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.

Fonctionnalités :

- 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
- Génère 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 de Message (`message_api_bench.exs`)

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

Fonctionnalités :

- 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

Documentation des opérations SMS-C

← Retour au README principal

Bienvenue dans la documentation des opérations SMS-C. Ce guide complet couvre tous les aspects de la configuration, de l'exploitation, de la surveillance et du dépannage du système SMS-C.

Vue d'ensemble de la documentation

Prise en main

- **Référence de configuration** - Options de configuration complètes et exemples

Opérations quotidiennes

- **Guide des opérations** - Tâches quotidiennes, surveillance et maintenance
- **Guide de routage SMS** - Gestion et configuration des routes
- **Recherche d'abonnés HSS** - Interface Diameter Sh pour la détection des abonnés sur le réseau
- **Fédération géographique** - Distribution multi-sites basée sur HTTP avec découverte DNS SRV
- **Référence API** - Documentation API complète avec exemples

Performance et surveillance

- **Optimisation des performances** - Optimisation pour différentes charges de travail
- **Guide des métriques** - Métriques Prometheus et surveillance

Dépannage

- **Guide de dépannage** - Problèmes courants et solutions

Conformité et réglementation

- **Conformité à l'interception ANSSI R226** - Spécifications techniques françaises pour l'interception légale
  - Intégration frontend multi-protocoles (IMS/SIP, SMPP, SS7/MAP)
  - Interfaces d'interception légale ETSI X1/X2/X3
  - Architecture de stockage à deux niveaux Mnesia + SQL
  - Schéma CDR pour les requêtes d'interception légale
  - Capacités de cryptage et de cryptanalyse

Liens rapides

Tâches courantes

- Soumettre un message
- Créer une route
- Vérifier l'état du message
- Surveiller la santé du système
- Gérer les échecs de livraison

Exemples de configuration

- Stockage et conservation des messages
- Configuration de l'export CDR
- Contrôles de confidentialité
- Configuration haute capacité
- Fédération géographique
- Routage géographique
- Équilibrage de charge
- Configuration Diameter Sh / HSS

- Configuration ENUM/NAPTR
- Chargement OCS
- Traduction de numéro

Surveillance et alertes

- Métriques clés
- Alertes recommandées
- Modèles de tableau de bord

Vue d'ensemble de l'architecture du système

Le SMS-C est une plateforme de routage de messages distribuée et haute performance avec les composants clés suivants :

Composants principaux

- **Stockage de messages** - Stockage rapide basé sur Mnesia avec conservation configurable et export CDR
- **Moteur de routage** - Règles de routage basées sur Mnesia avec correspondance de préfixe et équilibrage de charge
- **Traduction de numéro** - Normalisation de numéro basée sur Regex avec ordre de priorité
- **Intégration de facturation** - Facturation en ligne OCS avec politiques basées sur les routes
- **Recherche ENUM** - Routage de numéro basé sur DNS avec mise en cache
- **Journalisation des événements** - Suivi du cycle de vie des messages
- **Export CDR** - Exportation automatique vers une base de données SQL pour la facturation/analyse à long terme

Interfaces externes

- **API REST** - Soumission et gestion des messages (HTTPS)

- **Interface Web** - Gestion des routes, navigateur de messages, surveillance
- **Prometheus** - Exposition des métriques pour la surveillance
- **OCS** - Intégration de facturation
- **DNS** - Recherches ENUM/NAPTR pour le routage

Distribution et haute disponibilité

- **Fédération géographique** - Distribution multi-contrôleurs basée sur HTTP avec découverte DNS SRV
- **File d'attente de transfert** - Mise en file d'attente automatique des messages et réessai lorsque les contrôleurs distants sont inaccessibles
- **Équilibrage de charge** - Distribution de route pondérée

Documentation connexe

- **Références de performances** - Tests de performance et résultats
- **Référence du schéma CDR** - Schéma complet de la base de données CDR avec exemples SQL

Exigences du système

Exigences minimales

- **CPU** : 2 cœurs
- **RAM** : 4 Go
- **Disque** : 50 Go (augmente avec la conservation des messages)
- **OS** : Linux (recommandé), macOS (développement)
- **Erlang/OTP** : 26.x ou version ultérieure
- **Elixir** : 1.15.x ou version ultérieure
- **Base de données SQL** : MySQL 8.0+, MariaDB 10.5+, ou PostgreSQL 13+ (pour le stockage CDR)

Production recommandée

- **CPU** : 8+ cœurs
- **RAM** : 16+ Go
- **Disque** : 500+ Go SSD
- **Réseau** : 1 Gbps+
- **Base de données SQL** : Serveur dédié avec réplication (pour le stockage CDR)

Ports réseau

- **80/443** - Interface Web (HTTP/HTTPS)
- **8443** - API (HTTPS) et communication entre pairs de fédération
- **9568** - Métriques Prometheus

Support et ressources

Journaux

- **Journaux d'application** : `/var/log/sms_c/` (production) ou console (développement)
- **Journaux de l'interface Web** : Visionneuse de journaux en temps réel à `/logs`
- **Journaux d'événements** : Suivi des événements par message via l'API

Diagnostics

- **Vérification de l'état** : `GET /api/status`
- **Métriques** : `GET http://localhost:9568/metrics` (format Prometheus)
- **État du frontend** : Interface Web à `/frontend_status`
- **File d'attente de messages** : Interface Web à `/message_queue`

Obtenir de l'aide

1. Consultez le Guide de dépannage
2. Passez en revue les journaux d'application
3. Vérifiez les métriques Prometheus pour des anomalies
4. Utilisez le simulateur de routage pour tester la logique de routage
5. Examinez les journaux d'événements par message

Informations sur la version

Cette documentation est à jour au :

- **Dernière mise à jour** : 2025-10-30
- **Version SMS-C** : Dernière version de développement
- **Elixir pris en charge** : 1.15.x - 1.17.x
- **Erlang/OTP pris en charge** : 26.x - 27.x

Conventions de documentation

Tout au long de cette documentation :

- **Les exemples de configuration** montrent des valeurs typiques ; ajustez pour votre environnement
- **Les exemples d'API** utilisent le format de ligne de commande `curl`
- **Les adresses IP et les domaines** sont des exemples uniquement ; remplacez par vos valeurs réelles
- **Les noms de métriques** suivent les conventions de nommage Prometheus
- **Tous les horodatages** sont en UTC sauf indication contraire

Démarrage rapide

1. **Configuration** : Configurez via `config/runtime.exs` - voir Référence de configuration

2. **Routes initiales** : Créez des règles de routage via l'interface Web ou le fichier de configuration - voir Guide de routage SMS
3. **Soumettre un message de test** : Utilisez l'API ou l'interface Web - voir Référence API
4. **Surveiller** : Configurez le scraping Prometheus - voir Guide des métriques

Retour d'information sur la documentation

Cette documentation est maintenue parallèlement au code source de SMS-C. Pour des corrections ou des améliorations, veuillez mettre à jour les fichiers markdown dans le répertoire `docs/`.

Documentation de conformité à l'interception ANSSI R226

Objet 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 des préfixes
- Logique de réessai avec retour exponentiel
- Gestion de l'expiration des messages et de la file des lettres mortes
- Génération et archivage des enregistrements de détails d'appel (CDR)
- Performance : ~1 750 messages/seconde, 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 de messages)
  - Les CDR sont écrits lorsque les messages sont livrés, expirés, échoués ou rejetés
  - La base de données SQL est utilisée UNIQUEMENT pour l'exportation/archivage des CDR, PAS pour les opérations de message actives
  - Aucun impact sur les performances du routage des messages (écriture asynchrone)
- **Avantages de l'architecture à deux niveaux :**

    - File active : Extrêmement 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 en temps réel
- Correspondance basée sur des préfixes (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
- Contr��le de facturation par route (intégration CGRates)

 Architecture complète et fonctionnalités documentées dans README.md

1.2 Capacités d'interception

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 de mobile)
  - MSISDN de destination (numéro de mobile)
  - IMSI source (Identité d'abonné mobile international)
  - IMSI de destination
  - Corps du message (contenu texte)
  - Données PDU (Unité de données de protocole) brutes
  - Informations TP-DCS (Schéma de codage de données)
  - Codage 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 n'est pas livrable)
  - Statut (livré, expiré, échoué, rejeté)
  - Nombre de 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 complet des CDR 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 en RAM pour une vitesse maximale
  - Synchronisation automatique sur disque pour la récupération après un crash
  - Les messages persistent lors des redémarrages du système
- **Performance :** Opérations de lecture/écriture inférieures à la milliseconde
- **Conservation :** 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/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 (dépassé la période de validité)
  - Message échoué de manière permanente
  - Message rejeté par les règles de routage
- **Mode d'écriture :** Écriture par lots asynchrone (aucun impact sur les performances du routage des messages)
- **Conservation :** Long terme (mois à années), configurable selon les exigences réglementaires
- **Opérations :** Requêtes historiques, reporting, conformité, interception légale
- **Accès :** Requêtes SQL, REST API (à l'avenir), 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 lors d'un 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 vie privée
- 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'activité
- Suivi de l'état de santé (actif/expiré)
- Historique de disponibilité
- Suivi des adresses IP et des noms d'hôte
- Journalisation de la configuration spécifique au frontal

1.2.3 Capacités d'analyse

Surveillance en temps réel :

- Tableau de bord de l'interface 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 complet de surveillance dans OPERATIONS_GUIDE.md
📖 Documentation des métriques dans METRICS.md

Analyse historique :

- Base de données CDR interrogeable par :
    - Plage horaire
    - Numéro de la partie appelante/appelée
    - 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 sur les 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 des 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 Web
- Mécanismes d'authentification API
- Contrôles d'accès à la base de données
- Authentification d'enregistrement des frontaux

Journalisation d'audit :

- 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)
- Le champ du corps du message CDR peut être défini sur NULL pour des raisons de confidentialité

Sécurité de la base de données :

- Support de chiffrement des tables MySQL (ENCRYPTION='Y')
- Support de chiffrement des 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

Vue d'ensemble

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

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

Qu'est-ce que Mnesia ?

- Base de données distribuée intégrée dans le runtime Erlang/OTP
- Stockage hybride : Principalement en mémoire avec sauvegarde automatique sur disque
- Transactions conformes à ACID
- Réplication de cluster entre plusieurs nœuds

Mode de stockage : disc_copies

- **En mémoire principale :** Tous les messages actifs stockés en RAM
  - Opérations de lecture/écriture ultra-rapides (inférieures à la milliseconde)
  - Pas d'E/S disque lors des opérations de routage de messages normales
  - Permet un débit de plus de 1 750 messages/seconde
- **Sauvegarde sur disque (automatique) :** Mnesia synchronise la RAM sur 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 un 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. Passerelle met à jour le statut de livraison → Mise à jour Mnesia (mémoire + disque)
5. Après livraison/expiration → Message marqué pour nettoyage
6. Travail de nettoyage (24h par défaut) → Message supprimé de Mnesia

Caractéristique de performance critique :

- **AUCUNE requête de base de données SQL** pendant le routage/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 qu'un CDR (Call Detail Record) ?

- 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 (dépassé la période de validité 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 lots asynchrone :** Les CDR sont écrits dans le processus de travail en arrière-plan
- **Pas de blocage :** Le routage des messages n'attend jamais l'écriture SQL

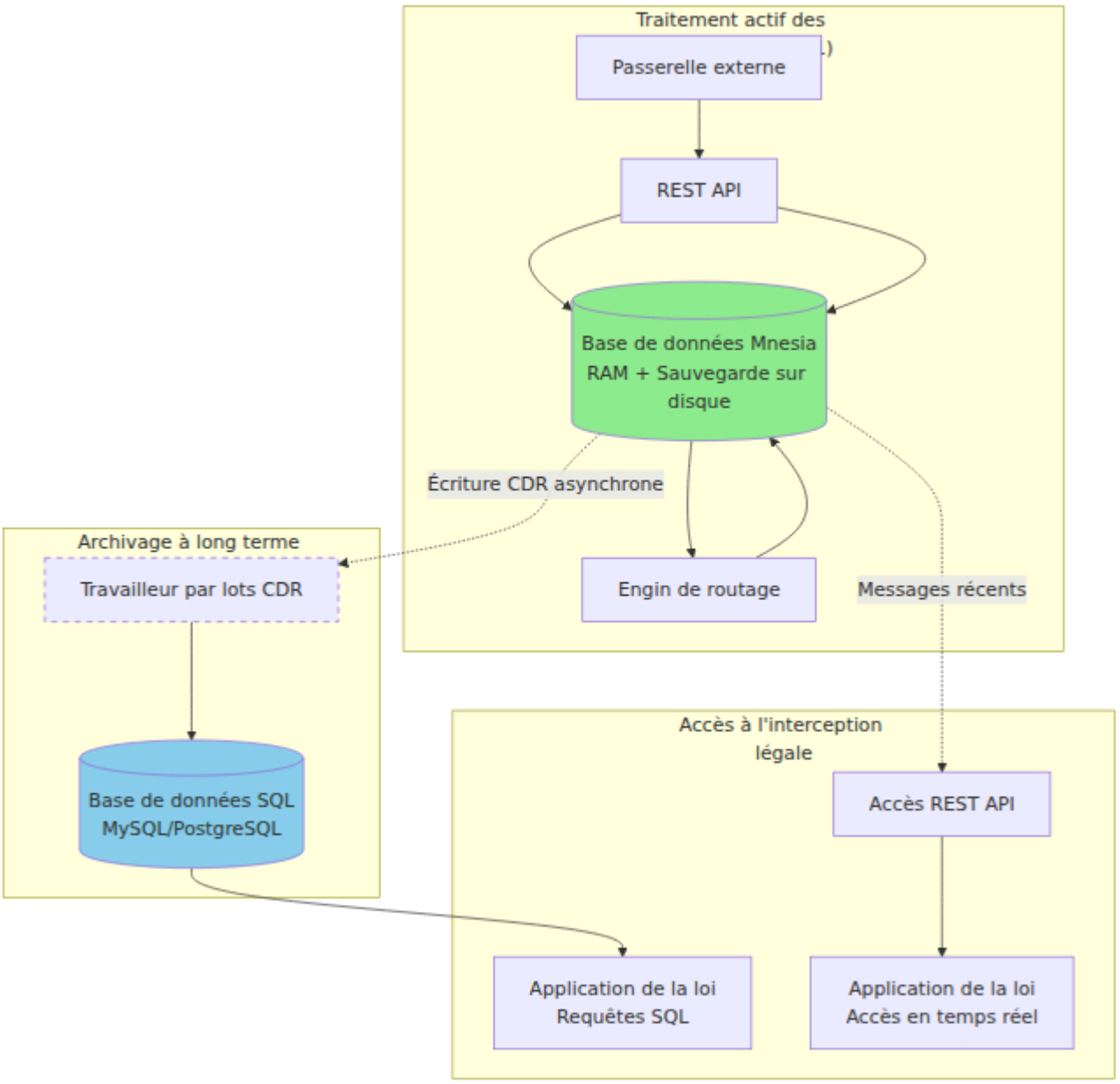
- **Inserts groupés :** Plusieurs CDR regroupés (par défaut 100) et écrits ensemble
- **Intervalle de vidage :** 100 ms par défaut (configurable)
- **Gestion des erreurs :** Les échecs d'écriture CDR sont enregistrés, le traitement des messages continue

```
# 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 (en 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 de requête SQL standard
- Métadonnées complètes des messages toujours disponibles
- Corps du message disponible (sauf si le mode de confidentialité est activé)

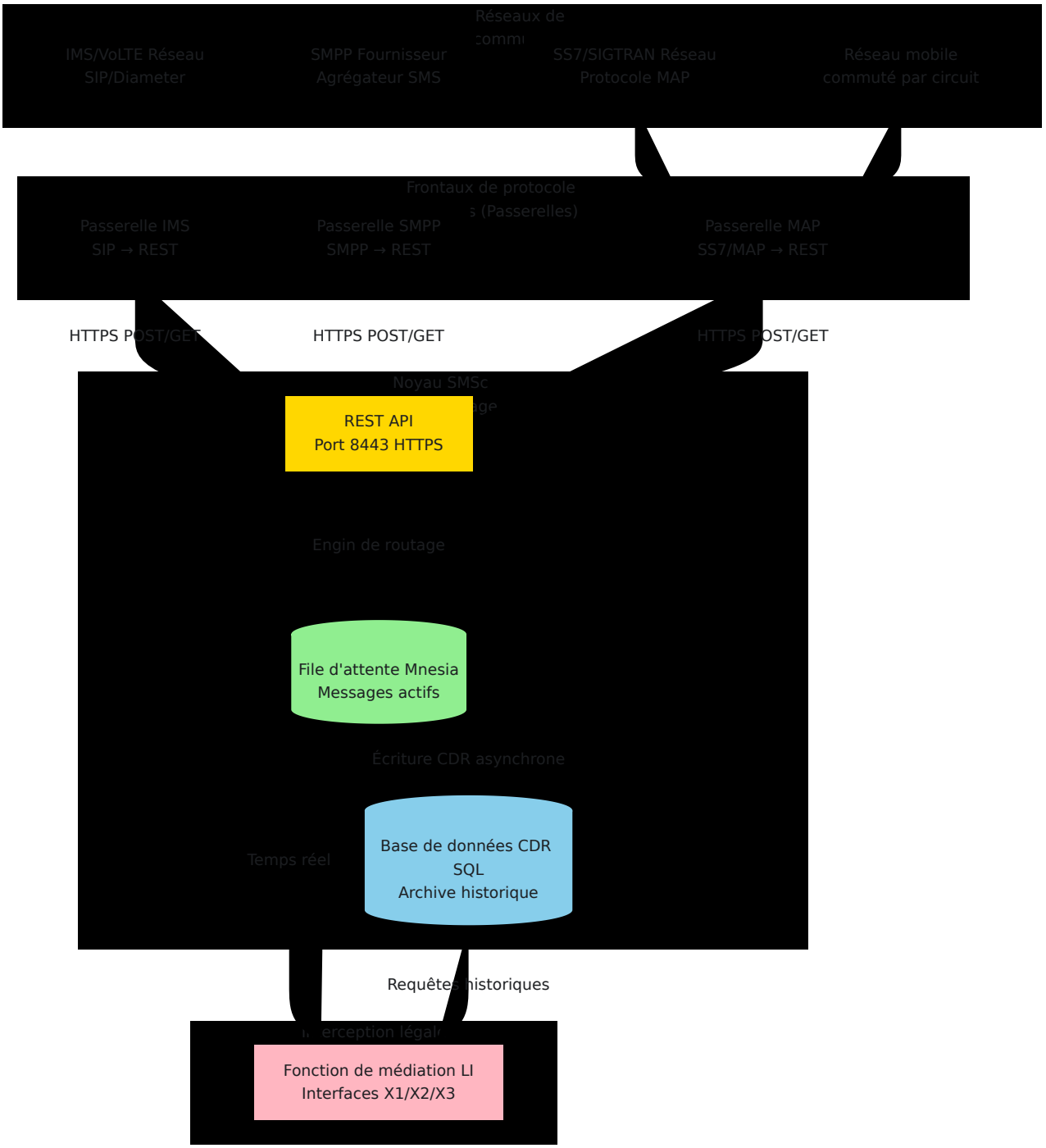
Avantages de conformité :

1. **Pas de perte de données :** Le mode `disc_copies` garantit que les messages survivent aux pannes
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 accessibles

1.5 Architecture d'intégration multi-protocoles des frontaux

Le SMSc OmniMessage utilise un design central indépendant 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.

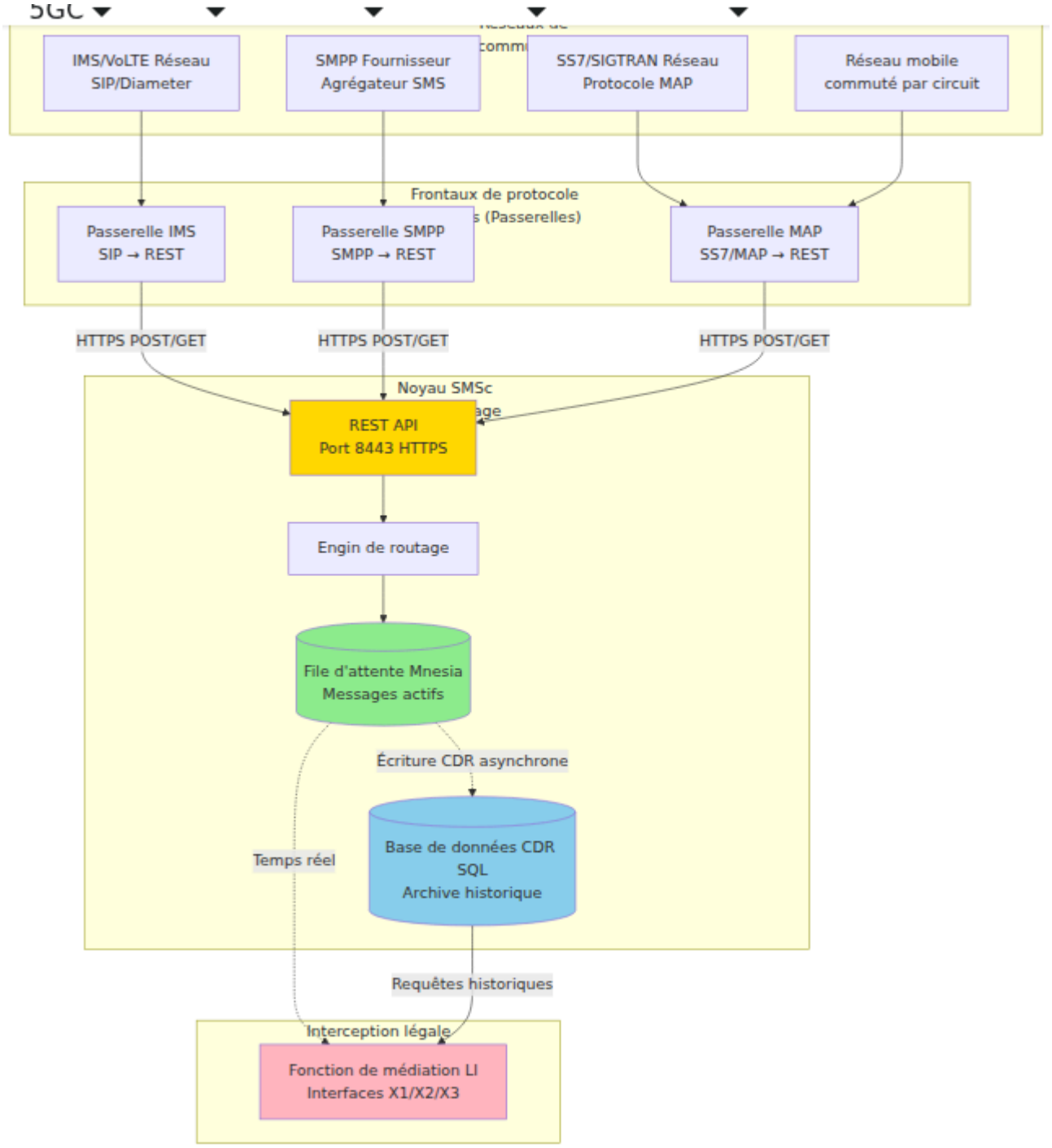
Vue d'ensemble de l'architecture



Détails d'intégration des frontaux de protocole

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.

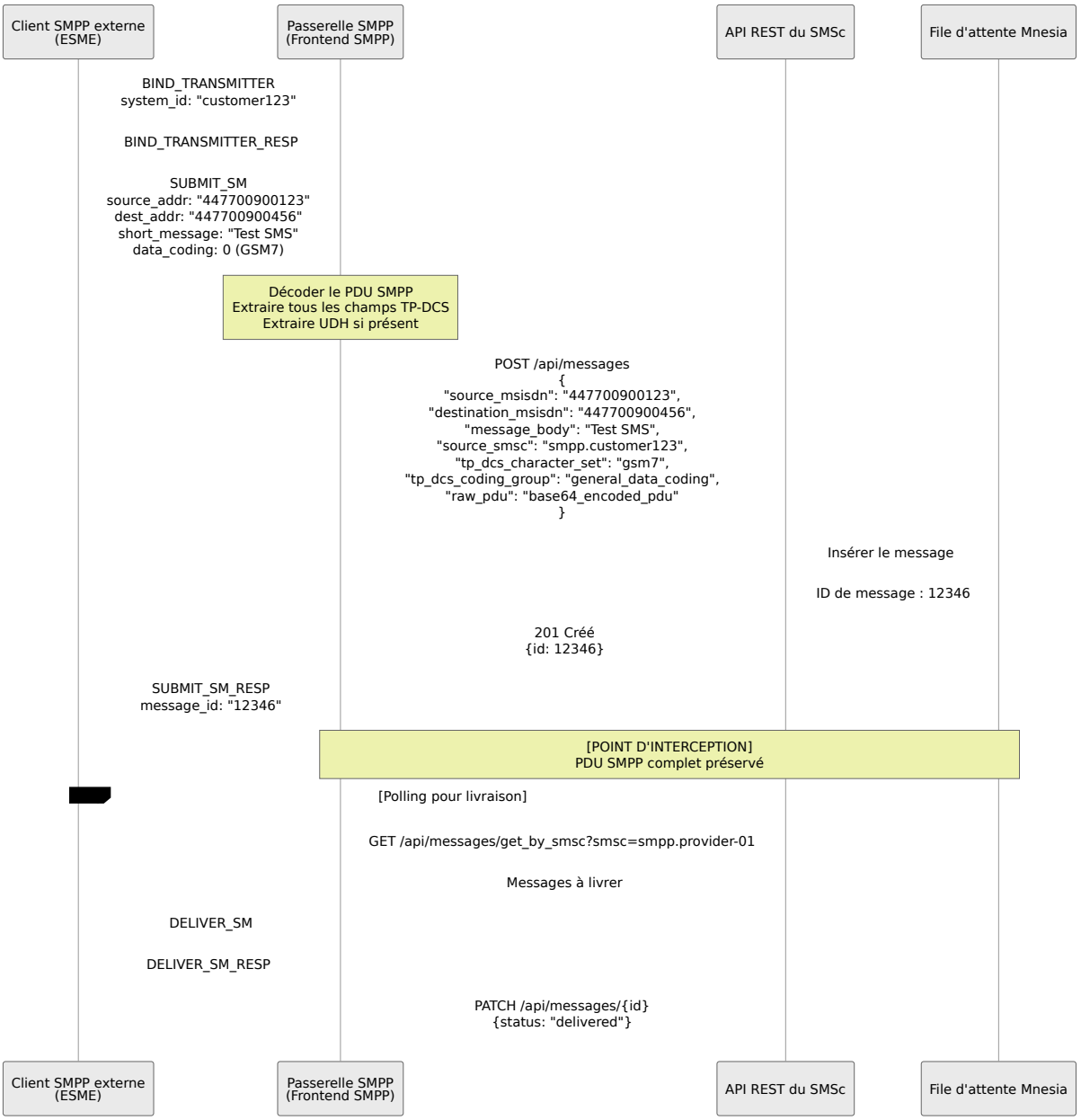


Données spécifiques à l'interception IMS :

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

2. Intégration du frontal SMPP

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

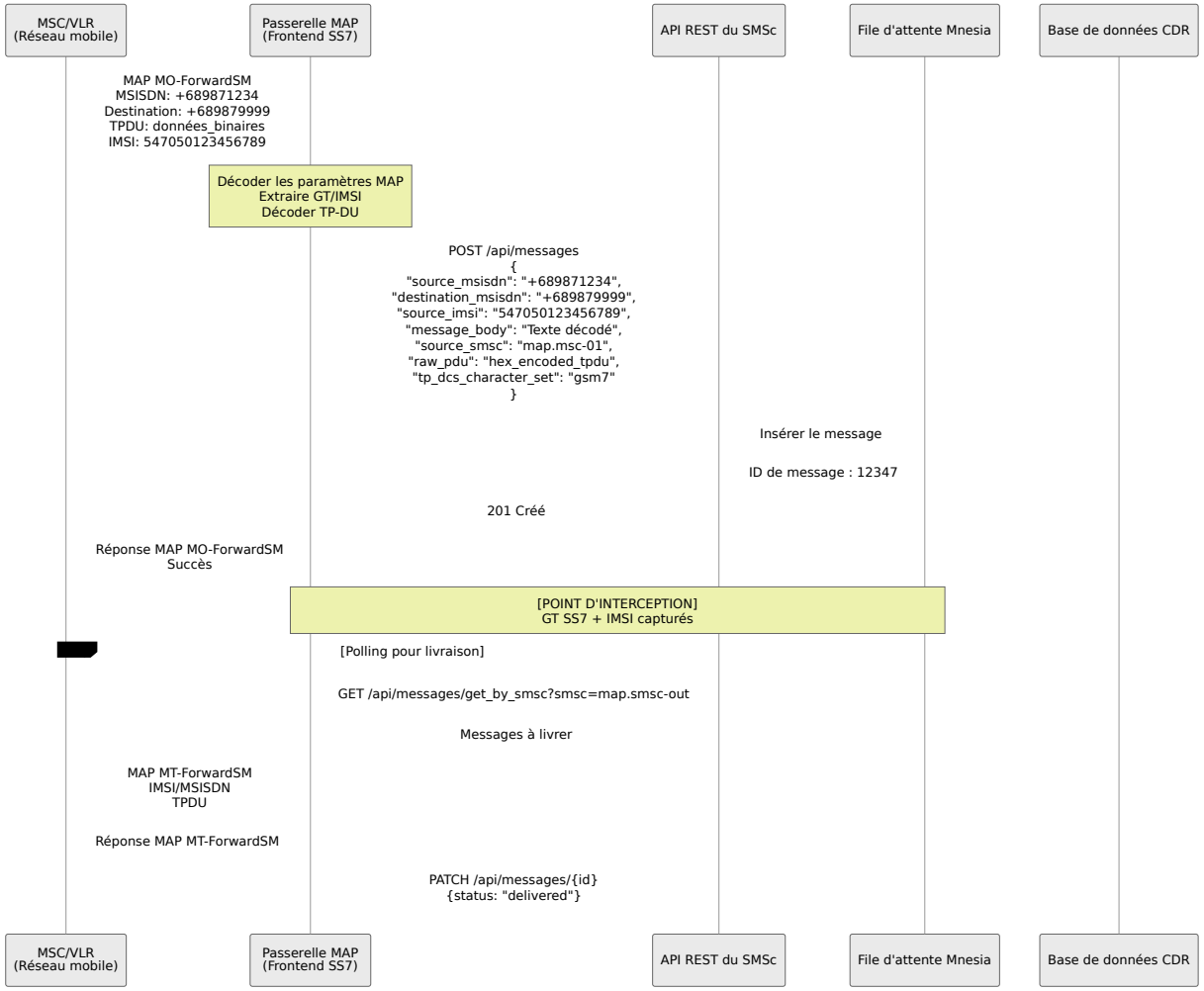


Données spécifiques à l'interception SMPP :

- PDU SMPP complet (format binaire préservé)
- Détails du schéma de codage de données (DCS)
- En-tête de données utilisateur (UDH) pour les messages concaténés
- ID système ESME (identification du client)
- Informations sur le plan de numérotation TON/NPI
- Indicateurs de livraison enregistrés

3. Intégration du frontal SS7/MAP

Les réseaux commutés par circuit hérités utilisent le protocole SS7 MAP pour les SMS. La passerelle MAP traduit entre le signalement SS7 et l'API REST.



Données spécifiques à l'interception SS7/MAP :

- IMSI des messages MAP
- Adresses Global Title (GT)
- Adresse MSC/VLR (identification de l'élément réseau)
- Adresses de la partie appelante/appelée SCCP
- Codes d'opération MAP
- Format binaire TP-User-Data

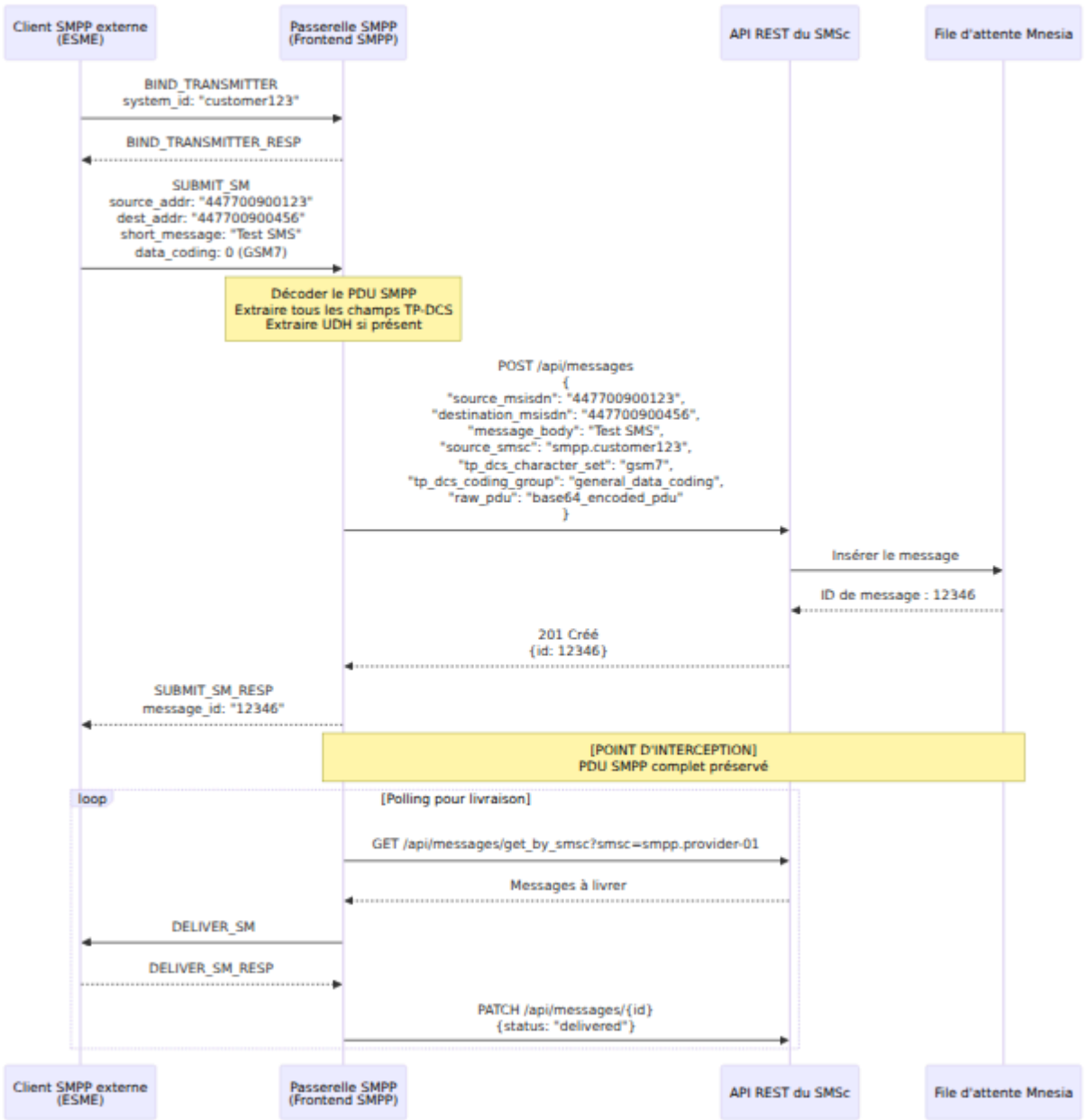
Interception unifiée à travers tous les protocoles

Avantage clé pour l'interception légale : Quel que soit le protocole utilisé (IMS/SIP, SMPP ou SS7/MAP), tous les messages convergent dans le noyau du

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-protocoles :** Suivre les messages à travers les frontières des protocoles
4. **Préservation complète des métadonnées :** Champs spécifiques au protocole préservés dans le CDR

Résumé du flux de données :



Identification de protocole dans le CDR :

- Le champ `source_smsc` indique le protocole frontal (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

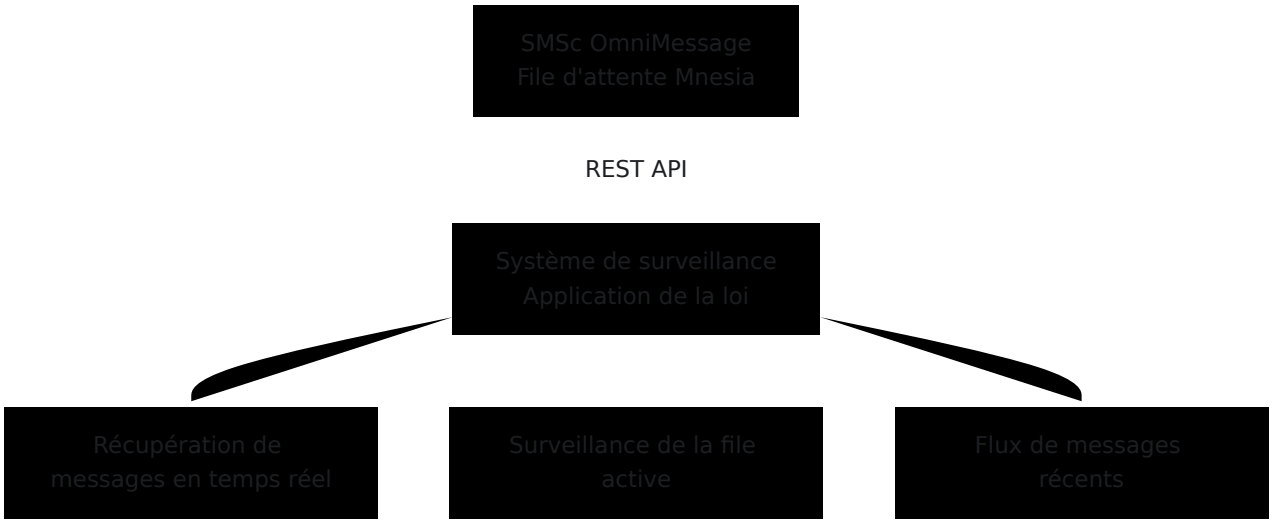
1.6 Architecture technique pour l'interception légale

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 (typiquement 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

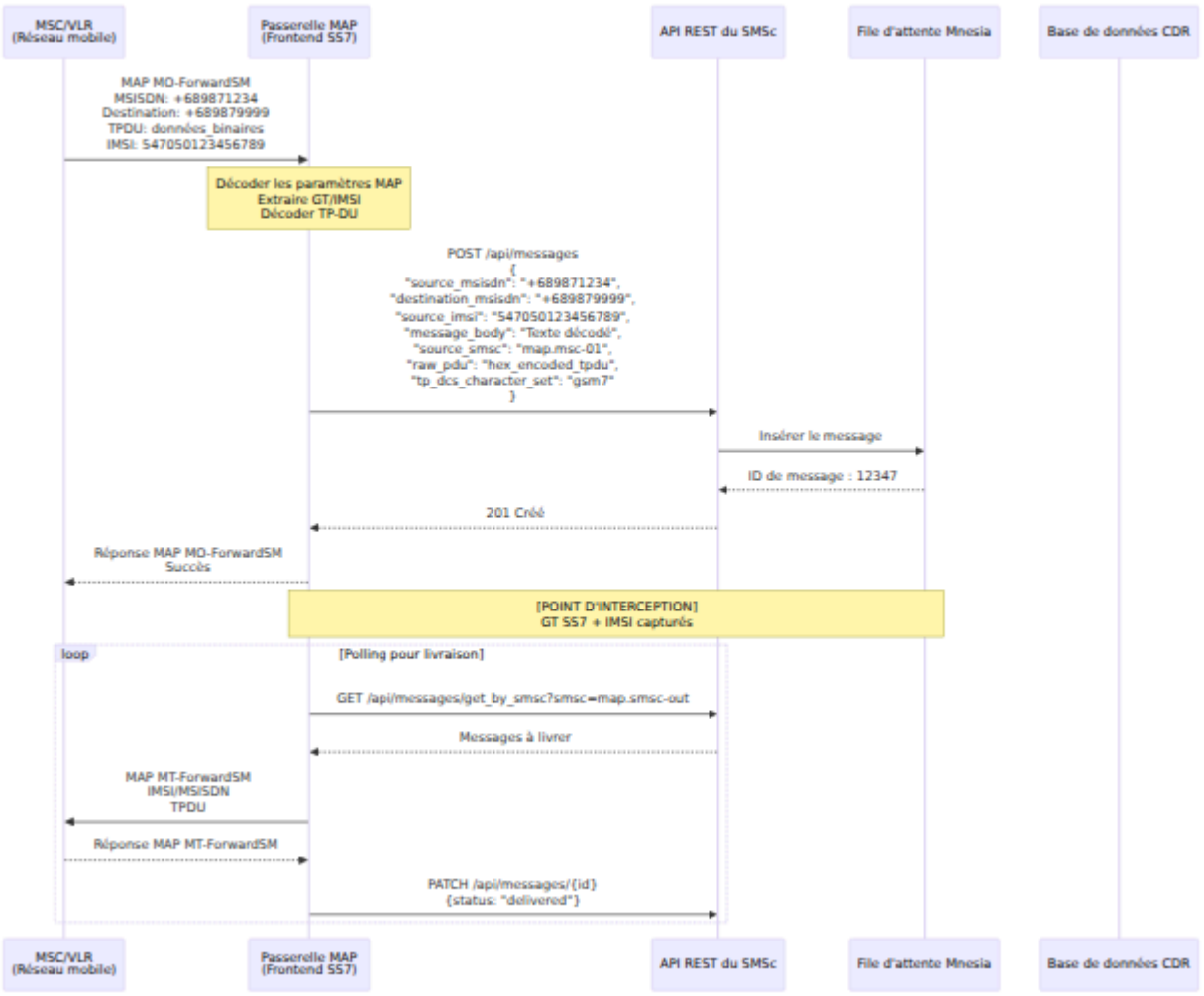
Remarque : Ces points de terminaison interrogent la file d'attente de messages active de Mnesia, fournissant un accès aux messages actuellement traités ou récemment livrés (dans la période de conservation).

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 aux requêtes SQL de la table `cdrs` (piste d'audit permanente)
- **Méthode d'accès :** Client SQL standard (mysql, psql, DBeaver, etc.)
- **Source de données :** Uniquement les messages archivés (pas de file 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) - Lorsque 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

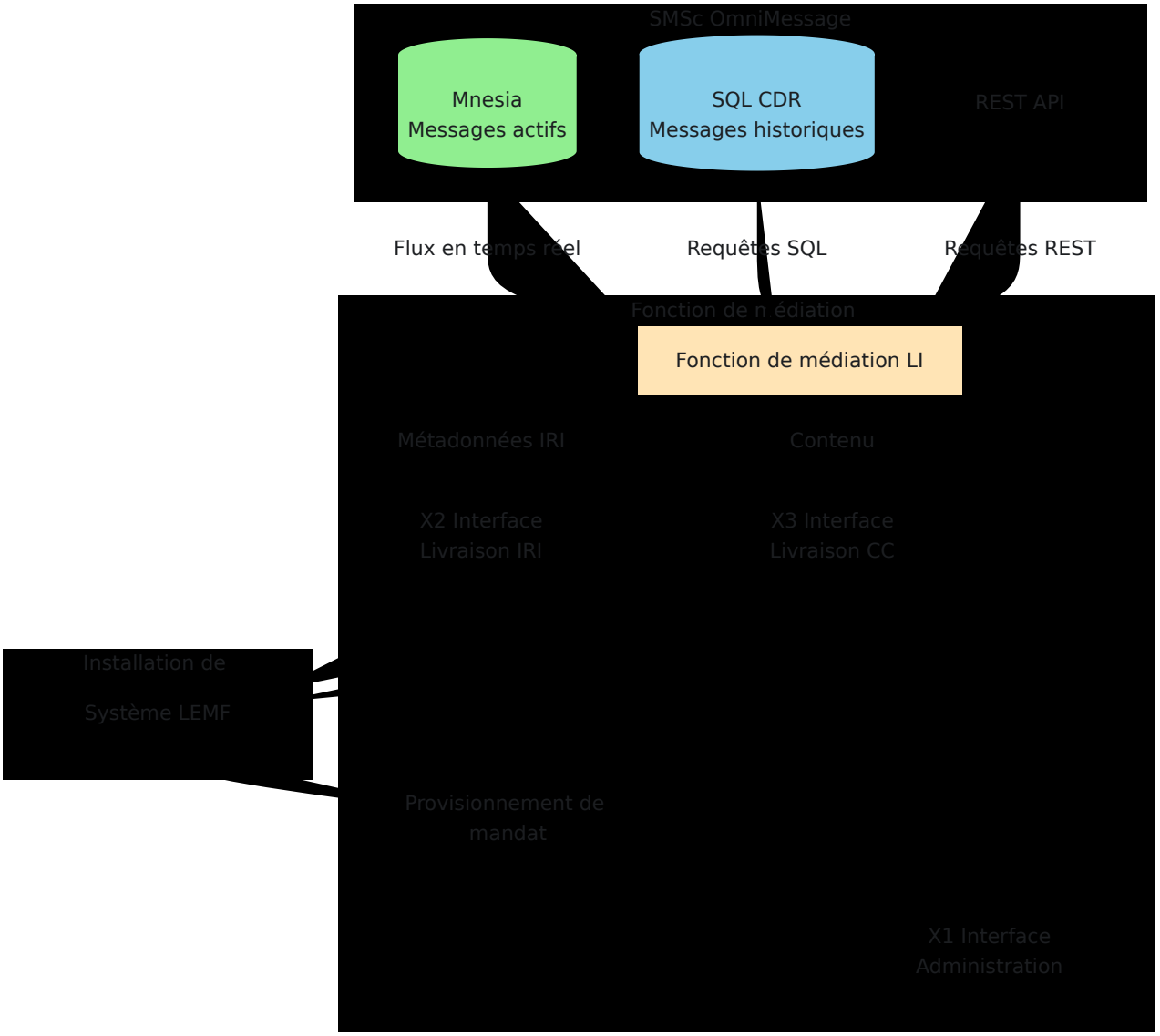
4. Interface d'exportation par lots :

- Export CSV des enregistrements CDR
- Export JSON pour un accès programmatique
- Champs d'exportation configurables
- Exportations basées sur la plage horaire
- Exportations conscientes de la vie privée (exclusion optionnelle du corps du message)

Interfaces standard d'interception légale ETSI

Le SMSc OmniMessage fournit la base pour la mise en œuvre des interfaces d'interception légale conformes à l'ETSI. Bien que le noyau du SMSc ne mette pas en œuvre nativement les interfaces X1/X2/X3, il fournit tous les points d'accès de données nécessaires qui peuvent être intégrés avec des systèmes externes de Fonction de médiation d'interception légale (LIMF).

Interfaces LI standard ETSI :



Descriptions des interfaces :

Interface X1 - Fonction d'administration :

- **Objet :** Provisionnement de mandat et de cible de l'application de la loi 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 des critères de filtrage (numéros de téléphone, fenêtres temporelles)
  - Récupérer l'état de l'interception

- **Intégration avec le SMSc :**
  - LIMF maintient la liste des cibles (base de données de mandats)
  - LIMF interroge le SMSc CDR/API pour les messages correspondants
  - LIMF filtre en fonction des critères fournis par X1

Interface X2 - Livraison d'IRI (Informations liées à l'interception) :

- **Objet :** Livrer les métadonnées des messages à l'application de la loi
- **Direction :** LIMF → LEMF (unidirectionnel)
- **Format de données :** XML/ASN.1 conforme à ETSI TS 102 232-x
- **Contenu du SMSc CDR :**
  - ID de message
  - Numéro appelant (MSISDN source)
  - Numéro appelé (MSISDN de 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 le SMSc :**
  - LIMF interroge la base de données CDR pour les numéros de téléphone cibles
  - LIMF transforme les enregistrements CDR en format IRI conforme à l'ETSI
  - LIMF livre l'IRI à LEMF via X2

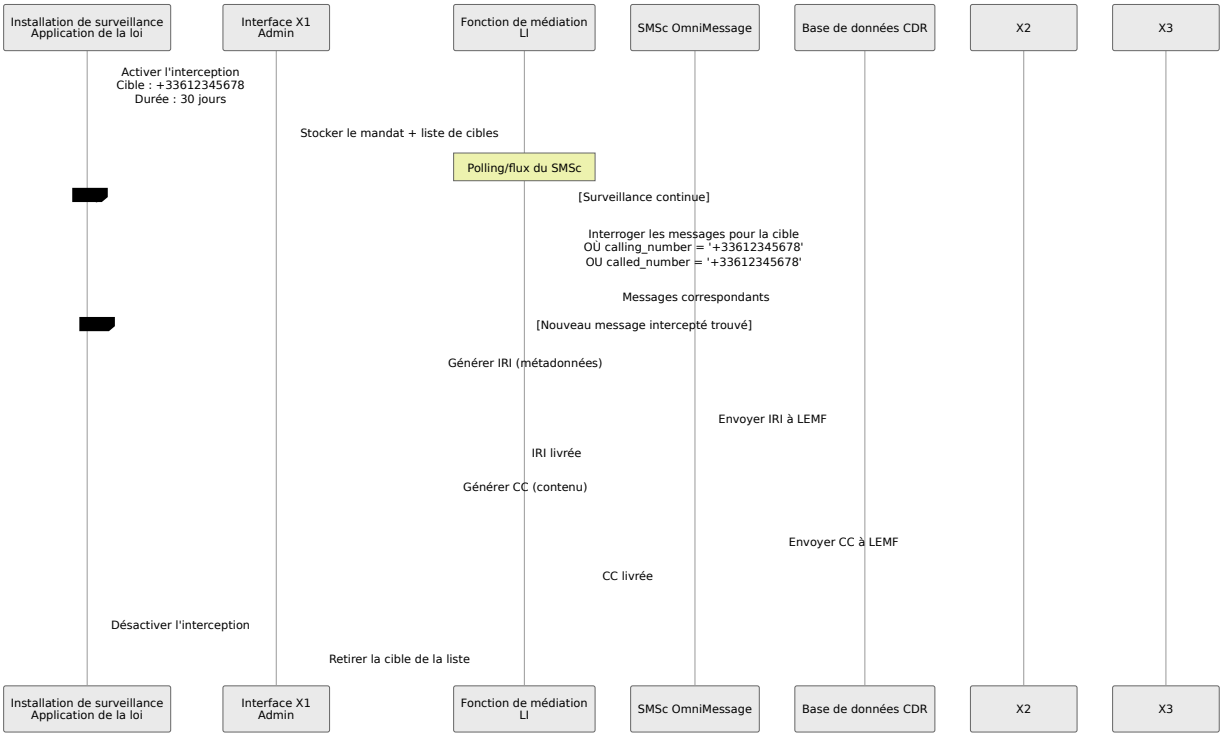
Interface X3 - Livraison CC (Contenu de la communication) :

- **Objet :** Livrer le contenu réel du message à l'application de la loi
- **Direction :** LIMF → LEMF (unidirectionnel)
- **Format de données :** Conforme à ETSI TS 102 232-x
- **Contenu du SMSc :**
  - Corps du message (contenu texte)

- PDU brut (données SMS binaires)
- Informations sur le codage de caractères
- Segments de message multipart
- Informations TP-DCS
- UDH (en-tête de données utilisateur)

- **Intégration avec le SMSc :**
  - LIMF récupère le contenu du message à partir du champ `message_body` CDR
  - LIMF récupère les données PDU brutes si disponibles
  - LIMF emballe le contenu au format CC conforme à l'ETSI
  - LIMF livre le CC à LEMF via X3

Architecture de mise en œuvre :



Mappage des données SMSc vers les interfaces LI :

Champ de données SMSc	X2 (IRI)	X3 (CC)	Colonne de la table CDR
ID de message	□ ID de corrélation	□ Référence	`message_id`
Numéro appelant	□ Partie A	-	`calling_number`
Numéro appelé	□ Partie B	-	`called_number`
Horodatage de soumission	□ Horodatage	-	`submission_time`
Horodatage 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 de destination	□ Élément réseau	-	`dest_smsc`
IMSI	□ ID d'abonné	-	(Via frontaux)

Options d'intégration LIMF :

Option 1 : Architecture de polling

- LIMF interroge périodiquement la base de données CDR (toutes les 1 à 60 secondes)

- Requête SQL filtrant par numéros de téléphone cibles de la liste de mandats X1
- 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

- Le SMSc PubSub 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 cible

Interception basée sur des événements :

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

Interception basée sur le temps :

- Filtrage par plage de dates/heures 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 pris en charge :

- TLS 1.2 (RFC 5246)
- TLS 1.3 (RFC 8446) - Recommandé
- SSL 2.0/3.0 : NON SOUTENU (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 Phoenix Framework

Suites de chiffrement :

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

Préféré - TLS 1.3 :

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256

- TLS_CHACHA20_POLY1305_SHA256

Pris en charge - 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 l'échange de clés ECDHE/DHE
- Groupes Diffie-Hellman forts (2048 bits minimum)
- Support de la cryptographie à courbe elliptique
- Support de l'indication de nom de serveur (SNI)

Gestion des certificats :

- Support de certificats X.509
- Tailles de clé RSA : 2048 bits minimum, 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: %{
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  }
```

