



Benchee SMS-C



1. SMS (raw_sms_bench.exs)

submit_message_raw API SMS PDU

- SMS PDU PDU @sample_pdus
-
- HTML

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

benchmarks/output/raw_sms_benchmark.html

2. API (message_api_bench.exs)

API

- insert_message
- get_messages_for_smsc
- list_message_queues
-

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

📄 `benchmarks/output/message_api_benchmark.html`

📄

📄📄📄📄📄 Benchee📄📄📄📄📄📄📄📄📄

- 📄📄2 📄
- 📄📄10 📄
- 📄📄📄📄2 📄
- 📄📄📄📄
- 📄📄📄 HTML 📄📄

📄📄

HTML 📄📄📄📄📄📄 `benchmarks/output/` 📄📄📄📄

- 📄📄📄📄📄
- 📄📄📄
- 📄📄📄📄
- 📄📄📄

SMS-C 目录

← [README](#)

本目录列出了 SMS-C 项目的所有子目录。SMS-C 项目目录

目录

子目录

- [目录](#) - 项目目录

子目录

- [目录](#) - 项目目录
- [SMS 目录](#) - 项目目录
- [HSS 目录](#) - 项目目录 Diameter Sh 子目录
- [目录](#) - 子目录 HTTP 子目录 DNS SRV 子目录
- [API 子目录](#) - 子目录 API 子目录

子目录

- [目录](#) - 项目目录
- [目录](#) - Prometheus 子目录

子目录

- [目录](#) - 项目目录

子目录

- [ANSI R226 目录](#) - 项目目录
 - 子目录 (IMS/SIP, SMPP, SS7/MAP)
 - ETSI X1/X2/X3 子目录

- Mnesia + SQL 資料庫
- 資料庫 CDR 表
- 資料庫

資料庫

資料庫

- 資料庫
- 資料庫
- 資料庫
- 資料庫
- 資料庫

資料庫

- 資料庫
- CDR 資料庫
- 資料庫
- 資料庫
- 資料庫
- 資料庫
- 資料庫
- Diameter Sh / HSS 表
- ENUM/NAPTR 表
- OCS 表
- 資料庫

資料庫

- 資料庫
- 資料庫
- 資料庫

□□□□

□□□□

- **CPU:** 2 □□
- **RAM:** 4 GB
- □□: 50 GB (□□□□□□□□)
- □□□□: Linux (□□), macOS (□□)
- **Erlang/OTP:** 26.x □□□□□
- **Elixir:** 1.15.x □□□□□
- **SQL □□□:** MySQL 8.0+ □ MariaDB 10.5+ □ PostgreSQL 13+ (□□ CDR □□)

□□□□

- **CPU:** 8+ □□
- **RAM:** 16+ GB
- □□: 500+ GB SSD
- □□: 1 Gbps+
- **SQL □□□:** □□□□□□□□□□ (□□ CDR □□)

□□□□

- **80/443** - Web UI (HTTP/HTTPS)
- **8443** - API (HTTPS) □□□□□□□□
- **9568** - Prometheus □□

□□□□□

□□

- □□□□: `/var/log/sms_c/` (□□) □□□□ (□□)
- **Web UI** □□: □□□□□□□□□□ `/logs`
- □□□□: □□ API □□□□□□□□□□

API

- 状态: `GET /api/status`
- 指标: `GET http://localhost:9568/metrics` (Prometheus 指标)
- 前端状态: Web UI 页面 `/frontend_status`
- 消息队列: Web UI 页面 `/message_queue`

部署

1. 准备环境
2. 安装依赖
3. 配置 Prometheus 监控
4. 部署应用
5. 验证部署

配置

系统配置

- 日期: 2025-10-30
- **SMS-C** 版本: 1.15.x - 1.17.x
- 依赖 **Elixir**: 1.15.x - 1.17.x
- 依赖 **Erlang/OTP**: 26.x - 27.x

部署

部署步骤

- 部署应用
- **API** 测试: `curl` 命令
- **IP** 地址: 192.168.1.100
- 配置 Prometheus 监控
- 设置时区为 UTC

ANSSI R226

OmniMessage (SMSc) R226-3 R226-7 ANSSI R226

(ANSSI)

R226 -

1.

1.1

OmniMessage SMSc

REST API (HTTPS) (SMPP/IMS/SS7/MAP)

Elixir/Erlang/Phoenix Mnesia/MySQL/PostgreSQL

- REST API
- SMPP/IMS/SS7/MAP
-
-
-
- (CDR)
- 1,750 / 1.5

- **Mnesia**
 - RAM
 - `disc_copies`
 -
 - 24
- **CDR** MySQL/PostgreSQL
 - CDR
 - SQL CDR
 -
- - 1,750 SQL
 - CDR
 - SQL
- Mnesia

API

- **REST API** HTTPS 8443
- HTTPS 8086 Web
- SMPP IMS SS7/MAP
- MySQL/PostgreSQL CDR

Database

-
-
- SMSC
-
-
- ENUM (E.164) DNS
-
- CGRates

• [README.md](#)

1.2 消息

1.2.1 消息

消息

- OmniMessage SMSc 消息
- 消息
 - 消息 MSISDN
 - 消息 MSISDN
 - 消息 IMSI
 - 消息 IMSI
 - 消息
 - 消息 PDU
 - TP-DCS
 - 消息 GSM7 UCS-2 8 Latin-1
 - 消息
 - 消息 UDH

消息

- 消息 CDR
 - 消息 ID
 - 消息 MSISDN
 - 消息 MSISDN
 - 消息
 - 消息
 - 消息
 - 消息
 - 消息
 - 消息
 - 消息 SMSC
 - 消息 SMSC
 - 消息 Erlang
 - 消息

- 資料庫

CDR 資料庫 CDR_SCHEMA.md

資料庫

- 資料庫
- REST API 資料庫
- 資料庫
- 資料庫
 - 資料庫/表
 - SMSC 表
 - 表
 - 表
 - 表

API 參考 API_REFERENCE.md

1.2.2 資料庫

資料庫

SMSc 資料庫

1. 資料庫 Mnesia

- 表
- Erlang Mnesia 表
- 表 disc_copies
 - RAM 表
 - 表
 - 表
- 表 表/表
- 表 24 表
- 表 CDR 表 Mnesia 表
- 表 表
- 表 表 SQL 表

2 CDR MySQL/PostgreSQL

- CDR data ingestion
- CDR data SQL MySQL PostgreSQL
- CDR data ingestion CDR
 - CDR data
 - CDR data
 - CDR data
 - CDR data
- CDR data ingestion
- CDR data ingestion
- CDR data ingestion
- CDR SQL REST API CSV/JSON

1. CDR data

1. CDR data ingestion SQL
2. CDR data Mnesia 1,750 CDR data SQL
3. CDR data disc_copies CDR data
4. CDR data CDR data
5. CDR data CDR data

2. CDR data

1. CDR data → CDR data Mnesia RAM + CDR data
2. CDR data → Mnesia CDR data
3. CDR data → CDR data SQL
4. 24 CDR data → CDR data Mnesia CDR data
5. CDR data SQL → CDR data

3. CDR data

- CDR data ingestion
- CDR data ingestion PDU CDR data Mnesia CDR data
- CDR data ingestion CDR data
- CDR data CDR data

□□□□

- □□□□ SMSC □□□SMPP□IMS□MAP □□□
- □□□□□□□□
- □□□□□□□□/□□□
- □□□□/□□□□
- IP □□□□□□□□
- □□□□□□□□□□

1.2.3 □□□□

□□□□

- Web UI □□□□□□
 - □□□□□□
 - □□□□□□□
 - □□□□□□□□□
 - □□□□□□
 - □□□□□□□
- Prometheus □□□□□□□□□□
- □□□□□□□□□□□□□□□□

□ □□□□□□□ **OPERATIONS_GUIDE.md** □ □□□□□ **METRICS.md**

□□□□

- CDR □□□□□□□□□□□□□□
 - □□□□
 - □□/□□□□□□
 - □□□□
 - SMSC □□
 - □□□□
 - □□□□□□□□□□□□□□□□□□
- □□□□□□□□
 - □□□□/□□□□□□□
 - □□□□□□□□/□□□□

- 國際號碼
- 國際號碼
- 國際號碼

國際號碼

- 國際號碼MSISDN國際號碼
- 國際 IMSI 國際號碼 IMS/MAP 國際號碼
- 國際號碼
- 國際號碼
- 國際號碼國際號碼國際號碼

國際號碼

- 國際號碼
- 國際號碼國際號碼
- 國際號碼
- 國際號碼國際號碼國際號碼
- 國際號碼
- 國際號碼

國際號碼

- E.164 國際號碼
- 國際號碼國際號碼/國際號碼
- 國際號碼國際號碼
- ENUM DNS 國際號碼國際號碼
- 國際號碼國際號碼

◻ 國際號碼 [number_translation_guide.md](#) ◻ 國際號碼 [sms_routing_guide.md](#)

1.3 國際號碼

1.3.1 國際號碼


國際號碼

- HTTPS/TLS 〇〇 REST API 〇〇
- 〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇〇〇 TLS〇
- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇

〇〇〇〇〇

- Web UI 〇〇〇〇
- API 〇〇〇〇〇〇
- 〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇

〇〇〇〇〇

- 〇〇〇〇〇〇〇〇〇〇
- 〇〇/〇〇〇〇
- 〇〇〇〇〇〇
- 〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇

1.3.2 〇〇〇〇〇〇

〇〇〇〇〇

- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇 UI 〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇
- CDR 〇〇〇〇〇〇〇〇〇〇〇〇 NULL 〇〇〇〇〇

〇〇〇〇〇〇

- MySQL 〇〇〇〇〇〇ENCRYPTION='Y'〇
- PostgreSQL 〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇

1. REST API → Mnesia RAM +
2. Mnesia →
3. → Mnesia
4. → Mnesia +
5. / →
6. 24 → Mnesia

SQL

- SQL
- SQL
- SMS-C I/O

2 SQL CDR

CDR

-
- MySQL PostgreSQL
-

CDR CDR

-
-
-
-

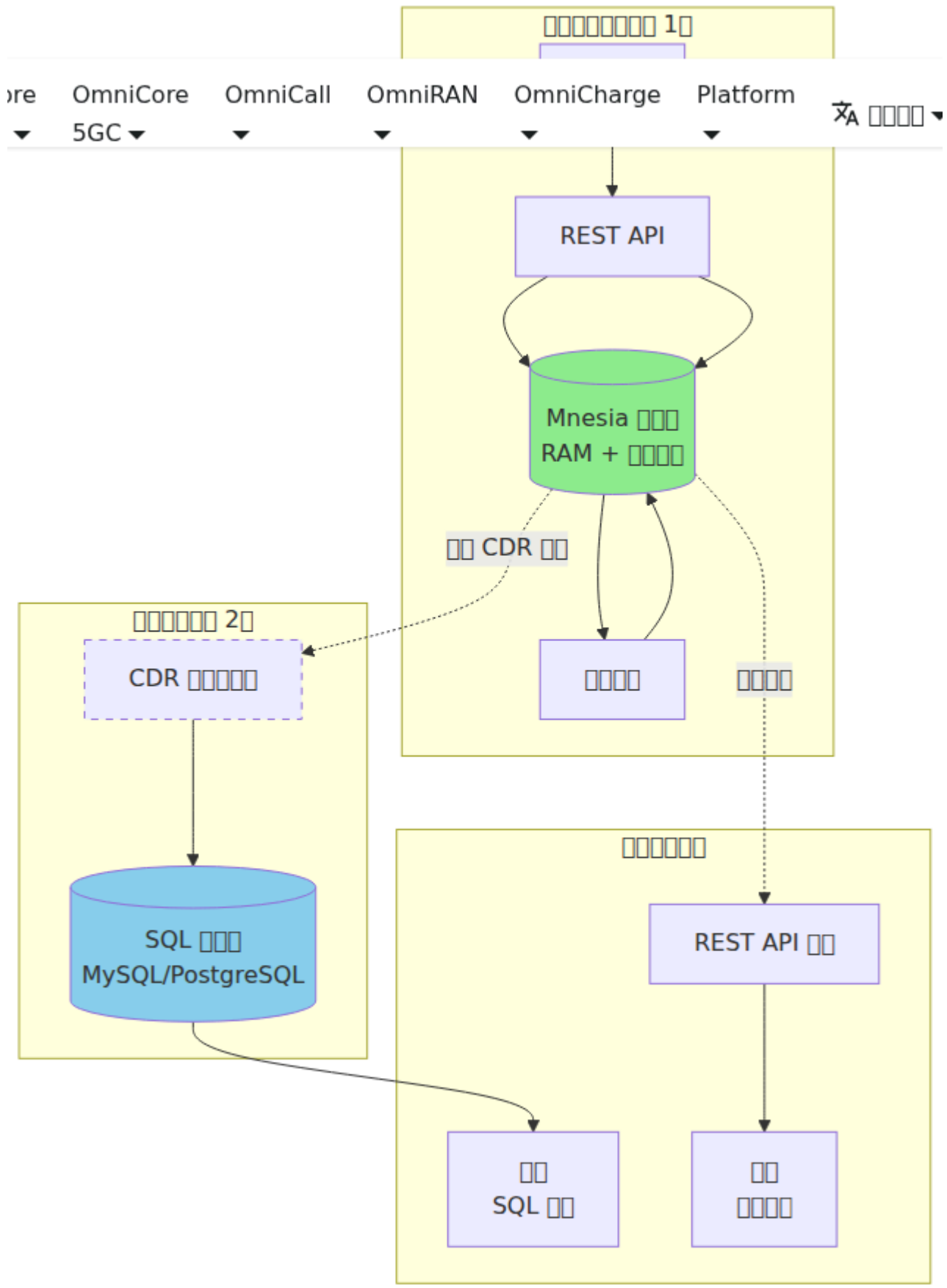
CDR

- CDR
- SQL
- CDR 100
- 100ms
- CDR

```
# config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,      # CDR
  batch_insert_flush_interval_ms: 100 #
```

SQL

-
-
-
- CDR
-
-



□□□

- □□□□□□□□□□

- 資料庫設計
- 資料庫設計
- 資料庫設計 SQL

資料庫設計

資料庫 < 24 資料庫

- 使用 Mnesia 提供 REST API
- 資料庫
- 資料庫設計
- 資料庫設計

資料庫 > 24 資料庫

- 使用 SQL 提供 CDR 查詢
- 使用 SQL 查詢
- 資料庫設計
- 資料庫設計

資料庫

1. 資料庫設計 disc_copies 資料庫設計
2. 資料庫設計 CDR 查詢 SQL 資料庫設計
3. 資料庫設計
4. 資料庫設計 Mnesia+ 資料庫設計 SQL 資料庫設計

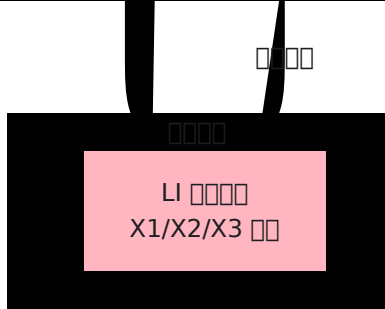
1.5 資料庫設計

OmniMessage SMS 提供 REST API 資料庫設計

資料庫



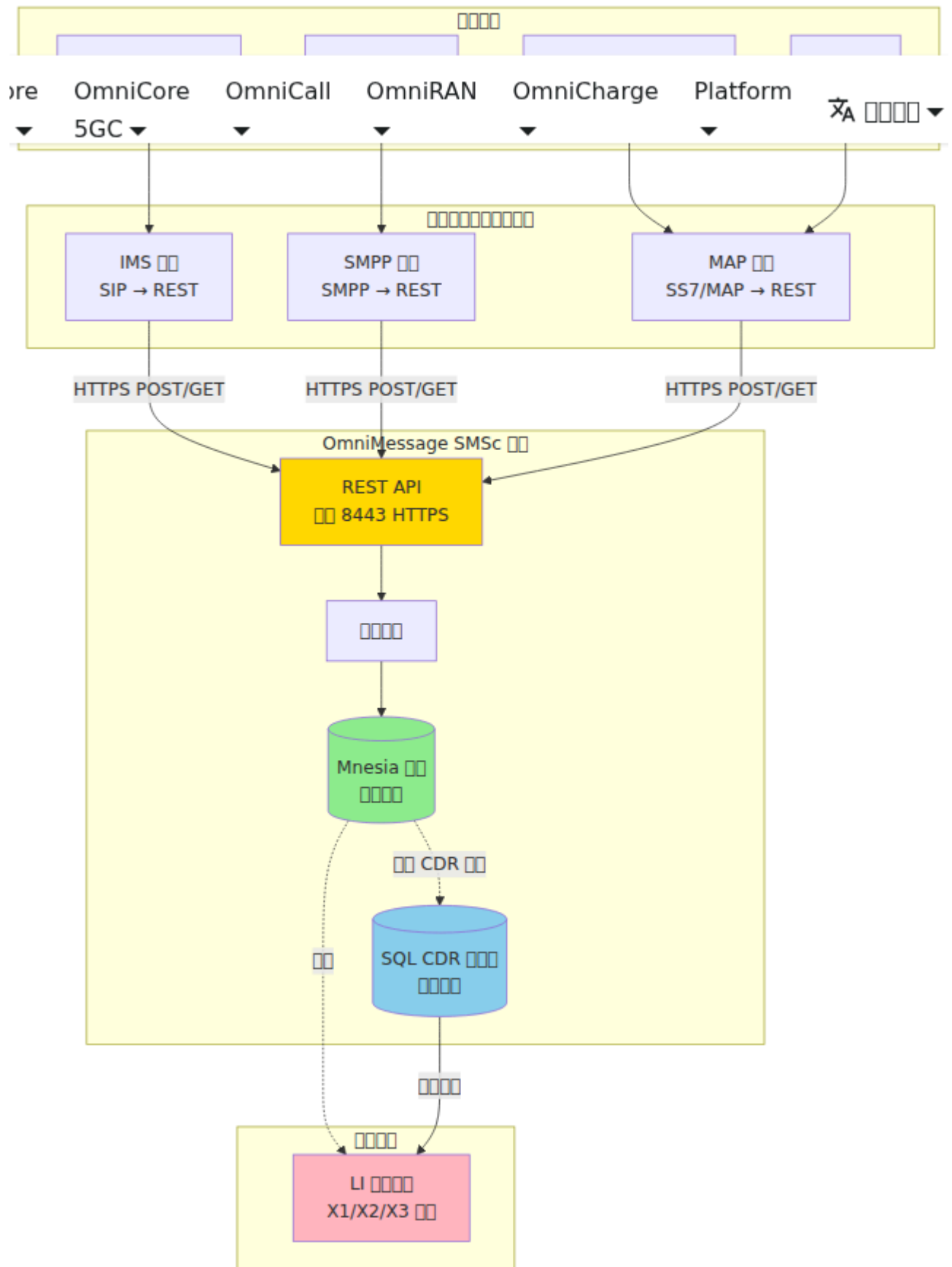
HTTPS POST/GET HTTPS POST/GET HTTPS POST/GET



