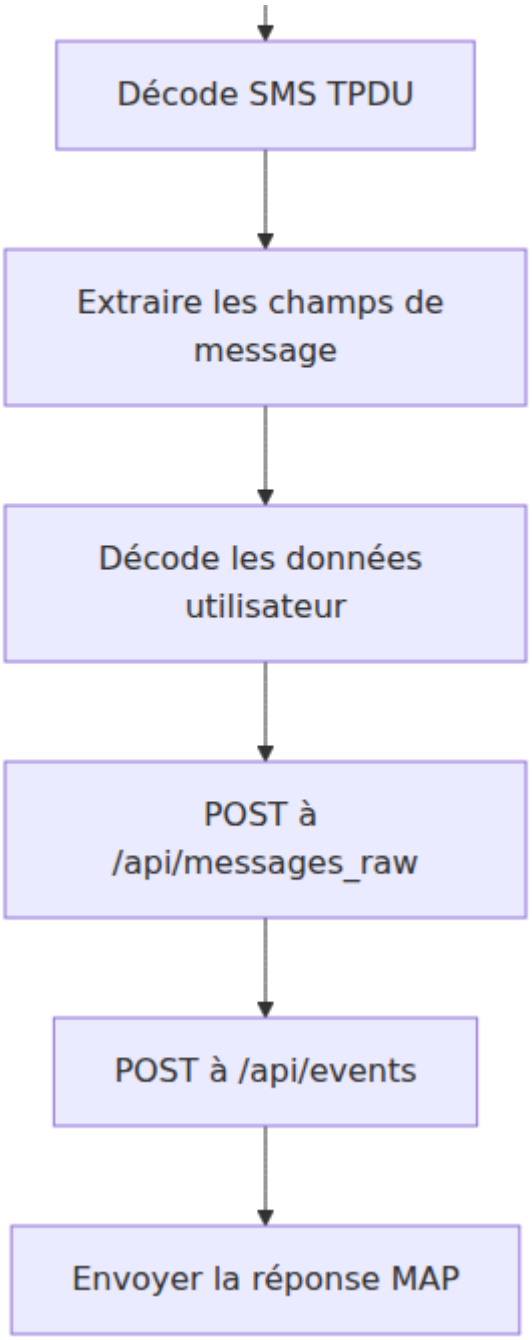

M3UA reçoit un paquet SCTP
M3UA décode le paquet
Extraire la charge utile SCCP
Décode le message SCCP
Extraire le message TCAP/MAP
Analyser l'opération MAP
Type d'Opération
Forward-SM




## Flux SMS Sortants (Terminés sur Mobile)

Lorsqu'une API HSS/HLR est configurée (`hlr_api_base_url`), le IP-SM-GW peut livrer des MT-SMS aux **abonnés sur le réseau** sans un aller-retour SRI-for-SM en interrogeant directement l'API HSS pour obtenir le véritable IMSI de l'abonné et l'adresse VLR de service. Un **cache VLR** dans le Suivi des Abonnés évite les recherches répétées dans le HSS pour le même abonné.


M3UA reçoit un paquet SCTP
M3UA décode le paquet
Extraire la charge utile SCCP
Décode le message SCCP
OmniCore 5GC
OmniCall
OmniRA
Extraire le message TCAP/MAP
Analyser l'opération MAP
Type d'Opération
Forward-SM




**Étapes Clés Expliquées** :

- **Vérification de Cache VLR** : Avant d'interroger l'API HSS, le système vérifie le cache VLR du Suivi des Abonnés pour une entrée valide (IMSI + adresse VLR dans le TTL). Un hit de cache évite complètement l'API HSS. Voir Mise en Cache de l'Adresse VLR pour plus de détails.

- **Livraison Directe via API HSS (sur réseau)** : Lorsque `hlr_api_base_url` est configuré, le IP-SM-GW interroge l'API HSS avec le MSISDN de destination pour obtenir le **véritable IMSI** et l'**adresse VLR de service** de l'abonné. Si les deux sont disponibles, le IP-SM-GW envoie le

MT-ForwardSM directement au VLR. Cela contourne complètement SRI-for-SM.

- **Abonné Absent (sur réseau)** : Si l'API HSS trouve l'abonné mais qu'aucun VLR de service n'est assigné, le message est marqué comme échoué avec une entrée de localisation expirée. Les réessais de livraison se produisent lorsque le HLR envoie un alertServiceCenter après que l'abonné se soit réenregistré.

- **Fallback SRI-for-SM (hors réseau)** : Lorsque l'API HSS n'est pas configurée ou que l'abonné n'est pas trouvé dans le HSS, le flux standard SRI-for-SM est utilisé. Le HLR répond avec un IMSI synthétique et un numéro de nœud réseau. Voir le Guide SRI-for-SM dans HLR.

```
- **VLR Effacer en cas d'échec** : Lorsque la livraison de MT-ForwardSM échoue (délai d'attente, erreur SCCP, etc.), l'adresse VLR mise en cache est effacée afin que la prochaine tentative interroge à nouveau l'API HSS pour un VLR frais.

- **Délai d'attente MT-ForwardSM** : Le dialogue MT-ForwardSM a un **délai d'attente de 30 secondes** pour tenir compte des réponses VLR 2G/CS plus lentes.

### Structure TPDU SMS

```mermaid
graph TB
    TPDU[SMS TPDU]
    TPDU --> MTI[Indicateur de Type de Message]
    TPDU --> Fields[Champs de Message]
    MTI --> DELIVER[SMS-DELIVER]
    MTI --> SUBMIT[SMS-SUBMIT]
    Fields --> OA[Adresse d'Origine]
    Fields --> DA[Adresse de Destination]
    Fields --> DCS[Schéma de Codage de Données]
    Fields --> UD[Données Utilisateur]
    DCS --> GSM7[GSM 7 bits<br/>160 caractères]
    DCS --> UCS2[UCS-2/Unicode<br/>70 caractères]
```

---

# Gestion du Centre de Service d'Alerte

Le SMSc peut recevoir des messages **alertServiceCenter** de l'HLR pour suivre l'état de disponibilité des abonnés.

Pour des informations sur la manière dont l'HLR envoie des messages alertServiceCenter, voir Intégration du Centre de Service d'Alerte dans le Guide HLR.
```

# Qu'est-ce que alertServiceCenter ?

Lorsqu'un abonné effectue une UpdateLocation à l'HLR (c'est-à-dire, s'enregistre avec un nouveau VLR/MSC), l'HLR peut notifier les systèmes SMSc que l'abonné est maintenant joignable en envoyant un message **alertServiceCenter** (opcode MAP 64).

# Configuration

Le temps d'expiration de la localisation est configuré dans l'HLR :

```
config :omniss7,
  # Temps d'expiration de la localisation lorsque le SMSc reçoit
alertServiceCenter (par défaut : 48 heures)
  hlr_alert_location_expiry_seconds: 172800
```

# Comportement

Lorsque le SMSc reçoit un message alertServiceCenter :

1. **Décoder MSISDN** : Extraire le MSISDN de l'abonné à partir du message (format TBCD)
2. **Supprimer le préfixe TON/NPI** : Retirer les préfixes courants comme "19", "11", "91" (par exemple, "19123123213" → "123123213")
3. **Résoudre IMSI** : Si `hlr_api_base_url` est configuré, interroger l'API HSS pour résoudre le **vrai IMSI** de l'abonné. Si la recherche HSS échoue ou qu'aucune API n'est configurée, revenir à la génération d'un IMSI synthétique en utilisant le même mappage que SRI-for-SM. Le vrai IMSI est critique pour la livraison MT sur le réseau à des abonnés 2G/CS. Le MSC/VLR nécessite le bon IMSI dans le MT-ForwardSM.
4. **POST à /api/locations** : Mettre à jour la base de données de localisation avec :
   - `msisdn` : Numéro de téléphone de l'abonné (nettoyé)
   - `imsi` : IMSI synthétique
   - `location` : Nom du SMSc (par exemple, "ipsmgw")
   - `expires` : Heure actuelle + `hlr_alert_location_expiry_seconds`

- `csfb` : true (abonné joignable via Circuit-Switched Fallback)
- `ims_capable` : false (il s'agit d'un enregistrement CS 2G/3G, pas d'IMS/VoLTE)
- `user_agent` : GT HLR qui a envoyé l'alerte (pour le suivi)
- `ran_location` : "SS7"

5. **Suivre dans le Suivi des Abonnés SMSc** : Enregistrer l'abonné avec GT HLR, statut=actif, compteurs de messages à 0
6. **Envoyer ACK** : Répondre à l'HLR avec un accusé de réception alertServiceCenter

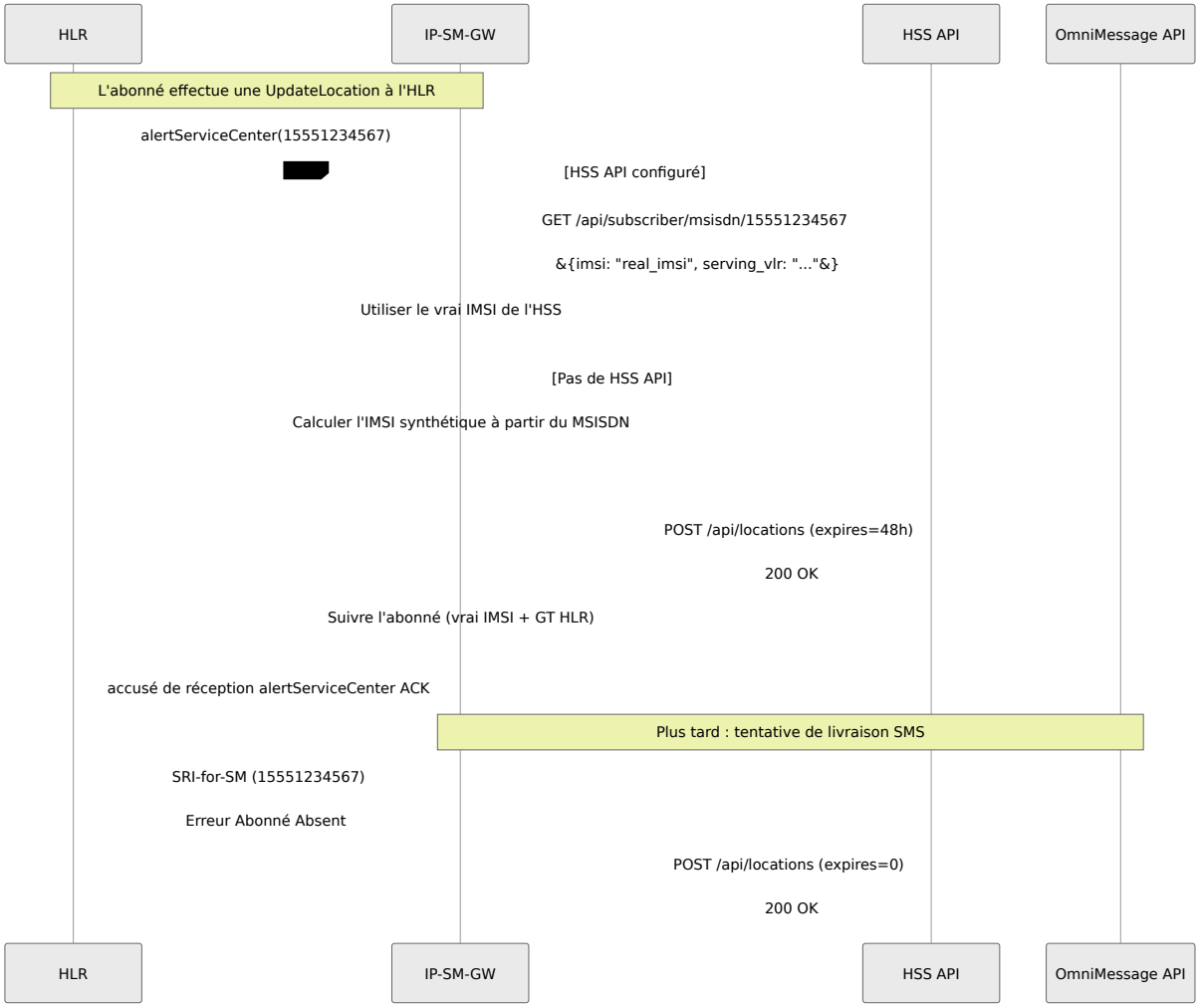
## Gestion des Abonnés Absents

Lorsque le SMSc tente de livrer un message et reçoit une erreur "abonné absent" pendant SRI-for-SM (pour plus d'informations sur SRI-for-SM, voir SRI-for-SM dans le Guide HLR) :

1. **Détecter l'absence** : SRI-for-SM retourne l'erreur `absentSubscriberDiagnosticSM`
2. **Expirer la localisation** : POST à /api/locations avec `expires=0` pour marquer l'abonné comme injoignable
3. **Agent utilisateur** : Défini sur "SS7_AbsentSubscriber" pour identifier la source
4. **Mettre à jour le suivi** : Marquer l'abonné comme `échoué` dans le Suivi des Abonnés SMSc

Cela garantit que la base de données de localisation et le suivi reflètent avec précision l'état de disponibilité des abonnés.

# Diagramme de Flux




# Point de terminaison API

## POST /api/locations

```
&amp;#123;
  "msisdn": "15551234567",
  "imsi": "001010123456789",
  "location": "ipsmgw",
  "ims_capable": false,
  "csfb": true,
  "expires": "2025-11-01T12:00:00Z",
  "user_agent": "15551111111",
  "ran_location": "SS7",
  "imei": "",
  "registered": "2025-10-30T12:00:00Z"
&amp;#125;
```


**Remarque** : Le champ `user_agent` contient le GT HLR qui a envoyé l'alertServiceCenter, permettant au SMSc de suivre quel HLR fournit les mises à jour de localisation.

Pour les abonnés absents, `expires` est défini sur l'heure actuelle (expiration immédiate).

---

# Prévention de Boucle

Le SMSc met en œuvre une **prévention automatique de boucle** pour éviter les boucles de routage de messages infinies lorsque les messages proviennent de réseaux SS7, tout en permettant la livraison légitime MO→MT entre abonnés sur le même réseau.

## Pourquoi la Prévention de Boucle est Importante

Lorsque le SMSc reçoit des messages SMS d'origine mobile (MO) du réseau SS7, il les insère dans la file d'attente de messages avec un champ `source_smsc` identifiant leur origine (par exemple, `"SS7_MO_15551234567"`). Sans prévention de boucle, ces messages pourraient être :

1. Reçus du réseau SS7 → Mis en file d'attente avec `source_smsc` contenant "SS7"
2. Récupérés de la file d'attente → Traitement pour livraison
3. Renvoyés au réseau SS7 → Créant une boucle

## Comment Cela Fonctionne

La logique de prévention de boucle distingue entre les destinations **sur le réseau** et **hors réseau**. Le SMSc bloque les messages provenant de SS7 uniquement lorsque l'abonné de destination **n'est pas sur le réseau**. Cela empêche les boucles de transit. Cela permet toujours la livraison légitime MO→MT entre les abonnés du même réseau (par exemple, MSC à MSC via l'IP-SM-GW).

Lexical error on line 2. Unrecognized text. ...e d'attente] --> B&fl° °123¶ßsource_smsc c ----------^

Réessayer

## Détection sur le Réseau

Lorsqu'un message provient de SS7, le système vérifie si l'abonné de destination est sur le réseau avant d'appliquer la prévention de boucle :

1. **Cache VLR** (le plus rapide) : Vérifier si l'abonné a une adresse VLR mise en cache avec un IMSI valide dans le Suivi des Abonnés
2. **API HSS** (autoritaire) : Si `hlr_api_base_url` est configuré et que le cache échoue, interroger l'API HSS par MSISDN. Si l'abonné existe dans l'HSS, il est sur le réseau.

Si aucune des vérifications ne trouve l'abonné, la destination est considérée comme **hors réseau** et le message est bloqué.

## Mise en œuvre

Lors du traitement des messages de la file d'attente, le SMSc vérifie le champ `source_smsc` :

- **Si `source_smsc` contient "SS7" et la destination est hors réseau** :
  - Le message est ignoré
  - Événement ajouté : `"Prévention de Boucle"` avec une description indiquant que la destination est hors réseau
  - Message marqué comme échoué via une requête PUT
  - Journalisé avec un niveau d'avertissement
- **Si `source_smsc` contient "SS7" et la destination est sur le réseau** :
  - Message traité normalement. Il s'agit d'un flux légitime MO→MT.
  - La livraison se poursuit via l'API HSS / chemin de cache VLR
- **Si `source_smsc` ne contient pas "SS7"** :
  - Message traité normalement (par exemple, messages de l'API web, SMPP ou d'autres sources non SS7)

## Valeurs Source SMSC

Les messages peuvent avoir diverses valeurs `source_smsc` :

| Source | Valeur Exemple | Action |
|---|---|---|
| Réseau SS7 (MO-FSM), dest hors réseau | `"SS7_MO_15551234567"` | **Ignoré** - Prévention de boucle |
| Réseau SS7 (MO-FSM), dest sur le réseau | `"SS7_MO_15551234567"` | **Autorisé** - Livraison MO→MT sur le réseau |
| API Externe/SMPP | `"ipsmgw"` ou `"api_gateway"` | Traité normalement |
| Autre SMSc | `"smsc-node-01"` | Traité normalement |

## Suivi des Événements

Lorsqu'un message est ignoré en raison de la prévention de boucle, un événement est enregistré :


```
&#123;
  "message_id": 12345,
  "name": "Prévention de Boucle",
  "description": "Message ignoré - source_smsc
'SS7_MO_15551234567' contient 'SS7' et la destination est hors
réseau, empêchant la boucle de message"
&#125;
```


Cet événement est visible dans :

- **Interface Web** : Page des Événements SS7 (`/events`)
- **Base de données** : Table `events` via API
- **Journaux** : Entrées de journal au niveau d'avertissement

## Configuration

La prévention de boucle est **toujours activée** et ne peut pas être désactivée. C'est une fonctionnalité de sécurité critique pour éviter les perturbations du réseau dues aux boucles de messages. Le contournement sur le réseau nécessite à la fois `smsc_mode_enabled: true` et soit le cache VLR soit `hlr_api_base_url` configuré.

## Scénarios d'Exemple

**Scénario 1** : MO-SMS de SS7 vers un abonn�� hors réseau (bloqué)

```
1. Téléphone mobile → MSC/VLR → IP-SM-GW (via MO-ForwardSM)
2. IP-SM-GW reçoit MO-FSM, insère dans la file d'attente :
source_smsc = "SS7_MO_9991900"
3. Auto-flush récupère le message de la file d'attente
4. Destination non trouvée dans le cache VLR ou HSS → hors réseau
5. IP-SM-GW détecte "SS7" dans source_smsc + dest hors réseau →
IGNORER
6. Événement enregistré : "Prévention de Boucle"
7. Aucun SRI-for-SM ou MT-ForwardSM envoyé (boucle empêchée)
```

**Scénario 2** : MO-SMS de SS7 vers un abonné sur le réseau (autorisé)

```
1. Téléphone mobile → MSC/VLR → IP-SM-GW (via MO-ForwardSM)
2. IP-SM-GW reçoit MO-FSM, insère dans la file d'attente :
source_smsc = "SS7_MO_9991900"
3. Auto-flush récupère le message de la file d'attente
4. Destination trouvée dans l'API HSS → abonné sur le réseau
5. IP-SM-GW autorise la livraison malgré l'origine SS7
6. HSS retourne le vrai IMSI + VLR → MT-ForwardSM envoyé
directement au VLR
```

---

# Suivi des Abonnés SMSc

Le SMSc comprend un **Suivi des Abonnés** GenServer qui maintient un état en temps réel pour les abonnés basé sur les messages alertServiceCenter et les tentatives de livraison de messages.

## Objectif

Le suivi fournit :

- **Surveillance de la disponibilité** : Quels abonnés sont actuellement joignables
- **Suivi HLR** : Quel HLR a envoyé l'alertServiceCenter pour chaque abonné
- **Comptes de messages** : Nombre de messages envoyés/reçus par abonné

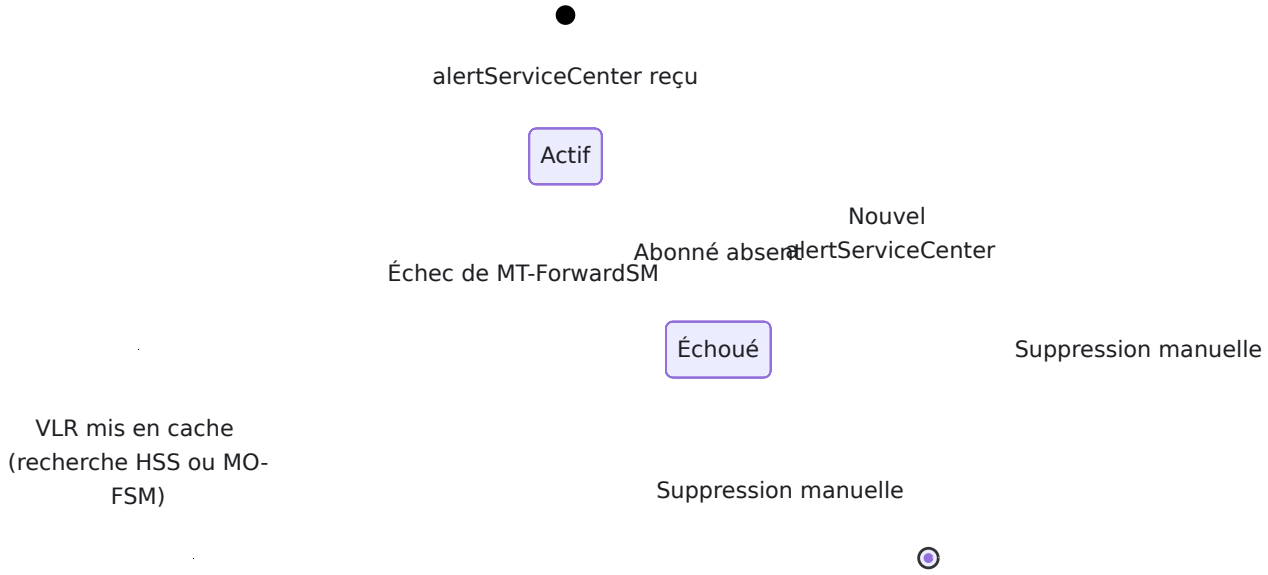
- **Suivi des échecs** : Marquer les abonnés comme échoués lorsque les tentatives de livraison échouent
- **Visibilité de l'Interface Web** : Tableau de bord en temps réel montrant tous les abonnés suivis

## Informations Suivies

Pour chaque abonné, le suivi stocke :

| Champ | Description | Exemple |
|---|---|---|
| `msisdn` | Numéro de téléphone de l'abonné (clé) | "15551234567" |
| `imsi` | IMSI de l'abonné (réel de l'HSS si disponible) | "001010123456789" |
| `hlr_gt` | GT HLR qui a envoyé alertServiceCenter | "15551111111" |
| `messages_sent` | Compte des messages MT-FSM envoyés | 5 |
| `messages_received` | Compte des messages MO-FSM reçus | 2 |
| `status` | `:actif` ou `:échoué` | `:actif` |
| `updated_at` | Horodatage Unix de la dernière mise à jour | 1730246400 |
| `vlr_address` | Adresse VLR de service mise en cache | "14155550100" |
| `vlr_cached_at` | Horodatage Unix lorsque le VLR a été mis en cache pour la dernière fois | 1730246400 |

# Transitions d'État




# Comportement

### Lorsqu'un alertServiceCenter est reçu :

- Créer ou mettre à jour l'entrée de l'abonné
- Définir `status = :actif`
- Enregistrer le GT HLR
- Résoudre le vrai IMSI à partir de l'API HSS (si configuré), sinon utiliser l'IMSI synthétique
- Préserver le cache VLR existant et les compteurs de messages

### Lorsque l'API HSS retourne un abonné avec VLR :

- Mettre en cache l'adresse VLR et le vrai IMSI dans le suivi
- Les livraisons ultérieures utilisent le VLR mis en cache sans interroger à nouveau l'HSS

### Lorsque MO-ForwardSM est reçu :

- Mettre en cache le GT appelant SCCP comme l'adresse VLR de l'abonné (le GT appelant sur un MO-ForwardSM est le VLR qui a relayé le message)
- Crée une entrée de suivi si elle n'existe pas

**Lorsque la livraison MT-ForwardSM échoue :**

- Effacer l'adresse VLR mise en cache afin que la prochaine tentative de livraison interroge à nouveau l'API HSS

**Lorsque SRI-for-SM réussit :**

- Incrémenter le compteur `messages_sent`
- Mettre à jour l'horodatage `updated_at`

**Lorsque SRI-for-SM échoue :**

- Définir `status = :échoué`
- Garder dans le suivi pour surveillance

**Lorsque l'abonné est supprimé :**

- Supprimer de la table ETS
- N'apparaît plus dans l'Interface Web

# Interface Web - Page des Abonnés SMSc

**Chemin** : `/smsc_subscribers` **Actualisation automatique** : Toutes les 2 secondes

**Remarque** : Cette page n'est disponible que lors de l'exécution en mode SMSc. Après avoir activé la configuration SMSc dans `config/config.exs`, vous devez redémarrer l'application pour que la route devienne disponible.

La page **Abonnés SMSc** fournit une surveillance en temps réel de tous les abonnés suivis :

**Fonctionnalités**

1. **Tableau des Abonnés**
    - MSISDN, IMSI, GT HLR, GT VLR (adresse VLR de service mise en cache)
    - Compteurs de messages envoyés/reçus
    - Badge de statut (Actif/Échoué) avec codage couleur

- Horodatage de dernière mise à jour et durée
- Bouton de suppression pour les abonnés individuels

2. **Statistiques Résumées**

   - Total des abonnés suivis
   - Compte des abonnés actifs
   - Compte des abonnés échoués
   - Nombre de HLR uniques

3. **Actions**

   - Tout Effacer : Supprimer tous les abonnés suivis
   - Supprimer : Supprimer un abonné individuel

**Exemple de Vue**

```
┌──────────────────────────────────────────────────────────────────────────┐
│ Abonnés Suivis SMSc                                    Total : 3         │
├──────────────────────────────────────────────────────────────────────────┤
│ MSISDN          IMSI              GT HLR         GT VLR         Msg      │
│                                                                  S/      │
├──────────────────────────────────────────────────────────────────────────┤
│ 15551234567     001010123456789   15551111111    14155550100    5/2     │
│ 15559876543     001010987654321   15551111111    -              0/0     │
│ 15551112222     001010111222233   15552222222    14155550200    3/1     │
└──────────────────────────────────────────────────────────────────────────┘

Résumé :  Total : 3  |  Actif : 2  |  Échoué : 1  |  HLRs Uniques : 2
```

# Fonctions API

Le suivi expose ces fonctions pour un accès programmatique :

```
# Appelé lorsque alertServiceCenter est reçu
SMSc.SubscriberTracker.alert_received(msisdn, imsi, hlr_gt)

# Incrémenter les compteurs de messages
SMSc.SubscriberTracker.message_sent(msisdn)
SMSc.SubscriberTracker.message_received(msisdn)

# Marquer comme échoué (échec SRI-for-SM)
SMSc.SubscriberTracker.mark_failed(msisdn)

# Supprimer du suivi
SMSc.SubscriberTracker.remove_subscriber(msisdn)

# Fonctions de requête
SMSc.SubscriberTracker.get_active_subscribers()
SMSc.SubscriberTracker.get_subscriber(msisdn)
SMSc.SubscriberTracker.count_subscribers()
SMSc.SubscriberTracker.clear_all()
```

## Intégration

Le suivi est automatiquement intégré avec :

- **Gestionnaire alertServiceCenter** : Appelle `alert_received/3` lors de la mise à jour de localisation réussie
- **Gestionnaire SRI-for-SM** : Incrémente `messages_sent` lors du routage réussi
- **Gestionnaire Abonné Absent** : Appelle `mark_failed/1` lorsque l'abonné est absent
- **Erreurs d'abonné inconnu** : Appelle `mark_failed/1` lorsque SRI-for-SM échoue

---

# Mise en Cache de l'Adresse VLR

Lorsqu'il fonctionne en tant qu'IP-SM-GW avec une API HSS configurée, le Suivi des Abonnés met en cache les adresses VLR pour éviter d'interroger l'HSS à chaque tentative de livraison MT.

## Comment Cela Fonctionne

Le cache VLR est peuplé à partir de deux sources :

1. **Recherche API HSS** : Lorsque le pipeline de livraison interroge l'API HSS et obtient un abonné avec un VLR de service, l'IMSI et l'adresse VLR sont mis en cache dans le Suivi des Abonnés.
2. **MO-ForwardSM** : Lorsqu'un MO-ForwardSM arrive d'un abonné, le GT de la partie appelante SCCP (qui est le VLR/MSC qui a relayé le message) est mis en cache comme l'adresse VLR de l'abonné.

Lors des tentatives de livraison MT ultérieures, le cache est vérifié en premier. Si une entrée valide existe (IMSI et VLR présents, dans le TTL), l'API HSS est complètement ignorée.

## Invalidité du Cache

Le cache VLR est effacé lorsque :

- **La livraison MT-ForwardSM échoue** (délai d'attente, erreur SCCP, rejet VLR) : le VLR mis en cache peut être obsolète
- **Le TTL expire** : le TTL est configurable via `vlr_cache_ttl_seconds`. Le cache considère les entrées plus anciennes que le TTL comme des échecs de cache.

Le cache **n'est pas** effacé lors de alertServiceCenter. L'entrée VLR existante est préservée car l'abonné peut toujours être joignable au même VLR.

## Configuration

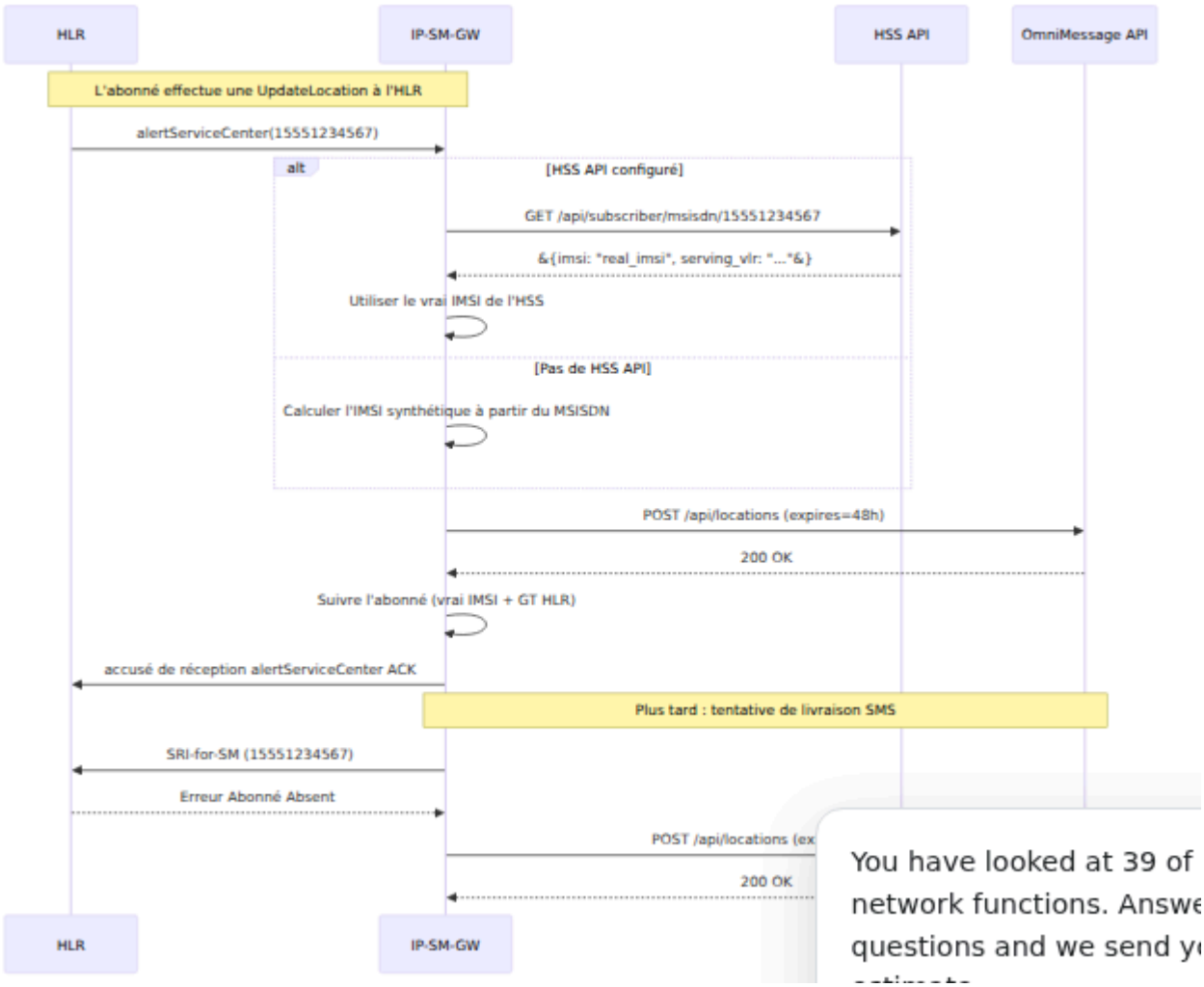
```
config :omniss7,
  # Point de terminaison API HSS/HLR (nécessaire pour la mise en
cache VLR et la livraison sur le réseau)
  hlr_api_base_url: "https://omnihss.example.com:8443",
  # Clé API OmniHSS (nécessaire lorsque hlr_api_base_url est
défini)
  hlr_api_key: "&lt;Clé API OmniHSS avec le rôle
hlr_interworking&gt;",

  # TTL du cache VLR en secondes (par défaut : 3600 = 1 heure)
  vlr_cache_ttl_seconds: 3600
```

| Paramètre | Type | Requis | Par défaut | Desc |
|---|---|---|---|---|
| `hlr_api_base_url` | Chaîne | Non | `nil` | URL de b l'API HSS Lorsqu'e définie, la reche d'abonn réseau, l cache VI livraison directe. `nil`, tou livraison le flux SI |
| `hlr_api_key` | Chaîne | Lorsque `hlr_api_base_url` est défini | `nil` | Clé API C OmniSS dans l'en `omnihss` chaque r API HSS. être la c qu'Omni attribue `hlr_int` Si `hlr_api` est défin clé est n ou vide, requête échoue a erreur n `hlr_api` |

| Paramètre | Type | Requis | Par défaut | Desc |
|---|---|---|---|---|
| `vlr_cache_ttl_seconds` | Entier | Non | `3600` | Âge max seconde qu'une e mise en considér valide. A période, prochain tentative livraison à nouvea HSS. Des plus bas augmen trafic de mais am précisior abonnés |

## Diagramme de Séquence




---

# Auto-Flush de la File d'Attente SMS

Le service **Auto-Flush** traite automatiquement les messages SMS en attente.

Pour référence des paramètres de configuration, voir Configuration Auto-Flush dans la Référence de Configuration.

## Configuration

```
config :omniss7,
  auto_flush_enabled: true,           # Activer/désactiver l'auto-
flush
  auto_flush_interval: 10_000,        # Intervalle de sondage en
millisecondes
  auto_flush_dest_smsc: nil,          # Filtre : nil = tous
  auto_flush_tps: 10                  # Transactions maximales par
seconde
```

## Comment Cela Fonctionne

1. **Sondage** : Toutes les `auto_flush_interval` millisecondes, interroge l'API pour les messages en attente
2. **Filtrage** : Filtrer éventuellement par `auto_flush_dest_smsc`
3. **Limitation de Taux** : Traiter jusqu'à `auto_flush_tps` messages par cycle
4. **Livraison** : Pour chaque message :
    - Envoyer **SRI-for-SM** (Send Routing Info for Short Message) à l'HLR pour obtenir les informations de routage
        - L'HLR retourne un IMSI synthétique calculé à partir du MSISDN
        - L'HLR retourne l'adresse GT SMSC où MT-ForwardSM doit être envoyé
        - Voir Détails SRI-for-SM dans le Guide HLR pour la documentation complète
    - En cas de succès, envoyer **MT-forwardSM** au MSC/VLR
    - Mettre à jour le statut du message via l'API (livré/échec)
    - Ajouter le suivi des événements via l'API

```
> **□ Plongée Technique** : Pour une explication complète de la façon
dont SRI-for-SM fonctionne, y compris le mappage MSISDN vers IMSI, la
configuration de l'adresse GT du centre de services, et la génération
d'IMSI synthétiques préservant la vie privée, consultez la [section
SRI-for-SM dans le Guide de Configuration HLR]
(./HLR_GUIDE.md#sendroutinginfoforsm-sri-for-sm-response).

---

## Métriques SMSc

OmniSS7 expose des métriques Prometheus. Les métriques MAP partagées
suivent l'activité SMS. Il n'y a pas de compteurs de file
d'attente/livraison spécifiques à SMSc. Les métriques pertinentes pou
les opérations SMS sont :

| Métrique | Type | Étiquettes | Pertinence pour SMS |
|----------|------|------------|---------------------|
| `map_requests_total` | Compteur | `operation` | Compte les
opérations MAP. Les opérations pertinentes pour SMS incluent
`sri_for_sm`, `forward_sm`, `MTforward_sm`, et `prn`. |
| `map_request_duration_milliseconds` | Histogramme | `operation` |
Durée des cycles de demande/réponse MAP, y compris SRI-for-SM et MT-
ForwardSM. |

L'ensemble complet des métriques déclarées (pas toutes spécifiques à
SMS) est : `map_requests_total`, `cap_requests_total`,
`ussd_requests_total`, `ussd_active_sessions`, `map_pending_requests`
`omniss7_license_status`, et `map_request_duration_milliseconds`.

**Exemples de Requêtes :**

```promql
# Taux d'opérations MT-ForwardSM au cours des 5 dernières minutes
rate(map_requests_total{operation="MTforward_sm"}[5m])

# Taux d'opérations SRI-for-SM au cours des 5 dernières minutes
rate(map_requests_total{operation="sri_for_sm"}[5m])

# Durée moyenne de MT-ForwardSM (dernières 5 minutes)
rate(map_request_duration_milliseconds_sum{operation="MTforward_sm"}
[5m]) /
```
```

```
rate(map_request_duration_milliseconds_count{operation="MTforward_sm"
[5m])
```

---

# Dépannage SMSc

## Problème : Messages Non Livrés

**Vérifications :**

1. Vérifiez que l'auto-flush est activé
2. Vérifiez la connexion à la base de données
3. Surveillez les journaux pour des erreurs
4. Vérifiez que la connexion M3UA est ACTIVE
5. Vérifiez les limites TPS

## Problème : Profondeur de File d'Attente Élevée

**Causes Possibles :**

- Limite TPS trop basse
- Problèmes de délai d'attente HLR
- Problèmes de connectivité réseau
- Numéros de destination invalides

**Solutions :**

- Augmentez `auto_flush_tps`
- Vérifiez la disponibilité du HLR
- Examinez les journaux des messages échoués

---

# API MT-forwardSM

## Envoyer un SMS via l'API

**Point de terminaison API :** `POST /api/MT-forwardSM`

Les quatre champs sont requis. Les noms de champs sont en snake_case (notez que `destination_service_centre` utilise l'orthographe britannique). `smsPDU` est le TPDU SMS encodé en hexadécimal.

