Benchmarks

Ce répertoire contient des benchmarks de performance pour le système SMS-C utilisant Benchee.

Benchmarks Disponibles

1. Benchmark SMS Brut (`raw_sms_bench.exs`)

Benchmark de l'API `submit_message_raw` en utilisant de vrais PDUs SMS.

Caractéristiques :

- Utilise de vrais PDUs SMS (ajoutez vos PDUs à la liste `@sample_pdus` dans le fichier)
- Désactive la détection des doublons en effaçant les empreintes avant chaque itération
- Produit des rapports à la fois en console et en HTML

Utilisation :

```
mix run benchmarks/raw_sms_bench.exs
```

Sortie : `benchmarks/output/raw_sms_benchmark.html`

2. Benchmark de l'API Message (`message_api_bench.exs`)

Benchmark de diverses opérations de l'API message, y compris l'insertion, la récupération et le routage.

Caractéristiques :

- Teste insert_message (simple et avec routage)

- Teste get_messages_for_smsc
- Teste list_message_queues
- Pré-remplit la base de données avec des données de test pour des scénarios réalistes

Utilisation :

```
mix run benchmarks/message_api_bench.exs
```

Sortie : `benchmarks/output/message_api_benchmark.html`

Configuration

Tous les benchmarks utilisent Benchee avec les paramètres par défaut suivants :

- Échauffement : 2 secondes
- Temps : 10 secondes
- Temps mémoire : 2 secondes
- Statistiques étendues activées
- Rapports HTML générés automatiquement

Sorties

Les rapports de benchmark HTML sont générés dans `benchmarks/output/` et incluent :

- Métriques de performance détaillées
- Graphiques de comparaison
- Statistiques d'utilisation de la mémoire
- Analyse statistique

Référence API SMS-C

← Retour à l'Index de Documentation

Référence pour l'API REST SMS-C, générée à partir de la table de routage en direct (`config/config.exs`) et des implémentations de contrôleur. Chaque point de terminaison ci-dessous est réellement routé par le service en cours d'exécution.

L'application elle-même sert une spécification OpenAPI lisible par machine et une interface Swagger UI interactive. Ceux-ci sont toujours synchronisés avec la version déployée :

```
GET /api/schema      # Spécification OpenAPI JSON
GET /api/docs        # Swagger UI
```

Table des Matières

- Aperçu
- Authentification
- Conventions de Réponse
- Statut
- Messages
- Indication de Message en Attente
- Emplacements
- Frontends
- Événements
- Campagnes
- Autres Familles de Points de Terminaison
- Codes de Statut HTTP
- Format d'Erreur
- Meilleures Pratiques

Aperçu

URL de Base

```
https://<host>:8443/api
```

Port par Défaut : 8443 (configurable via `config :api_ex, api: %{port: ...}`) **Protocole** : HTTPS lorsque `enable_tls` est activé (par défaut). Tous les itinéraires sont montés sous le préfixe `/api`.

Type de Contenu

Les corps de requête et de réponse sont en JSON :

```
Content-Type: application/json
```

Il n'y a pas de versionnement d'API basé sur l'URL. La table de routage est le contrat ; consultez `/api/schema` pour la forme exacte déployée.

Authentification

L'API n'a **aucune authentification au niveau de l'application**. Le pipeline de requête ne négocie que le type de contenu et attache la spécification OpenAPI ; il n'y a pas de clé API, de jeton d'authentification ou de vérification de session devant les itinéraires.

Le contrôle d'accès est donc censé être géré en dehors de l'application :

- **TLS** : lorsque `enable_tls` est vrai, l'auditeur termine HTTPS en utilisant le `tls_cert_path` / `tls_key_path` configuré. Utilisez-le en production.
- **Contrôles réseau** : restreindre le port API aux sources de confiance avec un pare-feu ou un proxy inverse en amont. Plusieurs points de terminaison enregistrent l'IP de l'appelant (à partir de `X-Forwarded-For` ou du socket) pour l'audit, mais ne bloquent pas sur cela.

Ne pas exposer le port API directement à des réseaux non fiables.

Conventions de Réponse

Les contrôleurs renvoient la ressource **directement**, sans enveloppe `data`.

Ressource unique :

```
{ "id": 12345, "status": "queued" }
```

Les collections sont renvoyées sous forme de tableau JSON brut :

```
[ { "id": 12345 }, { "id": 12346 } ]
```

Les erreurs au niveau du contrôleur utilisent une chaîne `error` plate :

```
{ "error": "payload_hex is required. For plaintext, use POST
/api/messages_plaintext" }
```

Les erreurs au niveau du framework (échecs de routage, corps non analysables) reviennent sous forme de :

```
{ "status": "error", "response": "Not Found" }
```

Statut

Vérification de la santé pour la surveillance et les équilibreurs de charge.

Requête :

```
GET /api/status
```

Réponse (200 OK) :

```
{
  "status": "ok",
  "application": "OmniMessage",
  "timestamp": "2025-11-01T23:00:00Z"
}
```

Exemple :

```
curl https://host:8443/api/status
```

Messages

La file d'attente de messages est la ressource principale. Le contenu du message est transporté sous `payload_hex` (les octets TP-User-Data) plus un octet `dcs` (schéma de codage de données) et un `udh_hex` optionnel (en-tête de données utilisateur). Les appelants en texte clair devraient utiliser Soumettre du Texte Clair, qui effectue l'encodage, la sélection DCS et le fractionnement multipart pour vous.

Objet Message

Champs renvoyés pour un message (voir `ApiEx.Schemas.MessageSchema`) :

Champ	Type	Remarques
`id`	entier	ID du message
`source_msisdn`	chaîne	Numéro source
`destination_msisdn`	chaîne	Numéro de destination
`payload_hex`	chaîne	Octets TP-User-Data en hex (contenu canonique)
`dcs`	chaîne	Schéma de codage de données sous forme d'octet hex (`00`=GSM7, `08`=UCS2, `04`=8bit, `C8`=MWI)
`udh_hex`	chaîne / null	Octets d'en-tête de données utilisateur en hex
`message_text`	chaîne / null	Texte UTF-8 dérivé pour affichage uniquement (non utilisé pour la livraison)
`source_smsc`	chaîne	Identifiant SMSC source
`dest_smsc`	chaîne / null	Identifiant SMSC de destination
`status`	chaîne	Un des `queued`, `backoff`, `sent`, `delivered`, `expired`, `dropped`, `auto_replied`, `balance_rejected`, `deleted`
`send_time`	datetime	
`deliver_time`	datetime / null	

Champ	Type	Remarques
`expires`	datetime	
`delivery_attempts`	entier	
`deadletter`	booléen	
`message_parts`	entier / null	
`message_part_number`	entier / null	
`smsc_node_name`	chaîne	Noeud qui possède le message

Lister les Messages

Requête :

```
GET /api/messages
```

En-têtes Optionnels :

- `smsc: <frontend_name>` - mode de sondage : renvoie les messages destinés à ce SMSC.
- `include-unrouted: true|1` - en mode de sondage, renvoie également les messages sans enregistrement de localisation (par défaut : faux).
- `x-decode: true|1` - inclut une vue décodée de chaque charge utile de message.

Paramètres de Requête (mode liste, utilisés uniquement lorsqu'aucun en-tête `smsc` n'est présent) :

- `status` - filtre par un seul atome de statut (voir la liste des statuts ci-dessus).

- `limit` - max enregistrements (par défaut 100, max 1000).
- `offset` - enregistrements à ignorer (par défaut 0).

Le point de terminaison est **toujours paginé** en mode liste : sans `limit`, il renvoie au maximum 100 enregistrements, jamais l'ensemble du magasin.

Réponse (200 OK) - un tableau JSON d'objets message.

Exemples :

```
# Sondage pour les messages d'un frontend
curl -H "smsc: uk_gateway" https://host:8443/api/messages

# Sondage incluant des messages non routés
curl -H "smsc: uk_gateway" -H "include-unrouted: true"
https://host:8443/api/messages

# Lister les messages livrés
curl "https://host:8443/api/messages?status=delivered&limit=50"
```

Obtenir un Message

Requête :

```
GET /api/messages/:id
```

L'en-tête optionnel `x-decode: true` ajoute une vue de charge utile décodée.

Réponse (200 OK) : un seul objet message. **404** avec `{"error": "Message not found"}` si l'id est inconnu.

```
curl https://host:8443/api/messages/12345
```

Soumettre un Message

Soumettez un message pré-encodé. **`payload_hex` est requis** ; les requêtes sans cela sont rejetées avec 422.

Requête :

```
POST /api/messages
Content-Type: application/json
```

Corps :

```
{
  "source_msisdn": "15555551234",
  "destination_msisdn": "15555555678",
  "payload_hex": "c8329bfd06dddf723619",
  "dcs": "00",
  "source_smsc": "api_client"
}
```

Champs :

- `source_msisdn`, `destination_msisdn` - numéros.
- `payload_hex` - **requis**. Octets TP-User-Data en hex.
- `dcs` - octet hex du schéma de codage de données (défauts appliqués en aval).
- `udh_hex` - en-tête de données utilisateur hex optionnel.
- `source_smsc` - identifie le système soumetteur.
- `source_type` - classification de source optionnelle.

Réponse (201 Created) : l'objet message créé. Les rejets retournent 422 avec une chaîne `error` (par exemple, `payload_hex` manquant).

```
curl -X POST https://host:8443/api/messages \
  -H "Content-Type: application/json" \
  -d '{
    "source_msisdn": "15555551234",
    "destination_msisdn": "15555555678",
    "payload_hex": "c8329bfd06dddf723619",
    "dcs": "00",
    "source_smsc": "api_client"
  }'
```

Soumettre du Texte Clair

Soumettez une chaîne UTF-8 et laissez le SMS-C gérer l'encodage (GSM7 vs UCS2), la sélection DCS, et le fractionnement multipart.

Requête :

```
POST /api/messages_plaintext
Content-Type: application/json
```

Corps :

```
{
  "source_msisdn": "15555551234",
  "destination_msisdn": "15555555678",
  "message": "Hello World",
  "source_smsc": "api_client"
}
```

Le texte du message peut être donné comme `message`, `text`, ou `body`. `source_msisdn` et `destination_msisdn` sont requis.

Réponse (201 Created) :

- Partie unique : l'objet message créé.
- Multipart : `{ "parts": <n>, "messages": [ ... ] }`.

```
curl -X POST https://host:8443/api/messages_plaintext \
  -H "Content-Type: application/json" \
  -d '{
    "source_msisdn": "15555551234",
    "destination_msisdn": "15555555678",
    "message": "Hello World",
    "source_smsc": "api_client"
  }'
```

Soumettre un PDU Brut

Soumettez un PDU SMS brut (hex) pour une compatibilité maximale avec les frontends hérités et SIP.

Requête :

```
POST /api/messages_raw
Content-Type: application/json
```

Corps :

```
{
  "message_body": "0001000B916407007009F0000004D4F29C0E",
  "source_smsc": "legacy_system"
}
```

Champs :

- `message_body` - **requis**. PDU SMS encodé en hex (ou TPDU, voir ci-dessous).
- `source_msisdn` - optionnel.
- `source_smsc` / `smsc_node_name` - requis en mode uniquement TPDU.
- `rp_originator_address`, `rp_destination_address` - fournissez-les pour soumettre un TPDU brut (la couche RP est synthétisée).

L'IP de l'appelant est prise à partir de `X-Forwarded-For` (ou du socket) pour résoudre le frontend d'origine.

Réponses :

- 201 Created avec `{ "status": "success", "response": { "id": <id> } }` (un champ hex `rp_ack` est inclus lorsqu'il est généré).
- 200 OK avec `{ "status": "success", "response": { "status": "delivery_report_received" } }` lorsque le PDU est un rapport de livraison plutôt qu'un nouveau message.

- 422 avec une chaîne `error` pour hex manquant/invalid ou échecs de décodage.

```
curl -X POST https://host:8443/api/messages_raw \
  -H "Content-Type: application/json" \
  -d '{
    "message_body": "0001000B916407007009F0000004D4F29C0E",
    "source_smsc": "legacy_system"
  }'
```

Le service décode le PDU (3GPP TS 23.040), extrait les numéros/texte/DCS, détecte les rapports de livraison et applique le routage. Les rapports de livraison sont enregistrés mais ne sont pas stockés comme de nouveaux messages.

Mettre à Jour un Message (PATCH)

Mise à jour partielle d'un message stocké.

Requête :

```
PATCH /api/messages/:id
Content-Type: application/json
```

Envoyez tous les champs de message modifiables (par exemple `dest_smsc`, `deliver_after`, `status`). Les modifications de métadonnées sont enregistrées dans le journal des événements.

Réponse (200 OK) : l'objet message mis à jour. 404 si non trouvé, 422 en cas de mise à jour rejetée.

```
curl -X PATCH https://host:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{ "dest_smsc": "backup_gateway" }'
```

Incrémenter la Tentative de Livraison

Enregistrer une livraison échouée : incrémente le compteur de tentatives, déplace le message vers `backoff`, et planifie la prochaine tentative avec un retour d'exponentiel.

Deux formes équivalentes existent :

```
PUT  /api/messages/:id                  # la ressource messages,
PUT = incrémenter
POST /api/messages_increment_attempt        (id dans le corps)
PUT  /api/messages_increment_attempt/:id    (id dans l'URL)
```

`PUT /api/messages/:id` est distinct de `PATCH /api/messages/:id` : PATCH effectue une mise à jour partielle, PUT incrémente la tentative de livraison. Passez `keep_route=true` pour conserver le `dest_smsc` actuel lors de la nouvelle tentative.

Réponse (200 OK) : le message mis à jour, par exemple :

```
{ "id": 12345, "status": "backoff", "delivery_attempts": 2 }
```

```
curl -X PUT https://host:8443/api/messages/12345
```

Marquer Comme Livré

Marquer un message comme livré (stock local ou fédéré).

```
POST /api/messages_mark_delivered           (id dans le corps)
PUT  /api/messages_mark_delivered/:id        (id dans l'URL)
```

L'en-tête optionnel `smsc` sélectionne le SMSC fédéré lorsque le message n'est pas conservé localement.

Réponse (200 OK) : l'objet message livré. 404 si non trouvé.

```
curl -X POST https://host:8443/api/messages_mark_delivered \
  -H "Content-Type: application/json" \
  -d '{ "id": 12345 }'
```

Réacheminer un Message

Réévaluer les règles de routage pour un message stocké et mettre à jour son `dest_smsc`.

Requête :

```
POST /api/messages/reroute
Content-Type: application/json
```

Corps : `{ "id": 12345 }`

Réponse (200 OK) :

```
{ "message_id": 12345, "new_dest_smsc": "uk_gateway", "status":
"rerouted" }
```

404 si le message n'est pas trouvé, 422 si aucune route ne correspond.

Effacer un Champ

Réinitialiser un champ de message spécifique à null.

Requête :

```
POST /api/messages/clear_field
Content-Type: application/json
```

Corps : `{ "id": 12345, "field": "dest_smsc" }`

`field` doit être l'un de `delivery_time`, `dest_smsc`, ou `deliver_after`; toute autre valeur renvoie 400. Le changement est enregistré dans le journal des

événements.

Réponse (200 OK) : l'objet message mis à jour.

Supprimer un Message

Requête :

```
DELETE /api/messages/:id
```

Réponse (204 No Content). 404 si non trouvé. La suppression est permanente.

```
curl -X DELETE https://host:8443/api/messages/12345
```

Indication de Message en Attente

Soumettez une notification MWI (Indication de Message en Attente). Le contrôleur construit le PDU MWI selon 3GPP TS 23.038 et stocke un message prêt à être livré avec un `payload_hex` et un `dcs` pré-calculés.

Requête :

```
POST /api/mwi
Content-Type: application/json
```

Corps :

```
{
  "source_msisdn": "12345",
  "destination_msisdn": "15555555678",
  "indication_type": "voicemail",
  "indication_active": true,
  "message_body": "Vous avez 1 nouveau message vocal",
  "source_smsc": "IMS_SMSC"
}
```

Champs :

- `indication_type` - `voicemail` (par défaut), `fax`, `email`, ou `autre`.
- `indication_active` - `true` pour définir l'indicateur, `false` pour le supprimer (par défaut `true`).
- `message_body` - texte d'accompagnement optionnel (encodé en GSM7).
- `source_msisdn`, `destination_msisdn`, `source_smsc`, `dest_smsc`.

Réponse (201 Created) : l'objet message créé. 422 en cas d'échec.

Emplacements

Enregistrements de localisation des abonnés pour la livraison mobile-terminée. Ressource REST standard ; le segment de chemin `:id` est l'IMSI.

```
GET    /api/locations           # lister (paginé : ?
limit=&offset=, en-tête x-total-count)
GET    /api/locations/:id       # obtenir par IMSI
POST   /api/locations           # créer ou mettre à jour
(identifié par IMSI)
PATCH  /api/locations/:id       # mise à jour partielle
PUT    /api/locations/:id       # mise à jour
DELETE /api/locations/:id       # supprimer
```

Corps de Création/Mise à Jour :

```
{
  "msisdn": "15555551234",
  "imsi": "001001000000001",
  "location": "msc1.region1.example.com",
  "ran_location": "cell_tower_12345",
  "imei": "123456789012345",
  "ims_capable": true,
  "csfb": false,
  "registered": true,
  "expires": "2025-11-01T23:30:00Z",
  "user_agent": "Samsung Galaxy"
}
```

`imsi` et `msisdn` sont requis ; les champs restants sont optionnels. Les réponses sont l'objet de localisation (200/201) ; les échecs de validation retournent 422. Appelés par les systèmes de gestion de mobilité lorsqu'un abonné s'enregistre ou se désenregistre.

Frontends

Suivez les connexions SMSC des frontends. Cette ressource expose **index, show, et create uniquement** (il n'y a pas de sous-routes `active`, `stats`, `history`, ou `register`).

```
GET  /api/frontends           # lister toutes les inscriptions
GET  /api/frontends/:id       # obtenir une inscription
POST /api/frontends           # enregistrer ou actualiser un
frontend
```

Corps d'Inscription (`POST /api/frontends`) :

```
{
  "frontend_name": "smpp-gateway-1",
  "frontend_type": "SMPP",
  "ip_address": "10.0.1.5",
  "hostname": "smpp-gw-01",
  "uptime_seconds": 3600,
  "configuration": "{\"port\": 2775, \"system_id\":
\"smpp_user\"}"
}
```

`frontend_name` est requis. Si `ip_address` est omis, il est rempli à partir de l'IP source de la requête (`X-Forwarded-For` ou socket). Si une inscription active existe déjà pour le `frontend_name`, elle est mise à jour ; sinon, une nouvelle est créée. Les inscriptions expirent après environ 90 secondes, donc les frontends doivent se réinscrire périodiquement.

Réponse (201 Created) : l'objet d'inscription.

Événements

Événements de cycle de vie par message. Les événements sont conservés dans ETS avec un TTL roulant.

Obtenir les Événements de Message

Requête :

```
GET /api/events/:id
```

`:id` est l'ID numérique du message.

Réponse (200 OK) :

```
{
  "message_id": 12345,
  "events": [
    { "event_epoch": 1698672000, "name": "message_inserted",
"description": "..." },
    { "event_epoch": 1698672005, "name": "message_delivered",
"description": "..." }
  ]
}
```

400 si l'id n'est pas un entier.

Enregistrer un Événement

Requête :

```
POST /api/events
Content-Type: application/json
```

Corps :

```
{
  "message_id": 12345,
  "name": "external_delivery_confirmed",
  "description": "Confirmé par le système en aval"
}
```

`message_id`, `name`, et `description` sont tous requis. La source de l'événement est détectée automatiquement à partir de l'IP de l'appelant (correspondant aux inscriptions de frontend actives), revenant au nom de nœud configuré.

Réponse (201 Created) :

```
{
  "status": "success",
  "message": "Événement enregistré avec succès",
  "event": {
    "message_id": 12345,
    "name": "external_delivery_confirmed",
    "description": "Confirmé par le système en aval",
    "event_source": "smpp-gateway-1",
    "event_epoch": 1698672010
  }
}
```

Campagnes

Messagerie en masse : téléchargez une liste de destinataires nommée, puis exécutez une campagne à débit limité contre celle-ci. Consultez le guide de Messagerie en Masse / Campagnes pour les concepts (filtre d'audience, taux de goutte, cycle de vie, statistiques).

Listes de Destinataires

```
GET    /api/campaign_lists           # toutes les listes
GET    /api/campaign_lists/:id       # une liste, inclut
"recipients"
POST   /api/campaign_lists           # créer une liste
PATCH  /api/campaign_lists/:id       # ajouter des destinataires
DELETE /api/campaign_lists/:id       # supprimer la liste et ses
destinataires
```

Chaque destinataire peut porter jusqu'à quatre variables de modèle (`var1`..`var4`). Fournissez les destinataires sous forme de tableau JSON, de chaîne CSV brute, ou les deux. Les destinataires JSON peuvent être des chaînes MSISDN simples ou des objets avec `var1`..`var4` :

```
{
  "name": "Clients VIP",
  "description": "Abonnés de grande valeur",
  "recipients": [
    { "msisdn": "12025550101", "var1": "Alice", "var2": "Gold" },
    "12025550102"
  ]
}
```

Pour CSV, une ligne d'en-tête nommant une colonne `msisdn` / `destination_msisdn` / `number` / `phone` sélectionne la colonne MSISDN (sinon, la première colonne est utilisée) ; les colonnes restantes deviennent `var1`..`var4` dans l'ordre des colonnes :

```
{ "name": "Importé", "csv":
"msisdn,first,plan\n12025550101,Alice,Gold\n" }
```

`PATCH` ajoute des destinataires (il ne les remplace pas).

Réponse (201 Created) : `{ "id": 1, "name": "Clients VIP", "recipient_count": 2 }`

Campagnes

```
GET    /api/campaigns        # toutes les campagnes (paginé, les
plus récentes en premier)
GET    /api/campaigns/:id    # une campagne, y compris les
statistiques en direct
POST   /api/campaigns        # créer une campagne (commence en
brouillon)
DELETE /api/campaigns/:id    # arrêter une campagne mais conserver
son enregistrement
```

Corps de Création :

```
{
  "name": "Avis de maintenance",
  "message": "Maintenance prévue ce soir de 02:00 à 03:00.",
  "source_msisdn": "12345",
  "source_smsc": "IMS_SMSC",
  "list_id": 1,
  "audience": "active_only",
  "drip_tps": 50
}
```

Champs :

- `name`, `message`, `source_msisdn`, `source_smsc` - **requis**. `message` prend en charge des variables de modèle par destinataire (`{{ var1 }}`..`{{ var4 }}`, `{{ msisdn }}`).
- `list_id` - liste de destinataires. **Omettez pour cibler tous les abonnés actuellement actifs (enregistrés).**
- `audience` - `active_only` (par défaut) ou `all`. Ignoré lorsque `list_id` est omis.
- `drip_tps` - messages soumis dans la file d'attente par seconde (par défaut à partir de la configuration).
- `validity_hours` - validité du message en heures ; définit l'`expires` de chaque message.
- `deliver_after` - heure ISO 8601 optionnelle pour maintenir les messages jusqu'à (envoi programmé).

`DELETE` arrête la livraison mais conserve l'enregistrement afin que l'historique et les statistiques restent visibles ; il retourne la campagne annulée (200) et est purgé automatiquement une fois passé le TTL de conservation.

Réponse (`GET /api/campaigns/:id`) :

```
{
  "id": 7,
  "name": "Avis de maintenance",
  "list_id": 1,
  "audience": "active_only",
  "status": "running",
  "drip_tps": 50,
  "total_targets": 1000,
  "stats": {
    "total": 1000, "pending": 600, "queued": 300,
    "delivered": 90, "failed": 10, "skipped_inactive": 0,
    "dispatched": 400, "progress_percent": 40, "delivery_percent":
22
  }
}
```

Contrôler une Campagne

Requête :

```
POST /api/campaigns/control
Content-Type: application/json
```

Corps : `{ "campaign_id": 7, "action": "start" }`

`action` est l'un de `start`, `pause`, `resume`, `cancel`. Retourne la campagne mise à jour (200), `404` si non trouvée, ou `422` si l'action est invalide pour l'état actuel.

Résultats par Destinataire

```
GET /api/campaign_targets/:id              # :id est l'ID de la
campagne
GET /api/campaign_targets/:id?state=failed # filtre par état de
cible
```

Prend en charge la pagination `limit`/`offset` ; le nombre total (après tout filtre `state`) est dans l'en-tête de réponse `x-total-count`. Chaque entrée porte l'`state` du destinataire (`pending` / `queued` / `delivered` / `failed` / `skipped_inactive`).

Autres Familles de Points de Terminaison

Les familles de routes suivantes font partie de la même API mais sont documentées en profondeur dans leurs guides dédiés. Les chemins sont listés ici pour complétude.

Routage

```
GET    /api/routes            # lister les routes
GET    /api/routes/:id        # obtenir une route
POST   /api/routes            # créer une route
PATCH  /api/routes/:id        # mettre à jour
PUT    /api/routes/:id        # mettre à jour
DELETE /api/routes/:id        # supprimer
GET    /api/routes/export     # exporter toutes les routes
POST   /api/routes/import     # importation en masse
POST   /api/routes/simulate   # simulation d'une décision de route
```

Voir le guide de Routage SMS.

Traductions de Numéros

```
GET    /api/translations           # lister
GET    /api/translations/:id       # obtenir
POST   /api/translations           # créer
PATCH  /api/translations/:id       # mettre à jour
DELETE /api/translations/:id       # supprimer
GET    /api/translations/export    # exporter
POST   /api/translations/import    # importation en masse
POST   /api/translations/simulate  # simulation d'une traduction
```

Voir le guide de Traduction de Numéros.

Opérations

```
GET  /api/operations                    # index des opérations
POST /api/operations/clear_messages     # purger le magasin de
messages
POST /api/operations/clear_routes       # purger les routes
POST /api/operations/clear_translations # purger les traductions
POST /api/operations/clear_frontends    # purger les inscriptions
de frontend
```

Ce sont des actions de maintenance destructrices ; protégez le port API en conséquence.

Fédération

```
GET  /api/federation/status          # statut de la fédération
GET  /api/federation/identity        # identité de fédération de
ce nœud
POST /api/federation/health          # rapport de santé des pairs
POST /api/federation/registry        # échange de registre
POST /api/federation/notify          # notification de message
entrant
POST /api/federation/delivery_status # rappel de statut de
livraison
GET  /api/federation_messages        # vue de message fédéré
```

Voir le guide de Fédération.

Diameter, Journaux, et DNS

```
GET  /api/diameter   # statut de pair/dip Diameter Sh
GET  /api/logs       # journaux structurés récents
POST /api/naptr      # test de résolution NAPTR
```

La recherche d'abonnés Diameter Sh est décrite dans le guide de recherche d'abonnés HSS.

Codes de Statut HTTP

Code	Signification	Utilisation Typique
200	OK	Lecture ou mise à jour réussie
201	Créé	Ressource créée
202	Accepté	Accepté pour traitement asynchrone
204	Pas de Contenu	Suppression réussie
400	Mauvaise Requête	Entrée mal formée (par exemple, id non entier, champ invalide)
404	Non Trouvé	Ressource inconnue
422	Entité Non Traitable	Échec de validation
500	Erreur Interne du Serveur	Erreur serveur inattendue

Format d'Erreur

Les erreurs au niveau du contrôleur retournent une chaîne `error` plate :

```
{ "error": "Message not found" }
```

Les erreurs au niveau du framework (itinéraires non correspondants, corps non analysables) retournent l'enveloppe normalisée :

```
{ "status": "error", "response": "Not Found" }
```

Meilleures Pratiques

Soumission de Message

1. **Pré-encodé vs texte clair** : envoyez `payload_hex` à `/api/messages` lorsque vous avez déjà encodé le TPDU ; sinon, utilisez `/api/messages_plaintext` et laissez le SMS-C encoder et fractionner.
2. **Toujours définir** `source_smsc` afin que les événements et le routage puissent attribuer le message.
3. **Utilisez des numéros au format E.164** de manière cohérente.
4. **Gérez 422** en inspectant la chaîne `error` plutôt qu'en réessayant à l'aveugle.

Intégration de Frontend

1. **Se réinscrire** sur `POST /api/frontends` bien avant la fenêtre d'expiration d'environ 90 secondes.
2. **Sonder** avec l'en-tête `smsc` pour vos messages ; utilisez `include-unrouted` uniquement si vous souhaitez délibérément du trafic non routé.
3. **Marquer comme livré** via `/api/messages_mark_delivered` après succès.
4. **Incrémenter en cas d'échec** via `PUT /api/messages/:id` (ou `/api/messages_increment_attempt`) pour planifier une nouvelle tentative

avec retour.

5. **Enregistrer des événements** pour les confirmations en aval.

Opérationnel

1. Terminez TLS et restreignez le port API au niveau réseau ; il n'y a pas d'authentification intégrée.
2. Réutilisez les connexions HTTP et définissez des délais d'attente client raisonnables.
3. Surveillez les métriques Prometheus (voir METRICS.md) pour la santé de la file d'attente et de la livraison.

Documentation de conformité à l'interception ANSSI R226

Objectif du document : Ce document fournit les spécifications techniques requises pour l'autorisation ANSSI R226 en vertu des articles R226-3 et R226-7 du Code pénal français pour le Centre de services SMS OmniMessage (SMSc).

Classification : Documentation de conformité réglementaire

Autorité cible : Agence nationale de la sécurité des systèmes d'information (ANSSI)

Réglementation : R226 - Protection de la vie privée des correspondances et interception légale

---

1. SPÉCIFICATIONS TECHNIQUES DÉTAILLÉES

1.1 Fiche technique commerciale

Nom du produit : OmniMessage SMSc (Centre de services SMS)
Type de produit : Centre de messages de télécommunications
Fonction principale : Routage, stockage et livraison de messages SMS
Protocoles réseau : REST API (HTTPS), protocoles SMS (SMPP, IMS, SS7/MAP via des frontaux externes)
Modèle de déploiement : Application serveur sur site
Technologie : Elixir/Erlang, Phoenix Framework, Mnesia, MySQL/PostgreSQL

Capacités principales

Traitement des messages :

- File d'attente de messages SMS centralisée avec REST API
- Conception indépendante du protocole prenant en charge les frontaux SMPP, IMS, SS7/MAP
- Moteur de routage dynamique avec routage basé sur les préfixes
- Logique de réessai avec retour exponentiel
- Gestion de l'expiration des messages et de la file d'attente des lettres mortes
- Génération et archivage des enregistrements de détails d'appel (CDR)
- Performance : ~1 750 messages/seconde de taux d'insertion, capacité de 150 millions de messages/jour

Stockage des messages :

- **File d'attente de messages active :** Base de données en mémoire Mnesia avec persistance sur disque optionnelle
  - Stockage principal : RAM pour un accès ultra-rapide (latence inférieure à la milliseconde)
  - Sauvegarde sur disque : le mode `disc_copies` écrit sur disque pour la récupération après un crash
  - Récupération automatique : Les messages survivent aux redémarrages du système
  - Conservation : Configurable (par défaut 24 heures), puis nettoyage automatique
- **Archive CDR à long terme :** Base de données MySQL/PostgreSQL (séparée de la file d'attente de messages)
  - Les CDR sont écrits lorsque les messages sont livrés, expirés, échoués ou rejetés
  - Base de données SQL utilisée UNIQUEMENT pour l'exportation/archivage des CDR, PAS pour les opérations de message actives
  - Aucun impact sur la performance du routage des messages (écriture asynchrone)
- **Avantages de l'architecture à deux niveaux :**

    - File active : Ultra-rapide (1 750 msg/sec) sans goulet d'étranglement SQL
    - Archive CDR : Conservation à long terme (mois/années) pour la facturation et l'interception légale
    - Séparation claire : Les opérations de message ne touchent jamais SQL
- Support de cluster pour haute disponibilité (réplication Mnesia entre les nœuds)

Interfaces réseau :

- **REST API :** HTTPS (port 8443) pour la communication avec les frontaux externes
- **Panneau de contrôle :** HTTPS (port 8086) pour la gestion basée sur le web
- **Protocoles frontaux :** SMPP, IMS, SS7/MAP (via des applications de passerelle externes)
- **Base de données :** MySQL/PostgreSQL pour le stockage des CDR

Routage et traitement :

- Routage SMS dynamique avec mises à jour de configuration à l'exécution
- Correspondance basée sur le préfixe (numéros appelants/appelés)
- Filtrage par SMSC source et type
- Équilibrage de charge basé sur la priorité et le poids
- Traduction et normalisation des numéros
- Support de recherche DNS ENUM (E.164 Number Mapping)
- Capacités de réponse automatique et de suppression de messages
- Contrôle de facturation par route (intégration OCS sur Diameter Ro)

 Architecture complète et fonctionnalités documentées dans README.md

1.2 Capacités d'interception

OmniMessage implémente les interfaces d'interception ETSI TS 103 221 (X1, X2, X3). OmniMessage agit comme l'élément réseau et le point d'interception.

Une installation de surveillance des forces de l'ordre provisionne les cibles via X1. OmniMessage livre les SMS interceptés en tant que 3GPP TS 33.128 xIRI via X2. Pour les rôles d'interface et la référence complète des paramètres, voir Interception légale (X1/X2/X3).

1.2.1 Acquisition de messages

Capture de messages SMS :

- Le SMSc OmniMessage traite tous les messages SMS entre les abonnés et les réseaux externes
- Accès complet aux métadonnées et au contenu des messages, y compris :
  - MSISDN source (numéro mobile)
  - MSISDN de destination (numéro mobile)
  - IMSI source (Identité d'abonné mobile internationale)
  - IMSI de destination
  - Corps du message (contenu texte)
  - Données brutes PDU (Unité de données de protocole)
  - Informations TP-DCS (Schéma de codage des données)
  - Encodage du message (GSM7, UCS-2, 8 bits, Latin-1)
  - Indicateurs de message multipart et données de réassemblage
  - Informations sur l'en-tête de données utilisateur (UDH)

Acquisition de métadonnées de message :

- Enregistrements de détails d'appel (CDR) complets stockés dans la base de données avec :
  - ID de message (identifiant unique)
  - Numéro appelant (MSISDN source)
  - Numéro appelé (MSISDN de destination)
  - Horodatage de soumission (lorsque le message est entré dans le système)
  - Horodatage de livraison (lorsque le message a été livré)
  - Horodatage d'expiration (lorsque le message a expiré s'il était non livrable)
  - Statut (livré, expiré, échoué, rejeté)

    - Compte des tentatives de livraison
    - Parties de message (pour les SMS concaténés/multi-part)
    - Identifiant SMSC source
    - Identifiant SMSC de destination
    - Nœud d'origine (nom du nœud de cluster Erlang)
    - Nœud de destination (pour les déploiements distribués)
    - Indicateur de lettre morte (indicateur d'épuisement des réessais)

📄 Schéma CDR complet documenté dans CDR_SCHEMA.md

Accès à la file d'attente de messages :

- Surveillance en temps réel de la file d'attente de messages
- Points de terminaison REST API pour la récupération de messages
- Requêtes de base de données pour la recherche historique de messages
- Capacités de filtrage par :
    - Numéro de téléphone (source/destination)
    - Passerelle SMSC
    - Plage horaire
    - Statut du message
    - Tentatives de livraison

📄 Documentation complète de l'API dans API_REFERENCE.md

1.2.2 Capacités de traitement des données

Architecture de stockage des messages (système à deux niveaux) :

Le SMSc utilise une architecture de stockage sophistiquée à deux niveaux qui sépare le traitement opérationnel des messages de l'archivage à long terme :

Niveau 1 : File d'attente de messages active (Mnesia)

- **Objectif :** Routage et opérations de livraison de messages en temps réel
- **Technologie :** Base de données distribuée Erlang Mnesia
- **Mode de stockage :** En mémoire avec sauvegarde disc_copies
    - Stockage principal dans la RAM pour une vitesse maximale

    - Synchronisation automatique du disque pour la récupération après un crash
    - Les messages persistent à travers les redémarrages du système
- **Performance :** Opérations de lecture/écriture inférieures à la milliseconde
- **Conservation :** À court terme (par défaut 24 heures), configurable
- **Nettoyage :** Archivage automatique vers la base de données CDR, puis suppression de Mnesia
- **Opérations :** Toutes les opérations de file d'attente de messages (insertion, mise à jour, statut de livraison, routage)
- **Caractéristique critique :** La base de données SQL n'est JAMAIS interrogée lors du routage/de la livraison des messages

Niveau 2 : Archive CDR (MySQL/PostgreSQL)

- **Objectif :** Stockage à long terme pour la facturation, l'analyse et l'interception légale
- **Technologie :** Base de données SQL traditionnelle (MySQL ou PostgreSQL)
- **Déclencheur d'écriture :** Les CDR sont écrits UNIQUEMENT lorsque les messages atteignent un état final :
    - Message livré avec succès
    - Message expiré (délai de validité dépassé)
    - Message échoué de manière permanente
    - Message rejeté par les règles de routage
- **Mode d'écriture :** Écriture par lot asynchrone (aucun impact sur la performance du routage des messages)
- **Conservation :** À long terme (mois à années), configurable selon les exigences réglementaires
- **Opérations :** Requêtes historiques, rapports, conformité, interception légale
- **Accès :** Requêtes SQL, REST API (futur), export CSV/JSON

Avantages architecturaux clés :

1. **Performance :** Les opérations de routage actives ne touchent jamais SQL (pas de goulet d'étranglement de base de données)

2. **Scalabilité :** Mnesia gère plus de 1 750 messages/seconde sans surcharge SQL
3. **Fiabilité :** Le mode disc_copies garantit aucune perte de message en cas de crash
4. **Conformité :** La base de données CDR fournit une piste d'audit permanente
5. **Séparation des préoccupations :** Données opérationnelles vs. données d'archivage clairement séparées

Cycle de vie du message :

```
1. Message soumis → Stocké dans Mnesia (RAM + sauvegarde sur
disque)
2. Message routé → Requête Mnesia (ultra-rapide)
3. Message livré/expiré → CDR écrit dans SQL (asynchrone)
4. Après 24h → Message supprimé de Mnesia (travail de nettoyage)
5. CDR reste dans SQL → Disponible pour les requêtes
d'interception légale (années)
```

Conservation et récupération des données :

- Conservation ou suppression configurable du corps du message pour la confidentialité
- Préservation des données binaires (stockage PDU brut dans Mnesia et CDR)
- Capacité de recherche en texte intégral (si activée sur la base de données CDR)
- Champs CDR indexés pour des requêtes d'interception légale rapides

Suivi des frontaux :

- Suivi en temps réel des frontaux SMSC externes (SMPP, IMS, passerelles MAP)
- Enregistrement des frontaux avec surveillance de l'état
- Suivi de l'état de santé (actif/expiré)
- Historique de disponibilité/indisponibilité
- Suivi des adresses IP et des noms d'hôte
- Journalisation de la configuration spécifique aux frontaux

1.2.3 Capacités d'analyse

Surveillance en temps réel :

- Tableau de bord de l'interface utilisateur web montrant :
  - File d'attente de messages active
  - Soumission et livraison de messages
  - Décisions de routage et sélection de passerelle
  - État de la passerelle frontale
  - Utilisation des ressources système
- Intégration des métriques Prometheus pour la surveillance opérationnelle
- Métriques de performance (débit, latence, taux de réussite)

 Guide de surveillance complet dans OPERATIONS_GUIDE.md
 Documentation des métriques dans METRICS.md

Analyse historique :

- Base de données CDR interrogeable par :
  - Plage horaire
  - Numéro de l'appelant/de l'appelé
  - Statut du message
  - Passerelle SMSC
  - Tentatives de livraison
  - Contenu du message (recherche en texte intégral si activée)
- Capacités d'analyse statistique :
  - Volume de messages par heure/jour/mois
  - Taux de succès/échec par route
  - Temps de livraison moyen
  - Analyse des messages multi-part
  - Modèles de livraison échouée

Suivi des abonnés :

- Historique des messages par numéro de téléphone (MSISDN)
- Suivi basé sur l'IMSI (lorsqu'il est disponible à partir des frontaux IMS/MAP)

- Analyse des modèles d'appel
- Corrélation des parties communicantes
- Analyse temporelle (fréquence des messages, modèles de timing)

Analyse réseau :

- Métriques de performance des routes
- Disponibilité et état de santé des passerelles
- Visualisation du flux de messages
- Distribution des nœuds de cluster (déploiements multi-nœuds)
- Analyse des tentatives de livraison
- Analyse des modèles de réessai

Intelligence des numéros :

- Normalisation des numéros E.164
- Identification du pays/région à partir du préfixe du numéro
- Règles de traduction et de réécriture des numéros
- Recherche DNS ENUM pour l'intelligence de routage
- Décisions de routage basées sur les préfixes

 Guide de traduction des numéros dans number_translation_guide.md
 Guide de routage dans sms_routing_guide.md

1.3 Capacités de contre-mesures

1.3.1 Mécanismes de protection de la vie privée

Confidentialité des communications :

- HTTPS/TLS pour les communications REST API
- Authentification basée sur des certificats
- Chiffrement de la connexion à la base de données (support TLS)
- Suppression configurable du corps du message après livraison

Contrôle d'accès :

- Contrôle d'accès à l'interface utilisateur web
- Mécanismes d'authentification API
- Contrôles d'accès à la base de données
- Authentification de l'enregistrement des frontaux

Journalisation des audits :

- Journalisation complète des événements système
- Journalisation de la soumission/livraison des messages
- Suivi des modifications de configuration
- Journalisation des actions administratives
- Journalisation structurée avec niveaux configurables

1.3.2 Fonctionnalités de protection des données

Vie privée des messages :

- Suppression configurable du corps du message après livraison
- Corps du message exclu de l'affichage de l'interface utilisateur (optionnel)
- Corps du message exclu des exports (optionnel)
- Champ du corps du message CDR peut être défini sur NULL pour la confidentialité

Sécurité de la base de données :

- Support de chiffrement des tables MySQL (ENCRYPTION='Y')
- Support de chiffrement de données transparent pour PostgreSQL
- Séparation des rôles d'accès à la base de données
- Comptes utilisateurs en lecture seule pour l'analyse
- Accès restreint au contenu des messages

Renforcement du système :

- Ports réseau exposés minimaux
- Gestion des certificats TLS
- Stockage sécurisé de la configuration
- Séparation de la configuration basée sur l'environnement

- Sécurité de cluster avec protocole de distribution Erlang

1.4 Architecture de stockage : Conception à deux niveaux Mnesia + SQL

Aperçu

Le SMSc OmniMessage emploie une architecture de stockage unique à deux niveaux spécifiquement conçue pour séparer le traitement opérationnel des messages à haute performance de l'archivage et de la conformité à long terme.

Niveau 1 : File d'attente de messages en mémoire Mnesia

Qu'est-ce que Mnesia ?

- Base de données distribuée intégrée dans l'environnement d'exécution Erlang/OTP
- Stockage hybride : Principalement en mémoire avec sauvegarde automatique sur disque
- Transactions conformes à ACID
- Réplication de cluster à travers plusieurs nœuds

Mode de stockage : disc_copies

- **Principal en mémoire :** Tous les messages actifs stockés dans la RAM
  - Opérations de lecture/écriture ultra-rapides (inférieures à la milliseconde)
  - Pas d'E/S disque pendant les opérations normales de routage de messages
  - Permet un débit de plus de 1 750 messages/seconde
- **Sauvegarde sur disque (automatique) :** Mnesia synchronise la RAM avec le disque
  - Les écritures se font de manière asynchrone en arrière-plan
  - La copie sur disque est mise à jour à chaque validation de transaction
  - Récupération après crash : Le système redémarre avec tous les messages intacts
  - Emplacement : répertoire `Mnesia.*/` dans les données de l'application

Cycle de vie du message dans Mnesia :

1. Message arrive via REST API → Inséré dans Mnesia RAM + sauvegarde sur disque
2. Moteur de routage interroge Mnesia → Réponse instantanée (accès mémoire)
3. Passerelle externe interroge pour des messages → Requête Mnesia (accès mémoire)
4. La passerelle met à jour le statut de livraison → Mise à jour Mnesia (mémoire + disque)
5. Après livraison/expiration → Message marqué pour nettoyage
6. Travailleur de nettoyage (par défaut 24h) → Message supprimé de Mnesia

Caractéristique de performance critique :

- **AUCUNE requête de base de données SQL** pendant le routage/de la livraison active des messages
- SQL est complètement contourné pour le traitement opérationnel des messages
- Cela élimine le goulet d'étranglement traditionnel du SMS-C (E/S de base de données)

Niveau 2 : Base de données SQL pour l'exportation/archivage des CDR

Qu'est-ce que CDR (Enregistrement de détails d'appel) ?

- Enregistrement d'audit permanent des métadonnées et du contenu des messages
- Écrit dans la base de données MySQL ou PostgreSQL
- Utilisé pour la facturation, l'analyse, la conformité et l'interception légale

Quand les CDR sont écrits : Les enregistrements CDR sont créés UNIQUEMENT lorsque les messages atteignent un état final :

- □ Message livré avec succès
- □ Message expiré (délai de validité dépassé sans livraison)
- □ Message échoué de manière permanente (numéro invalide, erreur de routage)

- □ Message rejeté (règles de routage, échec de validation)

Comment les CDR sont écrits :

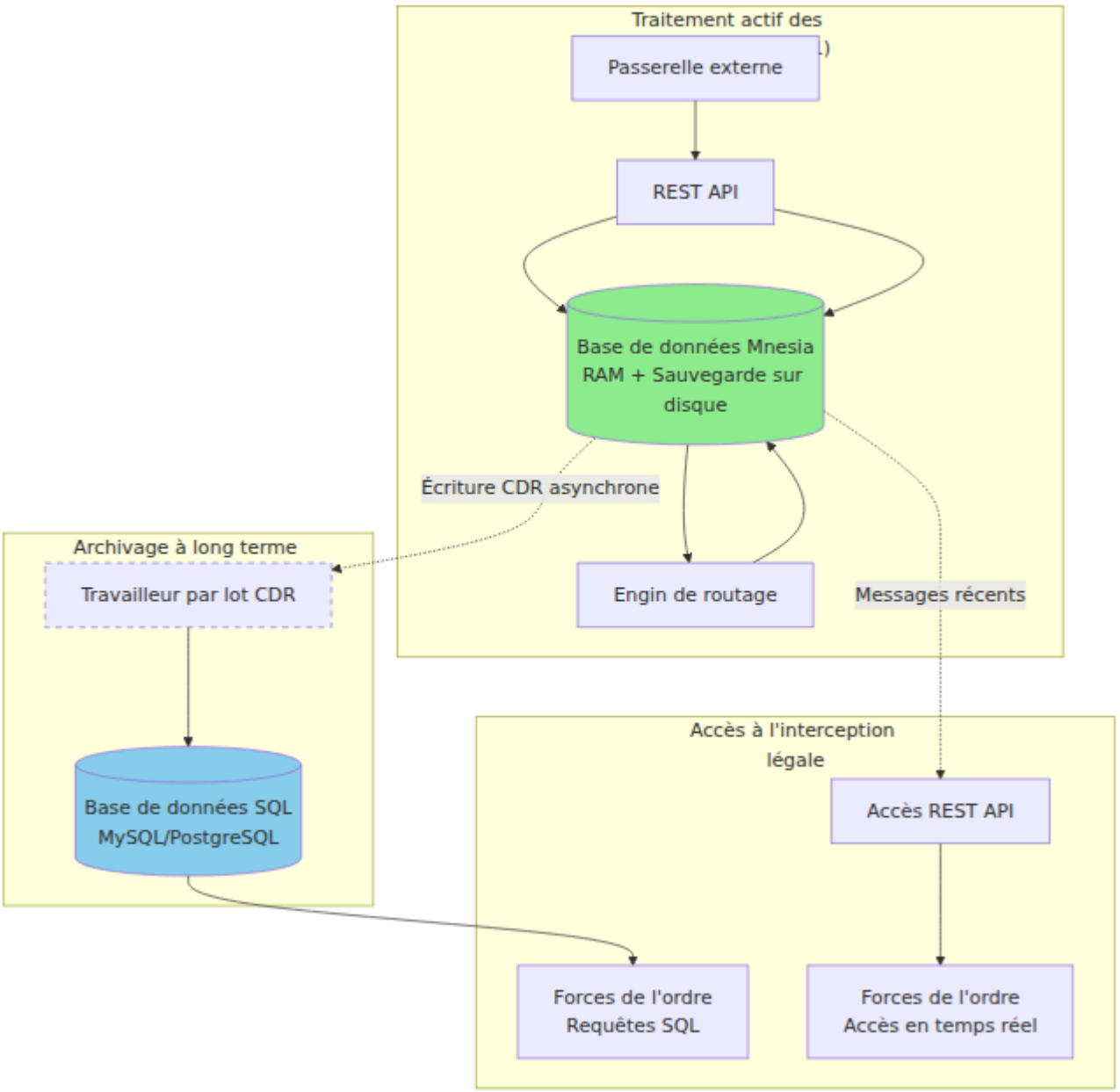
- **Écriture par lot asynchrone :** Les CDR sont écrits dans un processus de travail en arrière-plan
- **Pas de blocage :** Le routage des messages n'attend jamais l'écriture SQL
- **Inserts groupés :** Plusieurs CDR regroupés (par défaut 100) et écrits ensemble
- **Intervalle de vidage :** 100 ms par défaut (configurable)
- **Gestion des erreurs :** Les écritures CDR échouées sont enregistrées, le traitement des messages continue

```
# Configuration dans config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,          # Taille du lot pour les
écritures CDR
  batch_insert_flush_interval_ms: 100    # Intervalle de vidage
```

Objectif de la base de données SQL :

- □ NON utilisé pour : Opérations de file d'attente de messages actives
- □ NON utilisé pour : Décisions de routage des messages
- □ NON utilisé pour : Livraison de messages en temps réel
- □ UNIQUEMENT utilisé pour : Archivage à long terme des CDR et requêtes historiques
- □ UNIQUEMENT utilisé pour : Requêtes d'interception légale (mois/années d'historique)
- □ UNIQUEMENT utilisé pour : Rapports de facturation et d'analyse

Diagramme d'architecture



Légende :

- Lignes solides : opérations synchrones (temps réel)
- Lignes en pointillés : opérations asynchrones (arrière-plan)
- Vert : niveau haute performance (en mémoire)
- Bleu : niveau d'archivage (SQL persistant)

Implications de l'interception légale

Messages récents (< 24 heures) :

- Accessibles via Mnesia (requêtes REST API)
- Récupération ultra-rapide

- Contenu complet du message disponible
- Surveillance en temps réel possible

Messages historiques (> 24 heures) :

- Accessibles via la base de données SQL (table CDR)
- Performance standard des requêtes SQL
- Métadonnées complètes des messages toujours disponibles
- Corps du message disponible (sauf si le mode de confidentialité est activé)

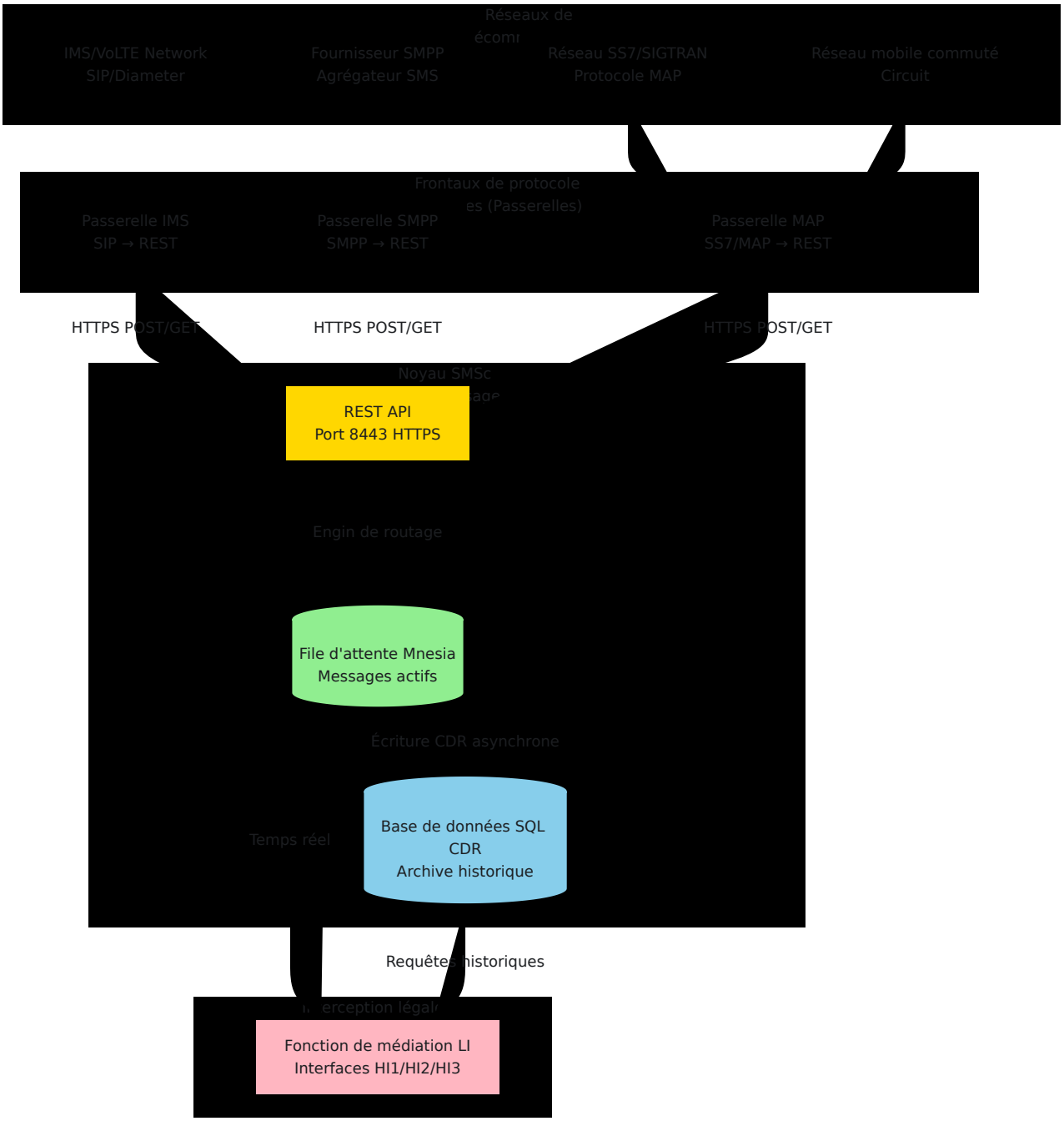
Avantages de conformité :

1. **Pas de perte de données :** Le mode disc_copies garantit que les messages survivent aux crashs
2. **Piste d'audit permanente :** Les CDR sont conservés pendant des années dans la base de données SQL
3. **Performance :** Les requêtes d'interception légale n'impactent pas le routage des messages
4. **Flexibilité :** Messages récents (Mnesia) + messages historiques (SQL) tous deux accessibles

1.5 Architecture d'intégration multi-protocoles des frontaux

Le SMSc OmniMessage emploie une conception centrale indépendante du protocole qui s'interface avec des passerelles spécifiques aux protocoles externes via une API REST unifiée. Cette architecture permet à l'interception légale de capturer des messages, quel que soit le protocole de télécommunications utilisé pour les envoyer ou les recevoir.

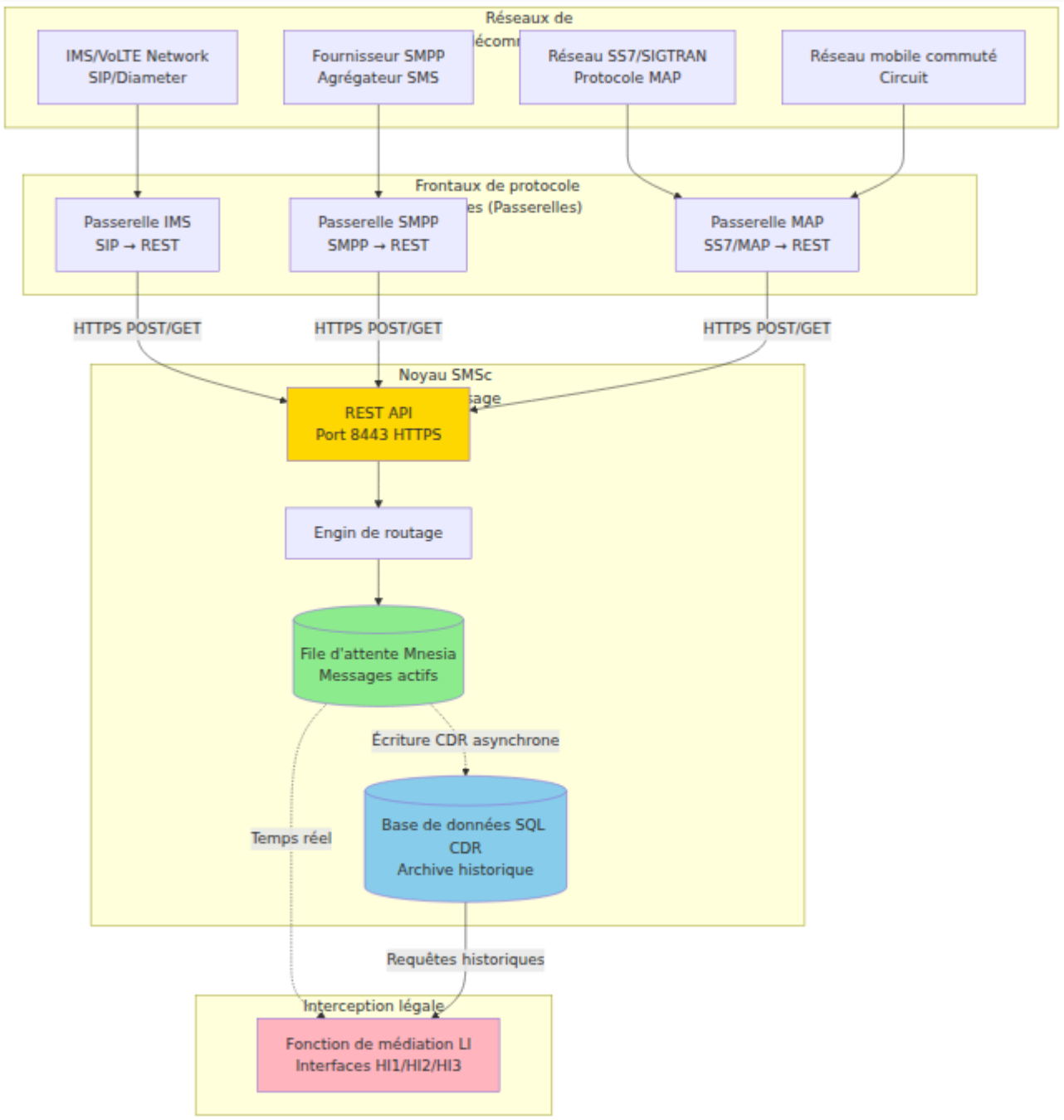
Aperçu de l'architecture



Détails de l'intégration des protocoles frontaux

1. Intégration du frontal IMS/SIP

Les réseaux IMS utilisent le protocole SIP pour la messagerie SMS sur IP. La passerelle IMS traduit entre SIP et l'API REST du SMSc.



Données d'interception spécifiques à IMS :

- IMSI source/destination (à partir de l'enregistrement IMS)
- En-têtes SIP P-Asserted-Identity
- ID d'appel SIP pour corrélation
- Emplacement réseau IMS (P-Access-Network-Info)
- Profils d'abonnés à partir de l'HSS IMS

2. INTÉGRATION DU FRONTEND SMPP

SMPP est le protocole standard de l'industrie pour les agrégateurs SMS et les fournisseurs de services. La passerelle SMPP traduit les messages SMPP basés sur PDU en appels d'API REST.

```mermaid
sequenceDiagram
    participant ESME as Client SMPP Externe<br/> (ESME)
    participant GW as Passerelle SMPP<br/>(Frontend SMPP)
    participant API as API REST SMSc
    participant Queue as File d'attente Mnesia

    ESME->>GW: BIND_TRANSMITTER<br/>system_id: "customer123"
    GW-->>ESME: BIND_TRANSMITTER_RESP

    ESME->>GW: SUBMIT_SM<br/>source_addr: "447700900123"
<br/>dest_addr: "447700900456"<br/>short_message: "Test SMS"
<br/>data_coding: 0 (GSM7)

    Note over GW: Décoder le PDU SMPP<br/>Extraire tous les champs
TP-DCS<br/>Extraire UDH si présent

    GW->>API: POST /api/messages<br/>&#123;<br/>  "source_msisdn":
"447700900123",<br/>  "destination_msisdn": "447700900456",<br/>
"message_body": "Test SMS",<br/>  "source_smsc":
"smpp.customer123",<br/>  "tp_dcs_character_set": "gsm7",<br/>
"tp_dcs_coding_group": "general_data_coding",<br/>  "raw_pdu":
"base64_encoded_pdu"<br/>&#125;

    API->>Queue: Insérer le message
    Queue-->>API: ID du message: 12346
    API-->>GW: 201 Created<br/>&#123;id: 12346&#125;

    GW-->>ESME: SUBMIT_SM_RESP<br/>message_id: "12346"

    Note over GW,Queue: [POINT D'INTERCEPTION]<br/>PDU SMPP
complet préservé

    loop Polling pour la livraison
        GW->>API: GET /api/messages/get_by_smsc?
smsc=smpp.provider-01
        API-->>GW: Messages à livrer
```

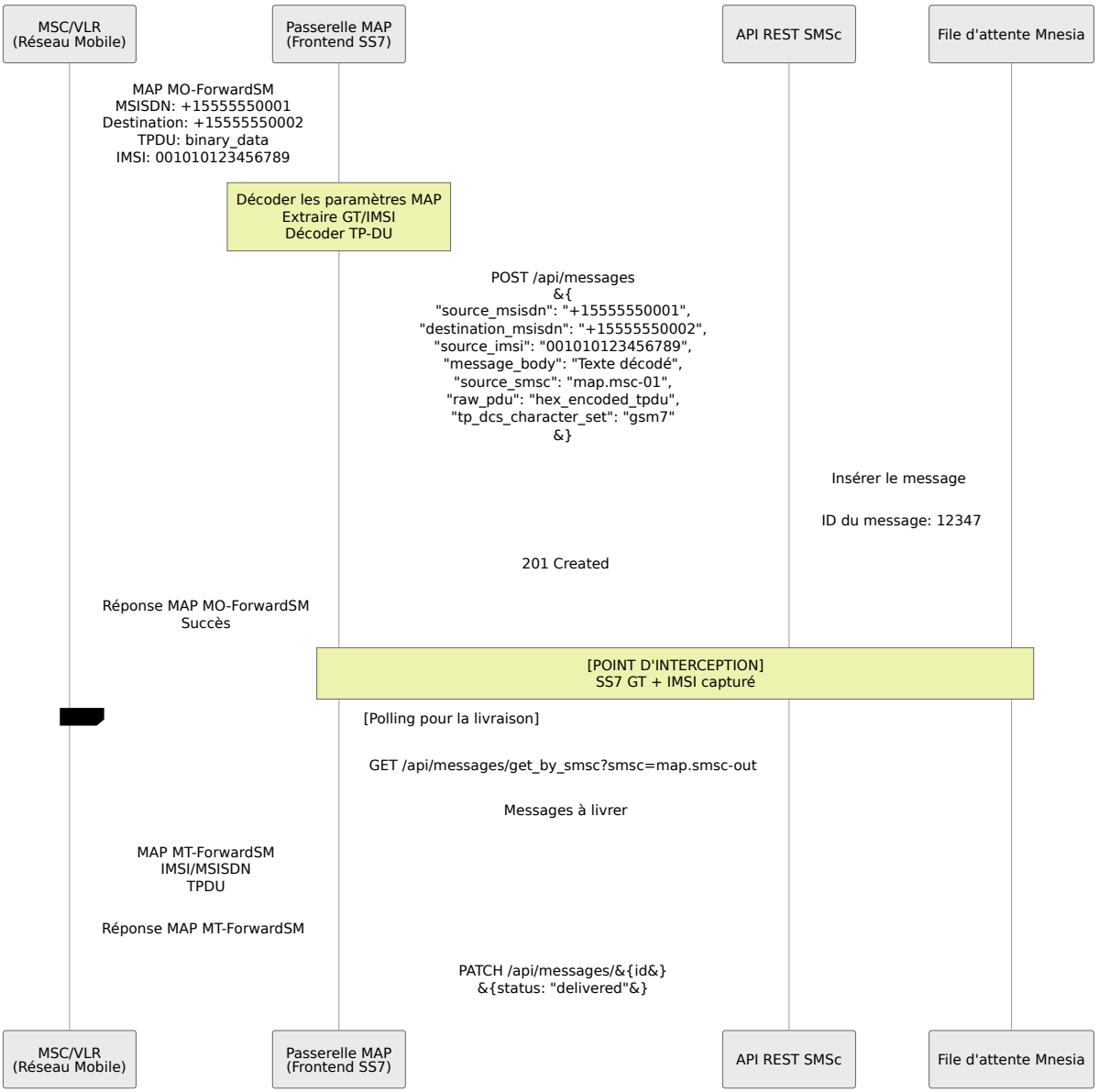
```
        GW->>ESME: DELIVER_SM
        ESME-->>GW: DELIVER_SM_RESP
        GW->>API: PATCH /api/messages/&#123;id&#125;
<br/>&#123;status: "delivered"&#125;
    end
```

Données d'Interception Spécifiques à SMPP :

- PDU SMPP complet (format binaire préservé)
- Détails du Schéma de Codage de Données (DCS)
- En-tête de Données Utilisateur (UDH) pour les messages concaténés
- system_id ESME (identification du client)
- Informations sur le plan de numérotation TON/NPI
- Indicateurs de livraison enregistrés

3. INTÉGRATION DU FRONTEND SS7/MAP

Les réseaux hérités à commutation de circuits utilisent le protocole SS7 MAP pour les SMS. La passerelle MAP traduit entre le signalement SS7 et l'API REST.



Données d'Interception Spécifiques à SS7/MAP :

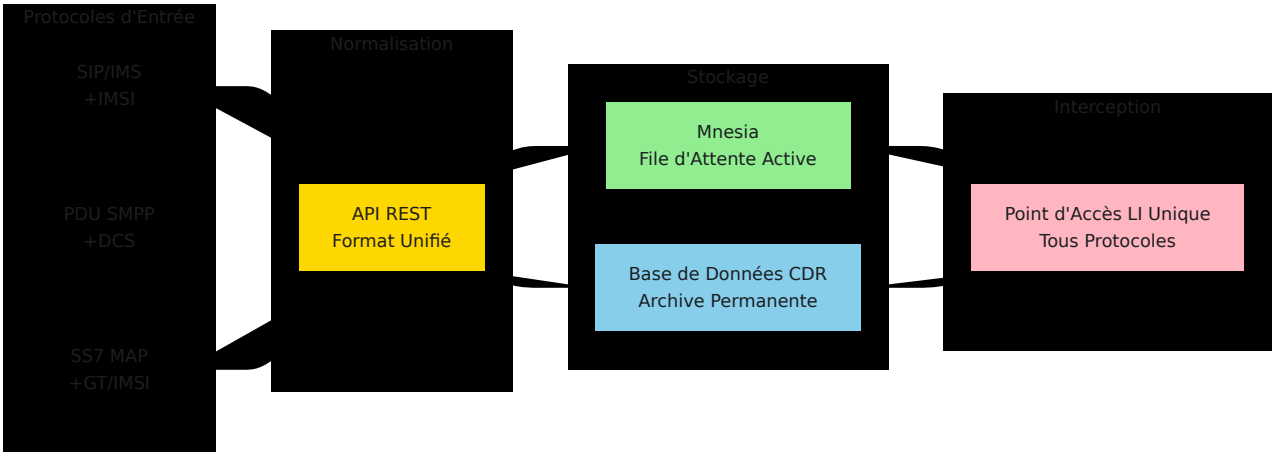
- IMSI des messages MAP
- Adresses de Titre Global (GT)
- Adresse MSC/VLR (identification de l'élément réseau)
- Adresses de parties appelantes/appelées SCCP
- Codes d'opération MAP
- Format binaire TP-User-Data

Interception Unifiée à Travers Tous les Protocoles

Avantage Clé pour l'Interception Légale : Quel que soit le protocole utilisé (IMS/SIP, SMPP ou SS7/MAP), tous les messages convergent dans le cœur SMSc avec une structure de données normalisée, permettant :

1. **Surveillance Indépendante du Protocole :** Un seul point d'interception capture tous les types de messages
2. **Format CDR Unifié :** Tous les protocoles écrivent dans le même schéma CDR
3. **Corrélation Inter-Protocole :** Suivre les messages à travers les frontières de protocole
4. **Préservation Complète des Métadonnées :** Champs spécifiques au protocole préservés dans le CDR

Résumé du Flux de Données :



Identification du Protocole dans le CDR :

- Le champ `source_smsc` indique le protocole frontend (par exemple, "ims.gateway-01", "smpp.customer123", "map.msc-01")
- Permet le filtrage et l'analyse par type de protocole
- Les requêtes d'interception légale peuvent cibler des protocoles spécifiques ou tous les protocoles

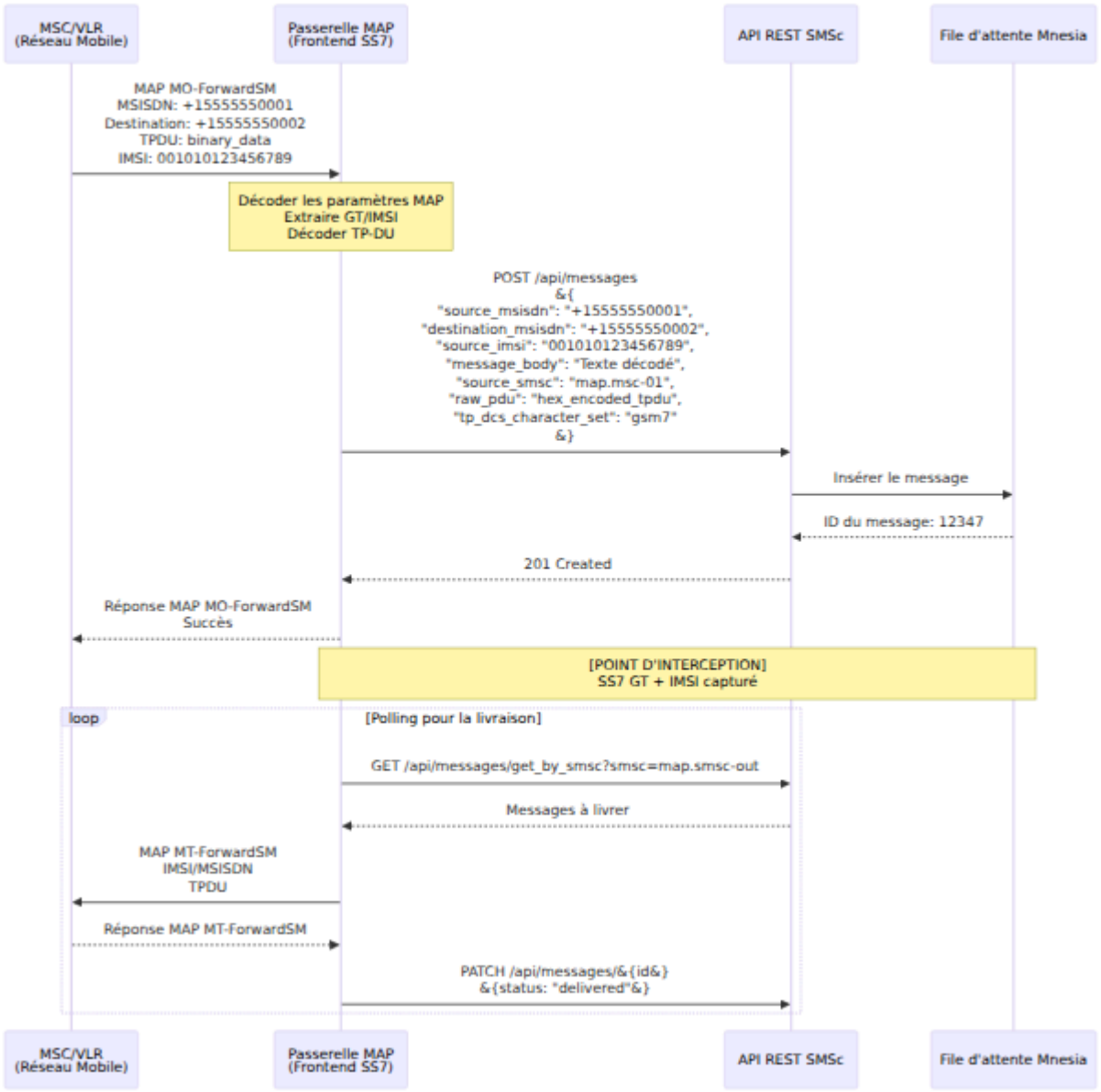
1.6 Architecture Technique pour l'Interception Légale

Points d'Intégration de l'Interception Légale

L'architecture de stockage à deux niveaux fournit plusieurs points d'accès pour l'interception légale, optimisés pour la surveillance en temps réel (Mnesia) et l'analyse historique (SQL).