□□□□□□□□

1. IMS/SIP

IMS SIP SMS-over-IP IMS SIP SCS REST API



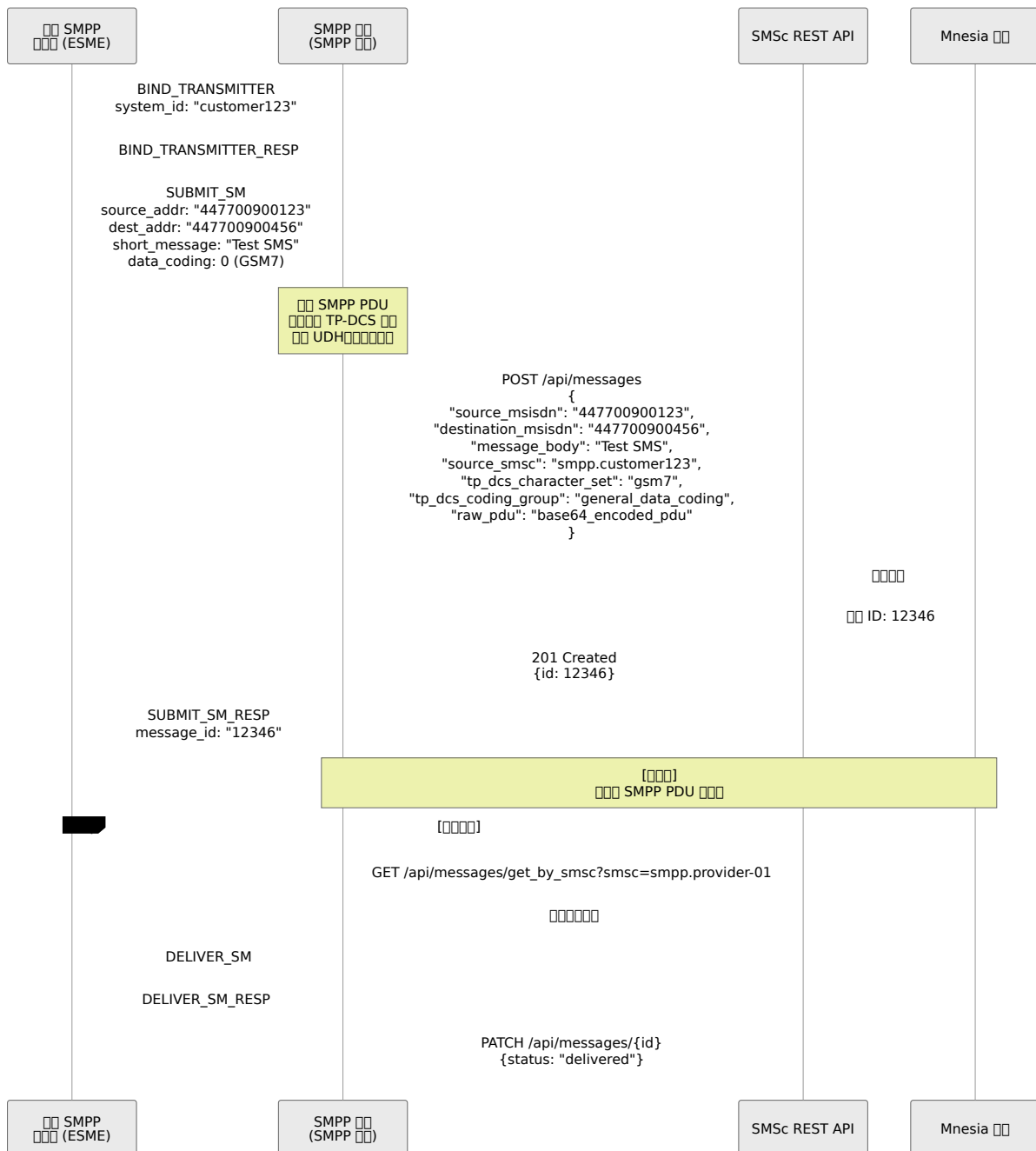
IMS

- IMS

- P-Asserted-Identity SIP
- SIP Call-ID
- IMS P-Access-Network-Info
- IMS HSS

2. SMPP

SMPP PDU SMPP PDU SMPP REST API

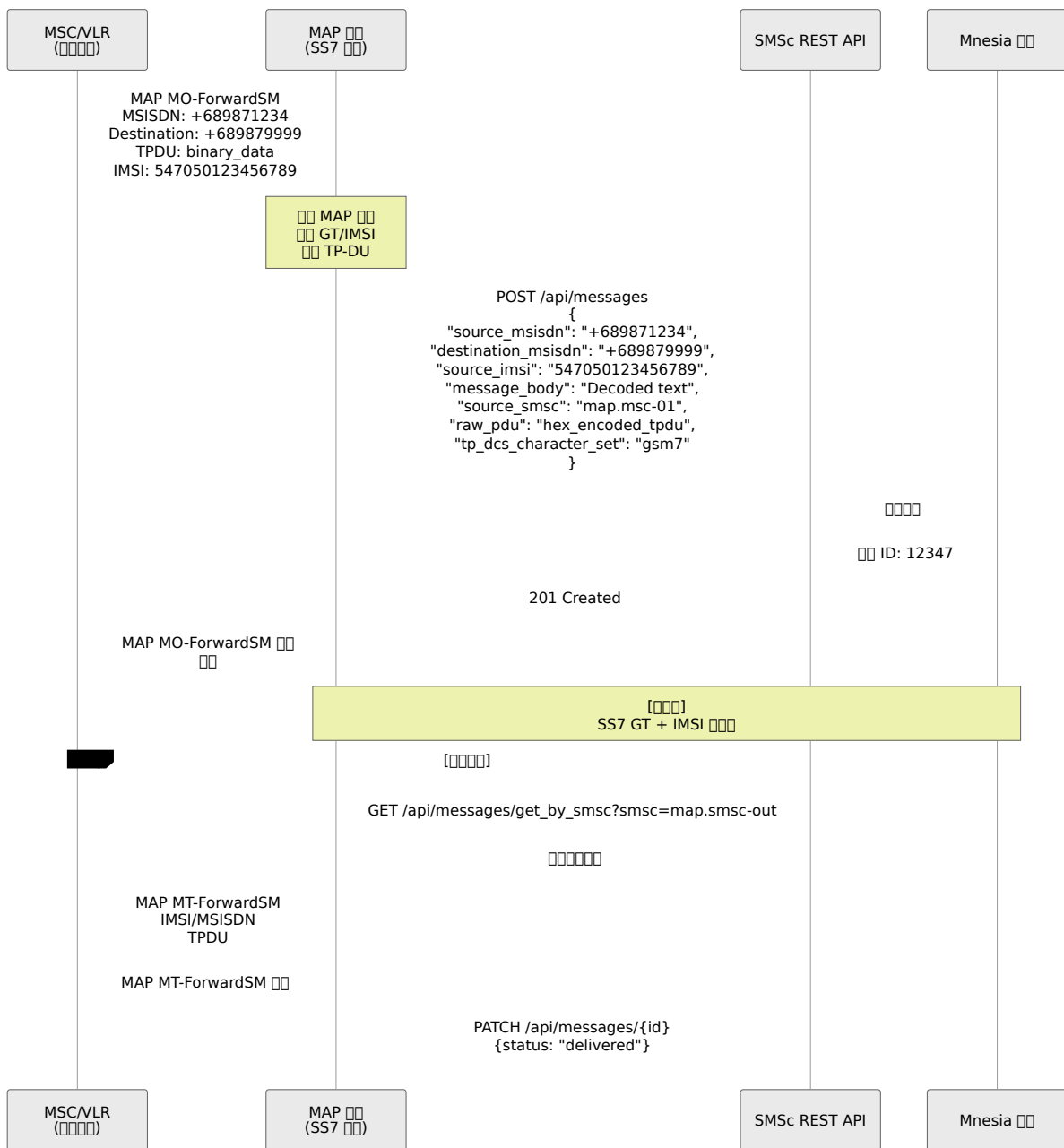


SMPP

- SMPP PDU
- DCS
- UDH
- ESME system_id
- TON/NPI
-

3. SS7/MAP

SS7 MAP REST API



SS7/MAP

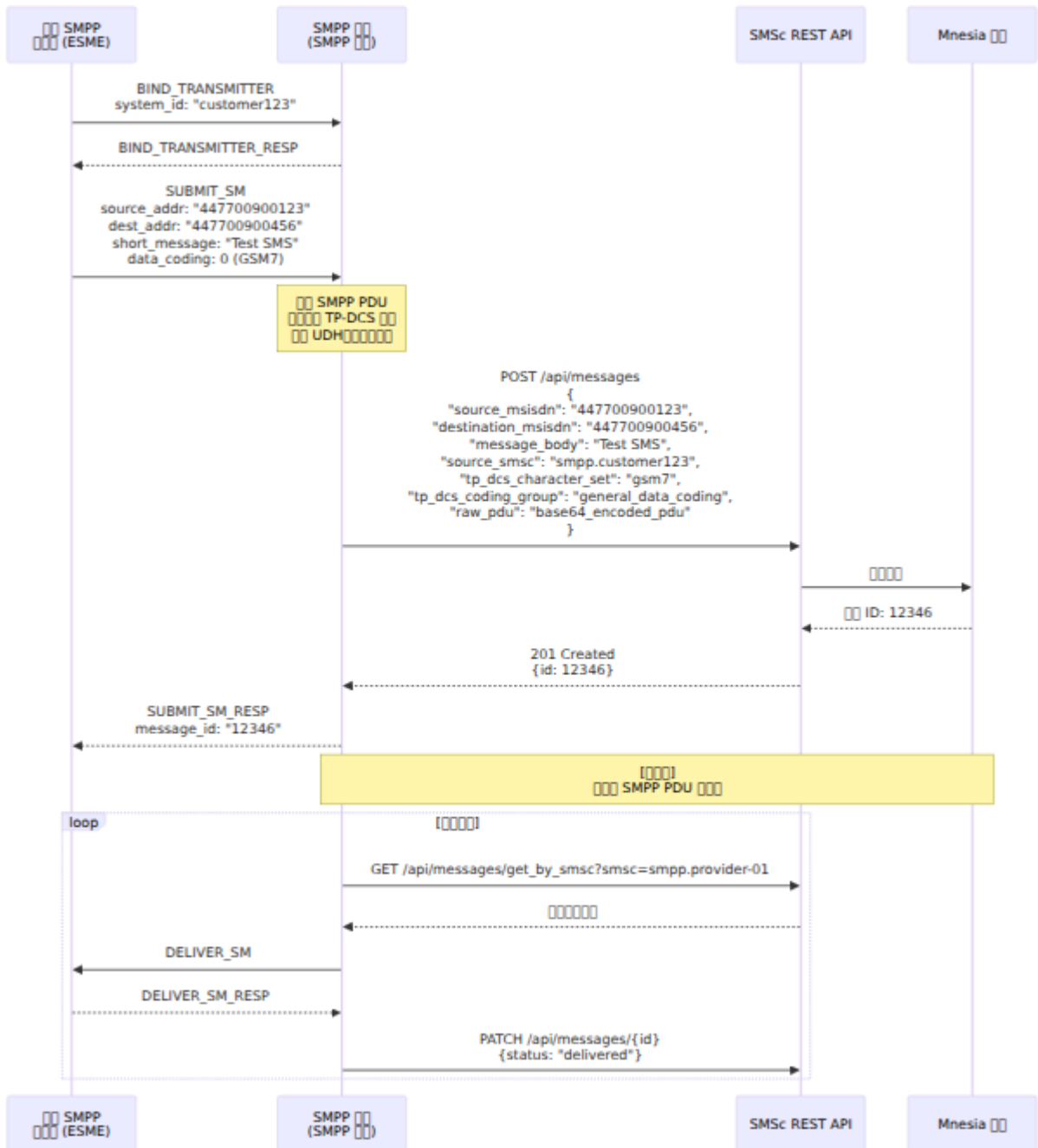
- MAP 與 IMSI
- 號碼 (GT) 碼
- MSC/VLR 與號碼
- SCCP 與號碼
- MAP 與
- TP-與號碼

號碼

號碼與號碼 IMS/SIP 與 SMPP 與 SS7/MAP 與號碼 SMSc 與號碼

1. 號碼與號碼
2. 與 **CDR** 與號碼與 CDR 與
3. 號碼與號碼
4. 號碼與號碼 CDR 與

號碼



CDR

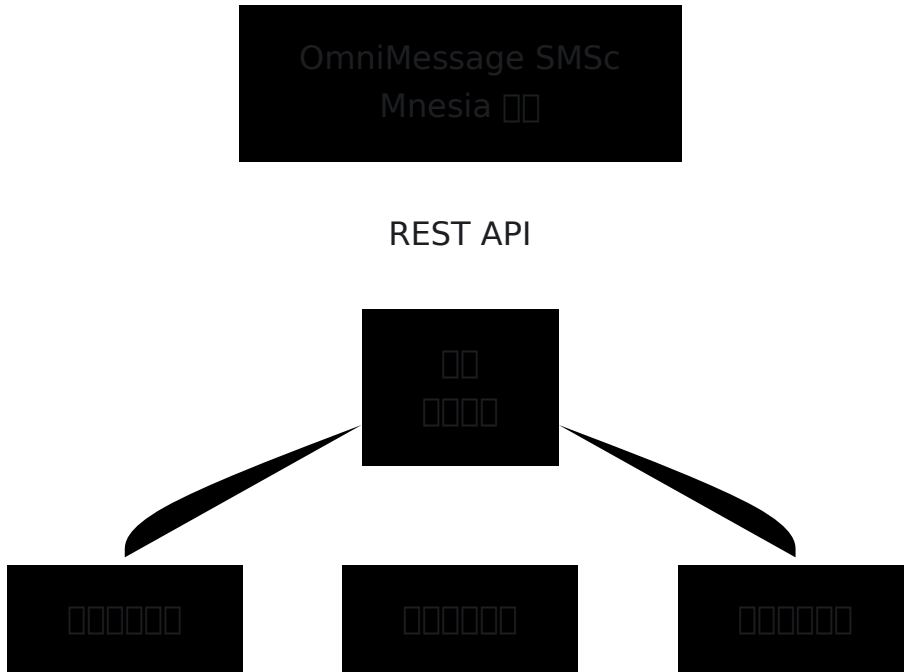
- source_smsc "ims.gateway-01" "smpp.customer123" "map.msc-01"
-
-

1.6

OmniMessage SMSc Mnesia SQL

1. REST API Mnesia

Mnesia 24



API

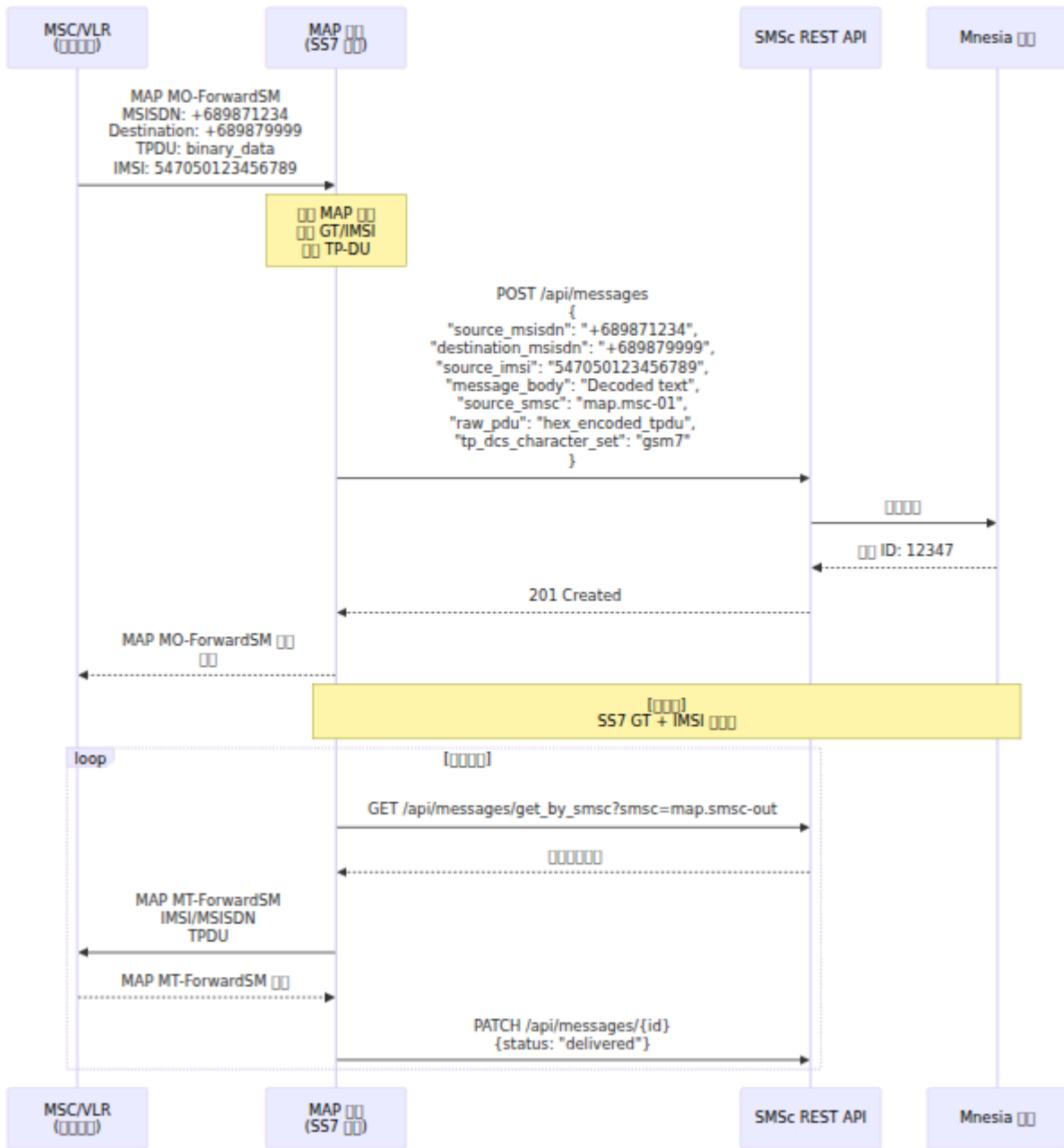
- GET /api/messages -
- GET /api/messages/{id} - Mnesia
- GET /api/messages/get_by_smsc?smc=X -
- Mnesia

Mnesia

- MSISDN
-
- SMSC
-
-

2. CDR SQL

SQL 数据库连接池/数据库连接池



SQL 数据库

- 数据库连接池
- 数据库连接池 SQL 数据库连接池
- 数据库连接池 SQL 数据库mysql/psql/DBBeaver
- 数据库连接池数据库连接池
- 数据库连接池数据库连接池
 - calling_number 数据库 - 数据库
 - called_number 数据库 - 数据库

- `message_id` -
- `submission_time` -
- `status` -
- `dest_smsc` -

CDR /

3. PubSub

- Phoenix PubSub
-
-
-
-
- WebSocket

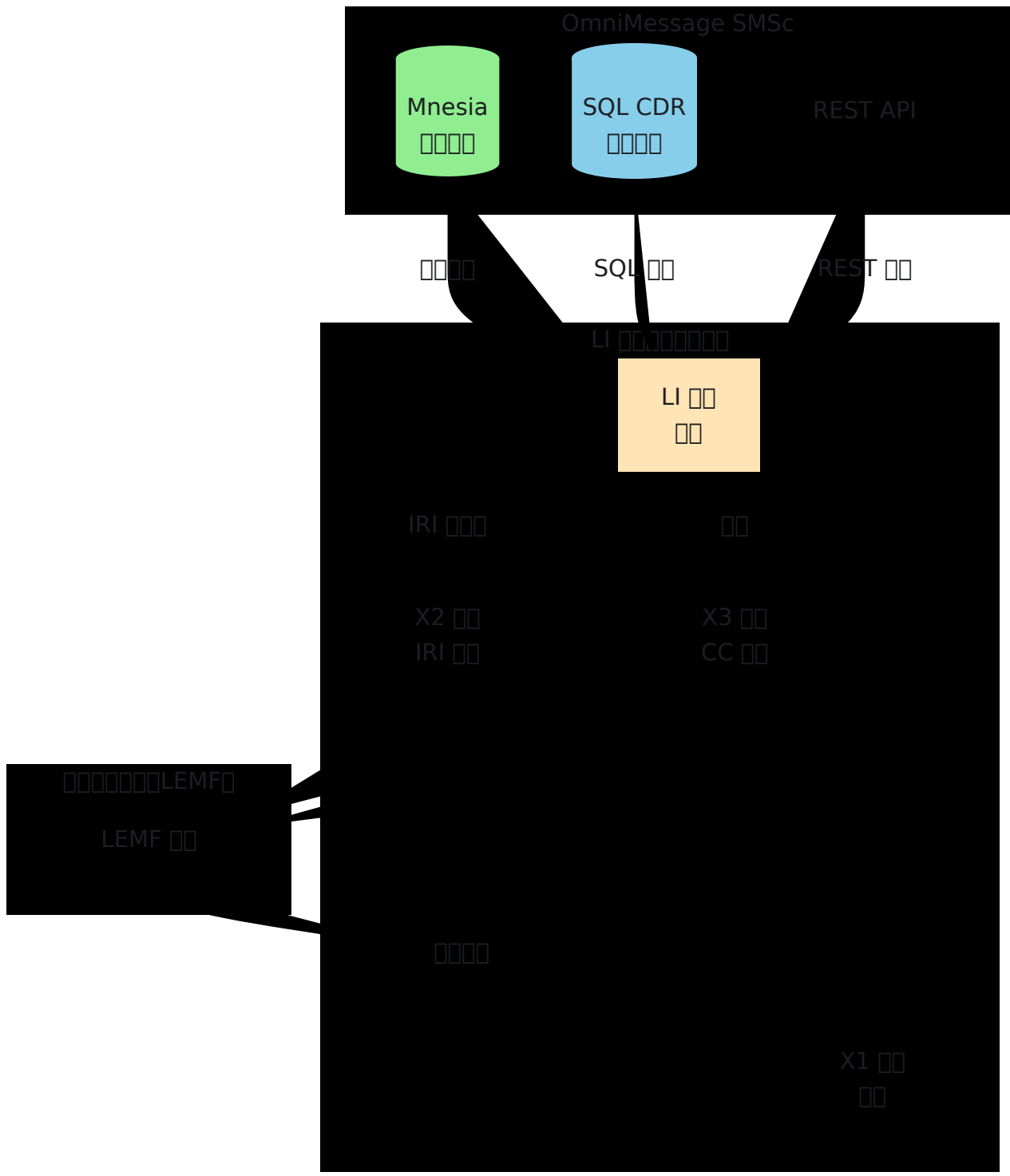
4.

