SMSc - Application du Centre de Service SMS

Application du Centre SMS construite avec Phoenix/Elixir, fournissant une file de messages centralisée et une API REST pour le routage et la livraison des SMS.

Documentation

Documentation Principale

- **Index de Documentation Complet** - Commencez ici pour toute la documentation
- **Référence de Configuration** - Options de configuration complètes
- **Référence API** - Documentation de l'API REST
- **Guide des Opérations** - Opérations quotidiennes et surveillance
- **Guide de Routage SMS** - Gestion et configuration des routes
- **Guide de Traduction de Numéros** - Normalisation et réécriture des numéros
- **Optimisation des Performances** - Optimisation pour différentes charges de travail
- **Guide des Métriques** - Métriques Prometheus et surveillance
- **Guide de Dépannage** - Problèmes courants et solutions
- **Schéma CDR** - Format des enregistrements de détails d'appel

Documentation de Conformité

- **Conformité à l'Interception ANSSI R226** - Spécifications techniques françaises pour l'interception légale

Performance

- **Benchmarks** - Tests de performance

Vue d'Ensemble de l'Architecture

Le cœur de SMS_C fournit une file de messages indépendante du protocole et une API REST. Les frontaux SMSC externes (SMPP, IMS, SS7/MAP) se connectent en tant que passerelles autonomes qui communiquent avec le cœur via l'API REST.

Flux de Messages

Flux de Messages Sortants (MT - Mobile Terminated)

Flux de Messages Entrants (MO - Mobile Originated)

Fonctionnalités Clés

1. Conception Indépendante du Protocole

- Le cœur de SMS_C gère la persistance des messages, le routage et l'API
- Les frontaux externes (SMPP, IMS, SS7/MAP) gèrent la communication spécifique au protocole
- Tous les frontaux communiquent via une API REST unifiée
- Ajoutez de nouveaux protocoles sans changer le cœur
- Chacun peut être mis à l'échelle indépendamment

2. Routage des Messages

- **Moteur de routage dynamique** avec configuration à l'exécution
- Routage basé sur les préfixes (numéros appelants/appelés)
- Filtrage par type de SMSC et source (IMS/Circuit Commuté/SMPP)
- Routage basé sur la priorité avec équilibrage de charge basé sur le poids
- Routage de réponse automatique et capacités de suppression de message
- Contrôle de facturation par route
- Interface Web pour la gestion des routes
- Mises à jour de route en temps réel sans interruption de service

📖 Voir Guide de Routage SMS pour une documentation complète

3. Logique de Réessai avec Retour Exponentiel

- Réessai automatique en cas d'échec de livraison
- Retour exponentiel : 1min, 2min, 4min, 8min, etc.
- Tentatives de réessai maximum configurables
- Gestion de l'expiration des messages
- Suivi des réessais par message

Guide des Opérations

Points d'Accès :

- API REST : `https://localhost:8443` (ou `http://localhost:8080` sans TLS)
- Panneau de Contrôle : `https://localhost:8086`
- Documentation API (Swagger UI) : `https://localhost:8443/api/docs`

Démarrer les Frontaux Externes : Chaque frontal de protocole est une application autonome. Consultez la documentation individuelle des frontaux pour les instructions de démarrage.

Configuration

Toute la configuration est gérée directement dans `config/runtime.exs`. Aucune variable d'environnement n'est utilisée.

Configuration Principale

Aucune variable d'environnement de configuration de l'application principale n'est actuellement utilisée. Les ports du serveur et les noms d'hôte sont configurés dans `config/runtime.exs` :

- Serveur API : Port 8443 (HTTPS), écoutant sur 0.0.0.0
- Panneau de Contrôle : Port 80 (HTTP), écoutant sur 0.0.0.0

Configuration de la Base de Données

Les paramètres de la base de données sont configurés dans `config/runtime.exs` :

- Nom d'utilisateur : `omnitouch`
- Mot de passe : `omnitouch2024`
- Nom d'hôte : `localhost`
- Port : `3306`
- Nom de la base de données : `smsc_new`
- Taille du pool : `1`

Configuration de Cluster

Les paramètres de cluster sont configurés dans `config/runtime.exs` :

- Nœuds de cluster : `""` (vide - pas de clustering par défaut)
- Requête DNS de cluster : `nil`

Configuration de la File de Messages

Les paramètres de la file de messages sont configurés dans `config/runtime.exs` :

- Temps de lettre morte : `1440` minutes (24 heures avant l'expiration du message)

Intégration de Facturation

Les paramètres de facturation sont configurés dans `config/runtime.exs` :

- URL : `http://localhost:2080/jsonrpc`
- Locataire : `mnc057.mcc505.3gppnetwork.org`
- Destination : `55512341234`
- Source : `00101900000257`
- Sujet : `00101900000257`
- Compte : `00101900000257`

Configuration de Routage SMS

Le système de routage SMS utilise des routes dynamiques, soutenues par une base de données, qui peuvent être gérées via l'interface Web ou le fichier de configuration. Les routes sont chargées depuis `config/runtime.exs` au premier démarrage.

Exemple de Configuration :

```
config :sms_c, :sms_routes, [
  %{
    called_prefix: "+44",
    dest_smsc: "InternationalGW",
    weight: 100,
    priority: 100,
    description: "SMS International UK",
    enabled: true
  },
  %{
    called_prefix: "1900",
    dest_smsc: "PremiumGW",
    charged: :yes,
    priority: 50,
    description: "Numéros Premium US",
    enabled: true
  }
]
```

Fonctionnalités :

- Correspondance basée sur les préfixes (numéros appelants/appelés)
- Filtrage par SMSC source et type
- Routage basé sur la priorité et le poids
- Capacités de réponse automatique et de suppression
- Contrôle de facturation par route
- Gestion à l'exécution via l'interface Web à `/routing`

🧭 Voir Guide de Routage SMS pour une documentation complète, des exemples et une référence API.

Points de Terminaison de l'API REST

Opérations sur la File de Messages

Soumettre un SMS (Créer un Message)

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

{
  "source_msisdn": "+1234567890",
  "destination_msisdn": "+0987654321",
  "message_body": "Bonjour, le monde !",
  "source_smsc": "web-app",
  "dest_smsc": "smpp-provider",          # Optionnel - le moteur de
routage attribue si nul
  "tp_dcs_character_set": "gsm7",        # Optionnel : gsm7, 8bit,
latin1, ucs2
  "tp_dcs_coding_group": "general_data_coding",
  "expires": "2025-10-17T10:30:00Z"      # Optionnel - par défaut 24h
à partir de maintenant
}
```

Champs Requis :

- `destination_msisdn` - Numéro de téléphone de destination
- `message_body` - Contenu du texte du message
- `source_msisdn` - Numéro de téléphone source
- `source_smsc` - Identifiant du système source

Champs Optionnels :

- `dest_smsc` - Passerelle de destination (le moteur de routage attribue si non fourni)
- `source_imsi`, `dest_imsi` - Identifiants IMSI
- `tp_dcs_character_set` - Encodage des caractères (gsm7, 8bit, latin1, ucs2)
- `tp_dcs_coding_group` - Groupe de codage DCS
- `tp_dcs_compressed` - Drapeau de compression (booléen)
- `tp_dcs_has_message_class` - Drapeau de classe de message (booléen)
- `tp_dcs_message_class` - Valeur de classe de message
- `tp_user_data_header` - En-tête de données utilisateur (carte)
- `message_part_number`, `message_parts` - Champs de message multipart
- `expires` - Horodatage d'expiration (par défaut 24 heures)

- `deliver_after` - Horodatage de livraison différée
- `deadletter`, `raw_data_flag`, `raw_sip_flag` - Drapeaux booléens

Réponse :

```
{
  "status": "success",
  "data": {
    "id": 123,
    "source_msisdn": "+1234567890",
    "destination_msisdn": "+0987654321",
    "dest_smsc": "smpp-provider",
    "message_body": "Bonjour, le monde !",
    "deliver_time": null,
    "delivery_attempts": 0,
    "expires": "2025-10-17T10:30:00Z",
    "inserted_at": "2025-10-16T10:30:00Z"
  }
}
```

Obtenir des Messages pour SMSC

```
GET /api/messages/get_by_smsc?smsc=nom-de-mon-smsc
```

Retourne tous les messages non livrés où :

- `destination_smsc` est nul OU correspond au nom SMSC fourni
- Le message n'est pas expiré
- Prêt à être envoyé (deliver_after est nul ou dans le passé)

Réponse :

```
{
  "status": "success",
  "data": [
    {
      "id": 123,
      "source_msisdn": "+1234567890",
      "destination_msisdn": "+0987654321",
      "message_body": "Bonjour",
      "destination_smsc": "nom-de-mon-smsc",
      "delivery_attempts": 0
    }
  ]
}
```

Lister les Messages avec Filtrage SMSC Optionnel

```
# Lister tous les messages dans la file
GET /api/messages

# Lister les messages pour un SMSC spécifique (avec filtrage par en-tête)
GET /api/messages
SMSc: nom-de-mon-smsc
```

Sans En-tête SMSc : Retourne tous les messages dans la file, quel que soit l'état de livraison ou l'expiration.

Avec En-tête SMSc : Retourne les messages non livrés où :

- `dest_smsc` correspond à la valeur de l'en-tête OU `dest_smsc` est nul
- `deliver_time` est nul (pas encore livré)
- `deliver_after` est nul ou avant/égal à l'heure actuelle (prêt à livrer)
- `expires` est après l'heure actuelle (non expiré)
- Ordonné par heure d'insertion (le plus ancien en premier)

Remarque : L'approche par en-tête SMSc permet aux frontaux externes de sonder leurs messages en utilisant le même modèle de point de terminaison, avec l'en-tête contrôlant le comportement de filtrage.

Réponse :

```
[
  {
    "id": 123,
    "source_msisdn": "+1234567890",
    "destination_msisdn": "+0987654321",
    "message_body": "Bonjour, le monde !",
    "dest_smsc": "nom-de-mon-smsc",
    "deliver_time": null,
    "delivery_attempts": 0,
    "expires": "2025-10-17T10:30:00Z",
    "inserted_at": "2025-10-16T10:30:00Z"
  }
]
```

Obtenir un Message Unique

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

Mettre à Jour un Message

```
PATCH /api/messages/{id}
```

```
Content-Type: application/json

{
  "status": "delivered",
  "delivered_at": "2025-10-16T10:30:00Z"
}
```

Supprimer un SMS

```
DELETE /api/messages/{id}
```

Gérer l'Échec de Livraison (Incrémenter le Compteur de Réessai)

Lorsqu'une livraison de message échoue temporairement, incrémentez le compteur de tentatives de livraison et planifiez un réessai avec retour exponentiel.

Méthode 1 : Utilisation de PUT (Recommandé)

```
# Simple et sémantique - PUT indique la mise à jour de l'état de
livraison
PUT /api/messages/{id}
```

Méthode 2 : Utilisation d'un Point de Terminaison Explicite

```
# Point de terminaison explicite alternatif
POST /api/messages/{id}/increment_delivery_attempt
```

Les deux méthodes incrémentent `delivery_attempts` et définissent le délai de retour exponentiel via `deliver_after` :

Tentative	Formule de Retour	Délai	Temps Total
1��re	2^1 minutes	2 min	2 min
2ème	2^2 minutes	4 min	6 min
3ème	2^3 minutes	8 min	14 min
4ème	2^4 minutes	16 min	30 min
5ème	2^5 minutes	32 min	1h 2min
6ème	2^6 minutes	64 min	2h 6min

Réponse :

```
{
  "id": 123,
  "delivery_attempts": 1,
  "deliver_after": "2025-10-20T19:05:00Z",
  "deliver_time": null,
  "expires": "2025-10-21T19:03:00Z",
```

```
  ...
}
```

Remarque : Les messages avec un `deliver_after` futur sont automatiquement filtrés des requêtes GET jusqu'à ce que la période de retour expire.

Mettre à Jour un Message (Mise à Jour Partielle)

Pour mettre à jour des champs spécifiques d'un message (comportement inchangé) :

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

{
  "dest_smsc": "passerelle-mise-à-jour",
  "status": "delivered"
}
```

Important : `PUT` et `PATCH` se comportent différemment :

- **PUT** → Incrémente les tentatives de livraison avec retour (aucun corps requis)
- **PATCH** → Effectue des mises à jour de champs partiels (corps requis)

Suivi de la Santé des Frontaux

Le cœur de SMS_C suit la santé et la disponibilité des frontaux externes via un système d'enregistrement. Cela permet de surveiller le temps de disponibilité des frontaux, de détecter les pannes et de maintenir des données historiques de disponibilité.

Remarque : L'enregistrement des frontaux n'est PAS utilisé pour la livraison ou le routage des messages. Les messages sont routés en fonction du champ `dest_smsc`. Le système d'enregistrement existe uniquement pour la surveillance de la santé et la visibilité opérationnelle.

Comment Fonctionne l'Enregistrement des Frontaux

Chaque frontal externe (passerelle SMPP, IMS, SS7/MAP) envoie périodiquement un enregistrement de cœur battant au cœur de SMS_C :

1. **Intervalle de Battement** : Les frontaux doivent s'enregistrer toutes les 30-60 secondes
2. **Temps d'Expiration** : Les enregistrements expirent après 90 secondes sans mise à jour
3. **Gestion Automatique des États** :
    - Les nouveaux frontaux créent un nouvel enregistrement

    - Les frontaux actifs existants mettent à jour leur enregistrement (prolonge l'expiration)
    - Les frontaux expirés qui reviennent en ligne créent une nouvelle période d'enregistrement

Points de Terminaison d'Enregistrement des Frontaux

Enregistrer/Mise à Jour un Frontal (Battement)

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

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

Champs Requis :

- `frontend_name` - Identifiant unique pour l'instance du frontal
- `frontend_type` - Type de frontal (SMPP, IMS, MAP, etc.)

Champs Optionnels :

- `ip_address` - Adresse IP du frontal (détectée automatiquement à partir de la source de la requête si non fournie)
- `hostname` - Nom d'hôte du serveur frontal
- `uptime_seconds` - Secondes depuis le démarrage du frontal
- `configuration` - Chaîne JSON avec la configuration spécifique au frontal

Remarque : Si `ip_address` n'est pas fourni, le cœur de SMS_C utilisera automatiquement l'IP source de la requête HTTP. Cela fonctionne avec les connexions directes et les requêtes proxy (via l'en-tête `X-Forwarded-For`).

Réponse :

```
{
  "id": 42,
  "frontend_name": "smpp-gateway-1",
  "frontend_type": "SMPP",
  "ip_address": "10.0.1.5",
  "hostname": "smpp-gw-01",
  "uptime_seconds": 3600,
  "status": "active",
```

```
  "last_seen_at": "2025-10-20T10:30:00Z",
  "expires_at": "2025-10-20T10:31:30Z",
  "inserted_at": "2025-10-20T10:00:00Z"
}
```

Lister Tous les Enregistrements de Frontaux

```
GET /api/frontends
```

Retourne tous les enregistrements de frontaux (actifs et expirés), ordonnés par activité la plus récente.

Lister Uniquement les Frontaux Actifs

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

Retourne uniquement les frontaux actuellement actifs (non expirés).

Obtenir des Statistiques sur les Frontaux

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

Retourne des statistiques récapitulatives :

```
{
  "active": 5,
  "expired": 12,
  "unique_frontends": 8
}
```

Obtenir l'Historique des Frontaux

```
GET /api/frontends/history/{frontend_name}
```

Retourne tous les enregistrements historiques pour un frontal spécifique, utile pour analyser les modèles de disponibilité/indisponibilité.

Exemple :

```
GET /api/frontends/history/smpp-gateway-1
```

Obtenir un Enregistrement Spécifique

```
GET /api/frontends/{id}
```

Mise en Œuvre dans les Frontaux Externes

Les frontaux externes doivent mettre en œuvre une tâche en arrière-plan qui envoie des battements :

Exemple (pseudocode) :

```
import time
import requests

def send_heartbeat():
    """Envoyer un battement toutes les 30 secondes"""
    while True:
        try:
            data = {
                "frontend_name": "mon-gateway-smpp",
                "frontend_type": "SMPP",
                "ip_address": get_local_ip(),
                "hostname": get_hostname(),
                "uptime_seconds": get_uptime()
            }

            response = requests.post(
                "https://smsc-core:8443/api/frontends",
                json=data,
                timeout=5
            )

            if response.status_code in [200, 201]:
                logger.debug("Battement envoyé avec succès")
            else:
                logger.error(f"Échec du battement :
{response.status_code}")

        except Exception as e:
            logger.error(f"Erreur de battement : {e}")

        time.sleep(30)  # Envoyer toutes les 30 secondes

# Démarrer le battement dans un thread en arrière-plan
threading.Thread(target=send_heartbeat, daemon=True).start()
```

Surveillance de la Santé des Frontaux

Panneau de Contrôle - L'interface Web à `https://localhost:8086` affiche :

- Les frontaux actuellement actifs
- Horodatage de la dernière vue pour chaque frontal

- Suivi du temps de disponibilité
- Disponibilité historique

Requêtes API :

```
# Obtenir tous les frontaux actifs
curl https://localhost:8443/api/frontends/active

# Vérifier si un frontal spécifique est en ligne
curl https://localhost:8443/api/frontends/history/smpp-gateway-1 | jq
'.[0].status'

# Obtenir des statistiques de santé
curl https://localhost:8443/api/frontends/stats
```

Autres Points de Terminaison

Statut

```
GET /api/status
```

Lieux

```
GET /api/locations
POST /api/locations
GET /api/locations/{id}
PATCH /api/locations/{id}
DELETE /api/locations/{id}
```

Événements SS7

```
GET /api/ss7_events
POST /api/ss7_events
GET /api/ss7_events/{id}
PATCH /api/ss7_events/{id}
DELETE /api/ss7_events/{id}
```

File de Messages MMS

```
GET /api/mms_message_queues
POST /api/mms_message_queues
GET /api/mms_message_queues/{id}
PATCH /api/mms_message_queues/{id}
DELETE /api/mms_message_queues/{id}
```

Performance

Le cœur de SMS_C offre un débit exceptionnel en utilisant Mnesia pour le stockage de messages en mémoire avec archivage automatique vers SQL pour la rétention à long terme des CDR.