1. Accès API REST pour les Messages Récents (Mnesia) :

Accès aux messages actifs dans la file d'attente Mnesia (généralement les dernières 24 heures) :



Points de Terminaison API pour l'Interception en Temps Réel :

- `GET /api/messages` - Lister les messages actifs avec filtrage
- `GET /api/messages/{id}` - Obtenir les détails d'un message spécifique (depuis Mnesia)
- `GET /api/messages/get_by_smsc?smsc=X` - Obtenir des messages par passerelle
- Toutes les requêtes touchent Mnesia (en mémoire) pour une réponse instantanée

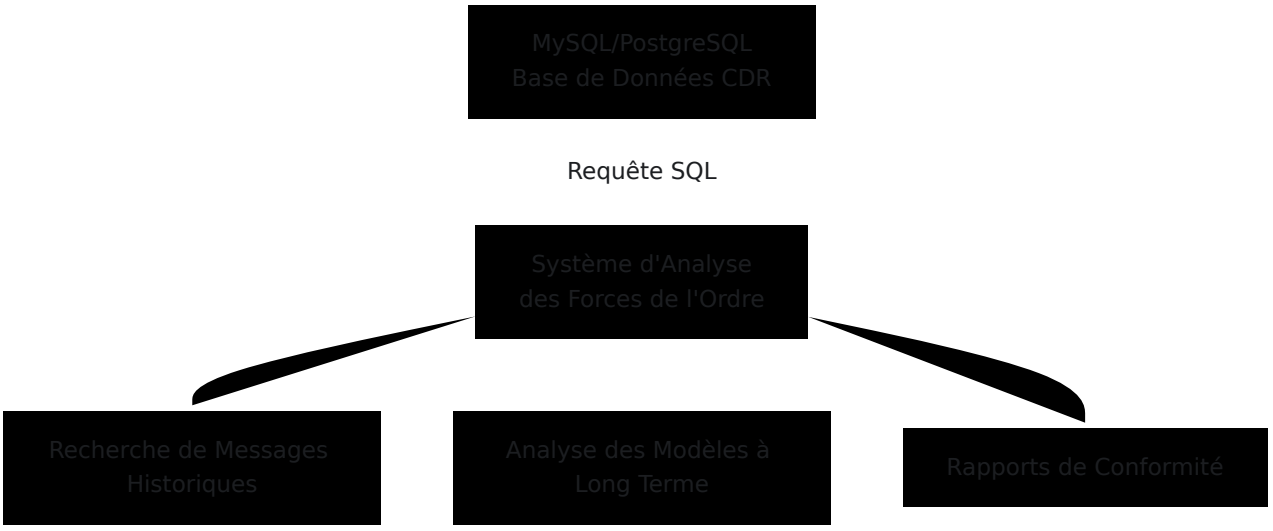
Remarque : Ces points de terminaison interrogent la file d'attente de messages active de Mnesia, fournissant l'accès aux messages actuellement traités ou récemment livrés (dans la période de conservation).

Paramètres de Requête :

- Filtrer par MSISDN source/destination
- Filtrer par plage horaire
- Filtrer par passerelle SMSC
- Filtrer par statut de message
- Support de tri et de pagination

2. Accès Direct à la Base de Données CDR pour les Messages Historiques (SQL) :

Accès aux messages archivés dans la base de données SQL (tous les messages livrés/expirés/échoués) :



Accès SQL Direct :

- Identifiants de base de données en lecture seule pour les systèmes autorisés
- Accès par requête SQL à la table `cdrs` (trace d'audit permanente)
- **Méthode d'Accès :** Client SQL standard (mysql, psql, DBeaver, etc.)
- **Source de Données :** Seulement les messages archivés (pas de file d'attente active)
- Champs indexés pour une recherche efficace :

- `calling_number` (indexé) - Numéro de téléphone source
- `called_number` (indexé) - Numéro de téléphone de destination
- `message_id` (indexé) - Identifiant unique du message
- `submission_time` (indexé) - Quand le message est entré dans le système
- `status` (indexé) - Statut final de livraison
- `dest_smsc` (indexé) - Passerelle utilisée pour la livraison

Remarque : La base de données CDR contient des enregistrements permanents de tous les messages traités. C'est la principale source de données pour les requêtes d'interception légale historiques (mois/années de données).

3. Flux de Messages en Temps Réel (PubSub) :

- Intégration Phoenix PubSub pour les événements en temps réel
- Notifications de soumission de messages
- Notifications de livraison de messages
- Événements de changement de statut de message
- Filtrage d'événements configurable par critères
- Support WebSocket pour la surveillance en direct

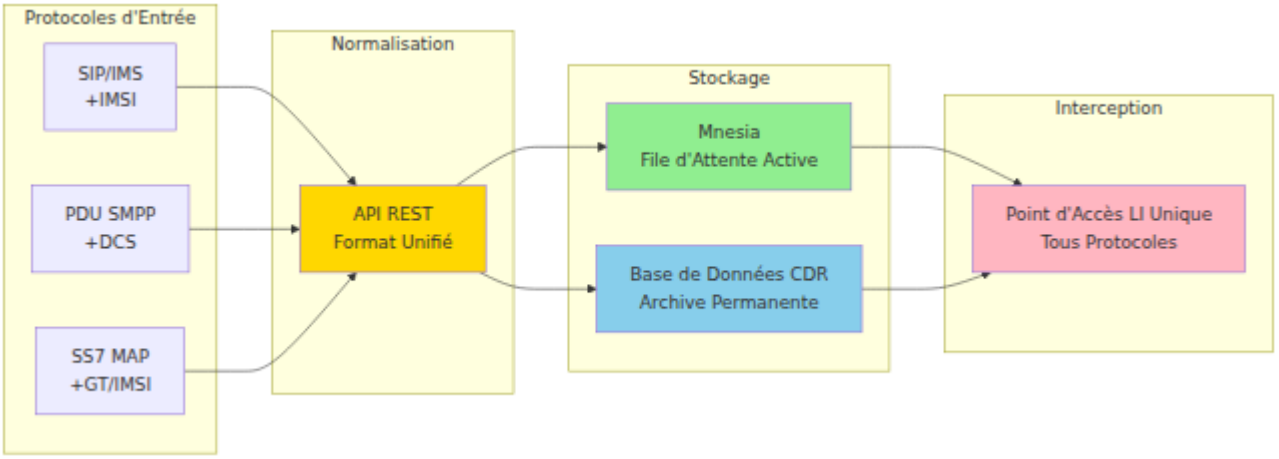
4. Interface d'Exportation par Lots :

- Exportation CSV des enregistrements CDR
- Exportation JSON pour un accès programmatique
- Champs d'exportation configurables
- Exportations basées sur des plages horaires
- Exportations respectueuses de la vie privée (exclusion optionnelle du corps du message)

Interfaces Standards d'Interception Légale ETSI

Le SMSc OmniMessage fournit la base pour la mise en œuvre des interfaces d'interception légale conformes à l'ETSI. Le cœur du SMSc n'implémente pas nativement les interfaces de transfert HI1/HI2/HI3. Il fournit tous les points d'accès de données nécessaires. Vous pouvez intégrer ces points d'accès avec des systèmes externes de Fonction de Médiation d'Interception Légale (LIMF).

Interfaces LI Standards ETSI :



Descriptions des Interfaces :

Interface HI1 - Fonction d'Administration :

- **But :** Provisionnement de mandat et de cible des forces de l'ordre au système d'interception
- **Direction :** LEMF → LIMF (bidirectionnel)
- **Fonctions :**
  - Activer/désactiver l'interception pour des cibles spécifiques (MSISDN, IMSIs)
  - Définir la durée et la période de validité de l'interception
  - Configurer les critères de filtrage (numéros de téléphone, fenêtres horaires)
  - Récupérer le statut de l'interception
- **Intégration avec SMSc :**
  - LIMF maintient la liste des cibles (base de données des mandats)
  - LIMF interroge le CDR/API du SMSc pour les messages correspondants
  - LIMF filtre en fonction des critères fournis par HI1

Interface HI2 - Livraison IRI (Informations Relatives à l'Interception) :

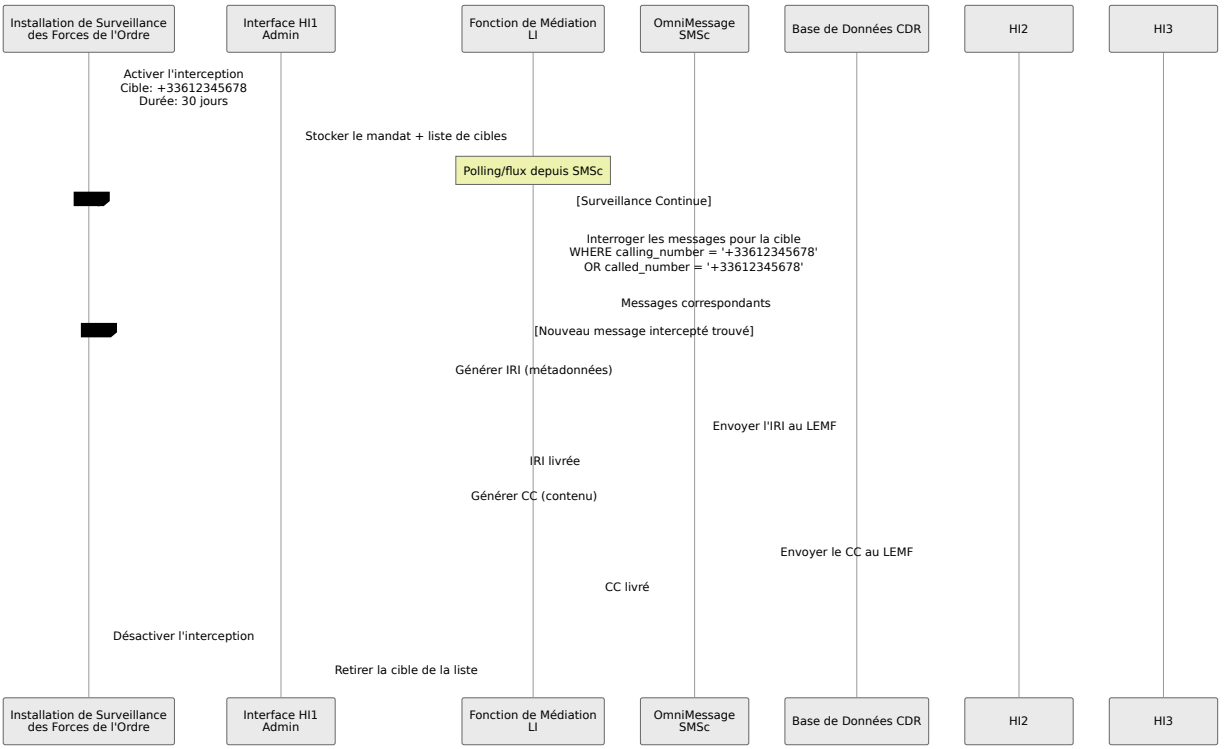
- **But :** Livrer les métadonnées des messages aux forces de l'ordre
- **Direction :** LIMF → LEMF (unidirectionnel)
- **Format de Données :** XML/ASN.1 conforme à l'ETSI TS 102 232-x
- **Contenu du CDR SMSc :**
  - ID du message

    - Numéro appelant (MSISDN source)
    - Numéro appelé (MSISDN destination)
    - IMSI (source et destination, si disponible)
    - Horodatage de soumission
    - Horodatage de livraison
    - Statut du message (livré/échoué/expiré)
    - Tentatives de livraison
    - Informations sur la passerelle SMSC (source/destination)
    - Localisation réseau (si disponible)
- **Intégration avec SMSc :**
    - LIMF interroge la base de données CDR pour les numéros de téléphone cibles
    - LIMF transforme les enregistrements CDR au format IRI conforme à l'ETSI
    - LIMF livre l'IRI au LEMF via HI2

Interface HI3 - Livraison CC (Contenu de la Communication) :

- **But :** Livrer le contenu réel du message aux forces de l'ordre
- **Direction :** LIMF → LEMF (unidirectionnel)
- **Format de Données :** Conforme à l'ETSI TS 102 232-x
- **Contenu du SMSc :**
    - Corps du message (contenu texte)
    - PDU brut (données SMS binaires)
    - Informations sur le codage des caractères
    - Segments de message multipart
    - Informations TP-DCS
    - En-tête de Données Utilisateur (UDH)
- **Intégration avec SMSc :**
    - LIMF récupère le contenu du message depuis le champ `message_body` du CDR
    - LIMF récupère les données PDU brutes si disponibles
    - LIMF emballe le contenu au format CC conforme à l'ETSI
    - LIMF livre le CC au LEMF via HI3

Architecture de Mise en Œuvre :



Mapping des Données SMSc vers les Interfaces LI :

Champ de Données SMSc	HI2 (IRI)	HI3 (CC)	Colonne de Table CDR
ID du message	□ ID de Corrélation	□ Référence	`message_id`
Numéro appelant	□ Partie A	-	`calling_number`
Numéro appelé	□ Partie B	-	`called_number`
Temps de soumission	□ Horodatage	-	`submission_time`
Temps de livraison	□ Achèvement	-	`delivery_time`
Statut	□ Résultat	-	`status`
Corps du message	-	□ Contenu	`message_body`
PDU brut	-	□ Binaire	(Mnesia/CDR)
SMSC Source	□ Élément réseau	-	`source_smsc`
SMSC Dest	□ Élément réseau	-	`dest_smsc`
IMSI	□ ID d'Abonné	-	(Via frontends)

Options d'Intégration LIMF :

Option 1 : Architecture de Polling

- LIMF interroge périodiquement la base de données CDR (toutes les 1 à 60 secondes)
- Requêtes SQL filtrées par numéros de téléphone cibles de la liste de mandats HI1

- Faible complexité, facile à mettre en œuvre
- Légère latence entre la livraison du message et la livraison LI

Option 2 : Architecture de Flux en Temps Réel

- PubSub SMSc publie des événements de message
- LIMF s'abonne au flux de messages en temps réel
- LIMF filtre en fonction de la liste de cibles
- Latence quasi nulle pour l'interception légale
- Nécessite un développement d'intégration personnalisé

Option 3 : Architecture Hybride

- Messages récents : Flux PubSub en temps réel (< 24 heures)
- Messages historiques : Polling de la base de données CDR
- Équilibre optimal entre latence et fiabilité

Mécanismes de Déclenchement de l'Interception

Interception Basée sur la Cible :

- Correspondance de numéro de téléphone (MSISDN)
- Ciblage basé sur l'IMSI (lorsqu'il est disponible)
- Listes de surveillance configurables
- Vues de base de données pour l'isolement des cibles
- Filtrage API par identifiants de cibles

Interception Basée sur les Événements :

- Tous les messages vers/depuis des numéros spécifiques
- Messages via des passerelles SMSC spécifiques
- Messages avec des caractéristiques spécifiques (multipart, livraison échouée, etc.)
- Routage géographique (via ENUM ou correspondance de préfixes)

Interception Basée sur le Temps :

- Filtrage par plage de date/heure dans les requêtes CDR

- Application de la période de conservation
- Archivage automatique des anciens messages
- Politiques de conservation des données configurables

Exemples de Requêtes SQL pour l'Interception Légale :

```sql
-- Obtenir tous les messages pour le numéro cible
SELECT * FROM cdrs
WHERE calling_number = '+33612345678'
   OR called_number = '+33612345678'
ORDER BY submission_time DESC;

-- Obtenir des messages dans une fenêtre temporelle spécifique
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' OR called_number =
'+33612345678')
  AND submission_time BETWEEN '2025-11-01 00:00:00' AND '2025-11-
30 23:59:59'
ORDER BY submission_time;

-- Obtenir la conversation entre deux parties
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' AND called_number =
'+33687654321')
   OR (calling_number = '+33687654321' AND called_number =
'+33612345678')
ORDER BY submission_time;
```

---

2. CAPACITÉS DE CHIFFREMENT ET DE CRYPTANALYSE

2.1 Vue d'Ensemble des Capacités Cryptographiques

Le SMSc OmniMessage met en œuvre des mécanismes cryptographiques pour sécuriser les communications et protéger les données sensibles. Cette section

documente toutes les capacités cryptographiques conformément aux exigences de l'ANSSI.

2.2 Chiffrement de la Couche de Transport

2.2.1 Mise en Œuvre TLS/SSL

Protocoles Supportés :

- TLS 1.2 (RFC 5246)
- TLS 1.3 (RFC 8446) - Recommandé
- SSL 2.0/3.0 : NON SUPPORTÉ (vulnérabilités connues)
- TLS 1.0/1.1 : DÉPRÉCIÉ (non recommandé)

Mise en Œuvre :

- Bibliothèque SSL/TLS Erlang/OTP (validée cryptographiquement)
- Serveur web Cowboy avec support TLS
- Points de terminaison HTTPS du Framework Phoenix

Suites de Chiffrement :

Le système utilise la sélection de suites de chiffrement sécurisées par défaut d'Erlang/OTP, qui inclut :

Préféré - TLS 1.3 :

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256

Supporté - TLS 1.2 :

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES128-GCM-SHA256

Fonctionnalités de Sécurité :

- Confidentialité Parfaite des Clés (PFS) via échange de clés ECDHE/DHE
- Groupes de Diffie-Hellman forts (minimum 2048 bits)
- Support de la Cryptographie à Courbe Elliptique
- Support de l'Indication de Nom de Serveur (SNI)

Gestion des Certificats :

- Support des certificats X.509
- Tailles de clés RSA : minimum 2048 bits, 4096 bits recommandé
- Support ECDSA
- Validation de la chaîne de certificats
- Certificats auto-signés (développement uniquement)
- Intégration CA externe

Emplacement de Configuration TLS :

```
# config/runtime.exs
config :api_ex,
  api: %&#123;
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  &#125;
```

 Référence complète de configuration dans CONFIGURATION.md

Applications :

- HTTPS pour l'API REST (port 8443)
- HTTPS pour le panneau de contrôle web (port 8086)
- Connexions à la base de données (MySQL/PostgreSQL via TLS)

2.3 Chiffrement des Données au Repos

2.3.1 Chiffrement de la Base de Données

Chiffrement MySQL/MariaDB :

- Support du chiffrement au niveau des tables
- Algorithme de chiffrement AES-256
- Chiffrement des données transparent (TDE)

```
-- Activer le chiffrement pour la table CDR
ALTER TABLE cdrs ENCRYPTION='Y';
```

Chiffrement PostgreSQL :

- Support du chiffrement des données transparent
- Chiffrement au niveau du système de fichiers
- Chiffrement au niveau des colonnes (extension pgcrypto)

2.3.2 Stockage sur Disque Mnesia

Base de Données Mnesia :

- Stockage de copies sur disque pour la persistance des messages
- Chiffrement au niveau du système de fichiers recommandé (LUKS, dm-crypt)
- Protection de la mémoire via l'isolation de la VM Erlang

2.3.3 Chiffrement du Système de Fichiers

Stockage de Données Sensibles :

- Fichiers de configuration : Chiffrement du système de fichiers recommandé
- Clés privées : Permissions de fichier (0600) + chiffrement du système de fichiers
- Fichiers journaux : Chiffrement configurable pour les journaux archivés
- Exportations CDR : Stockage chiffré pour les exportations sensibles

Stockage des Clés :

- Certificats et clés TLS stockés dans `priv/cert/`
- Magasins de clés basés sur des fichiers avec des permissions restreintes

- Procédures de rotation des clés sécurisées

2.4 Authentification et Contrôle d'Accès

2.4.1 Authentification API

Sécurité de l'API REST :

- Chiffrement de transport HTTPS/TLS obligatoire
- Authentification basée sur les en-têtes (en-tête SMSc pour identification du frontend)
- Contrôle d'accès basé sur l'IP (niveau pare-feu)
- Authentification client basée sur des certificats (optionnelle)

Enregistrement du Frontend :

- Identification unique du frontend (nom, type, IP, nom d'hôte)
- Authentification basée sur le heartbeat
- Gestion de session basée sur l'expiration (timeout de 90 secondes)
- Suivi et surveillance du frontend

2.4.2 Authentification de la Base de Données

Contrôle d'Accès à la Base de Données :

- Authentification par nom d'utilisateur/mot de passe
- Support de connexion TLS/SSL
- Restrictions de connexion basées sur l'IP
- Contrôle d'accès basé sur les rôles (RBAC)

Configuration :

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  username: "omnitouch",
  password: "omnitouch2024",  # Doit utiliser des mots de passe
forts en production
  hostname: "localhost",
  ssl: true  # Activer TLS pour les connexions à la base de
données
```

Recommandations de Contrôle d'Accès :

```
-- Créer un utilisateur en lecture seule pour l'accès des forces
de l'ordre
CREATE USER 'li_readonly'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c.cdrs TO 'li_readonly'@'%';

-- Créer un utilisateur limité sans accès au corps du message
CREATE USER 'analytics'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT (id, message_id, calling_number, called_number,
              source_smsc, dest_smsc, submission_time,
delivery_time,
              status, delivery_attempts)
ON sms_c.cdrs TO 'analytics'@'%';
```

2.5 Détails des Algorithmes Cryptographiques

2.5.1 Algorithmes de Hachage

Disponibles dans Erlang/OTP :

- SHA-256, SHA-384, SHA-512 (recommandé)
- SHA-1 (déprécié, compatibilité héritée uniquement)
- MD5 (déprécié, non utilisé pour la sécurité)
- BLAKE2 (disponible dans les versions modernes d'OTP)

Utilisation :

- Empreinte des messages (détection des doublons)

- Vérification de l'intégrité des données
- Intégrité des journaux d'audit

2.5.2 Chiffrement Symétrique

Algorithmes Disponibles :
- AES (Advanced Encryption Standard)
  - AES-128-GCM
  - AES-256-GCM
  - AES-128-CBC
  - AES-256-CBC
- ChaCha20-Poly1305

Tailles de Clé :
- 128 bits (minimum)
- 256 bits (recommandé)

Utilisation :
- Chiffrement de session TLS
- Chiffrement de base de données au repos
- Chiffrement optionnel du corps du message

2.5.3 Chiffrement Asymétrique

Algorithmes Supportés :
- RSA (2048 bits minimum, 4096 bits recommandé)
- ECDSA (Elliptic Curve Digital Signature Algorithm)
  - Courbes P-256, P-384, P-521
- Ed25519 (EdDSA)

Utilisation :
- Authentification de certificat TLS
- Signatures numériques
- Échange de clés

2.6 Sécurité du Protocole SMS

2.6.1 Encodage des Messages SMS

Support d'Encodage des Caractères :
- GSM 7 bits (encodage SMS standard)
- UCS-2 (Unicode, 16 bits)
- Données binaires 8 bits
- Latin-1

```
**TP-DCS (Data Coding Scheme) :**
- Indication de classe de message
- Drapeaux de compression
- Spécification du groupe de codage
- Identification de l'ensemble de caractères

**Pas de Chiffrement SMS Natif :**
- Le protocole SMS ne fournit pas de chiffrement de bout en bout
- Contenu du message accessible au niveau SMSc
- Permet l'interception légale si nécessaire

#### 2.6.2 Considérations de Sécurité du Protocole

**Protocole SMPP (Frontend Externe) :**
- Authentification nom d'utilisateur/mot de passe au niveau SMPP
- Support TLS disponible (SMPP sur TLS)
- Authentification de liaison

**Protocole IMS (Frontend Externe) :**
- Messagerie basée sur SIP
- Mécanismes d'authentification SIP
- Intégration avec la sécurité du réseau central IMS

**Protocole SS7/MAP (Frontend Externe) :**
- Sécurité du réseau SS7
- Authentification du protocole MAP
- Sécurité de la couche SCCP/TCAP

**Remarque :** La sécurité spécifique au protocole est mise en œuvre dans les passerelles frontend externes, pas dans le cœur SMSc.

### 2.7 Capacités d'Analyse Cryptographique et d'Évaluation de Sécurité

#### 2.7.1 Outils d'Analyse de Protocole

**Fonctionnalités de Débogage Intégrées :**
- Système de journalisation complet
- Traçage du flux de messages
- Journalisation des requêtes/réponses API
- Journalisation des requêtes de base de données
- Suivi des erreurs et des exceptions
```

```
**Intégration Externe :**
- Sortie de journalisation standard (stdout/fichiers)
- Support de capture PCAP pour l'analyse réseau
- Journalisation des requêtes de base de données pour les enquêtes
- Exportation de métriques Prometheus

#### 2.7.2 Considérations d'Évaluation des Vulnérabilités

**Limitations Connues :**
- Le protocole SMS est intrinsèquement non chiffré (par
conception, permet l'interception légale)
- Informations d'identification de base de données dans les
fichiers de configuration (doit utiliser la gestion des secrets)
- Support de certificats auto-signés (développement/test
uniquement)

**Recommandations de Renforcement de Sécurité :**
- Utiliser des suites de chiffrement TLS robustes
- Mettre en œuvre le chiffrement des connexions à la base de
données
- Utiliser une gestion des secrets externe (Vault, AWS Secrets
Manager)
- Mises à jour de sécurité régulières pour Erlang/OTP et ses
dépendances
- Restrictions de pare-feu sur les ports API
- Liste blanche d'IP pour l'accès frontend

**Tests de Sécurité :**
- Analyse régulière des vulnérabilités des dépendances
- Support pour les tests de pénétration
- Validation de la configuration TLS
- Audits de sécurité de la base de données
- Revue des contrôles d'accès

### 2.8 Infrastructure de Gestion des Clés

#### 2.8.1 Génération de Clé

**Génération de Certificat TLS :**
```bash
# Générer une clé privée (RSA 4096 bits)
openssl genrsa -out omnitouch.pem 4096

# Générer une demande de signature de certificat
```
```

```
openssl req -new -key omnitouch.pem -out omnitouch.csr

# Certificat auto-signé (développement)
openssl x509 -req -days 365 -in omnitouch.csr -signkey
omnitouch.pem -out omnitouch.crt

# Production : Obtenir un certificat d'une CA de confiance
```

Génération de Nombres Aléatoires :

- Erlang/OTP CSPRNG (Cryptographically Secure Pseudo-Random Number Generator)
- Pool d'entropie système (/dev/urandom)
- Forte randomité pour les clés de session, identifiants, jetons

2.8.2 Stockage et Protection des Clés

Stockage de Clé Privée :

- Système de fichiers avec permissions restreintes (0600)
- Stocké dans le répertoire `priv/cert/`
- Format PEM (optionnellement chiffré)
- Procédures de sauvegarde sécurisées

Rotation des Clés :

- Renouvellement de certificat TLS (annuel recommandé)
- Rotation des informations d'identification de la base de données
- Rotation des jetons API (si mis en œuvre)

2.8.3 Distribution des Clés

Distribution de Certificat :

- Installation manuelle dans `priv/cert/`
- Références dans le fichier de configuration
- Support possible du protocole ACME (Let's Encrypt)

Distribution de Clé Symétrique :

- Échange de clé hors bande pour les informations d'identification de la base de données
- Accord de clé Diffie-Hellman dans TLS
- Pas de transmission de clé en clair

2.9 Conformité et Normes

Cette section documente la conformité aux normes internationales de télécommunications, aux cadres réglementaires et aux spécifications de sécurité applicables au traitement des SMS à travers tous les protocoles supportés.

2.9.1 Conformité du Protocole SMS sur SS7/MAP

Normes 3GPP et ETSI :

- **3GPP TS 23.040 :** Réalisation technique du Service de Message Court (SMS) - Spécification du protocole SMS de base
- **3GPP TS 23.038 :** Alphabets et informations spécifiques à la langue - Encodage des caractères (GSM7, UCS-2)
- **3GPP TS 29.002 :** Spécification de la Partie d'Application Mobile (MAP) - Signalisation SS7 pour SMS
- **3GPP TS 23.003 :** Numérotation, adressage et identification - Formats MSISDN, IMSI
- **ETSI TS 100 901 :** Spécification du Service de Message Court Point à Point
- **ETSI TS 100 902 :** Spécification du Service de Message Court de Diffusion Cellulaire

Normes de Signalisation SS7 :

- **ITU-T Q.711-Q.716 :** Partie de Contrôle de Signalisation (SCCP)
- **ITU-T Q.771-Q.775 :** Partie d'Application des Capacités de Transaction (TCAP)
- **ITU-T Q.701-Q.710 :** Partie de Transfert de Message (MTP) Niveaux 1-3
- **ETSI EN 300 356 :** Système de Signalisation No.7 - Partie Utilisateur ISDN (ISUP)

Normes de Sécurité pour SS7/MAP :

- **GSMA FS.07 :** Sécurité de Signalisation SS7 et Diameter - Contrôles de sécurité de base
- **GSMA FS.11 :** Directives de Surveillance de la Sécurité SS7
- **3GPP TS 33.117 :** Catalogue des exigences générales d'assurance de sécurité
- **ETSI TS 133 210 :** Sécurité du domaine réseau - Sécurité de la couche réseau IP

Interception Légale pour SS7/MAP :

- **ETSI TS 101 671 :** Interception Légale (LI) ; Interface de remise pour l'interception légale du trafic de télécommunications
- **ETSI TS 102 232-1 :** Interception Légale (LI) ; Spécification de remise - Partie 1 : Interface de remise pour la gestion de LI
- **3GPP TS 33.107 :** Architecture et fonctions d'interception légale pour les réseaux 3G

2.9.2 Conformité du Protocole SMS sur IMS

Normes IMS 3GPP :

- **3GPP TS 23.228 :** Système Multimédia IP (IMS) - Architecture de niveau 2
- **3GPP TS 24.229 :** Protocole de Contrôle d'Appel Multimédia IP - Procédures SIP et SDP
- **3GPP TS 24.341 :** Support des SMS sur les réseaux IP - Spécification de niveau 3
- **3GPP TS 23.204 :** Support du Service de Message Court (SMS) sur l'accès IP générique 3GPP - Niveau 2
- **3GPP TS 29.228 :** Interfaces Cx et Dx du Système Multimédia (IM)

Normes de Sécurité IMS :

- **3GPP TS 33.203 :** Sécurité 3G ; Sécurité d'accès pour les services IP (IMS AKA)
- **3GPP TS 33.210 :** Sécurité 3G ; Sécurité du Domaine Réseau (NDS) ; Sécurité de la couche réseau IP (IPsec)

- **3GPP TS 33.310 :** Sécurité du Domaine Réseau (NDS) ; Cadre d'Authentification (AF)
- **ETSI TS 133 203 :** Sécurité d'accès pour les services IP

Normes du Protocole SIP :

- **RFC 3261 :** SIP : Protocole d'Initiation de Session - Spécification de base
- **RFC 3428 :** Extension SIP pour la Messagerie Instantanée - Méthode MESSAGE
- **RFC 3325 :** Extensions Privées à SIP pour l'Identité Affirmée
- **RFC 5765 :** Problèmes de Sécurité et Solutions dans les Systèmes Pair-à-Pair

Interception Légale pour IMS :

- **ETSI TS 102 232-5 :** Interception Légale (LI) ; Spécification de remise - Partie 5 : LI indépendante du service pour les services du Système Multimédia IP
- **3GPP TS 33.107 :** Exigences et architecture d'interception légale
- **3GPP TS 33.108 :** Interface de remise pour l'Interception Légale (LI)

2.9.3 Conformité du Protocole SMPP

Spécification SMPP :

- **SMPP v3.4 :** Spécification du Protocole de Message Court Peer-to-Peer - Norme de l'industrie
- **SMPP v5.0 :** Protocole SMPP étendu avec des fonctionnalités améliorées

Directives de Sécurité SMPP :

- **TLS sur SMPP :** Sécurité de la couche de transport pour les connexions SMPP (SMPP sur TLS)
- **Authentification de Liaison SMPP :** Authentification par ID système et mot de passe
- **Contrôle d'Accès basé sur l'IP :** Restrictions au niveau du réseau pour les connexions SMPP

Normes d'Interopérabilité :

- **GSM 03.40 (ETSI TS 100 901) :** Réalisation technique du SMS Point-à-Point (PP)
- **GSM 03.38 (ETSI TS 100 900) :** Alphabets et informations spécifiques à la langue
- **GSM 04.11 (ETSI TS 100 942) :** Support du SMS Point-à-Point sur l'interface radio mobile

Conformité à l'Encodage des Messages :

- **ITU-T T.50 :** Alphabet International No. 5 (IA5)
- **ISO/IEC 8859-1 :** Encodage de caractères Latin-1
- **ISO/IEC 10646 :** Ensemble de Caractères Universel (UCS-2/UTF-16)

2.9.4 Conformité aux Normes Cryptographiques

TLS et Sécurité Réseau :

- **NIST SP 800-52 :** Directives pour la Sélection, la Configuration et l'Utilisation des Implémentations TLS
- **NIST SP 800-131A :** Transition de l'Utilisation des Algorithmes Cryptographiques et des Longueurs de Clé
- **RFC 7525 :** Recommandations pour l'Utilisation Sécurisée de TLS et DTLS
- **RFC 8446 :** La Version 1.3 du Protocole de Sécurité de Transport (TLS)

Normes d'Algorithmes Cryptographiques :

- **FIPS 197 :** Standard de Chiffrement Avancé (AES)
- **FIPS 180-4 :** Standard de Hachage Sécurisé (famille SHA-2)
- **NIST SP 800-38D :** Recommandation pour les Modes de Fonctionnement des Chiffres de Bloc : Mode GCM
- **RFC 7539 :** ChaCha20 et Poly1305 pour les Protocoles IETF

Gestion des Clés :

- **NIST SP 800-57 :** Recommandation pour la Gestion des Clés

- **RFC 5280 :** Profil de Certificat et de CRL de l'Infrastructure de Clé Publique Internet X.509

2.10 Résistance à l'Analyse Cryptographique

2.10.1 Principes de Conception

Défense Contre l'Analyse Cryptographique :

- Pas d'algorithmes cryptographiques personnalisés/propriétaires
- Algorithmes standard de l'industrie, examinés par des pairs uniquement
- Mises à jour de sécurité régulières pour les bibliothèques cryptographiques
- Dépréciation des algorithmes faibles
- Utilisation de chiffrement authentifié (GCM, Poly1305)

2.10.2 Sécurité Opérationnelle

Rotation des Clés :

- Procédures de renouvellement de certificat TLS
- Rotation des clés de session (par session pour TLS)
- Politiques de rotation des informations d'identification de la base de données

Surveillance et Détection :

- Journalisation des échecs d'authentification
- Surveillance de l'expiration des certificats
- Journalisation des échecs de poignée de main TLS
- Détection d'anomalies pour les échecs de chiffrement
- Alerte d'événements de sécurité

---

3. CONTRÔLE D'INTERCEPTION ET AUTORISATION

3.1 Contrôle d'Accès pour l'Interception Légale

Autorisation Administrative :

- Accès administrateur système requis pour la configuration
- Contrôles d'accès au niveau de la base de données pour les requêtes CDR
- Accès API restreint par IP/authentification
- Journalisation des accès de tous les accès

Intégration du Cadre Légal :

- Suivi des mandats d'interception (intégration de système externe)
- Listes d'autorisation d'identificateurs cibles (vues de base de données)
- Requêtes à durée limitée (clauses WHERE SQL)
- Application automatique via des politiques d'accès

3.2 Conservation des Données et Confidentialité

Politiques de Conservation :

- Conservation des messages actifs : Configurable (par défaut 24 heures dans Mnesia)
- Conservation des CDR : Configurable (typique 6 mois à 2 ans)
- Archivage automatique de Mnesia à SQL
- Purge automatique des anciens CDR (basé sur cron)

Protections de Confidentialité :

- Option de suppression du corps du message après livraison
- Exclusion du corps du message de l'UI/exportations
- Chiffrement de la base de données au repos

- Journalisation et surveillance des accès
- Principe de collecte minimale de données

Configuration :

```
# config/runtime.exs
config :sms_c,
  # Conservation des messages Mnesia avant archivage
  message_retention_hours: 24,

  # Supprimer le corps du message après livraison pour la
confidentialité
  delete_message_body_after_delivery: false,  # Mettre true pour
le mode confidentialité

  # Contrôle d'écriture CDR
  cdr_enabled: true,

  # Paramètres d'archivage par lot
  batch_insert_batch_size: 100,
  batch_insert_flush_interval_ms: 100
```

 Voir CONFIGURATION.md pour tous les paramètres de conservation

3.3 Interfaces de Remise pour les Forces de l'Ordre

Interfaces Standard :

1. Accès API REST :

- Points de terminaison HTTPS pour la récupération de messages
- Échange de données au format JSON
- Authentification et autorisation
- Filtrage des requêtes par critères cibles

2. Accès Direct à la Base de Données :

- Informations d'identification SQL en lecture seule

- Requêtes SQL standard
- Accès à la table CDR
- Capacités de recherche indexées

3. Exportation par Lot :

- Format d'exportation CSV
- Format d'exportation JSON
- Exportations basées sur des plages horaires
- Sélection de champs configurable

Formats de Livraison :

IRI (Informations Relatives à l'Interception) :

- Champs de métadonnées CDR :
  - ID du message
  - Numéros appelants/appelés
  - Horodatages (soumission, livraison, expiration)
  - Statut
  - Tentatives de livraison
  - Informations de routage SMSC
  - Informations sur le nœud (suivi de cluster)

CC (Contenu de la Communication) :

- Corps du message (contenu textuel)
- Données PDU brutes
- Informations d'encodage
- Assemblage de message multipart

Exemple d'Exportation :

```
# Export CSV pour les forces de l'ordre
mysql -u li_readonly -p -D sms_c -e "
SELECT
  message_id,
  calling_number,
  called_number,
  message_body,
  submission_time,
  delivery_time,
  status
FROM cdrs
WHERE (calling_number = '+33612345678' OR called_number =
'+33612345678')
  AND submission_time BETWEEN '2025-11-01' AND '2025-11-30'
ORDER BY submission_time
" --batch --silent | sed 's/\t/,/g' > interception_report.csv
```

---

4. SÉCURITÉ ET INTÉGRITÉ DU SYSTÈME

4.1 Sécurité de l'Application

Sécurité Elixir/Erlang :

- Isolation et sandboxing de la VM Erlang
- Isolation et supervision des processus
- Récupération après panne et tolérance aux pannes
- Pas de vulnérabilités de débordement de tampon (runtime géré)

Gestion des Dépendances :

- Verrouillage des versions de dépendance (mix.lock)
- Analyse des vulnérabilités de sécurité
- Mises à jour régulières des dépendances
- Empreinte minimale de dépendance

4.2 Sécurité Réseau

Exposition Réseau :

- Ports exposés minimaux :
  - 8443 (API REST HTTPS)
  - 8086 (Panneau de Contrôle HTTPS)
- Configuration de pare-feu recommandée
- Liste blanche d'IP pour l'accès frontend
- Déploiement DMZ pour les services exposés à Internet

Segmentation Réseau :

- Réseau de gestion séparé
- Réseau de base de données isolé
- Séparation du réseau de passerelle frontend
- Réseau de communication de cluster (distribution Erlang)

4.3 Surveillance et Détection d'Intrusion

Capacités de Journalisation :

- Journalisation d'application structurée
- Niveaux de journalisation configurables
- Rotation et archivage des journaux
- Support d'intégration Syslog
- Journalisation centralisée (compatible avec la pile ELK)

Surveillance des Événements de Sécurité :

- Tentatives d'authentification échouées
- Modèles de message inhabituels
- Échecs de connexion à la base de données
- Échecs de poignée de main TLS
- Anomalies des ressources système

Métriques et Alerte :

- Exportation de métriques Prometheus
- Surveillance du débit de messages
- Suivi du taux d'erreur
- Utilisation des ressources système
- Règles d'alerte personnalisées

📖 Documentation complète de la surveillance dans OPERATIONS_GUIDE.md et METRICS.md

4.4 Haute Disponibilité et Récupération Après Sinistre

Support de Cluster :

- Capacité de cluster distribué Erlang
- Réplication Mnesia à travers les nœuds
- Basculement automatique
- Découverte et adhésion des nœuds

Redondance des Données :

- Copies de disque Mnesia sur tous les nœuds de cluster
- Réplication de base de données SQL (native MySQL/PostgreSQL)
- Procédures de sauvegarde CDR
- Sauvegarde de configuration

Procédures de Récupération :

- Sauvegarde et restauration de base de données
- Récupération de table Mnesia
- Restauration de configuration
- Procédures de remplacement de nœud

---

5. RÉFÉRENCES DOCUMENTAIRES

5.1 Manuels Techniques

Documentation disponible dans le dépôt du projet :

- **README.md** - Vue d'ensemble du système, architecture et fonctionnalités
- **CONFIGURATION.md** - Référence complète de configuration
- **API_REFERENCE.md** - Documentation de l'API REST
- **OPERATIONS_GUIDE.md** - Procédures opérationnelles et surveillance
- **CDR_SCHEMA.md** - Schéma de base de données des enregistrements de détails d'appel
- **sms_routing_guide.md** - Configuration de routage SMS
- **number_translation_guide.md** - Normalisation des numéros
- **METRICS.md** - Métriques Prometheus et surveillance
- **TROUBLESHOOTING.md** - Problèmes courants et solutions

5.2 Certifications de Sécurité

- **Rapports de Test de Pénétration :** [À fournir sur demande]
- **Rapports d'Audit de Sécurité :** [À fournir sur demande]
- **Évaluations de Vulnérabilité :** [À fournir sur demande]
- **Validation Cryptographique Erlang/OTP :** Bibliothèque cryptographique standard de l'industrie

5.3 Documentation de Conformité

- **Demande d'Autorisation ANSSI R226 :** Ce document
- **Conformité à l'Interception Légale :** Comme requis par les réglementations françaises en télécommunications
- **Conformité à la Protection des Données :** Considérations GDPR pour les données de message

---

6. INFORMATIONS DE CONTACT

Informations sur le Fournisseur/Opérateur :

- Nom de l'Entreprise : Omnitouch Network Services Pty Ltd
- Adresse : PO BOX 296, QUINNS ROCKS WA 6030, AUSTRALIE
- Personne de Contact : Équipe de Conformité
- Email : compliance@omnitouch.com.au

Contact Sécurité Technique :

- Nom : Équipe de Conformité
- Email : compliance@omnitouch.com.au

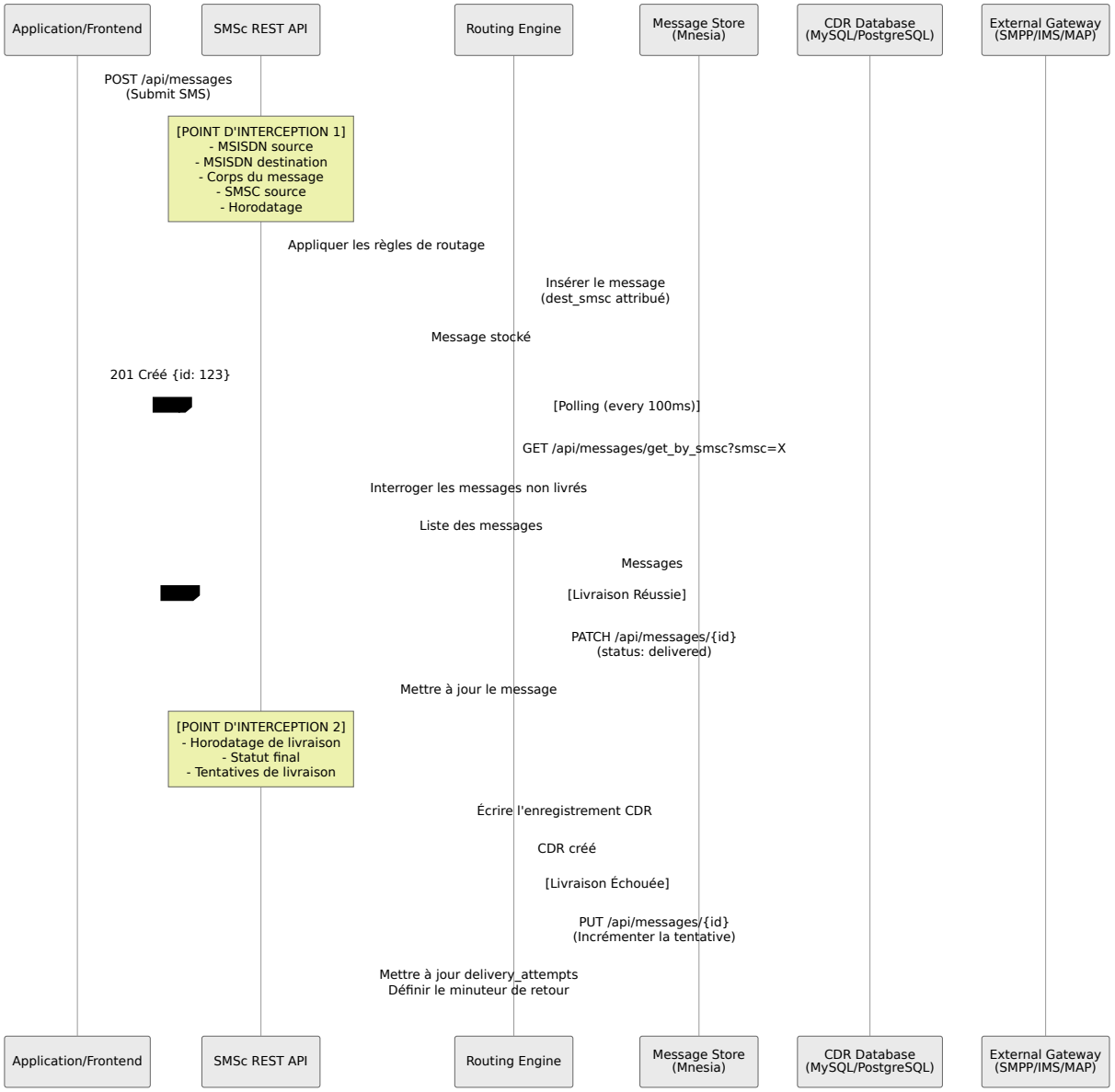
Contact Juridique/Conformité :

- Nom : Équipe de Conformité
- Email : compliance@omnitouch.com.au

ANNEXES

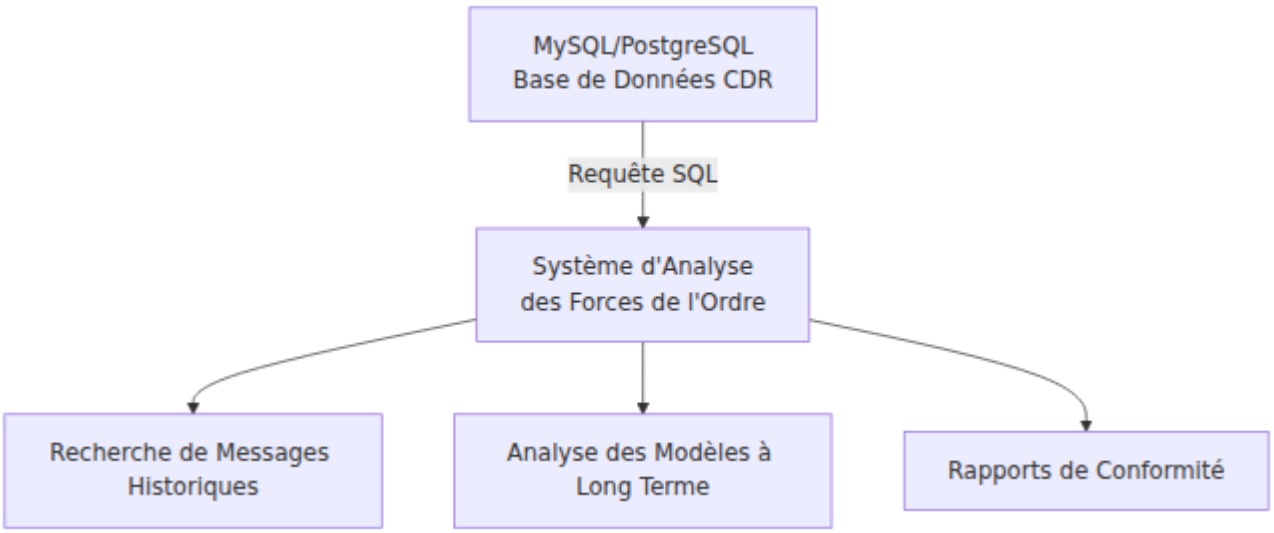
Annexe A : Flux de Message SMS avec Points d'Interception

A.1 Flux SMS Sortant (Mobile Terminé)



Légende : `[POINT D'INTERCEPTION]` = Points où les données d'interception légale sont capturées et stockées

A.2 Flux SMS Entrant (Mobile Originaire)



Annexe B : Schéma CDR pour l'Interception Légale

Le SMSc OmniMessage stocke les Enregistrements de Détails d'Appel dans une base de données SQL (MySQL ou PostgreSQL) pour une conservation à long terme et un accès à l'interception légale.

B.1 Champs Clés CDR pour l'Interception Légale

Nom du Champ	Type	Description	Pertinence de l'Interception
`id`	BIGINT	Clé primaire auto-incrémentée	Identifiant d'enregistrement
`message_id`	BIGINT	Identifiant unique du message	**Corrélation de session**
`calling_number`	VARCHAR(255)	MSISDN source	**Identifiant principal pour le suivi cible**
`called_number`	VARCHAR(255)	MSISDN destination	**Suivi de la destination cible**
`source_smsc`	VARCHAR(255)	Identifiant de la passerelle source	Suivi de protocole/passerelle
`dest_smsc`	VARCHAR(255)	Identifiant de la passerelle destination	Informations de routage
`origin_node`	VARCHAR(255)	Nœud de cluster Erlang (origine)	**Suivi du système**
`destination_node`	VARCHAR(255)	Nœud de cluster Erlang (livraison)	Distribution de cluster
`submission_time`	DATETIME	Horodatage de soumission du message	**Chronologie des événements**

Nom du Champ	Type	Description	Pertinence de l'Interception
`delivery_time`	DATETIME	Horodatage de livraison du message	**Confirmation de livraison**
`expiry_time`	DATETIME	Horodatage d'expiration du message	Suivi des échecs de livraison
`status`	VARCHAR(50)	Statut du message (livré/expiré/ échoué/rejeté)	**Disposition finale**
`delivery_attempts`	INT	Nombre de tentatives de livraison	Analyse de réessai
`message_parts`	INT	Nombre de segments SMS	Suivi des messages concaténés
`deadletter`	BOOLEAN	Indicateur de file d'attente de lettres mortes	Indicateur d'échec permanent
`message_body`	TEXT	Contenu du message SMS	**Contenu du message (optionnel)**
`inserted_at`	DATETIME	Horodatage de création du CDR	Piste d'audit

Nom du Champ	Type	Description	Pertinence de l'Interception
`updated_at`	DATETIME	Horodatage de mise à jour du CDR	Suivi des modifications

📖 Documentation complète du schéma avec des exemples SQL dans CDR_SCHEMA.md

B.2 Exemples de requêtes CDR pour interception légale

Interroger tous les messages pour le numéro cible :

```
SELECT * FROM cdrs
WHERE calling_number = '+33612345678'
   OR called_number = '+33612345678'
ORDER BY submission_time DESC;
```

Interroger les messages dans une fenêtre temporelle :

```
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' OR called_number =
'+33612345678')
  AND submission_time BETWEEN '2025-11-01 00:00:00' AND '2025-11-
30 23:59:59'
ORDER BY submission_time;
```

Exporter au format CSV pour les forces de l'ordre :

```
.mode csv
.output /tmp/interception_report.csv
SELECT message_id, calling_number, called_number, message_body,
       submission_time, delivery_time, status
FROM cdrs
WHERE calling_number = '+33612345678'
ORDER BY submission_time DESC;
```

B.3 Méthodes d'accès à la base de données CDR

1. Accès SQL direct :

- Identifiants de base de données en lecture seule
- Requêtes SQL standard
- Connectivité JDBC/ODBC
- Outils clients de base de données (MySQL Workbench, pgAdmin)

2. Accès API REST :

- Amélioration future : API REST pour les requêtes CDR
- Réponses au format JSON
- Authentification et autorisation
- Filtrage des paramètres de requête

3. Exportation par lot :

- Exportation CSV via la ligne de commande mysql/psql
- Scripts d'exportation automatisés
- Exportations planifiées via cron

B.4 Conservation et confidentialité des CDR

Configuration de conservation :

```
# config/runtime.exs
config :sms_c,
  # Supprimer le corps du message après livraison (mode de
confidentialité)
  delete_message_body_after_delivery: false,  # true pour la
confidentialité

  # Activer/désactiver l'écriture des CDR
  cdr_enabled: true,

  # Paramètres d'archivage Mnesia vers CDR
  message_retention_hours: 24
```

Options de confidentialité :

- Le corps du message peut être défini sur NULL après livraison
- Chiffrement de la table de base de données (MySQL ENCRYPTION='Y')
- Restrictions d'accès au niveau des colonnes
- Exportations masquées pour l'analyse

Annexe C : Référence API REST pour l'interception

C.1 Points de terminaison de récupération de messages

Obtenir tous les messages :

```
GET /api/messages
Authorization: Bearer <token>
```

Obtenir des messages par SMSC :

```
GET /api/messages/get_by_smsc?smsc=gateway-name
```

Obtenir un message spécifique :

```
GET /api/messages/{id}
```

Format de réponse :

```
{
  "status": "success",
  "data": [
    {
      "id": 12345,
      "message_id": 12345,
      "source_msisdn": "+33612345678",
      "destination_msisdn": "+33687654321",
      "message_body": "Contenu du message ici",
      "source_smsc": "ims.gateway",
      "dest_smsc": "smpp.provider",
      "status": "delivered",
      "delivery_attempts": 1,
      "inserted_at": "2025-11-29T10:30:00Z",
      "deliver_time": "2025-11-29T10:30:05Z",
      "expires": "2025-11-30T10:30:00Z"
    }
  ]
}
```


📖 Documentation API complète dans API_REFERENCE.md

C.2 Filtrage et recherche

Paramètres de requête :

- `source_smsc` - Filtrer par passerelle source
- `dest_smsc` - Filtrer par passerelle de destination
- Filtrage par plage horaire (via les requêtes de base de données CDR)
- Filtrage par statut (via les requêtes de base de données CDR)

Améliorations futures :

- API REST pour les requêtes CDR
- Filtrage avancé par numéro de téléphone
- Filtrage par plage de dates
- Recherche en texte intégral sur le corps du message

Annexe D : Exemples de configuration

D.1 Configuration du certificat TLS

Générer un certificat TLS :

```
# Générer une clé privée RSA de 4096 bits
openssl genrsa -out priv/cert/omnitouch.pem 4096

# Générer une demande de signature de certificat
openssl req -new -key priv/cert/omnitouch.pem -out
priv/cert/omnitouch.csr \
  -subj "/C=FR/ST=IDF/L=Paris/O=Omnitouch/CN=smsc.example.com"

# Certificat auto-signé (développement/test)
openssl x509 -req -days 365 -in priv/cert/omnitouch.csr \
  -signkey priv/cert/omnitouch.pem -out priv/cert/omnitouch.crt

# Production : Soumettre CSR à une CA de confiance pour signature
```

Configurer dans l'application :

```
# config/runtime.exs
config :api_ex,
  api: %{
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  }
```

D.2 Configuration du chiffrement de la base de données

Chiffrement de table MySQL :

```sql
-- Activer le chiffrement pour la table CDR
ALTER TABLE cdrs ENCRYPTION='Y';

-- Vérifier l'état du chiffrement
SELECT TABLE_NAME, CREATE_OPTIONS
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'sms_c' AND TABLE_NAME = 'cdrs';
```

Chiffrement de connexion PostgreSQL :

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  username: "omnitouch",
  password: "secure_password",
  hostname: "localhost",
  database: "sms_c",
  ssl: true,
  ssl_opts: [
    verify: :verify_peer,
    cacertfile: "/path/to/ca.crt",
    certfile: "/path/to/client.crt",
    keyfile: "/path/to/client.key"
  ]
```

D.3 Configuration de la confidentialité

Activer la suppression du corps du message :

```
# config/runtime.exs
config :sms_c,
  # Supprimer le corps du message après livraison réussie
  delete_message_body_after_delivery: true,

  # Masquer le corps du message dans l'UI
  hide_message_body_in_ui: true,

  # Masquer le corps du message dans les exportations
  hide_message_body_in_export: true
```

Créer une vue de base de données préservant la confidentialité :

```
-- Créer une vue sans corps de message pour l'analyse générale
CREATE VIEW cdrs_metadata AS
SELECT
  id, message_id, calling_number, called_number,
  source_smsc, dest_smsc, origin_node, destination_node,
  submission_time, delivery_time, expiry_time,
  status, delivery_attempts, message_parts, deadletter,
  inserted_at, updated_at
FROM cdrs;

-- Accorder l'accès aux utilisateurs d'analyse
GRANT SELECT ON cdrs_metadata TO 'analytics'@'%';
```

Annexe E : Glossaire

Organismes de réglementation et normes

- **ANSSI :** Agence nationale de la sécurité des systèmes d'information - Agence nationale française de cybersécurité
- **ETSI :** European Telecommunications Standards Institute
- **3GPP :** 3rd Generation Partnership Project - Normes de télécommunications mobiles
- **IETF :** Internet Engineering Task Force - Organisme de normalisation Internet

Termes de télécommunications

- **SMSc :** Centre de services SMS - Système central pour le routage et la livraison des messages SMS
- **SMPP :** Protocole de pair à pair de messages courts - Norme industrielle pour l'échange de SMS
- **ESME :** Entité de message court externe - Application cliente SMPP
- **IMS :** Système multimédia IP - Architecture réseau tout-IP pour les services multimédias
- **SIP :** Protocole de session d'initiation - Protocole de signalisation pour la messagerie IMS

- **P-CSCF :** Fonction de contrôle de session d'appel proxy - Point d'entrée du réseau IMS
- **S-CSCF :** Fonction de contrôle de session d'appel de service - Contrôle de session IMS
- **HSS :** Serveur d'abonnés à domicile - Base de données des abonnés IMS
- **SS7/MAP :** Système de signalisation 7 / Partie d'application mobile - Protocoles de signalisation mobile hérités
- **MSC :** Centre de commutation mobile - Élément de réseau à circuit commuté
- **VLR :** Registre de localisation des visiteurs - Base de données de localisation des abonnés
- **GT :** Titre global - Schéma d'adressage SS7
- **SCCP :** Partie de contrôle de connexion de signalisation - Couche réseau SS7
- **TCAP :** Partie des capacités de transaction - Couche d'application SS7
- **MSISDN :** Numéro de répertoire international d'abonnés de station mobile - Numéro de téléphone
- **IMSI :** Identité internationale d'abonné mobile - Identifiant unique de l'abonné
- **E.164 :** Plan de numérotation international pour les numéros de téléphone
- **ENUM :** E.164 Number Mapping - Mappage basé sur DNS du numéro de téléphone à l'URI
- **PDU :** Unité de données de protocole - Message SMS codé en binaire
- **TP-DCS :** Schéma de codage de données de protocole de transfert - Spécification de codage SMS
- **TP-DU :** Unité de données de protocole de transfert - Format PDU spécifique au SMS
- **UDH :** En-tête de données utilisateur - En-tête pour les messages SMS concaténés/spéciaux
- **TON/NPI :** Type de numéro / Indicateur de plan de numérotation - Classification du format de numéro
- **GSM7 :** Alphabet par défaut de 7 bits GSM - Codage de caractères SMS standard
- **UCS-2 :** Ensemble de caractères universel à 2 octets - Codage Unicode pour SMS

Composants du système

- **Mnesia :** Système de base de données distribué Erlang - Stockage en mémoire/disque
- **CDR :** Enregistrement des détails d'appel - Enregistrement de facturation et d'analyse pour les messages
- **REST API :** Transfert d'état représentatif - Architecture API basée sur HTTP
- **Phoenix :** Framework web Elixir
- **Cowboy :** Serveur HTTP Erlang
- **Ecto :** Wrapper de base de données Elixir et langage de requête
- **PubSub :** Modèle de messagerie Publish-Subscribe

Interception légale

- **LI :** Interception légale - Surveillance légale des télécommunications
- **LIMF :** Fonction de médiation d'interception légale - Système qui fait interface entre le réseau de télécommunications et les forces de l'ordre
- **LEMF :** Installation de surveillance des forces de l'ordre - Système des forces de l'ordre recevant les données interceptées
- **IRI :** Informations liées à l'interception - Métadonnées d'appel/message pour les forces de l'ordre
- **CC :** Contenu de la communication - Contenu réel du message
- **Interface HI1 :** Interface de transfert administratif LI ETSI - Provisionnement de mandat et activation de cible
- **Interface HI2 :** Interface de transfert LI ETSI pour la livraison d'IRI - Transfert de métadonnées aux forces de l'ordre
- **Interface HI3 :** Interface de transfert LI ETSI pour la livraison de CC - Transfert de contenu aux forces de l'ordre
- **R226 :** Articles R226-3 et R226-7 du Code pénal français régissant les équipements d'interception
- **ETSI :** European Telecommunications Standards Institute - Définit les normes LI
- **ETSI TS 102 232 :** Spécification technique pour les interfaces de transfert d'interception légale

Traitement des messages

- **MT :** Mobile Terminated - Message sortant vers un abonné mobile
- **MO :** Mobile Originated - Message entrant d'un abonné mobile
- **DLR :** Accusé de réception de livraison - Confirmation de la livraison du message
- **Lettre morte :** Message qui a échoué à la livraison après toutes les tentatives de réessai
- **Retard exponentiel :** Délai de réessai croissant (2min, 4min, 8min, etc.)

Sécurité et chiffrement

- **TLS :** Transport Layer Security - Protocole de chiffrement
- **PFS :** Confidentialité parfaite des clés - Propriété cryptographique pour la sécurité des clés de session
- **AES :** Standard de chiffrement avancé
- **RSA :** Rivest-Shamir-Adleman - Cryptographie à clé publique
- **ECDSA :** Algorithme de signature numérique à courbe elliptique
- **SHA :** Algorithme de hachage sécurisé
- **X.509 :** Norme de certificat
- **CA :** Autorité de certification
- **CSPRNG :** Générateur de nombres pseudo-aléatoires cryptographiquement sécurisé

Termes de base de données

- **MySQL :** Base de données relationnelle open-source
- **PostgreSQL :** Base de données objet-relationnelle open-source
- **TDE :** Chiffrement de données transparent
- **RBAC :** Contrôle d'accès basé sur les rôles

Version du document : 1.0
Date : 2025-11-29
Préparé pour : Demande d'autorisation ANSSI R226
Classification du document : Conformité réglementaire - Confidentiel

Messagerie en masse / Campagnes

← Retour à l'Index de Documentation

Aperçu

La fonctionnalité des campagnes envoie un seul message en texte brut à de nombreux destinataires tout en protégeant le SMSC de la surcharge. Vous téléchargez une **liste de destinataires** nommée une fois, puis exécutez une ou plusieurs **campagnes** contre celle-ci. Chaque campagne envoie des messages dans la file de messages normale à un rythme configurable et rapporte les progrès en temps réel ainsi que les statistiques de livraison.

Deux objets réutilisables :

Objet	Ce que c'est
Liste de campagnes	Un ensemble nommé et réutilisable de MSISDN de destinataires, chacun avec jusqu'à quatre variables optionnelles. Téléchargez une fois (tableau JSON ou CSV), réutilisez à travers les campagnes.
Campagne	Une exécution d'envoi : une liste (ou *tous les abonnés actifs*) + un modèle de message, envoyés dans la file.

Tout est exposé via l'API HTTP. Tout l'état est stocké dans Mnesia aux côtés du reste du SMSC.

Ciblage : listes et défaut d'abonnés actifs

Les destinataires d'une campagne proviennent de l'une des deux sources :

- **Une liste de destinataires** (`list_id`) - les MSISDN que vous avez téléchargés.
- **Aucune liste du tout** - si vous créez une campagne **sans** `list_id`, elle cible **tous les abonnés actuellement actifs (enregistrés)** - chaque MSISDN avec une inscription non expirée dans le magasin de localisation au moment où la campagne commence. C'est le moyen le plus simple d'atteindre "tout le monde sur le réseau en ce moment".

Variables par destinataire et modélisation

Chaque destinataire de la liste peut porter jusqu'à quatre variables - **`var1`**, **`var2`**, **`var3`**, **`var4`** - pour des choses comme le prénom, le nom de famille ou le nom du plan. Le message de la campagne est un **modèle** : `{{ var1 }}`…`{{ var4 }}` (et `{{ msisdn }}`) sont insérés pour chaque destinataire lorsque le message est envoyé. Les espaces à l'intérieur des accolades sont optionnels, et tout espace réservé sans valeur est rendu comme une chaîne vide.

> La modélisation est délibérément minimale - substitution de variables uniquement (pas de conditionnelles ou de filtres), donc il n'y a pas de dépendance supplémentaire et pas de surprises dans ce que reçoivent les abonnés.

Téléchargement de variables via CSV - la colonne MSISDN est celle nommée `msisdn`/`destination_msisdn`/`number`/`phone` (ou la première colonne si aucune n'est nommée) ; chaque autre colonne devient `var1`…`var4` dans l'ordre des colonnes :

```
msisdn,first,last,plan
12025550101,Alice,Smith,Gold
12025550102,Bob,Jones,Silver
```

Téléchargement de variables via JSON - les objets de destinataires peuvent définir les vars directement :

```
{ "name": "VIPs", "recipients": [
  { "msisdn": "12025550101", "var1": "Alice", "var2": "Gold" }
] }
```

Exemple de modèle :

```
Salut {{ var1 }}, votre plan {{ var2 }} se renouvelle bientôt.
Répondez STOP pour vous désinscrire.
```

Pour le destinataire `12025550101` ci-dessus (`var1=Alice`, `var2=Gold`), cela envoie : `Salut Alice, votre plan Gold se renouvelle bientôt. Répondez STOP pour vous désinscrire.`

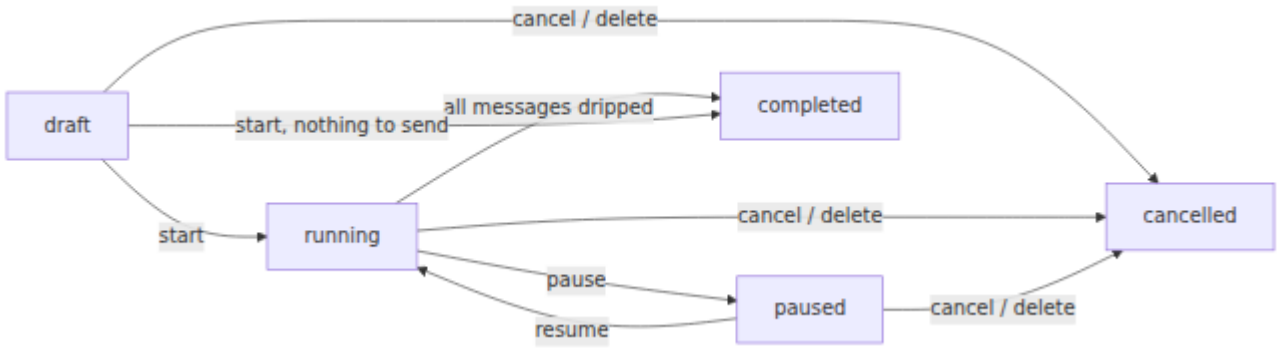
Filtre d'audience

Lorsqu'une campagne utilise une liste, le drapeau `audience` décide qui reçoit réellement un message :

Audience	Comportement
`active_only` (par défaut)	Seulement les membres de la liste qui sont **enregistrés** (une localisation non expirée) au moment où la campagne commence. Tous les autres sont enregistrés comme `skipped_inactive` et ne reçoivent jamais de message.
`all`	Chaque membre de la liste, indépendamment de l'enregistrement.

La vérification d'enregistrement est un **instantané pris au moment du démarrage** - les abonnés qui s'inscrivent plus tard ne sont pas pris en compte. Lorsqu'une campagne **n'a pas de liste**, l'ensemble des destinataires est déjà la base des abonnés actifs, donc `audience` est implicitement `active_only` et toute valeur fournie est ignorée.

Cycle de vie



1. **draft** - créé, pas encore de cibles.
2. **start** - la campagne prend un instantané de ses cibles (de sa liste ou de la base d'abonnés actifs), puis passe à **running**. S'il n'y a rien à envoyer, il passe directement à **completed**.
3. **running** - le travailleur de goutte soumet des messages dans la file.
4. **pause / resume** - arrêter temporairement / continuer à goutter.
5. **completed** - chaque cible a été gouttée dans la file (la livraison continue ensuite de manière asynchrone, et les statistiques de livraison continuent de se mettre à jour).

6. **cancelled** - arrêté définitivement.

Suppression d'une campagne

Supprimer une campagne **l'arrête mais conserve l'enregistrement** - elle est annulée plutôt que purgée, donc son historique et ses statistiques finales restent visibles. Les campagnes arrêtées et complètes sont automatiquement supprimées une fois qu'elles dépassent la durée de conservation TTL (`campaign_retention_hours`, par défaut 7 jours).

Taux de goutte (protection contre la surcharge)

Chaque campagne a un `drip_tps` (messages soumis dans la file par seconde, par défaut depuis la configuration). Un travailleur en arrière-plan se réveille à un intervalle fixe et soumet au maximum `drip_tps × interval` messages par campagne par tick - donc une campagne d'un million de destinataires ne submerge jamais la file. Le taux est **par campagne** ; plusieurs campagnes en cours gouttent chacune à leur propre rythme.

Progrès et statistiques de livraison

Chaque objet de campagne inclut un bloc `stats` en direct (interroger `GET /api/campaigns/:id`) :

Champ	Signification
`total`	Cibles prises en instantané au début
`pending`	Pas encore goutté dans la file
`queued`	Soumis, en attente d'un rapport de livraison
`delivered`	Confirmé livré
`failed`	Échec de la livraison / lettre morte
`skipped_inactive`	Ignoré car non enregistré au début
`dispatched`	`queued + delivered + failed`
`progress_percent`	Pourcentage de cibles n'étant plus `pending`
`delivery_percent`	Pourcentage de messages envoyés livrés

Les comptes de livraison et d'échec se mettent à jour automatiquement à mesure que les rapports de livraison affluent à travers le chemin de livraison normal du SMSC. Les détails par destinataire (et le filtrage par état, par exemple, tous ceux qui ont échoué) sont disponibles via `GET /api/campaign_targets/:id?state=failed`.

Configuration

```
config :sms_c, :campaigns,
  # Démarrer le travailleur de goutte (définir sur false pour
mettre en pause TOUTE livraison de campagne)
  enabled: true,
  # Intervalle entre les ticks de goutte en millisecondes
  drip_tick_ms: 1000,
  # Messages par seconde par campagne lorsqu'une campagne ne
définit pas son propre drip_tps
  default_drip_tps: 50,
  # Fenêtre de validité par défaut des messages lorsqu'une
campagne ne définit pas validity_hours
  message_validity_hours: 72,
  # Combien de temps une campagne arrêtée/complet persiste avant
d'être purgée (heures)
  campaign_retention_hours: 168
```

Validité et livraison programmée

Une campagne peut définir, au moment de sa création :

- **`validity_hours`** - la période de validité du message. Chaque message soumis est estampillé avec un `expires` afin que les messages non livrés expirent au lieu de rester dans la file. Par défaut, cela correspond à `message_validity_hours` de la configuration.
- **`deliver_after`** - un temps ISO8601 optionnel (doit être dans le futur) pour un envoi programmé.

