# Guide de l'API REST

← Retour à la documentation principale

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

## Table des matières



---

## Aperçu

OmniSS7 fournit une API REST pour un accès programmatique aux opérations MAP (Mobile Application Part). L'API vous permet de :

- Envoyer des requêtes MAP (SRI, SRI-for-SM, UpdateLocation, etc.)
- Récupérer des réponses MAP
- Surveiller les métriques système via Prometheus

### Architecture de l'API

# Configuration du serveur HTTP

## Détails du serveur

| Paramètre | Valeur | Configurable |
|---|---|---|
| **Protocole** | HTTP | Non |
| **Adresse IP** | 0.0.0.0 (toutes les interfaces) | Via code uniquement |
| **Port** | 8080 | Via code uniquement |
| **Transport** | Plug.Cowboy | Non |

**URL d'accès** : `http://[server-ip]:8080`

## Activation/Désactivation du serveur HTTP

Contrôlez si le serveur HTTP démarre :

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

**Par défaut** : `true` (activé)

**Lorsqu'il est désactivé** : Le serveur HTTP ne démarrera pas, et l'API REST/l'interface Swagger seront indisponibles.

---

# Interface Swagger

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

## Accéder à l'interface Swagger

**URL** : `http://[server-ip]:8080/swagger`

**Fonctionnalités** :

- Documentation API interactive
- Fonctionnalité d'essai pour tester les points de terminaison
- Schémas de requête/réponse
- Exemples de charges utiles

## Swagger JSON

La spécification OpenAPI est disponible à :

**URL** : `http://[server-ip]:8080/swagger.json`

**Cas d'utilisation** :

- Importer dans Postman ou d'autres clients API
- Générer des bibliothèques clientes
- Automatisation de la documentation API

---

# Points de terminaison de l'API

Tous les points de terminaison des opérations MAP suivent le modèle : `POST /api/{operation}`

## Résumé des points de terminaison

| Point de terminaison | Méthode | Objectif | Délai d'attente |
|---|---|---|---|
| `/api/sri` | POST | Envoyer des informations de routage | 10s |
| `/api/sri-for-sm` | POST | Envoyer des informations de routage pour SM | 10s |
| `/api/send-auth-info` | POST | Envoyer des informations d'authentification | 10s |
| `/api/MT-forwardSM` | POST | Mobile Terminated Forward SM | 10s |
| `/api/forwardSM` | POST | Forward SM | 10s |
| `/api/updateLocation` | POST | Mettre à jour la localisation | 10s |
| `/api/prn` | POST | Fournir un numéro de roaming | 10s |
| `/metrics` | GET | Métriques Prometheus | N/A |
| `/swagger` | GET | Interface Swagger | N/A |
| `/swagger.json` | GET | Spécification OpenAPI | N/A |

**Remarque** : Toutes les requêtes MAP ont un **délai d'attente de 10 secondes codé en dur**.

---

## 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"
}
```

**Paramètres** :

| 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",
    ...
  }
}
```

**Erreur** (504 Gateway Timeout) :

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

**Exemple cURL** :

```
curl -X POST http://localhost:8080/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"
}
```

**Paramètres** :

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

**Réponse** (200 OK) :

```
{
  "result": {
    "imsi": "001001234567890",
    "msc_number": "5551234800",
    "location_info": {...},
    ...
  }
}
```

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "1234567890",
    "service_center": "5551234567"
  }'
```

---

## SendAuthenticationInfo

Demander des vecteurs d'authentification pour un abonné.

**Point de terminaison** : `POST /api/send-auth-info`

**Corps de la requête** :

```
{
  "imsi": "001001234567890",
  "vectors": 3
}
```

**Paramètres** :

| Champ | Type | Requis | Description |
|---|---|---|---|
| `imsi` | String | Oui | IMSI de l'abonné |
| `vectors` | Integer | Oui | Nombre de vecteurs d'authentification à générer |

**Réponse** (200 OK) :

```
{
  "result": {
    "authentication_sets": [
      {
        "rand": "0123456789ABCDEF...",
        "xres": "...",
        "ck": "...",
        "ik": "...",
        "autn": "..."
      }
    ],
    ...
  }
}
```

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/send-auth-info \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vectors": 3
  }'
```

---

# MT-ForwardSM

Livrer un SMS mobile terminé à un abonné.

**Point de terminaison** : `POST /api/MT-forwardSM`

**Corps de la requête** :

```
{
  "imsi": "001001234567890",
  "destination_service_centre": "5551234567",
  "originating_service_center": "5551234568",
  "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
}
```

**Paramètres** :

| Champ | Type | Requis | Description |
|---|---|---|---|
| `imsi` | String | Oui | IMSI de l'abonné de destination |
| `destination_service_centre` | String | Oui | GT du centre de service de destination |
| `originating_service_center` | String | Oui | GT du centre de service d'origine |
| `smsPDU` | String | Oui | SMS TPDU au format hexadécimal |

**Remarque** : `smsPDU` doit être une chaîne encodée en hexadécimal (majuscules ou minuscules).

**Réponse** (200 OK) :

```
{
  "result": {
    "delivery_status": "success",
    ...
  }
}
```

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/MT-forwardSM \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "destination_service_centre": "5551234567",
    "originating_service_center": "5551234568",
    "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
  }'
```

---

# ForwardSM

Transférer un message SMS (MO-SMS de l'abonné).

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

**Corps de la requête** : Identique à MT-ForwardSM

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/forwardSM \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "destination_service_centre": "5551234567",
    "originating_service_center": "5551234568",
    "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
  }'
```

---

# UpdateLocation

Notifier le HLR du changement de localisation de l'abonné (enregistrement VLR).

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

**Corps de la requête** :

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

**Paramètres** :

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

**Réponse** (200 OK) :

```
{
  "result": {
    "hlr_number": "5551234567",
    "subscriber_data": {...},
    ...
  }
}
```

**Remarque** : En mode HLR, cela déclenche la séquence InsertSubscriberData (ISD) avec un délai d'attente de 10 secondes par ISD.

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/updateLocation \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vlr": "5551234800"
  }'
```

---

## ProvideRoamingNumber (PRN)

Demander un MSRN (Mobile Station Roaming Number) pour le routage des appels vers un abonné en itinérance.

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

**Corps de la requête** :

```
{
  "msisdn": "1234567890",
  "gmsc": "5551234567",
  "msc_number": "5551234800",
  "imsi": "001001234567890"
}
```

**Paramètres** :

| Champ | Type | Requis | Description |
|---|---|---|---|
| `msisdn` | String | Oui | MSISDN de l'abonné |
| `gmsc` | String | Oui | GT du MSC de passerelle |
| `msc_number` | String | Oui | Numéro MSC pour l'abonné |
| `imsi` | String | Oui | IMSI de l'abonné |

**Réponse** (200 OK) :

```
{
  "result": {
    "msrn": "5551234999",
    ...
  }
}
```

**Exemple cURL** :

```
curl -X POST http://localhost:8080/api/prn \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "1234567890",
    "gmsc": "5551234567",
    "msc_number": "5551234800",
    "imsi": "001001234567890"
  }'
```

---

# 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 - Corps de requête invalide
- 504 Gateway Timeout - Délai d'attente de la requête MAP (10 secondes)
- 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 | 400 | Champs requis manquants | Incluez tous les paramètres requis |
| Délai d'attente | 504 | La requête MAP a dépassé le délai d'attente de 10s | Vérifiez la connectivité M3UA, la disponibilité HLR/VLR |
| Non trouvé | 404 | Point de terminaison invalide | Vérifiez l'URL du point de terminaison |

## Comportement de délai d'attente

Toutes les requêtes MAP ont un **délai d'attente de 10 secondes codé en dur** :

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 accessible
- Vérifiez la configuration de routage

- Consultez les journaux d'événements SS7 pour des erreurs

---

# Métriques (Prometheus)

L'API expose des métriques Prometheus pour la surveillance.

## Point de terminaison des métriques

**URL** : `http://[server-ip]:8080/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
```

## Métriques disponibles

| Métrique | Type | Étiquettes | Desc |
|---|---|---|---|
| `map_requests_total` | Compteur | `operation` | Nomb de rec MAP p d'opér |
| `cap_requests_total` | Compteur | `operation` | Nomb de rec CAP p d'opér |
| `map_request_duration_milliseconds` | Histogramme | `operation` | Durée requê millise |
| `map_pending_requests` | Jauge | - | Nomb transa MAP e attent |

## Configuration de Prometheus

Ajoutez à votre `prometheus.yml` :

```yaml
scrape_configs:
  - job_name: 'omniss7'
    static_configs:
      - targets: ['server-ip:8080']
    metrics_path: '/metrics'
    scrape_interval: 15s
```

---

# Exemples de requêtes

## Exemple Python

```python
import requests
import json

# Requête SRI-for-SM
url = "http://localhost:8080/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:8080/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();
```

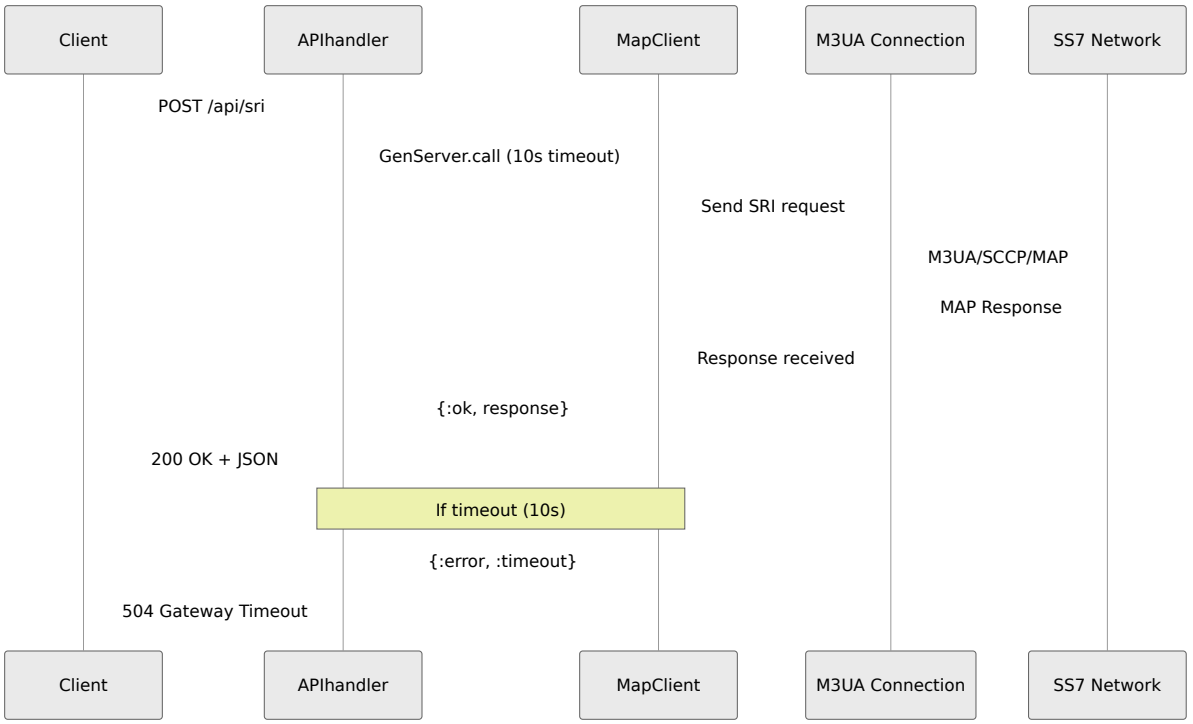
## Exemple Bash/cURL

```
#!/bin/bash

# Requête UpdateLocation
response=$(curl -s -w "\n%{http_code}" -X POST
http://localhost:8080/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 requêtes API




---

# Résumé

L'API REST d'OmniSS7 fournit :

 **Opérations MAP** - Support complet pour SRI, SRI-for-SM, UpdateLocation, livraison de SMS, authentification
 **Interface Swagger** - Documentation et tests interactifs de l'API
 **Métriques Prometheus** - Surveillance et observabilité
 **Délai d'attente codé en dur** - Délai d'attente de 10 secondes pour toutes les requêtes MAP
 **Serveur HTTP** - Fonctionne sur le port 8080 (configurable via `start_http_server`)

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

Pour les détails de configuration, consultez 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

| Opération | Opcode | But |
|---|---|---|
| updateLocation | 2 | Enregistrer l'emplacement de l'abonné |
| cancelLocation | 3 | Désenregistrer du VLR |
| provideRoamingNumber | 4 | Demander MSRN |
| sendRoutingInfo | 22 | Interroger le routage des appels |
| mt-forwardSM | 44 | Livrer un SMS à l'abonné |
| sendRoutingInfoForSM | 45 | Interroger le routage des SMS |
| mo-forwardSM | 46 | Transférer un SMS de l'abonné |
| sendAuthenticationInfo | 56 | Demander des vecteurs d'authentification |

---

# Types de Messages TCAP

- **BEGIN** - Démarrer la transaction
- **CONTINUE** - Mi-transaction
- **END** - Réponse finale
- **ABORT** - Annuler la transaction

---

# Addressage SCCP

## Formats de Titre Global

- **E.164** - Numéro de téléphone international (par exemple, 447712345678)
- **E.212** - Format IMSI (par exemple, 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** - Statut 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** - SCTP en cours de connexion
- **ASPUP_SENT** - En attente de l'ACK ASPUP
- **INACTIVE** - ASP actif mais non en service
- **ASPAC_SENT** - En attente de l'ACK ASPAC
- **ACTIVE** - Prêt pour le trafic

---

# Codes de Point SS7 Courants

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

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

- 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 de MTP
- **6** - Congestion du réseau
- **7** - Non qualifié
- **8** - Erreur dans le transport du 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 Passerelle CAMEL

## Vue d'ensemble

Le mode **Passerelle CAMEL (CAMELGW)** transforme OmniSS7 en une plateforme de Réseau Intelligent (IN) qui fournit des services de contrôle d'appel et de facturation en temps réel en utilisant le protocole CAMEL Application Part (CAP).

## Qu'est-ce que CAMEL ?

**CAMEL** (Applications Personnalisées pour la Logique Améliorée des Réseaux Mobiles) est un ensemble de normes conçu pour fonctionner 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 en fonction du temps/emplacement
- **Réseaux privés virtuels** - Plans de numérotation d'entreprise
- **Filtrage des appels** - Autoriser/bloquer les appels en fonction de critères

## Versions du Protocole CAP

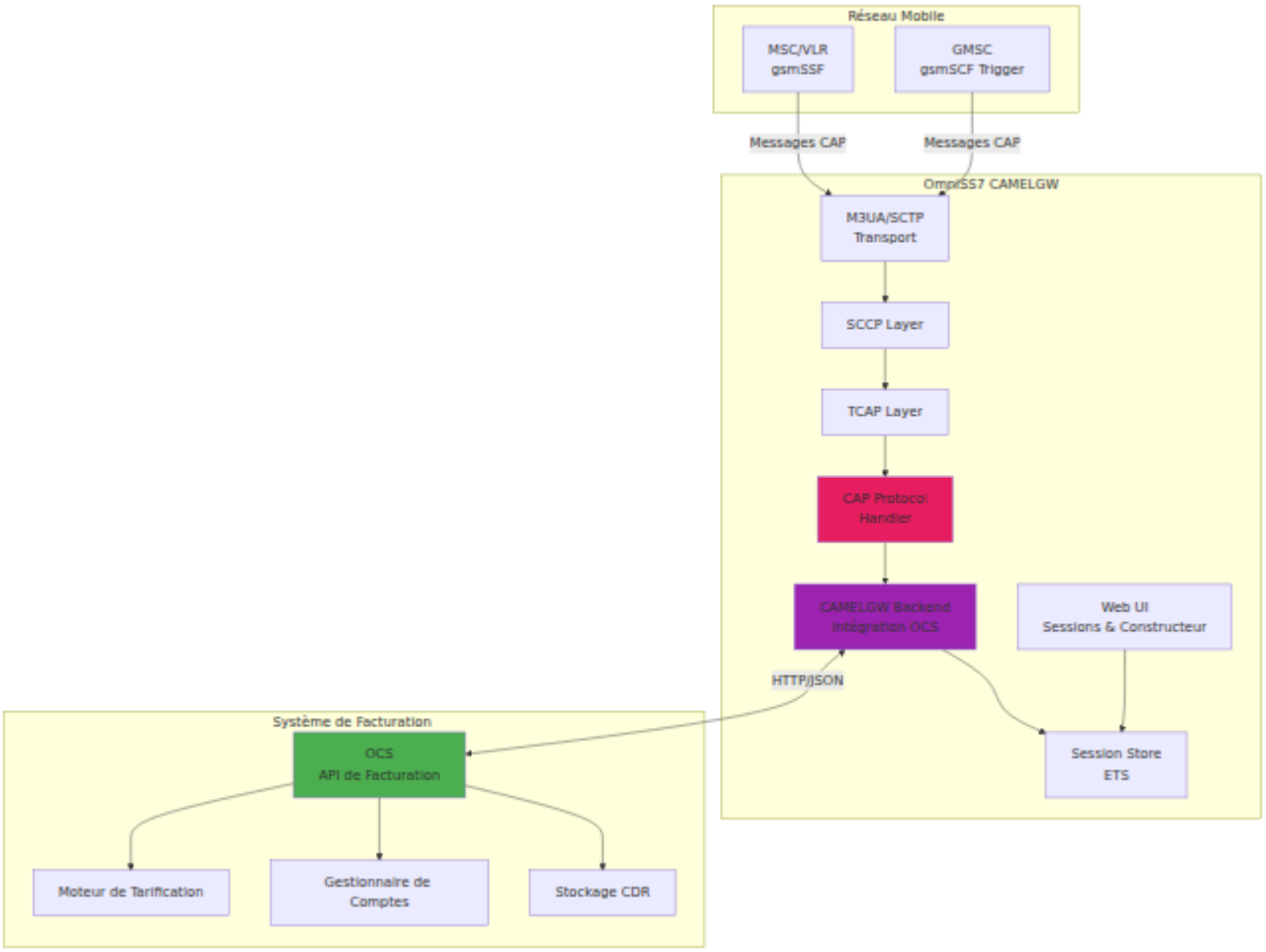
OmniSS7 CAMELGW prend en charge plusieurs versions de CAP :

| Version | Phase | Fonctionnalités |
|---|---|---|
| **CAP v1** | Phase CAMEL 1 | Contrôle d'appel de base, opérations limitées |
| **CAP v2** | Phase CAMEL 2 | Opérations améliorées, support SMS |
| **CAP v3** | Phase CAMEL 3 | Support GPRS, opérations supplémentaires |
| **CAP v4** | Phase CAMEL 4 | Fonctionnalités avancées, support multimédia |

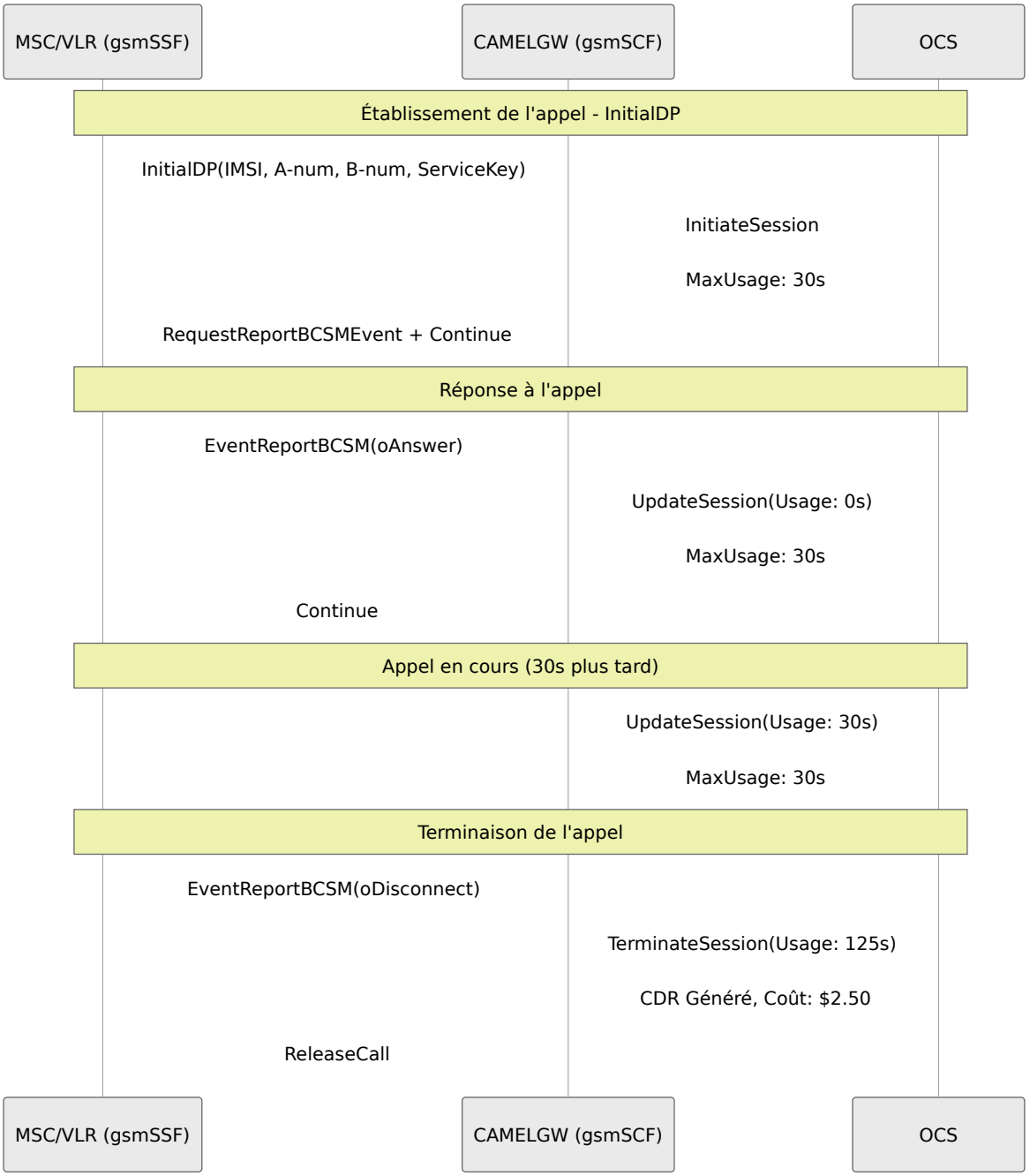
**Par défaut :** CAP v2 (le plus largement déployé)

---

# Architecture



## Exemple de Flux d'Appel




---

# Configuration

## Prérequis

- OmniSS7 installé et en cours d'exécution
- Connectivité M3UA vers MSC/GMSC (gsmSSF)

- Système de Facturation en Ligne (OCS) avec point de terminaison API (optionnel, pour la facturation en temps réel)

## Activer le Mode Passerelle CAMEL

Éditez `config/runtime.exs` et configurez la section Passerelle CAMEL :

```
config :omniss7,
  # Drapeaux de mode - Activer les fonctionnalités CAP/CAMEL
  cap_client_enabled: true,
  camelgw_mode_enabled: true,

  # Désactiver d'autres modes
  map_client_enabled: false,
  hlr_mode_enabled: false,
  smsc_mode_enabled: false,

  # Configuration de la Version CAP/CAMEL
  # Détermine quelle version CAP utiliser pour les requêtes
sortantes et le dialogue
  # Options : :v1, :v2, :v3, :v4
  cap_version: :v2,

  # Intégration OCS (pour la facturation en temps réel)
  ocs_enabled: true,
  ocs_url: "http://your-ocs-server/api/charging",
  ocs_timeout: 5000,  # millisecondes
  ocs_auth_token: "your-api-token"  # Optionnel, si OCS nécessite
une authentification

  # Configuration de la Connexion M3UA pour CAMEL
  # Se connecter en tant qu'ASP (Application Server Process) pour
les opérations CAP
  cap_client_m3ua: %{
    mode: "ASP",
    callback: {CapClient, :handle_payload, []},
    process_name: :camelgw_client_asp,

    # Point de terminaison local (système CAMELGW)
    local_ip: {10, 179, 4, 13},
    local_port: 2905,

    # Point de terminaison distant (MSC/GMSC - gsmSSF)
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,

    # Paramètres M3UA
    routing_context: 1,
    network_appearance: 0,
```

```
    asp_identifier: 13
  }
```

## Configurer les Pages de l'Interface Web

L'interface Web comprend des pages spécialisées pour les opérations CAMEL :

```
config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.TestClientLive, "/client", "Client SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
    {SS7.Web.CAMELSessionsLive, "/camel_sessions", "Sessions
CAP"},
    {SS7.Web.CAMELRequestLive, "/camel_request", "Requêtes CAP"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/camel_sessions",
               "/camel_request", "/application", "/configuration"]
```

---

# Opérations CAP Supportées

## Opérations Entrantes (de gsmSSF → gsmSCF)

| Opération | Opcode | Description | Ges |
|---|---|---|---|
| **InitialDP** | 0 | Point de Détection Initial - notification de mise en place de l'appel | `handle_initial_` |
| **EventReportBCSM** | 6 | Événement du Modèle d'État d'Appel de Base (réponse, déconnexion, etc.) | `handle_event_rep` |
| **ApplyChargingReport** | 71 | Rapport de facturation de gsmSSF | `handle_apply_cha` |
| **AssistRequestInstructions** | 16 | Demande d'assistance de gsmSRF | `handle_assist_re` |

## Opérations Sortantes (de gsmSCF → gsmSSF)

| Opération | Opcode | Description | |
|---|---|---|---|
| **Connect** | 20 | Connecter l'appel au numéro de destination | `CapRequestGen` |
| **Continue** | 31 | Continuer le traitement de l'appel sans modification | `CapRequestGen` |
| **ReleaseCall** | 22 | Libérer/terminer l'appel | `CapRequestGen` |
| **RequestReportBCSMEvent** | 23 | Demander la notification des événements d'appel | `CapRequestGen` |
| **ApplyCharging** | 35 | Appliquer la facturation à l'appel | `CapRequestGen` |

---

# Fonctionnalités de l'Interface Web

## Page des Sessions CAMEL

**URL :** `http://localhost/camel_sessions`

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

**Fonctionnalités :**

- **Liste des sessions en direct** - Se rafraîchit automatiquement toutes les 2 secondes
- **Détails de la session** - OTID, ID d'appel, État, Durée
- **Version CAP** - Affiche la version du protocole (CAP v1/v2/v3/v4) détectée à partir de l'InitialDP
- **Informations sur l'appel** - IMSI, Numéro A, Numéro B, Clé de Service
- **Suivi de l'état** - Initié, Répondu, Terminé
- **Minuteur de durée** - Affichage en temps réel de la durée de l'appel

**Colonnes du tableau :**

- ID d'appel, État, Version, IMSI, Numéro appelant, Numéro appelé, Clé de Service, Durée, Heure de début, OTID

**États de session :**

- □ **Initié** - InitialDP reçu, en attente de réponse
- □ **Répondu** - Appel répondu, facturation en cours
- □ **Terminé** - Appel terminé, CDR généré

**Détection de la Version CAP :** Le système détecte automatiquement la version du protocole CAP à partir de la partie dialogue de l'InitialDP et l'affiche dans la colonne Version. Cela aide à identifier quelle version CAP chaque MSC utilise.

# Constructeur de Requêtes CAMEL

**URL :** `http://localhost/camel_request`

Outil interactif pour construire et envoyer des requêtes CAP :

**Fonctionnalités :**

- **Sélecteur de type de requête** - InitialDP, Connect, ReleaseCall, etc.
- **Champs de formulaire dynamiques** - S'adapte au type de requête sélectionné
- **Options SCCP/M3UA** - Configuration d'adressage avancée
- **Historique des requêtes** - Dernières 20 requêtes avec statut

- **Suivi de session** - Maintient l'OTID pour les requêtes de suivi
- **Retour d'information en temps réel** - Messages de succès/erreur

**Types de Requêtes :**

1. **InitialDP** - Démarrer une nouvelle session d'appel
    - Clé de Service (entier)
    - Numéro appelant (A-partie)
    - Numéro appelé (B-partie)
2. **Connect** - Acheminer l'appel vers la destination
    - Numéro de destination
3. **ReleaseCall** - Terminer l'appel
    - Code de Cause (16=Normal, 17=Occupé, 31=Non spécifié)
4. **RequestReportBCSMEvent** - Demander des notifications d'événements
    - Événements : oAnswer, oDisconnect, tAnswer, tDisconnect
5. **Continue** - Continuer l'appel sans modification
    - Aucun paramètre requis
6. **ApplyCharging** - Appliquer des limites de durée d'appel
    - Durée (secondes, 1-864000)
    - Libérer à l'expiration (booléen)
    - Voir Guide du Constructeur de Requêtes CAMEL pour une utilisation détaillée

**Options SCCP Avancées :**

- Titre Global de la Partie Appelée
- Titre Global de la Partie Appelante
- SSN Appelée (par défaut : 146 = gsmSSF)
- SSN Appelante (par défaut : 146)

**Options M3UA :**

- OPC (Code de Point d'Origine, par défaut : 5013)
- DPC (Code de Point de Destination, par défaut : 5011)

---

# Intégration avec OCS

## Cycle de Vie de l'Appel avec Facturation

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

Lorsque le MSC envoie l'InitialDP, CAMELGW :

1. **Détecte la version CAP** - Examine la partie dialogue pour identifier CAP v1/v2/v3/v4
2. **Décode le message CAP** - Extrait IMSI, numéros appelants/appelés
3. **Appelle l'OCS** - API `InitiateSession`
4. **Reçoit l'autorisation** - MaxUsage (par exemple, 30 secondes)
5. **Stocke la session** - Dans SessionStore (table ETS) avec la version CAP
6. **Répond au MSC** - RequestReportBCSMEvent + Continue (en utilisant la même version CAP)

**Exemple :**

```
# Données InitialDP décodées
%{
  imsi: "310150123456789",
  calling_party_number: "14155551234",
  called_party_number: "14155556789",
  service_key: 1,
  msc_address: "19216800123",
  cap_version: :v2  # Détecté à partir du dialogue
}

# Réponse OCS
{:ok, %{max_usage: 30}}  # 30 secondes autorisées

# Entrée SessionStore
%{
  call_id: "CAMEL-4B000173",
  initial_dp_data: %{...},
  cap_version: :v2,  # Stocké pour la génération de réponse
  start_time: 1730246400,
  state: :initiated
}
```

**2. Réponse à l'Appel (EventReportBCSM - oAnswer)**

Lorsque l'appel est répondu :

1. **Reçoit l'événement oAnswer** - Du MSC
2. **Met à jour l'OCS** - `UpdateSession` avec usage=0
3. **Démarre la boucle de débit** - L'OCS commence à facturer
4. **Met à jour l'état de la session** - `:answered` dans SessionStore
5. **Continue l'appel** - Envoie Continue au MSC

**3. Mises à Jour Périodiques (Optionnel)**

Pour les longs appels, demander un crédit supplémentaire :