Référence de configuration complète 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 de 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 de 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 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

Gestion des clés :

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

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 des en-têtes (en-tête SMSc pour identification des frontaux)
- Contrôle d'accès basé sur l'IP (niveau pare-feu)
- Authentification client basée sur des certificats (optionnelle)

Enregistrement des frontaux :

- Identification unique des frontaux (nom, type, IP, nom d'hôte)
- Authentification basée sur des battements de cœur
- Gestion de session basée sur l'expiration (timeout de 90 secondes)
- Suivi et surveillance des frontaux

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",  # 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 de
l'application de la loi
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 D��tails 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 :

- Empreintes de message (détection de 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 clés :

- 128 bits (minimum)
- 256 bits (recommandé)

Utilisation :

- Chiffrement de session TLS
- Chiffrement des bases de données au repos
- Chiffrement optionnel du corps des messages

2.5.3 Chiffrement asymétrique

Algorithmes pris en charge :

- RSA (2048 bits minimum, 4096 bits recommandé)
- ECDSA (Algorithme de signature numérique à courbe elliptique)
  - Courbes P-256, P-384, P-521
- Ed25519 (EdDSA)

Utilisation :

- Authentification des certificats 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 (codage SMS standard)
- UCS-2 (Unicode, 16 bits)
- Données binaires 8 bits
- Latin-1

TP-DCS (Schéma de codage de données) :

- Indication de classe de message
- Indicateurs de compression
- Spécification de groupe de codage
- Identification de l'ensemble de caractères

Aucun chiffrement SMS natif :

- Le protocole SMS ne fournit pas de chiffrement de bout en bout
- Le contenu des messages est accessible au niveau du SMSc
- Permet l'interception légale comme requis

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 de cœur IMS

Protocole SS7/MAP (Frontend externe) :

- Sécurité du réseau SS7
- Authentification du protocole MAP
- Sécurité des couches SCCP/TCAP

Remarque : La sécurité spécifique au protocole est mise en œuvre dans les passerelles frontales externes, pas dans le noyau du SMSc.

2.7 Capacités de cryptanalyse et d'évaluation de la sécurité

2.7.1 Outils d'analyse de protocole

Capacités 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 analyses judiciaires
- Exportation des 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)
- Identifiants de base de données dans des fichiers de configuration (devrait utiliser la gestion des secrets)
- Support de certificats auto-signés (développement/test uniquement)

Recommandations de renforcement de la sécurité :

- Utiliser des suites de chiffrement TLS solides
- 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 régulières de sécurité pour Erlang/OTP et ses dépendances
- Restrictions de pare-feu sur les ports API
- Liste blanche des IP pour l'accès frontal

Tests de sécurité :

- Analyse régulière des vulnérabilités des dépendances
- Support de tests d'intrusion
- 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 clés

Génération de certificats TLS :

```
# 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 :

- CSPRNG Erlang/OTP (Générateur de nombres pseudo-aléatoires cryptographiquement sécurisé)
- Pool d'entropie système (/dev/urandom)
- Aléatoire fort pour les clés de session, IDs, tokens

2.8.2 Stockage et protection des clés

Stockage des clés privées :

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

Rotation des clés :

- Procédures de renouvellement de certificats TLS (annuellement recommandé)
- Rotation des identifiants de base de données
- Rotation des tokens API (si mis en œuvre)

2.8.3 Distribution des clés

Distribution des certificats :

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

Distribution des clés symétriques :

- Échange de clés hors bande pour les identifiants de base de données
- Accord de clés Diffie-Hellman dans TLS
- Pas de transmission de clés en clair

2.9 Conformité et normes

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

2.9.1 Conformité SMS sur protocole SS7/MAP

Normes 3GPP et ETSI :

- **3GPP TS 23.040 :** Réalisation technique du service de messages courts (SMS) - Spécification de protocole SMS de base
- **3GPP TS 23.038 :** Alphabets et informations spécifiques à la langue - Codage de 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 messages courts point à point
- **ETSI TS 100 902 :** Spécification du service de messages courts de diffusion cellulaire

Normes de signalisation SS7 :

- **ITU-T Q.711-Q.716 :** Partie de contrôle de connexion 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 messages (MTP) Niveaux 1-3
- **ETSI EN 300 356 :** Système de signalisation n° 7 - Partie utilisateur ISDN (ISUP)

Normes de sécurité pour SS7/MAP :

- **GSMA FS.07 :** Sécurité de la 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 transfert pour l'interception légale du trafic de télécommunications
- **ETSI TS 102 232-1 :** Interception légale (LI) ; Spécification de transfert - Partie 1 : Interface de transfert pour la gestion LI
- **3GPP TS 33.107 :** Architecture et fonctions d'interception légale pour les réseaux 3G

2.9.2 Conformité SMS sur protocole 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 de SMS sur réseaux IP - Spécification de niveau 3
- **3GPP TS 23.204 :** Support du service de messages courts (SMS) sur accès IP générique 3GPP - Niveau 2
- **3GPP TS 29.228 :** Interfaces Cx et Dx du sous-système multimédia (IM)

Normes de sécurité IMS :

- **3GPP TS 33.203 :** Sécurité 3G ; Sécurité d'accès pour les services basés sur 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 basés sur IP

Normes de 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 transfert - Partie 5 : LI indépendante du service pour les services du sous-système multimédia IP
- **3GPP TS 33.107 :** Exigences et architecture d'interception légale
- **3GPP TS 33.108 :** Interface de transfert pour l'interception légale (LI)

2.9.3 Conformité au protocole SMPP

Spécification SMPP :

- **SMPP v3.4 :** Spécification du protocole de messagerie courte 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 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 n° 5 (IA5)
- **ISO/IEC 8859-1 :** Codage de caractères Latin-1
- **ISO/IEC 10646 :** Ensemble de caractères universel (UCS-2/UTF-16)

2.9.4 Conformité aux normes cryptographiques

Sécurité TLS et réseau :

- **NIST SP 800-52 :** Directives pour la sélection, la configuration et l'utilisation des mises en œuvre 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 :** Le protocole de sécurité de transport (TLS) version 1.3

Normes des algorithmes cryptographiques :

- **FIPS 197 :** Norme de chiffrement avancé (AES)
- **FIPS 180-4 :** Norme de hachage sécurisé (famille SHA-2)

- **NIST SP 800-38D :** Recommandation pour les modes de fonctionnement des algorithmes de chiffrement par 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 clés publiques Internet X.509

2.10 Résistance à la cryptanalyse

2.10.1 Principes de conception

Défense contre la cryptanalyse :

- Aucun algorithme cryptographique personnalisé/propriétaire
- Algorithmes uniquement standards de l'industrie, examinés par des pairs
- Mises à jour régulières de sécurité 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 certificats TLS
- Rotation des clés de session (par session pour TLS)
- Politiques de rotation des identifiants de base de données

Surveillance et détection :

- Journalisation des échecs d'authentification
- Surveillance de l'expiration des certificats
- Journalisation des échecs de handshake TLS
- Détection d'anomalies pour les échecs de chiffrement
- Alerte sur les 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 d'audit de tous les accès

Intégration du cadre légal :

- Suivi des mandats d'interception (intégration de système externe)
- Listes d'autorisation d'identificateurs de cible (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 vie privée

Politiques de conservation :

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

Protections de la vie privée :

- Option de suppression du corps du message après livraison
- Exclusion du corps du message de l'interface utilisateur/exportations
- Chiffrement de la base de données au repos
- Journalisation et surveillance des accès

- Principe de collecte minimale des 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 vie
privée
  delete_message_body_after_delivery: false,  # Définir sur true
pour le mode de confidentialité

  # Contrôle d'écriture CDR
  cdr
```

Référence API SMS-C

← Retour à l'Index de Documentation | **README Principal**

Référence complète pour tous les points de terminaison de l'API REST SMS-C avec des exemples de requêtes/réponses.

Table des Matières

Aperçu de l'API

L'API REST SMS-C fournit un accès programmatique aux fonctions de soumission, de routage et de gestion des messages.

URL de Base

```
https://api.example.com:8443/api
```

Port par Défaut : 8443 (configurable)
Protocole : HTTPS (TLS requis en production)

Type de Contenu

Toutes les requêtes et réponses utilisent JSON :

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

Versionnage de l'API

L'API actuelle est la version 1 (implicite). Les futures versions utiliseront le versionnage par URL :

```
https://api.example.com:8443/api/v2/...
```

Authentification

Certificats Clients TLS (Recommandé)

Les déploiements en production doivent utiliser l'authentification par certificat client TLS :

```
curl --cert client.crt --key client.key \
  https://api.example.com:8443/api/status
```

Authentification par Clé API

Authentification par clé API personnalisée via l'en-tête `X-API-Key` :

```
curl -H "X-API-Key: your_api_key_here" \
  https://api.example.com:8443/api/status
```

Liste Blanche d'IP

Restreindre l'accès à l'API aux adresses IP de confiance au niveau du pare-feu.

Formats de Réponse Communs

Réponse de Succès

```
{
  "data": {
    ...
  }
}
```

Réponse d'Erreur

```
{
  "errors": {
    "detail": "Message d'erreur décrivant ce qui s'est mal passé"
  }
}
```

Réponse de Liste

```
{
  "data": [
    {...},
    {...}
  ]
}
```

Point de Terminaison de Statut

Point de vérification de santé pour la surveillance et les équilibreurs de charge.

Obtenir le Statut de l'API

Requête :

```
GET /api/status
```

Réponse (200 OK) :

```
{
  "status": "ok",
  "application": "OmniMessage",
  "timestamp": "2025-10-30T12:34:56Z"
}
```

Exemple :

```
curl https://api.example.com:8443/api/status
```

Cas d'Utilisation :

- Vérifications de santé des équilibreurs de charge
- Surveillance de la connectivité système
- Vérification de la disponibilité du service

API de File d'Attente de Messages

Points de terminaison principaux pour la soumission et la gestion des messages.

Lister les Messages

Récupérer les messages de la file d'attente.

Requête :

```
GET /api/messages
```

En-têtes Optionnels :

- `smsc: frontend_name` - Filtrer par SMSC de destination
- `include-unrouted: true|false|1|0` - Inclure les messages sans enregistrement de localisation (par défaut : false)
  - `false` (par défaut) : Ne retourner que les messages avec routage explicite ou enregistrement de localisation
  - `true` : Inclure les messages sans enregistrement de localisation (mode rétrocompatible)

Paramètres de Requête :

- `status` - Filtrer par statut : `pending`, `delivered`, `expired`, `dropped`
- `source_smsc` - Filtrer par SMSC source
- `dest_smsc` - Filtrer par SMSC de destination
- `limit` - Limiter les résultats (par défaut : 100, max : 1000)
- `offset` - Décalage de pagination

Réponse (200 OK) :

```
{
  "data": [
    {
      "id": 12345,
      "source_msisdn": "+15551234567",
      "destination_msisdn": "+447700900000",
      "message_body": "Hello World",
      "source_smsc": "api_client",
      "dest_smsc": "uk_gateway",
      "status": "pending",
      "send_time": "2025-10-30T12:00:00Z",
      "deliver_time": null,
      "delivery_attempts": 0,
      "inserted_at": "2025-10-30T12:00:00Z"
    }
  ]
}
```


Exemples :

Obtenir les messages en attente pour un SMSC spécifique (uniquement avec routage explicite ou localisation) :

```
curl -H "smsc: uk_gateway" \
  https://api.example.com:8443/api/messages
```

Obtenir les messages en attente y compris les messages non routés (rétrocompatible) :

```
curl -H "smsc: uk_gateway" \
  -H "include-unrouted: true" \
  https://api.example.com:8443/api/messages
```

Obtenir tous les messages livrés :

```
curl "https://api.example.com:8443/api/messages?
status=delivered&limit=50"
```

Obtenir un Message Unique

Récupérer les détails d'un message spécifique.

Requête :

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

Réponse (200 OK) :


```
{
  "data": {
    "id": 12345,
    "source_msisdn": "+15551234567",
    "destination_msisdn": "+447700900000",
    "message_body": "Hello World",
    "source_smsc": "api_client",
    "dest_smsc": "uk_gateway",
    "source_imsi": null,
    "dest_imsi": null,
    "message_parts": 1,
    "message_part_number": 1,
    "tp_data_coding_scheme": "00",
    "tp_user_data_header": null,
    "status": "pending",
    "send_time": "2025-10-30T12:00:00Z",
    "deliver_time": null,
    "expires": "2025-10-31T12:00:00Z",
    "deadletter": false,
    "delivery_attempts": 0,
    "charge_failed": false,
    "deliver_after": "2025-10-30T12:00:00Z",
    "raw_data_flag": false,
    "raw_sip_flag": false,
    "raw_pdu": null,
    "inserted_at": "2025-10-30T12:00:00Z",
    "updated_at": "2025-10-30T12:00:00Z"
  }
}
```

Exemple :

```
curl https://api.example.com:8443/api/messages/12345
```

Soumettre un Message (Synchronous)

Soumettre un message et recevoir immédiatement l'ID du message.

Requête :

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

Corps :

```
{
  "source_msisdn": "+15551234567",
  "destination_msisdn": "+447700900000",
  "message_body": "Hello World",
  "source_smsc": "api_client"
}
```

Champs Optionnels :

- `dest_smsc` - Remplacer la décision de routage
- `send_time` - Planifier pour une livraison future (ISO 8601)
- `message_parts` - Total des parties pour un message multipart
- `message_part_number` - Numéro de la partie (indexé à partir de 1)
- `tp_data_coding_scheme` - DCS SMS (par défaut : "00")
- `source_imsi` - IMSI de l'abonné source
- `dest_imsi` - IMSI de l'abonné de destination

Réponse (201 Created) :

```
{
  "data": {
    "id": 12345,
    "source_msisdn": "+15551234567",
    "destination_msisdn": "+447700900000",
    "message_body": "Hello World",
    "source_smsc": "api_client",
    "dest_smsc": "uk_gateway",
    "status": "pending",
    "send_time": "2025-10-30T12:00:00Z",
    "inserted_at": "2025-10-30T12:00:00Z"
  }
}
```


Exemple :

```
curl -X POST https://api.example.com:8443/api/messages \
  -H "Content-Type: application/json" \
  -d '{
    "source_msisdn": "+15551234567",
    "destination_msisdn": "+447700900000",
    "message_body": "Hello World",
    "source_smsc": "api_client"
  }'
```

Performance : ~70 messages/seconde, 14ms temps de réponse moyen

Utiliser Quand :

- Besoin de l'ID du message immédiatement
- Traitement de messages/seconde
- Nécessite une confirmation immédiate

Soumettre un Message (Asynchronous)

Soumettre un message avec un débit élevé (traitement par lots).

Requête :

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

Corps : Identique à l'endpoint synchrone

Réponse (202 Accepted) :

```
{
  "data": {
    "status": "accepted",
    "message": "Message mis en file d'attente pour traitement"
  }
}
```

Exemple :

```
curl -X POST
https://api.example.com:8443/api/messages/create_async \
  -H "Content-Type: application/json" \
  -d '{
    "source_msisdn": "+15551234567",
    "destination_msisdn": "+447700900000",
    "message_body": "Message de notification en masse",
    "source_smsc": "bulk_api"
  }'
```

Performance : ~4,650 messages/seconde, 0.22ms temps de réponse moyen

Latence : Le message apparaît dans la base de données dans les 100ms (configurable)

Utiliser Quand :

- Messagerie en masse à fort volume (> 100 msg/sec)
- Pas besoin de l'ID du message dans la réponse API
- Débit plus important que confirmation instantanée

Mettre à Jour un Message

Mettre à jour partiellement les champs d'un message.

Requête :

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

Corps :

```
{
  "dest_smsc": "alternate_gateway",
  "deliver_after": "2025-10-30T14:00:00Z"
}
```

Champs Modifiables :

- `dest_smsc` - Changer de destination
- `deliver_after` - Retarder la livraison
- `message_body` - Mettre à jour le texte du message
- `status` - Changer le statut

Réponse (200 OK) :

```
{
  "data": {
    "id": 12345,
    "dest_smsc": "alternate_gateway",
    "deliver_after": "2025-10-30T14:00:00Z",
    ...
  }
}
```

Exemple :

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

Marquer un Message Comme Livré

Marquer un message comme ayant été livré avec succès.

Requête :

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

Corps :

```
{
  "dest_smsc": "uk_gateway"
}
```

Réponse (200 OK) :

```
{
  "data": {
    "id": 12345,
    "status": "delivered",
    "deliver_time": "2025-10-30T12:05:30Z",
    "dest_smsc": "uk_gateway",
    ...
  }
}
```

Exemple :

```
curl -X POST
https://api.example.com:8443/api/messages/12345/mark_delivered \
  -H "Content-Type: application/json" \
  -d '{
    "dest_smsc": "uk_gateway"
  }'
```

Cas d'Utilisation : Appelé par les systèmes frontend après une livraison réussie

Incrémenter la Tentative de Livraison

Incrémenter le compteur de réessai et appliquer un backoff exponentiel.