Résultats des Benchmarks

Mesuré sur Intel i7-8650U @ 1.90GHz (8 cœurs) :

Performance d'Insertion de Messages :

- `insert_message` (avec routage) : **1 750 msg/sec** (latence moyenne de 0,58 ms)
- `insert_message` (simple) : **1 750 msg/sec** (latence moyenne de 0,57 ms)
- **Capacité d'environ 150 millions de messages par jour**

Performance de Requête :

- `get_messages_for_smsc`: 800 msg/sec (1,25 ms en moyenne)
- `list_message_queues`: Accès rapide en mémoire
- Utilisation de la mémoire : 62 Ko par opération d'insertion

Architecture

Stratégie de Stockage :

- **Messages Actifs** : Stockés dans Mnesia (mémoire + disque) pour un accès ultra-rapide
- **Archive de Messages** : Archivés automatiquement vers SQL pour le stockage à long terme des CDR
- **Rétention** : Période de rétention configurable (par défaut : 24 heures)
- **Pas de goulet d'étranglement SQL** : Toutes les opérations de messages actifs contournent SQL

Configuration

Le stockage et la rétention des messages sont configurés dans `config/runtime.exs` :

```
config :sms_c,
  message_retention_hours: 24,          # Archiver les messages de plus de 24 heures
  batch_insert_batch_size: 100,         # Taille du lot CDR pour l'archivage SQL
  batch_insert_flush_interval_ms: 100   # Intervalle de vidage CDR
```

Pour des conseils détaillés sur l'optimisation, voir : docs/PERFORMANCE_TUNING.md

Surveillance

Panneau de Contrôle - Interface Web à `https://localhost:8086`

- Voir la file de messages
- Soumettre des messages de test
- Gérer le routage SMS (voir Guide de Routage)
- Simuler des décisions de routage
- Voir les ressources système
- Suivre les statistiques des travailleurs de lot

Statistiques des Travailleurs de Lot :

```
# Obtenir les statistiques actuelles des travailleurs de lot
SmsC.Messaging.BatchInsertWorker.stats()
```

Retourne :

```
%{
  total_enqueued: 10000,
  total_flushed: 9900,
  current_queue_size: 100,
  last_flush_duration_ms: 45
}
```

Logs - Les logs de l'application sont écrits dans stdout

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

Dépannage

Port Déjà Utilisé

```
# Trouver le processus utilisant le port
lsof -i :4000

# Tuer le processus
kill -9 <PID>
```

Frontend Externe Ne Se Connecte Pas

Symptômes : Messages bloqués dans la file, les logs du frontal montrent des erreurs de connexion

Vérifiez :

- Vérifiez que `API_BASE_URL` est correctement défini dans le frontal
- Vérifiez que le cœur SMS_C fonctionne et est accessible
- Examinez les règles réseau/pare-feu
- Vérifiez la configuration du frontal

Solution :

```
# Tester la connectivité API depuis le frontal
curl http://localhost:4000/api/status

# Redémarrer le frontal
export API_BASE_URL="http://localhost:4000"
# Démarrer l'application frontal
```

Messages Non Livrés

Symptômes : Les messages restent non livrés, les tentatives de réessai s'incrémentent

Vérifiez :

1. Logs du frontal pour les erreurs d'envoi
2. Connectivité réseau externe
3. Configuration du frontal (identifiants, adresses)
4. Compatibilité du format de message

Voir les messages échoués :

```
# Obtenir les messages avec tentatives de réessai
curl https://localhost:8443/api/messages | jq '.data[] |
select(.delivery_attempts > 0)'
```

Latence Élevée des Messages

Symptômes : Messages prenant plus de temps que prévu, arriéré dans la file

Vérifiez :

1. Intervalle de sondage du frontal (peut n��cessiter une réduction pour un sondage plus fréquent)
2. Performance de la base de données
3. Latence réseau vers les systèmes externes

Surveiller la profondeur de la file :

```
watch -n 5 'curl -s https://localhost:8443/api/messages | jq ".data |
```

```
length"'
```

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`)

Benchmarks le point de terminaison API `submit_message_raw` en utilisant de véritables PDUs SMS.

Fonctionnalités :

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

Utilisation :

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

Sortie : `benchmarks/output/raw_sms_benchmark.html`

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

Benchmarks 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.

Aperçu 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
- **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
- Routage géographique
- Équilibrage de charge
- Configuration ENUM/NAPTR
- Chargement OCS
- Traduction de numéro

Surveillance et alertes

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

Aperçu 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

- **Clustering multi-nœuds** - Traitement de messages distribué
- **Réplication Mnesia** - Synchronisation des routes entre les nœuds
- **Basculement automatique** - Gestion des pannes de nœuds
- **Équilibrage de charge** - Distribution de routes pondérées

Documentation connexe

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

Exigences 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 ultérieur
- **Elixir**: 1.15.x ou ultérieur
- **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)
- **4369** - Mapper de port Erlang (clustering)
- **9100-9200** - Distribution Erlang (clustering)
- **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 API

Diagnostics

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

Obtenir de l'aide

1. Consultez le Guide de dépannage
2. Examinez 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 :

- **Exemples de configuration** montrent des valeurs typiques ; ajustez pour votre environnement
- **Exemples d'API** utilisent le format de ligne de commande `curl`
- **Adresses IP et domaines** sont des exemples seulement ; remplacez par vos valeurs réelles
- **Noms de métriques** suivent les conventions de nommage de 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

Retours sur la documentation

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

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 à la soumission, au routage et aux fonctions 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 contrôle 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 de l'équilibreur 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 des 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 des messages en attente incluant des 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 (Synchronisé)

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` - Programmer 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 Créé) :

```
{
  "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 (Asynchrone)

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 synchronisé

Réponse (202 Accepté) :

```
{
  "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 volume élevé (> 100 msg/sec)
- Pas besoin de l'ID du message dans la réponse API
- Le débit est plus important que la 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 la 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 retour en arrière exponentiel.

Requête :

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

Réponse (200 OK) :

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

Calcul de Retour en Arrière :

```
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 Pas de Contenu)

Exemple :

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

Avertissement : La suppression des messages les retire définitivement. Utiliser avec précaution.

API PDU SMS Brut

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

Soumettre un SMS Brut (Synchronisé)

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 Créé) :

```
{
  "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 un SMS Brut (Asynchrone)

Requête :

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

Corps : Identique à l'endpoint synchronisé

Réponse (202 Accepté) :

```
{
  "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 effectue automatiquement :

1. Décode 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 vers MSISDN si nécessaire
5. Applique les règles de routage
6. Stocke le PDU original pour référence

Détection des Rapports 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 mobiles terminés.

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 une 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 une Localisation

Crée une nouvelle localisation ou met à jour une 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 de cellule/secteur
- `imei` - Identifiant de l'appareil
- `ims_capable` - Capacité VoLTE IMS
- `csfb` - Indicateur de retour à circuit
- `registered` - Actuellement enregistré
- `expires` - Expiration de l'enregistrement
- `user_agent` - Modèle/info de l'appareil

Réponse (201 Créé 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 une 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 une Localisation

Requête :

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

Réponse (204 Pas de Contenu)

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 des Statistiques de 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 Créé) :

```
{
  "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 un statut actif

API de Journalisation des 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` - Rejeté 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 Créé) :

```
{
  "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) : Similaire 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 Créé) : Objet de message MMS complet

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 Créé) : Objet d'événement complet

Codes d'Erreur

Codes de Statut HTTP

Code	Signification	Description
200	OK	Requête réussie
201	Créé	Ressource créée avec succès
202	Accepté	Requête acceptée pour traitement
204	Pas de Contenu	Suppression réussie
400	Mauvaise Requête	Format de requête invalide
401	Non Autorisé	Authentification requise
403	Interdit	Permissions insuffisantes
404	Non Trouvé	La ressource n'existe pas
422	Entité Non Traitable	Erreurs de validation
429	Trop de Requêtes	Limite de taux dépassée
500	Erreur Interne du Serveur	Erreur du serveur
503	Service Indisponible	Temporairement indisponible

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 la 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 Trop de Requêtes) :

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

Meilleures Pratiques

Soumission de Message

1. **Utiliser Async pour le Bulk** : Utiliser `/create_async` pour > 100 msg/sec
2. **Inclure source_smsc** : Toujours identifier 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. **Mise en Cache des Connexions** : Réutiliser les connexions HTTP
2. **Requêtes en Lot** : 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 les clés régulièrement
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 les 4xx** : Les erreurs client nécessitent des corrections de code
3. **Retour en Arrière Exponentiel** : Attendre plus longtemps entre les réessais
4. **Disjoncteur** : Arrêter après des échecs répétés
5. **Alerte sur les Modèles** : Surveiller les taux d'erreur

Intégration Exemple (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 (asynchrone)
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 la localisation
- Enregistrement de frontend
- Journalisation des événements

Fonctionnalités Prévues

- Soumission de messages en lot (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 la table

MySQL / MariaDB

```sql
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 (ex : +15551234567).
`called_number`	VARCHAR(255)	NON	MSISDN de destination (numéro de mobile) du destinataire du message. Typiquement au format E.164 (ex : +15551234567).

Routage SMSC

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

Informations sur le nœud

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

Champ	Type	Nullable	Description
`origin_node`	VARCHAR(255)	OUI	Nom du nœud Erlang où le message a été initialement reçu (ex : "sms@node1.example.com"). Utile

Champ	Type	Nullable	Description
			pour le dépannage et l'analyse de distribution de charge.
`destination_node`	VARCHAR(255)	OUI	Nom du nœud Erlang d'où le message a été livré (si différent de l'origine). NULL pour les déploiements à nœud unique ou si le message n'a jamais été livré.

Horodatages

Tous les horodatages sont stockés en UTC :

Champ	Type	Nullable	Description
`submission_time`	DATETIME	NON	Quand le message a été d'abord soumis au SMS-C. Utilisé comme heure de début pour les calculs de facturation.
`delivery_time`	DATETIME	OUI	Quand le message a été livré avec succès. NULL si le message a expiré, échoué ou a été rejeté.
`expiry_time`	DATETIME	OUI	Quand le message a expiré (devenu non livrable). NULL si le message a été livré ou est 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 partie unique, 2+ pour multi-part. NULL si inconnu.
`deadletter`	BOOLEAN	NON	Indique si le message a été déplacé vers la file de lettres mortes. TRUE indique que le message n'a pas pu être livré et a épuisé toutes les

Champ	Type	Nullable	Description
			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 de livraison permanent (numéro invalide, etc.)	Dépend de la politique de facturation	NULL
`rejected`	Rejeté par des règles de routage ou validation	Non	NULL

Corps de message

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

Modes de confidentialité :

- **Conservation complète :** Corps de message stocké dans CDR pour conformité/archivage
- **Mode de confidentialité :** Corps de message défini sur NULL lorsque `delete_message_body_after_delivery: true`
- **Mode de conformité :** Corps stocké chiffré ou haché (nécessite une mise en œuvre personnalisée)

Horodatages d'audit

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

Exemples SQL

Requêtes de base

Trouver tous les CDR pour un numéro de téléphone spécifique :

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

Compter les messages par statut :

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

Temps de livraison moyen pour les messages livrés :

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

Requêtes de facturation

Volume de messages quotidien par SMSC de destination :

```
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) :

```
SELECT
  DATE(submission_time) AS date,
  COUNT(*) AS message_count,
  SUM(message_parts) AS total_segments,
```

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

Analyse de performance de routage :

```
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) :

```
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 :

```
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 :

```
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
!)
```

```sql
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 :

```sql
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 un enregistrement unique
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 de routage

Recommandations d'index supplémentaires

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

Index composite pour les requêtes de facturation :

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

Index composite pour l'analyse de routage :

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

Index composite pour les recherches de conformité :

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

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

```
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 prises en charge :

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

Considérations sur la vie privée

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

1. Confidentialité du corps de message

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

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

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

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

2. Masquage des numéros de téléphone

Pour 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 (pas d'accès au corps du
message)
CREATE USER 'analytics'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT (id, message_id, calling_number, called_number,
```

```
source_smsc,
              dest_smsc, submission_time, delivery_time, status,
              delivery_attempts, message_parts)
ON sms_c.cdrs TO 'analytics'@'%';
```

Conservation et archivage

Politiques de conservation

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

Industrie	Conservation typique	Base réglementaire
Télécom (US)	18-24 mois	FCC, lois 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 - À 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 carte tarifaire

Créer une table de tarifs séparée pour la facturation :

```
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'),
('+', '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

Exportation 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 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 CDR SQL

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 Pour
MySQL	8.0+	`Ecto.Adapters.MyXQL`	3306	Usage général, fiabilité prouvée
MariaDB	10.5+	`Ecto.Adapters.MyXQL`	3306	Compatible MySQL, open source
PostgreSQL	13+	`Ecto.Adapters.Postgres`	5432	Fonctionnalités avancées, support JSON

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

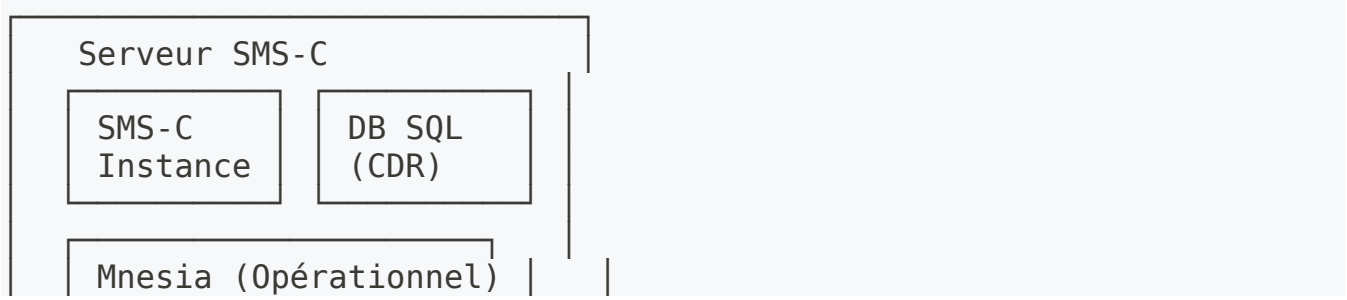
PostgreSQL - À considérer si vous avez besoin de :

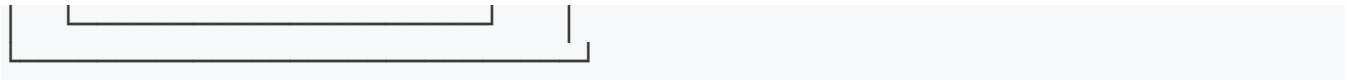
- Fonctionnalités avancées JSON/JSONB pour l'analyse
- Requêtes complexes sur les données CDR
- Infrastructure PostgreSQL existante
- PostGIS pour l'analyse géographique

Topologies de Déploiement

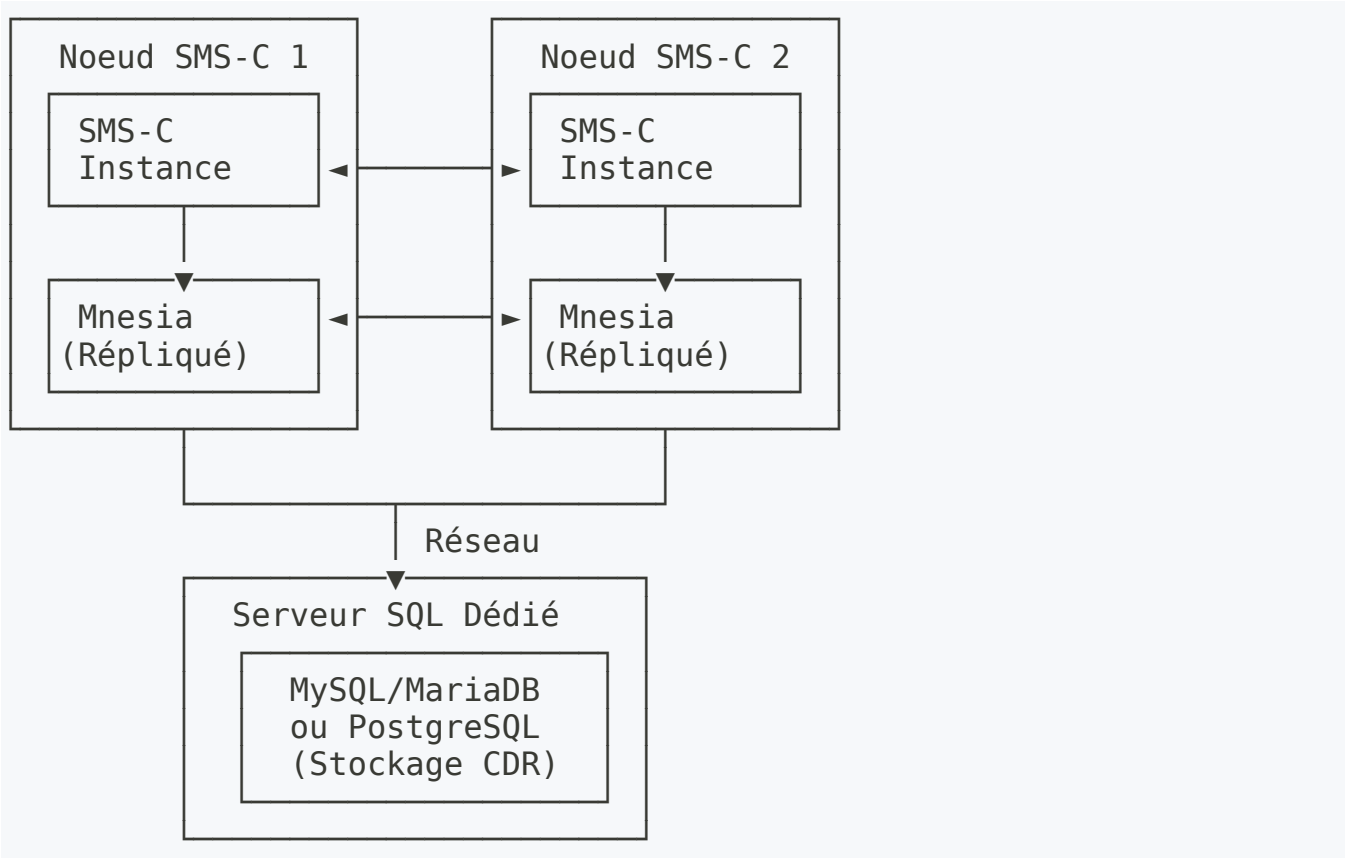
Important : La base de données SQL CDR peut fonctionner sur un **serveur séparé** de votre instance SMS-C. C'est l'approche recommandée pour les déploiements en production.

Déploiement sur un Serveur Unique (Développement/Test) :





Déploiement Distribué (Production - Recommandé) :



Avantages d'un Serveur SQL Séparé :

- **Isolation de Performance** : Les écritures 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** : Programmes 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

Charge de Travail	Taille de Pool	Description
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 de pool plus grand.

Configuration de l'API

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

Configuration de Base de l'API

```
# config/runtime.exs
config :api_ex,
  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 du Certificat API

Générez un certificat auto-signé pour les tests :

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

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

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

Contrôle d'Accès API

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

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

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

Authentification par Clé API (Niveau Application) :

Configurez via un plug personnalisé dans le routeur - voir le Guide des Opérations pour les détails d'implémentation.