```
# Toutes les 30 secondes
OCS.Client.update_session(call_id, %{}, current_usage)
```

Si MaxUsage retourne 0, l'abonné n'a pas de crédit → Envoyer ReleaseCall

**4. Terminaison de l'Appel (EventReportBCSM - oDisconnect)**

Lorsque l'appel se termine :

1. **Reçoit l'événement oDisconnect** - Du MSC
2. **Calcule la durée totale** - À partir de l'heure de début de la session
3. **Termine la session OCS** - API `TerminateSession`
4. **CDR généré** - Par l'OCS avec le coût final
5. **Nettoie la session** - Supprime de SessionStore
6. **Envoie ReleaseCall** - Confirme la terminaison au MSC

# Analyse des CDR

Les CDR sont générés par votre OCS et comprennent généralement :

**Champs CDR de CAMEL :**

- `Account` - IMSI ou numéro appelant
- `Destination` - Numéro de la partie appelée
- `OriginID` - Identifiant d'appel unique (CAMEL-OTID)
- `Usage` - Durée totale de l'appel (secondes)
- `Cost` - Coût calculé
- `IMSI` - IMSI de l'abonné
- `CallingPartyNumber` - A-partie
- `CalledPartyNumber` - B-partie
- `MSCAddress` - Code de point MSC servant
- `ServiceKey` - Clé de service CAMEL

---

# Tests

## Tests Manuels avec le Constructeur de Requêtes

1. **Naviguer vers le Constructeur de Requêtes :**

   ```
   http://localhost/camel_request
   ```

2. **Envoyer InitialDP :**

   - Sélectionner "InitialDP" dans le menu déroulant
   - Clé de Service : `100`
   - Numéro Appelant : `14155551234`
   - Numéro Appelé : `14155556789`
   - Cliquer sur "Envoyer la Requête InitialDP"
   - Noter l'OTID généré

3. **Surveiller la Session :**

   - Ouvrir un nouvel onglet : `http://localhost/camel_sessions`
   - Voir la session active avec l'état "Initié"

4. **Simuler la Réponse à l'Appel :**

   - Retourner au Constructeur de Requêtes
   - Sélectionner "EventReportBCSM"
   - Type d'Événement : `oAnswer`
   - Cliquer sur "Envoyer la Requête EventReportBCSM"
   - L'état de la session change en "Répondu"

5. **Terminer l'Appel :**

   - Sélectionner "ReleaseCall"
   - Code de Cause : `16` (Normal)
   - Cliquer sur "Envoyer la Requête ReleaseCall"

- L'état de la session change en "Terminé"

# Tests avec un MSC Réel

### Configurer le Service CAMEL sur le MSC

Sur votre MSC/VLR, configurez le service CAMEL :

```
# Exemple de configuration 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 Logs

Regardez les logs de CAMELGW pour les messages CAP entrants :

```
# Voir les logs en temps réel
tail -f /var/log/omniss7/omniss7.log

# Filtrer pour les événements CAP
grep "CAP:" /var/log/omniss7/omniss7.log

# Voir le journal des événements (format JSON)
curl http://localhost/api/events | jq '.[] | select(.map_event |
startswith("CAP:"))'
```

# Tests de Charge

Utilisez le Constructeur de Requêtes dans une boucle pour des tests de charge :

```
# Envoyer 100 requêtes InitialDP
for i in {1..100}; do
  curl -X POST http://localhost/api/camel/initial_dp \
    -H "Content-Type: application/json" \
    -d '{
      "service_key": 100,
      "calling_number": "1415555'$i'",
      "called_number": "14155556789"
    }'
  sleep 0.1
done
```

---

# Surveillance & Opérations

## Métriques Prometheus

CAMELGW expose des métriques à `http://localhost:8080/metrics` :

**Métriques spécifiques à CAP :**

- `cap_requests_total{operation}` - Total des requêtes CAP par type d'opération (par exemple, initialDP, requestReportBCSMEvent)

**Métriques MAP/API supplémentaires :**

- `map_requests_total{operation}` - Total des requêtes MAP par type d'opération
- `map_request_duration_milliseconds{operation}` - Histogramme de la durée des requêtes
- `map_pending_requests` - Nombre de transactions MAP en attente

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

- `m3ua_stp_messages_received_total{peer_name,point_code}` - Messages reçus des pairs

- `m3ua_stp_messages_sent_total{peer_name,point_code}` - Messages envoyés aux pairs
- `m3ua_stp_routing_failures_total{reason}` - Échecs de routage par raison

**Exemples de requêtes :**

```
# Requêtes CAP
curl http://localhost:8080/metrics | grep cap_requests_total

# Total des InitialDP reçus
curl http://localhost:8080/metrics | grep
'cap_requests_total{operation="initialDP"}'

# Requêtes MAP en attente
curl http://localhost:8080/metrics | grep map_pending_requests
```

## Vérifications de Santé

```
# Vérifier la connectivité M3UA
curl http://localhost/api/m3ua-status

# Vérifier la connectivité OCS
curl http://localhost/api/ocs-status

# Vérifier les sessions actives
curl http://localhost/api/camel/sessions/count
```

## Configuration des Logs

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

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

# Activer le logging de débogage CAP
config :logger, :console,
  metadata: [:cap_operation, :otid, :call_id]
```

---

# Dépannage

## Problème : Aucun message CAP reçu

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

**Vérifiez :**

1. État du lien M3UA : `curl http://localhost/api/m3ua-status`
2. Configuration du service CAMEL sur le MSC (Clé de Service, adresse gsmSCF)
3. Routage SCCP (Le Titre Global doit router vers CAMELGW)
4. Règles de pare-feu (autoriser le port SCTP 2905)

**Solution :**

```
# Vérifier la connectivité M3UA
tcpdump -i eth0 sctp

# Vérifier si le MSC peut atteindre CAMELGW
ss -tuln | grep 2905
```

## Problème : Erreurs OCS

**Symptômes :** `INSUFFICIENT_CREDIT` ou erreurs de délai d'attente

**Vérifiez :**

1. OCS est accessible : `curl http://your-ocs-server/api/health`
2. Le compte a un solde dans l'OCS
3. Plan de tarification configuré dans l'OCS
4. Connectivité réseau vers l'OCS
5. Le jeton d'authentification est valide (si requis)

**Solution :**

- Vérifiez la configuration de l'URL OCS dans `runtime.exs`
- Vérifiez les logs de l'OCS pour des erreurs
- Testez manuellement l'API OCS avec curl
- Vérifiez que les règles de pare-feu permettent la connectivité

# Problème : Session non trouvée

**Symptômes :** EventReportBCSM échoue avec "Session non trouvée"

**Cause :** Mismatch d'OTID ou session expirée

**Solution :**

1. Vérifiez l'OTID dans les logs
2. Vérifiez le délai d'expiration de la session (par défaut : pas d'expiration)
3. Assurez-vous que le DTID correspond à l'OTID dans les messages Continue/End

```
# Vérifiez les sessions actives
iex> CAMELGW.SessionStore.list_sessions()
```

# Problème : Erreurs de décodage

**Symptômes :** `Échec du décodage de l'InitialDP` dans les logs

**Cause :** Mismatch de version CAP ou message mal formé

**Solution :**

1. Vérifiez que la configuration de la version CAP correspond à celle du MSC
2. Vérifiez que l'encodage ASN.1 est correct
3. Capturez le PCAP et analysez-le avec Wireshark

```
# Capturez les messages CAP
tcpdump -i eth0 -w cap_trace.pcap sctp port 2905

# Analysez avec Wireshark (filtre : m3ua)
wireshark cap_trace.pcap
```

---

# Configuration Avancée

## Plusieurs Versions CAP

Supportez différentes versions CAP par clé de service :

```
config :omniss7,
  cap_version_map: %{
    100 => :v2,  # La clé de service 100 utilise CAP v2
    200 => :v3,  # La clé de service 200 utilise CAP v3
    300 => :v4   # La clé de service 300 utilise CAP v4
  },
  cap_version: :v2  # Par défaut
```

---

# Résumé

Le mode Passerelle CAMEL permet à OmniSS7 de fonctionner comme une plateforme complète de Réseau Intelligent avec :

 **Support complet du protocole CAP** (v1/v2/v3/v4)
 **Facturation en temps réel** via l'intégration OCS
 **Opérations de contrôle d'appel** (Connect, Release, Continue)
 **Gestion des sessions** avec stockage ETS

 **Tests interactifs** via le Constructeur de Requêtes de l'Interface Web
 **Surveillance en direct** des sessions d'appel actives
 **Génération de CDR** pour la facturation et l'analyse
 **Performance et fiabilité prêtes pour la production**

Pour des informations supplémentaires :

- Documentation du Constructeur de Requêtes CAMEL
- Référence Technique - Opérations CAP

---

**Produit :** Passerelle CAMEL OmniSS7
**Version de la Documentation :** 1.0
**Dernière Mise à Jour :** 2025-10-26

# CAMEL Request Builder - Résumé de l'Implémentation

## Vue d'ensemble

Un nouveau composant LiveView a été créé pour construire et envoyer des requêtes CAMEL/CAP à des fins de test. Cela fournit une interface utilisateur interactive pour créer des opérations InitialDP et d'autres opérations CAMEL.

# Nouveaux Composants

## 1. CAMEL Request Builder LiveView

**Fonctionnalités :**

- Interface utilisateur interactive basée sur un formulaire pour construire des requêtes CAMEL
- Support pour plusieurs types de requêtes :
  - **InitialDP** - Point de Détection Initiale (notification de configuration d'appel)
  - **Connect** - Connecter l'appel à la destination
  - **ReleaseCall** - Libérer/terminer l'appel
  - **RequestReportBCSMEvent** - Demander des notifications d'événements
  - **Continue** - Continuer le traitement de l'appel
  - **ApplyCharging** - Appliquer des limites de facturation/durée aux appels

**Capacités Clés :**

- Menu déroulant de sélection du type de requête
- Champs de formulaire dynamiques en fonction du type de requête sélectionné
- Options avancées SCCP/M3UA (section réductible)
  - Titres Globaux de la Partie Appelée/Appelante
  - Configuration du SSN (Numéro de Sous-système)
  - Paramètres OPC/DPC (Code de Point)
- Historique des requêtes en temps réel (dernières 20 requêtes)
- Suivi de session via OTID
- Retour d'information sur le succès/l'erreur
- Suivi de la taille de la requête

**Route :** `/camel_request`

# 2. Journal d'Événements Amélioré avec Support CAMEL

**Nouvelles Fonctions :**

- `paklog_camel/2` - Journalisation dédiée des messages CAMEL/CAP
- `lookup_cap_opcode_name/1` - Recherche de code d'opération CAP
- `find_cap_opcode/1` - Extraire l'opcode CAP à partir de JSON
- `extract_cap_tids/1` - Extraire OTID/DTID des messages CAP
- `format_cap_to_json/1` - Convertir les PDU CAP en format JSON

**Codes d'Opération CAP Supportés :**

```
0  => "initialDP"
5  => "connect"
6  => "releaseCall"
7  => "requestReportBCSMEvent"
8  => "eventReportBCSM"
10 => "continue"
13 => "furnishChargingInformation"
35 => "applyCharging"
... (47 opérations au total)
```

**Fonctionnalités :**

- Journalisation JSON de toutes les requêtes/réponses CAMEL
- Détection automatique des actions TCAP (Début/Continuer/Fin/Abandon)
- Extraction d'adressage SCCP
- Gestion des erreurs pour les messages mal formés
- Traitement des tâches en arrière-plan (non-bloquant)
- Événement préfixé par "CAP :" pour un filtrage facile

# 3. CapClient Mis à Jour

**Modifications :**

- Ajout d'appels `paklog_camel/2` pour les messages entrants et sortants

- Journalisation double : MAP (`paklog`) et CAP (`paklog_camel`) pour compatibilité
- Messages sortants journalisés dans `sccp_m3ua_maker/2`
- Messages entrants journalisés dans `handle_payload/1`

# Configuration

Les nouvelles pages LiveView ont été ajoutées à la configuration d'exécution :

```
# Fichier : config/runtime.exs

config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.TestClientLive, "/client", "Client SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
    {SS7.Web.HlrLinksLive, "/hlr_links", "Liens HLR"},
    {SS7.Web.CAMELSessionsLive, "/camel_sessions", "Sessions
CAMEL"},
    {SS7.Web.CAMELRequestLive, "/camel_request", "Constructeur de
Requêtes CAMEL"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/hlr_links",
               "/camel_sessions", "/camel_request",
               "/application", "/configuration"]
```

# Utilisation

## Accéder au Constructeur de Requêtes

1. Naviguez vers : `https://your-server:8087/camel_request`
2. Sélectionnez le type de requête dans le menu déroulant
3. Remplissez les paramètres requis
4. Optionnellement, développez "Options Avancées SCCP/M3UA" pour un réglage fin
5. Cliquez sur "Envoyer [RequestType] Requête"

# Flux de Requête

### InitialDP (Nouvel Appel)

1. Définir la Clé de Service (par exemple, 100)
2. Définir le Numéro Appelant (A-Party)
3. Définir le Numéro Appelé (B-Party)
4. Envoyer la requête → Génère un nouvel OTID
5. OTID stocké dans la session pour les requêtes de suivi

### Requêtes de Suivi (Connect, ReleaseCall, etc.)

1. Doit avoir un OTID actif de l'InitialDP
2. La requête utilise automatiquement l'OTID stocké
3. Avertissement affiché si aucun OTID actif

# Paramètres de Requête

**InitialDP :**

- Clé de Service (entier)
- Numéro Appelant (format ISDN)
- Numéro Appelé (format ISDN)

**Connect :**

- Numéro de Destination (où router l'appel)

**ReleaseCall :**

- Code de Cause (16 = Normal, 17 = Occupé, 31 = Non spécifié)

**RequestReportBCSMEvent :**

- Événements BCSM (séparés par des virgules : oAnswer, oDisconnect, etc.)

**Continue :**

- Aucun paramètre (utilise l'OTID actif)

**ApplyCharging :**

- Durée (secondes, 1-864000) - Durée maximale de l'appel avant action
- Libérer en cas de délai (booléen) - Si l'appel doit être libéré lorsque la durée expire

## Options Avancées

**Adressage SCCP :**

- GT de la Partie Appelée (Titre Global)
- GT de la Partie Appelante
- SSN Appelé (par défaut 146 = gsmSSF)
- SSN Appelant (par défaut 146)

**Codes de Point M3UA :**

- OPC (Code de Point d'Origine, par défaut 5013)
- DPC (Code de Point de Destination, par défaut 5011)

# Journalisation JSON

Tous les messages CAMEL sont maintenant journalisés au format JSON dans le journal des événements avec :

- **Direction :** entrant/sortant
- **Action TCAP :** Début/Continuer/Fin/Abandon
- **Opération CAP :** par exemple, "CAP:initialDP", "CAP:connect"
- **Adressage SCCP :** Informations sur la Partie Appelée/Appelante
- **TIDs :** OTID/DTID pour corrélation
- **Message Complet :** PDU CAP encodé en JSON

### Exemple d'Entrée de Journal


```
{
  "map_event": "CAP:initialDP",
  "direction": "outgoing",
  "tcap_action": "Begin",
  "otid": "A1B2C3D4",
  "sccp_called": {
    "SSN": 146,
    "GlobalTitle": {
      "Digits": "55512341234",
      "NumberingPlan": "isdn_tele",
      "NatureOfAddress_Indicator": "international"
    }
  },
  "event_message": "{ ... full CAP PDU ... }"
}
```


# Historique des Requêtes

L'interface utilisateur affiche les 20 dernières requêtes avec :

- Horodatage
- Type de requête (avec badge codé par couleur)
- OTID (premiers 8 caractères hex)
- Statut (envoyé/erreur)
- Taille du message en octets

# Suivi de Session

**Panneau d'Informations de Session Actuelle :**

- Affiche l'OTID actif
- Montre la taille en octets de la dernière requête
- Visible uniquement lorsque la session est active

# Flux de Travail de Test

1. **Démarrer un Nouvel Appel :**
    - Envoyer InitialDP → Obtenir OTID
    - Le système crée une session
2. **Contrôler l'Appel :**
    - Envoyer RequestReportBCSMEvent → Demander des notifications
    - Envoyer ApplyCharging → Définir la limite de durée de l'appel (par exemple, 290 secondes)
    - Envoyer Connect → Router à la destination
    - OU Envoyer ReleaseCall → Terminer
3. **Voir les Résultats :**
    - Vérifier l'historique des requêtes
    - Surveiller la page des Sessions CAMEL
    - Examiner les journaux d'événements avec le préfixe "CAP :"

# ApplyCharging - Contrôle de Durée d'Appel

## Vue d'ensemble

L'opération ApplyCharging vous permet de définir une durée maximale d'appel et, en option, de libérer l'appel lorsque cette durée expire. Cela est généralement utilisé pour des scénarios de facturation prépay��e ou pour imposer des limites de temps sur les appels.

## Cas d'Utilisation

- **Facturation Prépayée** : Limiter la durée de l'appel en fonction du solde de l'abonné

- **Facturation Basée sur le Temps** : Imposer des intervalles de facturation périodiques
- **Gestion des Ressources** : Empêcher les appels de durer indéfiniment
- **Intégration OCS** : Coordonner avec les Systèmes de Facturation en Ligne pour un contrôle de crédit en temps réel

# Paramètres

### Durée (maxCallPeriodDuration)

- **Type** : Entier (1-864000 secondes)
- **Description** : Nombre maximum de secondes que l'appel peut durer avant que le minuteur expire
- **Exemples** :
  - `60` = 1 minute
  - `290` = 4 minutes 50 secondes (valeur de test courante)
  - `3600` = 1 heure
  - `86400` = 24 heures

### Libérer en cas de délai (releaseIfDurationExceeded)

- **Type** : Booléen (vrai/faux)
- **Par défaut** : vrai
- **Description** : Que se passe-t-il lorsque la durée expire :
  - `vrai` : Libérer/déconnecter automatiquement l'appel
  - `faux` : Envoyer une notification mais garder l'appel actif (permet à gsmSCF d'agir)

# Structure du Message

Le message ApplyCharging est encodé en tant que TCAP Continue avec :

- **TCAP** : Message Continue (utilise la transaction existante)
- **Opcode** : 35 (applyCharging)
- **Paramètres** : ApplyChargingArg contenant :

- `aChBillingChargingCharacteristics` : Informations de facturation basées sur le temps
  - `timeDurationCharging` : Durée maximale et indicateur de libération
- `partyToCharge` : Quelle partie est facturée (par défaut : sendingSideID)

# Exemple d'Utilisation

**Scénario** : Appel prépayé avec une limite de 5 minutes

1. Envoyer **InitialDP** pour commencer la surveillance de l'appel

```
Clé de Service : 100
Appelant : 447700900123
Appelé : 447700900456
→ OTID : A1B2C3D4
```

2. Envoyer **ApplyCharging** pour définir la limite de 5 minutes

```
Durée : 300 (secondes)
Libérer en cas de délai : vrai
→ Utilise OTID : A1B2C3D4
```

3. Envoyer **Connect** pour compléter l'appel

```
Destination : 447700900456
→ Utilise OTID : A1B2C3D4
```

4. Après 5 minutes (300 secondes) :
   - L'appel est automatiquement libéré par le réseau
   - gsmSCF reçoit une notification de déconnexion

# Meilleures Pratiques

1. **Envoyer toujours ApplyCharging AVANT Connect**

    - Assure que la facturation est active lorsque l'appel se connecte
    - Empêche les segments d'appel non facturés

2. **Utiliser avec RequestReportBCSMEvent**

    - Demander les événements `oAnswer` et `oDisconnect`
    - Permet de suivre la durée réelle de l'appel
    - Permet de réappliquer la facturation si nécessaire

3. **Définir des durées raisonnables**

    - Trop court : Opérations de facturation fréquentes, mauvaise expérience utilisateur
    - Trop long : Risque de perte de revenus sur les appels prépayés
    - Typique : 60-300 secondes pour prépayé, plus long pour postpayé

4. **Gérer le délai avec grâce**

    - Si `release=false`, être prêt à gérer les notifications d'expiration du minuteur
    - Implémenter une logique pour prolonger la durée ou libérer l'appel

# Gestion des Erreurs

Problèmes courants :

- **Pas d'OTID actif** : Doit envoyer InitialDP d'abord
- **Durée invalide** : Doit être de 1 à 864000 secondes
- **Support réseau** : Certaines implémentations SSF peuvent ne pas supporter ApplyCharging
- **Précision du minuteur** : La résolution du minuteur réseau est généralement de 1 seconde, mais peut varier

# Surveillance

Suivre les opérations ApplyCharging via :

- **Historique des Requêtes** : Montre les requêtes ApplyCharging envoyées

- **Journal des Événements** : Rechercher "CAP:applyCharging"
- **Sessions CAMEL** : Surveiller les sessions actives avec facturation appliquée
- **Trace TCAP** : Déboguer les problèmes d'encodage/décodage

# Détails de l'Implémentation

## Gestion d'État

- LiveView assigne l'état du formulaire de suivi
- OTID stocké dans les assignations de socket
- Historique des requêtes limité à 20 entrées
- Auto-rafraîchissement désactivé (envoi manuel uniquement)

## Génération de Requête

- Utilise le module existant `CapRequestGenerator`
- Construit des structures TCAP/CAP appropriées
- Encode avec le codec `:TCAPMessages`
- Enveloppe dans SCCP via `CapClient.sccp_m3ua_maker/2`

## Mécanisme d'Envoi

- Envoie via M3UA à `:camelgw_client_asp`
- Utilise le contexte de routage 1
- Encapsulation automatique SCCP/M3UA

## Gestion des Erreurs

- Validation du formulaire avec retour d'information utilisateur
- Gestion gracieuse des OTID manquants
- Erreurs de parsing affichées dans l'UI
- Échecs d'encodage journalisés

# Améliorations Futures

Ajouts potentiels :

1. Modèles/presets de requêtes
2. Corrélation et affichage des réponses
3. Visualisation du flux d'appel
4. Détail des sessions en profondeur
5. Exporter l'historique des requêtes
6. Tests de charge (requêtes en masse)
7. Export PCAP des messages générés
8. Validation des paramètres CAP

# Notes d'Intégration

- Compatible avec la journalisation MAP existante (`paklog`)
- Partage la base de données de journal des événements avec les événements MAP
- Utilise la même infrastructure SCCP/M3UA
- Fonctionne avec CAMELSessionsLive pour la surveillance
- S'intègre avec le routage M3UA existant

# Fichiers Modifiés

- `config/runtime.exs` - MIS À JOUR

# Dépendances

- Générateur de Requêtes CapRequestGenerator existant
- CapClient pour l'envoi M3UA
- M3UA.Server pour la transmission de paquets
- EventLog pour la journalisation des messages

- Cadre Phoenix LiveView
- Panneau de Contrôle pour l'infrastructure de l'UI

# 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



---

## Aperçu de l'Interface Web

L'interface Web est accessible via l'adresse de votre serveur web configuré.

## Navigation Principale

- **Événements** - Événements de signalisation SS7 en temps réel et journaux de messages
- **Application** - État de l'application et informations d'exécution
- **Configuration** - Visualiseur de configuration système
- **État M3UA** - Connexions de pairs M3UA (mode STP)
- **File d'attente SMS** - Messages SMS sortants (mode SMSc)

## Accéder à l'Interface Web

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

## Documentation de l'API Swagger

Documentation interactive de l'API :

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

## Configuration de l'Interface Web

Configurer dans `config/runtime.exs` :

```
config :control_panel,
  # Ordre des pages dans le menu de navigation
  page_order: ["/events", "/application", "/configuration"],

  # Paramètres du serveur web
  web: %{
    listen_ip: "0.0.0.0",    # IP à lier (0.0.0.0 pour toutes les
interfaces)
    port: 80,                 # Port HTTP (443 pour HTTPS)
    hostname: "localhost",    # Nom d'hôte du serveur pour la
génération d'URL
    enable_tls: false,        # Définir à true pour activer HTTPS
    tls_cert: "cert.pem",    # Chemin vers le fichier de
certificat TLS
    tls_key: "key.pem"       # Chemin vers le fichier de clé
privée TLS
  }
```

**Paramètres de Configuration :**

| Paramètre | Type | Par Défaut | Description |
|---|---|---|---|
| `page_order` | Liste | `["/events", "/application", "/configuration"]` | Ordre des pages dans le menu de navigation |
| `listen_ip` | Chaîne | `"0.0.0.0"` | Adresse IP à lier au serveur web |
| `port` | Entier | `80` | Port HTTP (utiliser 443 pour HTTPS) |
| `hostname` | Chaîne | `"localhost"` | Nom d'hôte du serveur pour la génération d'URL |
| `enable_tls` | Booléen | `false` | Activer HTTPS avec TLS |
| `tls_cert` | Chaîne | `"cert.pem"` | Chemin vers le certificat TLS (lorsque TLS activé) |
| `tls_key` | Chaîne | `"key.pem"` | Chemin vers la clé privée TLS (lorsque TLS activé) |

## Configuration du Logger

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

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

**Niveaux de Journalisation :**

- `:debug` - Informations de débogage détaillées
- `:info` - Messages d'information généraux
- `:warning` - Messages d'avertissement pour des problèmes potentiels
- `:error` - Messages d'erreur uniquement

---

# Documentation de l'API

## URL de Base de l'API

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

## Codes de Réponse

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

## Spécification OpenAPI

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

---

# Surveillance et Métriques

## Point de Terminaison des Métriques Prometheus

```
http://your-server/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 des requêtes
- 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 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 débit
    - Utiliser des clés API ou OAuth
    - Journaliser toutes les requêtes pour audit

## Optimisation des Performances

1. **Limites TPS**
    - Configurer TPS approprié
    - Surveiller la charge système
    - Ajuster les tampons SCTP
2. **Optimisation de la Base de Données**

    - Ajouter des index
    - Archiver les anciens messages
    - Surveiller le pool de connexions

3. **Réglage 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 bascule automatiquement vers un chemin alternatif sans interrompre la connexion
- **Bascule sans Configuration** : Pas de logique au niveau de l'application 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 entre plusieurs chemins (dépendant de l'implémentation)

## Comment Cela Fonctionne

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

1. **Se Lie à Toutes 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

> **Note 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 de liste pour `remote_ip` n'est valide que pour `role: :client` (connexions sortantes).
>
> **Exemples :**

```
# ✓ CORRECT - Rôle serveur avec unique remote_ip
%&#123;
  role: :server,
  remote_ip: &#123;10, 0, 2, 100&#125;,  # Unique tuple
  # ...
&#125;

# ✗ FAUX - Rôle serveur avec liste ne fera PAS correspondre les
connexions entrantes
%&#123;
  role: :server,
  remote_ip: [&#123;10, 0, 2, 100&#125;, &#123;10, 0, 2,
101&#125;],  # La liste ne fonctionnera pas
  # ...
&#125;
```

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

```
# Configuration du 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 prise en charge du 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**

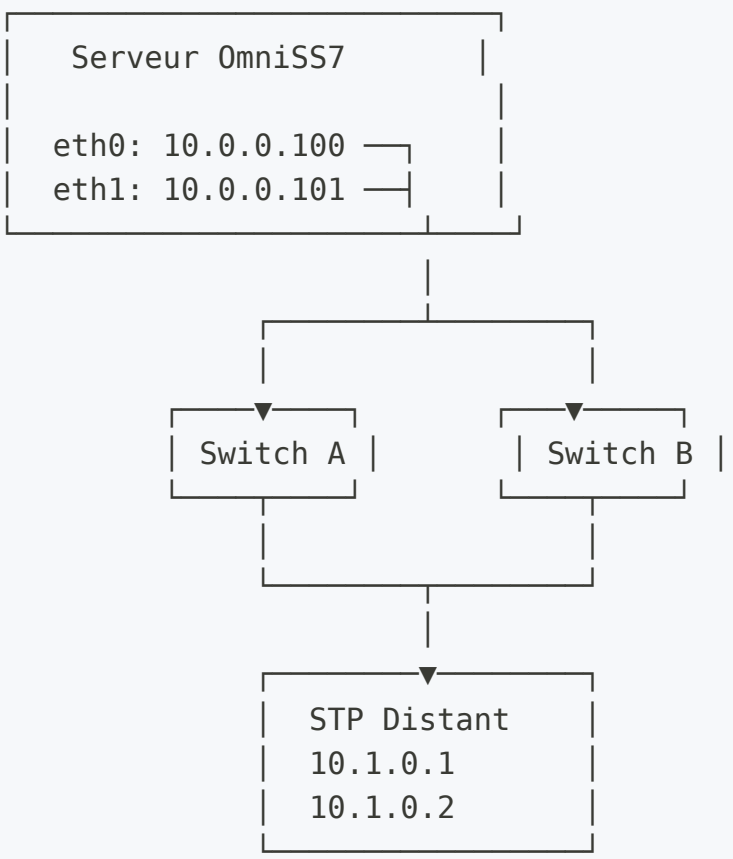
```
# Acceptation d'une connexion entrante d'un HLR multihomé
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "HLR",
      role: :server,  # ENTRANT - HLR se connecte à nous
      # Multihoming : nos IPs locales (la liste est OK)
      local_ip: [{10, 0, 1, 10}, {10, 0, 1, 11}],
      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 d'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 : Deux NICs (Déploiement Commun)