Requête :

```
PUT /api/messages/:id
```

Réponse (200 OK) :

```
{
  "data": {
    "id": 12345,
    "delivery_attempts": 2,
    "deliver_after": "2025-10-30T12:08:00Z",
    ...
  }
}
```

Calcul du Backoff :

```
deliver_after = maintenant + 2^(delivery_attempts) minutes
```

Exemple :

```
curl -X PUT https://api.example.com:8443/api/messages/12345
```

Cas d'Utilisation : Appelé par le frontend après un échec de livraison pour planifier un réessai

Supprimer un Message

Supprimer un message de la file d'attente.

Requête :

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

Réponse (204 No Content)

Exemple :

```
curl -X DELETE https://api.example.com:8443/api/messages/12345
```

Avertissement : Supprimer des messages les retire définitivement. Utilisez avec précaution.

API PDU SMS Brut

Soumettre des messages SMS sous forme de PDU brut (Unité de Données de Protocole) pour une compatibilité maximale avec les systèmes hérités.

Soumettre SMS Brut (Synchronous)

Requête :

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

Corps :

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

Format PDU : PDU TP SMS encodé en hexadécimal (Unité de Données de Protocole de Transport)

Réponse (201 Created) :

```
{
  "data": {
    "id": 12346,
    "source_msisdn": "+447700900000",
    "destination_msisdn": "+447700900000",
    "message_body": "Test",
    "source_smsc": "legacy_system",
    "raw_pdu": "0001000B916407007009F0000004D4F29C0E",
    ...
  }
}
```

Exemple :

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

Soumettre SMS Brut (Asynchronous)

Requête :

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

Corps : Identique à l'endpoint synchrone

Réponse (202 Accepted) :

```
{
  "data": {
    "status": "accepted",
    "message": "PDU mis en file d'attente pour traitement"
  }
}
```

Exemple :

```
curl -X POST https://api.example.com:8443/api/messages_raw/async \
  -H "Content-Type: application/json" \
  -d '{
    "pdu": "0001000B916407007009F0000004D4F29C0E",
    "source_smsc": "legacy_gateway"
  }'
```

Gestion des PDU

Le système décode automatiquement :

1. Le PDU en utilisant les normes SMS (3GPP TS 23.040)
2. Extrait les numéros de téléphone, le texte du message, DCS
3. Détecte les rapports de livraison (CP-ACK, RP-ACK, etc.)
4. Effectue une recherche IMSI à MSISDN si nécessaire
5. Applique les règles de routage
6. Stocke le PDU original pour référence

Détection de Rapport de Livraison :

- CP-ACK, CP-ERROR - Accusés de réception du protocole de connexion
- RP-ACK, RP-ERROR, RP-SMMA - Réponses du protocole de relais
- Les rapports de livraison sont enregistrés mais ne sont pas stockés en tant que messages

API de Gestion de Localisation

Gérer les informations de localisation des abonnés pour la livraison de messages terminés sur mobile.

Lister les Localisations

Requête :

```
GET /api/locations
```

Réponse (200 OK) :


```
{
  "data": [
    {
      "id": 1,
      "msisdn": "+15551234567",
      "imsi": "001001000000001",
      "location": "msc1.region1.example.com",
      "ran_location": "cell_tower_12345",
      "imei": "123456789012345",
      "ims_capable": true,
      "csfb": false,
      "registered": true,
      "expires": "2025-10-30T13:00:00Z",
      "user_agent": "Samsung Galaxy",
      "inserted_at": "2025-10-30T12:00:00Z",
      "updated_at": "2025-10-30T12:00:00Z"
    }
  ]
}
```


Exemple :

```
curl https://api.example.com:8443/api/locations
```

Obtenir la Localisation

Requête :

```
GET /api/locations/:id
```

Réponse (200 OK) :

```
{
  "data": {
    "id": 1,
    "msisdn": "+15551234567",
    "imsi": "001001000000001",
    ...
  }
}
```

Exemple :

```
curl https://api.example.com:8443/api/locations/1
```

Créer/Mettre à Jour la Localisation

Crée une nouvelle localisation ou met à jour l'existante en fonction de l'IMSI (identifiant unique).

Requête :

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

Corps :

```
{
  "msisdn": "+15551234567",
  "imsi": "001001000000001",
  "location": "msc1.region1.example.com",
  "ran_location": "cell_tower_12345",
  "imei": "123456789012345",
  "ims_capable": true,
  "csfb": false,
  "registered": true,
  "expires": "2025-10-30T13:00:00Z",
  "user_agent": "Samsung Galaxy"
}
```

Champs Requis :

- `imsi` - Identifiant unique de l'abonné
- `msisdn` - Numéro de téléphone

Champs Optionnels :

- `location` - Adresse MSC/VLR
- `ran_location` - ID de la tour/secteur cellulaire
- `imei` - Identifiant de l'appareil
- `ims_capable` - Capacité VoLTE IMS
- `csfb` - Drapeau de repli sur circuit
- `registered` - Actuellement enregistré
- `expires` - Expiration de l'enregistrement
- `user_agent` - Modèle/info de l'appareil

Réponse (201 Created ou 200 OK) :

```
{
  "data": {
    "id": 1,
    "msisdn": "+15551234567",
    ...
  }
}
```

Exemple :

```
curl -X POST https://api.example.com:8443/api/locations \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "+15551234567",
    "imsi": "001001000000001",
    "location": "msc1.region1.example.com",
    "ims_capable": true,
    "registered": true
  }'
```

Cas d'Utilisation : Appelé par les systèmes de gestion de mobilité (HSS, MME, etc.) lorsque l'abonné s'enregistre

Mettre à Jour la Localisation

Requête :

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

Corps : Mise à jour partielle avec n'importe quel champ de localisation

Réponse (200 OK) :

```
{
  "data": {
    "id": 1,
    ...
  }
}
```

Exemple :

```
curl -X PATCH https://api.example.com:8443/api/locations/1 \
  -H "Content-Type: application/json" \
  -d '{
    "location": "msc2.region2.example.com",
    "ran_location": "cell_tower_67890"
  }'
```

Supprimer la Localisation

Requête :

```
DELETE /api/locations/:id
```

Réponse (204 No Content)

Exemple :

```
curl -X DELETE https://api.example.com:8443/api/locations/1
```

Cas d'Utilisation : Appelé lorsque l'abonné se désinscrit ou expire

API d'Enregistrement Frontend

Suivre et gérer les connexions SMSC frontend.

Lister Tous les Frontends

Requête :

```
GET /api/frontends
```

Réponse (200 OK) :

```
{
  "data": [
    {
      "id": 1,
      "frontend_name": "uk_gateway_1",
      "frontend_type": "smpp",
      "ip_address": "10.0.1.50",
      "hostname": "gateway1.uk.example.com",
      "uptime_seconds": 86400,
      "configuration": {
        "max_throughput": 1000,
        "bind_type": "transceiver"
      },
      "status": "active",
      "expires_at": "2025-10-30T12:02:00Z",
      "last_seen_at": "2025-10-30T12:00:30Z",
      "inserted_at": "2025-10-29T12:00:00Z",
      "updated_at": "2025-10-30T12:00:30Z"
    }
  ]
}
```


Exemple :

```
curl https://api.example.com:8443/api/frontends
```

Lister Seulement les Frontends Actifs

Requête :

```
GET /api/frontends/active
```

Réponse (200 OK) : Même format, seulement les frontends actifs

Exemple :

```
curl https://api.example.com:8443/api/frontends/active
```

Cas d'Utilisation : Obtenir la liste des destinations disponibles pour le routage

Obtenir les Statistiques du Frontend

Requête :

```
GET /api/frontends/stats
```

Réponse (200 OK) :

```
{
  "data": {
    "active_count": 5,
    "expired_count": 2,
    "unique_frontends": 7,
    "total_registrations": 1523
  }
}
```

Exemple :

```
curl https://api.example.com:8443/api/frontends/stats
```

Obtenir l'Historique du Frontend

Requête :

```
GET /api/frontends/history/:name
```

Réponse (200 OK) :

```
{
  "data": [
    {
      "id": 1,
      "frontend_name": "uk_gateway_1",
      "status": "active",
      "inserted_at": "2025-10-30T12:00:00Z",
      ...
    },
    {
      "id": 2,
      "frontend_name": "uk_gateway_1",
      "status": "expired",
      "inserted_at": "2025-10-29T12:00:00Z",
      ...
    }
  ]
}
```

Exemple :

```
curl
https://api.example.com:8443/api/frontends/history/uk_gateway_1
```

Enregistrer un Frontend

Enregistrer ou mettre à jour la connexion frontend.

Requête :

```
POST /api/frontends/register
Content-Type: application/json
```

Corps :

```
{
  "frontend_name": "uk_gateway_1",
  "frontend_type": "smpp",
  "ip_address": "10.0.1.50",
  "hostname": "gateway1.uk.example.com",
  "uptime_seconds": 86400,
  "configuration": {
    "max_throughput": 1000,
    "bind_type": "transceiver",
    "system_id": "gateway1"
  }
}
```

Champs Requis :

- `frontend_name` - Identifiant unique pour le frontend
- `frontend_type` - Type : `smpp`, `sip`, `http`, etc.

Champs Optionnels :

- `ip_address` - IP du frontend
- `hostname` - Nom d'hôte du frontend
- `uptime_seconds` - Temps de fonctionnement depuis le démarrage
- `configuration` - Objet de configuration personnalisé

Réponse (201 Created) :

```
{
  "data": {
    "id": 1,
    "frontend_name": "uk_gateway_1",
    "status": "active",
    "expires_at": "2025-10-30T12:01:30Z",
    ...
  }
}
```

Exemple :

```
curl -X POST https://api.example.com:8443/api/frontends/register \
  -H "Content-Type: application/json" \
  -d '{
    "frontend_name": "uk_gateway_1",
    "frontend_type": "smpp",
    "ip_address": "10.0.1.50",
    "hostname": "gateway1.uk.example.com"
  }'
```

Délai d'Enregistrement : 90 secondes (les frontends doivent se réenregistrer toutes les 60-90 secondes)

Cas d'Utilisation : Appelé périodiquement par les systèmes frontend pour maintenir le statut actif

API de Journalisation d'Événements

Suivre les événements du cycle de vie des messages.

Obtenir les Événements de Message

Requête :

```
GET /api/events/:message_id
```

Réponse (200 OK) :

```
{
  "data": [
    {
      "event_epoch": 1698672000,
      "name": "message_inserted",
      "description": "Message inséré dans la file d'attente",
      "event_source": "node1@server.example.com"
    },
    {
      "event_epoch": 1698672001,
      "name": "message_routed",
      "description": "Routé vers uk_gateway via route_id=42",
      "event_source": "node1@server.example.com"
    },
    {
      "event_epoch": 1698672005,
      "name": "message_delivered",
      "description": "Livré avec succès",
      "event_source": "node2@server.example.com"
    }
  ]
}
```


Exemple :

```
curl https://api.example.com:8443/api/events/12345
```

Types d'Événements :

- `message_inserted` - Message créé
- `message_routed` - Décision de routage prise
- `message_delivered` - Livraison réussie
- `message_failed` - Échec de la livraison
- `message_dropped` - Écarté par la route
- `auto_reply_sent` - Réponse automatique déclenchée
- `number_translated` - Transformation de numéro appliquée
- `routing_failed` - Aucune route trouvée
- `charging_failed` - Erreur de facturation

Enregistrer un Événement

Requête :

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

Corps :

```
{
  "message_id": 12345,
  "name": "custom_event",
  "description": "Description de l'événement personnalisé",
  "event_source": "external_system"
}
```

Réponse (201 Created) :

```
{
  "data": {
    "message_id": 12345,
    "name": "custom_event",
    "description": "Description de l'événement personnalisé",
    "event_source": "external_system",
    "event_epoch": 1698672010
  }
}
```

Exemple :

```
curl -X POST https://api.example.com:8443/api/events \
  -H "Content-Type: application/json" \
  -d '{
    "message_id": 12345,
    "name": "external_delivery_confirmed",
    "description": "Confirmé par le système en aval"
  }'
```

Conservation des Événements : 7 jours (configurable)

API de Message MMS

Gérer les messages du Service de Messagerie Multimédia (MMS).

Lister les Messages MMS

Requête :

```
GET /api/mms_messages
```

Réponse (200 OK) : Semblable aux messages SMS avec des champs MMS supplémentaires

Créer un Message MMS

Requête :

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

Corps :

```
{
  "source_msisdn": "+15551234567",
  "destination_msisdn": "+447700900000",
  "subject": "Photo",
  "content_type": "image/jpeg",
  "content_location": "https://cdn.example.com/media/12345.jpg",
  "message_size": 524288
}
```

Réponse (201 Created) : Objet complet du message MMS

API d'Événements SS7

Suivre les événements de signalisation SS7.

Lister les Événements SS7

Requête :

```
GET /api/ss7_events
```

Réponse (200 OK) :

```
{
  "data": [
    {
      "id": 1,
      "event_type": "MAP_UPDATE_LOCATION",
      "imsi": "001001000000001",
      "msisdn": "+15551234567",
      "timestamp": "2025-10-30T12:00:00Z",
      ...
    }
  ]
}
```

Créer un Événement SS7

Requête :

```
POST /api/ss7_events
Content-Type: application/json
```

Corps :

```
{
  "event_type": "MAP_UPDATE_LOCATION",
  "imsi": "001001000000001",
  "msisdn": "+15551234567"
}
```

Réponse (201 Created) : Objet d'événement complet

Codes d'Erreur

Codes d'État HTTP

Code	Signification	Description
200	OK	Requête réussie
201	Created	Ressource créée avec succès
202	Accepted	Requête acceptée pour traitement
204	No Content	Suppression réussie
400	Bad Request	Format de requête invalide
401	Unauthorized	Authentification requise
403	Forbidden	Permissions insuffisantes
404	Not Found	La ressource n'existe pas
422	Unprocessable Entity	Erreurs de validation
429	Too Many Requests	Limite de taux dépassée
500	Internal Server Error	Erreur du serveur
503	Service Unavailable	Indisponible temporairement

Format de Réponse d'Erreur

```
{
  "errors": {
    "detail": "Échec de la validation : destination_msisdn est requis"
  }
}
```

Messages d'Erreur Communs

Erreur	Cause	Solution
"destination_msisdn est requis"	Champ requis manquant	Inclure destination_msisdn dans la requête
"Format de numéro de téléphone invalide"	Numéro mal formé	Utiliser le format E.164 : +15551234567
"Message trop long"	Dépasse la limite de taille	Diviser en plusieurs parties
"Aucune route trouvée"	Échec du routage	Vérifier la configuration de routage
"Échec de facturation"	Erreur OCS	Vérifier la connectivité du système de facturation
"Message non trouvé"	ID de message invalide	Vérifier que l'ID existe
"Frontend non enregistré"	SMSC inconnu	Enregistrer le frontend d'abord

Limitation de Taux

Limites par Défaut

Point de Terminaison	Limite	Fenêtre
POST /api/messages	100 req/sec	Par IP
POST /api/messages/create_async	1000 req/sec	Par IP
POST /api/messages_raw	100 req/sec	Par IP
GET /api/*	1000 req/sec	Par IP

En-têtes de Limitation de Taux

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1698672060
```

Limite de Taux Dépassée

Réponse (429 Too Many Requests) :

```
{
  "errors": {
    "detail": "Limite de taux dépassée. Réessayez après 5
secondes."
  }
}
```

Meilleures Pratiques

Soumission de Messages

1. **Utiliser Async pour le Bulk** : Utiliser `/create_async` pour > 100 msg/sec
2. **Inclure source_smsc** : Identifier toujours votre système
3. **Valider les Numéros** : Utiliser le format E.164 (+code pays)
4. **Gérer les Erreurs** : Implémenter une logique de réessai pour les erreurs 5xx
5. **Vérifier le Routage** : Tester les routes avant la soumission en masse

Intégration Frontend

1. **S'enregistrer Régulièrement** : Se réenregistrer toutes les 60 secondes
2. **Interroger pour les Messages** : Interroger avec l'en-tête `smsc` pour vos messages
3. **Utiliser include-unrouted Judicieusement** : Par défaut, seuls les messages avec routage explicite ou enregistrement de localisation sont retournés. Définir `include-unrouted: true` uniquement si vous avez besoin d'un comportement rétrocompatible pour recevoir tous les messages non routés
4. **Marquer Comme Livré** : Toujours appeler mark_delivered après succès
5. **Incrémenter en Cas d'Échec** : Utiliser l'endpoint PUT pour la logique de réessai
6. **Surveiller les Événements** : Vérifier le journal des événements pour les problèmes de livraison

Performance

1. **Pool de Connexion** : Réutiliser les connexions HTTP
2. **Requêtes par Lots** : Regrouper plusieurs messages par requête
3. **Traitement Parallèle** : Effectuer des appels API concurrents
4. **Surveiller les Métriques** : Surveiller Prometheus pour les goulets d'étranglement
5. **Définir des Délais** : Utiliser un délai de 30 secondes pour les appels API

Sécurité

1. **Utiliser TLS** : Toujours utiliser HTTPS en production
2. **Valider les Certificats** : Ne pas ignorer la validation des certificats
3. **Faire Tourner les Clés API** : Changer régulièrement les clés
4. **Liste Blanche d'IP** : Restreindre aux sources connues
5. **Journaliser l'Activité API** : Surveiller les modèles suspects

Gestion des Erreurs

1. **Réessayer les Erreurs 5xx** : Les erreurs serveur sont généralement temporaires
2. **Ne Pas Réessayer 4xx** : Les erreurs client nécessitent des corrections de code
3. **Backoff Exponentiel** : Attendre plus longtemps entre les réessais
4. **Disjoncteur** : Arrêter après des échecs répétés
5. **Alerter sur les Modèles** : Surveiller les taux d'erreur

Exemple d'Intégration (Python)

```python
import requests
import time

class SMSCClient:
    def __init__(self, base_url, api_key=None):
        self.base_url = base_url
        self.session = requests.Session()
        if api_key:
            self.session.headers.update({"X-API-Key": api_key})

    def submit_message(self, from_num, to_num, text,
async_mode=False):
        endpoint = "/messages/create_async" if async_mode else
"/messages"
        url = f"{self.base_url}{endpoint}"

        payload = {
            "source_msisdn": from_num,
            "destination_msisdn": to_num,
            "message_body": text,
            "source_smsc": "python_client"
        }

        try:
            response = self.session.post(url, json=payload,
timeout=30)
            response.raise_for_status()
            return response.json()["data"]
        except requests.exceptions.RequestException as e:
            print(f"Erreur API : {e}")
            return None

    def get_pending_messages(self, smsc_name,
include_unrouted=False):
        url = f"{self.base_url}/messages"
        headers = {"smsc": smsc_name}

        # Inclure les messages non routés si demandé (mode
rétrocompatible)
        if include_unrouted:
            headers["include-unrouted"] = "true"
```

```
        try:
            response = self.session.get(url, headers=headers,
timeout=30)
            response.raise_for_status()
            return response.json()["data"]
        except requests.exceptions.RequestException as e:
            print(f"Erreur API : {e}")
            return []

    def mark_delivered(self, message_id, smsc_name):
        url = f"
{self.base_url}/messages/{message_id}/mark_delivered"
        payload = {"dest_smsc": smsc_name}

        try:
            response = self.session.post(url, json=payload,
timeout=30)
            response.raise_for_status()
            return True
        except requests.exceptions.RequestException as e:
            print(f"Erreur API : {e}")
            return False

# Utilisation
client = SMSCClient("https://api.example.com:8443/api",
api_key="your_key")

# Soumettre un message unique
result = client.submit_message("+15551234567", "+447700900000",
"Hello")
print(f"ID du message : {result['id']}")

# Soumettre des messages en masse (async)
for i in range(1000):
    client.submit_message("+15551234567", f"+44770090{i:04d}",
f"Bulk {i}", async_mode=True)

# Boucle de sondage frontend
while True:
    # Obtenir des messages avec routage explicite ou
enregistrement de localisation
    messages = client.get_pending_messages("my_gateway")
```

```
    # Ou utiliser include_unrouted=True pour un comportement
rétrocompatible
    # messages = client.get_pending_messages("my_gateway",
include_unrouted=True)

    for msg in messages:
        # Livrer le message via votre protocole
        success = deliver_via_smpp(msg)

        if success:
            client.mark_delivered(msg["id"], "my_gateway")
        else:
            # Incrémenter pour réessayer
            requests.put(f"
{client.base_url}/messages/{msg['id']}")

    time.sleep(5)  # Sondage toutes les 5 secondes
```

Journal des Modifications de l'API

Version 1 (Actuelle)

- Version initiale
- CRUD de la file d'attente de messages
- Soumission de PDU brut
- Gestion de localisation
- Enregistrement de frontend
- Journalisation des événements

Fonctionnalités Prévues

- Soumission de messages par lots (une seule requête, plusieurs messages)
- Modèles de messages
- API de livraison programmée
- Webhooks en temps réel pour les événements
- Point de terminaison API GraphQL

- Authentification OAuth2

Pour des questions ou des problèmes avec l'API, consultez le Guide de Dépannage ou contactez le support.

Référence du Schéma CDR (Call Detail Record)

← Retour à l'Index de Documentation | **README Principal**

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

La table `cdrs` stocke les enregistrements de détails d'appel pour tous les messages SMS traités par le système. Les CDR sont écrits lorsque :

- Les messages sont livrés avec succès
- Les messages expirent sans livraison
- Les messages échouent de manière permanente
- Les messages sont rejetés

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

- La facturation et la facturation
- L'analyse et le reporting
- La conformité et l'audit
- L'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
  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)
);
```

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 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 unique du message provenant de la file d'attente de messages SMS-C. Référence l'ID du message original dans Mnesia.

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 de la 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 de la 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 reçu à l'origine (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 soumis pour la première fois à la 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 encore 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 une seule partie, 2+ pour multi-part. NULL si inconnu.
`deadletter`	BOOLEAN	NON	Indique si le message a été déplacé vers la file d'attente des lettres mortes. TRUE indique que le message n'a pas pu être livré 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 du 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 la confidentialité. La longueur maximale varie selon la base de données (typiquement 65 535 caractères pour le type TEXT).

Modes de Confidentialité :

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

- **Mode de conformité :** Corps stocké crypté ou haché (nécessite une implémentation personnalisée)

Horodatages d'Audit

Champ	Type	Nullable	Description
`inserted_at`	DATETIME	NON	Quand l'enregistrement CDR a été inséré pour la première fois 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. Identique à inserted_at s'il n'a jamais été mis à jour.

Exemples SQL

Requêtes de Base

Trouver tous les CDR 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 quotidien de messages 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 des routes :

```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 CDR) :

```
-- 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
précaution !)
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;
```

Indexes

Les index suivants sont créés pour optimiser les requêtes courantes :

Nom de l'Index	Colonnes	But
`PRIMARY`	`id`	Clé primaire, assure l'unicité de l'enregistrement
`idx_cdrs_message_id`	`message_id`	Recherche CDR par ID de message original
`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
`idx_cdrs_submission_time`	`submission_time`	Requêtes basées sur le temps, périodes de facturation
`idx_cdrs_dest_smsc`	`dest_smsc`	Analyse de performance des routes

Recommandations d'Index Supplémentaires

Pour les 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 des routes :

```
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 du message (MySQL) :

```
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 des types de champs à travers les bases de données supportées :

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). Envisagez ces mesures de confidentialité :

1. Confidentialité du Corps du 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 exportations
  hide_message_body_in_export: true
```

2. Masquage des Numéros de Téléphone

Pour les 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 la 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 transparent des données (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 (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, 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 des États
Télécom (UE)	6 mois - 2 ans	RGPD, ePrivacy
Financier	5-7 ans	SOX, SEC
Santé	6 ans	HIPAA

Stratégie d'Archivage

1. Partition 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 CDR 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 CDR 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 les Systèmes de Facturation

Export CSV :

```
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 CDR via l'API REST

Référence de Configuration SMS-C

← Retour à l'Index de Documentation | **README Principal**

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

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 journaliseur
- Paramètres de développement/test
- Paramètres d'optimisation des performances

config/runtime.exs

Configuration à l'exécution chargée au démarrage. Contient :

- Paramètres de connexion à la base de données
- Configuration du 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.

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 :

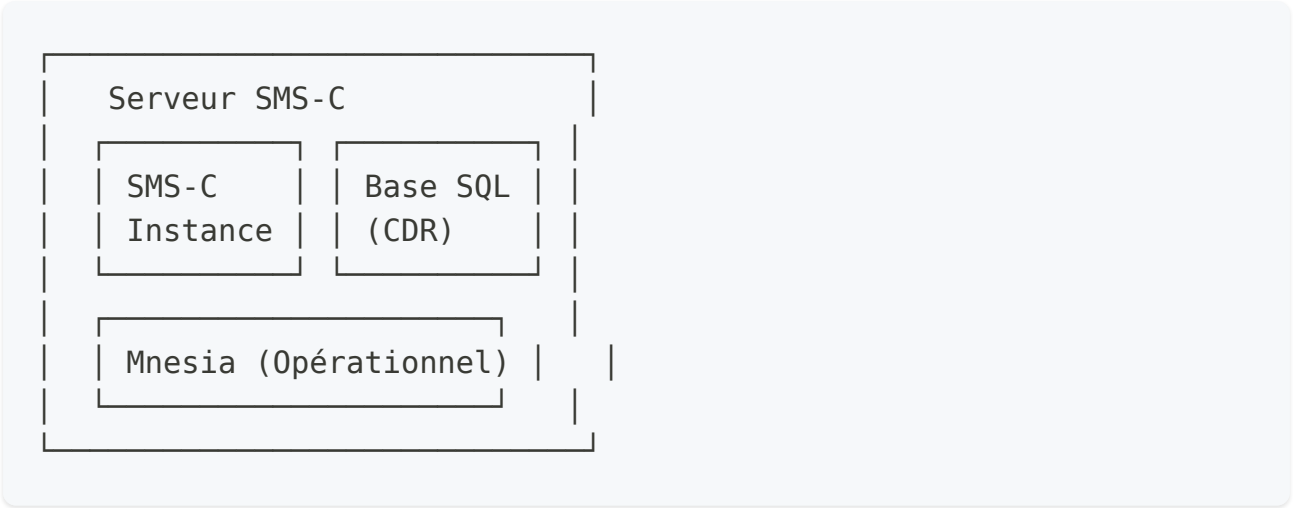
- Fonctionnalités avancées JSON/JSONB pour l'analyse

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

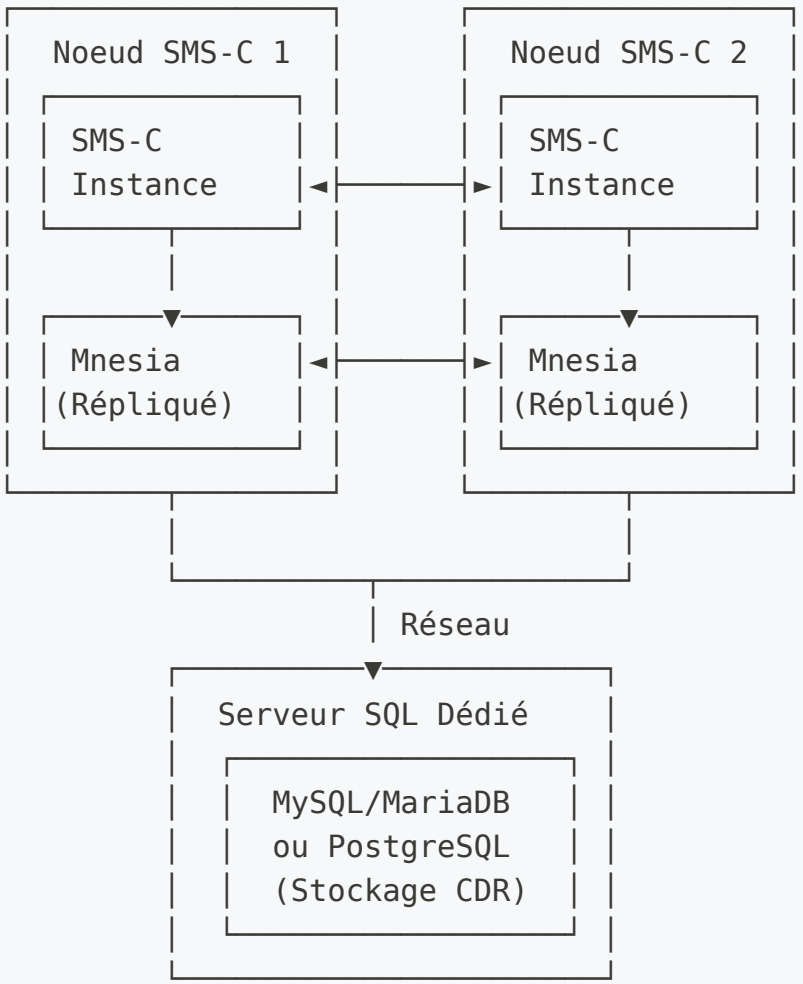
Topologies de Déploiement

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 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 fonctionnement de SMS-C
- **Gestion des Données** : Stockage CDR centralisé pour plusieurs instances SMS-C
- **Flexibilité de Sauvegarde** : Horaires de sauvegarde et politiques de conservation 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 haut 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 de messages.

Configuration de Base de l'API

```
# config/runtime.exs
config :api_ex,
  port: String.to_integer(System.get_env("API_PORT") || "8443"),
  listen_ip: System.get_env("API_LISTEN_IP") || "0.0.0.0",
  enable_tls: System.get_env("API_ENABLE_TLS") != "false"
```

Configuration TLS/SSL

Configuration de Production avec TLS (Recommandé) :

```
config :api_ex,
  port: 8443,
  listen_ip: "0.0.0.0",
  enable_tls: true,
  tls_cert_path: "/etc/sms_c/certs/server.crt",
  tls_key_path: "/etc/sms_c/certs/server.key"
```

Configuration de Développement sans TLS :

```
config :api_ex,
  port: 8080,
  listen_ip: "127.0.0.1",
  enable_tls: false
```

Configuration de Certificat API

Générez un certificat auto-signé pour les tests :

```
# Créer le répertoire de certificats
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 le 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) :

Configurer 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,
  port: String.to_integer(System.get_env("WEB_PORT") || "80"),
  hostname: System.get_env("WEB_HOSTNAME") || "localhost",
  enable_tls: System.get_env("WEB_ENABLE_TLS") == "true"
```

Configuration de l'Interface Web de Production

```
config :control_panel,
  port: 443,
  hostname: "sms-admin.example.com",
  enable_tls: true,
  tls_cert_path: "/etc/sms_c/certs/web.crt",
  tls_key_path: "/etc/sms_c/certs/web.key"
```

Configuration de Proxy Inverse (Recommandé)

Utilisez Nginx ou Apache comme proxy inverse 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 "SMS-C Admin";
  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 des déploiements multi-contrôleurs. Les contrôleurs se découvrent mutuellement 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 aux 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`	Boolean	`false`	Interrupteur principal pour la fédération.
`dns_srv_domain`	String	`""`	Domaine DNS SRV pour la découverte des pairs.
`dns_poll_interval_ms`	Integer	`30000`	Intervalle de re-résolution DNS.
`health_check_interval_ms`	Integer	`10000`	Intervalle de vérification de santé des pairs.
`registry_sync_interval_ms`	Integer	`15000`	Intervalle d'échange de registre frontend.
`forward_retry_interval_ms`	Integer	`5000`	Intervalle de réessai des messages en file d'attente.
`forward_max_retries`	Integer	`50`	Nombre maximal de tentatives de transfert avant de marquer comme échoué.
`http_timeout_ms`	Integer	`5000`	Délai d'attente pour tous les appels HTTP des pairs.

Paramètre	Type	Par Défaut	Description
`api_port`	Integer	`8443`	Port API pour construire les URLs des pairs.
`static_peers`	List	`[]`	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 (conservation prolongée)
- **10080** - 7 jours (conservation 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_de_tentatives) minutes
```

Tentative	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

Tentatives maximales 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** : Fréquence d'exécution du travail de nettoyage (par défaut : 10)
- **fingerprint_ttl_minutes** : Fenêtre de détection des doublons (par défaut : 5)
- **event_ttl_days** : Conservation des journaux d'événements (par défaut : 7)

Configuration de la Facturation

Intégration avec OCS pour la facturation et le chargement en ligne.

Activer la Facturation

```
# config/runtime.exs
config :sms_c,
  default_charging_enabled: true,
  ocs_url: "http://ocs.internal.example.com:2080/jsonrpc",
  ocs_tenant: "sms.example.com",
  ocs_destination: "default",
  ocs_source: "sms_platform",
  ocs_subject: "sms_user",
  ocs_account: "default_account"
```

Désactiver la Facturation

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

Lorsqu'elle est désactivée, tous les messages sont traités sans vérifications de facturation.

Configuration de Facturation par Locataire

```
config :sms_c,
  ocs_url: System.get_env("OCS_URL") ||
"http://localhost:2080/jsonrpc",
  ocs_tenant: System.get_env("OCS_TENANT") ||
"tenant1.example.com",
  ocs_account: System.get_env("OCS_ACCOUNT") || "default"
```

Variables d'Environnement par Locataire :

```
# Locataire 1
export OCS_TENANT=tenant1.example.com
export OCS_ACCOUNT=tenant1_account

# Locataire 2
export OCS_TENANT=tenant2.example.com
export OCS_ACCOUNT=tenant2_account
```

Comportement en Cas d'Échec de Facturation

Configurez ce qui se passe lorsque la facturation échoue :

```
config :sms_c,
  charging_failure_action: :allow  # ou :deny
```

- **:allow** - Traiter le message même si la facturation échoue (journaliser l'erreur)
- **:deny** - Rejeter le message si la facturation échoue

Exemple de Connexion OCS

Tester la Connectivité OCS :

```
# Tester l'API OCS
curl -X POST http://ocs.internal.example.com:2080/jsonrpc \
  -H "Content-Type: application/json" \
  -d '{
    "method": "SessionSv1.AuthorizeEvent",
    "params": [{
      "Tenant": "sms.example.com",
      "Account": "test_account",
      "Destination": "1234567890",
      "Usage": 100
    }],
    "id": 1
  }'
```

Réponse attendue :

```
{
  "id": 1,
  "result": {
    "Attributes": {},
    "MaxUsage": 100,
    ...
  }
}
```

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. Voir 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.mnc005.mcc547.3gppnetwork.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.mnc005.mcc547.3gppnetwork.org",
        ip: "10.0.0.1",
        port: 3868,
        realm: "epc.mnc005.mcc547.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
```

Paramètre	Type	Requis	Par Défaut	Description
`diameter_enabled`	Boolean	Non	`false`	Activer ou désactiver la recherche d'abonnés HSS dans le chemin de routage
`mock_sh`	Boolean	Non	`false`	Utiliser des réponses HSS fictives pour les tests (aucune pile Diameter d��marrée)
`mock_sh_on_net_numbers`	List	Non	`[]`	MSISDN traités comme sur le réseau en mode fictif

Pour la référence complète des paramètres de la pile Diameter (hôte, royaume, pairs, applications), voir 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 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 ce qu'une recherche réussisse :

```
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 Liens 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 de Numéros

Normalisation des numéros basée sur des expressions régulières appliquée avant le routage.