**Requête :**

```
{
  "imsi": "234509876543210",
  "destination_service_centre": "447999555111",
  "originating_service_center": "447999123456",
  "smsPDU":
"040B917477218345F60000157030185714 0C0BD4F29C0E9281C4E1F11A"
}
```

**Réponse :**

En cas de succès (HTTP 200), la réponse contient une seule clé `result` contenant le résultat TCAP formaté du dialogue MT-ForwardSM (il n'y a pas de `message_id`, et la valeur n'est pas la chaîne littérale `"success"`):

```
{
  "result": "<formatted TCAP result>"
}
```

Si le dialogue MAP expire, le point de terminaison renvoie HTTP 504 avec `{"error": "timeout"}`. Une requête malformée/manquante renvoie HTTP 400 avec `{"error": "invalid request"}`.

---

# Documentation Connexe

**Documentation OmniSS7 :**

- ← Retour à la Documentation Principale
- Guide de Configuration HLR - Configuration et opérations en mode HLR
  - **Détails Techniques SRI-for-SM** - Documentation complète sur le mappage MSISDN vers IMSI et la configuration du centre de services
- Guide des Fonctionnalités Communes - Interface Web, API, Surveillance
- Guide Client MAP - Opérations MAP
- Référence Technique - Spécifications des protocoles

**Documentation OmniMessage :** Pour la configuration du routage des messages, la gestion des files d'attente, le suivi des livraisons, la limitation de débit, et l'analytique, consultez la **documentation produit OmniMessage**. OmniMessage contient toute la logique de routage des messages, les algorithmes de réessai de file d'attente, la gestion des rapports de livraison, et le moteur de règles métier.

---

# Guide de Configuration STP M3UA & M2PA

← Retour à la Documentation Principale

Ce guide fournit des configurations détaillées pour utiliser OmniSS7 en tant que **Point de Transfert de Signalisation (STP)**.

## Table des Matières


1. Qu'est-ce qu'un STP ?
2. Rôles Réseau STP
3. Interface avec les Réseaux TDM
4. Activation du Mode STP
5. Configuration des Pairs
6. Support du Protocole M2PA
    - M3UA vs M2PA
    - Configuration des Pairs M2PA
    - Configuration SLTM
    - Gestion de M2PA via l'Interface Web
    - Métriques M2PA
7. Routage par Code de Point
8. Routage par Titre Global
9. Gestion du Transit SCCP
    - Route-sur-SSN Après GTT
    - Application du Compteur de Saut
    - Retour de Message en Cas d'Échec
10. État de Destination (Gestion de Route de Signalisation)
11. Filtrage de Passerelle (Pare-feu SCCP)
12. Fonctionnalités de Gestion de Route
    - Désactivation des Routes

    - DROP Routes - Prévention des Boucles de Routage
13. Routage Avancé
14. Tester la Configuration
15. Métriques et Surveillance
16. Surveillance des Pairs M3UA
    - Gestion du Contexte de Routage M3UA
    - Désambiguïsation Multi-Pair Same-IP


---

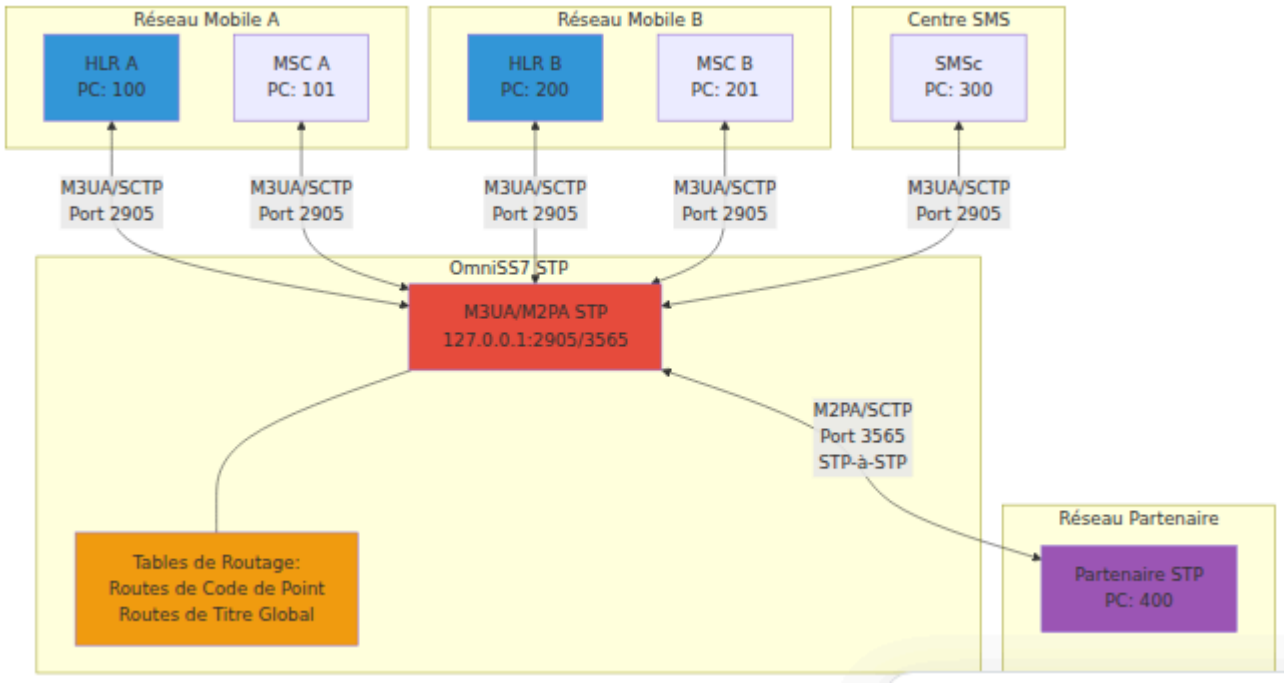
# Qu'est-ce qu'un Point de Transfert de Signalisation (STP) ?

Un **Point de Transfert de Signalisation (STP)** est un élément réseau critique dans les réseaux de signalisation SS7 et basés sur IP qui achemine les messages de signalisation entre les nœuds du réseau.

## Fonctions du STP

- **Routage de Message** : Achemine le trafic de signalisation SS7 basé sur le Code de Point de Destination (PC) ou le Titre Global (GT)
- **Traduction de Protocole** : Relie les réseaux SS7 traditionnels avec les réseaux M3UA/SCTP basés sur IP
- **Distribution de Charge** : Distribue le trafic entre plusieurs destinations en utilisant un routage basé sur la priorité
- **Passerelle Réseau** : Connecte différents réseaux de signalisation et fournisseurs de services
- **Masquage de Topologie** : Peut réécrire des adresses pour masquer la topologie interne du réseau

## Diagramme Réseau STP




---

# Rôles Réseau STP Expliqués

## ASP (Processus de Serveur d'Application)

- **Rôle** : Client se connectant à un SGP/STP distant

- **Direction** : Connexion sortante
- **Cas d'Utilisation** : Votre STP se connecte au STP d'un réseau partenaire

## SGP (Processus de Passerelle de Signalisation)

- **Rôle** : Serveur acceptant les connexions des ASP
- **Direction** : Connexion entrante
- **Cas d'Utilisation** : Les réseaux partenaires se connectent à votre STP

## AS (Serveur d'Application)

- **Définition** : Regroupement logique d'un ou plusieurs ASP
- **But** : Fournit redondance et partage de charge
- **Cas d'Utilisation** : Plusieurs ASP servant la même destination

---

# Interface avec les Réseaux TDM via des Passerelles de Signalisation

OmniSS7 est une plateforme de signalisation basée sur IP qui utilise les protocoles SIGTRAN (M3UA/M2PA sur SCTP). Pour échanger des signaux avec des **réseaux SS7 basés sur TDM** hérités, vous avez besoin d'une **Passerelle de Signalisation (SGW)** pour faire le pont entre les deux mondes.

## Qu'est-ce qu'une Passerelle de Signalisation ?

Une **Passerelle de Signalisation (SGW)** est un élément réseau qui convertit la signalisation SS7 entre :

- **Côté TDM** : SS7 traditionnel utilisant MTP1/MTP2/MTP3 sur des liaisons E1/T1
- **Côté IP** : SIGTRAN utilisant M3UA ou M2PA sur SCTP/IP

La SGW agit comme un traducteur de protocole. Elle permet aux applications basées sur IP comme OmniSS7 de communiquer avec des équipements TDM

hérités (commutateurs, STP, HLR) qui ne prennent pas en charge la signalisation IP.

## Architecture TDM vers IP




## Comparaison de la Pile de Protocoles

| Couche | TDM SS7 | Passerelle de Signalisation | IP (SIGTRAN) |
|---|---|---|---|
| **Partie Utilisateur** | SCCP/TCAP/MAP | ← Transparent → | SCCP/TCAP/MAP |
| **Réseau** | MTP3 | Conversion | M3UA/M2PA |
| **Liaison** | MTP2 | Terminaison | SCTP |
| **Physique** | MTP1 (E1/T1) | Terminaison | IP/Ethernet |

La SGW termine MTP1/MTP2 côté TDM et présente les données utilisateur MTP3 sur M3UA ou M2PA côté IP. Les couches supérieures (SCCP, TCAP, MAP, CAP) passent à travers de manière transparente.

## Connexion d'OmniSS7 à une Passerelle de Signalisation

OmniSS7 se connecte à une Passerelle de Signalisation en tant qu'**ASP (Processus de Serveur d'Application)**, tandis que la SGW agit en tant que

**SGP (Processus de Passerelle de Signalisation)**.

## Exemple de Configuration

```
config :omniss7,
  # Se connecter à la Passerelle de Signalisation en tant qu'ASP
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :sgw_connection,
    # Point de terminaison local (OmniSS7)
    local_ip: {10, 0, 0, 1},
    local_port: 0,                    # Port local dynamique
    # Point de terminaison de la Passerelle de Signalisation
    remote_ip: {10, 0, 0, 100},       # Adresse IP de la SGW
    remote_port: 2905,                # Port M3UA de la SGW
    routing_context: 1                # Assigné par la SGW
  },

  # Ou configurer comme un pair pour le routage STP
  peers: [
    %{
      peer_id: 1,
      name: "TDM_Gateway",
      role: :client,                  # OmniSS7 initie la
connexion
      local_ip: {10, 0, 0, 1},
      local_port: 0,
      remote_ip: {10, 0, 0, 100},     # Adresse IP de la SGW
      remote_port: 2905,
      routing_context: 1,
      point_code: 100,                # Plage de code de point
derrière la SGW
      network_indicator: :international
    }
  ],

  # Routage des codes de point du réseau TDM via la passerelle
  m3ua_routes: [
    %{
      dest_pc: 100,                   # Code de point MSC TDM
      peer_id: 1,                     # Route via le pair SGW
      priority: 1,
      network_indicator: :international
    },
    %{
      dest_pc: 200,                   # Code de point HLR TDM
```

```
      peer_id: 1,
      priority: 1,
      network_indicator: :international
    }
  ]
```

# Considérations de Configuration SGW

Lors de la configuration de votre Passerelle de Signalisation pour fonctionner avec OmniSS7 :

| Paramètre | Description | Valeur Typique |
|---|---|---|
| **Adresse IP Distant** | Adresse IP d'OmniSS7 | Votre adresse IP de serveur |
| **Port Distant** | Port SCTP d'OmniSS7 | 2905 |
| **Contexte de Routage** | Identifiant AS à la SGW | Assigné par l'administrateur SGW |
| **Codes de Point** | Codes de point du réseau TDM accessibles via SGW | Spécifique au réseau |
| **Mode de Trafic** | Mode de gestion du trafic ASP | Override ou Loadshare |

# Scénarios de Déploiement

### Scénario 1 : Application IP Accédant au Réseau TDM

OmniSS7 envoie des requêtes MAP (SRI-SM, PRN) aux HLR basés sur TDM via la SGW :

```
OmniSS7 (ASP) → SGW (SGP) → Réseau TDM → HLR
     M3UA          MTP2           MTP3
```

### Scénario 2 : OmniSS7 comme STP Entre IP et TDM

OmniSS7 achemine le trafic entre les éléments réseau basés sur IP et les réseaux TDM :

```
SMSC IP → OmniSS7 STP → SGW → HLR TDM
              ↓
         HLR basé sur IP
```

### Scénario 3 : SGW Double pour Redondance

Connectez-vous à deux Passerelles de Signalisation pour une haute disponibilité :

```
peers: [
  %{
    peer_id: 1,
    name: "SGW_Primary",
    role: :client,
    remote_ip: {10, 0, 0, 100},
    remote_port: 2905,
    point_code: 100,
    # ... autre config
  },
  %{
    peer_id: 2,
    name: "SGW_Backup",
    role: :client,
    remote_ip: {10, 0, 0, 101},
    remote_port: 2905,
    point_code: 100,
    # ... autre config
  }
],

m3ua_routes: [
  # Route principale via SGW_Primary
  %{dest_pc: 100, peer_id: 1, priority: 1, network_indicator:
:international},
  # Route de secours via SGW_Backup
  %{dest_pc: 100, peer_id: 2, priority: 2, network_indicator:
:international}
]
```

---

# Activation du Mode M3UA STP

OmniSS7 peut fonctionner en différents modes. Pour l'utiliser comme un STP, vous devez fournir une configuration de routage STP sous la clé d'application `:omniss7`.

## Où se Trouve la Configuration

| Fichier | But |
|---|---|
| `config/config.exs` | Configuration par défaut, toujours chargée (interface web, écouteur API sur le port 8445, journalisation) |
| `config/stp_example_runtime.exs` | Un **exemple STP** complet et prêt à éditer (`sctp_handler`, pairs, routes PC, routes GT) |
| `config/runtime.exs` | Remplacements d'exécution par déploiement. Dans ce dépôt, c'est un minimal stub ; les serveurs de production déploient les leurs |

> **Remarque** : Les versions antérieures contenaient trois blocs de mode commentés à l'intérieur de `config/runtime.exs`. Cette mise en page n'existe plus. L'exemple STP se trouve désormais dans son propre fichier, `config/stp_example_runtime.exs`.

## Passer au Mode STP

Pour mettre en place un STP, copiez la configuration d'exemple dans la configuration que votre déploiement charge à l'exécution :

1. **Ouvrez** `config/stp_example_runtime.exs` et examinez l'exemple STP (gestionnaire SCTP, pairs, routes PC, routes GT).
2. **Copiez** le bloc `config :omniss7, ...` dans le `config/runtime.exs` de votre déploiement (ou fusionnez les clés pertinentes).
3. **Personnalisez** les adresses IP, ports, codes de point, pairs et routes pour votre réseau.
4. **Redémarrez** l'application : `iex -S mix`

## Configuration du Mode STP

L'exemple STP dans `config/stp_example_runtime.exs` configure un écouteur STP entrant (`sctp_handler`), active le routage GT et définit des pairs ainsi que des routes PC et GT :

```
config :omniss7,
  # Écouteur SCTP (serveur STP entrant) - voir le Mode STP
Autonome ci-dessous
  sctp_handler: %{
    enabled: true,
    local_ip: {127, 0, 0, 1},
    local_port: 2905,
    point_code: 101            # Code de point de ce STP
  },

  # Activer le routage par Titre Global (en plus du routage par
PC)
  enable_gt_routing: true,

  # Les pairs M3UA, routes PC et routes GT suivent
  m3ua_peers: [ %{peer_id: 1, name: "STP_West", role: :client,
...} ],
  m3ua_routes: [ %{dest_pc: 100, peer_id: 1, priority: 1,
network_indicator: :international} ],
  m3ua_gt_routes: [ %{gt_prefix: "44", peer_id: 2, priority: 1,
description: "Numéros UK"} ]
```

Voir Configuration des Pairs M3UA, Routage par Code de Point, et Routage par Titre Global ci-dessous pour l'ensemble des paramètres.

## Paramètres de Configuration à Personnaliser

Pour une référence complète de tous les paramètres de configuration, voir la Référence de Configuration.

Voici les clés STP de niveau supérieur sous `config :omniss7` :

| Paramètre | Type | Par Défaut | Description | Exemple |
|---|---|---|---|---|
| `sctp_handler` | Map | *Optionnel* | Configuration de l'écouteur STP entrant (voir Mode STP Autonome) | `%{enabled: true, local_port: 2905, point_code: 101}` |
| `enable_gt_routing` | Boolean | `false` | Activer le routage par Titre Global (en plus du routage par PC) | `true` |
| `m3ua_peers` | List | `[]` | Définitions des pairs M3UA / M2PA (voir Configuration des Pairs M3UA) | Aucun |
| `m3ua_routes` | List | `[]` | Routes de Code de Point (voir Routage par Code de Point) | Aucun |
| `m3ua_gt_routes` | List | `[]` | Routes de Titre Global (voir Routage par Titre Global) | Aucun |

> **Astuce** : Utilisez le multihoming SCTP en fournissant une liste d'adresses IP pour `local_ip` et/ou `remote_ip` pour activer le basculement automatique. Voir Guide de Multihoming SCTP.

## Que Se Passe-t-il Lorsque le Mode STP est Activé

Une fois que les pairs et les routes sont configurés, l'interface web affiche :

- **Événements SS7** - Journalisation des événements
- **M3UA** - État de la connexion
- **Routage** - Gestion de la table de routage ← *Spécifique au STP*
- **Test de Routage** - Test de routage ← *Spécifique au STP*
- **Ressources** - Surveillance du système
- **Configuration** - Visualiseur de configuration

## Notes Importantes

- Le protocole SCTP (protocole IP 132) doit être autorisé à travers les pare-feu
- Le port M3UA par défaut est 2905 (norme de l'industrie)
- Assurez-vous de disposer de ressources système suffisantes pour gérer le trafic de routage
- **Persistance de Routage** : Toutes les routes configurées via l'interface Web ou l'API sont stockées dans la **base de données Mnesia** et **survivent aux redémarrages**
- **Fusion de Configuration** : Les routes de `runtime.exs` sont chargées au démarrage et fusionnées avec les routes Mnesia
- Après avoir modifié la configuration, vous devez redémarrer l'application pour que les modifications prennent effet
- **Interface Web** : Voir le Guide de l'Interface Web pour gérer les routes via l'interface web
- **Accès API** : Voir le Guide API pour la documentation de l'API REST et l'accès à Swagger UI

---

# Mode STP Autonome

Vous pouvez exécuter un **serveur STP M3UA autonome** qui écoute les connexions entrantes. C'est le mode utilisé par l'exemple dans `config/stp_example_runtime.exs`.

## Activation du STP Autonome

Ajoutez cette configuration sous `config :omniss7` dans votre configuration d'exécution :

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {127, 0, 0, 1},    # Adresse IP à écouter
    local_port: 2905,             # Port à écouter
    point_code: 100               # Code de point de ce STP
  }
```

## Paramètres de Configuration STP

| Paramètre | Type | Par Défaut | Description | Exemple |
|---|---|---|---|---|
| `enabled` | Boolean | `false` | Activer le serveur STP autonome | `true` |
| `local_ip` | Tuple | `{127, 0, 0, 1}` | Adresse IP à écouter pour les connexions | `{0, 0, 0, 0}` |
| `local_port` | Integer | `2905` | Port à écouter | `2905` |
| `point_code` | Integer | *Requis* | Code de point SS7 de ce STP | `100` |

## Quand Utiliser le STP Autonome

- **Routage Pur** : Lorsque vous avez seulement besoin de routage M3UA sans fonctionnalité de client MAP
- **STP Central** : Pour créer un routeur de signalisation central pour plusieurs éléments réseau
- **Architecture Hub** : Connecter plusieurs HLR, MSC et SMSC via un STP central

**Remarque** : Vous pouvez activer à la fois `map_client_m3ua` et `sctp_handler` simultanément si vous avez besoin à la fois de connexions sortantes et de fonctionnalités STP entrantes.

---

# Persistance de la Table de Routage (Mnesia)

Toutes les tables de routage (pairs, routes de Code de Point et routes de Titre Global) sont stockées dans une **base de données Mnesia** pour la persistance.

## Comment Fonctionne le Routage

1. **Routes runtime.exs** : Les routes définies dans `config/runtime.exs` sous `peers` (ou l'ancien `m3ua_peers`), `m3ua_routes`, et `m3ua_gt_routes` sont chargées au démarrage de l'application
2. **Routes Interface Web** : Les routes ajoutées via la page de Routage de l'Interface Web sont stockées dans Mnesia
3. **Fusion de Routes** : Au redémarrage, les routes runtime.exs sont fusionnées avec les routes Mnesia existantes (pas de doublons)
4. **Persistance** : Toutes les routes configurées via l'interface Web **survivent aux redémarrages de l'application**

## Type de Stockage Mnesia

Contrôlez comment les tables de routage sont stockées. Pour plus de détails sur la configuration de la base de données, voir Paramètres de Base de Données dans la Référence de Configuration.

```
config :omniss7,
  mnesia_storage_type: :disc_copies  # ou :ram_copies pour les
tests
```

| Type de Stockage | Description | Persistance | Cas d'Utilisation |
|---|---|---|---|
| `:disc_copies` | Stockage sur disque (par défaut) | **Survit aux redémarrages** | Environnements de production |
| `:ram_copies` | Seulement en mémoire | Perdu au redémarrage | Tests, développement |

**Par Défaut** : `:disc_copies`

## Emplacement de la Base de Données Mnesia

Mnesia stocke les tables de routage dans le répertoire Mnesia de l'application :

- **Emplacement** : `Mnesia.{node_name}/` (par exemple, `Mnesia.nonode@nohost/`)
- **Tables** (toutes créées dans `M3UARouting.init_database`):

| Table | But |
|---|---|
| `m3ua_peer` | Points de terminaison de connexion M3UA / M2PA |
| `m3ua_route` | Routes de Code de Point (DPC → pair, avec support de masque) |
| `m3ua_gt_route` | Routes de Titre Global (préfixe → pair, avec correspondance et réécritures SSN/TT/NPI/NAI) |
| `m3ua_dest_state` | État de disponibilité par destination géré par la gestion de route de signalisation MTP3 (TFP/TFA/TFR, Q.704 §13) |
| `ss7_screen_rule` | Règles de filtrage de passerelle (pare-feu SCCP) autoriser/refuser |

## Gestion des Routes

Vous avez trois options pour gérer les routes :

1. **Runtime.exs** - Configuration statique chargée au démarrage
2. **Interface Web** - Gestion interactive des routes (voir Guide de l'Interface Web)
3. **API REST** - Gestion programmatique des routes (voir Guide API)

**Meilleure Pratique** : Utilisez `runtime.exs` pour la configuration de base et l'interface Web pour les modifications dynamiques de routes pendant l'opération.

---

# Configuration des Pairs M3UA

Les pairs représentent les points de terminaison de connexion M3UA (autres STP, HLR, MSC, SMSC). Ajoutez des pairs à `config/runtime.exs`.

## Exemple de Configuration de Pair

> **Remarque** : La clé de configuration `peers` est la norme actuelle. L'ancienne clé `m3ua_peers` est toujours supportée pour la compatibilité ascendante.

```
config :omniss7,
  peers: [
    # Connexion sortante au STP Partenaire (rôle : :client)
    %{
      peer_id: 1,                          # Identifiant unique
      name: "Partner_STP_West",            # Nom descriptif
      role: :client,                       # :client pour
sortant, :server pour entrant
      local_ip: {10, 0, 0, 1},            # IP locale à lier
      local_port: 0,                       # 0 = affectation de
port dynamique
      remote_ip: {10, 0, 0, 10},          # IP du pair distant
      remote_port: 2905,                   # Port du pair distant
      routing_context: 1,                  # Contexte de routage
M3UA
      point_code: 100,                     # Code de point de ce
pair
      network_indicator: :international    # :international ou
:national
    },

    # Connexion au HLR Local (rôle : :client)
    %{
      peer_id: 2,
      name: "Local_HLR",
      role: :client,
      local_ip: {10, 0, 0, 1},
      local_port: 0,
      remote_ip: {10, 0, 0, 20},
      remote_port: 2905,
      routing_context: 2,
      point_code: 200,
      network_indicator: :international
    },

    # Connexion entrante depuis MSC Distant (rôle : :server)
    # Pour le rôle :server, le STP attend une connexion entrante
    %{
      peer_id: 3,
      name: "Remote_MSC",
      role: :server,                       # Accepter la
connexion entrante
      remote_ip: {10, 0, 0, 30},          # IP source attendue
```

```
      remote_port: 2905,                    # Port source attendu
(0 = accepter de n'importe quel port)
      routing_context: 3,
      point_code: 300,
      network_indicator: :international
    },

    # Connexion entrante avec port source dynamique (sans filtrage
de port)
    %{
      peer_id: 4,
      name: "Dynamic_Client",
      role: :server,
      remote_ip: {10, 0, 0, 40},           # IP source attendue
      remote_port: 0,                       # 0 = accepter les
connexions de n'importe quel port source
      routing_context: 4,
      point_code: 400,
      network_indicator: :international
    }
  ]
```

## Paramètres de configuration des pairs

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `peer_id` | Entier | Oui | Identifiant numérique unique pour le pair |
| `name` | Chaîne | Oui | Nom lisible par l'homme pour les journaux et la surveillance |
| `role` | Atome | Oui | `:client` (sortant) ou `:server` (entrant) |
| `local_ip` | Tuple ou Liste | Oui (client) | Adresse(s) IP locale(s) à lier. Unique : `{10, 0, 0, 1}` ou Multiple pour le multihoming SCTP : `[{10, 0, 0, 1}, {10, 0, 0, 2}]` |
| `local_port` | Entier | Oui (client) | Port local (0 pour dynamique) |
| `remote_ip` | Tuple ou Liste | Oui | **Rôle client** : Tuple ou liste unique pour un pair distant multihomé. **Rôle serveur** : Tuple unique uniquement - l'IP à partir de laquelle le pair distant se connecte (voir note ci-dessous) |
| `remote_port` | Entier | Oui | Port du pair distant (0 pour entrant = accepter tout port source) |
| `routing_context` | Entier | Oui | Identifiant de contexte de routage M3UA |

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `point_code` | Entier | Oui | Code de point SS7 de ce pair |
| `network_indicator` | Atome | Non | `:international` ou `:national` |

> **Multihoming SCTP pour le rôle serveur (connexions entrantes)** : Lors de l'acceptation de connexions entrantes d'un pair distant multihomé, vous devez uniquement spécifier l'**adresse IP unique** que le pair distant utilise pour initier la connexion SCTP (SCTP INIT). Cela doit être un **tuple unique**, pas une liste. SCTP découvrira automatiquement les autres adresses IP multihomées du pair lors de la poignée de main d'association. Le format de liste pour `remote_ip` n'est utilisé que pour le **rôle client** lors de la connexion à un pair distant multihomé.
>
> **Multihoming SCTP pour le rôle client (connexions sortantes)** : Pour la redondance réseau lors de la connexion à des pairs distants, vous pouvez configurer plusieurs adresses IP pour `local_ip` et `remote_ip` en utilisant des listes. Cela permet un basculement automatique si un chemin réseau échoue. Voir le Guide de multihoming SCTP pour des exemples de configuration détaillés et des meilleures pratiques.

## Filtrage de port source pour les connexions entrantes

Pour les **connexions entrantes** (rôle : `:server`), le paramètre `remote_port` contrôle le filtrage de port source :

- **Port spécifique** (par exemple, `remote_port: 2905`) : N'accepter que les connexions provenant de ce port source exact
  - Fournit une sécurité supplémentaire en validant le port source
  - Utilisez lorsque le pair distant utilise un port source fixe
- **Tout port** (`remote_port: 0`) : Accepter les connexions de tout port source
  - Utile lorsque le pair distant utilise des ports sources dynamiques/éphémères

- Valide uniquement l'adresse IP source
- Plus flexible mais légèrement moins sécurisé

**Exemple :**

```
# Accepter uniquement de 192.0.2.10:2905 (port spécifique)
%{
  peer_id: 1,
  name: "Strict_Peer",
  role: :server,
  remote_ip: {10, 5, 198, 200},
  remote_port: 2905,
  # ... autre configuration
}

# Accepter de 192.0.2.10 avec n'importe quel port source
%{
  peer_id: 2,
  name: "Flexible_Peer",
  role: :server,
  remote_ip: {10, 5, 198, 200},
  remote_port: 0,  # Accepter de tout port source
  # ... autre configuration
}
```

---

# Support du protocole M2PA

OmniSS7 prend en charge les protocoles **M3UA** et **M2PA** pour le transport de signalisation SS7.

## Qu'est-ce que M2PA ?

**M2PA** (MTP2 User Peer-to-Peer Adaptation Layer) est un protocole standardisé par l'IETF (RFC 4165) pour le transport des messages MTP3 SS7 sur des réseaux IP utilisant SCTP.

# M3UA vs M2PA : Principales différences

| Fonctionnalité | M3UA | M2PA |
|---|---|---|
| **Architecture** | Client/Serveur (ASP/SGW) | Pair à pair |
| **Cas d'utilisation** | Passerelle entre SS7 et IP | Liens directs point à point |
| **Gestion de l'état de lien** | Niveau application (ASPUP/ASPAC) | Style MTP2 (Alignement, Vérification, Prêt) |
| **Numéros de séquence** | Pas de séquençage inhérent | BSN/FSN de 24 bits pour livraison ordonnée |
| **Déploiement typique** | Passerelle SS7 vers IP, STP | Liens de signalisation directs entre nœuds |
| **RFC** | RFC 4666 | RFC 4165 |

# Conseils de sélection de protocole

**Recommandation : Utilisez M3UA par défaut. N'utilisez M2PA que si cela est spécifiquement requis.**

# Quand utiliser M3UA (recommandé)

M3UA est le protocole recommandé pour la plupart des déploiements :

- **Déploiements STP** : Implémentations standard de point de transfert de signalisation
- **Fonctions de passerelle** : Pont entre les réseaux SS7 et la signalisation basée sur IP
- **Connexions d'éléments de réseau** : Connexion des HLR, MSC, SMSC et autres éléments de réseau à votre STP

- **Passerelle de signalisation (SGW)** : Passerelle centrale acceptant des connexions de plusieurs serveurs d'application
- **Topologies flexibles** : Architectures client/serveur avec contrôle centralisé
- **Réseaux multi-fournisseurs** : Standard industriel largement supporté (RFC 4666)

**Utilisez M3UA pour connecter des éléments de réseau (HLR, MSC, SMSC, VLR, etc.) à votre STP.**

# Quand utiliser M2PA (cas spéciaux uniquement)

M2PA ne doit être utilisé que dans des scénarios spécifiques :

- **Liens STP à STP** : Connexions directes point à point entre des points de transfert de signalisation dans un réseau multi-STP
- **Remplacement TDM hérité** : Remplacement des liens TDM SS7 traditionnels lorsque le système distant exige spécifiquement M2PA
- **Compatibilité MTP2 requise** : Lors de la connexion à des systèmes hérités qui imposent une gestion d'état de lien de style MTP2
- **Exigence de partenaire** : Lorsqu'un partenaire ou un interconnexion exige spécifiquement le protocole M2PA

**Important** : Ne pas utiliser M2PA pour connecter des éléments de réseau (HLR, MSC, SMSC) à votre STP - utilisez plutôt M3UA. M2PA est conçu pour les interconnexions STP à STP où les deux côtés fonctionnent comme des nœuds de routage.

# Configuration des pairs M2PA

Les pairs M2PA sont configurés de la même manière que les pairs M3UA, avec un paramètre `protocol` supplémentaire.

**Configuration des pairs M2PA**

Ajoutez des pairs M2PA à votre configuration `peers` dans `config/runtime.exs` (les pairs M3UA et M2PA partagent la même section de configuration, différenciée par le paramètre `protocol`) :

**Paramètres clés pour M2PA :**

| Paramètre | Valeur | Description |
|---|---|---|
| `protocol` | `:m2pa` | Spécifie le protocole M2PA (par défaut `:m3ua` si omis) |
| `role` | `:client` ou `:server` | Direction de la connexion |
| `local_port` | Entier | Port SCTP local (le port standard M2PA est **3565**) |
| `remote_port` | Entier | Port SCTP distant (le port standard M2PA est **3565**) |
| `point_code` | Entier | Votre code de point |
| `adjacent_point_code` | Entier | Code de point du pair distant (spécifique à M2PA) |
| `send_sltm` | Booléen | Contrôle le comportement SLTM (voir Configuration SLTM ci-dessous) |
| `network_indicator` | Atome | `:international` ou `:national` - doit correspondre au pair distant |

**Remarque** : M2PA utilise **le port 3565** comme standard industriel (différent du port 2905 de M3UA).

# Configuration SLTM

**SLTM (Message de test de lien de signalisation)** et **SLTA (Accusé de réception du test de lien de signalisation)** sont des messages de maintenance MTP3 utilisés pour vérifier la connectivité de bout en bout après qu'un lien M2PA atteigne l'état PRÊT. Selon l'ITU-T Q.707, un côté envoie SLTM et l'autre répond avec SLTA pour confirmer que le lien est opérationnel.

### Comportement SLTM

Le paramètre `send_sltm` contrôle quel côté initie le test SLTM :

| Valeur `send_sltm` | Comportement |
|---|---|
| `true` | Nous envoyons SLTM lorsque le lien devient PRÊT, attendons SLTA |
| `false` | Nous attendons que le pair envoie SLTM, répondons avec SLTA |
| Non défini (par défaut) | Suit Q.707 : le répondeur SCTP envoie SLTM, l'initiateur SCTP attend |

**Comportement par défaut (conforme à Q.707) :**

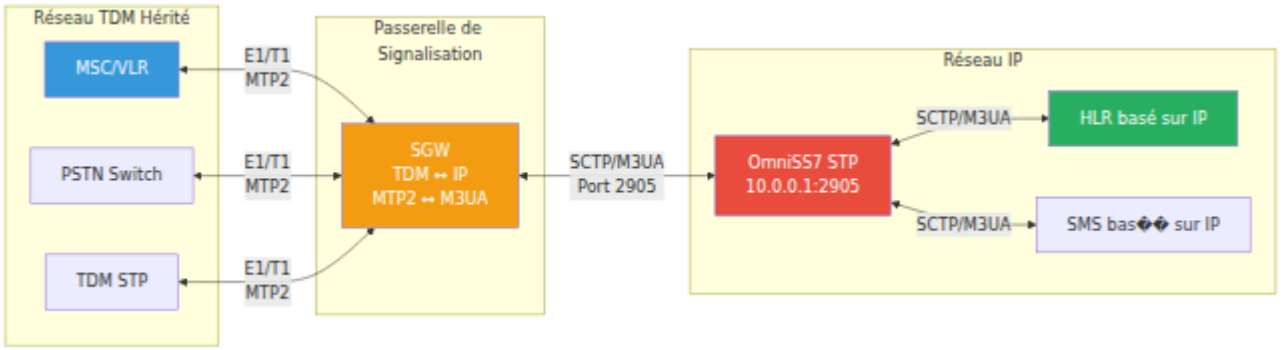
- Si `initiate_connection: true` (initiateur/client SCTP) → Nous attendons que le pair envoie SLTM
- Si `initiate_connection: false` (répondeur/server SCTP) → Nous envoyons SLTM

### Quand remplacer les valeurs par défaut SLTM

Remplacez le comportement par défaut lors de la connexion à un équipement qui ne suit pas les conventions Q.707 :

```
# Exemple : Forcer notre côté à envoyer SLTM indépendamment du
rôle SCTP
%{
  peer_id: 100,
  name: "Partner_STP",
  protocol: :m2pa,
  role: :client,
  local_ip: {10, 0, 0, 1},
  local_port: 3565,
  remote_ip: {10, 0, 0, 2},
  remote_port: 3565,
  point_code: 7415,
  adjacent_point_code: 15528,
  network_indicator: :international,
  send_sltm: true  # Remplacer : Nous envoyons SLTM même si nous
sommes l'initiateur SCTP
}
```

### Flux de messages SLTM




### Dépannage des problèmes SLTM

#### Symptôme : Le lien atteint PRÊT mais aucun trafic ne circule

Les deux côtés peuvent attendre que l'autre envoie SLTM. Vérifiez les journaux pour :