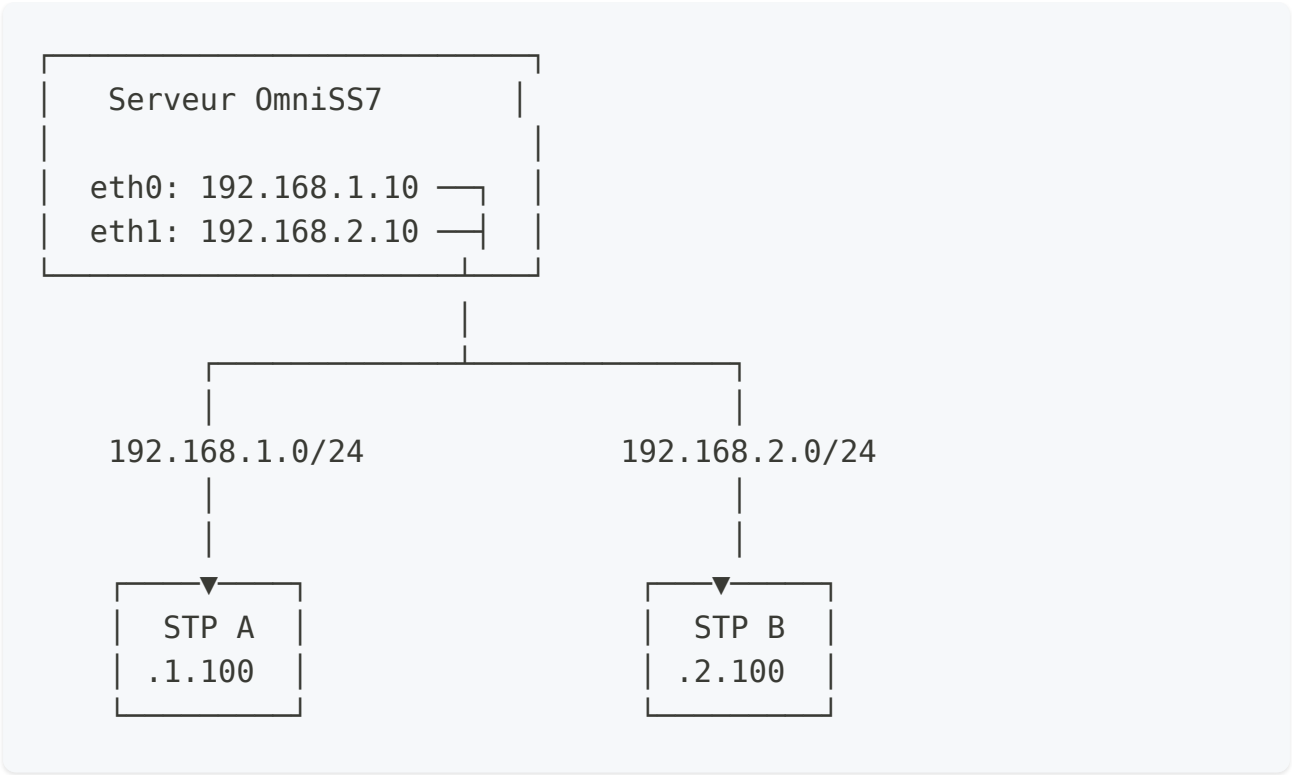

**Configuration :**

```
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 :**

- Survit à la panne d'une NIC
- Survit 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 :**

- Survit 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`
- **Plusieurs 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 des NICs différentes** pour une redondance maximale
- **Différents commutateurs** pour survivre aux pannes de commutateur
- **Différents sous-réseaux** si possible pour la diversité de routage
- **Même centre de données initialement** - tester avant séparation géographique

**2. Planification des Adresses IP**

- **La première IP est primaire** - assurez-vous qu'elle soit 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 locaux et distants utilisent plusieurs IPs
- **Acceptable** : Un seul côté utilise le multihoming
- **Remarque** : La redondance est meilleure lorsque les deux points de terminaison le 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 Basculer Pas**

**Symptômes** : Chemin principal 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 bloquant 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** : Typiquement <1 seconde de détection et de bascule
- **Récupération automatique** : Aucune intervention manuelle nécessaire

# Compatibilité

- **Rétrocompatible** : Le format de tuple d'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 support SCTP compilé

## Surveillance et Alerte

**Métriques Clés :**

- État de 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 Complète de Configuration

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

---

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

```
config :control_panel,
  page_order: ["/events", "/application", "/configuration"],
  web: %{
    listen_ip: "0.0.0.0",
    port: 80,
    hostname: "localhost",
    enable_tls: false,
    tls_cert: "cert.pem",
    tls_key: "key.pem"
  }
```

| Paramètre | Type | Requis | Par Défaut | Description |
|---|---|---|---|---|
| `page_order` | Liste de Chaînes | Non | `["/events", "/application", "/configuration"]` | Ordre des pages du menu de navigation |
| `web.listen_ip` | Chaîne | Oui | `"0.0.0.0"` | Adresse IP à lier au serveur web |
| `web.port` | Entier | Oui | `80` | Numéro de port HTTP/HTTPS |
| `web.hostname` | Chaîne | Oui | `"localhost"` | Nom d'hôte du serveur |
| `web.enable_tls` | Booléen | Non | `false` | Activer HTTPS |
| `web.tls_cert` | Chaîne | Si TLS activé | `"cert.pem"` | Chemin vers le certificat TLS |
| `web.tls_key` | Chaîne | Si TLS activé | `"key.pem"` | Chemin vers la clé privée TLS |

---

**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` | Booléen | 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` | Entier | Oui | `2905` | Port SCTP pour M3UA |
| `enable_gt_routing` | Booléen | Non | `false` | Activer le routage de titre global |

**Paramètres des Pairs M3UA :**

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `peer_id` | Entier | Oui | Identifiant unique du pair |
| `name` | Chaîne | Oui | Nom descriptif du pair |
| `role` | Atome | 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` | Entier | 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` | Entier | Si `:client` | Port du pair distant |
| `routing_context` | Entier | Oui | Contexte de routage M3UA |
| `point_code` | Entier | Oui | Code de point SS7 |
| `network_indicator` | Atome | Non | `:international` ou `:national` |

**Paramètres de Route M3UA :**

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

**Paramètres de Route GT M3UA :**

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `gt_prefix` | Chaîne | Oui | Préfixe de Titre Global à correspondre |
| `peer_id` | Entier | Oui | Pair de destination |
| `priority` | Entier | Oui | Priorité de la route |
| `description` | Chaîne | Non | Description de la route pour journalisation |
| `source_ssn` | Entier | Non | Correspondre uniquement si le SSN source correspond |
| `dest_ssn` | Entier | 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` | Booléen | Oui | `false` |
| `map_client_m3ua.mode` | Chaîne | Oui | `"ASP"` |
| `map_client_m3ua.callback` | Tuple | Oui | `{MapClient, :handle_payload, []}` |
| `map_client_m3ua.process_name` | Atome | Oui | `:map_client_asp` |
| `map_client_m3ua.local_ip` | Tuple | Oui | - |
| `map_client_m3ua.local_port` | Entier | Oui | `2905` |
| `map_client_m3ua.remote_ip` | Tuple | Oui | - |
| `map_client_m3ua.remote_port` | Entier | Oui | `2905` |
| `map_client_m3ua.routing_context` | Entier | 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` | Booléen | Non | `false` | Activer le vidage automatique de la file d'attente SMS |
| `auto_flush_interval` | Entier | Non | `10000` | Intervalle de sondage de la file d'attente (millisecondes) |
| `auto_flush_dest_smsc` | Chaîne/nil | Non | `nil` | Filtrer par SMSC de destination (nil = tous) |
| `auto_flush_tps` | Entier | 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://10.5.198.200:8443",
  frontend_name: "omni-smsc01"  # Optionnel : par défaut au nom
d'hôte_SMSc
```

**Paramètres de l'API :**

| Paramètre | Type | Requis | Par Défaut | |
|---|---|---|---|---|
| `smsc_api_base_url` | Chaîne | Oui | `"https://10.5.198.200:8443"` | |
| `frontend_name` | Chaîne | Non | `"{hostname}_SMSc"` | |

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

- `POST /api/frontends` - Enregistrer cette instance de 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 un suivi d'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



---

## Aperçu

La configuration d'OmniSS7 est gérée via `config/runtime.exs`. Le système prend en charge quatre modes opérationnels :

- **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

**Fichier de Configuration** : `config/runtime.exs`

---

# Drapeaux de Mode Opérationnel

Contrôle quelles fonctionnalités sont activées.

| Paramètre | Type | Par Défaut | Description | Modes |
|---|---|---|---|---|
| `map_client_enabled` | Boolean | `false` | Activer le client MAP et la connectivité M3UA | Tous |
| `hlr_mode_enabled` | Boolean | `false` | Activer les fonctionnalités spécifiques au HLR | HLR |
| `smsc_mode_enabled` | Boolean | `false` | Activer les fonctionnalités spécifiques au SMSc | SMSc |
| `cap_client_enabled` | Boolean | `false` | Activer le client CAP pour les opérations CAMEL | CAMEL GW |
| `camelgw_mode_enabled` | Boolean | `false` | Activer les fonctionnalités de la Passerelle CAMEL | CAMEL GW |
| `ussd_gateway_enabled` | Boolean | `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 de l'API HLR

| Paramètre | Type | Par Défaut | Requis | Desc |
|---|---|---|---|---|
| `hlr_api_base_url` | String | - | **Oui** | URL de termina l'API HL |
| `hlr_api_verify_ssl` | Boolean | `false` | Non | Activer/ la vérifi certifica pour l'A |
| `hlr_service_center_gt_address` | String | - | **Oui** | Adresse Global H retourné les répo UpdateL |
| `smsc_service_center_gt_address` | String | - | **Oui** | Adresse retourné les répo for-SM |

**Exemple** :

```
config :omniss7,
  hlr_api_base_url: "https://10.180.2.140:8443",
  hlr_api_verify_ssl: false,
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112"
```

## Configuration AlertServiceCenter

Lorsqu'un abonné effectue un UpdateLocation, le HLR envoie des messages alertServiceCenter aux GT SMSc configurés pour indiquer que l'abonné est maintenant joignable.

| Paramètre | Type | Par Défaut | Requis | De |
|---|---|---|---|---|
| `hlr_smsc_alert_gts` | Liste de Strings | `[]` | Non | Liste Globa pour notifi alertS |
| `hlr_alert_location_expiry_seconds` | Integer | `172800` | Non | Durée de la en se défau heure 0 pou l'expi |

**Exemple** :

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

## Mapping MSISDN ↔ IMSI

Configuration pour la génération IMSI synthétique à partir des 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` | String | `"50557"` | Non | Préfixe PLMN (MCC+MNC) pour la génération d'IMSI synthétique |
| `hlr_msisdn_country_code` | String | `"61"` | Non | Préfixe de code pays pour le mapping inverse IMSI→MSISDN |
| `hlr_msisdn_nsn_offset` | Integer | `0` | Non | Décalage dans le MSISDN où commence le NSN (typiquement la longueur du code pays) |
| `hlr_msisdn_nsn_length` | Integer | `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 de 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 de 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 UpdateLocation. 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` | Atom | `:packetAndCircuit` | Non | Typ<br>:<br>`:pa`<br>`:pa`<br>`:ci` |
| `isd_send_ss_data` | Boolean | `true` | Non | Env<br>ave<br>de<br>Sup |
| `isd_send_call_barring` | Boolean | `true` | Non | Env<br>ave<br>de |

**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` | Integer | `11_110` | No |
| `camel_trigger_detection_point` | Atom | `:termAttemptAuthorized` | No |
| `camel_gsmscf_gt_address` | String | (utilise le GT appelé) | No |

**Exemple** :

```
config :omniss7,
  camel_service_key: 11_110,
  camel_trigger_detection_point: :termAttemptAuthorized
```

## 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 de l'API SMSc

| Paramètre | Type | Par Défaut | Requis |
|---|---|---|---|
| `smsc_api_base_url` | String | - | **Oui** |
| `smsc_api_verify_ssl` | Boolean | `false` | Non |
| `smsc_name` | String | `"{hostname}_SMSc"` | Non |
| `smsc_service_center_gt_address` | String | - | **Oui** |

**Exemple** :

```
config :omniss7,
  smsc_api_base_url: "https://10.179.3.219:8443",
  smsc_api_verify_ssl: false,
  smsc_name: "ipsmgw",
  smsc_service_center_gt_address: "55512341112"
```

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

## Configuration Auto-Flush

| Paramètre | Type | Par Défaut | Requis | Description |
|---|---|---|---|---|
| `auto_flush_enabled` | Boolean | `true` | Non | Activer le traitemer automatique de la f d'attente SMS |
| `auto_flush_interval` | Integer | `10_000` | Non | Intervalle de traitement de la file d'attente en millisecondes |
| `auto_flush_dest_smsc` | String | - | **Oui** | Nom du SMSc de destination pour l'auto-flush |
| `auto_flush_tps` | Integer | `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` | Boolean | `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` | Integer | `2905` | Non | Port à écouter |
| `sctp_handler.point_code` | Integer | - | **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 la 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 du multihoming SCTP et ses avantages, voir Multihoming SCTP dans le Guide Commun.

## Routage de Titre Global

| Paramètre | Type | Par Défaut | Requis | Description |
|---|---|---|---|---|
| `enable_gt_routing` | Boolean | `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 à la fois 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` | Integer | - | **Oui** | Identifiant unique du p |
| `name` | String | - | **Oui** | Nom descri du pair |
| `protocol` | Atom | `:m3ua` | Non | Type de protocole : `:m3ua` ou `:` |
| `role` | Atom | `:client` | Non | Rôle de connexion : `:client`, `:server`, `:` ou `:sgp` |
| `local_ip` | Tuple ou Liste | - | **Oui** | Adresse(s) l locale(s) po binding |
| `local_port` | Integer | - | **Oui** | Port SCTP lo (M3UA : 29( M2PA : 356 |
| `remote_ip` | Tuple ou Liste | - | **Oui** | Adresse(s) l distante(s) |
| `remote_port` | Integer | - | **Oui** | Port SCTP distant |
| `routing_context` | Integer | - | Non | Identifiant c contexte de routage M3 (uniquemer M3UA) |

| Paramètre | Type | Par Défaut | Requis | Descripti |
|---|---|---|---|---|
| `point_code` | Integer | - | **Oui** | Code de po local |
| `network_indicator` | Atom | `:international` | Non | Indicateur d réseau : `:internati` ou `:nation` |
| `initiate_connection` | Boolean | `true` | Non | Indique s'il initier la connexion S |

**Paramètres Spécifiques à M2PA :**

| Paramètre | Type | Par Défaut | Requis | Description |
|---|---|---|---|---|
| `adjacent_point_code` | Integer | - | **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 de 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 Maps | `[]` | Non | Liste de 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 map avec :

- `dest_pc`: Code de point de destination (Integer)
- `peer_id`: ID du pair à travers lequel router (Integer)
- `priority`: Priorité de la route - valeur inférieure = priorité plus élevée (Integer)
- `network_indicator`: Indicateur de réseau (Atom) : `: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. Le plus long préfixe correspondant est utilisé en premier, puis la priorité.

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

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

**Paramètres de Base :**

- `gt_prefix`: Préfixe de Titre Global à faire correspondre (String) - chaîne vide correspond à tout

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

**Paramètres de Correspondance Optionnels :**

- `source_ssn`: Faire correspondre le Numéro de Sous-Système source (Integer)
- `source_tt`: Faire correspondre le Type de Traduction source (Integer)
- `source_npi`: Faire correspondre l'Indicateur de Plan de Numérotation source (Integer)
- `source_nai`: Faire correspondre l'Indicateur de Nature d'Adresse source (Integer)

**Paramètres de Transformation Optionnels :**

- `dest_ssn`: Transformer le SSN dans le message transféré (Integer)
- `dest_tt`: Transformer le Type de Traduction dans le message transféré (Integer)
- `dest_npi`: Transformer l'Indicateur de Plan de Numérotation dans le message transféré (Integer)
- `dest_nai`: Transformer l'Indicateur de Nature d'Adresse dans le message transféré (Integer)

**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 d'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 :**

```
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 de Type de Traduction
    %{
      gt_prefix: "61",
      peer_id: 3,
      priority: 1,
      description: "Numéros Australiens : transformation TT 0→1",
      source_tt: 0,  # Faire correspondre 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,  # Faire correspondre 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,   # Faire correspondre 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
```

```
    },

    # Route par défaut/De secours
    %{
      gt_prefix: "",
      peer_id: 1,
      priority: 99,
      description: "Route de secours par défaut"
    }
  ]
```

---

# Paramètres de Mode Passerelle CAMEL

Configuration pour le mode Passerelle CAMEL (protocole CAP).

## Drapeaux de 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` | Atom | `:v2` | Non | Version du protocole CAP : `:v1`, `:v2`, `:v3`, ou `:v4` |
| `camel_gsmscf_gt_address` | String | (utilise le GT appelé) | Non | Titre Global gsmSCF par défaut pour les réponses CAMEL |

**Mapping de Version CAP :**

- `:v1` → OID de Contexte d'Application : 0.4.0.0.1.0.50.0
- `:v2` → OID de Contexte d'Application : 0.4.0.0.1.0.50.1 (par défaut - le plus largement supporté)
- `:v3` → OID de Contexte d'Application : 0.4.0.0.1.21.3.4
- `:v4` → OID de 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: "68988411553"
```

# Intégration CGrateS

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

| Paramètre | Type | Par Défaut | Requis | Descrip |
|---|---|---|---|---|
| `cgrates_enabled` | Boolean | `false` | Non | Activer l'intégration CGrateS |
| `cgrates_url` | String | - | **Oui** (si activé) | URL de poir terminaison RPC de CGr |
| `cgrates_tenant` | String | `"cgrates.org"` | Non | Identifiant c locataire CC |
| `cgrates_request_type` | String | `"*prepaid"` | Non | Type de facturation `"*prepaid"` `"*postpaid` `"*pseudopr` |
| `cgrates_timeout` | Integer | `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 M3UA CAP

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

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

**Exemple :**

```
config :omniss7,
  cap_client_m3ua: %{
    mode: "ASP",
    callback: {CapClient, :handle_payload, []},
    process_name: :camelgw_client_asp,
    local_ip: {10, 5, 198, 200},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 4,
    opc: 5013,    # Code de Point d'Origine
    dpc: 5011     # Code de Point de Destination
  }
```

---

# Paramètres NAT de Titre Global

La Traduction d'Adresse Réseau de Titre Global permet des GT de réponse différents 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 NAT de Titre Global.

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

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

- `calling_prefix`: Préfixe de chaîne à faire correspondre contre le GT appelant (optionnel)
- `called_prefix`: Préfixe de chaîne à faire correspondre contre 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 préfixes `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 : Faire correspondre à la fois l'appelant de
"8772" ET le appelé à "555"
    %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

    # Priorité moyenne : Faire correspondre uniquement l'appelant
de "8772"
    %{calling_prefix: "8772", weight: 10, response_gt:
"68988411553"},

    # Priorité moyenne : Faire correspondre uniquement le appelé à
"555"
    %{called_prefix: "555", weight: 10, response_gt:
"68988411554"},

    # Faire correspondre uniquement l'appelant de "8773"
    %{calling_prefix: "8773", weight: 10, response_gt:
"68988411554"},

    # Règle de secours générique (correspond à tout, poids le plus
élevé)
    %{weight: 100, response_gt: "68988411555"}
  ]
```

**Voir Aussi** : Guide 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 Client MAP.

| Paramètre | Type | Par Défaut | Requis |
|---|---|---|---|
| `map_client_m3ua.mode` | String | - | **Oui** |
| `map_client_m3ua.callback` | Tuple | - | **Oui** |
| `map_client_m3ua.process_name` | Atom | - | **Oui** |
| `map_client_m3ua.local_ip` | Tuple ou Liste | - | **Oui** |
| `map_client_m3ua.local_port` | Integer | `2905` | **Oui** |
| `map_client_m3ua.remote_ip` | Tuple ou Liste | - | **Oui** |
| `map_client_m3ua.remote_port` | Integer | `2905` | **Oui** |
| `map_client_m3ua.routing_context` | Integer | - | **Oui** |
| `map_client_m3ua.opc` | Integer | `5013` | Non |

| Paramètre | Type | Par Défaut | Requis | |
|---|---|---|---|---|
| | | | | |
| `map_client_m3ua.dpc` | Integer | `5011` | Non | |
| `map_client_m3ua.receive_watchdog` | Boolean | `true` | Non | |
| `map_client_m3ua.receive_watchdog_idle` | Integer | `15` | Non | |

**Exemple (IP Unique)** :

```
config :omniss7,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    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)
  }
```

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

**Exemple (Multihoming SCTP)** :

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

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

# Receive Watchdog

Le **receive watchdog** surveille les connexions SCTP pour les sockets zombies — des associations qui restent 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, une connexion morte peut ne pas être détectée 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. Dans les environnements où le SG distant n'envoie pas de trafic périodique au niveau de la couche d'application (et ne send pas de M3UA BEATs), désactiver le watchdog évite des cycles de reconnexion inutiles.

| Paramètre | Type | Par Défaut | Description |
|---|---|---|---|
| `receive_watchdog` | Boolean | `true` | Activer ou désactiver le watchdog de réception. 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` | Integer | `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 — Désactivation du watchdog pour un client ASP :**

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

**Exemple — Personnalisation du seuil d'inactivité :**

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

---

# Paramètres d'Infrastructure

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

# Configuration de Licence

| Paramètre | Type | Par Défaut | Requis | |
|---|---|---|---|---|
| `license_client.license_server_api_urls` | Liste de Strings | - | **Oui** | |
| `license_client.licensee` | String | - | **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 | Par |
|---|---|---|
| `control_panel.parent_application_readable_name` | String | `"OmniSS7 S` |
| `control_panel.use_additional_pages` | Liste de Tuples | `[]` |
| `control_panel.page_order` | Liste de Strings | `[]` |
| `ControlPanelWeb.Endpoint.url.host` | String | `"0.0.0.0"` |
| `ControlPanelWeb.Endpoint.https.port` | Integer | `8087` |
| `ControlPanelWeb.Endpoint.https.keyfile` | String | `"priv/cert/` |
| `ControlPanelWeb.Endpoint.https.certfile` | String | `"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: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "Pairs"}
  ],
  page_order: ["/events", "/m3ua", "/application",
"/configuration"]
```

## Serveur API REST

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

**Exemple :**

```
config :omniss7,
  start_http_server: true

config :api_ex,
  api: %{
    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"
  }
```

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

- API REST : `https://[server-ip]:8445/api/*`
- Swagger UI : `http://[server-ip]:8080/swagger`
- Métriques Prometheus : `http://[server-ip]:8080/metrics`

# Configuration de Journalisation

| Paramètre | Type | Par Défaut | Requ |
|---|---|---|---|
| `logger.level` | Atom | `:debug` | Non |
| `logger.backends` | Liste | `[:console]` | Non |
| `logger.default_formatter.format` | String | - | Non |
| `logger.default_formatter.metadata` | Liste | `[]` | Non |
| `logger.default_formatter.truncate` | Atom/Integer | `: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` | Atom | `: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'Expiration

| Valeur | Impact | Contournement |
|---|---|---|
| Délai d'attente de requête MAP : **10 secondes** | Toutes les opérations MAP expirent après 10s | Modifier le code source |
| Délai 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 : **8080** | Le point de terminaison des métriques/Swagger fonctionne sur le port 8080 | Modifier le code source |

## Intervalles d'Enregistrement

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

## Actualisation Automatique de l'UI 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://10.180.2.140:8443",
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112",

  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

## Configuration SMSc Minimale

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

  smsc_api_base_url: "https://10.179.3.219:8443",
  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: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    local_ip: {10, 179, 4, 12},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

## 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: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    point_code: 100
  },

  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    local_ip: {10, 179, 4, 10},
    local_port: 2906,
    remote_ip: {10, 179, 4, 11},
    remote_port: 2905,
    routing_context: 1
  }
```

---

# Résumé

## Total des Paramètres de Configuration : 75+

**Par Catégorie** :

- Mode Opérationnel : 5 paramètres
- Mode HLR : 17 paramètres
- Mode SMSc : 8 paramètres

- Mode STP : 5+ paramètres (plus les listes m3ua_peers, m3ua_routes, m3ua_gt_routes)
- Mode Passerelle CAMEL : 14 paramètres
- NAT de Titre Global : 2 paramètres
- Connexion M3UA : 8 paramètres
- Infrastructure (Licence, Web, API, Journalisation) : 23 paramètres
- Base de Données : 1 paramètre

**Paramètres Requis par Mode** :

**Mode HLR :**

- `hlr_api_base_url`
- `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 Passerelle CAMEL :**

- `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 au HLR
- **Guide SMSc** - Configuration spécifique au SMSc
- **Guide STP** - Configuration de routage STP
- **Guide API** - Référence de l'API REST
- **Guide Passerelle USSD** - Configuration de la Passerelle USSD et protocole de rappel HTTP
- **Guide UI Web** - Documentation de l'interface web

# Guide de NAT de Titre Global

## Vue d'ensemble

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 de la nécessité de garantir 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 la 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 = plus haute priorité) 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
   - Les préfixes appelants et appelés
   - Aucun (règle de secours/caractère générique)

## Nouveau format de règle

**Champs requis :**

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

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

- `calling_prefix` : Correspondre au préfixe GT de la partie appelante
- `called_prefix` : Correspondre au 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"},

  # Secours générique - 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 d'un routage complexe. Avec GT NAT, une seule instance d'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 fonctionne

## 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 NATé)

## 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 du "55512341112"
    %{calling_prefix: "8772", response_gt: "55512341112"},

    # Règle 2 : Les appels du préfixe "8773" obtiennent une
réponse du "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 NAT de Titre Global dans la Référence de Configuration.

| Paramètre | Type | Requis | Description |
|---|---|---|---|
| `gt_nat_enabled` | Booléen | 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 faire
correspondre avec le GT appelant
  called_prefix: "555",        # (Optionnel) Préfixe à faire
correspondre avec le GT appelé
  weight: 10,                  # (Requis) Valeur de priorité (plus
bas = plus haute priorité)
  response_gt: "55512341112"   # (Requis) GT à utiliser dans les
réponses
}
```

**Champs de règle :**

- **`calling_prefix`** (Optionnel) : Préfixe de chaîne à faire correspondre avec le GT appelant entrant
  - La correspondance est effectuée par `String.starts_with?/2`
  - Chaîne vide `""` ou `nil` agit comme un caractère générique (correspond à n'importe quel GT appelant)
  - Peut être omis pour correspondre à n'importe quel GT appelant
- **`called_prefix`** (Optionnel) : Préfixe de chaîne à faire correspondre avec le GT appelé entrant
  - La correspondance est effectuée par `String.starts_with?/2`
  - Chaîne vide `""` ou `nil` agit comme un caractère générique (correspond à n'importe quel GT appelé)
  - Peut être omis pour correspondre à n'importe quel GT appelé
- **`weight`** (Requis) : Valeur de priorité entière
  - Poids inférieur = 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 l'un des `calling_prefix` ou `called_prefix` doit être spécifié pour un routage spécifique. Les deux peuvent être omis pour une règle de secours/caractère générique.**

# Logique de correspondance des règles

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

**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 des deux n'est défini, la règle agit comme un caractère générique
2. Trier les règles correspondantes par :
   - **Primaire** : Poids (croissant - valeurs inférieures 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"},  #
Appel uniquement
  %{called_prefix: "555", weight: 10, response_gt: "333333"},    #
Appelé uniquement

  # Poids 100 : Priorité la plus basse - secours générique
  %{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,
appel uniquement)
# Appelant : "999999999", Appelé : "555123" -> "333333" (poids 10,
appelé uniquement)
# Appelant : "999999999", Appelé : "888888" -> "444444" (poids
100, caractère générique)
```

# Exemples

## Exemple 1 : Deux partenaires de réseau

**Scénario** : Vous exploitez un SMSc avec deux partenaires de 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 (secours
générique)
    %{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 (NATé)
  networkNode-Number : 5551001 (dans la réponse MAP)
```

# Exemple 2 : HLR avec GT 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 (caractère générique)
    %{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 (caractère
générique)
    %{weight: 100, response_gt: "123456789"}  # Ancien GT
  ]
```

# Exemple 4 : Correspondance de préfixe de la partie appelée (NOUVEAU)

**Scénario** : Vous avez plusieurs GT pour différents services et souhaitez répondre avec le bon GT en fonction du GT 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"},

    # 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 des préfixes appelants + appelés (AVANCÉ)

**Scénario** : Différents partenaires appellent différents GT, 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"},

    # 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 (appel uniquement) → "555200"

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

# Appelant inconnu appelle le GT Voix
Appelant : "999999999", Appelé : "555200"
→ Correspond à la règle de poids 100 caractère générique →
"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"}  # Secours générique
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"}  # Secours générique
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"}  # Secours
générique par défaut
  ]
```

**Cas d'utilisation :** Lorsque vous opérez en tant que gsmSCF (Service Control Function) pour plusieurs réseaux, chaque gsmSSF de 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 la 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 :

```
# Vérifiez 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"}
]}

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

# Obtenez 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 :

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

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

# Testez 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"

# Testez 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"

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

# Dépannage

## Problème : GT NAT ne fonctionne pas

#### Vérifiez 1 : Est-il activé ?

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

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

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

**Vérifiez 3 : Vérifiez les journaux** Recherchez "GT NAT" dans les journaux pour voir si des traductions ont lieu.

# 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 attrape le trafic

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

```
# MAUVAIS : Caractère générique 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 inférieur, caractère
générique avec un poids plus élevé
gt_nat_rules: [
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},  #
Spécifique, poids bas
  %{weight: 100, response_gt: "111111"}  # Caractère générique,
poids élevé (secours)
]
```

## Problème : GT NAT non appliqué à un type de message spécifique

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

**Couverture actuelle :**

- □ GT appelant SCCP (toutes les réponses)
- □ Réponses SRI-for-SM (networkNode-Number)
- □ Messages UpdateLocation ISD (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 la 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 générique

```
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
caractère générique avec un poids élevé
]
```

## 2. Utiliser des préfixes et des 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 des 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 le déploiement