- CDR CSV
- JSON
-
-
-

ETSI

OmniMessage SMSc ETSI SMSc X1/X2/X3 (LIMF)

ETSI LI



000000

X1 00 - 000000

- 000 000000000000000000000000
- 000 LEMF → LIMF0000
- 000
 - 00/000000000000MSISDN00IMSIO

- 0000000000000000
- 000000000000000000000000
- 00000000
- **SMSc** 0000
 - LIMF 0000000000000000
 - LIMF 00 SMSc CDR/API 0000000000
 - LIMF 00 X1 0000000000

X2 00 - IRI 0000000000000000

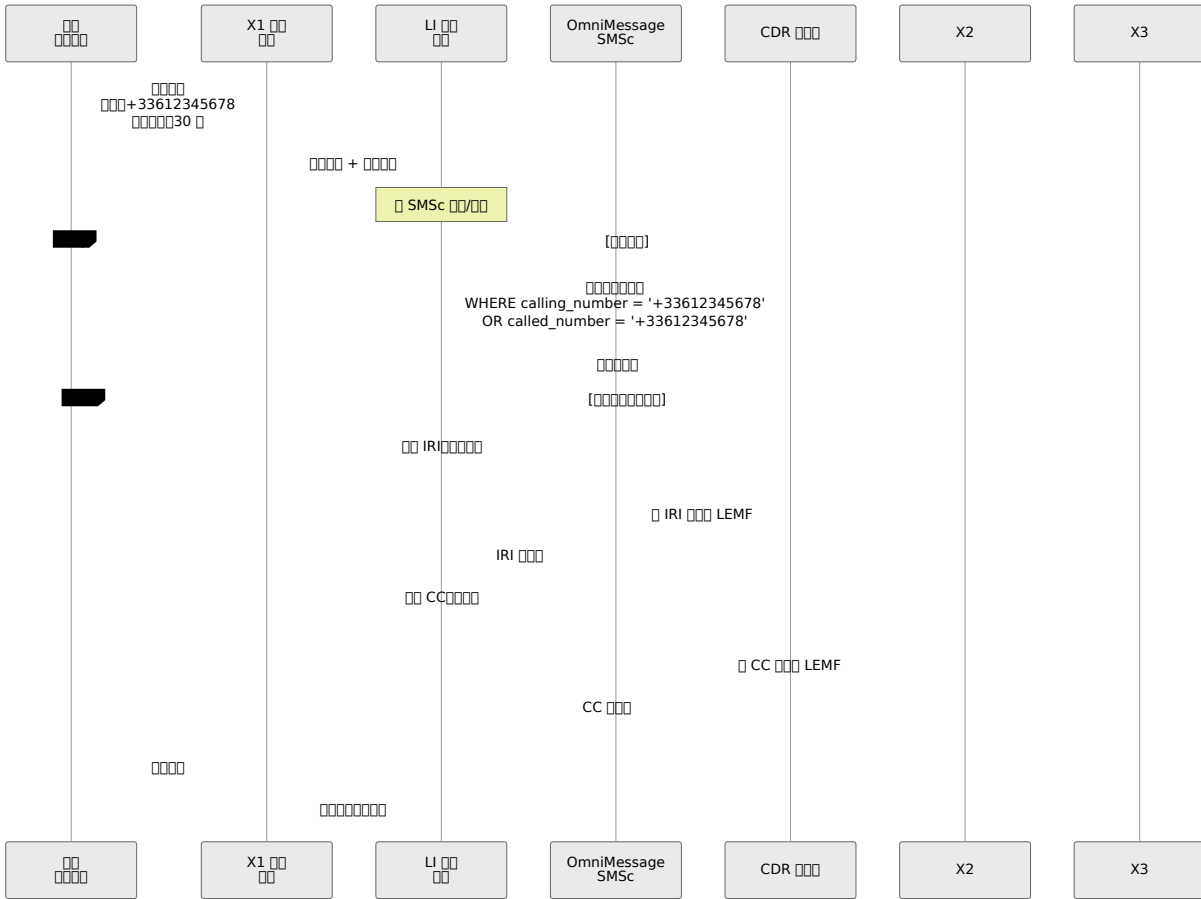
- 000 0000000000000000
- 000 LIMF → LEMF 0000
- 00000 00 ETSI TS 102 232-x 0 XML/ASN.1
- 00 **SMSc CDR** 0000
 - 00 ID
 - 000000 MSISDN 0
 - 00000000 MSISDN 0
 - IMSI 0000000000000000
 - 000000
 - 000000
 - 0000000000/00/0000
 - 000000
 - SMSC 00000000/0000
 - 00000000000000
- **SMSc** 0000
 - LIMF 00 CDR 000000000000000000000000
 - LIMF 0 CDR 000000 ETSI IRI 00
 - LIMF 00 X2 0 IRI 0000 LEMF

X3 00 - CC 000000000000

- 000 0000000000000000
- 000 LIMF → LEMF 0000
- 00000 00 ETSI TS 102 232-x 0000
- 00 **SMSc** 0000

- 000000000000
- 00 PDU0000 SMS 000
- 0000
- 000000
- TP-DCS 00
- 000000UDH0
- 0 **SMSc** 0000
 - LIMF 0 CDR message_body 00000000
 - LIMF 0000 PDU 00000000
 - LIMF 0 ETSI CC 000000
 - LIMF 00 X3 0 CC 000 LEMF

00000



SMSc 00000 LI 000

SMSc 欄名	X2 (IRI)	X3 (CC)	CDR 欄名
メッセージ ID	メッセージ ID	メッセージ	message_id
発信番号	A 欄	-	calling_number
着信番号	B 欄	-	called_number
送信時刻	送信時刻	-	submission_time
着信時刻	着信時刻	-	delivery_time
ステータス	ステータス	-	status
メッセージ本文	-	メッセージ	message_body
メッセージ PDU	-	メッセージ	(Mnesia/CDR)
発信 SMSc	発信 SMSc	-	source_smsc
着信 SMSc	着信 SMSc	-	dest_smsc
IMSI	メッセージ ID	-	(メッセージ)

LIMF 構築

ステップ 1

- LIMF 構築 CDR データ 1-60 日
- SQL 実行 X1 実行環境構築
- データ移行
- 実行環境構築 LI 実行環境構築

ステップ 2

- SMSc PubSub 構築
- LIMF 構築

- LIMF
-
-

3

- PubSub < 24
- CDR
-

- MSISDN
- IMSI
-
-
- API

-
- SMSC
-
- ENUM

- CDR
-
-
-

SQL


```

-- 検索範囲を指定
SELECT * FROM cdrs
WHERE calling_number = '+33612345678'
      OR called_number = '+33612345678'
ORDER BY submission_time DESC;

-- 検索範囲を指定
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;

-- 検索範囲を指定
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' AND called_number =
'+33687654321')
      OR (calling_number = '+33687654321' AND called_number =
'+33612345678')
ORDER BY submission_time;

```

2. 検索範囲を指定

2.1 検索範囲を指定

OmniMessage SMSc 検索範囲を指定する際に、ANSI 検索範囲を指定

2.2 検索範囲

2.2.1 TLS/SSL 検索

検索範囲

- TLS 1.2 (RFC 5246)
- TLS 1.3 (RFC 8446) - 検索
- SSL 2.0/3.0 検索範囲を指定

- TLS 1.0/1.1

- Erlang/OTP SSL/TLS
- Cowboy Web TLS
- Phoenix HTTPS

Erlang/OTP

TLS 1.3

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256

TLS 1.2

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

- ECDHE/DHE PFS
- Diffie-Hellman 2048
-
- SNI

- X.509
- RSA 2048 4096
- ECDSA
-

- 証明書を作成
- CA 証明書

TLS 設定

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

設定ファイル **CONFIGURATION.md**

設定

- HTTPS 対応 REST API 8443
- HTTPS 対応 Web 8086
- MySQL/PostgreSQL 対応 TLS

2.3 設定

2.3.1 設定

MySQL/MariaDB 設定

- 設定
- AES-256 暗号化
- TDE

```
-- CDR 暗号化
ALTER TABLE cdrs ENCRYPTION='Y';
```

PostgreSQL 設定

- 設定

- 00000000
- 000000pgcrypto 000

2.3.2 Mnesia 0000

Mnesia 0000

- 0000000000000000
- 000000000000LUKS0dm-crypt0
- 00 Erlang VM 0000000000

2.3.3 000000

000000000000

- 0000000000000000
- 000000000006000+ 0000000
- 00000000000000000000
- CDR 0000000000000000

000000

- TLS 0000000000 `priv/cert/`
- 00000000000000000000
- 0000000000

2.4 0000000000

2.4.1 API 0000

REST API 0000

- HTTPS/TLS 0000000000
- 0000000000SMSc 0000000000
- 00 IP 0000000000000000
- 00000000000000000000

000000

- IP
-
- 90
-

2.4.2

-
- TLS/SSL
- IP
- RBAC

```
# config/runtime.exs
config :sms_c, SmsC.Repo,
  username: "omnitouch",
  password: "omnitouch2024", #
  hostname: "localhost",
  ssl: true # TLS
```

```
--
CREATE USER 'li_readonly'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c.cdcs TO 'li_readonly'@'%';

--
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.cdcs TO 'analytics'@'%';
```

2.5 安全协议

2.5.1 认证

1 Erlang/OTP 认证

- SHA-256、SHA-384、SHA-512
- SHA-1
- MD5
- BLAKE2、OTP

2

-
-
-

2.5.2 加密

1

- AES
 - AES-128-GCM
 - AES-256-GCM
 - AES-128-CBC
 - AES-256-CBC
- ChaCha20-Poly1305

2

- 128
- 256

3

- TLS
-
-

2.5.3 暗号化

暗号化

- RSA 2048 鍵長 4096 鍵長
- ECDSA 楕円曲線暗号化
 - P-256 P-384 P-521 楕円曲線
- Ed25519 EdDSA

暗号

- TLS 暗号化
- 暗号
- 暗号

2.6 暗号化

2.6.1 暗号化

暗号化

- GSM 7 暗号化
- UCS-2 Unicode 16 暗号
- 8 暗号化
- Latin-1

TP-DCS 暗号化

- 暗号化
- 暗号
- 暗号化
- 暗号化

暗号化

- 暗号化
- 暗号化 SMSc 暗号
- 暗号化

2.6.2 2.6.2

SMPP

- SMPP
- TLS
-

IMS

- SIP
- SIP
- IMS

SS7/MAP

- SS7
- MAP
- SCCP/TCAP

SMSc

2.7

2.7.1

-
-
- API
-
-

- stdout/
- PCAP

- Prometheus

2.7.2

- Vault
- AWS Secrets Manager
- Erlang/OTP

- TLS
- API
- IP

- TLS
- API
- IP

2.8

2.8.1

TLS

```

# RSA 4096
openssl genrsa -out omnitouch.pem 4096

# CSR
openssl req -new -key omnitouch.pem -out omnitouch.csr

# Certificate
openssl x509 -req -days 365 -in omnitouch.csr -signkey
omnitouch.pem -out omnitouch.crt

# CA

```

• Erlang/OTP CSPRNG

- Erlang/OTP CSPRNG
- `/dev/urandom`
- ID

2.8.2

• 0600

- 0600
- `priv/cert/`
- PEM
-

• TLS

- TLS
-
- API

2.8.3

• `priv/cert/`

- `priv/cert/`
-

- 使用 ACME 证书 Let's Encrypt

其他

- 证书颁发机构
- TLS 使用 Diffie-Hellman 密钥交换
- 证书链

2.9 其他

其他 SMS 相关

2.9.1 使用 SS7/MAP 使用 SMS 使用

3GPP 使用 ETSI 使用

- **3GPP TS 23.040** 使用 SMS 使用 - 使用 SMS 使用
- **3GPP TS 23.038** 使用 - 使用 GSM7 使用 UCS-2
- **3GPP TS 29.002** 使用 MAP 使用 - SS7 使用 SMS
- **3GPP TS 23.003** 使用 - MSISDN 使用 IMSI 使用
- **ETSI TS 100 901** 使用
- **ETSI TS 100 902** 使用

SS7 使用

- **ITU-T Q.711-Q.716** 使用 SCCP
- **ITU-T Q.771-Q.775** 使用 TCAP
- **ITU-T Q.701-Q.710** 使用 MTP 使用 1-3
- **ETSI EN 300 356** 使用 No.7 - ISDN 使用 ISUP

SS7/MAP 使用

- **GSMA FS.07** 使用 SS7 使用 - 使用
- **GSMA FS.11** 使用 SS7 使用
- **3GPP TS 33.117** 使用
- **ETSI TS 133 210** 使用 - IP 使用

SS7/MAP 使用

- **SMPP v3.4** 的 消息 格式 - 消息
- **SMPP v5.0** 的 消息 格式

SMPP 消息

- **TLS** 的 **SMPP** 消息 格式
- **SMPP** 消息 ID 的 格式
- 消息 **IP** 地址 的 格式

消息

- **GSM 03.40 (ETSI TS 100 901)** 的 消息 格式
- **GSM 03.38 (ETSI TS 100 900)** 的 消息 格式
- **GSM 04.11 (ETSI TS 100 942)** 的 消息 格式

消息

- **ITU-T T.50** 的 消息 格式
- **ISO/IEC 8859-1** 的 消息 格式
- **ISO/IEC 10646** 的 消息 格式

2.9.4 消息

TLS 消息

- **NIST SP 800-52** 的 TLS 消息 格式
- **NIST SP 800-131A** 的 消息 格式
- **RFC 7525** 的 TLS 消息 格式
- **RFC 8446** 的 TLS 消息 格式

消息

- **FIPS 197** 的 消息 格式 (AES)
- **FIPS 180-4** 的 消息 格式 (SHA-2 消息)
- **NIST SP 800-38D** 的 消息 格式 (GCM 消息)
- **RFC 7539** 的 IETF 消息 格式 (ChaCha20 消息)

消息

- **NIST SP 800-57** 標準
- **RFC 5280** 及び X.509 標準による CRL 標準

2.10 暗号化

2.10.1 暗号化

暗号化

- 暗号化/復号化
- 暗号化アルゴリズムの種類
- 暗号化モード
- 鍵管理
- 暗号化アルゴリズム GCM/Poly1305

2.10.2 暗号化

暗号化

- TLS 暗号化
- 暗号化アルゴリズム TLS
- 暗号化モード

暗号化

- 暗号化アルゴリズム
 - 暗号化モード
 - TLS 暗号化
 - 暗号化アルゴリズム
 - 暗号化モード
-

3. 資料庫

3.1 資料庫

資料庫

- 資料庫
- CDR 資料庫
- API 查詢 IP/資料庫
- 資料庫

資料庫

- 資料庫
- 資料庫
- 資料庫 SQL WHERE 查詢
- 資料庫

3.2 資料庫

資料庫

- 資料庫 24 小時 Mnesia 查詢
- CDR 資料庫 6 個月 2 個月
- Mnesia 資料庫 SQL
- 資料庫 CDR 查詢 cron

資料庫

- 資料庫
- 資料庫 UI/資料庫
- 資料庫
- 資料庫
- 資料庫

查詢

```
# config/runtime.exs
config :sms_c,
  # Mnesia
  message_retention_hours: 24,

  #
  delete_message_body_after_delivery: false, # true

  # CDR
  cdr_enabled: true,

  #
  batch_insert_batch_size: 100,
  batch_insert_flush_interval_ms: 100
```

CONFIGURATION.md

3.3

1. REST API

- HTTPS
- JSON
-
-

2.

- SQL
- SQL
- CDR
-

3.

- CSV
- JSON

- 0000000000
- 00000000

00000

IRI0000000000

- CDR 00000000
 - 00 ID
 - 00/0000
 - 0000000000000000
 - 00
 - 0000
 - SMSC 0000
 - 0000000000

CC00000000

- 000000000000
- 00 PDU 00
- 0000
- 00000000

00000

```
# 生成 CSV
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. 数据库操作

4.1 数据库

Elixir/Erlang 数据库

- Erlang VM 数据库
- 数据库
- 数据库
- 数据库

数据库

- 数据库mix.lock
- 数据库
- 数据库
- 数据库

4.2 安全

安全

- 安全
 - 8443 HTTPS REST API
 - 8086 HTTPS 安全
- 安全
- 安全 IP 安全
- DMZ 安全

安全

- 安全
- 安全
- 安全
- 安全 Erlang 安全

4.3 安全

安全

- 安全
- 安全
- 安全
- Syslog 安全
- 安全 ELK 安全

安全

- 安全
- 安全
- 安全
- TLS 安全
- 安全

安全

- Prometheus [README](#)
- [README](#)
- [README](#)
- [README](#)
- [README](#)

[README](#) [OPERATIONS_GUIDE.md](#) [METRICS.md](#)

4.4 [README](#)

[README](#)

- Erlang [README](#)
- Mnesia [README](#)
- [README](#)
- [README](#)

[README](#)

- Mnesia [README](#) disc_copies
- SQL [README](#) MySQL/PostgreSQL [README](#)
- CDR [README](#)
- [README](#)

[README](#)

- [README](#)
 - Mnesia [README](#)
 - [README](#)
 - [README](#)
-

5. 目次

5.1 目次

目次

- **README.md** - 目次
- **CONFIGURATION.md** - 目次
- **API_REFERENCE.md** - REST API 目次
- **OPERATIONS_GUIDE.md** - 目次
- **CDR_SCHEMA.md** - 目次
- **sms_routing_guide.md** - 目次
- **number_translation_guide.md** - 目次
- **METRICS.md** - Prometheus 目次
- **PERFORMANCE_TUNING.md** - 目次
- **TROUBLESHOOTING.md** - 目次

5.2 目次

- 目次 [目次]
- 目次 [目次]
- 目次 [目次]
- **Erlang/OTP** 目次 [目次]

5.3 目次

- **ANSI R226** 目次 [目次]
- 目次 [目次]
- 目次 [目次] GDPR 目次

6. 目次

目次/目次

- Omnitech Network Services Pty Ltd
- PO BOX 296, QUINNS ROCKS WA 6030, AUSTRALIA
-
- compliance@omnitech.com.au

-
- compliance@omnitech.com.au

- - compliance@omnitech.com.au
-

A

A.1

Parse error on line 11: ... Note over -----^ Expecting 'ACTOR', got 'NEWLINE'

SMS-C API

[←](#) | [README](#)

SMS-C REST API

- API
-
-
-
- API
- SMS PDU API
- API
- API
- API
- MMS API
- SS7 API
-
-
-

API

SMS-C REST API

URL

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

ポート: 8443 (HTTPS)

プロトコル: HTTPS (SSL/TLS)

レスポンス

レスポンスが JSON 形式

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

API の URL

API の URL は 1 つの URL に指定されます。

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

例

TLS の設定

クライアントが TLS を使用する

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

API キー

API キーは X-API-Key として指定されます。

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


IP 地址

通过 API 获取 IP 地址

成功

响应

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

失败

```
{  
  "errors": {  
    "detail": "IP 地址不存在"  
  }  
}
```

列表

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

分页

分页响应

API

```
GET /api/status
```

(200 OK):

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

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

-
-
-

API

GET /api/messages

请求

- `smc: frontend_name` - 发送 SMSC 名称
- `include-unrouted: true|false|1|0` - 是否包含未路由的消息
 - `false` 不返回未路由的消息
 - `true` 返回未路由的消息

响应

- `status` - 消息状态 `pending` `delivered` `expired` `dropped`
- `source_smc` - 发送 SMSC 名称
- `dest_smc` - 接收 SMSC 名称
- `limit` - 返回消息数量 100 到 1000
- `offset` - 偏移量

成功 (200 OK):

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

错误

SMSC

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

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

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

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

(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"
  }
}
```

□□□

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

□□□□□□□□

□□□□□□□□□□ ID□

□□□

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

{} {}

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

{} {} {} {}

- `dest_smsc` - {} {} {} {} {}
- `send_time` - {} {} {} {} {} ISO 8601
- `message_parts` - {} {} {} {} {} {} {} {}
- `message_part_number` - {} {} {} {} {} 1 {} {}
- `tp_data_coding_scheme` - SMS DCS {} {} {} {} "00" {}
- `source_imsi` - {} {} {} {} IMSI
- `dest_imsi` - {} {} {} {} IMSI

{} {} (201 Created):

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

curl

```
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"
  }'
```

curl ~70 秒/メッセージ送信 14ms

レスポンス

- メッセージ ID
- ステータス
- メッセージ

HTTP ステータス

レスポンスヘッダー

ボディ

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

レスポンス

202 (202 Accepted):

```
{
  "data": {
    "status": "accepted",
    "message": "メッセージ"
  }
}
```

□□□

```
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": "□□□□□□",
  "source_smsc": "bulk_api"
}'
```

□□□~4,650 □□/□□□□□□□□□□ 0.22ms

□□□□□□ 100ms □□□□□□□□□□□□□□

□□□□□

- □□□□□□□□> 100 msg/sec□
- □□□□ API □□□□□□□□ ID
- □□□□□□□□□□□□

□□□□□

□□□□□□□□□□

□□□

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

□□□

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


□□□□□□

- `dest_smsc` - □□□□
- `deliver_after` - □□□□
- `message_body` - □□□□□□
- `status` - □□□□

□□ (200 OK):

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

□□□

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

□□□□□□□□□□

□□□□□□□□□□□□

□□□

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

□□□

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

200 (200 OK):

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

200

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

200

200

200

200

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

200 (200 OK):

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

□□□□□

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

□□□

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

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□

□□□

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

□□ (204 No Content)

□□□

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

□□□□□□□□□□□□□□□□□□□□□□□□

SMS PDU API

PDU SMS

SMS

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

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

PDU SMS TPDU

(201 Created):

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

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

SMS

POST

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

202 Accepted

202 (202 Accepted):

```
{
  "data": {
    "status": "accepted",
    "message": "PDU 0001000B916407007009F0000004D4F29C0E"
  }
}
```

POST

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

PDU

概要

1. SMS PDU (3GPP TS 23.040)
2. DCS
3. CP-ACK, RP-ACK
4. IMSI, MSISDN
- 5.
6. PDU

詳細

- CP-ACK, CP-ERROR -
- RP-ACK, RP-ERROR, RP-SMMA -
-

API

概要

API

GET

```
GET /api/locations
```

(200 OK):

```
{
  "data": [
    {
      "id": 1,
      "msisdn": "+15551234567",
      "imsi": "001001000000001",
      "location": "mnc1.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"
    }
  ]
}
```

□□□

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

□□□□

□□□

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

□□ (200 OK):

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

□□□

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

□□/□□□□

□□ IMSI□□□□□□□□□□□□□□□□□□□□

□□□

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

□□□

```
{
  "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"
}
```


请求体

- `imsi` - 国际移动用户识别码
- `msisdn` - 手机号码

响应体

- `location` - MSC/VLR 名称
- `ran_location` - RAN/小区 ID
- `imei` - 国际移动设备识别码
- `ims_capable` - IMS VoLTE 支持
- `csfb` - 电路域回落
- `registered` - 注册状态
- `expires` - 过期时间
- `user_agent` - 用户代理

成功 (201 Created) 或失败 (200 OK):

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

命令

```
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
}'
```

HTTP PATCH /api/locations/:id HSS MME

Request

Headers

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

Request Body

Response (200 OK):

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

Headers

```
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"
}'
```

Request

Headers

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

Response (204 No Content)

□□□

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

□□□□□□□□□□□□□□□□

□□□□ **API**

□□□□□□ SMSC □□□

□□□□□□

□□□

```
GET /api/frontends
```

□□ (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"
    }
  ]
}
```

□□□

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

□□□□□□□□

□□□

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

□□ (200 OK)□□□□□□□□□□

□□□

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

□□□□□□□□□□□□□□□□

□□□□□□□□

□□□

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

□□ (200 OK):

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

□□□

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

□□□□□□

□□□

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

□□ (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",
      ...
    }
  ]
}
```

□□□

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

□□□□

□□□□□□□□□□

□□□

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

□□□

```
{
  "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"
  }
}
```

□□□□

- `frontend_name` - □□□□□□□□
- `frontend_type` - □□□ `smpp` `sip` `http` □□

□□□□

- `ip_address` - □□ IP
- `hostname` - □□□□□
- `uptime_seconds` - □□□□□□□□□□
- `configuration` - □□□□□□□

□□ (201 Created):

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

□□□

```
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"
}'
```

90 60-90

API

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


(200 OK):


```
{
  "data": [
    {
      "event_epoch": 1698672000,
      "name": "message_inserted",
      "description": "消息插入",
      "event_source": "node1@server.example.com"
    },
    {
      "event_epoch": 1698672001,
      "name": "message_routed",
      "description": "消息路由 route_id=42 消息 uk_gateway",
      "event_source": "node1@server.example.com"
    },
    {
      "event_epoch": 1698672005,
      "name": "message_delivered",
      "description": "消息送达",
      "event_source": "node2@server.example.com"
    }
  ]
}
```

消息

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

消息类型

- `message_inserted` - 消息插入
- `message_routed` - 消息路由
- `message_delivered` - 消息送达
- `message_failed` - 消息失败
- `message_dropped` - 消息丢弃
- `auto_reply_sent` - 自动回复发送
- `number_translated` - 号码翻译 
- `routing_failed` - 路由失败
- `charging_failed` - 计费失败

□□□□

□□□

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

□□□

```
{
  "message_id": 12345,
  "name": "custom_event",
  "description": "□□□□□□□□",
  "event_source": "external_system"
}
```

□□ (201 Created):

```
{
  "data": {
    "message_id": 12345,
    "name": "custom_event",
    "description": "□□□□□□□□",
    "event_source": "external_system",
    "event_epoch": 1698672010
  }
}
```

□□□

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

000007 000000

MMS API

0000000000MMS0000

00 MMS 00

000

```
GET /api/mms_messages
```

00 (200 OK)0000 SMS 00000000 MMS 00

00 MMS 00

000

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

000

```
{
  "source_msisdn": "+15551234567",
  "destination_msisdn": "+447700900000",
  "subject": "00",
  "content_type": "image/jpeg",
  "content_location": "https://cdn.example.com/media/12345.jpg",
  "message_size": 524288
}
```

00 (201 Created)0000 MMS 0000

SS7 API

SS7 API

SS7 API

API

```
GET /api/ss7_events
```

200 OK:

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

SS7 API

API

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

API

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

{} (201 Created) {} {} {} {} {}

□□□□

HTTP □□□□

□□	□□	□□
200	OK	□□□□
201	Created	□□□□□□
202	Accepted	□□□□□□□
204	No Content	□□□□
400	Bad Request	□□□□□□
401	Unauthorized	□□□□
403	Forbidden	□□□□
404	Not Found	□□□□□
422	Unprocessable Entity	□□□□
429	Too Many Requests	□□□□□□
500	Internal Server Error	□□□□□
503	Service Unavailable	□□□□□

JSON 格式

```
{
  "errors": {
    "detail": "destination_msisdn 必填"
  }
}
```

错误码

错误码	原因	描述
"destination_msisdn is required"	必填	destination_msisdn 必填
"Invalid phone number format"	格式错误	不符合 E.164 格式 +15551234567
"Message too long"	长度限制	消息长度超过限制
"No route found"	路由失败	找不到路由
"Charging failed"	OCS 失败	计费失败
"Message not found"	消息 ID 不存在	消息 ID 不存在
"Frontend not registered"	SMSC 未注册	SMSC 未注册

□□□□

□□□□

□□	□□	□□
POST /api/messages	100 req/sec	□□ IP
POST /api/messages/create_async	1000 req/sec	□□ IP
POST /api/messages_raw	100 req/sec	□□ IP
GET /api/*	1000 req/sec	□□ IP

□□□□□

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

□□□□□□

□□ (429 Too Many Requests):

```
{  
  "errors": {  
    "detail": "□□□□□□□□ 5 □□□□□"  
  }  
}
```


5. API

1. 5xx
2. 4xx
- 3.
- 4.
- 5.