```
"waiting for peer to send SLTM"
```

**Résolution :** Configurez `send_sltm: true` d'un côté pour briser le blocage.

**Symptôme : SLTM envoyé mais aucun SLTA reçu**

Causes possibles :

1. `adjacent_point_code` est incorrect (SLTM envoyé avec un DPC erroné)
2. Mismatch d'indicateur de réseau (`:international` vs `:national`)
3. Pair distant non configuré pour répondre à notre code de point

**Résolution :** Vérifiez que `adjacent_point_code` correspond au code de point de l'STP (configuré dans `sctp_handler.point_code`).

**États de lien M2PA**

Les liens M2PA progressent à travers plusieurs états lors de l'initialisation :

1. **Hors ligne** - Aucune connexion établie
2. **Alignement** - Phase de synchronisation initiale (~1 seconde)
3. **Vérification** - Vérification de la qualité du lien (~2 secondes)
4. **Prêt** - Lien actif et prêt pour le trafic

La progression de l'état du lien garantit une signalisation fiable avant l'échange de trafic.

# Gestion des pairs M2PA via l'interface Web

La page **Routage** dans l'interface Web offre un support complet pour la gestion des pairs M2PA :

1. **Naviguer** vers la page Routage
2. **Sélectionner** l'onglet "Pairs"
3. **Cliquer** sur "Ajouter un nouveau pair"
4. **Choisir** "M2PA (RFC 4165)" dans le menu déroulant Protocole
5. **Remplir** la configuration du pair :
   - Nom du pair (identifiant descriptif)
   - Protocole : M2PA
   - Rôle : client ou serveur
   - Code de point (votre PC)

    - Adresses IP locales/distantes
    - Ports locaux/distants (typiquement 3565 pour M2PA)
    - Indicateur de réseau (international ou national)
6. **Cliquer** sur "Enregistrer le pair"

Le tableau des pairs affiche le type de protocole avec un code couleur :

- **Bleu** - pairs M3UA
- **Vert** - pairs M2PA

## Comportement de routage M2PA

Les pairs M2PA s'intègrent parfaitement au système de routage d'OmniSS7 :

- **Routes de code de point** : Fonctionnent de manière identique pour M2PA et M3UA
- **Routes de titre global** : Entièrement prises en charge sur les liens M2PA
- **Priorité de route** : Les pairs M2PA et M3UA peuvent être mélangés dans les mêmes tables de routage
- **Relais de message** : Les messages peuvent arriver sur M2PA et être routés vers M3UA, et vice versa

## Métriques M2PA

M2PA fournit des métriques Prometheus complètes pour surveiller la santé du lien et le trafic :

**Métriques de trafic :**

- `m2pa_messages_sent_total` - Total des messages MTP3 envoyés par lien
- `m2pa_messages_received_total` - Total des messages MTP3 reçus par lien
- `m2pa_bytes_sent_total` - Total des octets envoyés sur M2PA
- `m2pa_bytes_received_total` - Total des octets reçus sur M2PA

Toutes les métriques de trafic sont étiquetées par : `link_name`, `point_code`, `adjacent_pc`

**Métriques d'état de lien :**

- `m2pa_link_state_changes_total` - Transitions d'état de lien (HORS LIGNE → ALIGNEMENT → VÉRIFICATION → PRÊT)
  - Étiquettes : `link_name`, `from_state`, `to_state`

**Métriques d'erreur :**

- `m2pa_errors_total` - Total des erreurs par type
  - `decode_error` - Échecs de décodage de message M2PA
  - `encode_error` - Échecs d'encodage de message M2PA
  - `sctp_send_error` - Échecs de transmission SCTP
  - Étiquettes : `link_name`, `error_type`

**Métriques d'accès :**

- Point de terminaison Prometheus : `http://your-server:9090/metrics`
- Les métriques s'enregistrent automatiquement au démarrage de l'application

# Meilleures pratiques M2PA

1. **Sélection de port** : Utilisez le port 3565 pour M2PA (standard industriel)
2. **Surveillance de lien** : Surveillez les changements d'état de lien via les métriques
3. **Règles de pare-feu** : Assurez-vous que SCTP (protocole IP 132) est autorisé
4. **Codes de point** : Assurez-vous que les codes de point adjacents sont correctement configurés des deux côtés
5. **Indicateur de réseau** : Doit correspondre entre les pairs (international ou national)
6. **Test** : Utilisez la page de test de routage pour vérifier la connectivité après la configuration

# Exigences de socket M2PA

**M2PA utilise des sockets partagés SCTP.SocketHandler.** Tous les pairs M2PA utilisent automatiquement le SCTP.SocketHandler pour la gestion des sockets, ce qui permet à plusieurs pairs de partager efficacement le même port SCTP.

**Exigences :**

- SCTP.SocketHandler doit être activé et en cours d'exécution
- Le socket partagé doit être configuré avant le démarrage des pairs M2PA

**Exemple :**

```
# Activer SCTP.SocketHandler
sctp_handler: %{
  enabled: true,
  local_ip: {10, 179, 4, 10},
  local_port: 3565,
  point_code: 100
}

# Les pairs M2PA utilisent automatiquement le socket partagé
peers: [
  %{
    peer_id: 1,
    name: "M2PA_Link_1",
    protocol: :m2pa,
    role: :client,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,
    remote_ip: {10, 179, 4, 20},
    remote_port: 3565,
    point_code: 100,
    adjacent_point_code: 200
  },
  %{
    peer_id: 2,
    name: "M2PA_Link_2",
    protocol: :m2pa,
    role: :client,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,           # Même port partagé entre les
pairs
    remote_ip: {10, 179, 4, 30},
    remote_port: 3565,
    point_code: 100,
    adjacent_point_code: 300
  }
]
```

**Exigences SCTP :**

- Livraison ordonnée (pas de drapeau `unordered`)
- PPID 5 (identifiant M2PA selon RFC 4165)

---

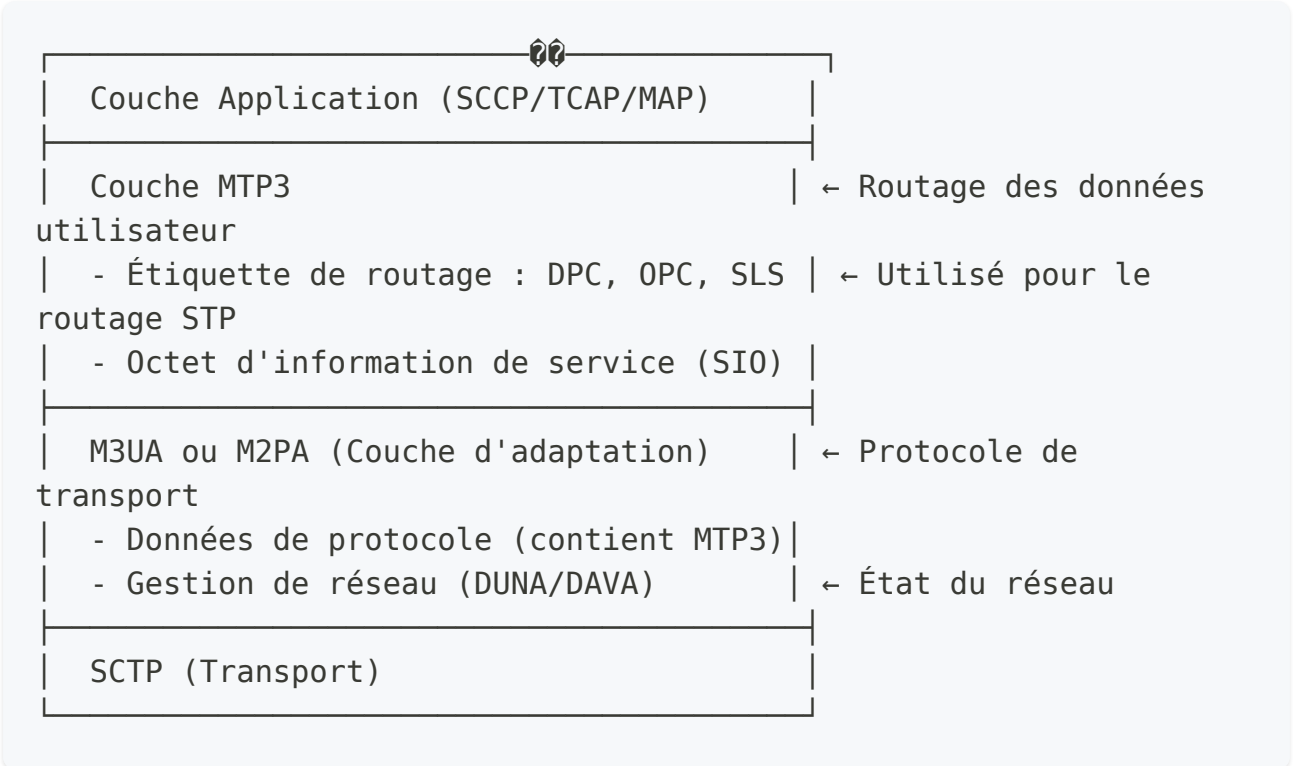
# Configuration du routage par code de point

Le routage par code de point dirige les messages en fonction du **Code de Point de Destination (DPC)** dans l'en-tête MTP3.

## Comprendre les codes de point dans la pile de protocoles SS7

Les codes de point existent à différents niveaux de la pile de protocoles SS7. Comprendre cette distinction est important :

**Couches de la pile de protocoles :**




**Deux types de codes de point :**

1. **Codes de point de couche MTP3** (Utilisés pour le routage) :
    - Situés dans l'étiquette de routage MTP3 (DPC, OPC)
    - Présents dans le paramètre de données de protocole M3UA (tag 528)
    - Présents dans les messages de donn��es utilisateur M2PA

    - **Le STP utilise ces valeurs DPC pour les décisions de routage**
    - Ceux-ci déterminent où le message est finalement livré

2. **Codes de point de couche M3UA** (Utilisés pour la gestion du réseau) :

    - Présents dans les messages de gestion M3UA (DUNA, DAVA, SCON, DUPU)
    - Indiquent les codes de point affectés pour l'état du réseau
    - Indiquent aux pairs quelles destinations sont disponibles/non disponibles
    - Non utilisés pour le routage des données utilisateur

**Comment fonctionne le routage STP :**

- **Pour les messages M3UA DATA** : Le STP extrait le message MTP3 du paramètre de données de protocole (tag 528), qui contient l'étiquette de routage MTP3 (DPC, OPC, SLS). Le DPC de la couche MTP3 est utilisé pour rechercher des routes.
- **Pour les messages de données utilisateur M2PA** : Le STP extrait le message MTP3 du champ de données utilisateur M2PA, puis lit le DPC de l'étiquette de routage MTP3.
- **Messages de gestion M3UA** : Les messages de gestion du réseau (DUNA, DAVA, SCON) contiennent des codes de point affectés au niveau M3UA pour signaler l'état du réseau entre les pairs.

# Routes de code de point de base

Ajoutez des routes à `config/runtime.exs` :

```
config :omniss7,
  m3ua_routes: [
    # Routage de tout le trafic pour le PC 100 vers le pair 1
(Partner STP)
    %{
      dest_pc: 100,                   # Code de point de
destination
      peer_id: 1,                     # Pair à travers lequel
router
      priority: 1,                    # Priorité (plus bas = plus
haute priorité)
      network_indicator: :international
      # mask: 14                      # Optionnel : par défaut 14
(correspondance exacte)
    },

    # Routage de tout le trafic pour le PC 200 vers le pair 2 (HLR
local)
    %{
      dest_pc: 200,
      peer_id: 2,
      priority: 1,
      network_indicator: :international
    },

    # Exemple d'équilibrage de charge : PC 300 avec routes
primaire et de secours
    %{
      dest_pc: 300,
      peer_id: 3,                     # Route primaire
      priority: 1,
      network_indicator: :international
    },
    %{
      dest_pc: 300,
      peer_id: 4,                     # Route de secours (numéro
de priorité plus élevé)
      priority: 2,
      network_indicator: :international
    }
  ]
```

**Remarque** : Le champ `mask` est optionnel et par défaut à `14` (correspondance exacte du code de point). Ne spécifiez `mask` que lorsque vous avez besoin d'un routage basé sur des plages (voir la section sur les masques de code de point ci-dessous).

# Logique de routage

1. Le STP reçoit un message M3UA DATA ou M2PA User Data
2. Le STP extrait le **message MTP3** du champ de données de protocole (M3UA) ou de données utilisateur (M2PA)
3. Le STP lit le **Code de Point de Destination (DPC)** de l'étiquette de routage MTP3
4. Recherche dans la table de routage le DPC correspondant (en tenant compte des masques)
5. Si plusieurs routes existent, sélectionne la route avec le **masque le plus spécifique** (valeur de masque la plus élevée), puis **le numéro de priorité le plus bas**
6. Enveloppe le message MTP3 dans M3UA DATA ou M2PA User Data pour le pair de destination
7. Route le message vers le pair correspondant
8. Si le pair sélectionné est hors ligne, essaie la route de priorité la plus élevée suivante

# Masques de code de point

Les codes de point sont des valeurs de 14 bits (plage 0-16383). Par défaut, les routes correspondent exactement à un seul code de point (masque `/14`). Cependant, vous pouvez utiliser des **masques de code de point** pour créer des routes qui correspondent à des **plages** de codes de point.

### Comprendre les masques

Le masque spécifie combien de **bits les plus significatifs** doivent correspondre entre le PC de destination de la route et le DPC du message entrant. Les bits restants peuvent avoir n'importe quelle valeur, créant une plage de codes de point correspondants.

### Tableau de référence des masques :

| Masque | Codes de point correspondants | Cas d'utilisation |
|---|---|---|
| `/14` | 1 PC (correspondance exacte) | Destination unique (par défaut) |
| `/13` | 2 PCs | Petite plage |
| `/12` | 4 PCs | Petite plage |
| `/11` | 8 PCs | Petite plage |
| `/10` | 16 PCs | Plage moyenne |
| `/9` | 32 PCs | Plage moyenne |
| `/8` | 64 PCs | Plage moyenne |
| `/7` | 128 PCs | Plage moyenne-grande |
| `/6` | 256 PCs | Grande plage |
| `/5` | 512 PCs | Grande plage |
| `/4` | 1,024 PCs | Très grande plage |
| `/3` | 2,048 PCs | Très grande plage |
| `/2` | 4,096 PCs | Plage extrêmement grande |
| `/1` | 8,192 PCs | La moitié de tous les PCs |
| `/0` | 16,384 PCs | Tous les PCs (route par défaut/de secours) |

**Exemples de masques de code de point**

**Remarque** : Le champ `mask` est **optionnel** dans tous les exemples. S'il est omis, il par défaut à `14` (correspondance exacte).

**Exemple 1 : Code de point unique (comportement par défaut)**

```
# Sans champ de masque (recommandé pour un seul PC)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  network_indicator: :international
}
# Masque par défaut à 14 - Correspond : Seulement PC 1000

# Masque explicite (même résultat)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  mask: 14,                                # Correspondance exacte
explicite
  network_indicator: :international
}
# Correspond : Seulement PC 1000
```

**Exemple 2 : Petite plage**

```
%{
  dest_pc: 1000,
  peer_id: 2,
  priority: 1,
  mask: 12,                                # Correspond à 4 PCs
  network_indicator: :international
}
# Correspond : PC 1000, 1001, 1002, 1003
```

**Exemple 3 : Plage moyenne**

```
%{
  dest_pc: 1000,
  peer_id: 3,
  priority: 1,
  mask: 8,                                # Correspond à 64 PCs
  network_indicator: :international
}
# Correspond : PC 1000-1063 (64 codes de point consécutifs)
```

**Exemple 4 : Route par défaut/de secours**

```
%{
  dest_pc: 0,
  peer_id: 4,
  priority: 10,                           # Faible priorité (numéro
élevé)
  mask: 0,                                # Correspond à tous les
PCs
  network_indicator: :international
}
# Correspond : Tous les codes de point (0-16383)
# Utiliser comme route par défaut/catch-all avec une faible
priorité
```

**Combinaison de routes spécifiques et de routes masquées**

Vous pouvez combiner des routes spécifiques avec des routes masquées pour un routage flexible :

```
config :omniss7,
  m3ua_routes: [
    # Route spécifique pour le PC 1000 (prend la priorité)
    %{
      dest_pc: 1000,
      peer_id: 1,
      priority: 1,
      network_indicator: :international
      # le masque par défaut est 14 (correspondance exacte)
    },

    # Route de plage pour les PC 1000-1063
    %{
      dest_pc: 1000,
      peer_id: 2,
      priority: 1,
      mask: 8,                          # Correspond à 64 PC
      network_indicator: :international
    },

    # Route par défaut/de secours pour tous les autres PC
    %{
      dest_pc: 0,
      peer_id: 3,
      priority: 10,                     # Priorité basse
      mask: 0,                          # Correspond à tous les
PC
      network_indicator: :international
    }
  ]
```

**Décision de routage pour le DPC 1000 :**

1. Correspond au masque `/14` (PC 1000 exactement) - **Sélectionné** (le plus spécifique)
2. Correspond également au masque `/8` (plage PC 1000-1063) - Ignoré (moins spécifique)
3. Correspond également au masque `/0` (tous les PC) - Ignoré (le moins spécifique)

**Décision de routage pour le DPC 1015 :**

1. Ne correspond pas au masque `/14` (PC 1000 uniquement)
2. Correspond au masque `/8` (plage PC 1000-1063) - **Sélectionné** (correspondance la plus spécifique)
3. Correspond également au masque `/0` (tous les PC) - Ignoré (moins spécifique)

**Décision de routage pour le DPC 5000 :**

1. Ne correspond pas au masque `/14`
2. Ne correspond pas au masque `/8`
3. Correspond au masque `/0` (tous les PC) - **Sélectionné** (seule correspondance, route de secours)

**Meilleures pratiques**

1. **Omettre `mask` pour les destinations uniques** : Pour les correspondances exactes de code de point, omettez complètement le champ `mask` (par défaut à `/14`)
2. **Utiliser `/14` explicitement uniquement si nécessaire** : Spécifiez `mask: 14` uniquement lorsque vous devez le clarifier dans la documentation ou lorsque vous mélangez avec des routes de plage
3. **Utiliser des masques de plage pour les blocs de réseau** : Routage de segments de réseau entiers vers des pairs spécifiques avec des masques de `/0` à `/13`
4. **Utiliser `/0` comme secours** : Créez une route par défaut avec une faible priorité pour capturer le trafic non apparié
5. **Le plus spécifique l'emporte** : Le moteur de routage sélectionne toujours la route correspondante la plus spécifique (valeur de masque la plus élevée) en premier
6. **Priorité comme critère de départage** : Si plusieurs routes ont le même masque, le numéro de priorité le plus bas l'emporte

---

# Configuration du routage par titre global (GT)

Le routage par titre global permet un **routage basé sur le contenu** en utilisant des numéros de téléphone ou des valeurs IMSI au lieu de codes de point. Pour une traduction d'adresse de titre global avancée basée sur l'appelant/l'appelé, consultez le Guide NAT de Titre Global.

## Prérequis

- Activer le routage GT : `enable_gt_routing: true` dans `config/runtime.exs`

## Configuration de la route GT

```
config :omniss7,
  # Activer le routage GT
  enable_gt_routing: true,

  m3ua_gt_routes: [
    # Route tous les numéros UK (préfixe 44) vers le pair 1
    %{
      gt_prefix: "44",                # Préfixe de Titre Global à
correspondre
      peer_id: 1,                     # Pair de destination
      priority: 1,                    # Priorité (plus bas = plus
élevé)
      description: "Numéros UK"       # Description pour la
journalisation
    },

    # Route les numéros US (préfixe 1) vers le pair 2
    %{
      gt_prefix: "1",
      peer_id: 2,
      priority: 1,
      description: "Numéros US"
    },

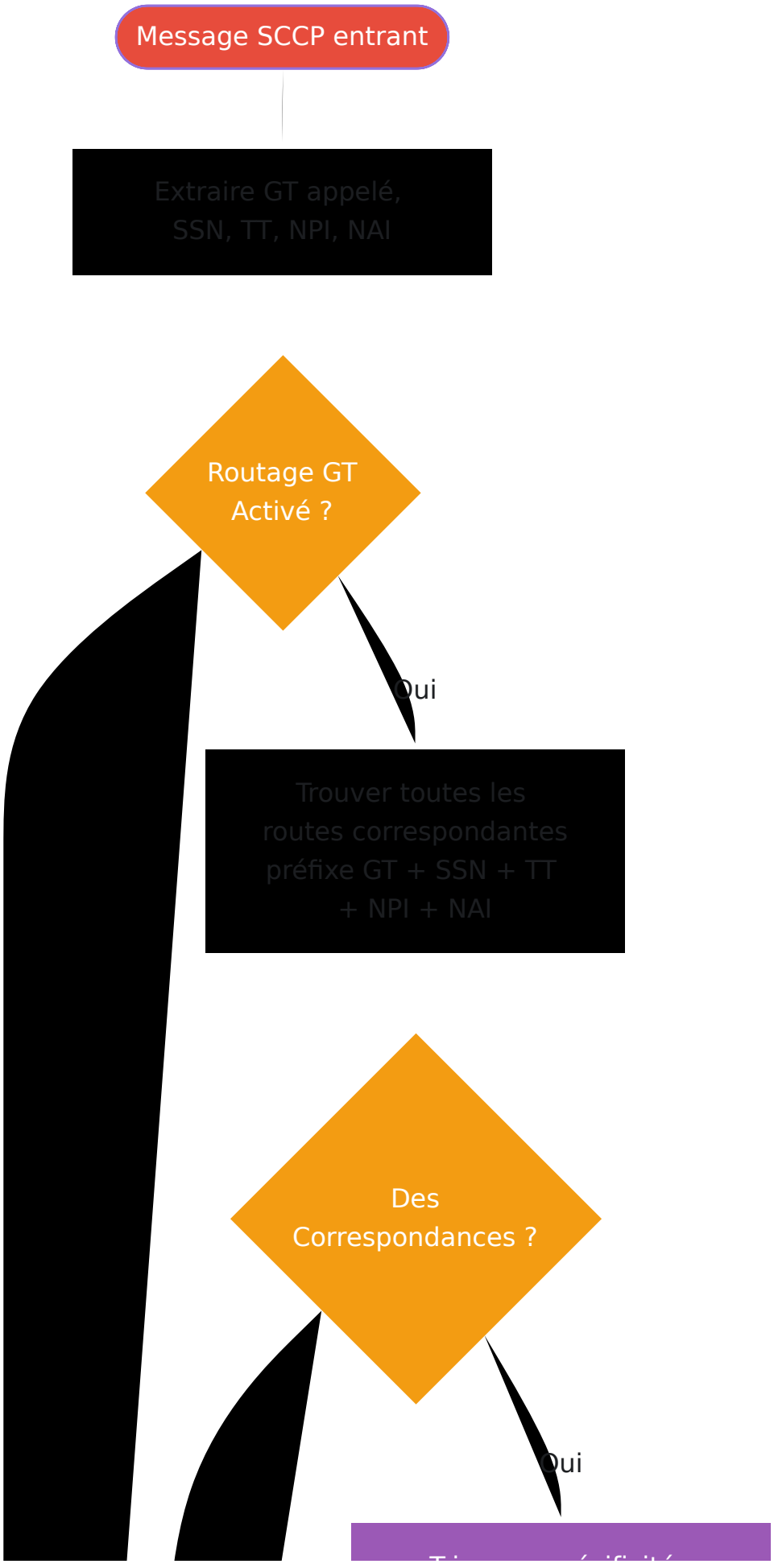
    # Route plus spécifique : numéros mobiles UK commençant par
447
    %{
      gt_prefix: "447",               # La correspondance de
préfixe la plus longue l'emporte
      peer_id: 3,
      priority: 1,
      description: "Numéros mobiles UK"
    },

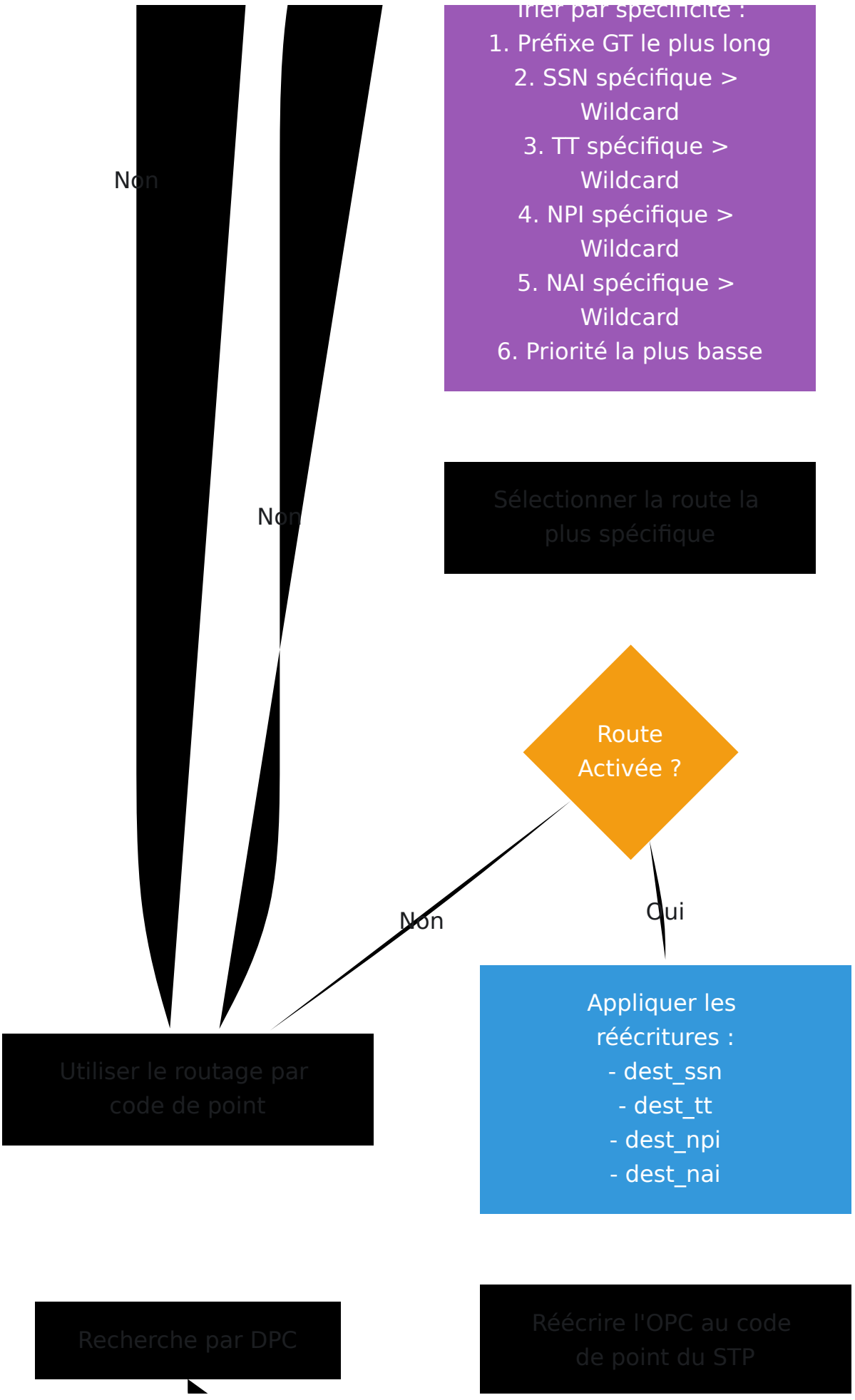
    # Routage spécifique SSN (optionnel)
    %{
      gt_prefix: "555",
      source_ssn: 8,                  # Correspond uniquement si
SSN source = 8 (SMSC)
      peer_id: 4,
      dest_ssn: 6,                    # Réécrit le SSN de
```

```
destination à 6 (HLR)
      priority: 1,
      description: "Trafic SMS pour le préfixe 61"
    }
  ]
```

## Logique de routage GT

L'algorithme de routage GT suit ce processus décisionnel :


Message SCCP entrant
Extraire GT appelé, SSN, TT, NPI, NAI
Routage GT Activé ?
Oui
Trouver toutes les routes correspondantes préfixe GT + SSN + TT + NPI + NAI
Des Correspondances ?
Oui


Trier par spécificité :
1. Préfixe GT le plus long
2. SSN spécifique > Wildcard
3. TT spécifique > Wildcard
4. NPI spécifique > Wildcard
5. NAI spécifique > Wildcard
6. Priorité la plus basse
Non
Non
Sélectionner la route la plus spécifique
Route Activée ?
Non
Oui
Appliquer les réécritures :
- dest_ssn
- dest_tt
- dest_npi
- dest_nai
Utiliser le routage par code de point
Recherche par DPC
Réécrire l'OPC au code de point du STP




**Étapes de routage :**

1. **Correspondance de préfixe la plus longue** : Le STP trouve toutes les routes GT où le préfixe correspond au début du Titre Global
    - Exemple : GT "447712345678" correspond à "44" et "447", mais "447" l'emporte (correspondance la plus longue)
2. **Correspondance SSN** (optionnelle) :
    - Si `source_ssn` est spécifié, la route ne correspond que lorsque le SSN de la partie appelée SCCP est égal à cette valeur
    - Si `source_ssn` est `nil`, la route correspond à n'importe quel SSN (wildcard)
3. **Correspondance TT/NPI/NAI** (optionnelle) :
    - Si `source_tt`, `source_npi` ou `source_nai` sont spécifiés, les routes doivent correspondre à ces indicateurs
    - Les valeurs `nil` agissent comme des wildcards (correspondent à n'importe quelle valeur)
4. **Sélection basée sur la spécificité** :
    - Les routes avec des critères de correspondance plus spécifiques l'emportent sur les wildcards
    - Ordre de priorité : Longueur du préfixe GT → SSN → TT → NPI → NAI → Numéro de priorité
5. **Réécriture des indicateurs** (optionnelle) :

- Si `dest_ssn`, `dest_tt`, `dest_npi` ou `dest_nai` sont spécifiés, le STP réécrit ces indicateurs
- Utile pour la normalisation des protocoles et l'interconnexion des réseaux

6. **Retour au code de point** :
    - Si aucune route GT ne correspond, le STP revient au routage par code de point en utilisant le DPC

## Routage GT avancé : Type de traduction, NPI et NAI

En plus de la correspondance de préfixe GT et de SSN, le STP prend en charge le routage et la transformation basés sur les indicateurs de Titre Global SCCP :

- **Type de traduction (TT)** : Identifie le plan de numérotation et le type d'adresse
- **Indicateur de plan de numérotation (NPI)** : Définit le plan de numérotation (par exemple, ISDN, Données, Télex)
- **Indicateur de nature d'adresse (NAI)** : Spécifie le format d'adresse (par exemple, International, National, Abonné)

#### Correspondance (Indicateurs source)

Les routes peuvent correspondre aux indicateurs de message entrant :

- `source_tt` : Correspond aux messages avec un Type de Traduction spécifique
- `source_npi` : Correspond aux messages avec un Indicateur de Plan de Numérotation spécifique
- `source_nai` : Correspond aux messages avec un Indicateur de Nature d'Adresse spécifique
- Valeur `nil` = wildcard (correspond à n'importe quelle valeur)

**Transformation (Indicateurs de destination)**

Les routes peuvent réécrire les indicateurs lors de la transmission :

- `dest_tt` : Transformer le Type de Traduction en nouvelle valeur
- `dest_npi` : Transformer l'Indicateur de Plan de Numérotation en nouvelle valeur
- `dest_nai` : Transformer l'Indicateur de Nature d'Adresse en nouvelle valeur
- Valeur `nil` = préserver la valeur originale (pas de transformation)

**Sélection basée sur la spécificité**

Lorsque plusieurs routes correspondent, la route la plus spécifique est sélectionnée en utilisant cet ordre de priorité :

1. Correspondance de préfixe GT la plus longue
2. SSN source spécifique sur SSN wildcard
3. TT source spécifique sur TT wildcard
4. NPI source spécifique sur NPI wildcard
5. NAI source spécifique sur NAI wildcard
6. Numéro de priorité le plus bas

**Exemples de configuration**

```
config :omniss7,
  enable_gt_routing: true,

  m3ua_gt_routes: [
    # Exemple 1 : Correspondre et transformer le Type de
Traduction
    %{
      gt_prefix: "44",
      peer_id: 1,
      source_tt: 0,      # Correspondre TT=0 (Inconnu)
      dest_tt: 3,        # Transformer en TT=3 (National)
      priority: 1,
      description: "Numéros UK : transformation TT 0→3"
    },

    # Exemple 2 : Correspondre un NPI spécifique et transformer
NAI
    %{
      gt_prefix: "1",
      peer_id: 2,
      source_npi: 1,     # Correspondre NPI=1 (ISDN/Téléphonie)
      source_nai: 4,     # Correspondre NAI=4 (International)
      dest_nai: 3,       # Transformer en NAI=3 (National)
      priority: 1,
      description: "Numéros US : NAI International→National"
    },

    # Exemple 3 : Routage combiné SSN et indicateur
    %{
      gt_prefix: "33",
      source_ssn: 8,     # Correspondre au trafic SMSC
      source_tt: 0,      # Correspondre TT=0
      dest_ssn: 6,       # Réécrire SSN en HLR
      dest_tt: 2,        # Transformer en TT=2
      dest_npi: 1,       # Définir NPI=1 (ISDN)
      dest_nai: 4,       # Définir NAI=4 (International)
      peer_id: 3,
      priority: 1,
      description: "SMS français : Normalisation complète"
    },

    # Exemple 4 : TT wildcard, NPI spécifique
    %{
```

```
      gt_prefix: "49",
      source_tt: nil,    # Correspondre à n'importe quel TT
(wildcard)
      source_npi: 6,     # Correspondre NPI=6 (Données)
      dest_npi: 1,       # Transformer en NPI=1 (ISDN)
      peer_id: 4,
      priority: 1,
      description: "Normalisation du réseau de données allemand"
    }
  ]
```

**Valeurs courantes de TT/NPI/NAI**

**Type de Traduction (TT) :**

- `0` = Inconnu
- `1` = International
- `2` = National
- `3` = Spécifique au réseau

**Indicateur de Plan de Numérotation (NPI) :**

- `0` = Inconnu
- `1` = ISDN/Téléphonie (E.164)
- `3` = Données (X.121)
- `4` = Télex (F.69)
- `6` = Mobile terrestre (E.212)

**Indicateur de Nature d'Adresse (NAI) :**

- `0` = Inconnu
- `1` = Numéro d'abonné
- `2` = Réservé à un usage national
- `3` = Numéro significatif national
- `4` = Numéro international

**Exemple de décision de routage**

Pour un message entrant avec :

- GT : "447712345678"
- SSN : 8
- TT : 0
- NPI : 1
- NAI : 4

Avec ces routes configurées :

```
# Route A : TT wildcard
%{gt_prefix: "447", peer_id: 1, priority: 1}

# Route B : TT spécifique
%{gt_prefix: "447", source_tt: 0, peer_id: 2, priority: 1}

# Route C : TT spécifique + NPI
%{gt_prefix: "447", source_tt: 0, source_npi: 1, peer_id: 3,
priority: 1}
```

**Résultat** : La route C est sélectionnée (la plus spécifique : correspond à GT + TT + NPI)

Le message est transmis avec les indicateurs transformés selon les valeurs `dest_tt`, `dest_npi`, `dest_nai` de la Route C.

## Exemples de routage GT

| GT appelé | SSN source | TT | NPI | NAI | Route correspondante | Rais |
|---|---|---|---|---|---|---|
| 447712345678 | 6 | - | - | - | "447" → pair 3 | Correspo de préfix plus long |
| 441234567890 | 6 | - | - | - | "44" → pair 1 | Correspo de préfix de route spécifiqu |
| 12125551234 | 6 | - | - | - | "1" → pair 2 | Correspo de préfix les numé |
| 555881234567 | 8 | - | - | - | "555" (SSN 8) → pair 4 | Correspo GT + SSN réécrit S |
| 555881234567 | 6 | - | - | - | "555" (SSN wildcard) → pair X | Correspo GT, pas d réécritur |
| 441234567890 | 6 | 0 | 1 | 4 | "44" (TT=0) → pair 1 | Correspo GT + TT, transforn en 3 |
| 12125551234 | 8 | 0 | 1 | 4 | "1" (TT=0, NPI=1, NAI=4) | La plus spécifiqu correspo GT+TT+I |

**Cas d'utilisation pratiques pour le routage TT/NPI/NAI**

1. **Normalisation de l'interconnexion des réseaux**
    - Différents réseaux peuvent utiliser différentes conventions d'indicateurs
    - Transformer les indicateurs au point d'interconnexion pour assurer la compatibilité
    - Exemple : Le réseau partenaire utilise TT=0 pour international, votre réseau utilise TT=1
2. **Conversion de protocole**
    - Convertir entre les plans de numérotation lors du routage entre différents types de réseaux
    - Exemple : Routage depuis un réseau mobile (NPI=6) vers le PSTN (NPI=1)
3. **Standardisation du format d'adresse**
    - Normaliser tout le trafic entrant pour utiliser des valeurs NAI cohérentes
    - Exemple : Convertir tous les formats internationaux (NAI=4) en format national (NAI=3) pour le routage national
4. **Routage spécifique au transporteur**
    - Routage basé sur le type de traduction vers différents fournisseurs de services
    - Exemple : TT=0 routé vers le transporteur A, TT=2 routé vers le transporteur B
5. **Intégration de systèmes hérités**
    - Les systèmes modernes peuvent utiliser des valeurs d'indicateurs différentes de celles des systèmes hérités
    - Transformer au niveau du STP pour maintenir la compatibilité descendante

---

# Gestion du transit SCCP

Au-delà de la sélection d'un pair de destination, le STP effectue les fonctions de transit de couche SCCP requises d'un véritable Point de Contrôle de Signalisation. Celles-ci s'appliquent aux classes de messages sans connexion que le STP relaye : **UDT** (0x09), **XUDT** (0x11) et **LUDT** (0x13).

## Routage par SSN après traduction de Titre Global

Par défaut, le STP **préserve l'indicateur de routage de la partie appelée comme routage par GT** lorsqu'il transmet un message correspondant à une route GT. Cela est délibéré : sélectionner un prochain saut par préfixe GT est *routage GT* (intermédiaire), pas *traduction finale de GT*. Selon ITU-T Q.714 §2.4, un nœud intermédiaire doit laisser l'indicateur de routage comme routage par GT afin que le nœud en aval effectue la traduction. Passer à un routage par SSN là fausse un résultat intermédiaire en tant que final. Le message porte alors un GT avec routage par SSN et aucun code de point, ce qu'un nœud en aval (par exemple, un MSS qui s'attend à exécuter son propre GTT) rejette avec `providerAbort(abnormalDialogue)`.

Définissez l'indicateur de routage sur routage par SSN **uniquement** lorsque *ce* STP est le traducteur final et que la route livre à un point de terminaison uniquement SSN qui ne peut pas effectuer de GTT supplémentaire. Cela est un opt-in explicite par route via le drapeau `route_on_ssn: true` sur la route GT :

```
m3ua_gt_routes: [
  # Routage de transit - préserve le routage par GT (par défaut) ;
en aval effectue GTT
  %{gt_prefix: "44", peer_id: 2, priority: 1, description:
"Numéros UK"},