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 du Proxy Inverse (Recommandé)

Utilisez Nginx ou Apache comme proxy inverse pour plus de sécurité et de fonctionnalités :

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 plus de sécurité
  auth_basic "Administrateur SMS-C";
  auth_basic_user_file /etc/nginx/.htpasswd;

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

  # Support WebSocket pour LiveView
  location /live {
    proxy_pass http://sms_web;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_read_timeout 86400;
  }
}
```

Configuration du Cluster

Le SMS-C prend en charge le clustering multi-nœuds pour une haute disponibilité et une distribution de charge.

Configuration d'un Nœud Unique

```
# config/runtime.exs
config :sms_c,
  cluster_nodes: [],  # Liste vide = mode nœud unique
  smsc_node_name: "node1"
```

Cluster Statique Multi-Nœuds

```
# Nœud 1 : config/runtime.exs
config :sms_c,
  cluster_nodes: [
    :"sms@node1.internal.example.com",
    :"sms@node2.internal.example.com",
    :"sms@node3.internal.example.com"
  ],
  smsc_node_name: "node1"

# Nœud 2 : config/runtime.exs
config :sms_c,
  cluster_nodes: [
    :"sms@node1.internal.example.com",
    :"sms@node2.internal.example.com",
    :"sms@node3.internal.example.com"
  ],
  smsc_node_name: "node2"
```

Découverte Automatique Basée sur DNS

```
config :sms_c,
  dns_cluster_query: "sms-cluster.internal.example.com",
  smsc_node_name: System.get_env("NODE_NAME") || "node1"
```

Configuration DNS pour la Découverte Automatique :

```
# Configurer les enregistrements SRV ou A pour les nœuds du cluster
# Enregistrement SRV (préféré) :
_sms._tcp.sms-cluster.internal.example.com. IN SRV 0 0 0
node1.internal.example.com.
_sms._tcp.sms-cluster.internal.example.com. IN SRV 0 0 0
node2.internal.example.com.
_sms._tcp.sms-cluster.internal.example.com. IN SRV 0 0 0
```

```
node3.internal.example.com.

# Enregistrements A (alternatif) :
sms-cluster.internal.example.com. IN A 10.0.1.10
sms-cluster.internal.example.com. IN A 10.0.1.11
sms-cluster.internal.example.com. IN A 10.0.1.12
```

Configuration de Distribution Erlang

Démarrer les Nœuds avec des Noms Appropriés :

```
# Nœud 1
export NODE_NAME=sms@node1.internal.example.com
export ERLANG_COOKIE=shared_secret_cookie_here
elixir --name $NODE_NAME --cookie $ERLANG_COOKIE -S mix phx.server

# Nœud 2
export NODE_NAME=sms@node2.internal.example.com
export ERLANG_COOKIE=shared_secret_cookie_here
elixir --name $NODE_NAME --cookie $ERLANG_COOKIE -S mix phx.server
```

Important : Tous les nœuds d'un cluster DOIVENT utiliser le même cookie Erlang pour des raisons de sécurité.

Exigences Réseau du Cluster

Ouvrez ces ports entre les nœuds du cluster :

Plage de Ports	Protocole	Objectif
4369	TCP	Daemon de Cartographie de Ports Erlang (EPMD)
9100-9200	TCP	Distribution Erlang

Exemple de Configuration de Pare-feu :

```
# Autoriser le trafic de cluster depuis le réseau interne
iptables -A INPUT -p tcp -s 10.0.0.0/8 --dport 4369 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.0.0/8 --dport 9100:9200 -j ACCEPT
```

Configuration de la File de Messages

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

Expiration des Messages

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

Valeurs Courantes :

- **60** - 1 heure (test/développement)
- **1440** - 24 heures (production typique)
- **4320** - 3 jours (rétention prolongée)
- **10080** - 7 jours (rétention maximale)

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

Configuration de Réessai de Livraison

Le comportement de réessai utilise un retour 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 Facturation

Intégration avec OCS pour la facturation 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 ENUM

Recherches de numéro 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 (fallback)
    {"1.1.1.1", 53}     # DNS Cloudflare (fallback)
  ],
  enum_timeout: 3000  # 3 secondes (échec plus rapide)
```

Priorité des Domaines ENUM

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

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

Optimisation des Performances ENUM

Pour les Réseaux à Faible Latence :

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

Pour les Liens à Haute Latence/Satellite :

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

Exemple de Configuration DNS ENUM

Configurer la 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 Traduction de Numéros

Normalisation des numéros basée sur des regex 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})$",           # Correspondre aux
numéros de 10 chiffres
      calling_replace: "+1\1",                # Prépend +1
      called_match: "^(\d{10})$",
      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 aux Transporteurs

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'appel
  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 (numéro inférieur = priorité plus élevée) :

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

    # Priorité 50 : Règles générales
    %{
      calling_prefix: nil,
      called_prefix: nil,
      source_smsc: nil,
      calling_match: "^(\d{10})$",
      calling_replace: "+1\1",
      called_match: "^(\d{10})$",
      called_replace: "+1\1",
      priority: 50,
      description: "Normalisation générale de 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_prefix: nil,
      called_prefix: "+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,
      weight: 100,
      priority: 50,
      description: "Routage Amérique du Nord",
      enabled: true
    },

    # Exemple de routage équilibré
    %{
      calling_prefix: nil,
      called_prefix: "+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,
      weight: 70,
      priority: 50,
      description: "Passerelle principale du Royaume-Uni (70%)",
      enabled: true
    },
    %{
      calling_prefix: nil,
      called_prefix: "+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,
      weight: 30,
      priority: 50,
      description: "Passerelle de secours du Royaume-Uni (30%)",
      enabled: true
    }
  ]
```

Ignorer le Chargement Initial des Routes

```
# Ne pas charger les routes depuis la config (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 par Lots

```
# config/config.exs
config :sms_c,
  batch_insert_batch_size: 100,           # Messages par lot
  batch_insert_flush_interval_ms: 100     # Temps d'attente maximal
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 des Journaux

Niveaux de Journalisation

```
# config/config.exs
```

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

Recommandé en Production : `:info` ou `:warning` **Recommandé en Développement** : `:debug`

Destinations de Sortie des Journaux

Console 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

Optimiser 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 performance)
config :sms_c,
  default_charging_enabled: false

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

Messagerie en Temps Réel pour Entreprises

Optimiser 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

Optimiser 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-region pour DR
  ],
  smsc_node_name: "us-east-1a"
```

Validation de la Configuration

Tester la configuration avant le déploiement :

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

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

# 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	Objectif	Exemple
DB_USERNAME	Nom d'utilisateur de la base de données	sms_prod_user
DB_PASSWORD	Mot de passe de la base de données	strong_password
DB_HOSTNAME	Hôte de la base de données	db.internal.example.com
DB_PORT	Port de la base de données	3306
DB_NAME	Nom de la base de données	sms_c_production
DB_POOL_SIZE	Taille du pool de connexions	30
API_PORT	Port d'écoute de l'API	8443
API_LISTEN_IP	IP d'écoute de l'API	0.0.0.0
WEB_PORT	Port de l'interface Web	443
NODE_NAME	Nom du nœud Erlang	sms@node1.example.com
ERLANG_COOKIE	Secret de cluster	shared_cookie_value
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 des valeurs sensibles (mots de passe, clés API)
2. **Testez les Changements de Configuration** en staging avant la production
3. **Documentez les Paramètres Personnalisés** dans les notes de déploiement
4. **Contrôlez les Fichiers de Config** (exclure les secrets)
5. **Surveillez Après les Changements** pour des régressions de performance
6. **Conservez des Sauvegardes** des configurations fonctionnelles
7. **Validez Avant le Redémarrage** 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 config	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 config 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 : ~1 Ko
- 10,000 messages : ~10 Mo
- 100,000 messages : ~100 Mo

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 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 confidentialité.

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

  # Cacher le corps du message dans l'interface web
  hide_message_body_in_ui: false,

  # Cacher 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 confidentialité
`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

Exemples de Configurations :

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 journalisé :

```
[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.

Documentation des métriques Prometheus SMS-C

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

Aperçu

Ce document décrit toutes les métriques Prometheus exposées par le système SMS-C. Ces métriques sont conçues pour que le personnel opérationnel puisse surveiller la santé du système, les performances et 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 :

- `message` - Métriques de traitement des messages
- `routing` - Métriques de décision de routage
- `enum` - Métriques de recherche ENUM/NAPTR
- `delivery` - Métriques de livraison des messages
- `queue` - Métriques de gestion de la file d'attente
- `charging` - Métriques de facturation
- `mnesia` - Métriques de base de données
- `frontend` - Métriques de connexion frontend
- `location` - Métriques de localisation/enregistrement
- `phoenix.endpoint` - Métriques de requêtes API HTTP
- `vm` - Métriques système de la VM Erlang

Métriques de traitement des messages

sms_c_message_received_count

Type : Compteur

Description : Nombre total de messages reçus par le SMS-C de toutes les sources.

Labels :

- `source_smsc` : Nom de la source SMSC 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 : Compteur

Description : Nombre total de validations de messages effectuées.

Labels :

- `valid` : Si la validation a réussi (vrai ou faux)

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

Alertes : Alerter lorsque le taux d'échec de validation dépasse le seuil (par exemple, > 5 % d'échecs).

---

sms_c_message_processing_stop_duration

Type : Histogramme

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` : Si le traitement a réussi (vrai ou faux)

Cas d'utilisation : Surveiller les performances 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 : Compteur

Description : Nombre total de fois qu'une route spécifique a été correspondue et sélectionnée pour le routage des messages.

Labels :

- `route_id` : Identifiant unique de la route correspondue
- `dest_smsc` : SMSC de destination sélectionné par la route
- `priority` : Valeur de priorité de la route correspondue

Cas d'utilisation : Comprendre quelles routes sont utilisées le plus fréquemment. Identifier les routes sous-utilisées ou surchargées. Utile pour la planification de capacité et l'optimisation des routes.

Alertes : Alerter si les routes à haute priorité sont rarement correspondues (peut indiquer une mauvaise configuration de routage).

---

sms_c_routing_failed_count

Type : Compteur

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 ils indiquent que les messages ne peuvent pas être livrés.

---

sms_c_routing_action_count

Type : Compteur

Description : Nombre total d'actions de routage spéciales effectuées.

Labels :

- `action` : Type d'action (drop, auto_reply, forward)
- `route_id` : Route qui a déclenché l'action

Cas d'utilisation : Surveiller les règles de suppression (anti-spam), l'utilisation des réponses automatiques et les modèles de transfert.

Alertes : Alerter sur des pics inattendus dans les actions de suppression (peut indiquer une attaque de spam).

---

sms_c_routing_stop_duration

Type : Histogramme

Description : Temps nécessaire pour évaluer toutes les routes et sélectionner la meilleure correspondance.

Unité : Millisecondes

Seaux : 1, 5, 10, 25, 50, 100, 250, 500 ms

Labels :

- `dest_smsc` : SMSC de destination sélectionné

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

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 : Compteur

Description : Nombre total de recherches ENUM servies à partir du cache (n'a pas nécessité de requête DNS).

Labels :

- `domain` : Domaine ENUM interrogé

Cas d'utilisation : Surveiller l'efficacité du cache. Des taux de réussite élevés réduisent la charge DNS et améliorent les performances.

Alertes : Alerter si le taux de réussite du cache tombe en dessous du seuil (peut indiquer des problèmes de cache ou un trafic inhabituel).

---

sms_c_enum_cache_miss_count

Type : Compteur

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 nombre de réussites 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'il 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 : Histogramme

Description : Temps nécessaire pour compléter une recherche ENUM (y compris la requête DNS si non mise en cache).

Unité : Millisecondes

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

Labels :

- `domain` : Domaine ENUM interrogé
- `success` : Si la recherche a réussi (vrai ou faux)
- `cache_hit` : Si le résultat a été servi à partir du cache (vrai ou faux)

Cas d'utilisation : Surveiller les performances 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 : Histogramme

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 : Compteur

Description : Nombre total de messages mis en file d'attente pour livraison à un 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 : Compteur

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 de messages mis en file d'attente indique un comportement de réessai.

---

sms_c_delivery_succeeded_count

Type : Compteur

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. Métrique de succès principale.

Alertes : Alerter si le taux de succès tombe en dessous du seuil SLA.

Calcul : `success_rate = succeeded / queued`

---

sms_c_delivery_failed_count

Type : Compteur

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 : Compteur

Description : Nombre total de messages déplacés vers la file d'attente des lettres mortes (non livrables).

Labels :

- `reason` : Raison pour 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 ils représentent un échec complet de livraison.

---

sms_c_delivery_succeeded_duration

Type : Histogramme

Description : Temps de bout en bout du message mis en file d'attente à la livraison réussie.

Unité : Millisecondes

Seaux : 100, 500, 1000, 5000, 10000, 30000, 60000 ms

Labels :

- `dest_smsc` : Nom de la SMSC de destination

Cas d'utilisation : Surveiller la latence de livraison. Identifier les destinations lentes ou les problèmes de réseau.

Alertes : Alerter lorsque le temps de livraison p95 dépasse les seuils SLA.

---

sms_c_delivery_succeeded_attempt_count

Type : Histogramme

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 dès la première tentative.

Alertes : Alerter si le nombre moyen de tentatives dépasse 2 (indique des problèmes de fiabilité de destination).

---

sms_c_delivery_failed_attempt_count

Type : Histogramme

Description : Nombre de tentatives de livraison effectuées avant l'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.

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/suppression 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 : Compteur

Description : Nombre total de demandes de facturation effectuées à l'OCS ou au système de facturation.

Labels :

- `account` : Identifiant du compte facturé

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

---

sms_c_charging_succeeded_count

Type : Compteur

Description : Nombre total d'opérations de facturation réussies.

Labels :

- `account` : Identifiant du compte facturé

Cas d'utilisation : Surveiller le taux de succès de facturation par compte.

Calcul : `success_rate = succeeded / requested`

---

sms_c_charging_failed_count

Type : Compteur

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 : Histogramme

Description : Temps nécessaire pour compléter une demande de facturation réussie.

Unité : Millisecondes

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

Labels :

- `account` : Identifiant du compte

Cas d'utilisation : Surveiller les performances du système de facturation. Une facturation lente peut retarder la livraison des messages.

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.

Labels :

- `table` : Nom de la table (par exemple, `sms_route`)

Cas d'utilisation : Surveiller la croissance de la base de données. Détecter une accumulation de données inattendue.

Alertes : Alerter sur des taux de croissance de table inattendus.

---

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 (connecté, déconnecté)

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

Alertes : Alerter lorsque des frontends attendus se déconnectent.

---

sms_c_location_registered_count

Type : Compteur

Description : Nombre total d'enregistrements de localisation/abonnés reçus par le système.

Labels :

- `location` : Nom du frontend/SMSC où l'abonné est enregistré
- `ims_capable` : Si l'abonné prend en charge l'IMS (vrai/faux)

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

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]))
```

Métriques de requêtes API HTTP

phoenix_endpoint_stop_duration

Type : Distribution (Histogramme)

Description : Durée de traitement des requêtes HTTP en millisecondes, du début de la requête à la fin de la réponse.

Labels :

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

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

Cas d'utilisation : Surveiller les performances de l'API. Identifier les endpoints lents. Suivre les SLA de temps de réponse.

Alertes : Définir des alertes pour :

- Latence P95 > 500ms pour les endpoints critiques
- Latence P99 > 1s pour tout endpoint
- Tendances de latence croissantes

Exemple de requête :

```
# Temps de réponse P95 par endpoint
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 : Compteur

Description : Nombre total de requêtes HTTP complétées, classées par route et code d'état HTTP.

Labels :

- `route` : Route de l'endpoint 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 succès. Suivre les taux d'erreur par endpoint.

Alertes : Définir des alertes pour :

- Taux d'erreur > 5 % pour tout endpoint
- Erreurs 5xx sur des endpoints critiques
- Baisses soudaines du volume des requêtes

Exemple de requête :

```
# Taux de requêtes par endpoint
sum by (route) (rate(phoenix_endpoint_stop_count[5m]))

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

# Taux de succès
sum(rate(phoenix_endpoint_stop_count{status=~"2.."}[5m])) /
sum(rate(phoenix_endpoint_stop_count[5m]))
```

---

phoenix_router_dispatch_exception_count

Type : Compteur

Description : Nombre total d'exceptions/erreurs levées lors du traitement des requêtes HTTP.

Labels :

- `route` : Route de l'endpoint API où l'exception s'est produite
- `kind` : Type d'exception (error, exit, throw)

Cas d'utilisation : Suivre les erreurs d'application. Identifier les endpoints problématiques. Surveiller la stabilité du système.

Alertes : Définir des alertes pour toute valeur non nulle sur des endpoints critiques.

Exemple de requête :

```
# Taux d'exception par endpoint
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 cela dépasse systématiquement 10 * le nombre de cœurs CPU.

---

vm_system_counts_process_count

Type : Gauge

Description : Nombre actuel de processus en cours d'exécution dans la VM.

Cas d'utilisation : Surveiller les modèles de création de processus. Détecter les fuites de processus.

Alertes : Alerter lorsque l'on approche de la limite de processus (par défaut 262,144).

Collecte et sondage des métriques

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

- Tailles et âges des files d'attente
- Tailles des tables Mnesia
- Statistiques du cache ENUM

Toutes les autres métriques sont déclenchées par des événements et émises lorsque l'action correspondante se produit.

Modèles de surveillance courants

Taux de succès de livraison par destination

Suivre le taux de succès de la livraison des messages pour chaque SMSC de destination :

Formule : `(sms_c_delivery_succeeded_count) / (sms_c_delivery_queued_count)`

Interprétation : Devrait être > 95 % pour des destinations saines. Des taux plus bas indiquent des problèmes de livraison.