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"API : {e}")
            return None

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

        #
        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"API 错误: {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"API 错误: {e}")
            return False

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

# 发送消息
result = client.submit_message("+15551234567", "+447700900000",
"Hello")
print(f"消息 ID: {result['id']}")

# 批量发送消息
for i in range(1000):
    client.submit_message("+15551234567", f"+44770090{i:04d}",
f"Bulk {i}", async_mode=True)

# 获取待处理消息
while True:
    # 获取待处理消息
    messages = client.get_pending_messages("my_gateway")

    # 包含未路由的消息
    # messages = client.get_pending_messages("my_gateway",

```

```

include_unrouted=True)

for msg in messages:
    # 成功メッセージ
    success = deliver_via_smpp(msg)

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

time.sleep(5) # 5 秒間待つ

```

API 開発

第 1 章

- 概要
- 基本的 CRUD
- PDU 送信
- 送信
- 受信
- 送信

API

- 送信
- 送信
- 送信 API
- 送信 Webhook
- GraphQL API
- OAuth2

API 開発   

CDR (Call Detail Record) 管理

[← 概要](#) | [README](#)

このプロジェクトは、CDR (Call Detail Record) の管理を目的としています。

特徴

- 高パフォーマンス
- 柔軟な拡張性
- 豊富な機能
- SQL 対応
- 簡単インストール
- 詳細なログ
- 柔軟な設定
- 高度なセキュリティ
- 高度なパフォーマンス

インストール

`cdrs` は、SMS 管理のための CDR 管理ツールです。

- インストール
- データベースの作成
- 設定
- 実行

CDR 管理には Mnesia を使用しています。

- インストール
- 設定
- 実行
- Mnesia の管理



MySQL / MariaDB

```
CREATE TABLE cdrs (  
  id BIGINT AUTO_INCREMENT PRIMARY KEY,  
  
  --   
  message_id BIGINT NOT NULL,  
  
  --   
  calling_number VARCHAR(255) NOT NULL,  
  called_number VARCHAR(255) NOT NULL,  
  
  -- SMSC   
  source_smsc VARCHAR(255),  
  dest_smsc VARCHAR(255),  
  
  --   
  origin_node VARCHAR(255),  
  destination_node VARCHAR(255),  
  
  --   
  submission_time DATETIME NOT NULL,  
  delivery_time DATETIME,  
  expiry_time DATETIME,  
  
  --   
  status VARCHAR(50) NOT NULL,  
  delivery_attempts INT DEFAULT 0,  
  message_parts INT,  
  deadletter BOOLEAN DEFAULT FALSE,  
  
  --   
  message_body TEXT,  
  
  --   
  inserted_at DATETIME NOT NULL,  
  updated_at DATETIME NOT NULL,  
  
  --   
  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

```
CREATE TABLE cdrs (  
  id BIGSERIAL PRIMARY KEY,  
  
  -- 消息 ID  
  message_id BIGINT NOT NULL,  
  
  -- 号码  
  calling_number VARCHAR(255) NOT NULL,  
  called_number VARCHAR(255) NOT NULL,  
  
  -- SMSC 号码  
  source_smsc VARCHAR(255),  
  dest_smsc VARCHAR(255),  
  
  -- 节点名称  
  origin_node VARCHAR(255),  
  destination_node VARCHAR(255),  
  
  -- 时间  
  submission_time TIMESTAMP NOT NULL,  
  delivery_time TIMESTAMP,  
  expiry_time TIMESTAMP,  
  
  -- 消息状态  
  status VARCHAR(50) NOT NULL,  
  delivery_attempts INTEGER DEFAULT 0,  
  message_parts INTEGER,  
  deadletter BOOLEAN DEFAULT FALSE,  
  
  -- 消息内容  
  message_body TEXT,  
  
  -- 插入和更新时间  
  inserted_at TIMESTAMP NOT NULL,  
  updated_at TIMESTAMP NOT NULL  
);  
  
-- 索引  
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);
```

□□□□

□□

□□	□□	□□	□□
id	BIGINT	NO	CDR □□□□□□□□

□□□□

□□	□□	□□	□□
message_id	BIGINT	NO	□□ SMS-C □□□□□□□□□□□□□□ Mnesia □□□□□□ ID□

□□□□

□□	□□	□□	□□
calling_number	VARCHAR(255)	NO	□□□□□□□ MSISDN□□□□□□□□□□ E.164 □□□□□□+15551234567□□
called_number	VARCHAR(255)	NO	□□□□□□□ MSISDN□□□□□□□□□□ E.164 □□□□□□+15551234567□□

SMSC 表

列名	データ型	Nullable	説明
source_smsc	VARCHAR(255)	YES	送信元 SMSC の ID。API から提供された SMSC ID を入力し、NULL は許可されません。
dest_smsc	VARCHAR(255)	YES	送信先 SMSC の ID。API から提供された SMSC ID を入力し、NULL は許可されません。

送信元

送信元ノードの ID

列名	データ型	Nullable	説明
origin_node	VARCHAR(255)	YES	Erlang ノード ID。例: "sms@node1.example.com"。NULL は許可されません。
destination_node	VARCHAR(255)	YES	Erlang ノード ID。NULL は許可されません。

時刻

時刻は UTC で指定されます。

欄名	データ型	Nullable	説明
submission_time	DATETIME	NO	SMS-C の送信日時
delivery_time	DATETIME	YES	SMS-C の配信日時 NULL
expiry_time	DATETIME	YES	SMS-C の有効期限日時 NULL

計算列

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

列定義

欄名	データ型	Nullable	説明
status	VARCHAR(50)	NO	送信ステータス delivered expired failed rejected
delivery_attempts	INT	NO	送信回数 0 0-255
message_parts	INT	YES	SMS のメッセージ部分数 1 2+ NULL
deadletter	BOOLEAN	NO	エラーメッセージかどうか TRUE FALSE

備考

ステータス	説明	メッセージボディ	メッセージボディ
delivered	メッセージが正常に送信された	メッセージボディ	メッセージボディ
expired	メッセージが送信期限を過ぎた	メッセージボディ	NULL
failed	メッセージの送信に失敗した	メッセージボディ	NULL
rejected	メッセージが送信拒否された	メッセージボディ	NULL

メッセージボディ

フィールド名	データ型	必須	説明
message_body	TEXT	YES	メッセージの本文。SMS の場合、delete_message_body_after_delivery が true の場合は NULL になる。TEXT の最大長は 65,535 文字である。

注意事項

- メッセージの送信履歴は CDR に記録される。
- delete_message_body_after_delivery: true の場合は message_body が NULL になる。
- メッセージの送信に失敗した場合は message_body が NULL になる。

メッセージ履歴

フィールド名	データ型	必須	説明
inserted_at	DATETIME	NO	メッセージが CDR に記録された時刻。delivery_time/expiry_time を参照してください。
updated_at	DATETIME	NO	メッセージが CDR に記録された時刻。inserted_at を参照してください。

SQL 00

0000

000000000000 **CDR**

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

00000000

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

0000000000000000

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

0000

000 **SMSC** 00000000

```

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;

```

□□□□□□□□□□□□□□□□

```

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);

```

□□□□□□□

```

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;

```

□□□□

□□□□□□□□□□□□

```

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;

```

□□□□□□□□


```

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;

```

□□□□□□□□

```

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;

```

□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

```

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;

```

■■■■■■■■■■ CDR■■

```

-- ■■■■■■■■■■■■■■■■■■■■■2 ■■
SELECT COUNT(*) FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);

-- ■■■■■■■■■■■■■■■■■■■■■
DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR)
LIMIT 10000; -- ■■■■■■■■■■■■■■■■■■■■■

```

■■■■

■■■■■■■■■■■■■■

```

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;

```

□□

□□□□□□□□□□□□□□□□

□□□□	□	□□
PRIMARY	id	□□□□□□□□
idx_cdrs_message_id	message_id	□□□□ ID □❓❓❓ CDR
idx_cdrs_calling_number	calling_number	□□□□□□□□□□
idx_cdrs_called_number	called_number	□□□□□□□□□□
idx_cdrs_status	status	□□□□□□□□
idx_cdrs_submission_time	submission_time	□□□□□□□□□□□□
idx_cdrs_dest_smsc	dest_smsc	□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□

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

□□□□□□□□□□

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

□□□□□□□□□□

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

□□□□□□□□□□MySQL□□

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

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

□□□□□□□□□□

□□□□□□□□□□□□□□

欄名	MySQL/MariaDB	PostgreSQL	説明
<code>id</code>	BIGINT AUTO_INCREMENT	BIGSERIAL	64 ビット
<code>message_id</code>	BIGINT	BIGINT	64 ビット
メッセージ本文	VARCHAR(255)	VARCHAR(255)	最大 255 文字
<code>message_body</code>	TEXT	TEXT	MySQL: 最大 65,535 文字 PostgreSQL: 最大 1GB
時刻	DATETIME	TIMESTAMP	UTC 時刻
フラグ	INT	INTEGER	32 ビット
有/無	BOOLEAN (TINYINT(1))	BOOLEAN	MySQL: 0/1

データベース

CDR データベーススキーマ

1. インストール

`config/runtime.exe` を実行

```
config :sms_c,  
  # 删除消息体  
  delete_message_body_after_delivery: true,  
  
  # 在 UI 中隐藏消息体  
  hide_message_body_in_ui: true,  
  
  # 在导出中隐藏消息体  
  hide_message_body_in_export: true
```

2. 数据脱敏

数据脱敏

```
-- 对 calling_number 和 called_number 进行脱敏  
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. 数据库加密

数据库加密

MySQL

```
-- 加密数据库  
ALTER TABLE cdrs ENCRYPTION='Y';
```

PostgreSQL 在 PostgreSQL 中启用 TDE 加密

4. 权限管理

创建 CDR 数据库

```
-- 创建数据库
CREATE USER 'billing_ro'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c cdrs TO 'billing_ro'@'%';

-- 创建数据库
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'@'%';
```

权限管理

权限

数据库权限

数据库	权限	数据库
数据库	18-24 天	FCC数据库
数据库	6 天 - 2 天	GDPR数据库
数据库	5-7 天	SOX数据库
数据库	6 天	HIPAA

数据库

1. 数据库 MySQL 8.0+ PostgreSQL 11+

```

-- MySQL 备份
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
);

-- 删除分区
ALTER TABLE cdrs DROP PARTITION p202510;

```

2. 备份

```

-- 创建 CDR 备份表
CREATE TABLE cdrs_archive LIKE cdrs;

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

-- 删除旧数据
DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);

```

3. 清理


```
#!/bin/bash
# cleanup_old_cdrs.sh - cron job

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

# MySQL command
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
```

Cron job

```
# cron job 2 days
0 2 * * * /usr/local/bin/cleanup_old_cdrs.sh >>
/var/log/sms_c/cleanup.log 2>&1
```

cron

mysql

mysql

```
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)  
);
```

```
-- 测试
```

```
INSERT INTO billing_rates (destination_prefix, description,  
  rate_per_message, rate_per_segment, effective_date) VALUES  
  ('+1', '00/000', 0.0050, 0.0050, '2025-01-01'),  
  ('+44', '00', 0.0080, 0.0080, '2025-01-01'),  
  ('+61', '0000', 0.0100, 0.0100, '2025-01-01'),  
  ('+', '0000', 0.0150, 0.0150, '2025-01-01');
```

0000

0 CDR 000000000000

```

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;

```

□□□□□□□□

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
```



- **CDR** - CDR
- **CDR** - CDR
- **API** - REST API CDR

SMS-C

← | [README](#)

SMS-C

-
-
- API
- Web UI
-
-
-
- Diameter Sh (HSS)
- ENUM
-
-
-
-
-

SMS-C

config/config.exs

-
-

- 数据库/数据库
- 数据库

config/runtime.exs

数据库/数据库

- 数据库
- 数据库
- 数据库 (OCS, ENUM)
- 数据库
- 数据库

config/prod.exs (数据库)

数据库

数据库 runtime.exs 数据库 API 数据库

SQL CDR 数据库

SMS-C 数据库 Mnesia 数据库 SQL 数据库 CDR数据库

数据库 SQL 数据库

数据库 SQL 数据库 CDR 数据库

数据库	版本	数据库	端口	数据库
MySQL	8.0+	Ecto.Adapters.MyXQL	3306	数据库
MariaDB	10.5+	Ecto.Adapters.MyXQL	3306	MySQL 数据库
PostgreSQL	13+	Ecto.Adapters.Postgres	5432	数据库JSON 数据库

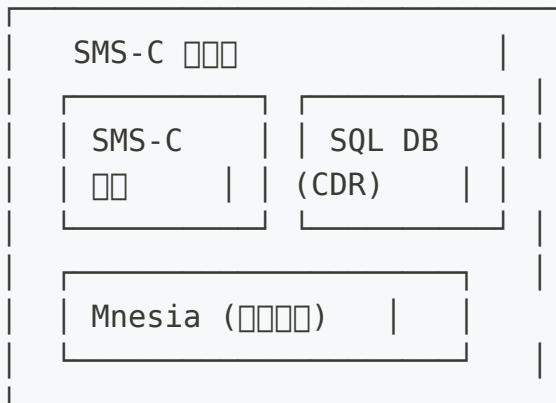
PostgreSQL - 数据库

- 数据库 JSON/JSONB 支持
- 支持 CDR 数据库
- 支持 PostgreSQL 数据库
- 支持数据库 PostGIS

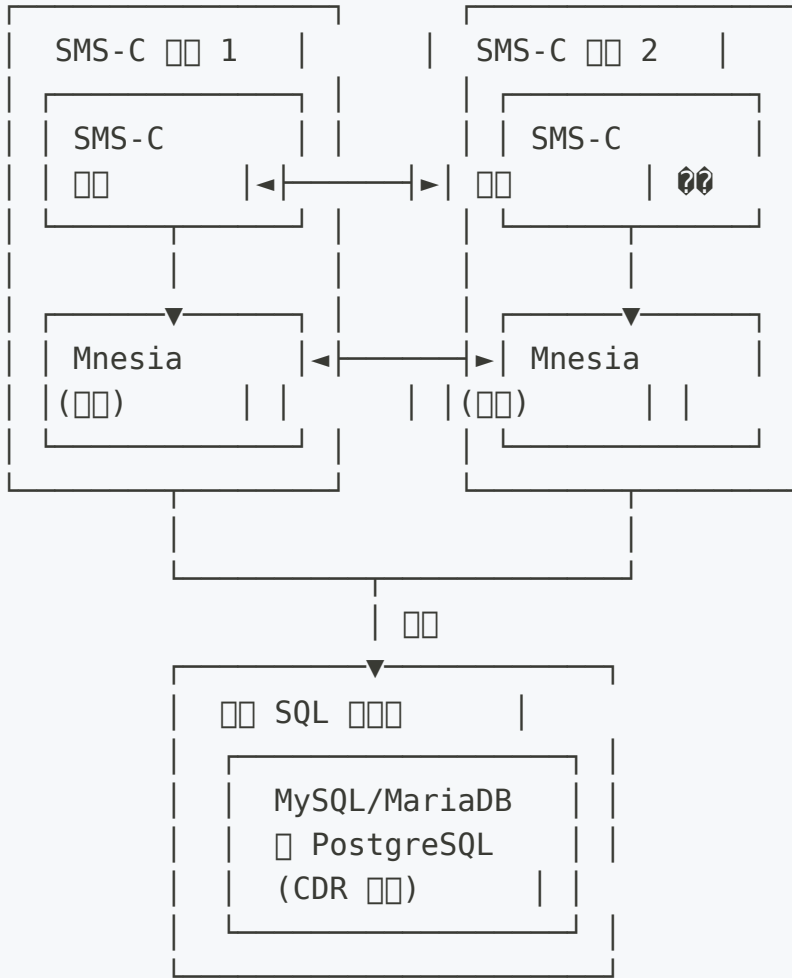
???