  # Livraison finale à un point de terminaison uniquement SSN -
bascule vers routage par SSN
  %{
    gt_prefix: "1234",
    peer_id: 1,
    priority: 1,
    dest_ssn: 8,
    route_on_ssn: true,
    description: "Livraison finale à un MSC uniquement SSN"
  }
]
```

Le drapeau peut également être défini par route via l'API de routage (`route_on_ssn` sur `POST /api/routing/gt-routes`) et par défaut à `false`. Lorsqu'il est activé, la bascule ne s'applique qu'à une route actuellement adressée par la partie appelée routage par GT qui porte un SSN non nul (ITU-T Q.714 §2.4).

## Application du compteur de sauts (XUDT / LUDT)

Les messages XUDT et LUDT portent un **compteur de sauts SCCP** qui protège contre les boucles de routage entre les nœuds de relais SCCP. À chaque saut de transit, le STP :

1. **Décrémente** le compteur de sauts dans le message transmis (Q.714 §2.4).
2. Si la décrémentation amènerait le compteur à **zéro**, le message est **jeté** plutôt que transmis (Q.714 §2.8.6).
3. Un rejet incrémente `m3ua_stp_routing_failures_total{reason="hcounter_violation"}`.
4. Si le message original a demandé un retour en cas d'erreur, un message `XUDTS`/`LUDTS` de retour est généré vers l'initiateur (voir ci-dessous).

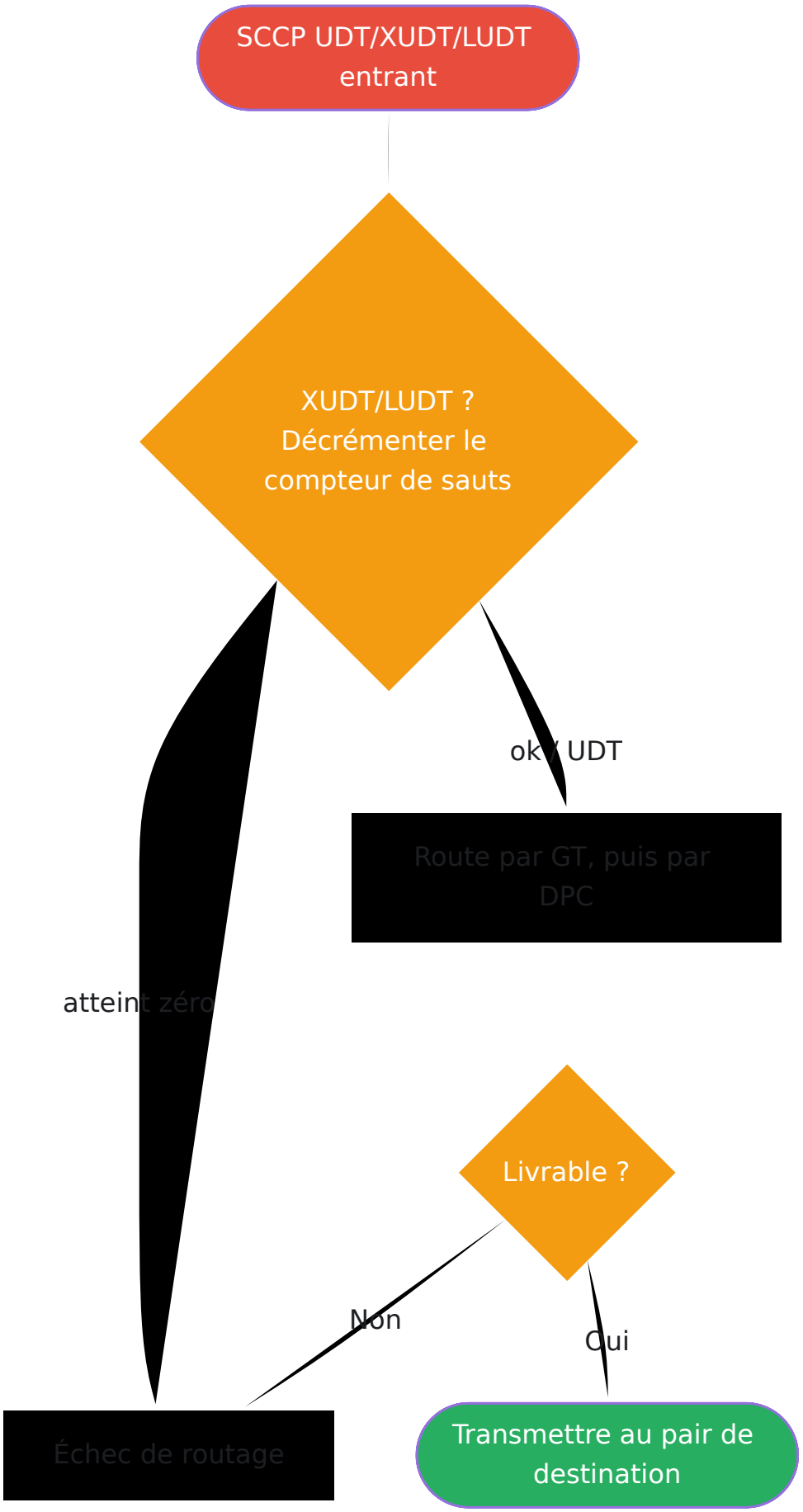
UDT n'a pas de compteur de sauts et est transmis sans changement à cette étape.

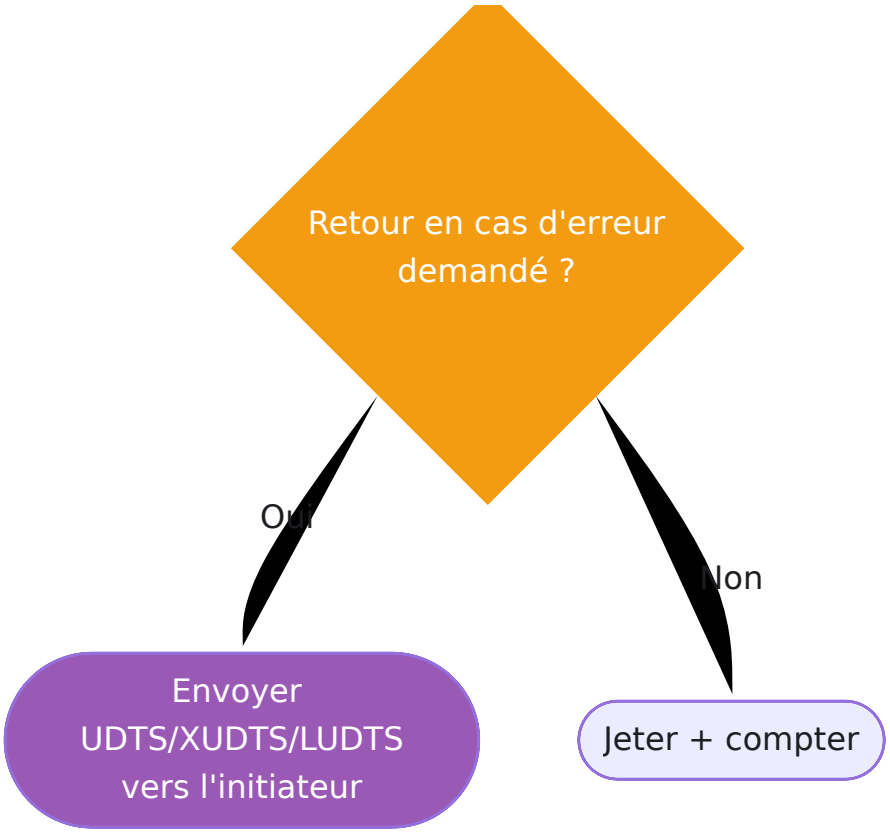
## Retour de message en cas d'échec de routage (UDTS / XUDTS / LUDTS)

Lorsqu'un message sans connexion ne peut pas être livré (aucune route, destination interdite, compteur de sauts épuisé, etc.) **et que l'initiateur a défini l'option de retour en cas d'erreur**, le STP construit un message de retour SCCP et l'envoie vers l'initiateur (ITU-T Q.714 §4.2) :

| Message original | Retourne comme |
|---|---|
| UDT (0x09) | UDTS (0x0A) |
| XUDT (0x11) | XUDTS (0x12) |
| LUDT (0x13) | LUDTS (0x14) |

La cause de retour portée dans le message reflète la raison de l'échec (par exemple, un échec MTP pour une destination interdite, ou une violation du compteur de sauts pour un compteur épuisé). Si l'initiateur n'a **pas** demandé de retour en cas d'erreur, le message est simplement jeté et compté.


SCCP UDT/XUDT/LUDT entrant
XUDT/LUDT ? Décrémenter le compteur de sauts
ok / UDT
Route par GT, puis par DPC
atteint zéro
Livrable ?
Non
Oui
Échec de routage
Transmettre au pair de destination




---

# État de destination (Gestion des routes de signalisation)

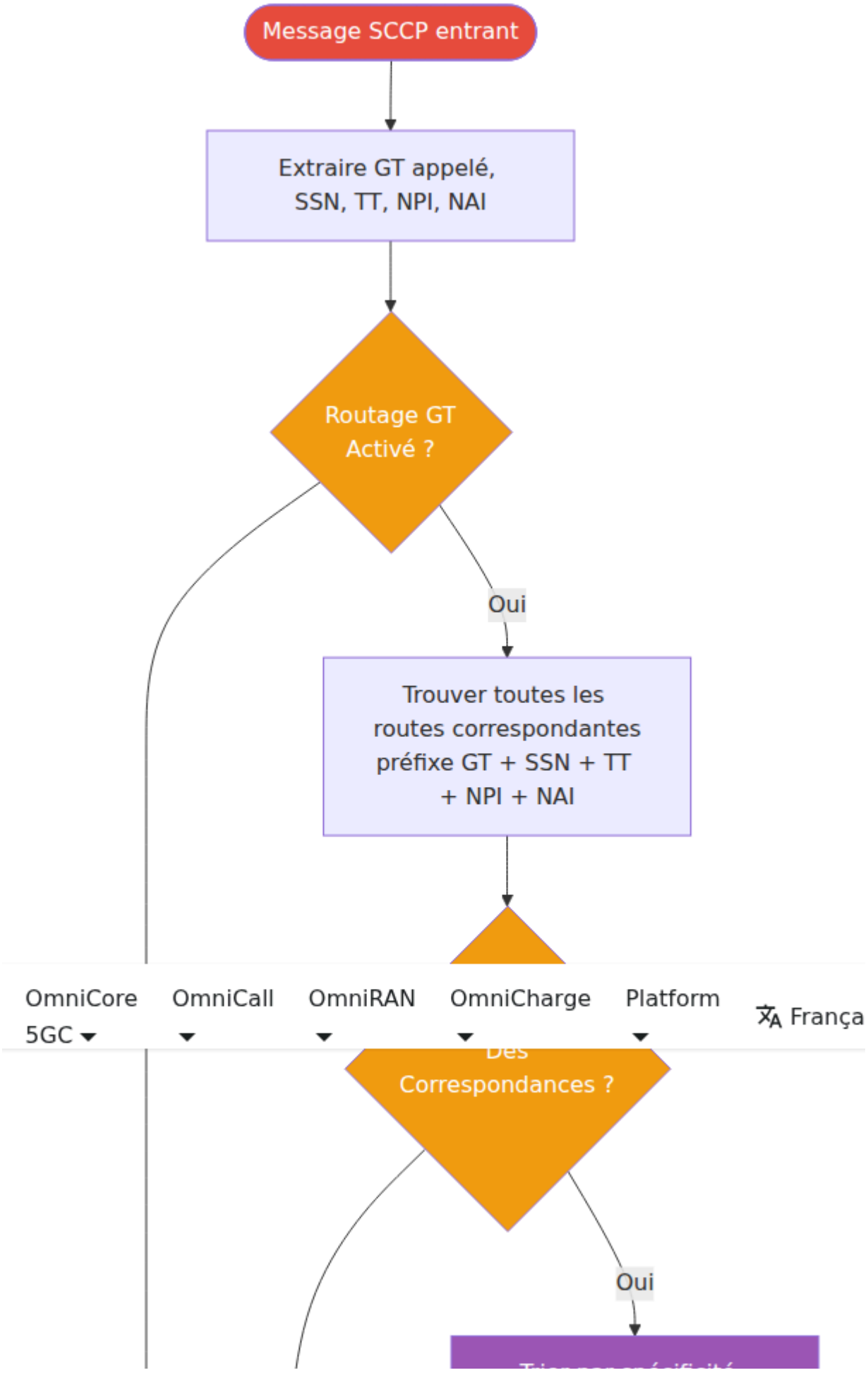
Le STP suit la disponibilité de chaque code de point de destination à l'aide de messages de gestion de route de signalisation MTP3, conformément à ITU-T Q.704 §13. L'état est conservé dans la table Mnesia `m3ua_dest_state` ; l'absence d'une ligne signifie que la destination est **autorisée** (par défaut).

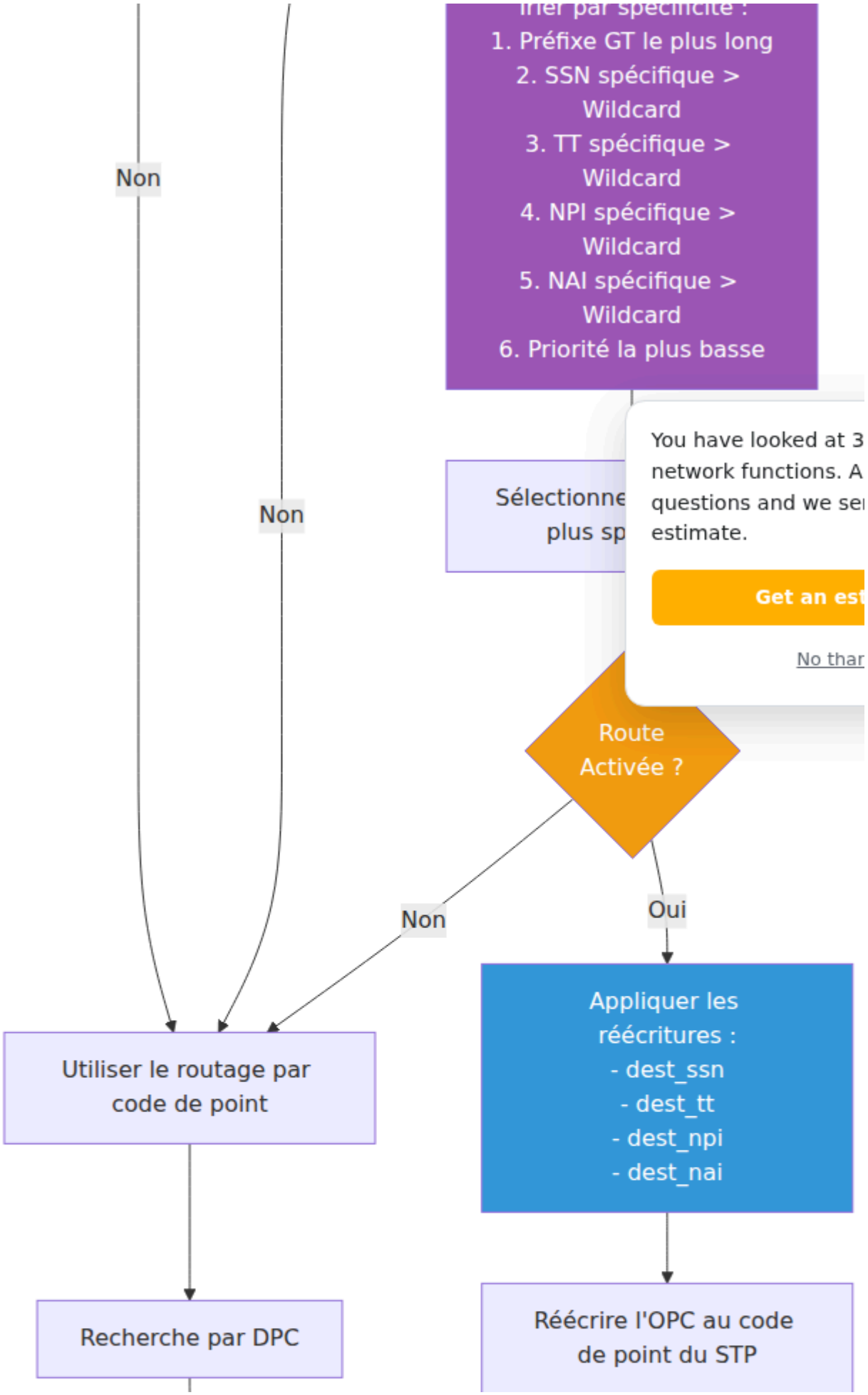
## Gestion TFP / TFA / TFR

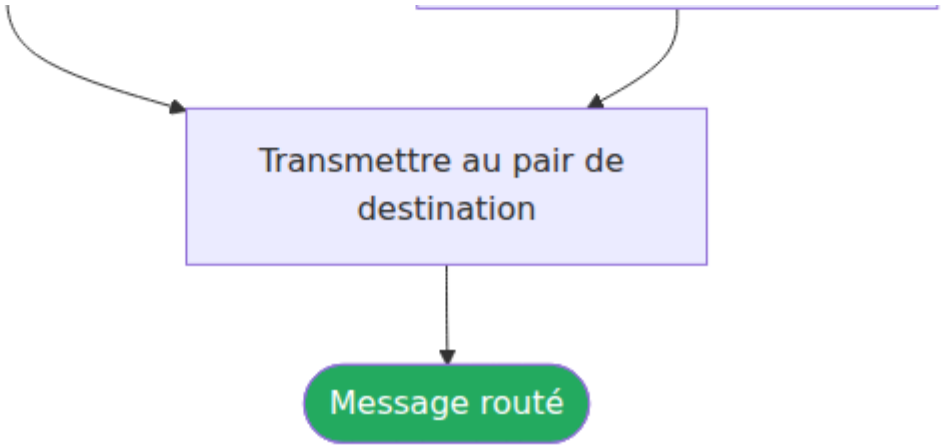
| Message | Signification | Effet sur l'état du STP |
|---|---|---|
| **TFP** (Transfert Interdit) | Une destination n'est plus accessible via cette route | Destination marquée `:prohibited` |
| **TFA** (Transfert Autorisé) | Une destination est de nouveau accessible | État de destination effacé (retour à autorisé) |
| **TFR** (Transfert Restreint) | Une destination ne doit être utilisée que lorsqu'aucune alternative n'existe | Destination marquée `:restricted` |

## Effet sur le routage

- Lors d'une recherche de code de point, le STP vérifie d'abord l'état de destination. Une destination marquée **transfert interdit** est ignorée. La recherche retourne `{:error, :prohibited}` et le message n'est pas routé là (Q.704 §13.2.3). Cela incrémente `m3ua_stp_routing_failures_total{reason="prohibited"}`.
- Les demandes **DAUD** (Audit de Destination) sont répondues à partir de cet état réel plutôt qu'avec un "disponible" global : les destinations accessibles sont rapportées avec **DAVA**, et les destinations inaccessibles ou interdites avec **DUNA** (RFC 4666 §3.4.3).


Message SCCP entrant
Extraire GT appelé,
SSN, TT, NPI, NAI
Routage GT
Activé ?
Oui
Trouver toutes les
routes correspondantes
préfixe GT + SSN + TT
+ NPI + NAI
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
França
Correspondances ?
Oui


Non
Non
1. Préfixe GT le plus long
2. SSN spécifique > Wildcard
3. TT spécifique > Wildcard
4. NPI spécifique > Wildcard
5. NAI spécifique > Wildcard
6. Priorité la plus basse
Route Activée ?
Non
Oui
Utiliser le routage par code de point
Appliquer les réécritures :
- dest_ssn
- dest_tt
- dest_npi
- dest_nai
Recherche par DPC
Réécrire l'OPC au code de point du STP




---

# Filtrage de passerelle (Pare-feu SCCP)

Le filtrage de passerelle permet au STP **d'autoriser ou de refuser le trafic de transit** avant qu'il ne soit routé. Sans cela, le STP transmet chaque message sans condition. Avec cela, les opérateurs peuvent bloquer le signalement indésirable à la passerelle. C'est un contrôle de sécurité SS7 essentiel.

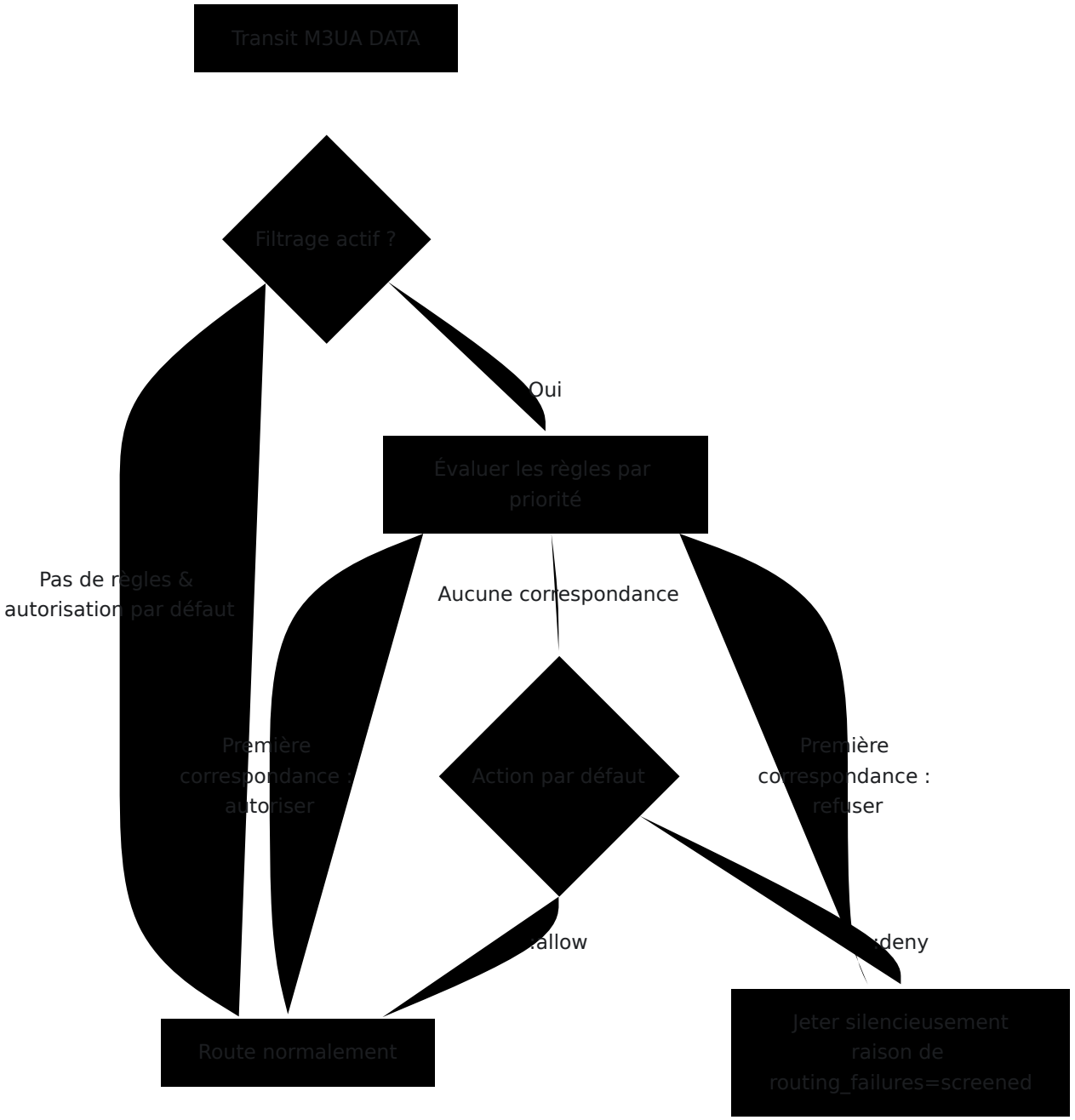
Chaque message M3UA DATA de transit est évalué par rapport à un ensemble ordonné de règles. Une règle correspond à toute combinaison des attributs ci-dessous (un attribut non défini est un wildcard) :

| Attribut | Correspond à |
|---|---|
| Pair source | Le pair d'où le message est arrivé |
| OPC | Code de point d'origine |
| DPC | Code de point de destination |
| Indicateur de service | SI MTP3 |
| Préfixe GTA de la partie appelée | Titre Global de la Partie Appelée SCCP (correspondance de préfixe) |
| Préfixe GTA de la partie appelante | Titre Global de la Partie Appelante SCCP (correspondance de préfixe) |
| SSN | Numéro de sous-système SCCP |

La **première règle activée correspondante** (valeur de `priority` la plus basse en premier) décide de l'issue : `allow` ou `deny`. Si aucune règle ne correspond, l'**action par défaut** configurable s'applique (`config :omniss7, :screening_default_action`, par défaut `:allow`).

Un message refusé est **jeté silencieusement** et compté comme `m3ua_stp_routing_failures_total{reason="screened"}`. Aucun UDTS/retour n'est généré, car un pare-feu ne doit pas divulguer le blocage au l'expéditeur.

## Comportement




L'analyse est complètement ignorée (coût zéro sur le chemin critique) lorsque aucune règle n'est configurée **et** que l'action par défaut est `:allow` ; elle devient active dès qu'une règle existe ou que l'action par défaut est définie sur `:deny`.

## Action par Défaut

| Paramètre | Type | Par Défaut | Description |
|---|---|---|---|
| `screening_default_action` | `:allow` \| `:deny` | `:allow` | Action appliquée lorsque aucune règle ne correspond. Définir sur `:deny` pour une posture de refus par défaut (liste blanche). |

```
config :omniss7,
  # Bloquer tout sauf ce que les règles autorisent explicitement.
  screening_default_action: :deny
```

Les règles sont stockées dans la table Mnesia `ss7_screen_rule` et gérées par programmation (chaque règle porte une priorité, les attributs de correspondance ci-dessus, une action `allow`/`deny` et un indicateur `enabled`). Voir la Référence de Configuration pour le paramètre d'action par défaut.

---

# Fonctionnalités de Gestion des Routes

## Désactivation des Routes

Les routes peuvent être temporairement désactivées sans les supprimer. Cela est utile pour les tests, la maintenance ou la gestion du trafic.

#### Indicateur Enabled

Les routes de Code de Point et de Titre Global prennent en charge un indicateur `enabled` optionnel :

```
config :omniss7,
  m3ua_routes: [
    # Route active
    %{
      dest_pc: 100,
      peer_id: 1,
      priority: 1,
      network_indicator: :international,
      enabled: true  # La route est active (par défaut si omis)
    },

    # Route désactivée (non évaluée lors du routage)
    %{
      dest_pc: 200,
      peer_id: 2,
      priority: 1,
      network_indicator: :international,
      enabled: false  # La route est désactivée
    }
  ],

  m3ua_gt_routes: [
    # Route GT désactivée
    %{
      gt_prefix: "44",
      peer_id: 1,
      priority: 1,
      description: "Numéros UK - temporairement désactivés",
      enabled: false
    }
  ]
```

**Comportement par Défaut :**

- Si `enabled` n'est pas spécifié, les routes par défaut sont `enabled: true`
- Les routes désactivées sont complètement ignorées lors de la recherche de routes
- Utilisez l'interface Web pour activer/désactiver les routes sans modifier la configuration

**Cas d'Utilisation :**

- Tester les changements de flux de trafic
- Fenêtres de maintenance temporaires
- Tests A/B de différents chemins de routage
- Déploiement progressif de nouvelles routes

---

# Routes DROP : Prévenir les Boucles de Routage

Les routes DROP (avec `peer_id: 0`) rejettent silencieusement le trafic au lieu de le transmettre. Cela empêche les boucles de routage et permet un filtrage avancé du trafic.

### Configuration des Routes DROP

```
config :omniss7,
  m3ua_routes: [
    # Route DROP pour un code de point spécifique
    %{
      dest_pc: 999,
      peer_id: 0,        # peer_id=0 signifie DROP
      priority: 1,
      network_indicator: :international
    }
  ],

  m3ua_gt_routes: [
    # Route DROP pour le préfixe GT
    %{
      gt_prefix: "999",
      peer_id: 0,        # peer_id=0 signifie DROP
      priority: 99,
      description: "Bloquer la plage de test"
    }
  ]
```

### Comment Fonctionnent les Routes DROP

Lorsqu'un message correspond à une route DROP :

1. Le moteur de routage identifie `peer_id: 0`
2. Le message est **silencieusement rejeté** (non transmis)
3. Un **journal INFO** est généré : `"DROP route matched for DPC 999"` ou `"DROP route matched for GT 999"`
4. La recherche de routage retourne `{:error, :dropped}`

**Important :** Le trafic rejeté est enregistré au niveau INFO pour le suivi et le dépannage.

**Cas d'Utilisation Commun : Liste Blanche de Préfixes**

L'un des usages les plus puissants des routes DROP est **la liste blanche de préfixes** - permettant uniquement des numéros spécifiques dans une large plage tout en bloquant tous les autres.

**Le Modèle :**

1. Créer une route DROP pour l'ensemble du préfixe avec un **numéro de priorité élevé** (par exemple, 99)
2. Créer des routes d'autorisation spécifiques pour des numéros individuels avec des **numéros de priorité faibles** (par exemple, 1)
3. Puisque les numéros de priorité inférieurs sont évalués en premier, les routes autorisées correspondent avant la route DROP
4. Tout numéro non explicitement autorisé est capturé par la route DROP

**Scénario d'Exemple :**

Vous avez un préfixe GT `1234` qui représente une plage de 10 000 numéros (1234000000 - 1234999999), mais vous ne souhaitez router que 3 numéros spécifiques : `1234567890`, `1234555000`, et `1234111222`.

```
config :omniss7,
  m3ua_gt_routes: [
    # Route DROP avec un numéro de priorité ÉLEVÉ (évaluée en
dernier)
    %{
      gt_prefix: "1234",
      peer_id: 0,              # DROP
      priority: 99,            # Numéro élevé = faible priorité =
évalué en dernier
      description: "Bloquer tous les 1234* sauf les numéros sur
liste blanche"
    },

    # Routes d'autorisation spécifiques avec des numéros de
priorité FAIBLES (évalués en premier)
    %{
      gt_prefix: "1234567890",
      peer_id: 1,              # Route vers le pair 1
      priority: 1,             # Numéro faible = haute priorité =
évalué en premier
      description: "Numéro autorisé 1"
    },
    %{
      gt_prefix: "1234555000",
      peer_id: 1,
      priority: 1,
      description: "Numéro autorisé 2"
    },
    %{
      gt_prefix: "1234111222",
      peer_id: 1,
      priority: 1,
      description: "Numéro autorisé 3"
    }
  ]
```

**Comportement de Routage :**

| GT Entrant | Routes Correspondantes | Route Sélectionnée | Action |
|---|---|---|---|
| 1234567890 | ✅ "1234567890" (priorité 1)<br>✅ "1234" DROP (priorité 99) | "1234567890" (la plus spécifique, la plus haute priorité) | **Routé vers le pair 1** |
| 1234555000 | ✅ "1234555000" (priorité 1)<br>✅ "1234" DROP (priorité 99) | "1234555000" (la plus spécifique, la plus haute priorité) | **Routé vers le pair 1** |
| 1234111222 | ✅ "1234111222" (priorité 1)<br>✅ "1234" DROP (priorité 99) | "1234111222" (la plus spécifique, la plus haute priorité) | **Routé vers le pair 1** |
| 1234999999 | ✅ "1234" DROP (priorité 99) | "1234" DROP (seule correspondance) | **Rejeté + enregistré** |
| 1234000000 | ✅ "1234" DROP (priorité 99) | "1234" DROP (seule correspondance) | **Rejeté + enregistré** |

**Résultat :**

- ✅ Seuls 3 numéros spécifiques sont routés vers le pair 1
- ✅ Tous les autres numéros `1234*` sont silencieusement rejetés
- ✅ Tout le trafic rejeté est enregistré pour le suivi

**Journaux Générés :**

```
[INFO] DROP route matched for GT 1234999999
[INFO] DROP route matched for GT 1234000000
```

**Routes DROP pour les Codes de Point**

Le même modèle de liste blanche fonctionne pour le routage des Codes de Point :

```
config :omniss7,
  m3ua_routes: [
    # DROP toute la plage /8 (64 codes de point : 1000-1063)
    %{
      dest_pc: 1000,
      peer_id: 0,
      priority: 99,
      mask: 8,
      network_indicator: :international
    },

    # Autoriser des PC spécifiques
    %{dest_pc: 1010, peer_id: 1, priority: 1, network_indicator:
:international},
    %{dest_pc: 1020, peer_id: 1, priority: 1, network_indicator:
:international},
    %{dest_pc: 1030, peer_id: 1, priority: 1, network_indicator:
:international}
  ]
```

**Résultat :** Seuls les PC 1010, 1020 et 1030 sont routés. Tous les autres PC de la plage 1000-1063 sont rejetés.

**Surveillance des Routes DROP**

**Vérifier les Journaux :**

```
# Surveiller le trafic rejeté
tail -f logs/app.log | grep "DROP route matched"

# Sortie attendue :
[INFO] DROP route matched for GT 1234999999
[INFO] DROP route matched for DPC 1050
```

**Via l'Interface Web :**

- Naviguez vers l'onglet **Journaux Système**

- Filtrer par niveau `INFO`
- Rechercher "DROP route matched"

**Meilleures Pratiques :**

1. ⚠ Surveiller régulièrement les journaux pour s'assurer que les routes DROP ne bloquent pas le trafic légitime
2. □ Utiliser des champs `description` descriptifs pour documenter pourquoi les routes sont rejetées
3. □ Utiliser des numéros de priorité élevés (90-99) pour les routes DROP afin de s'assurer qu'elles sont des routes de collecte
4. □ Tester le comportement des routes DROP avant de déployer en production
5. □ Configurer des alertes pour des augmentations inattendues du trafic rejeté

---

# Routage Avancé : Routage et Réécriture Basés sur le SSN

## Numéros de Sous-Système (SSN)

Les Numéros de Sous-Système identifient la couche d'application :

- **SSN 6** : HLR (Registre de Localisation Domiciliaire)
- **SSN 7** : VLR (Registre de Localisation Visiteur)
- **SSN 8** : MSC (Centre de Commutation Mobile) / SMSC (Centre SMS)
- **SSN 9** : GMLC (Centre de Localisation Mobile de Passerelle)

## Exemple de Routage Basé sur le SSN

Routage du trafic SMS vers différents HLR en fonction du préfixe de numéro :

```
m3ua_gt_routes: [
  # Routage SMS pour les numéros UK vers HLR UK, réécriture SSN de
8 (SMSC) à 6 (HLR)
  %{
    gt_prefix: "44",
    source_ssn: 8,                    # Correspondre au SSN
entrant 8 (SMSC)
    peer_id: 1,
    dest_ssn: 6,                      # Réécriture vers SSN 6
(HLR)
    priority: 1,
    description: "SMS UK vers HLR"
  },

  # Routage du trafic vocal pour les numéros UK (SSN 6) sans
réécriture
  %{
    gt_prefix: "44",
    source_ssn: 6,                    # Correspondre au SSN
entrant 6 (HLR)
    peer_id: 1,
    dest_ssn: nil,                    # Pas de réécriture de SSN
    priority: 1,
    description: "Trafic vocal UK"
  }
]
```

---

# Tester la Configuration de Routage STP

Après avoir configuré les pairs et les routes, vérifiez votre configuration :

## 1. Vérifier l'État des Pairs

**Via l'Interface Web :**

- Naviguez vers http://localhost

- Vérifiez la page d'État M3UA
- Vérifiez que les pairs affichent **État : ACTIF**

**Via la Console IEx :**

```
# Obtenir tous les états des pairs
SCTP.SocketHandler.get_peers_status()

# Sortie attendue :
# [
#   %{peer_id: 1, name: "Partner_STP_West", status: :active,
point_code: 100, ...},
#   %{peer_id: 2, name: "Local_HLR", status: :active, point_code:
200, ...}
# ]
```

# 2. Tester le Routage par Code de Point

```
# Envoyer un message M3UA de test au DPC 100
test_payload = <<1, 2, 3, 4>>  # Charge utile fictive
SCTP.SocketHandler.route_by_pc(100, test_payload, 0)

# Vérifiez les journaux pour la décision de routage
# Journal attendu : "Routage du message : OPC=... -> DPC=100 via
pair 1"
```

## 3. Tester le Routage par Titre Global

```
# Rechercher manuellement la route GT
M3UARouting.lookup_peer_by_gt("447712345678")

# Renvoie {:ok, peer, rewrites}. `peer` est l'enregistrement
m3ua_peer de 13 éléments ;
# `rewrites` est une MAP des réécritures d'indicateurs (nil =
conserver l'original) :
# {:ok, {:m3ua_peer, 3, "UK_Mobile_Peer", ...},
#       %{dest_ssn: nil, dest_tt: nil, dest_npi: nil, dest_nai:
nil}}

# Rechercher la route GT avec SSN (une route qui réécrit SSN 8 →
6)
M3UARouting.lookup_peer_by_gt("555881234567", 8)

# La réécriture de SSN apparaît dans la carte de réécritures, PAS
en tant qu'entier brut :
# {:ok, {:m3ua_peer, 4, "SMS_HLR_Peer", ...},
#       %{dest_ssn: 6, dest_tt: nil, dest_npi: nil, dest_nai:
nil}}
```

## 4. Surveiller les Métriques de Routage

Accédez aux métriques Prometheus à `/metrics`

Métriques clés :

```
# Messages reçus par pair
m3ua_stp_messages_received_total{peer_name="Partner_STP_West",point_c
1523

# Messages envoyés par pair
m3ua_stp_messages_sent_total{peer_name="Local_HLR",point_code="200"}

# Échecs de routage (par raison)
m3ua_stp_routing_failures_total{reason="no_route"} 5
m3ua_stp_routing_failures_total{reason="no_gt_route"} 2
m3ua_stp_routing_failures_total{reason="no_active_peer"} 1
m3ua_stp_routing_failures_total{reason="prohibited"} 3
m3ua_stp_routing_failures_total{reason="hcounter_violation"} 4
m3ua_stp_routing_failures_total{reason="screened"} 7
```

---

# Métriques et Surveillance STP

## Métriques Disponibles

**Métriques de Trafic par Pair :**

- `m3ua_stp_messages_received_total` - Total des messages reçus de chaque pair
  - Étiquettes : `peer_name`, `point_code`
- `m3ua_stp_messages_sent_total` - Total des messages transférés à chaque pair
  - Étiquettes : `peer_name`, `point_code`

**Métriques d'Échec de Routage :**

- `m3ua_stp_routing_failures_total` - Compte des échecs de routage par raison
  - Étiquettes : `reason`

| `reason` | Signification |
|---|---|
| `no_route` | Aucune route de Code de Point trouvée pour la destination |
| `no_gt_route` | Aucune route de Titre Global trouvée, et le routage PC a également échoué |
| `no_active_peer` | Une route a été trouvée, mais son pair est inconnu/non actif |
| `prohibited` | La destination est marquée comme interdite au transfert (un TFP a été reçu, Q.704 §13) |
| `hcounter_violation` | Le compteur de saut SCCP a atteint zéro sur un XUDT/LUDT (Q.714 §2.8.6) |

# Interprétation des Métriques

- **Comptes de messages élevés** : Indiquent un flux de trafic actif
- **Échecs de routage** : Indiquent des routes manquantes, une destination inaccessible ou un message en boucle
  - `no_route` : Aucune route de Code de Point trouvée pour la destination
  - `no_gt_route` : Aucune route de Titre Global trouvée, et le routage PC a également échoué
  - `no_active_peer` : Une route existe mais le pair cible est hors ligne/inconnu
  - `prohibited` : Destination marquée comme interdite au transfert par la gestion des routes de signalisation
  - `hcounter_violation` : Compteur de saut épuisé ; le message a été rejeté (et retourné si le retour en cas d'erreur était défini)

## Dépannage avec les Métriques

**Scénario : Aucun trafic atteignant la destination**

1. Vérifiez si des messages sont reçus :