Comment la livraison programmée interagit avec le goutte. La campagne commence toujours à goutter *immédiatement* à `drip_tps`, mais l'horloge de chaque message est décalée vers l'avant par un décalage fixe `delta = deliver_after − started_at`. Ainsi, un message goutté au temps réel *t* est estampillé `send_time`/`deliver_after = t + delta` et `expires = t + delta + validity_hours`. Étant donné que les messages sont insérés à `drip_tps` en temps réel, leurs valeurs `deliver_after` par message sortent **décalées à `drip_tps` à partir de `deliver_after`** - la livraison est limitée par

le taux plutôt qu'une ruée tonitruante au moment programmé. Sans `deliver_after`, `delta = 0` et les messages sont gouttés et livrés immédiatement.

Flux de travail typique

```
# 1. Téléchargez une liste de destinataires nommée avec des variables
destinataire (CSV ou JSON)
curl -k -X POST https://smsc:8443/api/campaign_lists \
  -H 'Content-Type: application/json' \
  -d '{"name":"Clients VIP",

"csv":"msisdn,first,plan\n12025550101,Alice,Gold\n12025550102,Bob,Sil
# => {"id": 1, "recipient_count": 2, ...}

# 2. Créez une campagne contre cette liste, en modélisant les variabl
curl -k -X POST https://smsc:8443/api/campaigns \
  -H 'Content-Type: application/json' \
  -d '{"name":"Avis de renouvellement","message":"Salut {{ var1 }}, v
plan {{ var2 }} se renouvelle bientôt.",
       "source_msisdn":"12345","source_smsc":"IMS_SMSC",
       "list_id":1,"audience":"active_only","drip_tps":50}'
# => {"id": 7, "status": "draft", ...}

# 3. Démarrez-la
curl -k -X POST https://smsc:8443/api/campaigns/control \
  -H 'Content-Type: application/json' \
  -d '{"campaign_id":7,"action":"start"}'

# 4. Interrogez les progrès / statistiques de livraison
curl -k https://smsc:8443/api/campaigns/7
# => {... "stats":
{"total":..,"delivered":..,"failed":..,"progress_percent":..}}
```

Pour envoyer un message **à tous les abonnés actifs**, omettez `list_id` à l'étape 2 (et sautez complètement l'étape 1).

Voir aussi

- Référence API des Campagnes
- Guide de Routage SMS - comment les messages gouttés sont routés/livrés
- Référence de Configuration

Référence du Schéma CDR (Call Detail Record)

← Retour à l'Index de Documentation

Référence complète pour la table de base de données CDR utilisée pour le stockage à long terme des messages, la facturation et l'analyse.

Table des Matières

- Aperçu
- Schéma de Table
- Descriptions des Champs
- Exemples SQL
- Index
- Types de Données par Base de Données
- Considérations de Confidentialité
- Conservation et Archivage
- Intégration de Facturation

Aperçu

La table `cdrs` stocke les enregistrements de détails d'appel pour tous les messages SMS que le système traite. Le système écrit un CDR au moment où un message atteint un état terminal : `livré`, `expiré`, `échoué` ou `rejeté`. Le système enregistre un `balance_rejected` hors ��quilibre comme `rejeté`. Un balayage de temps de conservation agit comme un filet de sécurité pour tout ce qui vieillit dans Mnesia sans écriture en direct. Voir Quand les CDRs sont Écrits pour le chemin d'écriture complet et les garanties d'idempotence.

Les CDRs fournissent un stockage à long terme séparé de la base de données opérationnelle Mnesia, permettant :

- Facturation et facturation
- Analyse et reporting
- Conformité et audit
- Historique des messages au-delà de la période de conservation de Mnesia

Schéma de Table

MySQL / MariaDB

```
CREATE TABLE cdrs (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,

  -- Identification du message
  message_id BIGINT NOT NULL,

  -- Numéros de téléphone
  calling_number VARCHAR(255) NOT NULL,
  called_number VARCHAR(255) NOT NULL,

  -- Routage SMSC
  source_smsc VARCHAR(255),
  dest_smsc VARCHAR(255),

  -- Informations sur le nœud (pour les déploiements en cluster)
  origin_node VARCHAR(255),
  destination_node VARCHAR(255),

  -- Horodatages
  submission_time DATETIME NOT NULL,
  delivery_time DATETIME,
  expiry_time DATETIME,

  -- Statut et métadonnées
  status VARCHAR(50) NOT NULL,
  delivery_attempts INT DEFAULT 0,
  message_parts INT,
  deadletter BOOLEAN DEFAULT FALSE,

  -- Corps de message optionnel (contrôles de confidentialité)
  message_body TEXT,

  -- Horodatages d'audit
  inserted_at DATETIME NOT NULL,
  updated_at DATETIME NOT NULL,

  -- Indexes
  UNIQUE KEY cdrs_origin_node_message_id_index (origin_node,
```

```
message_id),
  INDEX idx_cdrs_message_id (message_id),
  INDEX idx_cdrs_calling_number (calling_number),
  INDEX idx_cdrs_called_number (called_number),
  INDEX idx_cdrs_status (status),
  INDEX idx_cdrs_submission_time (submission_time),
  INDEX idx_cdrs_dest_smsc (dest_smsc)
);
```

> La clé unique `(origin_node, message_id)` donne à chaque CDR une identité unique à l'échelle mondiale à travers le cluster. Elle rend également les écritures CDR idempotentes. L'écriture quasi temps réel et le filet de sécurité de conservation peuvent tous deux tenter une insertion. La seconde insertion est un no-op via `ON CONFLICT DO NOTHING`.

PostgreSQL

```sql
CREATE TABLE cdrs (
  id BIGSERIAL PRIMARY KEY,

  -- Identification du message
  message_id BIGINT NOT NULL,

  -- Numéros de téléphone
  calling_number VARCHAR(255) NOT NULL,
  called_number VARCHAR(255) NOT NULL,

  -- Routage SMSC
  source_smsc VARCHAR(255),
  dest_smsc VARCHAR(255),

  -- Informations sur le nœud (pour les déploiements en cluster)
  origin_node VARCHAR(255),
  destination_node VARCHAR(255),

  -- Horodatages
  submission_time TIMESTAMP NOT NULL,
  delivery_time TIMESTAMP,
  expiry_time TIMESTAMP,

  -- Statut et métadonnées
  status VARCHAR(50) NOT NULL,
  delivery_attempts INTEGER DEFAULT 0,
  message_parts INTEGER,
  deadletter BOOLEAN DEFAULT FALSE,

  -- Corps de message optionnel (contrôles de confidentialité)
  message_body TEXT,

  -- Horodatages d'audit
  inserted_at TIMESTAMP NOT NULL,
  updated_at TIMESTAMP NOT NULL
);

-- Indexes
CREATE UNIQUE INDEX cdrs_origin_node_message_id_index ON
cdrs(origin_node, message_id);
CREATE INDEX idx_cdrs_message_id ON cdrs(message_id);
```

```
CREATE INDEX idx_cdrs_calling_number ON cdrs(calling_number);
CREATE INDEX idx_cdrs_called_number ON cdrs(called_number);
CREATE INDEX idx_cdrs_status ON cdrs(status);
CREATE INDEX idx_cdrs_submission_time ON cdrs(submission_time);
CREATE INDEX idx_cdrs_dest_smsc ON cdrs(dest_smsc);
```

Descriptions des Champs

Clé Primaire

Champ	Type	Nullable	Description
`id`	BIGINT	NON	Clé primaire auto-incrémentée pour l'enregistrement CDR

Identification du Message

Champ	Type	Nullable	Description
`message_id`	BIGINT	NON	Identifiant du message provenant de la file d'attente de messages SMS-C. Généré par un compteur **par nœud** (persisté dans Mnesia, donc il survit aux redémarrages), donc uniquement unique **au sein d'un seul nœud**. Pour une clé unique à l'échelle mondiale dans un cluster multi-nœuds, utilisez `(origin_node, message_id)` ensemble - cette paire est imposée par un index unique et est la clé de jointure/dé-duplication recommandée pour les systèmes de facturation.

Numéros de Téléphone

Champ	Type	Nullable	Description
`calling_number`	VARCHAR(255)	NON	MSISDN source (numéro de mobile) de l'expéditeur du message. Typiquement au format E.164 (par exemple, +15551234567).
`called_number`	VARCHAR(255)	NON	MSISDN de destination (numéro de mobile) du destinataire du message. Typiquement au format E.164 (par exemple, +15551234567).

Routage SMSC

Champ	Type	Nullable	Description
`source_smsc`	VARCHAR(255)	OUI	Nom ou identifiant du SMSC source qui a soumis le message. NULL si soumis via API ou autre interface non-SMSC.
`dest_smsc`	VARCHAR(255)	OUI	Nom ou identifiant du SMSC de destination qui a livré (ou tenté de livrer) le message. NULL si le message n'a jamais été routé.

Informations sur le Nœud

Pour les déploiements en cluster, suit quels nœuds ont géré le message :

Champ	Type	Nullable	Description
`origin_node`	VARCHAR(255)	OUI	Nom du nœud Erlang où le message a été initialemen reçu (par exemple, "sms@node1.example.cor Utile pour le dépannage et l'analyse de distribution de charge.
`destination_node`	VARCHAR(255)	OUI	Nom du nœud Erlang d'où message a été livré (si différent de l'origine). NUL pour les déploiements à nœud unique ou si le message n'a jamais été liv

Horodatages

Tous les horodatages sont stockés en UTC :

Champ	Type	Nullable	Description
`submission_time`	DATETIME	NON	Quand le message a été d'abord soumis au SMS-C. Utilisé comme heure de début pour les calculs de facturation.
`delivery_time`	DATETIME	OUI	Quand le message a été livré avec succès. NULL si le message a expiré, échoué ou a été rejeté.
`expiry_time`	DATETIME	OUI	Quand le message a expiré (devenu non livrable). NULL si le message a été livré ou est toujours en attente.

Calcul de la Durée de Livraison :

```
TIMESTAMPDIFF(SECOND, submission_time, delivery_time) AS
delivery_duration_seconds
```

Statut et Métadonnées

Champ	Type	Nullable	Description
`status`	VARCHAR(50)	NON	Statut final du message. Valeurs valides : `delivered`, `expired`, `failed`, `rejected`
`delivery_attempts`	INT	NON	Nombre de tentatives de livraison effectuées avant le statut final. Par défaut : 0. Plage : 0-255 typiquement.
`message_parts`	INT	OUI	Nombre de segments SMS pour les messages concaténés. 1 pour les messages à partie unique, 2+ pour multi-part. NULL si inconnu.
`deadletter`	BOOLEAN	NON	Indique si le système a déplacé le message vers la file d'attente des lettres mortes. TRUE indique que le système n'a pas pu livrer le message et a épuisé toutes les tentatives. Par défaut : FALSE

Valeurs de Statut :

Statut	Description	Facturable	Temps de Livraison
`delivered`	Livré avec succès au destinataire	Oui	Défini
`expired`	Dépassé la période de validité sans livraison	Dépend de la politique de facturation	NULL
`failed`	Échec permanent de la livraison (numéro invalide, etc.)	Dépend de la politique de facturation	NULL
`rejected`	Rejeté par les règles de routage ou de validation	Non	NULL

Corps de Message

Champ	Type	Nullable	Description
`message_body`	TEXT	OUI	Le contenu réel du message SMS. Peut être NULL si `delete_message_body_after_delivery` est activé pour des raisons de confidentialité. La longueur maximale varie selon la base de données (typiquement 65,535 caractères pour le type TEXT).

Modes de Confidentialité :

- **Conservation complète :** Corps de message stocké dans le CDR pour conformité/archivage
- **Mode de confidentialité :** Corps de message défini sur NULL lorsque `delete_message_body_after_delivery: true`

- **Mode de conformité :** Corps stocké chiffré ou haché (nécessite une mise en œuvre personnalisée)

Horodatages d'Audit

Champ	Type	Nullable	Description
`inserted_at`	DATETIME	NON	Quand l'enregistrement CDR a été d'abord inséré dans la base de données. Typiquement le même que ou peu après delivery_time/expiry_time.
`updated_at`	DATETIME	NON	Quand l'enregistrement CDR a été mis à jour pour la dernière fois. Même que inserted_at s'il n'a jamais été mis à jour.

Exemples SQL

Requêtes de Base

Trouver tous les CDRs pour un numéro de téléphone spécifique :

```sql
SELECT * FROM cdrs
WHERE calling_number = '+15551234567'
   OR called_number = '+15551234567'
ORDER BY submission_time DESC
LIMIT 100;
```

Compter les messages par statut :

```sql
SELECT status, COUNT(*) as count
FROM cdrs
GROUP BY status;
```

Temps de livraison moyen pour les messages livrés :

```sql
SELECT AVG(TIMESTAMPDIFF(SECOND, submission_time, delivery_time))
AS avg_delivery_seconds
FROM cdrs
WHERE status = 'delivered'
  AND delivery_time IS NOT NULL;
```

Requêtes de Facturation

Volume de messages quotidien par SMSC de destination :

```sql
SELECT
  DATE(submission_time) AS date,
  dest_smsc,
  COUNT(*) AS message_count,
  SUM(CASE WHEN status = 'delivered' THEN 1 ELSE 0 END) AS
delivered_count,
  SUM(message_parts) AS total_segments
FROM cdrs
WHERE submission_time >= DATE_SUB(NOW(), INTERVAL 30 DAY)
GROUP BY DATE(submission_time), dest_smsc
ORDER BY date DESC, message_count DESC;
```

Messages facturables pour un client (par préfixe de numéro appelant) :

```sql
SELECT
  DATE(submission_time) AS date,
  COUNT(*) AS message_count,
  SUM(message_parts) AS total_segments,
  SUM(message_parts) * 0.01 AS total_cost
FROM cdrs
WHERE calling_number LIKE '+1555%'
  AND status = 'delivered'
  AND submission_time >= '2025-10-01'
  AND submission_time < '2025-11-01'
GROUP BY DATE(submission_time);
```

Analyse de performance de routage :

```sql
SELECT
  dest_smsc,
  COUNT(*) AS total_messages,
  SUM(CASE WHEN status = 'delivered' THEN 1 ELSE 0 END) AS
delivered,
  ROUND(100.0 * SUM(CASE WHEN status = 'delivered' THEN 1 ELSE 0
END) / COUNT(*), 2) AS delivery_rate_pct,
  AVG(delivery_attempts) AS avg_attempts,
  AVG(TIMESTAMPDIFF(SECOND, submission_time, delivery_time)) AS
avg_delivery_seconds
FROM cdrs
WHERE submission_time >= DATE_SUB(NOW(), INTERVAL 7 DAY)
  AND dest_smsc IS NOT NULL
GROUP BY dest_smsc
ORDER BY delivery_rate_pct DESC;
```

Requêtes d'Analyse

Messages par heure de la journée (modèle de trafic) :

```sql
SELECT
  HOUR(submission_time) AS hour,
  COUNT(*) AS message_count
FROM cdrs
WHERE submission_time >= DATE_SUB(NOW(), INTERVAL 7 DAY)
GROUP BY HOUR(submission_time)
ORDER BY hour;
```

Analyse des messages multi-part :

```sql
SELECT
  message_parts,
  COUNT(*) AS message_count,
  AVG(TIMESTAMPDIFF(SECOND, submission_time, delivery_time)) AS
avg_delivery_seconds
FROM cdrs
WHERE message_parts IS NOT NULL
  AND status = 'delivered'
GROUP BY message_parts
ORDER BY message_parts;
```

Analyse des messages échoués :

```sql
SELECT
  called_number,
  COUNT(*) AS failure_count,
  AVG(delivery_attempts) AS avg_attempts,
  MAX(submission_time) AS last_failure
FROM cdrs
WHERE status IN ('failed', 'expired')
  AND submission_time >= DATE_SUB(NOW(), INTERVAL 7 DAY)
GROUP BY called_number
HAVING failure_count >= 5
ORDER BY failure_count DESC;
```

Requêtes de Conformité et d'Audit

Trouver tous les messages entre deux parties dans une plage de temps :

```
SELECT
  submission_time,
  calling_number,
  called_number,
  status,
  message_body,
  delivery_time
FROM cdrs
WHERE (
    (calling_number = '+15551234567' AND called_number =
'+15559876543')
    OR
    (calling_number = '+15559876543' AND called_number =
'+15551234567')
  )
  AND submission_time >= '2025-10-01'
  AND submission_time < '2025-11-01'
ORDER BY submission_time;
```

Application de la politique de conservation (supprimer les anciens CDRs) :

```
-- Trouver les enregistrements plus anciens que la période de
conservation (exemple : 2 ans)
SELECT COUNT(*) FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);

-- Supprimer les anciens enregistrements (à utiliser avec prudence
!)
DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR)
LIMIT 10000;  -- Suppression par lots pour éviter le verrouillage
```

Analyse de Cluster

Distribution des messages à travers les nœuds :

```
SELECT
  origin_node,
  COUNT(*) AS message_count,
  SUM(CASE WHEN status = 'delivered' THEN 1 ELSE 0 END) AS
delivered_count
FROM cdrs
WHERE submission_time >= DATE_SUB(NOW(), INTERVAL 1 DAY)
GROUP BY origin_node;
```

Index

Les index suivants sont créés pour optimiser les requêtes courantes :

Nom de l'Index	Colonnes	But
`PRIMARY`	`id`	Clé primaire, assure un enregistrement unique
`cdrs_origin_node_message_id_index`	`origin_node, message_id`	**Unique.** Identité CDR unique à l'échelle mondiale à travers le cluster ; rend les écritures idempotentes
`idx_cdrs_message_id`	`message_id`	Recherche CDR par ID de message d'origine
`idx_cdrs_calling_number`	`calling_number`	Trouver les messages d'un expéditeur spécifique
`idx_cdrs_called_number`	`called_number`	Trouver les messages à un destinataire spécifique
`idx_cdrs_status`	`status`	Filtrer par statut de livraison

Nom de l'Index	Colonnes	But
`idx_cdrs_submission_time`	`submission_time`	Requêtes basées sur le temps, périodes de facturation
`idx_cdrs_dest_smsc`	`dest_smsc`	Analyse de performance de routage

Recommandations d'Index Supplémentaires

Pour des déploiements à fort volume, envisagez ces index supplémentaires :

Index composite pour les requêtes de facturation :

```
CREATE INDEX idx_cdrs_billing ON cdrs(calling_number,
submission_time, status);
```

Index composite pour l'analyse de routage :

```
CREATE INDEX idx_cdrs_route_perf ON cdrs(dest_smsc,
submission_time, status);
```

Index composite pour les recherches de conformité :

```
CREATE INDEX idx_cdrs_party_time ON cdrs(calling_number,
called_number, submission_time);
```

Index de texte intégral pour les recherches dans le corps de message (MySQL) :

```sql
ALTER TABLE cdrs ADD FULLTEXT INDEX idx_cdrs_message_body_ft
(message_body);

-- Utilisation :
SELECT * FROM cdrs
WHERE MATCH(message_body) AGAINST('keyword' IN NATURAL LANGUAGE
MODE);
```

Types de Données par Base de Données

Mappages de types de champs à travers les bases de données prises en charge :

Champ	MySQL/MariaDB	PostgreSQL	Remarques
`id`	BIGINT AUTO_INCREMENT	BIGSERIAL	Entier 64 bits, auto-incrémenté
`message_id`	BIGINT	BIGINT	Entier 64 bits
Champs de chaîne	VARCHAR(255)	VARCHAR(255)	Chaîne de longueur variable, max 255 caractères
`message_body`	TEXT	TEXT	Grand texte, jusqu'à 65,535 octets (MySQL), illimité (PostgreSQL)
Horodatages	DATETIME	TIMESTAMP	Horodatages UTC recommandés
Entiers	INT	INTEGER	Entier signé 32 bits
Booléens	BOOLEAN (TINYINT(1))	BOOLEAN	MySQL stocke comme 0/1

Considérations de Confidentialité

La table CDR peut contenir des informations personnelles sensibles (numéros de téléphone, contenu des messages). Considérez ces mesures de confidentialité :

1. Confidentialité du Corps de Message

Options de configuration dans `config/runtime.exs` :

```
config :sms_c,
  # Supprimer le corps du message après livraison réussie
  delete_message_body_after_delivery: true,

  # Masquer le corps du message dans l'UI
  hide_message_body_in_ui: true,

  # Masquer le corps du message dans les exports
  hide_message_body_in_export: true
```

2. Masquage des Numéros de Téléphone

Pour des analyses qui ne nécessitent pas de numéros complets :

```
-- Masquer les 4 derniers chiffres des numéros de téléphone
SELECT
  CONCAT(SUBSTRING(calling_number, 1, LENGTH(calling_number) - 4),
'XXXX') AS masked_calling,
  CONCAT(SUBSTRING(called_number, 1, LENGTH(called_number) - 4),
'XXXX') AS masked_called,
  COUNT(*) AS message_count
FROM cdrs
GROUP BY masked_calling, masked_called;
```

3. Chiffrement de Base de Données

Activer le chiffrement au repos pour le serveur de base de données :

MySQL :

```
-- Activer le chiffrement de table
ALTER TABLE cdrs ENCRYPTION='Y';
```

PostgreSQL : Utiliser le chiffrement de données transparent (TDE) de PostgreSQL ou le chiffrement au niveau du système de fichiers.

4. Contrôles d'Accès

Restreindre l'accès à la table CDR :

```
-- Créer un utilisateur de facturation en lecture seule
CREATE USER 'billing_ro'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c.cdrs TO 'billing_ro'@'%';

-- Créer un utilisateur d'analyse limité (pas d'accès au corps du
message)
CREATE USER 'analytics'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT (id, message_id, calling_number, called_number,
source_smsc,
              dest_smsc, submission_time, delivery_time, status,
              delivery_attempts, message_parts)
ON sms_c.cdrs TO 'analytics'@'%';
```

Conservation et Archivage

Politiques de Conservation

Définir des périodes de conservation en fonction des exigences réglementaires et commerciales :

Industrie	Conservation Typique	Base Réglementaire
Télécom (US)	18-24 mois	FCC, lois d'État
Télécom (UE)	6 mois - 2 ans	RGPD, ePrivacy
Financier	5-7 ans	SOX, SEC
Santé	6 ans	HIPAA

Stratégie d'Archivage

1. Partitionner par Date (MySQL 8.0+, PostgreSQL 11+)

```
-- Partitionnement MySQL par mois
ALTER TABLE cdrs PARTITION BY RANGE (TO_DAYS(submission_time)) (
  PARTITION p202510 VALUES LESS THAN (TO_DAYS('2025-11-01')),
  PARTITION p202511 VALUES LESS THAN (TO_DAYS('2025-12-01')),
  PARTITION p202512 VALUES LESS THAN (TO_DAYS('2026-01-01')),
  PARTITION p_future VALUES LESS THAN MAXVALUE
);

-- Supprimer l'ancienne partition (archivage rapide)
ALTER TABLE cdrs DROP PARTITION p202510;
```

2. Archiver vers un Stockage Froid

```
-- Exporter les anciens CDRs vers une table d'archive
CREATE TABLE cdrs_archive LIKE cdrs;

INSERT INTO cdrs_archive
SELECT * FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);

-- Vérifier et supprimer de la table principale
DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);
```

3. Script de Nettoyage Automatisé

```
#!/bin/bash
# cleanup_old_cdrs.sh - Exécuter via cron

MYSQL_USER="cleanup_user"
MYSQL_PASS="secure_password"
MYSQL_DB="sms_c"
RETENTION_DAYS=730  # 2 ans

# Archiver les anciens enregistrements
mysql -u"$MYSQL_USER" -p"$MYSQL_PASS" "$MYSQL_DB" <<EOF
INSERT INTO cdrs_archive
SELECT * FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL $RETENTION_DAYS
DAY)
LIMIT 100000;

DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL $RETENTION_DAYS
DAY)
LIMIT 100000;
EOF
```

Entrée Cron :

```
# Exécuter quotidiennement à 2h du matin
0 2 * * * /usr/local/bin/cleanup_old_cdrs.sh >>
/var/log/sms_c/cleanup.log 2>&1
```

Intégration de Facturation

Schéma de Tarification

Créer une table de tarifs séparée pour la facturation :

```sql
CREATE TABLE billing_rates (
  id INT AUTO_INCREMENT PRIMARY KEY,
  destination_prefix VARCHAR(20) NOT NULL,
  description VARCHAR(255),
  rate_per_message DECIMAL(10, 6) NOT NULL,
  rate_per_segment DECIMAL(10, 6) NOT NULL,
  currency VARCHAR(3) DEFAULT 'USD',
  effective_date DATE NOT NULL,
  expiry_date DATE,
  INDEX idx_prefix (destination_prefix),
  INDEX idx_dates (effective_date, expiry_date)
);

-- Tarifs d'exemple
INSERT INTO billing_rates (destination_prefix, description,
rate_per_message, rate_per_segment, effective_date) VALUES
('+1', 'États-Unis/Canada', 0.0050, 0.0050, '2025-01-01'),
('+44', 'Royaume-Uni', 0.0080, 0.0080, '2025-01-01'),
('+61', 'Australie', 0.0100, 0.0100, '2025-01-01'),
('+', 'Par défaut international', 0.0150, 0.0150, '2025-01-01');
```

Requête de Facturation

Joindre les CDRs avec les tarifs pour la facturation :

```
SELECT
  DATE(c.submission_time) AS date,
  c.dest_smsc AS route,
  LEFT(c.called_number,
    CASE
      WHEN c.called_number LIKE '+1%' THEN 2
      WHEN c.called_number LIKE '+%' THEN
LENGTH(SUBSTRING_INDEX(c.called_number, '', 4))
      ELSE 0
    END
  ) AS destination_prefix,
  COUNT(*) AS message_count,
  SUM(c.message_parts) AS segment_count,
  COALESCE(r.rate_per_segment, 0.015) AS rate,
  SUM(c.message_parts) * COALESCE(r.rate_per_segment, 0.015) AS
total_cost
FROM cdrs c
LEFT JOIN billing_rates r ON c.called_number LIKE
CONCAT(r.destination_prefix, '%')
  AND c.submission_time >= r.effective_date
  AND (r.expiry_date IS NULL OR c.submission_time < r.expiry_date)
WHERE c.status = 'delivered'
  AND c.submission_time >= '2025-10-01'
  AND c.submission_time < '2025-11-01'
GROUP BY date, route, destination_prefix
ORDER BY date DESC, total_cost DESC;
```

Export pour Systèmes de Facturation

Exportation CSV intégrée (recommandée) :

Le SMS-C peut rendre les CDRs dans un fichier CSV conforme à la RFC 4180 directement, sans passer par le client SQL. Dans une version, pilotez-le depuis la console distante :

```
bin/omnimessage rpc
'SmsC.Messaging.CDR.export_csv_to_file("/tmp/cdrs-202607.csv",
since: ~U[2026-07-01 00:00:00Z], until: ~U[2026-08-01 00:00:00Z],
status: :delivered)'
```

Options : `:since`, `:until` (tous deux `DateTime`), `:status` (un atome/chaîne ou une liste, par exemple `[:delivered, :failed]`), et `:limit`. La valeur retournée est `{:ok, {path, row_count}}`. La colonne `message_body` respecte le paramètre `hide_message_body_in_export` - définissez-le sur `true` pour émettre `[REDACTED]` au lieu du contenu du message. Une fonction d'exportation compagnon retourne le même CSV sous forme de chaîne si vous préférez le diffuser ailleurs.

Exportation directe du client SQL (alternative) :

```
mysql -u billing_ro -p -D sms_c -e "
SELECT
  id,
  message_id,
  calling_number,
  called_number,
  dest_smsc,
  submission_time,
  delivery_time,
  status,
  message_parts
FROM cdrs
WHERE submission_time >= '2025-10-01'
  AND submission_time < '2025-11-01'
  AND status = 'delivered'
" --batch --silent | sed 's/\t/,/g' > billing_export_202510.csv
```

Voir Aussi

- Guide de Configuration - Configurer les paramètres d'exportation CDR
- Guide des Opérations - Procédures de maintenance des CDR
- Référence API - Interroger les CDRs via l'API REST

Référence de Configuration SMS-C

← Retour à l'Index de Documentation

Référence complète pour toutes les options de configuration SMS-C avec des exemples pour des scénarios de déploiement courants.

Table des Matières

- Fichiers de Configuration
- Configuration de la Base de Données
- Configuration de l'API
- Configuration de l'Interface Web
- Configuration de la Fédération
- Configuration de la File de Messages
- Configuration de Facturation
- Configuration Diameter Sh (HSS)
- Configuration ENUM
- Configuration de Traduction de Numéros
- Interception Légale (X1/X2/X3)
- Configuration de Routage
- Configuration de Réglage de Performance
- Configuration de Journalisation
- Scénarios de Configuration Courants

Fichiers de Configuration

Le SMS-C utilise trois fichiers de configuration principaux :

config/config.exs

Configuration statique chargée au moment de la compilation. Contient :

- Valeurs par défaut pour l'application
- Configuration du journal
- Paramètres de développement/test
- Paramètres de réglage de performance

config/runtime.exs

Configuration à l'exécution chargée au démarrage. Contient :

- Paramètres de connexion à la base de données
- Configuration de cluster
- Intégration de services externes (OCS, ENUM)
- Routes initiales et règles de traduction
- Paramètres spécifiques à l'environnement

config/prod.exs (optionnel)

Remplacements spécifiques à la production.

Meilleure Pratique : Utilisez des variables d'environnement dans `runtime.exs` pour des valeurs sensibles comme les mots de passe et les clés API.

Configuration de Stockage SQL CDR

Le SMS-C utilise **Mnesia** pour les données opérationnelles (file de messages, règles de routage, traductions de numéros) et prend en charge des **bases de données SQL externes** pour le stockage à long terme des CDR (Call Detail Record), la facturation et l'analyse.

Bases de Données SQL Supportées

Le système prend en charge les bases de données SQL suivantes pour l'exportation des CDR :

Base de Données	Version	Adaptateur	Port par Défaut	Meilleur P
MySQL	8.0+	`Ecto.Adapters.MyXQL`	3306	Usage géné fiabilité prouvée
MariaDB	10.5+	`Ecto.Adapters.MyXQL`	3306	Compatible MySQL, ope source
PostgreSQL	13+	`Ecto.Adapters.Postgres`	5432	Fonctionnal avancées, support JSC

Remarque : Mnesia est utilisé automatiquement pour les données opérationnelles (file de messages, routage, traductions) et ne nécessite aucune configuration. La base de données SQL est **uniquement** utilisée pour l'exportation des CDR et le stockage à long terme.

Quand les CDRs Sont Écrits

Les CDRs sont écrits dans la base de données SQL à deux moments :

1. **Quasi temps réel, à l'état terminal** - au moment où un message atteint un résultat final (`delivered`, `expired`, `failed`, `rejected`, ou `balance_rejected`), un CDR est écrit de manière asynchrone. C'est le chemin principal et cela maintient la facturation à jour sans attendre la rétention des messages.
2. **File d'attente de rétention** - lorsque le travailleur de rétention supprime un message expiré de Mnesia, il exporte d'abord un CDR. Il enregistre

`expired` pour tout ce qui n'a jamais atteint un état terminal. Le travailleur ne supprime un message qu'une fois que son exportation de CDR réussit. Rien n'est perdu si la base de données CDR est brièvement indisponible : le travailleur réessaie l'exportation lors du prochain passage.

Les deux chemins sont idempotents. Chaque CDR utilise la paire unique `(origin_node, message_id)` comme clé. Les écritures utilisent `ON CONFLICT DO NOTHING`. Le backstop ne duplique donc jamais un message que l'écriture quasi temps réel a déjà couvert.

Un compteur par nœud génère `message_id` (persisté dans Mnesia, donc il survit aux redémarrages). `message_id` est donc unique **au sein d'un seul nœud**. Lorsque plusieurs nœuds SMS-C écrivent dans une base de données CDR partagée, `(origin_node, message_id)` ensemble est la clé globalement unique. Voir la Référence du Schéma CDR. Définissez `cdr_enabled: false` pour désactiver complètement l'écriture des CDR.

Exportation CSV / Fichier Texte

Les CDRs peuvent être exportés de la base de données SQL vers un fichier CSV conforme à la RFC 4180. Dans une version (où `mix` n'est pas disponible), exécutez-le depuis la console distante :

```
bin/omnimessage rpc
'SmsC.Messaging.CDR.export_csv_to_file("/tmp/cdrs-202607.csv",
since: ~U[2026-07-01 00:00:00Z], until: ~U[2026-08-01 00:00:00Z])'
```

Options supportées : `:since`, `:until` (tous deux `DateTime`), `:status` (un atome/chaîne ou liste), et `:limit`. La colonne `message_body` respecte le paramètre `hide_message_body_in_export` (émis comme `[REDACTED]` lorsqu'il est activé).

Configuration MySQL / MariaDB

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  adapter: Ecto.Adapters.MyXQL,
  username: System.get_env("DB_USERNAME") || "sms_user",
  password: System.get_env("DB_PASSWORD") || "secure_password",
  hostname: System.get_env("DB_HOSTNAME") || "localhost",
  port: String.to_integer(System.get_env("DB_PORT") || "3306"),
  database: System.get_env("DB_NAME") || "sms_c_prod",
  pool_size: String.to_integer(System.get_env("DB_POOL_SIZE") ||
"20")
```

Configuration PostgreSQL

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  adapter: Ecto.Adapters.Postgres,
  username: System.get_env("DB_USERNAME") || "sms_user",
  password: System.get_env("DB_PASSWORD") || "secure_password",
  hostname: System.get_env("DB_HOSTNAME") || "localhost",
  port: String.to_integer(System.get_env("DB_PORT") || "5432"),
  database: System.get_env("DB_NAME") || "sms_c_prod",
  pool_size: String.to_integer(System.get_env("DB_POOL_SIZE") ||
"20")
```

Choisir une Base de Données SQL

MySQL/MariaDB - Recommandé pour la plupart des déploiements :

- Excellente performance pour les écritures de CDR
- Fiabilité prouvée dans les environnements de télécommunications
- Large support d'outils pour les systèmes de facturation
- Configuration de réplication facile

PostgreSQL - À considérer si vous avez besoin de :

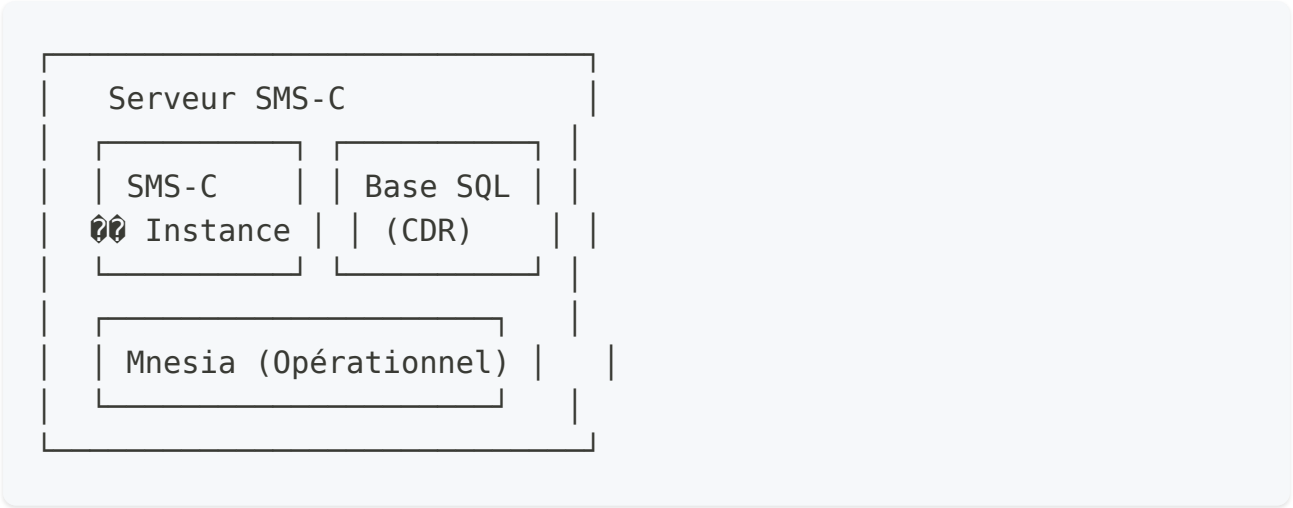
- Fonctionnalités avancées JSON/JSONB pour l'analyse

- Requêtes complexes sur les données CDR
- Infrastructure PostgreSQL existante
- PostGIS pour l'analyse géographique

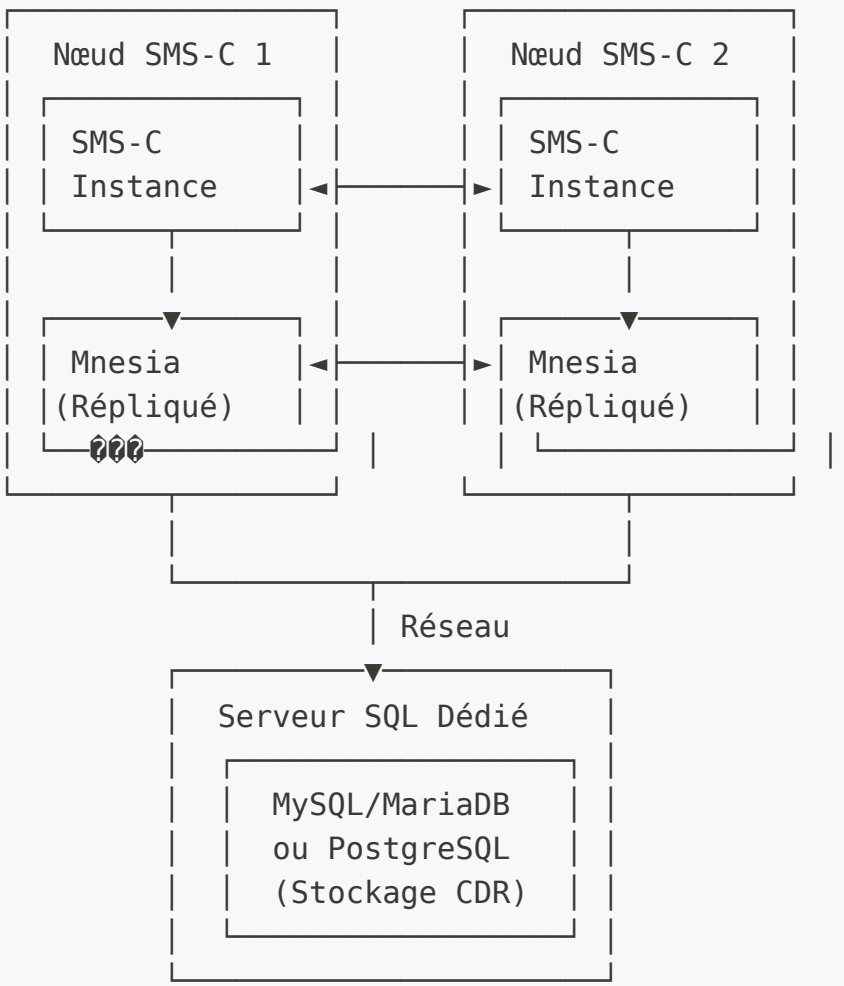
Topologies de Déploiement

Important : La base de données SQL CDR peut fonctionner sur un **serveur séparé** de vos instances SMS-C. C'est l'approche recommandée pour les déploiements en production.

Déploiement sur un Serveur Unique (Développement/Test) :



Déploiement Distribué (Production - Recommandé) :



Avantages d'un Serveur SQL Séparé :

- **Isolation de Performance** : Les écritures de CDR n'impactent pas le traitement des messages
- **Scalabilité** : Évoluer indépendamment la base de données et le traitement des messages
- **Fiabilité** : La maintenance de la base de données n'affecte pas le temps de disponibilité du SMS-C
- **Gestion des Données** : Stockage CDR centralisé pour plusieurs instances SMS-C
- **Flexibilité de Sauvegarde** : Horaires de sauvegarde et politiques de rétention indépendants

Directives de Taille de Pool

Charge de Travail	Taille de Pool	Description
Développement	5-10	Concurrence minimale
Faible Volume (< 100 msg/sec)	10-15	Petits déploiements
Volume Moyen (100-1000 msg/sec)	20-30	Production typique
Volume Élevé (> 1000 msg/sec)	40-100	Scénarios à fort débit

Calcul : `pool_size = (opérations DB concurrentes attendues) * 1.5`

Exemples de Connexion à la Base de Données

Utilisation de Variables d'Environnement (Recommandé pour la Production) :

```
# Définir les variables d'environnement
export DB_USERNAME=sms_prod_user
export DB_PASSWORD=strong_password_here
export DB_HOSTNAME=db-primary.internal.example.com
export DB_PORT=3306
export DB_NAME=sms_c_production
export DB_POOL_SIZE=30
```

Configuration Directe (Développement Seulement) :

```
config :sms_c, SmsC.Repo,
  username: "dev_user",
  password: "dev_password",
  hostname: "localhost",
  database: "sms_c_dev",
  pool_size: 5
```

Surveillance du Pool de Connexion

Surveillez l'utilisation du pool via les métriques Prometheus :

- `ecto_pools_queue_time` - Temps d'attente pour une connexion
- `ecto_pools_query_time` - Temps d'exécution de la requête
- `ecto_pools_connected_count` - Connexions actives

Alertez si le temps d'attente dépasse 100ms de manière constante - indique un besoin d'un pool plus grand.

Configuration de l'API

L'API REST fournit des capacités de soumission et de gestion des messages.

Configuration de Base de l'API

```
# config/runtime.exs
config :api_ex,
  api: %{
    port: 8443,
    listen_ip: "0.0.0.0",
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  }
```

Les paramètres de l'API se trouvent sous la carte `:api` de l'application `:api_ex`. La table des routes (`routes:`) est définie au moment de la

compilation dans `config/config.exs` et n'est pas modifiée ici.

Paramètre	Type	Requis	Par Défaut	Des
`port`	Entier	Non	`8443`	Port pou RES
`listen_ip`	Chaîne	Non	`0.0.0.0`	Adr liais l'AP `0.0` pou les inte
`enable_tls`	Booléen	Non	`true`	Ser via
`tls_cert_path`	Chaîne	Si `enable_tls`	`priv/cert/omnitouch.crt`	Che le c TLS
`tls_key_path`	Chaîne	Si `enable_tls`	`priv/cert/omnitouch.pem`	Che la c TLS

Configuration TLS/SSL

Configuration de Production avec TLS (Recommandé) :

```
config :api_ex,
  api: %{
    port: 8443,
    listen_ip: "0.0.0.0",
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  }
```

Configuration de Développement sans TLS :

```
config :api_ex,
  api: %{
    port: 8080,
    listen_ip: "127.0.0.1",
    enable_tls: false
  }
```

Configuration du Certificat API

Générez un certificat auto-signé pour les tests :

```
# Créer le répertoire du certificat
mkdir -p priv/cert

# Générer la clé privée
openssl genrsa -out priv/cert/server.key 2048

# Générer la demande de signature de certificat
openssl req -new -key priv/cert/server.key -out
priv/cert/server.csr \
  -subj "/C=US/ST=State/L=City/O=Organization/CN=sms-
api.example.com"

# Générer un certificat auto-signé (valide 365 jours)
openssl x509 -req -days 365 -in priv/cert/server.csr \
  -signkey priv/cert/server.key -out priv/cert/server.crt

# Définir les permissions
chmod 600 priv/cert/server.key
chmod 644 priv/cert/server.crt
```

Pour la production, utilisez des certificats d'une CA de confiance (Let's Encrypt, CA commerciale, etc.).

Contrôle d'Accès API

Liste Blanche d'IP (Pare-feu d'Application) :

```
# Utilisation d'iptables (Linux)
iptables -A INPUT -p tcp --dport 8443 -s 10.0.0.0/8 -j ACCEPT
iptables -A INPUT -p tcp --dport 8443 -j DROP

# Utilisation de firewalld (Red Hat/CentOS)
firewall-cmd --permanent --add-rich-rule='rule family="ipv4"
source address="10.0.0.0/8" port protocol="tcp" port="8443"
accept'
firewall-cmd --reload
```

Authentification par Clé API (Niveau Application) :

Configurez via un plug personnalisé dans le routeur - voir le Guide des Opérations pour les détails de mise en œuvre.

Configuration de l'Interface Web

L'interface web fournit la gestion des routes, la navigation des messages et la surveillance.

Configuration de Base de l'Interface Web

```
# config/runtime.exs
config :control_panel, ControlPanelWeb.Endpoint,
  https: [
    port: 443,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ]
```

L'interface web est un point de terminaison Phoenix. TLS est configuré via la liste de mots-clés `https:` du point de terminaison (pas `tls_cert_path`/`tls_key_path`), et HTTPS est imposé via `force_ssl` dans `config/config.exs`.

Paramètre	Type	Requis	Par Défaut	Description
`port`	Entier	Non	`443`	Port HTTPS pour le panneau de contrôle.
`keyfile`	Chaîne	Oui	`priv/cert/omnitouch.pem`	Chemin vers la clé privée TLS.
`certfile`	Chaîne	Oui	`priv/cert/omnitouch.crt`	Chemin vers le certificat TLS.

Configuration de Production de l'Interface Web

```
config :control_panel, ControlPanelWeb.Endpoint,
  url: [host: "sms-admin.example.com", path: "/"],
  https: [
    port: 443,
    keyfile: "priv/cert/web.pem",
    certfile: "priv/cert/web.crt"
  ]
```

Configuration de Proxy Inversé (Recommandé)

Utilisez Nginx ou Apache comme proxy inversé pour une sécurité et des fonctionnalités supplémentaires :

Exemple de Configuration Nginx :

```
upstream sms_web {
  server 127.0.0.1:4000;
  keepalive 32;
}

server {
  listen 80;
  server_name sms-admin.example.com;
  return 301 https://$server_name$request_uri;
}

server {
  listen 443 ssl http2;
  server_name sms-admin.example.com;

  ssl_certificate /etc/letsencrypt/live/sms-
admin.example.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/sms-
admin.example.com/privkey.pem;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers HIGH:!aNULL:!MD5;

  # Authentification de base pour une sécurité supplémentaire
  auth_basic "Administrateur SMS-C";
  auth_basic_user_file /etc/nginx/.htpasswd;

  location / {
    proxy_pass http://sms_web;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
  }

  # Support WebSocket pour LiveView
  location /live {
    proxy_pass http://sms_web;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
```

```
    proxy_read_timeout 86400;
  }
}
```

Configuration de la Fédération

OmniMessage utilise la fédération basée sur HTTP pour les déploiements multi-contrôleurs. Les contrôleurs se découvrent via des enregistrements DNS SRV ou des listes de pairs statiques, échangent des registres de santé et de frontend via HTTPS, et transmettent des messages à des contrôleurs distants si nécessaire.

Voir le **Guide de Fédération Géographique** pour l'architecture complète, des exemples de déploiement et des dépannages.

Démarrage Rapide

```
# config/runtime.exs
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.smsc.example.com"
```

Référence Complète des Paramètres

Paramètre	Type	Par Défaut	Description
`enabled`	Booléen	`false`	Interrupteur principal pour la fédération.
`dns_srv_domain`	Chaîne	`""`	Domaine DNS SRV pour la découverte des pairs.
`dns_poll_interval_ms`	Entier	`30000`	Intervalle de re-résolution DNS.
`health_check_interval_ms`	Entier	`10000`	Intervalle de vérification de santé des pairs.
`registry_sync_interval_ms`	Entier	`15000`	Intervalle d'échange de registre de frontend.
`forward_retry_interval_ms`	Entier	`5000`	Intervalle de réessai des messages en file d'attente.
`forward_max_retries`	Entier	`50`	Max d'essais de transfert avant de marquer comme échoué.
`http_timeout_ms`	Entier	`5000`	Délai d'attente pour tous les appels HTTP des pairs.

Paramètre	Type	Par Défaut	Description
`api_port`	Entier	`8443`	Port API pour la construction des URL des pairs.
`static_peers`	Liste	`[]`	Liste de pairs statiques (alternative aux DNS SRV).

Exigences Réseau

La fédération nécessite uniquement le port **8443** (HTTPS) entre les sites, par rapport aux ports 4369 + 9100-9200 pour le clustering Erlang.

```
# Autoriser le trafic de fédération des sites pairs
iptables -A INPUT -p tcp -s 10.0.1.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.2.0/24 --dport 8443 -j ACCEPT
```

Configuration de la File de Messages

Contrôle le comportement de rétention et d'expiration des messages.

Expiration des Messages

```
# config/runtime.exs
config :sms_c,
  dead_letter_time_minutes: 1440  # 24 heures
```

Valeurs Courantes :

- **60** - 1 heure (test/développement)

- **1440** - 24 heures (production typique)
- **4320** - 3 jours (rétention étendue)
- **10080** - 7 jours (rétention maximale)

Les messages plus anciens que cette valeur deviennent non livrables et sont marqués pour nettoyage.

Configuration de Réessai de Livraison

Le comportement de réessai utilise un backoff exponentiel :

```
Délai de Réessai = 2^(nombre_dessais) minutes
```

Essai	Délai
1	2 minutes
2	4 minutes
3	8 minutes
4	16 minutes
5	32 minutes
6	64 minutes
7	128 minutes
8	256 minutes

Essais maximum avant lettre morte : Limité par `dead_letter_time_minutes`.

Configuration de Nettoyage

```
# config/config.exs
config :sms_c,
  cleanup_interval_minutes: 10,
  fingerprint_ttl_minutes: 5,
  event_ttl_days: 7
```

Intervalles de Nettoyage :

- **cleanup_interval_minutes** : À quelle fréquence le travailleur de nettoyage s'exécute (par défaut : 10)
- **fingerprint_ttl_minutes** : Fenêtre de détection des doublons (par défaut : 5)
- **event_ttl_days** : Rétention des journaux d'événements (par défaut : 7)

Configuration de Facturation

La facturation en ligne est effectuée via l'interface **Diameter Ro** (Application de Contrôle de Crédit, RFC 4006). Pour chaque message facturé, le SMS-C envoie une Credit-Control-Request (Event-Request, `CC-Request-Type 4`) à l'OCS pour le **expéditeur** du message ; l'OCS autorise ou refuse en fonction du solde de l'abonné. Cela réutilise la pile Diameter (voir la section Diameter Sh ci-dessous) - l'application `:ro` doit être enregistrée.

Activer la Facturation

```
# config/runtime.exs
config :sms_c,
  default_charging_enabled: true

config :sms_c, :charging,
  # Que faire lorsque l'expéditeur est à découvert :
  #   :reject     - marquer le message :balance_rejected et ne pas
le livrer
  #   :auto_reply - envoyer également SMS
out_of_balance_reply_text à l'expéditeur
  out_of_balance_action: :reject,
  out_of_balance_reply_text: "Votre message n'a pas pu être envoyé
en raison d'un solde insuffisant.",
  # Mode d'échec lorsque l'OCS rencontre une erreur ou que la pile
Diameter n'est pas configurée
  # (PAS une réponse définitive de découvert) :
  #   :allow  - échouer ouvert, livrer le message (par défaut)
  #   :reject - échouer fermé, traiter comme à découvert
  on_charging_error: :allow,
  # Profil de tarification sélectionnant comment l'OCS tarifie le
message. Envoyé comme le CCR
  # Service-Context-Id.
  rating_profile: "sms@sms-c"
```

La facturation nécessite également que la pile Diameter soit activée avec l'application `:ro` (le Contrôle de Crédit est IETF de base, fournisseur 0) :

```
config :sms_c,
  diameter_enabled: true

config :diameter_ex,
  diameter: %{
    service_name: :omnimessage,
    host: "smsc01",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    # ... voir la section Diameter Sh pour la pile/peers complète
...
    applications: [
      %{
        application_name: :ro,
        application_dictionary: :diameter_gen_3gpp_ro,
        vendor_specific_application_ids: [
          %{vendor_id: 0, auth_application_id: 4,
acct_application_id: nil}
        ]
      }
    ]
  }
```

Facturation par Route

La politique `charged` de chaque route contrôle la facturation indépendamment de la valeur par défaut globale :

`charged`	Comportement
`:yes`	Toujours vérifier le solde via l'OCS
`:no`	Ne jamais facturer
`:default`	Utiliser `default_charging_enabled`

Désactiver la Facturation

```
config :sms_c,
  default_charging_enabled: false
```

Lorsqu'elle est désactivée (par défaut), les routes `:default` acheminent sans aucune vérification de solde OCS. Les routes individuelles peuvent toujours opter pour `charged: :yes`.

Valeurs par défaut et dégradation gracieuse

Le bloc `config :sms_c, :charging` est optionnel - chaque clé a une valeur par défaut raisonnable (`out_of_balance_action: :reject`, un texte de réponse intégré, et `rating_profile: "sms@sms-c"`), et `default_charging_enabled` par défaut à `false`.

Par défaut, la facturation **échoue en mode ouvert** (`on_charging_error: :allow`), correspondant au chemin de facturation HTTP précédent qui enregistrait les échecs mais ne bloquait jamais la livraison :

- Si une route demande une facturation mais que la pile Diameter **n'est pas** activée/configurée, le message est **autorisé** (enregistré comme un avertissement).
- Si l'OCS est injoignable ou renvoie une erreur de protocole, le message est **autorisé**.

Définissez `on_charging_error: :reject` pour **échouer en mode fermé** à la place - les erreurs OCS et une pile non configurée sont alors traitées comme hors équilibre et l'`out_of_balance_action` s'applique.

Dans tous les cas, une **réponse définitive hors équilibre** de l'OCS déclenche toujours l'`out_of_balance_action` configuré. Les messages rejetés sont marqués avec le statut `:balance_rejected`.

Configuration Diameter Sh (HSS)

OmniMessage peut interroger le HSS via l'interface Diameter Sh pour détecter les abonnés sur le réseau qui sont temporairement hors ligne, empêchant les messages d'être routés par la passerelle par défaut. Consultez le Guide de recherche d'abonnés HSS pour la documentation opérationnelle complète.

```
# Activer le dip HSS dans le chemin de routage
config :sms_c,
  diameter_enabled: true

# Configuration de la pile Diameter
config :diameter_ex,
  diameter: %{
    service_name: :omnimessage,
    listen_ip: "0.0.0.0",
    listen_port: 3868,
    decode_format: :map,
    host: "smsc01",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{vendor_id: 10415, auth_application_id: 16_777_217,
acct_application_id: nil}
        ]
      }
    ],
    peers: [
      %{
        host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
        ip: "10.0.0.1",
        port: 3868,
        realm: "epc.mnc001.mcc001.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
```

Paramètre	Type	Requis	Par défaut	Description
`diameter_enabled`	Booléen	Non	`false`	Activer les dips d'abonnés HSS lo de la correspondance d routes (pour les routes avec critèr `:on_net`/`:off_ne` Lorsque `false`, c critères échouent en mode fermé
`hss_cache_ttl_seconds`	Entier	Non	`7200`	TTL (secondes) p les résultats de recherche HSS m en cache
`mock_sh`	Booléen	Non	`false`	Utiliser des réponses HSS simulées pour les tests (aucune pile Diameter démarr
`mock_sh_on_net_numbers`	Liste	Non	`[]`	MSISDN traités comme sur le réseau en mode simulé

Pour la référence complète des paramètres de la pile Diameter (hôte, royaume, pairs, applications), consultez le Guide de recherche d'abonnés HSS.

Configuration ENUM

Recherches de numéros E.164 basées sur DNS pour un routage intelligent.

Désactiver ENUM (Par défaut)

```
# config/runtime.exs
config :sms_c,
  enum_enabled: false
```

Activer ENUM avec DNS par défaut

```
config :sms_c,
  enum_enabled: true,
  enum_domains: ["e164.arpa", "e164.org"],
  enum_dns_servers: [],  # Utiliser le DNS par défaut du système
  enum_timeout: 5000  # 5 secondes
```

Activer ENUM avec des serveurs DNS personnalisés

```
config :sms_c,
  enum_enabled: true,
  enum_domains: ["e164.internal.example.com", "e164.arpa"],
  enum_dns_servers: [
    {"10.0.1.53", 53},  # Serveur DNS interne
    {"8.8.8.8", 53},    # DNS public de Google (de secours)
    {"1.1.1.1", 53}     # DNS Cloudflare (de secours)
  ],
  enum_timeout: 3000  # 3 secondes (échec plus rapide)
```

Priorité des domaines ENUM

Les domaines sont interrogés dans l'ordre jusqu'à une recherche réussie :

```
config :sms_c,
  enum_domains: [
    "e164.internal.example.com",  # Essayer d'abord interne
    "e164.carrier.net",            # Puis le transporteur
    "e164.arpa"                    # Puis le registre public
  ]
```

Optimisation des performances ENUM

Pour les réseaux à faible latence :

```
enum_timeout: 2000  # 2 secondes
```

Pour les liaisons à haute latence/satellite :

```
enum_timeout: 10000  # 10 secondes
```

Exemple de configuration DNS ENUM

Configurer une zone ENUM privée (format BIND9) :

```
; Fichier de zone pour e164.internal.example.com
$ORIGIN e164.internal.example.com.
$TTL 300

; Numéro : +1-555-0100 devient
0.0.1.0.5.5.5.1.e164.internal.example.com
0.0.1.0.5.5.5.1.e164.internal.example.com. IN NAPTR 100 10 "u"
"E2U+sip" "!^.*$!sip:15550100@voip-gateway.example.com!" .
0.0.1.0.5.5.5.1.e164.internal.example.com. IN NAPTR 100 20 "u"
"E2U+pstn" "!^.*$!pstn:gateway-a.example.com!" .

; Numéro : +1-555-0200
0.0.2.0.5.5.5.1.e164.internal.example.com. IN NAPTR 100 10 "u"
"E2U+sip" "!^.*$!sip:15550200@voip-gateway.example.com!" .
```

Tester la résolution ENUM :

```
# Interroger le domaine ENUM
dig @10.0.1.53 NAPTR 0.0.1.0.5.5.5.1.e164.internal.example.com

# La sortie attendue inclut les enregistrements NAPTR :
# 0.0.1.0.5.5.5.1.e164.internal.example.com. 300 IN NAPTR 100 10
"u" "E2U+sip" "!^.*$!sip:15550100@voip-gateway.example.com!" .
```

Configuration de la traduction des numéros

Normalisation des numéros basée sur des regex appliquée avant le routage.

Désactiver la traduction des numéros

```
# config/runtime.exs
config :sms_c,
  translation_rules: []
```

Exemples de traduction de numéros de base

Ajouter le code pays aux numéros locaux :

```
config :sms_c,
  translation_rules: [
    %{
      calling_prefix: nil,
      called_prefix: "",
      source_smsc: nil,
      calling_match: "^(\d{10})$",          # Correspondre aux
numéros à 10 chiffres
      calling_replace: "+1\1",               # Préfixer +1
      called_match: "^(\d{10})$",
      called_replace: "+1\1",
      priority: 100,
      description: "Ajouter +1 aux numéros nord-américains à 10
chiffres",
      enabled: true
    }
  ]
```

Normaliser le format international :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^00(\d+)$",                # Correspondre au
préfixe 00
  calling_replace: "+\1",                    # Remplacer par +
  called_match: "^00(\d+)$",
  called_replace: "+\1",
  priority: 10,
  description: "Convertir le préfixe international 00 en +",
  enabled: true
}
```

Supprimer les caractères de formatage :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^\+?1?[\s\-\.\(\)]*(\d{3})[\s\-\.\)\(]*(\d{3})
[\s\-\.\(\)]*(\d{4})$",
  calling_replace: "+1\1\2\3",
  called_match: "^\+?1?[\s\-\.\(\)]*(\d{3})[\s\-\.\)\(]*(\d{3})
[\s\-\.\(\)]*(\d{4})$",
  called_replace: "+1\1\2\3",
  priority: 50,
  description: "Normaliser le format des numéros de téléphone
américains",
  enabled: true
}
```

Traduction spécifique au transporteur

Suppression du code de routage :

```
%{
  calling_prefix: nil,
  called_prefix: "101",                        # Seulement pour le
préfixe 101
  source_smsc: "carrier_a",                    # Seulement de ce
transporteur
  calling_match: nil,                          # Ne pas changer
l'appelant
  calling_replace: nil,
  called_match: "^101(\d+)$",                 # Supprimer le code
de routage 101
  called_replace: "\1",
  priority: 5,
  description: "Supprimer le code de routage du transporteur du
numéro appelé",
  enabled: true
}
```

Traduction multi-règles

Les règles sont évaluées par ordre de priorité (numéro inférieur = priorité plus élevée) :

```
config :sms_c,
  translation_rules: [
    # Priorité 1 : Règles les plus spécifiques en premier
    %{
      calling_prefix: "1555",
      called_prefix: nil,
      source_smsc: nil,
      calling_match: "^(1555\d{7})$",
      calling_replace: "+\1",
      called_match: nil,
      called_replace: nil,
      priority: 1,
      description: "Normalisation des numéros premium",
      enabled: true
    },

    # Priorité 50 : Règles générales
    %{
      calling_prefix: nil,
      called_prefix: nil,
      source_smsc: nil,
      calling_match: "^(\d{10})$",
      calling_replace: "+1\1",
      called_match: "^(\d{10})$",
      called_replace: "+1\1",
      priority: 50,
      description: "Normalisation générale à 10 chiffres",
      enabled: true
    }
  ]
```

Interception légale (X1/X2/X3)

OmniMessage agit comme un point d'interception de service de messages courts (POI). OmniMessage implémente les interfaces d'interception ETSI TS

103 221. OmniMessage fonctionne en tant qu'élément de réseau (NE). Une installation de surveillance des forces de l'ordre (LEMF) provisionne les cibles d'interception via X1. Lorsqu'un message correspond à une cible provisionnée, OmniMessage encode le message en tant qu'enregistrement xIRI TS 33.128. OmniMessage envoie ensuite l'enregistrement au LEMF via X2.

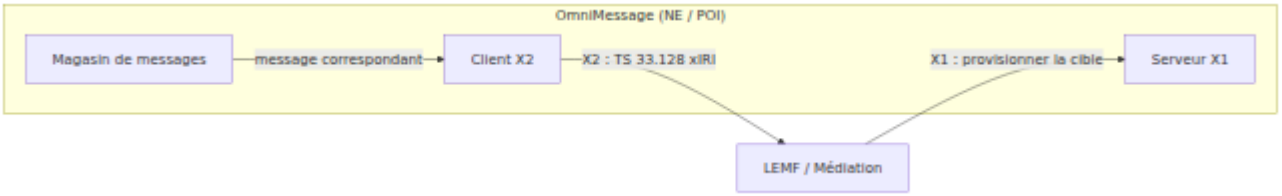
Vue d'ensemble de l'interface

Interface	Objectif	Rôle d'OmniMessage	Transport	Utilisé
X1	Provisionner et supprimer des tâches et des destinations d'interception	Serveur	HTTPS avec TLS mutuel	Oui
X2	Livrer des informations liées à l'interception (chaque SMS intercepté en tant que TS 33.128 xIRI)	Client	TCP, TLS optionnel	Oui
X3	Livrer le contenu de la communication (média)	Client	TCP	Non. Le SMS n'a pas de plan média.

Références aux normes :

Référence	Titre
ETSI TS 103 221-1	Interface d'interception interne X1 (provisionnement)
ETSI TS 103 221-2	Interface d'interception interne X2/X3 (livraison)
3GPP TS 33.128	Formats de charge utile d'interception légale

Architecture



Configuration

Configurez l'interception sous `config :sms_c, :li`. Définissez les valeurs dans `config/runtime.exs` pour chaque déploiement. L'interception est désactivée par défaut. Aucun processus d'interception ne démarre, et aucune interception ne se produit, jusqu'à ce que vous définissiez `enabled: true`.

```
config :sms_c, :li,
  enabled: true,
  ne_identifier: "smsc-poi-01",
  x1: [
    enabled: true,
    port: 8443,
    tls: [
      certfile: "/etc/omnimessage/li/x1-cert.pem",
      keyfile: "/etc/omnimessage/li/x1-key.pem",
      cacertfile: "/etc/omnimessage/li/lemf-ca.pem"
    ]
  ],
  x2: [enabled: true, address: "192.0.2.10", port: 12345, tls:
false]
```

Paramètres de niveau supérieur

Paramètre	Type	Requis	Par défaut	Description
`enabled`	Booléen	Non	`false`	Interrupteur principal. Lorsque `false`, aucun processus d'interception ne démarre.
`ne_identifier`	Chaîne	Non	`"smsc-01"`	Identifiant de fonction réseau (attribut 18 TS 103 221-2). OmniMessage envoie cette valeur sur chaque enregistrement X2. Le LEMF l'utilise pour étiqueter la connexion par nom au lieu de par adresse IP.
`x1`	Liste de mots-clés	Non	`[]`	Serveur de provisionnement X1. Voir Paramètres X1.
`x2`	Liste de mots-clés	Non	`[]`	Client de livraison X2. Voir Paramètres X2.
`x3`	Liste de mots-clés	Non	`[]`	Accepté pour la symétrie avec la bibliothèque d'interception. OmniMessage n'utilise pas X3. Laissez-le comme `[enabled: false]`.

Paramètres X1

OmniMessage exécute le serveur X1. Le LEMF se connecte au serveur pour provisionner des cibles. TLS est obligatoire (clause 7.2.2.1 TS 103 221-1). Le serveur ne démarre pas en mode texte clair à moins que vous ne définissiez `allow_insecure: true`.

Paramètre	Type	Requis	Par défaut	Description
`enabled`	Booléen	Non	`false`	Démarrer le serveur X1.
`port`	Entier	Non	`8443`	Port TCP pour le serveur X1.
`tls`	Liste de mots-clés	Oui	-	Paramètres TLS pour le serveur X1. Voir Paramètres TLS X1.
`allow_insecure`	Booléen	Non	`false`	Exécuter le serveur X1 sur HTTP en texte clair. Utilisez cette option uniquement dans un laboratoire. Ne pas utiliser cette option en production.

Paramètres TLS X1

Paramètre	Type	Requis	Par défaut	Description
`certfile`	Chaîne	Oui	-	Chemin vers le certificat du serveur, au format PEM.
`keyfile`	Chaîne	Oui	-	Chemin vers la clé privée du serveur, au format PEM.
`cacertfile`	Chaîne	Non	-	Chemin vers l'autorité de certification qui vérifie les certificats clients. Définissez cette valeur pour activer TLS mutuel. Les déploiements en production nécessitent TLS mutuel.

Paramètres X2

OmniMessage ex��cute le client X2. Le client se connecte au LEMF. Il envoie chaque SMS intercepté en tant qu'enregistrement TS 33.128 xIRI. L'enregistrement contient le message SMS et un identifiant de cible correspondante.

Paramètre	Type	Requis	Par défaut	Description
`enabled`	Booléen	Non	`false`	Démarrer le client X2.
`address`	Chaîne	Oui	-	Adresse IP ou nom d'hôte du point de terminaison X2 du LEMF.
`port`	Entier	Oui	-	Port TCP du point de terminaison X2 du LEMF.
`tls`	Booléen ou liste de mots-clés	Non	`false`	Définir `true` ou une liste de mots-clés d'options TLS pour livrer X2 via TLS. Définir `false` pour TCP en texte clair.

Persistance

OmniMessage conserve par défaut les tâches et destinations d'interception provisionnées en mémoire. Pour conserver les cibles provisionnées lors d'un redémarrage, activez le stockage persistant d'interception dans la configuration de déploiement. OmniMessage écrit les tâches et destinations sur disque. OmniMessage les recharge au prochain démarrage.

Conformité

Pour un aperçu de la conformité à l'interception, consultez Conformité à l'interception ANSSI R226.

Configuration de routage

Règles de routage initiales chargées au premier démarrage. Consultez le Guide de routage SMS pour la documentation complète sur le routage.

Charger les routes à partir de la configuration

```
# config/runtime.exs
config :sms_c,
  sms_routes: [
    # Exemple de routage géographique
    %{
      calling_regex: nil,
      called_regex: ~r/^\+1/,
      source_smsc: nil,
      dest_smsc: "north_america_gateway",
      source_type: nil,
      enum_domain: nil,
      enum_result_domain: nil,
      action: :route,
      auto_reply_message: nil,
      charged: :default,
      originating_network: :any,
      terminating_network: :any,
      weight: 100,
      priority: 50,
      description: "Routage Amérique du Nord",
      enabled: true
    },

    # Exemple de routage équilibré
    %{
      calling_regex: nil,
      called_regex: ~r/^\+44/,
      source_smsc: nil,
      dest_smsc: "uk_gateway_1",
      source_type: nil,
      enum_domain: nil,
      enum_result_domain: nil,
      action: :route,
      auto_reply_message: nil,
      charged: :default,
      originating_network: :any,
      terminating_network: :any,
      weight: 70,
      priority: 50,
      description: "Passerelle principale du Royaume-Uni (70%)",
      enabled: true
```

```
    },
    %{
      calling_regex: nil,
      called_regex: ~r/^\+44/,
      source_smsc: nil,
      dest_smsc: "uk_gateway_2",
      source_type: nil,
      enum_domain: nil,
      enum_result_domain: nil,
      action: :route,
      auto_reply_message: nil,
      charged: :default,
      originating_network: :any,
      terminating_network: :any,
      weight: 30,
      priority: 50,
      description: "Passerelle de secours du Royaume-Uni (30%)",
      enabled: true
    },

    # Sur le réseau uniquement - restreindre un lien SMPP aux
destinations sur le réseau uniquement
    %{
      calling_regex: nil,
      called_regex: nil,
      source_smsc: "carrier_smpp_bind",
      dest_smsc: "local_msc",
      source_type: :smpp,
      enum_domain: nil,
      enum_result_domain: nil,
      action: :route,
      auto_reply_message: nil,
      charged: :default,
      originating_network: :any,
      terminating_network: :on_net,
      weight: 100,
      priority: 50,
      description: "Transporteur X - terminaison sur le réseau
uniquement",
      enabled: true
    }
  ]
```

Ignorer le chargement initial des routes

```
# Ne pas charger les routes à partir de la configuration (gérer
uniquement via l'interface Web)
config :sms_c,
  sms_routes: []
```

Les routes définies dans la configuration ne sont CHARGÉES que si la table de routage est vide (premier démarrage).