SQL CDR 数据库 SMS-C 数据库

数据库/数据库



数据库 - 数据库



SQL 数据库

- 数据库 CDR 数据
- 数据库
- 数据库 SMS-C 数据
- 数据库 SMS-C 数据 CDR 数据
- 数据库

データベース

データベース	接続数	接続数
開発	5-10	開発
開発 (< 100 msg/sec)	10-15	開発
開発 (100-1000 msg/sec)	20-30	開発
開発 (> 1000 msg/sec)	40-100	開発

開発 `pool_size = (開発 DB 接続数) * 1.5`

開発

開発環境



```
# 開発環境
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
```

開発環境

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

□□□□□

□□ Prometheus □□□□□□□□□□

- `ecto_pools_queue_time` - □□□□□
- `ecto_pools_query_time` - □□□□□□
- `ecto_pools_connected_count` - □□□□□

□□□□□□□□□□ 100ms - □□□□□□□□□□

API □□

REST API □□□□□□□□□□□□

□□ **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"
```

TLS/SSL □□

□□□□□ **TLS**□□□□□

```
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"
```

□□□□□□□ **TLS**□

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

API 証明書

証明書生成コマンド

```
# ディレクトリ作成  
mkdir -p priv/cert  
  
# キー生成  
openssl genrsa -out priv/cert/server.key 2048  
  
# CSR生成  
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"  
  
# 証明書生成 (有効期限 365 日)  
openssl x509 -req -days 365 -in priv/cert/server.csr \  
  -signkey priv/cert/server.key -out priv/cert/server.crt  
  
# 権限設定  
chmod 600 priv/cert/server.key  
chmod 644 priv/cert/server.crt
```

証明書生成コマンド CA Let's Encrypt 証明書生成コマンド

API 証明書

IP 証明書生成コマンド

```
# 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

# 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
```

API

Web UI

Web

Web UI

```
# 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"
```

Web UI

```
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"
```

□□□□□□□□□□

□□ Nginx □ Apache □□□□□□□□□□□□□□□□

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;

    # 认证
    auth_basic "SMS-C Admin";
    auth_basic_user_file /etc/nginx/.htpasswd;

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

    # WebSocket 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;
  }
}
```

□□□□




OmniMessage □□□□ HTTP □□□□□□□□□□□□□□□□ DNS SRV □□□□□□□□□□□□□□
HTTPS □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□ □□□□□□□□

□□□□

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


配置参数

参数名	类型	默认值	描述
<code>enabled</code>	布尔	<code>false</code>	是否启用
<code>dns_srv_domain</code>	字符串	<code>""</code>	DNS SRV 域名
<code>dns_poll_interval_ms</code>	整数	<code>30000</code>	DNS 轮询间隔
<code>health_check_interval_ms</code>	整数	<code>10000</code>	健康检查间隔
<code>registry_sync_interval_ms</code>	整数	<code>15000</code>	注册表同步间隔
<code>forward_retry_interval_ms</code>	整数	<code>5000</code>	转发重试间隔
<code>forward_max_retries</code>	整数	<code>50</code>	转发最大重试次数
<code>http_timeout_ms</code>	整数	<code>5000</code>	HTTP 请求超时
<code>api_port</code>	整数	<code>8443</code>	API URL 端口   
<code>static_peers</code>	列表	<code>[]</code>	静态 DNS SRV 列表

防火墙

配置防火墙规则，允许 8443 端口 HTTPS 流量。Erlang 端口 4369 + 9100-9200

```
# 配置防火墙规则
iptables -A INPUT -p tcp -s 10.0.1.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.2.0/24 --dport 8443 -j ACCEPT
```

部署

部署命令

□□□□

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

□□□□

- **60** - 1 □□□□□/□□□
- **1440** - 24 □□□□□□□□
- **4320** - 3 □□□□□□□
- **10080** - 7 □□□□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□

```
□□□□ = 2^(□□□□) □□
```

階数	容量
1	2 MB
2	4 MB
3	8 MB
4	16 MB
5	32 MB
6	64 MB
7	128 MB
8	256 MB

デフォルト値は `dead_letter_time_minutes` 30分

設定例

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

パラメータ

- **cleanup_interval_minutes** デフォルト値は10分
- **fingerprint_ttl_minutes** デフォルト値は5分
- **event_ttl_days** デフォルト値は7日

□□□□

□ OCS □□□□□□□□

□□□□

```
# 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"
```

□□□□

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

□□□□□□□□□□□□□□□□□□

□□□□□□□□

```
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"
```

□□□□□□□□

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

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

□□□□□□

□□□□□□□□□□□□

```
config :sms_c,
  charging_failure_action: :allow # [] :deny
```

- **:allow** - □□□□□□□□□□□□□□□□
- **:deny** - □□□□□□□□□□□□

OCS □□□□




□□ OCS □□□□

```
# [] OCS API
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
  }'
```

□□□□□

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

Diameter Sh (HSS)

OmniMessage  Diameter Sh  HSS 
 HSS 

```

# HSS dip
config :sms_c,
  diameter_enabled: true

# Diameter
config :diameter_ex,
  diameter: %{
    service_name: :omnimessage,
    listen_ip: "0.0.0.0",
    listen_port: 3868,
    decode_format: :map,
    host: "smc01",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{vendor_id: 10415, auth_application_id: 16_777_217,
acct_application_id: nil}
        ]
      }
    ],
    peers: [
      %{
        host: "dra01.epc.mnc005.mcc547.3gppnetwork.org",
        ip: "10.0.0.1",
        port: 3868,
        realm: "epc.mnc005.mcc547.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
}

```

名前	型	デフォルト	説明
<code>diameter_enabled</code>	boolean	false	HSS と Diameter の接続を有効にするかどうか
<code>mock_sh</code>	boolean	false	HSS と Diameter の接続をシミュレートするかどうか
<code>mock_sh_on_net_numbers</code>	array	[]	シミュレーションする MSISDN のリスト

Diameter と HSS の接続を有効にするには、`diameter_enabled` を true に設定する必要があります。

ENUM

DNS を通じて E.164 番号を解析する

ENUM を有効にする

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

DNS を通じて ENUM を有効にする

```
config :sms_c,
  enum_enabled: true,
  enum_domains: ["e164.arpa", "e164.org"],
  enum_dns_servers: [], # DNS サーバのリスト
  enum_timeout: 5000 # 5 秒
```


ENUM DNS ENUM

```
config :sms_c,  
  enum_enabled: true,  
  enum_domains: ["e164.internal.example.com", "e164.arpa"],  
  enum_dns_servers: [  
    {"10.0.1.53", 53}, # DNS  
    {"8.8.8.8", 53}, # Google DNS  
    {"1.1.1.1", 53} # Cloudflare DNS  
  ],  
  enum_timeout: 3000 # 3
```

ENUM

```
config :sms_c,  
  enum_domains: [  
    "e164.internal.example.com", #  
    "e164.carrier.net", #  
    "e164.arpa" #  
  ]
```

ENUM

```
enum_timeout: 2000 # 2
```

```
enum_timeout: 10000 # 10
```

ENUM DNS

ENUM BIND9

```
; e164.internal.example.com
$ORIGIN e164.internal.example.com.
$TTL 300

; +1-555-0100 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!" .

; +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!" .
```

ENUM

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

# 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!" .
```

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

□□□□□□□□

□□□□□□□□□□

```
config :sms_c,  
  translation_rules: [  
    %{  
      calling_prefix: nil,  
      called_prefix: "",  
      source_smsc: nil,  
      calling_match: "^(\\d{10})$",           # □□ 10 □□□  
      calling_replace: "+1\\1",             # □□□ +1  
      called_match: "^(\\d{10})$",  
      called_replace: "+1\\1",  
      priority: 100,  
      description: "□ 10 □□□□□□□□ +1",  
      enabled: true  
    }  
  ]  
]
```

□□□□□□□□

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

□□□□□□□□

```

%{
  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: "XXXXXXXXXXXX",
  enabled: true
}

```

XXXXXXXX

XXXXXXXX

```

%{
  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: "XXXXXXXXXXXXXXXXXXXX",
  enabled: true
}

```

XXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

□□□□□□□□

```
# config/runtime.exs
config :sms_c,
  sms_routes: [
    # □□□□□□
    %{
      calling_regex: nil,
      called_regex: ~r/^\+1/,
      source_smsc: nil,
      dest_smsc: "north_america_gateway",
      source_type: nil,
      enum_domain: nil,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: false,
      weight: 100,
      priority: 50,
      description: "□□□□",
      enabled: true
    },

    # □□□□□□□□
    %{
      calling_regex: nil,
      called_regex: ~r/^\+44/,
      source_smsc: nil,
      dest_smsc: "uk_gateway_1",
      source_type: nil,
      enum_domain: nil,
      auto_reply: false,
      auto_reply_message: nil,
      drop: false,
      charged: :default,
      on_net_only: false,
      weight: 70,
      priority: 50,
      description: "□□□□□□70%□",
      enabled: true
    },

    %{
```

```

calling_regex: nil,
called_regex: ~r/^\+44/,
source_smsc: nil,
dest_smsc: "uk_gateway_2",
source_type: nil,
enum_domain: nil,
auto_reply: false,
auto_reply_message: nil,
drop: false,
charged: :default,
on_net_only: false,
weight: 30,
priority: 50,
description: "☎☎☎☎☎☎30%",
enabled: true
},

# ☎☎☎☎ - ☎ SMPP ☎☎☎☎☎☎☎☎☎☎☎☎
%{
calling_regex: nil,
called_regex: nil,
source_smsc: "carrier_smpp_bind",
dest_smsc: "local_msc",
source_type: :smpp,
enum_domain: nil,
auto_reply: false,
auto_reply_message: nil,
drop: false,
charged: :default,
on_net_only: true,
weight: 100,
priority: 50,
description: "☎☎☎ X - ☎☎☎☎☎☎☎",
enabled: true
}
]

```

Web UI

```
# config/config.exs Web UI
config :sms_c,
  sms_routes: []
```

config/config.exs

Batch

config/config.exs

Batch

```
# config/config.exs
config :sms_c,
  batch_insert_batch_size: 100,      # 100
  batch_insert_flush_interval_ms: 100 # 100ms
```

config/config.exs

Batch Size	Batch Size	Batch Interval	Throughput	Batch Interval
200	200	200ms	~5,000 msg/sec	200ms
100	100	100ms	~4,500 msg/sec	100ms
50	50	20ms	~3,000 msg/sec	20ms
10	10	10ms	~1,500 msg/sec	10ms

□□□□

□□□□

```
# 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]
```

□□□□ :info □ :warning □□□□ :debug

□□□□□□□□

□□□□□□□□

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

□□□□□□□□

```
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]
```

□□□□

□□ **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
        # systemctl reload sms_c
    endscript
}
```

□□□□□□

□□□□□□

□□□□□□□□□□5,000+ □□/□□□

```
# 连接池
config :sms_c, SmsC.Repo,
  pool_size: 50

# 批量插入
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# 死信队列
config :sms_c,
  dead_letter_time_minutes: 1440 # 24 小时

# 默认充电
config :sms_c,
  default_charging_enabled: false

# 清理间隔
config :sms_c,
  cleanup_interval_minutes: 30
```

连接池

批量插入 < 20ms

```
# 消息池
config :sms_c, SmsC.Repo,
  pool_size: 20

# 消息队列配置
config :sms_c,
  batch_insert_batch_size: 20,
  batch_insert_flush_interval_ms: 10

# 死信队列
config :sms_c,
  dead_letter_time_minutes: 4320 # 3 天

# 默认充电
config :sms_c,
  default_charging_enabled: true,
  ocs_url: "http://ocs.local:2080/jsonrpc"
```

消息池

消息队列配置

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

# 消息队列配置
config :sms_c,
  batch_insert_batch_size: 1,
  batch_insert_flush_interval_ms: 10

# 日志配置
config :logger, :console,
  level: :debug

# 死信队列配置
config :sms_c,
  dead_letter_time_minutes: 60 # 1 分钟

# 默认计费配置
config :sms_c,
  default_charging_enabled: false
```

环境变量配置

环境变量配置

```
# 租户 1 配置
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

# 租户 2 配置
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
```

□□□□

□□□□□□

```
# □□□□□□
config :sms_c,
  cluster_nodes: [
    : "sms@us-east-1a.example.com",
    : "sms@us-east-1b.example.com",
    : "sms@us-west-1a.example.com" # □□□□□ DR
  ],
  smsc_node_name: "us-east-1a"
```

□□□□

□□□□□□□□□□

```
# □□□□□□
mix compile

# □□□□□□□□
mix ecto.create
mix ecto.migrate

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

# □□□□□□□□□□□□
iex -S mix phx.server
```

□□□□□□□□

□□□□□□□□□□□□

名前	説明	値
DB_USERNAME	データベースユーザー名	sms_prod_user
DB_PASSWORD	データベースパスワード	strong_password
DB_HOSTNAME	データベースホスト名	db.internal.example.com
DB_PORT	データベースポート	3306
DB_NAME	データベース名	sms_c_production
DB_POOL_SIZE	データベース接続プールサイズ	30
API_PORT	API リスナーポート	8443
API_LISTEN_IP	API リスナー IP	0.0.0.0
WEB_PORT	Web UI ポート	443
NODE_NAME	Erlang ノード名	sms@node1.example.com
ERLANG_COOKIE	Erlang Cookie	shared_cookie_value
OCS_URL	OCS API URL	http://ocs.local:2080/jsonrpc
OCS_TENANT	OCS テナント	sms.example.com

インストール手順

1. 依存パッケージをインストールする
2. 設定ファイルを生成する
3. サービスを起動する
4. サービスの状態を確認する
5. テストを実行する

6. 0000000000
7. 00000000000000000000
8. 0000000000000000
9. 000000000000
10. 0000000000000000

0000000000

00	0000	0000
0000000000	0000000000	0000000000
00000000	00/0000	00 DB_* 0000
API 0000	000000/IP 00	00 API_PORT 0 listen_ip
0000000000	Cookie 00000000 0	00 ERLANG_COOKIE000000 4369, 9100-9200
0000	OCS 0000	0000 ocs_url 0000
ENUM 000 0	DNS 00000000	00 DNS 0000000000
000	00000000	0000000000
000000	000000	00 sms_routes 000 Web UI

000000000000 00000000

メッセージング (Mnesia)

メッセージ

メッセージ Mnesia によって管理されます

```
config :sms_c,  
  # Mnesia によって管理  
  message_retention_hours: 24,  
  
  # 管理  
  retention_check_interval_minutes: 60
```

メッセージ

- 24-72 時間管理
- 4-8 時間管理
- 12-24 時間管理

メッセージ

- 1KB
- 10,000 件 ~ 10MB
- 100,000 件 ~ 100MB

CDRメッセージ

メッセージ CDR によって管理 Ecto によって管理

```
config :sms_c,  
  # / CDR  
  cdr_enabled: true
```

CDRメッセージ

- ID/

- 0/00 SMSC
- 00/000000000000
- 000000000000
- 00000000
- 0000000000000000

00000

- 000 CDR 00000
- 0000000000000000

00000

000000000000000000000000

```

config :sms_c,
  # 0000000 Mnesia 0000000
  delete_message_body_after_delivery: false,

  # 0 Web UI 0000000
  hide_message_body_in_ui: false,

  # 0 CSV 0000000000
  hide_message_body_in_export: false

```

000

00	00
delete_message_body_after_delivery: true	00 Mnesia 0000000000
hide_message_body_in_ui: true	000000000000
hide_message_body_in_export: true	000000000000

00000

□□□□□□□□

```
config :sms_c,  
  delete_message_body_after_delivery: true,  
  hide_message_body_in_ui: true,  
  hide_message_body_in_export: true,  
  cdr_enabled: true # □□□□□□ CDR
```

□□□□□□□□

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

□□□□

□□□□□□□□□□□□□□□□

```
[info] □□□□Mnesia□□□□24h□  
[info] CDR □□□□□□  
[info] □□□□□□□□□□□□  
[info] OCS □□□□□□url: http://..., □□: ...□
```

□□□□□□□□□□□□□□□□

📄📄📄📄

← 📄📄📄📄 | 📄📄📄 | 📄📄📄

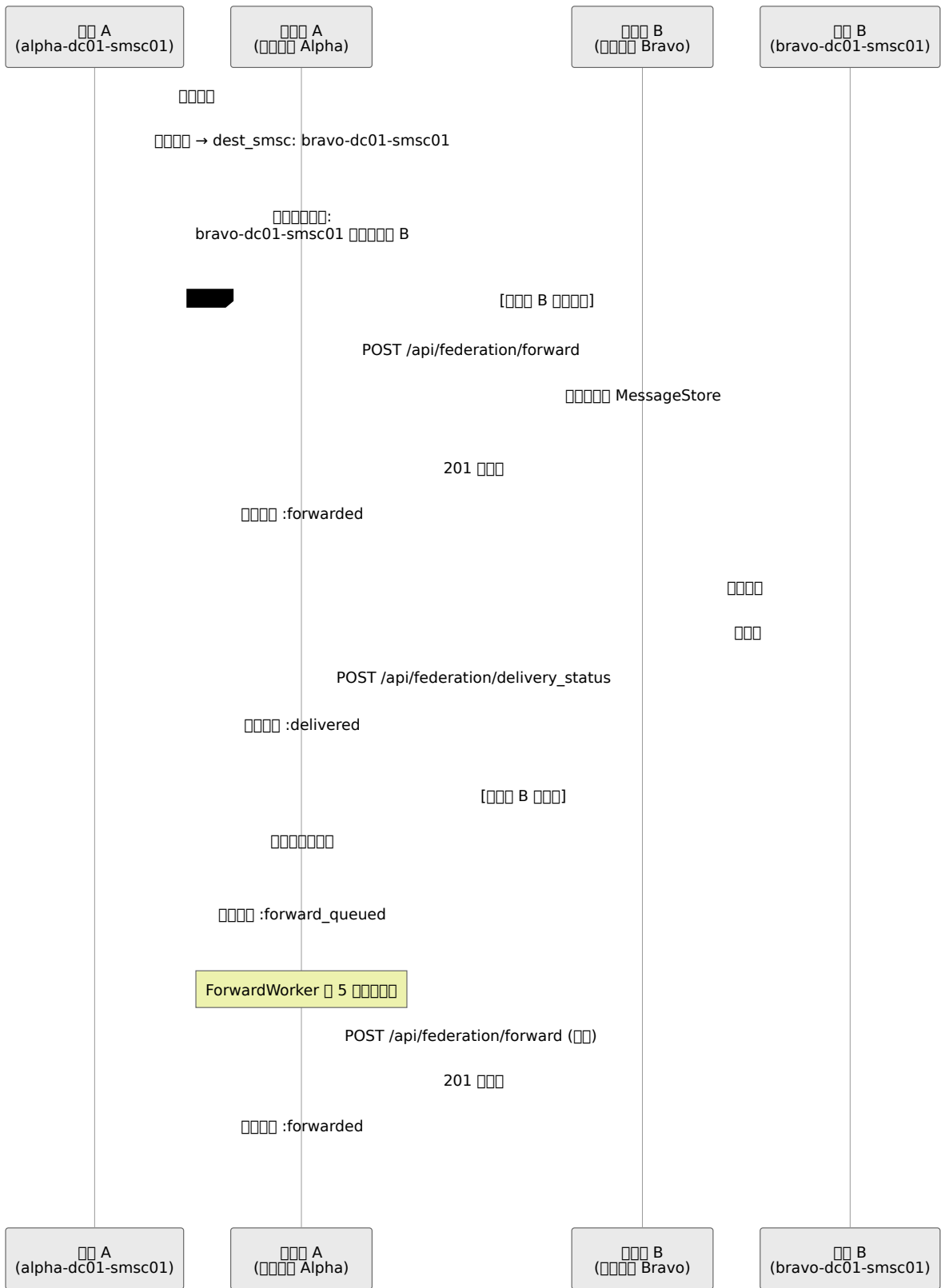
📄📄

OmniMessage 📄📄 **HTTP** 📄📄 📄📄📄📄📄📄📄📄📄📄📄📄📄——📄📄📄📄📄
📄📄📄📄📄📄📄📄📄📄📄📄 DNS SRV 📄📄📄📄📄📄📄📄📄📄📄📄 HTTPS 📄📄📄📄📄📄📄📄📄📄
📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

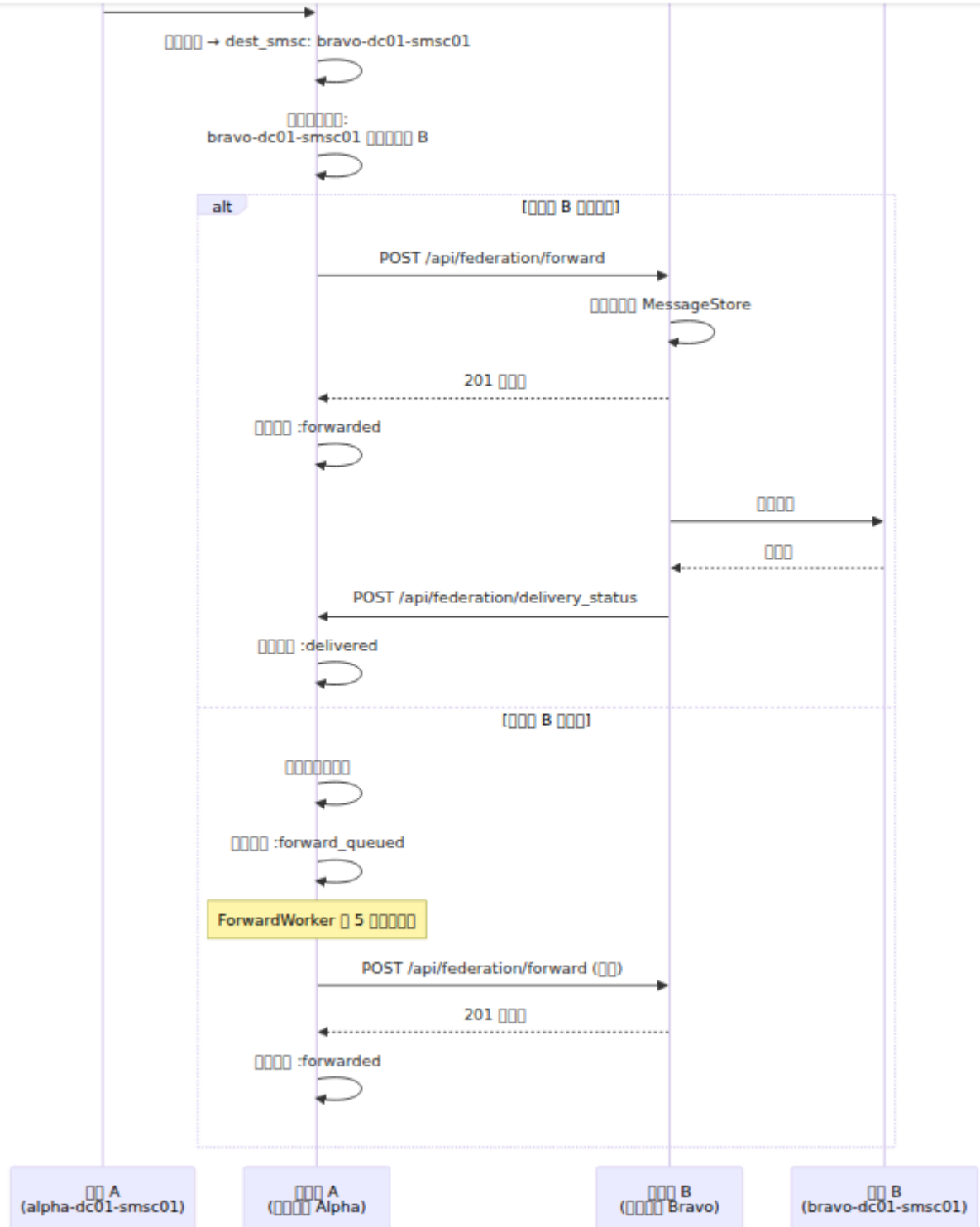
📄📄📄📄

📄📄	📄📄
📄📄	📄📄 WAN 📄📄📄📄📄📄📄📄
📄📄📄📄	📄 — 📄📄📄📄📄📄📄📄
📄📄📄	📄📄 TLS 📄 HTTPS
📄📄	📄📄 HTTPS 📄📄8443📄
📄📄📄📄	5-20 📄📄📄📄
📄📄📄📄	📄📄📄📄📄📄📄📄



□□□□□□□□

□□□□□□□□□□□□□□□□□□□□



XXXXXXXXXXXXXXXXXXXX

00	0000	0000	0000
00	000	00	000
00	00	0000	00
00	1-2 000	00	000
000	3 00000	00	000

00000

000000000000000000000000000000

00	00
:forwarded	0000000000000000
:forward_queued	0000000000000000000000000000
:forward_failed	000000000000000000000000

000000000000 :pending 00 :delivered 00 :failed 00 :retry 000000000000 Web UI 00 API
00000

00

000 config/runtime.exe 00 :federation 0000000000000000 00 000000000000

配置 DNS SRV

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

配置

```
# config/runtime.exs
config :sms_c, :federation,
  # 是否启用 - 默认为 false
  enabled: true,

  # 配置 DNS SRV 静态对等节点
  dns_srv_domain: "_smsc._tcp.smsc.example.com",

  # 配置 DNS SRV 轮询间隔
  dns_poll_interval_ms: 30_000,

  # 配置健康检查间隔
  health_check_interval_ms: 10_000,

  # 配置注册表同步间隔
  registry_sync_interval_ms: 15_000,

  # 配置转发重试间隔
  forward_retry_interval_ms: 5_000,

  # 配置转发失败后的最大重试次数
  :forward_failed_max_retries 50,
  forward_max_retries: 50,

  # 配置 API 的 HTTP 超时时间
  http_timeout_ms: 5_000,

  # 配置 DNS SRV 的 URL 和 API 端口
  api_port: 8443,

  # 配置静态对等节点 - 使用 dns_srv_domain 配置
  static_peers: []
```

□□□□

□□	□ □	□ □	□□	□□
enabled	□ □	□	false	□□□□□□ false □□□□□□□□ □□□□□□□□□□
dns_srv_domain	□ □ □	□	""	□□□□□□□□ DNS SRV □□□□ □□□□□ □□□□□□□ □
dns_poll_interval_ms	□ □	□	30000	DNS SRV □□□□□□□□□□□□ □□□□□□□□□□□□□□□□ DNS □□□
health_check_interval_ms	□ □	□	10000	□□□□□□□□□□□□□□□□□□ □□□
registry_sync_interval_ms	□ □	□	15000	□□□□□□□□□□□□□□□□□□ □□□□□
forward_retry_interval_ms	□ □	□	5000	□□□□□□□□□□□□□□□□□□ □□
forward_max_retries	□ □	□	50	□□□□□□□□□□□□□□□□□□ □□ :forward_failed□□□□ □□□□□□50 □□□ = ~4 □□□□ □□
http_timeout_ms	□ □	□	5000	□□□□□□□□□□ HTTPS □□□□□□ □□□□□□□□□□□□□□□□□□
api_port	□ □	□	8443	□ DNS SRV □□□□□□□□ URL □□ □□ API □□□□□□□ □□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□

名前	型	デフォルト	説明
<code>static_peers</code>	配列	<code>[]</code>	DNS SRV レコードを定義する。各レコードは <code>host</code> と <code>port</code> を含む。

例

DNS SRV レコードの例

```

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

```

名前	型	デフォルト	説明
<code>host</code>	文字列	-	ホスト IP アドレス
<code>port</code>	整数	8443	HTTPS API ポート

DNS SRV レコード

DNS SRV レコードの形式は `priority/weight/protocol/host:port` です。

DNS レコード

```

;  AAAA AAA AAA AAA
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
alpha.smsg.example.com.
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
bravo.smsg.example.com.
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
charlie.smsg.example.com.