Désactiver la Traduction de 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})$",           # Correspondre aux
numéros de 10 chiffres
      calling_replace: "+1\1",               # Préfixer +1
      called_match: "^(\d{10})$",
      called_replace: "+1\1",
      priority: 100,
      description: "Ajouter +1 aux numéros nord-américains de 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-Règle

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

```
config :sms_c,
  translation_rules: [
    # Priorité 1 : Règles les plus spécifiques d'abord
    %{
      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 des 10 chiffres",
      enabled: true
    }
  ]
```

Configuration de Routage

Les règles de routage initiales sont chargées au premier démarrage. Voir le Guide de Routage SMS pour la documentation complète sur le routage.

Charger les Routes depuis 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,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: false,
      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,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: false,
      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,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: false,
      weight: 30,
      priority: 50,
      description: "Passerelle de secours du Royaume-Uni (30%)",
      enabled: true
    },

    # Seulement sur le réseau — 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,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: true,
      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 depuis 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

Voir le Guide d'Optimisation des Performances pour des stratégies d'optimisation détaillées.

Travail d'Insertion en Lot

```
# 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 de 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 de Journalisation

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 pour la Production : `:info` ou `:warning` **Recommandé pour le Développement** : `:debug`

Destinations de Sortie des Journaux

Console Seulement (Développement) :

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

Journaliseur de Fichier (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 sms_c
    endscript
}
```

Scénarios de Configuration Courants

Agrégateur à Volume Élevé

Optimisé pour un débit maximal (5,000+ messages/seconde) :

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

# Travail par lot
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# Rétention des Messages
config :sms_c,
  dead_letter_time_minutes: 1440  # 24 heures

# Facturation (désactivée pour la performance)
config :sms_c,
  default_charging_enabled: false

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

Messagerie en Temps Réel pour Entreprises

Optimisé pour une faible latence (< 20ms) :

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

# Travail par lot (faible latence)
config :sms_c,
  batch_insert_batch_size: 20,
  batch_insert_flush_interval_ms: 10

# Rétention des Messages
config :sms_c,
  dead_letter_time_minutes: 4320  # 3 jours

# Facturation (activée)
config :sms_c,
  default_charging_enabled: true,
  ocs_url: "http://ocs.local:2080/jsonrpc"
```

Développement/Test

Optimisé pour le débogage et la visibilité :

```
# Base de Données
config :sms_c, SmsC.Repo,
  pool_size: 5

# Travail par lot (immédiat)
config :sms_c,
  batch_insert_batch_size: 1,
  batch_insert_flush_interval_ms: 10

# Journalisation (verbose)
config :logger, :console,
  level: :debug

# Rétention des Messages (courte)
config :sms_c,
  dead_letter_time_minutes: 60  # 1 heure

# Facturation (désactivée)
config :sms_c,
  default_charging_enabled: false
```

Fournisseur de Services Multi-Locataires

Configuration séparée par locataire :

```
# Environnement Locataire 1
export DB_NAME=sms_c_tenant1
export OCS_TENANT=tenant1.example.com
export OCS_ACCOUNT=tenant1_account
export NODE_NAME=sms_tenant1@node1.example.com

# Environnement Locataire 2
export DB_NAME=sms_c_tenant2
export OCS_TENANT=tenant2.example.com
export OCS_ACCOUNT=tenant2_account
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"  # Cross-région pour DR
  ],
  smsc_node_name: "us-east-1a"
```

Validation de Configuration

Testez 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

# Tester la connectivité OCS (si activée)
curl -X POST http://localhost:2080/jsonrpc -H "Content-Type:
application/json" \
  -d '{"method":"SessionSv1.Ping","params":[],"id":1}'

# 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 de nœud Erlang	`sms@node1.example.com`
`ERLANG_COOKIE`	Secret de cluster	`shared_cookie_value`
`OCS_URL`	URL de l'API OCS	`http://ocs.local:2080/jsonrpc`
`OCS_TENANT`	Locataire OCS	`sms.example.com`

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** dans un environnement de staging avant la production
3. **Documentez les Paramètres Personnalisés** dans les notes de déploiement
4. **Contrôlez les Fichiers de Configuration en Version** (exclure les 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
Échec de connexion à la base de données	Mauvaises informations d'identification/hôte	Vérifiez les variables d'environnement DB_*
API non accessible	Mauvais port/IP de liaison	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 injoignable	Testez la connectivité à ocs_url
Échecs de recherches ENUM	Serveur DNS injoignable	Testez la connectivité DNS, vérifiez le délai d'attente
Mauvaises performances	Mauvaises configurations de lot	Consultez le Guide d'Optimisation des Performances
Messages non 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)

Rétention 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)
  message_retention_hours: 24,

  # 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 : ~1Ko
- 10,000 messages : ~10Mo
- 100,000 messages : ~100Mo

Exportation 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 une analyse de facturation.

```
config :sms_c,
  # Activer/désactiver l'écriture CDR
  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 sur la base de données

Contrôles de Confidentialité

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

```
config :sms_c,
  # Supprimer le corps du message de Mnesia après 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 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 Exemples :

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 Complète)

```
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 (rétention : 24h)
[info] Exportation CDR : ACTIVÉE
[info] Suppression 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**

Aperçu

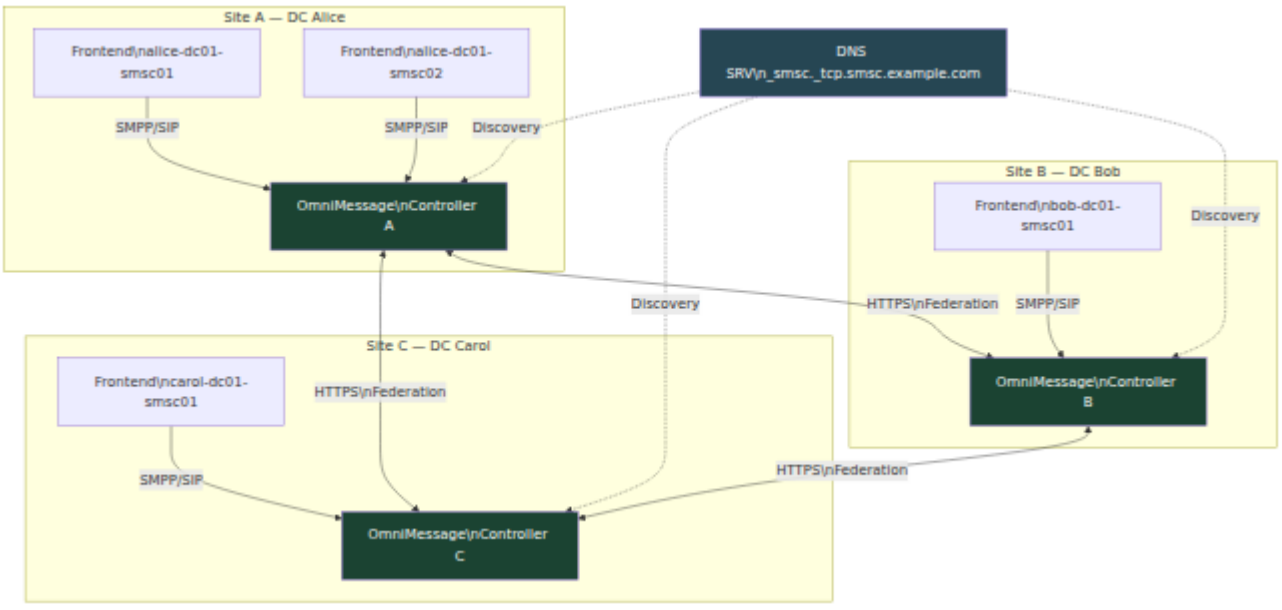
OmniMessage utilise une **fédération basée sur HTTP** avec un **modèle de tirage** pour des déploiements multi-contrôleurs à travers des centres de données ou des régions. Chaque contrôleur fonctionne indépendamment — gérant sa propre file de messages, sa table de routage et son ensemble de frontaux 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 non fiables
État partagé	Aucun — chaque contrôleur est indépendant
Propriété des messages	Les messages restent sur le contrôleur d'origine jusqu'à la livraison
Sécurité	HTTPS avec TLS
Ports	Port HTTPS unique (8443)
Cible de mise à l'échelle	5-20 contrôleurs
Gestion des partitions	Réessais du poller — pas de split-brain

Architecture



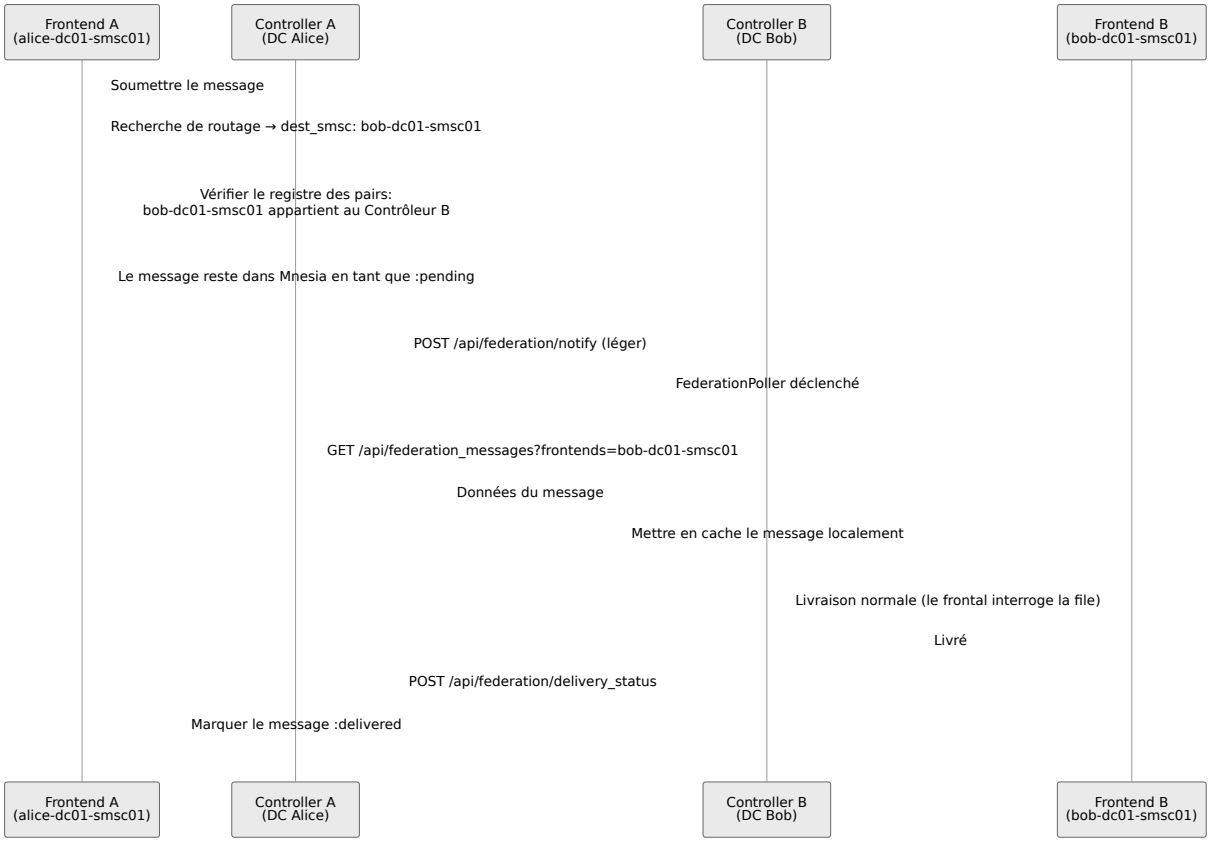
Décisions de Conception Clés

- **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 sur 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 sur la santé et les 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)

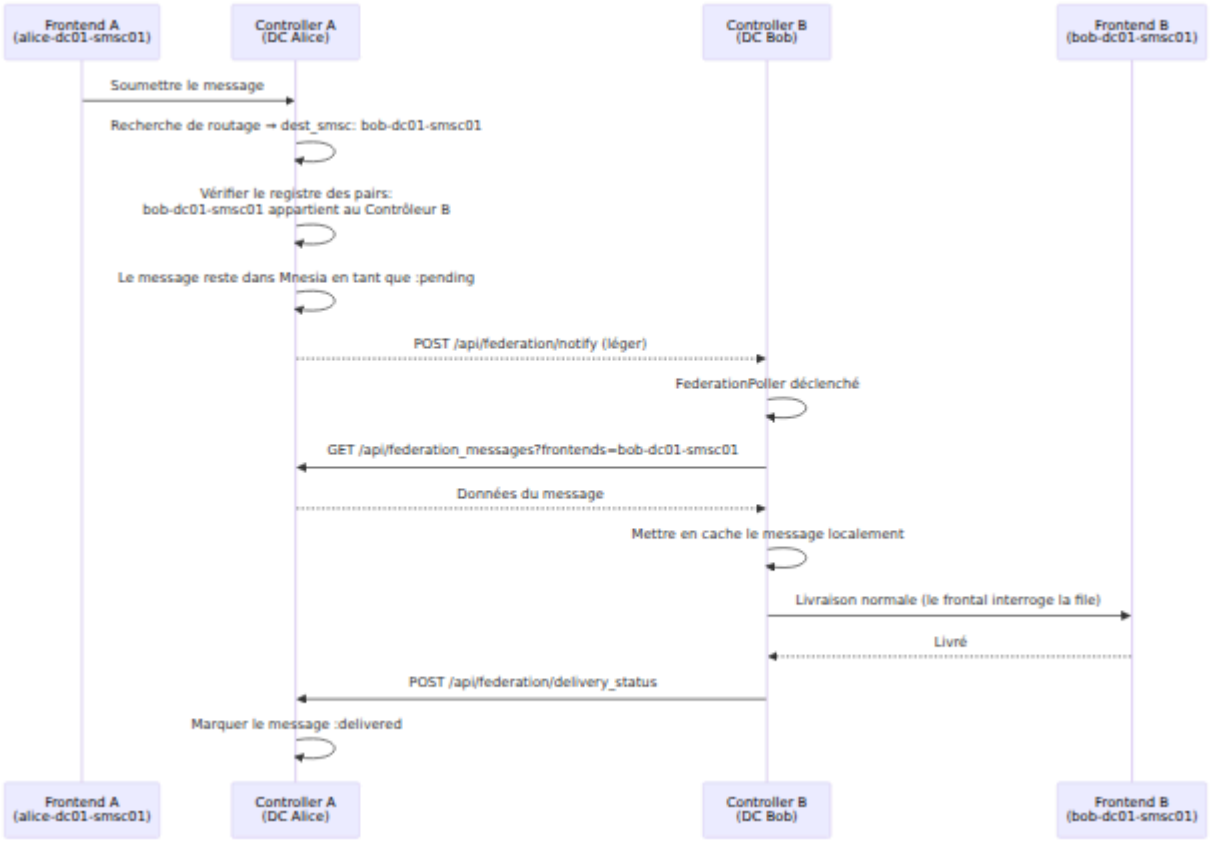
Lorsqu'un message arrive à un contrôleur et que le routage détermine que le frontal de destination se trouve sur un contrôleur distant :

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



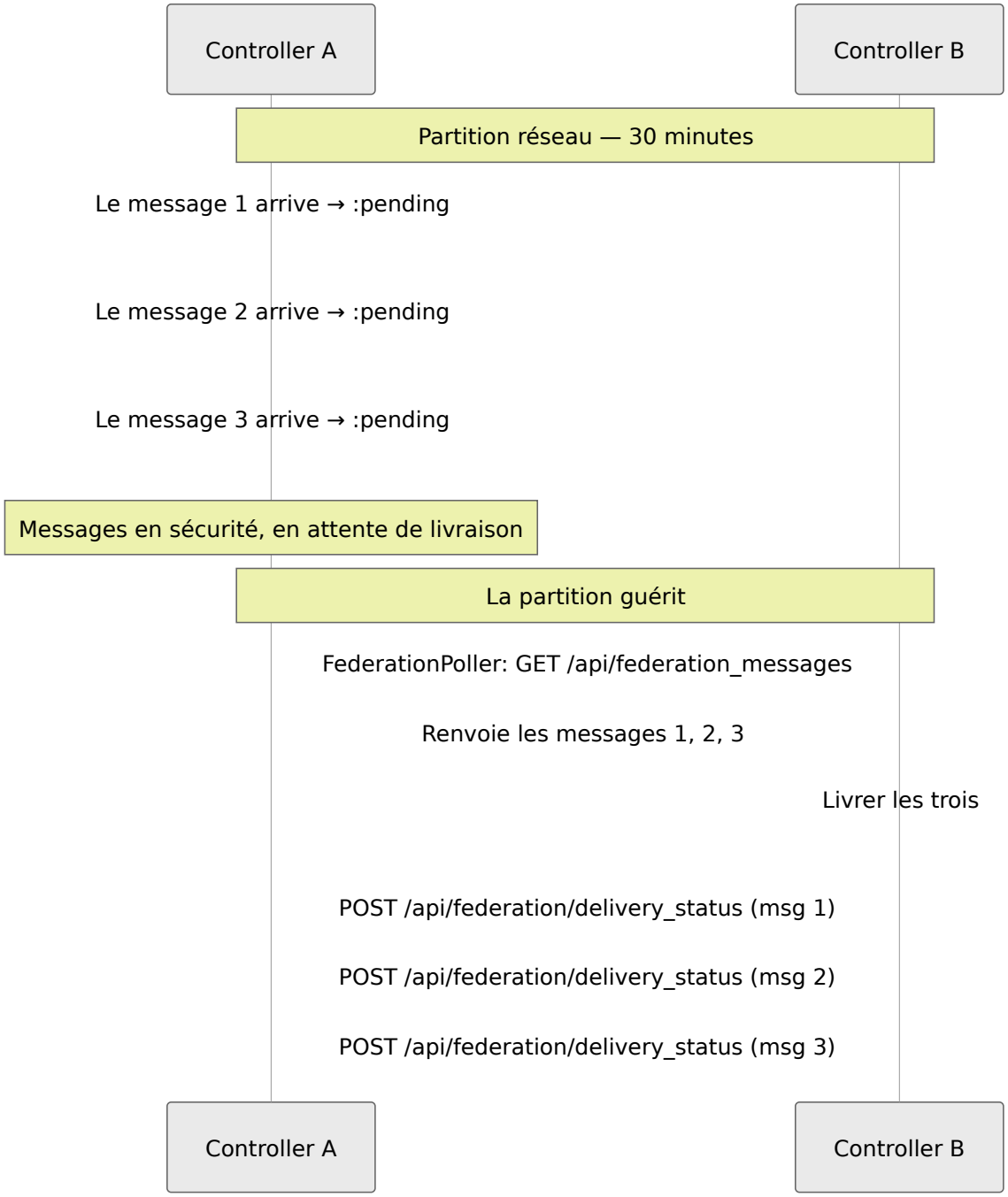
Pair Inaccessible — Échec de Notification

Si le contrôleur de destination est hors service 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.



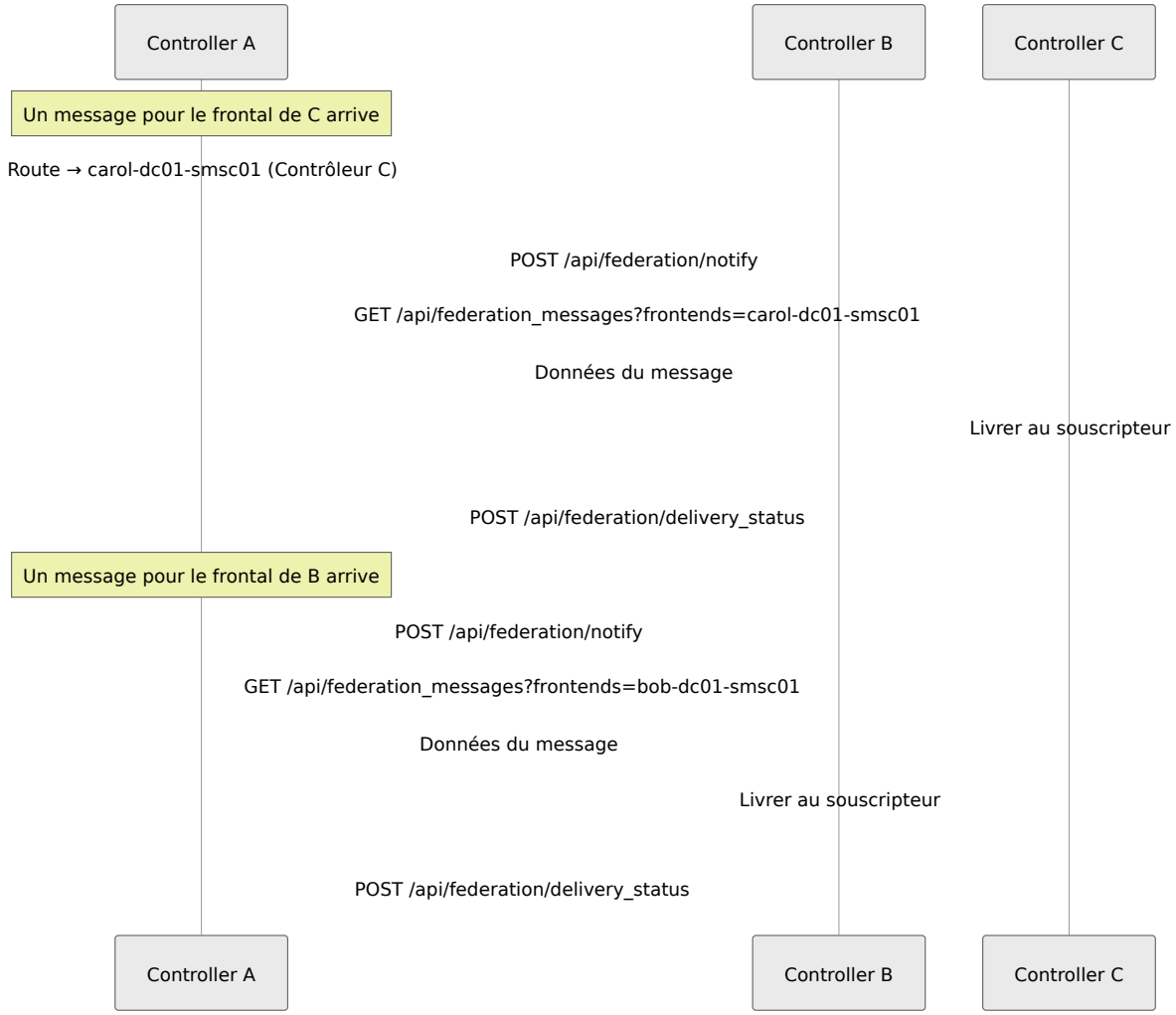
Récupération de Partition — Messages Enregistrés Multiples

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

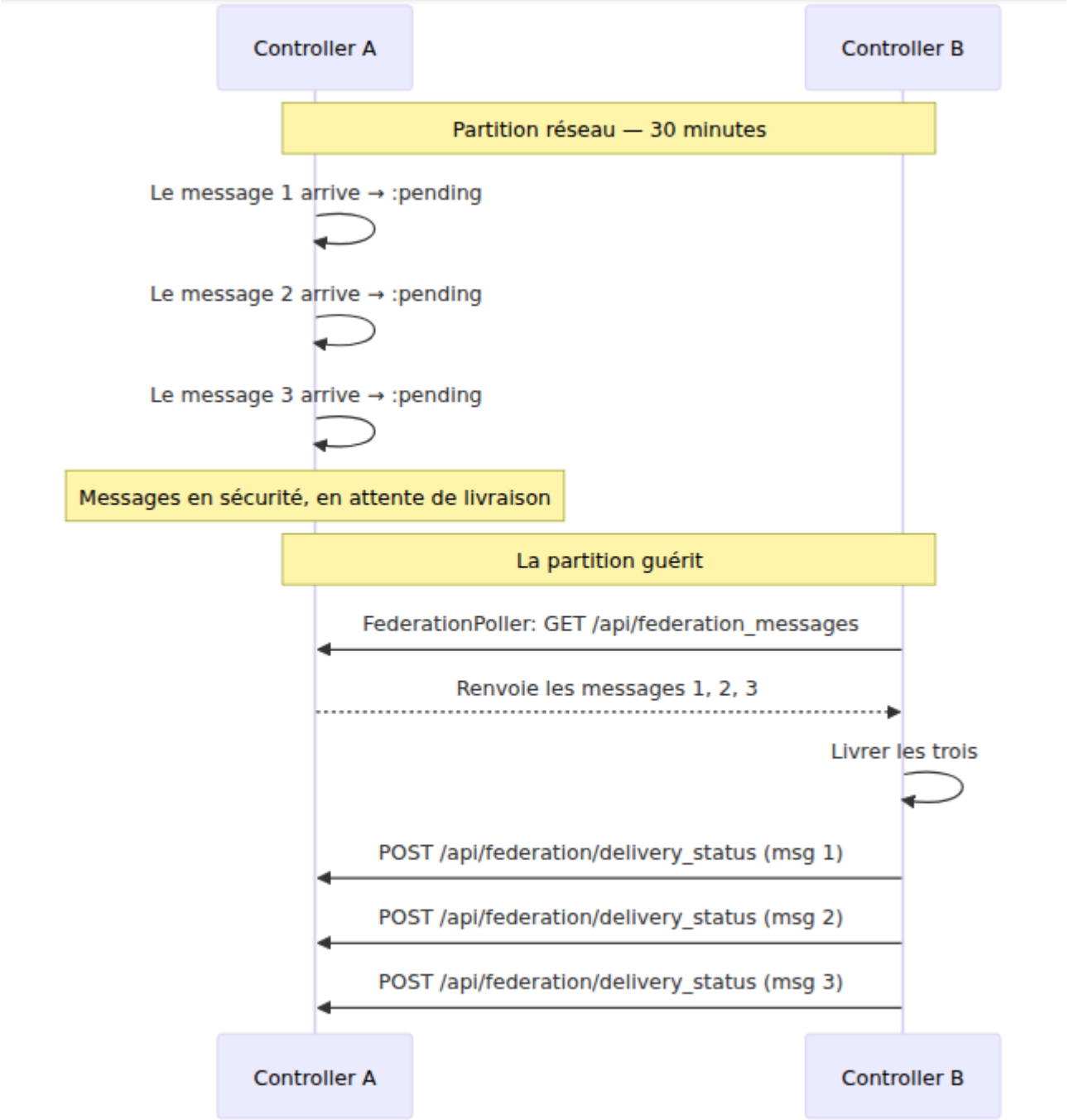
Les contrôleurs de fédération maintiennent une conscience les uns des autres à travers deux cycles périodiques :

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 lorsque false
  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 d'attente 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 pairs 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 des registres des 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 d'attente pour chaque appel HTTPS

Paramètre	Type	Requis	Par défaut	Description
				individuel à un pair. S'applique aux vérifications de santé, à la synchronisation des registres et à la récupération de messages.
`api_port`	Integer	Non	`8443`	Port API utilisé lors de la construction des URL de pairs à partir des enregistrements DNS SRV. Doit correspondre au port API configuré dans `config :api_ex`.

Paramètre	Type	Requis	Par défaut	Description
`static_peers`	List	Non	`[]`	Liste des configurations de pairs pour les environnemer sans DNS SRV Chaque entré 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
```

Ajouter 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).

Supprimer 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 uniquement 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 des 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

Tous les points de terminaison de fédération sont servis sur le port API standard (8443) sous `/api/federation/`. Ces points de terminaison sont appelés par les contrôleurs pairs, et non par des clients externes ou des 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 callee 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	Retourner 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 Statique

Deux centres de données connectés par un lien dédié. Aucun 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 font via 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 des vérifications de santé par pair **Étiquettes**:

- `peer` - Identifiant du pair
- `result` - `ok` ou `failed`

Requêtes d'exemple:

```
# 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 ne montrent aucun pair découvert ; `cluster_status` renvoie vide.

Causes possibles:

- Le domaine DNS SRV est incorrect ou non résolvable
- Le serveur DNS est 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 la fédération
3. Vérifiez les journaux d'application pour les messages "Découverte DNS de la fédération" au démarrage
4. Si vous utilisez des pairs statiques, vérifiez que la liste `static_peers` contient des entrées 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 service
- 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 permettent 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 de 0 sur la destination.

Causes possibles:

- Le contrôleur de destination est non sain (le FederationPoller n'interroge que 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 "Federation poller started")
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 la documentation | **Guide de routage SMS**

Aperçu

OmniMessage peut interroger le Serveur Abonné Domicile (HSS) via l'interface Diameter Sh pour déterminer si un abonné appelé appartient au réseau avant de prendre une décision de routage. Cela empêche les messages destinés aux abonnés sur le réseau d'être mal routés vers une passerelle par défaut lorsque l'abonné est temporairement hors ligne ou non enregistré.

Sans la recherche HSS activée, un message à un abonné sur le réseau qui n'est pas actuellement enregistré passera par la table de routage standard. Si une route de secours ou par défaut existe, le message sera envoyé à une passerelle externe — ce qui est incorrect pour un abonné qui appartient au réseau.

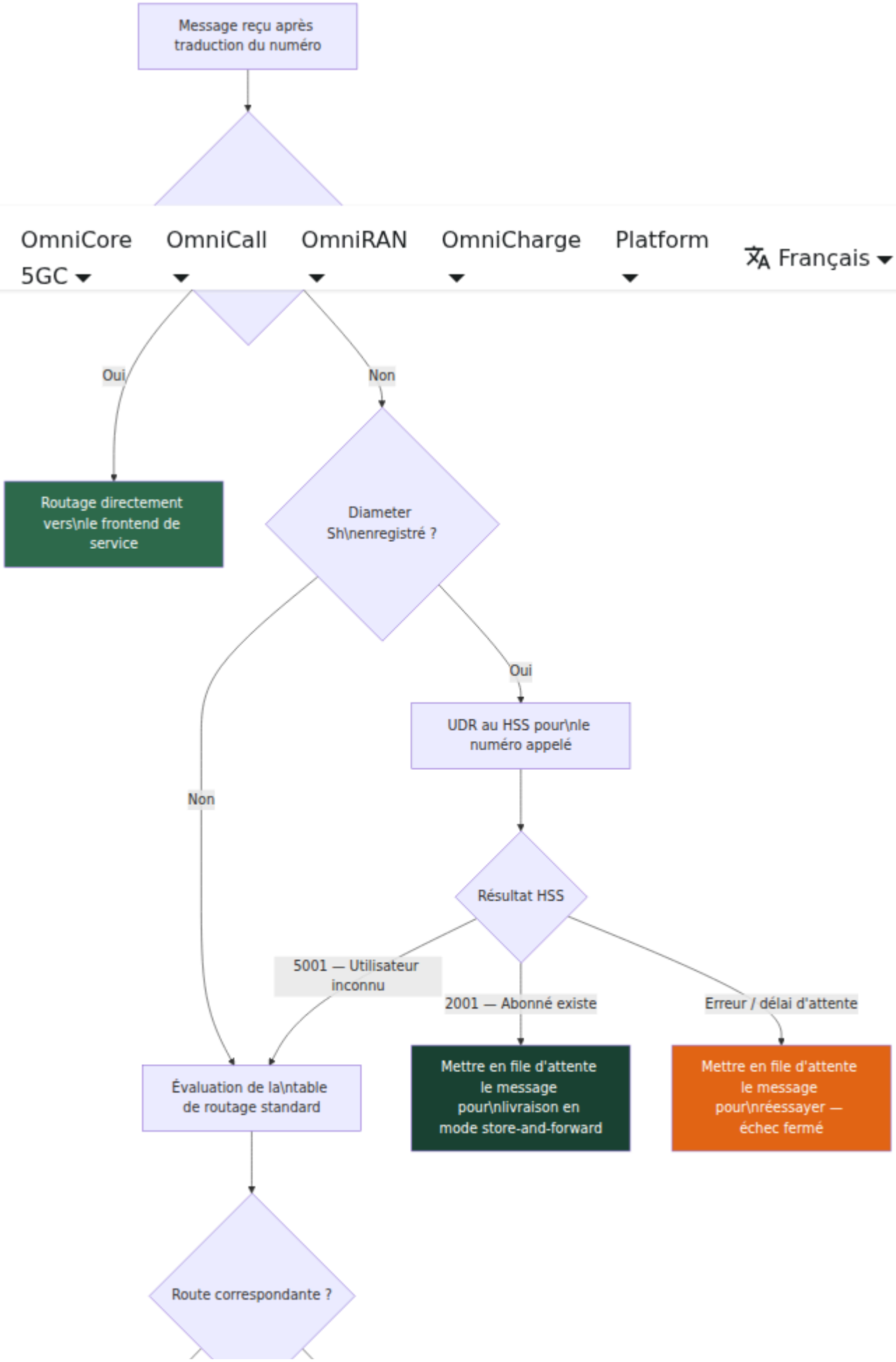
Avec la recherche HSS activée, OmniMessage envoie une **Demande de Données Utilisateur (UDR)** au HSS pour le numéro appelé. Si le HSS confirme que l'abonné existe (code de résultat 2001), le message est conservé dans la file d'attente des messages pour une livraison en mode store-and-forward lorsque l'abonné s'enregistre à nouveau. Si le HSS renvoie "utilisateur inconnu" (code de résultat 5001), l'abonné est hors réseau et le routage standard se poursuit normalement.

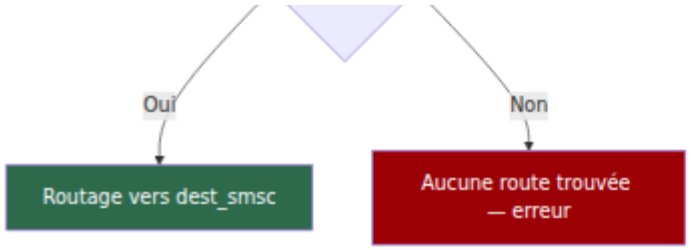
Spécifications pertinentes

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

Chaîne de priorité de routage

La recherche HSS s'inscrit dans la décision de routage d'OmniMessage comme **Priorité 2**, entre la vérification de localisation/enregistrement et la table de routage basée sur regex.


Message reçu après traduction du numéro
Oui
Non
Routage directement vers\nle frontend de service
Diameter Sh\nenregistré ?
Oui
Non
UDR au HSS pour\nle numéro appelé
Résultat HSS
5001 — Utilisateur inconnu
2001 — Abonné existe
Erreur / délai d'attente
Évaluation de la\ntable de routage standard
Mettre en file d'attente le message pour\nlivraison en mode store-and-forward
Mettre en file d'attente le message pour\nréessayer — échec fermé
Route correspondante ?



Priorité 1 — Routage basé sur la localisation

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.