Configuration d'optimisation des performances

Travailleur d'insertion par lots

```
# config/config.exs
config :sms_c,
  batch_insert_batch_size: 100,           # Messages par lot
  batch_insert_flush_interval_ms: 100     # Temps d'attente
maximum en ms
```

Profils de performance :

Profil	Taille du lot	Intervalle	Débit	Latence
Volume élevé	200	200ms	~5,000 msg/sec	Jusqu'à 200ms
Équilibré	100	100ms	~4,500 msg/sec	Jusqu'à 100ms
Faible latence	50	20ms	~3,000 msg/sec	Jusqu'à 20ms
Temps réel	10	10ms	~1,500 msg/sec	Jusqu'à 10ms

Configuration des journaux

Niveaux de journalisation

```
# config/config.exs
config :logger, :console,
  level: :info,  # :debug, :info, :warning, :error
  format: "$time $metadata[$level] $message\n",
  metadata: [:request_id, :message_id, :route_id]
```

Recommandé en production : `:info` ou `:warning` **Recommandé en développement** : `:debug`

Destinations de sortie des journaux

Console uniquement (Développement) :

```
config :logger,
  backends: [:console]
```

Journaliseur de fichiers (Production) :

```
config :logger,
  backends: [:console, {LoggerFileBackend, :file_log}]

config :logger, :file_log,
  path: "/var/log/sms_c/application.log",
  level: :info,
  format: "$time $metadata[$level] $message\n",
  metadata: [:request_id, :message_id]
```

Rotation des journaux

Utilisation de logrotate (Linux) :

```
# /etc/logrotate.d/sms_c
/var/log/sms_c/*.log {
    daily
    rotate 30
    compress
    delaycompress
    notifempty
    create 0644 sms_user sms_group
    sharedscripts
    postrotate
        # Signaler à l'application de rouvrir le fichier journal
        systemctl reload omnimessage
    endscript
}
```

Scénarios de configuration courants

Agrégateur à volume élevé

Optimiser pour un débit maximum (5,000+ messages/seconde) :

```
# Base de données
config :sms_c, SmsC.Repo,
  pool_size: 50

# Travailleur par lots
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# Conservation des messages
config :sms_c,
  dead_letter_time_minutes: 1440  # 24 heures

# Facturation (désactivée pour des performances)
config :sms_c,
  default_charging_enabled: false

# Nettoyage (intervalles prolongés)
config :sms_c,
  cleanup_interval_minutes: 30
```

Messagerie en temps réel pour les entreprises

Optimiser pour une faible latence (< 20ms) :

```
# Base de données
config :sms_c, SmsC.Repo,
  pool_size: 20

# Travailleur par lots (faible latence)
config :sms_c,
  batch_insert_batch_size: 20,
  batch_insert_flush_interval_ms: 10

# Conservation des messages
config :sms_c,
  dead_letter_time_minutes: 4320  # 3 jours

# Facturation (activée) - vérifications d'équilibre via Diameter
Ro
# (voir la section Configuration de la facturation pour les blocs
:charging + :diameter_ex)
config :sms_c,
  default_charging_enabled: true
```

Fournisseur de services multi-locataires

Configuration séparée par locataire :

```
# Environnement du locataire 1
export DB_NAME=sms_c_tenant1
export NODE_NAME=sms_tenant1@node1.example.com
# La facturation par OCS par locataire est sélectionnée via la
configuration :charging rating_profile

# Environnement du locataire 2
export DB_NAME=sms_c_tenant2
export NODE_NAME=sms_tenant2@node1.example.com
```

Redondance géographique

Cluster à travers les régions :

```
# Cluster US Est
config :sms_c,
  cluster_nodes: [
    :"sms@us-east-1a.example.com",
    :"sms@us-east-1b.example.com",
    :"sms@us-west-1a.example.com"  # Inter-régional pour DR
  ],
  smsc_node_name: "us-east-1a"
```

Validation de la configuration

Tester la configuration avant le déploiement :

```
# Vérifier la syntaxe de la configuration
mix compile

# Valider la connexion à la base de données
mix ecto.create
mix ecto.migrate

# Vérifier le pair Diameter Ro vers l'OCS (si la facturation est
activée), dans une console distante :
#   :diameter.service_info(:omnimessage, :connections)

# Démarrer l'application en mode interactif
iex -S mix phx.server
```

Référence des Variables d'Environnement

Variables d'environnement courantes utilisées dans la configuration :

Variable	But	Exemple
`DB_USERNAME`	Nom d'utilisateur de la base de données	`sms_prod_user`
`DB_PASSWORD`	Mot de passe de la base de données	`strong_password`
`DB_HOSTNAME`	Hôte de la base de données	`db.internal.example.com`
`DB_PORT`	Port de la base de données	`3306`
`DB_NAME`	Nom de la base de données	`sms_c_production`
`DB_POOL_SIZE`	Taille du pool de connexions	`30`
`API_PORT`	Port d'écoute de l'API	`8443`
`API_LISTEN_IP`	IP d'écoute de l'API	`0.0.0.0`
`WEB_PORT`	Port de l'interface Web	`443`
`NODE_NAME`	Nom du nœud Erlang	`sms@node1.example.com`
`ERLANG_COOKIE`	Secret de cluster	`shared_cookie_value`

Meilleures Pratiques de Configuration

1. **Utilisez des Variables d'Environnement** pour les valeurs sensibles (mots de passe, clés API)
2. **Testez les Changements de Configuration** en staging avant la production
3. **Documentez les Paramètres Personnalisés** dans les notes de déploiement

4. **Contrôlez les Fichiers de Configuration en Version** (à l'exclusion des secrets)
5. **Surveillez Après les Changements** pour détecter les régressions de performance
6. **Conservez des Sauvegardes** des configurations fonctionnelles
7. **Validez Avant de Redémarrer** pour éviter les échecs de démarrage
8. **Utilisez une Nomenclature Cohérente** à travers les environnements
9. **Définissez des Limites de Ressources** appropriées au matériel
10. **Révisez Périodiquement** pour supprimer les fonctionnalités inutilisées

Dépannage des Problèmes de Configuration

Symptôme	Cause Probable	Solution
L'application ne démarre pas	Erreur de syntaxe dans la configuration	Vérifiez les journaux, validez la syntaxe
La connexion à la base de données échoue	Mauvaises informations d'identification/hôte	Vérifiez les variables d'environnement DB_*
API non accessible	Mauvais port/lien IP	Vérifiez API_PORT et listen_ip
Les nœuds du cluster ne se connectent pas	Mismatch de cookie, pare-feu	Vérifiez ERLANG_COOKIE, vérifiez les ports 4369, 9100-9200
Échecs de facturation	OCS/DRA injoignable	Vérifiez le pair Diameter Ro (les échecs de facturation sont ouverts)
Les recherches ENUM échouent	Serveur DNS injoignable	Testez la connectivité DNS, vérifiez le délai d'attente
Mauvaises performances	Mauvaises configurations de lot	Consultez la section Configuration de l'Optimisation des Performances ci-dessus
Les messages ne sont pas routés	Routes non chargées	Vérifiez la configuration sms_routes ou l'interface Web

Pour une aide supplémentaire, consultez le Guide de Dépannage.

Configuration de Stockage des Messages (Mnesia)

Conservation des Messages

Les messages sont stockés dans Mnesia pour un accès rapide avec un nettoyage automatique configurable.

```
config :sms_c,
  # Combien de temps garder les messages dans Mnesia (heures).
  # La valeur par défaut intégrée est 24 si non définie ; le
runtime.exs fourni définit 168 (7 jours).
  message_retention_hours: 168,

  # À quelle fréquence vérifier les anciens messages (minutes)
  retention_check_interval_minutes: 60
```

Recommandations :

- **Production :** 24-72 heures (équilibrer les besoins opérationnels et la mémoire)
- **Développement :** 4-8 heures (nettoyage plus rapide pour les tests)
- **Volume élevé :** 12-24 heures (conserver la mémoire)

Impact sur la Mémoire :

- Message moyen : ~1 Ko
- 10 000 messages : ~10 Mo
- 100 000 messages : ~100 Mo

Exportation des CDR (Call Detail Record)

Lorsque les messages sont livrés ou expirés, les CDR peuvent être automatiquement écrits dans votre base de données Ecto pour un stockage à long terme et des analyses de facturation.

```
config :sms_c,
  # Activer/désactiver l'écriture des CDR (valeur par défaut
intégrée : false)
  cdr_enabled: true
```

Les Enregistrements CDR Incluent :

- ID du message, numéros appelants/appelés
- SMSC source/destination
- Nœud d'origine/destination (pour les clusters)
- Horodatages de soumission, de livraison, d'expiration
- Statut, tentatives de livraison
- Corps de message optionnel (voir les contrôles de confidentialité)

Quand Désactiver :

- Environnements de test où les CDR ne sont pas nécessaires
- Dépannage temporaire pour réduire la charge de la base de données

Contrôles de Confidentialité

Configurez la visibilité et la conservation du corps des messages pour la conformité à la vie privée.

```
config :sms_c,
  # Supprimer le corps du message de Mnesia après une livraison
réussie
  delete_message_body_after_delivery: false,

  # Masquer le corps du message dans l'interface Web
  hide_message_body_in_ui: false,

  # Masquer le corps du message dans les exports CSV
  hide_message_body_in_export: false
```

Cas d'Utilisation :

Configuration	Cas d'Utilisation
`delete_message_body_after_delivery: true`	Économiser de l'espace dans Mnesia, conformité à la vie privée
`hide_message_body_in_ui: true`	Empêcher l'opérateur de voir le contenu du message
`hide_message_body_in_export: true`	Conformité à l'exportation de données, rapports assainis

Configurations d'Exemple :

Confidentialité Maximale (Conformité)

```
config :sms_c,
  delete_message_body_after_delivery: true,
  hide_message_body_in_ui: true,
  hide_message_body_in_export: true,
  cdr_enabled: true  # Conserver les CDR sans corps
```

Développement (Visibilité Totale)

```
config :sms_c,
  delete_message_body_after_delivery: false,
  hide_message_body_in_ui: false,
  hide_message_body_in_export: false,
  cdr_enabled: true
```

Journalisation au Démarrage

Au démarrage de l'application, l'état de la configuration est enregistré :

```
[info] Stockage des messages : Mnesia (conservation : 24h)
[info] Exportation des CDR : ACTIVÉE
[info] Suppression du corps après livraison : DÉSACTIVÉE
[info] Facturation OCS : ACTIVÉE (url : http://..., locataire :
...)
```

Cela fournit une visibilité immédiate sur les fonctionnalités actives.

Fédération Géographique

← Retour à l'Index de Documentation | **Référence de Configuration** | **Guide des Opérations**

Vue d'ensemble

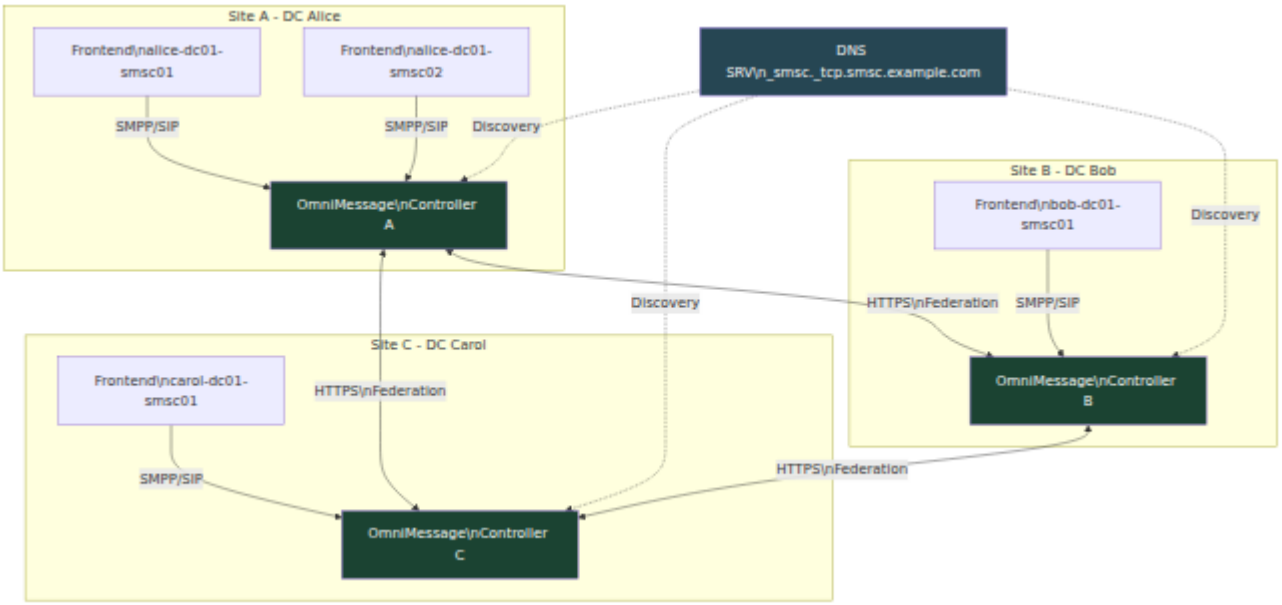
OmniMessage utilise une **fédération basée sur HTTP** avec un **modèle de tirage** pour des déploiements multi-contrôleurs à travers des centres de données ou des régions. Chaque contrôleur fonctionne de manière indépendante - gérant sa propre file de messages, sa table de routage et son ensemble de frontaux connectés. Les contrôleurs se découvrent mutuellement via des enregistrements DNS SRV (ou une configuration statique), échangent des informations sur la santé et le registre des frontaux via HTTPS.

Lorsque le routage détermine qu'un message appartient à un site distant, le message **reste sur le Mnesia du contrôleur d'origine**. Une notification légère est envoyée au contrôleur de destination, ce qui déclenche un sondage immédiat. Le FederationPoller du contrôleur de destination interroge périodiquement (et sur notification) les pairs d'origine pour des messages destinés à ses frontaux locaux, les met en cache et les rend disponibles aux frontaux locaux. Après la livraison, le contrôleur de destination rapporte l'état de livraison au contrôleur d'origine.

Caractéristiques Clés

Aspect	Détail
Réseau	Fonctionne sur n'importe quel WAN, y compris les liens peu fiables
État partagé	Aucun - chaque contrôleur est indépendant
Propriété des messages	Les messages restent sur le contrôleur d'origine jusqu'à livraison
Sécurité	HTTPS avec TLS
Ports	Port HTTPS unique (8443)
Cible d'échelle	5-20 contrôleurs
Gestion des partitions	Réessais du poller - pas de split-brain

Architecture



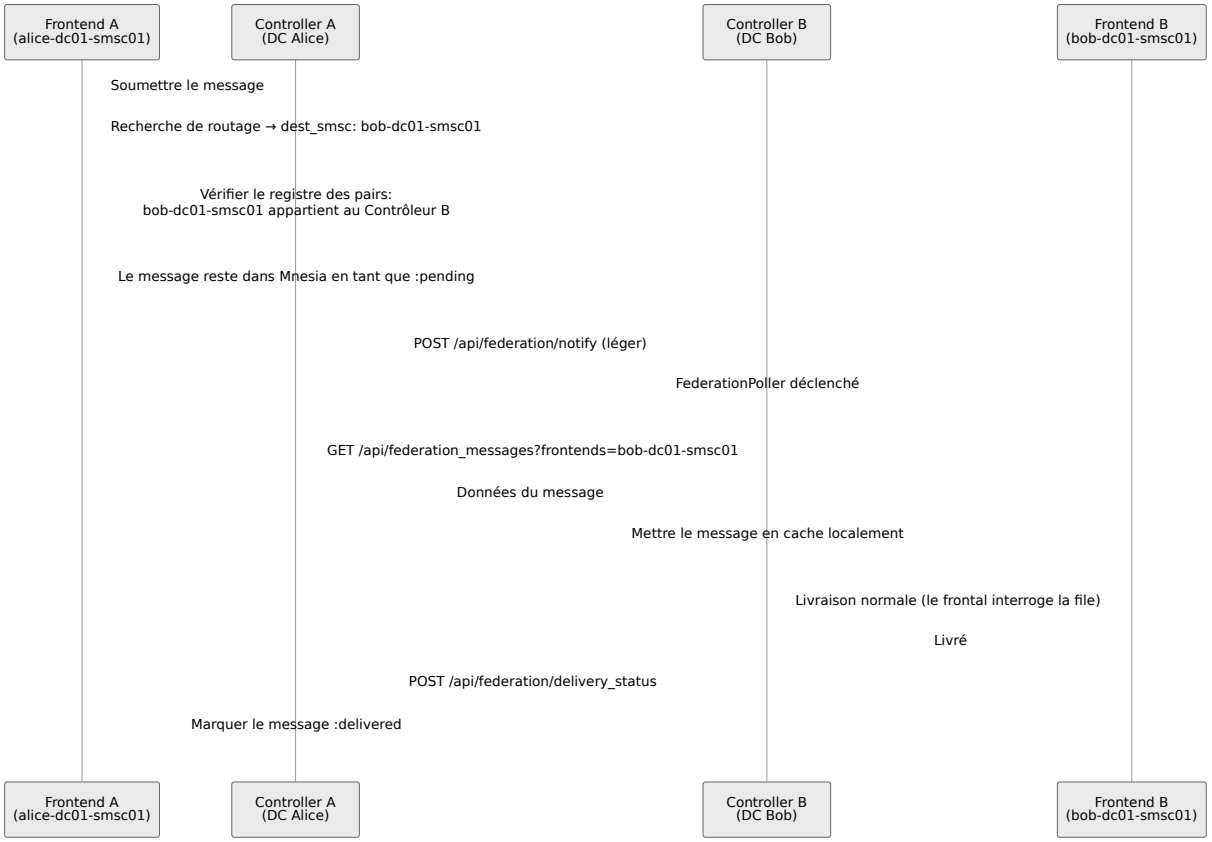
Décisions de Conception Clés

- **Flux de messages (modèle de tirage)** : Les messages restent sur le contrôleur d'origine. Le contrôleur de destination les tire via un sondage périodique et une notification à la demande. Les frontaux ne communiquent qu'avec leur contrôleur d'origine.
- **Gestion des partitions** : Les réessais du poller se font automatiquement. Les messages restent en sécurité sur le contrôleur d'origine jusqu'à ce qu'ils soient livrés avec succès.
- **Synchronisation de configuration** : Aucune. Chaque contrôleur gère sa propre table de routage de manière indépendante.
- **Maillage des pairs** : Maillage complet. Tous les contrôleurs échangent des informations de santé et des registres de frontaux avec tous les autres contrôleurs.
- **Découverte** : Enregistrements DNS SRV (recommandé) ou liste de pairs statiques (pour les environnements sans DNS dynamique).

Flux de Messages (Modèle de Tirage)

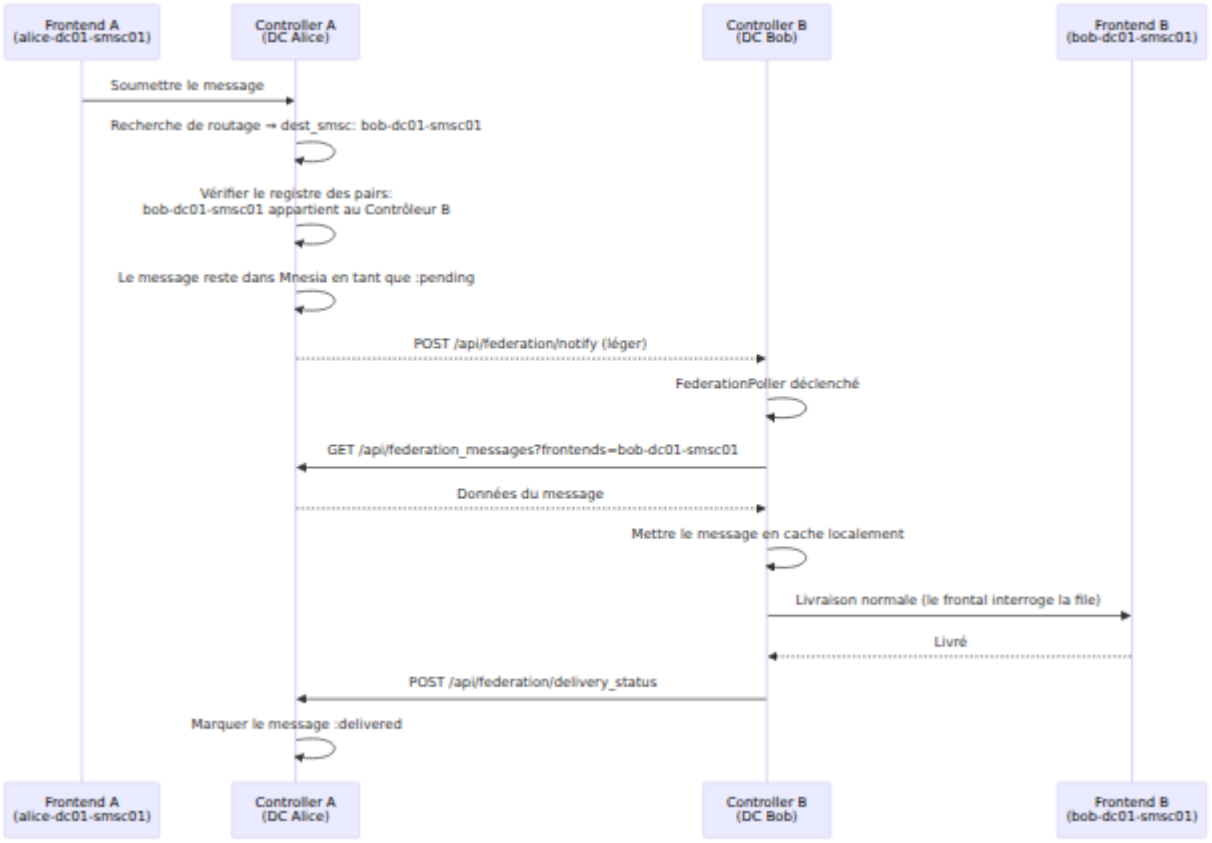
Lorsqu'un message arrive à un contrôleur et que le routage détermine que le frontal de destination se trouve sur un contrôleur distant :

1. Le message reste dans le Mnesia du contrôleur d'origine en tant que `:pending`
2. Une notification légère est envoyée au contrôleur de destination (fire-and-forget)
3. Le FederationPoller du contrôleur de destination récupère le message via GET /api/federation_messages
4. Le message est mis en cache localement et rendu disponible aux frontaux locaux
5. Après la livraison, la destination rapporte l'état au contrôleur d'origine



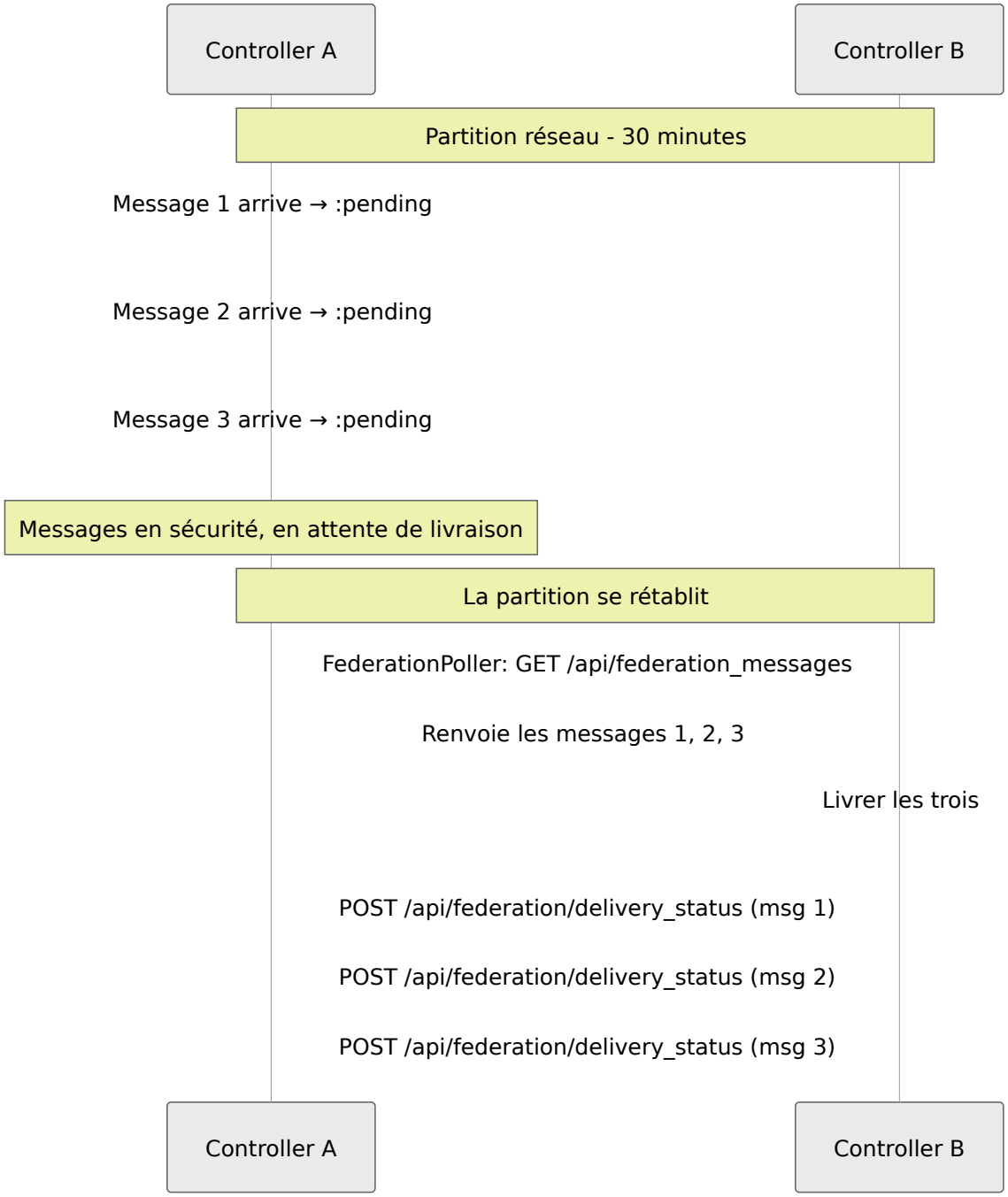
Pair Inaccessible - Échec de Notification

Si le contrôleur de destination est hors ligne lorsque la notification est envoyée, la notification échoue silencieusement. Le message reste `:pending` sur l'origine. Lorsque le contrôleur de destination se rétablit, son FederationPoller reprend le sondage périodique et découvre le message lors du cycle suivant (par défaut : toutes les 5 secondes). Aucun message n'est perdu.



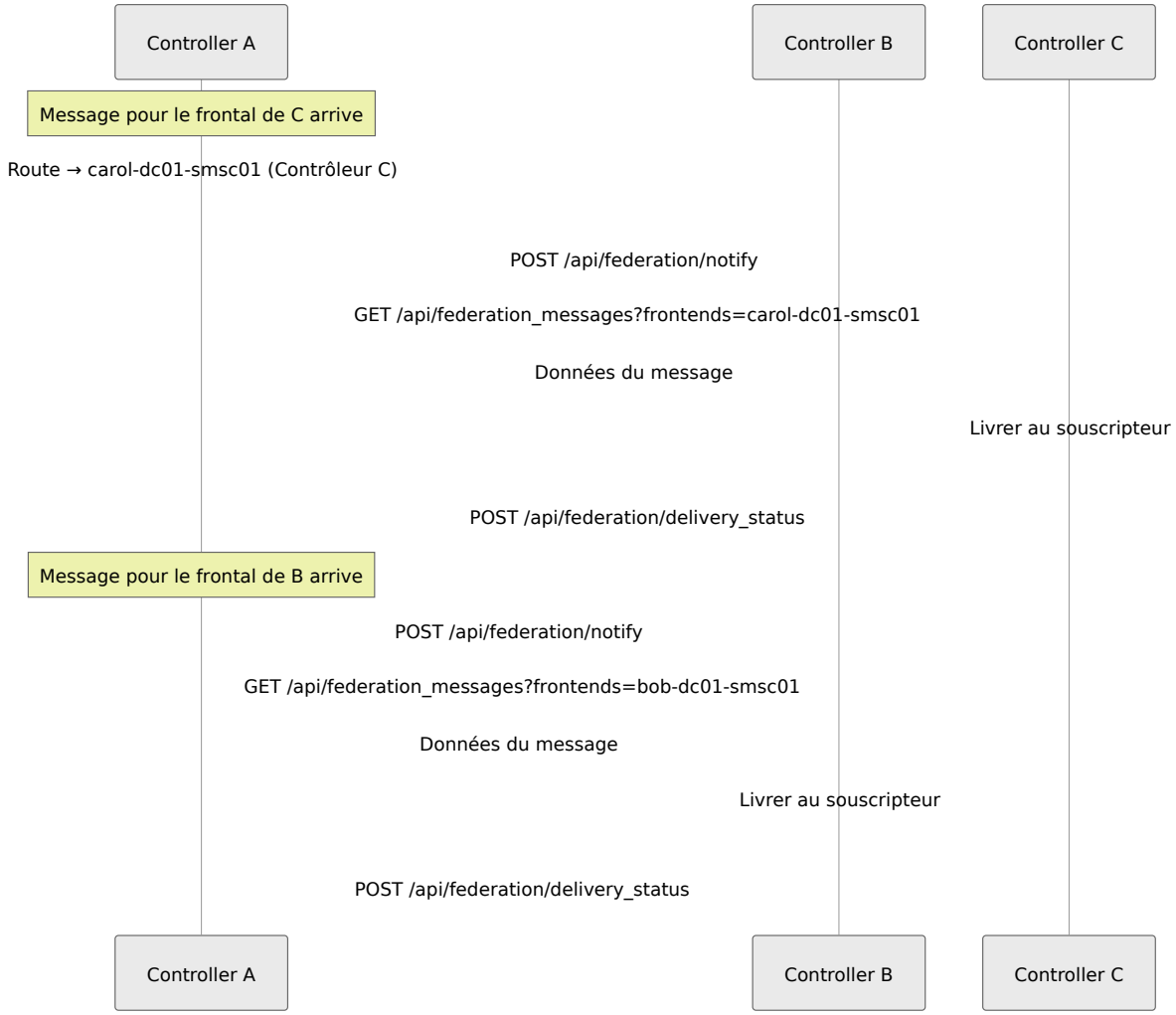
Récupération de Partition - Messages Enregistrés Multiples

Lors d'une partition prolongée, les messages s'accumulent sur le contrôleur d'origine. Lorsque le lien se rétablit, le FederationPoller se met à jour automatiquement - tous les messages en attente sont renvoyés dans une seule requête.



Flux de Messages à Trois Sites

Dans un maillage multi-sites, chaque contrôleur ne sonde que les messages destinés à ses propres frontaux.



Santé des Pairs et Synchronisation de Registre

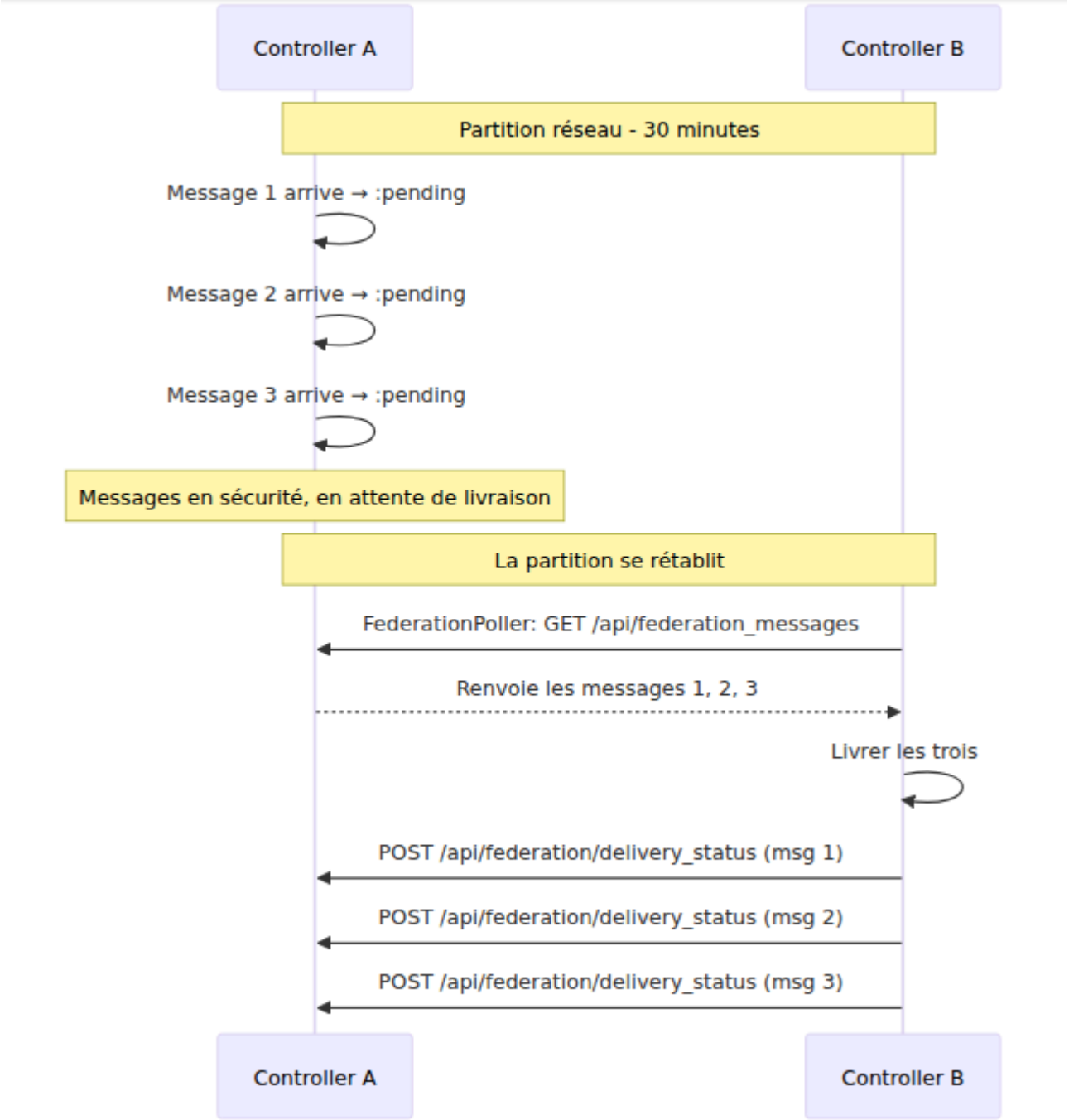
Les contrôleurs de fédération maintiennent la conscience les uns des autres à travers deux cycles périodiques :

Cycle de Vérification de Santé

Toutes les 10 secondes (configurable), chaque contrôleur appelle `POST /api/federation/health` sur chaque pair connu. L'appel est bidirectionnel - l'appelant envoie son propre statut et reçoit le statut du pair dans la réponse. Un pair est marqué **non sain** après 3 échecs consécutifs.

Cycle de Synchronisation de Registre

Toutes les 15 secondes (configurable), chaque contrôleur appelle `POST /api/federation/registry` sur chaque pair sain. L'appel échange des listes de frontaux - l'appelant envoie ses frontaux actifs et reçoit les frontaux actifs du pair. C'est ainsi que les contrôleurs apprennent quel pair possède quel frontal.



Statuts des pairs et leur effet sur le sondage de fédération :

Statut	Vérifications de santé	FederationPoller	Synchronisation de registre
Inconnu	En cours	Non sondé	Non tenté
Sain	Passant	Sondage actif	Actif
Dégradé	1-2 échecs	Non sondé	Non tenté
Non sain	3+ échecs	Non sondé	Non tenté

Configuration

La fédération est configurée dans `config/runtime.exs` sous la clé `:federation`. Elle est **désactivée par défaut** et doit être explicitement activée.

Configuration Minimale (Découverte DNS SRV)

```
# config/runtime.exs
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.smsc.example.com"
```

Référence de Configuration Complète

```
# config/runtime.exs
config :sms_c, :federation,
  # Interrupteur principal - la fédération est complètement
inactive quand faux
  enabled: true,

  # Domaine DNS SRV pour la découverte des pairs (laisser vide
pour utiliser static_peers à la place)
  dns_srv_domain: "_smsc._tcp.smsc.example.com",

  # À quelle fréquence résoudre à nouveau les enregistrements DNS
SRV (millisecondes)
  dns_poll_interval_ms: 30_000,

  # À quelle fréquence échanger l'état de santé avec tous les
pairs connus (millisecondes)
  health_check_interval_ms: 10_000,

  # À quelle fréquence échanger les registres de frontaux avec les
pairs sains (millisecondes)
  registry_sync_interval_ms: 15_000,

  # À quelle fréquence le FederationPoller interroge les pairs
sains pour des messages (millisecondes)
  poll_interval_ms: 5_000,

  # Délai HTTP pour tous les appels API pair-à-pair
(millisecondes)
  http_timeout_ms: 5_000,

  # Port API utilisé lors de la construction des URL de pair à
partir des enregistrements DNS SRV
  api_port: 8443,

  # Liste de pairs statiques - utilisée lorsque dns_srv_domain est
vide
  static_peers: []
```

Paramètres de Fédération

Paramètre	Type	Requis	Par défaut	Description
`enabled`	Boolean	Oui	`false`	Interrupteur principal. Lorsque `false` les services de fédération démarrent ma ne découvrent ni ne contacte les pairs.
`dns_srv_domain`	String	Non	`""`	Domaine DNS SRV pour la découverte automatique des pairs. Lorsqu'il est vide, `static_peers` est utilisé à la place.
`dns_poll_interval_ms`	Integer	Non	`30000`	Intervalle entr les tentatives de résolution DNS SRV. Des valeurs plus basses détectent de nouveaux pai plus rapideme mais

Paramètre	Type	Requis	Par défaut	Description
				augmentent la charge DNS.
`health_check_interval_ms`	Integer	Non	`10000`	Intervalle entre les tours de vérification de santé. Chaque tour contacte chaque pair connu.
`registry_sync_interval_ms`	Integer	Non	`15000`	Intervalle entre les tours de synchronisation de registre de frontaux. Chaque tour contacte chaque pair sain.
`poll_interval_ms`	Integer	Non	`5000`	À quelle fréquence le FederationPoller interroge les pairs sains pour des messages. Les notifications déclenchent des sondages immédiats.
`http_timeout_ms`	Integer	Non	`5000`	Délai pour chaque appel HTTPS

Paramètre	Type	Requis	Par défaut	Description
				individuel à un pair. S'applique aux vérifications de santé, à la synchronisation de registre et la récupération de messages.
`api_port`	Integer	Non	`8443`	Port API utilisé lors de la construction des URL de pair à partir des enregistrements DNS SRV. Doit correspondre au port API configuré dans `config :api_ex`.
`static_peers`	List	Non	`[]`	Liste de configurations de pairs pour les environnements sans DNS SRV. Chaque entrée est une carte avec des clés `host` et `port`

Configuration de Pair Statique

Pour les environnements où DNS SRV n'est pas disponible, configurez les pairs explicitement :

```
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "",
  static_peers: [
    %{host: "10.0.1.2", port: 8443},
    %{host: "10.0.2.2", port: 8443},
    %{host: "10.0.3.2", port: 8443}
  ]
```

Paramètre	Type	Requis	Par défaut	Description
`host`	String	Oui	-	Adresse IP ou nom d'hôte du contrôleur distant.
`port`	Integer	Non	`8443`	Port API HTTPS du contrôleur distant.

Configuration des Enregistrements DNS SRV

Les enregistrements DNS SRV permettent aux contrôleurs de se découvrir automatiquement. Chaque contrôleur résout le domaine configuré et utilise les paires hôte/port résultantes comme pairs.

Configuration de la Zone DNS :

```
; Priorité Poids Port Cible
_smsc._tcp.smsc.example.com. 86400 IN SRV 10 100 8443 smsc-
alice.smsc.example.com.
_smsc._tcp.smsc.example.com. 86400 IN SRV 10 100 8443 smsc-
bob.smsc.example.com.
_smsc._tcp.smsc.example.com. 86400 IN SRV 10 100 8443 smsc-
carol.smsc.example.com.

; Enregistrements A pour les cibles
smsc-alice.smsc.example.com.    86400 IN A 10.0.1.2
smsc-bob.smsc.example.com.    86400 IN A 10.0.2.2
smsc-carol.smsc.example.com.  86400 IN A 10.0.3.2
```

Ajout d'un nouveau site : Ajoutez un nouvel enregistrement SRV au DNS. Tous les contrôleurs existants découvriront le nouveau pair dans un cycle de `dns_poll_interval_ms` (par défaut 30 secondes).

Suppression d'un site : Supprimez l'enregistrement SRV du DNS. Les contrôleurs existants retireront le pair parti dans un cycle de sondage.

Exigences Réseau

La fédération nécessite seulement un port HTTPS unique entre les sites.

Port	Protocole	Direction	But
8443	TCP/TLS	Bidirectionnel	Tout le trafic de fédération (santé, registre, notification, messages, état de livraison)
53	UDP/TCP	Sortant	Résolution DNS SRV (si utilisation de la découverte DNS)

Règles de pare-feu (par site) :

```
# Autoriser le trafic de fédération des IPs de contrôleurs pairs
iptables -A INPUT -p tcp -s 10.0.1.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.2.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.3.0/24 --dport 8443 -j ACCEPT
```

Points de Terminaison de l'API de Fédération

Le contrôleur sert tous les points de terminaison de fédération sur le port API standard (8443) sous `/api/federation/`. Les contrôleurs pairs appellent ces points de terminaison, pas les clients externes ou les frontaux.

Point de terminaison	Méthode	But
`/api/federation/identity`	GET	Renvoie le nom de nœud de ce contrôleur et la version du protocole
`/api/federation/health`	POST	Échange de santé bidirectionnel - l'appelant envoie l'état, le appelé répond avec le sien
`/api/federation/registry`	POST	Échange de registre de frontaux bidirectionnel
`/api/federation/notify`	POST	Recevoir une notification légère qu'un message est disponible pour un frontal local
`/api/federation/delivery_status`	POST	Recevoir une confirmation de livraison du contrôleur de destination
`/api/federation_messages`	GET	Renvoie les messages en attente pour les frontaux spécifiés (utilisé par FederationPoller)

Toutes les requêtes et réponses utilisent JSON. L'identification des pairs est portée dans l'en-tête `X-Federation-Node`.

Exemples de Déploiement

Exemple 1 : Déploiement à Trois Sites avec DNS SRV

Trois centres de données, chacun avec des frontaux locaux pour leur base d'abonnés. DNS SRV fournit la découverte.

```
# Site A - DC Alice (config/runtime.exs)
config :sms_c,
  smsc_node_name: "alice-dc01-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.smsc.example.com"
```

```
# Site B - DC Bob (config/runtime.exs)
config :sms_c,
  smsc_node_name: "bob-dc01-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.smsc.example.com"
```

```
# Site C - DC Carol (config/runtime.exs)
config :sms_c,
  smsc_node_name: "carol-dc01-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.smsc.example.com"
```

Comment ça fonctionne : Tous les trois contrôleurs résolvent le même domaine SRV et se découvrent automatiquement. Chaque contrôleur enregistre ses frontaux locaux. Lorsqu'un message arrive à DC Alice destiné à un abonné servi par DC Bob, la table de routage sélectionne `bob-dc01-smsc01` comme destination. La couche de fédération détecte que ce frontal appartient au

Contrôleur B et envoie une notification. Le FederationPoller du Contrôleur B interroge le Contrôleur A et met en cache le message pour une livraison locale.

Exemple 2 : Deux Sites Actif-Actif avec Pairs Statiques

Deux centres de données connectés par un lien dédié. Pas de DNS SRV disponible.

```
# DC Principal (config/runtime.exs)
config :sms_c,
  smsc_node_name: "primary-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "",
  static_peers: [
    %{host: "10.200.1.5", port: 8443}
  ]
```

```
# DC Secondaire (config/runtime.exs)
config :sms_c,
  smsc_node_name: "secondary-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "",
  static_peers: [
    %{host: "10.100.1.5", port: 8443}
  ]
```

Comment ça fonctionne : Chaque contrôleur est configuré avec l'adresse IP de l'autre. Les vérifications de santé et les synchronisations de registre se déroulent sur le lien dédié. Les messages restent sur le contrôleur qui les a reçus. Le FederationPoller de chaque contrôleur interroge le pair pour des messages destinés aux frontaux locaux.

Exemple 3 : Sondage Aggressif pour Liens à Faible Latence

Pour les sites connectés par des liens fiables et à faible latence où une fédération rapide est importante :

```
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.cluster.internal",
  health_check_interval_ms: 5_000,
  registry_sync_interval_ms: 5_000,
  poll_interval_ms: 1_000,
  http_timeout_ms: 2_000
```

Cas d'utilisation : Déploiements sur campus ou dans des zones métropolitaines où les sites sont connectés par des liens < 10 ms RTT et où un basculement rapide est plus important que la minimisation du trafic de contrôle.

Métriques

La fédération expose les événements de télémétrie suivants qui peuvent être observés via Prometheus sur le port 9568.

Métrique: `sms_c_federation_message_received_count` **Type**: Compteur **Description**: Nombre de notifications de fédération reçues des pairs **Étiquettes**:

- `origin_node` - Pair qui a envoyé la notification

Métrique: `sms_c_federation_health_check_count` **Type**: Compteur **Description**: Résultats de vérification de santé par pair **Étiquettes**:

- `peer` - Identifiant du pair
- `result` - `ok` ou `failed`

Exemples de requêtes:

```
# Taux de notification par minute
rate(sms_c_federation_message_received_count[5m]) * 60

# Échecs de vérification de santé par pair
sum by (peer)
(rate(sms_c_federation_health_check_count{result="failed"}[5m]))
```

Indicateurs Clés à Surveiller

Ce qu'il faut surveiller	Où	Seuil d'alerte
Messages fédérés mis en cache	`GET /api/federation/status` ou Interface Web	Compte croissant (pair non livrant)
État de santé des pairs	`GET /api/federation/identity` sur chaque pair	Tout pair non sain > 5 minutes
Latence de vérification de santé	Journaux d'application	> 2 secondes (dégradation du lien)

Dépannage

Pair Non Découvert

Symptômes: Les journaux du contrôleur montrent aucun pair découvert ; `cluster_status` renvoie vide.

Causes possibles:

- Domaine DNS SRV incorrect ou non résolvable
- Serveur DNS inaccessible depuis l'hôte du contrôleur
- Les enregistrements SRV n'ont pas encore été propagés

- La fédération n'est pas activée (`enabled: false`)

Résolution:

1. Vérifiez la résolution DNS SRV depuis l'hôte du contrôleur : `dig SRV _smsc._tcp.smsc.example.com`
2. Vérifiez que `enabled: true` est défini dans la configuration de fédération
3. Vérifiez les journaux d'application pour les messages "Découverte DNS de fédération" au démarrage
4. Si vous utilisez des pairs statiques, vérifiez que la liste `static_peers` contient des entrées d'hôte/port correctes

Pair Marqué Non Sain

Symptômes: Les messages vers un site spécifique ne sont pas sondés. Les journaux montrent des messages "Échec de vérification de santé".

Causes possibles:

- Le contrôleur distant est hors ligne
- Pare-feu bloquant le port 8443 entre les sites
- Problèmes de certificat TLS
- Partition réseau entre les sites

Résolution:

1. Vérifiez que le contrôleur distant fonctionne : `curl -k https://<peer-host>:8443/api/federation/identity`
2. Vérifiez que les règles de pare-feu autorisent le trafic sur le port 8443 depuis l'IP du contrôleur local
3. Examinez les journaux d'application des deux côtés pour des erreurs TLS ou de connexion
4. Vérifiez la connectivité réseau entre les sites

Messages Non Livrés au Site Distant

Symptômes: Les messages restent `:pending` sur le contrôleur d'origine. Le compte de messages fédérés mis en cache est 0 sur la destination.

Causes possibles:

- Le contrôleur de destination est non sain (le FederationPoller interroge uniquement les pairs sains)
- Le FederationPoller ne fonctionne pas
- Le registre des frontaux n'est pas synchronisé (la destination ne sait pas quels frontaux sont locaux)

Résolution:

1. Vérifiez l'état des pairs sur le contrôleur de destination
2. Vérifiez que le FederationPoller est dans l'arbre de supervision (vérifiez les journaux d'application pour "Poller de fédération démarré")
3. Vérifiez que les registres des frontaux sont synchronisés (`GET /api/federation/status` - vérifiez les listes de frontaux)
4. Testez manuellement le point de terminaison federation_messages : `curl -k 'https://<origin>:8443/api/federation_messages?frontends=<frontend_name>&include_unrouted=false'`

Recherche d'abonné HSS (Diameter Sh)

← Retour à l'index de documentation | **Guide de routage SMS**

Vue d'ensemble

OmniMessage peut interroger le Serveur d'Abonnés à Domicile (HSS) via l'interface Diameter Sh pour déterminer si un MSISDN appartient au réseau (sur le réseau) ou non (hors réseau). Cette réponse est utilisée comme un **critère de correspondance de route** : les routes peuvent exiger que le numéro d'origine ou de terminaison soit sur le réseau ou hors réseau avant de s'appliquer.

Le dip est **paresseux** - il se produit *durant l'évaluation de la table de routage*, uniquement lorsqu'une route en cours d'évaluation déclare une exigence `originating_network` ou `terminating_network` de `:on_net` ou `:off_net`. Les routes qui laissent les deux à `:any` (la valeur par défaut) ne déclenchent jamais un dip. Aucune recherche HSS globale n'est effectuée avant le routage.

OmniMessage envoie une **Demande de Données Utilisateur (UDR)** pour le numéro. Si le HSS confirme que l'abonné existe (code de résultat 2001), le numéro est **sur le réseau** ; si le HSS renvoie *Utilisateur Inconnu* (code de résultat 5001), le numéro est **hors réseau**. Voir Guide de routage SMS → Correspondance sur le réseau / Hors réseau pour savoir comment ces critères se combinent avec le reste d'une route.

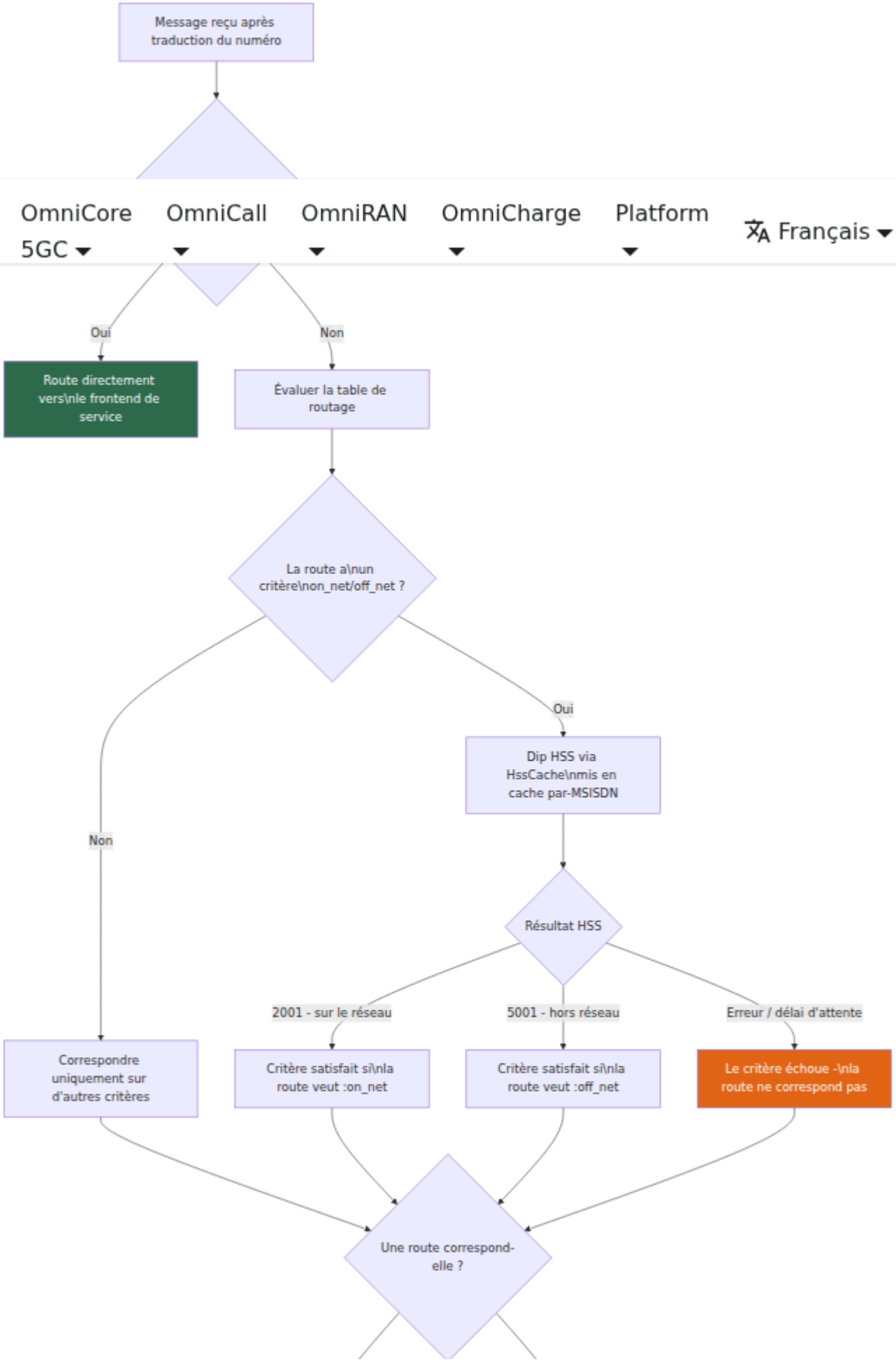
> **Sur le réseau vs hors réseau - HSS ou ENUM ?** Le HSS (Sh) répond "ce numéro est-il un abonné dans *notre* HSS ?" avec un résultat explicite sur le réseau/hors réseau. ENUM (DNS NAPTR) répond "à quel domaine de transport ce numéro se résout-il ?" et vous déduisez sur/hors réseau en faisant correspondre votre propre domaine. Utilisez Sh lorsque votre HSS est autoritaire ; utilisez ENUM pour la portabilité/routage inter-opérateurs. Voir Guide de routage SMS → Recherche ENUM.

Spécifications pertinentes

Spécification	Titre
3GPP TS 29.329	Interface Sh basée sur Diameter
3GPP TS 29.328	Interface IMS Sh - flux de signalisation et contenus des messages
RFC 6733	Protocole de base Diameter

Où s'inscrit le Dip HSS

OmniMessage évalue un message dans cet ordre :


Message reçu après traduction du numéro
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
Oui
Non
Route directement vers\nle frontend de service
Évaluer la table de routage
La route a\nun critère\non_net/off_net ?
Oui
Non
Dip HSS via HssCache\nmis en cache par-MSISDN
Résultat HSS
2001 - sur le réseau
5001 - hors réseau
Erreur / délai d'attente
Correspondre uniquement sur d'autres critères
Critère satisfait si\nla route veut :on_net
Critère satisfait si\nla route veut :off_net
Le critère échoue -\nla route ne correspond pas
Une route correspond-elle ?



Routage basé sur la localisation (vérifié en premier)

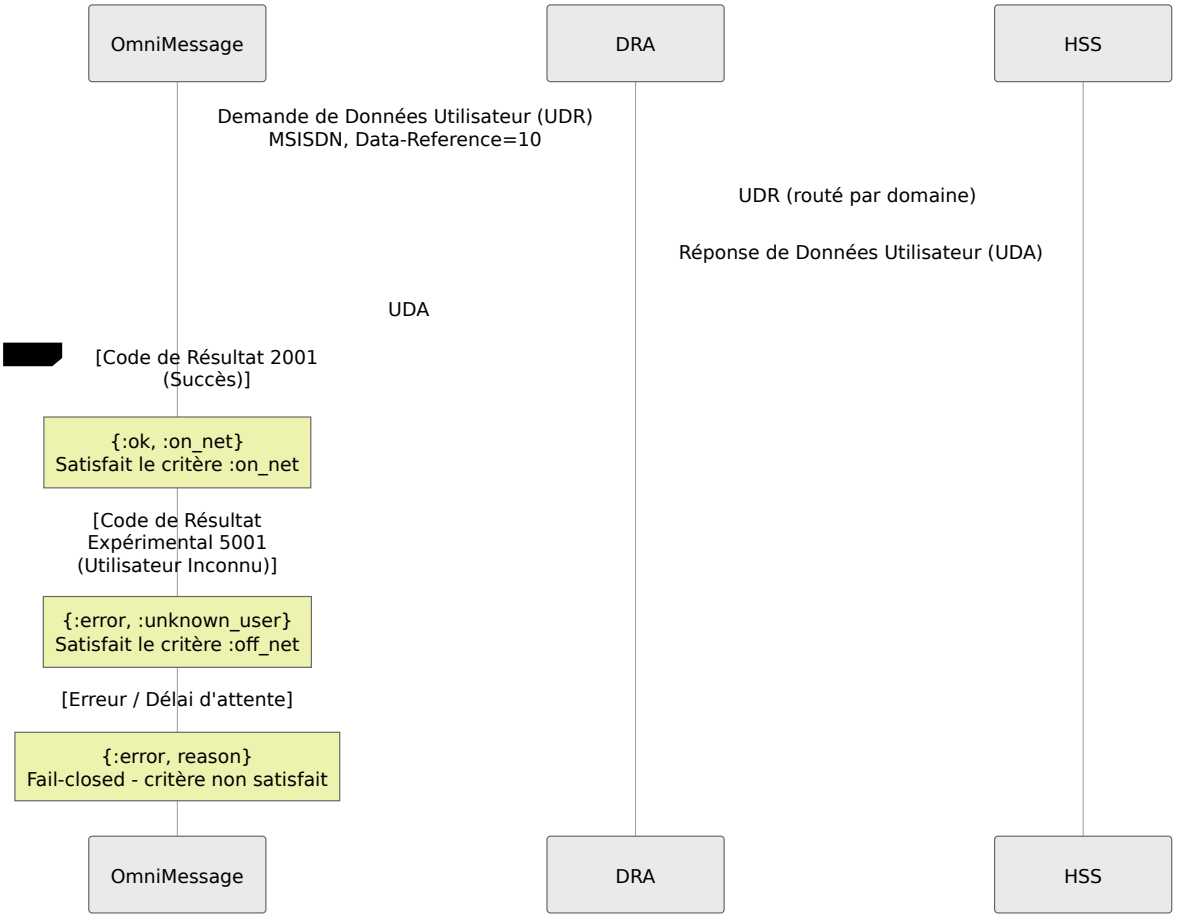
Si le MSISDN de destination est trouvé dans la table des enregistrements/localisations, le message est routé directement vers le frontend servant cet abonné. Aucune recherche HSS ou évaluation de la table de routage n'a lieu. (Voir Expiration de la localisation.)

Table de routage + Dip HSS paresseux

S'il n'y a pas d'enregistrement actif, le routeur évalue la table de routage. Pour chaque route qui déclare `originating_network`/`terminating_network` de `:on_net` ou `:off_net`, le routeur effectue un dip HSS (via le cache) dans le cadre de la correspondance de cette route. Une route ne correspond que si elle satisfait son critère de réseau ainsi que tous ses autres critères. Si Diameter est désactivé ou si le HSS ne donne pas de réponse définitive, le critère échoue (fail-closed) et la route ne correspond pas.

Retenir un message pour **stockage et transfert** est un concept distinct : l'*action* de la route `:saf` le déclenche, pas le dip HSS. Voir Livraison de stockage et transfert.

Flux de messages Diameter Sh



Paramètres UDR

AVP	Valeur	Description
Session-Id	Auto-généré	Identifiant de session unique
Auth-Session-State	1 (NO_STATE_MAINTAINED)	Aucun état de session requis pour les requêtes
Data-Reference	10	Demander toutes les données utilisateur
Origin-Host	Hôte configuré + domaine	Identité Diameter d'OmniMessage
Origin-Realm	Domaine configuré	Domaine Diameter d'OmniMessage
Destination-Realm	Domaine configuré	Domaine HSS (routé via DRA)
User-Identity	MSISDN encodé en TBCD	Le numéro de l'abonné
Server-Name	"OmniMessage"	Identité du serveur d'application
Vendor-Specific-Application-Id	Auth: 16777217, Vendor: 10415	Identifiant d'application Sh selon 3GPP TS 29.329

Codes de Résultat UDA

La vérification de l'existence de l'abonné mappe l'UDA à l'un des trois résultats :

Code	Type	Résultat	Signification pour la correspondance de route
2001	Code de Résultat	`{:ok, :on_net}`	Sur le réseau - satisfait `:on_net`, échoue `:off_net`
5001	Code de Résultat Expérimental	`{:error, :unknown_user}`	Hors réseau - satisfait `:off_net`, échoue `:on_net`
Autre / délai d'attente / pas de pairs	-	`{:error, reason}`	Fail-closed - ne satisfait aucun ; la route ne correspond pas

Configuration

Activation de la recherche HSS

Deux blocs de configuration dans `config/runtime.exs` contrôlent la recherche HSS :

1. Le drapeau `diameter_enabled` sous `:sms_c` - contrôle si des dips HSS sont effectués lors de la correspondance des routes. Lorsque `false`, toute route avec un critère `:on_net`/`:off_net` échoue à correspondre (fail-closed).
2. La configuration `:diameter_ex` - configure la pile Diameter elle-même (identité, pairs, applications)

Configuration minimale

```
# Activer les dips HSS lors de la correspondance des routes
config :sms_c,
  diameter_enabled: true

# Configuration de la pile Diameter
config :diameter_ex,
  diameter: %{
    service_name: :omnimessage,
    listen_ip: "0.0.0.0",
    listen_port: 3868,
    decode_format: :map,
    host: "smsc01",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    auth_application_ids: [],
    acct_application_ids: [],
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{
            vendor_id: 10415,
            auth_application_id: 16_777_217,
            acct_application_id: nil
          }
        ]
      }
    ],
    peers: [
      %{
        port: 3868,
        host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
```

```
        ip: "10.0.0.1",
        realm: "epc.mnc001.mcc001.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
```

Paramètres OmniMessage

Paramètre	Type	Requis	Par défaut	Description
`diameter_enabled`	Booléen	Non	`false`	Activer les dips HSS lors de la correspondance des routes. Lorsque `false`, le routes avec des critères `:on_net`/`:off_ne` échouent en mod fail-closed.
`hss_cache_ttl_seconds`	Entier	Non	`7200`	TTL (secondes) pour les résultats de recherche HSS mis en cache. Défini dans `runtime.exs` (no exposé dans le schéma YAML de l'opérateur).
`mock_sh`	Booléen	Non	`false`	Utiliser des réponses HSS fictives au lieu de vraies requêtes Diameter. Pour le tests uniquement
`mock_sh_on_net_numbers`	Liste	Non	`[]`	Liste de MSISDN traités comme su le réseau lorsque `mock_sh` est `true`

Paramètres de la pile Diameter

Paramètre	Type	Requis	Pa défa
`service_name`	Atome	Oui	-
`listen_ip`	Chaîne	Non	`"0.0.`
`listen_port`	Entier	Non	3868
`decode_format`	Atome	Oui	-
`host`	Chaîne	Oui	-
`realm`	Chaîne	Oui	-
`product_name`	Chaîne	Non	-
`request_timeout`	Entier	Non	5000
`peer_selection_algorithm`	Atome	Non	`:rand`

Paramètre	Type	Requis	Pa défa
`allow_undefined_peers_to_connect`	Booléen	Non	`true`
`log_unauthorized_peer_connection_attempts`	Booléen	Non	`true`
`control_module`	Module	Oui	-
`processor_module`	Module	Oui	-
`auth_application_ids`	Liste	Non	`[]`
`acct_application_ids`	Liste	Non	`[]`
`vendor_id`	Entier	Oui	-
`supported_vendor_ids`	Liste	Oui	-

Paramètres d'application

La liste `applications` configure quelles applications Diameter sont activées. Pour la recherche HSS, seule l'application Sh est requise.

Paramètre	Type	Requis	Par défaut	De
`application_name`	Atome	Oui	-	Identifiar l'applicat `:sh` pou HSS.
`application_dictionary`	Atome	Oui	-	Module d Diamete être `:diamet`
`vendor_specific_application_ids`	Liste	Oui	-	Identifiar spécifiqu fournisse dessous.

Identifiant d'application spécifique au fournisseur (Sh) :

Paramètre	Type	Valeur	Description
`vendor_id`	Entier	`10415`	Identifiant de fournisseur 3GPP
`auth_application_id`	Entier	`16_777_217`	Identifiant d'application Sh selon 3GPP TS 29.329
`acct_application_id`	-	`nil`	Non utilisé pour Sh

Paramètres des pairs

La liste `peers` définit les pairs Diameter (généralement DRAs ou le HSS directement) auxquels OmniMessage se connecte.

Paramètre	Type	Requis	Par défaut	Descripti
`host`	Chaîne	Oui	-	Identité d'Hô Diameter du (FQDN). Doit correspondre exactement l'identité configurée d pair.
`ip`	Chaîne	Oui	-	Adresse IP du pair pour la connexion TC
`port`	Entier	Non	3868	Port Diamete pair.
`realm`	Chaîne	Oui	-	Domaine Diameter du pair.
`tls`	Booléen	Non	`false`	Activer TLS p cette connex de pair.
`transport`	Atome	Non	`:diameter_tcp`	Protocole de transport : `:diameter_t` ou `:diameter_s`
`initiate_connection`	Booléen	Non	`true`	Si `true`, OmniMessag initie la connexion TC au pair. Si

Paramètre	Type	Requis	Par défaut	Descripti
				`false`, OmniMessag attend que le pair se conne

Multiples pairs

Configurez plusieurs pairs pour la redondance. OmniMessage sélectionne parmi les pairs connectés en utilisant l'`algorithme de sélection de pairs` :

```
peers: [
  %{
    port: 3868,
    host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.0.0.1",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  },
  %{
    port: 3868,
    host: "dra02.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.0.0.2",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  }
]
```

Mode fictif

Pour les environnements de test sans HSS en direct, activez le mode fictif :

```
config :sms_c,
  diameter_enabled: true,
  mock_sh: true,
  mock_sh_on_net_numbers: [
    "15555550001",
    "15555550002"
  ]
```

En mode fictif, tout MSISDN dans la liste `mock_sh_on_net_numbers` se résout en `{:ok, :on_net}` (équivalent au code de résultat 2001). Tous les autres MSISDN se résolvent en `{:error, :unknown_user}` (équivalent au code de résultat 5001). Aucune pile Diameter n'est démarrée et aucune connexion réseau n'est établie.

Cache de résultats HSS

Les recherches HSS sont mises en cache par `SmsC.Diameter.HssCache` par MSISDN pour éviter des allers-retours Diameter redondants :

- **TTL** : `hss_cache_ttl_seconds` (par défaut 7200 = 2 heures)
- **Partagé** : plusieurs routes nécessitant le même statut MSISDN dans une évaluation partagent un seul dip
- **Rafraîchissement forcé** : le cache est contourné (un nouveau dip est effectué) lorsqu'un message est explicitement réacheminé
- **Prédicats** : un résultat est traité comme sur le réseau uniquement pour `{:ok, :on_net}` et comme hors réseau uniquement pour `{:error, :unknown_user}` ; tout autre résultat (y compris `nil` lorsque Diameter est désactivé) est traité comme aucun - ce qui produit une correspondance fail-closed.

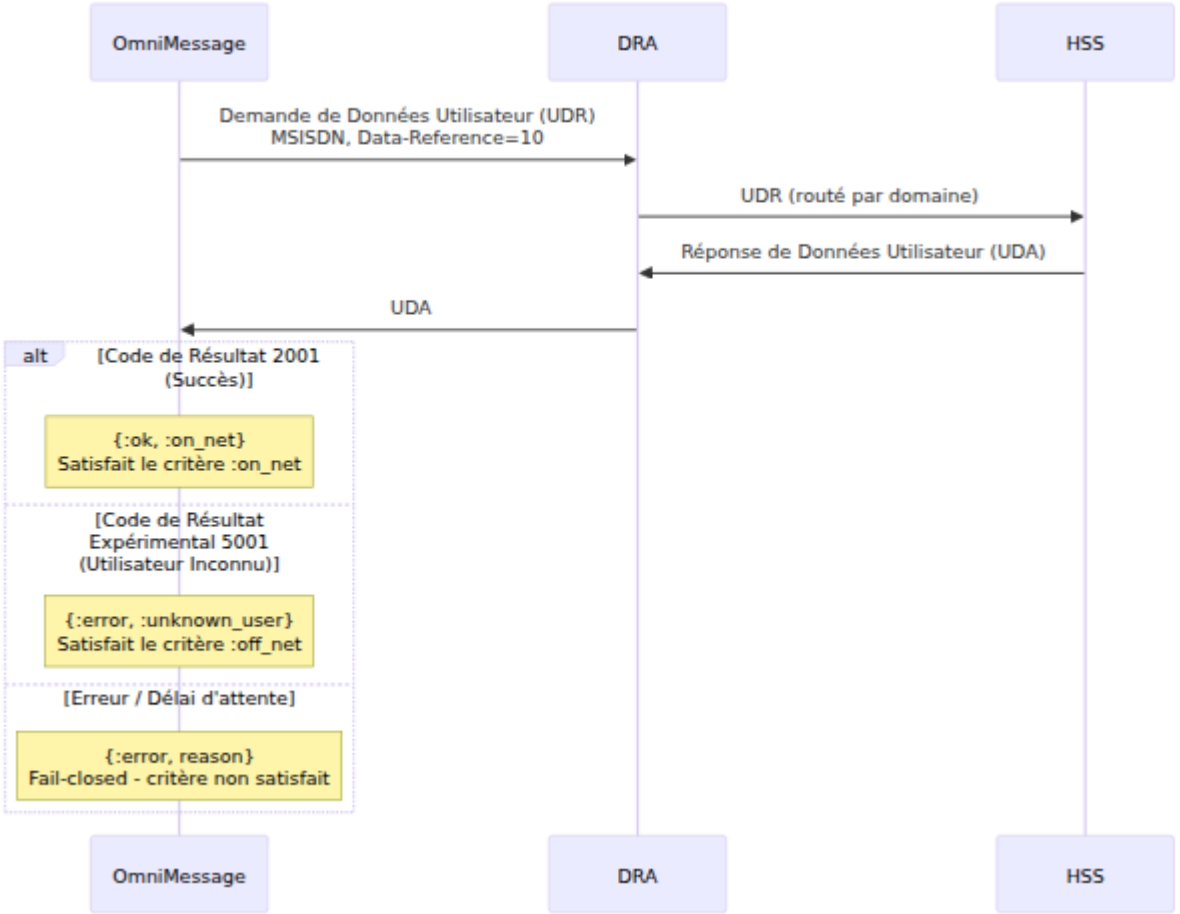
Livraison de stockage et transfert

L'*action* de route `:saf` entraîne le stockage et le transfert, pas le dip HSS. Une route avec `action: :saf` retient le message pour une livraison ultérieure. Pour retenir *uniquement* les abonnés sur le réseau, combinez `:saf` avec

`terminating_network: :on_net` et fournissez une route de secours hors réseau (voir Guide de routage SMS → `saf`).

Lorsqu'un message est retenu pour SAF, il reste dans la file d'attente de messages Mnesia avec `dest_smsc` non défini. Lorsque l'abonné s'enregistre et qu'une mise à jour de localisation arrive, la boucle de livraison récupère les messages en attente pour ce MSISDN et les achemine vers le frontend de service.

Les messages dans la file d'attente sont soumis à la politique de conservation standard. Si l'abonné ne s'enregistre pas avant l'expiration du message, le message est déplacé vers la file des lettres mortes selon le `dead_letter_time_minutes` configuré.



Notes opérationnelles

Journalisation au démarrage

OmniMessage journalise l'état Diameter Sh au démarrage :

- **Activé** : `Diameter Sh (HSS): ENABLED (hôte : smsc01, domaine : epc.mnc001.mcc001.3gppnetwork.org, pairs : 2)`
- **Désactivé** : `Diameter Sh (HSS): DISABLED`

Comportement Fail-Closed

Si une recherche HSS ne renvoie pas de réponse définitive - délai d'attente, pas de pairs connectés, réponse non gérée ou code de résultat inattendu - `subscriber_exists?/1` renvoie `{:error, reason}`. Pour une route nécessitant `:on_net` ou `:off_net`, ce critère **n'est pas** satisfait, donc la route **ne correspond pas** et l'évaluation se poursuit avec la route suivante.

C'est un design **fail-closed** : sans réponse définitive, OmniMessage ne considérera pas un numéro comme sur le réseau ou hors réseau, donc une route uniquement sur le réseau ne se déclenchera pas et une route de passerelle uniquement hors réseau ne se déclenchera pas non plus. Fournissez une route de secours générale (`terminating_network: :any`) si vous souhaitez que les messages soient routés pendant les pannes HSS.

Lorsque **Diameter est explicitement désactivé** (`diameter_enabled: false`), la recherche dans le cache renvoie `nil` et la même correspondance fail-closed s'applique à toute route avec un critère `:on_net`/`:off_net`. Les routes qui laissent les deux champs de réseau à `:any` ne sont pas affectées.

Observabilité

Les dips HSS sont observables à travers :

- **Journaux de route par message** - chaque événement `route_evaluated` de la route inclut la vérification du réseau, par exemple `term_network: ✓ (la terminaison nécessite on_net, +15551234 est sur le réseau)` ou

`✗ (... est hors réseau)`. Visible dans le journal d'activité des messages du panneau de contrôle et l'API des événements.

- **Télémétrie Diameter Sh** - `SmsC.Telemetry` émet des métriques de demande, de réponse (étiquetées par résultat), de durée et d'erreur (voir METRICS.md).
- **Journaux d'application** - lignes `HSS dip: <msisdn> est SUR LE RÉSEAU (2001)` / `n'est PAS dans HSS (5001)` de `SmsC.Diameter.Sh`.

Expiration de la localisation

Le routage basé sur la localisation vérifie l'horodatage `expires` sur les enregistrements d'abonnés. Si l'enregistrement d'un abonné a expiré, par exemple parce qu'il est passé en mode avion, le routeur considère la localisation comme inactive. L'évaluation se poursuit alors vers la table de routage, y compris tout dip HSS paresseux qu'une route correspondante exige.