Latence de message de bout en bout

Surveiller le temps total de la réception du message à la livraison :

Métriques :

- `sms_c_message_processing_stop_duration` (traitement)
- `sms_c_delivery_succeeded_duration` (livraison)

Interprétation : La somme représente la latence totale perçue par l'utilisateur.

Efficacité du cache ENUM

Mesurer l'efficacité du cache ENUM :

Formule : `(sms_c_enum_cache_hit_count) / (sms_c_enum_cache_hit_count + sms_c_enum_cache_miss_count)`

Interprétation : Devrait être > 80 % après la montée en température. Des taux plus bas peuvent indiquer un TTL court ou une variance de trafic élevée.

Utilisation des routes

Identifier quelles routes gèrent le plus de trafic :

Métrique : `sms_c_routing_route_matched_count` regroupée par `route_id`

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

Tendance de l'arriéré de la file d'attente

Surveiller si la file d'attente de messages est en croissance (arriéré) ou en diminution (rattrapage) :

Métriques :

- `sms_c_queue_size_pending` (en attente actuelle)
- `sms_c_queue_oldest_message_age_seconds` (tendance d'âge)

Interprétation : Compte en attente croissant + âge croissant = formation d'un arriéré.

Taux de réessai

Comprendre à quelle fréquence des réessais de livraison sont nécessaires :

Métrique : `sms_c_delivery_succeeded_attempt_count` histogramme des percentiles

Interprétation : Si p95 > 1, la plupart des messages nécessitent des réessais. Indique des problèmes de fiabilité de destination.

Alertes recommandées

Alerte	Condition	Gravité	Description
Taux d'échec de routage élevé	`routing_failed_count` augmentation	Critique	Les messages ne peuvent pas être routés
Arriéré de file d'attente	`queue_size_pending` > seuil	Avertissement	Messages s'accumulant
Anciens messages dans la file d'attente	`queue_oldest_message_age_seconds` > 300	Critique	Violation SLA
Pic d'échecs de livraison	`delivery_failed_count` pic	Élevé	Problèmes de destination
Événements de lettre morte	`delivery_dead_letter_count` > 0	Élevé	Messages non livrables
Délais d'attente de recherche ENUM	`enum_lookup_stop_duration` p95 > 5000ms	Avertissement	Problèmes DNS
Faible taux de réussite du cache	Taux de réussite du cache ENUM < 0.7	Avertissement	Cache inefficace
Frontend déconnecté	`frontend_status_count{status="disconnected"}` > 0	Élevé	Perte de connectivité
Échecs de facturation	`charging_failed_count` > seuil	Élevé	Problèmes de facturation
Traitement de message lent	`message_processing_stop_duration` p95 > 1000ms	Avertissement	Dégradation des performances

---

Recommandations de tableau de bord

Tableau de bord des opérations

Objectif : Surveillance de la santé du système en temps réel

Panneaux :

1. Débit de messages (reçus/traités/livrés par minute)
2. Tailles de file 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 des performances du système

Panneaux :

1. Histogramme de la durée de traitement des messages
2. Histogramme de la durée de routage
3. Histogramme de la durée de recherche ENUM
4. Histogramme de la durée de facturation
5. Distribution des tentatives de livraison
6. Taux de réussite du cache

Tableau de bord commercial

Objectif : Analyse du trafic et de l'utilisation

Panneaux :

1. Messages par SMSC source
2. Messages par SMSC de destination
3. Carte thermique d'utilisation des routes
4. Comptes d'actions de réponse automatique et de suppression
5. Statistiques d'utilisation ENUM
6. Volume de facturation par compte

Conservation des métriques

Paramètres de conservation recommandés pour Prometheus :

- **Métriques brutes** : 15 jours
- **Agrégats de 5 minutes** : 90 jours
- **Agrégats d'1 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_succeeded_duration` - 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 de 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_succeeded_duration_bucket)
```

Temps de livraison (p99) :

```
histogram_quantile(0.99, sms_c_delivery_succeeded_duration_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 tableaux de bord Grafana

Tableau de bord 1 : Opérations en temps réel

Objectif : Surveiller l'activité et la santé actuelles du système.

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 (Statistique)**
    - Requête : `sms_c_queue_oldest_message_age_seconds / 60`
    - Unité : minutes
    - Seuils :
        - Vert : < 5
        - Jaune : 5-10
        - Rouge : > 10

5. **Frontends actifs (Statistique)**
    - Requête : `sum(sms_c_frontend_status_count{status="connected"})`
    - Unité : compte
    - Couleur : Bleu
6. **Échecs de routage (Graphique)**
    - Requête : `rate(sms_c_routing_failed_count[5m]) * 60`
    - Unité : échecs/minute
    - Seuil d'alerte : > 0

Tableau de bord 2 : Analyse de performance

Objectif : Analyser les performances du système et identifier les goulets d'étranglement.

Panneaux :

1. **Latence de bout en bout (Graphique)**
    - Requête : `histogram_quantile(0.50, sms_c_message_processing_stop_duration_bucket)` (p50)
    - Requête : `histogram_quantile(0.95, sms_c_message_processing_stop_duration_bucket)` (p95)
    - Requête : `histogram_quantile(0.99, sms_c_message_processing_stop_duration_bucket)` (p99)
    - Unité : millisecondes
    - Légende : Percentile
2. **Latences des composants (Jauge à barres)**
    - Routage : `histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)`
    - ENUM : `histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)`
    - Facturation : `histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)`
    - Livraison : `histogram_quantile(0.95, sms_c_delivery_succeeded_duration_bucket)`
    - Unité : millisecondes
    - Barres horizontales
3. **Distribution des tentatives de livraison (Carte thermique)**
    - Requête : `sms_c_delivery_succeeded_attempt_count_bucket`
    - Montre combien de tentatives sont généralement nécessaires
    - Échelle de couleur : Bleu (1 tentative) à Rouge (plusieurs tentatives)
4. **Performance du cache ENUM (Graphique)**
    - Taux de réussite : `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`
    - Taille du cache : `sms_c_enum_cache_size_size`
    - Axe Y double (taux vs taille)
5. **Taux de succès de traitement (Jauge)**

- Requête : `(rate(sms_c_message_processing_stop_duration_count{success="true"}[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 (Diagramme circulaire)**
   - Requête : `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
   - Montre la distribution : IMS vs CS vs SMPP
2. **Messages par SMSC source (Graphique à barres)**
   - Requête : `sum by (source_smsc) (rate(sms_c_message_received_count[1h]))`
   - 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 aires empilées
   - Légende : {{dest_smsc}}
5. **Actions de suppression/réponse automatique (Statistique)**
   - Supprimés : `increase(sms_c_routing_action_count{action="drop"}[1h])`
   - Réponses automatiques : `increase(sms_c_routing_action_count{action="auto_reply"}[1h])`
   - Statistiques côte à côte
6. **Modèle de trafic horaire (Graphique)**
   - Requête : `rate(sms_c_message_received_count[1h]) * 3600`
   - Plage horaire : Derniers 7 jours
   - Montre les modèles quotidiens

Tableau de bord 4 : Capacité et ressources

Objectif : Surveiller l'utilisation des ressources et les limites de capacité.

Panneaux :

1. **Capacité de la file d'attente (Graphique)**
    - Actuel : `sms_c_queue_size_size`
    - Ligne de capacité : Valeur fixe basée sur les limites du système
    - Montre la tendance d'utilisation
2. **Croissance des tables de base de données (Graphique)**
    - Messages : `sms_c_mnesia_table_size_record_count{table="sms_route"}`
    - Traductions : `sms_c_mnesia_table_size_record_count{table="translation_rule"}`
    - Tendance sur les 30 derniers jours
3. **Tendance de l'arriéré 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 (Statistique)**
    - 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 les SLA (Statistique)**
    - Requête : `count(sms_c_delivery_succeeded_duration_bucket{le="5000"}) / count(sms_c_delivery_succeeded_duration_bucket)`
    - Montre le pourcentage livré dans les 5 secondes
    - Unité : pourcentage
3. **Violations SLA (Compteur)**
    - Messages dépassant 5 minutes : `increase(sms_c_queue_oldest_message_age_seconds{} > 300)[24h:]`
    - Devrait être 0
4. **Temps de disponibilité (Statistique)**
    - 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 d'attente des lettres mortes en croissance :

```
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 d'attente des lettres
mortes"
  description: "Les messages deviennent non livrables. Enquêtez sur les échecs."
```

Alertes d'avertissement

Faible taux de succès de livraison :

```
alert: LowDeliverySuccessRate
expr: (rate(sms_c_delivery_succeeded_count[10m]) /
rate(sms_c_delivery_queued_count[10m])) < 0.95
for: 10m
labels:
  severity: warning
annotations:
  summary: "Le taux de succès de livraison est {{ $value | humanizePercentage }}"
  description: "Le taux de succès est inférieur à 95 %. Enquêtez sur les échecs de
livraison."
```

Taux de réessai élevé :

```
alert: HighDeliveryRetryRate
expr: histogram_quantile(0.95, sms_c_delivery_succeeded_attempt_count_bucket) > 2
for: 15m
labels:
  severity: warning
annotations:
  summary: "Tentatives de livraison au 95e percentile : {{ $value }}"
```

```
  description: "Les messages nécessitant plusieurs tentatives. Vérifiez la
fiabilité de la destination."
```

Traitement de message lent :

```
alert: SlowMessageProcessing
expr: histogram_quantile(0.95, sms_c_message_processing_stop_duration_bucket) >
1000
for: 10m
labels:
  severity: warning
annotations:
  summary: "Temps de traitement au 95e percentile : {{ $value }}ms"
  description: "Le traitement des messages est lent. Vérifiez les ressources du
système."
```

Recherches ENUM échouant :

```
alert: HighEnumFailureRate
expr: rate(sms_c_enum_lookup_stop_duration_count{success="false"}[10m]) > 0.1
for: 10m
labels:
  severity: warning
annotations:
  summary: "Taux d'échec de recherche ENUM : {{ $value }}"
  description: "Les recherches DNS échouent. Vérifiez les serveurs DNS."
```

Faible taux de réussite du cache ENUM :

```
alert: LowEnumCacheHitRate
expr: rate(sms_c_enum_cache_hit_count[10m]) /
(rate(sms_c_enum_cache_hit_count[10m]) + rate(sms_c_enum_cache_miss_count[10m])) <
0.70
for: 30m
labels:
  severity: warning
annotations:
  summary: "Taux de réussite du cache ENUM : {{ $value | humanizePercentage }}"
  description: "Efficacité du cache faible. Peut indiquer un trafic de numéros
uniques."
```

Échecs de facturation :

```
alert: ChargingFailuresDetected
expr: rate(sms_c_charging_failed_count[10m]) > 0.05
for: 10m
labels:
  severity: warning
annotations:
  summary: "Taux d'échec de facturation : {{ $value }}"
  description: "Erreurs du système de facturation. Vérifiez la connectivité OCS."
```

Notes supplémentaires

- Toutes les métriques de durée utilisent une précision en nanosecondes en interne mais sont converties en millisecondes pour le reporting
- Les métriques de compteur sont cumulatives et doivent être utilisées avec les fonctions

`rate()` ou `increase()` dans les requêtes Prometheus
- Les métriques de gauge représentent des valeurs instantanées au moment de la collecte
- Les métriques d'histogramme fournissent des calculs de percentile (p50, p95, p99) et peuvent être utilisées pour créer des cartes thermiques
- Toutes les métriques incluent des labels par défaut ajoutés par Prometheus (instance, job, etc.)
- Lors de la création de tableaux de bord, utilisez des plages horaires 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 des opérations SMS-C

← Retour à l'index de 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 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 de lettres mortes (enquêter sur ceux présents)

4. Examiner l'état du frontend

Accédez à l'interface Web : `https://sms-admin.example.com/frontend_status`

Vérifiez :

- Tous les frontends attendus sont actifs
- Pas de déconnexions non expirées
- Pas d'erreurs de frontend au cours des 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

Vérifier les comptes de messages horaires :

Utilisez la requête Prometheus :

```
# Messages reçus par heure
increase(sms_c_message_received_count[1h])

# Messages livrés par heure
increase(sms_c_delivery_succeeded_count[1h])

# Calculer le taux de livraison
rate(sms_c_delivery_succeeded_count[1h]) /
rate(sms_c_message_received_count[1h])
```

Modèles attendus :

- Heures de bureau : Volume plus élevé
- Nuits/week-ends : Volume plus faible
- Taux de livraison : Doit être > 95%

Conditions d'alerte :

- Chute soudaine des messages (> 50% de diminution)

- Pic soudain des messages (> 200% d'augmentation)
- Taux de livraison tombant en dessous de 90%

Surveillance

Métriques clés à surveiller

Métriques de traitement des messages

Nombre de messages reçus (`sms_c_message_received_count`) :

- **Quoi** : Total des messages entrant dans le système
- **Alerte** : Chute ou pic soudain
- **Requête** : `rate(sms_c_message_received_count[5m])`

Durée de traitement des messages (`sms_c_message_processing_stop_duration`) :

- **Quoi** : Temps de traitement de bout en bout
- **Alerte** : p95 > 1000ms
- **Requête** : `histogram_quantile(0.95, sms_c_message_processing_stop_duration)`

Métriques de routage

Échecs de routage (`sms_c_routing_failed_count`) :

- **Quoi** : Messages qui n'ont pas pu être routés
- **Alerte** : Tout échec (> 0)
- **Requête** : `increase(sms_c_routing_failed_count[5m])`

Route correspondante (`sms_c_routing_route_matched_count`) :

- **Quoi** : Quelles routes sont utilisées
- **Alerte** : Routes prioritaires non correspondantes
- **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 le plus ancien en attente
- **Alerte** : Âge > 300 secondes
- **Requête** : `sms_c_queue_oldest_message_age_seconds`

Configuration du tableau de bord

Panneaux du tableau de bord opérationnel :

1. **Débit 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 la route
    - 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 la base de données (p95)

- Temps de recherche ENUM (p95)

Configuration des alertes

Alertes critiques (Réponse immédiate requise) :

```
# Aucune route trouvée - 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 au cours des 5
dernières minutes"

# Accumulation de la file d'attente - 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: "Tentatives de livraison au 95e percentile : {{ $value }}"

# Recherches ENUM lentes ou échouées
- alert: SlowEnumLookups
  expr: histogram_quantile(0.95, sms_c_enum_lookup_stop_duration) > 5000
  severity: warning
  description: "Recherches ENUM prenant > 5 secondes"

# Faible taux de réussite du cache ENUM
- alert: LowEnumCacheHitRate
```

```
  expr: rate(sms_c_enum_cache_hit_count[10m]) /
(rate(sms_c_enum_cache_hit_count[10m]) +
rate(sms_c_enum_cache_miss_count[10m])) < 0.70
  severity: warning
  description: "Taux de réussite du cache ENUM : {{ $value }}"
```

Suivi des messages

Trouver un message spécifique

Par ID de message :

1. **Interface Web** : Accédez à `/message_queue`
2. Entrez l'ID du message dans la boîte de recherche
3. Voir les détails complets et l'historique des événements

Via API :

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

Par numéro de téléphone :

1. **Interface Web** : Accédez à `/message_queue`
2. Entrez le numéro de téléphone dans la boîte de recherche
3. Voir tous les messages pour ce numéro

Suivre le cycle de vie du 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 - Nombres 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 la 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 interroge :

```
# Le frontend interroge pour des messages
GET /api/messages/queue?smsc=uk_gateway

# Renvoie le message même si dest_smsc est null
# parce que l'abonné de destination est enregistré à uk_gateway
```

Exigences de localisation :

Pour que le routage basé sur la localisation fonctionne :

- La table `locations` doit avoir une entrée pour le `destination_msisdn`
- Le champ `location` doit correspondre au nom du SMSC interrogeant
- L'horodatage `expires` doit être dans le futur

Surveillance du routage basé sur la localisation :

Vérifiez les enregistrements de localisation :

```
# Via API
GET /api/locations/{msisdn}

# Vérifiez si la localisation est expirée
# le champ expires doit être > heure actuelle
```

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 :

```
# Route 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 routes

Voir les routes actuelles

Interface Web : Accédez à `/sms_routing`

Via API :

```
# Lister toutes les routes
curl https://api.example.com:8443/api/routes
```

Vérifier l'utilisation des routes :

Requête Prometheus :

```
# Messages routés par chaque route (dernière heure)
increase(sms_c_routing_route_matched_count[1h])
```

Ajouter une nouvelle route

Interface Web :

1. Accédez à `/sms_routing`
2. Cliquez sur "Ajouter une nouvelle route"
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 réponse automatique/abandon)
    - **Priorité** : Priorité de la route (1-255, plus bas = plus haute priorité)
    - **Poids** : Poids de répartition de charge (1-100)

    - **Description** : Description lisible par l'homme
    - **Activé** : Cochez pour activer immédiatement
4. Cliquez sur "Sauvegarder la route"

Exemple : Route géographique :

- Préfixe appelé : +44
- SMSC de destination : `uk_gateway`
- Priorité : 50
- Poids : 100
- Description : "Routage UK"

Exemple : Route équilibrée par charge :

Créez deux routes avec les mêmes critères mais des poids différents :

Route 1 :

- Préfixe appelé : +44
- SMSC de destination : `uk_primary`
- Priorité : 50
- Poids : 70
- Description : "UK primaire (70%)"

Route 2 :

- Préfixe appelé : +44
- SMSC de destination : `uk_backup`
- Priorité : 50
- Poids : 30
- Description : "UK de secours (30%)"

Tester les routes

Simulateur de routage :

1. Accédez à `/simulator`
2. Entrez les paramètres de test :
    - Numéro d'appel : `+15551234567`
    - Numéro appelé : `+447700900000`
    - SMSC source : (facultatif)
    - Type source : (facultatif)
3. Cliquez sur "Simuler le routage"
4. Examinez les résultats :
    - **Route sélectionnée** : Quelle route a été choisie
    - **Toutes les correspondances** : Quelles routes ont correspondu aux critères
    - **Évaluation** : Pourquoi chaque route a correspondu ou n'a pas correspondu

Tester avant la production :

- Testez toutes les nouvelles routes dans le simulateur
- Vérifiez que la bonne route est sélectionnée
- Vérifiez l'ordre de priorité
- Validez la répartition des poids

Modifier une route existante

Interface Web :

1. Accédez à `/sms_routing`
2. Trouvez la route dans la liste
3. Cliquez sur "Modifier"
4. Modifiez les champs
5. Cliquez sur "Sauvegarder la route"

Modifications courantes :

- **Désactiver la route** : Décochez "Activé" (suppression temporaire)
- **Ajuster le poids** : Changer la répartition de la charge
- **Changer la priorité** : Réorganiser l'évaluation de la route
- **Mettre à jour la destination** : Passer à un autre SMSC

Supprimer une route

Interface Web :

1. Accédez à `/sms_routing`
2. Trouvez la route dans la liste
3. Cliquez sur "Supprimer"
4. Confirmez la suppression

Avertissement : La suppression des routes est permanente. Envisagez de désactiver à la place.

Exporter/Importer des routes

Exporter des routes (Sauvegarde) :

1. Accédez à `/sms_routing`
2. Cliquez sur "Exporter des routes"
3. Sauvegardez le fichier JSON

Importer des routes :

1. Accédez à `/sms_routing`
2. Cliquez sur "Importer des routes"
3. Sélectionnez le fichier JSON
4. Choisissez le mode d'importation :
    - **Fusionner** : Ajouter aux routes existantes
    - **Remplacer** : Supprimer toutes et importer

Cas d'utilisation :

- Sauvegarde avant des changements majeurs
- Copier des routes 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 de vue sont récentes (< 90 secondes)
- Pas de frontends expirés inattendus

Via API :