```
m3ua_stp_messages_received_total{peer_name="Source_Peer"} > 0
```

2. Vérifiez si des messages sont envoyés :

```
m3ua_stp_messages_sent_total{peer_name="Dest_Peer"} > 0
```

3. Vérifiez les échecs de routage :

```
m3ua_stp_routing_failures_total{reason="no_route"} > 0
```

**Solution** : Si les échecs de routage sont élevés, ajoutez les routes manquantes dans la configuration.

---

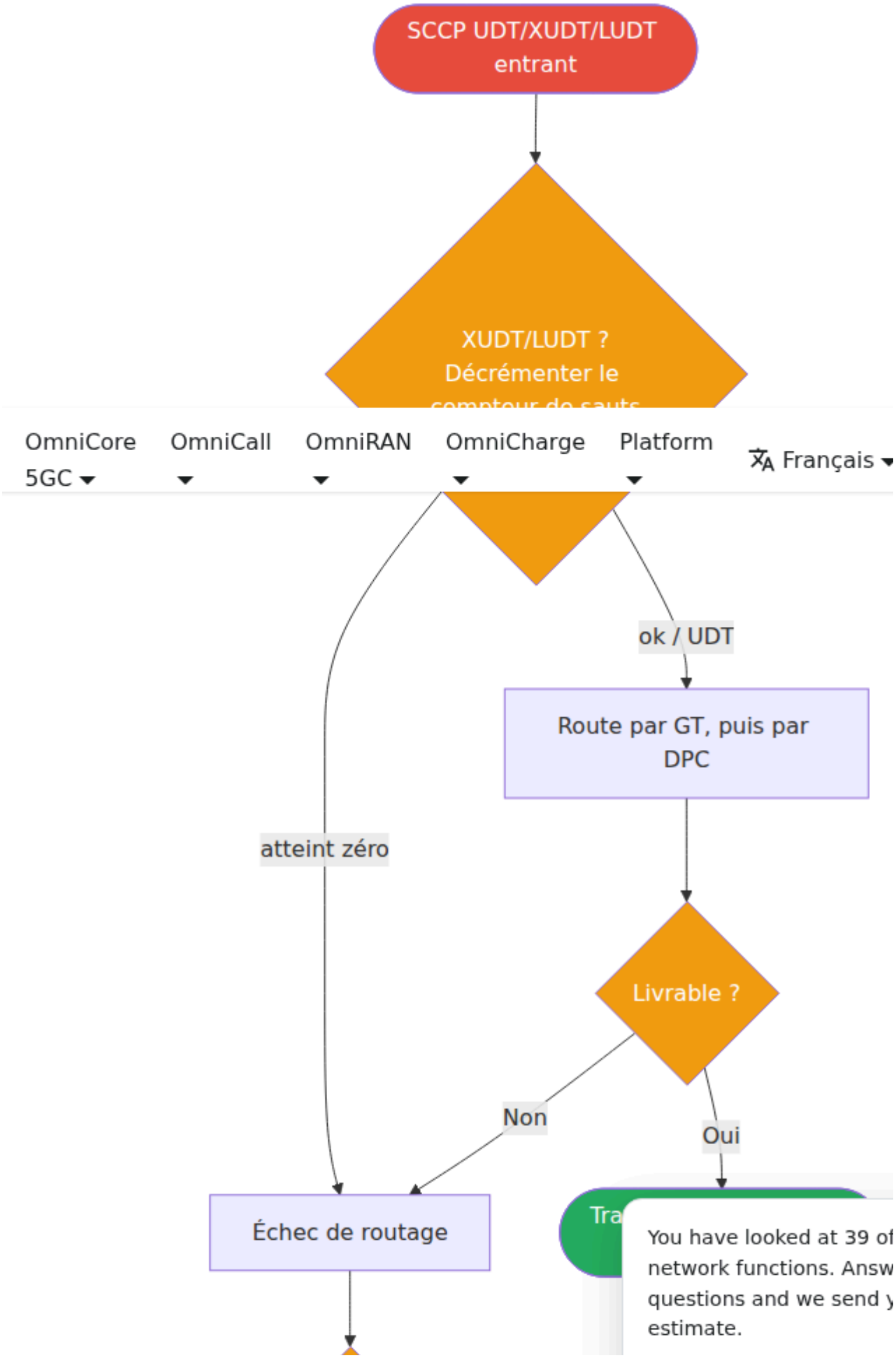
# Surveillance de l'État des Pairs M3UA

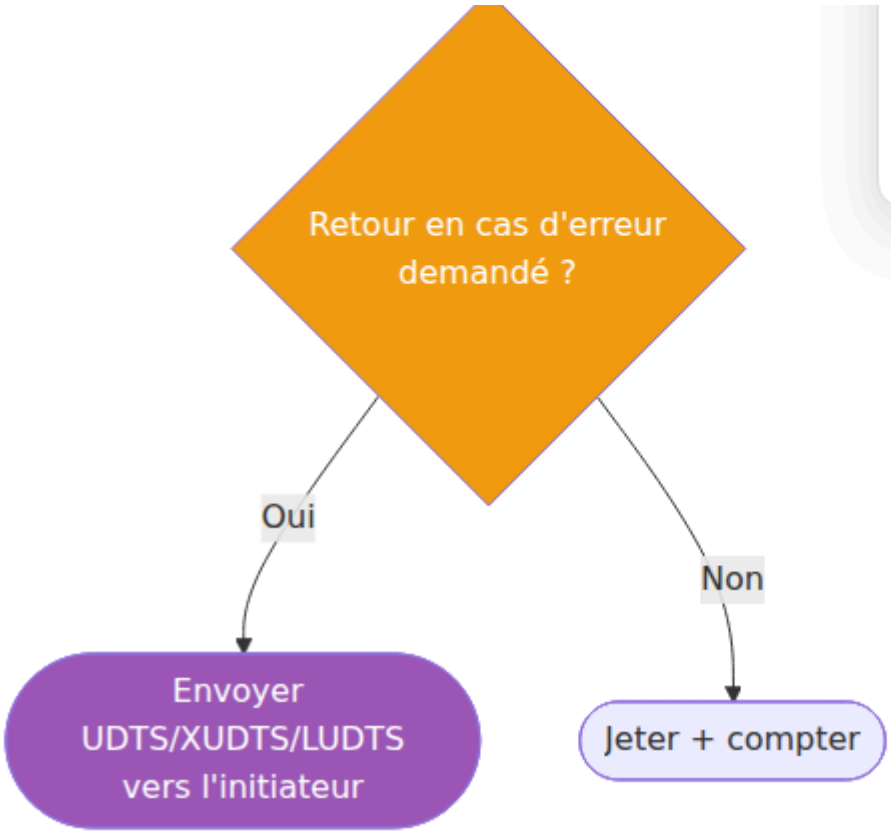
## Comprendre M3UA

M3UA (Couche d'Adaptation Utilisateur MTP3) est un protocole qui permet le transport du signalement SS7 sur des réseaux IP utilisant SCTP.

## États de Connexion M3UA

Les connexions M3UA progressent à travers plusieurs états :


SCCP UDT/XUDT/LUDT entrant
XUDT/LUDT ? Décrémenter le
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
ok / UDT
Route par GT, puis par DPC
atteint zéro
Livrable ?
Non
Oui
Échec de routage
You have looked at 39 of
network functions. Answ
questions and we send y
estimate.




**Descriptions des États :**

- **DOWN** - Pas de connexion SCTP
- **CONNECTING** - Connexion SCTP en cours
- **ASPUP_SENT** - En attente de l'accusé de réception ASPUP
- **INACTIVE** - ASP est actif mais ne transporte pas activement de trafic (peut passer à ACTIVE via ASPAC)
- **ASPAC_SENT** - En attente de l'accusé de réception ASPAC
- **ACTIVE** - Prêt pour le trafic, entièrement opérationnel
- **ASPDOWN_SENT** - Arrêt gracieux en cours

## Surveillance des Pairs M3UA via l'Interface Web

L'interface Web fournit une surveillance en temps réel des connexions de pairs M3UA.

**Accéder à la Page d'État M3UA :**

1. Naviguez vers la page d'accueil de l'interface Web

2. Cliquez sur "État M3UA" dans le menu de navigation
3. La page se rafraîchit automatiquement chaque seconde

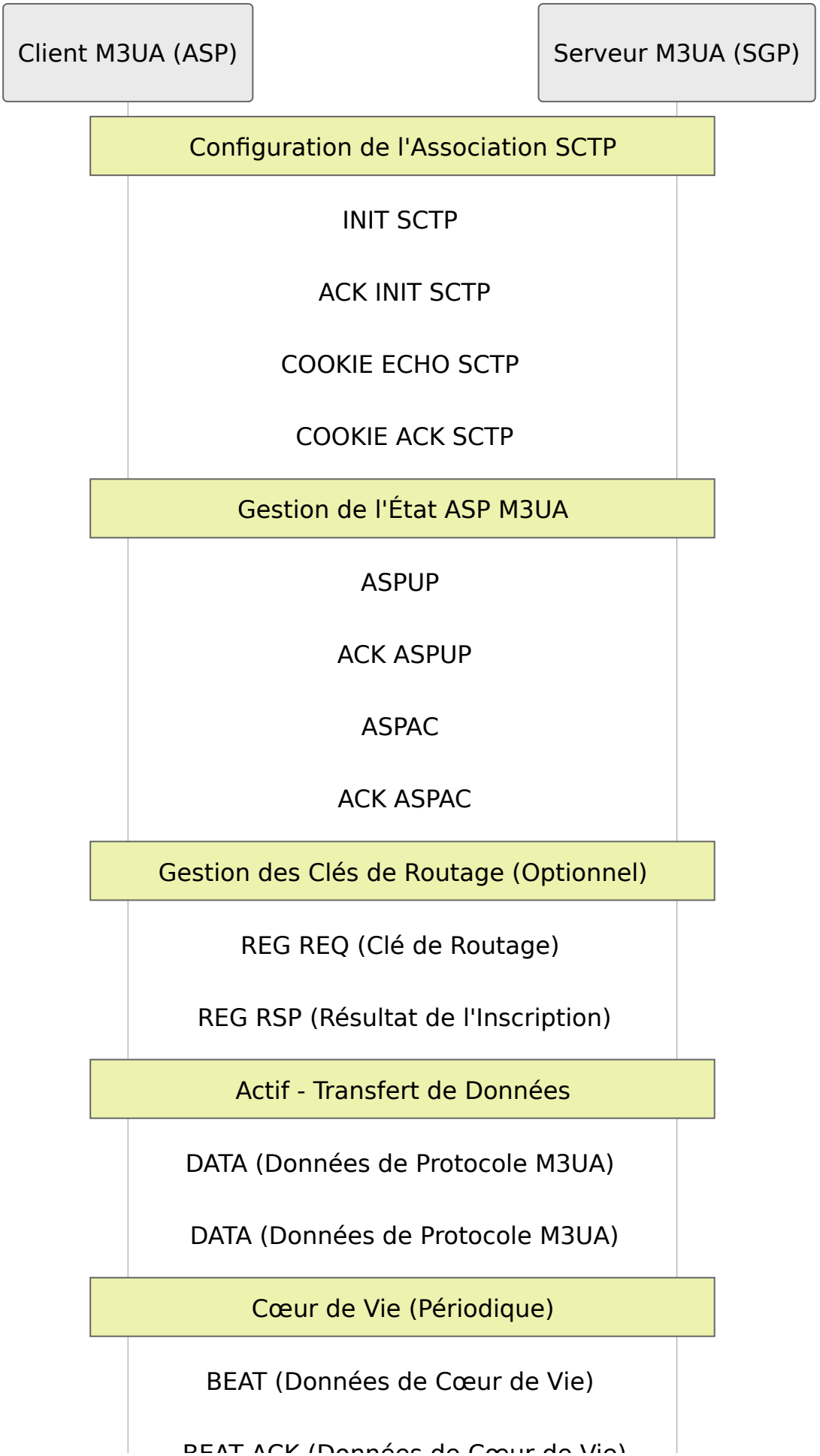
**Tableau d'État M3UA :**

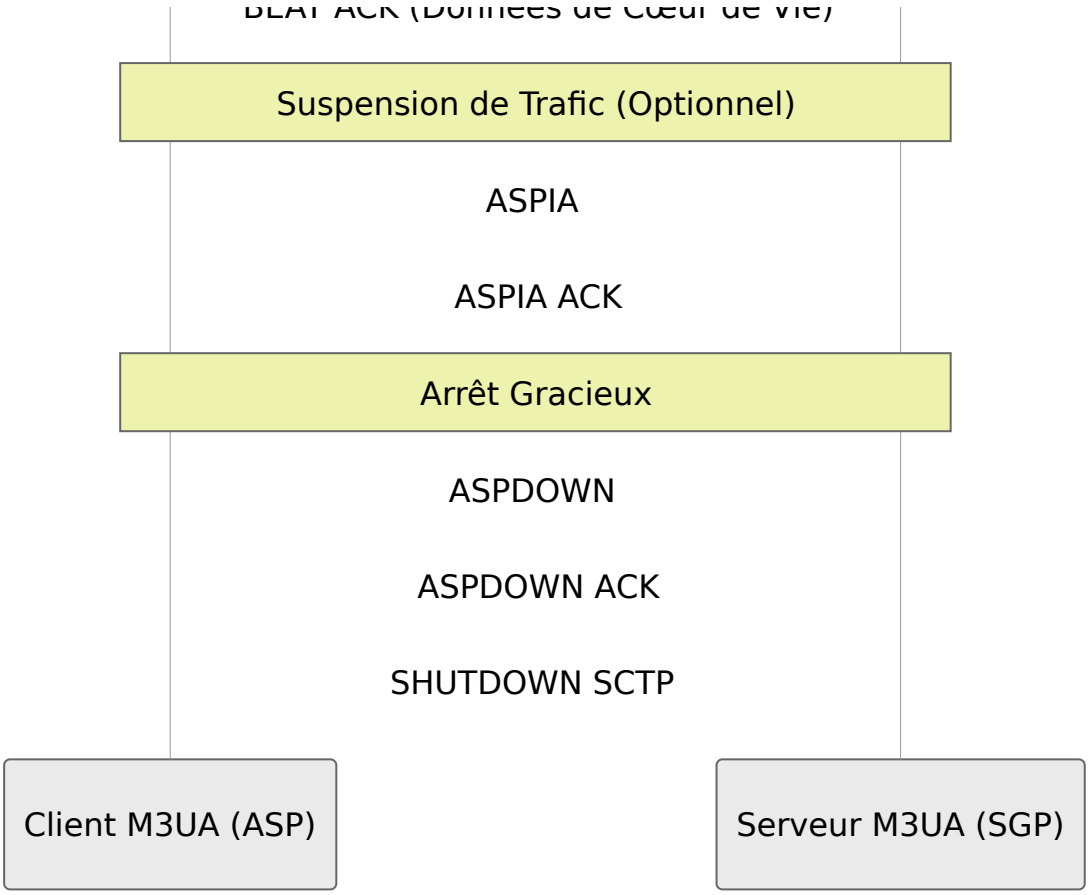
| Colonne | Description |
|---|---|
| **Nom** | Nom de la connexion (par exemple, `testASP`) |
| **PID** | Identifiant de processus |
| **État** | UP (vert) ou DOWN (rouge) |
| **État ASP** | État M3UA actuel (par exemple, ACTIVE, INACTIVE) |
| **Assoc/SCTP** | État d'association SCTP |
| **Local** | IP:Port local |
| **Remote** | IP:Port distant |
| **RC** | Identifiant de Contexte de Routage |

**Indicateurs d'État :**

- **Vert (UP)** - La connexion est active et saine
- **Rouge (DOWN)** - La connexion est hors ligne ou indisponible
- **État ASP** - Affiche l'état de connexion M3UA actuel
- **Assoc/SCTP** - Affiche l'état de l'association SCTP

# Flux de Messages M3UA


Client M3UA (ASP)
Serveur M3UA (SGP)
Configuration de l'Association SCTP
INIT SCTP
ACK INIT SCTP
COOKIE ECHO SCTP
COOKIE ACK SCTP
Gestion de l'État ASP M3UA
ASPUP
ACK ASPUP
ASPAC
ACK ASPAC
Gestion des Clés de Routage (Optionnel)
REG REQ (Clé de Routage)
REG RSP (Résultat de l'Inscription)
Actif - Transfert de Données
DATA (Données de Protocole M3UA)
DATA (Données de Protocole M3UA)
Cœur de Vie (Périodique)
BEAT (Données de Cœur de Vie)




## Support des Messages M3UA selon la RFC 4666

OmniSS7 implémente l'ensemble complet des types de messages M3UA définis dans la RFC 4666 :

| Classe | Type | Message | Direction | Comportement |
|---|---|---|---|---|
| 0 (MGMT) | 0 | ERR | Les deux | Enregistre le code d'erreur et les informations de diagnostic |
| 0 (MGMT) | 1 | NTFY | Les deux | Enregistre la notification |
| 1 (XFER) | 1 | DATA | Les deux | Route la charge utile via PC ou GT |
| 2 (SSNM) | 1 | DUNA | Recevoir | Enregistre la destination indisponible |
| 2 (SSNM) | 2 | DAVA | Recevoir | Enregistre la destination disponible |
| 2 (SSNM) | 3 | DAUD | Recevoir | Répond selon l'état de destination : DAVA pour les PC accessibles, DUNA pour les PC inaccessibles/interdits |
| 2 (SSNM) | 4 | SCON | Recevoir | Enregistre la congestion |
| 2 (SSNM) | 5 | DUPU | Recevoir | Enregistre la partie utilisateur indisponible |
| 2 (SSNM) | 6 | DRST | Recevoir | Enregistre la destination restreinte |
| 3 (ASPSM) | 1 | ASPUP | Les deux | Envoie/reçoit ACK ASPUP |
| 3 (ASPSM) | 2 | ASPDN | Les deux | Envoie ACK ASPDN, supprime les routes de pair |

| Classe | Type | Message | Direction | Comportement |
|---|---|---|---|---|
| 3 (ASPSM) | 3 | BEAT | Les deux | Répond aux données de cœur de vie dans BEAT ACK |
| 3 (ASPSM) | 4 | ACK ASPUP | Recevoir | Transitionne le pair vers UP |
| 3 (ASPSM) | 5 | ACK ASPDN | Recevoir | Transitionne le pair vers DOWN |
| 3 (ASPSM) | 6 | ACK BEAT | Recevoir | Accuse réception du cœur de vie |
| 4 (ASPTM) | 1 | ASPAC | Les deux | Envoie/reçoit ACK ASPAC, active les routes |
| 4 (ASPTM) | 2 | ASPIA | Les deux | Envoie ACK ASPIA, transitionne le pair vers INACTIVE |
| 4 (ASPTM) | 3 | ACK ASPAC | Recevoir | Transitionne le pair vers ACTIVE |
| 4 (ASPTM) | 4 | ACK ASPIA | Recevoir | Transitionne le pair vers INACTIVE |
| 9 (RKM) | 1 | REG REQ | Réception SGP | Analyse la clé de routage, répond avec REG RSP |
| 9 (RKM) | 2 | REG RSP | Réception ASP | Enregistre le résultat de l'inscription |
| 9 (RKM) | 3 | DEREG REQ | Réception SGP | Répond avec DEREG RSP |

| Classe | Type | Message | Direction | Comportement |
|---|---|---|---|---|
| 9 (RKM) | 4 | DEREG RSP | Réception ASP | Enregistre le résultat de la désinscription |

**Cœur de Vie (BEAT/BEAT ACK) :** Lorsque le STP reçoit un message BEAT, il renvoie les données de cœur de vie dans un BEAT ACK. Cela permet aux pairs de détecter des pannes de lien sans se fier uniquement aux délais d'attente au niveau SCTP.

**Gestion des Clés de Routage (RKM) :** Lorsque un ASP envoie un REG REQ, le SGP analyse la clé de routage (identifiant RK local, contexte de routage, mode de trafic, DPC) et répond avec un REG RSP contenant l'état de l'inscription. Cela permet un approvisionnement dynamique des routes sans nécessiter de configuration statique.

**ASP Down (ASPDN) :** Lorsque le SGP reçoit un ASPDN, il répond avec un ACK ASPDN, marque le pair comme DOWN, et supprime les routes associées de la table de routage.

**ASP Inactive (ASPIA) :** Lorsque le SGP reçoit un ASPIA, il répond avec un ACK ASPIA et transitionne le pair vers INACTIVE, suspendant le trafic sans détruire l'association.

# Gestion du Contexte de Routage M3UA

Le **Contexte de Routage (RC)** est un identifiant de 32 bits dans M3UA qui identifie le contexte du Serveur d'Application (AS) à chaque point de terminaison. Comprendre comment le RC est géré à différents points du chemin de signalisation est important pour une configuration réseau correcte.

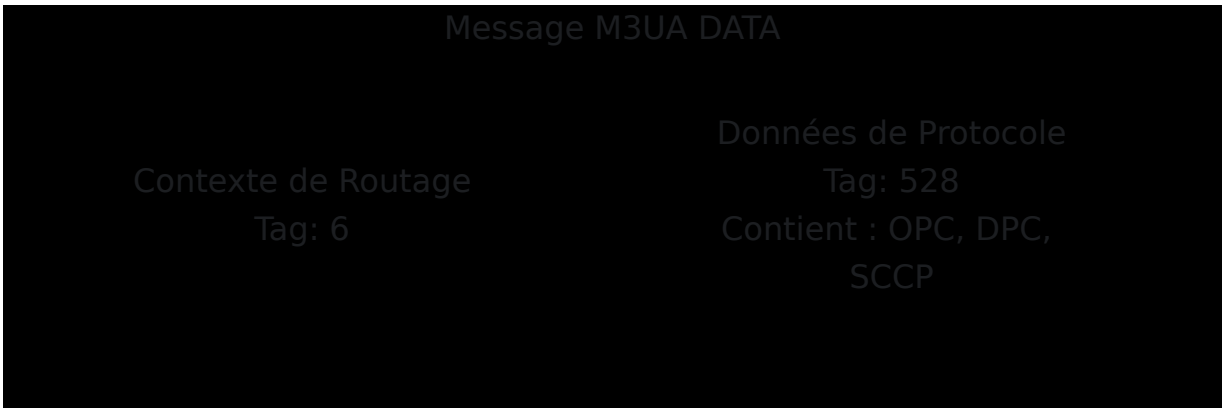
### Qu'est-ce que le Contexte de Routage ?

Selon RFC 4666 Section 3.3.1, le Contexte de Routage :

- Identifie de manière unique un Serveur d'Application (AS) à un point de terminaison
- Est transporté dans les messages M3UA DATA (tag de paramètre 6)

- Permet à une Passerelle de Signalisation de déterminer quel AS doit recevoir un message
- Est significatif par saut, pas de bout en bout

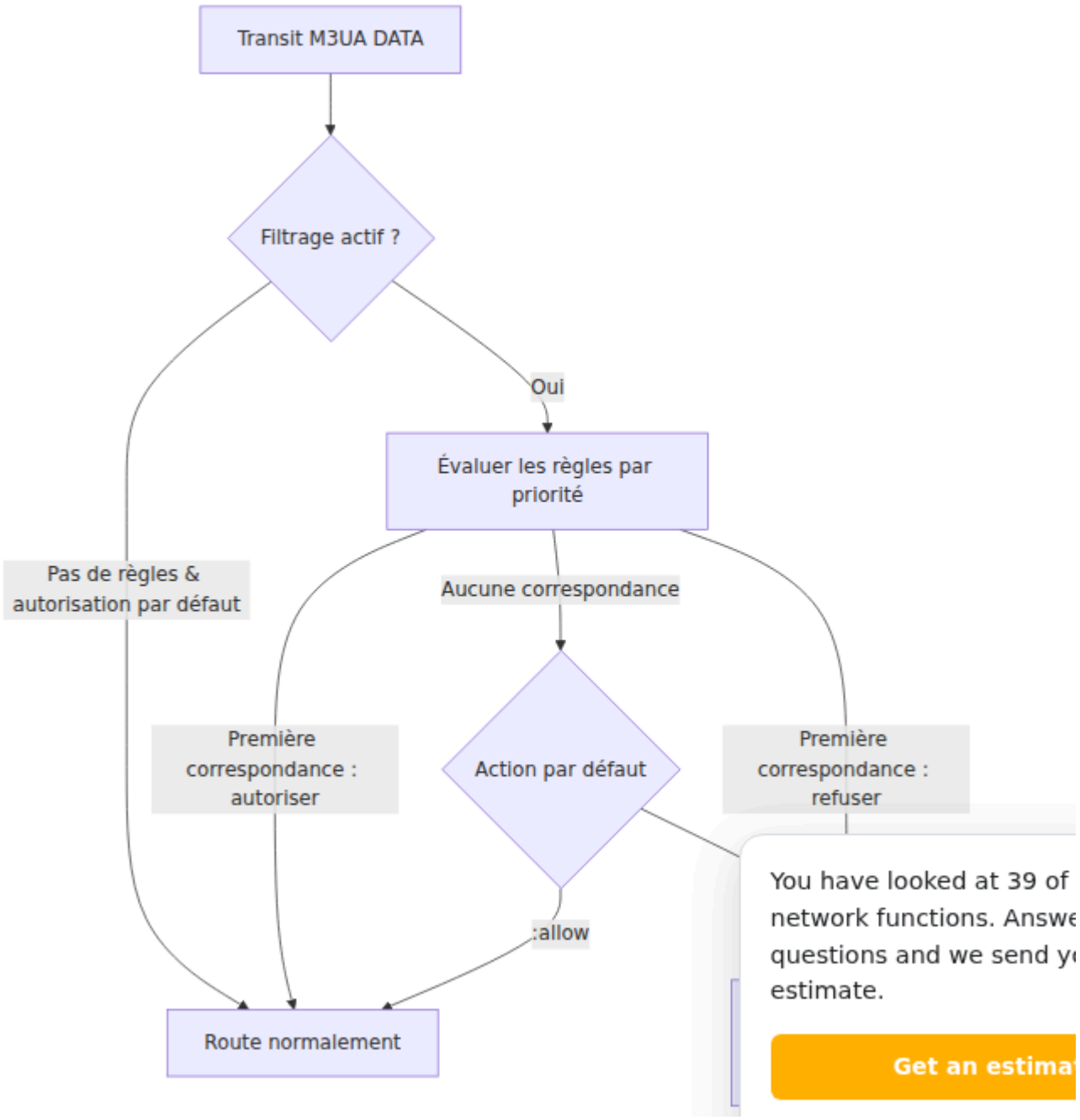
### Contexte de Routage dans le Protocole M3UA




Le paramètre de Contexte de Routage (tag 6) apparaît dans les messages M3UA DATA aux côtés du paramètre de Données de Protocole (tag 528) qui contient la charge utile MTP3/SCCP réelle.

### Comment OmniSS7 Gère le Contexte de Routage

OmniSS7 gère le Contexte de Routage différemment selon le composant et la direction du message :




**Client MAP et Client CAP**

Pour les clients de couche application (MAP et CAP), le contexte de routage est géré comme suit :

| Scénario | Gestion du RC | Configuration |
| --- | --- | --- |
| **Réponse à une demande entrante** | Miroir du RC entrant dans la réponse | Automatique - aucune configuration nécessaire |
| **Demande initiée par le client** (SRI, PRN, IDP, etc.) | Utiliser le RC configuré à partir de la configuration de l'application | Définir `routing_context` dans `map_client_m3ua` ou `cap_client_m3ua` |
| **Pas de RC dans la demande entrante** | Omettre le RC de la réponse | Maintient la compatibilité ascendante |
| **Pas de RC configuré pour les demandes des clients** | Omettre le RC du message sortant | Maintient la compatibilité ascendante |

**Exemple de Configuration :**

```
config :omniss7,
  # Configuration du RC du Client MAP
  map_client_m3ua: %{
    mode: "ASP",
    local_ip: {10, 0, 0, 1},
    local_port: 2905,
    remote_ip: {10, 0, 0, 10},
    remote_port: 2905,
    routing_context: 1    # Utilisé pour les demandes MAP initiées
par le client (SRI, PRN, etc.)
  },

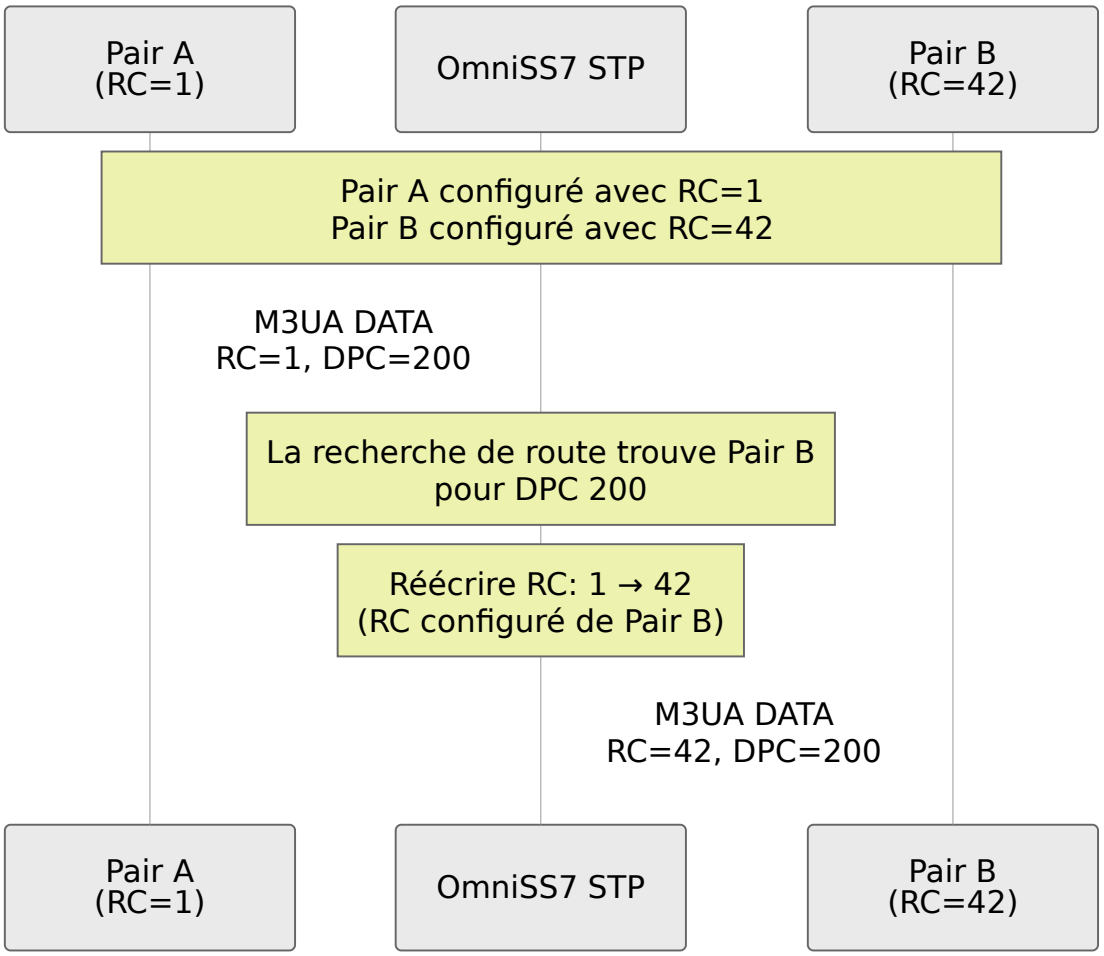
  # Configuration du RC du Client CAP
  cap_client_m3ua: %{
    mode: "ASP",
    local_ip: {10, 0, 0, 1},
    local_port: 2906,
    remote_ip: {10, 0, 0, 20},
    remote_port: 2906,
    routing_context: 2    # Utilisé pour les demandes CAP initiées
par le client (InitialDP, etc.)
  }
```

**Réécriture du Contexte de Routage STP**

Lorsque le STP transfère des messages entre pairs, il **réécrit** le Contexte de Routage au RC configuré du pair de destination. C'est un comportement correct selon la RFC 4666 car :

1. Le RC identifie le contexte AS à chaque point de terminaison
2. Différents pairs peuvent être associés à différents contextes AS
3. Le STP traduit les valeurs RC lors du routage entre différents AS

**Flux de Réécriture du RC STP :**




**Configuration :**

Le contexte de routage de chaque pair est configuré dans la définition du pair :

```
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "Network_A_STP",
      role: :server,
      remote_ip: {10, 0, 0, 100},
      remote_port: 2905,
      routing_context: 1,        # RC attendu par/de ce pair
      point_code: 100,
      network_indicator: :international
    },
    %{
      peer_id: 2,
      name: "Network_B_HLR",
      role: :client,
      local_ip: {10, 0, 0, 1},
      local_port: 0,
      remote_ip: {10, 0, 0, 200},
      remote_port: 2905,
      routing_context: 42,       # RC attendu par ce pair
      point_code: 200,
      network_indicator: :international
    }
  ]
```

Lorsqu'un message arrive de Pair 1 (RC=1) destiné à Pair 2, le STP :

1. Route le message basé sur DPC (200 → Pair 2)
2. Réécrit le RC de 1 à 42 (RC configuré de Pair 2)
3. Transmet le message à Pair 2

**Résumé de la Gestion du RC**

| Composant | Demande Entrante | Réponse Sortante | Initiée par le Client |
|---|---|---|---|
| **Client MAP** | Extraire RC | Miroir du RC entrant | Utiliser le RC configuré |
| **Client CAP** | Extraire RC | Miroir du RC entrant | Utiliser le RC configuré |
| **STP** | Extraire RC | N/A | Réécrire au RC du pair de destination |

**Lorsque le RC est Omis**

Si un pair n'a pas de `routing_context` configuré (ou s'il est défini sur `nil`) :

- **Réponses des Clients MAP/CAP** : Le paramètre RC est omis de la réponse
- **Demandes des Clients MAP/CAP** : Le paramètre RC est omis de la demande
- **Transfert STP** : Le RC n'est pas réécrit (RC original préservé ou omis)

Cela maintient la compatibilité ascendante avec les pairs qui n'utilisent pas ou n'attendent pas de contexte de routage.

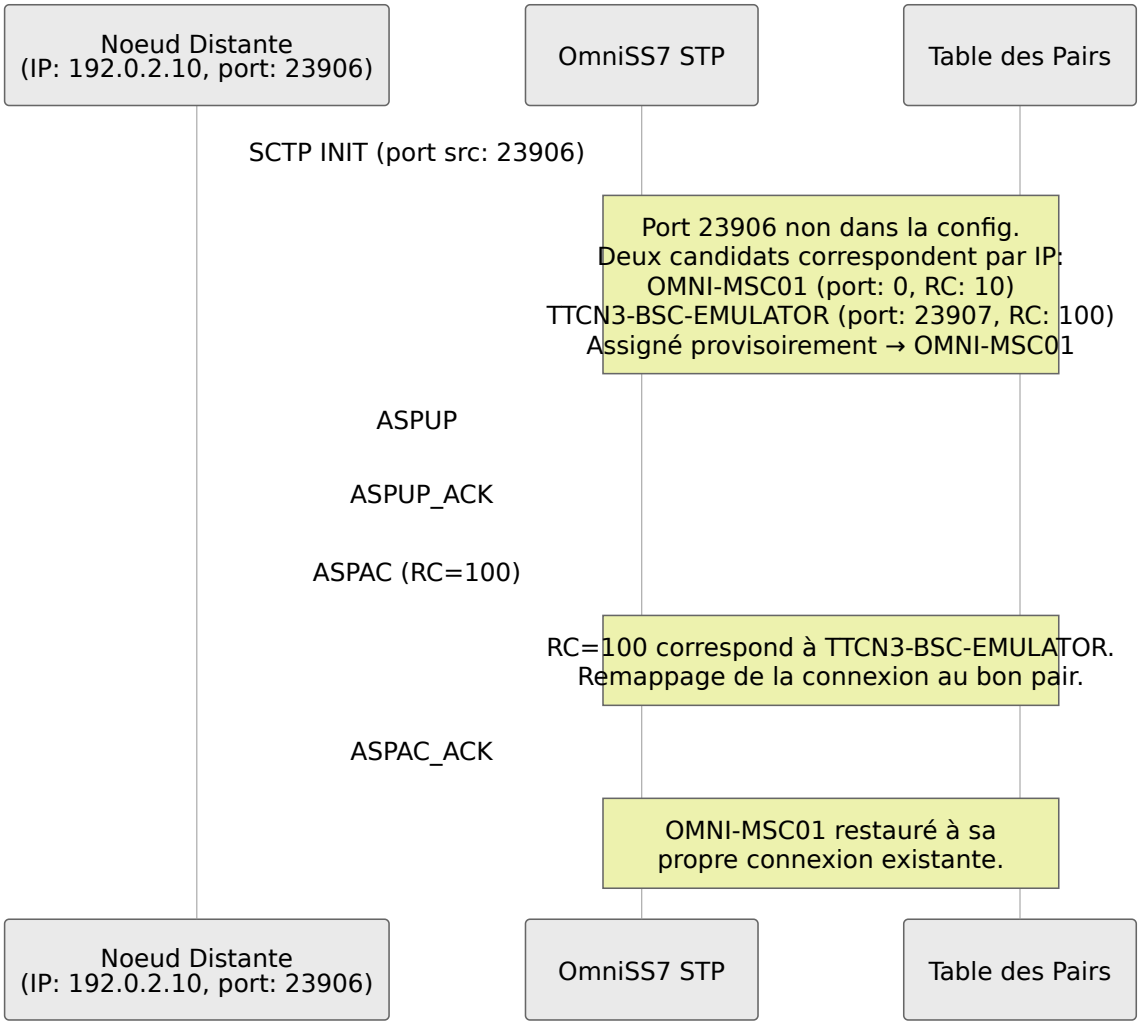
---

# Désambiguïsation Multi-Pair avec la Même IP

Deux pairs configurés peuvent partager la même adresse IP distante : par exemple, un MSC de production et un émulateur de test fonctionnant sur le même hôte. Lorsqu'une connexion SCTP entrante arrive, OmniSS7 ne peut pas déterminer la bonne identité du pair à partir de l'IP source seule. Un pair configuré avec `remote_port: 0` (joker) correspondra à toute connexion provenant de cette IP, ce qui peut entraîner qu'une nouvelle connexion de l'autre pair soit provisoirement assignée à la mauvaise entrée.

OmniSS7 résout cela automatiquement en utilisant le **contexte de routage ASPAC M3UA**. La connexion est acceptée et provisoirement assignée au pair le mieux correspondant (une correspondance de port spécifique est préférée à un joker). Lorsque l'extrémité distante envoie un message `ASPAC` portant un contexte de routage, le système compare ce RC avec le `routing_context` configuré de chaque pair. Si le RC correspond à un autre pair, la connexion est silencieusement remappée au bon.

### Comment Cela Fonctionne




### Sortie de Journal

Lorsque plusieurs pairs correspondent à une connexion entrante, OmniSS7 enregistre tous les candidats afin que les opérateurs puissent vérifier que la désambiguïsation fonctionne correctement :

```
Connexion entrante de 192.0.2.10:23906 correspond aux pairs :
["OMNI-MSC01", "TTCN3-BSC-EMULATOR"] - utilisant provisoirement
"OMNI-MSC01" jusqu'à ce que le RC ASPAC confirme l'identité
```

Après la réception du contexte de routage ASPAC :

```
RC ASPAC=100 correspond au pair 9 (TTCN3-BSC-EMULATOR), mais la
connexion a été assignée au pair 8 (OMNI-MSC01) - remappage
```

### Exigences de Configuration

Pour que la désambiguïsation des pairs avec la même IP fonctionne correctement, **chaque pair doit avoir un `routing_context` unique configuré**. Si deux pairs sur la même IP partagent le même RC (ou si les deux ont `nil`), le remappage ne peut pas déterminer quel pair est correct.