Cela empêche l'envoi de messages vers un frontend qui ne peut plus atteindre l'abonné, évitant ainsi des échecs de livraison inutiles et des cycles de réessai.

Application des exigences sur le réseau / hors réseau au niveau de la route

Les routes appliquent les exigences sur le réseau/hors réseau à travers deux champs - **pas** des indicateurs booléens et **pas** une action de route :

Champ	Valeurs	Vérifications	Comportement
`originating_network`	`:any` | `:on_net` | `:off_net`	Numéro appelant	La route ne correspond que si le statut HSS du MSISDN source correspond (fail-closed)
`terminating_network`	`:any` | `:on_net` | `:off_net`	Numéro appelé	La route ne correspond que si le statut HSS du MSISDN de destination correspond (fail-closed)

La valeur par défaut pour les deux est `:any` (pas de vérification HSS). Une valeur de `:on_net` nécessite le code de résultat 2001 ; `:off_net` nécessite 5001 ; toute réponse indéfinie échoue le critère.

Exemple de configuration

```
config :sms_c, :sms_routes, [
  # Destinations sur le réseau uniquement - livrer au passerelle
locale
  %{
    calling_regex: nil,
    called_regex: ~r/^1555/,
    source_smsc: nil,
    dest_smsc: "local-gateway",
    source_type: nil,
    action: :route,
    charged: :default,
    weight: 100,
    priority: 10,
    description: "Passerelle locale - destinations sur le réseau
uniquement",
    originating_network: :any,
    terminating_network: :on_net,
    enabled: true
  },

  # Route de secours hors réseau - attraper tout ce que la route
sur le réseau n'a pas pris
  %{
    calling_regex: nil,
    called_regex: nil,
    source_smsc: nil,
    dest_smsc: "external-gateway",
    source_type: :ims,
    action: :route,
    charged: :default,
    weight: 100,
    priority: 255,
    description: "Passerelle externe - secours",
    originating_network: :any,
    terminating_network: :any,
    enabled: true
  }
]
```

Comment ça fonctionne : pour un message à un abonné non enregistré :

1. **Vérification de localisation** - aucun enregistrement actif trouvé
2. **Évaluation de la route** - les routes sont vérifiées par ordre de priorité
3. La route P10 "local-gateway" a `terminating_network: :on_net`, donc un dip HSS est effectué sur le numéro appelé. Si le HSS renvoie 2001 (sur le réseau), la route correspond et le message est livré à `local-gateway`. S'il renvoie 5001 (hors réseau) ou échoue (fail-closed), la route ne correspond pas.
4. La route P255 "external-gateway" a `terminating_network: :any`, donc elle n'a pas besoin de réponse HSS et correspond en tant que secours.

Cas d'utilisation : maintenir le trafic sur le réseau local tout en permettant au trafic hors réseau (ou incertain HSS) d'atteindre un opérateur externe. Pour *retenir* les abonnés sur le réseau pour un stockage et un transfert au lieu de les livrer à une passerelle, définissez l'`action` de la route P10 sur `:saf`.

Dépannage

Abonnés sur le Réseau Routés vers la Passerelle de Secours

Symptômes : Les messages destinés aux abonnés sur le réseau prennent la route hors réseau/de secours au lieu de la route sur le réseau.

Causes possibles :

- `diameter_enabled` est `false` - le critère `:on_net` échoue, donc la route sur le réseau ne correspond jamais
- Aucun pair Diameter n'est connecté (vérifiez les journaux d'application pour l'état de connexion des pairs)
- Le HSS renvoie des erreurs ou expire (vérifiez la télémétrie des erreurs Sh et les lignes de journal "HSS dip")
- La traduction de numéro produit un format de numéro différent de celui attendu par le HSS

Résolution :

1. Vérifiez `diameter_enabled: true` dans `runtime.exs`
2. Vérifiez les journaux pour l'état de connexion des pairs Diameter au démarrage (`Diameter Sh (HSS): ENABLED (... pairs : N)`)
3. Recherchez les lignes de journal `HSS dip:` et les métriques d'erreur Sh
4. Confirmez que le format de numéro après traduction correspond à ce que le HSS attend (E.164 sans `+`)

Messages Retenus Mais Jamais Livrés

Symptômes : Les messages sont correctement retenus (`:saf`) pour les abonnés sur le réseau mais ne sont jamais livrés, même après l'enregistrement de l'abonné.

Causes possibles :

- Les mises à jour de localisation n'atteignent pas OmniMessage
- Le MSISDN dans l'enregistrement de localisation ne correspond pas au `destination_msisdn` du message en file d'attente
- Les messages ont expiré et ont été déplacés vers la file d'attente des lettres mortes

Résolution :

1. Vérifiez que les mises à jour de localisation/enregistrement sont reçues (vérifiez la page des Enregistrements dans le Panneau de Contrôle)
2. Confirmez que le format MSISDN est cohérent entre la file d'attente des messages et la table d'enregistrement
3. Vérifiez `dead_letter_time_minutes` - augmentez si les abonnés peuvent être hors ligne pendant de longues périodes

Pair Diameter Ne Se Connecte Pas

Symptômes : Les journaux ne montrent aucune connexion de pair Diameter au démarrage.

Causes possibles :

- IP ou port de pair incorrect dans la configuration

- Pare-feu bloquant le port 3868
- Mismatch d'identité de l'hôte de pair (l'`hôte` configuré doit correspondre exactement à l'identité Diameter du pair)
- Mismatch de domaine entre OmniMessage et le pair

Résolution :

1. Vérifiez que l'adresse IP et le port du pair sont corrects
2. Confirmez que les règles de pare-feu permettent le trafic TCP sur le port configuré
3. Vérifiez que la valeur `host` du pair correspond exactement à ce que le pair annonce dans son échange de capacités
4. Assurez-vous que le `realm` correspond entre les configurations d'OmniMessage et du pair

Documentation des métriques Prometheus de SMS-C

← Retour à l'index de documentation

Aperçu

Ce document décrit toutes les métriques Prometheus exposées par le système SMS-C. Ces métriques sont conçues pour le personnel opérationnel afin de surveiller la santé du système, les performances et de résoudre les problèmes.

Accès aux métriques

Le point de terminaison des métriques Prometheus est disponible à l'adresse :

```
http://localhost:9568/metrics
```

Ce point de terminaison expose des métriques au format texte Prometheus que le serveur Prometheus peut récupérer. Le système met à jour les métriques en temps réel au fur et à mesure qu'il traite les messages.

Convention de nommage des métriques

Toutes les métriques suivent le modèle : `sms_c.<category>.<metric_name>.<type>`

Catégories :

- `license` - Métriques de statut de licence
- `message` - Métriques de traitement des messages
- `routing` - Métriques de décision de routage
- `enum` - Métriques de recherche ENUM/NAPTR
- `delivery` - Métriques de livraison des messages
- `queue` - Métriques de gestion de la file d'attente
- `charging` - Métriques de facturation
- `mnesia` - Métriques de base de données
- `frontend` - Métriques de connexion frontend
- `location` - Métriques de localisation/enregistrement
- `phoenix.endpoint` - Métriques de requêtes API HTTP
- `vm` - Métriques système de la VM Erlang

Métriques de licence

sms_c_license_status

Type : Gauge

Description : Statut actuel de la licence du système SMS-C OmniMessage.

Valeurs :

- `1` - Licence valide
- `0` - Licence invalide/expirée

Labels : Aucun

Nom du produit : `omnimessage`

Cas d'utilisation : Surveiller la validité de la licence pour s'assurer que le système fonctionne avec une licence valide. Lorsqu'elle est invalide, les messages sont toujours reçus mais routés vers la destination "NOLICENCE" au lieu du routage normal.

Comportement lorsque la licence est invalide :

- Les messages entrants sont **acceptés** et stockés
- La destination du message (`dest_smsc`) est **automatiquement définie sur "NOLICENCE"**
- Le routage normal est **contourné**
- L'interface utilisateur et la surveillance restent **accessibles**
- La base de données et tous les services restent **opérationnels**

Alerting :

```
- alert: SMS_C_License_Invalid
  expr: sms_c_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "Licence SMS-C invalide ou expirée"
    description: "Le statut de la licence est invalide - les
messages sont routés vers NOLICENCE"
```

Exemples de requêtes Prometheus :

```
# Vérifier si la licence est valide
sms_c_license_status == 1

# Alerte sur licence invalide
sms_c_license_status == 0

# Compter les messages routés vers NOLICENCE (indique un problème
de licence)
sms_c_routing_route_matched_count{dest_smsc="NOLICENCE"}
```

Métriques de traitement des messages

sms_c_message_received_count

Type : Counter

Description : Nombre total de messages reçus par le SMS-C de toutes les sources.

Labels :

- `source_smsc` : Nom du SMSC source qui a envoyé le message
- `source_type` : Type de connexion source (ims, circuit_switched, smpp)
- `message_type` : Type de message (sms, mms)

Cas d'utilisation : Surveiller le volume de messages entrants par source et type. Utilisé pour détecter les modèles de trafic, identifier les périodes de forte activité et repérer les anomalies dans le flux de messages.

Alerting : Définir des alertes pour des baisses soudaines (problèmes de connectivité potentiels) ou des pics (attaque/spam potentiel).

---

sms_c_message_validated_count

Type : Counter

Description : Nombre total de validations de messages effectuées.

Labels :

- `valid` : Si la validation a réussi (true ou false)

Cas d'utilisation : Suivre les taux de succès/échec de validation. Des taux d'échec élevés peuvent indiquer des messages mal formés ou des problèmes d'intégration.

Alerting : Alerter lorsque le taux d'échec de validation dépasse un seuil (par exemple, > 5 % d'échecs).

---

sms_c_message_processing_stop_duration

Type : Histogram

Description : Temps nécessaire pour traiter un message de la réception à l'achèvement (inclut validation, routage et mise en file d'attente).

Unité : Millisecondes

Seaux : 10, 50, 100, 250, 500, 1000, 2500, 5000 ms

Labels :

- `success` : Si le traitement a réussi (true ou false)

Cas d'utilisation : Surveiller les performances de traitement des messages de bout en bout. Identifier les ralentissements dans le pipeline de traitement.

Alerting : Alerter lorsque la latence p95 ou p99 dépasse les seuils SLA.

Métriques de routage

sms_c_routing_route_matched_count

Type : Counter

Description : Nombre total de fois qu'une route spécifique a été correspondue et sélectionnée pour le routage des messages.

Labels :

- `route_id` : Identifiant unique de la route correspondue
- `dest_smsc` : SMSC de destination sélectionné par la route

- `priority` : Valeur de priorité de la route correspondue

Cas d'utilisation : Comprendre quelles routes sont utilisées le plus fréquemment. Identifier les routes sous-utilisées ou surchargées. Utile pour la planification de capacité et l'optimisation des routes.

Alerting : Alerter si des routes de haute priorité sont rarement correspondues (peut indiquer une mauvaise configuration de routage).

---

sms_c_routing_failed_count

Type : Counter

Description : Nombre total d'échecs de routage où aucune route appropriée n'a pu être trouvée.

Labels :

- `reason` : Raison de l'échec (no_route_found, validation_failed, etc.)

Cas d'utilisation : Suivre les échecs de routage pour identifier les lacunes de configuration ou les modèles de trafic inattendus.

Alerting : Alerter sur tout échec de routage car cela indique que les messages ne peuvent pas être livrés.

---

sms_c_routing_action_count

Type : Counter

Description : Nombre total d'actions de routage spéciales effectuées.

Labels :

- `action` : Type d'action (drop, auto_reply, forward)
- `route_id` : Route qui a déclenché l'action

Cas d'utilisation : Surveiller les règles de suppression (anti-spam), l'utilisation des réponses automatiques et les modèles de transfert.

Alerting : Alerter sur des pics inattendus dans les actions de suppression (peut indiquer une attaque de spam).

---

sms_c_routing_stop_duration

Type : Histogram

Description : Temps nécessaire pour évaluer toutes les routes et sélectionner la meilleure correspondance.

Unité : Millisecondes

Seaux : 1, 5, 10, 25, 50, 100, 250, 500 ms

Labels :

- `dest_smsc` : SMSC de destination sélectionné

Cas d'utilisation : Surveiller les performances du moteur de routage. Un routage lent indique trop de routes ou une logique de correspondance complexe.

Alerting : Alerter lorsque le routage prend systématiquement plus de temps que prévu (par exemple, p95 > 50 ms).

Métriques de recherche ENUM/NAPTR

sms_c_enum_cache_hit_count

Type : Counter

Description : Nombre total de recherches ENUM servies à partir du cache (n'a pas nécessité de requête DNS).

Labels :

- `domain` : Domaine ENUM interrogé

Cas d'utilisation : Surveiller l'efficacité du cache. Des taux de réussite élevés réduisent la charge DNS et améliorent les performances.

Alerting : Alerter si le taux de réussite du cache tombe en dessous d'un seuil (peut indiquer des problèmes de cache ou un trafic inhabituel).

---

sms_c_enum_cache_miss_count

Type : Counter

Description : Nombre total de recherches ENUM qui ont nécessité une requête DNS (non dans le cache).

Labels :

- `domain` : Domaine ENUM interrogé

Cas d'utilisation : Suivre les échecs de cache pour comprendre l'efficacité du cache. Utiliser avec le compte de réussite pour calculer le taux de réussite.

Calcul : `cache_hit_rate = hits / (hits + misses)`

---

sms_c_enum_cache_size_size

Type : Gauge

Description : Nombre actuel d'entrées dans le cache ENUM.

Cas d'utilisation : Surveiller la taille du cache pour s'assurer qu'elle ne croît pas de manière illimitée. Aider à ajuster les paramètres TTL du cache.

Alerting : Alerter si la taille du cache dépasse les limites attendues (peut indiquer une fuite de mémoire).

---

sms_c_enum_lookup_stop_duration

Type : Histogram

Description : Temps nécessaire pour compléter une recherche ENUM (y compris la requête DNS si non mise en cache).

Unité : Millisecondes

Seaux : 10, 50, 100, 250, 500, 1000, 2500, 5000 ms

Labels :

- `domain` : Domaine ENUM interrogé
- `success` : Si la recherche a réussi (true ou false)
- `cache_hit` : Si le résultat a été servi à partir du cache (true ou false)

Cas d'utilisation : Surveiller les performances des recherches ENUM. Identifier les serveurs DNS lents ou les problèmes de réseau.

Alerting : Alerter lorsque le temps de recherche p95 dépasse le seuil de délai d'attente.

---

sms_c_enum_naptr_records_record_count

Type : Histogram

Description : Nombre d'enregistrements NAPTR retournés par une recherche ENUM réussie.

Seaux : 0, 1, 2, 3, 5, 10

Labels :

- `domain` : Domaine ENUM interrogé

Cas d'utilisation : Comprendre la distribution des enregistrements ENUM. La plupart des recherches devraient retourner 1 à 3 enregistrements.

Alerting : Alerter si 0 enregistrements sont fréquemment retournés (problème de configuration DNS).

Métriques de livraison

sms_c_delivery_queued_count

Type : Counter

Description : Nombre total de messages mis en file d'attente pour livraison à un SMSC de destination.

Labels :

- `dest_smsc` : Nom du SMSC de destination

Cas d'utilisation : Surveiller le flux de messages vers chaque destination. Utile pour la planification de capacité.

Alerting : Comparer avec les comptes de succès/échec de livraison pour détecter l'accumulation.

---

sms_c_delivery_attempted_count

Type : Counter

Description : Nombre total de tentatives de livraison effectuées (inclut les réessais).

Labels :

- `dest_smsc` : Nom du SMSC de destination

Cas d'utilisation : Suivre le volume des tentatives de livraison. Un nombre élevé de tentatives par rapport au nombre mis en file d'attente indique un comportement de réessai.

---

sms_c_delivery_succeeded_count

Type : Counter

Description : Nombre total de messages livrés avec succès au SMSC de destination.

Labels :

- `dest_smsc` : Nom du SMSC de destination

Cas d'utilisation : Suivre les livraisons réussies par destination. Principal indicateur de succès.

Alerting : Alerter si le taux de succès tombe en dessous du seuil SLA.

Calcul : `success_rate = succeeded / queued`

---

sms_c_delivery_failed_count

Type : Counter

Description : Nombre total de messages qui ont échoué à la livraison après toutes les tentatives de réessai.

Labels :

- `dest_smsc` : Nom du SMSC de destination
- `reason` : Raison de l'échec

Cas d'utilisation : Suivre les échecs de livraison pour identifier les destinations problématiques ou les modèles d'échec.

Alerting : Alerter sur des taux d'échec élevés ou des raisons d'échec spécifiques.

---

sms_c_delivery_dead_letter_count

Type : Counter

Description : Nombre total de messages déplacés vers la file de lettres mortes (non livrables).

Labels :

- `reason` : Raison de la lettre morte (par exemple, `max_retries_exceeded`, `expired`)

Cas d'utilisation : Surveiller les messages non livrables nécessitant une intervention manuelle.

Alerting : Alerter sur tout événement de lettre morte car cela représente un échec complet de livraison.

---

sms_c_delivery_succeeded_attempt_count

Type : Histogram

Description : Nombre de tentatives de livraison nécessaires avant une livraison réussie.

Seaux : 1, 2, 3, 5, 10

Labels :

- `dest_smsc` : Nom du SMSC de destination

Cas d'utilisation : Comprendre le comportement de réessai. La plupart des livraisons devraient réussir dès la première tentative.

Alerting : Alerter si le nombre moyen de tentatives dépasse 2 (indique des problèmes de fiabilité de destination).

---

sms_c_delivery_failed_attempt_count

Type : Histogram

Description : Nombre de tentatives de livraison effectuées avant l'échec final.

Seaux : 1, 2, 3, 5, 10

Labels :

- `dest_smsc` : Nom du SMSC de destination

Cas d'utilisation : Comprendre combien de réessais se produisent avant d'abandonner.

---

sms_c_delivery_time_delta_delta_ms

Type : Histogram

Description : Temps écoulé entre la soumission du message et la confirmation de livraison. Cette métrique capture la latence de bout en bout complète depuis qu'un message entre dans le SMS-C jusqu'à ce que le système confirme la livraison. Elle fournit des labels détaillés pour l'analyse par source, destination et comportement de réessai.

Unité : Millisecondes

Seaux : 50, 100, 250, 500, 1000, 2000, 5000, 10000, 20000, 60000, 600000 ms (50 ms à 10 minutes)

Labels :

- `source_smsc` : SMSC source qui a soumis le message
- `dest_smsc` : SMSC de destination qui a livré le message

- `delivery_attempts` : Nombre de tentatives de livraison nécessaires (0 = succès à la première tentative)

Cas d'utilisation : Analyser les performances de livraison de bout en bout par paire source-destination. Identifier les chemins lents, les combinaisons de SMSC problématiques et corréler la latence avec le comportement de réessai. Essentiel pour la surveillance SLA et la planification de capacité.

Alerting :

- Alerter lorsque p95 dépasse le seuil SLA pour des combinaisons source/dest spécifiques
- Alerter lorsque les messages nécessitent systématiquement plusieurs tentatives

Exemples de requêtes Prometheus :

```
# Temps de livraison p95 par SMSC source et destination
histogram_quantile(0.95,
  sum by (source_smsc, dest_smsc, le) (
    rate(sms_c_delivery_time_delta_delta_ms_bucket[5m])
  )
)

# Temps de livraison p99 global
histogram_quantile(0.99,
  sum by (le) (
    rate(sms_c_delivery_time_delta_delta_ms_bucket[5m])
  )
)

# Temps de livraison moyen par SMSC source
sum by (source_smsc) (rate(sms_c_delivery_time_delta_delta_ms_sum[5m]
/
sum by (source_smsc)
(rate(sms_c_delivery_time_delta_delta_ms_count[5m]))

# Temps de livraison pour les messages nécessitant des réessais vs
succès à la première tentative
histogram_quantile(0.95,
  sum by (le)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{delivery_attempts="0"
[5m]))
)
histogram_quantile(0.95,
  sum by (le)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{delivery_attempts!="0
[5m]))
)

# Paires source-destination les plus lentes (par p95)
topk(10,
  histogram_quantile(0.95,
    sum by (source_smsc, dest_smsc, le) (
      rate(sms_c_delivery_time_delta_delta_ms_bucket[1h])
    )
  )
)

# Pourcentage de messages livrés en moins de 2 secondes
```

```
sum(rate(sms_c_delivery_time_delta_delta_ms_bucket{le="2000"}[5m])) /
sum(rate(sms_c_delivery_time_delta_delta_ms_count[5m])) * 100

# Messages livrés en moins de 2 secondes par destination
sum by (dest_smsc)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{le="2000"}[5m])) /
sum by (dest_smsc) (rate(sms_c_delivery_time_delta_delta_ms_count[5m])
* 100
```

---

sms_c_delivery_time_by_smsc_delta_ms

Type : Histogram

Description : Temps écoulé entre la soumission du message et la confirmation de livraison, étiqueté uniquement par SMSC source et destination. C'est un compagnon de faible cardinalité à `sms_c_delivery_time_delta_delta_ms` - il omet le label `delivery_attempts` afin que le temps de livraison moyen par paire source/destination puisse être lu directement sans agrégation à travers les comptes de tentative. Émis à partir du même événement `[:sms_c, :delivery, :time_delta]`.

Unité : Millisecondes

Seaux : 50, 100, 250, 500, 1000, 2000, 5000, 10000, 20000, 60000, 600000 ms (50 ms à 10 minutes)

Labels :

- `source_smsc` : SMSC source qui a soumis le message
- `dest_smsc` : SMSC de destination qui a livré le message

Cas d'utilisation : Tableau de bord du temps de livraison moyen de bout en bout par paire source/destination. Préférer cette métrique à `sms_c_delivery_time_delta_delta_ms` pour les panneaux moyens/SLA où la répartition des comptes de tentative n'est pas nécessaire ; utiliser la variante `_delta` lorsque vous devez découper par comportement de réessai.

Exemples de requêtes Prometheus :

```
# Temps de livraison moyen (ms) par SMSC source/destination
sum by (source_smsc, dest_smsc)
(rate(sms_c_delivery_time_by_smsc_delta_ms_sum[5m])) /
sum by (source_smsc, dest_smsc)
(rate(sms_c_delivery_time_by_smsc_delta_ms_count[5m]))

# Temps de livraison moyen global
sum(rate(sms_c_delivery_time_by_smsc_delta_ms_sum[5m])) /
sum(rate(sms_c_delivery_time_by_smsc_delta_ms_count[5m]))

# Temps de livraison p95 par SMSC source/destination
histogram_quantile(0.95,
  sum by (source_smsc, dest_smsc, le) (
    rate(sms_c_delivery_time_by_smsc_delta_ms_bucket[5m])
  )
)
```

Métriques de file d'attente

sms_c_queue_size_size

Type : Gauge

Description : Nombre total actuel de messages dans la file d'attente (tous états combinés).

Labels :

- `queue_type` : Type de file d'attente (message_queue, dead_letter)

Cas d'utilisation : Surveiller la profondeur de la file d'attente pour détecter les arriérés ou les problèmes de traitement.

Alerting : Alerter lorsque la taille de la file d'attente dépasse les seuils de capacité.

---

sms_c_queue_size_pending

Type : Gauge

Description : Nombre actuel de messages en attente de livraison (pas encore tentés).

Labels :

- `queue_type` : Type de file d'attente

Cas d'utilisation : Surveiller le nombre de messages en attente. Des comptes en attente élevés indiquent des retards de traitement.

Alerting : Alerter lorsque le compte en attente dépasse le seuil pendant une période prolongée.

---

sms_c_queue_size_failed

Type : Gauge

Description : Nombre actuel de messages dans un état d'échec (en attente de réessai).

Labels :

- `queue_type` : Type de file d'attente

Cas d'utilisation : Surveiller l'accumulation de messages échoués. Indique des problèmes de livraison.

Alerting : Alerter sur un compte échoué élevé car cela impacte les taux de livraison.

---

sms_c_queue_size_delivered

Type : Gauge

Description : Nombre actuel de messages livrés en attente de nettoyage/retirement de la file d'attente.

Labels :

- `queue_type` : Type de file d'attente

Cas d'utilisation : Surveiller le retard de nettoyage. Des comptes élevés indiquent que le processus de nettoyage est à la traîne.

Alerting : Alerter si les messages livrés s'accumulent de manière significative.

---

sms_c_queue_oldest_message_age_seconds

Type : Gauge

Description : Âge (en secondes) du message le plus ancien actuellement en état d'attente.

Labels :

- `queue_type` : Type de file d'attente

Cas d'utilisation : Détecter le vieillissement des messages et les arrêts de traitement. Critique pour la surveillance SLA.

Alerting : Alerter lorsque l'âge du message le plus ancien dépasse le seuil SLA (par exemple, > 300 secondes).

Métriques de facturation

La facturation est effectuée via l'interface **Diameter Ro** (Contrôle de crédit). Le label `account` porte le MSISDN de l'abonné facturé (l'expéditeur du message). Un événement `charging_failed` est émis à la fois pour un refus définitif hors équilibre (`reason: :no_credit`) et pour une erreur OCS (transport/protocole). Seul un refus `:no_credit` rejette un message (statut

`:balance_rejected`) ; les erreurs OCS échouent ouvertement et le message est livré.

sms_c_charging_requested_count

Type : Counter

Description : Nombre total de demandes de facturation (Requêtes d'événements Diameter Ro CCR) envoyées à l'OCS.

Labels :

- `account` : MSISDN de l'abonné facturé (expéditeur du message)

Cas d'utilisation : Suivre le volume de facturation par compte. Utile pour la réconciliation de facturation.

---

sms_c_charging_succeeded_count

Type : Counter

Description : Nombre total d'opérations de facturation réussies.

Labels :

- `account` : Identifiant du compte facturé

Cas d'utilisation : Surveiller le taux de succès de la facturation par compte.

Calcul : `success_rate = succeeded / requested`

---

sms_c_charging_failed_count

Type : Counter

Description : Nombre total d'opérations de facturation échouées.

Labels :

- `account` : Identifiant du compte
- `reason` : Raison de l'échec

Cas d'utilisation : Identifier les échecs de facturation qui peuvent impacter les revenus ou nécessiter une intervention sur le compte.

Alerting : Alerter sur des taux d'échec de facturation élevés.

---

sms_c_charging_succeeded_duration

Type : Histogram

Description : Temps nécessaire pour compléter une demande de facturation réussie.

Unité : Millisecondes

Seaux : 10, 50, 100, 250, 500, 1000, 2500, 5000 ms

Labels :

- `account` : Identifiant du compte

Cas d'utilisation : Surveiller les performances du système de facturation. Une facturation lente peut retarder la livraison des messages.

Alerting : Alerter lorsque le temps de facturation p95 dépasse le seuil.

Métriques de santé du système

sms_c_mnesia_table_size_record_count

Type : Gauge

Description : Nombre actuel d'enregistrements dans chaque table de base de données Mnesia. Interrogé toutes les 10 secondes.

Labels :

- `table` : Nom de la table

Tables suivies :

- `sms_route` - Règles de routage SMS
- `message_store` - File d'attente de messages (messages en attente, livrés, échoués)
- `location_store` - Données de localisation/enregistrement des abonnés
- `frontend_store` - Enregistrements frontend/SMSC
- `translation_rule` - Règles de traduction de numéros
- `cell_tower_store` - Données de localisation des tours cellulaires
- `message_events` - Journaux d'événements de messages

Cas d'utilisation : Surveiller la croissance de la base de données. Détecter une accumulation de données inattendue. Planification de capacité.

Alerting : Alerter sur des taux de croissance de table inattendus ou lorsque les limites de capacité sont atteintes.

Exemples de requêtes Prometheus :

```
# Toutes les tailles de table
sms_c_mnesia_table_size_record_count

# Taille de la file d'attente de messages
sms_c_mnesia_table_size_record_count{table="message_store"}

# Enregistrements d'abonnés actifs
sms_c_mnesia_table_size_record_count{table="location_store"}

# Nombre de règles de routage
sms_c_mnesia_table_size_record_count{table="sms_route"}
```

---

sms_c_frontend_status_count

Type : Gauge

Description : Nombre de frontends dans chaque statut de connexion.

Labels :

- `frontend_name` : Identifiant du frontend
- `status` : Statut de connexion (connecté, déconnecté)

Cas d'utilisation : Surveiller la connectivité des frontends. Détecter les échecs de connexion.

Alerting : Alerter lorsque des frontends attendus se déconnectent.

---

sms_c_location_registered_count

Type : Counter

Description : Nombre total d'enregistrements de localisation/abonnés reçus par le système.

Labels :

- `location` : Nom du frontend/SMSC où l'abonné est enregistré
- `ims_capable` : Si l'abonné prend en charge l'IMS (true/false)

Cas d'utilisation : Surveiller l'activité d'enregistrement des abonnés. Suivre les abonnés IMS vs non-IMS. Détecter les tempêtes d'enregistrement ou les échecs.

Alerting : Définir des alertes pour :

- Baisse des taux d'enregistrement (peut indiquer des problèmes de réseau)
- Pics inhabituels dans les enregistrements
- Ratio élevé d'enregistrements non-IMS (afflux d'appareils anciens)

Exemple de requête :

```
# Taux d'enregistrement par minute
rate(sms_c_location_registered_count[1m])

# Ratio d'enregistrement IMS vs non-IMS
sum(rate(sms_c_location_registered_count{ims_capable="true"}[5m]))
/
sum(rate(sms_c_location_registered_count[5m]))
```

---

sms_c_location_active_registrations_count

Type : Gauge

Description : Nombre actuel d'enregistrements d'abonnés actifs, regroupés par localisation (frontend/SMSC) et capacité IMS. Cette métrique est interrogée toutes les 10 secondes et reflète l'état actuel du magasin de localisation.

Labels :

- `location` : Nom du frontend/SMSC où les abonnés sont enregistrés
- `ims_capable` : Si les enregistrements sont capables d'IMS (true/false)

Cas d'utilisation : Surveiller la distribution actuelle des abonnés à travers les frontends. Suivre l'adoption de l'IMS. Identifier les déséquilibres de charge entre les frontends. Planification de capacité pour chaque instance SMSC.

Alerting : Définir des alertes pour :

- Baisse soudaine des enregistrements pour une localisation (peut indiquer des problèmes de frontend)
- Distribution déséquilibrée entre les frontends
- Total des enregistrements approchant les limites de capacité

Exemples de requêtes Prometheus :

```
# Total des enregistrements actifs
sum(sms_c_location_active_registrations_count)

# Enregistrements actifs par localisation
sum by (location) (sms_c_location_active_registrations_count)

# Enregistrements actifs capables d'IMS par localisation
sum by (location)
(sms_c_location_active_registrations_count{ims_capable="true"})

# Taux d'adoption IMS par localisation
sum by (location)
(sms_c_location_active_registrations_count{ims_capable="true"}) /
sum by (location) (sms_c_location_active_registrations_count) *
100

# Total des enregistrements IMS vs non-IMS
sum by (ims_capable) (sms_c_location_active_registrations_count)

# Localisations avec le plus d'enregistrements
topk(10, sum by (location)
(sms_c_location_active_registrations_count))

# Distribution des enregistrements en pourcentage du total
sum by (location) (sms_c_location_active_registrations_count) /
sum(sms_c_location_active_registrations_count) * 100
```

Métriques de requêtes API HTTP

phoenix_endpoint_stop_duration

Type : Distribution (Histogram)

Description : Durée de traitement des requêtes HTTP en millisecondes, du début de la requête à l'achèvement de la réponse.

Labels :

- `method` : Méthode HTTP (par exemple, `GET`, `POST`)
- `path` : chemin de la requête avec des segments de haute cardinalité (UUID, IDs numériques) réduits à `/:id` (par exemple, `/api/messages`, `/api/messages/:id`)

Seaux : 10ms, 50ms, 100ms, 250ms, 500ms, 1s, 2.5s, 5s

Cas d'utilisation : Surveiller les performances de l'API. Identifier les points de terminaison lents. Suivre les SLA de temps de réponse.

Alerting : Définir des alertes pour :

- Latence P95 > 500ms pour les points de terminaison critiques
- Latence P99 > 1s pour tout point de terminaison
- Tendances de latence croissantes

Exemple de requête :

```
# Temps de réponse P95 par point de terminaison
histogram_quantile(0.95,
  rate(phoenix_endpoint_stop_duration_bucket[5m]))

# Requêtes plus lentes qu'une seconde
sum(rate(phoenix_endpoint_stop_duration_bucket{le="1000"}[5m]))
```

---

phoenix_endpoint_stop_count

Type : Counter

Description : Nombre total de requêtes HTTP terminées, catégorisées par méthode, chemin et code d'état HTTP.

Labels :

- `method` : Méthode HTTP (par exemple, `GET`, `POST`)
- `path` : chemin de la requête (segments de haute cardinalité réduits à `/:id`)

- `status` : code d'état HTTP (200, 201, 400, 404, 500, etc.)

Cas d'utilisation : Surveiller le volume des requêtes API et les taux de succès. Suivre les taux d'erreur par point de terminaison.

Alerting : Définir des alertes pour :

- Taux d'erreur > 5 % pour tout point de terminaison
- Erreurs 5xx sur des points de terminaison critiques
- Baisses soudaines du volume de requêtes

Exemple de requête :

```
# Taux de requêtes par chemin
sum by (path) (rate(phoenix_endpoint_stop_count[5m]))

# Taux d'erreur par chemin
sum by (path) (rate(phoenix_endpoint_stop_count{status=~"5.."}
[5m])) /
sum by (path) (rate(phoenix_endpoint_stop_count[5m]))

# Taux de succès
sum(rate(phoenix_endpoint_stop_count{status=~"2.."}[5m])) /
sum(rate(phoenix_endpoint_stop_count[5m]))
```

---

phoenix_router_dispatch_exception_count

Type : Counter

Description : Nombre total d'exceptions/erreurs levées lors du traitement des requêtes HTTP.

Labels :

- `route` : Route de point de terminaison API où l'exception s'est produite
- `kind` : Type d'exception (error, exit, throw)

Cas d'utilisation : Suivre les erreurs d'application. Identifier les points de terminaison problématiques. Surveiller la stabilité du système.

Alerting : Définir des alertes pour toute valeur non nulle sur des points de terminaison critiques.

Exemple de requête :

```
# Taux d'exception par point de terminaison
rate(phoenix_router_dispatch_exception_count[5m])

# Total des exceptions dans la dernière heure
increase(phoenix_router_dispatch_exception_count[1h])
```

Métriques Diameter Sh

Le SMS-C interroge le HSS via l'interface **Diameter Sh** (3GPP TS 29.329) pour classer les abonnés comme étant sur le réseau ou hors réseau lors de l'évaluation des routes. Ces métriques ne sont émises que lorsque Diameter est activé.

sms_c_diameter_peer_status

Type : Gauge

Description : Si le pair Diameter est connecté (`1`) ou non (`0`).

Labels :

- `origin_host` : hôte d'origine Diameter configuré du pair
- `ip` : adresse IP du pair

Exemple de requête :

```
# Pairs actuellement hors ligne
sms_c_diameter_peer_status == 0
```

---

sms_c_diameter_sh_request_count

Type : Counter

Description : Nombre total de requêtes de recherche Diameter Sh (UDR) envoyées.

---

sms_c_diameter_sh_response_count

Type : Counter

Description : Nombre total de réponses de recherche Diameter Sh (UDA), par résultat.

Labels :

- `result` : résultat de la recherche (par exemple, `on_net`, `unknown_user`, `error`)
- `result_code` : Résultat expérimental Diameter / Code de résultat (par exemple, `2001`, `5001`)

Exemple de requête :

```
# Ratio d'erreur de recherche Sh
sum(rate(sms_c_diameter_sh_response_count{result="error"}[5m]))
  / sum(rate(sms_c_diameter_sh_response_count[5m]))
```

---

sms_c_diameter_sh_response_duration_ms

Type : Distribution (Histogram)

Description : Durée de recherche Diameter Sh en millisecondes.

Labels :

- `result` : résultat de la recherche

Seaux : 5, 10, 25, 50, 100, 250, 500, 1000, 2500 ms

Exemple de requête :

```
# Latence de recherche Sh P95
histogram_quantile(0.95,
rate(sms_c_diameter_sh_response_duration_ms_bucket[5m]))
```

---

sms_c_diameter_sh_error_count

Type : Counter

Description : Nombre total d'erreurs de recherche Diameter Sh (échecs de transport/timeout/décodage, distincts d'un UDA négatif).

Labels :

- `reason` : raison de l'erreur

Métriques de fédération

La fédération transfère des messages entre les contrôleurs SMS-C via un modèle de pair basé sur le tirage. Voir Fédération.

sms_c_federation_forward_attempted_count

Type : Counter

Description : Nombre total de messages transférés vers des contrôleurs distants.

Labels :

- `dest_controller` : identité du contrôleur de destination

sms_c_federation_forward_completed_count

Type : Counter

Description : Nombre total de messages livrés avec succès à des contrôleurs distants.

Labels :

- `dest_controller` : identité du contrôleur de destination

---

sms_c_federation_forward_queued_count

Type : Counter

Description : Nombre total de messages mis en file d'attente parce qu'un pair était injoignable.

Labels :

- `dest_controller` : identité du contrôleur de destination

---

sms_c_federation_forward_failed_count

Type : Counter

Description : Nombre total de messages ayant dépassé le nombre maximum de réessais.

Labels :

- `dest_controller` : identité du contrôleur de destination

---

sms_c_federation_message_received_count

Type : Counter

Description : Nombre total de messages reçus de contrôleurs distants.

Labels :

- `origin_node` : nœud de contrôleur d'origine

---

sms_c_federation_health_check_count

Type : Counter

Description : Nombre total de tentatives de vérification de santé des pairs, par pair et résultat.

Labels :

- `peer` : identité du pair
- `result` : résultat de la vérification de santé (par exemple, `healthy`, `unhealthy`)

---

sms_c_federation_peer_status_count

Type : Gauge

Description : Nombre actuel de pairs de fédération par statut.

Labels :

- `status` : statut du pair (par exemple, `healthy`, `degraded`, `unhealthy`)

---

sms_c_federation_queue_depth_count

Type : Gauge

Description : Profondeur actuelle de la file d'attente de transfert de fédération.

Exemple de requête :

```
# Alerter si la file d'attente de transfert est en cours de
remplissage
sms_c_federation_queue_depth_count > 1000
```

Métriques de la VM Erlang

vm_memory_total

Type : Gauge

Description : Mémoire totale allouée par la VM Erlang en octets.

Cas d'utilisation : Surveiller l'utilisation globale de la mémoire. Détecter les fuites de mémoire. Planifier la capacité.

Alerting : Alerter lorsque l'utilisation de la mémoire > 80 % de la mémoire système disponible.

---

vm_memory_processes

Type : Gauge

Description : Mémoire utilisée par les processus Erlang en octets.

Cas d'utilisation : Suivre la consommation de mémoire des processus. Source la plus courante de croissance de la mémoire.

Alerting : Alerter sur un taux de croissance soutenu élevé.

---

vm_total_run_queue_lengths_total

Type : Gauge

Description : Nombre total de processus en attente d'être planifiés sur tous les planificateurs CPU.

Cas d'utilisation : Mesurer la charge système. Des valeurs élevées indiquent une saturation du CPU.

Alerting : Alerter lorsque cela dépasse systématiquement 10 * nombre de cœurs CPU.

---

vm_system_counts_process_count

Type : Gauge

Description : Nombre actuel de processus en cours d'exécution dans la VM.

Cas d'utilisation : Surveiller les modèles de création de processus. Détecter les fuites de processus.

Alerting : Alerter lorsque l'on approche de la limite de processus (par défaut 262 144).

Collecte et sondage des métriques

Le système collecte automatiquement les métriques suivantes toutes les 10 secondes :

- Tailles et âges de la file d'attente
- Tailles des tables Mnesia
- Statistiques de cache ENUM

Toutes les autres métriques sont déclenchées par des événements et émises lorsque l'action correspondante se produit.

Modèles de surveillance courants

Taux de succès de livraison par destination

Suivre le taux de succès de la livraison des messages pour chaque SMSC de destination :

Formule : `(sms_c_delivery_succeeded_count) / (sms_c_delivery_queued_count)`

Interprétation : Devrait être > 95 % pour des destinations saines. Des taux plus bas indiquent des problèmes de livraison.

Latence de message de bout en bout

Surveiller le temps total depuis la réception du message jusqu'à la livraison :

Métriques :

- `sms_c_message_processing_stop_duration` (traitement)
- `sms_c_delivery_time_delta_delta_ms` (livraison)

Interprétation : La somme représente la latence totale perçue par l'utilisateur.

Efficacité du cache ENUM

Mesurer l'efficacité du cache ENUM :

Formule : `(sms_c_enum_cache_hit_count) / (sms_c_enum_cache_hit_count + sms_c_enum_cache_miss_count)`

Interprétation : Devrait être > 80 % après la montée en température. Des taux plus bas peuvent indiquer un TTL court ou une variance de trafic élevée.

Utilisation des routes

Identifier quelles routes gèrent le plus de trafic :

Métrique : `sms_c_routing_route_matched_count` groupée par `route_id`

Interprétation : Utiliser pour identifier les routes chaudes pour l'optimisation et la planification de capacité.

Tendance d'arriéré de file d'attente

Surveiller si la file d'attente de messages est en croissance (arriéré) ou en diminution (rattrapage) :

Métriques :

- `sms_c_queue_size_pending` (en attente actuelle)
- `sms_c_queue_oldest_message_age_seconds` (âge en tendance)

Interprétation : Compte en attente croissant + âge croissant = formation d'un arriéré.

Taux de réessai

Comprendre à quelle fréquence des réessais de livraison sont nécessaires :

Métrique : `sms_c_delivery_succeeded_attempt_count` histogramme des percentiles

Interprétation : Si p95 > 1, la plupart des messages nécessitent des réessais. Indique des problèmes de fiabilité de destination.

Alertes recommandées

Alerte	Condition	Gravi
Taux d'échec de routage élevé	Augmentation de `routing_failed_count`	Critique
Arriéré de file d'attente	`queue_size_pending` > seuil	Avertisse
Anciens messages dans la file d'attente	`queue_oldest_message_age_seconds` > 300	Critique
Pic d'échecs de livraison	Pic de `delivery_failed_count`	Élevé
Événements de lettre morte	`delivery_dead_letter_count` > 0	Élevé
Délais d'attente de recherche ENUM	`enum_lookup_stop_duration` p95 > 5000ms	Avertisse
Faible taux de réussite du cache	Taux de réussite du cache ENUM < 0,7	Avertisse
Frontend déconnecté	`frontend_status_count{status="disconnected"}` > 0	Élevé

Alerte	Condition	Gravi
Échecs de facturation	`charging_failed_count` > seuil	Élevé
Traitement de message lent	`message_processing_stop_duration` p95 > 1000ms	Avertisse

Recommandations de tableau de bord

Tableau de bord des opérations

Objectif : Surveillance de la santé du système en temps réel

Panneaux :

1. Débit de messages (reçus/traités/livrés par minute)
2. Tailles de la file d'attente (en attente, échoués, livrés)
3. Taux de succès de livraison par destination
4. Latence de traitement et de livraison p95
5. Statut des frontends actifs
6. Alertes actuelles

Tableau de bord de performance

Objectif : Analyse des performances du système

Panneaux :

1. Histogramme de la durée de traitement des messages
2. Histogramme de la durée de routage

3. Histogramme de la durée de recherche ENUM
4. Histogramme de la durée de facturation
5. Distribution des tentatives de livraison
6. Taux de réussite du cache

Tableau de bord commercial

Objectif : Analyse du trafic et de l'utilisation

Panneaux :

1. Messages par SMSC source
2. Messages par SMSC de destination
3. Carte thermique d'utilisation des routes
4. Comptes d'actions de réponse automatique et de suppression
5. Statistiques d'utilisation ENUM
6. Volume de facturation par compte

Rétention des métriques

Paramètres de rétention recommandés pour Prometheus :

- **Métriques brutes** : 15 jours
- **Agrégats de 5 minutes** : 90 jours
- **Agrégats d'une heure** : 2 ans

Cela fournit un historique récent détaillé tout en maintenant des tendances à long terme pour la planification de capacité.

Dépannage avec les métriques

Scénario : Messages non livrés

Étapes d'investigation :

1. Vérifier `sms_c_message_received_count` - Les messages sont-ils reçus ?
2. Vérifier `sms_c_routing_failed_count` - Sont-ils routés ?
3. Vérifier `sms_c_delivery_queued_count` - Sont-ils mis en file d'attente ?
4. Vérifier `sms_c_delivery_failed_count` - Les tentatives de livraison échouent-elles ?
5. Vérifier les labels `dest_smsc` pour identifier la destination problématique

---

Scénario : Traitement de message lent

Étapes d'investigation :

1. Vérifier l'histogramme `sms_c_message_processing_stop_duration` - Temps de traitement global
2. Vérifier `sms_c_routing_stop_duration` - Le routage est-il lent ?
3. Vérifier `sms_c_enum_lookup_stop_duration` - Les recherches ENUM sont-elles lentes ?
4. Vérifier `sms_c_charging_succeeded_duration` - La facturation est-elle lente ?
5. Identifier le goulet d'étranglement et enquêter sur le composant spécifique

---

Scénario : Croissance de la file d'attente de messages

Étapes d'investigation :

1. Vérifier la tendance `sms_c_queue_size_pending` - Est-elle en croissance ?
2. Vérifier `sms_c_delivery_attempted_count` - Des tentatives de livraison ont-elles lieu ?

3. Vérifier `sms_c_delivery_failed_count` - Échouent-elles ?
4. Vérifier `sms_c_delivery_time_delta_delta_ms` - La livraison prend-elle trop de temps ?
5. Vérifier les labels `dest_smsc` pour identifier les destinations lentes

---

Exemples de requêtes Prometheus

Débit de messages

Messages reçus par seconde (moyenne sur 5 minutes) :

```
rate(sms_c_message_received_count[5m])
```

Messages reçus par minute (moyenne sur 1 heure) :

```
rate(sms_c_message_received_count[1h]) * 60
```

Total des messages aujourd'hui :

```
increase(sms_c_message_received_count[24h])
```

Messages par type de source :

```
sum by (source_type) (rate(sms_c_message_received_count[5m]))
```

Messages par SMSC source :

```
sum by (source_smsc) (rate(sms_c_message_received_count[5m]))
```

Performance de livraison

Taux de succès de livraison (pourcentage) :

```
(rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])) * 100
```

Taux d'échec de livraison (pourcentage) :

```
(rate(sms_c_delivery_failed_count[5m]) /
rate(sms_c_delivery_queued_count[5m])) * 100
```

Tentatives de livraison moyennes (p95) :

```
histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count_bucket)
```

Succès de livraison par destination :

```
sum by (dest_smsc) (rate(sms_c_delivery_succeeded_count[5m]))
```

Raisons d'échec de livraison :

```
sum by (reason) (rate(sms_c_delivery_failed_count[5m]))
```

Temps de livraison (p95) :

```
histogram_quantile(0.95,
sms_c_delivery_time_delta_delta_ms_bucket)
```

Temps de livraison (p99) :

```
histogram_quantile(0.99,
sms_c_delivery_time_delta_delta_ms_bucket)
```

Métriques de file d'attente

Messages en attente actuels :

```
sms_c_queue_size_pending
```

Messages échoués en attente de réessai :

```
sms_c_queue_size_failed
```

Âge du message le plus ancien (minutes) :

```
sms_c_queue_oldest_message_age_seconds / 60
```

Taux de croissance de la file d'attente (messages/heure) :

```
rate(sms_c_queue_size_size[1h]) * 3600
```

Messages entrant dans la file d'attente :

```
rate(sms_c_delivery_queued_count[5m])
```

Messages sortant de la file d'attente :

```
rate(sms_c_delivery_succeeded_count[5m]) +
rate(sms_c_delivery_failed_count[5m])
```

Arriéré de file d'attente (entrant - sortant) :

```
rate(sms_c_delivery_queued_count[5m]) -
(rate(sms_c_delivery_succeeded_count[5m]) +
rate(sms_c_delivery_failed_count[5m]))
```

Performance de routage

Taux de succès de routage :

```
(1 - (rate(sms_c_routing_failed_count[5m]) /
(rate(sms_c_routing_route_matched_count[5m]) +
rate(sms_c_routing_failed_count[5m])))) * 100
```

Routes les plus utilisées :

```
topk(10, sum by (route_id, dest_smsc)
(rate(sms_c_routing_route_matched_count[1h])))
```

Échecs de routage (par minute) :

```
rate(sms_c_routing_failed_count[5m]) * 60
```

Latence de routage (p50, p95, p99) :

```
histogram_quantile(0.50, sms_c_routing_stop_duration_bucket)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
histogram_quantile(0.99, sms_c_routing_stop_duration_bucket)
```

Actions de suppression par heure :

```
increase(sms_c_routing_action_count{action="drop"}[1h])
```

Actions de réponse automatique par heure :

```
increase(sms_c_routing_action_count{action="auto_reply"}[1h])
```

Performance ENUM

Taux de réussite du cache ENUM :

```
rate(sms_c_enum_cache_hit_count[5m]) /
(rate(sms_c_enum_cache_hit_count[5m]) +
rate(sms_c_enum_cache_miss_count[5m]))
```

Pourcentage de réussite du cache ENUM :

```
(rate(sms_c_enum_cache_hit_count[5m]) /
(rate(sms_c_enum_cache_hit_count[5m]) +
rate(sms_c_enum_cache_miss_count[5m]))) * 100
```

Latence de recherche ENUM (p95) :

```
histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)
```

Recherches ENUM par seconde (cachées vs non cachées) :

```
# Cachées (rapides)
rate(sms_c_enum_cache_hit_count[5m])

# Non cachées (requiert une requête DNS)
rate(sms_c_enum_cache_miss_count[5m])
```

Nombre moyen d'enregistrements NAPTR retournés :

```
rate(sms_c_enum_naptr_records_record_count_sum[5m]) /
rate(sms_c_enum_naptr_records_record_count_count[5m])
```

Taille du cache ENUM :

```
sms_c_enum_cache_size_size
```

Performance de traitement

Latence de traitement des messages (p95) :

```
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)
```

Latence de traitement des messages (p99) :

```
histogram_quantile(0.99,
sms_c_message_processing_stop_duration_bucket)
```

Échecs de traitement :

```
rate(sms_c_message_processing_stop_duration_count{success="false"}
[5m])
```

Taux d'échec de validation :

```
rate(sms_c_message_validated_count{valid="false"}[5m]) /
rate(sms_c_message_validated_count[5m])
```

Métriques de facturation

Taux de succès de facturation :

```
rate(sms_c_charging_succeeded_count[5m]) /
rate(sms_c_charging_requested_count[5m])
```

Échecs de facturation par minute :

```
rate(sms_c_charging_failed_count[5m]) * 60
```

Latence de facturation (p95) :

```
histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)
```

Volume de facturation par compte :

```
sum by (account) (rate(sms_c_charging_requested_count[1h]))
```

Santé des frontends

Frontends actifs :

```
sum(sms_c_frontend_status_count{status="connected"})
```

Frontends déconnectés :

```
sum(sms_c_frontend_status_count{status="disconnected"})
```

Frontends par nom :

```
sum by (frontend_name)
(sms_c_frontend_status_count{status="connected"})
```

Santé du système

Tailles des tables Mnesia :

```
sms_c_mnesia_table_size_record_count
```

Nombre de routes :

```
sms_c_mnesia_table_size_record_count{table="sms_route"}
```

Nombre de règles de traduction :

```
sms_c_mnesia_table_size_record_count{table="translation_rule"}
```

Exemples de tableau de bord Grafana

Tableau de bord 1 : Opérations en temps réel

Objectif : Surveiller l'activité et la santé du système actuelles.

Panneaux :

1. **Débit de messages (Graphique)**
   - Requête : `rate(sms_c_message_received_count[5m])`
   - Requête : `rate(sms_c_delivery_succeeded_count[5m])`
   - Unité : messages/seconde
   - Légende : {{source_type}}
2. **Taux de succès de livraison (Jauge)**
   - Requête : `(rate(sms_c_delivery_succeeded_count[5m]) / rate(sms_c_delivery_queued_count[5m])) * 100`
   - Unité : pourcentage (0-100)
   - Seuils :
     - Rouge : < 90
     - Jaune : 90-95
     - Vert : > 95
3. **Profondeur de la file d'attente (Graphique)**
   - Requête : `sms_c_queue_size_pending`

- Requête : `sms_c_queue_size_failed`
- Unité : messages
- Légende : {{queue_type}}

4. **Âge du message le plus ancien (Stat)**
   - Requête : `sms_c_queue_oldest_message_age_seconds / 60`
   - Unité : minutes
   - Seuils :
     - Vert : < 5
     - Jaune : 5-10
     - Rouge : > 10
5. **Frontends actifs (Stat)**
   - Requête : `sum(sms_c_frontend_status_count{status="connected"})`
   - Unité : compte
   - Couleur : Bleu
6. **Échecs de routage (Graphique)**
   - Requête : `rate(sms_c_routing_failed_count[5m]) * 60`
   - Unité : échecs/minute
   - Seuil d'alerte : > 0

Tableau de bord 2 : Analyse de performance

Objectif : Analyser les performances du système et identifier les goulets d'étranglement.

Panneaux :

1. **Latence de bout en bout (Graphique)**
   - Requête : `histogram_quantile(0.50, sms_c_message_processing_stop_duration_bucket)` (p50)
   - Requête : `histogram_quantile(0.95, sms_c_message_processing_stop_duration_bucket)` (p95)

- Requête : `histogram_quantile(0.99, sms_c_message_processing_stop_duration_bucket)` (p99)
- Unité : millisecondes
- Légende : Percentile

2. **Latences des composants (Jauge à barres)**

- Routage : `histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)`
- ENUM : `histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)`
- Facturation : `histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)`
- Livraison : `histogram_quantile(0.95, sms_c_delivery_time_delta_delta_ms_bucket)`
- Unité : millisecondes
- Barres horizontales

3. **Distribution des tentatives de livraison (Carte thermique)**

- Requête : `sms_c_delivery_succeeded_attempt_count_bucket`
- Montre combien de tentatives sont généralement nécessaires
- Échelle de couleur : Bleu (1 tentative) à Rouge (beaucoup de tentatives)

4. **Performance du cache ENUM (Graphique)**

- Taux de réussite : `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`
- Taille du cache : `sms_c_enum_cache_size_size`
- Axe Y double (taux vs taille)

5. **Taux de succès de traitement (Jauge)**

- Requête : `(rate(sms_c_message_processing_stop_duration_count{success="tru`

`e"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m])) * 100`
    - Unité : pourcentage
    - Seuils :
        - Rouge : < 95
        - Jaune : 95-99
        - Vert : > 99

Tableau de bord 3 : Analyse du trafic

Objectif : Analyser les modèles de trafic de messages et la distribution de routage.

Panneaux :

1. **Messages par type de source (Graphique circulaire)**
    - Requête : `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
    - Montre la distribution : IMS vs CS vs SMPP
2. **Messages par SMSC source (Graphique à barres)**
    - Requête : `sum by (source_smsc) (rate(sms_c_message_received_count[1h]))`
    - Top 10 des sources
    - Barres horizontales
3. **Utilisation des routes (Tableau)**
    - Colonnes :
        - Route ID
        - SMSC de destination
        - Messages (1h) : `sum by (route_id, dest_smsc) (increase(sms_c_routing_route_matched_count[1h]))`
        - Priorité
        - Taux de succès

- Trié par nombre de messages

4. **Livraison par destination (Graphique)**
   - Requête : `sum by (dest_smsc) (rate(sms_c_delivery_succeeded_count[5m]))`
   - Unité : messages/seconde
   - Graphique en zone empilée
   - Légende : {{dest_smsc}}
5. **Actions de suppression/réponse automatique (Stat)**
   - Supprimé : `increase(sms_c_routing_action_count{action="drop"}[1h])`
   - Réponse automatique : `increase(sms_c_routing_action_count{action="auto_reply"}[1h])`
   - Statistiques côte à côte
6. **Modèle de trafic horaire (Graphique)**
   - Requête : `rate(sms_c_message_received_count[1h]) * 3600`
   - Plage horaire : Derniers 7 jours
   - Montre les modèles quotidiens

Tableau de bord 4 : Capacité et ressources

Objectif : Surveiller l'utilisation des ressources et les limites de capacité.

Panneaux :

1. **Capacité de la file d'attente (Graphique)**
   - Actuel : `sms_c_queue_size_size`
   - Ligne de capacité : valeur fixe basée sur les limites du système
   - Montre la tendance d'utilisation
2. **Croissance des tables de base de données (Graphique)**

- Messages : `sms_c_mnesia_table_size_record_count{table="sms_route"}`
- Traductions : `sms_c_mnesia_table_size_record_count{table="translation_rule"}`
- Tendance sur les 30 derniers jours

3. **Tendance d'arriéré de livraison (Graphique)**
   - Requête : `rate(sms_c_delivery_queued_count[5m]) - (rate(sms_c_delivery_succeeded_count[5m]) + rate(sms_c_delivery_failed_count[5m]))`
   - Positif = arriéré croissant
   - Négatif = rattrapage
4. **Trafic de pointe (Stat)**
   - Requête : `max_over_time(rate(sms_c_message_received_count[5m])[24h:])`
   - Montre le taux le plus élevé de 5 minutes au cours des dernières 24 heures
   - Unité : messages/seconde
5. **Utilisation de la capacité (Jauge)**
   - Requête : `(rate(sms_c_message_received_count[5m]) / MAX_CAPACITY) * 100`
   - Remplacer MAX_CAPACITY par votre limite système
   - Unité : pourcentage
   - Seuils :
     - Vert : < 70
     - Jaune : 70-85
     - Rouge : > 85

Tableau de bord 5 : Conformité SLA

Objectif : Suivre les métriques SLA et la conformité.

Panneaux :

1. **Conformité SLA (Jauge)**
    - Succès de livraison : `(rate(sms_c_delivery_succeeded_count[1h]) / rate(sms_c_delivery_queued_count[1h])) * 100`
    - Ligne cible à 99 %
    - Seuils :
        - Rouge : < 95
        - Jaune : 95-99
        - Vert : >= 99
2. **Messages livrés dans le SLA (Stat)**
    - Requête : `count(sms_c_delivery_time_delta_delta_ms_bucket{le="5000"}) / count(sms_c_delivery_time_delta_delta_ms_bucket)`
    - Montre le pourcentage livré en moins de 5 secondes
    - Unité : pourcentage
3. **Violations SLA (Counter)**
    - Messages dépassant 5 minutes : `increase(sms_c_queue_oldest_message_age_seconds{} > 300)[24h:]`
    - Devrait être 0
4. **Uptime (Stat)**
    - Requête : `up{job="sms-c"}`
    - Binaire : 1 = en ligne, 0 = hors ligne
    - Montre l'état actuel
5. **Tendance du taux de succès quotidien (Graphique)**
    - Requête : `avg_over_time((rate(sms_c_delivery_succeeded_count[1h]) / rate(sms_c_delivery_queued_count[1h]))[24h:1h])`
    - Plage horaire : Derniers 30 jours
    - Ligne SLA à 99 %

Exemples de règles d'alerte

Alertes critiques

Échecs de routage :

```
alert: RoutingFailuresDetected
expr: increase(sms_c_routing_failed_count[5m]) > 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "{{ $value }} échecs de routage dans les 5 dernières
minutes"
  description: "Les messages ne peuvent pas être routés. Vérifiez
la configuration de routage."
```

Arriéré de file d'attente :

```
alert: MessageQueueBacklog
expr: sms_c_queue_size_pending > 10000
for: 5m
labels:
  severity: critical
annotations:
  summary: "La file d'attente de messages a {{ $value }} messages
en attente"
  description: "La file d'attente est en cours de remplissage.
Vérifiez les performances de livraison."
```

Anciens messages dans la file d'attente :

```
alert: OldMessagesInQueue
expr: sms_c_queue_oldest_message_age_seconds > 300
for: 2m
labels:
  severity: critical
annotations:
  summary: "Le message le plus ancien a {{ $value }} secondes"
  description: "Les messages ne sont pas livrés. Vérifiez les
frontends."
```

Tous les frontends déconnectés :

```
alert: NoActiveFrontends
expr: sum(sms_c_frontend_status_count{status="connected"}) == 0
for: 1m
labels:
  severity: critical
annotations:
  summary: "Aucun frontend connecté"
  description: "Aucun chemin de livraison disponible. Vérifiez la
connectivité des frontends."
```

Croissance de la file de lettres mortes :

```
alert: DeadLetterMessagesIncreasing
expr: rate(sms_c_delivery_dead_letter_count[10m]) > 0
for: 5m
labels:
  severity: critical
annotations:
  summary: "{{ $value }} messages déplacés vers la file de lettres
mortes"
  description: "Les messages deviennent non livrables. Enquêtez
sur les échecs."
```

Alertes d'avertissement

Faible taux de succès de livraison :

```
alert: LowDeliverySuccessRate
expr: (rate(sms_c_delivery_succeeded_count[10m]) /
rate(sms_c_delivery_queued_count[10m])) < 0.95
for: 10m
labels:
  severity: warning
annotations:
  summary: "Le taux de succès de livraison est {{ $value |
humanizePercentage }}"
  description: "Le taux de succès est inférieur à 95 %. Enquêtez
sur les échecs de livraison."
```

Taux de réessai élevé :

```
alert: HighDeliveryRetryRate
expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count_bucket) > 2
for: 15m
labels:
  severity: warning
annotations:
  summary: "Tentatives de livraison au 95e percentile : {{ $value
}}"
  description: "Les messages nécessitant plusieurs tentatives.
Vérifiez la fiabilité de la destination."
```

Traitement de message lent :

```
alert: SlowMessageProcessing
expr: histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket) > 1000
for: 10m
labels:
  severity: warning
annotations:
  summary: "Temps de traitement au 95e percentile : {{ $value
}}ms"
  description: "Le traitement des messages est lent. Vérifiez les
ressources système."
```

Recherches ENUM échouant :

```
alert: HighEnumFailureRate
expr: rate(sms_c_enum_lookup_stop_duration_count{success="false"}
[10m]) > 0.1
for: 10m
labels:
  severity: warning
annotations:
  summary: "Taux d'échec de recherche ENUM : {{ $value }}"
  description: "Les recherches DNS échouent. Vérifiez les serveurs
DNS."
```

Faible taux de réussite du cache ENUM :

```
alert: LowEnumCacheHitRate
expr: rate(sms_c_enum_cache_hit_count[10m]) /
(rate(sms_c_enum_cache_hit_count[10m]) +
rate(sms_c_enum_cache_miss_count[10m])) < 0.70
for: 30m
labels:
  severity: warning
annotations:
  summary: "Taux de réussite du cache ENUM : {{ $value |
humanizePercentage }}"
  description: "Efficacité du cache faible. Peut indiquer un
trafic de numéros uniques."
```

Échecs de facturation :

```
alert: ChargingFailuresDetected
expr: rate(sms_c_charging_failed_count[10m]) > 0.05
for: 10m
labels:
  severity: warning
annotations:
  summary: "Taux d'échec de facturation : {{ $value }}"
  description: "Erreurs du système de facturation. Vérifiez la
connectivité OCS."
```

Notes supplémentaires

- Toutes les métriques de durée utilisent une précision en nanosecondes en interne mais sont converties en millisecondes pour le reporting.
- Les métriques de compteur sont cumulatives et doivent être utilisées avec les fonctions `rate()` ou `increase()` dans les requêtes Prometheus.
- Les métriques de gauge représentent des valeurs instantanées au moment de la collecte.
- Les métriques d'histogramme fournissent des calculs de percentile (p50, p95, p99) et peuvent être utilisées pour créer des cartes thermiques.
- Toutes les métriques incluent des labels par défaut ajoutés par Prometheus (instance, job, etc.).
- Lors de la création de tableaux de bord, utilisez des plages de temps appropriées : 5m pour le temps réel, 1h pour les tendances, 24h+ pour la planification de capacité.
- Configurez des règles d'enregistrement dans Prometheus pour des requêtes complexes fréquemment utilisées afin d'améliorer les performances des tableaux de bord.
- Utilisez le templating de variables dans Grafana pour des tableaux de bord dynamiques (sélectionner dest_smsc, source_smsc, etc.)

Centre MMS (MMSc)

← Retour à l'Index de Documentation | **Référence de Configuration** | **Référence API**

Table des Matières

- Aperçu
- Architecture
- Traitement de Soumission de Message
- Flux de Livraison
- Notification MMS
- Détermination On/Off-Net
- Configuration
- Stockage
- Comportement à l'Exécution
- Observabilité
- Dépannage

Aperçu

OmniMessage comprend un **Centre MMS (MMSc)** natif qui reçoit, stocke et livre des messages multimédias. Il met en œuvre les deux interfaces d'interconnexion MMS définies par 3GPP TS 23.140 : **MM1** pour la communication directe avec le terminal (HTTP/WAP) et **MM4** pour le relais inter-opérateurs (SMTP).

Le MMSc est intégré au cœur d'OmniMessage. Il ne fonctionne pas en tant que service séparé et n'a aucune dépendance externe à une base de donn��es ou à un serveur web. Lorsqu'un abonné envoie ou reçoit un MMS, le MMSc :

1. Stocke le corps brut du MMS sur le système de fichiers local et ses métadonnées dans Mnesia.

2. Construit une **notification MMS** et la soumet comme un SMS ordinaire terminé par un mobile via le même pipeline de traitement utilisé pour chaque SMS, qui le traduit, le route et le livre ensuite.
3. Sert le MMS stocké au terminal qui le récupère via MM1 lorsqu'il accède à l'URL de notification.

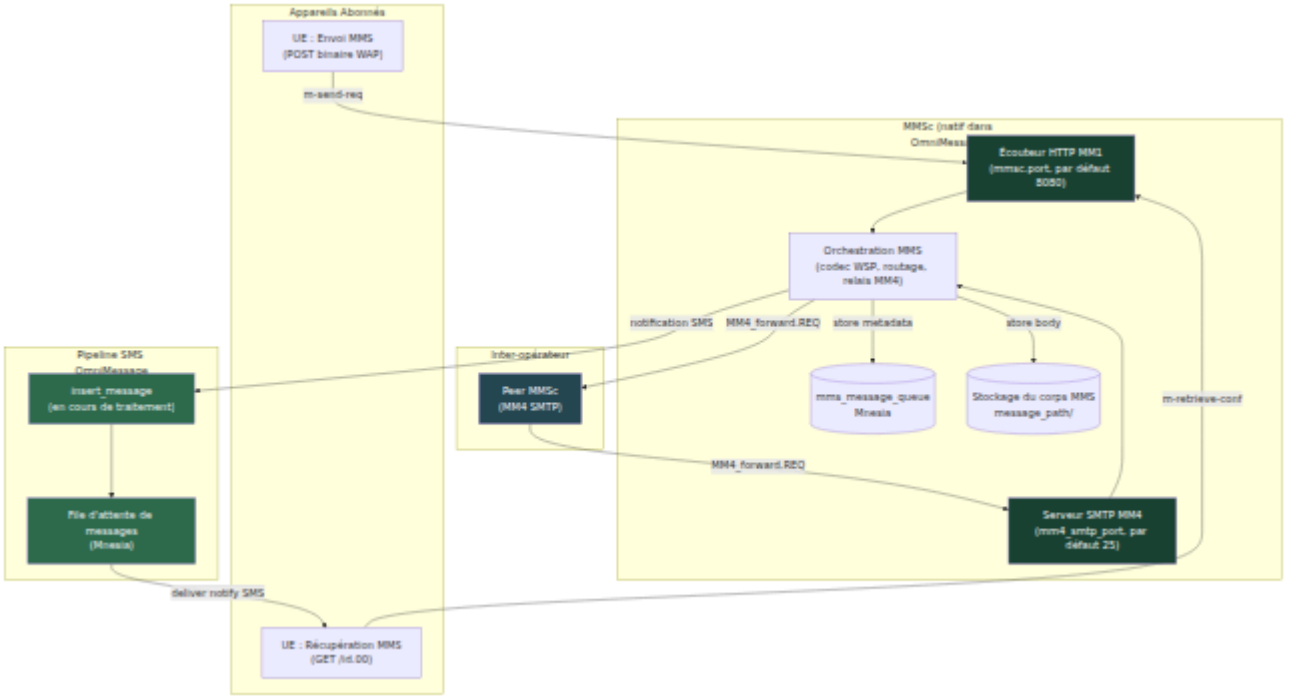
Parce que la notification est injectée comme un SMS normal, le MMS réutilise toute la machinerie existante de routage, de traduction et de livraison de SMS sans appels HTTP autodirigés.

Spécifications Pertinentes

Spécification	Titre
3GPP TS 23.140	Service de Messagerie Multimédia (MMS) : description fonctionnelle
3GPP TS 26.140	MMS : formats et codecs multimédias
Protocole d'Encapsulation MMS OMA	Encodage binaire WAP MM1

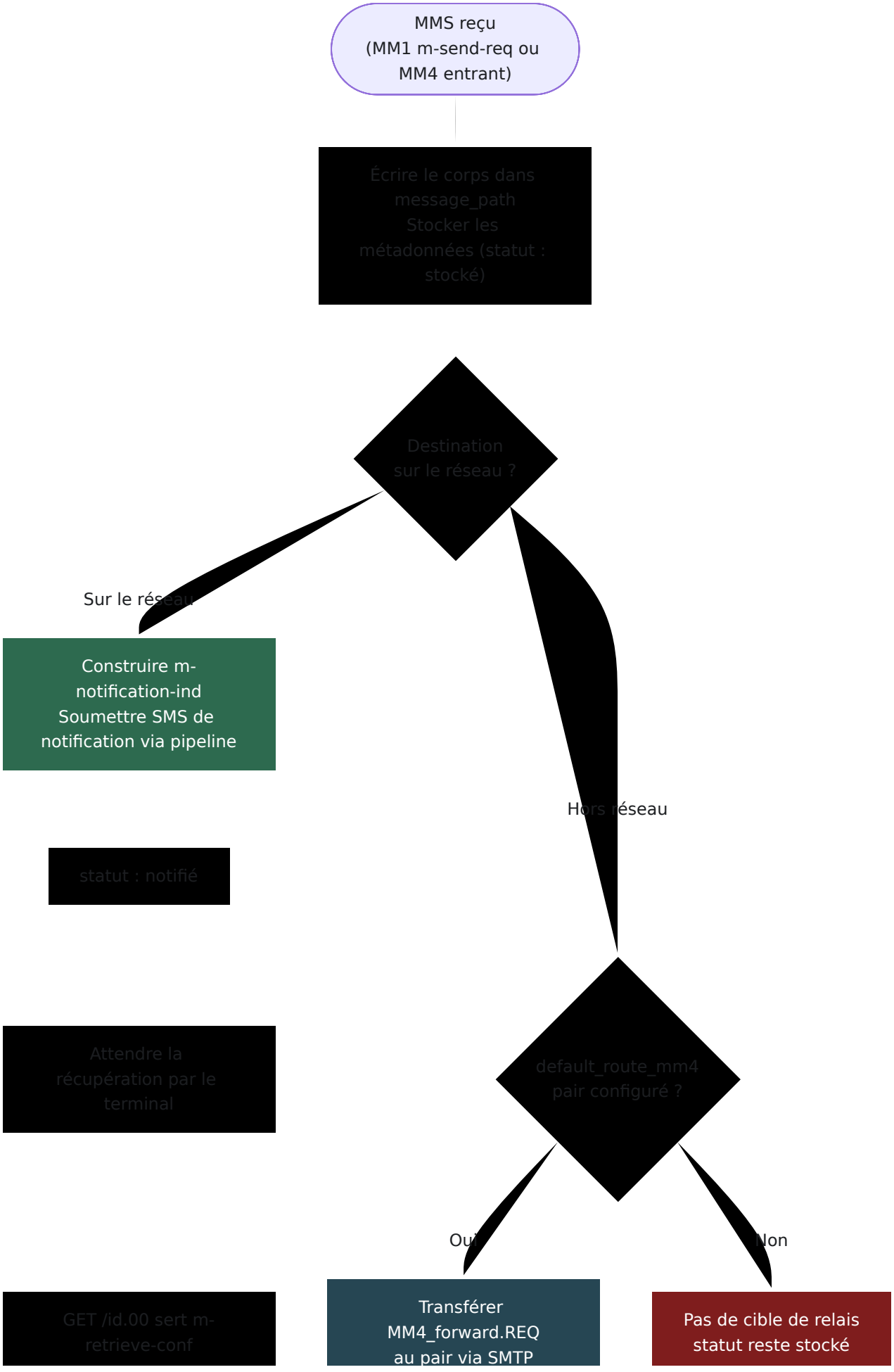
Architecture

Les interfaces MM1 (HTTP du terminal) et MM4 (SMTP inter-opérateur) écoutent chacune sur leur propre port dédié, séparé de l'API OmniMessage. Les métadonnées MMS résident dans Mnesia aux côtés du stockage des messages SMS. Le MMSc conserve les corps MMS bruts sous forme de fichiers sur le système de fichiers local. L'enregistrement des métadonnées ne conserve que le chemin du fichier.