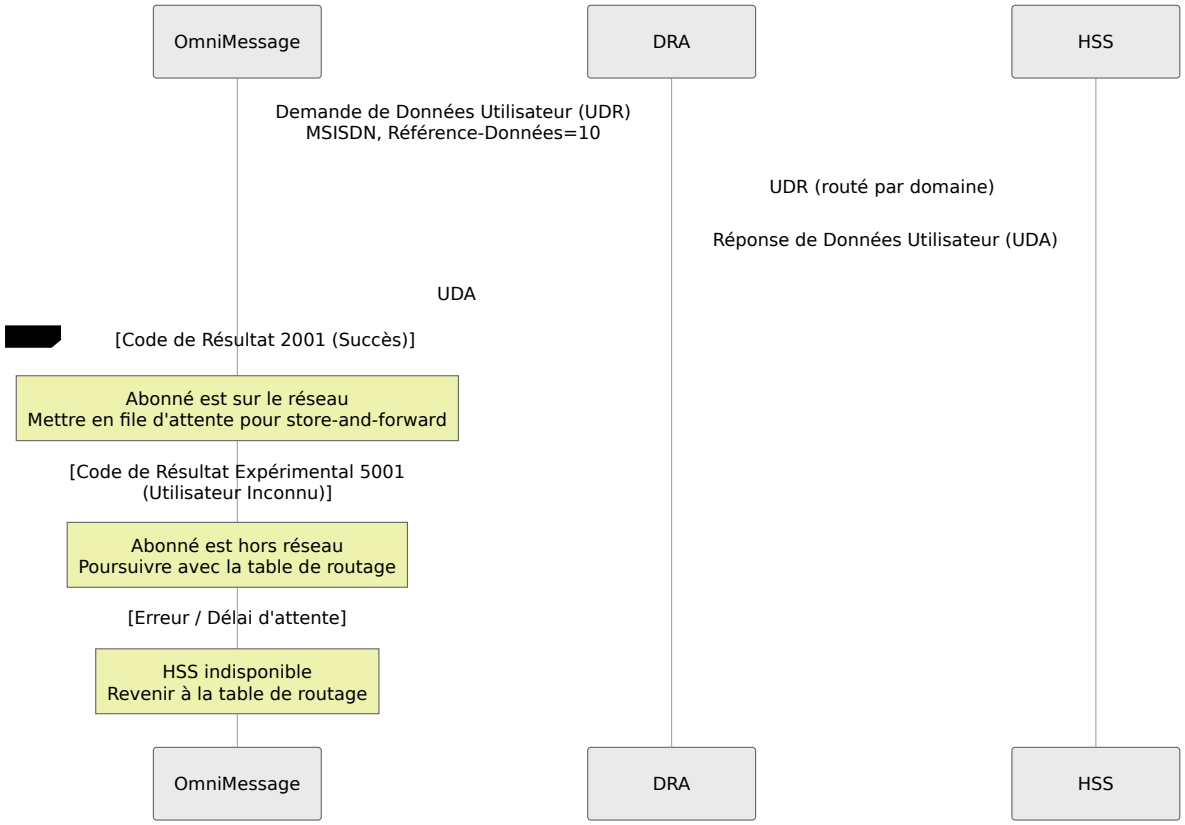
Priorité 2 — Recherche d'abonné HSS

Si aucun enregistrement n'est trouvé et que Diameter Sh est activé, OmniMessage interroge le HSS pour déterminer si l'abonné est sur le réseau. Un abonné sur le réseau qui est simplement hors ligne aura son message mis en file d'attente — il sera livré lorsque l'abonné s'enregistrera et qu'une mise à jour de localisation arrivera.

Priorité 3 — Table de routage

Si le HSS confirme que l'abonné est inconnu (hors réseau), ou si Diameter n'est pas activé, la table de routage standard basée sur regex est évaluée. Voir le Guide de routage SMS pour plus de détails.

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 TBCD	Le numéro de l'abonné appelé
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

Code	Type	Signification	Action d'OmniMessage
2001	Code de Résultat	Succès — abonné trouvé	Mettre en file d'attente le message pour store-and-forward
5001	Code de Résultat Expérimental	Utilisateur inconnu — non dans le HSS	Poursuivre avec la table de routage
Autre	Code de Résultat Expérimental	Erreur inattendue	Journaliser un avertissement, mettre en file d'attente pour réessayer (échec fermé)

Configuration

Activation de la recherche HSS

La recherche HSS est contrôlée par deux blocs de configuration dans `config/runtime.exs` :

1. Le drapeau `diameter_enabled` sous `:sms_c` — contrôle si le dip HSS est effectué dans le chemin de routage
2. La configuration `:diameter_ex` — configure la pile Diameter elle-même (identité, pairs, applications)

Configuration minimale

```
# 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.mnc005.mcc547.3gppnetwork.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.mnc005.mcc547.3gppnetwork.org",
```

```
        ip: "10.0.0.1",
        realm: "epc.mnc005.mcc547.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
```

Paramètres OmniMessage

Paramètre	Type	Requis	Par défaut	Description
`diameter_enabled`	Booléen	Non	`false`	Activer ou désactiver le dip d'abonné HSS dans le chemin de routage. Lorsque `false`, tous les messages sautent la vérification HSS et vont directement à la table de routage.
`mock_sh`	Booléen	Non	`false`	Utiliser des réponses HSS fictives au lieu de vraies requêtes Diameter. Pour les tests uniquement.
`mock_sh_on_net_numbers`	Liste	Non	`[]`	Liste des MSISDN traités comme sur 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`	ID fournisseur 3GPP
`auth_application_id`	Entier	`16_777_217`	ID 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

Plusieurs pairs

Configurer 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.mnc005.mcc547.3gppnetwork.org",
    ip: "10.0.0.1",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  },
  %{
    port: 3868,
    host: "dra02.epc.mnc005.mcc547.3gppnetwork.org",
    ip: "10.0.0.2",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  }
]
```

Mode fictif

Pour les environnements de test sans HSS en direct, activez le mode fictif :

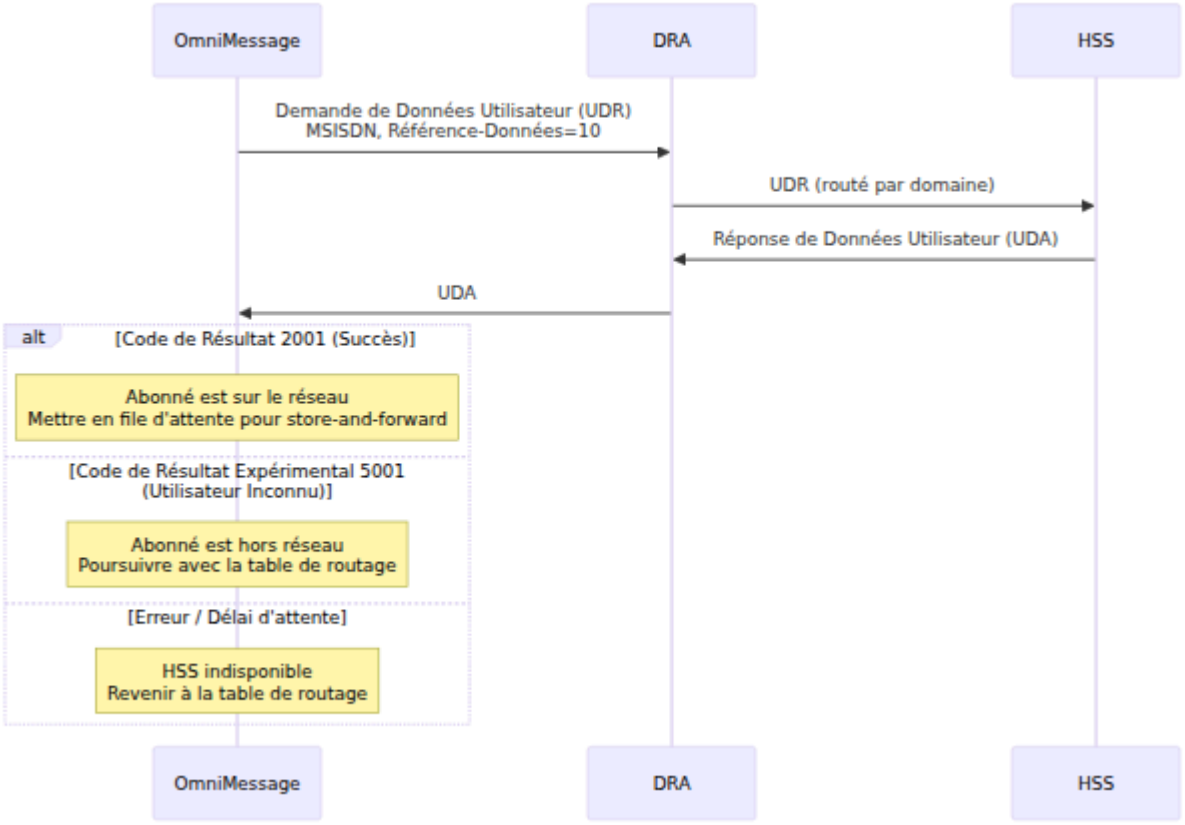
```
config :sms_c,
  diameter_enabled: true,
  mock_sh: true,
  mock_sh_on_net_numbers: [
    "68987654321",
    "68912345678"
  ]
```

En mode fictif, tout MSISDN dans la liste `mock_sh_on_net_numbers` est traité comme un abonné sur le réseau (équivalent au code de résultat HSS 2001). Tous les autres MSISDN sont traités comme inconnus (équivalent au code de résultat 5001). Aucune pile Diameter n'est démarrée et aucune connexion réseau n'est établie.

Livraison en mode store-and-forward

Lorsqu'un message est mis en file d'attente pour un abonné sur le réseau qui n'est pas enregistré, le message reste dans la file d'attente des messages Mnesia avec `dest_smsc` défini sur `nil`. 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 route 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 de lettres mortes selon le `dead_letter_time_minutes` configuré.



Notes opérationnelles

Journalisation au démarrage

OmniMessage journalise l'état de Diameter Sh au démarrage :

- **Activé** : `Diameter Sh (HSS): ENABLED (host: smsc01, realm: epc.mnc005.mcc547.3gppnetwork.org, peers: 2)`
- **Désactivé** : `Diameter Sh (HSS): DISABLED`

HSS Inaccessible (Échec fermé)

Si la recherche HSS échoue pour une raison quelconque — délai d'attente, aucun pair connecté, réponse non gérée — OmniMessage journalise un avertissement et **met le message en file d'attente pour réessayer**. Le message reste avec `dest_smsc` défini sur `nil` et sera réévalué lors du prochain cycle de sondage, moment auquel le dip HSS est tenté à nouveau.

C'est un design **échec fermé** : sans réponse définitive du HSS, OmniMessage ne routera pas le message vers une passerelle hors réseau. Cela empêche les abonnés sur le réseau d'avoir leurs messages envoyés incorrectement à des transporteurs externes pendant les pannes du HSS.

Lorsque **Diameter est explicitement désactivé** (`diameter_enabled: false`), les messages sont routés via la table de routage standard sans aucune vérification HSS. C'est un choix délibéré de l'opérateur — le comportement d'échec fermé ne s'applique que lorsque Diameter est activé mais que la recherche échoue.

Entrées de journal d'événements

Les types d'événements suivants sont enregistrés dans le journal des événements de message lorsque la recherche HSS est active :

Type d'événement	Description
`hss_dip_on_net`	Abonné trouvé dans le HSS (code de résultat 2001). Message mis en file d'attente pour store-and-forward.
`hss_dip_off_net`	Abonné non trouvé dans le HSS (code de résultat 5001). Poursuite avec le routage standard.
`hss_dip_error`	Échec de la recherche HSS (délai d'attente, pas de connexion, etc.). Message mis en file d'attente pour réessayer.

Tous les événements incluent des métadonnées structurées : `result_code`, `hss` (nom d'hôte du pair), `duration_ms`, `session_id`, `destination_msisdn`, et `message_id`.

Les événements de routage standard (`sms_routing_started`, `sms_routing_selected`, etc.) continuent d'être enregistrés pour les messages qui passent à la table de routage.

Application de la contrainte sur le réseau au niveau de la route

En plus du dip HSS de Priorité 2 (qui s'applique globalement avant le routage), des routes individuelles peuvent appliquer des exigences sur le réseau à l'aide de trois drapeaux. Ceux-ci sont vérifiés **après** qu'une route correspond et **avant** que le message soit attribué à `dest_smsc` de cette route.

Drapeaux de route

Drapeau	Description	Vérification HSS	Co en
`on_net_only`	Alias obsolète pour `terminating_on_net_only`	Vérifie le numéro appelé	Su me
`originating_on_net_only`	Exiger que l'expéditeur (partie appelante) soit sur le réseau	Vérifie le numéro appelant	Su me
`terminating_on_net_only`	Exiger que la destination (partie appelée) soit sur le réseau	Vérifie le numéro appelé	Su me

Tous les trois drapeaux **échouent fermés** : si le HSS est inaccessible ou si Diameter est désactivé, le message est supprimé plutôt que livré à une destination non vérifiée.

Exemple de configuration

```
config :sms_c, :sms_routes, [
  # Passerelle SMPP sur le réseau — n'accepte que les messages
pour des destinations confirmées sur le réseau
  %{
    calling_regex: nil,
    called_regex: ~r/^1907/,
    source_smsc: nil,
    dest_smsc: "local-gateway",
    source_type: nil,
    auto_reply: false,
    drop: false,
    charged: :default,
    weight: 100,
    priority: 10,
    description: "Passerelle locale — destinations sur le réseau
uniquement",
    on_net_only: false,
    originating_on_net_only: false,
    terminating_on_net_only: true,
    enabled: true
  },

  # Sauvegarde hors réseau — catch-all pour les abonnés confirmés
hors réseau par le HSS
  %{
    calling_regex: nil,
    called_regex: nil,
    source_smsc: nil,
    dest_smsc: "external-gateway",
    source_type: :ims,
    auto_reply: false,
    drop: false,
    charged: :default,
    weight: 100,
    priority: 255,
    description: "Passerelle externe — sauvegarde hors réseau",
    on_net_only: false,
    originating_on_net_only: false,
    terminating_on_net_only: false,
    enabled: true
```

```
  }
]
```

Comment ça fonctionne : Lorsqu'un message arrive pour un abonné non enregistré :

1. **Vérification de localisation** — aucun enregistrement actif trouvé
2. **Dip HSS** — abonné confirmé sur le réseau → message mis en file d'attente pour store-and-forward (pas de routage)
3. **Dip HSS** — abonné confirmé hors réseau → table de routage évaluée
4. La route "local-gateway" correspond mais a `terminating_on_net_only: true` — le HSS a déjà confirmé hors réseau, donc cette route est ignorée (message supprimé à cette route)
5. La route "external-gateway" correspond sans exigence sur le réseau → message routé vers la passerelle externe

Cas d'utilisation : Empêcher le trafic hors réseau d'être envoyé à une infrastructure locale qui ne peut servir que des abonnés sur le réseau, tout en s'assurant que le trafic hors réseau atteigne un transporteur externe.

Interaction avec le dip HSS de Priorité 2

Le dip HSS de Priorité 2 et les drapeaux de niveau de route sur le réseau servent des objectifs différents :

Mécanisme	Quand	Objectif	Comportemer en cas d'éche
Dip HSS de Priorité 2	Avant le routage	Mettre en file d'attente les abonnés sur le réseau pour store-and-forward	Échec fermé (mettre en file d'attente pour réessayer)
`terminating_on_net_only`	Après correspondance de la route	Appliquer l'exigence sur le réseau sur des routes spécifiques	Échec fermé (supprimer le message)
`originating_on_net_only`	Après correspondance de la route	Appliquer l'exigence d'origine sur le réseau	Échec fermé (supprimer le message)

Expiration de la localisation

Le routage basé sur la localisation (Priorité 1) vérifie le timestamp `expires` sur les enregistrements d'abonnés. Si l'enregistrement d'un abonné a expiré — par exemple, s'il est passé en mode avion — la localisation est considérée comme inactive et le routage se poursuit vers la Priorité 2 (dip HSS) et la Priorité 3 (table de routage).

Cela empêche les messages d'être envoyés à un frontend qui ne peut plus atteindre l'abonné, évitant ainsi des échecs de livraison inutiles et des cycles

de réessai.

Dépannage

Messages toujours routés vers la passerelle par défaut pour les abonnés sur le réseau

Symptômes : Les messages à des abonnés sur le réseau qui sont hors ligne sont envoyés à la passerelle par défaut au lieu d'être mis en file d'attente.

Causes possibles :

- `diameter_enabled` est défini sur `false`
- 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 les journaux pour les avertissements "Échec du dip HSS")
- 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
3. Recherchez les avertissements "Échec du dip HSS" dans le journal d'application
4. Confirmez que le format du numéro appelé après traduction correspond à ce que le HSS attend (E.164 sans +)

Messages mis en file d'attente mais jamais livrés

Symptômes : Les messages sont correctement mis en file d'attente pour les abonnés sur le réseau mais ne sont jamais livrés, même après que l'abonné s'enregistre.

Causes possibles :

- Les mises à jour de localisation n'atteignent pas OmniMessage
- Le MSISDN dans l'enregistrement de localisation ne correspond pas à `destination_msisdn` du message mis en file d'attente
- Les messages ont expiré et ont été déplacés vers la file de 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 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 d'hôte de pair (le `host` 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 | **README principal**

Vue d'ensemble

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, la performance 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 qui peuvent être récupérées par un serveur Prometheus. Les métriques sont mises à jour en temps réel à mesure que le système traite les messages.

Convention de nommage des métriques

Toutes les métriques suivent le modèle : `sms_c.<category>.<metric_name>.<type>`

Catégories :

- `license` - Métriques d'état de la 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 des files 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 : État 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 garantir que le système fonctionne avec une licence valide. En cas d'invalidité, les messages sont toujours reçus mais routés vers la destination "NOLICENCE" au lieu du routage normal.

Comportement en cas de licence 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**

Alertes :

```
- 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: "L'état 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 de la 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. Utiliser pour détecter les modèles de trafic, identifier les périodes de forte activité et repérer les anomalies dans le flux de messages.

Alertes : 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` : Indique si la validation a réussi (true ou false)

Cas d'utilisation : Suivre les taux de réussite/échec de validation. Des taux d'échec élevés peuvent indiquer des messages malformés ou des problèmes d'intégration.

Alertes : 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 la validation, le routage et la mise en file d'attente).

Unité : Millisecondes

Seaux : 10, 50, 100, 250, 500, 1000, 2500, 5000 ms

Labels :

- `success` : Indique si le traitement a réussi (true ou false)

Cas d'utilisation : Surveiller la performance de traitement des messages de bout en bout. Identifier les ralentissements dans le pipeline de traitement.

Alertes : 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 correspondante
- `dest_smsc` : SMSC de destination sélectionnée par la route

- `priority` : Valeur de priorité de la route correspondante

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.

Alertes : 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.

Alertes : 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 de réponses automatiques et les modèles de transfert.

Alertes : 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 sélectionnée

Cas d'utilisation : Surveiller la performance du moteur de routage. Un routage lent indique trop de routes ou une logique de correspondance complexe.

Alertes : 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 de cache élevés réduisent la charge DNS et améliorent les performances.

Alertes : 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.

Alertes : 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` : Indique si la recherche a réussi (true ou false)
- `cache_hit` : Indique si le résultat a été servi à partir du cache (true ou false)

Cas d'utilisation : Surveiller la performance des recherches ENUM. Identifier les serveurs DNS lents ou les problèmes de réseau.

Alertes : 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.

Alertes : 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 une SMSC de destination.

Labels :

- `dest_smsc` : Nom de la SMSC de destination

Cas d'utilisation : Surveiller le flux de messages vers chaque destination. Utile pour la planification de capacité.

Alertes : 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 de la 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 à la SMSC de destination.

Labels :

- `dest_smsc` : Nom de la SMSC de destination

Cas d'utilisation : Suivre les livraisons réussies par destination. Principal indicateur de succès.

Alertes : 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 de la 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.

Alertes : 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 d'attente des 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.

Alertes : 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 de la SMSC de destination

Cas d'utilisation : Comprendre le comportement de réessai. La plupart des livraisons devraient réussir du premier coup.

Alertes : 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 un échec final.

Seaux : 1, 2, 3, 5, 10

Labels :

- `dest_smsc` : Nom de la 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 totale de bout en bout depuis qu'un message entre dans le SMS-C jusqu'à ce que la livraison soit confirmée, avec 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 requises (0 = succès du premier essai)

Cas d'utilisation : Analyser la performance 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.

Alertes :

- 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 par
rapport au succès du premier essai
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 dans les 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 dans les 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
```

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 confondus).

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.

Alertes : 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.

Alertes : 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.

Alertes : 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.

Alertes : 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.

Alertes : Alerter lorsque l'âge du message le plus ancien dépasse le seuil SLA (par exemple, > 300 secondes).

Métriques de facturation

sms_c_charging_requested_count

Type : Counter

Description : Nombre total de demandes de facturation effectuées vers l'OCS ou le système de facturation.

Labels :

- `account` : Identifiant du compte en cours de facturation

Cas d'utilisation : Suivre le volume de facturation par compte. Utile pour la réconciliation des factures.

---

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 réussite 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.

Alertes : 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 la performance du système de facturation. Une facturation lente peut retarder la livraison des messages.

Alertes : 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` - Nombre de règles de traduction
- `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.

Alertes : 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 des 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 état de connexion.

Labels :

- `frontend_name` : Identifiant du frontend
- `status` : État de connexion (connected, disconnected)

Cas d'utilisation : Surveiller la connectivité des frontends. Détecter les échecs de connexion.

Alertes : Alerter lorsque les 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` : Indique si l'abonné prend en charge l'IMS (true/false)

Cas d'utilisation : Surveiller l'activité d'enregistrement des abonnés. Suivre les abonnés IMS par rapport aux abonnés non-IMS. Détecter les tempêtes d'enregistrement ou les échecs.

Alertes : Définir des alertes pour :

- Baisse du 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 emplacement (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` : Indique 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 de SMSC.

Alertes : Définir des alertes pour :

- Baisse soudaine des enregistrements pour un emplacement (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 emplacement
sum by (location) (sms_c_location_active_registrations_count)

# Enregistrements actifs capables d'IMS par emplacement
sum by (location)
(sms_c_location_active_registrations_count{ims_capable="true"})

# Taux d'adoption IMS par emplacement
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)

# Emplacements 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 :

- `route` : Route de l'API (par exemple, `/api/messages`, `/api/frontends`)

Seaux : 10ms, 50ms, 100ms, 250ms, 500ms, 1s, 2.5s, 5s

Cas d'utilisation : Surveiller la performance de l'API. Identifier les points de terminaison lents. Suivre les SLA de temps de réponse.

Alertes : 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 complétées, catégorisées par route et code d'état HTTP.

Labels :

- `route` : Route de l'API
- `status` : Code d'état HTTP (200, 201, 400, 404, 500, etc.)

Cas d'utilisation : Surveiller le volume des requêtes API et les taux de réussite. Suivre les taux d'erreur par point de terminaison.

Alertes : 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 point de terminaison
sum by (route) (rate(phoenix_endpoint_stop_count[5m]))

# Taux d'erreur par point de terminaison
sum by (route) (rate(phoenix_endpoint_stop_count{status=~"5.."}
[5m])) /
sum by (route) (rate(phoenix_endpoint_stop_count[5m]))

# Taux de réussite
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 l'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.

Alertes : Définir des alertes pour toute valeur non nulle sur des points de terminaison critiques.

Exemple de requête :

```
# Taux d'exceptions 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 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.

Alertes : 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.

Alertes : 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.

Alertes : Alerter lorsque > 10 * nombre de cœurs CPU de manière constante.

---

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.

Alertes : Alerter lorsque le nombre de processus approche la limite (par défaut 262 144).

Collecte et sondage des métriques

Le système collecte automatiquement les métriques suivantes toutes les 10 secondes :

- Tailles et âges des files 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 totale des messages de bout en bout

Surveiller le temps total de la réception du message 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 pour 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 forte variance de trafic.

Utilisation des routes

Identifier quelles routes gèrent le plus de trafic :

Métrique : `sms_c_routing_route_matched_count` regroupée par `route_id`

Interprétation : Utiliser pour identifier les routes chaudes pour l'optimisation et la planification de capacité.

Tendance des arriérés 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` (actuel en attente)
- `sms_c_queue_oldest_message_age_seconds` (âge en tendance)

Interprétation : Compte en attente croissant + âge en augmentation = arriéré en formation.

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	Sévéri
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	Sévéri
Échecs de facturation	`charging_failed_count` > seuil	Élevé
Traitement des messages lent	`message_processing_stop_duration` p95 > 1000ms	Avertisse

Recommandations pour les tableaux 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 des files d'attente (en attente, échouées, livrées)
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 de la performance 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 de l'utilisation des routes
4. Comptes d'actions de réponse automatique et de suppression
5. Statistiques d'utilisation ENUM
6. Volume de facturation par compte

Conservation des métriques

Paramètres de conservation 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 des messages 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])))
```

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)
```

Échecs de routage par minute :

```
rate(sms_c_routing_failed_count[5m]) * 60
```

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 (nécessite 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 la performance 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 couleurs : 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 du 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 du routage.

Panneaux :

1. **Messages par type de source (Graphique en secteurs)**
    - 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]))`
    - 10 principales sources
    - Barres horizontales
3. **Utilisation des routes (Tableau)**
    - Colonnes :
        - ID de route
        - 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)**
   - Supprimés : `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 au cours des 30 derniers jours

3. **Tendance des arriérés de messages (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 dans les 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 retard. 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."
```

File de lettres mortes croissante :