;  AAAA A AAA
smsg-alpha.smsg.example.com.      86400 IN A 10.0.1.2
smsg-bravo.smsg.example.com.      86400 IN A 10.0.2.2
smsg-charlie.smsg.example.com.    86400 IN A 10.0.3.2

```

Configure DNS SRV records with `dns_poll_interval_ms` to 30

Configure DNS SRV records with `forward_max_retries` to `:forward_failed`

Ports

Configure HTTPS ports

Port	Protocol	Priority	Weight
8443	TCP/TLS	10	100
53	UDP/TCP	10	DNS SRV records DNS

Configure iptables

```

# Allow traffic from IP ranges
iptables -A INPUT -p tcp -s 10.0.1.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.2.0/24 --dport 8443 -j ACCEPT
iptables -A INPUT -p tcp -s 10.0.3.0/24 --dport 8443 -j ACCEPT

```

API

API 8443 /api/federation/

URI	Method	Description
/api/federation/identity	GET	Retrieve identity information
/api/federation/health	POST	Health check endpoint
/api/federation/registry	POST	Registry management
/api/federation/forward	POST	Forwarding configuration
/api/federation/delivery_status	POST	Delivery status reporting

JSON X-Federation-Node

1

DNS SRV

DNS SRV

```
# A - Alpha (config/runtime.exs)
config :sms_c,
  smsc_node_name: "alpha-dc01-smsc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smc._tcp.smc.example.com"
```

```
# Node B - Node Bravo (config/runtime.exs)
config :sms_c,
  smsc_node_name: "bravo-dc01-smisc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smisc._tcp.smisc.example.com"
```

```
# Node C - Node Charlie (config/runtime.exs)
config :sms_c,
  smsc_node_name: "charlie-dc01-smisc01"

config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smisc._tcp.smisc.example.com"
```

Node Alpha SRV records for Node Bravo **bravo-dc01-smisc01** Node B
HTTPS

2 Node-Node

DNS SRV

```
# Node (config/runtime.exs)
config :sms_c,
  smsc_node_name: "primary-smisc01"

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

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

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

この構成では、各ノードが IP アドレスを指定して他のノードと接続できるように設定されています。

3. フェデレーション

フェデレーションを有効にするには、以下のように設定します。

```
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smsc._tcp.cluster.internal",
  health_check_interval_ms: 5_000,
  registry_sync_interval_ms: 5_000,
  forward_retry_interval_ms: 1_000,
  http_timeout_ms: 2_000
```

健康チェックの頻度を < 10 秒の RTT に設定し、フェデレーションを有効にします。

監視

フェデレーションのメトリクスは Prometheus に 9568 で公開されています。

例: `sms_c_federation_forward_completed_count` のメトリクスは、以下のように表示されます。

- `dest_controller` - フェデレーションの宛先

監視


```
# 概要
rate(sms_c_federation_forward_completed_count[5m]) * 60

# 詳細
sum by (dest_controller)
(rate(sms_c_federation_forward_completed_count[5m]))
```

監視対象: :forwarded, :forward_queued, :forward_failed (Web UI)

監視対象

監視対象	監視対象	閾値
概要	GET /api/status (Web UI)	> 100 (回)
詳細	GET /api/federation/identity	> 5 (回)
監視対象	Prometheus :forward_failed	
監視対象	監視対象	> 2 (回)

監視対象

監視対象

監視対象 cluster_status

監視対象

- DNS SRV 監視対象
- 監視対象 DNS
- SRV 監視対象
- 監視対象 enabled: false

配置

1. 配置 DNS SRV 记录 `dig SRV _smc._tcp.smc.example.com`
2. 配置 `enabled: true`
3. 配置“DNS 记录”
4. 配置 `static_peers` 列表/端口



验证

配置完成后，使用“验证”命令

配置

- 配置
- 配置 8443 端口
- TLS 配置
- 配置

配置

1. 配置 `curl -k https://<peer-host>:8443/api/federation/identity`
2. 配置 IP 8443 端口
3. 配置 TLS 配置
4. 配置  

配置

配置 `forward_queued` 列表

配置

- 配置
- 配置
- 配置 API `/api/federation/forward` 列表

配置

1. 配置参数——配置参数
2. 配置参数——配置参数“ForwardWorker”
3. 配置参数 curl -k -X POST
 https://<peer>:8443/api/federation/forward -H 'Content-Type: application/json' -d
 '{"source_msisdn":"test","destination_msisdn":"test","message_body":"test","source_smsc":"test","dest_smsc":"test"}'
4. 配置 forward_max_retries ——配置参数 :forward_failed

配置 forward_failed

配置参数 :forward_failed

配置

- 配置参数 forward_max_retries * forward_retry_interval_ms
- 配置参数 50 配置 5 配置参数 4

配置

1. 配置参数
2. 配置参数
3. 配置 forward_max_retries
4. 配置参数 forward_max_retries: 1000 配置 5 配置 83

HSS (Diameter Sh)

← |

OmniMessage Diameter Sh (HSS) —

HSS —

HSS OmniMessage HSS (UDR) HSS (2001) HSS “” (5001)

3GPP TS 29.329	Diameter Sh
3GPP TS 29.328	IMS Sh —
RFC 6733	Diameter

HSS OmniMessage 2/

□□□□□□□□□□

□□□□\n□□□□?

ore OmniCore OmniCall OmniRAN OmniCharge Platform 7A □□□□

□□□□\n□□□□

Diameter Sh\n□□?

UDR □ HSS\n□□□□□□

HSS □□

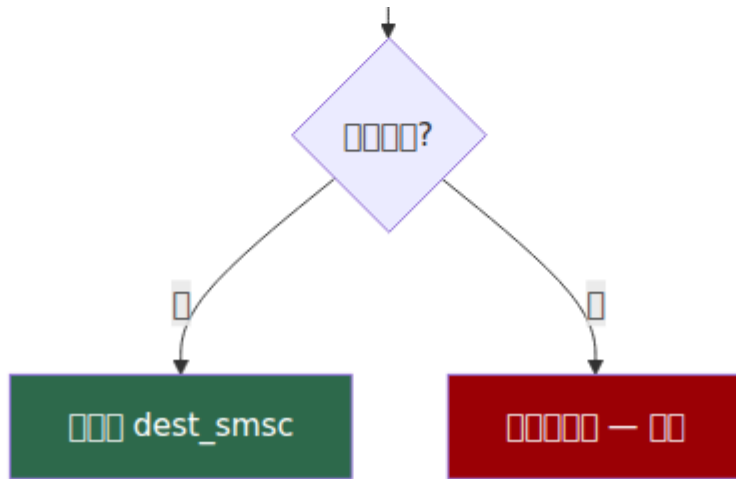
5001 — □□□□

□□ / □□

2001 — □□□□□

□□□□\n□□□□

□□□□□□\n□□□□□□□□



步骤 1 — 检查 MSISDN

检查 MSISDN 是否/是否匹配 HSS 中的 MSISDN 列表

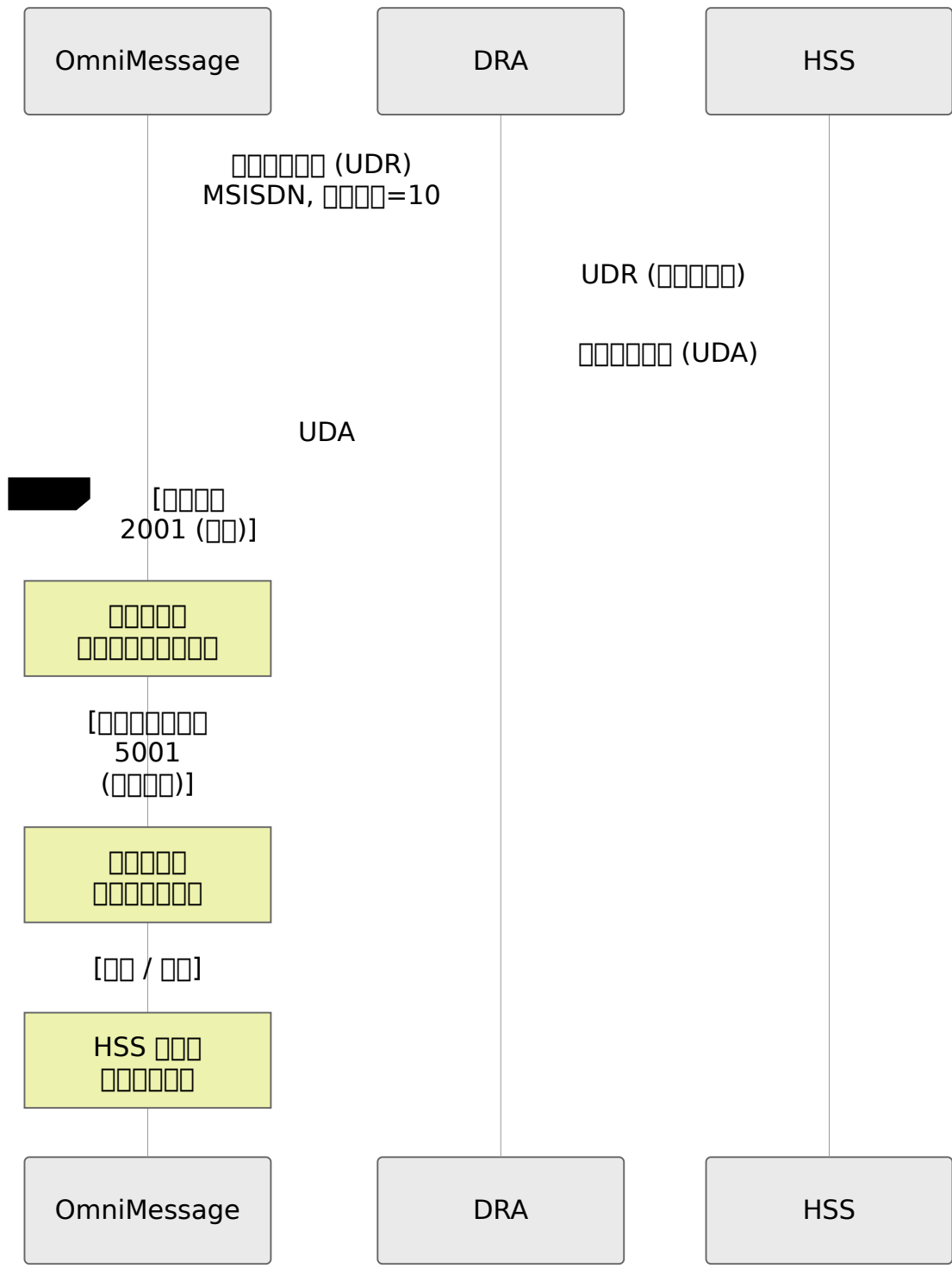
步骤 2 — HSS 检查

通过 Diameter Sh 消息向 HSS 请求 MSISDN 列表

步骤 3 — 返回

HSS 返回 Diameter 消息，包含 MSISDN 列表

Diameter Sh 000



UDR 表

AVP	値	説明
Session-Id	0000	00000000
Auth-Session-State	1 (NO_STATE_MAINTAINED)	0000000000
Data-Reference	10	00000000
Origin-Host	000000 + 00	OmniMessage 0 Diameter 00
Origin-Realm	000000	OmniMessage 0 Diameter 00
Destination-Realm	000000	HSS 00 (00 DRA 00)
User-Identity	TBCD 000 MSISDN	0000000000
Server-Name	"OmniMessage"	00000000
Vendor-Specific-Application-Id	Auth: 16777217, Vendor: 10415	00 3GPP TS 29.329 0 Sh 000000

UDA 表

値	説明	値	OmniMessage 説明
2001	0000	00 — 000000	00000000000000
5001	00000000	0000 — 00 HSS 0	00000000
00	00000000	0000	00000000000000

□□

□□ **HSS** □□

HSS □□□ `config/runtime.exs` □□□□□□□□□□

1. `:sms_c` □□ `diameter_enabled` □□ — □□□□□□□□□□□□ HSS □□
2. `:diameter_ex` □□ — □□ Diameter □□□□□□□□□□□□□□□□

□□□□

```
# □□□□□□□□ HSS □□
config :sms_c,
  diameter_enabled: true

# Diameter □□□□
config :diameter_ex,
  diameter: %{
    service_name: :omnimessage,
    listen_ip: "0.0.0.0",
    listen_port: 3868,
    decode_format: :map,
    host: "smsc01",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    auth_application_ids: [],
    acct_application_ids: [],
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{
            vendor_id: 10415,
            auth_application_id: 16_777_217,
            acct_application_id: nil
          }
        ]
      }
    ],
    peers: [
      %{
        port: 3868,
        host: "dra01.epc.mnc005.mcc547.3gppnetwork.org",
```

```

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

```

OmniMessage []

属性	型	デフォルト値	説明
diameter_enabled	Boolean	false	HSS と接続するかどうかを指定する。true の場合は HSS と接続し、false の場合は接続しない。
mock_sh	Boolean	false	HSS と接続する際に Diameter のメッセージをモックするかどうかを指定する。
mock_sh_on_net_numbers	Array	[]	mock_sh が true の場合に、MSISDN のリストを指定する。

Diameter □□□□

□□	□ □	□ □	□□	
<code>service_name</code>	□ □	□	-	Diam
<code>listen_ip</code>	□ □ □	□	<code>"0.0.0.0"</code>	□□□□□
<code>listen_port</code>	□ □	□	3868	□□□□ 6733
<code>decode_format</code>	□ □	□	-	Diam
<code>host</code>	□ □ □	□	-	Omni □□□□□
<code>realm</code>	□ □ □	□	-	Omni 3GPP <code>epc.n</code>
<code>product_name</code>	□ □ □	□	-	□ Dial
<code>request_timeout</code>	□ □	□	5000	□□ HS
<code>peer_selection_algorithm</code>	□ □	□	<code>:random</code>	□□□□□ :rour

配置项	数据类型	默认值	取值范围	说明
<code>allow_undefined_peers_to_connect</code>	boolean	true	true	是否允许未定义的对等方连接
<code>log_unauthorized_peer_connection_attempts</code>	boolean	true	true	是否记录未经授权的对等方连接尝试
<code>control_module</code>	string	-	-	Diameter SmsControl
<code>processor_module</code>	string	-	-	Diameter Diameter
<code>auth_application_ids</code>	list	[]	[]	认证应用ID列表
<code>acct_application_ids</code>	list	[]	[]	计费应用ID列表
<code>vendor_id</code>	string	-	-	3GPP
<code>supported_vendor_ids</code>	list	-	-	支持的对等方ID列表

配置项

`applications` 配置Diameter应用，HSS应用Sh

項目	型	長	値	説明
application_name	文字列	100	-	HSS アプリケーション名 :sh
application_dictionary	文字列	100	-	Erlang Diameter アプリケーション辞書 :diameter_gen_3gpp_sh
vendor_specific_application_ids	文字列	100	-	ベンダー固有のアプリケーション ID

3GPP ID (Sh)

項目	型	値	説明
vendor_id	整数	10415	3GPP ID
auth_application_id	整数	16_777_217	3GPP TS 29.329 Sh ID
acct_application_id	-	nil	Sh

peers

peers: OmniMessage Diameter DRA HSS

項目	型	必須	デフォルト	説明
host	文字列	必須	-	Diameter サーバの FQDN (FQDN) を指定します。
ip	文字列	必須	-	TCP/IP アドレスを指定します。
port	整数	必須	3868	Diameter サーバのポート番号を指定します。
realm	文字列	必須	-	Diameter サーバのドメインを指定します。
tls	ブール値	必須	false	TLS を有効にするかどうかを指定します。
transport	文字列	必須	:diameter_tcp	使用するトランスポートプロトコルを指定します。:diameter_tcp または :diameter_sctp を指定できます。
initiate_connection	ブール値	必須	true	接続を初期化するかどうかを指定します。true を指定すると、OmniMessage を送信して TCP 接続を確立し、false を指定すると、OmniMessage を送信せずに接続を試みます。

注意事項

OmniMessage の peer_selection_algorithm を指定します。

```
peers: [
  %{
    port: 3868,
    host: "dra01.epc.mnc005.mcc547.3gppnetwork.org",
    ip: "10.0.0.1",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  },
  %{
    port: 3868,
    host: "dra02.epc.mnc005.mcc547.3gppnetwork.org",
    ip: "10.0.0.2",
    realm: "epc.mnc005.mcc547.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  }
]
```

□□□□

□□□□□□ HSS □□□□□□□□□□□□□□

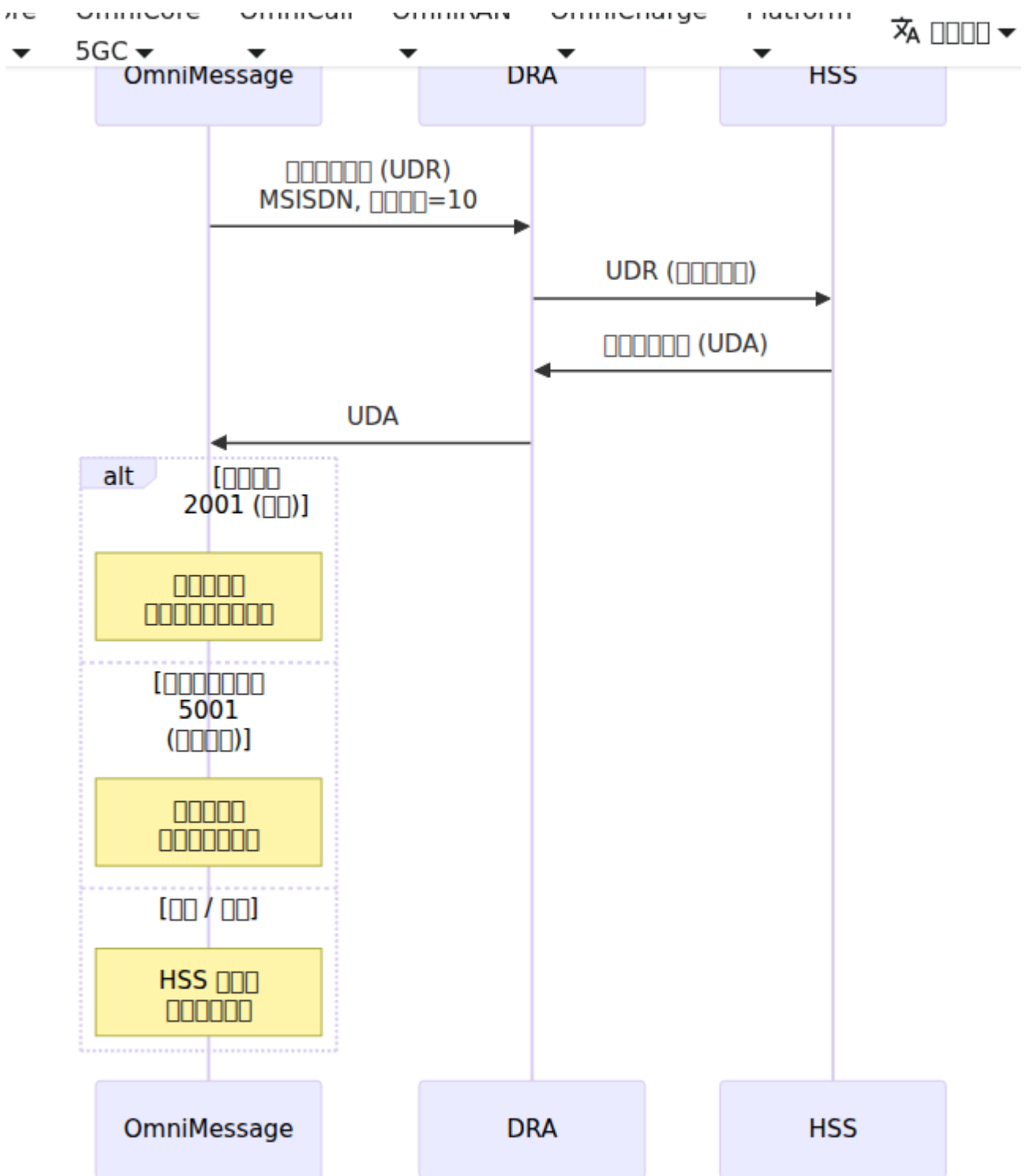
```
config :sms_c,
  diameter_enabled: true,
  mock_sh: true,
  mock_sh_on_net_numbers: [
    "68987654321",
    "68912345678"
  ]
```