```
# Obtenir les frontends actifs
curl https://api.example.com:8443/api/frontends/active

# Obtenir des statistiques
curl https://api.example.com:8443/api/frontends/stats
```

Enquêter sur les déconnexions

Frontend expiré :

1. Vérifiez les journaux du frontend pour les erreurs
2. Vérifiez la connectivité réseau au 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 de l'API pour les erreurs d'enregistrement
3. Vérifiez le format de la charge utile JSON
4. Testez l'enregistrement manuellement avec curl

Exemple d'enregistrement manuel :

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

Voir l'historique du frontend

Interface Web :

1. Accédez à `/frontend_status`
2. Trouvez le frontend dans la liste

3. Cliquez sur "Historique"
4. Examinez les enregistrements passés

Via API :

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

Cas d'utilisation :

- Enquêter sur la fiabilité de la connexion
- Suivre les modèles de disponibilité du frontend
- Identifier les changements de configuration

Gestion de la traduction des numéros

Les règles de traduction des numéros sont gérées via `config/runtime.exs`. Les changements nécessitent un redémarrage de l'application.

Voir les règles de traduction actives

Vérifiez le fichier de configuration :

```
cat config/runtime.exs | grep -A 20 "translation_rules:"
```

Tâches de traduction courantes

Ajouter un code pays aux numéros locaux :

Éditez `config/runtime.exs` :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^(\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\d{10})$",
  called_replace: "+1\1",
  priority: 100,
  description: "Ajouter +1 aux numéros américains à 10 chiffres",
  enabled: true
}
```

Normaliser le format international :

```
%{
  calling_prefix: nil,
  called_prefix: nil,
  source_smsc: nil,
  calling_match: "^00(\d+)$",
  calling_replace: "+\1",
  called_match: "^00(\d+)$",
  called_replace: "+\1",
  priority: 10,
```

```
  description: "Convertir le préfixe 00 en +",
  enabled: true
}
```

Suppression de code spécifique à l'opérateur :

```
%{
  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 de l'opérateur 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** : Requête `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 enregistrements plus anciens vers un entrepôt de données ou un stockage à froid
- Supprimez les enregistrements archivés de la base de données opérationnelle par lots pour éviter la contention de verrouillage

Optimiser les tables :

Optimisez périodiquement les tables de la base de données pour maintenir la performance :

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

À exécuter chaque semaine pendant une période de faible trafic pour maintenir une performance optimale.

Maintenance de la base de données Mnesia

Vérifier la taille de la table Mnesia :

```
# Dans la console IEx
:mnesia.table_info(:sms_route, :size)
:mnesia.table_info(:translation_rule, :size)
```

Sauvegarder les tables Mnesia :

```
# Exporter les routes (Interface Web)
# Accédez à /sms_routing
# Cliquez sur "Exporter des routes"

# Ou via la commande de 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 les 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 existe)
2. **Tables de routage** (Mnesia) :
    - Exporter via l'interface Web
    - Ou commande de sauvegarde Mnesia
3. **Base de données SQL CDR** :
    - Sauvegarde complète quotidienne
    - Sauvegardes des journaux de transactions (continue)
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

# Sauvegarder la configuration
cp -r /opt/sms_c/config $BACKUP_DIR/

# Sauvegarder les 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

# Sauvegarder la base de données SQL CDR
# MySQL/MariaDB : Utiliser mysqldump avec --single-transaction pour la
cohérence
# PostgreSQL : Utiliser pg_dump -F c pour le format personnalisé

# Exemple de structure (adapter à votre base de données) :
# - Utiliser l'outil de sauvegarde approprié (mysqldump, pg_dump)
# - Activer les sauvegardes sûres pour la cohérence
# - Compresser la sortie pour économiser de l'espace
# - Configurer la période de conservation (par exemple, 30 jours)

# Supprimer les anciennes sauvegardes
find $BACKUP_DIR -name "sms_c_*.gz" -mtime +30 -delete

echo "Sauvegarde de la base de données terminée : sms_c_${DATE}"
```

Sauvegarde 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 API
curl https://api.example.com:8443/api/routes/export \
  > $BACKUP_DIR/routes_${DATE}.json

echo "Sauvegarde des routes terminée : routes_${DATE}.json"
```

Planifier les 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 config
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** : Décompresser et pipeliner vers le client mysql
- **PostgreSQL** : Utiliser 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 routes"
3. Sélectionnez le fichier JSON de sauvegarde
4. Choisissez le mode "Remplacer"
5. Confirmez l'importation

Planification de la capacité

Surveiller les tendances de croissance

Tendance du volume de messages :

Requête Prometheus (moyenne sur 30 jours) :

```
avg_over_time(sms_c_message_received_count[30d])
```

Taux de croissance de la base de données :

```sql
-- Croissance des données mensuelles
SELECT
  DATE_FORMAT(inserted_at, '%Y-%m') AS month,
  COUNT(*) AS message_count,
  ROUND(SUM(LENGTH(message_body)) / 1024 / 1024, 2) AS data_mb
FROM message_queues
GROUP BY month
ORDER BY month DESC
LIMIT 12;
```

Indicateurs de capacité

Utilisation du CPU :

- **Normal** : < 50% en moyenne
- **Élevé** : > 70% soutenu
- **Critique** : > 90%

Utilisation de la mémoire :

- **Normal** : < 70% de disponible
- **Élevé** : > 80%
- **Critique** : > 90%

Utilisation du disque :

- **Normal** : < 60% plein
- **Élevé** : > 75%
- **Critique** : > 85%

Profondeur de la file d'attente :

- **Normal** : < 1000 en attente
- **Élevé** : > 5000 en attente
- **Critique** : > 10,000 en attente

Recommandations de mise à l'échelle

Quand mettre à l'échelle verticalement (Améliorer les ressources) :

- CPU constamment > 70%

- Mémoire constamment > 80%
- Goulot d'étranglement à nœud unique

Quand mettre à l'échelle horizontalement (Ajouter des nœuds) :

- CPU > 50% sur tous les nœuds
- Volume de messages > 5,000 msg/sec
- Distribution géographique nécessaire
- Haute disponibilité requise

Mise à l'échelle de la base de données :

- Réplicas de lecture pour les requêtes de rapport
- Optimisation du pool de connexions
- Optimisation des index
- Partitionner de grandes tables par date

Réponse aux incidents

Niveaux de gravité

Critique (Réponse immédiate) :

- Aucun message 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 de route 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 :

- Arrêter les changements en cours
- Annuler les déploiements récents si cause suspectée
- Activer les journaux détaillés si nécessaire
- Informer les parties prenantes

4. Enquête :

- Examiner les journaux d'application
- Vérifier l'utilisation des ressources système
- Examiner la performance de la base de données
- Tester les dépendances externes

5. Résolution :

- Appliquer le correctif
- Tester dans le simulateur
- Déployer en production
- Surveiller pour amélioration

6. Après l'incident :

- Documenter la cause racine
- Mettre à jour la surveillance/alertes
- Mettre en œuvre des mesures préventives
- Mettre 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 la performance de la base de données
4. Examinez Prometheus pour les goulots d'étranglement
5. Envisagez d'augmenter la taille/l'intervalle des lots

Échecs de routage :

1. Vérifiez la configuration de routage
2. Testez dans le simulateur de routage
3. Vérifiez les routes manquantes
4. Vérifiez qu'une route 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 la performance des requêtes de base de données
2. Examinez la configuration des travailleurs par lots
3. Vérifiez les ressources adéquates (CPU/Mémoire)
4. Vérifiez les retards de recherche ENUM
5. Examinez la performance du système de facturation

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

Guide d'Optimisation de Performance

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

Ce guide explique comment optimiser la performance de SMS-C pour différents scénarios de charge de travail.

Vue d'Ensemble de la Performance

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

Principales Métriques de Performance

Mesuré sur Intel i7-8650U @ 1.90GHz (8 cœurs) :

Opération	Débit	Latence (moyenne)	Amélioration
Insertion de Message (avec routage)	1,750 msg/sec	0.58ms	21x plus rapide que SQL
Insertion de Message (simple)	1,750 msg/sec	0.57ms	21x plus rapide que SQL
Récupérer des Messages pour SMSC	800 msg/sec	1.25ms	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 une performance optimale :

Magasin de Messages Actif (Mnesia)

- **But** : 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 0.58ms
- **Conservation** : Configurable (par défaut : 24 heures)
- **Clustering** : Prend en charge Mnesia distribué pour un scalabilité horizontale

Archive CDR (Base de Données SQL)

- **But** : 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 Conservation des Messages

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

Directives de Réglage :

- **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 les 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 d'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 de lot CDR
  batch_insert_flush_interval_ms: 100     # Intervalle d'auto-flush
```

Directives de Réglage des CDR

Archivage à Volume Élevé

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

- Des lots plus grands 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 100ms

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)  # Retourne tous les messages
```

Pool de Connexion MySQL

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

```
Nom                                          ips        moyenne  écart
médiane         99e %
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
```

Métriques clés :

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

Base de Performance

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

Métrique	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)	0.58ms	16ms
Temps de Réponse (p99)	<5ms	30ms
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 lot :

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

Retourne :

```
%{
  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
}
```

Métriques 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 < 100ms 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

Configurez des alertes de surveillance pour :

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

Dépannage

Symptôme : Faible Débit

Causes possibles :

1. Pool de connexions de base de données épuisé : Augmenter `pool_size`
2. Base de données lente : Vérifier la performance des requêtes, ajouter des index
3. Latence réseau : Optimiser le chemin réseau vers la base de données
4. Taille de lot trop petite : Augmenter `batch_insert_batch_size`

Symptôme : Haute Latence

Causes possibles :

1. Intervalle de flush trop élevé : Réduire `batch_insert_flush_interval_ms`
2. Taille de lot trop élevée : Réduire `batch_insert_batch_size`
3. Écritures lentes de la base de données : Vérifier l'I/O disque, optimiser les tables
4. Utilisation de l'API asynchrone alors que vous avez besoin de synchronisation : Passer à un point de terminaison synchronisé

Symptôme : Problèmes de Mémoire

Causes possibles :

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

Meilleures Pratiques

1. **Commencer avec les valeurs par défaut** (100/100ms) et régler en fonction du comportement observé
2. **Surveiller en production** pendant au moins 1 semaine avant d'optimiser
3. **Tester les changements de configuration** en staging avec une charge semblable à celle de la production
4. **Utiliser des benchmarks** pour valider les changements de configuration
5. **Documenter vos décisions de réglage** pour référence future
6. **Configurer des alertes** avant d'optimiser pour détecter les régressions
7. **Considérer 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

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 modèles d'erreur spécifiques
grep "routing_failed" /var/log/sms_c/application.log

# Trouver des erreurs de base de données SQL
grep -i "database\|sql\|ecto" /var/log/sms_c/application.log | grep
error
```

Surveiller les Journaux en Temps Réel :

```
# Suivre les journaux avec filtre
tail -f /var/log/sms_c/application.log | grep -E
"(error|warning|critical)"
```

Requêtes de Métriques

Vérifier le Taux de Traitement des Messages :

```
# Messages par seconde
rate(sms_c_message_received_count[5m])

# Taux de succès de livraison
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])
```

Vérifier l'État de la File d'Attente :

```
# Profondeur actuelle de la file d'attente
sms_c_queue_size_pending

# Âge du message le plus ancien (secondes)
sms_c_queue_oldest_message_age_seconds
```

Vérifier la Performance du Système :

```
# Latence de traitement des messages (p95)
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)

# Latence de routage (p95)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
```

Problèmes de Livraison de Messages

Messages Non Livrés

Symptômes :

- Messages bloqués 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 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 :

- Réviser 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 l'horodatage `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 des Réessais

Symptômes :

- Compteur `delivery_attempts` en augmentation
- Messages avec un nombre élevé de tentatives (> 3)
- Retards exponentiels

Étapes de Diagnostic :

1. **Vérifier le Journal des Événements** :

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

Rechercher :

- Événements d'échec de livraison
- Descriptions d'erreur
- Horodatages de réessai

2. **Vérifier les Journaux Frontend** :

- Pourquoi le frontend échoue-t-il à livrer ?
- Erreurs réseau ?
- Erreurs de protocole ?
- Système en aval indisponible ?

Solutions :

Problèmes Réseau Temporaires :

- Attendre le réessai (automatique)
- Surveiller pour une livraison réussie

Échecs Persistants :

```
# Route 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 compteur `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 n'ont pas correspondu

3. **Vérifier les Critères de Route** :

- Les préfixes doivent-ils correspondre ?
- Le filtre SMSC source est-il trop restrictif ?
- Toutes les routes sont-elles désactivées ?

Solutions :

Aucune Route Configurée :

Ajouter une route par défaut :

```
Préfixe d'Appel : (vide)
Préfixe Appelé : (vide)
SMSC Source : (vide)
SMSC Dest : default_gateway
Priorité : 255
Poids : 100
Activé : ✓
Description : Route par défaut
```

Routes Trop Spécifiques :

Ajouter une route plus large :

```
Préfixe Appelé : +
SMSC Dest : international_gateway
Priorité : 200
Poids : 100
Activé : ✓
Description : Route internationale
```

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 "Tous les Correspondances"
- Vérifier les scores de priorité et de spécificité

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

- Un nombre plus bas = une priorité plus élevée
- Les routes sont évaluées par ordre de priorité
- Au sein de la même priorité, les poids s'appliquent

3. **Vérifier la Spécificité des Routes** :

Scoring 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 plus basse)
```

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 envoyé
- Journal des événements manquant l'événement 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 bas) :

```
Route de Réponse Automatique :
  Priorité : 10

Route Normale :
  Priorité : 50
```

Problèmes de Performance

Latence Élevée de Traitement des Messages

Symptômes :

- `sms_c_message_processing_stop_duration` p95 > 1000ms
- Réponses API lentes
- File d'attente 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 routage

Recherches ENUM Lentes :

- Vérifier la latence du serveur DNS
- Augmenter le délai d'attente
- Utiliser des serveurs DNS plus rapides/proches
- Désactiver ENUM si non nécessaire

Facturation Lente :

- Vérifier la performance de l'OCS
- Augmenter le délai d'attente de l'OCS
- Désactiver la facturation si non nécessaire
- Utiliser la facturation asynchrone

Base de Données Lente :

- Augmenter la taille du pool de connexions
- Ajouter des index
- Optimiser les requêtes
- Améliorer les ressources de la base de données

Modifications de Configuration :

```
# config/config.exs
# Augmenter la taille du lot pour le débit
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# Augmenter le pool de base de données
config :sms_c, SmsC.Repo,
  pool_size: 50
```

Faible Débit de Messages

Symptômes :

- Traitement < 100 msg/sec
- Utilisation de l'API asynchrone mais toujours lente
- Temps de réponse API élevés

Étapes de Diagnostic :

1. **Vérifier le Travailleur de Lot** :

```
# Dans la console de production (iex)
SmsC.Messaging.BatchInsertWorker.stats()
```

Rechercher :

- `current_queue_size` 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 la 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 Frontend** :

- Les frontends interrogent-ils suffisamment souvent ?
- Les frontends traitent-ils les messages assez rapidement ?
- Des erreurs 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 frontend
- Vérifier l'intervalle de polling (devrait être de 5 à 10 secondes)
- Redémarrer les services frontend

Frontends Trop Lents :

- Ajouter plus d'instances frontend
- Optimiser le traitement frontend
- Augmenter la concurrence frontend

Taux de Réessai Élevé :

- Enquêter sur les échecs de livraison
- Corriger les problèmes en aval
- Router vers des passerelles alternatives

Pointe Temporaire :

- Attendre que la file d'attente se vide
- Surveiller jusqu'à normal
- Envisager des mises à niveau de capacité si récurrent

Problèmes de Base de Données

Échecs de Connexion

Symptômes :

- Erreur : "impossible de se connecter à la base de données"
- API retournant des erreurs 500
- L'application ne démarre pas

Étapes de Diagnostic :

1. **Vérifier l'État de la Base de Données SQL CDR** :

```
# MySQL/MariaDB
systemctl status mysql

# PostgreSQL
systemctl status postgresql

# Tester la connectivité (MySQL/MariaDB)
mysql -u sms_user -p -h db.example.com -e "SELECT 1"

# Tester la connectivité (PostgreSQL)
psql -U sms_user -h db.example.com -d sms_c_prod -c "SELECT 1"
```

2. **Vérifier le Réseau** :

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

# Vérifier la connectivité 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 à partir 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** :

```
-- 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** :

```
-- 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** :

```
-- 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 :

```
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 :

```
-- 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 :

```
-- 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 &#123;&#125; + | sort -rh |
head -20

# Trouver les plus gros fichiers (PostgreSQL)
find /var/lib/postgresql -type f -exec du -h &#123;&#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 Tourner les Journaux :