```
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: "Taux de succès 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 du 95e percentile : {{ $value
}}"
  description: "Les messages nécessitent plusieurs tentatives.
Vérifiez la fiabilité de la destination."
```

Traitement des messages 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 du 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 dans le 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 jauge 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.)

Guide de Traduction des Numéros SMS-C

← Retour à l'Index de Documentation | **README Principal**

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 aux 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 prévention des boucles
- **Transformations indépendantes 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	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
`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

Champ	Type	Description	Requis
`continue`	booléen	Continuer à évaluer les règles après correspondance (par défaut : false)	Non

Remarque : Les règles sont évaluées par ordre de priorité (le plus bas en premier). Seules les règles activées sont évaluées.

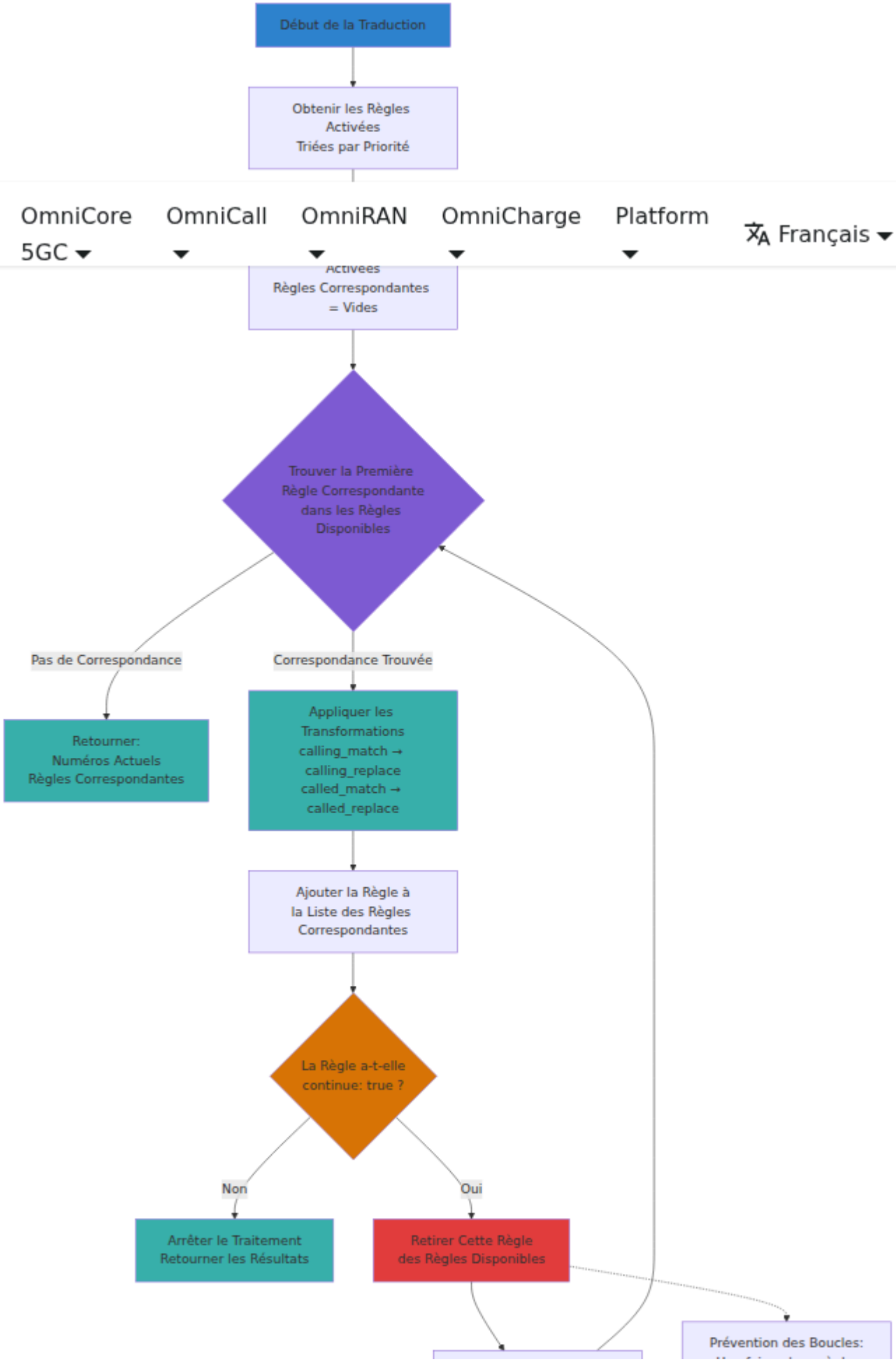
Algorithme de Traduction

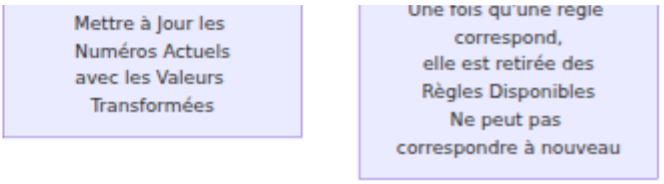
Lors de la traduction des numéros, le système :

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 des règles tout en prévenant les boucles infinies :


Début de la Traduction
Obtenir les Règles Activées Triées par Priorité
Activées
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 (nil match/replace) 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

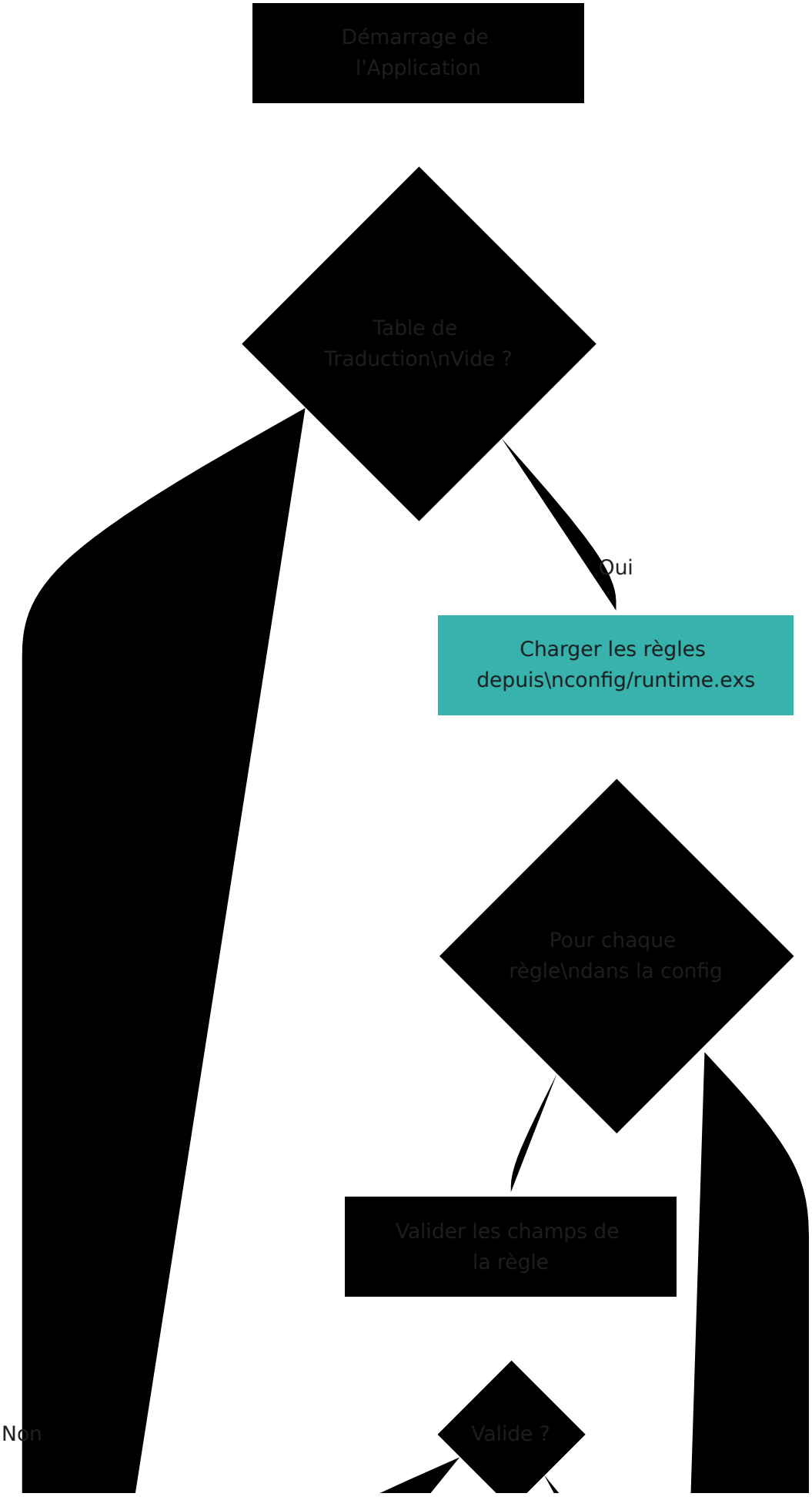
Configuration

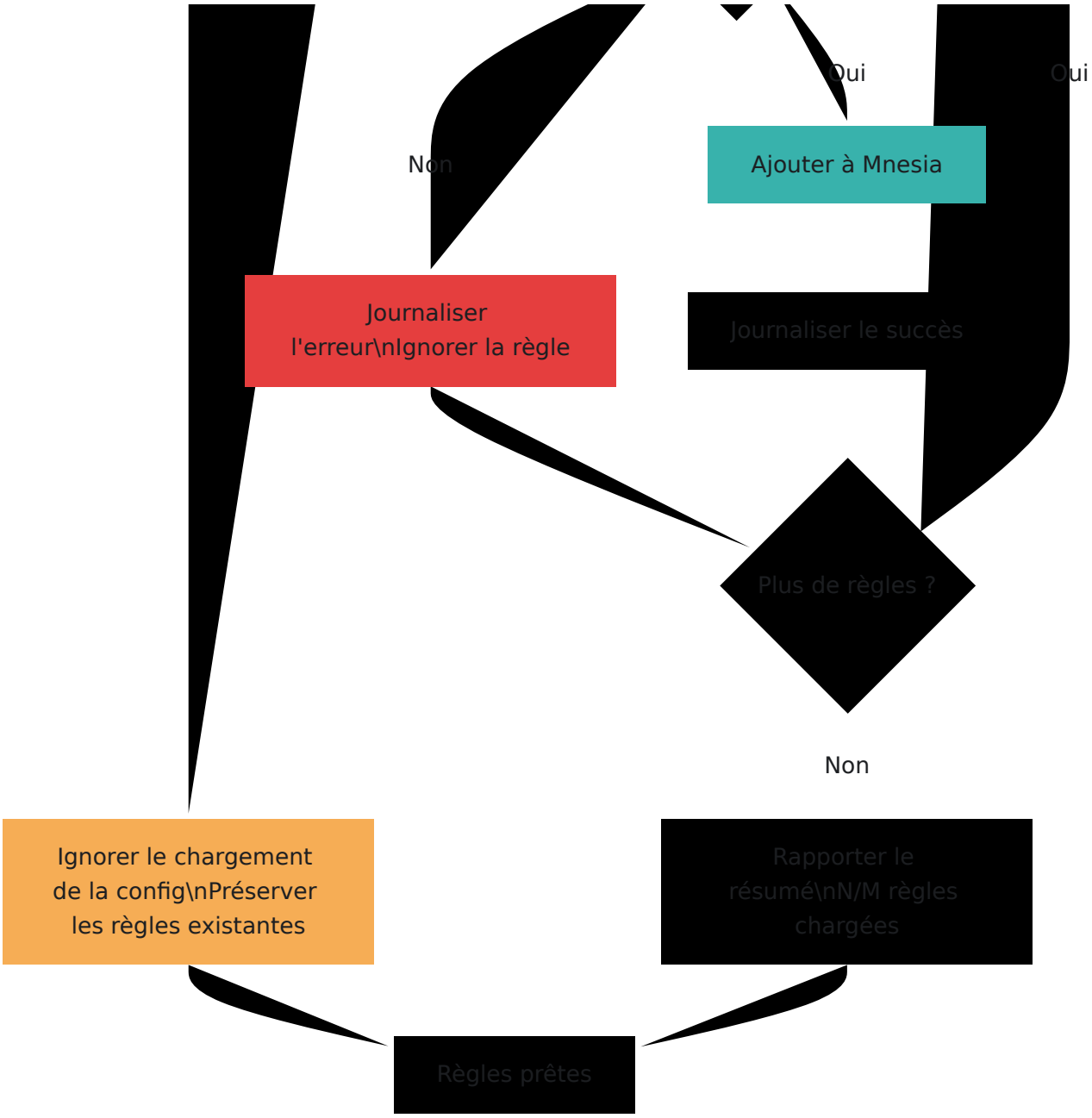
Chargement des Règles depuis le Fichier de Configuration

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


Démarrage de l'Application
Table de Traduction\nVide ?
Oui
Charger les règles depuis\nconfig/runtime.exs
Pour chaque règle\ndans la config
Valider les champs de la règle
Valide ?
Non



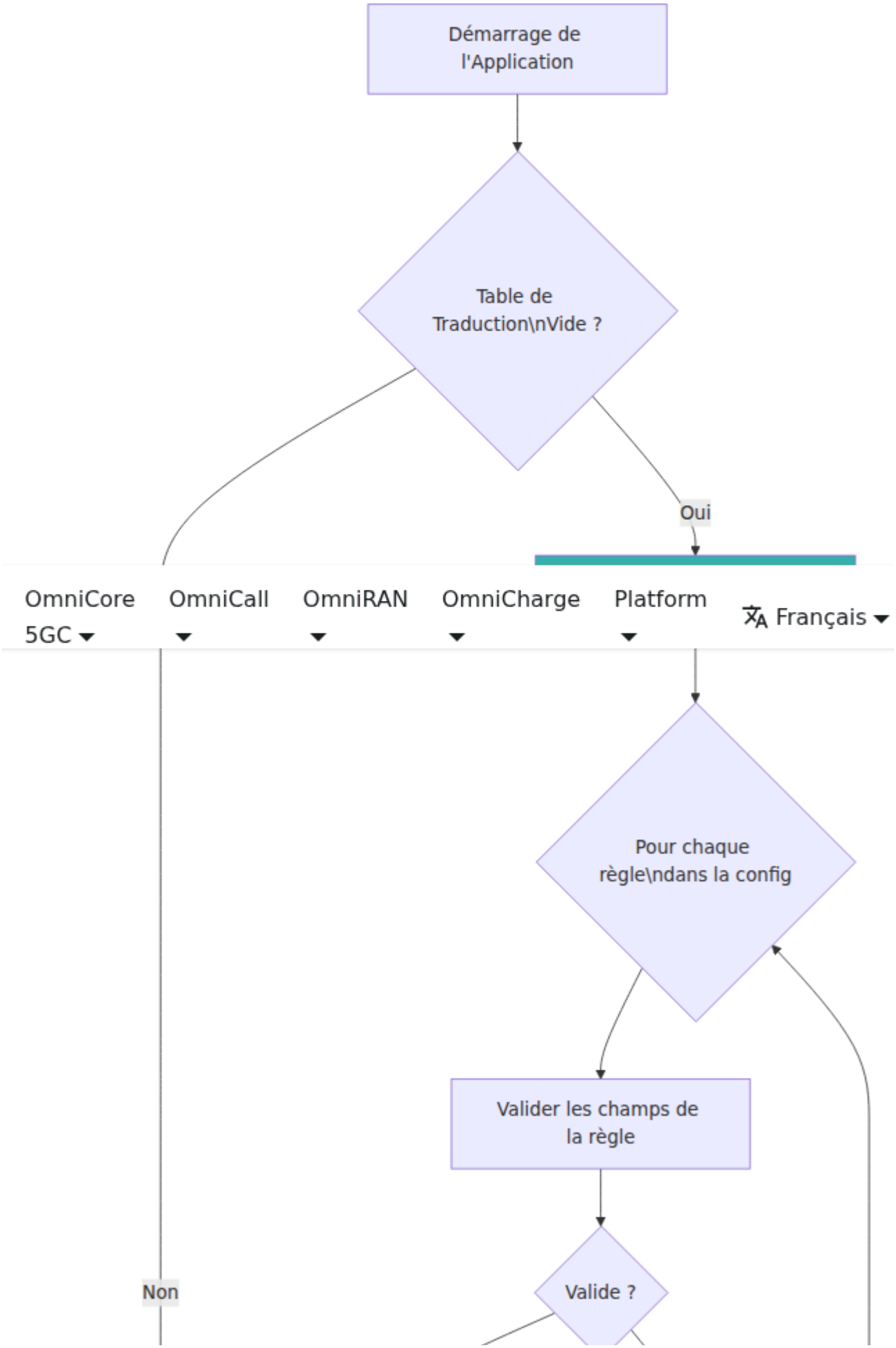
Exemple de Configuration

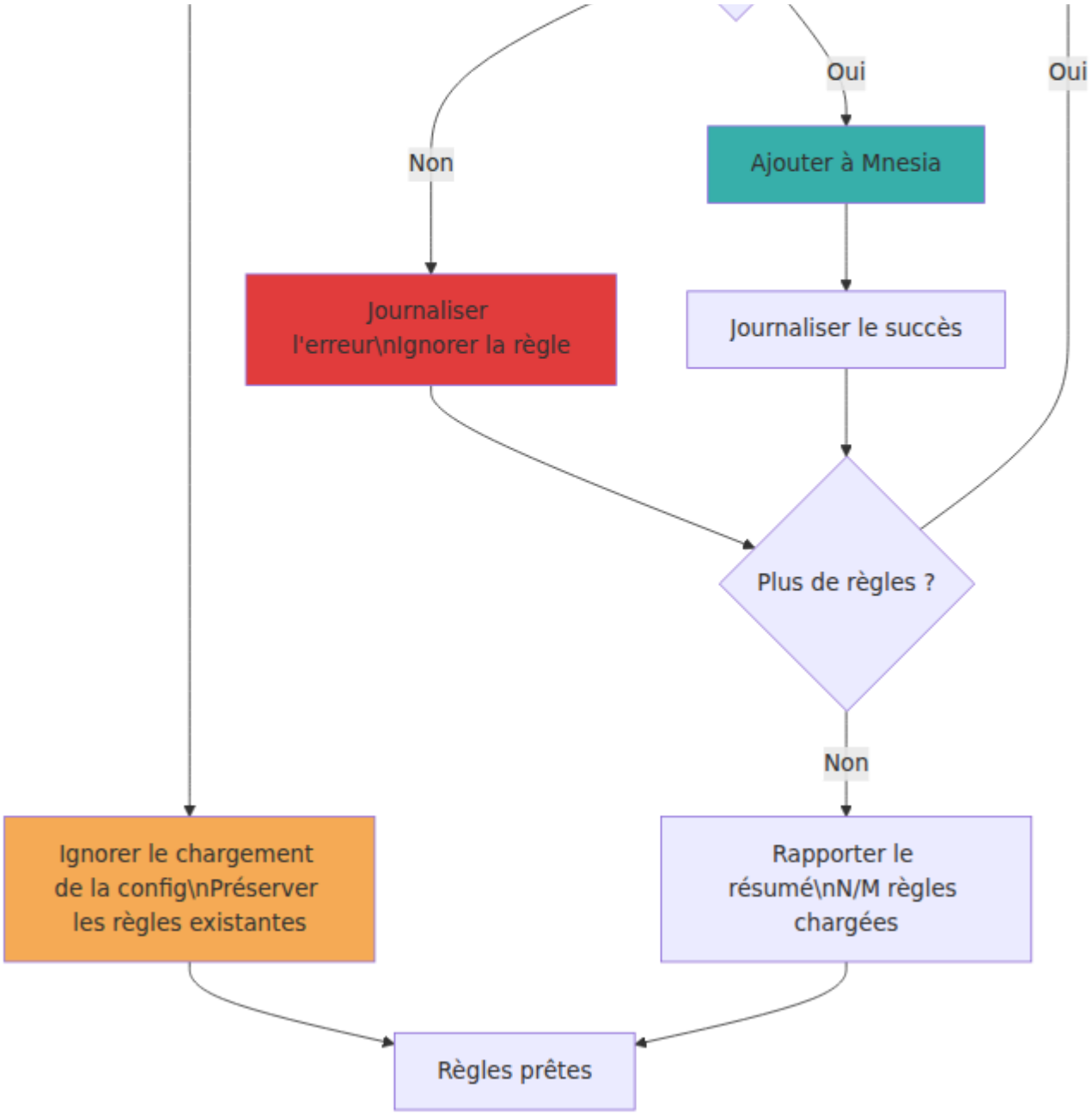
```
# config/runtime.exs
config :sms_c, :translation_rules, [
  # Ajouter +1 aux numéros américains à 10 chiffres
  %{
    calling_prefix: nil,
    called_prefix: nil,
    source_smsc: "us_domestic_smsc",
    calling_match: "^(\d{10})$",
    calling_replace: "+1\1",
    called_match: "^(\d{10})$",
    called_replace: "+1\1",
    priority: 10,
    description: "Ajouter +1 aux numéros américains à 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
  },

  # Formater les numéros britanniques 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 britanniques pour une
passerelle héritée",
    enabled: true,
    continue: false
  }
]
```

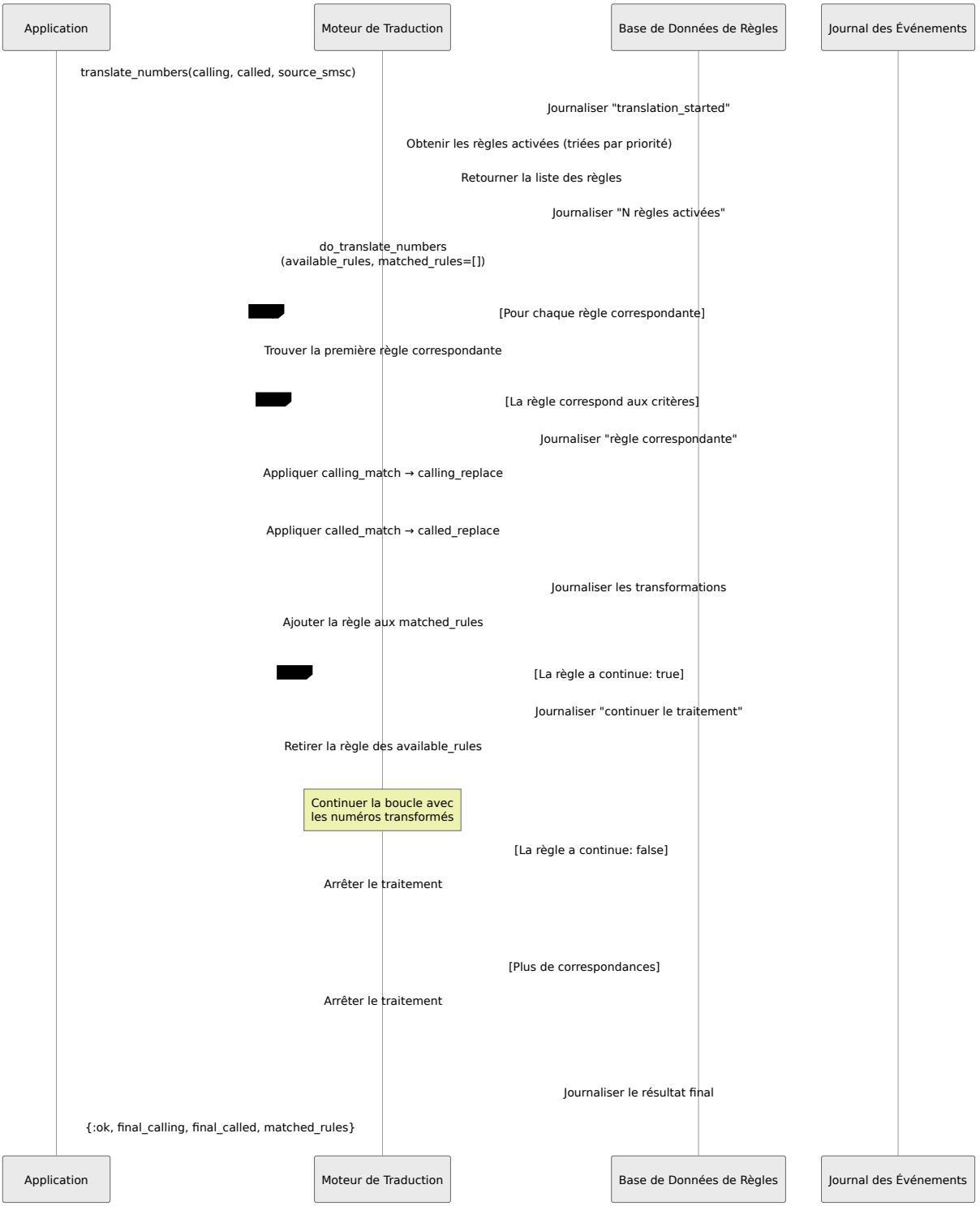
Démarrage

Flux d'Initialisation


Démarrage de l'Application
Table de Traduction\nVide ?
Oui
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Français
Pour chaque règle\ndans la config
Valider les champs de la règle
Valide ?
Non



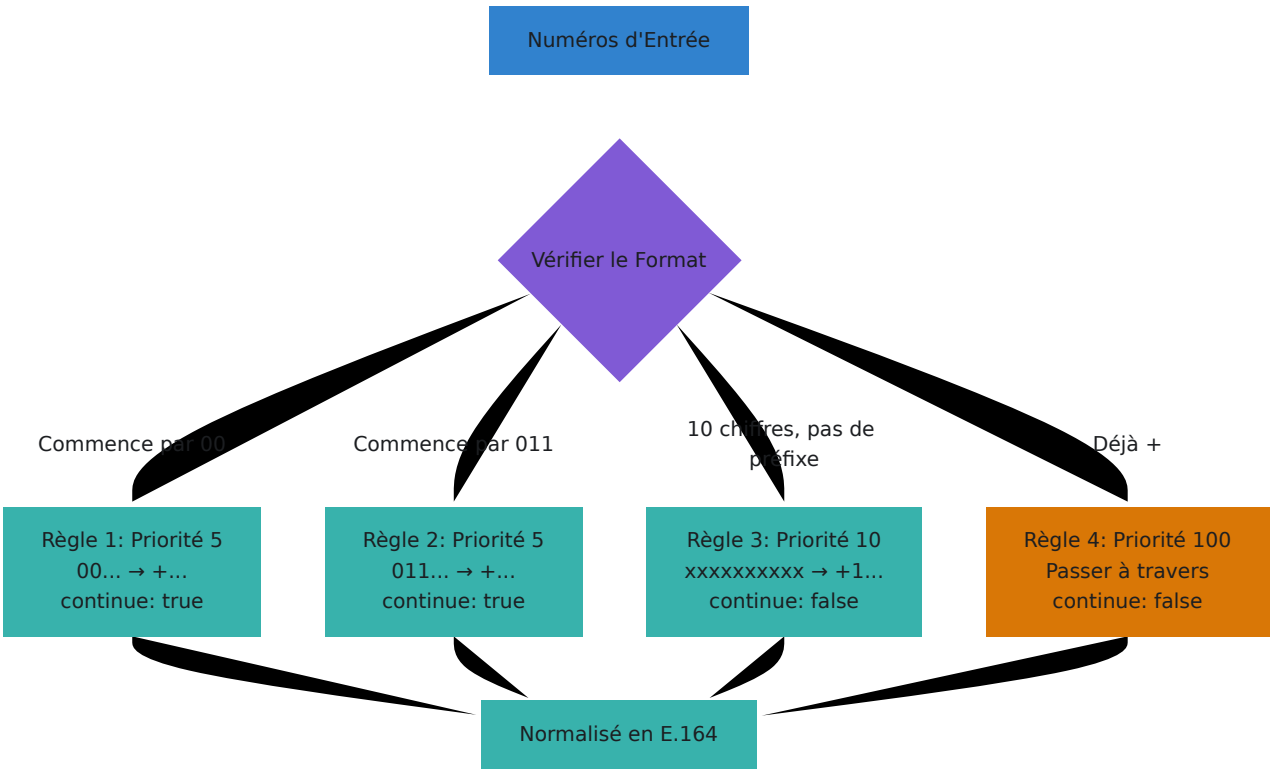
Flux de Traduction des Messages



Cas d'Utilisation Courants

Normalisation des Numéros Internationaux

Normaliser divers formats internationaux en E.164 :



Formatage Spécifique à la Passerelle

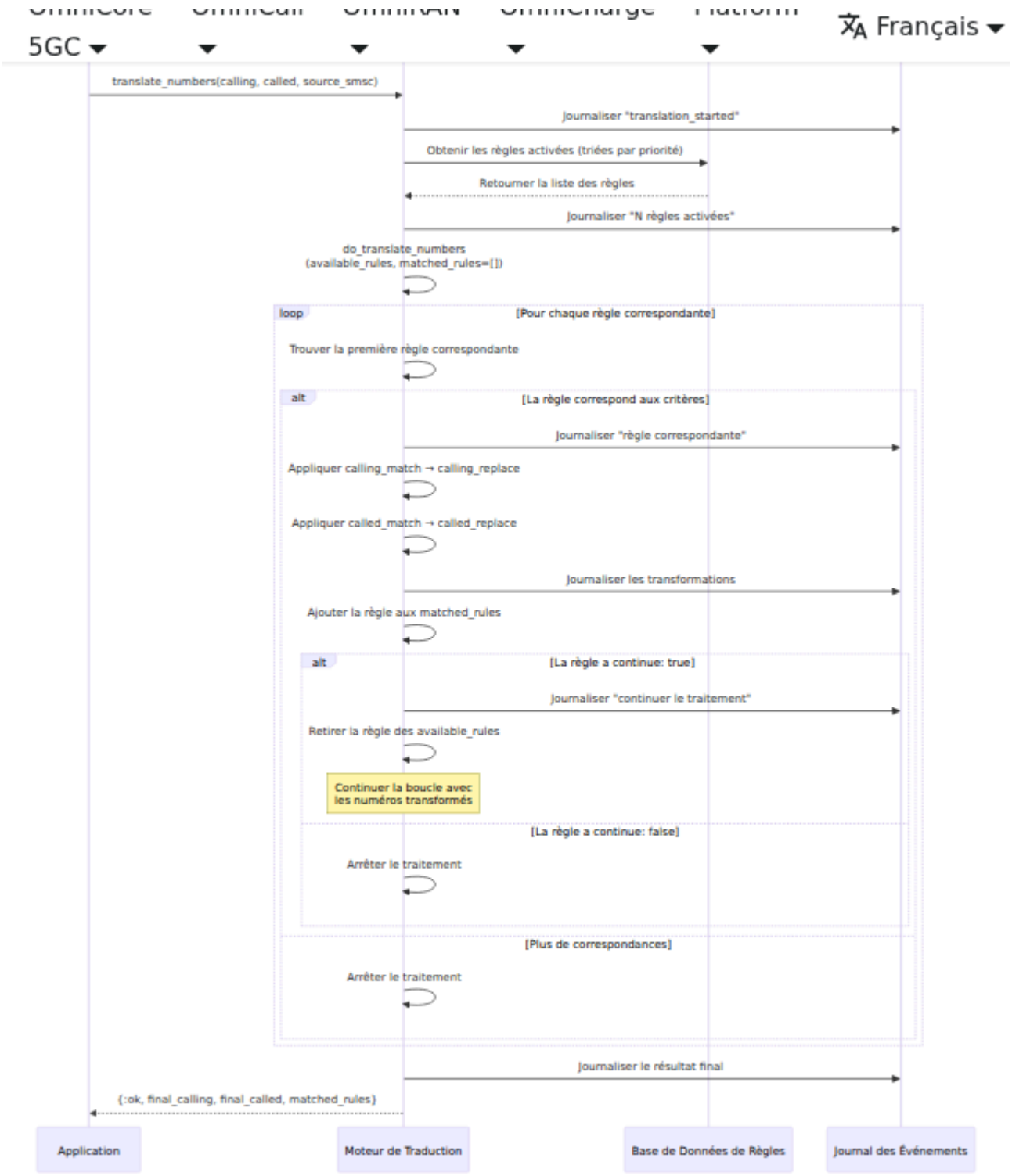
Enchaîner des règles pour formater des numéros selon les exigences spécifiques de la passerelle :

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

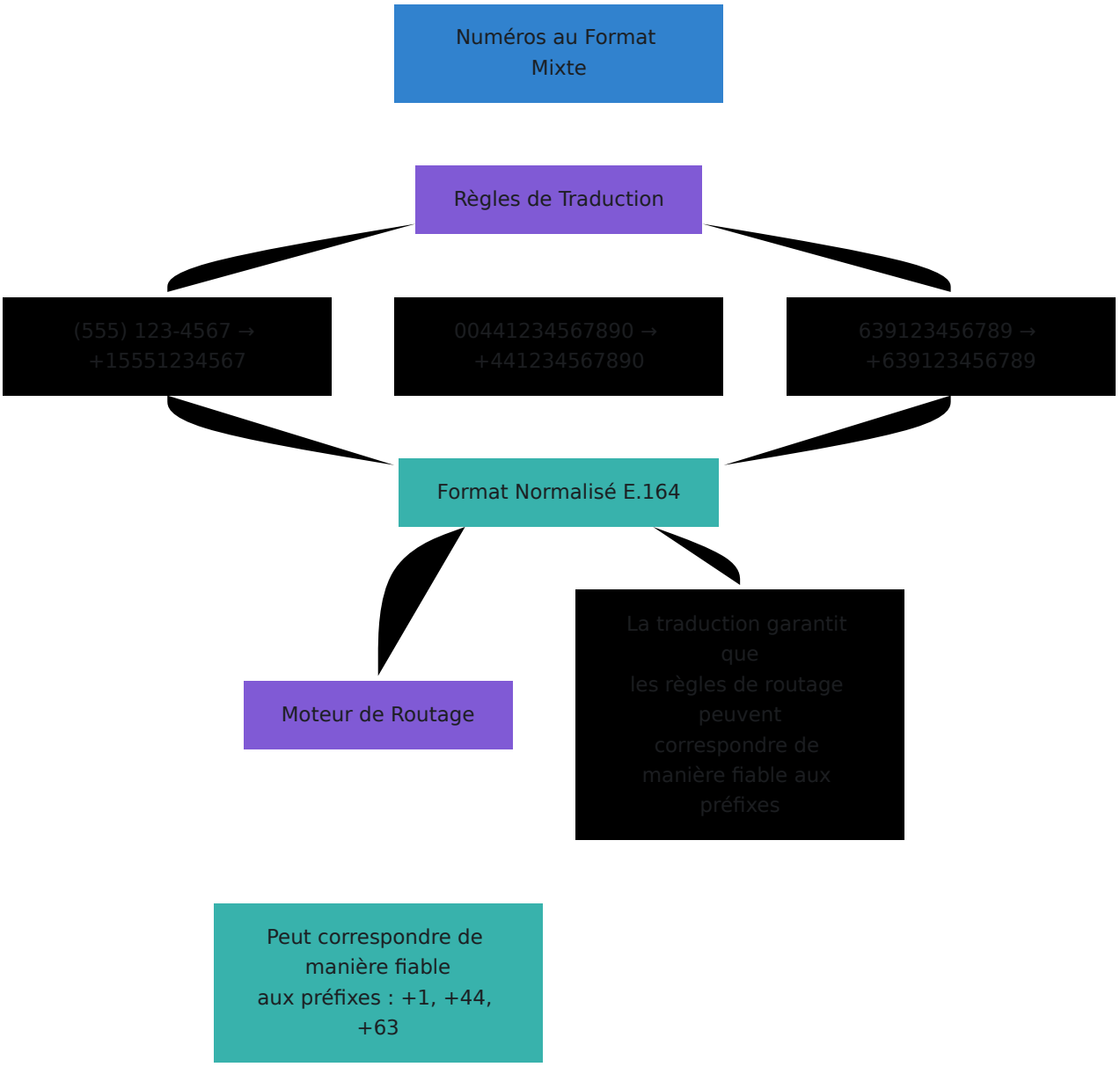
Traductions Spécifiques à la SMSC

Appliquer différentes traductions en fonction de la source du message :



Préparation au Routage Basée sur le Préfixe

Normaliser les numéros avant le routage pour assurer 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 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. Remplir les critères de correspondance (facultatif) :
   - Préfixe appelant (ex. : "+1", "44")
   - Préfixe appelé (ex. : "+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 priorité (1-255, plus bas = plus haute priorité)
4. Définir le statut :
   - **Activé** : La règle est active
   - **Continuer le Traitement** : Continuer à évaluer plus de règles après celle-ci
5. Ajouter une description
6. Cliquer sur "Ajouter une Règle" ou "Mettre à Jour la Règle"

Basculer le Traitement Continu :

- **Arrêter** (par défaut) : Arrêter le traitement après que cette règle corresponde
- **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. Cliquer sur "Modifier" à côté de la règle
2. Modifier les champs selon les besoins
3. Cliquer 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 modèles 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. Entrer les paramètres de test :

- Numéro appelant (de)
- Numéro appelé (à)
- SMSC source (facultatif)

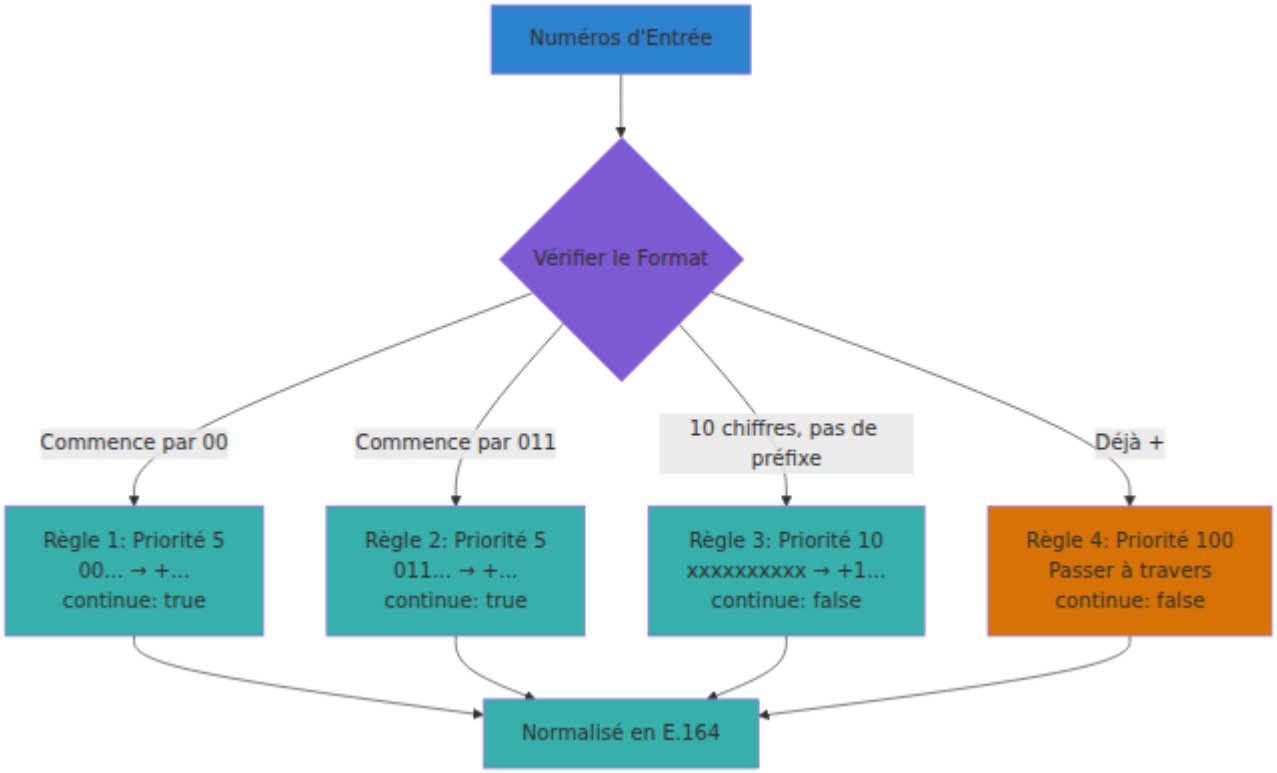
2. Cliquer sur "Tester la Traduction"
3. Voir les résultats complets :
   - **Résultat de Traduction** : Numéros finaux après toutes les transformations
   - **Règles Appliquées** : Compte et liste de toutes les règles qui ont correspondu
   - **Transformations Étape par Étape** : Vue détaillée de chaque règle :
     - Numéro d'étape et informations sur la règle
     - Description de la règle
     - Avant → Après pour les numéros appelant et appelé
     - Indicateur "↓ Continuer" pour les règles qui ont continué le traitement
     - Transformations mises en évidence en vert
     - Valeurs inchangées marquées comme "passées à travers"
4. Charger des exemples pré-configurés en utilisant les boutons d'exemple
5. Revoir l'historique des tests pour comparer différents scénarios

Exemple de Sortie :

```
Résultat de Traduction
═══════════════════════════════════════════════

Numéro Appelant:  5551234567 → +1-555-123-4567
Numéro Appelé:   9078720155 → +1-907-872-0155
✓ Traduit par 3 règle(s)

Transformations Étape par Étape
═══════════════════════════════════════════════

┌─ Étape 1 ──────────────────────────────────────┐
│ Règle #1 (Priorité 10)          ↓ Continuer │
│ Ajouter le code pays aux numéros à 10 chiffres     │
│                                                │
│ Appelé: 9078720155 → +19078720155       │
└────────────────────────────────────────────────┘

┌─ Étape 2 ──────────────────────────────────────┐
│ Règle #2 (Priorité 20)          ↓ Continuer │
│ Formater le code régional avec des tirets            │
│                                                │
│ Appelé: +19078720155 → +1-907-8720155   │
└────────────────────────────────────────────────┘

┌─ Étape 3 ──────────────────────────────────────┐
│ Règle #3 (Priorité 30)                      │
│ Formatage final pour la passerelle             │
│                                                │
│ Appelé: +1-907-8720155 → +1-907-872-0155│
└────────────────────────────────────────────────┘
```

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 de la SMSC source
- `message_id` (facultatif) : Pour la journalisation des événements

Renvoie :

- `{:ok, translated_calling, translated_called, [rules_applied]}` - Toujours réussi
- Renvoie les numéros originaux si aucune règle ne correspond
- Renvoie la liste de toutes les règles qui ont été appliquées (dans l'ordre)

```
# Exemple d'utilisation
{:ok, new_calling, new_called, rules} =
  NumberTranslation.translate_numbers(
    calling_number: "5551234567",
    called_number: "9078720155",
    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 à 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()
# Renvoie : %{
#   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 l'existant)
{:ok, %{imported: 3, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :merge)

# Importer des règles (remplacer tout l'existant)
{:ok, %{imported: 5, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :replace)
```

Meilleures Pratiques

Conception des Règles

1. **Maintenir les priorités organisées** :
   - **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 (ex. : "5551234567 → +15551234567")
   - Documenter le but et les entrées/sorties attendues

4. **Tester les modèles regex** :

   - Tester les modèles 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 par préfixe pour réduire les évaluations regex
   - Supprimer ou désactiver les règles inutilisées

2. **Optimiser les modèles regex** :

   - Utiliser d'abord la correspondance par préfixe (plus rapide que regex)
   - Garder les modèles regex simples
   - Éviter les modèles 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 de télémétrie

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 des 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. **Modèles courants** :
   - Numéro américain à 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 : La 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 modèle 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 des préfixes (sensible à la casse)
4. Tester le modèle regex séparément

5. Vérifier l'ordre des priorités

Mauvaise Transformation Appliquée

Symptôme : Numéro transformé mais le résultat est incorrect

Causes possibles :

- Le modèle regex correspond mais le modèle de remplacement est incorrect
- Plusieurs règles s'appliquent dans un ordre inattendu
- Les 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 modèle regex capture les bons groupes
3. Vérifier la syntaxe du modèle de remplacement
4. Tester le regex dans un testeur regex en ligne
5. Revoir la priorité des règles et les drapeaux continue

Boucle Infinie / Dégradation des Performances

Symptôme : La traduction prend très longtemps 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 les erreurs
2. Revoir les règles avec continue: true
3. Simplifier les modèles 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 des priorités 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 :

- Traduction non intégrée dans le flux de message
- Traduction se produisant après le routage
- Code d'application contournant la traduction

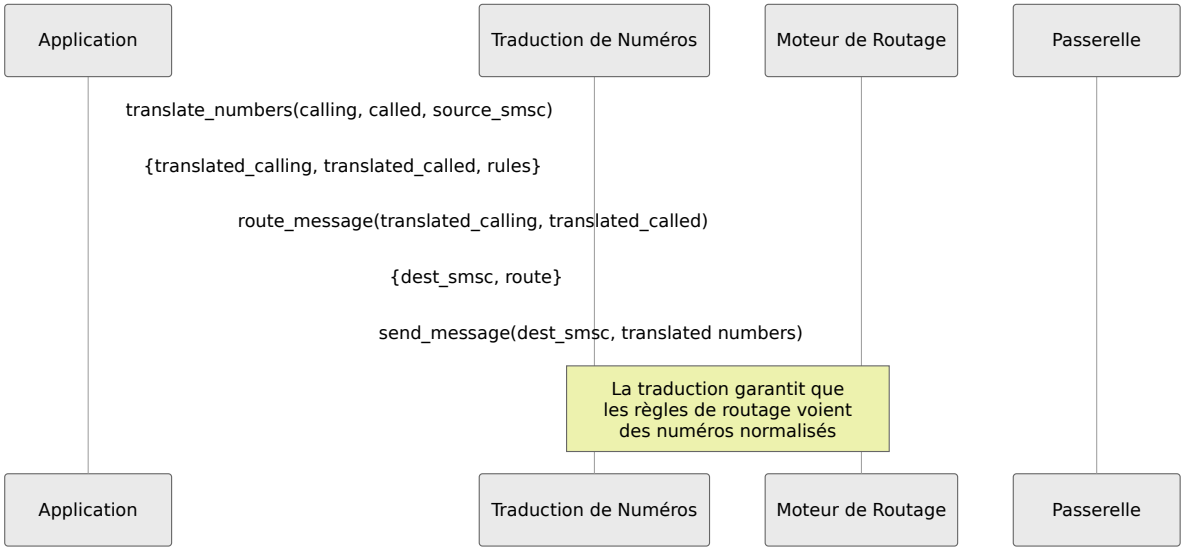
Solutions :

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 de Télémétrie

Surveiller les performances de traduction avec Télémétrie :