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

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

## 5. Surveiller les journaux

Activez la journalisation de 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** avec les 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)
- S'il n'est pas 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 départ)
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"}  # Secours générique
  ]
```

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 une fenêtre de maintenance

- Surveiller les 24 premières heures de près
- Avoir un plan de retour prêt (définir `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 configuration

# Guide de Configuration HLR

← Retour à la Documentation Principale

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

# Intégration OmniHSS

**Le mode HLR d'OmniSS7 fonctionne comme un frontend de signalisation SS7** qui interagit 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 provisionnement 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 le provisionnement 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 de réseau CS (circuit-switched) et PS (packet-switched)
- **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 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 de réseau sur un seul appareil compatible eSIM
- **Changement de Réseau** : Changement transparent entre les réseaux sans changer de MSISDN
- **Coordination Dual SIM** : Coordination entre plusieurs SIM physiques ou virtuelles
- **Tests & Développement** : Plusieurs IMSI de test pointant vers le même abonné

**Comment cela fonctionne :**

- Chaque IMSI a ses propres informations d'authentification (Ki, OPc, algorithme)
- Chaque IMSI peut avoir des enregistrements de sessions 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 d'abonné appropriées
- 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 National 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 la correspondance IMSI-abonné et garantit une authentification et un provisionnement appropriés pour chaque IMSI.

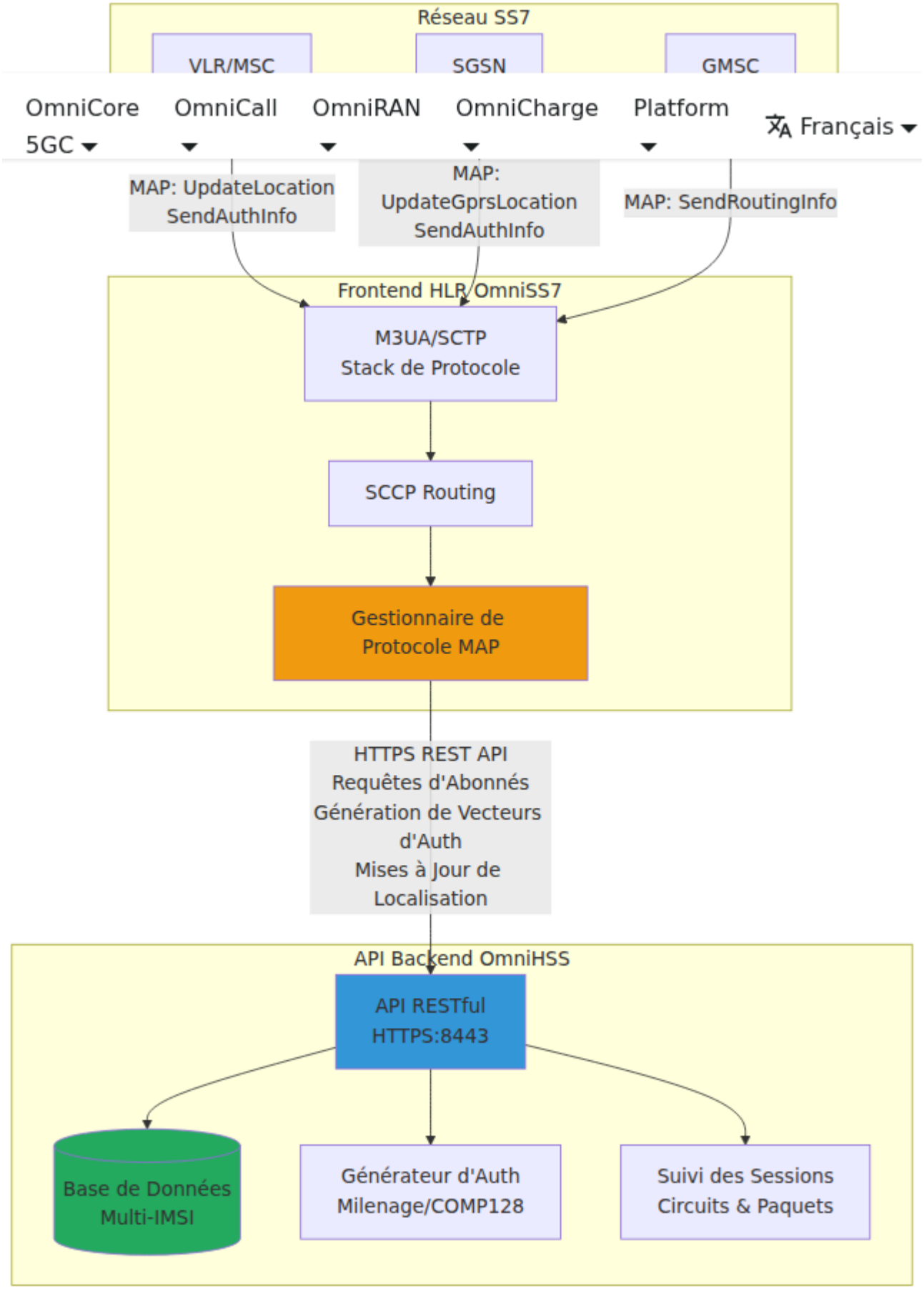
# Table des Matières

# 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. Pour l'utiliser comme HLR, vous devez activer le mode HLR dans la configuration.

## Changement vers le Mode HLR

Le fichier `config/runtime.exs` d'OmniSS7 contient trois modes opérationnels préconfigurés. Pour activer le mode HLR :

1. **Ouvrir** `config/runtime.exs`
2. **Trouver** les trois sections de configuration (lignes 53-174) :
   - Configuration 1 : Mode STP (lignes 53-85)
   - Configuration 2 : Mode HLR (lignes 87-123)
   - Configuration 3 : Mode SMSc (lignes 125-174)
3. **Commenter** la configuration actuellement active (ajouter `#` à chaque ligne)
4. **Décommenter** la configuration HLR (retirer `#` des lignes 87-123)
5. **Personnaliser** les paramètres de configuration selon les besoins
6. **Redémarrer** l'application : `iex -S mix`

## Configuration du Mode HLR

La configuration complète HLR 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://10.180.2.140:8443",

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

  # Configuration de la Correspondance MSISDN ↔ IMSI
  # Voir : section de Correspondance 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 de 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 à Domicile
  # Liste des préfixes d'adresses VLR considérés comme réseau
"domestique"
  # Si le VLR de l'abonné commence par l'un de ces préfixes,
```

```
utiliser la réponse SRI standard
  # Sinon, l'abonné est en itinérance et nous devons envoyer PRN
pour obtenir MSRN
  home_vlr_prefixes: ["123456"],

  # Configuration de Connexion M3UA
  # Se connecter en tant qu'ASP pour recevoir les 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_service_center_gt_address` | String | *Requis* |
| `smsc_service_center_gt_address` | String | *Requis* |
| `hlr_smsc_alert_gts` | List | `[]` |
| `hlr_alert_location_expiry_seconds` | Integer | `172800` |
| `hlr_imsi_plmn_prefix` | String | `"50557"` |
| `hlr_msisdn_country_code` | String | `"61"` |

| Paramètre | Type | Par Défaut |
| --- | --- | --- |
| | | |
| `hlr_msisdn_nsn_offset` | Integer | `0` |
| `hlr_msisdn_nsn_length` | Integer | `9` |
| `isd_network_access_mode` | Atom | `:packetAndCircuit` |
| `isd_send_ss_data` | Boolean | `true` |
| `isd_send_call_barring` | Boolean | `true` |
| `camel_service_key` | Integer | `11_110` |

| Paramètre | Type | Par Défaut |
|---|---|---|
| `camel_trigger_detection_point` | Atom | `:termAttemptAuthorized` |
| `home_vlr_prefixes` | List | `["5551231"]` |
| `local_ip` | Tuple | *Requis* |
| `local_port` | Integer | `2905` |
| `remote_ip` | Tuple | *Requis* |
| `remote_port` | Integer | `2905` |
| `routing_context` | Integer | `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
- **M3UA** - État de la connexion
- **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 cach��s.

## Notes 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 backend doit être accessible à l'URL `hlr_api_base_url` configurée
- **Délai d'Attente des Requêtes API** : Toutes les requêtes API OmniHSS ont un **délai d'attente de 5 secondes codé en dur**
- **Délai d'Attente des Requêtes MAP** : Toutes les requêtes MAP (SRI, UpdateLocation, SendAuthInfo, etc.) ont un **délai d'attente de 10 secondes codé en dur**
- **Délai d'ISD** : Chaque message InsertSubscriberData (ISD) dans une séquence UpdateLocation a un **délai d'attente de 10 secondes codé en dur**
- La connexion M3UA au STP est requise pour recevoir les 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** : Sélection de Ki, OPc et algorithme (Milenage ou COMP128)
- **Profils de service** : Catégorie d'abonné, services autorisés, paramètres QoS
- **Suivi de localisation** : Suivi indépendant de l'emplacement 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'Authentification

**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 g��nérer 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"
```

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

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

---

# Mises à Jour de Localisation

## Traitement des Mises 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 GTs 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

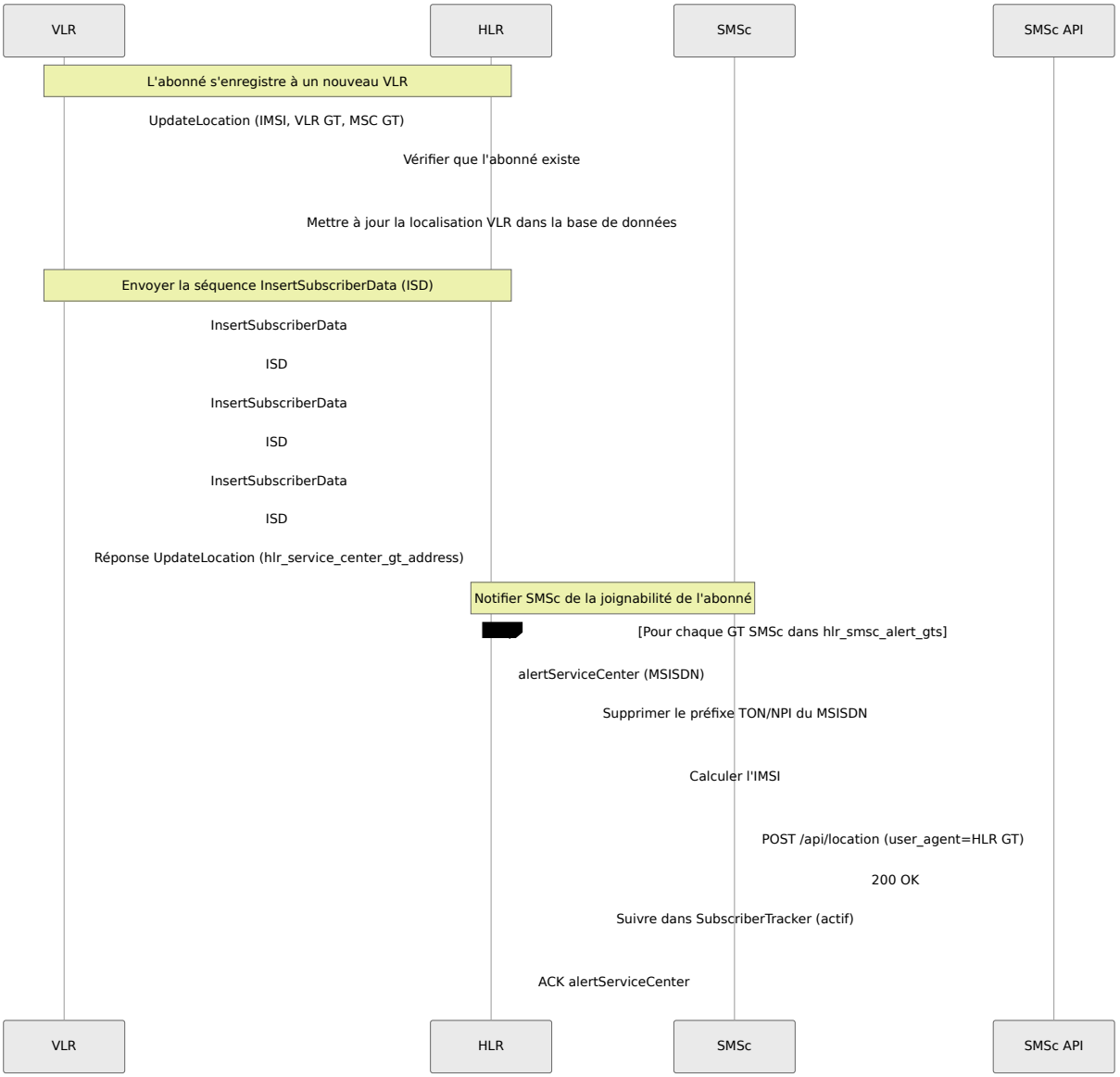
Après une mise à jour de localisation réussie, le HLR peut automatiquement notifier les systèmes SMSc qu'un abonné est maintenant joignable en envoyant des messages **alertServiceCenter** (opcode MAP 64). Pour des informations sur la manière dont le SMSc gère ces alertes, voir Gestion du Centre de Service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

### Configuration

Configurer la liste des Titres Globaux SMSc à notifier :

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

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

### Diagramme de Flux




### Comportement

Lorsqu'un abonné effectue UpdateLocation :

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/location)
- **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 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 des tests :

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

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

## Configuration InsertSubscriberData (ISD)

Après une mise à jour de localisation réussie, le HLR envoie des données de provisionnement d'abonné au VLR en utilisant des messages **InsertSubscriberData** (ISD). La configuration ISD vous permet de personnaliser quelles données sont 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
    - État de l'abonné (serviceGranted)
    - Liste des services porteurs
    - Liste des téléservices
    - Mode d'accès au réseau
2. **ISD #2** (Optionnel) - Données de Services Supplémentaires (SS) :
    - Paramètres de transfert d'appels (inconditionnel, occupé, pas de réponse, non joignable)
    - Attente d'appels
    - Mise en attente d'appels
    - Service multi-parties
    - État et fonctionnalités des services supplémentaires
3. **ISD #3** (Optionnel) - Données de Barrage d'Appels :
    - Barrage de tous les appels sortants (BAOC)
    - Barrage des appels internationaux sortants (BOIC)
    - Données de restriction d'accès

**Options de Configuration**

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

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

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

### Mode d'Accès Réseau

Le paramètre `isd_network_access_mode` contrôle quel type d'accès 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, appareils IoT |
| `:circuitOnly` | Commuté par circuits uniquement (voix/SMS) | Appareils anciens, plans uniquement vocaux |

### Contrôle des Messages ISD

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

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

```
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 les données de base + services supplémentaires** (Pas de barrage 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) - Barrage 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 de 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/barrage d'appels pour les appareils uniquement de données
- **Interopérabilité** : Certains anciens VLR peuvent ne pas prendre en charge tous les services supplémentaires - désactivez `isd_send_ss_data` si vous rencontrez des problèmes
- **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 requêtes **SendRoutingInfo** (SRI) d'un GMSC (Gateway MSC), le HLR peut demander au GMSC d'invoquer des services CAMEL pour un routage d'appels intelligent et un contrôle de service.

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 pr��payée
- Filtrage et barrage d'appels
- Réseaux Privés Virtuels (VPN)
- Services à tarif premium
- Transfert 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 de service CAMEL
camel_service_key: 11_110,

# Point de Détection de Déclenchement 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 d'appel prépayé entrant (par défaut) |
| `100` | Service prépayé d'origine |
| `200` | Transfert d'appels avec logique personnalisée |
| `300` | Réseau Privé Virtuel (VPN) |
| Personnalisé | Services spécifiques à l'opérateur |

**Exemple de Configuration :**

```
# Pour le contrôle d'appel prépayé entrant
camel_service_key: 11_110,

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

**Point de Détection de Déclenchement**

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 en terminaison | Lorsque l'abonné est occupé |
| `:tNoAnswer` | Pas de réponse en terminaison | Lorsque l'abonné ne répond pas |
| `:tAnswer` | Réponse en terminaison | 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 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 de Déclenchement : termAttemptAuthorized
    * Gestion des Appels par Défaut : continueCall

Le GMSC contacte le 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 prépayés entrants
- **Gestion des Retours** : `defaultCallHandling: :continueCall` garantit que les appels se poursuivent si le gsmSCF est injoignable

---

# Gestion des Abonnés en Itinérance

## Détection du VLR à Domicile vs VLR en Itinérance

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

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

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

# Configuration

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

**Exemple de Configuration** :

```
# Un seul opérateur de réseau domestique
home_vlr_prefixes: ["555123"],

# Plusieurs opérateurs de réseau domestique (par exemple,
différentes régions ou filiales)
home_vlr_prefixes: ["555123", "555124", "555125"],
```

# Comment Cela Fonctionne

### 1. Flux Abonné à Domicile (Standard)

Lorsque l'adresse VLR de l'abonné commence par un préfixe domestique 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 domestique
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 de Déclenchement : termAttemptAuthorized
```

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

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

```
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" → Itinérance
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 en Itinérance (MSRN) : "49170999888777"
```

## Différences de Structure de Réponse

#### Réponse SRI Abonné à 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 Itinérance

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

# Opération Provide Roaming Number (PRN)

### Structure de Requête PRN

La requête PRN envoyée au MSC/VLR contient :

| Champ | Source | Description |
|---|---|---|
| **MSISDN** | Requête SRI | Numéro de téléphone de l'abonné |
| **IMSI** | API HLR | IMSI de l'abonné |
| **Numéro MSC** | API HLR | MSC servant l'abonné en itinérance (`serving_msc`) |
| **Adresse GMSC** | Requête SRI | GMSC effectuant la requête SRI d'origine |
| **Numéro de Référence d'Appel** | Statique | Identifiant de référence d'appel |
| **Phases CAMEL Supportées** | Statique | Phases CAMEL prises en charge par le GMSC |

**Gestion de la Réponse PRN**

Le HLR s'attend à une réponse PRN contenant :

- **MSRN** (Numéro de Roaming de 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 de Réseau Domestique Unique

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

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

### Opérateur Multi-Régions

```
# Plusieurs réseaux domestiques 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` → Itinérance (international)

### Configuration de Test

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

```
# Tous les VLR sont traités comme en itinérance (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 d'Itinérance** : Assurez-vous que le PRN est correctement pris en charge par les réseaux partenaires en itinérance
- **Tests** : Testez soigneusement les scénarios domestiques et en itinérance avant le déploiement en production
- **Surveillance** : Surveillez les taux de délai d'attente PRN pour identifier les problèmes de connectivité avec les partenaires en itinérance

# Dépannage

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

- **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 requêtes PRN expirent

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

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

- **Cause** : Le format de réponse PRN du partenaire en itinérance 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

- `updateLocation` (Opcode 2) - Enregistrer la localisation VLR
- `sendAuthenticationInfo` (Opcode 56) - Générer des vecteurs d'authentification
- `sendRoutingInfo` (Opcode 22) - Fournir MSRN pour les appels avec support CAMEL
- `sendRoutingInfoForSM` (Opcode 45) - Fournir GT MSC pour les SMS
- `cancelLocation` (Opcode 3) - Désenregistrer du vieux VLR
- `insertSubscriberData` (Opcode 7) - Pousser le profil d'abonné

## Mapping 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 réponses différents en fonction de si l'abonné est sur un VLR domestique ou en itinérance :

##### Réponse Abonné à 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** | runtime.exs | Identifiant de service CAMEL (`camel_service_key`) |
| **Point de Détection de Déclenchement** | runtime.exs | Quand déclencher CAMEL (`camel_trigger_detection_point`) |
| **Gestion des Capacités CAMEL** | Statique | Niveau de support des phases CAMEL |
| **Gestion des Appels par Défaut** | Statique | Retour si gsmSCF injoignable |

#### Réponse Abonné en Itinérance (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 en Itinérance (MSRN)** | Réponse PRN | MSRN obtenu de la requête ProvideRoamingNumber | `"4` |

**Logique de Décision de Routage** :

```
1. OmniSS7 reçoit la requête 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 le VLR commence par le préfixe domestique :
     → Retourner les informations de routage CAMEL (flux abonné à
domicile)

   Si le VLR ne correspond à aucun préfixe domestique :
     → Envoyer ProvideRoamingNumber (PRN) au MSC
     → Extraire le MSRN de la réponse PRN
     → Retourner les informations de routage avec le MSRN (flux
abonné en itinérance)
```

**Flux de Données** :

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

**Exigences de Configuration** :

```
# Dans runtime.exs
home_vlr_prefixes: ["555123"],  # Liste des préfixes VLR à
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 requête PRN expire (cas d'itinérance) : Retourne l'erreur 27 (Abonné Absent)
- Si la réponse PRN est invalide (cas d'itinérance) : Retourne l'erreur 27 (Abonné Absent)

---

### Réponse UpdateLocation avec InsertSubscriberData

**But** : Enregistre l'abonné à un nouveau VLR et provisionne les données de l'abonné.

#### Réponse UpdateLocation FIN

| Champ | Source | Description | Exemple |
|---|---|---|---|
| **Numéro HLR** | runtime.exs | Titre Global de ce HLR (`hlr_service_center_gt_address`) | `"555123456` |
| **Type de Message TCAP** | Statique | Réponse finale après tous les ISD | `FIN` |

#### InsertSubscriberData #1 (Données de Base de l'Abonné)

| Champ | Source | Description | Exemple |
|---|---|---|---|
| **IMSI** | Requête | Provenant de la requête UpdateLocation | `"999999876543` |
| **MSISDN** | API OmniHSS | Numéro de téléphone de l'abonné provenant d'OmniHSS | `"555123456"` |
| **Catégorie** | Statique | Catégorie d'abonné | `"\n"` (0x0A) |
| **État de l'Abonné** | Statique | État de service | `:serviceGrant` |
| **Liste des Services Porteurs** | Statique | Services porteurs pris en charge | `[<<31>>]` |
| **Liste des Téléservices** | Statique | Téléservices pris en charge | `[<<17>>, "` `"\""]` |
| **Mode d'Accès au Réseau** | runtime.exs | Accès paquet/circuit (`isd_network_access_mode`) | `:packetAndCir` |

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

| Champ | Source | Description | Contrôlé Par |
|---|---|---|---|
| **SS Provisionnés** | Statique | Données des services supplémentaires | `isd_send_ss_data: true` |
| **Transfert d'Appels** | Statique | Paramètres de transfert (inconditionnel, occupé, pas de réponse, non joignable) | Config activée |
| **Attente d'Appels** | Statique | État du service d'attente d'appels | Config activée |
| **Service Multi-Parties** | Statique | Support d'appel conférence | Config activée |

**ISD #2 inclut** :

- Transfert d'appels inconditionnel (code SS 21)
- Transfert d'appels en cas d'occupation (code SS 41)
- Transfert d'appels en cas de non réponse (code SS 42)
- Transfert d'appels en cas de non joignabilité (code SS 62)
- Attente d'appels (code SS 43)
- Service multi-parties (code SS 51)
- Services CLIP/CLIR

#### InsertSubscriberData #3 (Barrage d'Appels) - Optionnel

| Champ | Source | Description | Contrôlé Par |
|---|---|---|---|
| **Informations de Barrage d'Appels** | Statique | Configurations de barrage d'appels | `isd_send_call_barring: true` |
| **BAOC** | Statique | Barrage de Tous les Appels Sortants (code SS 146) | Config activée |
| **BOIC** | Statique | Barrage des Appels Internationaux Sortants (code SS 147) | Config activée |
| **Données de Restriction d'Accès** | Statique | Restrictions d'accès au réseau | Config activée |

**Contrôle de Séquence ISD** :

- ISD #1 : **Toujours envoyé** - Contient les données essentielles de l'abonné
- ISD #2 : Envoyé uniquement si `isd_send_ss_data: true` dans runtime.exs
- ISD #3 : Envoyé uniquement si `isd_send_call_barring: true` dans runtime.exs

---

**Réponse SendRoutingInfoForSM (SRI-for-SM)**

**But** : 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 requête SRI-for-SM au HLR pour déterminer où router le message.

**Structure de Réponse** :

| Champ | Source | Description | Comment Généré |
|---|---|---|---|
| **IMSI** | Calculé | IMSI synthétique dérivé de MSISDN | `PLMN_PREFIX + zero_padded_MSISDN` |
| **Numéro de Nœud Réseau** | runtime.exs | Adresse GT du SMSC pour le routage SMS | `smsc_service_center_gt_address` |

**Paramètres de Configuration** (provenant de `runtime.exs`) :

```
# Adresse du Centre de Service GT (renvoyée dans les réponses SRI-for-SM)
# Cela indique au SMSc demandeur où envoyer les messages MT-ForwardSM
smsc_service_center_gt_address: "5551234567",  # Requis

# Configuration de la Correspondance MSISDN ↔ IMSI
# Préfixe PLMN : MCC (001 = Réseau de Test) + MNC (01 = Opérateur de Test)
hlr_imsi_plmn_prefix: "001001",                # Seul paramètre de configuration nécessaire !
```

---

# Correspondance 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 correspondance inverse IMSI→MSISDN (par exemple, "61" pour l'Australie, "1" pour les États-Unis/Canada)
- `hlr_msisdn_nsn_offset` : Position du caractère où commence le Numéro d'Abonné National (NSN) dans le MSISDN (typiquement 0 si le MSISDN n'inclut pas le code pays, ou longueur du code pays s'il l'inclut)
- `hlr_msisdn_nsn_length` : Nombre de chiffres à extraire du MSISDN comme NSN

Pour des détails supplémentaires sur la configuration, voir Correspondance MSISDN ↔ IMSI dans la Référence de Configuration.

**Pourquoi la Correspondance MSISDN vers IMSI est-elle N��cessaire ?**

Le protocole MAP pour **SendRoutingInfoForSM** (SRI-for-SM) exige que le HLR renvoie un **IMSI** (Identité Internationale d'Abonné Mobile) dans sa réponse. Cependant, le SMSc demandeur ne connaît que le **MSISDN** (numéro de téléphone) de l'abonné.

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 de MT-ForwardSM au MSC/VLR

**Approche d'OmniSS7 - IMSIs Synthétiques** :

Au lieu de maintenir une base de données complète d'abonnés avec des correspondances MSISDN vers IMSI, OmniSS7 utilise un schéma d'encodage simple pour **calculer** les IMSIs synthétiques directement à partir des MSISDN. Cette approche offre deux avantages clés :

1. **Confidentialité** : Les véritables IMSIs 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 d'interroger 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 Cela Fonctionne** :

Les MSISDN sont encodés directement dans la partie abonnée de l'IMSI (les chiffres après MCC+MNC) :

```
IMSI = PLMN_PREFIX + MSISDN avec zéros ajoutés
```

Où :

- **PLMN_PREFIX** : MCC + MNC (par exemple, "001001" pour Réseau de Test)
- **MSISDN** : Tous les chiffres numériques du numéro de téléphone
- **Ajout de Zéros** : 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 requête 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 Fina |
|---|---|---|---|---|
| `"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

**Gestion de la Correspondance Inverse (IMSI → MSISDN)** :

Le SMSc peut inverser cette correspondance 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 Correspondance Inverse** :

| IMSI d'Entrée | Préfixe PLMN | Portion Abonnée | Supprimer les Zéros à Gauche | MSISDI Final |
|---|---|---|---|---|
| `"001001555123456"` | `"001001"` | `"555123456"` | `"555123456"` | `"5551234` |
| `"001001000000099"` | `"001001"` | `"000000099"` | `"99"` | `"99"` |
| `"001001999999999"` | `"001001"` | `"999999999"` | `"999999999"` | `"9999999` |

**Propriétés de Cette Correspondance** :

- □ **Déterministe** : Le même MSISDN produit toujours le même IMSI
- □ **Récupérable** : Peut être converti de l'IMSI au MSISDN
- □ **Configuration Minimale** : Nécessite uniquement `hlr_imsi_plmn_prefix`
- □ **Préservation de la Confidentialité** : Les véritables IMSIs ne sont jamais exposés
- □ **Aucune 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 requête 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. **Extraire les Chiffres** : Conserver uniquement les chiffres numériques, supprimer tous les caractères non numériques
3. **Normaliser** : Si plus long que l'espace disponible, prendre les chiffres de droite ; si plus court, compléter par des zéros
4. **Encoder** : Concaténer le préfixe PLMN + MSISDN normalisé

**Considérations de Sécurité** :

Les IMSIs synthétiques renvoyés dans les réponses SRI-for-SM sont uniquement à des fins de routage. Ils ne sont PAS les véritables IMSIs stockés dans la base de données des abonnés HLR. Cela fournit une couche de protection supplémentaire de la confidentialité, car les véritables IMSIs des abonnés ne sont exposés que lorsque cela est absolument nécessaire (par exemple, lors des opérations UpdateLocation ou SendAuthenticationInfo qui nécessitent de véritables vecteurs d'authentification).

**Flux de Réponse** :

```
1. SMSc → HLR: Requête SRI-for-SM
   - MSISDN (TBCD): "91123456789" (inclut TON/NPI)

2. Traitement HLR :
   - Décodage TBCD : "91123456789"
   - Extraire les chiffres : "91123456789" (11 chiffres)
   - Adapter à 9 chiffres : "555123456" (9 chiffres de droite)
   - Ajouter PLMN : "001001" + "555123456" = "001001555123456"
   - Obtenir GT SMSC à partir de la configuration : "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 `runtime.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 pays)
hlr_msisdn_country_code: "1",      # Utilisé pour la
correspondance inverse (IMSI→MSISDN)
hlr_msisdn_nsn_offset: 1,          # Ignorer 1 chiffre du code
pays
hlr_msisdn_nsn_length: 10          # Extraire 10 chiffres NSN
```

**Configuration d'Extraction NSN** :

Si vos MSISDN incluent le code pays (par exemple, `"68988000088"` au lieu de juste `"88000088"`), vous devez configurer l'extraction NSN :

- **`hlr_msisdn_nsn_offset`** : Position où commence le NSN (typiquement la longueur de votre code pays)

- **`hlr_msisdn_nsn_length`** : Nombre de chiffres dans le NSN

**Exemples** :

| Exemple | Code Pays | Exemple MSISDN | nsn_offset | nsn_length | NSN Ex |
|---|---|---|---|---|---|
| Code Pays 1 chiffre | `"9"` | `"95551234567"` | `1` | `10` | `"555123` |
| Code Pays 2 chiffres | `"99"` | `"99412345678"` | `2` | `9` | `"412345` |
| Code Pays 3 chiffres | `"999"` | `"99988000088"` | `3` | `8` | `"880000` |

**Comment Cela Fonctionne** :

1. **MSISDN → IMSI** : Extraire le NSN du MSISDN, compléter par des zéros, 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, supprimer les zéros à gauche, préfixer le code pays

```
IMSI : "547050088000088"
Portion abonnée : "088000088"
Supprimer les zéros : "88000088"
MSISDN : "+999" + "88000088" = "+99988000088"
```

# Exigences API : Aucune - SRI-for-SM utilise uniquement des valeurs calculées et des configurations. Aucun appel à la base de données n'est requis.

## Résumé des Sources de Champs

| Type de Source | Description | Exemples |
|---|---|---|
| **API OmniHSS** | Données dynamiques de la base de données d'abonnés OmniHSS | IMSI, MSISDN, VLR/MSC servant à partir de circuit_session |
| **runtime.exs** | Paramètres de configuration OmniSS7 | `smsc_service_center_gt_address`, `camel_service_key`, `isd_network_access_mode` |
| **Statique** | Valeurs codées en dur dans le générateur de réponse | État de l'abonné, services porteurs, codes SS |
| **Requête** | Champs extraits de la requête MAP entrante | IMSI de UpdateLocation, MSISDN de SRI |
| **Calculé** | Valeurs dérivées utilisant la logique | IMSI synthétique dans SRI-for-SM (hlr_imsi_prefix + NSN) |

## Dépendances de Configuration

**Requis dans runtime.exs** :

- `hlr_service_center_gt_address` - Utilisé dans les réponses UpdateLocation
- `smsc_service_center_gt_address` - Utilisé dans les réponses SRI-for-SM (où les MT-ForwardSM doivent être routés)

**Optionnel dans runtime.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 la correspondance MSISDN↔IMSI)

**Requis d'OmniHSS** :

OmniHSS doit fournir des points de terminaison API REST pour :

- Recherche d'abonné par IMSI et MSISDN
- Mises à jour de localisation de session circuit (assignation VLR/MSC)
- Génération de vecteurs d'authentification
- Requêtes sur l'état de l'abonné et les profils de service

---

# Documentation Connexe

**Documentation OmniSS7 :**


- ← Retour à la Documentation Principale
- Guide des Fonctionnalités Communes
- Guide Client MAP

- Référence Technique
- Référence de Configuration

**Documentation OmniHSS :** Pour la gestion des abonnés, le provisionnement, la configuration de l'authentification et les opérations administratives, reportez-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 service et les capacités de gestion Multi-IMSI.