□□□□□□□ mock_sh_on_net_numbers □□□□□□ MSISDN □□□□□□◆◆□□□□□□ HSS □□□□
2001□□□□□□ MSISDN □□□□□□□□□□□□□□ 5001□□□□□□□ Diameter □□□□□□□□□□□□□□

Sequence Diagram

Scenario: Mnesia dest_smsc is nil
 MSISDN is provided

dead_letter_time_minutes is set



- 检查 Diameter 配置是否正确
- HSS 返回 "HSS dip failed" 错误
- 检查 HSS 配置

检查:

1. 检查 `runtime.exe` 中 `diameter_enabled: true`
2. 检查 Diameter 配置
3. 检查 "HSS dip failed" 错误
4. 检查 HSS 配置 E.164 号码 +

检查配置

检查: 检查配置

检查:

- 检查 OmniMessage
- 检查 MSISDN 配置 `destination_msisdn`
- 检查配置

检查:

1. 检查/配置
2. 检查 MSISDN 配置
3. 检查 `dead_letter_time_minutes` — 检查配置

Diameter 配置

检查: 检查 Diameter 配置

检查:

- 检查 IP 配置
- 检查 3868
- 检查 `host` 配置 Diameter 配置
- OmniMessage 配置

□□□□:

1. □□□□□□ IP □□□□□□□□□□
2. □□□□□□□□□□□□□□□□ TCP □□
3. □□□□□□ `host` □□□□□□□□□□□□□□□□□□□□□□□□□□
4. □□ OmniMessage □□□□□□□□□□ `realm` □□

SMS-C Prometheus



[←](#) [README](#)



SMS-C Prometheus



Prometheus

`http://localhost:9568/metrics`

Prometheus



`sms_c.<category>.<metric_name>.<type>`

- `license` -
- `message` -
- `routing` -
- `enum` - ENUM/NAPTR
- `delivery` -
- `queue` -
- `charging` - /
- `mnesia` -

- `frontend` - 管理画面
- `location` - 位置情報
- `phoenix.endpoint` - HTTP API 接続先
- `vm` - Erlang VM 接続先

管理画面

sms_c_license_status

管理画面 Gauge

管理画面 OmniMessage SMS-C 接続先

管理画面

- `1` - 接続成功
- `0` - 接続失敗

管理画面

管理画面 `omnimessage`

管理画面 `omnimessage` の値が `"NOLICENCE"` の場合

管理画面

- 接続先が `dest_smsc` の場合
- 接続先が `dest_smsc` の場合 **"NOLICENCE"** の場合
- 接続先が `dest_smsc` の場合
- UI 上に表示される
- 接続先が `dest_smsc` の場合

管理画面

```

- alert: SMS_C_License_Invalid
  expr: sms_c_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "SMS-C 异常"
    description: "异常 - 无许可证"

```

Prometheus

```

# 异常
sms_c_license_status == 1

# 正常
sms_c_license_status == 0

# 无许可证 异常
sms_c_routing_route_matched_count{dest_smsc="NOLICENCE"}

```

异常

sms_c_message_received_count

Counter

SMS-C 异常

异常

- source_smsc 异常 SMSC
- source_type 异常ims 异常circuit_switched 异常smpp
- message_type 异常sms 异常mms

异常

Counter/Counter

sms_c_message_validated_count

Counter

Counter

Counter

- `valid` Counter `true` | `false`

Counter/Counter

Counter > 5% Counter

sms_c_message_processing_stop_duration

Histogram

Counter

Counter

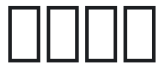
10, 50, 100, 250, 500, 1000, 2500, 5000 ms

Counter

- `success` Counter `true` | `false`

Counter

Counter p95 | Counter p99 Counter SLA Counter



sms_c_routing_route_matched_count

Counter

Counter

Counter

- route_id Counter
- dest_smsc Counter SMSC
- priority Counter

Counter

Counter

sms_c_routing_failed_count

Counter

Counter

Counter

- reason Counter no_route_found validation_failed

Counter

Counter

sms_c_routing_action_count

Counter

Counter

□□□

- `action` □□□□□□ `drop` □ `auto_reply` □ `forward` □
- `route_id` □□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
◆◆

sms_c_routing_stop_duration

□□□ Histogram

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

□□ 1, 5, 10, 25, 50, 100, 250, 500 ms

□□□

- `dest_smsc` □□□□□□□□ SMSC

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
p95 > 50ms □□

ENUM/NAPTR □□□□

sms_c_enum_cache_hit_count

□□□ Counter

□□□□□□□□□□ ENUM □□□□□□□□□□ DNS □□□□

□□□

- `domain` ENUM

DNS

`sms_c_enum_cache_miss_count`

Counter

DNS ENUM

- `domain` ENUM

`cache_hit_rate = hits / (hits + misses)`

`sms_c_enum_cache_size_size`

Gauge

ENUM

TTL

`sms_c_enum_lookup_stop_duration`

Histogram

ENUM DNS

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

Options

- `domain` ENUM
- `success` true | false
- `cache_hit` true | false

ENUM DNS

p95

sms_c_enum_naptr_records_record_count

Histogram

ENUM NAPTR

0, 1, 2, 3, 5, 10

Options

- `domain` ENUM

ENUM 1-3

0 DNS

Options

sms_c_delivery_queued_count

Counter

SMSC

Options

- `dest_smsc` SMSC

sms_c_delivery_attempted_count

Counter

- `dest_smsc` SMSC

sms_c_delivery_succeeded_count

Counter

SMSC

- `dest_smsc` SMSC

SLA

`success_rate = succeeded / queued`

sms_c_delivery_failed_count

Counter

□□□□□□□□□□□□□□□□□□□□

□□□

- `dest_smsc` □□□ SMSC □□
- `reason` □□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

sms_c_delivery_dead_letter_count

□□□ Counter

□□□□□□□□□□□□□□□□□□□□

□□

- `reason` □□□□□□□□□□ `max_retries_exceeded` □ `expired` □

□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

sms_c_delivery_succeeded_attempt_count

□□□ Histogram

□□□□□□□□□□□□□□□□□□□□

□□ 1, 2, 3, 5, 10

□□□

- `dest_smsc` □□□ SMSC □□

□□□□□□□□□□□□□□□□□□□□

- p95 latency/throughput SLA requirements
- Performance optimization techniques

Prometheus Metrics


```

# P95 SMSC P95
histogram_quantile(0.95,
  sum by (source_smsc, dest_smsc, le) (
    rate(sms_c_delivery_time_delta_delta_ms_bucket[5m])
  )
)

# P99
histogram_quantile(0.99,
  sum by (le) (
    rate(sms_c_delivery_time_delta_delta_ms_bucket[5m])
  )
)

# SMSC
sum by (source_smsc) (rate(sms_c_delivery_time_delta_delta_ms_sum[5m]
/
sum by (source_smsc)
(rate(sms_c_delivery_time_delta_delta_ms_count[5m])))

#
histogram_quantile(0.95,
  sum by (le)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{delivery_attempts="0"
[5m]))
)
histogram_quantile(0.95,
  sum by (le)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{delivery_attempts!="0"
[5m]))
)

# - p95
topk(10,
  histogram_quantile(0.95,
    sum by (source_smsc, dest_smsc, le) (
      rate(sms_c_delivery_time_delta_delta_ms_bucket[1h])
    )
  )
)

# 2
sum(rate(sms_c_delivery_time_delta_delta_ms_bucket{le="2000"}[5m])) /

```

```
sum(rate(sms_c_delivery_time_delta_delta_ms_count[5m])) * 100

# 000000 2 0000
sum by (dest_smsc)
(rate(sms_c_delivery_time_delta_delta_ms_bucket{le="2000"}[5m])) /
sum by (dest_smsc) (rate(sms_c_delivery_time_delta_delta_ms_count[5m]
* 100
```

0000

sms_c_queue_size_size

000Gauge

00000000000000000000000000000000

000

- queue_type 000000 message_queue 0 dead_letter 0

00000000000000000000000000000000

00000000000000000000000000000000

sms_c_queue_size_pending

000Gauge

00000000000000000000000000000000

000

- queue_type 000000

00000000000000000000000000000000

00000000000000000000000000000000

sms_c_queue_size_failed

Gauge

Queue size of failed messages

Tags

- queue_type

Queue size of failed messages

Queue size of failed messages

sms_c_queue_size_delivered

Gauge

Queue size of delivered messages

Tags

- queue_type

Queue size of delivered messages

Queue size of delivered messages

sms_c_queue_oldest_message_age_seconds

Gauge

Age of the oldest message in the queue

Tags

- queue_type

SLA

SLA > 300

sms_c_charging_requested_count

Counter

OCS

- account

sms_c_charging_succeeded_count

Counter

- account

$success_rate = \frac{succeeded}{requested}$

sms_c_charging_failed_count

Counter

□□□□□□□□□□□□

□□□

- account □□□□□□
- reason □□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□

sms_c_charging_succeeded_duration

□□□ Histogram

□□□□□□□□□□□□□□□□

□□□□□

□□ 10, 50, 100, 250, 500, 1000, 2500, 5000 ms

□□□

- account □□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ p95 □□□□□□□□□□□□□□□□

□□□□□□□□

sms_c_mnesia_table_size_record_count

□□□ Gauge

□□□□□ Mnesia □□□□□□□□□□□□□□ 10 □□□□□□

表名

- table 表名

表名

- sms_route - SMS 表名
- message_store - 消息存储表名
- location_store - 位置存储表名
- frontend_store - 前端/SMSC 表名
- translation_rule - 翻译规则表名
- cell_tower_store - 基站存储表名
- message_events - 消息事件表名

表名

表名

表名 **Prometheus** 表名

```
# 表名  
sms_c_mnesia_table_size_record_count  
  
# 表名  
sms_c_mnesia_table_size_record_count{table="message_store"}  
  
# 表名  
sms_c_mnesia_table_size_record_count{table="location_store"}  
  
# 表名  
sms_c_mnesia_table_size_record_count{table="sms_route"}
```

sms_c_frontend_status_count

表名 Gauge

表名

概要

- `frontend_name` 名前
- `status` `connected` / `disconnected`

概要

概要

sms_c_location_registered_count

概要 Counter

概要

概要

- `location` 名前 / SMSC 名前
- `ims_capable` IMS 名前 `true` / `false`

概要 IMS 名前 IMS 名前

概要

- 概要
- 概要
- IMS 概要

概要

```
# 概要
rate(sms_c_location_registered_count[1m])

# IMS 名前 IMS 名前
sum(rate(sms_c_location_registered_count{ims_capable="true"}[5m]))
/
sum(rate(sms_c_location_registered_count[5m]))
```

sms_c_location_active_registrations_count

📊 Gauge

📍📞📡📶📱📧/SMSC📡 IMS 📡📶📱📧📞 10 📍📞📡📶📱📧📞

📍📞

- `location` 📍📞📡📶📱📧/SMSC 📡
- `ims_capable` 📍📞📡📶📱📧 IMS 📡📶📱📧📞 true/false📡

📍📞📡📶📱📧📞📡📶📱📧📞 IMS 📡📶📱📧📞📡 SMSC 📍📞📡📶📱📧📞

📍📞📡📶📱📧📞

- 📍📞📡📶📱📧📞
- 📍📞📡📶📱📧📞
- 📍📞📡📶📱📧📞

📍📞 **Prometheus** 📍📞


```

# 总数
sum(sms_c_location_active_registrations_count)

# 按位置分组
sum by (location) (sms_c_location_active_registrations_count)

# 按位置分组 IMS 支持
sum by (location)
(sms_c_location_active_registrations_count{ims_capable="true"})

# 按位置分组 IMS 占比
sum by (location)
(sms_c_location_active_registrations_count{ims_capable="true"}) /
sum by (location) (sms_c_location_active_registrations_count) *
100

# 按 IMS 支持分组
sum by (ims_capable) (sms_c_location_active_registrations_count)

# 按位置分组 TopK
topk(10, sum by (location)
(sms_c_location_active_registrations_count))

# 按位置分组占比
sum by (location) (sms_c_location_active_registrations_count) /
sum(sms_c_location_active_registrations_count) * 100

```

HTTP API 接口

phoenix_endpoint_stop_duration

分布 (Histogram)

HTTP 接口

接口

- route API 接口 `/api/messages` `/api/frontends`

10ms 50ms 100ms 250ms 500ms 1s 2.5s 5s

API SLA

- P95 > 500ms
- P99 > 1s
-

```
# P95
histogram_quantile(0.95,
  rate(phoenix_endpoint_stop_duration_bucket[5m]))

# 1
sum(rate(phoenix_endpoint_stop_duration_bucket{le="1000"}[5m]))
```

phoenix_endpoint_stop_count

Counter

HTTP HTTP

- route API
- status HTTP 200 201 400 404 500

API SLA

- > 5%
- 5xx
-

□□□□□

```
# □□□□□□□□  
sum by (route) (rate(phoenix_endpoint_stop_count[5m]))  
  
# □□□□□□□□  
sum by (route) (rate(phoenix_endpoint_stop_count{status=~"5.."}  
[5m])) /  
sum by (route) (rate(phoenix_endpoint_stop_count[5m]))  
  
# □□□  
sum(rate(phoenix_endpoint_stop_count{status=~"2.."}[5m])) /  
sum(rate(phoenix_endpoint_stop_count[5m]))
```

phoenix_router_dispatch_exception_count

□□□ Counter

□□□□ HTTP □□□□□□□□□□□□/□□□□□

□□□

- route □□□□□□ API □□□□
- kind □□□□□□ error □ exit □ throw □

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□

□□□□□

```
# □□□□□□□□  
rate(phoenix_router_dispatch_exception_count[5m])  
  
# □□□□□□□□□□  
increase(phoenix_router_dispatch_exception_count[1h])
```

Erlang VM

vm_memory_total

Gauge

Erlang VM

> 80%

vm_memory_processes

Gauge

Erlang

vm_total_run_queue_lengths_total

Gauge

CPU

CPU

10 * CPU

vm_system_counts_process_count

Gauge

VM 262,144

262,144

10

-
- Mnesia
- ENUM

SMSC

$(\text{sms_c_delivery_succeeded_count}) / (\text{sms_c_delivery_queued_count})$

> 95%

- sms_c_message_processing_stop_duration
- sms_c_delivery_time_delta_delta_ms

ENUM 命中率

ENUM 命中率

ENUM 命中率

$(\text{sms_c_enum_cache_hit_count}) / (\text{sms_c_enum_cache_hit_count} + \text{sms_c_enum_cache_miss_count})$

命中率 > 80% 则设置 TTL 为 1000000

路由匹配

路由匹配

$\text{sms_c_routing_route_matched_count}$ 按 route_id 分组

路由匹配

队列大小

队列大小

队列

- $\text{sms_c_queue_size_pending}$ 队列大小
- $\text{sms_c_queue_oldest_message_age_seconds}$ 队列最老消息的年龄

队列大小 + 队列最老消息的年龄 = 队列大小

交付

交付

$\text{sms_c_delivery_succeeded_attempt_count}$ histogram percentiles

□□□□ p95 > 1□□□□□□□□□□□□□□□□□□□□

报警规则

报警名称	报警规则	报警级别	报警模板
消息路由失败	<code>routing_failed_count > 10</code>	Critical	消息路由失败: {value}
消息队列积压	<code>queue_size_pending > 100</code>	Warning	消息队列积压: {value}
消息队列最老消息过期	<code>queue_oldest_message_age_seconds > 300</code>	Critical	消息队列最老消息过期: {value} SLA
消息投递失败	<code>delivery_failed_count > 10</code>	High	消息投递失败: {value}
消息投递死信	<code>delivery_dead_letter_count > 0</code>	High	消息投递死信: {value}
ENUM 查找耗时	<code>enum_lookup_stop_duration p95 > 5000ms</code>	Warning	ENUM 查找耗时: {value} DNS
ENUM 命中率	<code>ENUM hit_ratio < 0.7</code>	Warning	ENUM 命中率: {value}
前端断开连接	<code>frontend_status_count{status="disconnected"} > 0</code>	High	前端断开连接: {value}
消息充电失败	<code>charging_failed_count > 10</code>	High	消息充电失败: {value}
消息处理耗时	<code>message_processing_stop_duration p95 > 1000ms</code>	Warning	消息处理耗时: {value}

□□□□□

□□□□□

□□□□□□□□□□

□□□

1. □□□□□□□□□□/□□/□□□
 2. □□□□□□□□□□□□□□□□
 3. □□□□□□□□□□
 4. p95 □□□□□□□□
 5. □□□□□□□
 6. □□□□□
-

□□□□□

□□□□□□□□□□

□□□

1. □□□□□□□□□□□□
 2. □□□□□□□□□□
 3. ENUM □□□□□□□□□□
 4. □□□□□□□□□□
 5. □□□□□□□
 6. □□□□□
-

□□□□□

□□□□□□□□□□

□□□

1. SMSC
 2. SMSC
 3. ? ? ?
 - 4.
 5. ENUM
 - 6.
-

Prometheus

- 15
- **5** 90
- **1** 2

1. sms_c_message_received_count -
 2. sms_c_routing_failed_count -
 3. sms_c_delivery_queued_count -
 4. sms_c_delivery_failed_count -
 5. dest_smsc
-

短信处理时长

指标

1. `sms_c_message_processing_stop_duration` 秒 - 短信处理时长
 2. `sms_c_routing_stop_duration` - 路由时长
 3. `sms_c_enum_lookup_stop_duration` - ENUM 查找时长
 4. `sms_c_charging_succeeded_duration` - 计费成功时长
 5. 短信接收时长
-

短信队列大小

指标

1. `sms_c_queue_size_pending` 个 - 待处理短信数量
 2. `sms_c_delivery_attempted_count` - 尝试投递次数
 3. `sms_c_delivery_failed_count` - 投递失败次数
 4. `sms_c_delivery_time_delta_delta_ms` - 投递时间差
 5. `dest_smsc` 短信中心号码
-

Prometheus 配置

配置

短信接收速率5分钟

```
rate(sms_c_message_received_count[5m])
```

短信接收速率1小时

```
rate(sms_c_message_received_count[1h]) * 60
```

增加计数

```
increase(sms_c_message_received_count[24h])
```

按源类型求和

```
sum by (source_type) (rate(sms_c_message_received_count[5m]))
```

按 SMSC 求和

```
sum by (source_smsc) (rate(sms_c_message_received_count[5m]))
```

成功率

成功消息数 / 队列消息数

```
(rate(sms_c_delivery_succeeded_count[5m]) /  
rate(sms_c_delivery_queued_count[5m])) * 100
```

失败率

```
(rate(sms_c_delivery_failed_count[5m]) /  
rate(sms_c_delivery_queued_count[5m])) * 100
```

成功消息的 p95 分位数

```
histogram_quantile(0.95,  
sms_c_delivery_succeeded_attempt_count_bucket)
```

按目标 SMSC 求和

```
sum by (dest_smsc) (rate(sms_c_delivery_succeeded_count[5m]))
```

□□□□□□

```
sum by (reason) (rate(sms_c_delivery_failed_count[5m]))
```

□□□□□**p95**□□

```
histogram_quantile(0.95,  
sms_c_delivery_time_delta_delta_ms_bucket)
```

□□□□□**p99**□□

```
histogram_quantile(0.99,  
sms_c_delivery_time_delta_delta_ms_bucket)
```

□□□□

□□□□□□□□

```
sms_c_queue_size_pending
```

□□□□□□□□□□

```
sms_c_queue_size_failed
```

□□□□□□□□□□□□

```
sms_c_queue_oldest_message_age_seconds / 60
```

□□□□□□□□□□□□

```
rate(sms_c_queue_size_size[1h]) * 3600
```

□□□□□□□□

```
rate(sms_c_delivery_queued_count[5m])
```

队列速率

```
rate(sms_c_delivery_succeeded_count[5m]) +  
rate(sms_c_delivery_failed_count[5m])
```

队列速率 - 成功速率

```
rate(sms_c_delivery_queued_count[5m]) -  
(rate(sms_c_delivery_succeeded_count[5m]) +  
rate(sms_c_delivery_failed_count[5m]))
```

成功率

成功率公式

```
(1 - (rate(sms_c_routing_failed_count[5m]) /  
(rate(sms_c_routing_route_matched_count[5m]) +  
rate(sms_c_routing_failed_count[5m]))) * 100
```

路由成功率

```
topk(10, sum by (route_id, dest_smsc)  
(rate(sms_c_routing_route_matched_count[1h])))
```

路由停止时长分布

```
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)
```

路由停止时长分布

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

ENUM 0000000000

```
increase(sms_c_routing_action_count{action="drop"}[1h])
```

ENUM 000000000000

```
increase(sms_c_routing_action_count{action="auto_reply"}[1h])
```

ENUM 00

ENUM 00000000

```
rate(sms_c_enum_cache_hit_count[5m]) /  
(rate(sms_c_enum_cache_hit_count[5m]) +  
rate(sms_c_enum_cache_miss_count[5m]))
```

ENUM 0000000000

```
(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
```

ENUM 000000p9500

```
histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)
```

00 ENUM 00000000000000

```
# 命中率
rate(sms_c_enum_cache_hit_count[5m])

# 命中率 DNS 失败
rate(sms_c_enum_cache_miss_count[5m])
```

命中率 **NAPTR** 命中率

```
rate(sms_c_enum_naptr_records_record_count_sum[5m]) /
rate(sms_c_enum_naptr_records_record_count_count[5m])
```

ENUM 命中率

```
sms_c_enum_cache_size_size
```

命中率

命中率 **p95** 命中率

```
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)
```

命中率 **p99** 命中率

```
histogram_quantile(0.99,
sms_c_message_processing_stop_duration_bucket)
```