```
: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

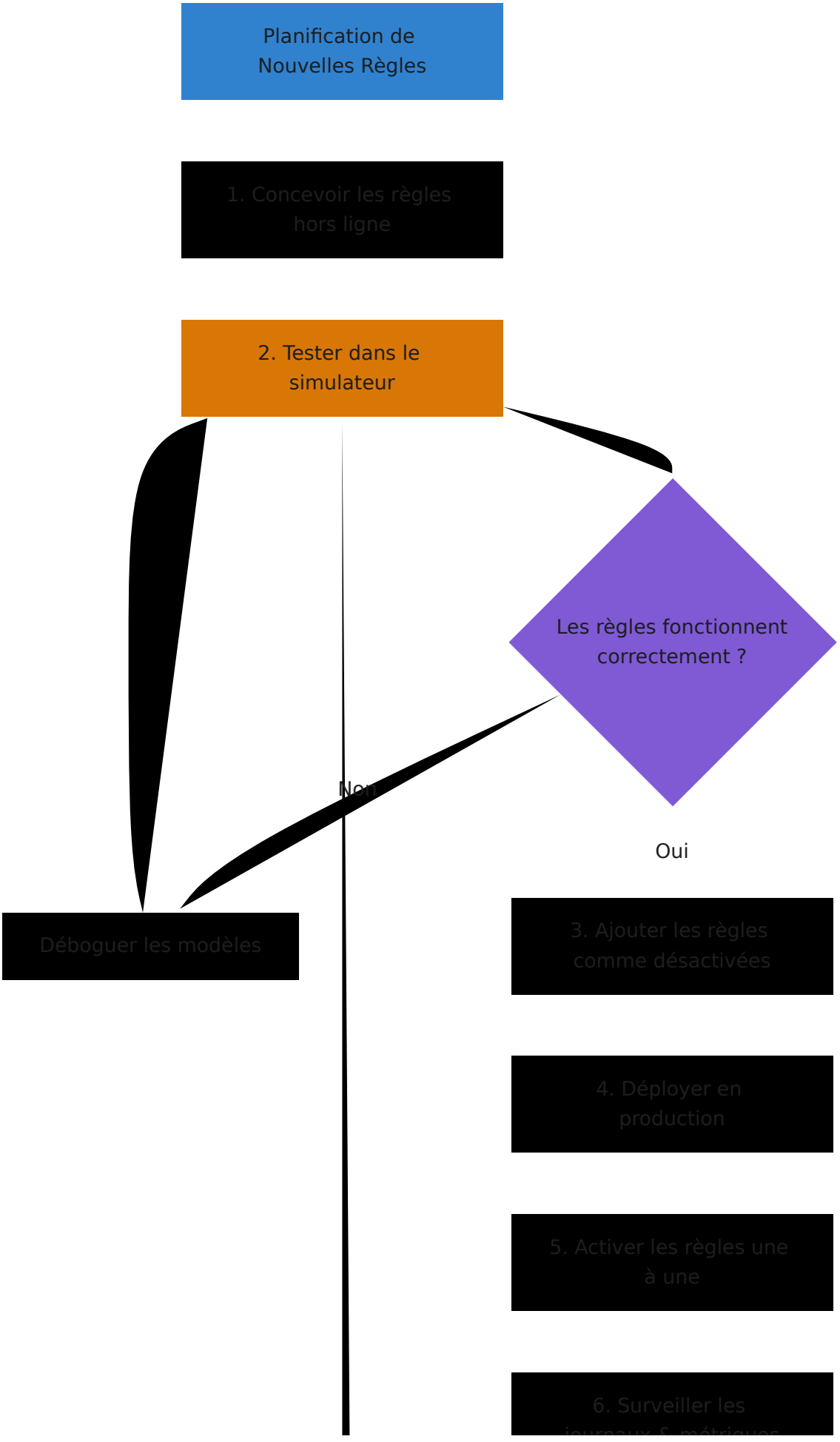
Les tables Mnesia sont automatiquement distribuées à travers les nœuds en cluster. Les règles de traduction sont répliquées pour une haute disponibilité.

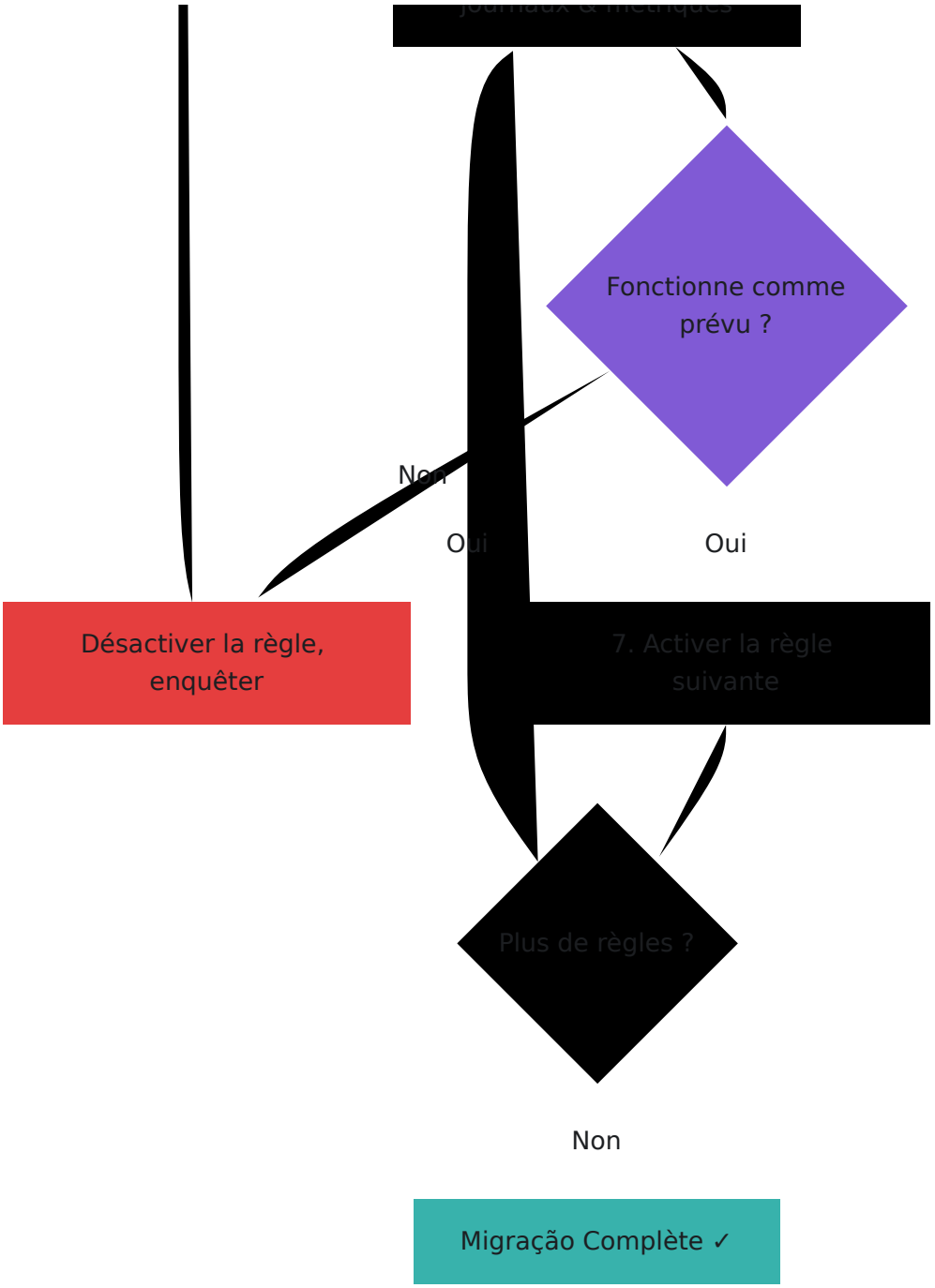
Parse error on line 25: ... style New fill:#3182CE style P ----------^ Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW', 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW', 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS', 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

Réessayer

Stratégies de Migration

Lors du déploiement de nouvelles règles de traduction :


Planification de Nouvelles Règles
1. Concevoir les règles hors ligne
2. Tester dans le simulateur
Les règles fonctionnent correctement ?
Non
Oui
Déboguer les modèles
3. Ajouter les règles comme désactivées
4. Déployer en production
5. Activer les règles une à une
6. Surveiller les



Exemples

Exemple 1 : Normalisation des Numéros Américains

Exigence : Convertir divers formats de numéros américains en E.164 (+1XXXXXXXXXX)

```
# Règle 1 : numéros à 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 à 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, continue)
# 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 à 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 à 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, continue)
# Étape 2 : "5551234567" → "+15551234567" (Règle 2, continue)
# 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 :

- Vérifiez la suite de tests à `test/sms_c/messaging/number_translation_test.exs` pour des exemples
- Utilisez le simulateur pour déboguer la logique de traduction
- Consultez 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 de télémétrie pour des problèmes de performance

Guide des opérations SMS-C

← Retour à l'index de la documentation | **README principal**

Procédures opérationnelles quotidiennes, surveillance et tâches de maintenance pour les équipes d'opérations SMS-C.

Table des matières

Opérations quotidiennes

Vérification de la 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 des messages (derières 24 heures)
- Taux d'échec de routage (doit être < 1%)
- Arriéré de la file d'attente (doit être < 1000 en attente)
- Taux de réussite 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 de nombreuses tentatives de livraison (enquêter si > 3)
- Messages en 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
- Aucune déconnexion non expirée
- Aucun erreur 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 horaires des messages :

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 qui n'ont pas pu être routés
- **Alerte** : Tout échec (> 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 non correspondants
- **Requête** : `sms_c_routing_route_matched_count`

Métriques de livraison

Taux de réussite 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** : Réessais nécessaires pour la livraison
- **Alerte** : p95 > 2 (trop de réessais)
- **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 en attente le plus ancien
- **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 des 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 réussite 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"

# La file d'attente s'accumule - le traitement prend du 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 réussite 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 réussite de livraison est {{ $value }}"

# Trop de réessais de livraison
- alert: HighRetryRate
  expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count) > 2
  severity: warning
  description: "95e percentile des tentatives de livraison : {{
$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 de 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. Voir tous les détails 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. Voir tous les messages pour ce numéro

Suivre le cycle de vie d'un message

Voir l'historique des événements :

1. **Interface Web** : Cliquez sur le message dans la file d'attente, voir 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

Messages en attente :

- État : "en attente"
- deliver_after : Horodatage futur
- delivery_attempts : 0 ou faible nombre

Messages livrés :

- État : "livré"
- deliver_time : Horodatage de livraison
- dest_smsc : Frontend qui a livré

Messages échoués :

- État : "en attente" avec de nombreuses delivery_attempts
- deadletter : true (si expiré)
- Vérifiez le journal des événements pour les raisons d'échec

Routage des 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 ça fonctionne :

Lorsqu'un frontend interroge pour des messages en attente en utilisant `get_messages_for_smsc(smsc_name)`, le système renvoie les 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 interrogera :

```
# 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 🔍 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 (obligatoire pour le routage géographique)
   - **SMSC source** : Filtre du système source (facultatif)
   - **SMSC de destination** : Passerelle de destination (obligatoire 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 sauvegarde (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`
- Uni que 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 est indisponible, les messages sont abandonnés (échec fermé).

Tester les itinéraires

Simulateur de routage :

1. Accédez à `/simulator`
2. Entrez les paramètres de test :
   - Numéro appelant : `+15551234567`
   - Numéro appelé : `+447700900000`
   - SMSC source : (facultatif)
   - Type source : (facultatif)
3. Cliquez sur "Simuler le routage"
4. Examinez les résultats :
   - **Itinéraire sélectionné** : Quel itinéraire a été choisi
   - **Toutes les correspondances** : Quels itinéraires ont correspondu aux critères
   - **Évaluation** : Pourquoi chaque itinéraire a correspondu ou n'a pas correspondu

Tester avant la production :

- Testez tous les nouveaux itinéraires dans le simulateur
- Vérifiez que le bon itinéraire 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 de l'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)
- Aucun frontend expiré inattendu

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 fonctionne
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 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 modifications 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 :

Modifiez `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 de 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 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 selon votre politique de conservation :

- Configurez l'archivage automatique en fonction des exigences commerciales (typiquement 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 les performances :

- **MySQL/MariaDB** : Exécutez la commande `OPTIMIZE TABLE` pendant les périodes de faible trafic
- **PostgreSQL** : Exécutez `VACUUM ANALYZE` régulièrement (ou activez l'autovacuum)

Exécutez chaque semaine pendant une période de faible trafic pour maintenir des performances optimales.

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 sms_c || true
    endscript
}
```

Redémarrer l'application

Redémarrage gracieux (zéro temps d'arrêt dans le cluster) :

```
# Redémarrez un nœud à la fois
systemctl restart sms_c

# Attendez que le nœud rejoigne le cluster
# Répétez pour chaque nœud
```

Redémarrage d'urgence (tous les nœuds) :

```
systemctl restart sms_c
```

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) :
   - Exportez via l'interface Web
   - Ou commande de sauvegarde Mnesia
3. **Base de données SQL CDR** :
   - Sauvegarde complète quotidienne
   - Sauvegardes des journaux de transactions (continues)
4. **Certificats TLS** :
   - `priv/cert/*.crt`
   - `priv/cert/*.key`

Procédures de sauvegarde

Sauvegarde quotidienne de la configuration :

```
#!/bin/bash
# /opt/sms_c/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/sms_c/config $BACKUP_DIR/

# Sauvegarde des certificats
cp -r /opt/sms_c/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/sms_c/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 sûres pour la cohérence
# - Compressez la sortie pour économiser de l'espace
# - Configurez une 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 de la table de routage :

```
#!/bin/bash
# /opt/sms_c/scripts/backup_routes.sh

BACKUP_DIR="/var/backups/sms_c/routes"
DATE=$(date +%Y%m%d)

mkdir -p $BACKUP_DIR

# Exporter via l'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/sms_c/scripts/backup_config.sh
0 2 * * * /opt/sms_c/scripts/backup_database.sh
0 2 * * * /opt/sms_c/scripts/backup_routes.sh
```

Procédures de récupération

Restaurer la configuration :

```
# Arrêter l'application
systemctl stop sms_c

# Restaurer les fichiers de configuration
cp -r /var/backups/sms_c/20251030/config/* /opt/sms_c/config/

# Restaurer les certificats
cp -r /var/backups/sms_c/20251030/cert/* /opt/sms_c/priv/cert/

# Démarrer l'application
systemctl start sms_c
```

Restaurer la base de données SQL CDR :

Utilisez les outils de restauration appropriés pour votre base de données :

- **MySQL/MariaDB** : Décompressez et redirigez vers le client mysql
- **PostgreSQL** : Utilisez pg_restore avec des sauvegardes 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 des 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 mensuelle des données
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
- Partitionnement des 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 réussite 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 réussite de livraison 80-95%
- Traitement des messages lent
- Recherches ENUM échouées

Faible (Réponse dans les 24 heures) :

- Légère dégradation 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
- Revenez aux déploiements récents si suspectés comme cause
- Activez les journaux détaillés si nécessaire
- Informez les parties prenantes

4. Enquête :

- Examinez les journaux d'application
- Vérifiez l'utilisation des ressources système
- Examinez les performances 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
- Implémentez des mesures préventives
- Mettez à jour les manuels d'exploitation

Incidents courants

Arriéré de file d'attente élevé :

1. Vérifiez le taux de réussite de livraison
2. Vérifiez que les frontends sont connectés et interrogent
3. Vérifiez les performances 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 qu'un itinéraire de secours existe
5. Vérifiez les journaux d'événements pour les raisons d'échec

Déconnexions de frontend :

1. Vérifiez l'état du système 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 les performances des requêtes de base de données
2. Examinez la configuration des travailleurs de lot
3. Vérifiez les ressources adéquates (CPU/Mémoire)
4. Vérifiez les retards de recherche ENUM
5. Examinez les performances du système de facturation

Pour des procédures de dépannage détaillées, consultez le Guide de dépannage.

Guide d'Optimisation des Performances

← Retour à l'Index de Documentation | **README Principal**

Ce guide explique comment optimiser les performances de SMS-C pour différents scénarios de charge de travail.

Aperçu des Performances

SMS-C offre un débit de **1 750 messages/seconde** en utilisant Mnesia pour le stockage de messages en mémoire avec archivage automatique de la base de données SQL pour la conservation des CDR.

Principaux Indicateurs de Performance

Mesuré sur Intel i7-8650U @ 1,90 GHz (8 cœurs) :

Opération	Débit	Latence (moyenne)	Amélioration
Insertion de Message (avec routage)	1 750 msg/sec	0,58 ms	21x plus rapide que SQL
Insertion de Message (simple)	1 750 msg/sec	0,57 ms	21x plus rapide que SQL
Récupérer des Messages pour SMSC	800 msg/sec	1,25 ms	Requête en mémoire
Mémoire par Insertion	62 Ko	-	Réduction de 50 %

Capacité : ~150 millions de messages par jour sur un nœud unique

Table des Matières

Architecture de Stockage des Messages

SMS-C utilise une architecture de stockage dual pour des performances optimales :

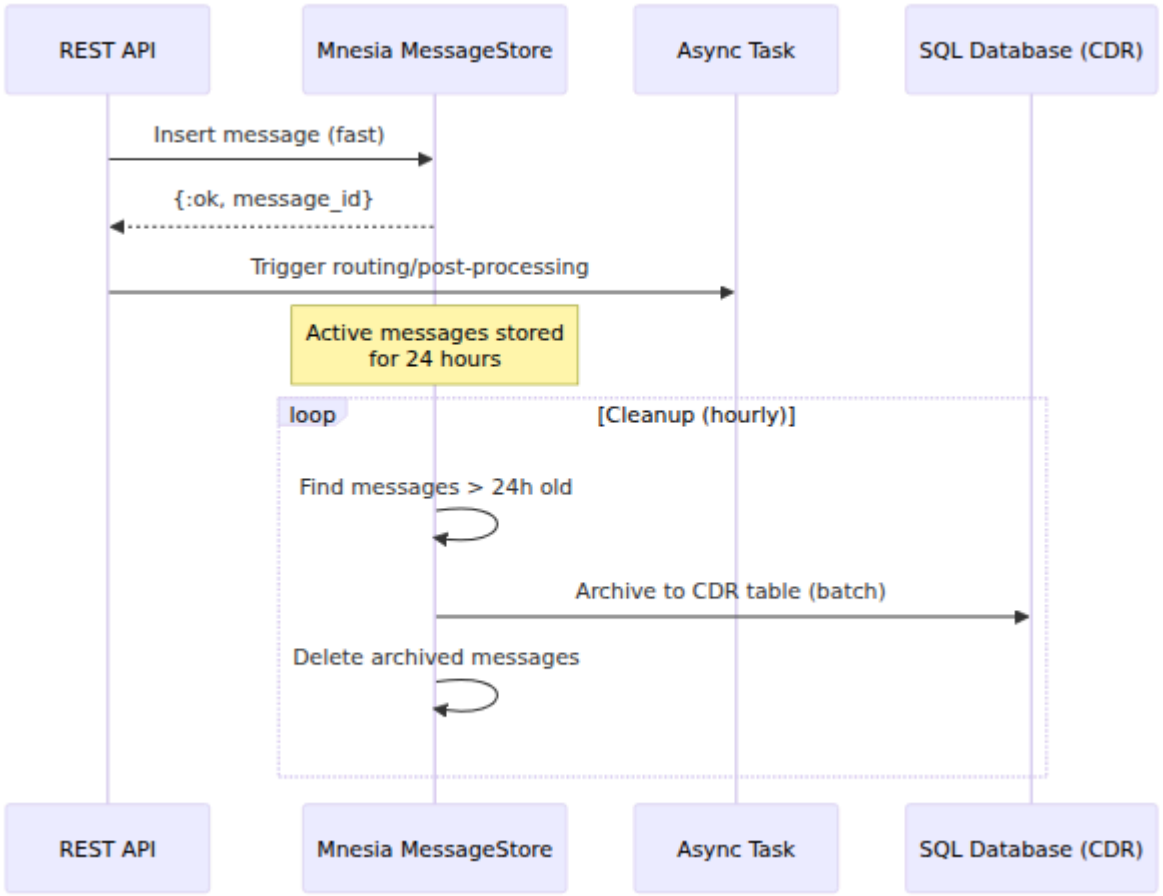
Magasin de Messages Actif (Mnesia)

- **Objectif** : Insertion, routage et livraison de messages ultra-rapides
- **Stockage** : En mémoire avec persistance sur disque (disc_copies)
- **Performance** : Débit d'insertion de 1 750 msg/sec, latence de 0,58 ms
- **Conservation** : Configurable (par défaut : 24 heures)
- **Clustering** : Prend en charge Mnesia distribué pour l'évolutivité horizontale

Archive CDR (Base de Données SQL)

- **Objectif** : Historique des messages à long terme et reporting
- **Stockage** : Base de données SQL (MySQL/MariaDB ou PostgreSQL) pour archivage durable
- **Performance** : Écritures par lots pour minimiser la charge de la base de données
- **Conservation** : Permanente (ou selon la politique de conservation des données)
- **Requêtes** : Analytique, reporting, conformité

Flux de Données



Optimisation de Mnesia

Configuration de la Conservation des Messages

```
# config/runtime.exs
config :sms_c,
  message_retention_hours: 24  # Par défaut : 24 heures
```

Directives d'Optimisation :

- **Volume élevé (>1M msg/jour)** : 12-24 heures de conservation
    - Minimiser la taille de la table Mnesia
    - Requêtes plus rapides

    - Archivage plus fréquent vers MySQL
- **Volume moyen (100K-1M msg/jour)** : 24-48 heures de conservation
    - Bon équilibre pour la plupart des déploiements
    - Tampon adéquat pour la logique de réessai
- **Faible volume (<100K msg/jour)** : 48-168 heures de conservation
    - Historique des messages plus long dans un stockage rapide
    - Archivage moins fréquent

Indices de Table Mnesia

MessageStore crée automatiquement des indices sur :

- `status` - Pour filtrer les messages en attente/livrés
- `dest_smsc` - Pour des requêtes spécifiques à SMSC
- `expires` - Pour la gestion des expirations
- `destination_msisdn` - Pour les requêtes d'abonnés
- `source_msisdn` - Pour les requêtes d'abonnés

Persistance Disque Mnesia

Les messages sont stockés sous forme de `disc_copies` offrant :

- □ Performance en mémoire
- □ Persistance automatique sur disque
- □ Récupération après crash
- □ Pas de perte de données au redémarrage

Configuration de l'Archivage des CDR

Le `BatchInsertWorker` gère l'archivage des CDR vers MySQL en utilisant des écritures par lots :

```
# config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,           # Taille du lot CDR
  batch_insert_flush_interval_ms: 100     # Intervalle d'auto-
flush
```

Directives d'Optimisation des CDR

Archivage à Volume Élevé

```
batch_insert_batch_size: 200
batch_insert_flush_interval_ms: 200
```

- Des lots plus importants réduisent la charge sur MySQL
- Latence plus élevée pour les écritures CDR (acceptable pour l'archivage)

Équilibré (Recommandé)

```
batch_insert_batch_size: 100
batch_insert_flush_interval_ms: 100
```

- Bon équilibre pour la plupart des déploiements
- CDR écrits dans les 100 ms

Exigences CDR en Temps Réel

```
batch_insert_batch_size: 20
batch_insert_flush_interval_ms: 20
```

- Écritures CDR plus rapides pour la conformité
- Plus d'opérations d'écriture MySQL

Optimisation des Requêtes

Utilisation Efficace des Indices Mnesia

Les requêtes qui utilisent des champs indexés sont les plus rapides :

```
# Requêtes rapides (utiliser des indices)
MessageStore.list(status: :pending)
MessageStore.list(dest_smsc: "gateway-1")
Messaging.get_messages_for_smsc("gateway-1")

# Requêtes plus lentes (scan complet de la table)
MessageStore.list(limit: :infinity)  # Renvoie tous les messages
```

Pool de Connexion MySQL

Pour les requêtes CDR et l'archivage, configurez le pool de connexion MySQL :

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  pool_size: 10  # Augmenter pour un reporting CDR lourd
```

Directives :

- Déploiement standard : `pool_size: 10`
- Reporting CDR lourd : `pool_size: 20-30`
- Archivage uniquement : `pool_size: 5`

Benchmarking

Exécution des Benchmarks

Le projet inclut des benchmarks basés sur Benchee pour les tests de performance :

```
# Benchmark API SMS brut (compare sync vs async)
mix run benchmarks/raw_sms_bench.exs

# Benchmark API de message général
mix run benchmarks/message_api_bench.exs
```

Interprétation des Résultats

Exemple de sortie :

```
Name                                          ips        average
deviation         median         99th %
submit_message_raw_async (batch)        4.65 K        0.22 ms
±41.72%       0.184 ms        0.55 ms
submit_message_raw (sync)             0.0696 K       14.36 ms
±33.42%       12.57 ms       33.71 ms
```

Indicateurs clés :

- **ips** : Itérations par seconde (plus c'est élevé, mieux c'est)
- **average** : Temps d'exécution moyen (plus c'est bas, mieux c'est)
- **median** : Valeur médiane, plus représentative que la moyenne pour des distributions asymétriques
- **99th %** : Latence au 99e percentile (important pour la conformité SLA)

Base de Performance

Performance attendue sur du matériel moderne (Intel i7-8650U, 8 cœurs) :

Indicateur	insert_message (Mnesia)	Précédent (MySQL)
Débit (avec routage)	1 750 msg/sec	83 msg/sec
Débit (simple)	1 750 msg/sec	89 msg/sec
Temps de Réponse (moyenne)	0,58 ms	16 ms
Temps de Réponse (p99)	<5 ms	30 ms
Mémoire par opération	62 Ko	121 Ko
Gain de Performance	**21x plus rapide**	-

Améliorations Clés :

- □ Suppression des appels de traduction de numéro en double
- □ Post-traitement asynchrone (routage, facturation, événements)
- □ Stockage en mémoire Mnesia vs I/O disque MySQL
- □ Réduction de 50 % de la mémoire

Surveillance

Statistiques d'Exécution

Vérifiez les statistiques du travailleur par lots :

```
SmsC.Messaging.BatchInsertWorker.stats()
```

Renvoie :

```
%{
  total_enqueued: 10000,
  total_flushed: 9900,
  total_batches: 99,
  current_queue_size: 100,
  flush_errors: 0,
  last_flush_at: ~U[2025-10-22 12:34:56Z],
  last_flush_count: 100,
  last_flush_duration_ms: 45
}
```

Indicateurs Clés à Surveiller

1. **Taille de la File d'Attente** : `current_queue_size` - Doit être en dessous de `batch_size` la plupart du temps
2. **Durée de Flush** : `last_flush_duration_ms` - Doit être < 100 ms pour batch_size=100
3. **Erreurs de Flush** : `flush_errors` - Doit être 0 ou très faible
4. **Débit** : `total_flushed / uptime` - Doit correspondre à la charge attendue

Alertes

Mettez en place des alertes de surveillance pour :

- Taille de la file d'attente constamment au maximum (indique une pression de retour)
- Durée de flush augmentant (dégradation des performances de la base de données)
- Erreurs de flush > 0 (problèmes de connectivité à la base de données)
- Débit en dessous des attentes (dégradation des performances)

Dépannage

Symptôme : Faible Débit

Causes possibles :

1. Pool de connexion à la base de données épuisé : Augmentez `pool_size`
2. Base de données lente : Vérifiez la performance des requêtes, ajoutez des index
3. Latence réseau : Optimisez le chemin réseau vers la base de données
4. Taille de lot trop petite : Augmentez `batch_insert_batch_size`

Symptôme : Haute Latence

Causes possibles :

1. Intervalle de flush trop élevé : Réduisez `batch_insert_flush_interval_ms`
2. Taille de lot trop élevée : Réduisez `batch_insert_batch_size`
3. Écritures lentes de la base de données : Vérifiez l'I/O disque, optimisez les tables
4. Utilisation de l'API asynchrone alors que vous avez besoin de synchronisation : Passez à l'endpoint synchrone

Symptôme : Problèmes de Mémoire

Causes possibles :

1. File d'attente en attente : Messages s'accumulant plus rapidement que le flush
2. Taille de lot trop grande : Réduisez `batch_insert_batch_size`
3. Échecs de flush : Vérifiez `flush_errors` dans les statistiques
4. Besoin de redémarrer le travailleur : `Supervisor.terminate_child/2` et redémarrer

Meilleures Pratiques

1. **Commencez avec les valeurs par défaut** (100/100ms) et ajustez en fonction du comportement observé
2. **Surveillez en production** pendant au moins 1 semaine avant d'optimiser
3. **Testez les changements de configuration** en staging avec une charge similaire à la production
4. **Utilisez des benchmarks** pour valider les changements de configuration
5. **Documentez vos décisions d'optimisation** pour référence future
6. **Mettez en place des alertes** avant d'optimiser pour détecter les régressions
7. **Considérez les fuseaux horaires** - la charge de pointe varie selon la région

Exemples de Configurations

Configuration : Agrégateur à Volume Élevé

```
# config/prod.exs
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

config :sms_c, SmsC.Repo,
  pool_size: 50
```

Configuration : Messagerie en Temps Réel pour Entreprises

```
# config/prod.exs
config :sms_c,
  batch_insert_batch_size: 20,
  batch_insert_flush_interval_ms: 10

config :sms_c, SmsC.Repo,
  pool_size: 20
```

Configuration : Développement/Test

```
# config/dev.exs
config :sms_c,
  batch_insert_batch_size: 10,
  batch_insert_flush_interval_ms: 50

config :sms_c, SmsC.Repo,
  pool_size: 5
```

Lectures Complémentaires

- Guide de Performance Ecto
- Documentation Benchee
- Phoenix Sous Pression

Routage SMS

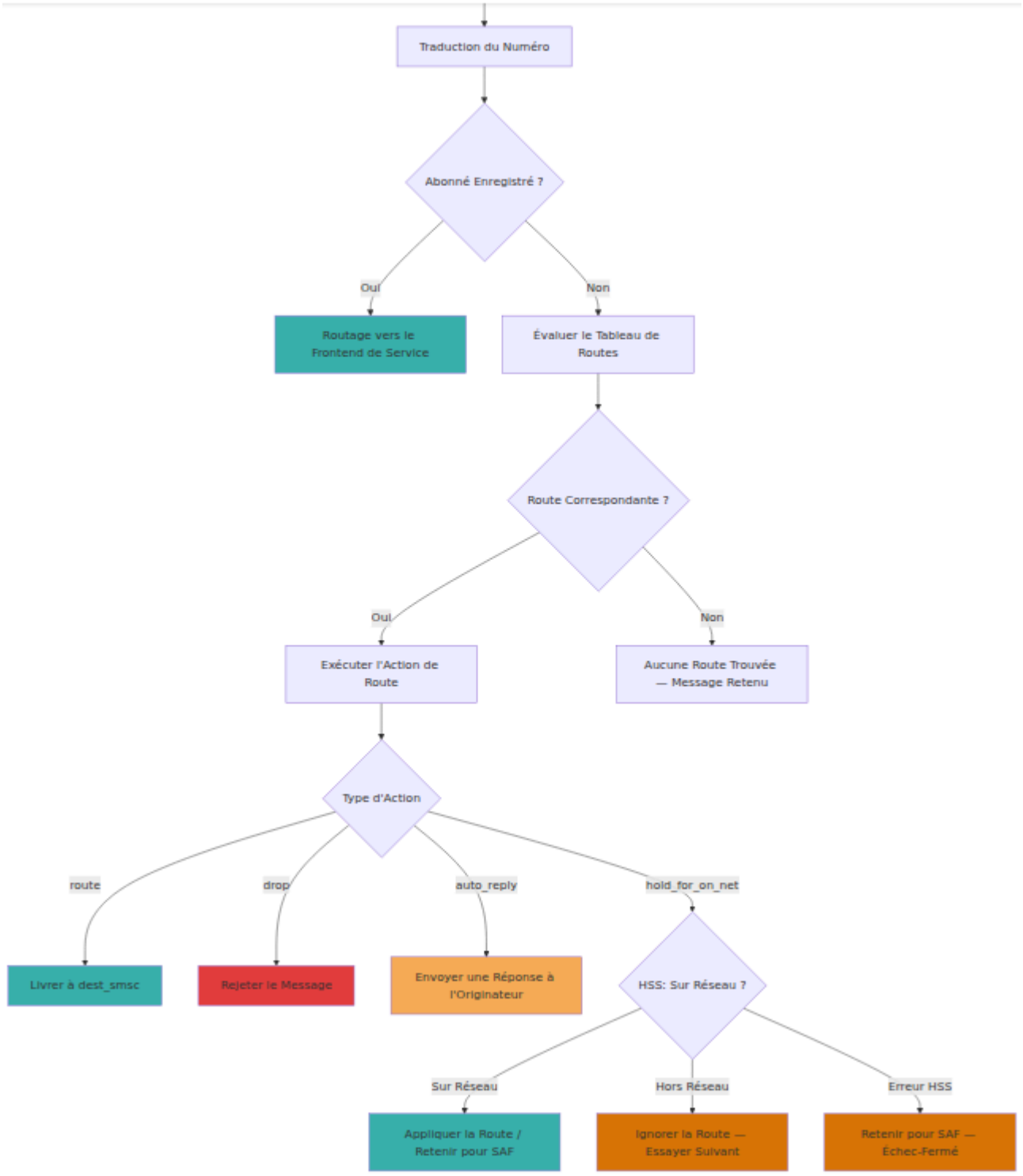
← Retour à l'Index de Documentation

Vue d'ensemble

Le moteur de routage SMS-C détermine où chaque message va après être entré dans le système. Les routes sont stockées dans Mnesia, peuvent être modifiées à l'exécution via l'API ou le panneau de contrôle, et prennent en charge la correspondance regex, l'ordre de priorité, l'équilibrage de charge pondéré, et les décisions basées sur HSS pour les connexions sur réseau/hors réseau.

Pipeline de Traitement des Messages

Chaque message suit ce pipeline depuis la soumission jusqu'au routage :



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 du tableau 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.
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 généré automatiquement
`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 l'SMSC source (`nil` = joker)
`dest_smsc`	Chaîne	Destination SMSC. Requis pour l'action `route`, optionnel pour `hold_for_on_net`
`source_type`	Chaîne	Filtrer par type source : `ims`, `circuit_switched`, `smpp`, ou `nil`
`action`	Chaîne	Action de route : `route`, `drop`, `auto_reply`, `hold_for_on_net`
`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)

Champ	Type	Description
`description`	Chaîne	Étiquette lisible par l'homme
`enabled`	Booléen	Bascule actif/inactif
`originating_on_net_only`	Booléen	Rejeter si le MSISDN source est hors réseau selon HSS
`terminating_on_net_only`	Booléen	Rejeter si le MSISDN de destination est hors réseau selon HSS

Actions de Route

`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 supporté.
Veuillez composer le 911.",
  "priority": 1,
  "description": "Réponse automatique 911"
}
```