---

# Guide de Configuration du Client MAP

← Retour à la Documentation Principale

Ce guide fournit une configuration détaillée pour utiliser OmniSS7 en tant que **Client MAP** pour envoyer des requêtes de protocole MAP aux éléments du réseau.

## Table des Matières

6. Dépannage

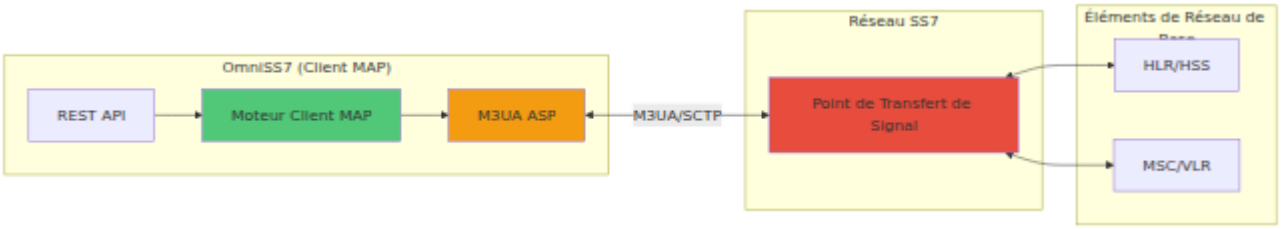
---

# Qu'est-ce que le Mode Client MAP ?

**Mode Client MAP** permet à OmniSS7 de se connecter en tant que **Processus Serveur d'Application (ASP)** à un pair M3UA (STP ou SGP) et d'envoyer/recevoir des messages **MAP (Mobile Application Part)** pour des services tels que :

- **Requêtes HLR** : SRI (Send Routing Info), SRI-for-SM, Authentication Info
- **Mises à Jour de Localisation** : Update Location, Cancel Location
- **Gestion des Abonnés** : Provide Roaming Number (PRN), Insert Subscriber Data

## Architecture Réseau




---

# Activation du Mode Client MAP

Modifiez `config/runtime.exs` et configurez les paramètres du client MAP. Pour une référence de configuration complète, voir Paramètres de Connexion M3UA dans la Référence de Configuration.

## Configuration de Base

```
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",                      # Mode M3UA : "ASP" (client)
ou "SGP" (serveur)
    callback: {MapClient, :handle_payload, []},  # Callback pour
les messages entrants
    process_name: :map_client_asp,    # Nom du processus
enregistré
    local_ip: {10, 0, 0, 100},       # Adresse IP locale
    local_port: 2905,                 # Port SCTP local
    remote_ip: {10, 0, 0, 1},        # IP STP/SGP distante
    remote_port: 2905,                # Port STP/SGP distant
    routing_context: 1                # Contexte de routage M3UA
  }
```

## Exemple de Configuration en Production

```
config :omniss7,
  # Activer le Client MAP pour la production
  map_client_enabled: true,

  # Connexion M3UA en production
  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},        # IP STP de production
    remote_port: 2905,
    routing_context: 1
  }

config :control_panel,
  web: %{
    listen_ip: "0.0.0.0",
    port: 443,
    hostname: "ss7-gateway.example.com",
    enable_tls: true,
    tls_cert: "/etc/ssl/certs/gateway.crt",
    tls_key: "/etc/ssl/private/gateway.key"
  }
```

---

# Opérations MAP Disponibles

# 1. Envoyer des Informations de Routage pour SM (SRI-for-SM)

Interroge le HLR pour déterminer le MSC de service pour la livraison de SMS. Pour des informations détaillées sur la façon dont le HLR traite les requêtes SRI-for-SM, voir SRI-for-SM dans le Guide HLR.

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

**Requête :**

```
{
  "msisdn": "447712345678",
  "serviceCenter": "447999123456"
}
```

**Réponse :**

```
{
  "result": {
    "imsi": "234509876543210",
    "locationInfoWithLMSI": {
      "networkNode-Number": "447999555111"
    }
  }
}
```

**Exemple cURL :**

```
curl -X POST http://localhost/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "447712345678",
    "serviceCenter": "447999123456"
  }'
```

---

## 2. Envoyer des Informations de Routage (SRI)

Interroge le HLR pour des informations de routage d'appel vocal.

**Point de Terminaison API :** `POST /api/sri`

**Requête :**

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456"
}
```

**Réponse :**

```
{
  "result": {
    "imsi": "234509876543210",
    "extendedRoutingInfo": {
      "routingInfo": {
        "roamingNumber": "447999555222"
      }
    }
  }
}
```

---

## 3. Fournir un Numéro de Roaming (PRN)

Demande un numéro de roaming temporaire (MSRN) au MSC de service.

**Point de Terminaison API :** `POST /api/prn`

**Requête :**

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456",
  "msc_number": "447999555111",
  "imsi": "234509876543210"
}
```

---

## 4. Envoyer des Informations d'Authentification

Demande des vecteurs d'authentification au HLR pour l'authentification des abonnés.

**Point de Terminaison API :** `POST /api/send-auth-info`

**Requête :**

```
{
  "imsi": "234509876543210",
  "vectors": 5
}
```

**Réponse :**

```
{
  "result": {
    "authenticationSetList": [
      {
        "rand": "0123456789ABCDEF0123456789ABCDEF",
        "xres": "ABCDEF0123456789",
        "ck": "0123456789ABCDEF0123456789ABCDEF",
        "ik": "FEDCBA9876543210FEDCBA9876543210",
        "autn": "0123456789ABCDEF0123456789ABCDEF"
      }
    ]
  }
}
```

---

## 5. Mettre à Jour la Localisation

Enregistre la localisation actuelle d'un abonné auprès du HLR. Pour des informations détaillées sur le traitement de UpdateLocation et les séquences d'InsertSubscriberData, voir Mises à Jour de Localisation dans le Guide HLR.

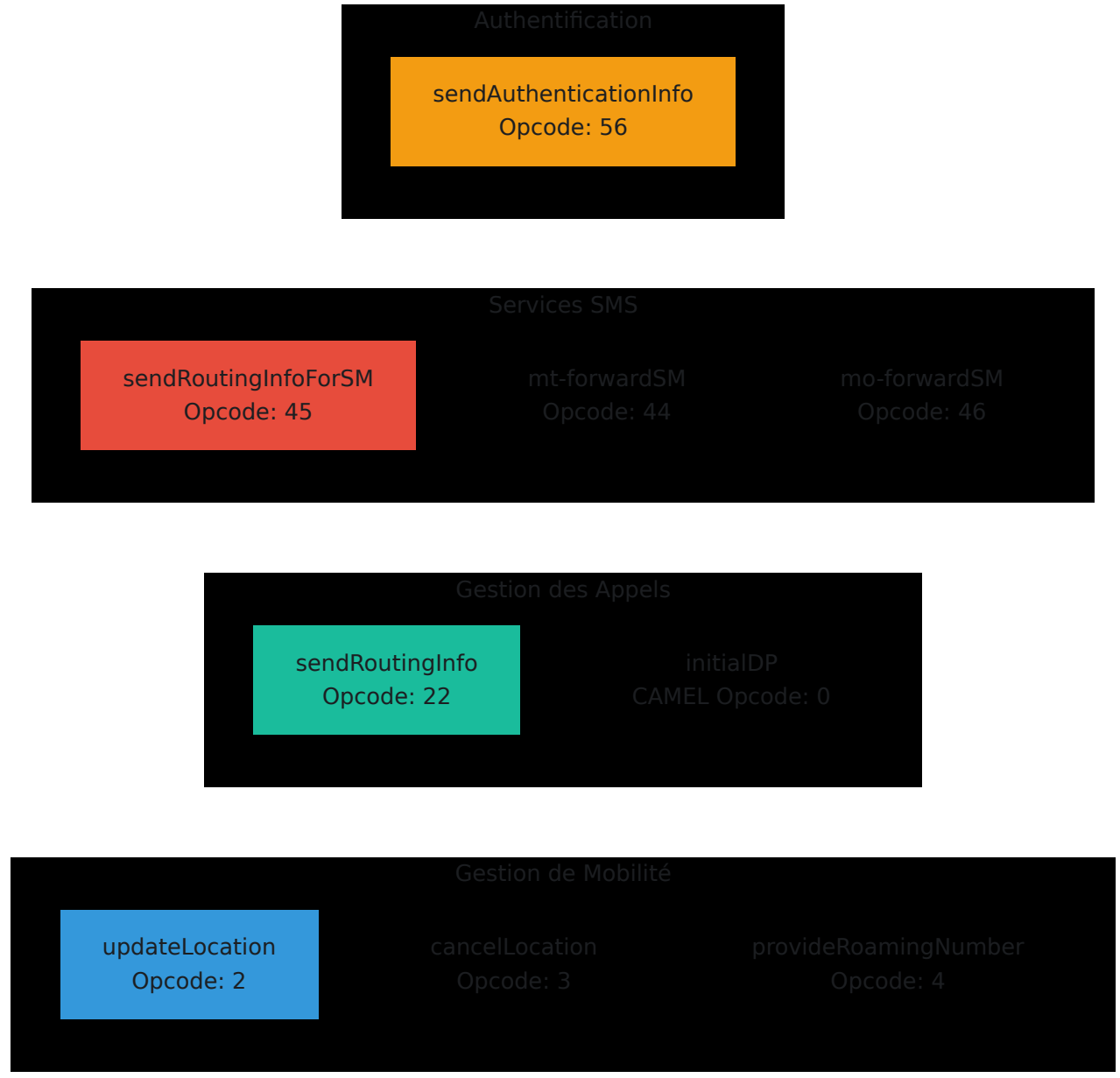
**Point de Terminaison API :** `POST /api/updateLocation`

**Requête :**

```
{
  "imsi": "234509876543210",
  "vlr": "447999555111"
}
```

---

## Résumé des Opérations MAP




---

# Envoi de Requêtes via l'API

## Utilisation de Swagger UI

L'interface Swagger UI fournit une interface interactive pour l'envoi de requêtes SS7.

**Accéder à Swagger UI :**

1. Naviguez vers `http://your-server/swagger`
2. Parcourez les points de terminaison API disponibles
3. Cliquez sur n'importe quel point de terminaison pour développer ses détails

**Envoi d'une Requête :**

1. Cliquez sur le point de terminaison que vous souhaitez utiliser (par exemple, `/api/sri-for-sm`)
2. Cliquez sur le bouton "Essayez-le"
3. Remplissez les paramètres requis dans le corps de la requête
4. Cliquez sur "Exécuter"
5. Consultez la réponse ci-dessous

## Codes de Réponse API

- **200** - Succès, résultat retourné dans le corps de la réponse
- **400** - Mauvaise Requête, paramètres invalides
- **504** - Délai d'Attente de la Passerelle, pas de réponse du réseau SS7 dans les 10 secondes

---

# Métriques du Client MAP

## Métriques Disponibles

**Métriques de Requête :**

- `map_requests_total` - Nombre total de requêtes MAP envoyées
  - Étiquettes : `operation` (valeurs : `sri`, `sri_for_sm`, `prn`, `authentication_info`, etc.)
- `map_request_errors_total` - Nombre total d'erreurs de requêtes MAP
  - Étiquettes : `operation`

- `map_request_duration_milliseconds` - Histogramme des durées de requêtes MAP
  - Étiquettes : `operation`
- `map_pending_requests` - Nombre actuel de requêtes MAP en attente (gauge)

## Exemples de Requêtes Prometheus

```
# Total des requêtes SRI-for-SM au cours de la dernière heure
increase(map_requests_total{operation="sri_for_sm"}[1h])

# Temps de réponse moyen pour les requêtes SRI
rate(map_request_duration_milliseconds_sum{operation="sri"}[5m]) /
rate(map_request_duration_milliseconds_count{operation="sri"}[5m])

# Taux d'erreur pour toutes les opérations MAP
sum(rate(map_request_errors_total[5m])) by (operation)

# Requêtes en attente actuelles
map_pending_requests
```

---

# Dépannage du Client MAP

## Problème : Délai d'Attente des Requêtes

**Symptômes :**

- L'API retourne 504 Délai d'Attente de la Passerelle
- Pas de réponse du HLR/MSC

**Vérifications :**

1. Vérifiez que la connexion M3UA est ACTIVE :

```
# Dans la console IEx
:sys.get_state(:map_client_asp)
```

2. Vérifiez la connectivité réseau vers le STP
3. Vérifiez le contexte de routage et l'adressage SCCP
4. Vérifiez les journaux pour les erreurs SCCP

---

## Problème : Erreurs SCCP

**Symptômes :**

- L'API retourne des réponses d'erreur SCCP
- Les journaux montrent des messages "service unitdata SCCP"

**Codes d'Erreur SCCP Courants :**

- **Pas de Traduction** : Titre Global non trouvé dans la table de routage STP
- **Échec de Sous-système** : Sous-système de destination (HLR SSN 6) indisponible
- **Échec Réseau** : Congestion ou échec du réseau

**Solutions :**

- Contactez l'administrateur STP pour vérifier la configuration de routage
- Vérifiez si le Titre Global de destination est accessible
- Vérifiez si le sous-système de destination est opérationnel

---

# Documentation Connexe


- ← Retour à la Documentation Principale
- Guide des Fonctionnalités Communes - Interface Web, API, Surveillance
- Guide STP - Configuration de routage

- Guide du Centre SMS - Livraison de SMS
- Référence Technique - Spécifications de protocole

---

**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 tant que 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 tous les signaux de protocole 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 éviter la congestion du réseau
- **Conception Axée sur l'API** : API HTTP RESTful pour l'intégration avec les systèmes de facturation, les portails clients et les applications tierces
- **Analytique et 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

---

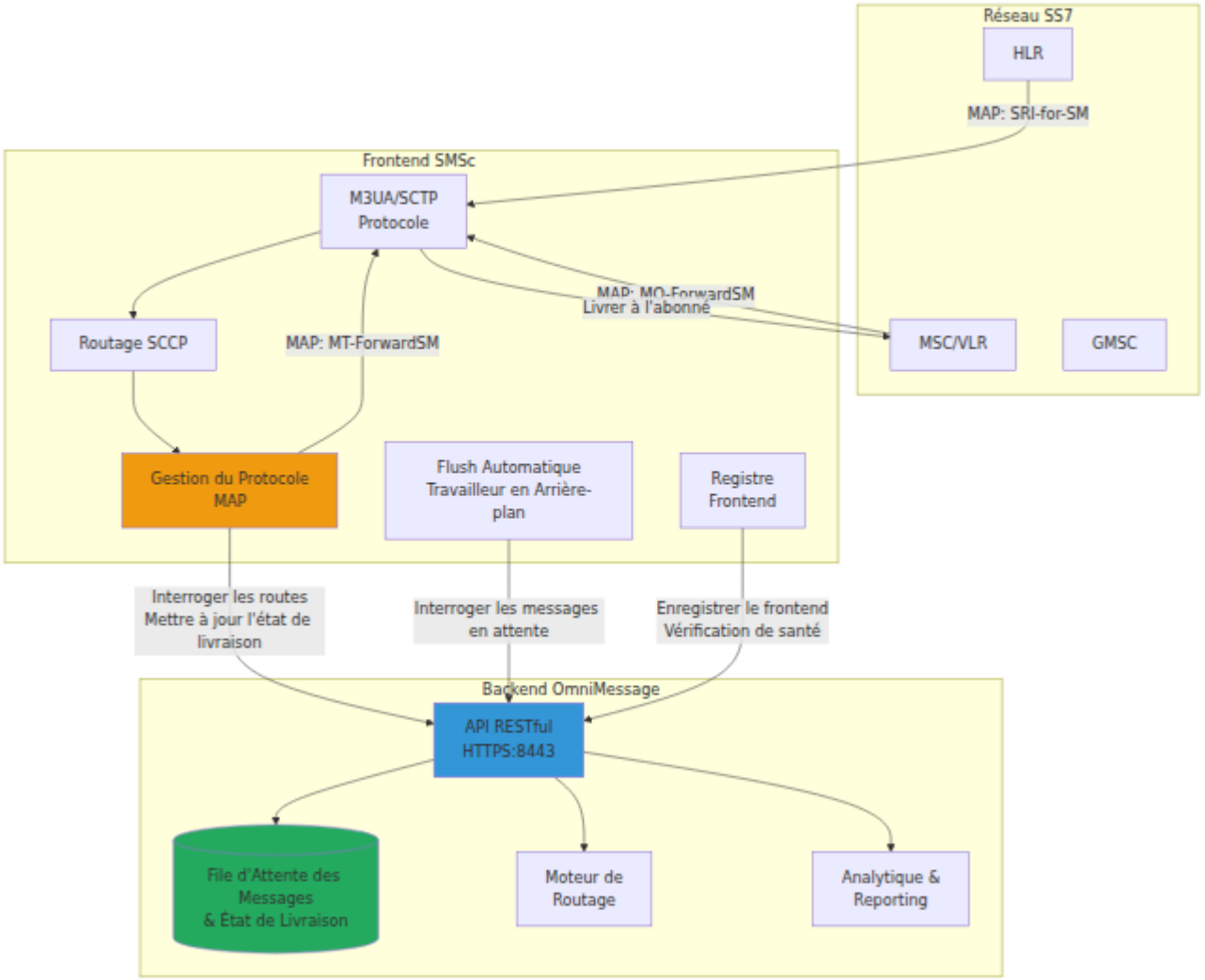
# Qu'est-ce que le Mode Centre SMS ?

**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. Pour l'utiliser comme un SMSc, vous devez activer le mode SMSc dans la configuration.

## Changement vers le Mode SMSc

Le fichier `config/runtime.exs` d'OmniSS7 contient trois modes opérationnels préconfigurés. Pour activer le mode SMSc :

1. **Ouvrir** `config/runtime.exs`
2. **Trouver** les trois sections de configuration (lignes 53-204) :

    - Configuration 1 : Mode STP (lignes 53-95)
    - Configuration 2 : Mode HLR (lignes 97-142)
    - Configuration 3 : Mode SMSc (lignes 144-204)
3. **Commenter** toute autre configuration active (ajouter `#` à chaque ligne)
4. **Décommenter** la configuration SMSc (retirer `#` des lignes 144-204)
5. **Personnaliser** les paramètres de configuration selon les besoins
6. **Redémarrer** l'application : `iex -S mix`

## Configuration du Mode SMSc

La configuration complète du SMSc ressemble à ceci :

```
config :omniss7,
  # Drapeaux de mode - Activer les fonctionnalités STP + SMSc
  # Remarque : map_client_enabled est vrai car 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://10.179.3.219:8443",
  # 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
  }

config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Événements SS7"},
    {SS7.Web.TestClientLive, "/client", "Client SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
```

```
    {SS7.Web.RoutingLive, "/routing", "Routage"},
    {SS7.Web.RoutingTestLive, "/routing_test", "Test de Routage"},
    {SS7.Web.SmscLinksLive, "/smsc_links", "Liens SMSc"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/routing",
"/routing_test", "/smsc_links", "/application", "/configuration"]
```

## 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` | String | *Requis* | **Point d termina** backend OmniMe |
| `smsc_name` | String | `" {hostname}_SMSc"` | Votre id pour l'e |
| `smsc_service_center_gt_address` | String | *Requis* | Titre Gl de Serv |
| `auto_flush_enabled` | Boolean | `true` | Activer automa d'attent |
| `auto_flush_interval` | Integer | `10_000` | Interval traiteme d'attent millisec |
| `auto_flush_dest_smsc` | String | *Requis* | Nom du destinat flush au |
| `auto_flush_tps` | Integer | `10` | Taux de des mes (transac |
| `local_ip` | Tuple | *Requis* | Adresse systèm |
| `local_port` | Integer | `2905` | Port SC |
| `remote_ip` | Tuple | *Requis* | Adresse connect |

| Paramètre | Type | Par défaut | De |
|---|---|---|---|
| `remote_port` | Integer | `2905` | Port SC |
| `routing_context` | Integer | `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
- **M3UA** - État de la connexion
- **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** - Visualiseur de configuration

L'onglet **Liens HLR** sera masqué.

## Notes Importantes

- Le mode SMSc nécessite `map_client_enabled: true` pour les capacités de routage
- **Backend OmniMessage** : L'API OmniMessage backend doit être accessible à l'URL `smsc_api_base_url` configurée
- **Enregistrement du Frontend** : Le système s'enregistre automatiquement avec OmniMessage toutes les **5 minutes** via le module `SMS.FrontendRegistry`

- **Délai d'Attente des Requêtes API** : Toutes les requêtes API OmniMessage ont un **délai d'attente de 5 secondes codé en dur**
- **Délai d'Attente des Requêtes MAP** : SRI-for-SM a un délai d'attente de dialogue de 10 secondes. MT-ForwardSM a un **délai d'attente 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 en tant que frontend actif :

```
config :omniss7,
  # URL de base de l'API OmniMessage
  smsc_api_base_url: "https://10.5.198.200:8443",
  # Identifiant du nom du SMSc pour l'enregistrement (par défaut
au 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` | String | Oui | `"https://localhost` |
| `smsc_name` | String | Non | `""` (utilise `"{hostname}_SMSc"`) |
| `smsc_service_center_gt_address` | String | Non | `"5551234567"` |

## Enregistrement du Frontend

Le système s'enregistre automatiquement avec OmniMessage au démarrage et **se réenregistre toutes les 5 minutes** via le module `SMS.FrontendRegistry`. Cela permet à OmniMessage de :

- Suivre les frontends actifs pour l'équilibrage de charge
- Surveiller la disponibilité et l'état de santé

- Collecter des informations de configuration
- Gérer le routage SMS distribué à travers plusieurs frontends

**Détails d'Implémentation :**

- **Intervalle d'Enregistrement** : 5 minutes (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 avec l'API 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 en tant que 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 la Requête |
|---|---|---|---|
| `/api/frontends` | POST | Enregistrer l'instance de frontend | `{"frontend_name": "...", "frontend_type": "SMSc", "hostname": "...", "uptime_seconds": ...}` |
| `/api/messages_raw` | POST | Insérer un nouveau message SMS | `{"source_msisdn": "...", "source_smsc": "...", "message_body": "..."}` |
| `/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_time": "...", "dest_smsc": "..."}` |
| `/api/messages/{id}` | PUT | Mettre à jour l'état du message | `{"dest_smsc": null}` |
| `/api/locations` | POST | Insérer/mettre à jour la localisation de l'abonné | `{"msisdn": "...", "imsi": "...", "location": "...", "ims_capable": true, "csfb": false, "expires": "...",` |

| Endpoint | Méthode | But | Corps de la Requête |
|---|---|---|---|
| | | | `"user_agent": "...", "ran_location": "...", "imei": "...", "registered": "..."}` |
| `/api/events` | POST | Ajouter le suivi des événements | `{"message_id": ..., "name": "...", "description": "..."}` |
| `/api/status` | GET | Vérification de santé | - |

# Format de Réponse de l'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 (ex. : `"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 client API principal fournissant toute la communication HTTP API avec OmniMessage :

- `frontend_register/4` - Enregistrer le frontend avec OmniMessage
- `insert_message/3` - Insérer un message SMS brut (version Python-compatible à 3 paramètres)
- `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 continue 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 5 minutes
- Utilise `smsc_name` de la config (retombe sur le nom d'hôte)
- Collecte des informations de configuration et de disponibilité du système

**3. SMS.Utils**

Fonctions utilitaires pour les opérations SMS :

- `generate_tp_scts/0` - Générer un horodatage SMS au format TPDU

---

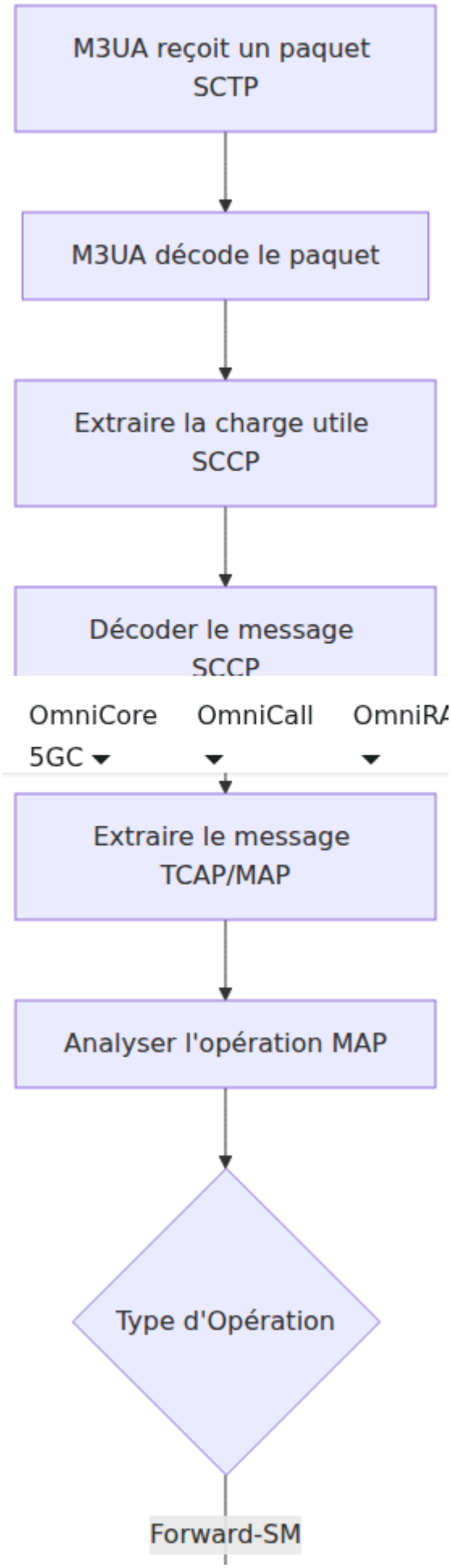
# Flux de Messages SMS

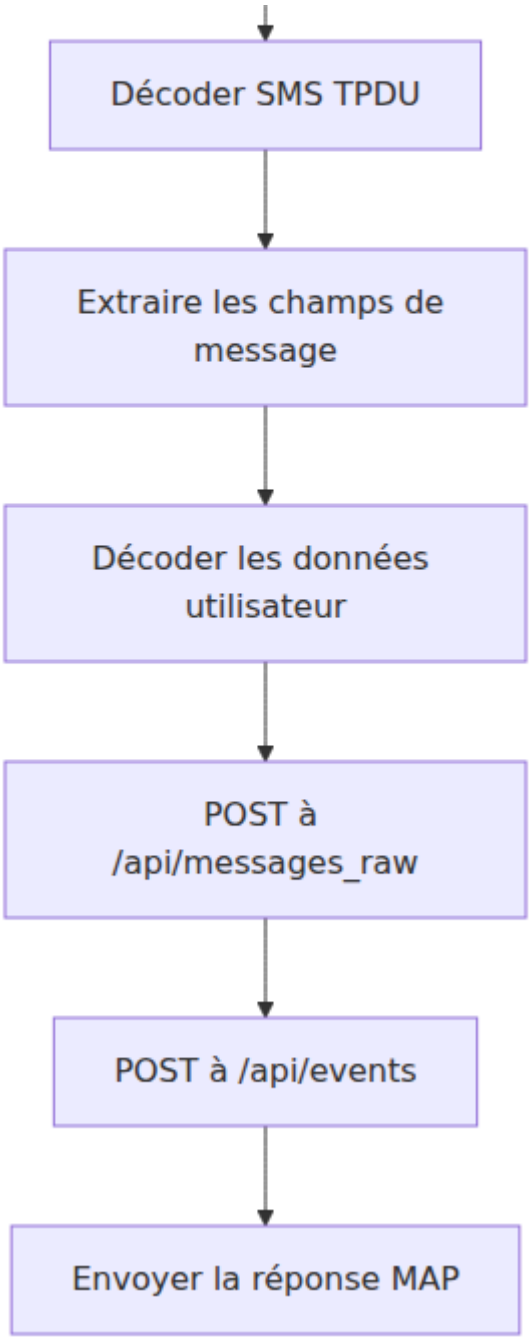
## Flux de SMS entrants (D'origine Mobile)


M3UA reçoit un paquet SCTP
M3UA décode le paquet
Extraire la charge utile SCCP
Décoder le message SCCP
Extraire le message TCAP/MAP
Analyser l'opération MAP
Type d'Opération
Forward-SM


## Flux de SMS sortants (Terminés sur Mobile)

Lorsqu'une API HSS/HLR est configurée (`hlr_api_base_url`), l'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 l'IMSI réel 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
Décoder 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 du 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. Consultez Mise en Cache des Adresses VLR pour plus de détails.

- **Livraison Directe via l'API HSS (sur réseau)** : Lorsque `hlr_api_base_url` est configuré, l'IP-SM-GW interroge l'API HSS avec le MSISDN de destination pour obtenir l'**IMSI réel** de l'abonné et l'**adresse**

**VLR de service**. Si les deux sont disponibles, le MT-ForwardSM est envoyé directement au VLR — contournant complètement SRI-for-SM.

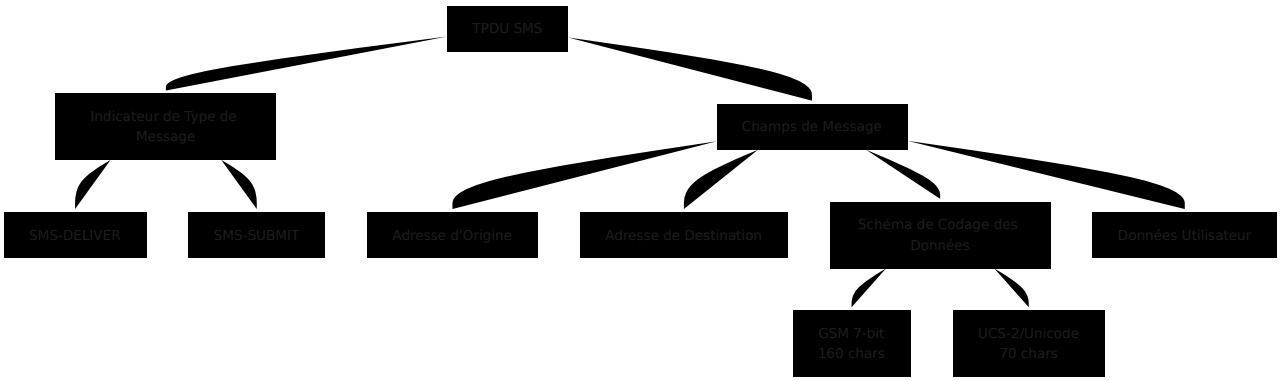
- **Abonné Absent (sur réseau)** : Si l'API HSS trouve l'abonné mais qu'aucun VLR de service n'est assigné, le message est marqué comme échoué avec une entrée de localisation expirée. Les réessais de livraison se produisent lorsque le HLR envoie un alertServiceCenter après que l'abonné se soit réenregistré.

- **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. Consultez le Guide SRI-for-SM dans HLR.

- **Effacement du VLR en cas d'Échec** : Lorsque la livraison 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 accommoder les réponses plus lentes des VLR 2G/CS.

## Structure du TPDU SMS




---

# Gestion du Service Center d'Alerte

Le SMSc peut recevoir des messages **alertServiceCenter** du HLR pour suivre l'état de disponibilité des abonnés.

Pour des informations sur la façon dont le HLR envoie des messages alertServiceCenter, consultez le Guide d'Intégration du Service Center d'Alerte dans HLR.

## Qu'est-ce que alertServiceCenter ?

Lorsqu'un abonné effectue une UpdateLocation au HLR (c'est-à-dire, s'enregistre avec un nouveau VLR/MSC), le 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 le 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 du message (format TBCD)
2. **Supprimer le préfixe TON/NPI** : Retirer les préfixes communs comme "19", "11", "91" (ex. : "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. L'utilisation de l'IMSI réel est critique pour la livraison MT aux abonnés 2G/CS sur le réseau — le MSC/VLR nécessite l'IMSI correct dans le MT-ForwardSM.
4. **POST à /api/location** : 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 (ex. : "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 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 HLR GT, status=actif, compteurs de messages à 0
6. **Envoyer ACK** : Répondre au HLR avec l'accusé de réception alertServiceCenter

## Gestion des Abonnés Absents

Lorsque le SMSc tente de livrer un message et reçoit une erreur "abonné absent" lors de SRI-for-SM (pour plus d'informations sur SRI-for-SM, consultez le Guide SRI-for-SM dans HLR) :

1. **Détecter l'absence** : SRI-for-SM retourne l'erreur `absentSubscriberDiagnosticSM`
2. **Expiration de la localisation** : POST à /api/location 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 tracker** : Marquer l'abonné comme `failed` dans le Suivi des Abonnés SMSc

Cela garantit que la base de données de localisation et le tracker reflètent avec précision l'état de disponibilité des abonnés.

# Diagramme de Flux

## Endpoint API

**POST /api/location**