```
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 8,
      name: "OMNI-MSC01",
      role: :server,
      remote_ip: {10, 5, 198, 200},
      remote_port: 0,              # Joker - accepte tout port
source
      routing_context: 10,         # Doit être unique entre les
pairs partageant cette IP
      point_code: 500,
      network_indicator: :national
    },
    %{
      peer_id: 9,
      name: "TTCN3-BSC-EMULATOR",
      role: :server,
      remote_ip: {10, 5, 198, 200},
      remote_port: 23907,          # Port source attendu (peut
différer pour les ports éphémères)
      routing_context: 100,        # RC différent - utilisé pour
identifier ce pair après ASPAC
      point_code: 193,
      network_indicator: :national
    }
  ]
```

> **Remarque sur les ports sources éphémères** : La correspondance des ports est exacte. Si un pair est configuré avec `remote_port: 23907` mais se connecte à partir du port 23906, ce pair est exclu de la correspondance initiale et le pair joker l'emporte à la place. Cela est attendu. Le contexte de routage ASPAC est le signal d'identité autoritaire. Le `remote_port` configuré sur un pair n'est utilisé que pour l'heuristique d'assignation de connexion initiale, pas pour l'identité finale. Donnez à chaque pair un `routing_context` distinct et le remappage résoudra correctement quel que soit le port source utilisé par le distant.

---

# Dépannage des Connexions M3UA

**Problème : La connexion ne s'établit pas**

**Symptômes :**

- Le statut indique DOWN
- Pas d'association SCTP

**Vérifications :**

1. Vérifiez la connectivité réseau : `ping remote_ip`
2. Vérifiez que le pare-feu autorise SCTP (protocole 132)
3. Vérifiez que le STP/SGP distant écoute sur le port correct
4. Vérifiez `remote_ip` et `remote_port` dans la config
5. Consultez les journaux de l'application pour les erreurs SCTP

**Problème : Connexion établie mais ASP non actif**

**Symptômes :**

- L'association SCTP existe
- L'état de l'ASP est bloqué en INACTIVE ou ASPUP_SENT

**Vérifications :**

1. Vérifiez que le contexte de routage correspond à la configuration distante
2. Vérifiez que le STP distant accepte votre code de point
3. Consultez les journaux pour les rejets ASPUP/ASPAC
4. Vérifiez qu'il n'y a pas d'exigences d'authentification/sécurité

**Problème : Données ne circulent pas**

**Symptômes :**

- L'état de l'ASP indique ACTIVE
- Aucun message n'est routé

**Vérifications :**

1. Vérifiez le contexte de routage dans les messages
2. Vérifiez l'adressage SCCP (format GT, valeurs SSN)
3. Vérifiez que les tables de routage sont configurées correctement
4. Consultez la page `/events` pour les erreurs SCCP
5. Vérifiez le routage du code de point au niveau STP

---

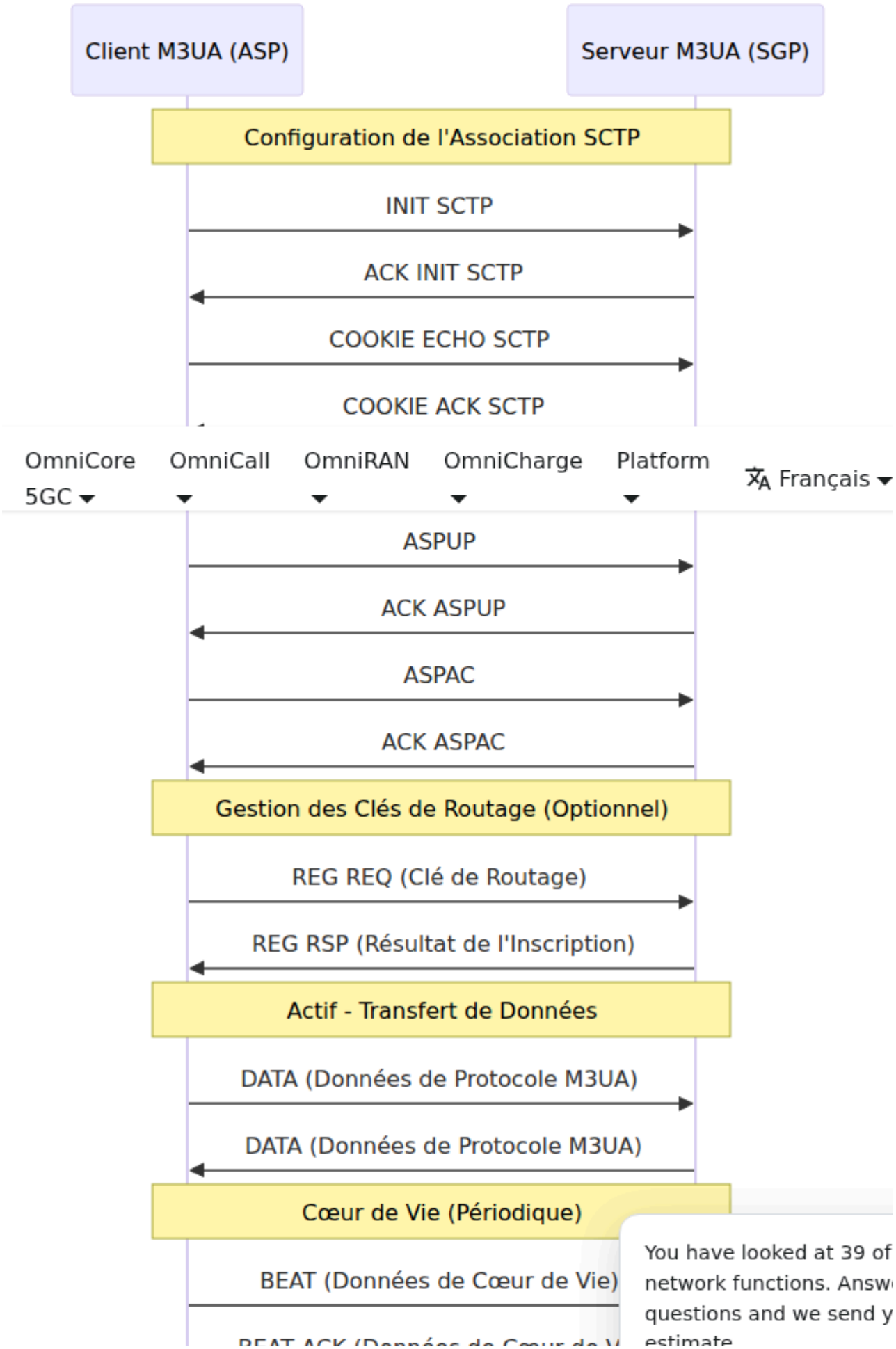
# Surveillance de l'État des Pairs M2PA

## Comprendre M2PA

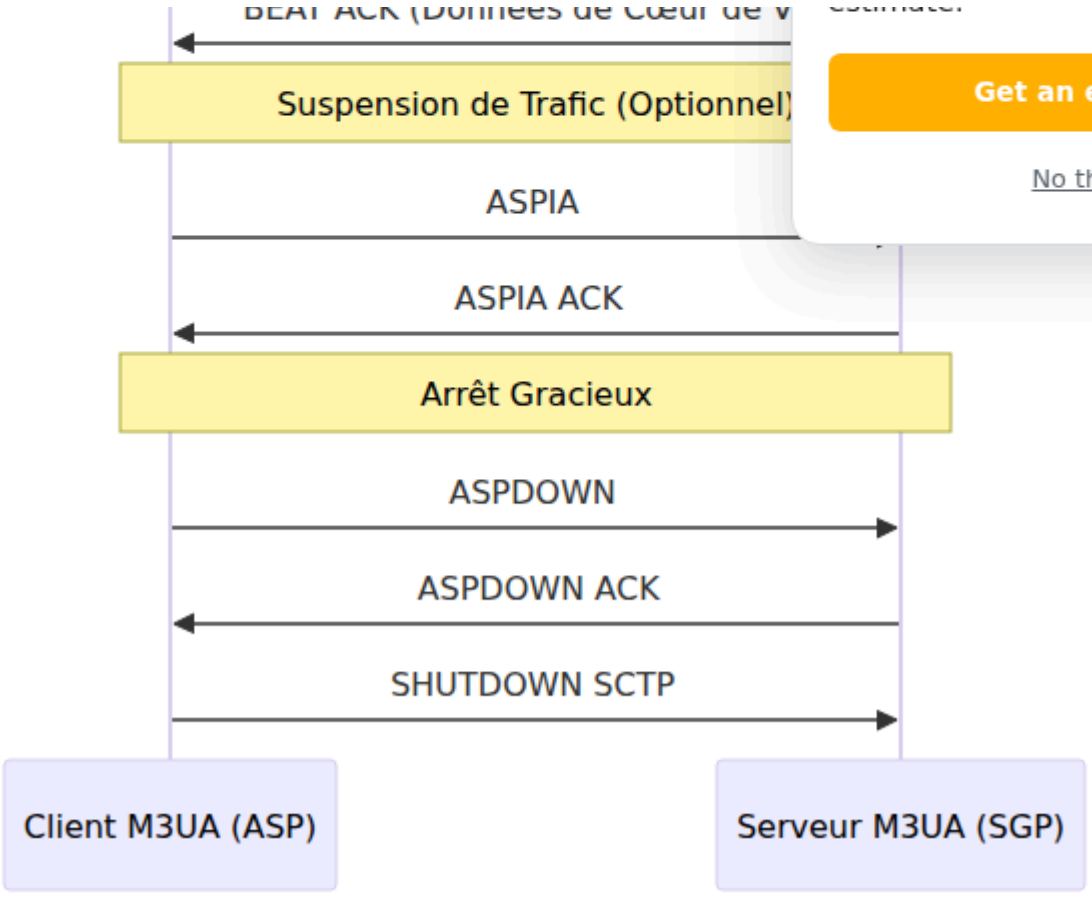
M2PA (MTP2 User Peer-to-Peer Adaptation Layer) est un protocole défini dans la RFC 4165 qui fournit un transport de messages MTP3 point à point sur SCTP. Contrairement à M3UA qui utilise une architecture ASP/SGP, M2PA fournit des liens pair à pair similaires aux liens SS7 TDM traditionnels.

## États des Liens M2PA

Les liens M2PA progressent à travers plusieurs états lors de l'établissement :




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




**Descriptions des États :**

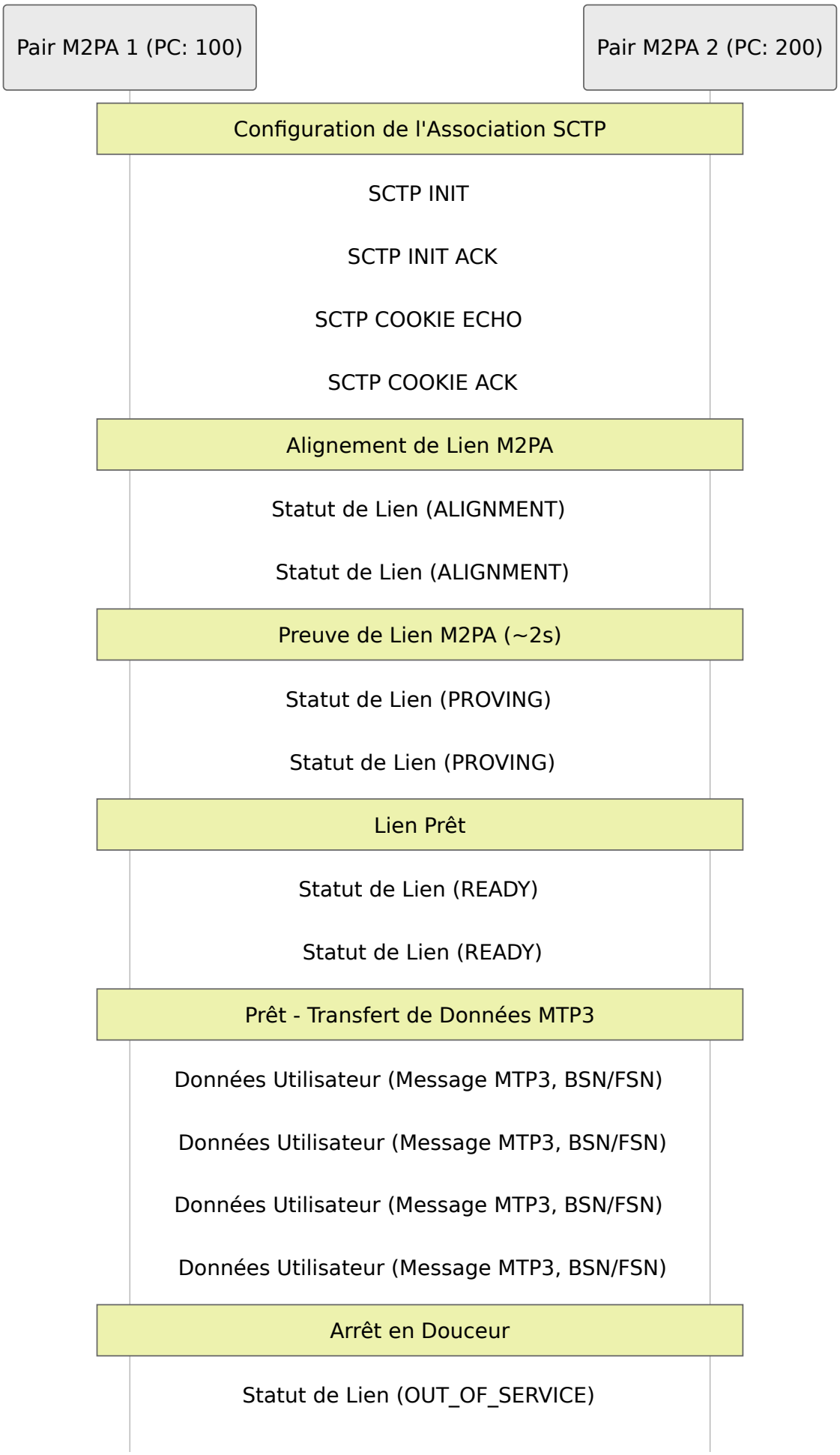
- **DOWN** - Pas de connexion SCTP, lien inactif
- **CONNECTING** - Association SCTP en cours
- **ALIGNMENT** - Messages de Statut de Lien échangés (~1 seconde)
- **PROVING** - Période de preuve de lien, test de l'intégrité du lien (~2 secondes)
- **READY** - Lien opérationnel, prêt pour le transfert de données utilisateur MTP3
- **ALIGNMENT (réentrée)** - Changement de statut de lien nécessite une nouvelle alignement

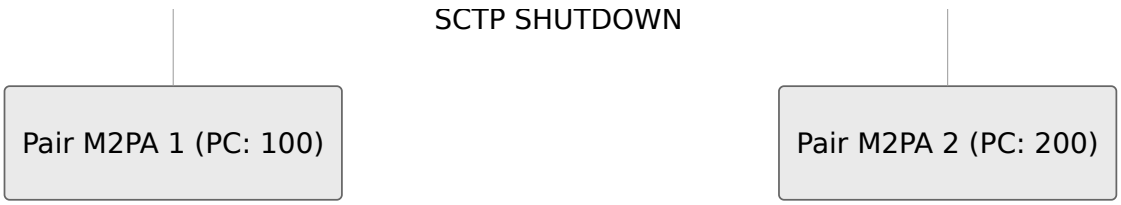
**Progression des États de Lien :**

1. **Connexion SCTP** : Établit l'association SCTP (DOWN → CONNECTING)
2. **Alignement** : Échange de messages de Statut de Lien pour synchroniser (CONNECTING → ALIGNMENT)
3. **Preuve** : Teste la fiabilité du lien et la synchronisation des numéros de séquence (ALIGNMENT → PROVING)

4. **Prêt** : Le lien devient opérationnel pour le transfert de données (PROVING → READY)

# Flux de Messages M2PA


Pair M2PA 1 (PC: 100)
Pair M2PA 2 (PC: 200)
Configuration de l'Association SCTP
SCTP INIT
SCTP INIT ACK
SCTP COOKIE ECHO
SCTP COOKIE ACK
Alignement de Lien M2PA
Statut de Lien (ALIGNMENT)
Statut de Lien (ALIGNMENT)
Preuve de Lien M2PA (~2s)
Statut de Lien (PROVING)
Statut de Lien (PROVING)
Lien Prêt
Statut de Lien (READY)
Statut de Lien (READY)
Prêt - Transfert de Données MTP3
Données Utilisateur (Message MTP3, BSN/FSN)
Données Utilisateur (Message MTP3, BSN/FSN)
Données Utilisateur (Message MTP3, BSN/FSN)
Données Utilisateur (Message MTP3, BSN/FSN)
Arrêt en Douceur
Statut de Lien (OUT_OF_SERVICE)




# Surveillance des Pairs M2PA via l'Interface Web

L'interface Web fournit une surveillance en temps réel des connexions des pairs M2PA.

**Accéder à la Page de Gestion du Routage :**

1. Naviguez vers la page d'accueil de l'interface Web
2. Cliquez sur "Gestion du Routage" dans le menu de navigation
3. Consultez le tableau "Pairs M3UA/M2PA"

**Tableau des Pairs M2PA :**

| Colonne | Description |
|---|---|
| **ID du Pair** | Identifiant unique du pair |
| **Nom** | Nom du pair (par exemple, `M2PA_Link_STP_A`) |
| **Protocole** | Affiche "M2PA" en vert |
| **Code de Point** | Code de point local |
| **PC Adjacent** | Code de point du pair adjacent |
| **Local** | IP:Port local (typiquement port 3565) |
| **Distant** | IP:Port distant |
| **Statut** | État du lien (par exemple, READY, ALIGNMENT, DOWN) |

**Indicateurs de Statut :**

- **READY (Vert)** - Le lien est opérationnel et passe le trafic
- **ALIGNMENT (Jaune)** - Le lien est en alignement, pas encore prêt
- **PROVING (Jaune)** - Le lien est en état de preuve
- **DOWN (Rouge)** - Le lien est hors service ou indisponible

# Dépannage des Connexions M2PA

**Problème : Lien bloqué en ALIGNMENT**

**Symptômes :**

- L'état du lien indique ALIGNMENT pendant une période prolongée
- Pas de progression vers PROVING ou READY

**Vérifications :**

1. Vérifiez que les deux côtés sont configurés avec les bons codes de point
2. Vérifiez que le pare-feu SCTP autorise le protocole 132
3. Vérifiez que `point_code` et `adjacent_point_code` sont correctement définis
4. Consultez les journaux de l'application pour les erreurs de message de Statut de Lien
5. Assurez-vous que le pair distant est également en état d'ALIGNMENT

**Problème : Lien bloqué en PROVING**

**Symptômes :**

- Le lien atteint PROVING mais ne passe pas à READY
- La période de preuve dépasse 2-3 secondes

**Vérifications :**

1. Vérifiez la stabilité du réseau (pas de perte de paquets)
2. Vérifiez les erreurs d'association SCTP
3. Consultez les journaux pour les incohérences de numéros de séquence
4. Assurez-vous que le pair distant est également en état de PROVING
5. Vérifiez que le multihoming SCTP ne cause pas de problèmes de routage

**Problème : Flapping de Lien (DOWN ↔ READY)**

**Symptômes :**

- Le lien passe de manière répétée entre READY et DOWN
- Ré-alignements fréquents

**Vérifications :**

1. Vérifiez la stabilité de la connectivité réseau
2. Vérifiez les paramètres de cœur de SCTP
3. Consultez les paramètres de délai d'expiration de session du pare-feu
4. Vérifiez les problèmes de MTU/fragments
5. Vérifiez qu'il n'y a pas d'adresses IP en double

**Problème : Données ne circulent pas**

**Symptômes :**

- L'état du lien indique READY
- Aucun message MTP3 n'est transféré

**Vérifications :**

1. Vérifiez que les tables de routage incluent des routes vers ce pair
2. Vérifiez que le routage du code de point MTP3 est configuré
3. Consultez les valeurs DPC dans les messages correspondent aux routes attendues
4. Consultez la page `/events` pour les erreurs de routage
5. Vérifiez que les numéros de séquence (BSN/FSN) s'incrémentent

---

# Documentation Connexe


- ← Retour à la Documentation Principale
- Guide des Fonctionnalités Communes - Interface Web, API, Surveillance
- Guide du Client MAP - Envoi de demandes MAP

- Guide du Centre SMS - Livraison SMS
- Référence Technique - Spécifications des protocoles

---

**OmniSS7** par Omnitouch Network Services

# Guide de la passerelle USSD

← Retour à la documentation principale

Ce guide couvre la **passerelle USSD** OmniSS7, qui relie les dialogues USSD SS7/MAP aux rappels HTTP/JSON, permettant aux développeurs tiers de créer des applications USSD avec un simple point de terminaison HTTP.

## Table des matières

1. Aperçu
2. Architecture
3. Activation de la passerelle USSD
4. Configuration
5. Protocole de rappel HTTP
6. USSD d'origine réseau (API Push)
7. Cycle de vie de la session
8. Gestion des erreurs
9. Métriques et surveillance
10. Serveur de rappel d'exemple
11. Dépannage

---

## Aperçu

La passerelle USSD gère deux directions de trafic USSD :

- **D'origine mobile (entrant)** : Un abonné compose un code court (par exemple `*100#`). La passerelle reçoit la MAP `processUnstructuredSS-Request` (opcode 59). Elle transmet la demande à votre rappel HTTP. Elle renvoie ensuite votre réponse par SS7.

- **D'origine réseau (sortant)** : Votre application envoie un message USSD à un abonné via l'API REST. La passerelle envoie une MAP `unstructuredSS-Request` (opcode 60) par SS7. Elle achemine ensuite la réponse de l'abonné vers votre rappel.

Les deux directions prennent en charge des **dialogues multi-tours** : des menus interactifs où l'abonné répond et reçoit des invites de suivi.

## Caractéristiques clés

| Propriété | Valeur |
|---|---|
| **Transport** | HTTP POST synchrone par tour |
| **Encodage** | Alphabet par défaut GSM 7 bits (DCS 0x0F) selon 3GPP TS 23.038 |
| **Longueur maximale du texte** | 182 caractères (configurable) |
| **Suivi de session** | UUID généré par la passerelle par dialogue |
| **Authentification** | Aucune (fait confiance au réseau SS7) |
| **Routage** | Correspondance de préfixe de code court aux URL de rappel |

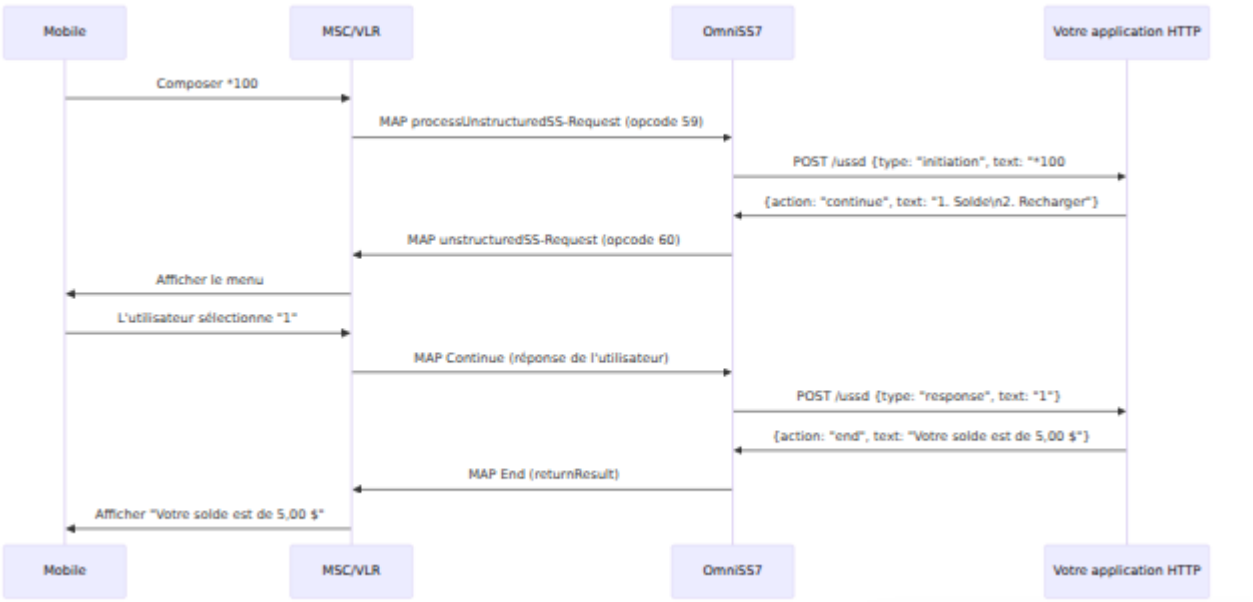
## Références 3GPP

| Spécification | Pertinence |
|---|---|
| 3GPP TS 23.090 | USSD Étape 2 : architecture et procédures |
| 3GPP TS 24.090 | USSD Étape 3 : détails du protocole |
| 3GPP TS 29.002 | Protocole MAP : USSD-Arg, USSD-Res, opcodes 59/60/61 |
| 3GPP TS 23.038 | Alphabet par défaut GSM 7 bits et schéma de codage des données |

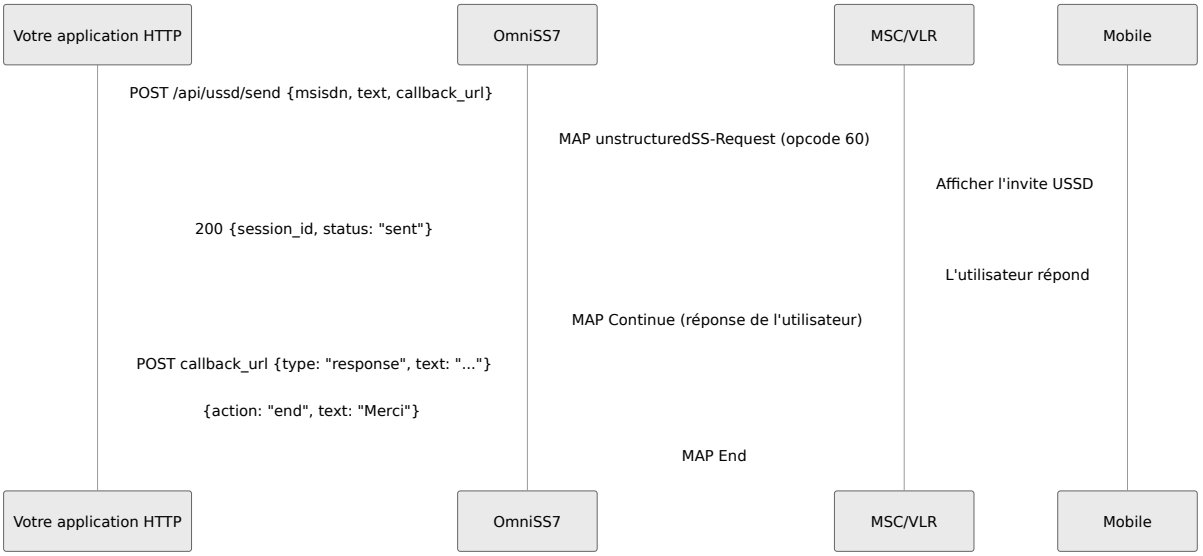
---

# Architecture

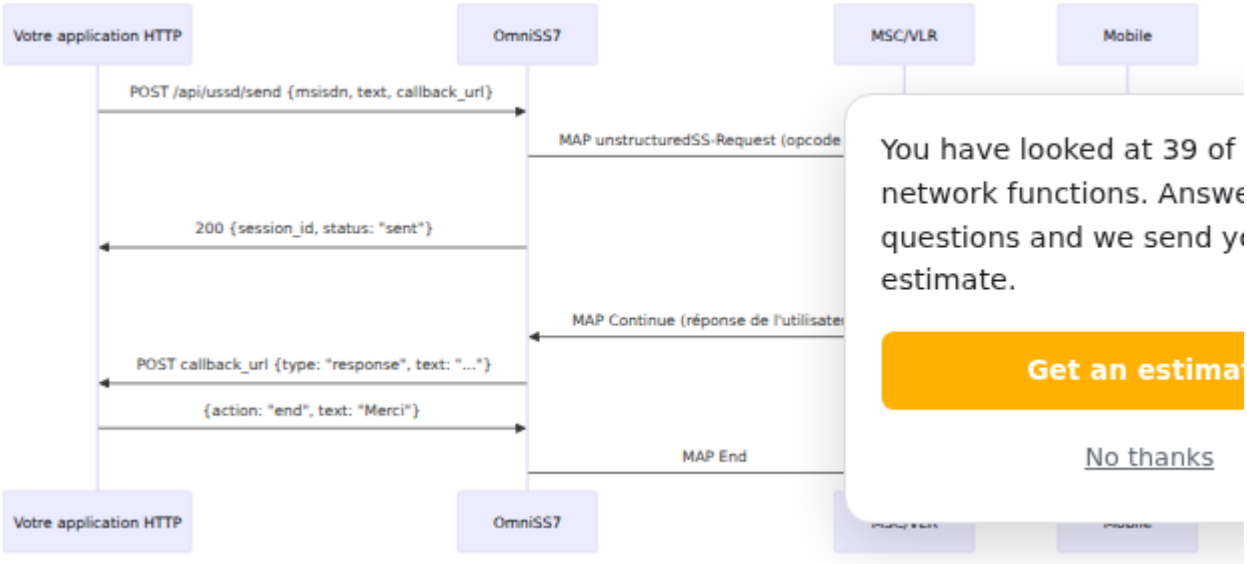
## Flux d'origine mobile (entrant)



## Flux d'origine réseau (Push sortant)




## Vue d'ensemble des composants




---

# Activation de la passerelle USSD

La passerelle USSD nécessite que le **mode client MAP** soit activé, ainsi que son propre drapeau de fonctionnalité.

```
config :omniss7,
  map_client_enabled: true,
  ussd_gateway_enabled: true
```

La passerelle nécessite également une connexion M3UA fonctionnelle (voir Guide du client MAP pour la configuration M3UA).

---

# Configuration

## Paramètres de la passerelle USSD

```
config :omniss7,
  ussd_gateway_enabled: true,
  ussd_gateway: %{
    # Routage de code court - correspondance de préfixe la plus
longue
    routes: [
      %{pattern: "*100", url: "http://balance-app:9000/ussd"},
      %{pattern: "*200", url: "http://topup-app:9000/ussd"},
      %{pattern: "*",    url: "http://default-app:9000/ussd"}
    ],

    # Délais d'expiration de session
    session_timeout_ms: 180_000,    # Durée totale de la session
(3 minutes)
    turn_timeout_ms: 30_000,        # Temps d'attente maximal pour
la réponse de l'abonné par tour (30 secondes)

    # Paramètres de rappel HTTP
    http_timeout_ms: 5_000,         # Délai d'expiration pour le
POST HTTP vers votre application (5 secondes)

    # Limites de texte
    max_text_length: 182            # Max GSM 7 bits (tronque avec
avertissement si dépassé)
  }
```

## Référence des paramètres

| Paramètre | Type | Requis | Par défaut | Des |
|---|---|---|---|---|
| `ussd_gateway_enabled` | Booléen | Oui | `false` | Interr princi la fonct de la USSD |
| `ussd_gateway.routes` | Liste de Maps | Oui | `[]` | Règle routag préfix court. entrée `patte` (chaîn préfix (URL rappe corres de pr plus l l'emp |
| `ussd_gateway.session_timeout_ms` | Entier | Non | `180_000` | Durée maxi sessio en millis La ses termi une e elle e dépas |

| Paramètre | Type | Requis | Par défaut | Des |
|---|---|---|---|---|
| `ussd_gateway.turn_timeout_ms` | Entier | Non | `30_000` | Temps maxin d'atte la rép l'abon un dia multi- millise |
| `ussd_gateway.http_timeout_ms` | Entier | Non | `5_000` | Délai d'expi la req pour l rappe votre applic millise Couvr le tem conne temps répon |
| `ussd_gateway.max_text_length` | Entier | Non | `182` | Nomb maxin caract une cl texte textes dépas sont t et un averti est en |

## Paramètres de routage

Chaque entrée dans la liste `routes` est une carte :

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `pattern` | Chaîne | Oui | Préfixe de code court à faire correspondre. Utilisez `"*"` comme solution de secours. Les préfixes plus longs ont la priorité. |
| `url` | Chaîne | Oui | URL de point de terminaison HTTP pour recevoir les POST de rappel pour les codes courts correspondants. |

## Correspondance de routage

Les routes sont correspondantes par **préfixe le plus long en premier**. Pour la chaîne de composition `*100#` :

1. `"*100"` correspond (longueur 4) : sélectionné
2. `"*10"` correspond (longueur 3) : ignoré, plus court
3. `"*"` correspond (longueur 1) : solution de secours

Si aucune route ne correspond, la passerelle renvoie une erreur MAP au mobile et enregistre un avertissement.

---

# Protocole de rappel HTTP

Votre application reçoit des requêtes HTTP POST de la passerelle et répond avec des instructions JSON.

## Requête de la passerelle à votre application

**Content-Type**: `application/json`

**Premier tour (initiation de session) :**

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "msisdn": "+254712345678",
  "type": "initiation",
  "text": "*100#",
  "turn": 1
}
```

**Tours suivants (l'abonné a répondu) :**

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "msisdn": "+254712345678",
  "type": "response",
  "text": "1",
  "turn": 2
}
```

## Champs de requête

| Champ | Type | Description |
|---|---|---|
| `session_id` | Chaîne | UUID généré par la passerelle. Unique par dialogue USSD. Utilisez ceci pour corréler les tours. |
| `msisdn` | Chaîne | MSISDN de l'abonné (si disponible à partir du message MAP). Peut être vide pour certains réseaux. |
| `type` | Chaîne | `"initiation"` pour le premier tour, `"response"` pour les réponses suivantes de l'abonné. |
| `text` | Chaîne | La chaîne de composition (par exemple `*100#`) lors de l'initiation, ou l'entrée de l'abonné (par exemple `1`) lors de la réponse. |
| `turn` | Entier | Compteur de tour commençant à 1. S'incrémente à chaque interaction de l'abonné. |

## Réponse de votre application

Votre application doit répondre avec JSON contenant une `action` et un `text` :

**Continuer (afficher le menu, attendre l'entrée de l'abonné) :**

```
{
  "action": "continue",
  "text": "1. Solde\n2. Recharger\n3. Transférer"
}
```

**Fin (afficher le message final, fermer la session) :**

```
{
  "action": "end",
  "text": "Votre solde est de 5,00 $"
}
```

### Champs de réponse

| Champ | Type | Requis | Description |
|---|---|---|---|
| `action` | Chaîne | Oui | `"continue"` pour garder la session ouverte et attendre l'entrée de l'abonné, ou `"end"` pour afficher un message final et fermer la session. |
| `text` | Chaîne | Oui | Texte à afficher sur le téléphone de l'abonné. La longueur maximale est régie par `max_text_length` (par défaut 182). Utilisez `\n` pour les sauts de ligne. |

---

## USSD d'origine réseau (API Push)

Envoyez un message USSD à un abonné depuis votre application.

### Point de terminaison

`POST /api/ussd/send`

## Requête

```
{
  "msisdn": "+254712345678",
  "text": "Vous avez une facture en attente. Répondez 1 pour
payer.",
  "callback_url": "http://billing-app:9000/ussd"
}
```

## Champs de requête

| Champ | Type | Requis | Description |
|---|---|---|---|
| `msisdn` | Chaîne | Oui | MSISDN de l'abonné destinataire au format international. |
| `text` | Chaîne | Oui | Texte USSD initial à afficher. Encodé en GSM 7 bits. |
| `callback_url` | Chaîne | Oui | URL pour recevoir la réponse de l'abonné via le protocole de rappel standard. |

## Réponse

**Succès (200 OK) :**