Traitement de Soumission de Message

Chaque MMS soumis (d'un terminal via MM1, ou d'un pair via MM4) suit le même chemin de traitement : stocker le corps, stocker les métadonnées, classifier la destination, puis soit notifier localement soit relayer hors réseau.

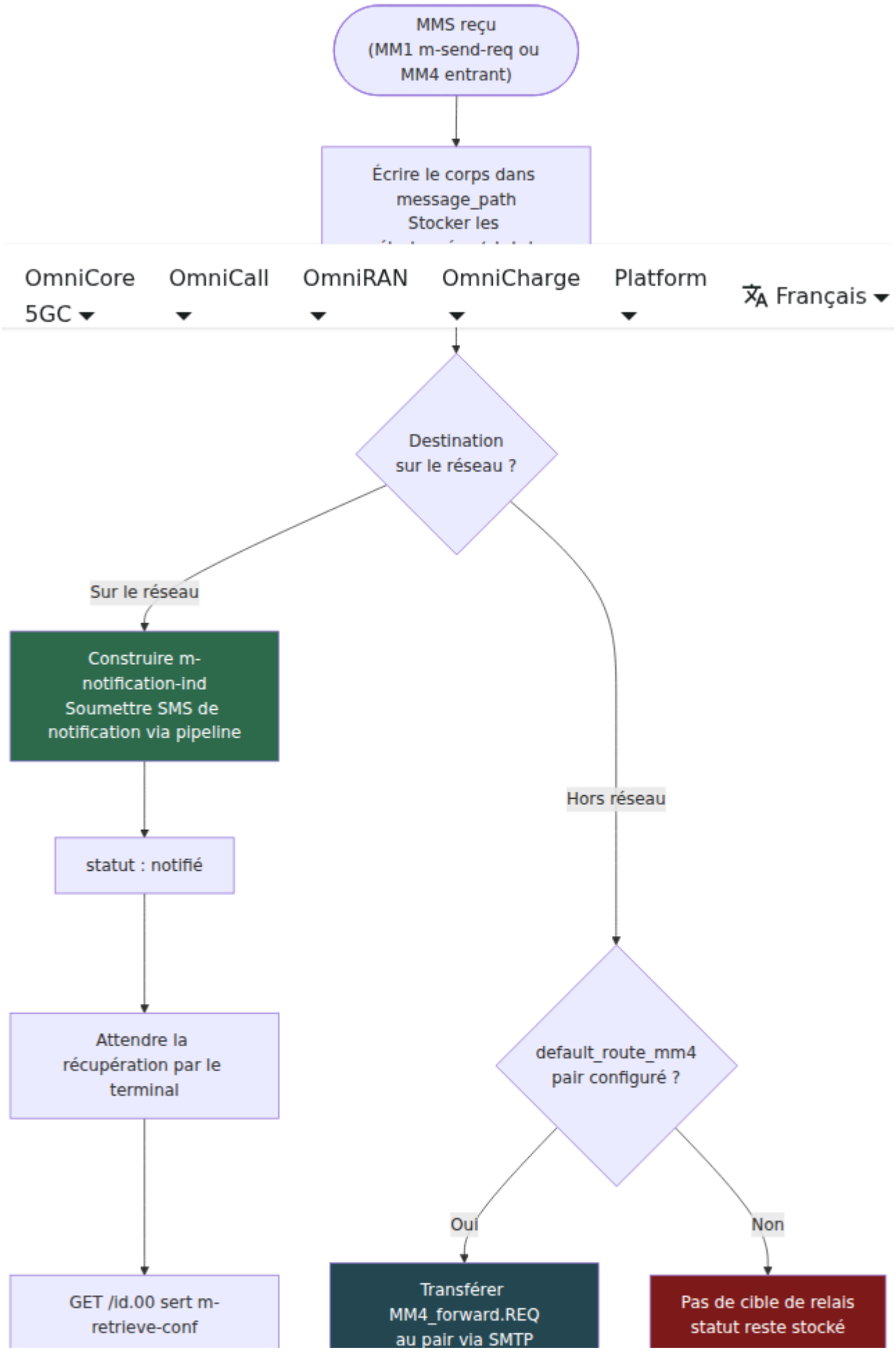

MMS reçu
(MM1 m-send-req ou
MM4 entrant)
Écrire le corps dans
message_path
Stocker les
métadonnées (statut :
stocké)
Destination
sur le réseau ?
Sur le réseau
Construire m-
notification-ind
Soumettre SMS de
notification via pipeline
statut : notifié
Hors réseau
Attendre la
récupération par le
terminal
default_route_mm4
pair configuré ?
Oui
Non
GET /id.00 sert m-
retrieve-conf
Transférer
MM4_forward.REQ
au pair via SMTP
Pas de cible de relais
statut reste stocké

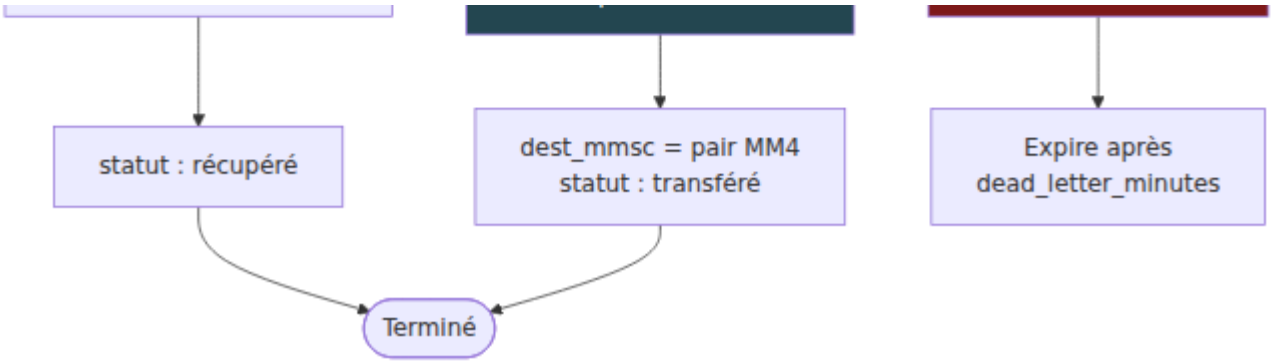


Flux de Livraison

Livraison Sur le Réseau (MM1)

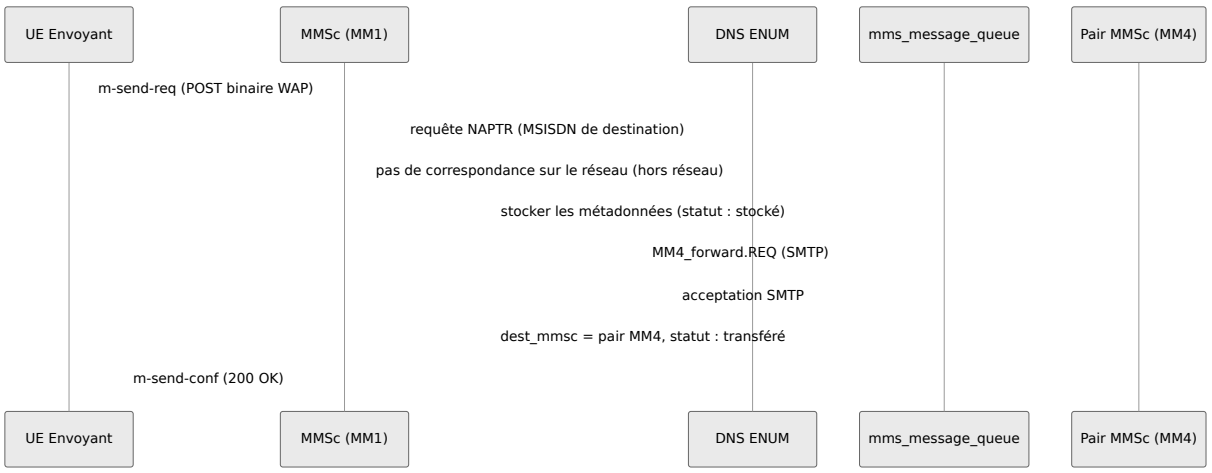
Un abonné envoie un MMS à un destinataire servi par le même réseau.


MMS reçu
(MM1 m-send-req ou
MM4 entrant)
Écrire le corps dans
message_path
Stocker les
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
Destination
sur le réseau ?
Sur le réseau
Construire m-
notification-ind
Soumettre SMS de
notification via pipeline
statut : notifié
Attendre la
récupération par le
terminal
GET /id.00 sert m-
retrieve-conf
Hors réseau
default_route_mm4
pair configuré ?
Oui
Non
Transférer
MM4_forward.REQ
au pair via SMTP
Pas de cible de relais
statut reste stocké



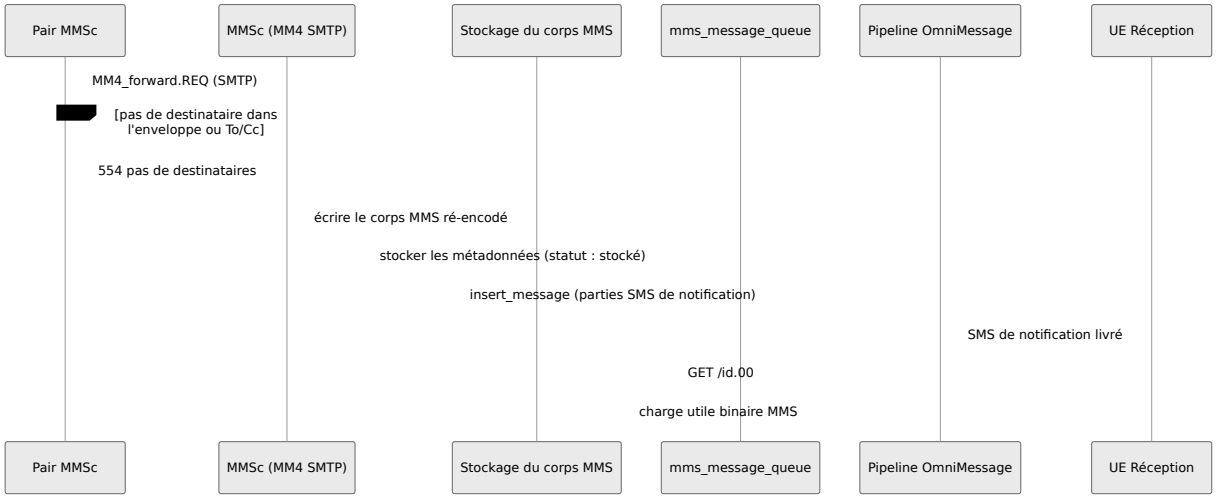
Livraison Hors Réseau (MM4)

Un abonné envoie un MMS à un destinataire chez un autre opérateur.



MM4 entrant (d'un Pair Opérateur)

Un abonné sur un autre réseau envoie un MMS à un abonné sur le réseau.



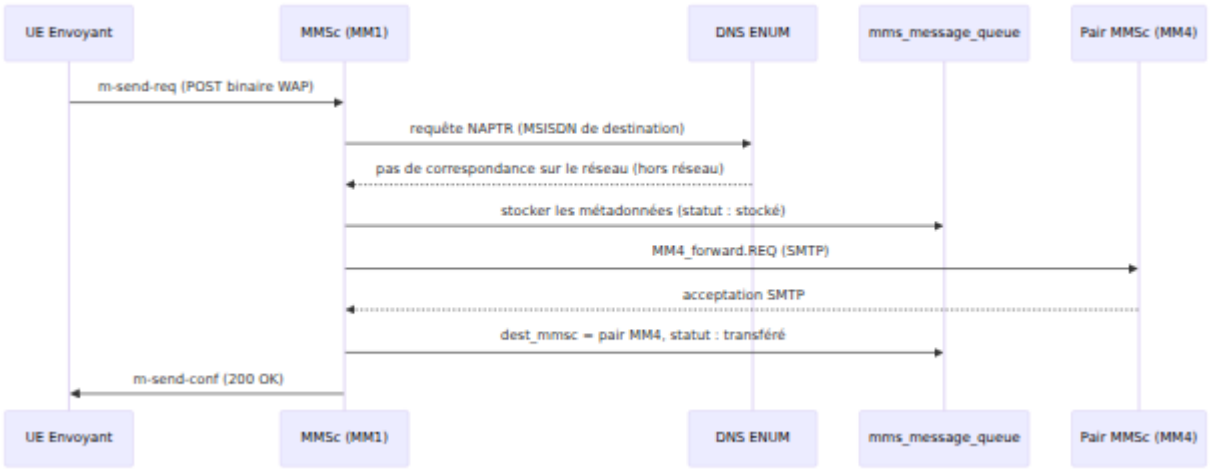
- **La livraison** va aux destinataires de l'enveloppe SMTP (`RCPT TO`) ; les pairs adressent une enveloppe par abonné sur le réseau. Ce n'est que

lorsque l'enveloppe ne contient aucun destinataire que les en-têtes `To`/`Cc` décident.

- **Membres du groupe** : `To` + `Cc` forment ensemble la liste des membres stockée dans `recipient_list` et listée dans le `m-retrieve-conf`, de sorte que le terminal conserve un MMS de groupe dans son fil de discussion de groupe.
- **Média** : chaque partie non-SMIL de l'arbre MIME (y compris les multiparts imbriqués) est ré-encodée dans le corps WSP stocké. Les parties texte sont converties de leur charset déclaré en UTF-8.
- Un message sans destinataire dans l'enveloppe ou `To`/`Cc` est refusé avec `554` plutôt qu'accepté et abandonné.

Notification MMS

La notification MMS est un SMS binaire portant un **PDU `m-notification-ind` encodé en WSP**. Il utilise un codage de données 8 bits (DCS `0x44`), est divisé en segments SMS concaténés, et est soumis à la file d'attente de messages comme des parties SMS MT ordinaires via le pipeline en cours de traitement. Chaque règle de traduction, de routage et de livraison SMS s'applique donc à elle sans changement.



L'emplacement de récupération encodé dans la notification est :

```
http://<mms_domain>/<unique_id>.00
```

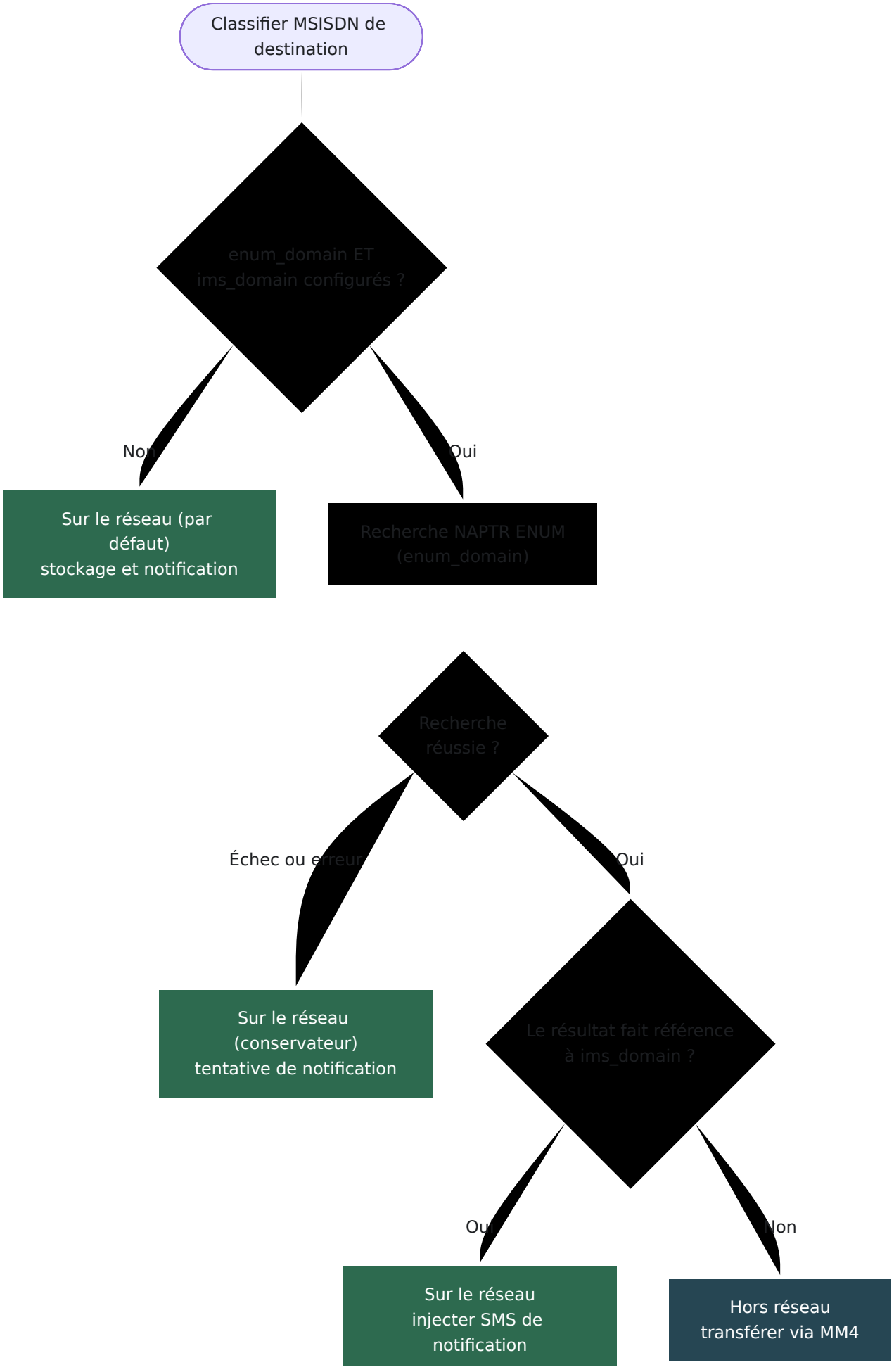
- `mms_domain` est le `mmsc.mms_domain` configuré.

- `<unique_id>.00` est la clé de récupération. Le gestionnaire MM1 supprime le suffixe `.00` pour résoudre le message stocké.

Le terminal récepteur ouvre cette URL via MM1 pour récupérer le MMS sous forme de `m-retrieve-conf`.

Détermination On/Off-Net

Avant que le MMSc ne décide s'il doit notifier localement ou relayer via MM4, il réutilise la machinerie ENUM (NAPTR) de l'SMS-C pour classifier la destination. Un résultat ENUM dont la regexp/remplacement fait référence au `ims_domain` configuré détermine le statut sur le réseau.


Classifier MSISDN de destination
enum_domain ET ims_domain configurés ?
Non
Oui
Sur le réseau (par défaut)
stockage et notification
Recherche NAPTR ENUM (enum_domain)
Recherche réussie ?
Échec ou erreur
Oui
Sur le réseau
(conservateur)
tentative de notification
Le résultat fait référence à ims_domain ?
Oui
Non
Sur le réseau
injecter SMS de notification
Hors réseau
transférer via MM4

Condition	Classification	Action
`ims_domain` ou `enum_domain` non configurés	Sur le réseau (par défaut)	Stocker localement et injecter le SMS de notification
Le résultat NAPTR ENUM fait référence à `ims_domain`	Sur le réseau	Stocker localement et injecter le SMS de notification
Le résultat NAPTR ENUM ne fait pas référence à `ims_domain`	Hors réseau	Stocker localement et transférer au pair via MM4
Échec de recherche ENUM ou erreur de recherche	Sur le réseau (conservateur)	Stocker localement et tenter la notification

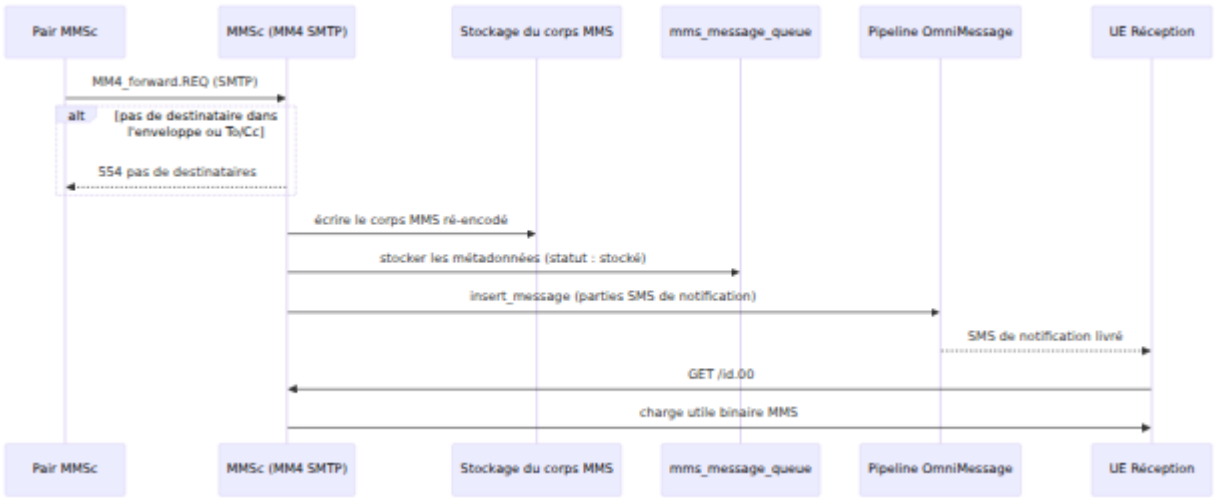
Lorsque ENUM n'est pas configuré, toutes les destinations sont traitées comme sur le réseau (stockage et notification). Le relais MM4 n'est tenté que pour les destinations classées positivement comme hors réseau.

Configuration

Vous configurez le MMSc dans `runtime.exs` sous la clé `:mmsc` de l'application `:sms_c`. Il est **désactivé par défaut**. Si le bloc `:mmsc` est absent, la fonctionnalité est complètement inactive : le MMSc ne lie aucun port et ne démarre aucun écouteur. Les déploiements existants ne sont donc pas affectés tant que vous ne l'activez pas explicitement.

Structure de Configuration

Chaque clé de configuration pilote une partie spécifique du MMSc. Le diagramme montre quelles clés contrôlent quel processus.



Exemple de Configuration

```
# config/runtime.exs
config :sms_c, :mmsc,
  # Noyau
  enabled: true,
  port: 8080,
  host: "0.0.0.0",
  mms_domain: "mms.example.com",
  message_path: "/var/lib/omnimessage/stored_mms",
  notify_source_msisdn: "6675",
  dead_letter_minutes: 1440,

  # Classification on/off-net (ENUM)
  enum_domain: "e164.arpa",
  ims_domain: "ims.mnc001.mcc001.3gppnetwork.org",

  # Résolution de l'abonné (IP client MM1 vers MSISDN)
  subscriber_ip_api_urls: ["https://10.0.0.10:8443",
"https://10.0.0.11:8443"],
  pcrf_api_url: nil,
  hss_api_url: nil,

  # MM4 entrant/sortant (inter-opérateur)
  mm4_enabled: false,
  mm4_smtp_port: 25,
  mm4_domain: "mms.example.com",
  default_route_mm4: nil
```

Paramètres `:mmsc`

Paramètre	Type	Requis	Par défaut
`enabled`	Booléen	Non	`false`
`port`	Entier	Non	`8080`
`host`	Chaîne	Non	`0.0.0.0`
`mms_domain`	Chaîne	Oui (si activé)	-
`message_path`	Chaîne	Non	`/var/lib/omnimessage/sto`
`notify_source_msisdn`	Chaîne	Non	`6675`
`dead_letter_minutes`	Entier	Non	`1440`
`enum_domain`	Chaîne	Non	-

Paramètre	Type	Requis	Par défaut
`ims_domain`	Chaîne	Non	`""`
`subscriber_ip_api_urls`	Liste de chaînes	Non	`[]`
`pcrf_api_url`	Chaîne	Non	`nil`
`hss_api_url`	Chaîne	Non	`nil`
`mm4_enabled`	Booléen	Non	`false`
`mm4_smtp_port`	Entier	Non	`25`

Paramètre	Type	Requis	Par défaut
`mm4_domain`	Chaîne	Non	revient à `mms_domain`
`default_route_mm4`	Chaîne	Non	`nil`

Pairs MM4

Les cibles de relais MM4 sortantes sont configurées séparément, car elles sont des enregistrements hôte/port/domaine structurés plutôt que des paramètres scalaires :

```
# config/runtime.exs
config :sms_c, :mmsc_mm4_peers, %{
  "carrier-a" => [
    %{host: "mm4.carrier-a.example.net", port: 25, mms_domain:
"mms.carrier-a.example.net"}
  ]
}
```

`mmsc.default_route_mm4` sélectionne quel groupe de pairs est utilisé. Chaque groupe est une liste ; lorsqu'il contient plusieurs entrées, une est choisie par tentative de relais afin que la charge soit répartie entre les hôtes entrants d'un pair.

Paramètres des pairs MM4 :

Paramètre	Type	Requis	Par défaut	Description
`host`	Chaîne	Oui	-	Nom d'hôte ou IP SMTP MM4 du pair. Utilisé comme relais SMTP nom d'hôte HELO.
`port`	Entier	Non	`25`	Port SMTP MM4 du pair.
`mms_domain`	Chaîne	Non	`""`	Domaine MMS du pair, utilisé po construire l'adresse RCPT (`+<msisdn>/TYPE=PLMN@<mms_doma` et enregistré comme `dest_mmsc`

Stockage

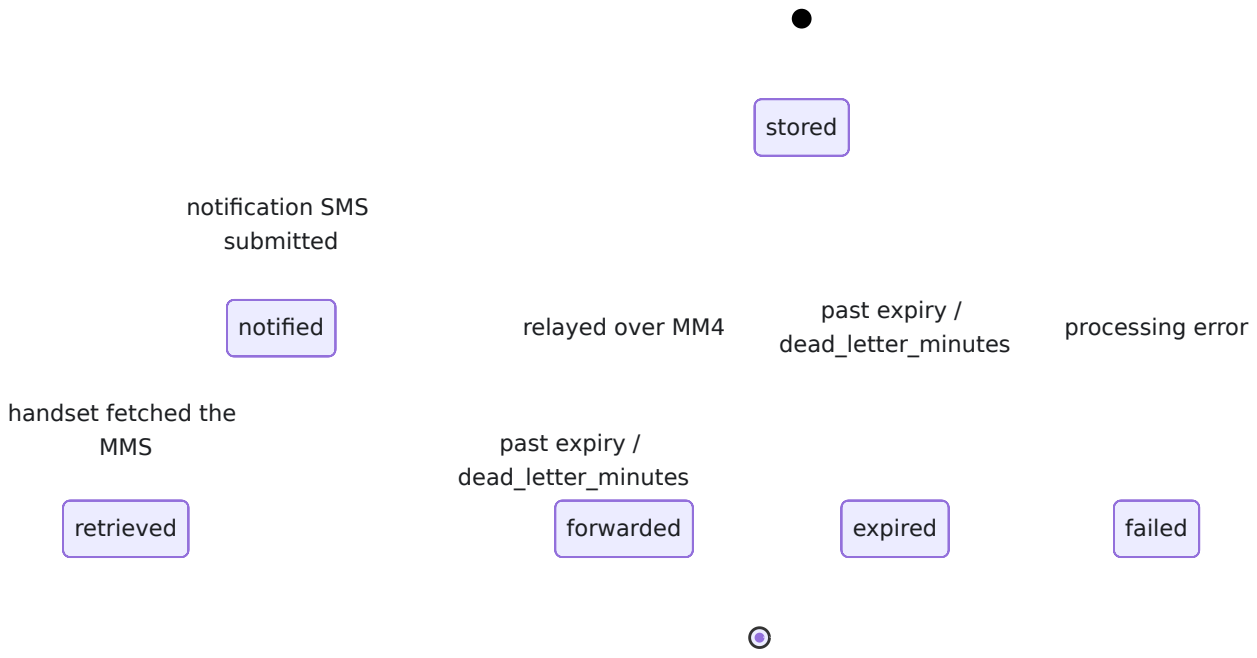
Les **métadonnées** MMS sont stockées dans la table Mnesia `:mms_message_queue`, créée automatiquement au démarrage aux côtés du stockage des messages SMS. Il n'y a pas de base de données séparée à provisionner. Le **corps** brut du MMS est stocké sous forme de fichier `.mms` dans `message_path` ; seul son chemin est conservé dans l'enregistrement des métadonnées.

Champs `mms_message_queue`

Champ	Description
`mms_id`	Clé primaire entière d'un compteur Mnesia.
`unique_id`	Identifiant externe utilisé dans le chemin d'URL de récupération (`<unique_id>.00`).
`source_msisdn`	MSISDN de l'abonné expéditeur.
`destination_msisdn`	MSISDN du destinataire principal.
`recipient_list`	Liste complète To/Cc (CSV) pour le MMS de groupe et le retrieve-conf.
`subject`	Ligne de sujet du MMS.
`file_path`	Chemin vers le corps `.mms` stocké sous `message_path`.
`source_mmsc`	Identité du MMSc ou du pair MM4 qui a originaire le message.
`dest_mmsc`	Défini sur `MM4_<pair>` après relais sortant, ou la cible de livraison après livraison locale.
`on_net`	Indique si le MMSc a classé la destination comme étant sur le réseau.
`notify_message_ids`	Identifiants de message SMS créés pour la notification (liens vers le stockage de messages SMS).
`send_time`	Quand ce MMSc a reçu le MMS.

Champ	Description
`notify_time`	Quand le système a soumis le SMS de notification.
`retrieve_time`	Quand l'abonné a récupéré le MMS.
`expires`	Temps d'expiration après lequel le MMS n'est pas livré.
`status`	État du cycle de vie (voir ci-dessous).
`deadletter`	Défini lorsque le message est en lettre morte.
`inserted_at` / `updated_at`	Horodatages du cycle de vie de l'enregistrement.

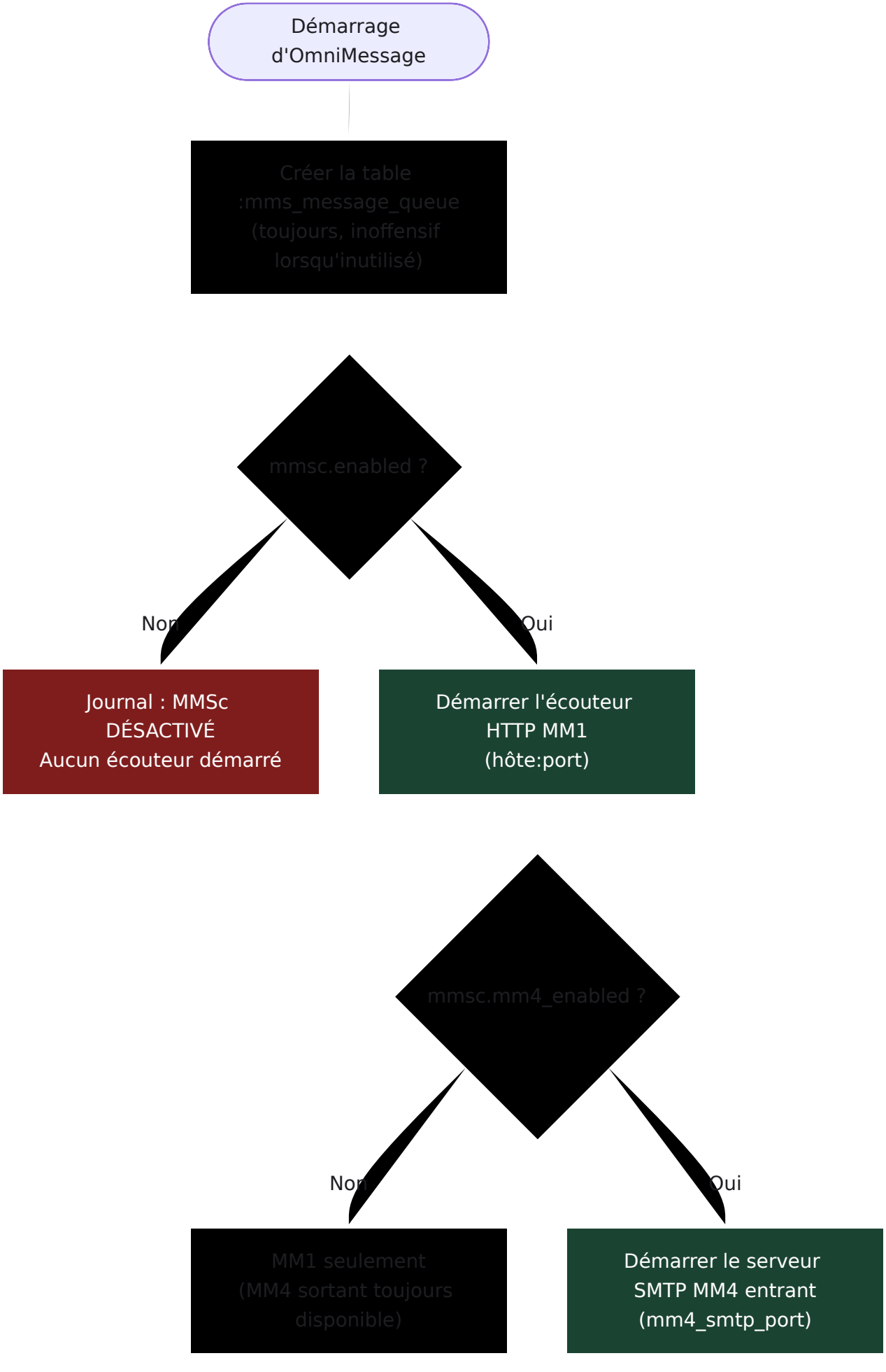
Cycle de Vie du Message



Statut	Signification
`stored`	Corps écrit et métadonnées enregistrées ; aucune action de livraison n'a encore été prise.
`notified`	SMS de notification soumis au pipeline pour un destinataire sur le réseau.
`retrieved`	Le destinataire a récupéré le MMS via MM1.
`forwarded`	Transféré à un pair opérateur via MM4.
`expired`	Passé sa date d'expiration ou `dead_letter_minutes` sans livraison.
`failed`	Erreur de traitement ou de livraison.

Comportement à l'Exécution

Le MMSc fonctionne comme des processus supervisés à l'intérieur de la version d'OmniMessage. Rien n'est déployé ou démarré séparément. Le diagramme montre quelles configurations contrôlent chaque écouteur au démarrage.


Démarrage d'OmniMessage
Créer la table :mms_message_queue (toujours, inoffensif lorsqu'inutilisé)
mmsc.enabled ?
Non
Oui
Journal : MMSc DÉSACTIVÉ Aucun écouteur démarré
Démarrer l'écouteur HTTP MM1 (hôte:port)
mmsc.mm4_enabled ?
Non
Oui
MM1 seulement (MM4 sortant toujours disponible)
Démarrer le serveur SMTP MM4 entrant (mm4_smtp_port)

- **Le listener HTTP MM1** démarre uniquement lorsque `mmsc.enabled` est `true`. Il se lie à `mmsc.host:mmsc.port` et sert les soumissions des terminaux (`m-send-req`), les réponses de notification (`m-notifyresp-ind`), les accusés de réception (`m-acknowledge-ind`), et les récupérations (`GET /<unique_id>.00`).
- **Le serveur SMTP MM4** démarre uniquement lorsque `mmsc.enabled` et `mmsc.mm4_enabled` sont tous deux `true`. Il se lie à `mmsc.mm4_smtp_port` et accepte les messages entrants `MM4_forward.REQ` des opérateurs partenaires.
- **Le relais MM4 sortant** est effectué en interne lorsque la destination est classée hors réseau ; il se connecte à l'opérateur configuré via SMTP. Aucun agent de transfert de mail externe n'est requis.
- **La création de table** s'exécute toujours au démarrage. Lorsque le MMSc est désactivé, cela crée simplement une table vide `:mms_message_queue`, ce qui est inoffensif.

Liaison de port : le port MM4 par défaut (25) est privilégié. Le service OmniMessage doit soit s'exécuter avec la capacité de lier des ports bas (`CAP_NET_BIND_SERVICE`), soit utiliser un `mm4_smtp_port` plus élevé derrière un transfert de port. Le listener MM1 utilise par défaut le port non privilégié 8080.

TLS : le listener MM1 sert du HTTP simple. Terminez TLS à un proxy inverse devant `mmsc.port` si un chiffrement côté terminal est requis.

Observabilité

Le MMSc n'émet actuellement pas de métriques MMS Prometheus dédiées. La livraison sur le réseau injecte un SMS de notification dans le pipeline standard. Les métriques de message OmniMessage normales montrent donc les notifications MMS. Ces notifications apparaissent dans la file de messages et les CDR comme tout autre SMS. Voir Référence des métriques.

Pour le traçage par message, les champs `status`, `send_time`, `notify_time` et `retrieve_time` de l'enregistrement de métadonnées suivent un message à travers son cycle de vie.

Dépannage

Le MMSc ne démarre pas

Symptômes : Pas de listener MM1 sur le port configuré ; les journaux de démarrage montrent `MMSc: DISABLED`.

Causes possibles :

- Le bloc `:mmsc` est absent de `runtime.exs`.
- `mmsc.enabled` est `false`.

Résolution :

1. Ajoutez le bloc de configuration `:mmsc` et définissez `enabled: true`.
2. Redémarrez le service et confirmez que le journal de démarrage indique `MMSc: ENABLED` avec le port et le domaine attendus.

Le terminal ne peut pas récupérer le MMS

Symptômes : Le SMS de notification est livré, mais le terminal échoue à récupérer le contenu.

Causes possibles :

- `mms_domain` n'est pas résolvable ou non routable depuis l'APN MMS du terminal.
- Un proxy inverse ou un pare-feu bloque `mmsc.port`.
- Le fichier `.mms` stocké est manquant sous `message_path`.

Résolution :

1. Confirmez que l'hôte de l'URL de notification (`mms_domain`) se résout et est accessible depuis l'APN MMS.
2. Vérifiez l'accès entrant à `mmsc.host:mmsc.port` (et tout proxy terminant TLS).
3. Vérifiez le `status` du message et que le fichier `.mms` existe à `file_path`.

Le MMS hors réseau n'est pas relayé

Symptômes : Le MMS vers un autre opérateur est stocké mais jamais transféré ; le `status` reste `stored`.

Causes possibles :

- `default_route_mm4` n'est pas défini, donc aucun groupe partenaire n'est sélectionné.
- L'entrée partenaire dans `:mmsc_mm4_peers` a le mauvais hôte/port.
- La destination a été mal classée comme sur le réseau parce que `ims_domain`/`enum_domain` ne sont pas définis (tout est traité comme sur le réseau).

Résolution :

1. Définissez `default_route_mm4` sur une clé présente dans `:mmsc_mm4_peers`.
2. Vérifiez l'`hôte`/`port` partenaire et que l'envoi SMTP vers celui-ci est autorisé.
3. Configurez `enum_domain` et `ims_domain` afin que les destinations hors réseau soient correctement classées.

MM4 entrant rejeté ou non reçu

Symptômes : Un opérateur partenaire ne peut pas livrer via MM4.

Causes possibles :

- `mm4_enabled` est `false`, donc le serveur SMTP n'est pas en cours d'exécution.
- Le message ne nomme aucun destinataire dans l'enveloppe SMTP ou les en-têtes `To`/`Cc`, donc il est refusé avec `554 no_recipients`.
- Le service n'a pas la permission de lier le port privilégié 25.
- Un pare-feu bloque le SMTP entrant.

Résolution :

1. Définissez `mm4_enabled: true` et redémarrez.
2. Accordez `CAP_NET_BIND_SERVICE` ou définissez un `mm4_smtp_port` non privilégié derrière un transfert de port.
3. Autorisez le trafic entrant vers `mm4_smtp_port` depuis les adresses de sortie du partenaire.

Échec de la résolution de l'adresse de l'expéditeur

Symptômes : Les soumissions provenant de terminaux non-locaux sont rejetées avec une erreur de recherche d'abonné.

Causes possibles :

- Ni `subscriber_ip_api_urls` ni `pcrf_api_url` / `hss_api_url` n'est défini, donc seul localhost se résout à un MSISDN de test.
- Aucun nœud HSS configuré ne détient une liaison IP pour le terminal (la session de données de l'UE n'est pas encore connue de l'HSS).
- Les points de terminaison HSS/PCRF sont inaccessibles ou renvoient des réponses non-200.

Résolution :

1. Configurez `subscriber_ip_api_urls` avec chaque nœud HSS (ou les deux `pcrf_api_url` et `hss_api_url`) pour la résolution de l'expéditeur en production.
2. Vérifiez que les points de terminaison sont accessibles et renvoient le mappage attendu abonné/MSISDN.

Guide de Traduction des Numéros SMS-C

← Retour à l'Index de Documentation

Vue d'ensemble

Le système de Traduction des Numéros SMS-C fournit une transformation flexible basée sur des regex des numéros de téléphone avant le routage. Les règles de traduction peuvent normaliser les numéros, ajouter des préfixes internationaux, formater les numéros pour des passerelles spécifiques et enchaîner plusieurs transformations. Les règles sont stockées dans Mnesia pour la persistance et peuvent être modifiées à l'exécution sans interruption de service.

Caractéristiques Clés

- **Correspondance basée sur le préfixe** : Correspondre les numéros par préfixe avant d'appliquer les transformations
- **Transformation basée sur regex** : Correspondance et remplacement puissants avec des groupes de capture
- **Filtrage par SMSC source** : Appliquer différentes traductions en fonction de l'origine du message
- **Évaluation basée sur la priorité** : Contrôler l'ordre des règles avec des priorités configurables (1-255)
- **Enchaînement des règles** : Continuer le traitement à travers plusieurs règles avec pr��vention des boucles
- **Transformations séparées pour appelant/appelé** : Transformation indépendante pour les numéros d'origine et de destination
- **Chargement de fichier de configuration** : Charger les règles initiales depuis `runtime.exs` au premier démarrage

- **Configuration à l'exécution** : Ajouter, modifier ou désactiver des règles sans redémarrer
- **Interface Web** : Interface CRUD complète pour la gestion des règles
- **Outil de simulation** : Tester la logique de traduction avec une évaluation étape par étape
- **Sauvegarde/Restauration** : Exporter et importer des configurations de traduction
- **Intégration pré-routage** : Traductions appliquées avant le routage pour des formats de numéro cohérents

Architecture

Modèle de Données

Chaque règle de traduction contient les champs suivants :

Champ	Type	Description	Requis
`rule_id`	entier	Identifiant unique auto-incrémenté	Oui (auto)
`calling_prefix`	chaîne/nil	Correspondance de préfixe pour le numéro appelant (nil = joker)	Non
`called_prefix`	chaîne/nil	Correspondance de préfixe pour le numéro appelé (nil = joker)	Non
`source_smsc`	chaîne/nil	Nom de la SMSC source (nil = joker)	Non
`calling_match`	chaîne/nil	Modèle regex pour correspondre au numéro appelant. Agit également comme le **gate** pour la substitution TON/NPI source	Non
`calling_replace`	chaîne/nil	Modèle de remplacement pour le numéro appelant	Non
`called_match`	chaîne/nil	Modèle regex pour correspondre au numéro appelé	Non
`called_replace`	chaîne/nil	Modèle de remplacement pour le numéro appelé	Non
`source_ton`	chaîne/nil	Forcer le **type** de numéro de l'adresse source. Vide/absent = laisser inféré	Non
`source_npi`	chaîne/nil	Forcer l'indicateur de plan de numérotation de l'adresse source. Vide/absent = laisser inféré	Non

Champ	Type	Description	Requis
`priority`	entier	Priorité de la règle (1-255, plus bas = plus haute priorité)	Oui
`description`	chaîne	Description lisible par l'homme	Non
`enabled`	booléen	Activer/désactiver la règle	Oui
`continue`	booléen	Continuer à évaluer les règles après correspondance (par défaut : faux)	Non

Remarque : Les règles sont évaluées dans l'ordre de priorité (le plus petit numéro en premier). Seules les règles activées sont évaluées.

Voir Forçage du Type de Numéro (TON) et de l'Indicateur de Plan de Numérotation (NPI) pour la documentation complète des champs `source_ton` et `source_npi`.

Algorithme de Traduction

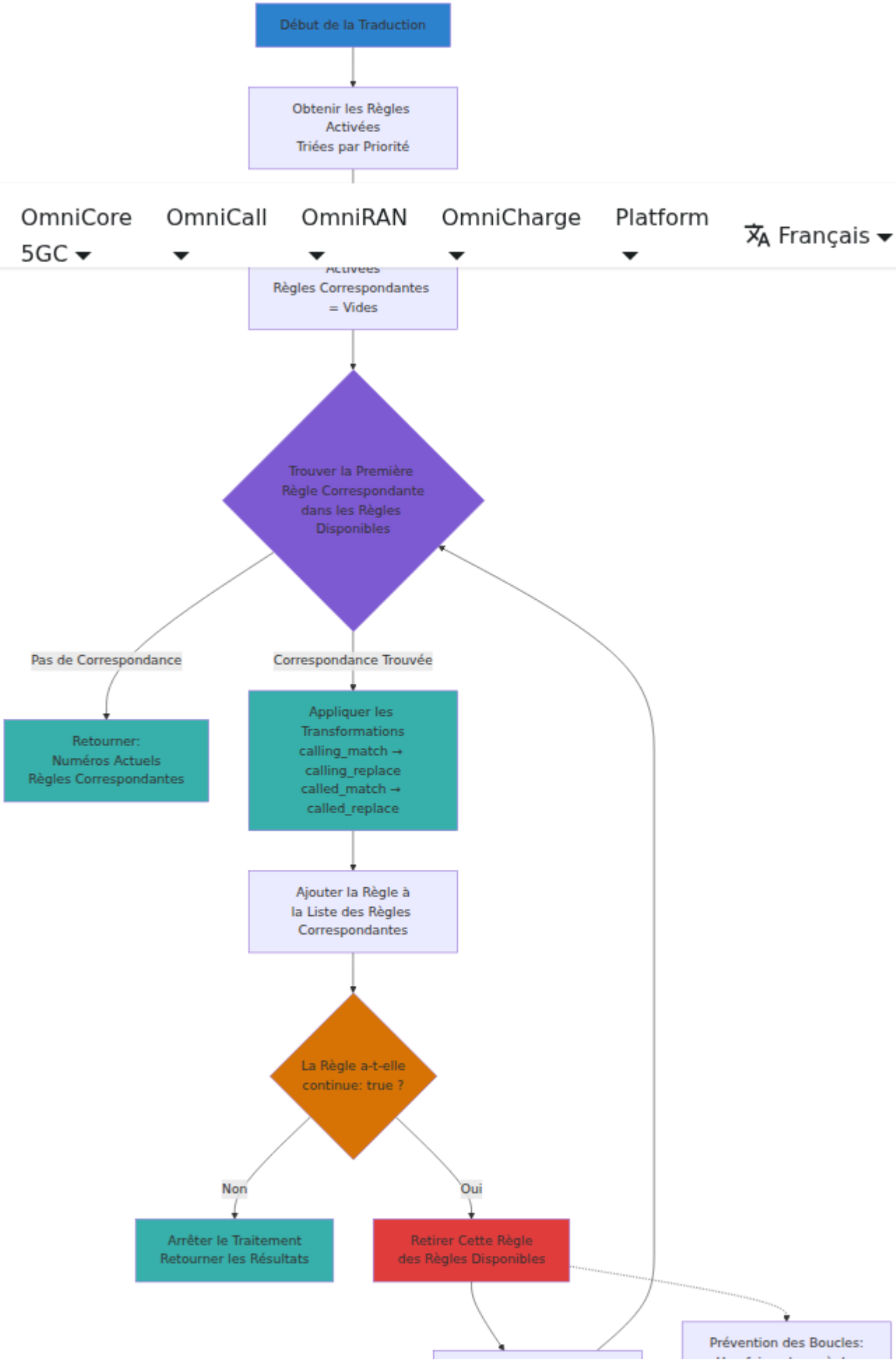
Lors de la traduction des numéros, le système :

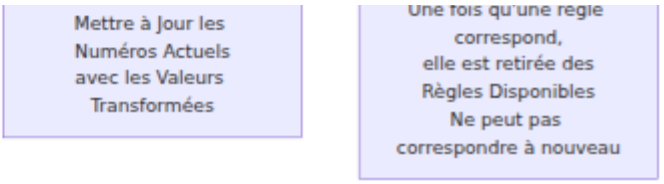
1. **Récupère les règles activées** triées par priorité (les plus basses en premier)
2. **Évalue les règles séquentiellement** par rapport aux paramètres du message :
   - Correspondre `calling_prefix` (si spécifié)
   - Correspondre `called_prefix` (si spécifié)
   - Correspondre `source_smsc` (si spécifié)
3. **Applique la première règle correspondante** :
   - Transformer le numéro appelant en utilisant `calling_match` et `calling_replace`
   - Transformer le numéro appelé en utilisant `called_match` et `called_replace`

4. **Vérifie le drapeau continue** :
    - Si `continue: false` → Arrêter le traitement, retourner le résultat
    - Si `continue: true` → Retirer la règle correspondante des règles disponibles, continuer avec l'étape 2 en utilisant **les numéros transformés**
5. **Retourne les numéros finaux** et la liste de toutes les règles appliquées

Enchaînement des Règles avec Prévention des Boucles

Le drapeau `continue` permet un puissant enchaînement de règles tout en prévenant les boucles infinies :


Début de la Traduction
Obtenir les Règles Activées Triées par Priorité
Règles Correspondantes = Vides
Trouver la Première Règle Correspondante dans les Règles Disponibles
Pas de Correspondance
Correspondance Trouvée
Retourner: Numéros Actuels Règles Correspondantes
Appliquer les Transformations calling_match → calling_replace called_match → called_replace
Ajouter la Règle à la Liste des Règles Correspondantes
La Règle a-t-elle continue: true ?
Non
Oui
Arrêter le Traitement Retourner les Résultats
Retirer Cette Règle des Règles Disponibles
Prévention des Boucles:



Jokers

- `nil` ou des valeurs vides agissent comme des jokers qui correspondent à n'importe quelle valeur
- Une règle sans critères de correspondance est une règle de rattrapage
- Une règle sans modèles de transformation (correspondance/remplacement nil) passe les numéros sans changement

Exemple : Scénario d'Enchaînement de Règles

Parse error on line 20: ...] style R1 fill:#38B2AC style R ----------^
Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW', 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW', 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

Réessayer

Forçage du Type de Numéro (TON) et de l'Indicateur de Plan de Numérotation (NPI)

En plus de la réécriture des numéros basée sur regex, une règle de traduction peut **forcer** le Type de Numéro (TON) et l'Indicateur de Plan de Numérotation (NPI) de l'adresse source (d'origine). Cela est utile lorsque la chaîne de numéro de l'expéditeur ne transmet pas seule la sémantique d'adresse correcte. Par exemple, un code court doit être signalé comme `unknown`/`unknown` plutôt que traité comme un numéro national ou international.

Ces substitutions sont indépendantes de la réécriture des numéros : une règle peut forcer TON/NPI **sans** changer le numéro du tout.

Les Champs de Substitution

Champ	S'applique à	Effet
`source_ton`	Adresse source (d'origine)	Force le Type de Numéro sur l'adresse d'origine SMS-DELIVER construite (le TPDU)
`source_npi`	Adresse source (d'origine)	Force l'Indicateur de Plan de Numérotation sur l'adresse d'origine SMS-DELIVER construite (le TPDU)

Vide ou absent = pas de substitution. Lorsqu'un champ est laissé vide (ou que la règle est complètement absente), la valeur de l'adresse est **inférée** à la place - voir Retour d'Inférence ci-dessous.

> **Remarque :** `source_ton`/`source_npi` sont intégrés directement dans l'adresse d'origine encodée dans le **TPDU** (le SMS-DELIVER que le frontend IMS/SS7 transmet). La valeur vit dans le TPDU encodé - rien de plus n'est stocké dans l'enregistrement du message. Il n'y a pas de substitution TON/NPI de destination : le TPDU SMS-DELIVER ne transporte pas d'adresse de destination, et les frontends dérivent eux-mêmes l'adressage de destination.

Valeurs Autorisées

Type de Numéro (TON) :

Valeur	Signification
`unknown`	Type de numéro inconnu
`international`	Numéro international (E.164 complet avec code pays)
`national`	Numéro national (significatif)
`network_specific`	Numéro spécifique au réseau
`subscriber`	Numéro d'abonné
`alphanumeric`	Adresse alphanumérique (par exemple, un ID d'expéditeur de marque)
`abbreviated`	Numéro abrégé

Indicateur de Plan de Numérotation (NPI) :

Valeur	Signification
`unknown`	Plan de numérotation inconnu
`isdn`	Plan de numérotation ISDN/téléphonie (E.164)
`data`	Plan de numérotation de données (X.121)
`telex`	Plan de numérotation Telex
`land_mobile`	Plan de numérotation mobile terrestre (E.212)
`national`	Plan de numérotation national
`private`	Plan de numérotation privé
`ermes`	Plan de numérotation ERMES

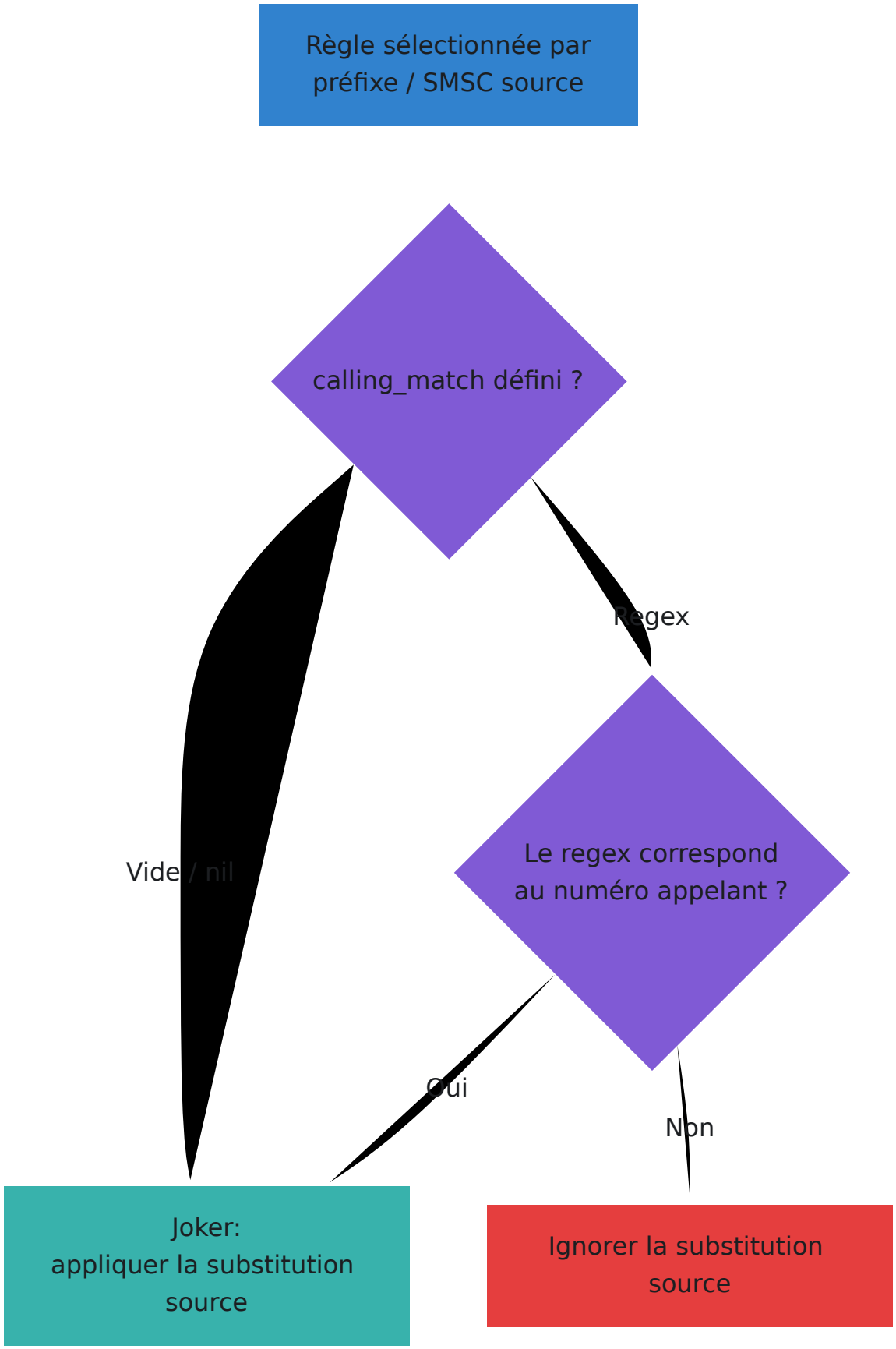
Toute valeur en dehors de ces ensembles est rejetée lors de la création ou de la mise à jour de la règle.

Gating de Correspondance : Scoping d'une Substitution à un Modèle d'Expéditeur

La substitution est **gated** par le regex **Correspondance de Numéro Appelant (`calling_match`)** de la règle, ce qui permet à une seule règle de cibler un sous-ensemble d'expéditeurs par motif :

- La substitution source (`source_ton`/`source_npi`) est appliquée uniquement lorsque le regex `calling_match` de la règle correspond au numéro appelant.
- Une **correspondance vide est un joker** - la substitution s'applique chaque fois que la règle est autrement sélectionnée (par préfixe et/ou SMSC source).

Une règle peut porter un `calling_match` sans correspondance `calling_replace` correspondante. Dans cette configuration, la correspondance agit purement comme un **gate** pour la substitution et ne réécrit **pas** le numéro. C'est la manière recommandée d'appliquer une substitution TON/NPI uniquement à ces numéros correspondant à un motif.



Lorsque `continue: true` enchaîne plusieurs règles, les substitutions TON/NPI s'accumulent avec une sémantique de **dernier écrit gagne** : une règle

ultérieure qui définit une valeur non vide écrase une précédente ; une valeur vide ne supprime jamais une substitution précédemment définie.

Retour d'Inférence

Lorsque **aucune** règle ne force une substitution, les valeurs d'adresse source sont **inférées** à partir de la chaîne de numéro au moment de la construction du TPDU :

Entrée	TON Inféré	NPI Inféré
Numérique, ≥ 5 chiffres	`international`	`isdn`
Numérique, < 5 chiffres	`national`	`isdn`
Non numérique (expéditeur alphanumérique, par exemple `Acme`)	`alphanumeric`	`unknown`

Une substitution forcée prend toujours le pas sur la valeur inférée. Forcer uniquement le TON (laisser le NPI vide) remplace le TON tout en laissant le NPI inféré, et vice versa.

Définition des Substitutions TON/NPI

Ces champs sont définissables via trois interfaces :

- **Interface Web** - l'éditeur de règles de Traduction de Numéros expose des menus déroulants pour le TON Source et le NPI Source. Laisser un menu déroulant non défini signifie "pas de substitution".
- **API REST** - les points de terminaison `/api/translations` acceptent `source_ton` et `source_npi` en tant que valeurs de chaîne (voir Exemples ci-dessous).
- **Configuration** - la liste `config :sms_c, :translation_rules` accepte les mêmes champs en tant que valeurs de chaîne, chargées au premier démarrage.

Exemple de Fonctionnement : Expéditeurs de Codes Courts

Un expéditeur qui est un code court de 3 à 6 chiffres doit être signalé comme TON/NPI `unknown`/`unknown` **sans** réécrire le numéro. La règle utilise `calling_match` uniquement comme un gate (pas de `calling_replace`) :

Configuration (`config/runtime.exs`) :

```
config :sms_c, :translation_rules, [
  %{
    calling_match: "^\d{3,6}$",
    calling_replace: nil,
    source_ton: "unknown",
    source_npi: "unknown",
    priority: 5,
    continue: true,
    description: "Codes courts (3-6 chiffres) => TON/NPI
inconnu/inconnu",
    enabled: true
  }
]
```

Comment ça fonctionne : La règle est évaluée pour chaque message (aucun `calling_prefix`, `called_prefix`, ou `source_smsc` ne la restreint). Le regex `calling_match` `^\d{3,6}$` gate la substitution source, donc `source_ton`/`source_npi` sont forcés à `unknown`/`unknown` **uniquement** lorsque le numéro appelant est une simple chaîne de 3 à 6 chiffres. Comme `calling_replace` est `nil`, le numéro lui-même reste inchangé. `continue: true` permet également à d'autres règles de s'exécuter.

Cas d'utilisation : Trafic d'origine d'application provenant de codes courts qui seraient autrement mal inférés comme des numéros `nationaux` (4 chiffres) ou `internationaux` (5 à 6 chiffres).

Requête API REST équivalente - `POST /api/translations` :

```
{
  "calling_match": "^\d{3,6}$",
  "calling_replace": null,
  "source_ton": "unknown",
  "source_npi": "unknown",
  "priority": 5,
  "continue": true,
  "description": "Codes courts opérateurs (3-6 chiffres) =>
TON/NPI inconnu/inconnu",
  "enabled": true
}
```

L'API retourne la règle créée (y compris son `rule_id` attribué) avec les valeurs TON/NPI sérialisées en tant que chaînes.

Exemple de Fonctionnement : Forcer National pour une Plage Connue

Forcer l'adresse source d'une plage de numéros locaux connue à `national`/`isdn`, sans réécrire le numéro :

Configuration (`config/runtime.exs`) :

```
%{
  calling_prefix: "07",
  calling_match: "^07\d{8}$",
  calling_replace: nil,
  source_ton: "national",
  source_npi: "isdn",
  priority: 20,
  continue: false,
  description: "Numéros locaux 07xxxxxxx => national/isdn",
  enabled: true
}
```

Comment ça fonctionne : `calling_prefix: "07"` restreint la règle aux numéros commençant par `07`, et le regex `calling_match` gate la substitution à exactement un `07` suivi de 8 chiffres. Lorsqu'il est correspondue, l'adresse source est signalée comme `national`/`isdn` plutôt que l'inféré

`international`/`isdn` (qui s'appliquerait parce que le numéro a ≥ 5 chiffres). Le numéro n'est pas réécrit.

Cas d'utilisation : Une passerelle ou un pair qui exige que les numéros locaux formatés nationalement soient signalés comme TON `national` même s'ils sont stockés sous leur forme de numérotation brute.

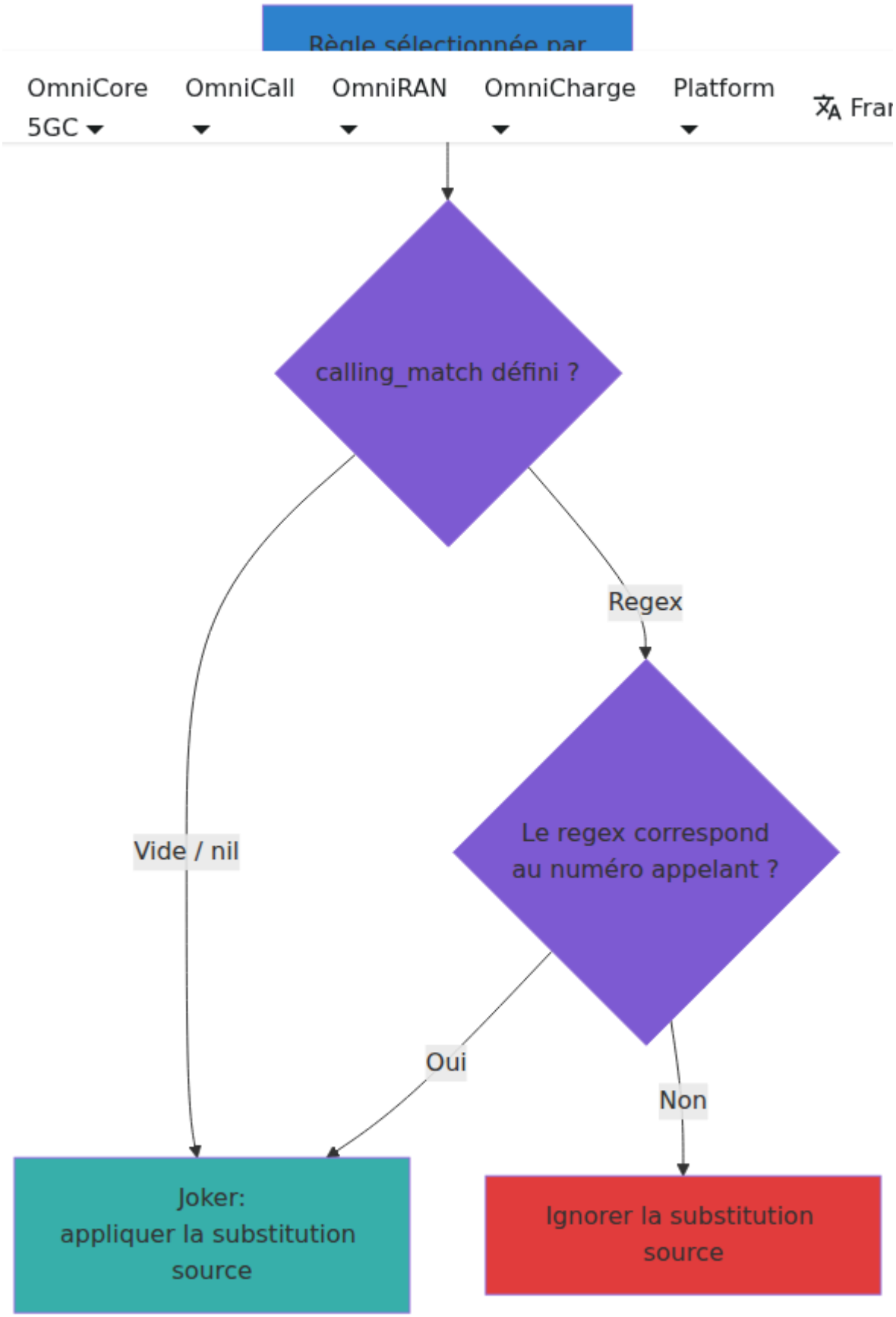
Configuration

Chargement des Règles depuis le Fichier de Configuration

Les règles de traduction peuvent être définies dans `config/runtime.exs` et seront automatiquement chargées au premier démarrage.

Important : Les règles de configuration ne sont chargées que lorsque la table de traduction est **vide** (premier démarrage). Cela préserve les règles ajoutées via l'interface Web pendant l'exécution et empêche les doublons lors des redémarrages.