```
# Forcer logrotate
logrotate -f /etc/logrotate.d/sms_c

# Effacer les anciens fichiers journaux
find /var/log/sms_c -name "*.log.*" -mtime +30 -delete
```

Étendre le Disque :

- Redimensionner le volume (cloud)
- Ajouter un nouveau disque et étendre le volume
- Déplacer les données vers un disque plus grand

Problèmes de Connexion Frontend

Frontend Ne S'affiche Pas Comme Actif

Symptômes :

- Le statut du frontend montre "expiré"
- Le frontend n'est pas dans la liste active
- Les messages ne sont pas livrés au frontend

Étapes de Diagnostic :

1. **Vérifier l'Enregistrement** :

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

2. **Vérifier les Journaux Frontend** :

- Le frontend appelle-t-il `/api/frontends/register` ?
- Des erreurs API ?
- Fréquence d'enregistrement (devrait 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 de l'enregistrement toutes les 60 secondes :

```
# Le frontend doit appeler l'enregistrement 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 frontend
- Vérifier les journaux 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 :

Intervalle correct :

```
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 '&#123;
    "method": "SessionSv1.Ping",
    "params": [],
    "id": 1
  &#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** :

```
# Pinger 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: [
    &amp;#123;"8.8.8.8", 53&amp;#125;,  # Google Public DNS
    &amp;#123;"1.1.1.1", 53&amp;#125;   # Cloudflare DNS
  ]
```

Domaine ENUM Incorrect :

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 :

Faible Taux de Réussite (Attendu) :

- Trafic vers des numéros uniques (normal)
- Surveiller mais ne pas alerter 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 retournent 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 que c'est 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 ﷺ 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 en premier
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 d'attente 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 à l'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 Retourne 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 la configuration 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 d'attente 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
```

Acc��der au 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 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 Développement (F12)
- Rechercher des erreurs WebSocket
- Vérifier l'onglet réseau pour les requêtes échouées

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

Si un proxy inverse est utilisé, 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 :

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

Routage Inefficace :

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

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 d'Attente 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 :

- Évoluer verticalement
- Ajouter du 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`

Documentation de conformité à l'interception ANSSI R226

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

Classification : Documentation de conformité réglementaire

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

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

---

1. SPÉCIFICATIONS TECHNIQUES DÉTAILLÉES

1.1 Fiche technique commerciale

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

Capacités principales

Traitement des messages :

- File d'attente de messages SMS centralisée avec REST API
- Conception indépendante du protocole prenant en charge les frontaux SMPP, IMS, SS7/MAP
- Moteur de routage dynamique avec routage basé sur les préfixes
- Logique de réessai avec retour exponentiel
- Gestion de l'expiration des messages et de la file d'attente des lettres mortes
- Génération et archivage des enregistrements de détails d'appel (CDR)
- Performance : ~1 750 messages/seconde, 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 une milliseconde)
    - Sauvegarde sur disque : le mode `disc_copies` écrit sur disque pour la récupération après un crash
    - Récupération automatique : Les messages survivent aux redémarrages du système
    - Conservation : Configurable (par défaut 24 heures), puis nettoyage automatique
- **Archive CDR à long terme :** Base de données MySQL/PostgreSQL (séparée de la file d'attente de messages)
    - Les CDR sont écrits lorsque les messages sont livrés, expirés, échoués ou rejetés
    - 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 nœuds)

Interfaces réseau :

- **REST API :** HTTPS (port 8443) pour la communication avec les frontaux externes
- **Panneau de contrôle :** HTTPS (port 8086) pour la gestion web
- **Protocoles frontaux :** SMPP, IMS, SS7/MAP (via des applications passerelles 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 les 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
- Contrôle 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 internationale d'abonné mobile)
    - 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 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 du disque pour la récupération après un crash
  - Les messages persistent entre les redémarrages du système
- **Performance :** Opérations de lecture/écriture inférieures à une milliseconde
- **Conservation :** À court terme (par défaut 24 heures), configurable
- **Nettoyage :** Archivage automatique vers la base de données CDR, puis suppression de Mnesia
- **Opérations :** Toutes les opérations de la file d'attente de messages (insertion, mise à jour, statut de livraison, routage)
- **Caractéristique critique :** La base de données SQL n'est JAMAIS interrogée pendant le 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 la performance du routage des messages)
- **Conservation :** À long terme (mois à années), configurable selon les exigences réglementaires
- **Opérations :** Requêtes historiques, reporting, conformité, interception légale
- **Accès :** Requêtes SQL, REST API (futur), export CSV/JSON

Avantages architecturaux clés :

1. **Performance :** Les opérations de routage actives ne touchent jamais SQL (pas de goulet d'étranglement de base de données)
2. **Scalabilité :** Mnesia gère plus de 1 750 messages/seconde sans surcharge SQL
3. **Fiabilité :** Le mode `disc_copies` garantit aucune perte de message en cas de crash
4. **Conformité :** La base de données CDR fournit une piste d'audit permanente
5. **Séparation des préoccupations :** Données opérationnelles vs. données d'archivage clairement séparées

Cycle de vie des messages :

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

Conservation et récupération des données :

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

Suivi des frontaux :

- Suivi en temps réel des frontaux SMSC externes (SMPP, IMS, passerelles

MAP)
- Enregistrement des frontaux avec surveillance de l'état de santé
- Suivi de l'état de santé (actif/expiré)
- Historique de disponibilité/indisponibilité
- Suivi des adresses IP et des noms d'hôte
- Journalisation de la configuration spécifique aux frontaux

1.2.3 Capacités d'analyse

Surveillance en temps réel :

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

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

Analyse historique :

- Base de données CDR interrogeable par :
  - Plage horaire
  - Numéro de 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 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 les préfixes

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

1.3 Capacités de contre-mesures

1.3.1 Mécanismes de protection de la vie privée

Confidentialité des communications :

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

Contrôle d'accès :

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

Confidentialité 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 la 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 des configurations
- Séparation des configurations basées 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 et du stockage de conformité à long terme.

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

Qu'est-ce que Mnesia ?

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

Mode de stockage : disc_copies

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

Cycle de vie des messages dans Mnesia :

1. Message arrive via REST API → Inséré dans Mnesia RAM + sauvegarde disque
2. Le moteur de routage interroge Mnesia → Réponse instantanée (accès mémoire)
3. La passerelle externe interroge pour des messages → Requête Mnesia (accès mémoire)
4. La passerelle met à jour le statut de livraison → Mise à jour Mnesia (mémoire + disque)
5. Après livraison/expiration → Message marqué pour nettoyage
6. Travail de nettoyage (par défaut 24h) → 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 (enregistrement de détails d'appel) ?

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

Quand les CDR sont écrits : Les enregistrements CDR sont créés

UNIQUEMENT lorsque les messages atteignent un état final :

- ✅ Message livré avec succès
- ✅ Message expiré (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 par lots :** Plusieurs CDR regroupés (par défaut 100) et écrits ensemble
- **Intervalle de vidage :** 100 ms par défaut (configurable)
- **Gestion des erreurs :** Les écritures CDR échouées sont enregistrées, le traitement des messages continue

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

Objectif de la base de données SQL :

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

Diagramme d'architecture

Légende :

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

Implications de l'interception légale

Messages récents (< 24 heures) :

- Accessibles via Mnesia (requêtes REST API)
- Récupération ultra-rapide
- Contenu complet du message disponible
- Surveillance en temps réel possible

Messages historiques (> 24 heures) :

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

1.5 Architecture d'intégration multi-protocoles des frontaux

Le SMSc OmniMessage utilise une conception de cœur indépendante du protocole qui s'interface avec des passerelles spécifiques à des 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 frontaux IMS/SIP

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

Données d'interception spécifiques à IMS :

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

2. Intégration du frontaux 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 d'interception spécifiques à SMPP :

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

3. Intégration du frontaux 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 d'interception spécifiques à SS7/MAP :

- IMSI des messages MAP
- Adresses Global Title (GT)
- Adresse MSC/VLR (identification de l'élément réseau)
- Adresses de 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 cœur 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 de protocole
4. **Préservation complète des métadonnées :** Champs spécifiques au protocole préservés dans le CDR

Résumé du flux de données :

Identification du protocole dans le CDR :

- Le champ `source_smsc` indique le protocole 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 pour 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 REST API 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 (de 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 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 à 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 message
- Notifications de livraison de message
- Événements de changement de statut de message
- Filtrage d'événements configurable par critères
- Support WebSocket pour la surveillance en direct

4. Interface d'exportation par lots :

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

Interfaces standard d'interception légale ETSI

Le SMSc OmniMessage fournit les bases pour la mise en œuvre des interfaces d'interception légale conformes à l'ETSI. Bien que le cœur du SMSc n'implémente pas nativement les interfaces X1/X2/X3, il fournit tous les points d'accès 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 :

- **Objectif :** Suivi des mandats et provisionnement des cibles de l'application des lois au système d'interception
- **Direction :** LEMF → LIMF (bidirectionnelle)
- **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 le statut 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) :

- **Objectif :** Fournir des métadonnées de message aux forces de l'ordre
- **Direction :** LIMF → LEMF (unidirectionnelle)
- **Format de données :** Conforme à ETSI TS 102 232-x XML/ASN.1
- **Contenu provenant 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) :

- **Objectif :** Fournir le contenu réel du message aux forces de l'ordre
- **Direction :** LIMF → LEMF (unidirectionnelle)
- **Format de données :** Conforme à ETSI TS 102 232-x
- **Contenu provenant du SMSc :**
  - Corps du message (contenu texte)
  - PDU brut (données SMS binaires)
  - Informations sur le codage des caractères

    - Segments de message multipart
    - Informations TP-DCS
    - En-tête de données utilisateur (UDH)
- **Intégration avec 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 dans le format CC conforme à l'ETSI
    - LIMF livre le CC à LEMF via X3

Architecture d'implémentation :

Mapping 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 filtrée 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 des 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 d'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 cibles

Interception basée sur les 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 de 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 :

```
-- 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;
```

```sql
-- 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 de 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 comprend :

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

Gestion des certificats :

- Support des certificats X.509
- Tailles de 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 sur 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 de la mémoire via l'isolation de la VM Erlang

2.3.3 Chiffrement du système de fichiers

Stockage de données sensibles :

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

Stockage des clés :

- Certificats et clés TLS stockés dans `priv/cert/`
- Magasins de clés basés sur des fichiers avec des permissions restreintes
- Procédures de rotation 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 l'en-tête (en-tête SMSc pour identification du frontal)
- Contrôle d'accès basé sur l'IP (niveau pare-feu)
- Authentification client basée sur des certificats (optionnel)

Enregistrement des frontaux :

- Identification unique des frontaux (nom, type, IP, nom d'hôte)
- Authentification basée sur le heartbeat
- Gestion de session basée sur l'expiration (timeout de 90 secondes)
- Suivi et surveillance 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",  # Doit utiliser des mots de passe forts
en production
  hostname: "localhost",
  ssl: true  # Activer TLS pour les connexions à la base de données
```

Recommandations de contrôle d'accès :

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

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

2.5 Détails des algorithmes cryptographiques

2.5.1 Algorithmes de hachage

Disponibles dans Erlang/OTP :

- SHA-256, SHA-384, SHA-512 (recommandés)
- SHA-1 (déprécié, compatibilité héritée uniquement)

- MD5 (déprécié, non utilisé pour la sécurité)
- BLAKE2 (disponible dans les versions modernes d'OTP)

Utilisation :

- Empreinte des messages (détection 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 clé :

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

Utilisation :

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

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 de certificat TLS
- Signatures numériques
- Échange de clés

2.6 Sécurité du protocole SMS

2.6.1 Encodage des messages SMS

Support de codage 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

Pas de 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 des protocoles

Protocole SMPP (Frontal externe) :

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

Protocole IMS (Frontal externe) :

- Messagerie basée sur SIP
- Mécanismes d'authentification SIP
- Intégration avec la sécurité du réseau IMS

Protocole SS7/MAP (Frontal 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 cœur 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 la criminalistique
- Exportation de métriques Prometheus

2.7.2 Considérations d'évaluation des vulnérabilités

Limitations connues :

- Le protocole SMS est intrinsèquement non chiffré (par conception, permet l'interception légale)
- 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 robustes
- Mettre en œuvre le chiffrement des connexions à la base de données
- Utiliser une gestion externe des secrets (Vault, AWS Secrets Manager)
- Mises à jour régulières de sécurité pour Erlang/OTP et les 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 de pénétration
- Validation de la configuration TLS
- Audits de sécurité de la base de données
- Revue des contrôles d'accès

2.8 Infrastructure de gestion des clés

2.8.1 Génération de clés

Génération de certificat 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 d'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 :

- Renouvellement de certificat TLS (annuel recommandé)
- Rotation des identifiants de base de données
- Rotation des tokens API (si implémenté)

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

Conformité aux normes cryptographiques :

- NIST SP 800-52 : Directives TLS
- NIST SP 800-131A : Transitions d'algorithmes cryptographiques
- RFC 7525 : Recommandations TLS
- ETSI TS 133 310 : Sécurité réseau (pour l'intégration IMS)

Réglementations françaises en matière de cryptographie :

- Pas de cryptographie restreinte à l'exportation (tous les algorithmes standard)
- Déclaration des moyens cryptographiques (si applicable)
- Certification de produit cryptographique ANSSI (si requise)

2.10 Résistance à la cryptanalyse

2.10.1 Principes de conception

Défense contre la cryptanalyse :

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

2.10.2 Sécurité opérationnelle

Rotation des clés :

- Procédures de renouvellement de certificat TLS
- Rotation des clés de session (par session pour TLS)
- Politiques de rotation des 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 des 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'identifiants cibles (vues de base de données)
- Requêtes à durée limitée (clauses WHERE SQL)
- Application automatique via des politiques d'accès

3.2 Conservation des données et confidentialité

Politiques de conservation :

- Conservation des messages actifs : configurable (par défaut 24 heures dans Mnesia)
- Conservation des CDR : configurable (typique 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 UI/exportations
- Chiffrement de la base de données au repos
- Journalisation et surveillance des accès
- Principe de collecte minimale de données

Configuration :

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

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

  # Contrôle d'écriture CDR
  cdr_enabled: true,

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

🔀 Voir [CONFIGURATION.md] pour tous les paramètres de conservation

3.3 Interfaces de transfert pour les forces de l'ordre

Interfaces standard :

1. Accès REST API :

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

2. Accès direct à la base de données :

- Identifiants SQL en lecture seule
- Requêtes SQL standard
- Accès à la table CDR
- Capacités de recherche indexées

3. Exportation par lots :

- Format d'exportation CSV
- Format d'exportation JSON
- Exportations basées sur la plage horaire
- Sélection de champs configurable

Formats de livraison :

IRI (Informations liées à l'interception) :

- Champs de métadonnées CDR :
  - ID de message
  - Numéros appelants/appelés
  - Horodatages (soumission, livraison, expiration)
  - Statut
  - Tentatives de livraison

    - Informations de routage SMSC
    - Informations sur le nœud (suivi de cluster)

CC (Contenu de la communication) :

- Corps du message (contenu texte)
- Données PDU brutes
- Informations sur le codage
- Assemblage de message multipart

Exemple d'exportation :

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

---

4. SÉCURITÉ ET INTÉGRITÉ DU SYSTÈME

4.1 Sécurité de l'application

Sécurité Erlang/Elixir :

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

Gestion des dépendances :

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

4.2 Sécurité réseau

Exposition réseau :

- Ports exposés minimaux :
  - 8443 (API REST HTTPS)
  - 8086 (Panneau de contrôle HTTPS)
  - 3306/5432 (Base de données - doit être protégée par un pare-feu)
- Configuration de pare-feu recommandée
- Liste blanche des IP pour l'accès frontal
- Déploiement DMZ pour les services exposés à Internet

Segmentation réseau :

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

4.3 Surveillance et détection d'intrusion

Capacités de journalisation :

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

Surveillance des événements de sécurité :

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

Métriques et alertes :

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

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

4.4 Haute disponibilité et récupération après sinistre

Support de cluster :

- Capacité de cluster distribué Erlang
- Réplication Mnesia entre nœuds
- Basculement automatique
- Découverte et adhésion des nœuds

Redondance des données :

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

Procédures de récupération :

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

---

5. RÉFÉRENCES DOCUMENTAIRES

5.1 Manuels techniques

Documentation disponible dans le dépôt du projet :

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

5.2 Certifications de sécurité

- **Rapports de test de pénétration :** [À fournir sur demande]
- **Rapports d'audit de sécurité :** [À fournir sur demande]
- **Évaluations des vulnérabilités :** [À fournir sur demande]

- **Validation cryptographique Erlang/OTP :** Bibliothèque cryptographique standard de l'industrie

5.3 Documentation de conformité

- **Demande d'autorisation ANSSI R226 :** Ce document
- **Conformité à l'interception légale :** Comme requis par

Guide de Traduction des Numéros SMS-C

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

Aperçu

Le système de Traduction des Numéros SMS-C fournit une transformation flexible basée sur des regex des numéros de téléphone avant le routage. Les règles de traduction peuvent normaliser les numéros, ajouter des préfixes internationaux, formater les numéros pour des passerelles spécifiques et enchaîner plusieurs transformations. Les règles sont stockées dans Mnesia pour la persistance et peuvent être modifiées à l'exécution sans interruption de service.

Caractéristiques Clés

- **Correspondance basée sur le préfixe** : Correspondre les numéros par préfixe avant d'appliquer les transformations
- **Transformation basée sur regex** : Puissante correspondance de motifs et remplacement avec des groupes de capture
- **Filtrage 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 de règles** : Continuer le traitement à travers plusieurs règles avec prévention des boucles
- **Transformations d'appelant/appelé séparées** : 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 du SMSC source (nil = joker)	Non
`calling_match`	chaîne/nil	Motif regex pour correspondre au numéro appelant	Non
`calling_replace`	chaîne/nil	Motif de remplacement pour le numéro appelant	Non
`called_match`	chaîne/nil	Motif regex pour correspondre au numéro appelé	Non
`called_replace`	chaîne/nil	Motif de remplacement pour le numéro appelé	Non
`priority`	entier	Priorité de la règle (1-255, plus bas = priorité plus élevée)	Oui
`description`	chaîne	Description lisible par l'homme	Non
`enabled`	booléen	Activer/désactiver la règle	Oui
`continue`	booléen	Continuer à évaluer les règles après correspondance (par défaut : faux)	Non

Remarque : Les règles sont évaluées par ordre de priorité (le plus petit nombre 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 de Règles avec Prévention des Boucles

Le drapeau `continue` permet un puissant enchaînement de règles tout en prévenant les boucles infinies :

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 motifs de transformation (correspondance/remplacement nil) passe les numéros sans changement

Exemple : Scénario d'Enchaînement de Règles

Configuration

Chargement des Règles depuis le Fichier de Configuration

Les règles de traduction peuvent être définies dans `config/runtime.exs` et seront automatiquement chargées au premier démarrage.