```
{
  "session_id": "xyz-789-abc-123",
  "status": "sent"
}
```

**Réponses d'erreur :**

| Statut HTTP | Corps | Cause |
|---|---|---|
| 400 | `{"error": "requête invalide", "required": ["msisdn", "text", "callback_url"]}` | Un ou plusieurs champs requis manquants, ou le corps n'est pas un JSON valide |
| 500 | `{"error": "{:gsm7_encode_failed, ...}"}` | Le texte contient des caractères non présents dans l'alphabet GSM 7 bits. L'échec d'encodage est signalé par la branche d'erreur générique, donc le statut est **500**, pas 400. |
| 500 | `{"error": "..."}` | Toute autre erreur d'envoi (par exemple, connectivité M3UA). Le champ `error` contient la raison interne inspectée. |
| 503 | `{"error": "passerelle USSD non activée"}` | `ussd_gateway_enabled` est `false` |

> **Remarque sur les erreurs de validation vs d'envoi :** Les problèmes de validation des champs (clés manquantes, JSON mal formé) renvoient **400** **`requête invalide`**. Une fois la validation réussie, tout échec du chemin d'envoi (y compris les échecs d'encodage GSM 7 bits) est signalé comme **500** avec la raison inspectée dans le champ `error`.

## Exemple cURL

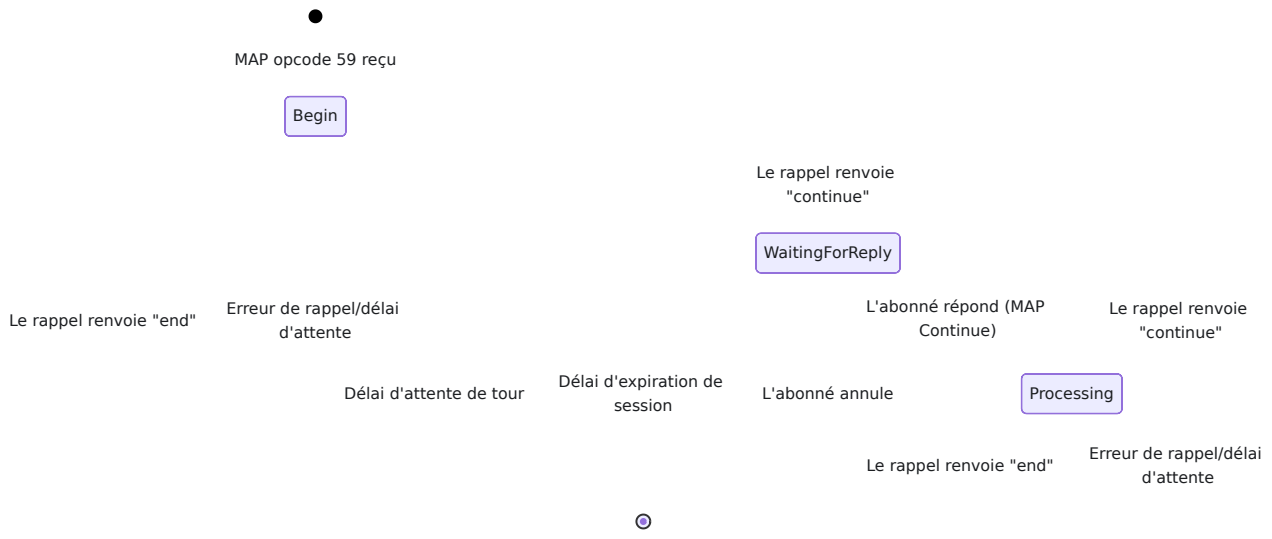
```
curl -X POST http://localhost:9090/api/ussd/send \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "+254712345678",
    "text": "Vous avez une facture en attente. Répondez 1 pour
payer.",
    "callback_url": "http://billing-app:9000/ussd"
  }'
```

---

# Cycle de vie de la session

Chaque dialogue USSD est suivi comme une **session** avec un `session_id` unique.

## États de session




## Session à un tour

Si votre rappel renvoie `"end"` au premier tour, aucune session persistante n'est créée. La passerelle envoie une MAP End avec le résultat et renvoie immédiatement.

## Session multi-tours

Si votre rappel renvoie `"continue"`, la passerelle :

1. Crée un **Session GenServer** enregistré dans `UssdGateway.Registry`
2. Envoie une MAP Continue avec l'opcode 60 (unstructuredSS-Request) au mobile
3. Attend la réponse de l'abonné (jusqu'à `turn_timeout_ms`)
4. Transmet la réponse à votre rappel
5. Répète jusqu'à ce que votre rappel renvoie `"end"` ou qu'un délai d'attente se produise

## Comportement de délai d'attente

| Délai d'attente | Par défaut | Effet |
|---|---|---|
| **Délai d'attente de tour** | 30 secondes | Si l'abonné ne répond pas dans cette fenêtre, la session est terminée avec une erreur MAP. |
| **Délai d'expiration de session** | 3 minutes | Durée totale de la session. Termine la session indépendamment de l'activité. |
| **Délai d'expiration de rappel HTTP** | 5 secondes | Si votre application ne répond pas à temps, la passerelle envoie une erreur MAP au mobile et termine la session. |

---

# Gestion des erreurs

La passerelle gère les échecs avec grâce et tente toujours d'envoyer une réponse d'erreur MAP au mobile afin que l'abonné voit un message significatif plutôt qu'un délai d'attente réseau.

| Scénario | Action de la passerelle | Code d'erreur MAP |
|---|---|---|
| Délai d'attente de rappel HTTP ou 5xx | Terminer la session, envoyer MAP End avec erreur | 34 (systemFailure) |
| JSON invalide du rappel | Terminer la session, envoyer MAP End avec erreur | 34 (systemFailure) |
| Le texte USSD dépasse `max_text_length` | Tronquer le texte, enregistrer un avertissement, continuer normalement | N/A (tronqué, pas une erreur) |
| Délai d'attente de l'abonné (pas de réponse) | Terminer la session, envoyer MAP End avec erreur | 34 (systemFailure) |
| Aucune route ne correspond au code court | Envoyer MAP End avec erreur, enregistrer un avertissement | 34 (systemFailure) |
| Plantage du Session GenServer | La session meurt, l'abonné voit un délai d'attente réseau | N/A (sortie de processus) |
| Passerelle USSD non activée | Retourner Facility Not Supported | 21 (facilityNotSupported) |

---

# Métriques et surveillance

La passerelle USSD expose des métriques Prometheus sur le point de terminaison standard `/metrics` (port 9090).

# Métriques USSD

**Métrique**: `ussd_requests_total` **Type**: Compteur **Description**: Nombre total de requêtes USSD traitées **Étiquettes**:

- `direction`: `"inbound"` (d'origine mobile) ou `"outbound"` (push d'origine réseau)

**Métrique**: `ussd_active_sessions` **Type**: Jauge **Description**: Déclarée comme une jauge pour le nombre de sessions USSD actives.

> **Avertissement**: Cette jauge est **déclarée mais jamais mise à jour** dans la version actuelle : elle lit toujours `0`. Ne comptez pas dessus pour les comptes de session en direct. Suivez l'activité des sessions via `ussd_requests_total` à la place.

**Métrique**: `map_request_duration_milliseconds` **Type**: Histogramme **Description**: Durée des opérations d'envoi USSD en millisecondes **Étiquettes**:

- `operation`: `"ussd_send"` pour les requêtes push sortantes

# Exemples de requêtes Prometheus

```
# Taux de requêtes USSD par direction (entrant vs sortant)
rate(ussd_requests_total[5m])

# Latence USSD sortante (p95)
histogram_quantile(0.95,
rate(map_request_duration_milliseconds_bucket{operation="ussd_send"}
[5m]))
```

---

# Serveur de rappel d'exemple

## Python (Flask)

```
from flask import Flask, request, jsonify

app = Flask(__name__)
sessions = {}

@app.route('/ussd', methods=['POST'])
def ussd():
    data = request.json
    session_id = data['session_id']
    text = data['text']
    turn = data['turn']

    if data['type'] == 'initiation':
        sessions[session_id] = {'state': 'main_menu'}
        return jsonify({
            'action': 'continue',
            'text': 'Bienvenue!\n1. Vérifier le solde\n2. Acheter
du crédit\n3. Transférer'
        })

    state = sessions.get(session_id, {}).get('state')

    if state == 'main_menu':
        if text == '1':
            del sessions[session_id]
            return jsonify({
                'action': 'end',
                'text': 'Votre solde est de 5,00 $'
            })
        elif text == '2':
            sessions[session_id]['state'] = 'buy_airtime'
            return jsonify({
                'action': 'continue',
                'text': 'Entrez le montant :'
            })
        else:
            del sessions[session_id]
            return jsonify({
```

```
                'action': 'end',
                'text': 'Option invalide. Au revoir.'
            })

    elif state == 'buy_airtime':
        del sessions[session_id]
        return jsonify({
            'action': 'end',
            'text': f'Vous avez acheté {text} $ de crédit. Merci!'
        })

    return jsonify({'action': 'end', 'text': 'Session expirée.'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=9000)
```

## Node.js (Express)

```
const express = require('express');
const app = express();
app.use(express.json());

const sessions = new Map();

app.post('/ussd', (req, res) => {
  const { session_id, text, type } = req.body;

  if (type === 'initiation') {
    sessions.set(session_id, { state: 'main_menu' });
    return res.json({
      action: 'continue',
      text: 'Bienvenue!\n1. Vérifier le solde\n2. Acheter du
crédit'
    });
  }

  const session = sessions.get(session_id);
  if (!session) {
    return res.json({ action: 'end', text: 'Session expirée.' });
  }

  if (session.state === 'main_menu' && text === '1') {
    sessions.delete(session_id);
    return res.json({ action: 'end', text: 'Votre solde est de
5,00 $' });
  }

  sessions.delete(session_id);
  return res.json({ action: 'end', text: 'Au revoir.' });
});

app.listen(9000, () => console.log('Rappel USSD sur le port
9000'));
```

---

# Dépannage

## Le numéro USSD renvoie "Service non disponible"

**Symptômes** : L'abonné compose un code court et obtient immédiatement une erreur réseau.

**Causes possibles** :

- `ussd_gateway_enabled` est `false`
- Aucune route ne correspond au code court composé
- La connexion M3UA est hors service

**Résolution** :

1. Vérifiez `ussd_gateway_enabled: true` dans la configuration
2. Assurez-vous qu'un modèle de route correspond au code court (n'oubliez pas d'inclure `*` comme solution de secours)
3. Vérifiez l'état du pair M3UA dans l'interface Web (page Pairs)

## Le rappel ne reçoit pas de requêtes

**Symptômes** : Les journaux de la passerelle montrent un début USSD mais votre application ne reçoit jamais le POST HTTP.

**Causes possibles** :

- L'URL de rappel est inaccessible depuis l'hôte OmniSS7
- Pare-feu bloquant le HTTP sortant depuis OmniSS7
- Application de rappel non en cours d'exécution

**Résolution** :

1. Testez la connectivité : `curl -v http://your-app:9000/ussd` depuis l'hôte OmniSS7
2. Vérifiez les règles de pare-feu pour le HTTP sortant

3. Vérifiez que votre application de rappel écoute sur le port configuré

## Les sessions expirent prématurément

**Symptômes** : Les sessions multi-tours se terminent par "systemFailure" avant que l'abonné puisse répondre.

**Causes possibles** :

- `turn_timeout_ms` est trop court pour votre base d'abonnés
- `http_timeout_ms` est trop court pour le temps de traitement de votre application
- Latence réseau entre OmniSS7 et votre serveur de rappel

**Résolution** :

1. Augmentez `turn_timeout_ms` (30 secondes par défaut devrait suffire pour la plupart des cas)
2. Augmentez `http_timeout_ms` si votre application a besoin de plus de temps de traitement
3. Déployez le serveur de rappel près de l'OmniSS7 pour réduire la latence

## Erreurs d'encodage GSM 7 bits

**Symptômes** : erreurs `gsm7_encode_failed` dans les journaux, ou une réponse **500** de `/api/ussd/send` dont le champ `error` contient `{:gsm7_encode_failed, ...}`.

**Causes possibles** :

- Le texte contient des caractères en dehors de l'alphabet GSM 7 bits par défaut (par exemple, emoji, caractères CJK)

**Résolution** :

- Restreindre le texte USSD à l'ensemble de caractères de base GSM : lettres ASCII, chiffres, ponctuation courante, et quelques caractères grecs/nordiques

- Voir 3GPP TS 23.038 Section 6.2.1 pour le tableau complet des caractères

---

# Documentation connexe

- **Guide API** : Référence complète de l'API REST (tous les points de terminaison y compris `/api/ussd/send`)
- **Guide du client MAP** : Configuration de la connexion M3UA requise pour USSD
- **Référence de configuration** : Tous les paramètres de configuration
- **Guide des fonctionnalités communes** : Interface Web, surveillance et configuration de Prometheus

# Guide de l'interface utilisateur Web

← Retour à la documentation principale

Ce guide fournit une documentation complète pour l'utilisation de l'OmniSS7 **Interface Utilisateur Web** (interface Phoenix LiveView).

## Table des matières

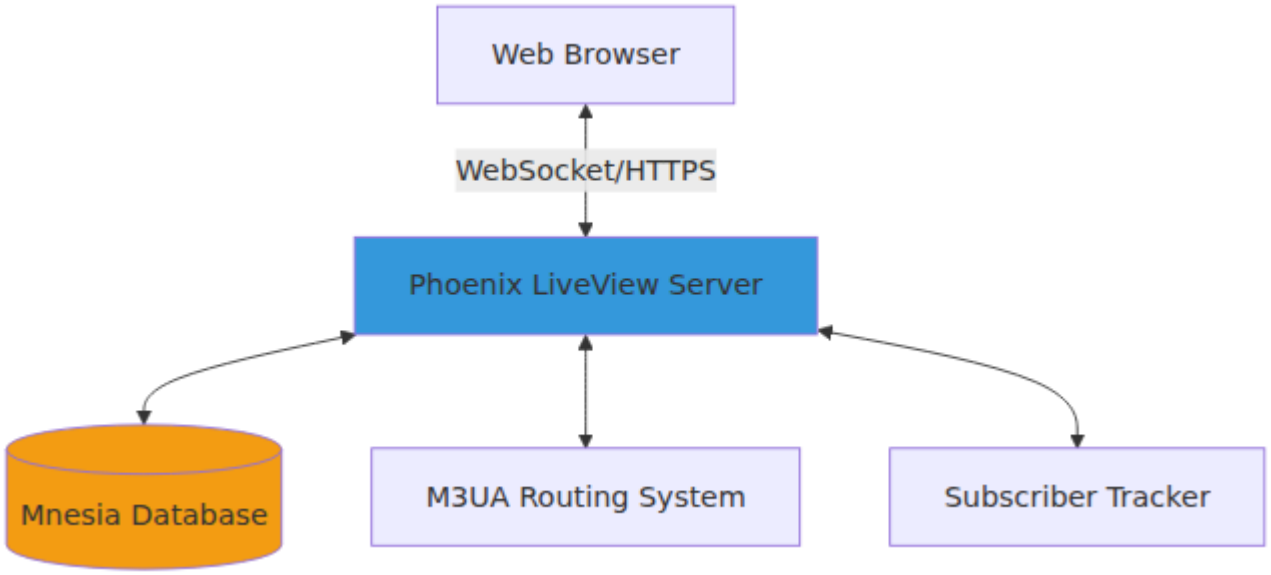
1. Aperçu
2. Accéder à l'interface utilisateur Web
3. Page de gestion du routage
4. Page des abonnés actifs
5. Opérations courantes
6. Comportement d'auto-rafraîchissement

---

## Aperçu

L'interface utilisateur Web OmniSS7 est une application **Phoenix LiveView** qui fournit des capacités de surveillance et de gestion en temps réel. Les pages disponibles dépendent du mode opérationnel actif (STP, HLR ou SMSc).

## Architecture de l'interface utilisateur Web




## Configuration du serveur

| Paramètre | Valeur |
|---|---|
| **Protocole** | HTTPS (`force_ssl` activé) |
| **Port** | `8087` (configuré sous `config :control_panel, ControlPanelWeb.Endpoint` → `https: [port: 8087]`) |
| **IP par défaut** | `0.0.0.0` (écoute sur toutes les interfaces) |
| **Certificat** | `priv/cert/omnitouch.crt` |
| **Clé privée** | `priv/cert/omnitouch.pem` |
| **Adaptateur** | Bandit (Phoenix) |

**URL d'accès** : `https://[server-ip]:8087`

---

# Accéder à l'interface utilisateur Web

## Prérequis

1. **Certificats SSL** : Assurez-vous que des certificats SSL valides sont présents dans `priv/cert/` :
   - `omnitouch.crt` - Fichier de certificat
   - `omnitouch.pem` - Fichier de clé privée
2. **Application en cours d'exécution** : Démarrez l'application avec `iex -S mix`
3. **Pare-feu** : Assurez-vous que le port `8087` est ouvert pour le trafic HTTPS

## Pages disponibles par mode

L'étiquette de la page affichée ici est exactement l'étiquette enregistrée dans `config/config.exs` (`use_builtin_pages` / `use_additional_pages`). Les colonnes de mode indiquent le mode opérationnel (STP/HLR/SMSc) dans lequel une page est généralement pertinente ; la configuration d'exécution filtre la liste des pages par déploiement.

| Page (route) | Étiquette | Mode STP | Mode HLR | Mode SMSc | Descrip |
|---|---|---|---|---|---|
| `/application` | **Ressources** | ✓ | ✓ | ✓ | Ressourc système surveillar |
| `/configuration` | **Configuration** | ✓ | ✓ | ✓ | Visualiseu configura |
| `/license` | **Licence** | ✓ | ✓ | ✓ | Statut de licence |
| `/events` | **Événements SS7** | ✓ | ✓ | ✓ | Journalisa des événeme capture d messages SCCP |
| `/logs` | **Journaux système** | ✓ | ✓ | ✓ | Visualiseu journaux d'applicat en direct |
| `/client` | **Client SS7** | ✓ | ✓ | ✓ | Test de fonctionn manuel M |
| `/m3ua` | **Pairs** | ✓ | ✓ | ✓ | État de la connexion M3UA pai ASP |
| `/sctp` | **Connexions SCTP** | ✓ | ✓ | ✓ | État de l'associat SCTP en |

| Page (route) | Étiquette | Mode STP | Mode HLR | Mode SMSc | Descrip |
|---|---|---|---|---|---|
| `/routing` | **Routage** | □ | □ | □ | Gestion d table de routage M |
| `/routing_test` | **Test de routage** | □ | □ | □ | Test et validation route |
| `/hlr_links` | **Liens HLR** | □ | □ | □ | Statut de HLR et ge des abon |
| `/subscribers` | **Abonnés actifs** | □ | □ | □ | Suivi de localisati abonnés temps ré (HLR) |
| `/smsc_links` | **Liens SMSc** | □ | □ | □ | Statut de SMSc et gestion d file d'atte |
| `/smsc_subscribers` | **Abonnés SMSc** | □ | □ | □ | Suivi des abonnés temps ré (SMSc) |
| `/camel_sessions` | **Sessions CAMEL** | □ | □ | □ | Surveillar dialogue CAMEL/C direct |

| Page (route) | Étiquette | Mode STP | Mode HLR | Mode SMSc | Descrip |
|---|---|---|---|---|---|
| `/camel_request` | **Constructeur de requêtes CAMEL** | □ | □ | □ | Construct de requêt CAP inter |
| `/msisdn_imsi_test` | **Test MSISDN/IMSI** | □ | □ | □ | Recherch test de conversic MSISDN ↔ |

> La page M3UA est étiquetée **"Pairs"** dans la navigation (route `/m3ua`).

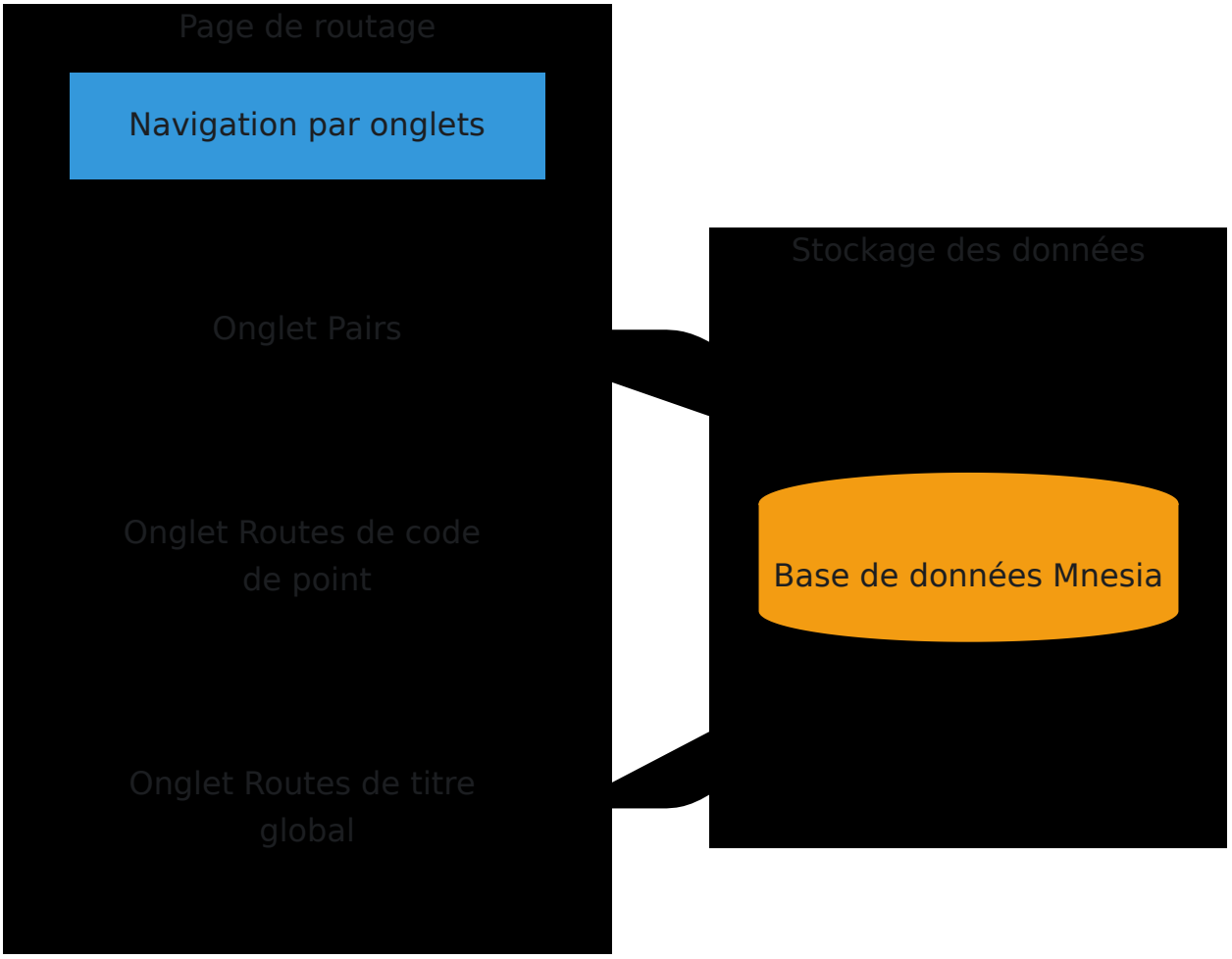
---

# Page de gestion du routage

**Page** : `/routing` **Modes** : STP, SMSc **Auto-rafraîchissement** : Toutes les 5 secondes

La page de gestion du routage fournit une interface à onglets pour gérer les tables de routage M3UA.

## Mise en page de la page




## Onglet Pairs

Gérez les connexions M3UA aux pairs (autres STP, HLR, MSC, SMSC).

**Colonnes du tableau des pairs**

| Colonne | Description | Exemple |
|---|---|---|
| **ID** | Identifiant unique du pair | `1` |
| **Nom** | Nom du pair lisible par l'homme | `"STP_West"` |
| **Rôle** | Rôle de connexion | `client`, `server`, `stp` |
| **Code de point** | Code de point SS7 du pair | `100` |
| **Distant** | IP:Port distant | `10.0.0.10:2905` |
| **Statut** | Statut de la connexion | `active`, `aspup`, `down` |
| **Actions** | Boutons Modifier/Supprimer | - |

**Ajouter un pair**

1. **Cliquez** sur l'onglet Pairs
2. **Remplissez** les champs du formulaire :
   - **ID du pair** : Généré automatiquement s'il est laissé vide
   - **Nom du pair** : Nom descriptif (obligatoire)
   - **Rôle** : Sélectionnez `client`, `server` ou `stp`
   - **Code de point** : Code de point SS7 (obligatoire)
   - **IP locale** : Adresse IP de votre système
   - **Port local** : 0 pour l'attribution dynamique de port
   - **IP distante** : Adresse IP du pair
   - **Port distant** : Port du pair (généralement 2905)
   - **Contexte de routage** : ID de contexte de routage M3UA
   - **Indicateur de réseau** : `international` ou `national`
3. **Cliquez** sur "Ajouter un pair"

**Persistance** : Le pair est immédiatement enregistré dans Mnesia et survit au redémarrage.

**Modifier un pair**

1. **Cliquez** sur le bouton "Modifier" de la ligne du pair
2. **Modifiez** les champs du formulaire selon vos besoins
3. **Cliquez** sur "Mettre à jour le pair"

**Remarque** : Si vous changez l'ID du pair, l'ancien pair est supprimé et un nouveau est créé.

#### Supprimer un pair

1. **Cliquez** sur le bouton "Supprimer" de la ligne du pair
2. **Confirmez** la suppression (toutes les routes utilisant ce pair seront également supprimées)

#### Indicateurs de statut des pairs

| Statut | Couleur | Description |
|---|---|---|
| `active` | □ Vert | Le pair est connecté et routant des messages |
| `aspup` | □ Jaune | L'ASP est opérationnel mais pas encore actif |
| `down` | □ Rouge | Le pair est déconnecté |

---

## Onglet Routes de code de point

Configurez des règles de routage basées sur les codes de point de destination.

#### Colonnes du tableau des routes

| Colonne | Description | Exemple |
|---|---|---|
| **PC de destination** | Code de point cible (format zone.area.id) | `1.2.3` (100) |
| **Masque** | Masque de sous-réseau pour la correspondance des PC | `/14` (exact), `/8` (plage) |
| **ID du pair** | Pair cible pour cette route | `1` |
| **Nom du pair** | Nom du pair cible | `"STP_West"` |
| **Priorité** | Priorité de la route (1 = la plus élevée) | `1` |
| **Réseau** | Indicateur de réseau | `international` |
| **Actions** | Boutons Modifier/Supprimer | - |

**Ajouter une route de code de point**

1. **Cliquez** sur l'onglet "Routes de code de point"
2. **Remplissez** les champs du formulaire :
   - **Code de point de destination** : Entrez au format `zone.area.id` (par exemple, `1.2.3`) ou entier (0-16383)
   - **Masque** : Sélectionnez le masque `/14` pour une correspondance exacte, des valeurs inférieures pour des plages
   - **ID du pair** : Sélectionnez le pair cible dans le menu déroulant
   - **Priorité** : Entrez la priorité (1 = la plus élevée, par défaut)
   - **Indicateur de réseau** : Sélectionnez `international` ou `national`
3. **Cliquez** sur "Ajouter une route"

**Format de code de point** : Vous pouvez entrer des codes de point dans deux formats :

- **Format 3-8-3** : `zone.area.id` (par exemple, `1.2.3`)
- **Format entier** : `0-16383` (par exemple, `1100`)

Le système convertit automatiquement entre les formats.

**Comprendre les masques**

Les codes de point sont des valeurs de 14 bits (0-16383). Le masque spécifie combien de bits les plus significatifs doivent correspondre :

| Masque | PC correspondants | Cas d'utilisation |
|---|---|---|
| `/14` | 1 (correspondance exacte) | Route vers une destination spécifique |
| `/13` | 2 PC | Petite plage |
| `/8` | 64 PC | Plage moyenne |
| `/0` | Tous les 16 384 PC | **Route par défaut/de secours** |

**Exemples** :

- `PC 1000 /14` → Correspond uniquement au PC 1000
- `PC 1000 /8` → Correspond aux PC 1000-1063 (64 PC consécutifs)
- `PC 0 /0` → Correspond à tous les codes de point (route par défaut)

**Carte de référence des masques de code de point**

La page Web comprend une référence interactive montrant toutes les valeurs de masque et leurs plages.

---

# Onglet Routes de titre global

Configurez des règles de routage basées sur les adresses de titre global SCCP.

**Exigence** : Le routage par titre global doit être activé dans la configuration :

```
config :omniss7,
  enable_gt_routing: true
```

### Colonnes du tableau des routes

| Colonne | Description | Exemple |
|---|---|---|
| **Préfixe GT** | Préfixe GT du parti appelé (vide = secours) | `"1234"`, `""` |
| **SSN source** | Correspondance sur le SSN du parti appelé (facultatif) | `6` (HLR), `any` |
| **ID du pair** | Pair cible | `1` |
| **Pair** | Nom du pair | `"HLR_West (1)"` |
| **SSN de destination** | Réécrire le SSN lors du transfert (facultatif) | `6`, `preserve` |
| **Priorité** | Priorité de la route | `1` |
| **Description** | Description de la route | `"Numéros US"` |
| **Actions** | Boutons Modifier/Supprimer | - |

### Ajouter une route de titre global

1. **Cliquez** sur l'onglet "Routes de titre global"
2. **Remplissez** les champs du formulaire :
   - **Préfixe GT** : Laissez vide pour une route de secours, ou entrez des chiffres (par exemple, `"1234"`)
   - **SSN source** : Facultatif - filtre par SSN du parti appelé
   - **ID du pair** : Sélectionnez le pair cible

    - **SSN de destination** : Facultatif - réécrire le SSN lors du transfert
    - **Priorité** : Priorité de la route (1 = la plus élevée)
    - **Description** : Description lisible par l'homme
3. **Cliquez** sur "Ajouter une route"

**Routes de secours** : Si le préfixe GT est vide, la route agit comme un attrape-tout pour les GT qui ne correspondent à aucune autre route.

#### Valeurs SSN courantes

La page comprend une carte de référence avec des valeurs SSN courantes :

| SSN | Élément de réseau |
|---|---|
| 6 | HLR (Registre de localisation de l'abonné) |
| 7 | VLR (Registre de localisation des visiteurs) |
| 8 | MSC (Centre de commutation mobile) |
| 9 | EIR (Registre d'identité d'équipement) |
| 10 | AUC (Centre d'authentification) |
| 142 | RANAP |
| 145 | gsmSCF (Fonction de contrôle de service) |
| 146 | SGSN |

#### Réécriture de SSN

- **SSN source** : Correspondance sur le SSN du parti appelé dans les messages entrants
- **SSN de destination** : Si défini, réécrit le SSN du parti appelé lors du transfert
    - Vide = préserver le SSN original

- Valeur = remplacer par ce SSN

**Cas d'utilisation** : Routage des messages avec SSN=6 (HLR) vers un pair, et réécriture à SSN=7 (VLR) du côté sortant.

---

## Persistance de la table de routage

**Toutes les routes sont stockées dans Mnesia et survivent aux redémarrages de l'application.**

### Comment les routes persistent

1. **Modifications de l'interface utilisateur Web** : Toutes les opérations d'ajout/modification/suppression sont immédiatement enregistrées dans Mnesia
2. **Redémarrage de l'application** : Les routes sont chargées depuis Mnesia au démarrage
3. **Fusion de Runtime.exs** : Les routes statiques de `config/runtime.exs` sont fusionnées avec les routes Mnesia (pas de doublons)

### Priorité de la route

Lorsque plusieurs routes correspondent à une destination :

1. **Plus spécifique d'abord** : Les valeurs de masque plus élevées (plus spécifiques) ont la priorité
2. **Champ de priorité** : Les numéros de priorité plus bas routent en premier (1 = priorité la plus élevée)
3. **Statut du pair** : Seules les routes vers des pairs `actifs` sont utilisées

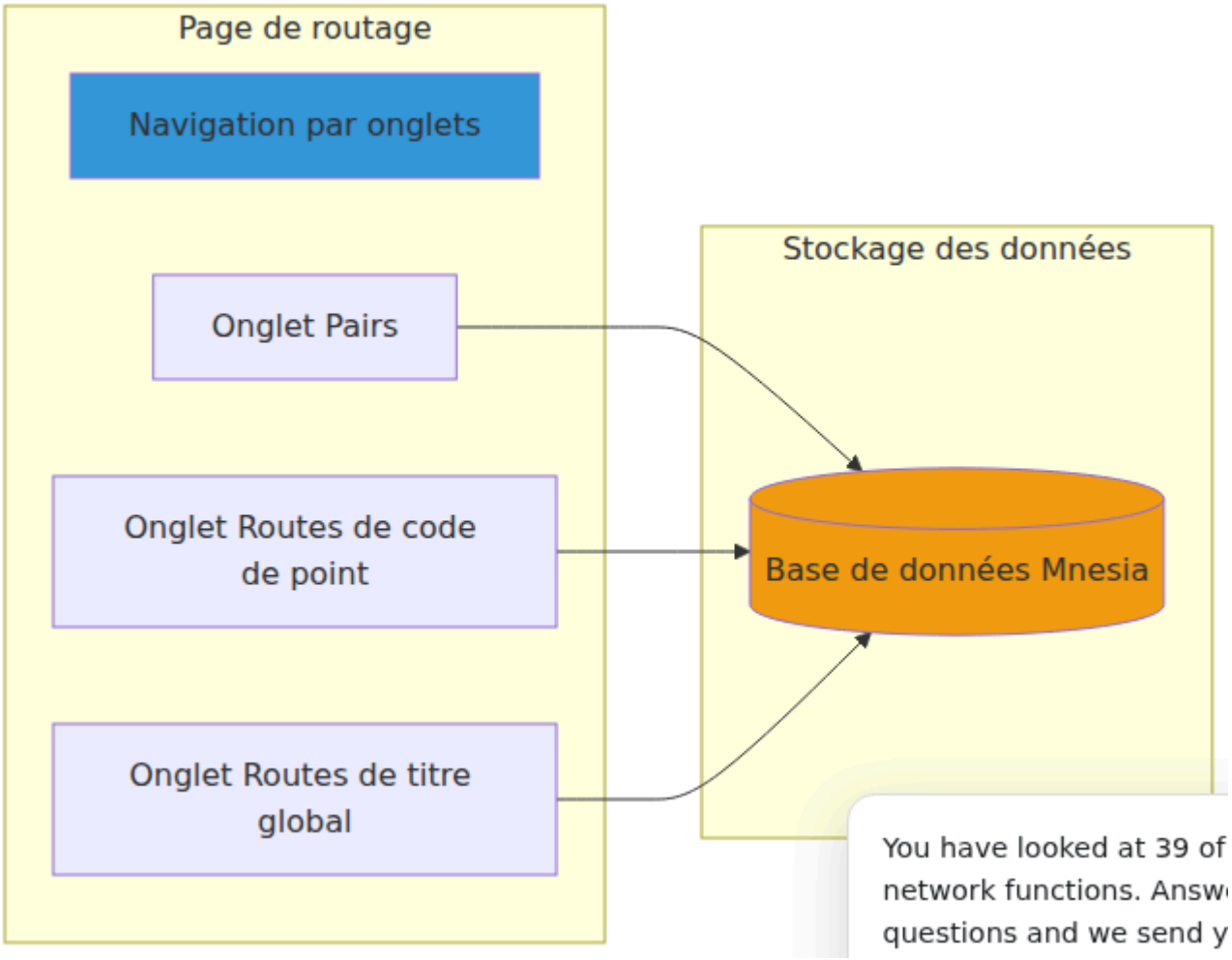
---

# Page des abonnés actifs

**Page** : `/subscribers` **Mode** : HLR uniquement **Auto-rafraîchissement** : Toutes les 2 secondes

Affiche le suivi en temps réel des abonnés ayant envoyé des demandes UpdateLocation.

## Fonctionnalités de la page



## Colonnes du tableau des abonnés

| Colonne | Description | Exemple |
|---|---|---|
| **IMSI** | IMSI de l'abonné | `"50557123456789"` |
| **Numéro VLR** | Adresse GT VLR actuelle | `"555123155"` |
| **Numéro MSC** | Adresse GT MSC actuelle | `"555123155"` |
| **Mis à jour à** | Horodatage de la dernière mise à jour de localisation | `"2025-10-25 14:23:45 UTC"` |
| **Durée** | Temps écoulé depuis l'enregistrement | `"2h 15m 34s"` |

## Résumé des statistiques

Lorsque des abonnés sont présents, une carte de résumé affiche :

- **Total Actif** : Nombre total d'abonnés enregistrés
- **VLR Uniques** : Nombre d'adresses VLR distinctes
- **MSC Uniques** : Nombre d'adresses MSC distinctes

## Effacer les abonnés

**Bouton Effacer tout** : Supprime tous les abonnés actifs du tracker.

**Confirmation** : Nécessite une confirmation avant de vider (ne peut pas être annulé).

**Cas d'utilisation** : Effacer les enregistrements d'abonnés obsolètes après une maintenance ou un test du réseau.

## Auto-rafraîchissement

La page se rafraîchit automatiquement toutes les **2 secondes** pour afficher les mises à jour des abonnés en temps réel.

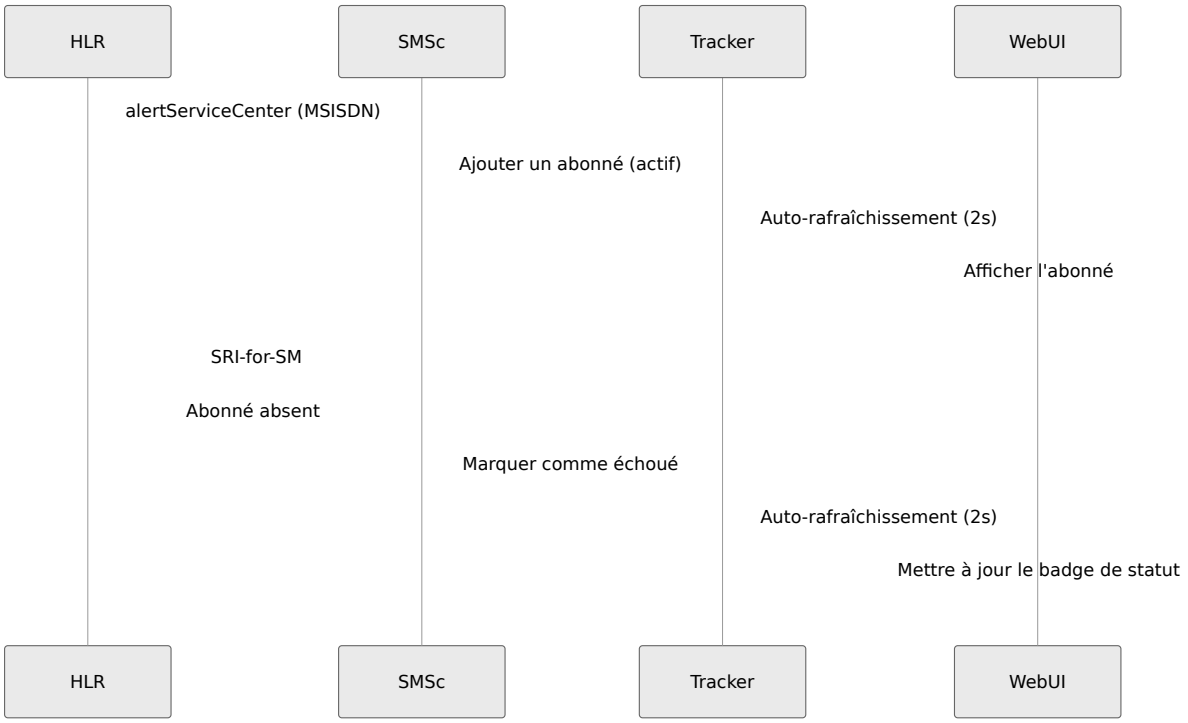
---

# Page des abonnés SMSc

**Page** : `/smsc_subscribers` **Mode** : SMSc uniquement **Auto-rafraîchissement** : Toutes les 2 secondes

Affiche le suivi en temps réel des abonnés basé sur les messages alertServiceCenter reçus des HLR, le statut de livraison des messages et le suivi des échecs.

## Fonctionnalités de la page



## Colonnes du tableau des abonnés

| Colonne | Description | Exemple |
|---|---|---|
| **MSISDN** | Numéro de téléphone de l'abonné | `"15551234567"` |
| **IMSI** | IMSI de l'abonné | `"001010123456789"` |
| **HLR GT** | HLR GT qui a envoyé alertServiceCenter | `"15551111111"` |
| **Msgs Envoyés** | Nombre de messages MT-FSM envoyés | `5` |
| **Msgs Reçus** | Nombre de messages MO-FSM reçus | `2` |
| **Statut** | `Actif` ou `Échoué` (codé par couleur) | `● Actif` |
| **Dernière mise à jour** | Horodatage de la dernière mise à jour | `"2025-10-30 14:23:45 UTC"` |
| **Durée** | Temps écoulé depuis la dernière mise à jour | `"15m 34s"` |

## Indicateurs de statut

- ● **Actif** (Vert) : L'abonné est joignable, dernier alertServiceCenter reçu avec succès
- ○ **Échoué** (Rouge) : La dernière tentative de livraison a échoué (erreur SRI-for-SM ou abonné absent)

## Résumé des statistiques

Lorsque des abonnés sont présents, une carte de résumé affiche :

- **Total Suivi** : Nombre total d'abonnés suivis
- **Actifs** : Nombre d'abonnés avec statut actif
- **Échoués** : Nombre d'abonnés avec statut échoué
- **HLR Uniques** : Nombre de HLR distincts envoyant des alertes

## Gestion des abonnés

**Bouton Supprimer** : Supprime un abonné individuel du suivi.

**Bouton Effacer tout** : Supprime tous les abonnés suivis.

**Confirmation** : Effacer tout nécessite une confirmation avant de vider (ne peut pas être annulé).

**Cas d'utilisation** :

- Supprimer les entrées obsolètes après des problèmes de réseau
- Effacer les données de test après le développement
- Surveiller quels HLR envoient des alertes

## Compteurs de messages

Le tracker incrémente automatiquement les compteurs :

- **Messages envoyés** : Incrémenté lorsque SRI-for-SM réussit et que MT-FSM est envoyé
- **Messages reçus** : Incrémenté lorsque MO-FSM est reçu de l'abonné

## Auto-rafraîchissement

La page se rafraîchit automatiquement toutes les **2 secondes** pour afficher les mises à jour en temps réel des abonnés et des statuts.

---

# Opérations courantes

## Recherche et filtrage

Actuellement, l'interface utilisateur Web n'inclut pas de fonctionnalité de recherche/filtrage intégrée. Pour trouver des routes spécifiques :

1. Utilisez la fonction de recherche de votre navigateur (Ctrl+F / Cmd+F)
2. Recherchez des noms de pairs, des codes de point ou des préfixes GT

## Opérations en masse

Pour effectuer des modifications de routes en masse :

1. **Option 1** : Utilisez l'API REST pour un accès programmatique
2. **Option 2** : Modifiez `config/runtime.exs` et redémarrez l'application
3. **Option 3** : Utilisez l'interface utilisateur Web pour des modifications de routes individuelles

## Exporter/Importer

**Remarque** : L'interface utilisateur Web ne prend actuellement pas en charge l'exportation ou l'importation de tables de routage. Les routes sont :

- Stockées dans des fichiers de base de données Mnesia
- Configurées dans `config/runtime.exs`

Pour sauvegarder les routes :

1. **Mnesia** : Sauvegardez le répertoire `Mnesia.{node_name}/`
2. **Configuration** : Contrôle de version de `config/runtime.exs`

---

# Comportement d'auto-rafraîchissement

Différentes pages ont différents intervalles de rafraîchissement :

| Page | Intervalle de rafraîchissement | Raison |
|---|---|---|
| Gestion du routage | 5 secondes | Les changements de route sont peu fréquents |
| Abonnés actifs | 2 secondes | L'état des abonnés change fréquemment |
| Statut M3UA | Varie selon la page | Surveillance de l'état de connexion |

**Connexion WebSocket** : Toutes les pages utilisent des connexions WebSocket Phoenix LiveView pour des mises à jour en temps réel.

**Interruption de réseau** : Si la connexion WebSocket est perdue, la page tentera de se reconnecter automatiquement.

---

# Dépannage

## Page ne se charge pas

1. **Vérifiez le certificat HTTPS** : Assurez-vous que `priv/cert/omnitouch.crt` et `priv/cert/omnitouch.pem` sont présents
2. **Vérifiez le port 8087** : Vérifiez que les règles de pare-feu autorisent le trafic HTTPS sur le port `8087`
3. **Application en cours d'exécution** : Confirmez que l'application fonctionne avec `iex -S mix`

4. **Console du navigateur** : Vérifiez les erreurs de certificat SSL (avertissements de certificat auto-signé)

## Routes ne persistant pas

1. **Vérifiez le stockage Mnesia** : Vérifiez `mnesia_storage_type: :disc_copies` dans la configuration
2. **Répertoire Mnesia** : Assurez-vous que le répertoire Mnesia est accessible en écriture
3. **Vérifiez les journaux** : Recherchez des erreurs Mnesia dans les journaux de l'application

## Auto-rafraîchissement ne fonctionne pas

1. **Connexion WebSocket** : Vérifiez la console du navigateur pour des erreurs WebSocket
2. **Réseau** : Vérifiez la connexion réseau stable
3. **Rafraîchissement de la page** : Essayez de rafraîchir la page (F5)

---

# Documentation connexe

- **Guide STP** - Configuration détaillée du routage
- **Guide HLR** - Gestion des abonnés
- **Guide API** - API REST pour un accès programmatique
- **Référence de configuration** - Tous les paramètres de configuration

---

# Résumé

L'interface utilisateur Web OmniSS7 fournit une gestion intuitive et en temps réel des tables de routage et du suivi des abonnés :

 **Mises à jour en temps réel** - L'auto-rafraîchissement maintient les données à jour

 **Stockage persistant** - Mnesia garantit que les routes survivent aux redémarrages
 **Interface utilisateur basée sur les rôles** - Les pages s'adaptent au mode opérationnel (STP/HLR/SMSc)
 **Gestion interactive** - Ajouter, modifier, supprimer des routes sans redémarrage
 **Surveillance de l'état** - État de connexion et statut des pairs en direct

Pour des opérations avancées ou de l'automatisation, consultez le Guide API.

# Exemples Avancés de Routage de Titre Global

← Retour à la Documentation Principale

Ce document fournit des exemples pratiques d'utilisation du Type de Traduction (TT), de l'Indicateur de Plan de Numérotation (NPI) et de l'Indicateur de Nature d'Adresse (NAI) pour des scénarios avancés de routage GT.

## Table des Matières

1. Concepts de Base
2. Exemples de Type de Traduction (TT)
3. Exemples d'Indicateur de Plan de Numérotation (NPI)
4. Exemples d'Indicateur de Nature d'Adresse (NAI)
5. Exemples Combinés
6. Cas d'Utilisation Réels

---

## Concepts de Base

### Référence des Valeurs d'Indicateur

**Type de Traduction (TT) :**

- `0` = Inconnu
- `1` = International
- `2` = National
- `3` = Spécifique au Réseau

**Indicateur de Plan de Numérotation (NPI) :**

- `0` = Inconnu
- `1` = ISDN/Téléphonie (E.164)
- `3` = Données (X.121)
- `4` = Telex (F.69)
- `6` = Mobile Terrestre (E.212)

**Indicateur de Nature d'Adresse (NAI) :**

- `0` = Inconnu
- `1` = Numéro d'Abonné
- `2` = Réservé à un Usage National
- `3` = Numéro Significatif National
- `4` = Numéro International

## Correspondance vs Transformation

- **Indicateurs source** (source_tt, source_npi, source_nai) : Utilisés pour **correspondre** aux messages entrants
- **Indicateurs de destination** (dest_tt, dest_npi, dest_nai) : Utilisés pour **transformer** les valeurs lors du transfert
- Laissez les champs vides (`nil`) pour une correspondance par joker ou pour préserver les valeurs originales

---

# Exemples de Type de Traduction (TT)

## Exemple 1 : Conversion de TT Inconnu à International

**Scénario :** Le réseau partenaire envoie tous les GT avec TT=0 (Inconnu), mais votre HLR en aval s'attend à TT=1 (International).