Flux de Chargement de Configuration



Exemple de Configuration

```
# config/runtime.exs
config :sms_c, :translation_rules, [
  # Ajouter +1 aux numéros américains de 10 chiffres
  %{
    calling_prefix: nil,
    called_prefix: nil,
    source_smsc: "us_domestic_smsc",
    calling_match: "^(\d{10})$",
    calling_replace: "+1\1",
    called_match: "^(\d{10})$",
    called_replace: "+1\1",
    priority: 10,
    description: "Ajouter +1 aux numéros américains de 10 chiffres
provenant de la SMSC domestique",
    enabled: true,
    continue: false
  },

  # Supprimer les zéros initiaux du format international
  %{
    calling_prefix: "00",
    called_prefix: nil,
    source_smsc: nil,
    calling_match: "^00(.+)$",
    calling_replace: "+\1",
    called_match: nil,
    called_replace: nil,
    priority: 5,
    description: "Convertir le préfixe international 00 en +",
    enabled: true,
    continue: true  # Continuer à appliquer plus de formatage
  },

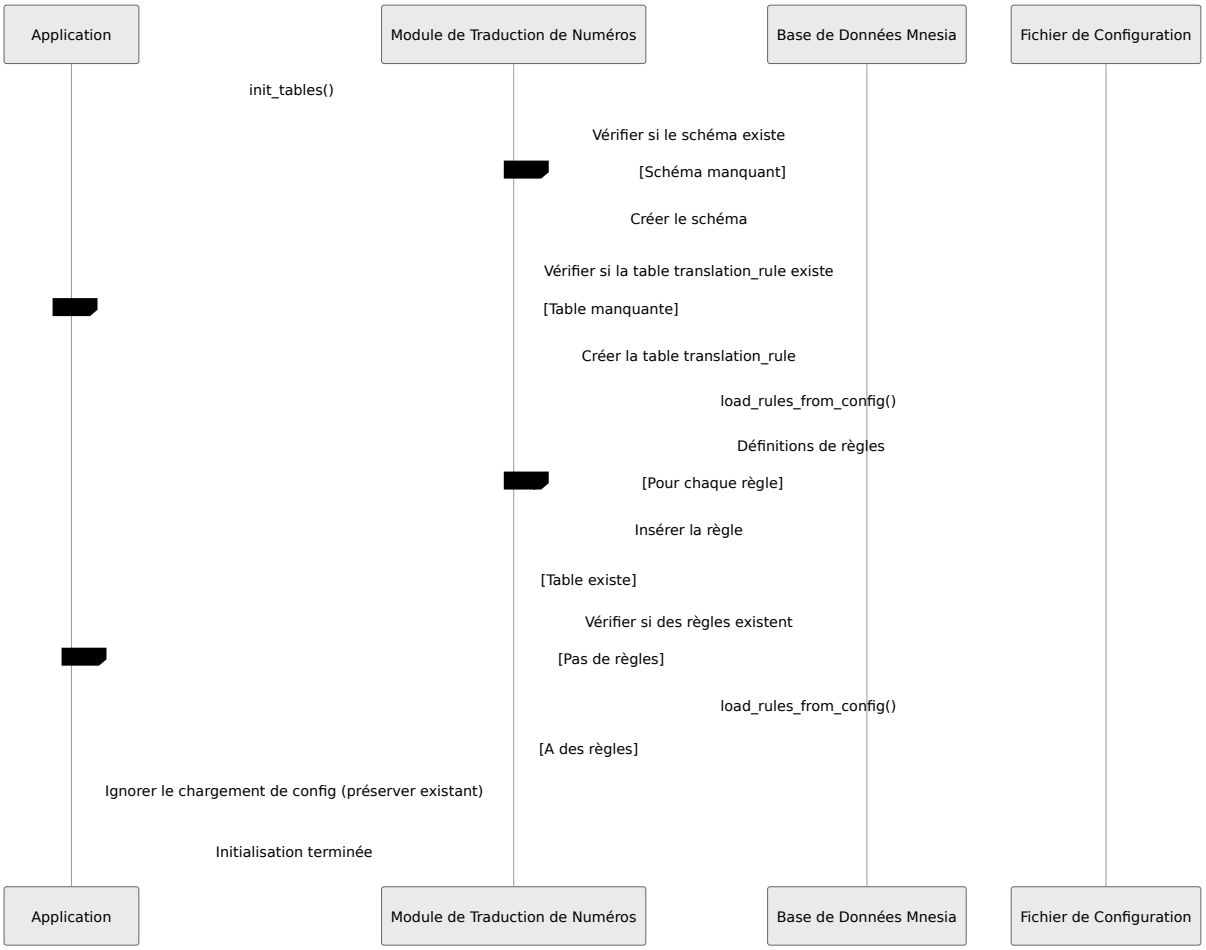
  # Forcer les expéditeurs de codes courts à TON/NPI
inconnu/inconnu (gate uniquement, pas de réécriture)
  %{
    calling_match: "^\d{3,6}$",
    calling_replace: nil,
    source_ton: "unknown",
    source_npi: "unknown",
    priority: 6,
    description: "Codes courts (3-6 chiffres) => TON/NPI
inconnu/inconnu",
```

```
    enabled: true,
    continue: true
  },

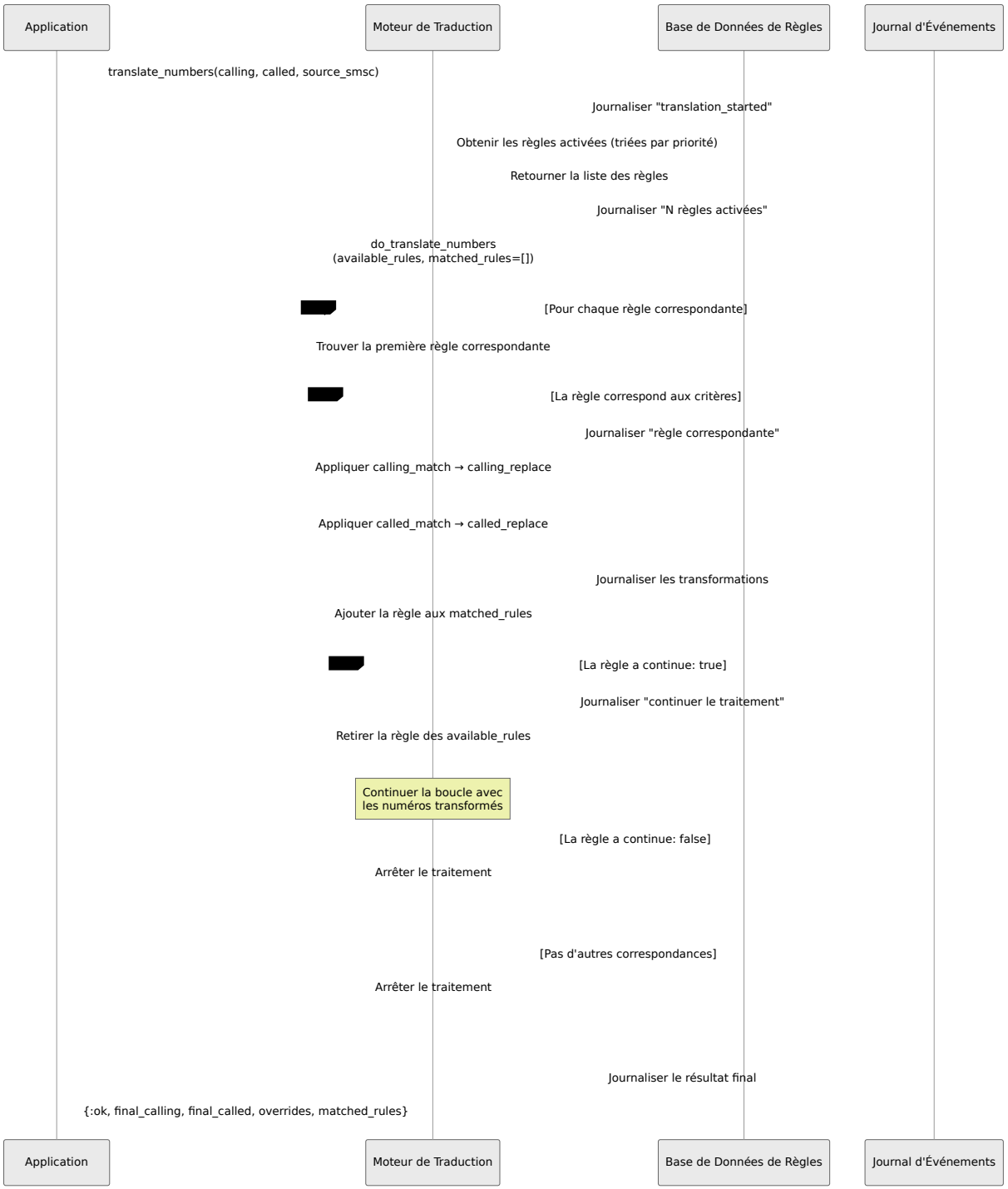
  # Formater les numéros du Royaume-Uni pour une passerelle
spécifique
  %{
    calling_prefix: "+44",
    called_prefix: "+44",
    source_smsc: nil,
    calling_match: "^\+44(.*)$",
    calling_replace: "0044\1",
    called_match: "^\+44(.*)$",
    called_replace: "0044\1",
    priority: 20,
    description: "Formater les numéros du Royaume-Uni pour une
passerelle héritée",
    enabled: true,
    continue: false
  }
]
```

Démarrer

Flux d'Initialisation



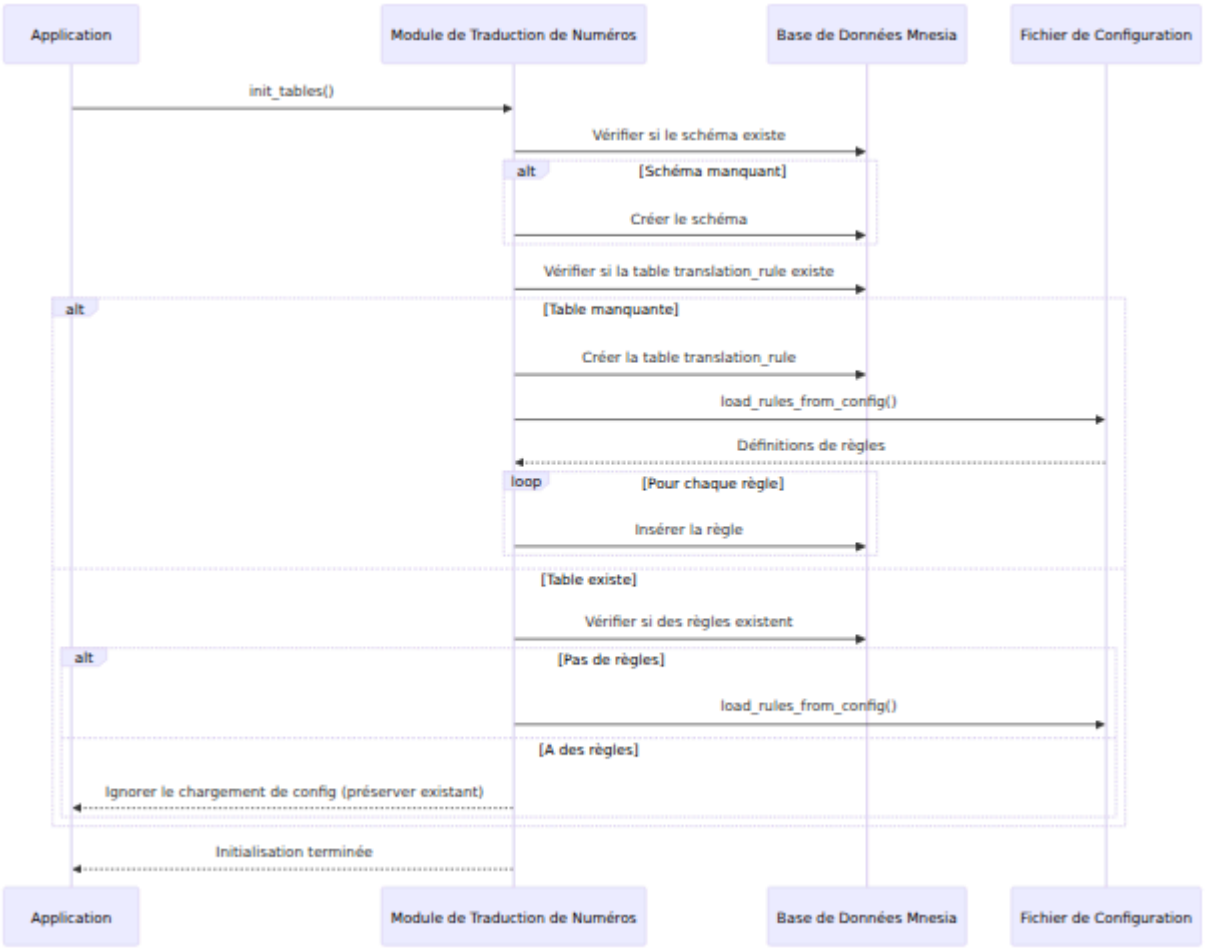
Flux de Traduction de Message



Cas d'Utilisation Courants

Normalisation des Numéros Internationaux

Normaliser divers formats internationaux en E.164 :



Formatage Spécifique à la Passerelle

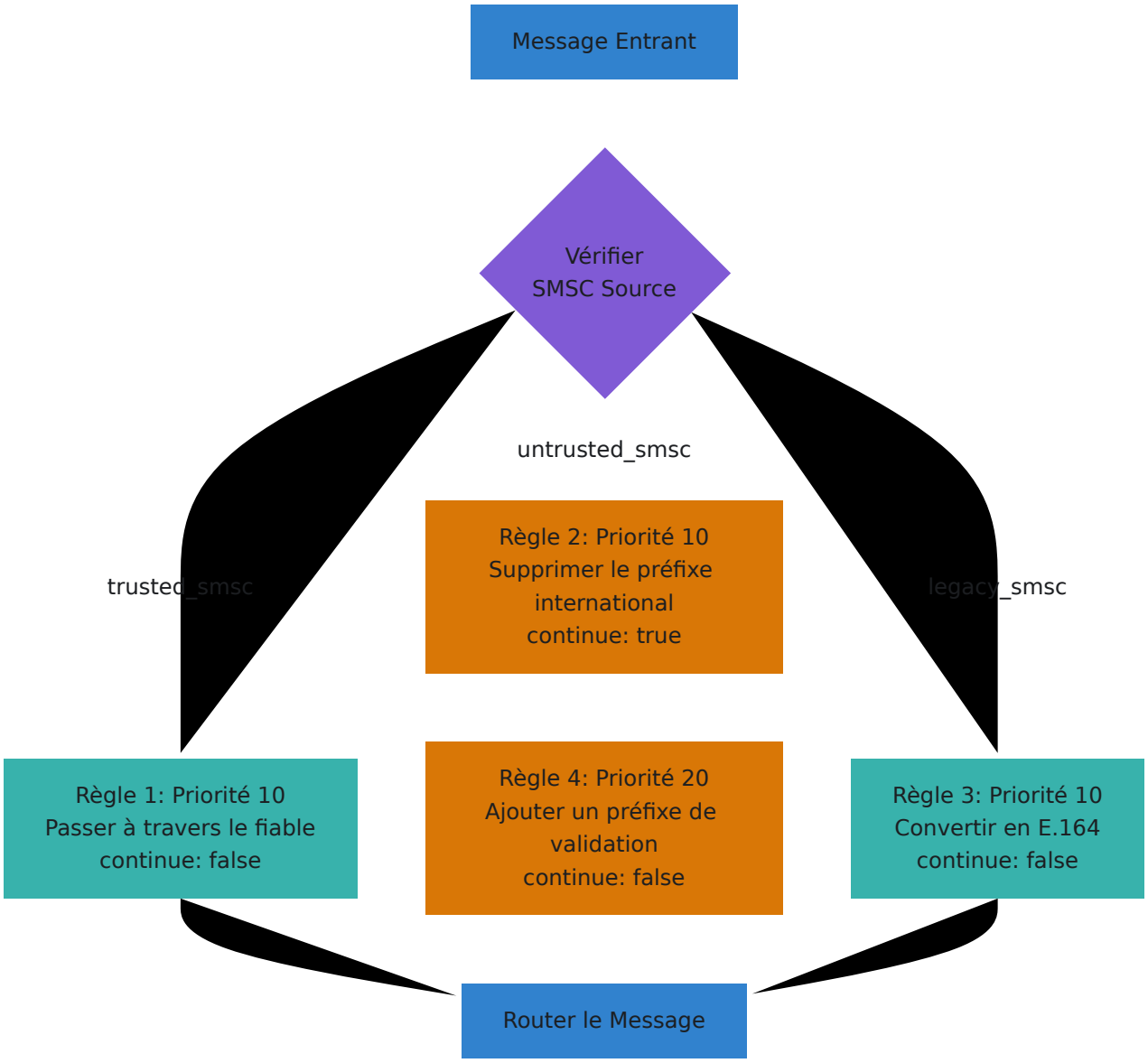
Enchaîner des règles pour formater des numéros selon les exigences spécifiques de la passerelle :

Parse error on line 2: ...t TD I[Entrée: "5551234567"] --> S1[----------^ Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND', 'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND_STOP', 'TAGEND', 'TRAPEND', 'INVTRAPEND', 'UNICODE_TEXT', 'TEXT', 'TAGSTART', got 'STR'

Réessayer

Traductions Spécifiques à la SMSC

Appliquer différentes traductions en fonction de la source du message :



Préparation au Routage Basée sur le Préfixe

Normaliser les numéros avant le routage pour garantir une correspondance de préfixe cohérente :



Gestion de la Portabilité des Numéros

Gérer les numéros portés qui nécessitent des changements de préfixe :

Parse error on line 18: ... style Input fill:#3182CE style R ----------^
Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW', 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW',

'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

Réessayer

Interface Web

Interface de Gestion des Règles de Traduction

Accédez à l'interface de gestion des règles à `/number_translation` (via le menu de navigation) :

Fonctionnalités :

- Voir toutes les règles dans un tableau triable par priorité
- Ajouter de nouvelles règles avec validation de formulaire
- Modifier des règles existantes
- Activer/désactiver des règles sans les supprimer
- Supprimer des règles avec confirmation
- Indicateur visuel pour les règles avec `continue: true`
- Importer/Exporter des règles au format JSON

Ajouter une Règle :

1. Remplissez les critères de correspondance (facultatif) :
   - Préfixe appelant (par exemple, "+1", "44")
   - Préfixe appelé (par exemple, "+639", "1555")
   - SMSC source (laisser vide pour n'importe quel)
2. Définir les transformations (facultatif) :
   - Correspondance regex du numéro appelant et remplacement
   - Correspondance regex du numéro appelé et remplacement
3. Définir la substitution d'adresse source (facultatif) :
   - Menus déroulants **Source TON / Source NPI** - forcer le Type de Numéro de l'adresse d'origine / Indicateur de Plan de Numérotation

(intégré dans le TPDU). Gate par le regex de Correspondance de Numéro Appelant.
    - Laisser un menu déroulant non défini pour laisser la valeur inférée. Voir Forçage du Type de Numéro (TON) et de l'Indicateur de Plan de Numérotation (NPI).
4. Définir la priorité (1-255, plus bas = plus haute priorité)
5. Définir le statut :
    - **Activé** : La règle est active
    - **Continuer le Traitement** : Continuer à évaluer plus de règles après celle-ci
6. Ajouter une description
7. Cliquez sur "Ajouter une Règle" ou "Mettre à Jour la Règle"

Basculer le Traitement Continu :

- **Arrêter** (par défaut) : Arrêter le traitement après que cette règle correspond
- **Continuer** : Appliquer cette règle et continuer à évaluer les règles restantes
- Les règles avec continue activé affichent un badge bleu "↓ Continuer" dans le tableau

Modifier une Règle :

1. Cliquez sur "Modifier" à côté de la règle
2. Modifiez les champs selon les besoins
3. Cliquez sur "Mettre à Jour la Règle"

Indicateurs du Tableau des Règles :

- Le badge **Activé/Désactivé** montre le statut de la règle
- Le badge **↓ Continuer** montre les règles qui continueront le traitement
- Le badge **Priorité** montre l'ordre d'évaluation
- Les motifs regex sont affichés en police monospace pour plus de clarté

Simulateur de Traduction

Accédez au simulateur à `/translation_simulator` (via le menu de navigation) :

Fonctionnalités :

- Tester la logique de traduction avec des numéros réels
- **Transformation Étape par Étape** montrant chaque règle appliquée
- Voir les valeurs avant/après pour chaque transformation
- Voir quelles règles ont correspondu et pourquoi
- Charger des scénarios d'exemple pour des tests rapides
- Voir l'historique des tests (derniers 10 tests)

Utiliser le Simulateur :

1. Entrez les paramètres de test :
    - Numéro appelant (de)
    - Numéro appelé (à)
    - SMSC source (facultatif)
2. Cliquez sur "Tester la Traduction"
3. Voir les résultats complets :
    - **Résultat de Traduction** : Numéros finaux après toutes les transformations
    - **Règles Appliquées** : Compte et liste de toutes les règles qui ont correspondu
    - **Transformations Étape par Étape** : Vue détaillée de chaque règle :
        - Numéro d'étape et informations sur la règle
        - Description de la règle
        - Avant → Après pour les numéros appelants et appelés
        - Indicateur "↓ Continuer" pour les règles qui ont continué le traitement
        - Transformations mises en surbrillance en vert
        - Valeurs inchangées marquées comme "passées à travers"
4. Charger des exemples préconfigurés en utilisant les boutons d'exemple
5. Examiner l'historique des tests pour comparer différents scénarios

Exemple de Sortie :

```
Résultat de Traduction
═══════════════════════════════════════════════
Numéro Appelant:  5551234567 → +1-555-123-4567
Numéro Appelé:   5558720155 → +1-555-872-0155
✓ Traduit par 3 règle(s)

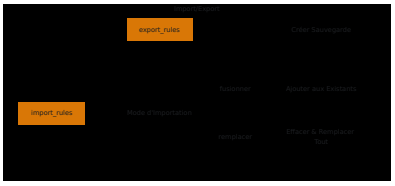
Transformations Étape par Étape
═══════════════════════════════════════════════
┌─ Étape 1 ──────────────────────────────────┐
│ Règle #1 (Priorité 10)         ↓ Continuer │
│ Ajouter le code pays aux numéros de 10 chiffres     │
│                                            │
│ Appelé: 5558720155 → +15558720155       │
└────────────────────────────────────────────┘

┌─ Étape 2 ──────────────────────────────────┐
│ Règle #2 (Priorité 20)         ↓ Continuer │
│ Formater l'indicatif régional avec des tirets             │
│                                            │
│ Appelé: +15558720155 → +1-555-8720155   │
└────────────────────────────────────────────┘

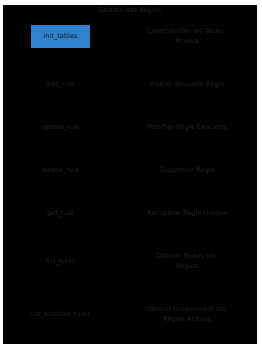
┌─ Étape 3 ──────────────────────────────────┐
│ Règle #3 (Priorité 30)                     │
│ Formatage final pour la passerelle             │
│                                            │
│ Appelé: +1-555-8720155 → +1-555-872-0155│
└────────────────────────────────────────────┘
```

Référence API

Vue d'ensemble des Opérations Principales







Paramètres de Traduction

translate_numbers accepte les paramètres suivants :

- `calling_number` (facultatif) : Numéro de téléphone d'origine
- `called_number` (facultatif) : Numéro de téléphone de destination
- `source_smsc` (facultatif) : Identifiant SMSC source
- `message_id` (facultatif) : Pour la journalisation des événements

Retourne :

- `{:ok, translated_calling, translated_called, overrides, [rules_applied]}` - Toujours réussi
- `overrides` est une carte `%{source_ton:, source_npi:}` (chacun `nil` lorsque aucune règle correspondante ne l'a défini ; dernier écrit gagne lorsque plusieurs règles définissent une valeur)
- Retourne les numéros originaux et des substitutions vides si aucune règle ne correspond
- Retourne la liste de toutes les règles qui ont été appliquées (dans l'ordre)

```
# Exemple d'utilisation
{:ok, new_calling, new_called, _overrides, rules} =
  NumberTranslation.translate_numbers(
    calling_number: "5551234567",
    called_number: "5558720155",
    source_smsc: "domestic_gateway",
    message_id: "msg_123"
  )

# Vérifier si une traduction a eu lieu
if rules != [] do
  Logger.info("Appliqué #{length(rules)} règles de traduction")
  Enum.each(rules, fn rule ->
    Logger.info("  - Règle ##{rule.rule_id}: #{rule.description}")
  end)
end
```

Opérations de Gestion des Règles

```
# Ajouter une nouvelle règle
{:ok, rule} = NumberTranslation.add_rule(%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: "gateway1",
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 10,
  description: "Ajouter +1 aux numéros de 10 chiffres",
  enabled: true,
  continue: false
})

# Mettre à jour une règle
{:ok, updated_rule} = NumberTranslation.update_rule(rule_id, %{
  enabled: false,
  description: "Désactivé pour test"
})

# Supprimer une règle
:ok = NumberTranslation.delete_rule(rule_id)

# Obtenir une règle spécifique
rule = NumberTranslation.get_rule(rule_id)

# Lister toutes les règles
all_rules = NumberTranslation.list_rules()

# Lister uniquement les règles activées (triées par priorité)
enabled_rules = NumberTranslation.list_enabled_rules()
```

Opérations d'Import/Export

```
# Exporter toutes les règles
backup = NumberTranslation.export_rules()
# Retourne : %{
#   version: "1.0",
#   exported_at: ~U[2024-01-15 10:30:00Z],
#   count: 5,
#   rules: [...]
# }

# Sauvegarder dans un fichier JSON
json = Jason.encode!(backup, pretty: true)
File.write!("translation_rules_backup.json", json)

# Importer des règles (fusionner avec les existantes)
{:ok, %{imported: 3, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :merge)

# Importer des règles (remplacer toutes les existantes)
{:ok, %{imported: 5, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :replace)
```

Meilleures Pratiques

Conception des Règles

1. **Organiser les priorités** :
    - **1-10** : Règles de normalisation critiques (ajouter des codes pays, corriger les formats)
    - **11-50** : Formatage spécifique à la passerelle
    - **51-100** : Transformations optionnelles
    - **101+** : Règles de rattrapage ou de débogage
2. **Utiliser continue de manière stratégique** :

- Activer `continue: true` pour les règles de normalisation qui préparent les numéros pour un traitement ultérieur
- Désactiver `continue: false` pour les règles de formatage final
- Éviter les longues chaînes (maximum 3-4 règles) pour maintenir les performances

3. **Documenter vos règles** :

   - Toujours ajouter des descriptions claires
   - Inclure des exemples dans la description (par exemple, "5551234567 → +15551234567")
   - Documenter l'objectif et l'entrée/sortie attendus

4. **Tester les motifs regex** :

   - Tester les motifs avec le simulateur avant de déployer
   - Utiliser des groupes de capture (\1, \2) pour des transformations flexibles
   - Échapper les caractères spéciaux regex (points, parenthèses, etc.)

Performance

1. **Minimiser le nombre de règles** :

   - Combiner des règles similaires lorsque cela est possible
   - Utiliser la correspondance de préfixe pour réduire les évaluations regex
   - Supprimer ou désactiver les règles inutilisées

2. **Optimiser les motifs regex** :

   - Utiliser d'abord la correspondance de préfixe (plus rapide que regex)
   - Garder les motifs regex simples
   - Éviter les motifs lourds en backtracking

3. **Limiter l'enchaînement des règles** :

   - Les longues chaînes (5+ règles) peuvent impacter les performances
   - Envisager de combiner plusieurs étapes en une seule règle si possible

- Surveiller la latence de traduction avec des métriques Telemetry

Opérations

1. **Tester avant de déployer** :
   - Utiliser le simulateur avec des exemples du monde réel
   - Tester les cas limites (numéros vides, caractères spéciaux)
   - Vérifier le comportement du drapeau continue
2. **Sauvegarder régulièrement** :
   - Exporter les règles avant d'apporter des modifications majeures
   - Contrôler la version de vos exports
   - Tester les imports en non-production d'abord
3. **Surveiller les traductions** :
   - Activer la journalisation de message_id pour le débogage
   - Vérifier les journaux d'événements pour les décisions de traduction
   - Surveiller quelles règles sont appliquées
4. **Déploiement progressif** :
   - Ajouter de nouvelles règles comme désactivées d'abord
   - Tester avec le simulateur
   - Activer et surveiller
   - Ajuster si nécessaire

Conseils Regex

1. **Motifs courants** :
   - Numéro américain de 10 chiffres : `^(\d{10})$`
   - Format international : `^\+(\d+)$`
   - Supprimer les zéros initiaux : `^0+(.+)$`
   - Ajouter des tirets : `^(\d{3})(\d{3})(\d{4})$` → `\1-\2-\3`

2. **Groupes de capture** :
    - Utiliser des parenthèses pour capturer : `^(\d{3})(\d{7})$`
    - Référencer dans le remplacement : `+1\1\2`
    - Captures multiples : `^\+(\d{1,3})(\d+)$` → `00\1\2`
3. **Échapper les caractères spéciaux** :
    - Point littéral : `\.`
    - Plus littéral : `\+`
    - Parenthèse littérale : `\(` ou `\)`

Dépannage

Règle Non Correspondante

Symptôme : Règle attendue ne correspond pas, les numéros passent sans changement

Causes possibles :

- Le préfixe ne correspond pas (vérifier la correspondance exacte du préfixe)
- La SMSC source ne correspond pas
- Le motif regex ne correspond pas au format d'entrée
- La règle est désactivée
- Une règle de priorité supérieure a été correspondue en premier (avec continue: false)

Solutions :

1. Utiliser le simulateur pour voir quelles règles sont évaluées
2. Vérifier le statut de la règle (activée/désactivée)
3. Vérifier la correspondance de préfixe (sensible à la casse)
4. Tester le motif regex séparément
5. Vérifier l'ordre de priorité

Mauvaise Transformation Appliquée

Symptôme : Numéro transformé mais résultat incorrect

Causes possibles :

- Le motif regex correspond mais le motif de remplacement est incorrect
- Plusieurs règles s'appliquent dans un ordre inattendu
- Références de groupes de capture incorrectes (\1, \2, etc.)

Solutions :

1. Utiliser le simulateur pour voir les transformations étape par étape
2. Vérifier que le motif regex capture les bons groupes
3. Vérifier la syntaxe du motif de remplacement
4. Tester le regex dans un testeur de regex en ligne
5. Revoir la priorité des règles et les drapeaux continue

Boucle Infinie / Dégradation des Performances

Symptôme : La traduction prend beaucoup de temps ou semble se bloquer

Remarque : Cela ne devrait pas se produire en raison de la prévention des boucles, mais si cela se produit :

Causes possibles :

- Bug dans la logique de prévention des boucles
- Évaluation regex extrêmement longue
- Chaîne de règles très longue

Solutions :

1. Vérifier les journaux d'application pour des erreurs
2. Revoir les règles avec continue: true
3. Simplifier les motifs regex
4. Réduire le nombre de règles enchaînées
5. Signaler un bug si la prévention des boucles a échoué

Enchaînement de Règles Inattendu

Symptôme : Plus de règles appliquées que prévu

Causes possibles :

- Les règles ont continue: true alors qu'elles ne devraient pas
- L'ordre de priorité permet plusieurs correspondances
- Le numéro transformé correspond à des règles supplémentaires

Solutions :

1. Utiliser le simulateur pour voir la chaîne de règles exacte
2. Revoir les drapeaux continue sur toutes les règles
3. Ajuster les priorités pour contrôler l'ordre
4. Définir continue: false sur la règle finale

Traduction Non Appliquée Avant le Routage

Symptôme : Le routeur voit des numéros non traduits

Causes possibles :

- La traduction n'est pas intégrée dans le flux de message
- La traduction se produit après le routage
- Le code de l'application contourne la traduction

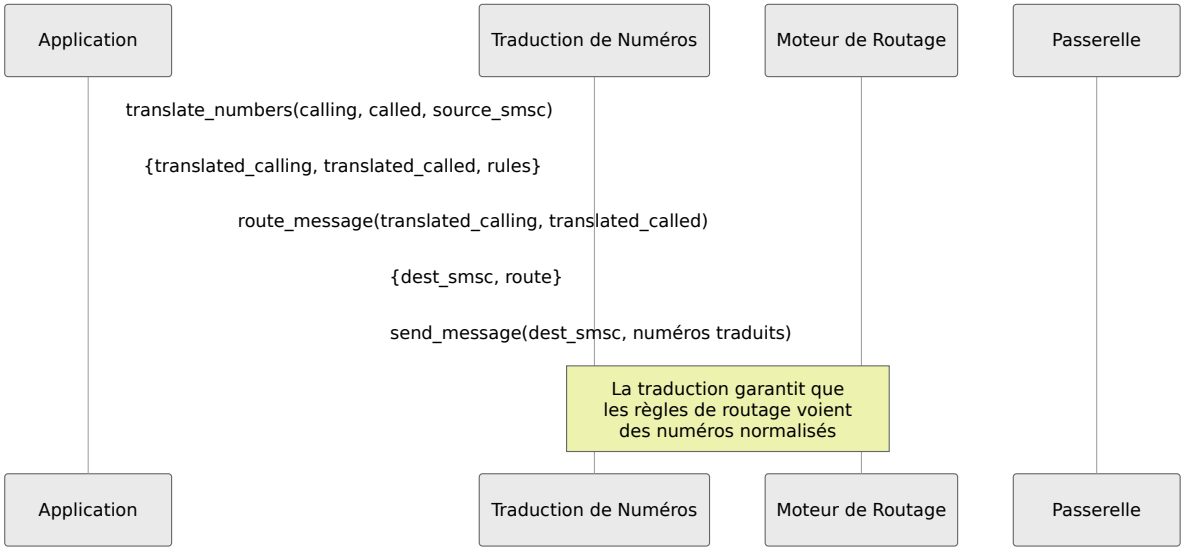
Solutions :

1. Vérifier l'intégration de l'application : la traduction doit être appelée avant le routage
2. Vérifier le pipeline de traitement des messages
3. Revoir les journaux d'événements pour les événements de traduction
4. S'assurer que translate_numbers est appelé dans le bon ordre

Sujets Avancés

Intégration avec le Routage

La traduction se produit **avant** le routage pour garantir des formats de numéro cohérents :



Journalisation des Événements

Les décisions de traduction sont journalisées via l'EventLogger :

- `translation_started` : La traduction commence
- `translation_candidates` : Nombre de règles activées
- `translation_matched` : Règle correspondante et appliquée
- `translation_calling` : Numéro appelant transformé
- `translation_called` : Numéro appelé transformé
- `translation_continue` : La règle a continue=true, continuation de l'évaluation
- `translation_none` : Aucune règle correspondante

Activer la journalisation en passant `message_id` à `translate_numbers/1`.

Métriques Telemetry

Surveiller les performances de traduction avec Telemetry :

```
:telemetry.attach(
  "number-translation-handler",
  [:sms_c, :number_translation, :translate, :stop],
  fn _event_name, measurements, metadata, _config ->
    # measurements: %{duration: microsecondes}
    # metadata: %{rules_applied: count, ...}
  end,
  nil
)
```

Métriques clés à surveiller :

- Durée de traduction (p50, p95, p99)
- Règles appliquées par message
- Règles correspondantes vs non correspondantes
- Utilisation du drapeau continue

Clustering

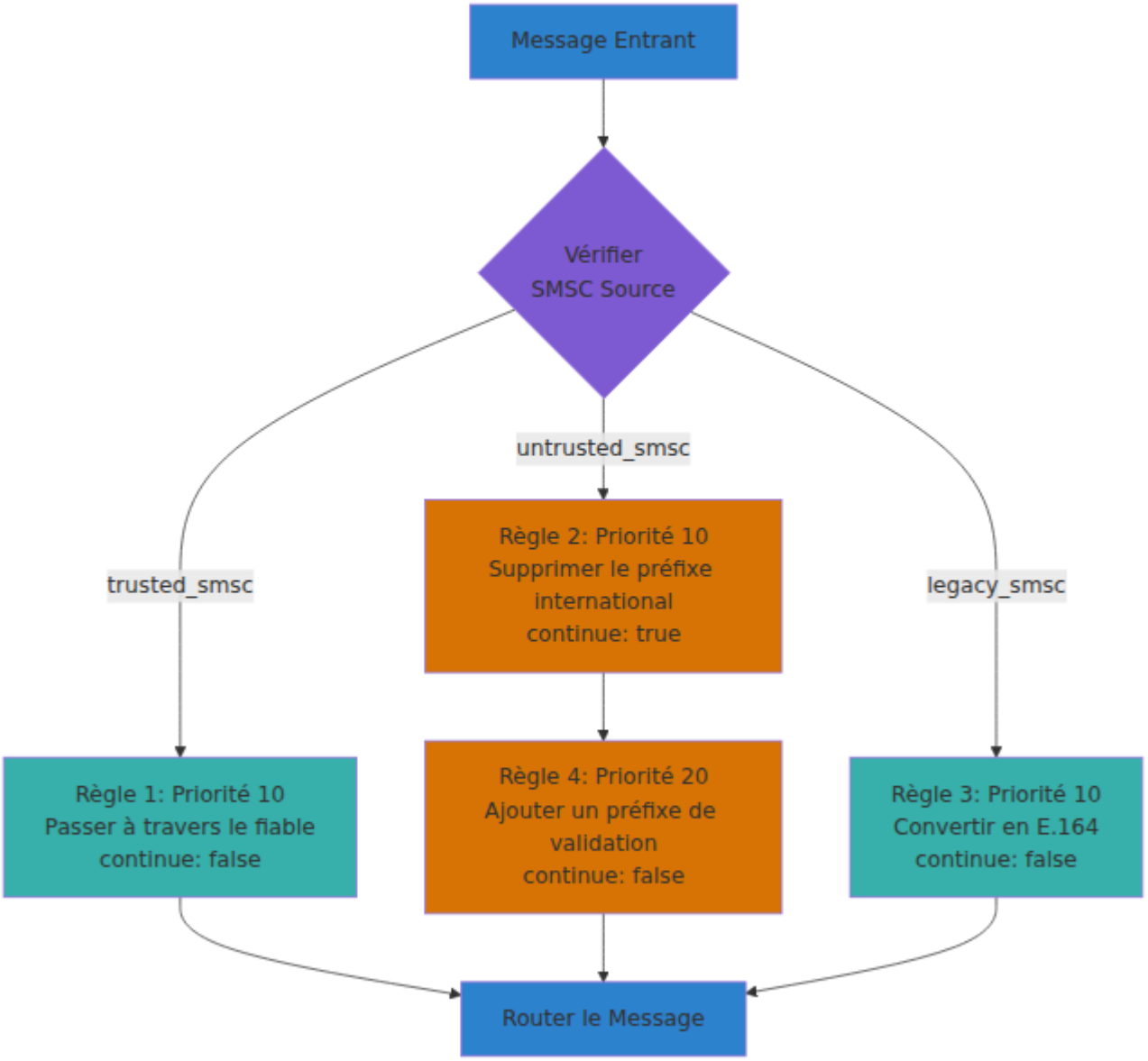
Les tables Mnesia sont automatiquement distribuées à travers les nœuds en cluster. Les règles de traduction sont répliquées pour une haute disponibilité.

Parse error on line 25: ... style New fill:#3182CE style P ----------^ Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW', 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW', 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

Réessayer

Stratégies de Migration

Lors du déploiement de nouvelles règles de traduction :



Exemples

Exemple 1 : Normalisation des Numéros Américains

Exigence : Convertir divers formats de numéros américains en E.164 (+1XXXXXXXXXX)

```
# Règle 1 : Numéros de 10 chiffres (priorité la plus élevée)
%{
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 5,
  description: "Ajouter +1 aux numéros de 10 chiffres nus",
  enabled: true,
  continue: false
}

# Règle 2 : 1 + 10 chiffres (priorité moyenne)
%{
  calling_match: "^1(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^1(\d{10})$",
  called_replace: "+1\1",
  priority: 10,
  description: "Convertir 1XXXXXXXXXX en +1XXXXXXXXXX",
  enabled: true,
  continue: false
}

# Cas de test :
# "5551234567" → "+15551234567" (Règle 1)
# "15551234567" → "+15551234567" (Règle 2)
# "+15551234567" → "+15551234567" (Pas de correspondance, passer à
travers)
```

Exemple 2 : Conversion de Préfixe International avec Enchaînement

Exigence : Convertir le préfixe 00 en +, puis formater pour la passerelle

```
# Règle 1 : Convertir 00 en + (continue vers la règle suivante)
%{
  calling_match: "^00(.+)$",
  calling_replace: "+\1",
  called_match: "^00(.+)$",
  called_replace: "+\1",
  priority: 5,
  description: "Convertir le préfixe international 00 en +",
  enabled: true,
  continue: true  # Continuer à formater
}

# Règle 2 : Formater pour la passerelle (arrête le traitement)
%{
  calling_match: "^\+(\d+)$",
  calling_replace: "00\1",
  called_match: "^\+(\d+)$",
  called_replace: "00\1",
  priority: 10,
  description: "Formater les numéros + comme 00 pour la
passerelle",
  enabled: true,
  continue: false  # Arrêter après cela
}

# Cas de test :
# Étape 1 : "00441234567890" → "+441234567890" (Règle 1,
continuer)
# Étape 2 : "+441234567890" → "00441234567890" (Règle 2, arrêter)
# Résultat : "00441234567890"
# Règles appliquées : [Règle 1, Règle 2]
```

Exemple 3 : Gestion Spécifique à la SMSC

Exigence : Appliquer différentes règles en fonction de la SMSC source

```
# Règle 1 : SMSC de confiance - passer à travers (priorité 5)
%{
  source_smsc: "trusted_gateway",
  calling_match: nil,  # Pas de transformation
  calling_replace: nil,
  called_match: nil,
  called_replace: nil,
  priority: 5,
  description: "Passer à travers les numéros de la passerelle de
confiance",
  enabled: true,
  continue: false
}

# Règle 2 : SMSC non fiable - normaliser (priorité 10)
%{
  source_smsc: "untrusted_gateway",
  calling_match: "^(.*)$",
  calling_replace: "+VALIDATE\1",
  called_match: "^(.*)$",
  called_replace: "+VALIDATE\1",
  priority: 10,
  description: "Ajouter un préfixe de validation pour la source
non fiable",
  enabled: true,
  continue: false
}

# Règle 3 : Rattrapage pour d'autres SMSCs (priorité 100)
%{
  source_smsc: nil,  # Joker
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 100,
  description: "Par défaut : Ajouter +1 aux numéros de 10
chiffres",
  enabled: true,
  continue: false
}
```

Exemple 4 : Chaîne de Formatage Multi-Étapes

Exigence : Normaliser → Ajouter le code pays → Formater avec des tirets

```
# Règle 1 : Supprimer les zéros initiaux (continue)
%{
  calling_match: "^0+(.+)$",
  calling_replace: "\1",
  called_match: "^0+(.+)$",
  called_replace: "\1",
  priority: 5,
  description: "Supprimer les zéros initiaux",
  enabled: true,
  continue: true
}

# Règle 2 : Ajouter le code pays si manquant (continue)
%{
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 10,
  description: "Ajouter +1 aux numéros de 10 chiffres",
  enabled: true,
  continue: true
}

# Règle 3 : Formater avec des tirets (arrête)
%{
  calling_match: "^\+1(\d{3})(\d{3})(\d{4})$",
  calling_replace: "+1-\1-\2-\3",
  called_match: "^\+1(\d{3})(\d{3})(\d{4})$",
  called_replace: "+1-\1-\2-\3",
  priority: 15,
  description: "Formater comme +1-XXX-XXX-XXXX",
  enabled: true,
  continue: false
}

# Cas de test :
# Entrée : "005551234567"
# Étape 1 : "005551234567" → "5551234567" (Règle 1, continuer)
# Étape 2 : "5551234567" → "+15551234567" (Règle 2, continuer)
# Étape 3 : "+15551234567" → "+1-555-123-4567" (Règle 3, arrêter)
```

```
# Résultat : "+1-555-123-4567"
# Règles appliquées : [Règle 1, Règle 2, Règle 3]
```

Support

Pour des problèmes ou des questions :

- Utilisez le simulateur pour travailler à travers des exemples fonctionnels et déboguer la logique de traduction
- Vérifiez les journaux d'événements pour les décisions de traduction
- Vérifiez le contenu de la table Mnesia : `:mnesia.table_info(:translation_rule, :size)`
- Surveillez les métriques Telemetry pour les problèmes de performance

Guide des opérations SMS-C

← Retour à l'index de documentation

Procédures opérationnelles quotidiennes, surveillance et tâches de maintenance pour les équipes d'opérations SMS-C.

Table des matières

- Opérations quotidiennes
- Surveillance
- Suivi des messages
- Gestion des itinéraires
- Gestion du frontend
- Gestion de la traduction des numéros
- Maintenance du système
- Sauvegarde et récupération
- Planification de la capacité
- Réponse aux incidents

Opérations quotidiennes

Vérification de santé du matin

Effectuez ces vérifications au début de chaque jour :

1. Vérifier l'état du système

```
# Vérification de la santé de l'API
curl https://api.example.com:8443/api/status

# Réponse attendue :
# {"status":"ok","application":"OmniMessage","timestamp":"2025-10-
30T08:00:00Z"}
```

2. Examiner les métriques Prometheus

Accédez au tableau de bord Prometheus et vérifiez :

- Débit de messages (dernières 24 heures)
- Taux d'échec de routage (doit être < 1%)
- Arriéré de la file d'attente (doit être < 1000 en attente)
- Taux de succès de livraison (doit être > 95%)
- État de connexion du frontend (tous les frontends attendus actifs)

3. Vérifier la file d'attente des messages

Accédez à l'interface Web : `https://sms-admin.example.com/message_queue`

Examinez :

- Total des messages en attente (doit être faible)
- Âge du message le plus ancien (doit être < 5 minutes)
- Messages avec un nombre élevé de tentatives de livraison (enquêter si > 3)
- Messages de lettre morte (enquêter sur ceux présents)

4. Examiner l'état du frontend

Accédez à l'interface Web : `https://sms-admin.example.com/frontend_status`

Vérifiez :

- Tous les frontends attendus sont actifs
- Pas de déconnexions non expirées
- Pas d'erreurs de frontend dans les dernières 24 heures

5. Vérifier les journaux d'application

Accédez à l'interface Web : `https://sms-admin.example.com/logs` ou vérifiez les fichiers journaux

Recherchez :

- Messages de niveau erreur
- Échecs de routage
- Échecs de facturation
- Problèmes de connexion à la base de données
- Problèmes de nœud de cluster

Surveillance du volume des messages

Vérifiez les comptes de messages horaires :

Utilisez la requête Prometheus :

```
# Messages reçus par heure
increase(sms_c_message_received_count[1h])

# Messages livrés par heure
increase(sms_c_delivery_succeeded_count[1h])

# Calculer le taux de livraison
rate(sms_c_delivery_succeeded_count[1h]) /
rate(sms_c_message_received_count[1h])
```

Modèles attendus :

- Heures de bureau : Volume plus élevé
- Nuits/week-ends : Volume plus faible
- Taux de livraison : Doit être > 95%

Conditions d'alerte :

- Chute soudaine des messages (> 50% de diminution)

- Pic soudain des messages (> 200% d'augmentation)
- Taux de livraison tombant en dessous de 90%

Surveillance

Métriques clés à surveiller

Métriques de traitement des messages

Nombre de messages reçus (`sms_c_message_received_count`) :

- **Quoi** : Total des messages entrant dans le système
- **Alerte** : Chute ou pic soudain
- **Requête** : `rate(sms_c_message_received_count[5m])`

Durée de traitement des messages (`sms_c_message_processing_stop_duration`) :

- **Quoi** : Temps de traitement de bout en bout
- **Alerte** : p95 > 1000ms
- **Requête** : `histogram_quantile(0.95, sms_c_message_processing_stop_duration)`

Métriques de routage

Échecs de routage (`sms_c_routing_failed_count`) :

- **Quoi** : Messages que le système n'a pas pu router
- **Alerte** : Toute défaillance (> 0)
- **Requête** : `increase(sms_c_routing_failed_count[5m])`

Itinéraire correspondant (`sms_c_routing_route_matched_count`) :

- **Quoi** : Quels itinéraires sont utilisés
- **Alerte** : Itinéraires prioritaires ne correspondant pas
- **Requête** : `sms_c_routing_route_matched_count`

Métriques de livraison

Taux de succès de livraison :

- **Quoi** : Pourcentage de livraisons réussies
- **Alerte** : Taux < 95%
- **Requête** : `rate(sms_c_delivery_succeeded_count[5m]) / rate(sms_c_delivery_queued_count[5m])`

Tentatives de livraison (`sms_c_delivery_succeeded_attempt_count`) :

- **Quoi** : Tentatives nécessaires pour la livraison
- **Alerte** : p95 > 2 (trop de tentatives)
- **Requête** : `histogram_quantile(0.95, sms_c_delivery_succeeded_attempt_count)`

Métriques de file d'attente

Taille de la file d'attente (`sms_c_queue_size_size`) :

- **Quoi** : Total des messages dans la file d'attente
- **Alerte** : Taille > 10,000
- **Requête** : `sms_c_queue_size_size`

Âge du message le plus ancien (`sms_c_queue_oldest_message_age_seconds`) :

- **Quoi** : Âge du message le plus ancien en attente
- **Alerte** : Âge > 300 secondes
- **Requête** : `sms_c_queue_oldest_message_age_seconds`

Configuration du tableau de bord

Panneaux du tableau de bord opérationnel :

1. **Débit de messages** (Graphique)
    - Messages reçus (taux de 5 minutes)
    - Messages livrés (taux de 5 minutes)

- Plage horaire : Dernières 24 heures

2. **État de la file d'attente** (Statistiques uniques)
   - Messages en attente actuels
   - Âge du message le plus ancien
   - Nombre de messages échoués
3. **Performance de livraison** (Graphique)
   - Taux de succès au fil du temps
   - Taux d'échec au fil du temps
   - Plage horaire : Dernières 24 heures
4. **État du routage** (Tableau)
   - ID de l'itinéraire
   - Nombre de correspondances (dernière heure)
   - SMSC de destination
   - Priorité
5. **État du frontend** (Tableau)
   - Nom du frontend
   - État (actif/expiré)
   - Dernière vue
   - Nombre de messages (dernière heure)
6. **Santé du système** (Statistiques uniques)
   - Temps de réponse de l'API (p95)
   - Temps de requête de base de données (p95)
   - Temps de recherche ENUM (p95)

Configuration des alertes

Alertes critiques (Réponse immédiate requise) :

```yaml
# Aucun itinéraire trouvé - les messages ne peuvent pas être livrés
- alert: RoutingFailures
  expr: increase(sms_c_routing_failed_count[5m]) > 0
  severity: critical
  description: "{{ $value }} messages ont échoué au routage dans les 5 dernières minutes"

# File d'attente en augmentation - traitement en retard
- alert: QueueBacklog
  expr: sms_c_queue_size_pending > 10000
  severity: critical
  description: "La file d'attente a {{ $value }} messages en attente"

# Messages vieillissants - livraison bloquée
- alert: OldMessagesInQueue
  expr: sms_c_queue_oldest_message_age_seconds > 300
  severity: critical
  description: "Le message le plus ancien a {{ $value }} secondes"

# Frontend déconnecté - aucun chemin de livraison
- alert: FrontendDisconnected
  expr: sms_c_frontend_status_count{status="disconnected"} > 0
  severity: critical
  description: "{{ $value }} frontends déconnectés"
```

Alertes d'avertissement (Enquête nécessaire) :

```
# Taux de succès de livraison en baisse
- alert: LowDeliveryRate
  expr: rate(sms_c_delivery_succeeded_count[10m]) /
rate(sms_c_delivery_queued_count[10m]) < 0.90
  severity: warning
  description: "Le taux de succès de livraison est {{ $value }}"

# Trop de tentatives de livraison
- alert: HighRetryRate
  expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count) > 2
  severity: warning
  description: "Tentatives de livraison au 95e percentile : {{
$value }}"

# Recherches ENUM lentes ou échouées
- alert: SlowEnumLookups
  expr: histogram_quantile(0.95, sms_c_enum_lookup_stop_duration)
> 5000
  severity: warning
  description: "Recherches ENUM prenant > 5 secondes"

# Faible taux de réussite du cache ENUM
- alert: LowEnumCacheHitRate
  expr: rate(sms_c_enum_cache_hit_count[10m]) /
(rate(sms_c_enum_cache_hit_count[10m]) +
rate(sms_c_enum_cache_miss_count[10m])) < 0.70
  severity: warning
  description: "Taux de réussite du cache ENUM : {{ $value }}"
```

Suivi des messages

Trouver un message spécifique

Par ID de message :

1. **Interface Web** : Accédez à `/message_queue`
2. Entrez l'ID du message dans la boîte de recherche
3. Affichez les détails complets et l'historique des événements

Via API :

```
curl https://api.example.com:8443/api/messages/12345
```

Par numéro de téléphone :

1. **Interface Web** : Accédez à `/message_queue`
2. Entrez le numéro de téléphone dans la boîte de recherche
3. Affichez tous les messages pour ce numéro

Suivre le cycle de vie du message

Afficher l'historique des événements :

1. **Interface Web** : Cliquez sur le message dans la file d'attente, affichez la section "Événements"
2. **API** : `GET /api/events/12345`

Séquence d'événements courante :

```
1. message_inserted - Message créé
   ↓
2. number_translated - Numéros normalisés (si configuré)
   ↓
3. message_routed - Décision de routage prise
   ↓
4. charging_attempted - Vérification de facturation (si activée)
   ↓
5. message_delivered - Livré avec succès
```

Séquence de livraison échouée :

```
1. message_inserted
   ↓
2. message_routed
   ↓
3. delivery_attempt_1 - Première tentative échouée
   ↓
4. delivery_attempt_2 - Deuxième tentative échouée (délai de 2
minutes)
   ↓
5. delivery_attempt_3 - Troisième tentative échouée (délai de 4
minutes)
   ↓
6. message_dead_letter - Limite de réessai dépassée
```

Vérifier l'état de livraison

En attente de livraison :

- État : "queued" (prêt à être envoyé maintenant) ou "backoff" (attente d'un réessai)
- deliver_after : nil/passe pour `queued`, un horodatage futur pour `backoff`
- delivery_attempts : 0 pour un message `queued` frais, plus élevé pour `backoff`

Messages livrés :

- État : "delivered" (livraison confirmée) ou "sent" (soumis, pas encore de DLR)
- deliver_time : Horodatage de la livraison
- dest_smsc : Frontend qui a livré

Messages non livrés / échoués :

- État : "backoff" avec un nombre élevé de delivery_attempts - toujours en réessai (limité à un intervalle de 30 minutes, jusqu'à ce que le `expires` du message soit atteint)
- État : "expired", "dropped", "balance_rejected", ou "auto_replied" - terminal
- Vérifiez le journal des événements pour les raisons d'échec

Routage de messages basé sur la localisation

Le SMS-C prend en charge la récupération de messages basée sur la localisation, permettant aux frontends de recevoir automatiquement des messages destinés aux abonnés enregistrés à leur emplacement.

Comment cela fonctionne :

Lorsqu'un frontend interroge pour des messages en attente en utilisant `get_messages_for_smsc(smsc_name)`, le système renvoie des messages de deux manières :

1. **Routage explicite** - Messages où `dest_smsc` correspond explicitement au nom du frontend
2. **Routage basé sur la localisation** - Messages où :
   - `dest_smsc` est `null` (non routé explicitement)
   - `destination_msisdn` a un enregistrement de localisation actif
   - Le champ `location` de la localisation correspond au nom du frontend
   - La localisation n'a pas expiré

Scénario d'exemple :

Un abonné avec MSISDN `+447700900123` s'enregistre au frontend `uk_gateway` :

```
# L'abonné s'enregistre (crée un enregistrement de localisation)
POST /api/locations
{
  "msisdn": "+447700900123",
  "imsi": "234150123456789",
  "location": "uk_gateway",
  "expires": "2025-11-01T12:00:00Z"
}
```

Lorsqu'un message arrive pour cet abonné sans routage explicite :

```
# Message soumis sans dest_smsc
POST /api/messages
{
  "source_msisdn": "+15551234567",
  "destination_msisdn": "+447700900123",
  "message_body": "Hello",
  "source_smsc": "api"
  # Remarque : dest_smsc est null
}
```

Le frontend `uk_gateway` recevra automatiquement ce message lorsqu'il interroge :

```
# Le frontend interroge pour des messages
GET /api/messages/queue?smsc=uk_gateway

# Renvoie le message même si dest_smsc est null
# car l'abonné de destination est enregistré à uk_gateway
```

Exigences de localisation :

Pour que le routage basé sur la localisation fonctionne :

- La table `locations` doit avoir une entrée pour le `destination_msisdn`
- Le champ `location` doit correspondre au nom du SMSC interrogeant
- L'horodatage `expires` doit être dans le futur

Surveillance du routage basé sur la localisation :

Vérifiez les enregistrements de localisation :

```
# Via API
GET /api/locations/{msisdn}

# Vérifiez si la localisation a expiré
# le champ expires doit être > temps actuel
```

Problèmes courants :

- **Message non livré** : Vérifiez si la localisation a expiré
- **Mauvais frontend** : Vérifiez que le champ `location` correspond au nom de frontend attendu
- **Localisation non trouvée** : L'abonné peut avoir besoin de se réenregistrer

Interventions manuelles

Réessayer un message échoué :

```
# Réinitialiser delivery_attempts et deliver_after
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{
    "delivery_attempts": 0,
    "deliver_after": "2025-10-30T12:00:00Z"
  }'
```

Changer de destination :

```
# Routage vers un SMSC différent
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{
    "dest_smsc": "backup_gateway"
  }'
```

Supprimer un message bloqué :

```
curl -X DELETE https://api.example.com:8443/api/messages/12345
```

Gestion des itinéraires

Voir les itinéraires actuels

Interface Web : Accédez à `/sms_routing`

Via API :

```
# Lister tous les itinéraires
curl https://api.example.com:8443/api/routes
```

Vérifier l'utilisation des itinéraires :

Requête Prometheus :

```
# Messages routés par chaque itinéraire (dernière heure)
increase(sms_c_routing_route_matched_count[1h])
```

Ajouter un nouvel itinéraire

Interface Web :

1. Accédez à `/sms_routing`
2. Cliquez sur "Ajouter un nouvel itinéraire"
3. Remplissez les champs :
   - **Préfixe d'appel** : Préfixe du numéro source (facultatif)
   - **Préfixe appelé** : Préfixe du numéro de destination (requis pour le routage géographique)
   - **SMSC source** : Filtre du système source (facultatif)
   - **SMSC de destination** : Passerelle de destination (requis sauf pour auto-réponse/abandon)
   - **Uniquement sur le réseau** : Restreindre cet itinéraire aux destinations sur le réseau uniquement (nécessite Diameter/HSS)
   - **Priorité** : Priorité de l'itinéraire (1-255, plus bas = plus haute priorité)
   - **Poids** : Poids d'équilibrage de charge (1-100)

- **Description** : Description lisible par l'homme
- **Activé** : Cochez pour activer immédiatement

4. Cliquez sur "Enregistrer l'itinéraire"

Exemple : Itinéraire géographique :

- Préfixe appelé : `+44`
- SMSC de destination : `uk_gateway`
- Priorité : 50
- Poids : 100
- Description : "Routage UK"

Exemple : Itinéraire équilibré :

Créez deux itinéraires avec les mêmes critères mais des poids différents :

Itinéraire 1 :

- Préfixe appelé : `+44`
- SMSC de destination : `uk_primary`
- Priorité : 50
- Poids : 70
- Description : "UK primaire (70%)"

Itinéraire 2 :

- Préfixe appelé : `+44`
- SMSC de destination : `uk_backup`
- Priorité : 50
- Poids : 30
- Description : "UK de secours (30%)"

Exemple : Itinéraire uniquement sur le réseau :

Restreindre un transporteur externe ou une application interne à n'envoyer qu'aux abonnés sur le réseau. Les messages vers des destinations hors réseau sont abandonnés.

- SMSC source : `carrier_smpp_bind`
- SMSC de destination : `local_msc`
- Uniquement sur le réseau : coché
- Priorité : 50
- Poids : 100
- Description : "Transporteur X - uniquement sur le réseau"

Cela nécessite que Diameter/HSS soit activé. Si le HSS n'est pas disponible, les messages sont abandonnés (échec-clos).

Tester les itinéraires

Simulateur de routage :

1. Accédez à `/simulator`
2. Entrez les paramètres de test :
   - Numéro d'appel : `+15551234567`
   - Numéro appelé : `+447700900000`
   - SMSC source : (facultatif)
   - Type source : (facultatif)
3. Cliquez sur "Simuler le routage"
4. Examinez les résultats :
   - **Itinéraire sélectionné** : Quel itinéraire le système a choisi
   - **Toutes les correspondances** : Quels itinéraires ont correspondu aux critères
   - **Évaluation** : Pourquoi chaque itinéraire a correspondu ou non

Tester avant la production :

- Testez tous les nouveaux itinéraires dans le simulateur
- Vérifiez que l'itinéraire correct est sélectionné
- Vérifiez l'ordre de priorité
- Validez la distribution des poids

Modifier un itinéraire existant

Interface Web :

1. Accédez à `/sms_routing`
2. Trouvez l'itinéraire dans la liste
3. Cliquez sur "Modifier"
4. Modifiez les champs
5. Cliquez sur "Enregistrer l'itinéraire"

Modifications courantes :

- **Désactiver l'itinéraire** : Décochez "Activé" (suppression temporaire)
- **Ajuster le poids** : Changer la distribution d'équilibrage de charge
- **Changer la priorité** : Réorganiser l'évaluation de l'itinéraire
- **Mettre à jour la destination** : Passer à un SMSC différent

Supprimer un itinéraire

Interface Web :

1. Accédez à `/sms_routing`
2. Trouvez l'itinéraire dans la liste
3. Cliquez sur "Supprimer"
4. Confirmez la suppression

Avertissement : La suppression des itinéraires est permanente. Envisagez de désactiver à la place.

Exporter/Importer des itinéraires

Exporter des itinéraires (Sauvegarde) :

1. Accédez à `/sms_routing`
2. Cliquez sur "Exporter des itinéraires"
3. Enregistrez le fichier JSON

Importer des itinéraires :

1. Accédez à `/sms_routing`
2. Cliquez sur "Importer des itinéraires"
3. Sélectionnez le fichier JSON
4. Choisissez le mode d'importation :
   - **Fusionner** : Ajouter aux itinéraires existants
   - **Remplacer** : Supprimer tout et importer

Cas d'utilisation :

- Sauvegarde avant des changements majeurs
- Copier des itinéraires entre environnements
- Récupération après sinistre
- Versionnage de configuration

Gestion du frontend

Surveiller les connexions du frontend

Interface Web : Accédez à `/frontend_status`

Vérifiez :

- Tous les frontends attendus sont "actifs"
- Les dernières heures vues sont récentes (< 90 secondes)
- Pas de frontends expirés inattendus

Via API :

```
# Obtenir les frontends actifs
curl https://api.example.com:8443/api/frontends/active

# Obtenir des statistiques
curl https://api.example.com:8443/api/frontends/stats
```

Enquêter sur les déconnexions

Frontend expiré :

1. Vérifiez les journaux du frontend pour des erreurs
2. Vérifiez la connectivité réseau vers le SMS-C
3. Confirmez que le frontend est en cours d'exécution
4. Vérifiez la logique d'enregistrement du frontend (doit se réenregistrer toutes les 60s)

Enregistrement non affiché :

1. Vérifiez que le frontend appelle POST `/api/frontends/register`
2. Vérifiez les journaux de l'API pour des erreurs d'enregistrement
3. Vérifiez le format de la charge utile JSON
4. Testez l'enregistrement manuellement avec curl

Exemple d'enregistrement manuel :

```
curl -X POST https://api.example.com:8443/api/frontends/register \
  -H "Content-Type: application/json" \
  -d '{
    "frontend_name": "test_gateway",
    "frontend_type": "smpp",
    "ip_address": "10.0.1.50",
    "hostname": "gateway.example.com"
  }'
```

Voir l'historique du frontend

Interface Web :

1. Accédez à `/frontend_status`
2. Trouvez le frontend dans la liste
3. Cliquez sur "Historique"
4. Examinez les enregistrements passés

Via API :

```
curl https://api.example.com:8443/api/frontends/history/uk_gateway
```

Cas d'utilisation :

- Enquêter sur la fiabilité de la connexion
- Suivre les modèles de disponibilité du frontend
- Identifier les changements de configuration

Gestion de la traduction des numéros

Les règles de traduction des numéros sont gérées via `config/runtime.exs`. Les changements nécessitent un redémarrage de l'application.

Voir les règles de traduction actives

Vérifiez le fichier de configuration :

```
cat config/runtime.exs | grep -A 20 "translation_rules:"
```

Tâches de traduction courantes

Ajouter le code pays aux numéros locaux :

Modifier `config/runtime.exs` :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 100,
  description: "Ajouter +1 aux numéros américains à 10 chiffres",
  enabled: true
}
```

Normaliser le format international :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^00(\d+)$",
  calling_replace: "+\1",
  called_match: "^00(\d+)$",
  called_replace: "+\1",
  priority: 10,
  description: "Convertir le préfixe 00 en +",
  enabled: true
}
```

Suppression de code spécifique au transporteur :

```
%{
  calling_prefix: nil,
  called_prefix: "101",
  source_smsc: "carrier_a",
  calling_match: nil,
  calling_replace: nil,
  called_match: "^101(\d+)$",
  called_replace: "\1",
  priority: 5,
  description: "Supprimer le code transporteur du transporteur A",
  enabled: true
}
```

Tester les règles de traduction

Après les modifications de configuration :

1. Redémarrez l'application pour charger les nouvelles règles
2. Soumettez un message de test avec une source/destination qui devrait correspondre
3. Vérifiez le journal des événements pour l'événement `number_translated`
4. Vérifiez que les numéros ont été transformés correctement

Désactiver une règle de traduction

Définissez `enabled: false` dans la règle :

```
%{
  ...
  enabled: false
}
```

Redémarrez l'application.

Maintenance du système

Maintenance de la base de données

Vérifier la taille de la base de données :

Utilisez vos outils de gestion de base de données pour surveiller la taille de stockage des CDR :

- **MySQL/MariaDB** : Interrogez `information_schema.tables` pour la taille de la base de données
- **PostgreSQL** : Utilisez la fonction `pg_database_size()` ou la commande `\l+` dans psql

Nettoyer les anciens enregistrements CDR :

Les enregistrements CDR doivent être archivés et purgés périodiquement en fonction de votre politique de conservation :

- Configurez l'archivage automatique en fonction des exigences commerciales (généralement 30-90 jours dans la base de données opérationnelle)
- Archivez les anciens enregistrements vers un entrepôt de données ou un stockage à froid
- Supprimez les enregistrements archivés de la base de données opérationnelle par lots pour éviter la contention de verrouillage

Optimiser les tables :

Optimisez périodiquement les tables de la base de données pour maintenir la performance :

- **MySQL/MariaDB** : Exécutez la commande `OPTIMIZE TABLE` pendant les périodes de faible trafic
- **PostgreSQL** : Exécutez régulièrement `VACUUM ANALYZE` (ou activez l'autovacuum)

Exécutez chaque semaine pendant une période de faible trafic pour maintenir une performance optimale.

Maintenance de la base de données Mnesia

Vérifier la taille de la table Mnesia :

```
# Dans la console IEx
:mnesia.table_info(:sms_route, :size)
:mnesia.table_info(:translation_rule, :size)
```

Sauvegarder les tables Mnesia :

```
# Exporter les itinéraires (Interface Web)
# Accédez à /sms_routing
# Cliquez sur "Exporter les itinéraires"

# Ou via la sauvegarde Mnesia
:mnesia.backup("/var/backups/sms_c/mnesia_backup.bup")
```

Restaurer Mnesia :

```
# Via l'importation de l'interface Web
# Ou restaurer la sauvegarde :
:mnesia.restore("/var/backups/sms_c/mnesia_backup.bup", [])
```

Rotation des journaux

Configurez logrotate pour les journaux d'application :

```
# /etc/logrotate.d/sms_c
/var/log/sms_c/*.log {
    daily
    rotate 30
    compress
    delaycompress
    notifempty
    create 0644 sms_user sms_group
    sharedscripts
    postrotate
        systemctl reload omnimessage || true
    endscript
}
```

Redémarrer l'application

Redémarrage gracieux (temps d'arrêt nul dans le cluster) :

```
# Redémarrez un nœud à la fois
systemctl restart omnimessage

# Attendez que le nœud rejoigne le cluster
# Répétez pour chaque nœud
```

Redémarrage d'urgence (tous les nœuds) :

```
systemctl restart omnimessage
```

Après le redémarrage :

- Vérifiez que tous les frontends se reconnectent
- Vérifiez Prometheus pour la continuité des métriques
- Surveillez les journaux pour des erreurs
- Vérifiez que le traitement des messages reprend

Sauvegarde et récupération

Que sauvegarder

1. **Fichiers de configuration** :
   - `config/runtime.exs`
   - `config/config.exs`
   - `config/prod.exs` (si existant)
2. **Tables de routage** (Mnesia) :
   - Exporter via l'interface Web
   - Ou commande de sauvegarde Mnesia
3. **Base de données SQL CDR** :
   - Sauvegarde complète quotidienne
   - Sauvegardes des journaux de transactions (continu)
4. **Certificats TLS** :
   - `priv/cert/*.crt`
   - `priv/cert/*.key`

Procédures de sauvegarde

Sauvegarde quotidienne de la configuration :

```
#!/bin/bash
# /opt/omnimessage/scripts/backup_config.sh

BACKUP_DIR="/var/backups/sms_c/$(date +%Y%m%d)"
mkdir -p $BACKUP_DIR

# Sauvegarde de la configuration
cp -r /opt/omnimessage/config $BACKUP_DIR/

# Sauvegarde des certificats
cp -r /opt/omnimessage/priv/cert $BACKUP_DIR/

# Définir les permissions
chmod 600 $BACKUP_DIR/cert/*

echo "Sauvegarde de la configuration terminée : $BACKUP_DIR"
```

Sauvegarde de la base de données :

```
#!/bin/bash
# /opt/omnimessage/scripts/backup_database.sh

BACKUP_DIR="/var/backups/sms_c/database"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p $BACKUP_DIR

# Sauvegarde de la base de données SQL CDR
# MySQL/MariaDB : Utilisez mysqldump avec --single-transaction
pour la cohérence
# PostgreSQL : Utilisez pg_dump -F c pour le format personnalisé

# Exemple de structure (adaptez à votre base de données) :
# - Utilisez l'outil de sauvegarde approprié (mysqldump, pg_dump)
# - Activez les sauvegardes sécurisées pour la cohérence
# - Compressez la sortie pour économiser de l'espace
# - Configurez la période de conservation (par exemple, 30 jours)

# Supprimer les anciennes sauvegardes
find $BACKUP_DIR -name "sms_c_*.gz" -mtime +30 -delete

echo "Sauvegarde de la base de données terminée : sms_c_${DATE}"
```

Sauvegarde des tables de routage :

```
#!/bin/bash
# /opt/omnimessage/scripts/backup_routes.sh

BACKUP_DIR="/var/backups/sms_c/routes"
DATE=$(date +%Y%m%d)

mkdir -p $BACKUP_DIR

# Exporter via API
curl https://api.example.com:8443/api/routes/export \
  > $BACKUP_DIR/routes_${DATE}.json

echo "Sauvegarde des itinéraires terminée : routes_${DATE}.json"
```

Planifier des sauvegardes (crontab) :

```
# Quotidiennement à 2h du matin
0 2 * * * /opt/omnimessage/scripts/backup_config.sh
0 2 * * * /opt/omnimessage/scripts/backup_database.sh
0 2 * * * /opt/omnimessage/scripts/backup_routes.sh
```

Procédures de récupération

Restaurer la configuration :

```
# Arrêter l'application
systemctl stop omnimessage

# Restaurer les fichiers de configuration
cp -r /var/backups/sms_c/20251030/config/*
/opt/omnimessage/config/

# Restaurer les certificats
cp -r /var/backups/sms_c/20251030/cert/*
/opt/omnimessage/priv/cert/

# Démarrer l'application
systemctl start omnimessage
```

Restaurer la base de données SQL CDR :

Utilisez les outils de restauration appropriés pour votre base de données :

- **MySQL/MariaDB** : Décompressez et pipez vers le client mysql
- **PostgreSQL** : Utilisez pg_restore avec des dumps au format personnalisé

Important : Arrêtez l'application SMS-C avant de restaurer la base de données pour éviter les conflits de données.

Restaurer les tables de routage :

1. Accédez à l'interface Web `/sms_routing`
2. Cliquez sur "Importer des itinéraires"
3. Sélectionnez le fichier JSON de sauvegarde

4. Choisissez le mode "Remplacer"
5. Confirmez l'importation

Planification de la capacité

Surveiller les tendances de croissance

Tendance du volume de messages :

Requête Prometheus (moyenne sur 30 jours) :

```
avg_over_time(sms_c_message_received_count[30d])
```

Taux de croissance de la base de données :

```sql
-- Croissance des données mensuelles
SELECT
  DATE_FORMAT(inserted_at, '%Y-%m') AS month,
  COUNT(*) AS message_count,
  ROUND(SUM(LENGTH(message_body)) / 1024 / 1024, 2) AS data_mb
FROM message_queues
GROUP BY month
ORDER BY month DESC
LIMIT 12;
```

Indicateurs de capacité

Utilisation du CPU :

- **Normal** : < 50% en moyenne
- **Élevé** : > 70% soutenu
- **Critique** : > 90%

Utilisation de la mémoire :

- **Normal** : < 70% de disponible

- **Élevé** : > 80%
- **Critique** : > 90%

Utilisation du disque :

- **Normal** : < 60% plein
- **Élevé** : > 75%
- **Critique** : > 85%

Profondeur de la file d'attente :

- **Normal** : < 1000 en attente
- **Élevé** : > 5000 en attente
- **Critique** : > 10,000 en attente

Recommandations de mise à l'échelle

Quand mettre à l'échelle verticalement (Améliorer les ressources) :

- CPU constamment > 70%
- Mémoire constamment > 80%
- Goulot d'étranglement à nœud unique

Quand mettre à l'échelle horizontalement (Ajouter des nœuds) :

- CPU > 50% sur tous les nœuds
- Volume de messages > 5,000 msg/sec
- Distribution géographique nécessaire
- Haute disponibilité requise

Mise à l'échelle de la base de données :

- Réplicas de lecture pour les requêtes de rapport
- Optimisation du pool de connexions
- Optimisation des index
- Partitionner de grandes tables par date

Réponse aux incidents

Niveaux de gravité

Critique (Réponse immédiate) :

- Aucun message n'est livré
- Tous les frontends déconnectés
- Base de données indisponible
- API complètement hors service

Élevé (Réponse dans l'heure) :

- Taux de succès de livraison < 80%
- Plusieurs frontends déconnectés
- Échecs de routage > 10%
- Arriéré de file d'attente croissant

Moyen (Réponse dans les 4 heures) :

- Un seul frontend déconnecté
- Taux de succès de livraison 80-95%
- Traitement des messages lent
- Recherches ENUM échouées

Faible (Réponse dans les 24 heures) :

- Dégradation mineure des performances
- Problème d'itinéraire unique
- Alertes d'avertissement non critiques

Liste de contrôle des incidents

1. Évaluer la gravité :

- Vérifiez les alertes Prometheus
- Examinez les métriques du tableau de bord

- Vérifiez l'état de la file d'attente des messages
- Vérifiez les connexions des frontends

2. Rassembler des informations :

- Changements de configuration récents ?
- Déploiements récents ?
- État des dépendances externes (OCS, DNS) ?
- Messages d'erreur dans les journaux ?

3. Actions immédiates :

- Arrêtez les changements en cours
- Rétrogradez les déploiements récents si cause suspectée
- Activez les journaux détaillés si nécessaire
- Informez les parties prenantes

4. Enquête :

- Examinez les journaux d'application
- Vérifiez l'utilisation des ressources système
- Examinez la performance de la base de données
- Testez les dépendances externes

5. Résolution :

- Appliquez la correction
- Testez dans le simulateur
- Déployez en production
- Surveillez pour des améliorations

6. Post-incident :

- Documentez la cause racine
- Mettez à jour la surveillance/alertes
- Mettez en œuvre des mesures préventives
- Mettez à jour les manuels d'exploitation

Incidents courants

Arriéré de file d'attente élevé :

1. Vérifiez le taux de succès de livraison
2. Vérifiez que les frontends sont connectés et interrogent
3. Vérifiez la performance de la base de données
4. Examinez Prometheus pour les goulots d'étranglement
5. Envisagez d'augmenter la taille/l'intervalle des lots

Échecs de routage :

1. Examinez la configuration de routage
2. Testez dans le simulateur de routage
3. Vérifiez les itinéraires manquants
4. Vérifiez que l'itinéraire de secours existe
5. Vérifiez les journaux d'événements pour les raisons d'échec

Déconnexions de frontend :

1. Vérifiez l'état du système du frontend
2. Vérifiez la connectivité réseau
3. Examinez les journaux du frontend
4. Testez l'enregistrement API manuel
5. Vérifiez les règles de pare-feu

Traitement des messages lent :

1. Vérifiez la performance des requêtes de base de données
2. Examinez la configuration des travailleurs par lots
3. Vérifiez les ressources adéquates (CPU/Mémoire)
4. Vérifiez les retards de recherche ENUM
5. Examinez la performance du système de facturation

Pour des procédures de dépannage détaillées, consultez le Guide de dépannage.