```
{
  "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"
}
```

**Remarque** : Le champ `user_agent` contient le GT HLR qui a envoyé l'alertServiceCenter, permettant au SMSc de suivre quel HLR fournit des mises à jour de localisation.

Pour les abonnés absents, `expires` est défini sur l'heure actuelle (expiration immédiate).

---

# Prévention des Boucles

Le SMSc met en œuvre une **prévention automatique des boucles** pour éviter des boucles de routage de messages infinies lorsque les messages proviennent de réseaux SS7, tout en permettant une livraison légitime MO→MT entre abonnés sur le même réseau.

## Pourquoi la Prévention des Boucles est Importante

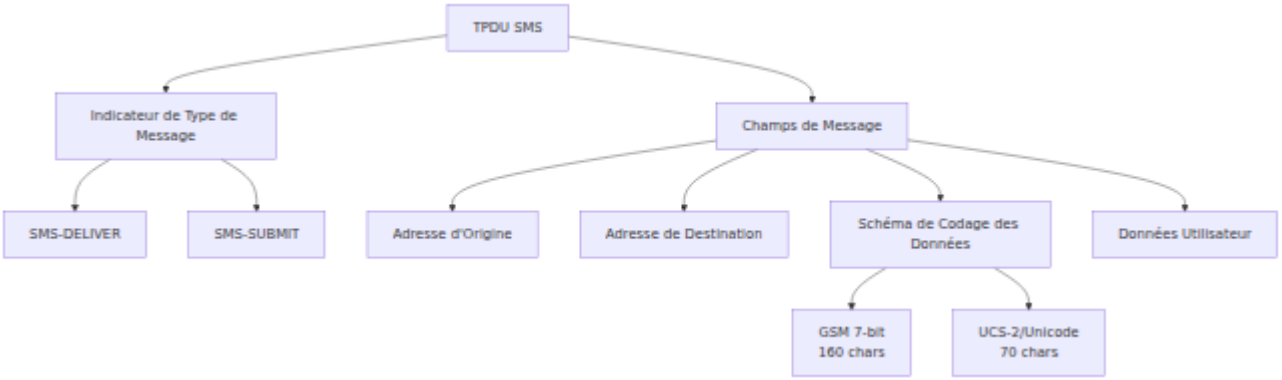
Lorsque le SMSc reçoit des messages SMS d'origine mobile (MO) du réseau SS7, il les insère dans la file d'attente de messages avec un champ `source_smsc`

identifiant leur origine (ex. : `"SS7_MO_15551234567"`). Sans prévention des boucles, 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 → Traités pour livraison
3. Renvoyés au réseau SS7 → Créant une boucle

## Comment Cela Fonctionne

La logique de prévention des boucles distingue entre les destinations **sur réseau** et **hors réseau**. Les messages provenant de SS7 sont uniquement bloqués lorsque l'abonné de destination n'est **pas sur réseau** — cela empêche les boucles de transit tout en permettant une livraison légitime MO→MT entre abonnés sur le même réseau (ex. : MSC à MSC via l'IP-SM-GW).




## Détection sur Réseau

Lorsqu'un message provient de SS7, le système vérifie si l'abonné de destination est sur réseau avant d'appliquer la prévention des boucles :

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 le HSS, il est sur réseau.

Si aucune des vérifications ne trouve l'abonné, la destination est considérée comme **hors réseau** et le message est bloqué.

# Implémentation

Lors du traitement des messages de la file d'attente, le SMSc vérifie le champ `source_smsc` :

- **Si `source_smsc` contient "SS7" et que la destination est hors réseau** :
  - Le message est passé
  - Événement ajouté : `"Prévention des Boucles"` avec une description notant que la destination est hors réseau
  - Message marqué comme échoué via la requête PUT
  - Journalisé avec un niveau d'avertissement
- **Si `source_smsc` contient "SS7" et que la destination est sur réseau** :
  - Message traité normalement — il s'agit d'un flux légitime MO→MT
  - La livraison se poursuit via le chemin API HSS / cache VLR
- **Si `source_smsc` ne contient pas "SS7"** :
  - Message traité normalement (ex. : messages provenant de l'API web, SMPP ou d'autres sources non-SS7)

# Valeurs Source SMSc

Les messages peuvent avoir diverses valeurs `source_smsc` :

| Source | Exemple de Valeur | Action |
|---|---|---|
| Réseau SS7 (MO-FSM), dest hors réseau | `"SS7_MO_15551234567"` | **Passé** - Prévention des boucles |
| Réseau SS7 (MO-FSM), dest sur réseau | `"SS7_MO_15551234567"` | **Autorisé** - Livraison MO→MT sur réseau |
| API Externe/SMPP | `"ipsmgw"` ou `"api_gateway"` | Traité normalement |
| Autre SMSc | `"smsc-node-01"` | Traité normalement |

## Suivi des Événements

Lorsqu'un message est passé en raison de la prévention des boucles, un événement est enregistré :

```
{
  "message_id": 12345,
  "name": "Prévention des Boucles",
  "description": "Message passé - source_smsc 'SS7_MO_15551234567' contient 'SS7' et la destination est hors réseau, empêchant la boucle de message"
}
```

Cet événement est visible dans :

- **Interface Web** : Page des Événements SS7 (`/events`)
- **Base de Données** : Table `events` via l'API
- **Journaux** : Entrées de journal de niveau d'avertissement

## Configuration

La prévention des boucles est **toujours activée** et ne peut pas être désactivée. C'est une fonctionnalité de sécurité critique pour éviter la

perturbation du réseau due aux boucles de messages. Le contournement sur 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 abonné 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 : source_smsc =
"SS7_MO_2471900"
3. Le flush automatique récupère le message de la file
4. Destination non dans le cache VLR ou HSS → hors réseau
5. IP-SM-GW détecte "SS7" dans source_smsc + dest hors réseau →
PASSER
6. Événement enregistré : "Prévention des Boucles"
7. Aucun SRI-for-SM ou MT-ForwardSM envoyé (boucle empêchée)
```

**Scénario 2** : MO-SMS de SS7 vers un abonné sur réseau (autorisé)

```
1. Téléphone mobile → MSC/VLR → IP-SM-GW (via MO-ForwardSM)
2. IP-SM-GW reçoit MO-FSM, insère dans la file : source_smsc =
"SS7_MO_2471900"
3. Le flush automatique récupère le message de la file
4. Destination trouvée dans l'API HSS → abonné sur réseau
5. IP-SM-GW autorise la livraison malgré l'origine SS7
6. HSS retourne IMSI réel + VLR → MT-ForwardSM envoyé directement
au VLR
```

---

# Suivi des Abonnés SMSc

Le SMSc comprend un GenServer de **Suivi des Abonnés** qui maintient l'état en temps réel des abonnés basé sur les messages alertServiceCenter et les tentatives de livraison de messages.

# Objectif

Le tracker fournit :

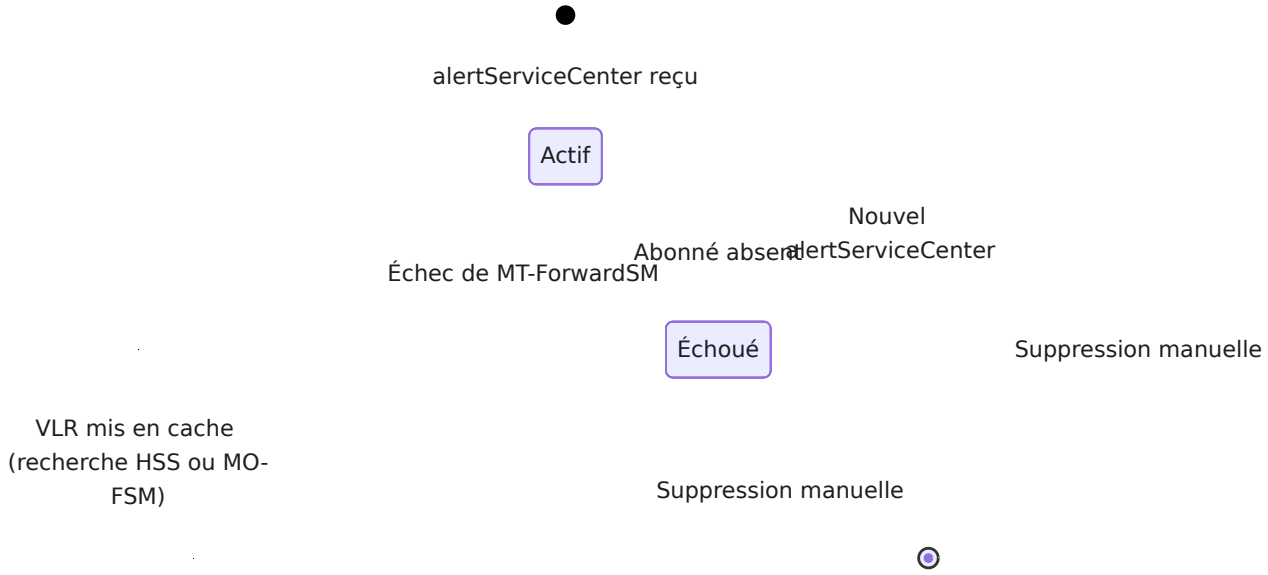
- **Surveillance de la disponibilité** : Quels abonnés sont actuellement joignables
- **Suivi HLR** : Quel HLR a envoyé l'alertServiceCenter pour chaque abonné
- **Compteurs de messages** : Nombre de messages envoyés/reçus par abonné
- **Suivi des échecs** : Marquer les abonnés comme échoués lorsque les tentatives de livraison échouent
- **Visibilité dans l'Interface Web** : Tableau de bord en temps réel montrant tous les abonnés suivis

# Informations Suivies

Pour chaque abonné, le tracker stocke :

| Champ | Description | Exemple |
|---|---|---|
| `msisdn` | Numéro de téléphone de l'abonné (clé) | "15551234567" |
| `imsi` | IMSI de l'abonné (réel du HSS si disponible) | "001010123456789" |
| `hlr_gt` | HLR GT qui a envoyé alertServiceCenter | "15551111111" |
| `messages_sent` | Compte des messages MT-FSM envoyés | 5 |
| `messages_received` | Compte des messages MO-FSM reçus | 2 |
| `status` | `:active` ou `:failed` | `:active` |
| `updated_at` | Horodatage Unix de la dernière mise à jour | 1730246400 |
| `vlr_address` | Adresse VLR GT 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

### Lorsque alertServiceCenter est reçu :

- Créer ou mettre à jour l'entrée de l'abonné
- Définir `status = :active`
- Enregistrer HLR GT
- Résoudre l'IMSI réel à partir de l'API HSS (si configurée), 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 l'IMSI réel dans le tracker
- Les livraisons suivantes utilisent le VLR mis en cache sans interroger à nouveau le HSS

### Lorsque MO-ForwardSM est reçu :

- Mettre en cache le GT appelant SCCP comme adresse VLR de l'abonné (le GT appelant sur un MO-ForwardSM est le VLR qui a relayé le message)
- Crée une entrée de tracker si elle n'existe pas

**Lorsque la livraison MT-ForwardSM échoue :**

- Effacer l'adresse VLR mise en cache pour que la prochaine tentative de livraison interroge à nouveau l'API HSS

**Lorsque SRI-for-SM réussit :**

- Incrémenter le compteur `messages_sent`
- Mettre ﷺ jour l'horodatage `updated_at`

**Lorsque SRI-for-SM échoue :**

- Définir `status = :failed`
- Garder dans le tracker 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 lorsque le mode SMSc est activé. Après avoir décommenté la configuration SMSc dans `config/runtime.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, HLR GT, VLR GT (adresse VLR de service mise en cache)
    - Compteurs de messages envoyés/reçus
    - Badge de statut (Actif/Échoué) avec codage couleur

- Horodatage de la 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              HLR GT         VLR GT         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 tracker 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 tracker 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 des Adresses VLR

Lorsqu'il fonctionne comme un IP-SM-GW avec une API HSS configurée, le Suivi des Abonnés met en cache les adresses VLR pour éviter d'interroger le 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 adresse VLR de l'abonné.

Lors des tentatives de livraison MT suivantes, le cache est vérifié en premier. Si une entrée valide existe (IMSI et VLR présents, dans le TTL), l'API HSS est complètement contournée.

## Invalidation 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** — configurable via `vlr_cache_ttl_seconds`, les entrées plus anciennes que le TTL sont considérées 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.

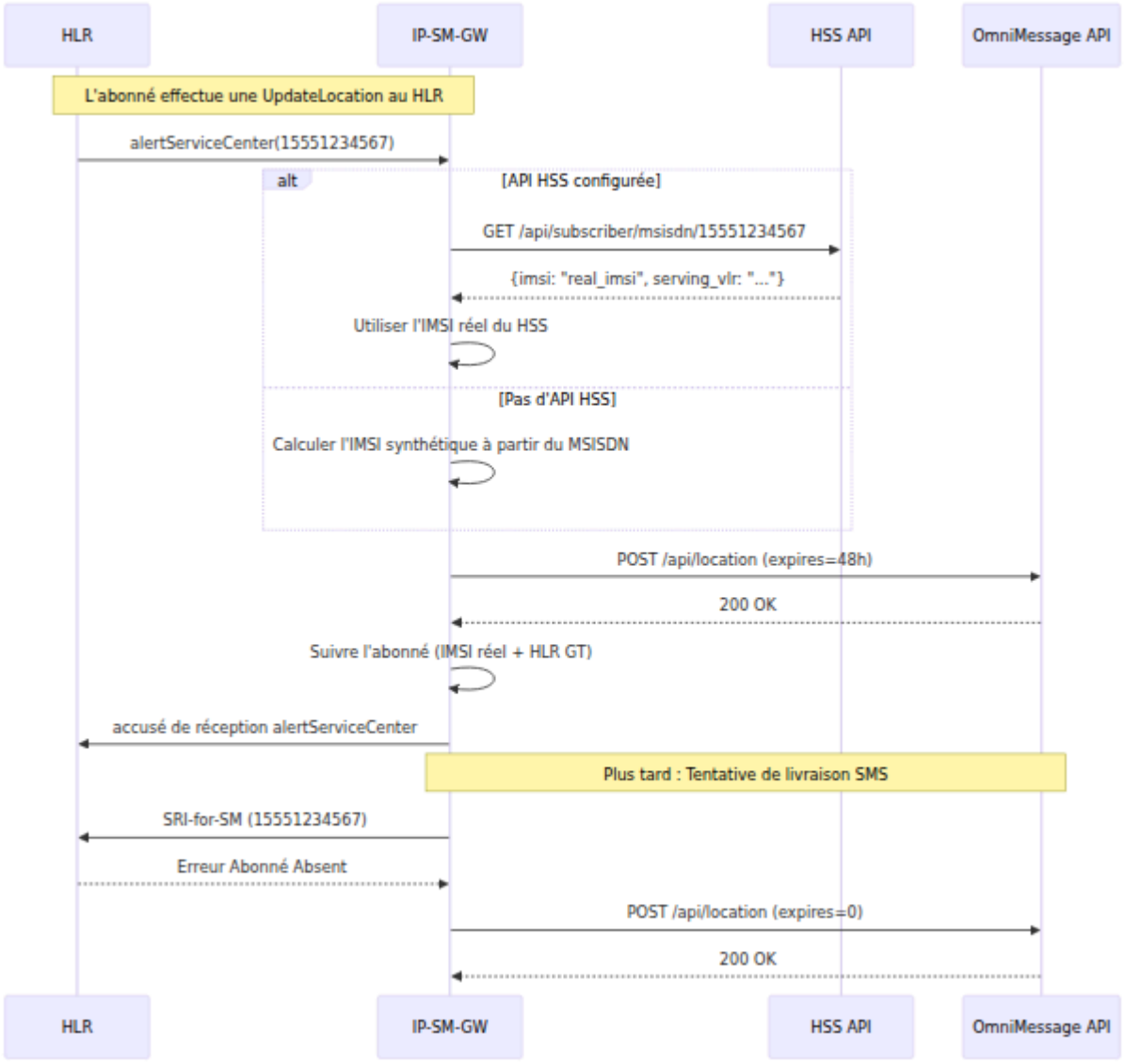
## Configuration

```
config :omniss7,
  # Point de terminaison API HSS/HLR (requis pour la mise en cache
VLR et la livraison sur réseau)
  hlr_api_base_url: "https://10.179.2.140:8443",

  # TTL de cache VLR en secondes (par défaut : 3600 = 1 heure)
  vlr_cache_ttl_seconds: 3600
```

| Paramètre | Type | Requis | Par défaut | Description |
| --- | --- | --- | --- | --- |
| `hlr_api_base_url` | String | Non | `nil` | URL de base de l'API HSS/HLR. Lorsqu'elle est définie, elle active la recherche d'abonnés sur réseau, la mise en cache VLR et la livraison directe MT. Lorsque `nil`, toutes les livraisons utilisent le flux SRI-for-SM. |
| `vlr_cache_ttl_seconds` | Integer | Non | `3600` | Âge maximum en secondes pour qu'une entrée VLR mise en cache soit considérée comme valide. Après cette période, la prochaine tentative de livraison interroge à nouveau l'API HSS. Des valeurs plus basses augmentent le |

| Paramètre | Type | Requis | Par défaut | Description |
|---|---|---|---|---|
| | | | | trafic API HSS mais améliorent l'exactitude pour les abonnés mobiles. |

## Diagramme de Séquence



# Configuration de Flush Automatique de la File d'Attente SMS

Le service **Flush Automatique** traite automatiquement les messages SMS en attente.

Pour référence des paramètres de configuration, consultez Configuration de Flush Automatique dans la Référence de Configuration.

## Configuration

```
config :omniss7,
  auto_flush_enabled: true,           # Activer/désactiver le
flush automatique
  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** (Envoyer les Informations de Routage pour Message Court) au HLR pour obtenir des informations de routage
        - Le HLR retourne un IMSI synthétique calculé à partir du MSISDN
        - Le HLR retourne l'adresse GT du SMSc où le MT-ForwardSM doit être envoyé
        - Consultez Détails SRI-for-SM dans le Guide HLR pour une documentation complète

- En cas de succès, envoyer **MT-forwardSM** au MSC/VLR
- Mettre à jour l'état du message via l'API (livré/échoué)
- 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 à IMSI, la configuration de l'adresse GT du centre de service, et la génération d'IMSI synthétique préservant la vie privée, consultez la section SRI-for-SM dans le Guide de Configuration HLR.

---

# Métriques SMSc

## Métriques Disponibles

### Métriques de la File d'Attente SMS :

- `smsc_queue_depth` - Nombre actuel de messages en attente
- `smsc_messages_delivered_total` - Total des messages livrés avec succès
- `smsc_messages_failed_total` - Total des messages ayant échoué à la livraison
- `smsc_delivery_duration_milliseconds` - Histogramme des temps de livraison

### Exemples de Requêtes :

```
# Profondeur de file d'attente actuelle
smsc_queue_depth

# Taux de succès de livraison (dernières 5 minutes)
rate(smsc_messages_delivered_total[5m]) /
(rate(smsc_messages_delivered_total[5m]) +
rate(smsc_messages_failed_total[5m]))

# Temps de livraison moyen
rate(smsc_delivery_duration_milliseconds_sum[5m]) /
rate(smsc_delivery_duration_milliseconds_count[5m])
```

---

# Dépannage SMSc

## Problème : Messages Non Livrés

**Vérifications :**

1. Vérifiez que le flush automatique est activé
2. Vérifiez la connexion à la base de données
3. Surveillez les journaux pour les 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
- Passez en revue les journaux de messages échoués

---

# API MT-forwardSM

## Envoyer un SMS via l'API

**Endpoint API :** `POST /api/MT-forwardSM`

**Requête :**

```
{
  "imsi": "234509876543210",
  "destination_serviceCentre": "447999555111",
  "originating_serviceCenter": "447999123456",
  "smsPDU":
"040B917477218345F60000157030185714 0C0BD4F29C0E9281C4E1F11A"
}
```

**Réponse :**

```
{
  "result": "success",
  "message_id": "12345"
}
```

---

# 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 à IMSI et la configuration du centre de service
- 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 de livraison, la limitation de taux et l'analytique, reportez-vous à la **documentation produit d'OmniMessage**. OmniMessage contient toute la logique de routage des messages, les algorithmes de réessai de la file d'attente, la gestion des rapports de livraison et le moteur de règles commerciales.

# Guide de Configuration M3UA & M2PA STP

← Retour à la Documentation Principale

Ce guide fournit des instructions détaillées pour utiliser OmniSS7 en tant que **Point de Transfert de Signalisation (STP)**.

## Table des Matières

---

# Qu'est-ce qu'un Point de Transfert de Signalisation (STP) ?

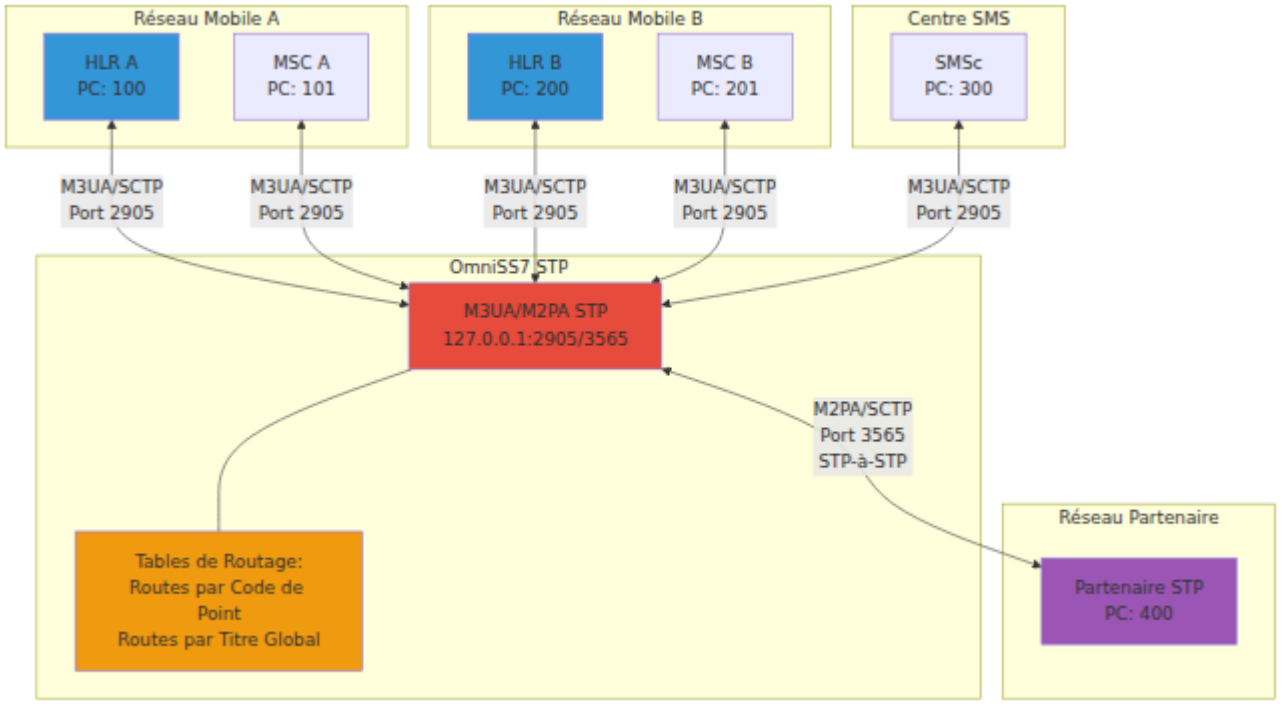
Un **Point de Transfert de Signalisation (STP)** est un élément critique du réseau dans les réseaux de signalisation SS7 et basés sur IP qui achemine les messages de signalisation entre les nœuds du réseau.

## Fonctions du STP

- **Routage de Messages** : Achemine le trafic de signalisation SS7 en fonction du Code de Point de Destination (PC) ou du 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 les 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
- **Objectif** : 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 relier les deux mondes.

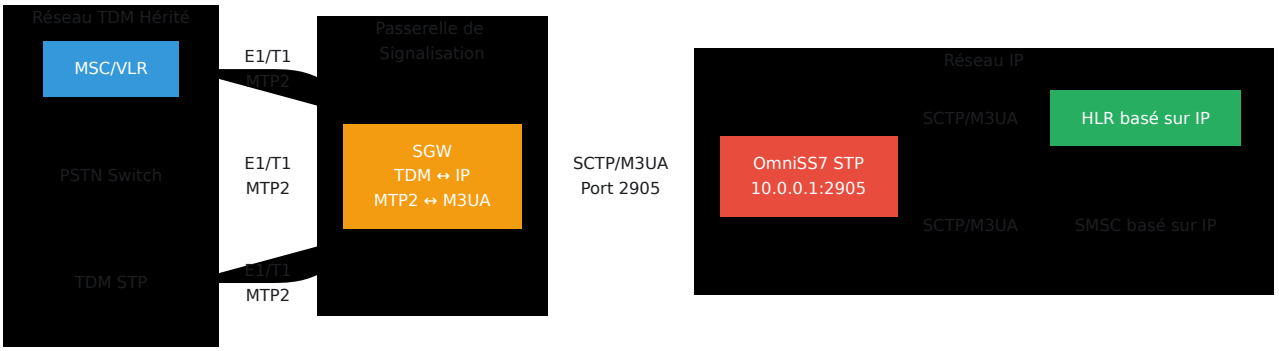
## Qu'est-ce qu'une Passerelle de Signalisation ?

Une **Passerelle de Signalisation (SGW)** est un élément de 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 en tant que traducteur de protocole, permettant 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 à 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 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 |
|---|---|---|
| **Remote IP** | Adresse IP d'OmniSS7 | Adresse IP de votre serveur |
| **Remote Port** | Port SCTP d'OmniSS7 | 2905 |
| **Routing Context** | Identifiant AS à la SGW | Assigné par l'administrateur SGW |
| **Point Codes** | Codes de point du réseau TDM accessibles via SGW | Spécifique au réseau |
| **Traffic Mode** | 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 en tant que STP entre IP et TDM

OmniSS7 achemine le trafic entre les éléments de réseau basés sur IP et les réseaux TDM :

```
IP SMSC → OmniSS7 STP → SGW → HLR TDM
                ↓
          HLR basé sur IP
```

### Scénario 3 : Dual SGW 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 primaire 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 STP, vous devez activer le mode STP dans la configuration.

# Changement au Mode STP

Le fichier `config/runtime.exs` d'OmniSS7 contient trois modes opérationnels préconfigurés. Pour activer le mode STP :

1. **Ouvrir** `config/runtime.exs`
2. **Trouver** les trois sections de configuration (lignes 53-174) :
   - Configuration 1 : Mode STP (lignes 53-85)
   - Configuration 2 : Mode HLR (lignes 87-123)
   - Configuration 3 : Mode SMSc (lignes 125-174)
3. **Commenter** la configuration actuellement active (ajouter `#` à chaque ligne)
4. **Décommenter** la configuration STP (retirer `#` des lignes 53-85)
5. **Personnaliser** les paramètres de configuration selon les besoins
6. **Redémarrer** l'application : `iex -S mix`

# Configuration du Mode STP

La configuration complète du STP ressemble à ceci :

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

  # Configuration de Connexion M3UA
  # Se connecter en tant qu'ASP (Processus de Serveur
d'Application) au STP/SGW distant
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    # Point de terminaison local (ce système)
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    # Point de terminaison STP/SGW distant
    remote_ip: {10, 179, 4, 11},
    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 | Description | Exemple |
|---|---|---|---|---|
| `map_client_enabled` | Booléen | `true` | Activer les capacités de client MAP et de routage | `true` |
| `local_ip` | Tuple ou Liste | *Requis* | Adresse(s) IP de votre système. Unique : `{10, 0, 0, 1}` ou Liste pour multihoming : `[{10, 0, 0, 1}, {10, 0, 0, 2}]` | `{10, 179, 4, 10}` |
| `local_port` | Entier | `2905` | Port SCTP local | `2905` |
| `remote_ip` | Tuple ou Liste | *Requis* | Adresse(s) IP de la STP/SGW distante. Unique ou Liste pour multihoming | `{10, 179, 4, 11}` |
| `remote_port` | Entier | `2905` | Port SCTP distant | `2905` |
| `routing_context` | Entier | `1` | Identifiant de contexte de routage M3UA | `1` |
| `enable_gt_routing` | Booléen | `false` | Activer le routage par Titre Global (en plus du routage par PC) | `true` |

> **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 le Guide de Multihoming SCTP.

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

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

- **Événements SS7** - Journalisation des événements
- **Client SS7** - Tests de fonctionnement MAP
- **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

Les onglets **Liens HLR** et **Liens SMSc** seront masqués.

## 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 du 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 changé de modes, 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

En plus des capacités de routage STP disponibles lorsque `map_client_enabled: true`, vous pouvez exécuter un **serveur STP M3UA autonome** qui écoute les connexions entrantes.

## Activation du STP Autonome

Ajoutez cette configuration à `config/runtime.exs` :

```
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 propre à ce STP
  }
```

## Paramètres de Configuration STP

| Paramètre | Type | Par Défaut | Description | Exemple |
|---|---|---|---|---|
| `enabled` | Booléen | `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` | Entier | `2905` | Port à écouter | `2905` |
| `point_code` | Entier | *Requis* | Code de point SS7 propre à ce STP | `100` |

## Quand Utiliser le STP Autonome

- **Routage Pur** : Lorsque vous avez uniquement besoin de routage M3UA sans fonctionnalité de client MAP
- **STP Central** : Pour créer un routeur de signalisation central pour plusieurs éléments de 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 par Code de Point et routes par 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** : `m3ua_peer`, `m3ua_route`, `m3ua_gt_route`

## 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'exploitation.

---

# 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 prise en charge pour des raisons de compatibilité.