```
%{
  gt_prefix: "44",
  peer_id: 1,
  priority: 1,
  description: "UK: Convertir TT Inconnu en International",
  source_tt: 0,      # Correspondre à TT=0 entrant
  dest_tt: 1         # Transformer en TT=1
}
```

**Flux de trafic :**

- Entrant : GT=441234567890, TT=0
- Routage : Correspond à gt_prefix="44", source_tt=0
- Sortant : GT=441234567890, TT=1 (transformé)

# Exemple 2 : Routes Séparées par Type de Traduction

**Scénario :** Routage de différentes valeurs TT vers différents réseaux.

```
# Routage TT=0 vers le Réseau A
%{
  gt_prefix: "33",
  peer_id: 1,
  priority: 1,
  description: "Numéros français avec TT=0 vers le Réseau A",
  source_tt: 0
},

# Routage TT=2 vers le Réseau B
%{
  gt_prefix: "33",
  peer_id: 2,
  priority: 1,
  description: "Numéros français avec TT=2 vers le Réseau B",
  source_tt: 2
}
```

**Flux de trafic :**

- GT="33123456789", TT=0 → Routé vers le pair 1
- GT="33123456789", TT=2 → Routé vers le pair 2

---

# Exemples d'Indicateur de Plan de Numérotation (NPI)

## Exemple 1 : Conversion de Mobile à ISDN NPI

**Scénario :** Le réseau mobile utilise NPI=6 (E.212), le PSTN s'attend à NPI=1 (E.164).

```
%{
  gt_prefix: "49",
  peer_id: 1,
  priority: 1,
  description: "Conversion mobile→ISDN allemande",
  source_npi: 6,     # Correspondre à NPI=6 (Mobile/E.212)
  dest_npi: 1        # Transformer en NPI=1 (ISDN/E.164)
}
```

**Cas d'utilisation :** Interconnexion des réseaux mobiles et fixes avec des plans de numérotation différents.

## Exemple 2 : Routage Basé sur le Plan de Numérotation

**Scénario :** Séparer le trafic réseau de données du trafic vocal.

```
# Routage du trafic réseau de données (NPI=3)
%{
  gt_prefix: "1",
  peer_id: 1,
  priority: 1,
  description: "Trafic réseau de données",
  source_npi: 3      # Correspondre à NPI=3 (Données/X.121)
},

# Routage du trafic réseau vocal (NPI=1)
%{
  gt_prefix: "1",
  peer_id: 2,
  priority: 1,
  description: "Trafic réseau vocal",
  source_npi: 1      # Correspondre à NPI=1 (ISDN)
}
```

---

# Exemples d'Indicateur de Nature d'Adresse (NAI)

## Exemple 1 : Conversion de NAI International à National

**Scénario :** Format international entrant (NAI=4), convertir en format national (NAI=3) pour le routage domestique.

```
%{
  gt_prefix: "1",
  peer_id: 1,
  priority: 1,
  description: "US: Format International→National",
  source_nai: 4,     # Correspondre à NAI=4 (International)
  dest_nai: 3        # Transformer en NAI=3 (National)
}
```

**Flux de trafic :**

- Entrant : GT=12125551234, NAI=4 (format international)
- Sortant : GT=12125551234, NAI=3 (format national)

## Exemple 2 : Routage Abonné vs Réseau NAI

**Scénario :** Routage basé sur le fait qu'il s'agit d'un numéro d'abonné ou d'un numéro de réseau.

```
# Routage des numéros d'abonnés vers HLR
%{
  gt_prefix: "44",
  peer_id: 1,
  priority: 1,
  description: "Numéros d'abonnés UK vers HLR",
  source_nai: 1      # Correspondre à NAI=1 (Numéro d'Abonné)
},

# Routage des numéros de réseau vers MSC
%{
  gt_prefix: "44",
  peer_id: 2,
  priority: 1,
  description: "Adresses réseau UK vers MSC",
  source_nai: 3      # Correspondre à NAI=3 (Numéro Significatif
National)
}
```

---

# Exemples Combinés

## Exemple 1 : Normalisation du Trafic SMS

**Scénario :** Normaliser tout le trafic SMS provenant de diverses sources avant de le router vers HLR.

```
%{
  gt_prefix: "86",
  source_ssn: 8,           # Correspondre à SSN=8 (SMSC)
  peer_id: 1,
  dest_ssn: 6,             # Réécrire en SSN=6 (HLR)
  priority: 1,
  description: "Normalisation SMS chinoise",
  source_tt: 0,            # Correspondre à TT=0 (Inconnu)
  dest_tt: 2,              # Transformer en TT=2 (National)
  source_npi: 6,           # Correspondre à NPI=6 (Mobile)
  dest_npi: 1,             # Transformer en NPI=1 (ISDN)
  source_nai: 4,           # Correspondre à NAI=4 (International)
  dest_nai: 3              # Transformer en NAI=3 (National)
}
```

**Ce que cela fait :**

- Correspond au trafic SMS (SSN=8) pour les numéros chinois (préfixe GT "86")
- Normalise tous les indicateurs à des valeurs standard
- Réécrit le SSN de destination de SMSC (8) à HLR (6)
- Parfait pour les scénarios d'interconnexion avec une utilisation incohérente des indicateurs

## Exemple 2 : Routage Multi-Opérateurs avec Conversion de Format

**Scénario :** Routage vers différents opérateurs en fonction du TT, en convertissant les formats si nécessaire.

```
# Opérateur A : S'attend à un TT National et un NPI ISDN
%{
  gt_prefix: "44",
  peer_id: 1,
  priority: 1,
  description: "Opérateur A UK : Normaliser en National/ISDN",
  source_tt: 0,            # Correspondre à tout TT=0
  dest_tt: 2,              # Convertir en TT=2 (National)
  source_npi: 6,           # Correspondre au NPI Mobile
  dest_npi: 1              # Convertir au NPI ISDN
},

# Opérateur B : S'attend à un TT International, préserver NPI
%{
  gt_prefix: "44",
  peer_id: 2,
  priority: 2,             # Priorité inférieure (numéro plus
élevé)
  description: "Opérateur B UK : Convertir en TT International",
  source_tt: 2,            # Correspondre au TT National
  dest_tt: 1               # Convertir en TT International
  # NPI laissé vide = préserver l'original
}
```

---

# Cas d'Utilisation Réels

## Cas d'Utilisation 1 : Intégration de Système Hérité

**Problème :** Intégrer un réseau moderne avec un système hérité qui utilise des conventions d'indicateurs différentes.

**Solution :** Transformer tous les indicateurs au format hérité.

```
%{
  gt_prefix: "555",
  peer_id: 1,
  priority: 1,
  description: "Normalisation HLR hérité",
  # Correspondre à tout format moderne
  source_tt: 1,           # Moderne : International
  dest_tt: 0,             # Hérité : Inconnu
  source_npi: 1,          # Moderne : ISDN
  dest_npi: 0,            # Hérité : Inconnu
  source_nai: 4,          # Moderne : International
  dest_nai: 0             # Hérité : Inconnu
}
```

## Cas d'Utilisation 2 : Normalisation de l'Interconnexion Internationale

**Problème :** Plusieurs partenaires internationaux envoient du trafic avec des indicateurs incohérents.

**Solution :** Normaliser tout le trafic entrant au format cohérent.

```
# Partenaire A envoie TT=0, convertir en TT=1 standard
%{
  gt_prefix: "1",
  peer_id: 1,
  priority: 1,
  description: "Normalisation Partenaire A US",
  source_tt: 0,
  dest_tt: 1,
  dest_npi: 1,             # Assurer NPI ISDN
  dest_nai: 4              # Assurer NAI International
},

# Partenaire B envoie TT=3 (Spécifique au Réseau), convertir en
TT=1
%{
  gt_prefix: "44",
  peer_id: 2,
  priority: 1,
  description: "Normalisation Partenaire B UK",
  source_tt: 3,
  dest_tt: 1,
  dest_npi: 1,
  dest_nai: 4
}
```

## Cas d'Utilisation 3 : Gestion du Trafic d'Itinérance

**Problème :** Le réseau d'origine et les partenaires d'itinérance utilisent différents formats d'adresse.

**Solution :** Convertir le trafic d'itinérance au format du réseau d'origine.

```
# Itinérance entrante : Convertir au format du réseau d'origine
%{
  gt_prefix: "12025",      # Zone de Washington DC
  peer_id: 1,
  priority: 1,
  description: "Normalisation itinérante entrante",
  source_nai: 4,           # International du partenaire
d'itinérance
  dest_nai: 3,             # National pour le réseau d'origine
  source_npi: 6,           # Réseau mobile
  dest_npi: 1              # ISDN pour le HLR d'origine
}
```

## Cas d'Utilisation 4 : Migration de Réseau

**Problème :** Migration progressive d'un ancien schéma d'adressage vers un nouveau schéma.

**Solution :** Routage basé sur les indicateurs source, transformation pour la destination.

```
# Trafic au format ancien : Transformer en nouveau format
%{
  gt_prefix: "999",
  peer_id: 1,
  priority: 1,
  description: "Migration du format ancien au nouveau",
  source_tt: 0,            # Ancien : Inconnu
  dest_tt: 2,              # Nouveau : National
  source_npi: 0,           # Ancien : Inconnu
  dest_npi: 1,             # Nouveau : ISDN
  source_nai: 0,           # Ancien : Inconnu
  dest_nai: 3              # Nouveau : National
},

# Trafic au format nouveau : Passer sans changement
%{
  gt_prefix: "999",
  peer_id: 1,
  priority: 2,
  description: "Passage au format nouveau",
  source_tt: 2,            # Déjà au nouveau format
  # Pas de champs dest_ = préserver toutes les valeurs
}
```

---

# Tester Vos Routes

## Via l'Interface Web

1. Naviguez vers **Routage → Routes de Titre Global**
2. Cliquez sur **"Afficher le Routage Avancé (TT/NPI/NAI)"**
3. Remplissez vos indicateurs source et de destination
4. Cliquez sur **Ajouter une Route**
5. Allez à la page **Test de Routage** pour tester la route

## Via l'API

Le point de terminaison de la route GT est monté à `/routing/gt-routes` sur l'écouteur API (port **8445**, TLS activé). Le corps JSON utilise le champ `prefix` (pas `gt_prefix`), et `prefix` et `peer_id` sont requis. Les correspondances et réécritures TT/NPI/NAI sont optionnelles.

```
curl -k -X POST https://localhost:8445/routing/gt-routes \
  -H "Content-Type: application/json" \
  -d '{
    "prefix": "44",
    "peer_id": 1,
    "priority": 1,
    "description": "Conversion TT UK",
    "source_tt": 0,
    "dest_tt": 1
  }'
```

| Champ | Type | Requis | Description |
|---|---|---|---|
| `prefix` | Chaîne | Oui | Préfixe GT à correspondre |
| `peer_id` | Entier | Oui | ID du pair cible (`0` = DROP) |
| `priority` | Entier | Non | Priorité de la route (par défaut `1`, le plus bas gagne) |
| `description` | Chaîne | Non | Description lisible par l'homme |
| `source_ssn` / `dest_ssn` | Entier | Non | Correspondre / réécrire le SSN de la Partie Appelée |
| `source_tt` / `dest_tt` | Entier | Non | Correspondre / réécrire le Type de Traduction |
| `source_npi` / `dest_npi` | Entier | Non | Correspondre / réécrire l'Indicateur de Plan de Numérotation |
| `source_nai` / `dest_nai` | Entier | Non | Correspondre / réécrire l'Indicateur de Nature d'Adresse |

Pour supprimer une route, émettez un `DELETE` à `/routing/gt-routes/{prefix}:{peer_id}` (par exemple `/routing/gt-routes/44:1`).

## Tester la Correspondance des Routes

Utilisez la page de Test de Routage pour simuler le trafic entrant avec des indicateurs spécifiques :

1. Entrez le préfixe GT (par exemple, "441234567890")
2. Définissez les valeurs TT, NPI, NAI
3. Cliquez sur "Tester la Route"
4. Voyez quelle route correspond et quelles transformations s'appliquent

---

# Dépannage

## Routes Non Correspondantes

**Vérifiez l'ordre de spécificité :**

1. Le préfixe GT le plus long gagne
2. Ensuite, le source_ssn spécifique sur wildcard
3. Ensuite, le source_tt spécifique sur wildcard
4. Ensuite, le source_npi spécifique sur wildcard
5. Ensuite, le source_nai spécifique sur wildcard
6. Enfin, le numéro de priorité le plus bas gagne

## Transformations Non Appliquées

**Vérifiez que les champs dest_ sont définis :**

- Un dest_tt/dest_npi/dest_nai vide signifie "préserver la valeur originale"
- Doit explicitement définir dest_* pour appliquer la transformation

## Problèmes de Routes Dupliquées

**:bag table permet des duplicatas :**

- La table des routes GT est de type `:bag` (permet plusieurs enregistrements avec la même clé)
- La clé est `{gt_prefix, source_ssn}`
- Utilisez TT/NPI/NAI pour différencier les routes avec le même préfixe+SSN
- Supprimez les routes dupliquées non désirées

---

# Meilleures Pratiques

1. **Documentez vos conversions :** Utilisez des descriptions de route descriptives

2. **Testez minutieusement :** Utilisez la page de Test de Routage avant la production
3. **Surveillez les journaux :** Vérifiez les journaux INFO pour les transformations appliquées
4. **Commencez simple :** Ajoutez d'abord des routes de préfixe de base, puis ajoutez des règles d'indicateur
5. **Utilisez les jokers judicieusement :** Les champs source_* vides correspondent à n'importe quelle valeur
6. **Préservez lorsque c'est possible :** Transformez uniquement ce qui doit être changé
7. **Considérez la spécificité :** Les routes plus spécifiques remplacent celles moins spécifiques

---

Pour plus d'informations, voir :


- ← Retour à la Documentation Principale
- Guide STP - Documentation complète sur le routage
- Référence de Configuration - Toutes les options de configuration
- Guide API - Documentation de l'API REST


---

**OmniSS7** par Omnitouch Network Services

# Routage de Titre Global M3UA STP

← Retour à la Documentation Principale

## Aperçu

Le M3UA STP prend en charge le **routage de Titre Global (GT)** en plus du routage par Code de Point (PC). Le routage GT achemine les messages SS7 en fonction du Titre Global de la Partie Appelée dans la couche SCCP, avec un retour automatique au routage PC.

## Comment ça fonctionne

### Priorité de Routage

1. **Routage GT (si activé)** : Extraire le GT Appelé de SCCP → Rechercher dans la table de routage GT
2. **Routage PC (retour)** : Extraire DPC de MTP3 → Rechercher dans la table de routage PC

### Correspondance de Préfixe GT

- Utilise l'algorithme de **correspondance de préfixe le plus long**
- Exemple : GT "447123456789"
  - Correspond à "447" (longueur 3) plutôt qu'à "44" (longueur 2)
  - Achemine vers le pair configuré pour le préfixe "447"

# Configuration

## Activer le Routage GT

Dans `config/runtime.exs` :

```
config :omniss7,
  enable_gt_routing: true,  # Mettre à false pour utiliser
uniquement le routage PC
```

## Définir les Routes GT

```
config :omniss7,
  m3ua_gt_routes: [
    # Route les numéros US (commençant par 1234) vers le pair 1
    %{gt_prefix: "1234", peer_id: 1, priority: 1, description:
"Numéros US"},

    # Route les numéros UK (commençant par 44) vers le pair 2
    %{gt_prefix: "44", peer_id: 2, priority: 1, description:
"Numéros UK"},

    # Route plus spécifique pour mobile UK (447xxx)
    %{gt_prefix: "447", peer_id: 2, priority: 1, description:
"Mobile UK"},

    # Route les numéros Australiens (commençant par 61) vers le
pair 3
    %{gt_prefix: "61", peer_id: 3, priority: 1, description:
"Numéros Australiens"}
  ]
```

> Une configuration STP complète, prête à être modifiée, est fournie dans `config/stp_example_runtime.exs`. Consultez le Guide STP pour savoir comment l'activer.

# Exemples d'Utilisation

## Exemple 1 : Routage GT de Base

**Message** : SCCP UDT avec GT Appelé = "447700123456"

**Décision de Routage** :

- GT extrait : "447700123456"
- Meilleure correspondance : "447" (mobile UK) → pair 2
- Message acheminé vers le pair 2

## Exemple 2 : GT Non Trouvé, Retour au PC

**Message** : SCCP UDT avec GT Appelé = "331234567890", DPC = 100

**Décision de Routage** :

- GT extrait : "331234567890"
- Aucune route GT trouvée pour "33"
- Retour au routage PC
- DPC 100 → pair 1 (de la table de routage PC)
- Message acheminé vers le pair 1

## Exemple 3 : Pas de GT Présent

**Message** : M3UA DATA sans SCCP, uniquement MTP3 avec DPC = 200

**Décision de Routage** :

- Pas de couche SCCP, pas de GT
- Utiliser uniquement le routage PC
- DPC 200 → pair 2
- Message acheminé vers le pair 2

# Gestion à l'Exécution

Ces fonctions sont exposées par `M3UARouting` et peuvent être appelées depuis une console `iex -S mix`.

## Interroger les Routes GT

```
# Obtenir toutes les routes GT
{:ok, routes} = M3UARouting.get_all_gt_routes()

# Chaque enregistrement est un tuple de 14 éléments. Les champs
sont :
# {:m3ua_gt_route, gt_prefix, source_ssn, peer_id, dest_ssn,
priority,
#  description, enabled, source_tt, dest_tt, source_npi, dest_npi,
#  source_nai, dest_nai}
#
# Exemple de sortie (source_ssn et les champs de
réécriture/indicateur par défaut à nil) :
[
  {:m3ua_gt_route, "1234", nil, 1, nil, 1, "Numéros US", true,
nil, nil, nil, nil, nil, nil},
  {:m3ua_gt_route, "44", nil, 2, nil, 1, "Numéros UK", true, nil,
nil, nil, nil, nil, nil},
  {:m3ua_gt_route, "447", nil, 2, nil, 1, "Mobile UK", true, nil,
nil, nil, nil, nil, nil},
  {:m3ua_gt_route, "61", nil, 3, nil, 1, "Numéros Australiens",
true, nil, nil, nil, nil, nil, nil}
]
```

## Ajouter une Route GT Dynamiquement

`add_gt_route/3` prend le préfixe et l'ID du pair en position ; tout le reste (`priority`, `description`, `source_ssn`, `dest_ssn`, les correspondances et réécritures TT/NPI/NAI, et `enabled`) est une option par mot-clé.

```
# Signature : add_gt_route(gt_prefix, peer_id, opts \ [])

# Ajouter une route pour les numéros allemands
M3UARouting.add_gt_route("49", 4, priority: 1, description:
"Numéros Allemagne")

# Ajouter une route SMS qui correspond à SSN 8 (SMSC) et réécrit
le SSN à 6 (HLR)
M3UARouting.add_gt_route("55", 4,
  source_ssn: 8,
  dest_ssn: 6,
  priority: 1,
  description: "Trafic SMS vers HLR"
)
```

## Supprimer une Route GT

```
# Supprimer la route (gt_prefix, peer_id, source_ssn optionnel)
M3UARouting.remove_gt_route("49", 4)
```

## Tester la Recherche GT

```
# Tester quel pair gérerait ce GT
# Signature : lookup_peer_by_gt(global_title, source_ssn \ nil,
opts \ [])
{:ok, peer, rewrites} =
M3UARouting.lookup_peer_by_gt("447700123456")

# `peer` est l'enregistrement m3ua_peer de 13 éléments :
# {:m3ua_peer, peer_id, peer_name, protocol, role, local_ip,
local_port,
#  remote_ip, remote_port, routing_context, point_code,
network_indicator, status}
#
# `rewrites` est une MAP des réécritures d'indicateur à appliquer
lors de l'acheminement.
# Les valeurs nil signifient "garder l'original" :
#   %{dest_ssn: nil, dest_tt: nil, dest_npi: nil, dest_nai: nil}
```

Lorsqu'une route spécifie des réécritures, elles apparaissent dans la carte. Par exemple, une route "55" avec `dest_ssn: 6` :

```
{:ok, peer, rewrites} =
M3UARouting.lookup_peer_by_gt("5512345678", 8)
# rewrites => %{dest_ssn: 6, dest_tt: nil, dest_npi: nil,
dest_nai: nil}
```

Une route avec `peer_id: 0` (DROP) retourne `{:error, :dropped}` ; un GT non apparié retourne `{:error, :no_gt_route}`.

# Journalisation

Le STP journalise les décisions de routage GT :

```
[info] Routage M3UA DATA : OPC=1000, DPC=200, GT
Appelé="447700123456" (du pair 1)
[info] Routage par GT : 447700123456
[info] Message routé du pair 1 au pair 2
```

Si le routage GT échoue :

```
[info] Routage M3UA DATA : OPC=1000, DPC=200, GT
Appelé="331234567" (du pair 1)
[info] Aucune route GT trouvée pour 331234567
[info] Message routé du pair 1 au pair 2 (via routage PC)
```

# Support SCCP

## Messages SCCP Supportés

| Message | Code | Support |
|---|---|---|
| **UDT** (Unitdata) | 0x09 | Complet : extraction GT, réécriture d'indicateur, routage |
| **UDTS** (Unitdata Service) | 0x0A | Complet : généré comme message de retour en cas d'échec de routage |
| **XUDT** (Extended Unitdata) | 0x11 | Complet : extraction GT, réécriture d'indicateur, décrément du compteur de sauts, message de retour |
| **XUDTS** (Extended Unitdata Service) | 0x12 | Complet : généré comme message de retour en cas d'échec de routage |
| **LUDT** (Long Unitdata) | 0x13 | Complet : extraction GT, réécriture d'indicateur, décrément du compteur de sauts, message de retour |
| **LUDTS** (Long Unitdata Service) | 0x14 | Complet : généré comme message de retour en cas d'échec de routage |

XUDT et LUDT transportent un compteur de sauts SCCP. Le STP le décrémente à chaque saut de transit et, s'il atteint zéro, il rejette le message et le renvoie à l'initiateur (lorsqu'un retour en cas d'erreur est demandé). Consultez le Guide STP pour plus de détails.

## Formats GT Supportés

- **GTI 0-4** : Tous les types d'indicateurs GT
- **Encodage BCD** : Encodage standard des chiffres de téléphonie
- **Nature de l'Adresse** : Tous les types (International, National, etc.)

# Performance

- **Recherche GT** : O(n) où n = nombre de routes GT
- **Correspondance de Préfixe** : Optimisé avec tri par correspondance la plus longue en premier
- **Recommandation** : Gardez le nombre de routes GT < 1000 pour des performances optimales

# Dépannage

## GT Non Extrait

**Problème** : Les journaux montrent `Called GT=nil`

**Solutions** :

1. Vérifiez que le type de message SCCP est UDT/XUDT/LUDT
2. Vérifiez que l'Adresse de la Partie Appelée porte un Titre Global
3. Activez la journalisation de débogage pour voir les détails de l'analyse SCCP

## Mauvais Pair Sélectionné

**Problème** : Message acheminé vers le mauvais pair

**Solutions** :

1. Vérifiez la configuration du préfixe GT (la correspondance la plus longue l'emporte)
2. Vérifiez que `peer_id` correspond aux pairs configurés
3. Vérifiez les valeurs de priorité (plus bas = plus haute priorité)

## Retour au PC Non Fonctionnel

**Problème** : Message rejeté lorsque GT non trouvé

**Solutions** :

1. Vérifiez que `enable_gt_routing` est `true`
2. Vérifiez que la table de routage PC a une route pour le DPC
3. Assurez-vous que le pair pour le PC est actif

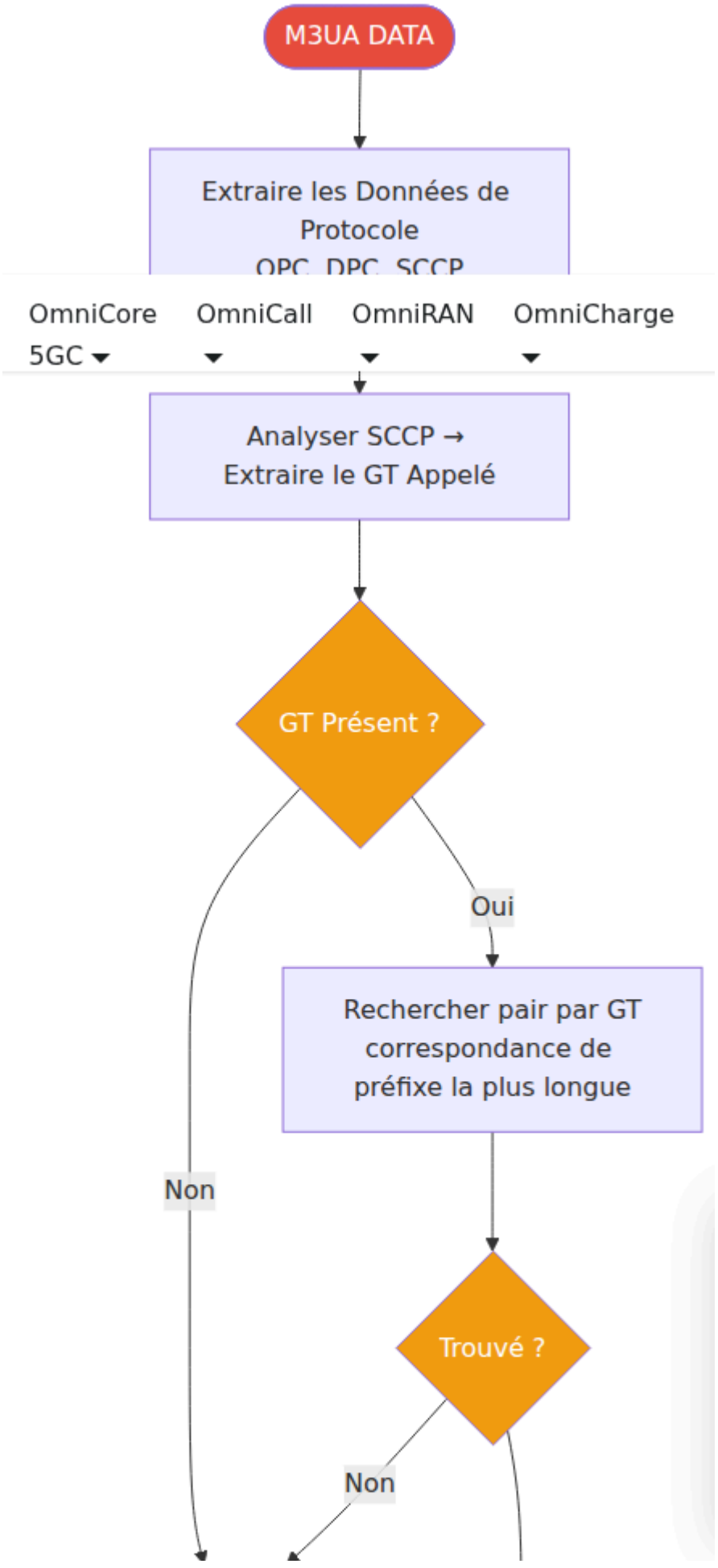
# Meilleures Pratiques

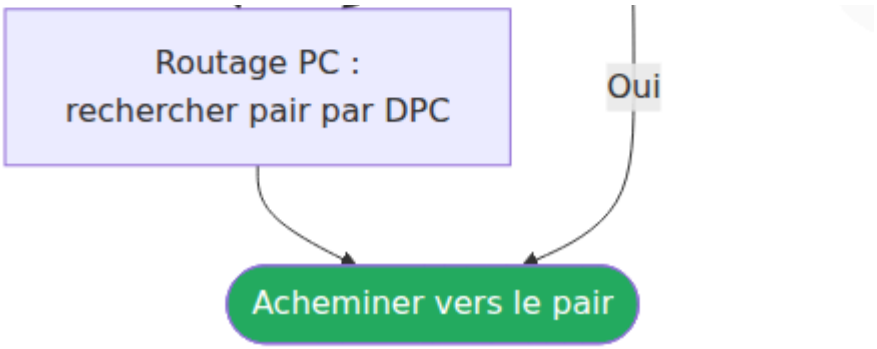
1. **Utilisez des Préfixes Spécifiques** : Des routes plus spécifiques remplacent les moins spécifiques
   - Bon : "447", "448", "449" pour différents opérateurs UK
   - Mauvais : Un seul "44" pour tous les UK
2. **Documentez les Routes** : Utilisez le champ de description

```
M3UARouting.add_gt_route("1234", 1, priority: 1,
  description: "ACME Corp - Côte Est des États-Unis")
```

3. **Testez les Routes** : Utilisez `lookup_peer_by_gt/1` pour vérifier
4. **Surveillez** : Vérifiez les journaux pour les échecs de routage
5. **Retour** : Configurez toujours des routes PC comme sauvegarde

# Architecture


M3UA DATA
Extraire les Données de Protocole
OPC, DPC, SCCP
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Analyser SCCP → Extraire le GT Appelé
GT Présent ?
Oui
Non
Rechercher pair par GT correspondance de préfixe la plus longue
Trouvé ?
Non




---

**OmniSS7** par Omnitouch Network Services