Routage SMS

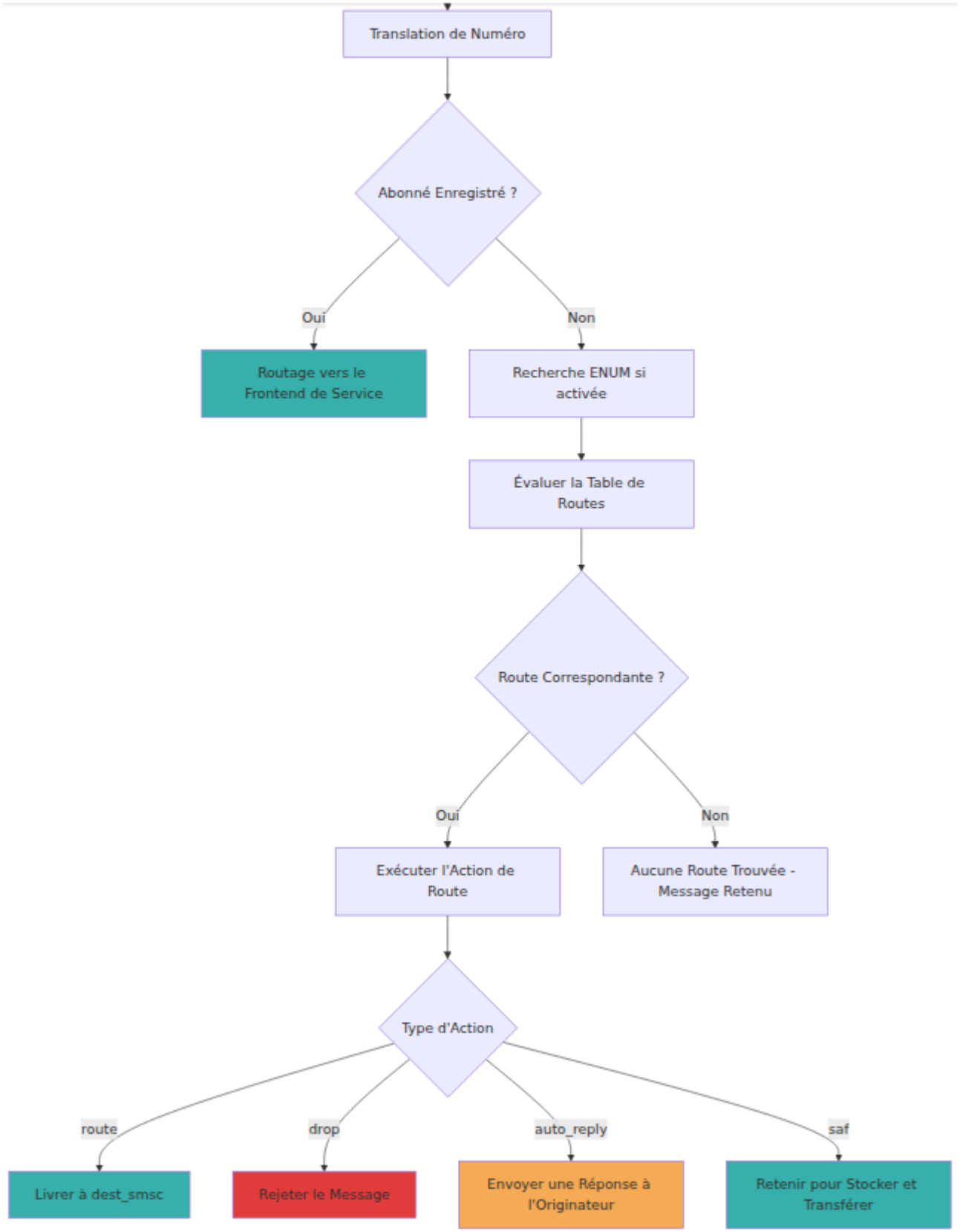
← Retour à l'Index de Documentation

Aperçu

Le moteur de routage SMS-C détermine où chaque message va après être entré dans le système. Mnesia stocke les routes. Vous pouvez les modifier à l'exécution via l'API ou le panneau de contrôle. Les routes prennent en charge la correspondance regex, l'ordre de priorité, l'équilibrage de charge pondéré, la correspondance des résultats ENUM (DNS NAPTR) et la correspondance sur réseau/hors réseau basée sur HSS.

Pipeline de Traitement des Messages

Chaque message suit ce pipeline de la soumission au routage :



Le statut sur réseau/hors réseau **n'est pas** une action de route. C'est un *critère de correspondance* (`originating_network` / `terminating_network`) évalué

pendant que la table de routes est filtrée - voir Correspondance Sur Réseau / Hors Réseau.

Ordre de Priorité

1. **Routage basé sur la localisation** - si l'abonné de destination a une inscription active, routez directement vers le frontend de service. Toutes les règles de route sont ignorées.
2. **Évaluation de la table de routes** - toutes les routes activées sont évaluées par rapport au message. Chaque route est enregistrée avec des détails de réussite/échec par critère de correspondance (y compris toute vérification HSS sur réseau/hors réseau).
3. **Exécution de l'action de route** - l'action de la route sélectionnée détermine ce qui arrive au message.

Modèle de Données de Route

Champ	Type	Description
`route_id`	Entier	Identifiant unique auto-généré
`calling_regex`	Chaîne	Regex pour le numéro source (`nil` = joker)
`called_regex`	Chaîne	Regex pour le numéro de destination (`nil` = joker)
`source_smsc`	Chaîne	Nom de la SMSC source (`nil` = joker)
`dest_smsc`	Chaîne	SMSC de destination. Requis pour `route`; optionnel pour `saf`; inutilisé par `drop`/`auto_reply`
`source_type`	Chaîne	Filtrer par type source : `ims`, `circuit_switched`, `smpp`, ou `nil`
`enum_domain`	Chaîne	Étiquette de domaine ENUM pour la route (`nil` = aucun). Stocké sur la route et ajoute de la spécificité, mais la requête DNS réelle utilise la configuration globale `enum_domains` - voir Recherche ENUM
`enum_result_domain`	Chaîne	Critère de correspondance : la route ne correspond que si un résultat NAPTR ENUM résout ce domaine (`nil` = joker)
`action`	Chaîne	Action de route : `route`, `drop`, `auto_reply`, `saf`

Champ	Type	Description
`auto_reply_message`	Chaîne	Texte de réponse (requis lorsque l'action est `auto_reply`)
`charged`	Chaîne	Facturation : `yes`, `no`, `default`
`weight`	Entier	Poids d'équilibrage de charge (1–100)
`priority`	Entier	Priorité (1–255, plus bas = plus haute priorité)
`description`	Chaîne	Étiquette lisible par l'homme
`enabled`	Booléen	Bascule actif/inactif
`originating_network`	Atome	Critère de correspondance sur l'état HSS MSISDN source : `:any` | `:on_net` | `:off_net` (par défaut `:any`)
`terminating_network`	Atome	Critère de correspondance sur l'état HSS MSISDN de destination : `:any` | `:on_net` | `:off_net` (par défaut `:any`)

Actions de Route

Le champ `action` a exactement quatre valeurs valides : `:route`, `:drop`, `:auto_reply`, `:saf`.

`route` (Par Défaut)

Livrer le message à `dest_smsc`. C'est l'action de routage standard.

```
{
  "action": "route",
  "called_regex": "^61",
  "dest_smsc": "australia-gw",
  "priority": 100,
  "description": "Route les numéros australiens vers la passerelle
AU"
}
```

`drop`

Rejette le message. Aucun `dest_smsc` requis.

```
{
  "action": "drop",
  "called_regex": "^900",
  "priority": 5,
  "description": "Bloquer les numéros à tarif premium"
}
```

`auto_reply`

Envoie `auto_reply_message` à l'originateur. Le message original est marqué comme ayant reçu une réponse automatique.

```
{
  "action": "auto_reply",
  "called_regex": "^911$",
  "auto_reply_message": "Urgent : SMS au 911 n'est pas pris en
charge. Veuillez composer le 911.",
  "priority": 1,
  "description": "Réponse automatique 911"
}
```

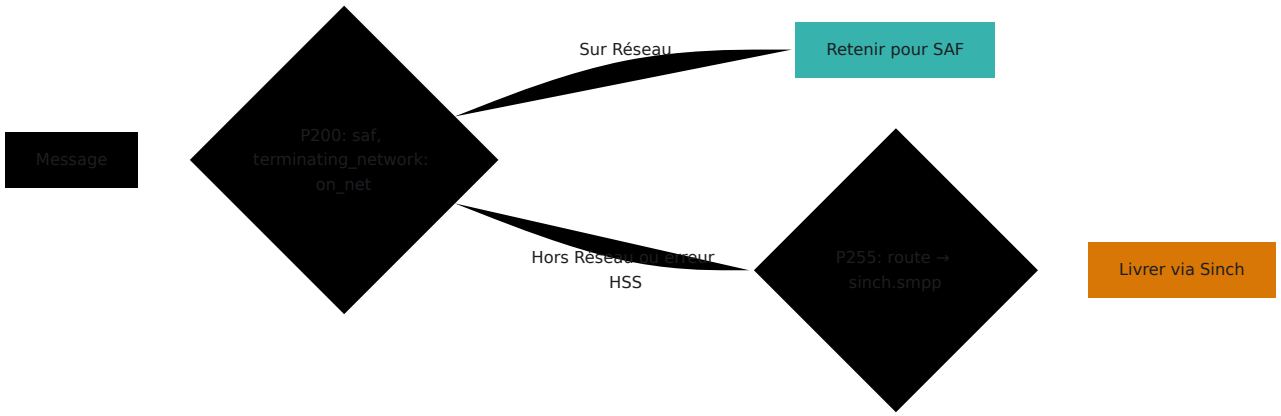
`saf`

Stocker et transférer. Le message est retenu et livré plus tard, lorsque l'abonné de destination s'inscrit. `dest_smsc` est optionnel (s'il est défini, il est utilisé lorsque le message retenu est finalement livré).

`saf` lui-même est **inconditionnel** - il ne réalise pas de vérification HSS. Pour retenir *uniquement* les abonnés sur réseau pour stockage et transfert, combinez `saf` avec un critère de correspondance `terminating_network: :on_net` (voir ci-dessous), et fournissez une route de secours hors réseau.

```
{
  "action": "saf",
  "terminating_network": "on_net",
  "priority": 200,
  "description": "Retenir les abonnés sur réseau pour stockage et transfert"
}
```

Configuration Typique : Retenue Sur Réseau + Route de Secours Hors Réseau



La route P200 ne *correspond* que lorsque la destination est confirmée sur réseau, donc les abonnés sur réseau sont retenus pour SAF. Les numéros hors réseau (et les erreurs HSS - échec fermé) échouent cette route et tombent sur la P255 de secours. Cela garantit que les abonnés sur réseau ne fuient jamais vers la passerelle hors réseau.

Recherche ENUM

ENUM permet aux décisions de routage de dépendre d'une recherche DNS E.164 NAPTR du numéro de destination. C'est typique pour la portabilité des numéros ou le routage inter-opérateurs, où DNS (et non votre HSS) est l'autorité sur laquelle appartient un numéro.

Comment ça fonctionne

1. **Activé globalement.** ENUM ne fonctionne que lorsque `enum_enabled: true` et `enum_domains` n'est pas vide. C'est une seule recherche effectuée **une fois par message, avant que la table de routes soit évaluée** - pas par route.
2. **Requête.** Le numéro appelé est converti au format ENUM inversé (`+15551234` → `4.3.2.1.5.5.5.1.e164.arpa`) et une requête NAPTR est émise. Chaque domaine configuré dans `enum_domains` est essayé **dans l'ordre**, s'arrêtant au premier qui retourne des enregistrements.
3. **Extraction du domaine de résultat.** Chaque enregistrement NAPTR retourné est analysé et réduit à un `result_domain` - les deux dernières étiquettes du champ `replacement`, ou de l'hôte dans une réécriture `sip:user@host` (`sip:+15551234@ims.example.com` → `example.com`).
4. **Correspondance.** Une route avec `enum_result_domain` défini ne correspond que si **n'importe quel** enregistrement NAPTR retourné a un `result_domain` égal exactement à cette chaîne. `enum_result_domain: nil` est un joker (correspond toujours, même lorsqu'aucune recherche ENUM n'a été effectuée).
5. **Mise en cache.** Les résultats sont mis en cache dans ETS pendant 15 minutes par `(number, domain)`. Les échecs sont également mis en cache, pour éviter de surcharger DNS lors de répétitions d'échecs.

Dérivation sur réseau/hors réseau à partir d'ENUM

ENUM ne retourne pas un statut booléen sur réseau/hors réseau - il retourne un domaine. Vous exprimez "sur réseau" en pointant `enum_result_domain` d'une route vers votre propre domaine IMS/opérateur, et en donnant à tout le reste

une route de secours. Contrairement à cela, la correspondance HSS (ci-dessous) retourne une réponse explicite sur réseau/hors réseau.

Ce que ENUM ne fait *pas*

- Le résultat NAPTR brut (`enum_result`) est utilisé **uniquement** comme clé de correspondance de route. Il n'est **pas** utilisé pour réécrire l'URI de destination ou choisir un prochain saut - où va un message correspondant est entièrement `route.dest_smsc`. Le pipeline de livraison en direct rejette le résultat ENUM brut après le routage.
- `enum_domain` sur une route n'est **pas** utilisé pour définir la portée de la requête DNS (la requête utilise toujours la liste globale `enum_domains`) et **n'est pas** un critère de correspondance. Il contribue uniquement à l'évaluation de spécificité.

Configuration

Le comportement DNS ENUM (`enum_enabled`, `enum_domains`, `enum_dns_servers`, `enum_timeout`) est configuré globalement - voir CONFIGURATION.md → Configuration ENUM.

Correspondance Sur Réseau / Hors Réseau

Un **appel Diameter Sh au HSS** répond à la question de savoir si une destination (ou un originateur) est sur réseau. Les routes exposent cela comme deux critères de correspondance. Ce ne sont **pas** des indicateurs booléens et **pas** une action de route. Ce sont des critères de valeur enum que le routeur évalue aux côtés des vérifications regex/SMSC/type. Une route ne correspond que si elle satisfait ses critères réseau.

`originating_network` / `terminating_network`

Chacun prend l'une des trois valeurs :

Valeur	Signification
`:any`	Pas de vérification HSS ; correspond à la fois sur réseau et hors réseau (par défaut)
`:on_net`	La route ne correspond que si le MSISDN est confirmé **sur réseau** (HSS retourne succès, code de r��sultat 2001)
`:off_net`	La route ne correspond que si le MSISDN est confirmé **hors réseau** (HSS retourne *Utilisateur Inconnu*, code de résultat 5001)

- `originating_network` vérifie le numéro **appelant** ; `terminating_network` vérifie le numéro **appelé**.
- **Cas d'utilisation** : restreindre une route à livrer uniquement aux abonnés sur réseau (`terminating_network: :on_net`) ; restreindre un lien SMPP ou un opérateur externe à accepter uniquement des originateurs sur réseau (`originating_network: :on_net`) ; envoyer une route de secours uniquement hors réseau via une passerelle externe (`terminating_network: :off_net`).

Comportement de fermeture en cas d'échec

Toute valeur autre que `:any` nécessite une réponse HSS définitive. Si la réponse n'est pas définitive, la route **ne correspond pas** (et le moteur essaie la route suivante) :

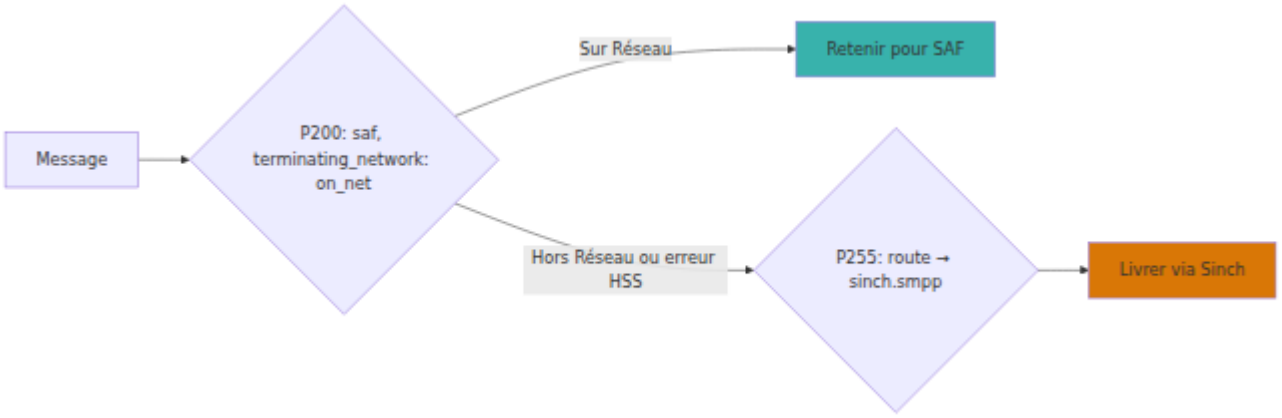
- Diameter désactivé → la route ne correspond pas
- HSS inaccessible / délai d'attente / erreur de transport → la route ne correspond pas
- HSS retourne une erreur autre que *Utilisateur Inconnu* → la route ne correspond pas

Notez que même `:off_net` échoue en fermeture : elle ne correspond qu'à un *Utilisateur Inconnu* explicite (5001), et non à un échec de transport. Cela empêche la fuite de trafic vers une passerelle hors réseau lorsque le HSS est simplement hors service.

Cache de Résultats HSS

Les recherches HSS sont coûteuses (un aller-retour Diameter). Le système met en cache les résultats par MSISDN pour éviter les dips redondants.

- **TTL du cache** : 2 heures (configurable via `hss_cache_ttl_seconds`)
- **Dips paresseux** : le HSS est interrogé uniquement lorsqu'une route évaluée a `originating_network` ou `terminating_network` défini sur `:on_net` ou `:off_net`. Les routes qui laissent les deux à `:any` ne déclenchent jamais un dip.
- **Partagé entre les routes** : si plusieurs routes ont besoin du même statut MSISDN, un seul dip HSS se produit
- **Re-route forcée** : le cache est contourné (un nouveau dip est effectué) lorsqu'un message est explicitement re-routé



Journalisation de l'Évaluation des Routes

Chaque route évaluée pour un message est enregistrée comme un événement `route_evaluated` sur ce message. Le journal comprend :

- ID de la route, action et priorité
- Statut de réussite/échec pour chaque critère de correspondance (y compris `enum_result` et les vérifications HSS `orig_network`/`term_network`)
- Détail de ce qui a été comparé (modèle regex vs numéro, nom SMSC vs source, réseau requis vs état HSS réel, etc.)

Exemples d'entrées de journal d'événements :

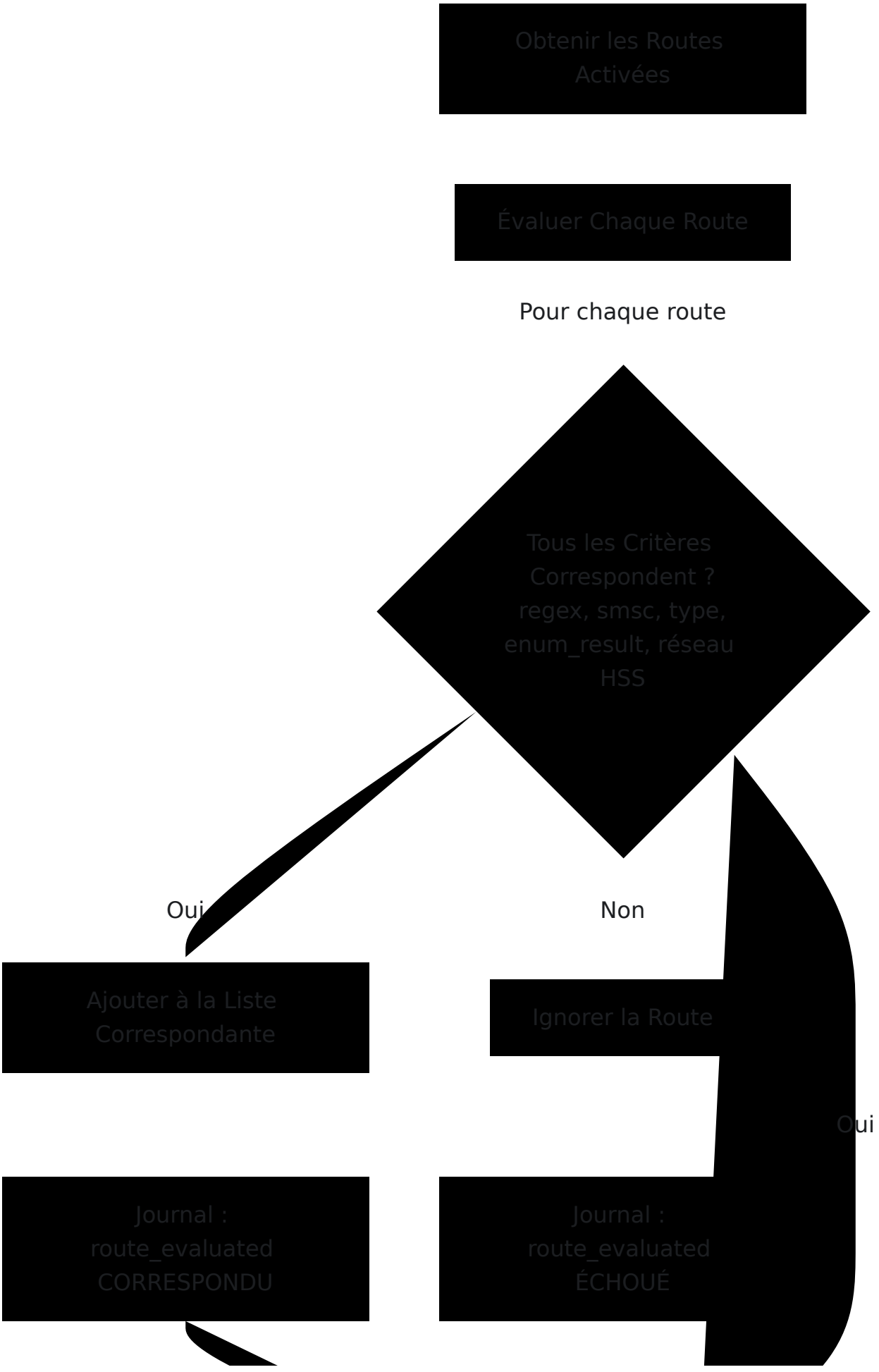
```
Route #4 [saf] CORRESPONDU : P200 "Retenir sur réseau" |
  calling_regex: ✓ (joker), called_regex: ✓ (joker),
  source_smsc: ✓ (joker), source_type: ✓ (joker),
  enum_result: ✓ (joker),
  term_network: ✓ (terminating nécessite on_net, +15551234 est sur
réseau)

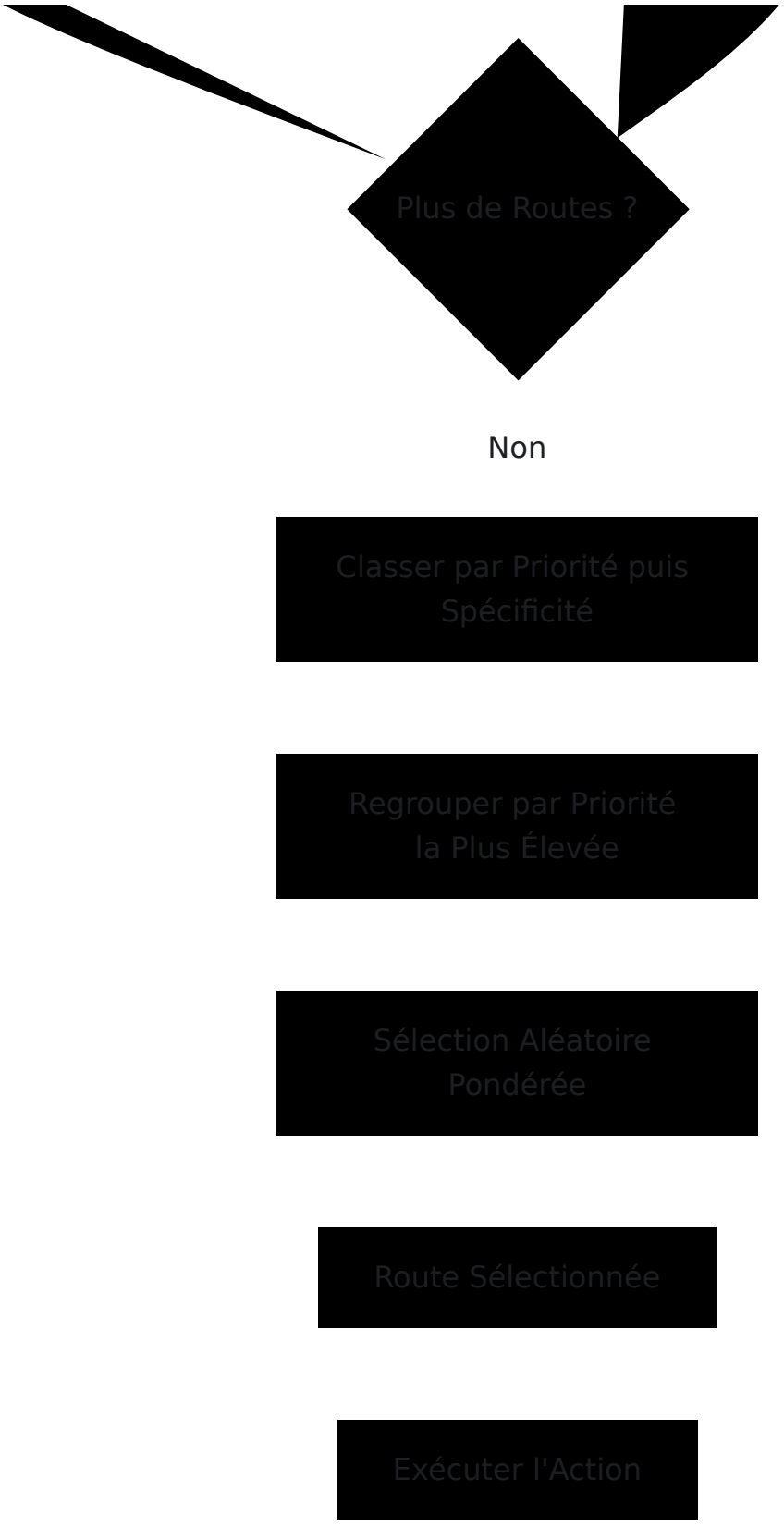
Route #3 [route] ÉCHOUÉ : P255 "Secours Sinch" → sinch.smpp |
  calling_regex: ✓ (joker), called_regex: ✓ (joker),
  source_smsc: ✓ (joker), source_type: ✗ (ims vs
circuit_switched),
  enum_result: ✓ (joker), term_network: ✓ (any)
```

Ces événements sont visibles dans le journal d'activité des messages du panneau de contrôle et via l'API des Événements.

Algorithme d'Évaluation des

Routes


Obtenir les Routes Activées
Évaluer Chaque Route
Pour chaque route
Tous les Critères Correspondent ? regex, smsc, type, enum_result, réseau HSS
Oui
Non
Ajouter à la Liste Correspondante
Ignorer la Route
Oui
Journal : route_evaluated CORRESPONDU
Journal : route_evaluated ÉCHOUÉ



La vérification HSS sur réseau/hors réseau fait partie de **la correspondance des critères** (une route qui échoue ne correspond tout simplement pas) - il n'y a pas d'étape HSS séparée après la sélection.

Évaluation de Spécificité

Lorsque plusieurs routes correspondent, la spécificité détermine laquelle est préférée au sein du même niveau de priorité :

Critère	Points
Longueur du modèle `called_regex`	longueur × 100
Longueur du modèle `calling_regex`	longueur × 50
`source_smsc` spécifié	+25
`enum_result_domain` spécifié	+15
`source_type` spécifié	+10
`enum_domain` spécifié	+5

Des modèles plus longs et plus spécifiques l'emportent toujours sur les jokers.

Configuration

Chargement des Routes depuis la Configuration

Les routes définies dans `config/runtime.exs` sous `:sms_routes` sont chargées au premier démarrage lorsque la table de routage est vide. Les redémarrages ultérieurs conservent les routes ajoutées via l'API.

```
config :sms_c, :sms_routes, [
  %{
    called_regex: ~r/^15555/,
    dest_smsc: "example-gw",
    action: :route,
    priority: 100,
    description: "Routage national exemple"
  },
  %{
    action: :saf,
    terminating_network: :on_net,
    priority: 200,
    description: "Retenir les abonnés sur réseau pour stockage et
transfert"
  },
  %{
    action: :auto_reply,
    called_regex: ~r/^911$/,
    auto_reply_message: "Urgent : Veuillez composer le 911.",
    priority: 1,
    description: "Réponse automatique 911"
  }
]
```

Paramètres de Configuration des Routes

Paramètre	Type	Requis	Par Défaut	Des
`action`	Atome/Chaîne	Non	`:route`	Action c `:route` `:auto_`
`calling_regex`	Regex	Non	`nil`	Modèle le nume `nil` co tous
`called_regex`	Regex	Non	`nil`	Modèle le nume destina corresp
`source_smsc`	Chaîne	Non	`nil`	Filtrer p SMSC s corresp
`dest_smsc`	Chaîne	Conditionnel	`nil`	Requis Optionr Non uti `drop`/`a`
`source_type`	Atome	Non	`nil`	Filtrer p source `:circu` `:smpp`
`enum_domain`	Chaîne	Non	`nil`	Étiquett domain la route la spéci

Paramètre	Type	Requis	Par Défaut	Des
				définit p de la re
`enum_result_domain`	Chaîne	Non	`nil`	Corresp uniquer résultat ENUM r domain corresp
`auto_reply_message`	Chaîne	Conditionnel	`nil`	Requis l'action `:auto_`
`charged`	Atome	Non	`:default`	Politiqu factura `:no`, `:c`
`weight`	Entier	Non	`100`	Poids d' de char
`priority`	Entier	Non	`100`	Priorité (1–255, plus éle
`description`	Chaîne	Non	`""`	Descrip par l'ho
`enabled`	Booléen	Non	`true`	Indique est acti

Paramètre	Type	Requis	Par Défaut	Des
`originating_network`	Atome	Non	`:any`	Exigenc MSISDN `:any`, `:off_n`
`terminating_network`	Atome	Non	`:any`	Exigenc MSISDN destina `:on_ne`

Configuration du Cache HSS

Paramètre	Type	Par Défaut	Description
`hss_cache_ttl_seconds`	Entier	`7200`	Durée de vie des résultats de recherche HSS mis en cache (secondes). Défini dans `runtime.exs`

API

Créer une Route

```
POST /api/routes
```

```
{
  "action": "saf",
  "terminating_network": "on_net",
  "priority": 200,
  "description": "Retenir les abonnés sur réseau pour stockage et
transfert"
}
```

Lister les Routes

```
GET /api/routes
```

Mettre à Jour une Route

```
PUT /api/routes/:id
```

Supprimer une Route

```
DELETE /api/routes/:id
```

Importer/Exporter

```
POST /api/routes/import
GET /api/routes/export
```

L'importation prend en charge les modes `merge` (ajouter à l'existant) et `replace` (vider et remplacer tout).

Dépannage

Aucune Route Trouvée

Symptômes : Le message reste en attente avec `dest_smsc: null`, le journal des événements montre `sms_routing_failed`

Causes possibles :

- Aucune route activée ne correspond aux critères du message
- Les modèles regex de route ne correspondent pas au format du numéro (par exemple, code de pays manquant)
- Chaque route candidate a une exigence `terminating_network`/`originating_network` que le HSS n'a pas satisfaite (hors réseau, ou une erreur HSS sous échec fermé), et aucune route de secours `:any` n'existe

Résolution :

1. Vérifiez le journal des événements du message - chaque route évaluée est enregistrée avec réussite/échec par critère (y compris la vérification du réseau HSS)
2. Utilisez le simulateur de routage dans le panneau de contrôle pour tester avec les mêmes numéros
3. Assurez-vous qu'une route de secours générale (`originating_network: :any`, `terminating_network: :any`) existe à un numéro de priorité élevé (par exemple, 255)

Abonné Sur Réseau Routé Hors Réseau

Symptômes : Les messages vers des abonnés sur réseau sont livrés via une passerelle externe au lieu d'être retenus/livrés localement

Causes possibles :

- Aucune route avec `terminating_network: :on_net` configurée pour capturer les destinations sur réseau

- La route sur réseau a un *numéro* de priorité plus élevé (préférence inférieure) que la route de secours hors réseau
- Échec de la recherche HSS (vérifiez `hss_dip_error` / télémétrie Sh) - sous échec fermé, la route sur réseau ne correspond pas et le trafic tombe sur la route de secours
- Diameter non activé dans la configuration

Résolution :

1. Ajoutez une route avec `terminating_network: :on_net` à un numéro de priorité inférieur à celui de la route de secours hors réseau
2. Vérifiez que Diameter est activé et que les pairs HSS sont connectés via le panneau de contrôle
3. Vérifiez l'état du cache HSS et les résultats de dip récents dans le journal des événements

Cache HSS Obsolète

Symptômes : Abonné porté ou provisionné mais toujours routé en fonction de l'ancien statut sur réseau/hors réseau

Résolution :

1. Re-routez le message via l'API - la re-route forcée contourne le cache HSS
2. Attendez l'expiration du TTL du cache (par défaut 2 heures)
3. Réduisez `hss_cache_ttl_seconds` si des mises à jour plus rapides sont nécessaires

Guide de Dépannage SMS-C

← Retour à l'Index de Documentation

Guide complet pour diagnostiquer et résoudre les problèmes courants de SMS-C.

Table des Matières

- Outils de Diagnostic
- Problèmes de Livraison de Messages
- Problèmes de Routage
- Problèmes de Performance
- Problèmes de Base de Données
- Problèmes de Connexion Frontend
- Problèmes de Facturation
- Problèmes de Recherche ENUM
- Problèmes de Cluster
- Problèmes d'API
- Problèmes d'Interface Web
- Problèmes de Ressources Système

Outils de Diagnostic

Vérification Rapide de la Santé

```
# 1. Vérifier l'état de l'API
curl https://api.example.com:8443/api/status

# 2. Vérifier le point de terminaison des métriques Prometheus
curl https://api.example.com:9568/metrics | grep sms_c

# 3. Vérifier les journaux de l'application
tail -f /var/log/sms_c/application.log

# 4. Vérifier l'état du processus
systemctl status omnimessage

# 5. Vérifier la connectivité à la base de données SQL CDR
(MySQL/MariaDB)
mysql -u sms_user -p -h db.example.com -e "SELECT 1"

# Pour PostgreSQL :
# psql -U sms_user -h db.example.com -d sms_c_prod -c "SELECT 1"
```

Analyse des Journaux

Voir les Erreurs Récentes :

```
# Dernières 100 entrées de journal de niveau erreur
tail -1000 /var/log/sms_c/application.log | grep "\[error\]"

# Rechercher des motifs d'erreur spécifiques
grep "routing_failed" /var/log/sms_c/application.log

# Trouver des erreurs de base de données SQL
grep -i "database\|sql\|ecto" /var/log/sms_c/application.log |
grep error
```

Surveiller les Journaux en Temps Réel :

```
# Suivre les journaux avec filtre
tail -f /var/log/sms_c/application.log | grep -E "
(error|warning|critical)"
```

Requêtes de Métriques

Vérifier le Taux de Traitement des Messages :

```
# Messages par seconde
rate(sms_c_message_received_count[5m])

# Taux de succès de livraison
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])
```

Vérifier l'État de la File d'Attente :

```
# Profondeur actuelle de la file d'attente
sms_c_queue_size_pending

# Âge du message le plus ancien (secondes)
sms_c_queue_oldest_message_age_seconds
```

Vérifier la Performance du Système :

```
# Latence de traitement des messages (p95)
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)

# Latence de routage (p95)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
```

Problèmes de Livraison de Messages

Messages Non Livrés

Symptômes :

- Messages bloqués dans l'état "en file d'attente"/"retard"
- Nombre élevé de messages non livrés
- Pas de notifications de livraison

Étapes de Diagnostic :

1. **Vérifier les Connexions Frontend** :

```
curl https://api.example.com:8443/api/frontends/active
```

Attendu : Liste des frontends actifs
Problème : Liste vide ou frontends attendus manquants

2. **Vérifier la File d'Attente des Messages** :

Accéder à l'Interface Web : `/message_queue`

- Filtrer par statut : "en file d'attente" ou "retard"
- Vérifier la valeur `dest_smsc`
- Vérifier que `deliver_after` n'est pas dans le futur

3. **Vérifier le Routage** :

Accéder à l'Interface Web : `/simulator`

- Tester avec des paramètres de message réels
- Vérifier que la route correspond et que la destination est correcte

4. **Vérifier le Polling Frontend** :

Examiner les journaux du système frontend :

- Le frontend interroge-t-il `/api/messages` ?
- Le frontend envoie-t-il correctement l'en-tête `smsc` ?

Solutions :

Aucun Frontend Connecté :

```
# Vérifier l'état du système frontend
systemctl status frontend_service

# Vérifier que le frontend peut atteindre l'API
curl -k https://api.example.com:8443/api/status

# Enregistrer manuellement le frontend
curl -X POST https://api.example.com:8443/api/frontends/register \
  -H "Content-Type: application/json" \
  -d '{
    "frontend_name": "test_gateway",
    "frontend_type": "smpp",
    "ip_address": "10.0.1.50"
  }'
```

Messages Routés vers le Mauvais SMSC :

- Examiner la configuration de routage
- Vérifier les priorités de route
- Tester dans le simulateur de routage
- Vérifier que le nom du frontend correspond à `dest_smsc` dans les messages

Messages Planifiés pour le Futur :

- Vérifier le timestamp `deliver_after`
- Réinitialiser si nécessaire :

```
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{"deliver_after": "2025-10-30T12:00:00Z"}'
```

Messages Échouant avec Réessais

Symptômes :

- Compteur `delivery_attempts` augmentant
- Messages avec un nombre élevé de tentatives (> 3)
- Retards exponentiels

Étapes de Diagnostic :

1. **Vérifier le Journal des Événements** :

```
curl https://api.example.com:8443/api/events/12345
```

Rechercher :

- Événements d'échec de livraison
- Descriptions d'erreur
- Timestamps de réessai

2. **Vérifier les Journaux Frontend** :

- Pourquoi le frontend échoue-t-il à livrer ?
- Erreurs réseau ?
- Erreurs de protocole ?
- Système en aval indisponible ?

Solutions :

Problèmes Réseau Temporaires :

- Attendre le réessai (automatique)
- Surveiller pour une livraison réussie

Échecs Persistants :

```
# Routage vers une passerelle alternative
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{"dest_smsc": "backup_gateway"}'

# Réinitialiser le compteur de réessai
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{"delivery_attempts": 0, "deliver_after": "2025-10-
30T12:00:00Z"}'
```

Numéro de Destination Invalide :

- Vérifier le format du numéro
- Vérifier les règles de traduction de numéro
- Supprimer le message s'il est vraiment invalide

Messages en Lettre Morte

Symptômes :

- `deadletter: true` dans le message
- Messages dépassant le temps d'expiration
- Statut toujours "en file d'attente"/"retard" (les livraisons échouées ne sont pas mises en lettre morte ; elles cessent d'être servies une fois dépassé `expires`)

Étapes de Diagnostic :

1. **Trouver les Messages en Lettre Morte** :

Accéder à l'Interface Web : `/message_queue`

- Filtrer par statut expiré
- Vérifier les timestamps d'expiration

2. **Vérifier Pourquoi Expiré** :

- Examiner le journal des événements

- Vérifier l'historique des tentatives de livraison
- Vérifier que le routage a été réussi

Solutions :

Étendre l'Expiration :

```
# Ajouter 24 heures à l'expiration
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
  -H "Content-Type: application/json" \
  -d '{"expires": "2025-10-31T12:00:00Z", "deadletter": false}'
```

Problèmes de Routage

Aucune Route Trouvée

Symptômes :

- Erreur : `no_route_found`
- La métrique `sms_c_routing_failed_count` augmente
- Le journal des événements montre "routing_failed"

Étapes de Diagnostic :

1. **Vérifier que les Routes Existent** :

Accéder à l'Interface Web : `/sms_routing`

- Vérifier que les routes sont configurées
- Vérifier qu'au moins une route est activée

2. **Tester le Routage** :

Accéder à l'Interface Web : `/simulator`

- Entrer les paramètres de message (numéro appelant, numéro appelé, SMSC source)
- Examiner les résultats d'évaluation

- Vérifier pourquoi les routes ne correspondaient pas

3. **Vérifier les Critères de Route** :

- Les préfixes doivent-ils correspondre ?
- Le filtre SMSC source est-il trop restrictif ?
- Toutes les routes sont-elles désactivées ?

Solutions :

Aucune Route Configurée :

Ajouter une route de secours :

```
Préfixe d'Appel : (vide)
Préfixe Appelé : (vide)
SMSC Source : (vide)
SMSC Destinataire : default_gateway
Priorité : 255
Poids : 100
Activé : ✓
Description : Route par défaut de secours
```

Routes Trop Spécifiques :

Ajouter une route plus large :

```
Préfixe Appelé : +
SMSC Destinataire : international_gateway
Priorité : 200
Poids : 100
Activé : ✓
Description : Route internationale par défaut
```

Toutes les Routes Désactivées :

- Activer les routes appropriées via l'Interface Web
- Vérifier que la configuration n'a pas accidentellement désactivé les routes

Mauvaise Route Sélectionnée

Symptômes :

- Messages routés vers une destination inattendue
- Mauvaise passerelle recevant le trafic
- Équilibrage de charge ne distribuant pas comme prévu

Étapes de Diagnostic :

1. **Utiliser le Simulateur de Routage** :

Accéder à l'Interface Web : `/simulator`

- Tester avec des paramètres de message réels
- Examiner la section "Toutes les Correspondances"
- Vérifier les priorités et les scores de spécificité

2. **Vérifier les Priorités de Route** :

- Un nombre plus bas = une priorité plus élevée
- Les routes sont évaluées par ordre de priorité
- Au sein de la même priorité, les poids s'appliquent

3. **Vérifier la Spécificité de la Route** :

Notation de spécificité :

- Préfixe appelé plus long : +100 points par caractère
- Préfixe appelant plus long : +50 points par caractère
- SMSC source spécifié : +25 points
- Type source spécifié : +10 points
- Domaine ENUM spécifié : +15 points

Solutions :

Ajuster les Priorités :

Faire une route spécifique de priorité plus élevée :

```
Route Premium :
  Préfixe Appelé : +1555
  Priorité : 10 (haute priorité)

Route Générale :
  Préfixe Appelé : +1
  Priorité : 50 (priorité plus basse)
```

Ajuster les Poids :

Changer la distribution de l'équilibrage de charge :

```
Principal (70%) :
  Poids : 70

De Secours (30%) :
  Poids : 30
```

Ajouter une Route Plus Spécifique :

Remplacer la route générale pour un cas spécifique :

```
Route Spécifique :
  Préfixe Appelé : +15551234
  SMSC Destinataire : dedicated_gateway
  Priorité : 1

Route Générale :
  Préfixe Appelé : +1
  SMSC Destinataire : general_gateway
  Priorité : 50
```

Réponse Automatique Ne Fonctionne Pas

Symptômes :

- Route de réponse automatique configurée mais ne se déclenche pas
- Aucun message de réponse envoyé

- Journal des événements manquant l'événement de réponse automatique

Étapes de Diagnostic :

1. **Vérifier la Configuration de la Route** :

- `action: :auto_reply`
- `auto_reply_message` contient du texte
- Route est activée
- Route correspond aux critères du message

2. **Tester dans le Simulateur** :

- Vérifier que la route est sélectionnée
- Vérifier l'indication "auto_reply"

3. **Vérifier le Journal des Événements** :

```
curl https://api.example.com:8443/api/events/12345 | grep
auto_reply
```

Solutions :

Route Ne Correspond Pas :

- Élargir les critères (supprimer les filtres)
- Vérifier la priorité (doit être plus élevée que les routes normales)
- Vérifier le statut activé

Message Non Défini :

Modifier la route, ajouter un message :

```
Réponse Automatique : ✓
Message de Réponse Automatique : "Merci pour votre message. Nous
répondrons bientôt."
```

Mauvaise Priorité :

Les routes de réponse automatique doivent avoir une haute priorité (nombre bas) :

```
Route de Réponse Automatique :
  Priorité : 10

Route Normale :
  Priorité : 50
```

Problèmes de Performance

Latence Élevée de Traitement des Messages

Symptômes :

- `sms_c_message_processing_stop_duration` p95 > 1000ms
- Réponses API lentes
- File d'attente qui s'accumule

Étapes de Diagnostic :

1. **Vérifier les Latences des Composants** :

```
# Latence de routage
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)

# Latence de recherche ENUM
histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)

# Latence de facturation
histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)

# Latence de livraison
histogram_quantile(0.95, sms_c_delivery_succeeded_duration_bucket)
```

2. **Vérifier les Ressources Système** :

```
# Utilisation du CPU
top -b -n 1 | grep sms_c

# Utilisation de la mémoire
ps aux | grep beam.smp
```

Solutions :

Routage Lent (Beaucoup de routes) :

- Réduire le nombre de routes activées
- Combiner des routes similaires
- Optimiser les critères de routage

Recherches ENUM Lentes :

- Vérifier la latence du serveur DNS
- Augmenter le délai d'attente
- Utiliser des serveurs DNS plus rapides/proches
- Désactiver ENUM si non nécessaire

Facturation Lente :

- Vérifier la performance de l'OCS
- Augmenter le délai d'attente de l'OCS
- Désactiver la facturation si non nécessaire
- Utiliser la facturation asynchrone

Base de Données Lente :

- Augmenter la taille du pool de connexions
- Ajouter des index
- Optimiser les requêtes
- Améliorer les ressources de la base de données

Modifications de Configuration :

```
# config/config.exs
# Augmenter la taille du lot pour le débit
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# Augmenter le pool de base de données
config :sms_c, SmsC.Repo,
  pool_size: 50
```

Faible Débit de Messages

Symptômes :

- Traitement < 100 msg/sec
- Utilisation de l'API asynchrone mais toujours lente
- Temps de réponse API élevés

Étapes de Diagnostic :

1. **Vérifier le Travailleur de Lot** :

```
# Dans la console de production (iex)
SmsC.Messaging.BatchInsertWorker.stats()
```

Rechercher :

- `current_queue_size` proche du maximum
- `flush_errors` > 0
- `last_flush_duration_ms` très élevé

2. **Vérifier les Goulots d'Étranglement** :

```
# Temps de requête de base de données
ecto_pools_query_time

# Temps de file d'attente du pool de connexions
ecto_pools_queue_time
```

Solutions :

Goulot d'Étranglement de Base de Données :

Augmenter la taille du pool :

```
config :sms_c, SmsC.Repo,
  pool_size: 50  # Augmenter de 20
```

Configuration de Lot :

Ajuster pour le débit :

```
config :sms_c,
  batch_insert_batch_size: 200,  # Lots plus grands
  batch_insert_flush_interval_ms: 200  # Intervalle plus long
```

Utiliser le Point de Terminaison Asynchrone :

```
# Débit élevé : utiliser /create_async
curl -X POST
https://api.example.com:8443/api/messages/create_async

# PAS : /api/messages (synchronisé)
```

Accumulation de File d'Attente Croissante

Symptômes :

- `sms_c_queue_size_pending` augmentant
- Âge du message le plus ancien augmentant

- Le traitement ne peut pas suivre le taux entrant

Étapes de Diagnostic :

1. **Vérifier le Taux Entrant vs Taux de Livraison** :

```
# Taux entrant
rate(sms_c_message_received_count[5m])

# Taux de livraison
rate(sms_c_delivery_succeeded_count[5m])
```

2. **Vérifier la Capacité du Frontend** :

- Les frontends interrogent-ils assez fréquemment ?
- Les frontends traitent-ils les messages assez rapidement ?
- Des erreurs de frontend ?

3. **Vérifier le Taux de Succès de Livraison** :

```
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_attempted_count[5m])
```

Solutions :

Frontends Ne Pollent Pas :

- Vérifier la connectivité du frontend
- Vérifier l'intervalle de polling (doit être de 5 à 10 secondes)
- Redémarrer les services frontend

Frontends Trop Lents :

- Ajouter plus d'instances frontend
- Optimiser le traitement frontend
- Augmenter la concurrence frontend

Taux de Réessai Élevé :

- Enquêter sur les échecs de livraison
- Corriger les problèmes en aval
- Routage vers des passerelles alternatives

Pointe Temporaire :

- Attendre que la file d'attente se vide
- Surveiller jusqu'à ce que cela redevienne normal
- Envisager des mises à niveau de capacité si récurrent

Problèmes de Base de Données

Échecs de Connexion

Symptômes :

- Erreur : "impossible de se connecter à la base de données"
- L'API renvoie des erreurs 500
- L'application ne démarre pas

Étapes de Diagnostic :

1. **Vérifier l'État de la Base de Données SQL CDR** :

```
# MySQL/MariaDB
systemctl status mysql

# PostgreSQL
systemctl status postgresql

# Tester la connectivité (MySQL/MariaDB)
mysql -u sms_user -p -h db.example.com -e "SELECT 1"

# Tester la connectivité (PostgreSQL)
psql -U sms_user -h db.example.com -d sms_c_prod -c "SELECT 1"
```

2. **Vérifier le Réseau** :

```
# Pinger l'hôte de la base de données
ping db.example.com

# Vérifier la connectivité au port (MySQL/MariaDB : 3306,
PostgreSQL : 5432)
telnet db.example.com 3306
# ou
telnet db.example.com 5432
```

3. **Vérifier les Identifiants** :

```
# Vérifier les variables d'environnement
echo $DB_USERNAME
echo $DB_HOSTNAME
echo $DB_PORT

# Essayer la connexion manuelle avec les mêmes identifiants
(MySQL/MariaDB)
mysql -u $DB_USERNAME -p$DB_PASSWORD -h $DB_HOSTNAME

# Pour PostgreSQL :
# psql -U $DB_USERNAME -h $DB_HOSTNAME -d sms_c_prod
```

Solutions :

Base de Données Hors Service :

```
# Démarrer la base de données (MySQL/MariaDB)
systemctl start mysql

# Démarrer la base de données (PostgreSQL)
systemctl start postgresql
```

Identifiants Incorrects :

Mettre à jour la configuration :

```
export DB_USERNAME=correct_user
export DB_PASSWORD=correct_password

# Redémarrer l'application
systemctl restart omnimessage
```

Problème Réseau :

- Vérifier les règles de pare-feu
- Vérifier les groupes de sécurité (cloud)
- Vérifier la connectivité VPN/réseau

Pool de Connexion Épuisé :

Augmenter la taille du pool :

```
config :sms_c, SmsC.Repo,
  pool_size: 50  # Augmenter de la valeur actuelle
```

Requêtes Lentes

Symptômes :

- Temps de requête de base de données élevé
- Réponses API lentes
- File d'attente du pool de connexions qui s'accumule

Étapes de Diagnostic :

1. **Vérifier le Journal des Requêtes Lentes** :

```sql
-- MySQL/MariaDB : Activer le journal des requêtes lentes
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 1;  -- Journaliser les requêtes > 1 seconde

-- Voir les requêtes lentes (MySQL/MariaDB)
SELECT * FROM mysql.slow_log ORDER BY query_time DESC LIMIT 10;

-- PostgreSQL : Activer le journal des requêtes lentes dans postgresql.conf
-- log_min_duration_statement = 1000  # millisecondes
-- Puis vérifier les journaux PostgreSQL
```

2. **Vérifier les Index Manquants** :

```sql
-- Vérifier les index de table
SHOW INDEX FROM message_queues;

-- Index attendus :
-- - source_smsc
-- - dest_smsc
-- - send_time
-- - inserted_at
```

3. **Vérifier les Statistiques de Table** :

```sql
-- Tailles de table (MySQL/MariaDB)
SELECT
  table_name,
  table_rows,
  ROUND(data_length / 1024 / 1024, 2) AS data_mb,
  ROUND(index_length / 1024 / 1024, 2) AS index_mb
FROM information_schema.tables
WHERE table_schema = 'sms_c_prod';

-- Tailles de table (PostgreSQL)
-- SELECT schemaname, tablename,
--
pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename))
AS size
-- FROM pg_tables WHERE schemaname = 'public';
```

Solutions :

Index Manquants :

```sql
CREATE INDEX idx_message_queues_source_smsc ON
message_queues(source_smsc);
CREATE INDEX idx_message_queues_dest_smsc ON
message_queues(dest_smsc);
CREATE INDEX idx_message_queues_send_time ON
message_queues(send_time);
CREATE INDEX idx_message_queues_status ON message_queues(status);
```

Fragmentation de Table :

```sql
-- MySQL/MariaDB
OPTIMIZE TABLE message_queues;
OPTIMIZE TABLE frontend_registrations;

-- PostgreSQL
-- VACUUM ANALYZE message_queues;
-- VACUUM ANALYZE frontend_registrations;
```

Trop de Données :

Nettoyer les anciens enregistrements :

```sql
-- Supprimer les messages livrés de plus de 30 jours
DELETE FROM message_queues
WHERE status = 'delivered'
AND deliver_time < DATE_SUB(NOW(), INTERVAL 30 DAY)
LIMIT 10000;
```

Espace Disque Plein

Symptômes :

- Erreur : "Disque plein"
- Impossible d'écrire dans la base de données
- L'application plante

Étapes de Diagnostic :

1. **Vérifier l'Utilisation du Disque** :

```
df -h

# Vérifier le répertoire de la base de données SQL (MySQL/MariaDB)
du -sh /var/lib/mysql

# Vérifier le répertoire de la base de données SQL (PostgreSQL)
du -sh /var/lib/postgresql
```

2. **Trouver les Fichiers Larges** :

```
# Trouver les fichiers les plus volumineux (MySQL/MariaDB)
find /var/lib/mysql -type f -exec du -h &amp;#123;&amp;#125; + | sort -rh
| head -20

# Trouver les fichiers les plus volumineux (PostgreSQL)
find /var/lib/postgresql -type f -exec du -h &amp;#123;&amp;#125; + | sort
-rh | head -20

# Vérifier les fichiers journaux
du -sh /var/log/sms_c/*
```

Solutions :

Nettoyer les Anciennes Données :

```sql
-- Supprimer les anciens messages
DELETE FROM message_queues
WHERE inserted_at < DATE_SUB(NOW(), INTERVAL 90 DAY)
LIMIT 100000;
```

Faire Tourner les Journaux :

```
# Forcer logrotate
logrotate -f /etc/logrotate.d/sms_c

# Effacer les anciens fichiers journaux
find /var/log/sms_c -name "*.log.*" -mtime +30 -delete
```

Étendre le Disque :

- Redimensionner le volume (cloud)
- Ajouter un nouveau disque et étendre le volume
- Déplacer les données vers un disque plus grand

Problèmes de Connexion Frontend

Frontend Ne S'affiche Pas Comme Actif

Symptômes :

- Le statut du frontend montre "expiré"
- Le frontend n'est pas dans la liste active
- Les messages ne sont pas livrés au frontend

Étapes de Diagnostic :

1. **Vérifier l'Inscription** :

```
curl https://api.example.com:8443/api/frontends/active | grep
frontend_name
```

2. **Vérifier les Journaux Frontend** :

- Le frontend appelle-t-il `/api/frontends/register` ?
- Des erreurs API ?
- Fréquence d'inscription (doit être toutes les 60s)

3. **Vérifier les Journaux de l'API** :

```
grep "frontend.*register" /var/log/sms_c/application.log | tail
-20
```

Solutions :

Frontend Ne S'inscrit Pas :

Tester l'inscription manuelle :

```
curl -X POST https://api.example.com:8443/api/frontends/register \
  -H "Content-Type: application/json" \
  -d '&amp;#123;
    "frontend_name": "uk_gateway",
    "frontend_type": "smpp",
    "ip_address": "10.0.1.50"
  &amp;#125;'
```

Si réussi, le problème est dans le code/configuration du frontend.

Inscription Expirant :

Les frontends expirent après 90 secondes. Assurez-vous de l'inscription toutes les 60 secondes :

```
# Le frontend doit appeler l'inscription toutes les 60 secondes
while True:
    register_with_smsc()
    time.sleep(60)
```

Problèmes Réseau :

- Vérifier le pare-feu entre le frontend et l'API
- Vérifier la résolution DNS
- Tester avec curl depuis le serveur frontend

Frontend Se Connecte/Déconnecte Répétitivement

Symptômes :

- Le statut du frontend bascule entre actif/expiré
- Compte d'inscription élevé dans l'historique
- Connexion instable

Étapes de Diagnostic :

1. **Vérifier la Santé du Frontend** :

- Le processus frontend est-il stable ?
- Des plantages ou redémarrages ?
- Problèmes de ressources (CPU/mémoire) ?

2. **Vérifier la Stabilité du Réseau** :

```
# Vérifier la perte de paquets
ping -c 100 api.example.com

# Vérifier les réinitialisations de connexion
netstat -s | grep -i reset
```

3. **Vérifier le Timing d'Inscription** :

- Trop fréquent ? (toutes les quelques secondes)
- Trop peu fréquent ? (> 90 secondes)

Solutions :

Frontend Instable :

- Corriger les problèmes de l'application frontend
- Augmenter les ressources du frontend
- Vérifier les journaux frontend pour des erreurs

Problèmes Réseau :

- Vérifier la connectivité intermittente
- Examiner les journaux de pare-feu
- Vérifier les vérifications de santé de l'équilibreur de charge

Mauvais Intervalle d'Inscription :

Corriger l'intervalle :

```
REGISTRATION_INTERVAL = 60  # secondes
```

Charging est effectué via **Diameter Ro** (Contrôle de Crédit). Notez que le chargement **échoue ouvert** : une erreur OCS ou une pile Diameter non configurée enregistre un échec mais délivre toujours le message. Seule une réponse définitive hors équilibre rejette un message (statut `:balance_rejected`).

Symptômes :

- `sms_c_charging_failed_count` en augmentation
- Le journal des événements montre "charging_failed" ou "balance_rejected"
- Messages marqués avec le statut `:balance_rejected` (hors équilibre), ou avertissements dans le journal concernant la pile Diameter non activée

Étapes de Diagnostic :

1. **Vérifiez l'état du pair Diameter Ro** (CER/CEA établi vers le DRA/OCS) :

```
# Dans une console distante
:diameter.service_info(:omnimessage, :connections)
```

2. **Confirmez que la pile est activée et que l'application `:ro` est enregistrée** :

```
grep -E "diameter_enabled|application_name: :ro"
config/runtime.exs
```

Une ligne de journal `Charging requested but the Diameter Ro stack is not enabled/configured` signifie que le chargement est ignoré (messages autorisés) - activez la pile.

3. **Vérifiez que le profil de tarification** correspond au contexte de tarification configuré sur l'OCS :

```
grep rating_profile config/runtime.exs
```

Solutions :

OCS / DRA Inaccessible :

- Vérifiez la connectivité IP et que le DRA a ce nœud provisionné comme pair Diameter (Origin-Host/Realm).
- Confirmez que le CCR `Service-Context-Id` (`rating_profile`) correspond au profil de tarification SMS de l'OCS.

Gestion hors équilibre (`config :sms_c, :charging`) :

```
config :sms_c, :charging,
  out_of_balance_action: :reject,    # ou :auto_reply
  out_of_balance_reply_text: "...",
  rating_profile: "sms@sms-c"
```

Désactiver temporairement le chargement :

```
config :sms_c,
  default_charging_enabled: false
```

Redémarrez l'application.

Problèmes de Compte :

- Vérifiez que le compte existe dans l'OCS
- Vérifiez que le compte a un solde
- Vérifiez que les plans de tarification sont configurés

Chargement Trop Lent

Symptômes :

- `sms_c_charging_succeeded_duration` p95 > 500ms
- Traitement des messages lent lorsque le chargement est activé
- Rapide lorsque le chargement est désactivé

Étapes de Diagnostic :

1. **Vérifiez la Latence de Chargement** :

```
histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)
```

2. **Vérifiez la performance de l'OCS / DRA** (aller-retour Diameter Ro). Une latence élevée de `sms_c_charging_succeeded_duration` pointe généralement vers une latence de tarification OCS ou un pair DRA lent/flottant.

3. **Vérifiez la Latence Réseau vers le DRA** :

```
# Ping le DRA / hôte OCS
ping -c 10 dra01.example.com
```

Solutions :

OCS Lent :

- Optimisez la configuration de l'OCS
- Ajoutez des ressources OCS
- Utilisez un moteur de tarification plus rapide

Latence Réseau :

- Déployez l'OCS plus près de SMS-C
- Utilisez un chemin réseau direct
- Évitez les VPN/tunnels si possible

Délai Trop Bas :

Augmentez le délai :

```
config :sms_c,
  ocs_timeout: 5000  # 5 secondes
```

Problèmes de Recherche ENUM

Échecs de Recherches ENUM

Symptômes :

- `sms_c_enum_lookup_stop_duration` montrant des échecs
- Le journal des événements montre des erreurs ENUM
- Routes avec `enum_result_domain` ne correspondant pas

Étapes de Diagnostic :

1. **Vérifiez la Configuration ENUM** :

```
grep -A 10 "enum_" config/runtime.exs
```

2. **Testez la Connectivité DNS** :

```
# Testez le serveur DNS
dig @8.8.8.8 e164.arpa

# Testez la requête ENUM
# Pour +15551234567 :
dig @8.8.8.8 NAPTR 7.6.5.4.3.2.1.5.5.5.1.e164.arpa
```

3. **Vérifiez le Serveur DNS** :

```
# Le DNS personnalisé est-il accessible ?
ping 10.0.1.53

# Testez le port
nc -zv 10.0.1.53 53
```

Solutions :

Serveur DNS Inaccessible :

Utilisez un DNS alternatif :

```
config :sms_c,
  enum_dns_servers: [
    &amp;#123;"8.8.8.8", 53&amp;#125;,  # Google Public DNS
    &amp;#123;"1.1.1.1", 53&amp;#125;   # Cloudflare DNS
  ]
```

Domaine ENUM Incorrect :

Mettez à jour le domaine :

```
config :sms_c,
  enum_domains: ["e164.arpa"]  # Utilisez le domaine standard
```

Délai Trop Court :

Augmentez le délai :

```
config :sms_c,
  enum_timeout: 10000  # 10 secondes
```

Désactiver ENUM (si non nécessaire) :

```
config :sms_c,
  enum_enabled: false
```

Problèmes de Cache ENUM

Symptômes :

- Faible taux de réussite du cache (< 70%)
- Taille du cache en croissance illimitée
- Utilisation de la mémoire élevée

Étapes de Diagnostic :

1. **Vérifiez les Statistiques du Cache** :

```
# Taux de réussite du cache
rate(sms_c_enum_cache_hit_count[5m]) /
(rate(sms_c_enum_cache_hit_count[5m]) +
rate(sms_c_enum_cache_miss_count[5m]))

# Taille du cache
sms_c_enum_cache_size_size
```

2. **Vérifiez le Modèle de Trafic** :

- Les numéros se répètent-ils ?
- Le TTL du cache est-il approprié ?

Solutions :

Faible Taux de Réussite (Attendu) :

- Trafic vers des numéros uniques (normal)
- Surveillez mais ne déclenchez pas d'alarme si < 70%

Cache en Croissance :

Effacez le cache via la page de test NAPTR ou redémarrez l'application.

Utilisation Élevée de la Mémoire :

- Attendu avec un grand cache
- Surveillez la mémoire système globale
- Envisagez un ajustement du TTL

Problèmes de Cluster

Nœud Ne Peut Pas Rejoindre le Cluster

Symptômes :

- Nœud unique en cours d'exécution
- Les requêtes de cluster ne retournent que des résultats locaux
- Erreurs de distribution Erlang

Étapes de Diagnostic :

1. **Vérifiez les Noms des Nœuds** :

```
# Dans la console IEx
Node.self()
# Attendu : :sms@node1.example.com

Node.list()
# Attendu : Liste des autres nœuds
```

2. **Vérifiez le Cookie Erlang** :

```
# Vérifiez le fichier cookie
cat ~/.erlang.cookie

# Vérifiez qu'il est identique sur tous les nœuds
```

3. **Vérifiez le Réseau** :

```
# Les nœuds peuvent-ils se joindre ?
ping node2.example.com

# Vérifiez les ports
nc -zv node2.example.com 4369
nc -zv node2.example.com 9100-9200
```

Solutions :

Mismatch de Cookie :

Définissez le même cookie sur tous les nœuds :

```
export ERLANG_COOKIE=same_secret_value_here

# Ou mettez à jour ~/.erlang.cookie
echo "same_secret_value_here" > ~/.erlang.cookie
chmod 400 ~/.erlang.cookie
```

Pare-feu Bloquant :

Ouvrez les ports requis :

```
# EPMD
iptables -A INPUT -p tcp --dport 4369 -j ACCEPT

# Distribution Erlang
iptables -A INPUT -p tcp --dport 9100:9200 -j ACCEPT
```

Problèmes DNS :

Utilisez des adresses IP au lieu de noms d'hôtes :

```
config :sms_c,
  cluster_nodes: [
    :"sms@10.0.1.10",
    :"sms@10.0.1.11"
  ]
```

Split Brain de Cluster

Symptômes :

- Nœuds en cours d'exécution mais déconnectés
- Données différentes sur différents nœuds
- Incohérences Mnesia

Étapes de Diagnostic :

1. **Vérifiez la Connectivité des Nœuds** :

```
# Sur chaque nœud (IEx)
Node.list()
```

2. **Vérifiez Mnesia** :

```
:mnesia.system_info(:running_db_nodes)
```

Solutions :

Reconnecter les Nœuds :

```
# Arrêtez tous les nœuds
systemctl stop omnimessage

# Démarrez d'abord un nœud
systemctl start omnimessage  # Sur node1

# Attendez qu'il démarre complètement, puis démarrez les autres
systemctl start omnimessage  # Sur node2
systemctl start omnimessage  # Sur node3
```

Incohérence Mnesia :

- Exportez les routes depuis le nœud correct
- Arrêtez tous les nœuds
- Supprimez le répertoire Mnesia
- Démarrez les nœuds
- Importez les routes

Problèmes API

API Ne Répond Pas

Symptômes :

- Délai de connexion

- Connexion refusée
- Pas de réponse

Étapes de Diagnostic :

1. **Vérifiez le Processus API** :

```
# L'application est-elle en cours d'exécution ?
systemctl status omnimessage

# Vérifiez les ports à l'écoute
netstat -tlnp | grep 8443
```

2. **Vérifiez le Pare-feu** :

```
# Vérifiez iptables
iptables -L -n | grep 8443

# Testez la connectivité locale
curl -k https://localhost:8443/api/status
```

3. **Vérifiez la Configuration TLS** :

```
# Vérifiez que le certificat existe
ls -l priv/cert/server.crt priv/cert/server.key

# Vérifiez la validité du certificat
openssl x509 -in priv/cert/server.crt -noout -dates
```

Solutions :

Application Non Exécutée :

```
systemctl start omnimessage
```

Pare-feu Bloquant :

```
# Autoriser le port API
iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

Problèmes de Certificat :

Générez un nouveau certificat (voir le Guide de Configuration).

Mauvais Port :

Vérifiez la configuration :

```
grep "port:" config/runtime.exs
```

API Renvoie des Erreurs 500

Symptômes :

- Erreur Interne du Serveur
- Code d'état 500
- Erreur dans les journaux

Étapes de Diagnostic :

1. **Vérifiez les Journaux de l'Application** :

```
tail -100 /var/log/sms_c/application.log | grep "\[error\]"
```

2. **Vérifiez la Base de Données** :

```
mysql -u sms_user -p -e "SELECT 1"
```

3. **Vérifiez les Ressources** :

```
# Mémoire
free -h

# CPU
top -b -n 1

# Disque
df -h
```

Solutions :

Base de Données Indisponible :

- Démarrez la base de données
- Corrigez la configuration de connexion

Hors Mémoire :

- Redémarrez l'application
- Augmentez la mémoire système
- Vérifiez les fuites de mémoire

Erreur d'Application :

- Vérifiez l'erreur spécifique dans les journaux
- Corrigez le problème de configuration
- Redémarrez l'application

Problèmes d'Interface Web

Impossible d'Accéder à l'Interface Web

Symptômes :

- Délai de connexion
- 404 Non Trouvé
- La page ne se charge pas

Étapes de Diagnostic :

1. **Vérifiez l'État de l'Application** :

```
systemctl status omnimessage
```

2. **Vérifiez le Port** :

```
netstat -tlnp | grep 80
```

3. **Vérifiez l'URL** :

- Nom d'hôte correct ?
- Port correct ?
- HTTP vs HTTPS ?

Solutions :

Mauvais Port :

Vérifiez la configuration :

```
grep "control_panel" config/runtime.exs
```

Accédez au bon port (par défaut : 80 ou 4000).

Application Non Exécutée :

```
systemctl start omnimessage
```

Pare-feu :

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

LiveView Ne Se Met Pas à Jour

Symptômes :

- La page se charge mais ne se met pas à jour
- Les données sont obsolètes
- Erreurs WebSocket dans la console du navigateur

Étapes de Diagnostic :

1. **Vérifiez la Console du Navigateur** :

- Ouvrez les Outils de Développeur (F12)
- Recherchez les erreurs WebSocket
- Vérifiez l'onglet réseau pour les requêtes échouées

2. **Vérifiez la Configuration du Proxy** :

Si vous utilisez un proxy inverse, assurez-vous du support WebSocket :

```
location /live &#123;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
&#125;
```

Solutions :

WebSocket Bloqué :

- Configurez le proxy pour WebSocket
- Vérifiez le pare-feu
- Vérifiez les extensions du navigateur

Rafraîchir la Page :

- Rafraîchissement forcé (Ctrl+F5)
- Effacez le cache du navigateur

Problèmes de Ressources Système

Utilisation Élevée du CPU

Symptômes :

- CPU constamment > 80%
- Système lent
- Application non réactive

Étapes de Diagnostic :

1. **Vérifiez le Processus** :

```
top -b -n 1 | grep beam.smp
```

2. **Vérifiez les Métriques** :

```
# Taux de traitement des messages
rate(sms_c_message_received_count[5m])

# Opérations de routage
rate(sms_c_routing_route_matched_count[5m])
```

Solutions :

Trafic Élevé :

- Évoluez horizontalement (ajoutez des nœuds)
- Évoluez verticalement (ajoutez du CPU)

Routage Inefficace :

- Réduisez le nombre de routes
- Optimisez les critères de route

Trop de Recherches ENUM :

- Vérifiez le taux de réussite du cache
- Envisagez de désactiver si non nécessaire

Utilisation Élevée de la Mémoire

Symptômes :

- Utilisation de la mémoire > 90%
- Application se bloque
- Erreurs de mémoire insuffisante

Étapes de Diagnostic :

1. **Vérifiez la Mémoire** :

```
free -h

ps aux | grep beam.smp
```

2. **Vérifiez les Tailles du Cache** :

```
sms_c_enum_cache_size_size
```

Solutions :

Cache ENUM Trop Grand :

- Effacez le cache
- Réduisez le TTL
- Désactivez ENUM si non nécessaire

File d'Attente de Batch en Croissance :

```
# Vérifiez les statistiques des travailleurs (IEx)
SmsC.Messaging.BatchInsertWorker.stats()
```

Si la file d'attente est grande, videz-la manuellement ou redémarrez.

Ajouter de la Mémoire :

- Évoluez verticalement
- Ajoutez du swap (temporaire)

Fuite de Mémoire :

- Redémarrez l'application
- Signalez le problème pour enquête

Pour une assistance supplémentaire, consultez :

- Guide des Opérations - Procédures quotidiennes
- Guide de Configuration - Options de configuration
- Guide des Métriques - Configuration de la surveillance
- Journaux d'application - `/var/log/sms_c/application.log`