```
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 = attribution
dynamique du port
      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 du 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 unique ou liste pour un pair distant multihomé. **Rôle Serveur** : Seulement un tuple - 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 de n'importe quel 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 liste pour `remote_ip` est uniquement utilisé 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 de ce port source exact
  - Fournit une sécurité supplémentaire en validant le port source
  - Utiliser lorsque le pair distant utilise un port source fixe
- **N'importe quel Port** (`remote_port: 0`) : Accepter les connexions de n'importe quel port source

- Utile lorsque le pair distant utilise des ports source dynamiques/éphémères
- Valide uniquement l'adresse IP source
- Plus flexible mais légèrement moins sécurisé

**Exemple :**

```
# Accepter uniquement de 10.5.198.200:2905 (port spécifique)
%{
  peer_id: 1,
  name: "Strict_Peer",
  role: :server,
  remote_ip: {10, 5, 198, 200},
  remote_port: 2905,
  # ... autre config
}

# Accepter de 10.5.198.200 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 n'importe quel port source
  # ... autre config
}
```

---

# Support du Protocole M2PA

OmniSS7 prend en charge à la fois 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 normalisé par l'IETF (RFC 4165) pour transporter les messages MTP3 SS7 sur des réseaux IP utilisant SCTP.

# M3UA vs M2PA : Différences Clés

| Fonctionnalité | M3UA | M2PA |
|---|---|---|
| **Architecture** | Client/Serveur (ASP/SGW) | Pair-à-Pair |
| **Cas d'Utilisation** | Passerelle entre SS7 et IP | Liaisons directes point à point |
| **Gestion de l'État de Liaison** | Niveau application (ASPUP/ASPAC) | Style MTP2 (Alignement, Test, Prêt) |
| **Numéros de Séquence** | Pas de séquençage inhérent | BSN/FSN 24 bits pour livraison ordonnée |
| **Déploiement Typique** | Passerelle SS7 vers IP, STP | Liaisons de signalisation directes entre nœuds |
| **RFC** | RFC 4666 | RFC 4165 |

# Conseils de Sélection de Protocole

**Recommandation : Utilisez M3UA par défaut. N'utilisez M2PA que lorsque 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** : Relier les réseaux SS7 avec la signalisation basée sur IP
- **Connexions d'Éléments de Réseau** : Connecter HLR, MSC, SMSC et d'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** : Norme de l'industrie largement supportée (RFC 4666)

**Utilisez M3UA pour connecter des éléments de réseau (HLR, MSC, SMSC, VLR, etc.) à votre STP.**

# Quand Utiliser M2PA (Uniquement dans des Cas Spécifiques)

M2PA ne doit être utilisé que dans des scénarios spécifiques :

- **Liaisons STP à STP** : Connexions directes point à point entre Points de Transfert de Signalisation dans un réseau multi-STP
- **Remplacement TDM Hérité** : Remplacer des liaisons SS7 TDM traditionnelles 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 liaison de style MTP2
- **Exigence Partenaire** : Lorsqu'un partenaire ou une interconnexion exige spécifiquement le protocole M2PA

**Important** : Ne pas utiliser M2PA pour connecter des éléments de réseau (HLR, MSC, SMSC) à votre STP - utilisez M3UA à la place. 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 de Pair 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 norme de l'industrie (différent du port 2905 de M3UA).

# Configuration SLTM

**SLTM (Message de Test de Liaison de Signalisation)** et **SLTA (Accusé de Réception de Test de Liaison de Signalisation)** sont des messages de maintenance MTP3 utilisés pour vérifier la connectivité de bout en bout après qu'une liaison M2PA atteigne l'état PRÊT. Selon l'ITU-T Q.707, un côté envoie SLTM et l'autre répond avec SLTA pour confirmer que la liaison est opérationnelle.

### Comportement SLTM

Le paramètre `send_sltm` contrôle quel côté initie le test SLTM :

| Valeur `send_sltm` | Comportement |
|---|---|
| `true` | Nous envoyons SLTM lorsque la liaison devient PRÊTE, attendons SLTA |
| `false` | Nous attendons que le pair envoie SLTM, répondons avec SLTA |
| Non défini (par défaut) | Suit Q.707 : le répondant SCTP envoie SLTM, l'initiateur SCTP attend |

### Comportement par Défaut (Conforme à Q.707) :

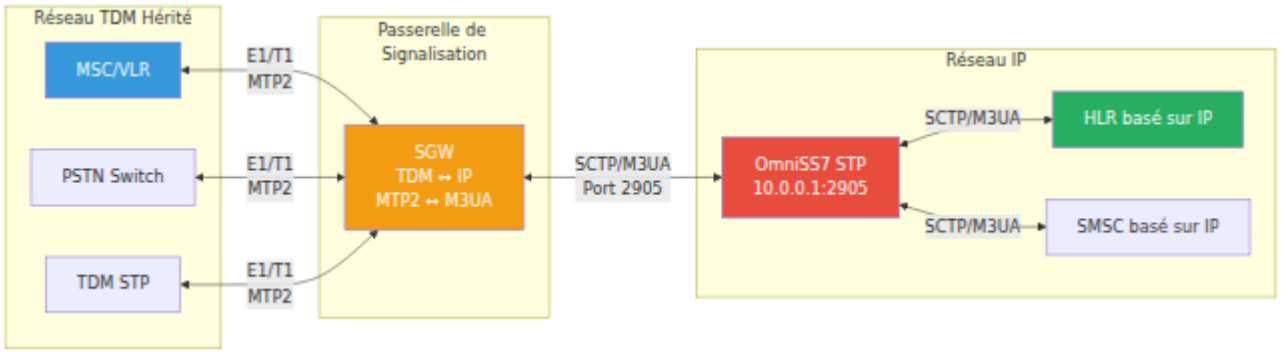
- Si `initiate_connection: true` (initiateur/client SCTP) → Nous attendons que le pair envoie SLTM
- Si `initiate_connection: false` (répondant/server SCTP) → Nous envoyons SLTM

### Quand Surcharger les Valeurs par Défaut de SLTM

Surchargez le comportement par défaut lors de la connexion à des équipements qui ne suivent 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  # Surcharge : Nous envoyons SLTM même si nous
sommes l'initiateur SCTP
}
```

### Flux de Messages SLTM




### Dépannage des Problèmes SLTM

#### Symptôme : La liaison 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 DPC erroné)
2. Mismatch de l'indicateur de réseau (`:international` vs `:national`)
3. Le pair distant n'est pas configuré pour répondre à notre code de point

**Résolution :** Vérifiez que `adjacent_point_code` correspond au code de point de STP (configuré dans `sctp_handler.point_code`).

**États de Liaison M2PA**

Les liaisons M2PA progressent à travers plusieurs états lors de l'initialisation :

1. **Down** - Aucune connexion établie
2. **Alignment** - Phase de synchronisation initiale (~1 seconde)
3. **Proving** - Vérification de la qualité de la liaison (~2 secondes)
4. **Ready** - Liaison active et prête pour le trafic

La progression des états de liaison garantit une signalisation fiable avant l'échange de trafic.

# Gestion des Pairs M2PA via l'Interface Web

La page **Routage** dans l'Interface Web fournit un support complet pour la gestion des pairs M2PA :

1. **Naviguer** vers la page de 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 "Sauvegarder le Pair"

La table 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 par Code de Point** : Fonctionnent de manière identique pour M2PA et M3UA
- **Routes par Titre Global** : Entièrement prises en charge sur les liaisons 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 des liaisons et le trafic :

**Métriques de Trafic :**

- `m2pa_messages_sent_total` - Total des messages MTP3 envoyés par liaison
- `m2pa_messages_received_total` - Total des messages MTP3 reçus par liaison
- `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 Liaison :**

- `m2pa_link_state_changes_total` - Transitions d'état de liaison (DOWN → ALIGNMENT → PROVING → READY)
  - 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`

**Accès aux Métriques :**

- Point de terminaison Prometheus : `http://your-server:8080/metrics`
- Les métriques s'enregistrent automatiquement au démarrage de l'application

# Meilleures Pratiques M2PA

1. **Sélection de Port** : Utilisez le port 3565 pour M2PA (norme de l'industrie)
2. **Surveillance de Liaison** : Surveillez les changements d'état de liaison 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. **Tests** : Utilisez la page de Test de Routage pour vérifier la connectivité après 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 que les pairs M2PA ne démarrent

**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 `unordered` flag)
- PPID 5 (identifiant M2PA selon RFC 4165)

---

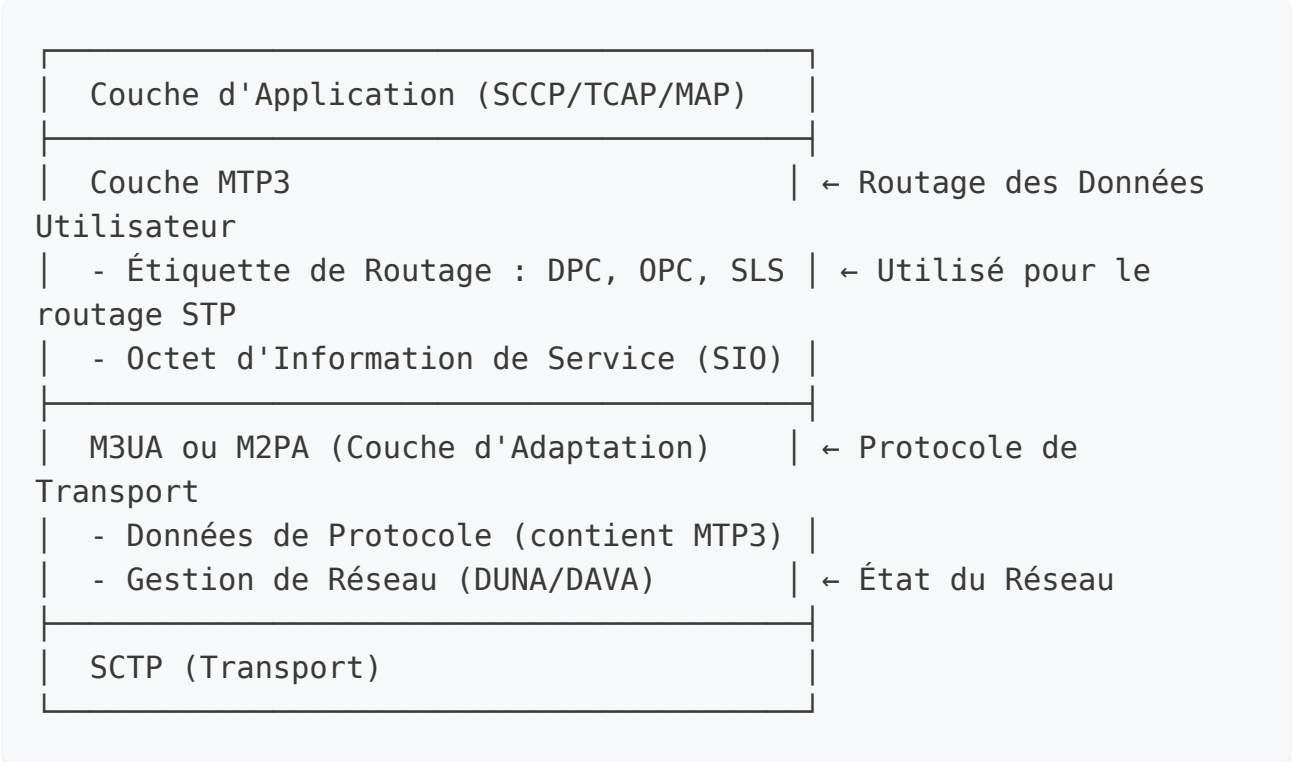
# Configuration du Routage par Code de Point

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 Protocole SS7

Les codes de point existent à différentes couches de la pile de protocole SS7. Comprendre cette distinction est important :

**Couches de la Pile de Protocole :**




**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 Données de Protocole M3UA (tag 528)
    - Présents dans les messages de Données Utiles M2PA

    - **Le STP utilise ces valeurs DPC pour les décisions de routage**
    - Elles 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/inaccessibles
    - 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 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 Utiles 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 le statut de signalisation entre pairs.

# Routes de Code de Point de Base

Ajoutez des routes à `config/runtime.exs` :

```
config :omniss7,
  m3ua_routes: [
    # Route tout le trafic pour le PC 100 vers le pair 1 (STP
Partenaire)
    %{
      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)
    },

    # Route 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). 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 Données Utiles
2. Le STP extrait le **message MTP3** du paramètre Données de Protocole (M3UA) ou du champ de Données Utiles (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 pour un 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 Données Utiles pour le pair de destination
7. Route le message vers le pair correspondant
8. Si le pair sélectionné est hors service, essaie la route de priorité suivante

## Masques de Code de Point

Les codes de point sont des valeurs de 14 bits (plage 0-16383). Par défaut, les routes correspondent à un code de point unique exactement (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 PC unique)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  network_indicator: :international
}
# Masque par défaut à 14 - Correspond à : Seulement le PC 1000

# Masque explicite (même résultat)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  mask: 14,                              # Correspondance exacte
explicite
  network_indicator: :international
}
# Correspond à : Seulement le PC 1000
```

**Exemple 2 : Petite Plage**

```
%{
  dest_pc: 1000,
  peer_id: 2,
  priority: 1,
  mask: 12,                              # Correspond à 4 PCs
  network_indicator: :international
}
# Correspond à : 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 à : PC 1000-1063 (64 codes de point consécutifs)
```

**Exemple 4 : Route par Défaut / de Secours**

```
%{
  dest_pc: 0,
  peer_id: 4,
  priority: 10,                           # Basse priorité (numéro
élevé)
  mask: 0,                                # Correspond à tous les
PCs
  network_indicator: :international
}
# Correspond à : Tous les codes de point (0-16383)
# Utiliser comme route de secours/catch-all avec faible priorité
```

**Combinaison de Routes Spécifiques et 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
      # masque par défaut à 14 (correspondance exacte)
    },

    # Route de plage pour les PCs 1000-1063
    %{
      dest_pc: 1000,
      peer_id: 2,
      priority: 1,
      mask: 8,                          # Correspond à 64 PCs
      network_indicator: :international
    },

    # Route par défaut pour tous les autres PCs
    %{
      dest_pc: 0,
      peer_id: 3,
      priority: 10,                     # Basse priorité
      mask: 0,                          # Correspond à tous les
PCs
      network_indicator: :international
    }
  ]
```

**Décision de Routage pour le DPC 1000 :**

1. Correspond à la route masque `/14` (PC 1000 exactement) - **Sélectionnée** (la plus spécifique)
2. Correspond également à la route masque `/8` (plage PC 1000-1063) - Ignorée (moins spécifique)
3. Correspond également à la route masque `/0` (tous les PCs) - Ignorée (moins spécifique)

**Décision de Routage pour le DPC 1015 :**

1. Ne correspond pas à la route masque `/14` (PC 1000 uniquement)
2. Correspond à la route masque `/8` (plage PC 1000-1063) - **Sélectionnée** (correspondance la plus spécifique)
3. Correspond également à la route masque `/0` (tous les PCs) - Ignorée (moins spécifique)

**Décision de Routage pour le DPC 5000 :**

1. Ne correspond pas à la route masque `/14`
2. Ne correspond pas à la route masque `/8`
3. Correspond à la route masque `/0` (tous les PCs) - **Sélectionnée** (seule correspondance, route de secours)

**Meilleures Pratiques**

1. **Omettre `mask` pour les Destinations Uniques** : Pour les correspondances exactes de code de point, omettez le champ `mask` entièrement (par défaut à `/14`)
2. **Utiliser `/14` Explicitement Seulement Lorsque Nécessaire** : Ne spécifiez `mask: 14` que 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 Réseau** : Routez des segments de réseau entiers vers des pairs spécifiques avec des masques `/0` à `/13`
4. **Utiliser `/0` comme Route de Secours** : Créez une route par défaut avec une faible priorité pour capturer le trafic non apparié
5. **La Plus Spécifique Gagne** : Le moteur de routage sélectionne toujours la route 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 gagne

---

# Configuration du Routage par Titre Global (GT)

Le routage par Titre Global permet un **routage basé sur le contenu** utilisant des numéros de téléphone ou des valeurs IMSI au lieu de codes de point. Pour un routage avancé par Titre Global basé sur le numéro appelant/appelé, voir le Guide NAT par Titre Global.

## Prérequis

- Activer le routage GT : `enable_gt_routing: true` dans `config/runtime.exs`

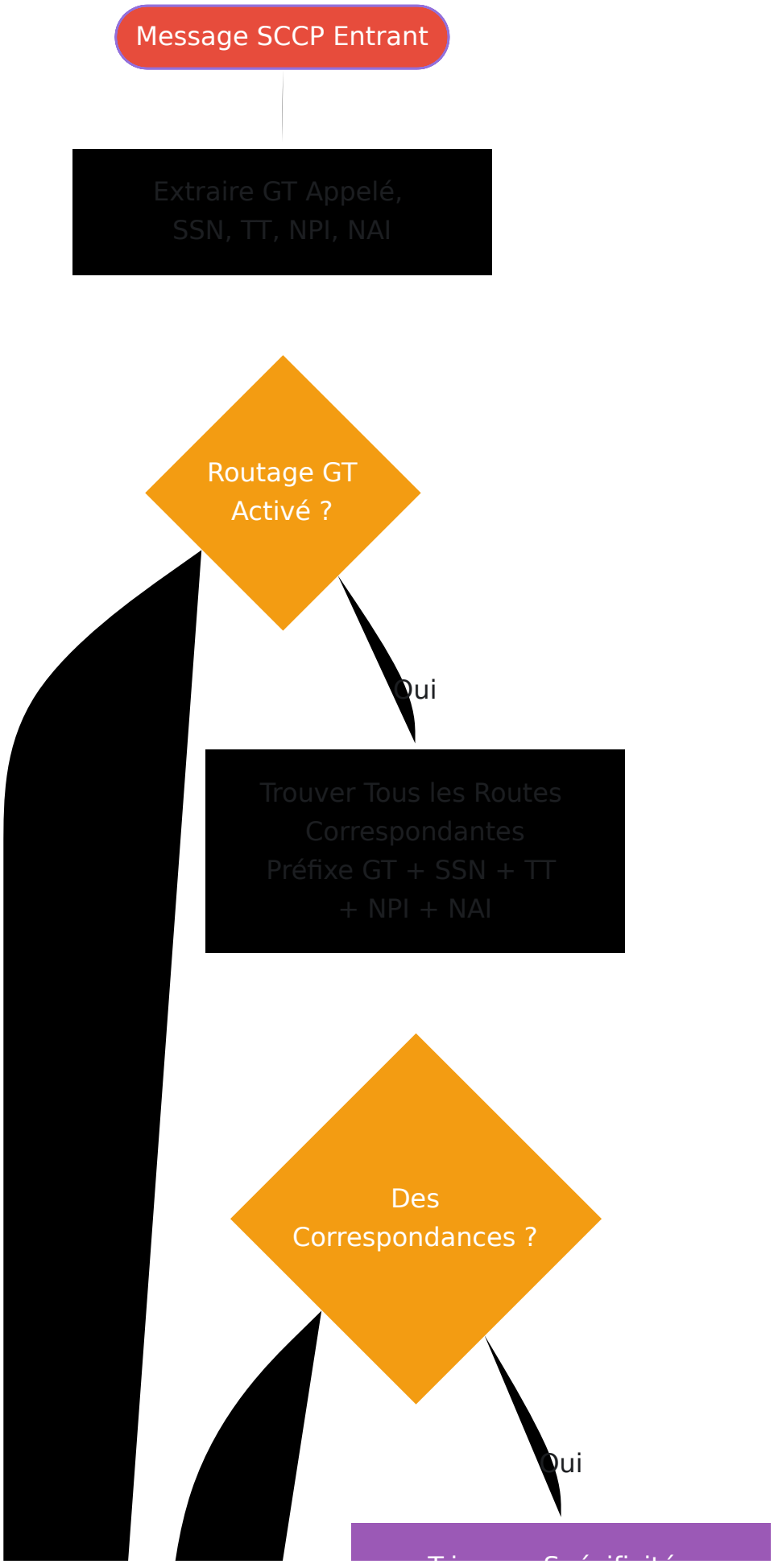
## Configuration des Routes GT

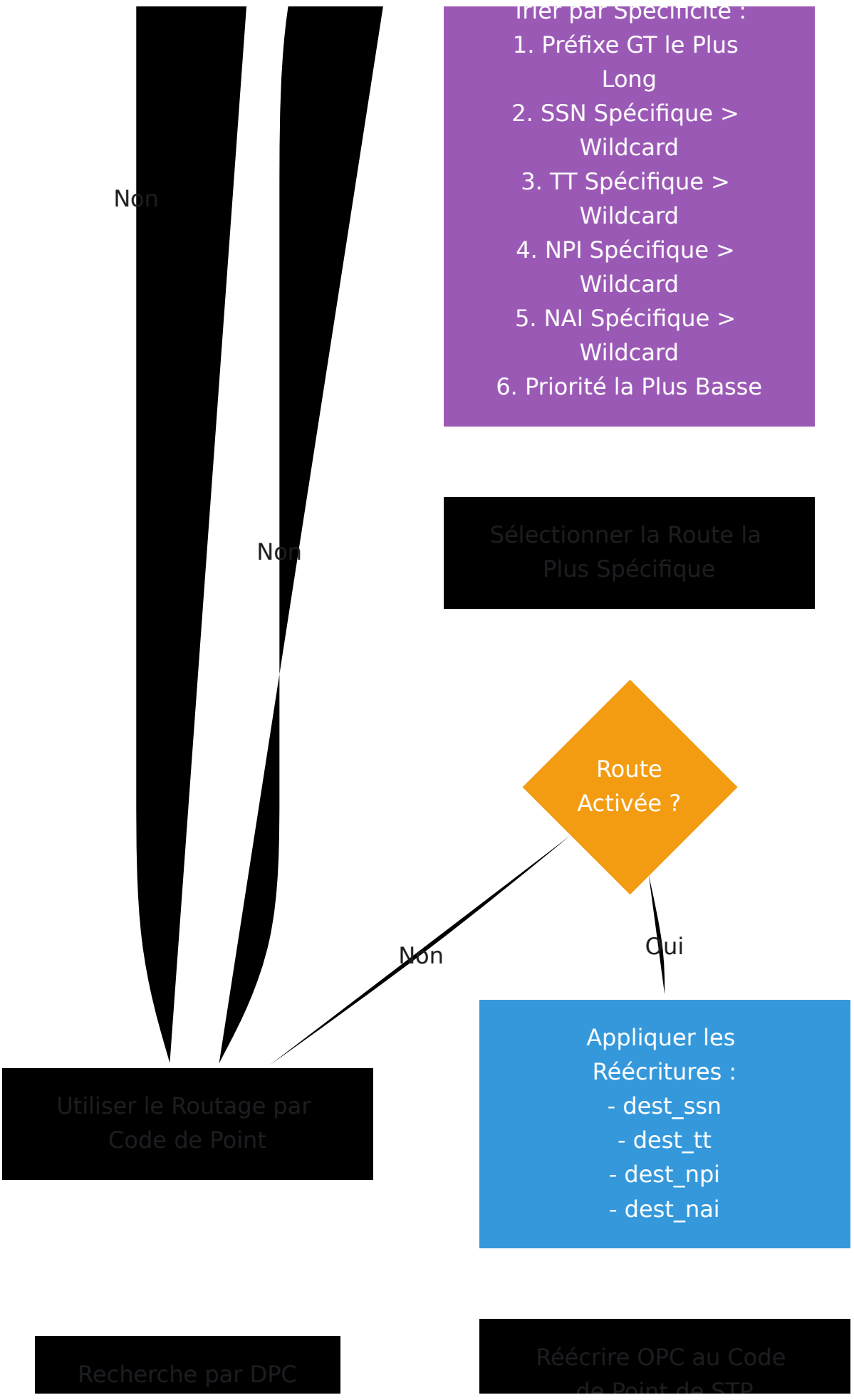
```
config :omniss7,
  # Activer le routage GT
  enable_gt_routing: true,

  m3ua_gt_routes: [
    # Route tous les numéros du Royaume-Uni (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
haut)
      description: "Numéros du Royaume-Uni"        # Description
pour la journalisation
    },

    # Route les numéros américains (préfixe 1) vers le pair 2
    %{
      gt_prefix: "1",
      peer_id: 2,
      priority: 1,
      description: "Numéros américains"
    },

    # Route plus spécifique : numéros mobiles britanniques
commençant par 447
    %{
      gt_prefix: "447",               # La correspondance du
préfixe le plus long gagne
      peer_id: 3,
      priority: 1,
      description: "Numéros mobiles britanniques"
    },

    # Routage spécifique à SSN (optionnel)
    %{
      gt_prefix: "555",
      source_ssn: 8,                  # Ne correspond que si SSN
source = 8 (SMSC)
      peer_id: 4,
```

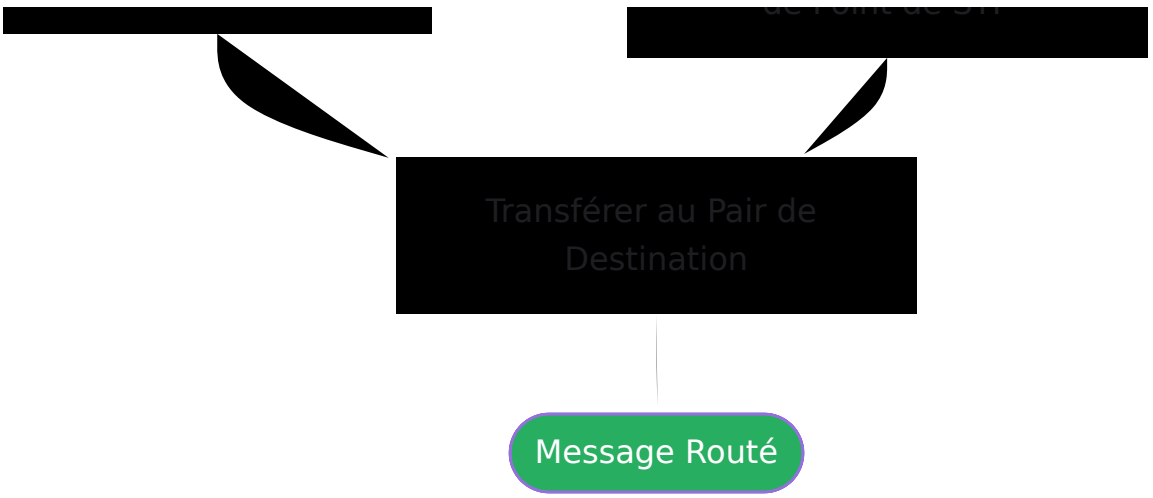
```
      dest_ssn: 6,                    # Réécrit SSN de
destination à 6 (HLR)
      priority: 1,
      description: "Trafic SMS pour le préfixe 61"
    }
  ]
```

## Logique de Routage GT

L'algorithme de routage GT suit ce processus de décision :


Message SCCP Entrant
Extraire GT Appelé,
SSN, TT, NPI, NAI
Routage GT
Activé ?
Oui
Trouver Tous 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
Utiliser le Routage par Code de Point
Appliquer les Réécritures :
- dest_ssn
- dest_tt
- dest_npi
- dest_nai
Recherche par DPC
Réécrire OPC au Code de Point de STP




**Étapes de Routage :**

1. **Correspondance du Préfixe le Plus Long** : 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" gagne (correspondance la plus longue)
2. **Correspondance SSN** (Optionnel) :
    - 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** (Optionnel) :
    - 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 gagnent sur les wildcards
    - Ordre de priorité : Longueur du Préfixe GT → SSN Spécifique → TT → NPI → NAI → Numéro de Priorité le Plus Bas
5. **Réécriture des Indicateurs** (Optionnel) :

- 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

## Exemples de Routage GT

| GT Appelé | SSN Source | TT | NPI | NAI | Route Correspondante | Rai |
|---|---|---|---|---|---|---|
| 447712345678 | 6 | - | - | - | "447" → pair 3 | Correspo de préfix plus long |
| 441234567890 | 6 | - | - | - | "44" → pair 1 | Correspo de préfix aucune plus spé |
| 12125551234 | 6 | - | - | - | "1" → pair 2 | Correspo de préfix les num américa |
| 555881234567 | 8 | - | - | - | "555" (SSN 8) → pair 4 | Correspo GT + SS réécrit S |
| 555881234567 | 6 | - | - | - | "555" (wildcard SSN) → pair X | Correspo GT, aucu réécritur |
| 441234567890 | 6 | 0 | 1 | 4 | "44" (TT=0) → pair 1 | Correspo GT + TT transfor 3 |
| 12125551234 | 8 | 0 | 1 | 4 | "1" (TT=0, NPI=1, NAI=4) | La plus spécifiqu correspo GT+TT+ |

**Utilisations Pratiques pour le Routage TT/NPI/NAI**

1. **Normalisation de l'Interconnexion Réseau**
    - Différents réseaux peuvent utiliser des conventions d'indicateurs différentes
    - 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 des plans de numérotation lors du routage entre différents types de réseaux
    - Exemple : Route d'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 domestique
4. **Routage Spécifique au Fournisseur**
    - Router en fonction du type de traduction vers différents fournisseurs de services
    - Exemple : TT=0 routé vers le Fournisseur A, TT=2 routé vers le Fournisseur B
5. **Intégration de Systèmes Hérités**
    - Les systèmes modernes peuvent utiliser des valeurs d'indicateur différentes des systèmes hérités
    - Transformer au STP pour maintenir la compatibilité descendante

---

# Fonctionnalités de Gestion des Routes

## Désactivation des Routes

Les routes peuvent être temporairement désactivées sans être supprimées. Cela est utile pour les tests, la maintenance ou la gestion du trafic.