Important : Les règles de configuration ne sont chargées que lorsque la table de traduction est **vide** (premier démarrage). Cela préserve les règles ajoutées via l'interface Web pendant l'exécution et empêche les doublons lors des redémarrages.

Flux de Chargement de Configuration

Exemple de Configuration

```
# config/runtime.exs
config :sms_c, :translation_rules, [
  # Ajouter +1 aux numéros américains 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 du 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
  }
]
```

Démarrer

Flux d'Initialisation

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 les numéros selon les exigences spécifiques de la passerelle :

Traductions Spécifiques au SMSC

Appliquer différentes traductions en fonction de la source du message :

Préparation au Routage Basée sur le Préfixe

Normaliser les numéros avant le routage pour garantir une correspondance de préfixe cohérente :

Gestion de la Portabilité des Numéros

Gérer les numéros portés qui nécessitent des changements de préfixe :

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 les règles existantes
- Activer/désactiver des règles sans suppression
- 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 = priorité plus élevée)
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 correspond
- **Continuer** : Appliquer cette règle et continuer à évaluer les règles restantes
- Les règles avec continue activé affichent un badge bleu "↓ Continuer" dans le tableau

Modifier une Règle :

1. 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 motifs regex sont affichés en police monospace pour plus de clarté

Simulateur de Traduction

Accédez au simulateur à `/translation_simulator` (via le menu de navigation) :

Fonctionnalités :

- Tester la logique de traduction avec des numéros réels
- **Transformation Étape par Étape** montrant chaque règle appliquée

- Voir les valeurs avant/après pour chaque transformation
- Voir quelles règles ont correspondu et pourquoi
- Charger des scénarios d'exemple pour des tests rapides
- Voir l'historique des tests (derniers 10 tests)

Utiliser le Simulateur :

1. 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 de l'étape et informations sur la règle
        - Description de la règle
        - Avant → Après pour les numéros appelants et appelés
        - Indicateur "↓ Continuer" pour les règles qui ont continué le traitement
        - Transformations mises en évidence en vert
        - Valeurs inchangées marquées comme "passées à travers"
4. Charger des exemples pré-configurés en utilisant les boutons d'exemple
5. Examiner l'historique des tests pour comparer différents scénarios

Exemple de Sortie :

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

Numéro Appelant :  5551234567 → +1-555-123-4567
Numéro Appelé :   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

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

Retourne :

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

```
# Exemple d'utilisation
{:ok, new_calling, new_called, 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 les tests"
})

# Supprimer une règle
:ok = NumberTranslation.delete_rule(rule_id)

# Obtenir une règle spécifique
rule = NumberTranslation.get_rule(rule_id)

# Lister toutes les règles
all_rules = NumberTranslation.list_rules()

# Lister uniquement les règles activées (triées par priorité)
enabled_rules = NumberTranslation.list_enabled_rules()
```

Opérations d'Import/Export

```
# Exporter toutes les règles
backup = NumberTranslation.export_rules()
# Retourne : %{
#   version: "1.0",
#   exported_at: ~U[2024-01-15 10:30:00Z],
#   count: 5,
#   rules: [...]
# }

# Sauvegarder dans un fichier JSON
json = Jason.encode!(backup, pretty: true)
```

```
File.write!("translation_rules_backup.json", json)

# Importer des règles (fusionner avec les existantes)
{:ok, %{imported: 3, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :merge)

# Importer des règles (remplacer toutes les existantes)
{:ok, %{imported: 5, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :replace)
```

Meilleures Pratiques

Conception des Règles

1. **Maintenir les priorités organisées** :
    - **1-10** : Règles de normalisation critiques (ajouter des codes pays, corriger des 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 la performance
3. **Documenter vos règles** :
    - Toujours ajouter des descriptions claires
    - Inclure des exemples dans la description (ex. : "5551234567 → +15551234567")
    - Documenter l'objectif et l'entrée/sortie attendue
4. **Tester les motifs regex** :
    - Tester les motifs avec le simulateur avant de déployer
    - Utiliser des groupes de capture (\1, \2) pour des transformations flexibles
    - Échapper les caractères spéciaux regex (points, parenthèses, etc.)

Performance

1. **Minimiser le nombre de règles** :

    - Combiner des règles similaires lorsque cela est possible
    - Utiliser la correspondance de préfixe pour réduire les évaluations regex
    - Supprimer ou désactiver les règles inutilisées

2. **Optimiser les motifs regex** :

    - Utiliser d'abord la correspondance de préfixe (plus rapide que regex)
    - Garder les motifs regex simples
    - Éviter les motifs lourds en backtracking

3. **Limiter l'enchaînement des règles** :

    - Les longues chaînes (5+ règles) peuvent impacter la performance
    - 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 de message_id pour le débogage
    - Vérifier les journaux d'événements pour les décisions de traduction
    - Surveiller quelles règles sont appliquées

4. **Déploiement progressif** :

    - Ajouter de nouvelles règles comme désactivées d'abord
    - Tester avec le simulateur
    - Activer et surveiller
    - Ajuster si nécessaire

Conseils Regex

1. **Motifs courants** :

    - Numéro américain 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})$`
    - Référencer dans le remplacement : `+1\1\2`
    - Multiples captures : `^\+(\d{1,3})(\d+)$` → `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)
- Le SMSC source ne correspond pas
- Le motif regex ne correspond pas au format d'entrée
- La règle est désactivée
- Une règle de priorité supérieure a été correspondue en premier (avec continue: false)

Solutions :

1. Utiliser le simulateur pour voir quelles règles sont évaluées
2. Vérifier le statut de la règle (activée/désactivée)
3. Vérifier la correspondance des préfixes (sensible à la casse)
4. Tester le motif regex séparément
5. Vérifier l'ordre de priorité

Mauvaise Transformation Appliquée

Symptôme : Numéro transformé mais le résultat est incorrect

Causes possibles :

- Le motif regex correspond mais le motif 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 motif regex capture les bons groupes
3. Vérifier la syntaxe du motif 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 de Performance

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 des erreurs
2. Revoir les règles avec continue: true
3. Simplifier les motifs regex
4. Réduire le nombre de règles enchaînées
5. Signaler un bug si la prévention des boucles a échoué

Enchaînement de Règles Inattendu

Symptôme : Plus de règles appliquées que prévu

Causes possibles :

- Les règles ont continue: true alors qu'elles ne devraient pas
- L'ordre de priorité permet plusieurs correspondances
- Le numéro transformé correspond à des règles supplémentaires

Solutions :

1. Utiliser le simulateur pour voir la chaîne de règles exacte
2. Revoir les drapeaux continue sur toutes les règles
3. Ajuster les priorités pour contrôler l'ordre
4. Définir continue: false sur la règle finale

Traduction Non Appliquée Avant le Routage

Symptôme : Le routeur voit des numéros non traduits

Causes possibles :

- 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 la performance 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 clusterisés. Les règles de traduction sont répliquées pour une haute disponibilité.

Stratégies de Migration

Lors du déploiement de nouvelles règles de traduction :