`hold_for_on_net`

Vérifie le MSISDN de destination contre le HSS via Diameter Sh. Le comportement dépend du résultat :

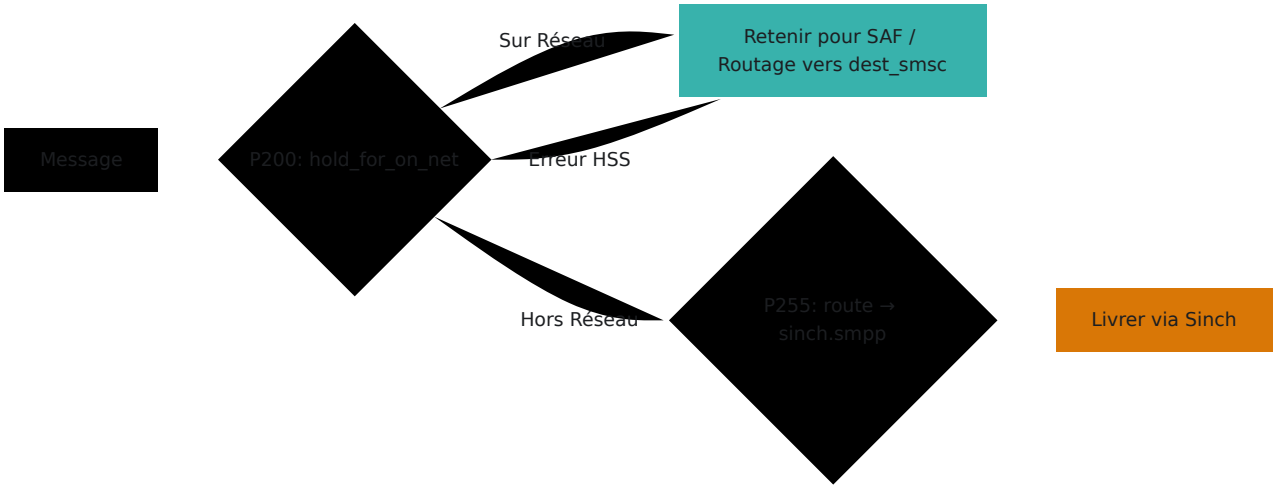
- **Sur réseau** : La route s'applique. Si `dest_smsc` est défini, livrer là. Si `dest_smsc` est `null`, retenir le message pour une livraison store-and-forward (SAF) lorsque l'abonné s'inscrit.
- **Hors réseau** : La route ne s'applique pas. Le moteur de routage ignore cette route et continue d'évaluer la prochaine route correspondante.
- **Erreur HSS ou Diameter désactivé** : Échec-fermé. Le message est retenu pour SAF.

```
{
  "action": "hold_for_on_net",
  "priority": 200,
  "description": "Retenir les abonnés sur réseau pour une
livraison locale"
}
```

Avec destination explicite :

```
{
  "action": "hold_for_on_net",
  "dest_smsc": "ims-frontend-01",
  "priority": 200,
  "description": "Routage sur réseau vers le frontend IMS"
}
```

Configuration Typique : Retenir Sur Réseau + Repli Hors Réseau

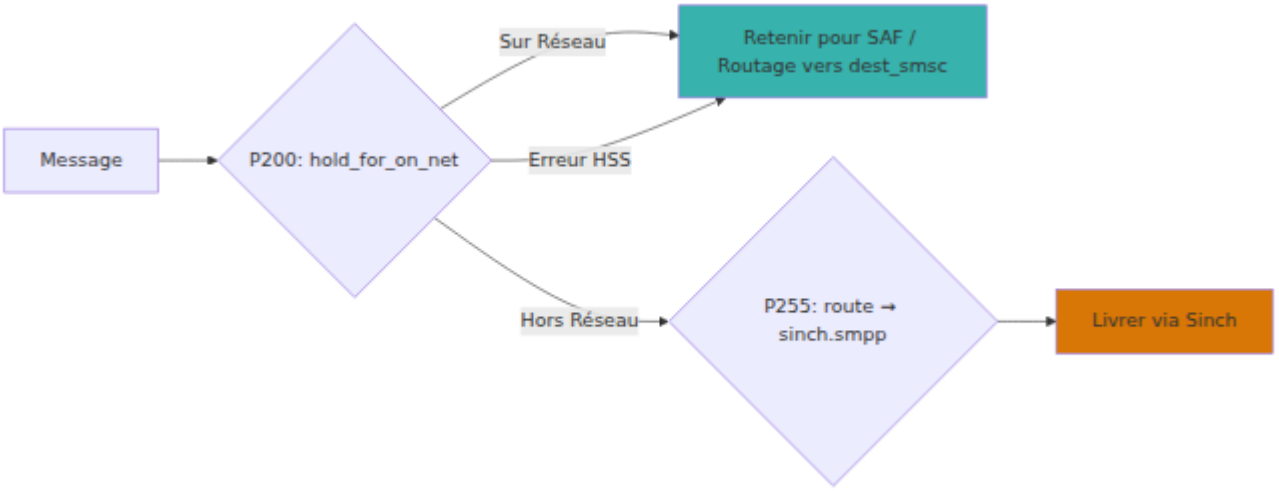


Cela garantit que les abonnés sur réseau ne fuient jamais vers la passerelle hors réseau. Les numéros hors réseau passent proprement.

Cache de Résultat HSS

Les recherches HSS sont coûteuses (aller-retour Diameter vers HSS). Le système met en cache les résultats par MSISDN pour éviter les recherches redondantes.

- **TTL du cache** : 2 heures (configurable via `hss_cache_ttl_seconds`)
- **Recherche paresseuse** : HSS n'est interrogé que lorsque une route a réellement besoin du statut sur réseau (`hold_for_on_net`, `originating_on_net_only`, `terminating_on_net_only`)
- **Partagé entre les routes** : Si plusieurs routes ont besoin du même statut de MSISDN, une seule recherche HSS est effectuée
- **Re-routage forcé** : Contourne le cache (nouvelle recherche) lorsqu'un message est explicitement re-routé



Drapeaux d'Application Sur Réseau HSS

Ce sont des drapeaux booléens sur n'importe quelle route qui déterminent si la route s'applique en fonction du statut HSS. Ils fonctionnent indépendamment de l'action de route.

`originating_on_net_only`

Si `true`, le MSISDN source est vérifié contre le HSS. S'il est hors réseau, le message est rejeté. Utilise le cache HSS.

Cas d'utilisation : restreindre un lien SMPP ou un transporteur externe à n'accepter que le trafic des originators sur réseau.

`terminating_on_net_only`

Si `true`, le MSISDN de destination est vérifié contre le HSS. S'il est hors réseau, le message est rejeté. Utilise le cache HSS.

Cas d'utilisation : restreindre une route à ne livrer qu'aux abonnés sur réseau.

Comportement Échec-Fermé

Tous les contrôles sur réseau échouent de manière fermée :

- HSS renvoie une erreur → message rejeté
- Diameter désactivé → message rejeté
- HSS injoignable → message rejeté

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
- Détail de ce qui a été comparé (modèle regex vs numéro, nom SMSC vs source, etc.)

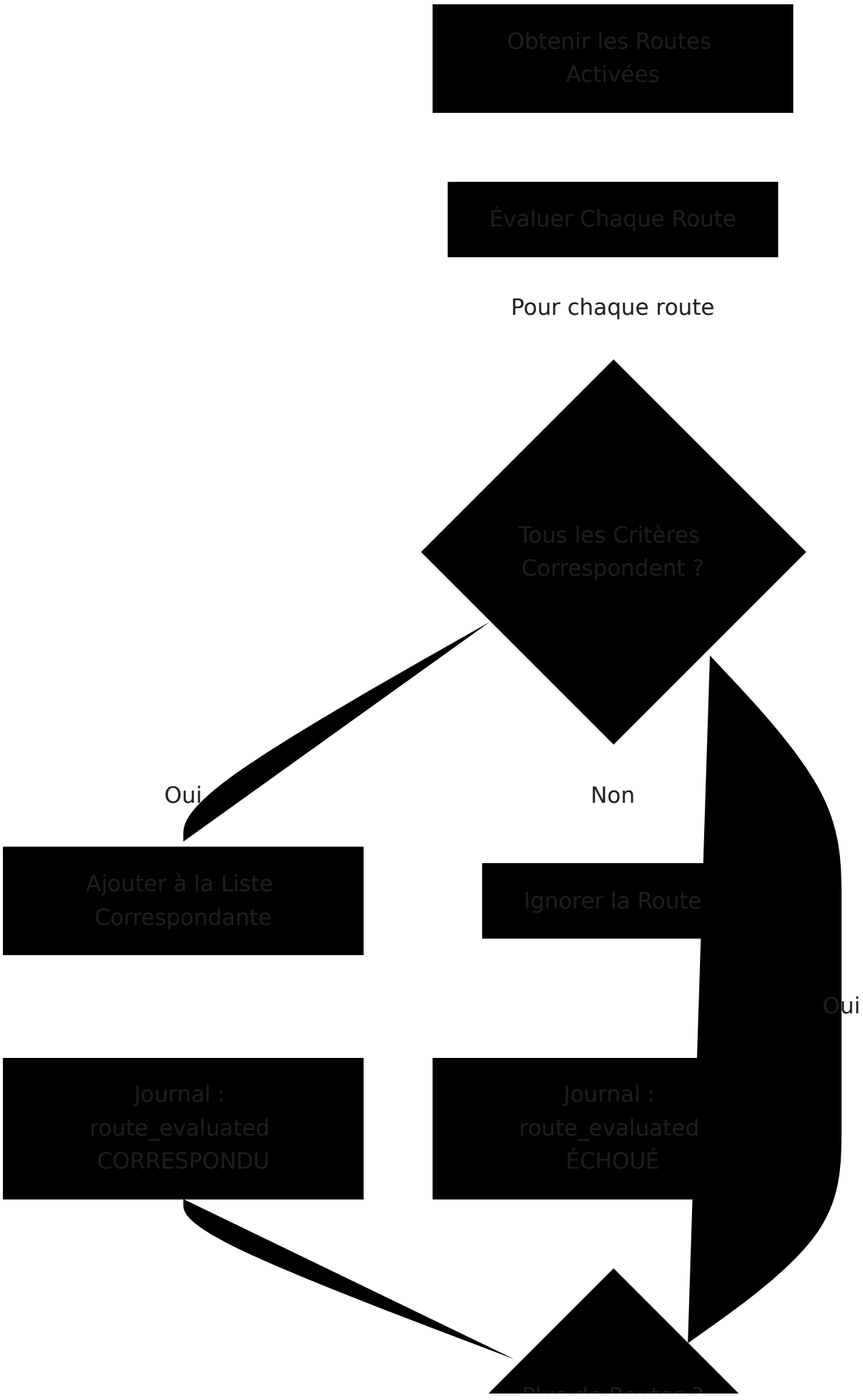
Exemples d'entrées de journal d'événements :

```
Route #4 [hold_for_on_net] CORRESPONDU : P200 "Retenir sur réseau"
|
  calling_regex: ✓ (joker), called_regex: ✓ (joker),
  source_smsc: ✓ (joker), source_type: ✓ (joker),
  enum_result: ✓ (joker)

Route #3 [route] ÉCHOUÉ : P255 "Repli Sinch" → sinch.smpp |
  calling_regex: ✓ (joker), called_regex: ✓ (joker),
  source_smsc: ✓ (joker), source_type: ✗ (ims vs
circuit_switched),
  enum_result: ✓ (joker)
```

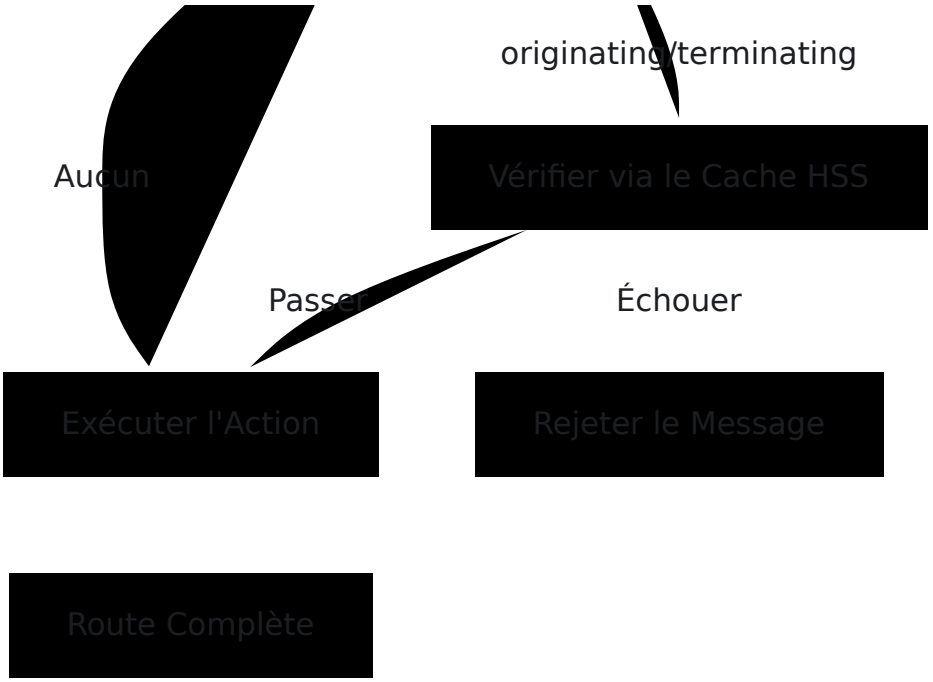
Ces événements sont visibles dans le journal d'activité des messages du panneau de contrôle et via l'API des Événements.

Algorithme d'Évaluation des

Routes


Obtenir les Routes Activées
Évaluer Chaque Route
Pour chaque route
Tous les Critères Correspondent ?
Oui
Non
Ajouter à la Liste Correspondante
Ignorer la Route
Journal : route_evaluated CORRESPONDU
Journal : route_evaluated ÉCHOUÉ
Oui
Plus de Routes ?


Plus de Routes ?
Non
SELECT
Trier par Priorité puis Spécificité
Regrouper par Priorité la Plus Élevée
Sélection Aléatoire Pondérée
Route Sélectionnée
Contrôles Sur Réseau ?



Score 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 lors du premier démarrage lorsque le tableau de routage est vide. Les redémarrages suivants préservent les routes ajoutées via l'API.

```
config :sms_c, :sms_routes, [
  %{
    called_regex: ~r/^68987/,
    dest_smsc: "pacific-gw",
    priority: 100,
    description: "Routage vers les Îles du Pacifique"
  },
  %{
    action: :hold_for_on_net,
    priority: 200,
    description: "Retenir sur réseau pour SAF"
  },
  %{
    action: :auto_reply,
    called_regex: ~r/^911$/,
    auto_reply_message: "Urgent : Veuillez composer le 911.",
    priority: 1,
    description: "Réponse automatique 911"
  }
]
```

Paramètres de Configuration de Route

Paramètre	Type	Requis	Par défaut	

`action`	Atome/Chaîne	Non	`:route`	Act
				`:r`
				`:a`
				`:h`

`calling_regex`	Regex	Non	`nil`	Mo le `ni` tou
`called_regex`	Regex	Non	`nil`	Mo le des cor
`source_smsc`	Chaîne	Non	`nil`	Filt SM cor

`dest_smsc`	Chaîne	Conditionnel	`nil`	Re
				`ro`
				pou
				`ho`
				No
				`dr`

`source_type`	Atome	Non	`nil`	Filt
				sou
				`:c`
				`:s`

Paramètre	Type	Requis	Par défaut	
`auto_reply_message`	Chaîne	Conditionnel	`nil`	Re l'ac `:a`
`charged`	Atome	Non	`:default`	Pol fac `:n`
`weight`	Entier	Non	`100`	Poi de
`priority`	Entier	Non	`100`	Pri (1- plu
`description`	Chaîne	Non	`""`	De par
`enabled`	Booléen	Non	`true`	Ind est
`originating_on_net_only`	Booléen	Non	`false`	Exi MS sur HS
`terminating_on_net_only`	Booléen	Non	`false`	Exi MS des rés

Configuration du Cache HSS

Paramètre	Type	Par défaut	Description
`hss_cache_ttl_seconds`	Entier	`7200`	Durée de vie pour les résultats de recherche HSS mis en cache (secondes)

API

Créer une Route

```
POST /api/routes
```

```
{
  "action": "hold_for_on_net",
  "priority": 200,
  "description": "Retenir les abonnés sur réseau pour une
livraison locale"
}
```

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 pays manquant)
- Toutes les routes correspondantes ont `hold_for_on_net` et la destination est hors réseau, mais aucune route de repli n'existe

Résolution :

1. Vérifiez le journal des événements du message — chaque route évaluée est enregistrée avec réussite/échec par critère
2. Utilisez le simulateur de routage dans le panneau de contrôle pour tester avec les mêmes numéros
3. Assurez-vous qu'une route de repli générale 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 pour une livraison locale

Causes possibles :

- Aucune route `hold_for_on_net` configurée
- La route `hold_for_on_net` a un numéro de priorité plus élevé que la route de repli hors réseau
- Échec de la recherche HSS (vérifiez les événements `hss_dip_error`)
- Diameter non activé dans la configuration

Résolution :

1. Ajoutez une route `hold_for_on_net` avec un numéro de priorité inférieur à celui du repli 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 récents de recherche dans le journal des événements

Cache HSS Obsolète

Symptômes : Abonné porté ou provisionné mais toujours routé en fonction d'un ancien statut sur réseau/hors réseau

Résolution :

1. Re-routage du message via l'API — le re-routage forcé contourne le cache HSS
2. Attendre l'expiration du TTL du cache (par défaut 2 heures)
3. Réduire `hss_cache_ttl_seconds` si des mises à jour plus rapides sont nécessaires

Guide de Dépannage SMS-C

← Retour à l'Index de Documentation | **README Principal**

Guide complet pour diagnostiquer et résoudre les problèmes courants de SMS-C.

Table des Matières

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 sms_c

# 5. Vérifier la connectivité de 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 réussite 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 en statut "en attente"
- Nombre élevé de messages en attente
- 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 attente"
- 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 du 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` en augmentation
- Messages avec un nombre élevé de tentatives (> 3)
- Retards d'exponential backoff

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
- Horodatages de réessai

2. **Vérifier les Journaux du 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 des numéros
- Supprimer le message s'il est vraiment invalide

Messages de Lettre Morte

Symptômes :

- `deadletter: true` dans le message
- Messages dépassant le temps d'expiration
- Statut toujours "en attente"

Étapes de Diagnostic :

1. **Trouver les Messages de Lettre Morte** :

Accéder à l'Interface Web : `/message_queue`

- Filtrer par statut expiré
- Vérifier les horodatages 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`
- Le métrique `sms_c_routing_failed_count` augmente
- Le journal des événements montre "routing_failed"

Étapes de Diagnostic :

1. **Vérifier l'Existence des Routes** :

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** :

- Des correspondances de préfixe sont-elles requises ?
- 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 Dest : default_gateway
Priorité : 255
Poids : 100
Activé : ✓
Description : Route par défaut de secours
```

Routes Trop Spécifiques :

Ajouter une route plus large :

```
Préfixe Appelé : +
SMSC Dest : 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
- L'équilibrage de charge ne distribue 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 scores de priorité et de spécificité

2. **Vérifier les Priorités de Route** :

- Un nombre inférieur = 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 :

Rendre une route spécifique de priorité plus élevée :

```
Route Premium :
  Préfixe Appelé : +1555
  Priorité : 10 (haute priorité)

Route Générale :
  Préfixe Appelé : +1
  Priorité : 50 (priorité inférieure)
```

Ajuster les Poids :

Changer la distribution de l'équilibrage de charge :

```
Principal (70%) :
  Poids : 70

Sauvegarde (30%) :
  Poids : 30
```

Ajouter une Route Plus Spécifique :

Remplacer la route générale pour un cas spécifique :

```
Route Spécifique :
  Préfixe Appelé : +15551234
  SMSC Dest : dedicated_gateway
  Priorité : 1

Route Générale :
  Préfixe Appelé : +1
  SMSC Dest : general_gateway
  Priorité : 50
```

Réponse Automatique Ne Fonctionne Pas

Symptômes :

- Route de réponse automatique configurée mais ne se déclenche pas
- Aucun message de réponse n'est envoyé

- Journal des événements manquant l'év��nement de réponse automatique

Étapes de Diagnostic :

1. **Vérifier la Configuration de la Route** :

- `auto_reply: true`
- `auto_reply_message` contient du texte
- La route est activée
- La 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 faible) :

```
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 en augmentation

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 route

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` près 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)
```

Croissance de la File d'Attente

Symptômes :

- `sms_c_queue_size_pending` en augmentation
- Âge du message le plus ancien en augmentation

- Le traitement ne peut pas suivre le taux d'entrée

Étapes de Diagnostic :

1. **Vérifier le Taux d'Entrée vs de Livraison** :

```
# Taux d'entrée
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 suffisamment souvent ?
- Les frontends traitent-ils les messages assez rapidement ?
- Des erreurs de frontend ?

3. **Vérifier le Taux de Réussite 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 de frontend
- Optimiser le traitement du frontend
- Augmenter la concurrence du 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 la normalité
- Envisager des mises à niveau de capacité si récurrentes

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** :

```
# Ping de l'hôte de la base de données
ping db.example.com

# Vérifier la connectivité du 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 une 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 sms_c
```

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 en augmentation

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 plus gros fichiers (MySQL/MariaDB)
find /var/lib/mysql -type f -exec du -h &amp;#123;&amp;#125; + | sort -rh
| head -20

# Trouver les plus gros fichiers (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 :

```
-- Supprimer les anciens messages
DELETE FROM message_queues
WHERE inserted_at < DATE_SUB(NOW(), INTERVAL 90 DAY)
LIMIT 100000;
```

Faire Pivoter 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'Enregistrement** :

```
curl https://api.example.com:8443/api/frontends/active | grep
frontend_name
```

2. **Vérifier les Journaux du Frontend** :

- Le frontend appelle-t-il `/api/frontends/register` ?
- Des erreurs API ?
- Fréquence d'enregistrement (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'enregistre Pas :

Tester l'enregistrement manuel :

```
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.

Délai d'Enregistrement :

Les frontends expirent après 90 secondes. Assurez-vous d'un enregistrement toutes les 60 secondes :

```
# Le frontend doit appeler enregistrer 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'enregistrement 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 Délai d'Enregistrement** :

- 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 du frontend pour des erreurs

Problèmes Réseau :

- Vérifier la connectivité intermittente
- Examiner les journaux du pare-feu
- Vérifier les vérifications de santé de l'équilibreur de charge

Mauvais Intervalle d'Enregistrement :

Corriger l'intervalle :

```
REGISTRATION_INTERVAL = 60  # secondes
```

Problèmes de Facturation/Chargement

Échecs de Facturation

Symptômes :

- `sms_c_charging_failed_count` en augmentation
- Le journal des événements montre "charging_failed"
- Messages marqués comme `charge_failed: true`

Étapes de Diagnostic :

1. **Vérifier la Connectivité OCS** :

```
# Tester l'API OCS
curl -X POST http://ocs.example.com:2080/jsonrpc \
  -H "Content-Type: application/json" \
  -d '&amp;#123;
    "method": "SessionSv1.Ping",
    "params": [],
    "id": 1
  &amp;#125;'
```

Attendu : `{"result":"Pong"}`

2. **Vérifier les Journaux OCS** :

```
tail -f /var/log/ocs/ocs.log
```

3. **Vérifier la Configuration** :

```
# Vérifier l'URL OCS
grep ocs_url config/runtime.exs
```

Solutions :

OCS Indisponible :

```
# Vérifier l'état de l'OCS
systemctl status ocs

# Démarrer si nécessaire
systemctl start ocs
```

Erreur de Configuration :

Mettre à jour la configuration :

```
config :sms_c,
  ocs_url: "http://correct-host:2080/jsonrpc",
  ocs_tenant: "correct_tenant"
```

Désactiver la Facturation Temporairement :

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

Redémarrer l'application.

Problèmes de Compte :

- Vérifier que le compte existe dans l'OCS
- Vérifier que le compte a un solde
- Vérifier que les plans de tarification sont configurés

Facturation Trop Lente

Symptômes :

- `sms_c_charging_succeeded_duration` p95 > 500ms
- Traitement des messages lent lorsque la facturation est activée
- Rapide lorsque la facturation est désactivée

Étapes de Diagnostic :

1. **Vérifier la Latence de Facturation** :

```
histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)
```

2. **Vérifier la Performance de l'OCS** :

```
# Temps de réponse OCS
curl -w "%&#123;time_total&#125;\n" -X POST
http://ocs.example.com:2080/jsonrpc \
  -H "Content-Type: application/json" \
  -d '&#123;"method":"SessionSv1.Ping","params":[],"id":1&#125;'
```

3. **Vérifier la Latence Réseau** :

```
# Ping de l'hôte OCS
ping -c 10 ocs.example.com
```

Solutions :

OCS Lent :

- Optimiser la configuration de l'OCS
- Ajouter des ressources à l'OCS
- Utiliser un moteur de tarification plus rapide

Latence Réseau :

- Déployer l'OCS plus près de SMS-C
- Utiliser un chemin réseau direct
- Éviter les VPN/tunnels si possible

Délai d'Attente Trop Court :

Augmenter le délai d'attente :

```
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. **Vérifier la Configuration ENUM** :

```
grep -A 10 "enum_" config/runtime.exs
```

2. **Tester la Connectivité DNS** :

```
# Tester le serveur DNS
dig @8.8.8.8 e164.arpa

# Tester 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. **Vérifier le Serveur DNS** :

```
# Le DNS personnalisé est-il accessible ?
ping 10.0.1.53

# Tester le port
nc -zv 10.0.1.53 53
```

Solutions :

Serveur DNS Inaccessible :

Utiliser un DNS alternatif :

```
config :sms_c,
  enum_dns_servers: [
    &#123;"8.8.8.8", 53&#125;,  # DNS Public Google
    &#123;"1.1.1.1", 53&#125;   # DNS Cloudflare
  ]
```

Domaine ENUM Incorrect :

Mettre à jour le domaine :

```
config :sms_c,
  enum_domains: ["e164.arpa"]  # Utiliser le domaine standard
```

Délai d'Attente Trop Court :

Augmenter le délai d'attente :

```
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 croissante sans limite
- Utilisation de la mémoire élevée

Étapes de Diagnostic :

1. **Vérifier 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. **Vérifier le Modèle de Trafic** :

- Les numéros se répètent-ils ?
- Le TTL du cache est-il approprié ?

Solutions :

Taux de Réussite Bas (Attendu) :

- Trafic vers des numéros uniques (normal)
- Surveiller mais ne pas alarmer si < 70%

Cache Croissant :

Effacer le cache via la page de Test NAPTR ou redémarrer l'application.

Utilisation Élevée de la Mémoire :

- Attendu avec un grand cache
- Surveiller la mémoire système globale
- Envisager un ajustement du TTL

Problèmes de Cluster

Le Nœud Ne Peut Pas Rejoindre le Cluster

Symptômes :

- Un seul nœud en cours d'exécution
- Les requêtes de cluster renvoient uniquement des résultats locaux
- Erreurs de distribution Erlang

Étapes de Diagnostic :

1. **Vérifier les Noms de Nœud** :

```
# Dans la console IEx
Node.self()
# Attendu : :sms@node1.example.com

Node.list()
# Attendu : Liste des autres nœuds
```

2. **Vérifier le Cookie Erlang** :

```
# Vérifier le fichier cookie
cat ~/.erlang.cookie

# Vérifier le même sur tous les nœuds
```

3. **Vérifier le Réseau** :

```
# Les nœuds peuvent-ils se joindre ?
ping node2.example.com

# Vérifier les ports
nc -zv node2.example.com 4369
nc -zv node2.example.com 9100-9200
```

Solutions :

Mismatch de Cookie :

Définir le même cookie sur tous les nœuds :

```
export ERLANG_COOKIE=same_secret_value_here

# Ou mettre à jour ~/.erlang.cookie
echo "same_secret_value_here" > ~/.erlang.cookie
chmod 400 ~/.erlang.cookie
```

Pare-feu Bloquant :

Ouvrir 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 :

Utiliser 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. **Vérifier la Connectivité des Nœuds** :

```
# Sur chaque nœud (IEx)
Node.list()
```

2. **Vérifier Mnesia** :

```
:mnesia.system_info(:running_db_nodes)
```

Solutions :

Reconnecter les Nœuds :

```
# Arrêter tous les nœuds
systemctl stop sms_c

# Démarrer un nœud d'abord
systemctl start sms_c  # Sur node1

# Attendre qu'il démarre complètement, puis démarrer les autres
systemctl start sms_c  # Sur node2
systemctl start sms_c  # Sur node3
```

Incohérence Mnesia :

- Exporter les routes depuis le nœud correct
- Arrêter tous les nœuds
- Supprimer le répertoire Mnesia
- Démarrer les nœuds
- Importer les routes

Problèmes d'API

L'API Ne Répond Pas

Symptômes :

- Délai de connexion
- Connexion refusée
- Pas de réponse

Étapes de Diagnostic :

1. **Vérifier le Processus API** :

```
# L'application est-elle en cours d'exécution ?
systemctl status sms_c

# Vérifier les ports d'écoute
netstat -tlnp | grep 8443
```

2. **Vérifier le Pare-feu** :

```
# Vérifier iptables
iptables -L -n | grep 8443

# Tester la connectivité locale
curl -k https://localhost:8443/api/status
```

3. **Vérifier la Configuration TLS** :

```
# Vérifier que le certificat existe
ls -l priv/cert/server.crt priv/cert/server.key

# Vérifier la validité du certificat
openssl x509 -in priv/cert/server.crt -noout -dates
```

Solutions :

Application Ne Fonctionne Pas :

```
systemctl start sms_c
```

Pare-feu Bloquant :

```
# Autoriser le port API
iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

Problèmes de Certificat :

Générer un nouveau certificat (voir le Guide de Configuration).

Mauvais Port :

Vérifier la configuration :

```
grep "port:" config/runtime.exs
```

L'API Renvoie des Erreurs 500

Symptômes :

- Erreur Interne du Serveur
- Code d'état 500
- Erreur dans les journaux

Étapes de Diagnostic :

1. **Vérifier les Journaux de l'Application** :

```
tail -100 /var/log/sms_c/application.log | grep "\[error\]"
```

2. **Vérifier la Base de Données** :

```
mysql -u sms_user -p -e "SELECT 1"
```

3. **Vérifier les Ressources** :

```
# Mémoire
free -h

# CPU
top -b -n 1

# Disque
df -h
```

Solutions :

Base de Données Indisponible :

- Démarrer la base de données
- Corriger le problème de connexion

Hors Mémoire :

- Redémarrer l'application
- Augmenter la mémoire système
- Vérifier les fuites de mémoire

Erreur d'Application :

- Vérifier l'erreur spécifique dans les journaux
- Corriger le problème de configuration
- Redémarrer l'application

Problèmes d'Interface Web

Impossible d'Accéder à l'Interface Web

Symptômes :

- Délai de connexion
- 404 Not Found
- La page ne se charge pas

Étapes de Diagnostic :

1. **Vérifier l'État de l'Application** :

```
systemctl status sms_c
```

2. **Vérifier le Port** :

```
netstat -tlnp | grep 80
```

3. **Vérifier l'URL** :

- Nom d'hôte correct ?
- Port correct ?
- HTTP vs HTTPS ?

Solutions :

Mauvais Port :

Vérifier la configuration :

```
grep "control_panel" config/runtime.exs
```

Accéder sur le port correct (par défaut : 80 ou 4000).

Application Ne Fonctionne Pas :

```
systemctl start sms_c
```

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. **Vérifier la Console du Navigateur** :

- Ouvrir les Outils de Développeur (F12)
- Rechercher des erreurs WebSocket
- Vérifier l'onglet réseau pour des requêtes échouées

2. **Vérifier la Configuration du Proxy** :

Si utilisation d'un proxy inverse, assurer le support WebSocket :

```
location /live &amp;#123;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
&amp;#125;
```

Solutions :

WebSocket Bloqué :

- Configurer le proxy pour WebSocket
- Vérifier le pare-feu
- Vérifier les extensions du navigateur

Rafraîchir la Page :

- Rafraîchissement forcé (Ctrl+F5)
- Effacer 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. **Vérifier le Processus** :

```
top -b -n 1 | grep beam.smp
```

2. **Vérifier 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 :

- Échelonner horizontalement (ajouter des nœuds)
- Échelonner verticalement (ajouter du CPU)

Routage Inefficace :

- Réduire le nombre de routes
- Optimiser les critères de route

Trop de Recherches ENUM :

- Vérifier le taux de réussite du cache
- Envisager de désactiver si non nécessaire

Utilisation Élevée de la Mémoire

Symptômes :

- Utilisation de la mémoire > 90%
- Application plante
- Erreurs de mémoire insuffisante

Étapes de Diagnostic :

1. **Vérifier la Mémoire** :

```
free -h

ps aux | grep beam.smp
```

2. **Vérifier les Tailles de Cache** :

```
sms_c_enum_cache_size_size
```

Solutions :

Cache ENUM Trop Grand :

- Effacer le cache

- Réduire le TTL
- Désactiver ENUM si non nécessaire

File de Lot Croissante :

```
# Vérifier les statistiques du travailleur (IEx)
SmsC.Messaging.BatchInsertWorker.stats()
```

Si la file est grande, vider manuellement ou redémarrer.

Ajouter de la Mémoire :

- Échelonner verticalement
- Ajouter un swap (temporaire)

Fuite de Mémoire :

- Redémarrer l'application
- Signaler 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 de l'application - `/var/log/sms_c/application.log`