### Drapeau Activé

Les routes par Code de Point et par Titre Global prennent en charge un drapeau `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 du Royaume-Uni - 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 basculer les routes activées/désactivées sans modifier la configuration

**Cas d'Utilisation :**

- Tester les changements de flux de trafic
- Fenêtres de maintenance temporaires
- Test A/B de différents chemins de routage
- Déploiement progressif de nouvelles routes

---

# Routes DROP : Prévention des Boucles de Routage

Les routes DROP (avec `peer_id: 0`) rejettent silencieusement le trafic au lieu de le transférer. 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 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 transféré)
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}`

**

# Guide de la passerelle USSD

← Retour à la documentation principale

Ce guide couvre la **passerelle USSD** d'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



---

## Aperçu

La passerelle USSD gère deux directions du trafic USSD :

- **D'origine mobile (entrant)** — Un abonné compose un code court (par exemple `*100#`). La passerelle reçoit la MAP `processUnstructuredSS-Request` (opcode 59), la transmet à votre rappel HTTP et renvoie 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 et achemine 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 de texte max** | 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 |

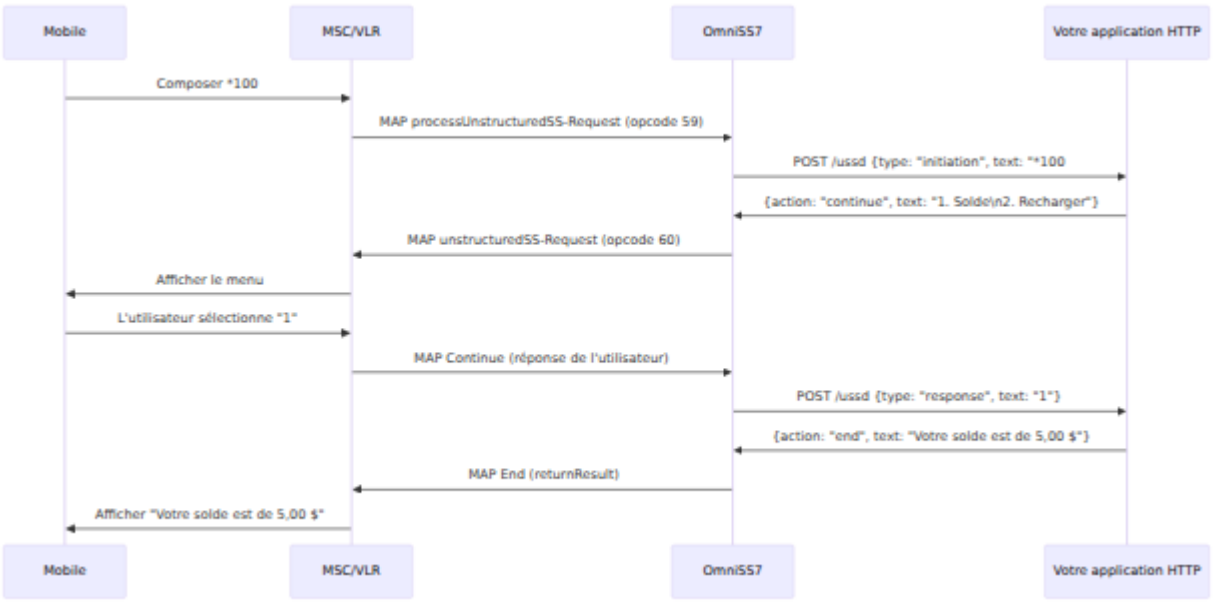
## Références 3GPP

| 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 de données |

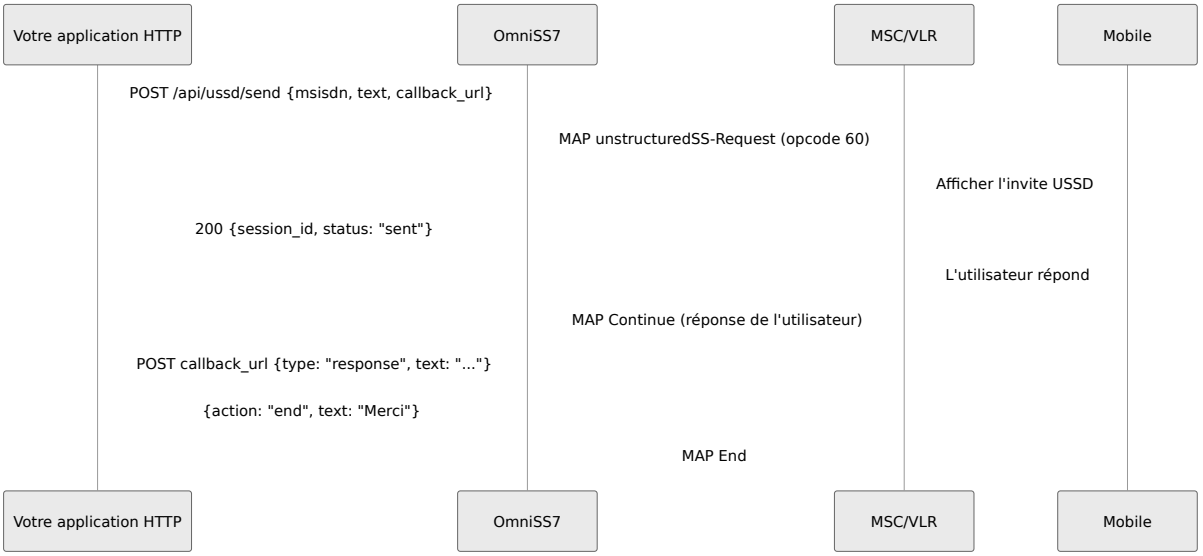
---

# Architecture

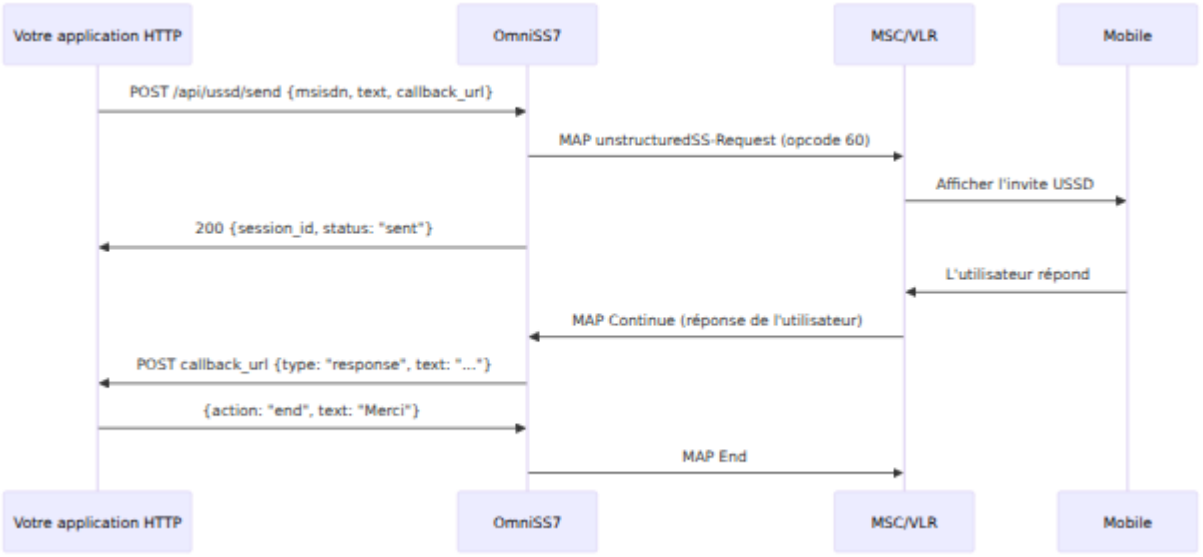
## Flux d'origine mobile (entrant)



## 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 max pour la
réponse de l'abonné par tour (30 secondes)

    # Paramètres de rappel HTTP
    http_timeout_ms: 5_000,         # Délai d'expiration pour HTTP
POST 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 cartes | 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 dépas |

| Paramètre | Type | Requis | Par défaut | Des |
|---|---|---|---|---|
| `ussd_gateway.turn_timeout_ms` | Entier | Non | `30_000` | Temps maxin attend 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 de rép |
| `ussd_gateway.max_text_length` | Entier | Non | `182` | Nomb maxin caract une cl texte textes dépas cette 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 correspondre. Utilisez `"*"` comme solution de repli. 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 des routes

Les routes sont correspondantes par **préfixe le plus long d'abord**. 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 repli

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 mobile de l'abonné. Longueur maximale 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": "invalid request", "required": ["msisdn", "text", "callback_url"]}` | Champs requis manquants |
| 400 | `{"error": "gsm7_encode_failed", ...}` | Le texte contient des caractères non présents dans l'alphabet GSM 7 bits |
| 500 | `{"error": "send_failed", ...}` | ��chec de l'envoi M3UA (vérifiez la connectivité) |
| 503 | `{"error": "USSD gateway not enabled"}` | `ussd_gateway_enabled` est `false` |

## Exemple cURL

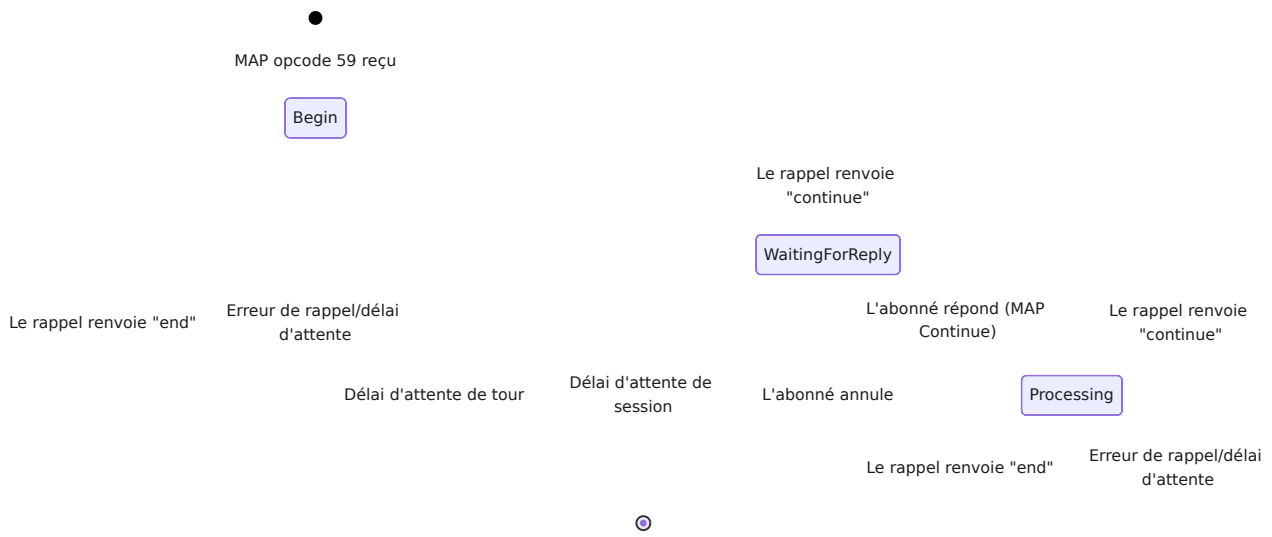
```
curl -X POST http://localhost:8080/api/ussd/send \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "+254712345678",
    "text": "Vous avez une facture en attente. Répondez 1 pour
payer.",
    "callback_url": "http://billing-app:9000/ussd"
  }'
```

---

# 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 retourne 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 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'attente de session** | 3 minutes | Durée totale de la session. Termine la session indépendamment de l'activité. |
| **Délai d'attente 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 de manière élégante et tente toujours d'envoyer une réponse d'erreur MAP au mobile afin que l'abonné voie 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) |
| Échec 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 8080).

# Métriques USSD

**Métrique**: `ussd_requests_total` **Type**: Compteur **Description**: Total des requêtes USSD traitées **Étiquettes**:

- `direction` — `"inbound"` (d'origine mobile) ou `"outbound"` (push d'origine réseau)

**M��trique**: `ussd_active_sessions` **Type**: Jauge **Description**: Nombre de sessions USSD actuellement actives

**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
rate(ussd_requests_total[5m])

# Sessions actives
ussd_active_sessions

# Latence USSD sortante (p95)
histogram_quantile(0.95,
rate(map_request_duration_milliseconds_bucket{operation="ussd_send"}
[5m]))
```

---

# Serveur de rappel 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 reçoit 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 repli)
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 le 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 HTTP sortant depuis OmniSS7
- L'application de rappel ne fonctionne pas

**Résolution**:

1. Testez la connectivité : `curl -v http://your-app:9000/ussd` depuis l'hôte OmniSS7
2. Vérifiez les règles de pare-feu pour 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 avec "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 d'OmniSS7 pour réduire la latence

## Erreurs d'encodage GSM 7 bits

**Symptômes**: Erreurs `gsm7_encode_failed` dans les journaux ou réponses 400 de `/api/ussd/send`.

**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

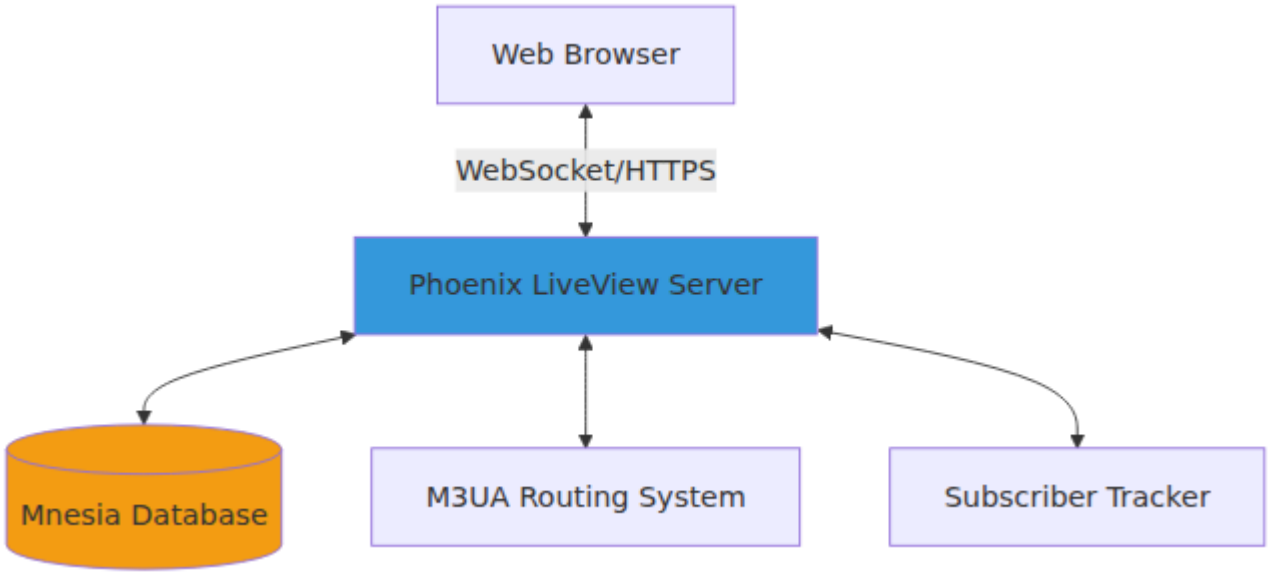


---

## 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

- **Protocole**: HTTPS
- **Port**: 443 (configuré dans `config/runtime.exs`)
- **IP par défaut**: 0.0.0.0 (écoute sur toutes les interfaces)
- **Certificats**: Situés dans `priv/cert/`

**URL d'accès**: `https://[server-ip]:443`

---

# 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 443 est ouvert pour le trafic HTTPS

## Pages disponibles par mode

| Page | Mode STP | Mode HLR | Mode SMSc | Description |
|---|---|---|---|---|
| **Événements SS7** | ✅ | ✅ | ✅ | Journalisation des événements et capture des messages SCCP |
| **Client SS7** | ✅ | ✅ | ✅ | Tests manuels des opérations MAP |
| **M3UA** | ✅ | ✅ | ✅ | État de la connexion M3UA |
| **Routage** | ✅ | ✅ | ✅ | Gestion de la table de routage M3UA |
| **Test de routage** | ✅ | ✅ | ✅ | Test et validation des routes |
| **Liens HLR** | ✅ | ✅ | ✅ | État de l'API HLR et gestion des abonnés |
| **Abonnés actifs** | ✅ | ✅ | ✅ | Suivi de la localisation des abonnés en temps réel (HLR) |
| **Liens SMSc** | ✅ | ✅ | ✅ | État de l'API SMSc et gestion des files d'attente |
| **Abonnés SMSc** | ✅ | ✅ | ✅ | Suivi des abonnés en temps réel (SMSc) |
| **Application** | ✅ | ✅ | ✅ | Ressources système et surveillance |

| Page | Mode STP | Mode HLR | Mode SMSc | Description |
|---|---|---|---|---|
| **Configuration** | □ | □ | □ | Visualiseur de configuration |

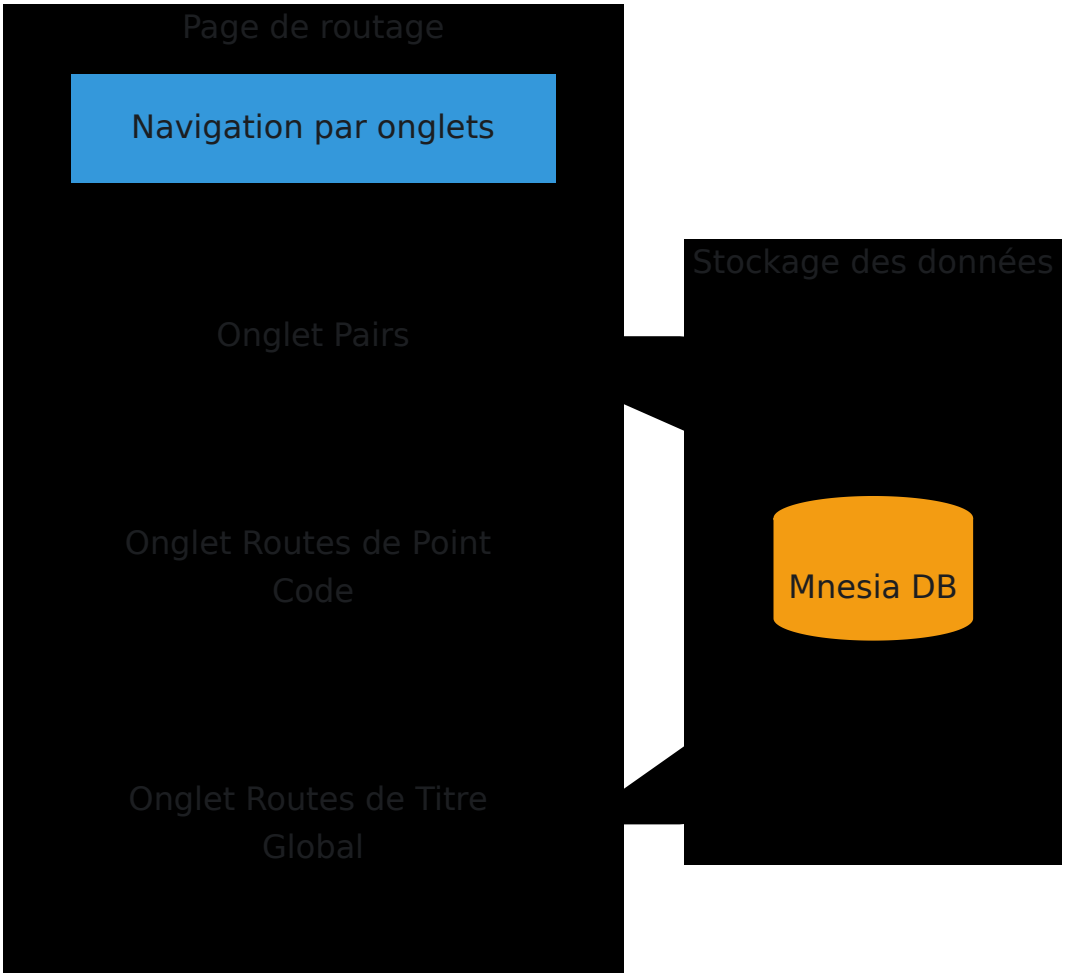
---

# 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 la connexion | `client`, `server`, `stp` |
| **Point Code** | Point code 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`
   - **Point Code**: Point code 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 si nécessaire
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 en ligne mais pas encore actif |
| `down` |  Rouge | Le pair est déconnecté |

---

## Onglet Routes de Point Code

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 point code**

1. **Cliquez** sur l'onglet "Routes de Point Code"
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 de masque de point code

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 (optionnel) | `6` (HLR), `any` |
| **ID du pair** | Pair cible | `1` |
| **Pair** | Nom du pair | `"HLR_West (1)"` |
| **SSN Dest** | Réécrire le SSN lors du transfert (optionnel) | `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 la route de secours, ou entrez des chiffres (par exemple, `"1234"`)
   - **SSN Source**: Optionnel - filtrer par SSN du parti appelé
   - **ID du pair**: Sélectionnez le pair cible
   - **SSN Dest**: Optionnel - 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 (Home Location Register) |
| 7 | VLR (Visitor Location Register) |
| 8 | MSC (Mobile Switching Center) |
| 9 | EIR (Equipment Identity Register) |
| 10 | AUC (Authentication Center) |
| 142 | RANAP |
| 145 | gsmSCF (Service Control Function) |
| 146 | SGSN |

#### Réécriture de SSN

- **SSN Source**: Correspondance sur le SSN du parti appelé dans les messages entrants
- **SSN Dest**: 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 vers SSN=7 (VLR) sur le 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 inférieurs sont routés en premier (1 = priorité la plus élevée)
3. **Statut du pair**: Seules les routes vers des pairs `actifs` sont utilisées

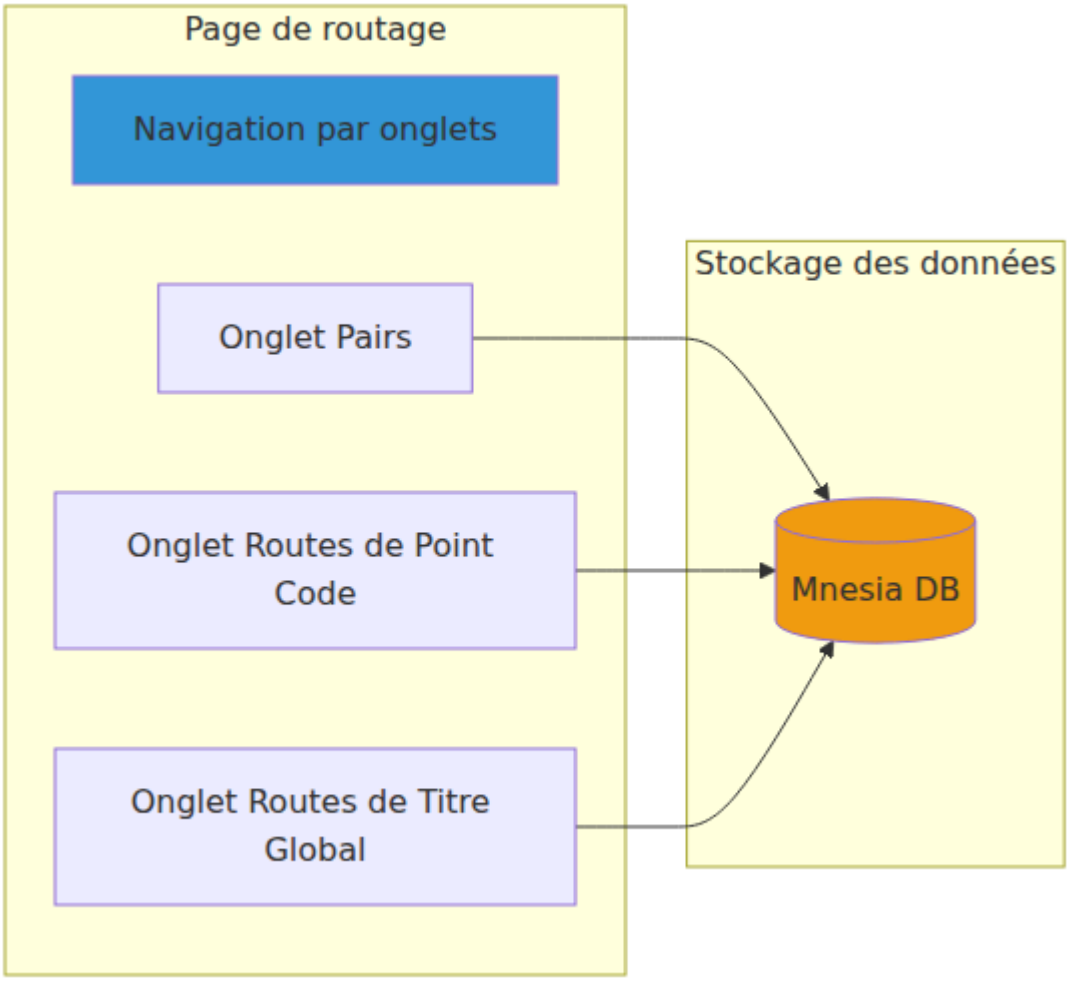
---

# 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** | Dernière horodatage UpdateLocation | `"2025-10-25 14:23:45 UTC"` |
| **Durée** | Temps 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
- **VLRs Uniques**: Nombre d'adresses VLR distinctes
- **MSCs 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 en temps réel des abonnés.

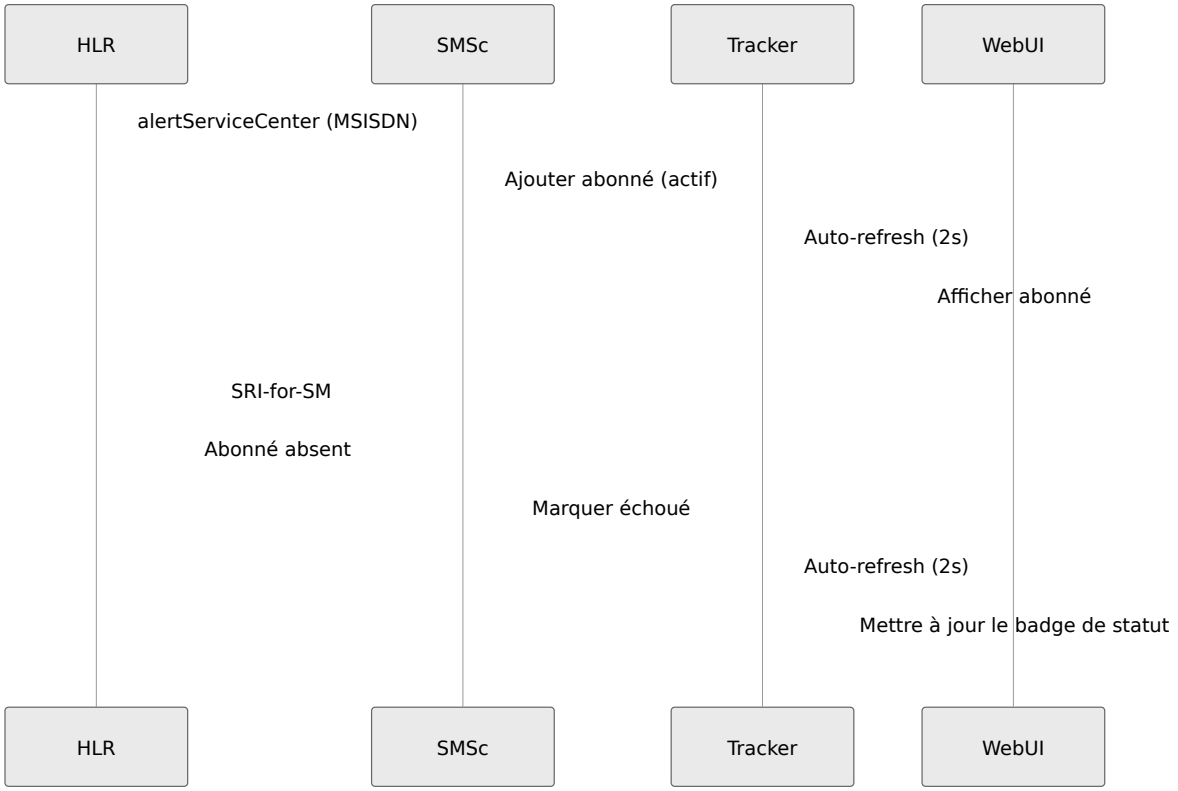
---

# 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, l'état 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** | Dernière horodatage de mise à jour | `"2025-10-30 14:23:45 UTC"` |
| **Durée** | Temps depuis la dernière mise à jour | `"15m 34s"` |

## Indicateurs de statut

- ● **Actif** (Vert): L'abonné est joignable, dernier alertServiceCenter reçu avec succès
- ○ **Échoué** (Rouge): Dernière tentative de livraison échouée (SRI-for-SM ou erreur d'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 un statut actif
- **Échoués**: Nombre d'abonnés avec un statut échoué
- **HLRs 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 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 ne comprend 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 route 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 route 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 |
| État 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 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 `.pem` sont présents
2. **Vérifiez le port 443**: Vérifiez que les règles de pare-feu autorisent le trafic HTTPS
3. **Application en cours d'exécution**: Confirmez que l'application est en cours d'exécution avec `iex -S mix`

4. **Console du navigateur**: Vérifiez les erreurs de certificat SSL (avertissements de certificat auto-signé)

## Routes ne persistent 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 de pair en direct

Pour des opérations avancées ou de l'automatisation, consultez le Guide API.