Exemples

Exemple 1 : Normalisation des Numéros Américains

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

```
# Règle 1 : numéros 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 pr��fixe 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 au SMSC

Exigence : Appliquer des règles différentes en fonction du 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 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 SMSC (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 en +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
- Examinez 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 de Routage SMS-C

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

Vue d'ensemble

Le système de routage SMS-C fournit un routage flexible et haute performance des messages SMS basé sur plusieurs critères, y compris les préfixes de numéros, les identifiants SMSC, les types de connexion, et plus encore. Les routes sont stockées dans Mnesia pour la persistance et peuvent être modifiées à l'exécution sans interruption de service.

Caractéristiques Clés

- **Routage basé sur les préfixes** : Routage basé sur les préfixes de numéros appelants/appelés avec une logique de correspondance la plus longue gagnante
- **Routage basé sur le SMSC** : Routage basé sur le SMSC source ou destination
- **Routage basé sur le type** : Routage basé sur le type de connexion source (IMS, Circuit Commuté, SMPP)
- **Routage basé sur la priorité** : Contrôle de l'ordre de sélection des routes avec des priorités configurables
- **Équilibrage de charge basé sur le poids** : Distribution du trafic sur plusieurs routes en utilisant des poids
- **Routage de réponse automatique** : Envoi automatique de réponses aux expéditeurs de messages
- **Routage de suppression** : Rejet des messages correspondant à des critères spécifiques (filtrage de spam, etc.)
- **Contrôle de facturation** : Configuration du comportement de facturation par route (Oui/Non/Par défaut)
- **Chargement de fichier de configuration** : Chargement des routes initiales depuis `runtime.exs` au premier démarrage
- **Configuration à l'exécution** : Ajout, modification ou désactivation des routes sans redémarrer
- **Interface Web** : Interface CRUD complète pour la gestion des routes avec menu déroulant frontal
- **Outil de simulation** : Tester la logique de routage avant le déploiement
- **Sauvegarde/Restaurer** : Exporter et importer des configurations de routage
- **Support ENUM** : Recherche de numéro basée sur DNS (pour une implémentation future)

Architecture

Modèle de Données

Chaque route contient les champs suivants :

Champ	Type	Description	Requis
`route_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 du SMSC source (nil = joker)	Non
`dest_smsc`	chaîne/nil	Nom du SMSC de destination (requis sauf si auto_reply ou drop est vrai)	Conditionnel
`source_type`	atome/nil	Type source : `:ims`, `:circuit_switched`, `:smpp`, ou nil	Non
`enum_domain`	chaîne/nil	Domaine DNS ENUM pour la recherche	Non
`auto_reply`	booléen	Si vrai, envoie une réponse à l'expéditeur	Non (par défaut : faux)
`auto_reply_message`	chaîne/nil	Texte du message pour la réponse automatique (requis si auto_reply est vrai)	Conditionnel
`drop`	booléen	Si vrai, rejette le message (filtrage de spam)	Non (par défaut : faux)
`charged`	atome	Comportement de facturation : `:yes`, `:no`, ou `:default`	Non (par défaut : `:default`)
`weight`	entier	Poids d'équilibrage de charge (1-100, par défaut 100)	Oui
`priority`	entier	Priorité de la route (1-255, plus bas = plus haute priorité)	Oui
`description`	chaîne	Description lisible par l'homme	Non
`enabled`	booléen	Activer/désactiver la route	Oui

Remarque : Une route doit être l'un des trois types :

1. **Routage normal** : `auto_reply=false`, `drop=false`, nécessite `dest_smsc`
2. **Réponse automatique** : `auto_reply=true`, nécessite `auto_reply_message`
3. **Suppression** : `drop=true`, rejette le message

Algorithme de Routage

Lors du routage d'un message, le système suit cet ordre de priorité :

PRIORITÉ 1 : Routage Basé sur la Localisation (Plus Haute)

1. **Vérifier l'enregistrement de l'abonné** : Si le MSISDN de destination est enregistré dans la table des localisations
2. **Routage directement vers le frontend de service** : Ignorer toutes les règles de routage et envoyer directement au frontend servant cet abonné
3. **Cela se produit APRÈS la traduction des numéros** pour garantir la cohérence avec les enregistrements de localisation

PRIORITÉ 2 : Règles de Routage Standard (si aucun enregistrement de localisation trouvé)

1. **Filtre les routes activées** qui correspondent à TOUS les critères spécifiés
2. **Trie par spécificité** (routes plus spécifiques en premier) :
    - Préfixe appelé plus long = plus haute spécificité (×100 points)
    - Préfixe appelant plus long = spécificité moyenne (×50 points)
    - SMSC source spécifié = +25 points
    - Domaine de résultat ENUM spécifié = +15 points
    - Type source spécifié = +10 points
    - Domaine ENUM spécifié = +5 points
3. **Regroupe par priorité** (numéro inférieur = priorité plus élevée)
4. **Sélectionne dans le groupe de la plus haute priorité** en utilisant une sélection aléatoire pondérée
5. **Exécute l'action de la route** :
    - **Route normale** : Retourne le SMSC de destination pour la livraison du message
    - **Route de réponse automatique** : Envoie une réponse à l'expéditeur de manière asynchrone
    - **Route de suppression** : Rejette le message et enregistre l'événement

Jokers

- `nil` ou des valeurs vides agissent comme des jokers qui correspondent à n'importe quelle valeur
- Une route sans critères spécifiés est une route de rattrapage

Configuration

Chargement des Routes depuis le Fichier de Configuration

Les routes peuvent être définies dans `config/runtime.exs` et seront automatiquement chargées au premier démarrage. Cela est utile pour définir des

règles de routage de base qui devraient être présentes lorsque le système démarre pour la première fois.

Important : Les routes provenant de la configuration ne sont chargées que lorsque la table de routage est **vide** (premier démarrage). Cela préserve les routes ajoutées via l'interface Web pendant l'exécution et empêche les doublons lors des redémarrages.

Flux de Chargement de Configuration

Exemple de Structure de Configuration de Route

Voir `config/runtime.exs` et `config/sms_routes.example.exs` pour des exemples complets incluant :

- Routage géographique
- Routes de réponse automatique
- Routes de suppression (filtrage de spam)
- Routes équilibrées par charge
- Routage de numéros premium avec facturation

Prise en Main

Flux d'Initialisation

Vue d'ensemble des Types de Routes

Flux de Routage des Messages

Cas d'utilisation Courants

Routage Basé sur la Localisation (Priorité la Plus Élevée)

Routage des messages directement vers le frontend servant un abonné enregistré, contournant toutes les règles de routage :

Comment cela fonctionne :

1. Le message arrive avec le numéro de destination
2. Les numéros sont traduits (si configuré)
3. Le système vérifie si le MSISDN de destination traduit est dans la table des localisations
4. Si enregistré, le message est routé directement vers le frontend servant cet abonné
5. Les règles de routage standard sont **complètement ignorées**
6. Si non enregistré, les règles de routage normales s'appliquent

Avantages :

- **Livraison garantie** au bon frontend pour les abonnés enregistrés
- **Routage le plus rapide** - aucune évaluation de la table de routage nécessaire
- **Routage précis** - la localisation de l'abonné est la source de vérité
- **Outrepasse toutes les règles de routage** - garantit la joignabilité de l'abonné

Cas d'utilisation :

- Abonnés IMS/VoLTE enregistrés sur des cœurs IMS spécifiques
- Abonnés mobiles attachés à des MSC spécifiques
- Abonnés SIP enregistrés sur des serveurs d'applications spécifiques

Routage Géographique

Routage des messages vers des SMSCs régionaux en fonction du pays de destination :

Équilibrage de Charge

Distribuer le trafic sur plusieurs SMSCs avec des poids :

Routage de Numéros Premium

Routage des numéros premium vers un traitement spécial avec priorité :

Routage Spécifique au Protocole

Routage basé sur le type de connexion source :

Migration de Réseau

Lors de la migration, routage de préfixes spécifiques vers une nouvelle infrastructure :

Routage Multi-Critères Complexe

Combiner plusieurs critères pour un contrôle précis :

Interface Web

Interface de Gestion des Routes

Accédez à l'interface de gestion des routes à `/sms_routing` (configurer dans votre routeur) :

Fonctionnalités :

- Voir toutes les routes dans un tableau triable
- Ajouter de nouvelles routes avec validation de formulaire
- Modifier des routes existantes
- Activer/désactiver des routes sans supprimer
- Supprimer des routes avec confirmation
- Mises à jour en temps réel (rafraîchissement toutes les 5 secondes)

Ajout d'une Route :

1. Cliquez sur "Ajouter une Nouvelle Route"
2. Remplissez les champs du formulaire (seul le SMSC de destination est requis)
3. Définissez le poids (1-100, par défaut 100) et la priorité (1-255, par défaut 100)
4. Cochez "Activé" pour activer immédiatement
5. Cliquez sur "Enregistrer la Route"

Modification d'une Route :

1. Cliquez sur "Modifier" à côté de la route
2. Modifiez les champs selon les besoins
3. Cliquez sur "Enregistrer la Route"

Désactivation d'une Route :

- Cliquez sur "Désactiver" pour désactiver temporairement sans supprimer
- Cliquez sur "Activer" pour réactiver

Simulateur de Routage

Accédez au simulateur à `/simulator` (via le menu de navigation) :

Fonctionnalités :

- Tester la logique de routage avec divers paramètres
- **Évaluation détaillée champ par champ** montrant pourquoi chaque route a correspondu ou non
- Voir toutes les routes évaluées par ordre de priorité
- Indicateurs visuels pour les routes correspondantes/sélectionnées
- Charger des scénarios d'exemple pour des tests rapides
- Voir l'historique des tests (derniers 10 tests)

Utilisation du Simulateur :

1. Entrez les paramètres de test :
    - Numéro appelant (de)
    - Numéro appelé (à)

    - SMSC source (facultatif)
    - Type source (Tout/IMS/Circuit Commuté/SMPP)
2. Cliquez sur "Simuler le Routage"
3. Voir les résultats complets :
    - **Résultat de Routage** : Route sélectionnée et destination (ou "Aucune Route Trouvée")
    - **Évaluation de la Route** : Toutes les routes avec analyse champ par champ :
        - ✓ Coche verte = Champ correspondant
        - ✗ Croix rouge = Champ ne correspondant pas
        - Raison pour chaque correspondance/non-correspondance de champ
    - **Indicateurs visuels** :
        - Bordure verte + badge "SÉLECTIONNÉ" = Route effectivement utilisée
        - Bordure violette + badge "CORRESPONDANTE" = Routes qui ont correspondu mais n'ont pas été sélectionnées
        - Bordure grise = Routes qui n'ont pas correspondu
4. Charger des exemples préconfigurés à l'aide des boutons d'exemple
5. Revoir l'historique des tests pour comparer différents scénarios

Exemple de Sortie d'Évaluation : Pour chaque route, vous verrez pourquoi elle a correspondu ou non :

- **Préfixe appelant** : "Correspond au préfixe '1234'" ou "Ne commence pas par '44'"
- **Préfixe appelé** : "Joker (correspond à tout)" ou "Ne commence pas par '639'"
- **SMSC source** : "Correspond à 'smsc1'" ou "Attendu 'untrusted_smsc', obtenu 'none'"
- **Type source** : "Joker (correspond à tout)" ou "Attendu 'smpp', obtenu 'IMS'"

Référence API

Vue d'ensemble des Opérations Principales

Opérations de Gestion des Routes

Paramètres de Routage des Messages

route_message accepte les paramètres suivants :

- `calling_number` (facultatif) : Numéro de téléphone d'origine
- `called_number` (facultatif) : Numéro de téléphone de destination
- `source_smsc` (facultatif) : Identifiant SMSC source
- `source_type` (facultatif) : Type de connexion (`:ims`, `:circuit_switched`,

`:smpp`)
- `message_id` (facultatif) : Pour la journalisation des événements

Retourne :

- `{:ok, dest_smsc, route}` - Route trouvée et sélectionnée
- `{:error, :no_route_found}` - Aucune route correspondante

Opérations d'Import/Export

Meilleures Pratiques

Conception de Route

1. **Utilisez les priorités judicieusement** : Réservez les priorités basses (1-10) pour les routes critiques
2. **Gardez-le simple** : Commencez par des routes larges et ajoutez des spécifiques au besoin
3. **Documentez les routes** : Ajoutez toujours des descriptions aux routes
4. **Utilisez un joker** : Ayez toujours une route par défaut avec une priorité basse

Performance

1. **Minimisez le nombre de routes** : Combinez les routes similaires lorsque cela est possible
2. **Utilisez les préfixes les plus longs** : Des préfixes plus spécifiques réduisent le temps d'évaluation
3. **Désactivez les routes inutilisées** : Ne supprimez pas les routes dont vous pourriez avoir besoin plus tard ; désactivez-les

Opérations

1. **Testez avant de déployer** : Utilisez le simulateur pour vérifier la logique de routage
2. **Sauvegardez régulièrement** : Exportez les routes avant d'apporter des modifications majeures
3. **Surveillez le routage** : Vérifiez les journaux d'événements pour les décisions de routage
4. **Déploiement progressif** : Utilisez des poids pour déplacer progressivement le trafic vers de nouvelles routes

Tests

1. **Écrivez des tests d'intégration** : Testez vos scénarios de routage spécifiques
2. **Test de charge** : Vérifiez les performances de routage sous charge

3. **Tests de basculement** : Assurez-vous que les routes de secours fonctionnent lorsque les primaires échouent

Dépannage

Aucune Route Trouvée

Symptôme : `{:error, :no_route_found}` retourné

Causes possibles :

- Aucune route configurée
- Toutes les routes correspondantes sont désactivées
- Les critères de route ne correspondent pas aux paramètres du message
- Le préfixe ne correspond pas (vérifiez les fautes de frappe)

Solutions :

1. Vérifiez que les routes existent : `SmsRouting.list_enabled_routes()`
2. Utilisez le simulateur pour tester le routage avec les paramètres de message réels
3. Ajoutez une route de rattrapage pour le débogage : `add_route(%{dest_smsc: "debug_smsc", priority: 255})`
4. Vérifiez les journaux d'événements pour les détails d'évaluation du routage

Mauvaise Route Sélectionnée

Symptôme : Message routé vers une destination inattendue

Causes possibles :

- Mauvaise configuration de priorité
- Route joker a une priorité plus élevée
- Le calcul de spécificité favorise une route différente
- Plusieurs routes avec les mêmes critères utilisant des poids

Solutions :

1. Utilisez le simulateur pour voir toutes les routes correspondantes
2. Vérifiez les valeurs de priorité (plus bas = plus haute priorité)
3. Vérifiez les scores de spécificité dans le simulateur
4. Examinez la distribution des poids pour les routes équilibrées par charge

Problèmes de Performance

Symptôme : Le routage est lent

Causes possibles :

- Trop de routes dans la base de données
- Modèles de route complexes
- Table Mnesia mal indexée

Solutions :

1. Consolidez les routes similaires
2. Supprimez les routes désactivées qui ne sont plus nécessaires
3. Assurez-vous que les index Mnesia sont créés (automatique dans `init_tables`)
4. Envisagez de mettre en cache les décisions de routage fréquemment utilisées

Sujets Avancés

Intégration ENUM/NAPTR

ENUM (E.164 Number Mapping) fournit une recherche de numéro basée sur DNS utilisant des enregistrements NAPTR. Le SMS-C inclut un support ENUM complet avec mise en cache, serveurs DNS configurables, et correspondance de route basée sur les résultats de recherche ENUM.

Qu'est-ce qu'ENUM ?

ENUM mappe les numéros de téléphone E.164 vers des noms DNS en utilisant une transformation simple :

- **Numéro de Téléphone** : +1-212-555-1234
- **Requête ENUM** : 4.3.2.1.5.5.5.2.1.2.1.e164.arpa
- **Type d'Enregistrement DNS** : NAPTR (Nom d'Autorité Pointeur)
- **Résultat** : URI SIP, informations de routage, ou autres données de service

Configuration

La fonctionnalité ENUM est configurée dans `config/runtime.exs` :

Activer les Recherches ENUM :

Définissez `enum_enabled: true` pour activer les recherches ENUM avant le routage. Lorsqu'elle est activée, le système effectuera des recherches DNS ENUM pour les messages entrants et utilisera les résultats dans les décisions de routage.

Domaines ENUM :

Listez les domaines ENUM à interroger par ordre de priorité. Le système essaiera chaque domaine jusqu'à ce qu'une recherche réussie se produise.

Domaines ENUM courants :

- `e164.arpa` - Domaine ENUM officiel de l'IETF
- `e164.org` - Registre ENUM alternatif
- Domaines ENUM privés personnalisés

Serveurs DNS :

Configurez des serveurs DNS spécifiques pour les requêtes ENUM. Format : `{ip_address, port}`

Laissez vide ou définissez sur `[]` pour utiliser les serveurs DNS par défaut du système.

Exemple de configuration DNS personnalisée :

- DNS Public de Google : `{"8.8.8.8", 53}`, `{"8.8.4.4", 53}`
- DNS Cloudflare : `{"1.1.1.1", 53}`, `{"1.0.0.1", 53}`
- DNS ENUM personnalisé : `{"10.0.0.53", 53}`

Délai d'Attente :

Définissez le délai d'attente de la requête DNS en millisecondes (par défaut : 5000ms). Augmentez pour les réseaux lents, diminuez pour un basculement plus rapide.

Comment Fonctionnent les Recherches ENUM

Mise en Cache ENUM

Le système met en cache les résultats de recherche ENUM pendant 15 minutes pour améliorer les performances et réduire la charge DNS.

Avantages du Cache :

- Réduit la charge des requêtes DNS
- Améliore la latence de routage
- Protège contre les pannes de serveur DNS (les résultats mis en cache restent disponibles)

Statistiques de Cache :

- Voir la taille et l'état du cache dans la page de Test NAPTR
- Surveiller les taux de réussite/échec du cache via les métriques Prometheus
- Effacer le cache manuellement si nécessaire (modifications de configuration, tests, etc.)

Comportement du Cache :

- Les recherches réussies et échouées sont toutes deux mises en cache
- Les recherches échouées sont mises en cache pour éviter des requêtes répétées pour des numéros invalides
- Le cache expire automatiquement après 15 minutes
- Le cache survit aux redémarrages d'application (stocké dans ETS)

Utilisation d'ENUM dans les Routes

Les routes peuvent correspondre aux résultats de recherche ENUM en utilisant le champ `enum_result_domain` :

Scénario Exemple :

La recherche ENUM pour +1-555-0100 retourne un enregistrement NAPTR :

- Service : E2U+sip
- Remplacement : sip:customer@voip-carrier.com
- **Domaine de Résultat** : voip-carrier.com

Configuration de Route :

Créez une route avec `enum_result_domain: "voip-carrier.com"` pour correspondre aux messages où la recherche ENUM a retourné ce domaine.

Logique de Correspondance :

- Si la route a `enum_result_domain: nil` - correspond à tous les messages (joker)
- Si la route a `enum_result_domain: "specific.com"` - ne correspond que si ENUM a retourné ce domaine
- Les routes avec des domaines ENUM correspondants reçoivent des scores de spécificité plus élevés

Calcul de Priorité :

Les routes avec des domaines de résultats ENUM reçoivent +15 points de spécificité, les priorisant par rapport aux routes génériques.

Tester les Recherches ENUM

Accédez à la page de Test NAPTR à `/naptr_test` (via le menu de navigation).

Fonctionnalités :

- Effectuer des recherches ENUM en direct contre les serveurs DNS configurés

- Voir des informations détaillées sur les enregistrements NAPTR
- Voir les domaines de résultat extraits des enregistrements NAPTR
- Surveiller les statistiques de cache
- Effacer le cache pour les tests

Flux de Test :

1. Entrez un numéro de téléphone (avec ou sans préfixe +)
2. Spécifiez le domaine ENUM (par défaut : e164.arpa)
3. Cliquez sur "Effectuer la Recherche"
4. Revoir les résultats :
    - Enregistrements NAPTR trouvés
    - Ordre et valeurs de préférence
    - Types de services (E2U+sip, E2U+tel, etc.)
    - Expressions régulières
    - Valeurs de remplacement
    - **Domaines de résultat extraits** (utilisés pour la correspondance de route)

Affichage de la Configuration Actuelle :

- Serveurs DNS utilisés (ou "Système Par Défaut")
- Paramètre de délai d'attente
- Taille et état du cache
- Bouton d'effacement du cache

Comprendre les Résultats :

Chaque enregistrement NAPTR contient :

- **Ordre** : Priorité pour le traitement (plus bas en premier)
- **Préférence** : Au sein du même ordre (plus bas en premier)
- **Drapeaux** : Instructions de traitement (u=terminal, s=continuer)
- **Service** : Type de service (E2U+sip, E2U+tel, etc.)
- **Regexp** : Expression de substitution
- **Remplacement** : Domaine ou adresse alternative
- **Domaine de Résultat** : Domaine extrait pour la correspondance de route

Cas d'utilisation Courants ENUM

1. Peering VoIP

Utilisez ENUM pour identifier les numéros hébergés sur des réseaux SIP/VoIP et les router directement vers des passerelles VoIP :

- ENUM retourne URI SIP : sip:number@voip-carrier.com
- Domaine de résultat : voip-carrier.com
- Route avec `enum_result_domain: "voip-carrier.com"` sélectionnée

- Trafic envoyé à la passerelle de peering VoIP directe

2. Identification du Transporteur

Identifiez le transporteur servant un numéro et routez en conséquence :

- ENUM retourne des informations sur le transporteur
- Domaine de résultat : carrier-a.com
- Route vers l'interconnexion du transporteur A
- Optimisez les coûts et la qualité du routage

3. Portabilité de Numéro

Gérez les numéros portés qui ont déménagé entre les transporteurs :

- La recherche ENUM retourne le transporteur actuel
- Route vers la destination correcte automatiquement
- Pas de mises à jour manuelles de la table de routage nécessaires

4. Routage à Moindre Coût

Combinez ENUM avec plusieurs routes :

- ENUM identifie le réseau de destination
- Plusieurs routes pour le même domaine avec des coûts différents
- Utilisez priorité et poids pour préférer les routes à coût inférieur

5. Services d'Urgence

Routez les numéros d'urgence (911, 112, etc.) vers les services d'urgence appropriés :

- La recherche ENUM identifie la passerelle d'urgence locale
- Route de haute priorité assure un routage immédiat
- Pas de retard dû à l'évaluation normale des routes

Stratégie de Routage ENUM

Configuration Recommandée :

1. **Routes ENUM de Haute Priorité (Priorité 1-10)**
    - Routes qui correspondent à des domaines de résultats ENUM spécifiques
    - Utilisées pour le peering direct, le routage VoIP
    - Spécificité la plus élevée, sélectionnées en premier
2. **Routes de Préfixe de Priorité Moyenne (Priorité 50-100)**

    - Routage standard basé sur le préfixe
    - Utilisées lorsque la recherche ENUM échoue ou ne retourne aucun enregistrement
    - Sauvegarde fiable

3. **Route de Rattrapage de Basse Priorité (Priorité 200+)**

    - Route par défaut pour tout le reste
    - Assure qu'aucun message ne reste non routé

Hiérarchie d'Exemple de Route :

- Priorité 1 : `enum_result_domain: "sip.carrier.com"` → Passerelle VoIP directe
- Priorité 10 : `enum_result_domain: "tel.carrier.com"` → Passerelle PSTN du transporteur
- Priorité 50 : `called_prefix: "+1"` → Passerelle par défaut Amérique du Nord
- Priorité 100 : `called_prefix: "+"` → Passerelle par défaut Internationale
- Priorité 200 : Aucun critère → Dernière sauvegarde

Considérations de Performance

Latence de Requête DNS :

Les recherches ENUM ajoutent un temps de requête DNS au routage :

- **Mise en cache** : < 1ms (rapide)
- **Non mise en cache** : 10-100ms (dépend du serveur DNS)

Recommandations :

- Utilisez des serveurs DNS géographiquement proches
- Configurez un délai d'attente approprié (5000ms par défaut)
- Surveillez les taux de réussite du cache (objectif > 80%)
- Envisagez de préchauffer le cache pour les numéros connus

Scalabilité :

Le système de cache gère des scénarios à fort volume :

- Le cache est partagé entre tous les processus
- Table ETS en lecture-concurrente pour la performance
- Nettoyage automatique du cache via TTL
- Évolue jusqu'à des millions d'entrées mises en cache

Gestion des Échecs :

Les échecs ENUM tombent gracieusement en arrière vers le routage régulier :

- Délai d'attente DNS → Passer à la route suivante
- Aucun enregistrement NAPTR → Utiliser les routes basées sur le préfixe
- Format NAPTR invalide → Enregistrer l'erreur, continuer le routage
- Serveur DNS indisponible → Utiliser les résultats mis en cache ou passerelle

Surveillance des Opérations ENUM

Utilisez les métriques Prometheus pour surveiller les performances ENUM :

- `sms_c_enum_lookup_stop_duration` - Latence de recherche
- `sms_c_enum_cache_hit_count` - Hits de cache
- `sms_c_enum_cache_miss_count` - Misses de cache
- `sms_c_enum_cache_size_size` - Taille actuelle du cache
- `sms_c_enum_naptr_records_record_count` - Enregistrements NAPTR par recherche

Métriques Clés à Surveiller :

- **Taux de réussite du cache** : Devrait être > 70% après préchauffage
- **Durée de recherche p95** : Devrait être < 1000ms
- **Recherches échouées** : Surveillez les problèmes DNS

Voir `docs/METRICS.md` pour la documentation complète des métriques.

Dépannage ENUM

Problème : Aucun Enregistrement NAPTR Trouvé

- Vérifiez la configuration du domaine ENUM
- Testez la connectivité du serveur DNS
- Vérifiez si le numéro est réellement dans le registre ENUM
- Essayez un domaine ENUM alternatif (par exemple, e164.org)
- Utilisez la page de Test NAPTR pour diagnostiquer

Problème : Recherches ENUM Lentes

- Vérifiez la latence du serveur DNS
- Vérifiez la connectivité réseau
- Augmentez le délai d'attente si nécessaire
- Envisagez d'utiliser des serveurs DNS plus proches
- Vérifiez le taux de réussite du cache

Problème : Mauvaise Route Sélectionnée Après ENUM

- Vérifiez le champ `enum_result_domain` dans les routes
- Utilisez le Simulateur de Route pour tester la logique de routage
- Vérifiez que l'extraction du domaine de résultat est correcte

- Passez en revue le format de l'enregistrement NAPTR dans la page de Test

Problème : Recherches ENUM Désactivées

- Vérifiez `enum_enabled: true` dans config/runtime.exs
- Vérifiez que la liste `enum_domains` n'est pas vide
- Redémarrez l'application après les modifications de configuration
- Vérifiez les journaux d'application pour l'initialisation ENUM

Considérations de Sécurité

Empoisonnement de Cache DNS :

- Utilisez uniquement des serveurs DNS de confiance
- Envisagez DNSSEC si disponible
- Validez les formats d'enregistrement NAPTR
- Surveillez les domaines de résultat inattendus

Épuisement des Ressources :

- Les limites de cache empêchent l'épuisement de la mémoire
- Le délai d'attente empêche de rester bloqué sur un DNS lent
- Les recherches échouées sont mises en cache pour éviter les tempêtes de réessai

Divulgation d'Informations :

- Les recherches ENUM révèlent les intentions de routage aux serveurs DNS
- Utilisez des serveurs DNS privés pour un routage sensible
- Envisagez VPN/DNS chiffré pour la confidentialité

Journalisation des Événements

Les décisions de routage sont enregistrées via l'EventLogger :

- `sms_routing_started` : Évaluation du routage commence
- `sms_routing_candidates` : Nombre de routes activées trouvées
- `sms_routing_matches` : Nombre de routes correspondantes
- `sms_routing_selected` : Détails de la route sélectionnée
- `sms_routing_failed` : Aucune route trouvée

Activez la journalisation en passant `message_id` à `route_message/1`.

Clustering

Les tables Mnesia sont automatiquement distribuées entre les nœuds du cluster. Les routes sont répliquées pour une haute disponibilité.

Exemples

Voir la suite de tests à `test/sms_c/messaging/sms_routing_test.exs` pour des exemples complets de :

- Correspondance de préfixe
- Routage basé sur la priorité
- Équilibrage de charge basé sur le poids
- Routage multi-critères
- Cas limites

Migration depuis l'Ancien Routage

Si vous migrez depuis l'ancien routage basé sur la configuration, suivez ce processus :

Détails des Étapes de Migration

1. **Initialiser les Tables**
    - Crée les tables de routage Mnesia
    - Prépare le système pour le nouveau routage
2. **Analyser les Anciennes Routes**
    - **Modèles Regex** → Routes basées sur le préfixe
    - **Réponses préenregistrées** → Routes de réponse automatique
    - **Logique personnalisée** → Routes multi-critères
3. **Tester en Profondeur**
    - Utilisez le simulateur de routage
    - Vérifiez tous les scénarios
    - Vérifiez les cas limites
4. **Mettre à Jour le Code**
    - Remplacez les anciens appels de routage
    - Utilisez l'API `route_message/1`
    - Mettez à jour la gestion des erreurs
5. **Déployer & Surveiller**
    - Déployez le nouveau système de routage
    - Surveillez les problèmes
    - Conservez l'ancienne configuration comme sauvegarde initialement

6. **Nettoyer**
    - Supprimez l'ancienne configuration de routage
    - Supprimez le code de migration
    - Mettez à jour la documentation

Support

Pour des problèmes ou des questions :

- Vérifiez la suite de tests pour des exemples
- Utilisez le simulateur pour déboguer la logique de routage
- Consultez les journaux d'événements pour les décisions de routage
- Vérifiez le contenu de la table Mnesia : `:mnesia.table_info(:sms_route, :size)`