命中率

```
rate(sms_c_message_processing_stop_duration_count{success="false"}
[5m])
```

命中率


```
rate(sms_c_message_validated_count{valid="false"}[5m]) /  
rate(sms_c_message_validated_count[5m])
```

□□□□

□□□□□□

```
rate(sms_c_charging_succeeded_count[5m]) /  
rate(sms_c_charging_requested_count[5m])
```

□□□□□□□□

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

□□□□□**p95**□□

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

□□□□□□□□

```
sum by (account) (rate(sms_c_charging_requested_count[1h]))
```

□□□□

□□□□□

```
sum(sms_c_frontend_status_count{status="connected"})
```

□□□□□□□□

```
sum(sms_c_frontend_status_count{status="disconnected"})
```

□□□□□□

```
sum by (frontend_name)
(sms_c_frontend_status_count{status="connected"})
```

□□□□

Mnesia □□□□

```
sms_c_mnesia_table_size_record_count
```

□□□□

```
sms_c_mnesia_table_size_record_count{table="sms_route"}
```

□□□□□□

```
sms_c_mnesia_table_size_record_count{table="translation_rule"}
```

Grafana □□□□□

□□□ **1**□□□□□

□□□□□□□□□□□□□□

□□□

1. □□□□□□□□□□

- □□□ `rate(sms_c_message_received_count[5m])`
- □□□ `rate(sms_c_delivery_succeeded_count[5m])`
- □□□□□/□
- □□□{{source_type}}

2. 消息成功率

- $\text{rate}(\text{sms_c_delivery_succeeded_count}[5\text{m}]) / \text{rate}(\text{sms_c_delivery_queued_count}[5\text{m}]) * 100$
- 范围0-100%
- 阈值
 - < 90
 - $90-95$
 - > 95

3. 队列大小

- `sms_c_queue_size_pending`
- `sms_c_queue_size_failed`
- 范围
- `{{queue_type}}`

4. 队列最老消息年龄

- $\text{sms_c_queue_oldest_message_age_seconds} / 60$
- 范围
- 阈值
 - < 5
 - $5-10$
 - > 10

5. 连接数

- `sum(sms_c_frontend_status_count{status="connected"})`
- 范围
- 阈值

6. 消息路由失败率

- $\text{rate}(\text{sms_c_routing_failed_count}[5\text{m}]) * 60$
- 范围/阈值
- > 0

0000 20000000

00000000000000000000

0000

1. 0000000000

- 0000 histogram_quantile(0.50, sms_c_message_processing_stop_duration_bucket) p50
- 0000 histogram_quantile(0.95, sms_c_message_processing_stop_duration_bucket) p95
- 0000 histogram_quantile(0.99, sms_c_message_processing_stop_duration_bucket) p99
- 000000
- 000000

2. 0000000000

- 0000 histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
- ENUM 0000 histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)
- 0000 histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)
- 0000 histogram_quantile(0.95, sms_c_delivery_time_delta_delta_ms_bucket)
- 000000
- 0000

3. 0000000000

- 0000 sms_c_delivery_succeeded_attempt_count_bucket
- 000000000000
- 00000000001 00000000000000

4. ENUM 00000000

- `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`
- `sms_c_enum_cache_size_size`
- Y `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`

5. `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`

- `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`
- `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`
- `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`
 - `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100 < 95`
 - `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100 95-99`
 - `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100 > 99`

3. `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`

`rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`

`rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`

1. `rate(sms_c_message_processing_stop_duration_count{success="true"}[5m]) / rate(sms_c_message_processing_stop_duration_count[5m]) * 100`

- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`

2. `sum by (source_type) (increase(sms_c_message_received_count[1h]))`

- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
- `sum by (source_type) (increase(sms_c_message_received_count[1h]))`

3. `sum by (source_type) (increase(sms_c_message_received_count[1h]))`

- 统计
 - 统计 ID
 - 统计 SMSC
 - 统计 1h 统计 `sum by (route_id, dest_smsc) (increase(sms_c_routing_route_matched_count[1h]))`
 - 统计
 - 统计
- 统计

4. 统计

- 统计 `sum by (dest_smsc) (rate(sms_c_delivery_succeeded_count[5m]))`
- 统计
- 统计
- 统计 `{{dest_smsc}}`

5. 统计

- 统计 `increase(sms_c_routing_action_count{action="drop"}[1h])`
- 统计 `increase(sms_c_routing_action_count{action="auto_reply"}[1h])`
- 统计

6. 统计

- 统计 `rate(sms_c_message_received_count[1h]) * 3600`
- 统计 7
- 统计

统计 4

统计

统计

1. 统计

- `rate sms_c_queue_size_size`
- `rate sms_c_queue_size_size`
- `rate sms_c_queue_size_size`

2. `rate sms_c_mnesia_table_size_record_count`

- `rate sms_c_mnesia_table_size_record_count{table="sms_route"}`
- `rate sms_c_mnesia_table_size_record_count{table="translation_rule"}`
- `rate sms_c_mnesia_table_size_record_count{table="translation_rule"}`
- `rate sms_c_mnesia_table_size_record_count{table="translation_rule"}`

3. `rate sms_c_delivery_queued_count`

- `rate sms_c_delivery_queued_count[5m] - (rate sms_c_delivery_succeeded_count[5m] + rate sms_c_delivery_failed_count[5m])`
- `rate sms_c_delivery_queued_count[5m]`
- `rate sms_c_delivery_succeeded_count[5m]`
- `rate sms_c_delivery_failed_count[5m]`

4. `max_over_time rate sms_c_message_received_count`

- `max_over_time rate sms_c_message_received_count[5m][24h:]`
- `max_over_time rate sms_c_message_received_count[5m][24h:]`
- `max_over_time rate sms_c_message_received_count[5m][24h:]`

5. `rate sms_c_message_received_count`

- `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100`
- `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100`
- `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100`
- `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100`
 - `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100 < 70`
 - `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100 70-85`
 - `rate sms_c_message_received_count[5m] / MAX_CAPACITY * 100 > 85`

5 SLA

SLA

1. SLA

- $(\text{rate}(\text{sms_c_delivery_succeeded_count}[1h]) / \text{rate}(\text{sms_c_delivery_queued_count}[1h])) * 100$
- 99%
- - < 95
 - 95-99
 - ≥ 99

2. SLA

- $\text{count}(\text{sms_c_delivery_time_delta_delta_ms_bucket}\{\text{le}="5000"\}) / \text{count}(\text{sms_c_delivery_time_delta_delta_ms_bucket})$
- 5
-

3. SLA

- 5 $\text{increase}(\text{sms_c_queue_oldest_message_age_seconds}\{\} > 300)[24h:]$
- 0

4.

- $\text{up}\{\text{job}="sms-c"\}$
- $1 = 0 =$
-

5.

- $\text{avg_over_time}((\text{rate}(\text{sms_c_delivery_succeeded_count}[1h]) / \text{rate}(\text{sms_c_delivery_queued_count}[1h]))[24h:1h])$

- 平均遅延 30 秒
- SLA 達成 99%

監視項目

監視

監視 

```
alert: RoutingFailuresDetected
expr: increase(sms_c_routing_failed_count[5m]) > 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "{ $value } 平均遅延 5 秒"
  description: "監視項目"
```

監視

```
alert: MessageQueueBacklog
expr: sms_c_queue_size_pending > 10000
for: 5m
labels:
  severity: critical
annotations:
  summary: "バックログ { $value } あり"
  description: "監視項目"
```

監視項目

```
alert: OldMessagesInQueue
expr: sms_c_queue_oldest_message_age_seconds > 300
for: 2m
labels:
  severity: critical
annotations:
  summary: "队列消息最老年龄 > 300s"
  description: "队列消息最老年龄 > 300s"
```

队列消息最老年龄 > 300s

```
alert: NoActiveFrontends
expr: sum(sms_c_frontend_status_count{status="connected"}) == 0
for: 1m
labels:
  severity: critical
annotations:
  summary: "没有活跃的客户端"
  description: "没有活跃的客户端"
```

没有活跃的客户端

```
alert: DeadLetterMessagesIncreasing
expr: rate(sms_c_delivery_dead_letter_count[10m]) > 0
for: 5m
labels:
  severity: critical
annotations:
  summary: "死信消息数量增加"
  description: "死信消息数量增加"
```

死信消息数量增加

死信消息数量增加

```
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: "95% 交付成功率が低下しています"
  description: "95%のメッセージが正常に配信されています"
```

95%

```
alert: HighDeliveryRetryRate
expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count_bucket) > 2
for: 15m
labels:
  severity: warning
annotations:
  summary: "95% 再試行率が上昇しています"
  description: "95%のメッセージが再試行されています"
```

95%

```
alert: SlowMessageProcessing
expr: histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket) > 1000
for: 10m
labels:
  severity: warning
annotations:
  summary: "95% のメッセージ処理が遅延しています"
  description: "95%のメッセージが1000ms以上遅延しています"
```

ENUM 95%

- Prometheus `rate()` `increase()`
- Gauge
- Histogram p50 p95 p99
- Prometheus
- 5m 1h 24h+
- Prometheus
- Grafana `dest_smsc` `source_smsc`

SMS-C

[←](#) [README](#)

概要

SMS-Cは、Mnesiaを用いたSMS管理システムです。

インストール

- 依存関係のインストール
- 実行環境の準備
- SMSC**のインストール
- ポート番号の指定 (1-255)
- デフォルトの設定
- 起動/停止コマンド
- 実行ファイル `runtime.exe` の実行
- 設定ファイルの読み込み
- Web UI** の CRUD 操作
- データベースの接続
- ログ出力の設定
- ヘルプ表示

使い方

起動

実行コマンド

欄名	型別	説明	デフォルト値
rule_id	整数	規則ID	0
calling_prefix	文字列/nil	発信番号文字列 (nil = 未指定)	空文字列
called_prefix	文字列/nil	着信番号文字列 (nil = 未指定)	空文字列
source_smsc	文字列/nil	送信元SMS-C番号 (nil = 未指定)	空文字列
calling_match	文字列/nil	発信番号一致条件	空文字列
calling_replace	文字列/nil	発信番号置換条件	空文字列
called_match	文字列/nil	着信番号一致条件	空文字列
called_replace	文字列/nil	着信番号置換条件	空文字列
priority	整数	優先順位 (1-255)	1
description	文字列	説明	空文字列
enabled	ブール値	有効/無効	有効
continue	ブール値	続行フラグ (false)	有効

注意事項: 特定のフィールドには制約が適用されます。

フィールド

フィールド名

1. 必須フィールド
2. 任意フィールド
 - calling_prefix
 - called_prefix
 - source_smsc

3. `calling_match` `calling_replace`

- `calling_match` `calling_replace`

- `called_match` `called_replace`

4. `continue`

- `continue: false` →

- `continue: true` → 2

5. `continue`

`continue`

`continue`

□□□□

□□□□□□
□□□□□□

□□□□ = □□□□□□□□
□□□□□□ = □

ore OmniCore OmniCall OmniRAN OmniCharge Platform 5A □□□□

□□□□□□
□□□□□□□□□□

□□□□

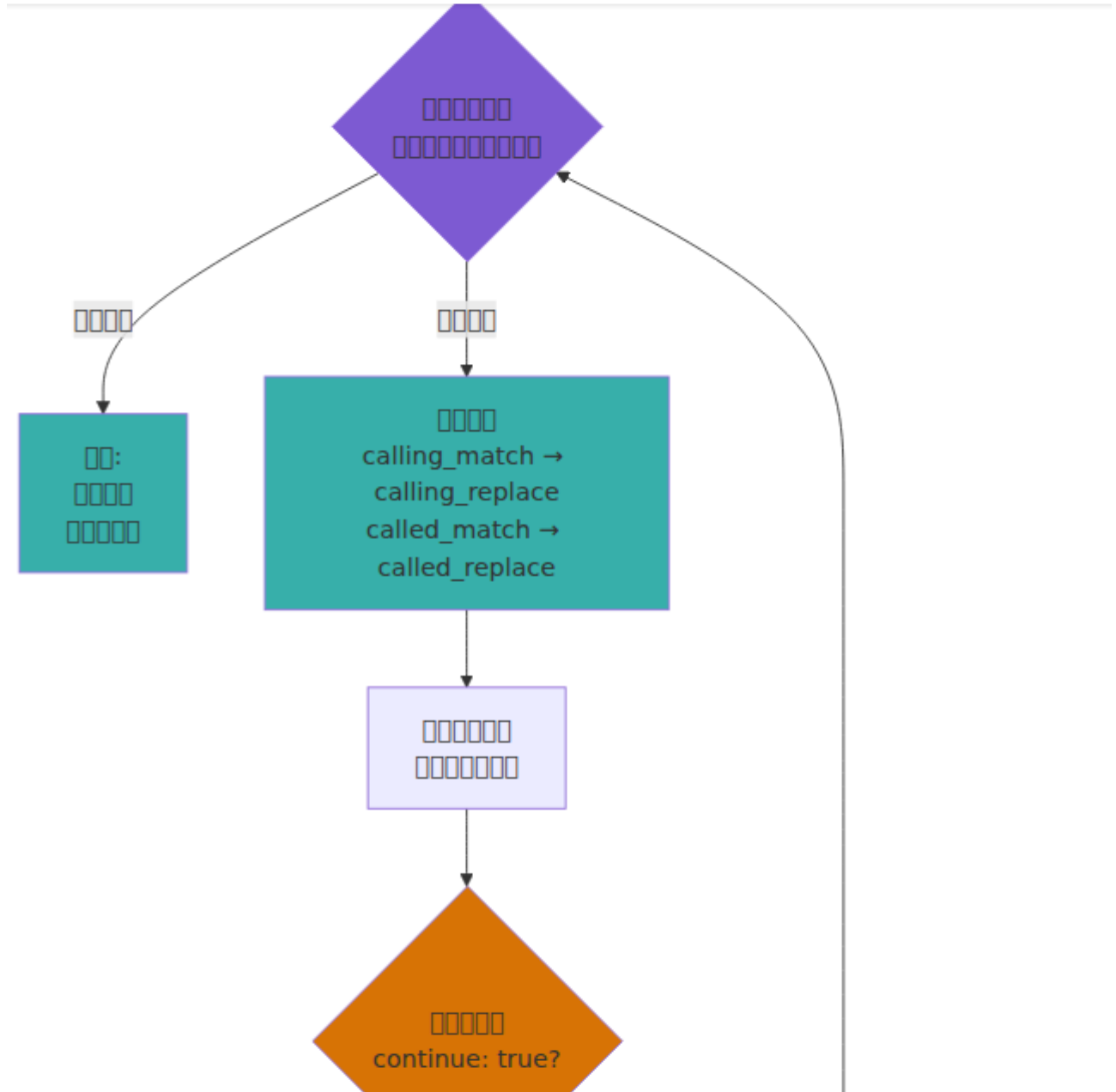
□□:
□□□□
□□□□□□

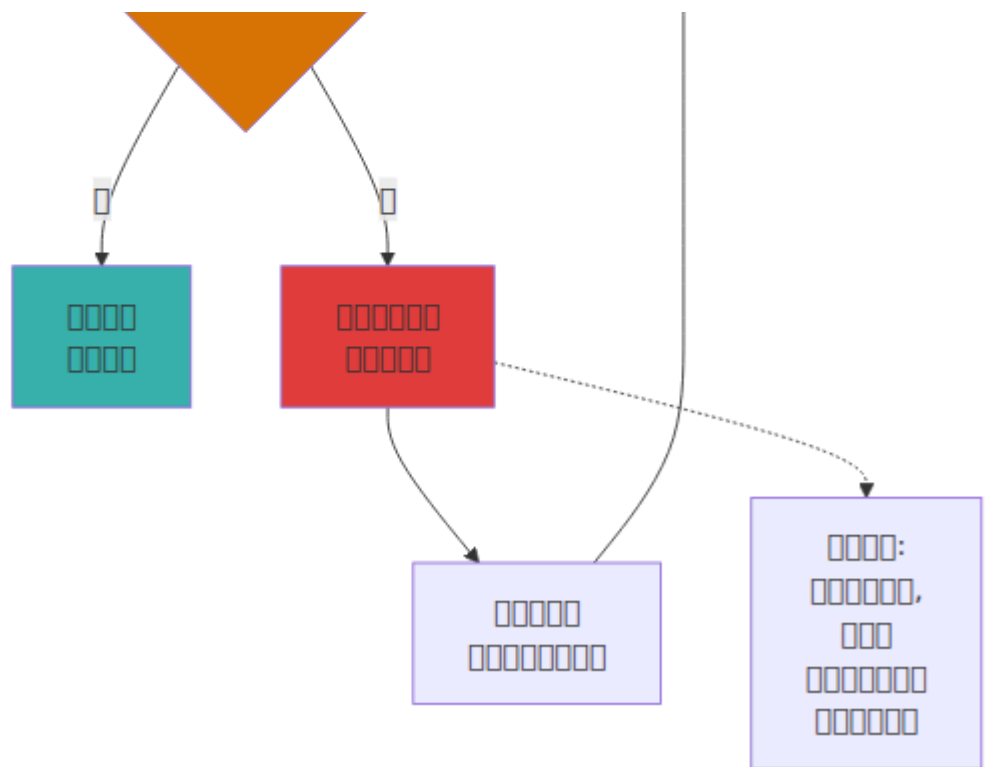
□□□□

□□□□
calling_match →
calling_replace
called_match →
called_replace

□□□□□□
□□□□□□□□

□□□□□□
continue: true?





□□□

- nil □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□ nil □□/□□□□□□□□□□□□□□

□□□□□□□□□□□□

Parse error on line 20: [...] style R1 fill:#38B2AC style R -----^
 Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW',
 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW',
 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS',
 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

□□

□□□□

□□□\n□□?

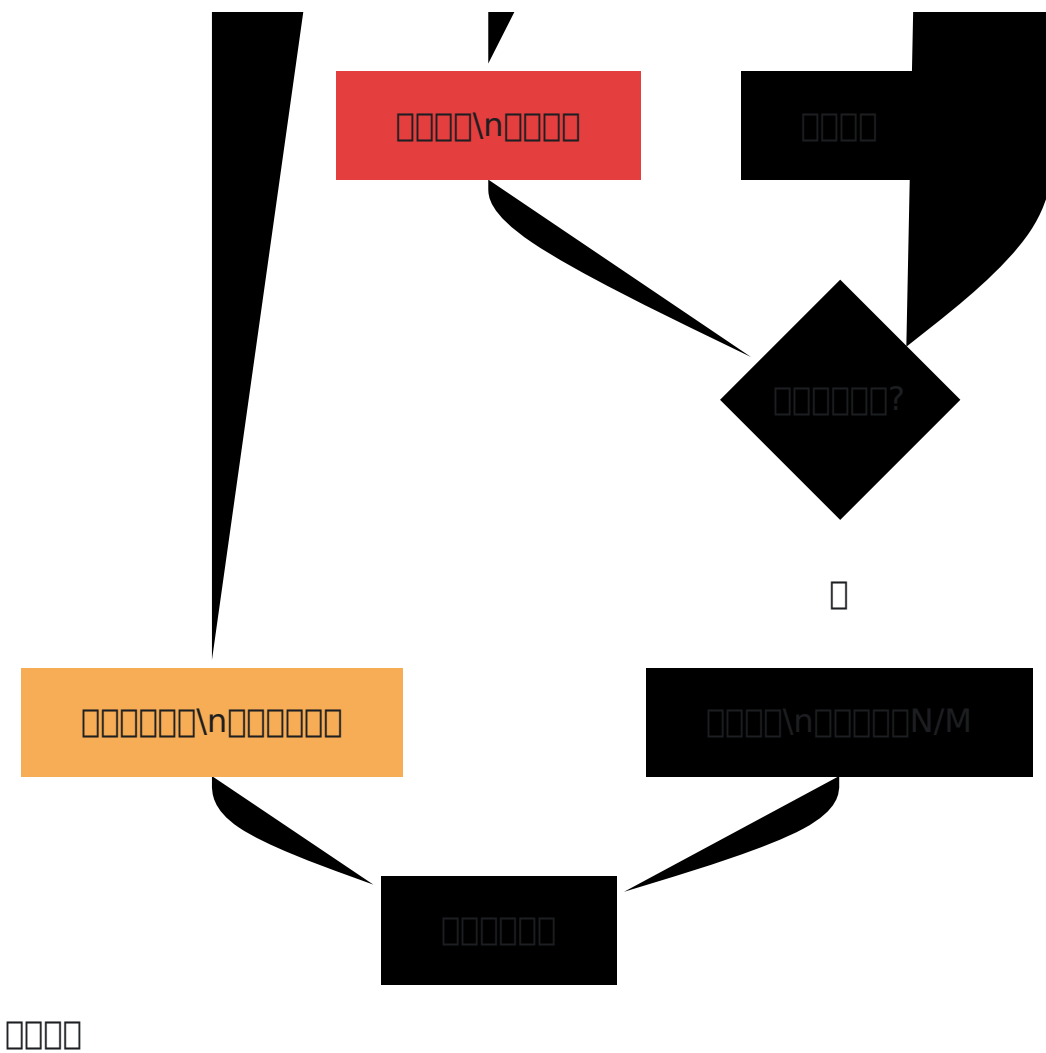
□\nconfig/runtime.exs□□
□□

□□□□□□□□□□

□□□□□□

□□?

□□□Mnesia




```

# config/runtime.exs
config :sms_c, :translation_rules, [
  # 1000000000+1
  %{
    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: "1000000000SMSC1000000000+1",
    enabled: true,
    continue: false
  },

  # 0000000000
  %{
    calling_prefix: "00",
    called_prefix: nil,
    source_smsc: nil,
    calling_match: "^00(.+)$",
    calling_replace: "+\\1",
    called_match: nil,
    called_replace: nil,
    priority: 5,
    description: "0000000000+",
    enabled: true,
    continue: true # 00000000
  },

  # 00000000000000
  %{
    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: "00000000000000",
  }
]

```

```
enabled: true,  
continue: false  
}  
]
```


□□

□□□□

□□□□

□□□\n□□?

□□□□□□□□□□

ore OmniCore OmniCall OmniRAN OmniCharge Platform
▼ 5GC ▼ ▼ ▼ ▼ ▼ ▼

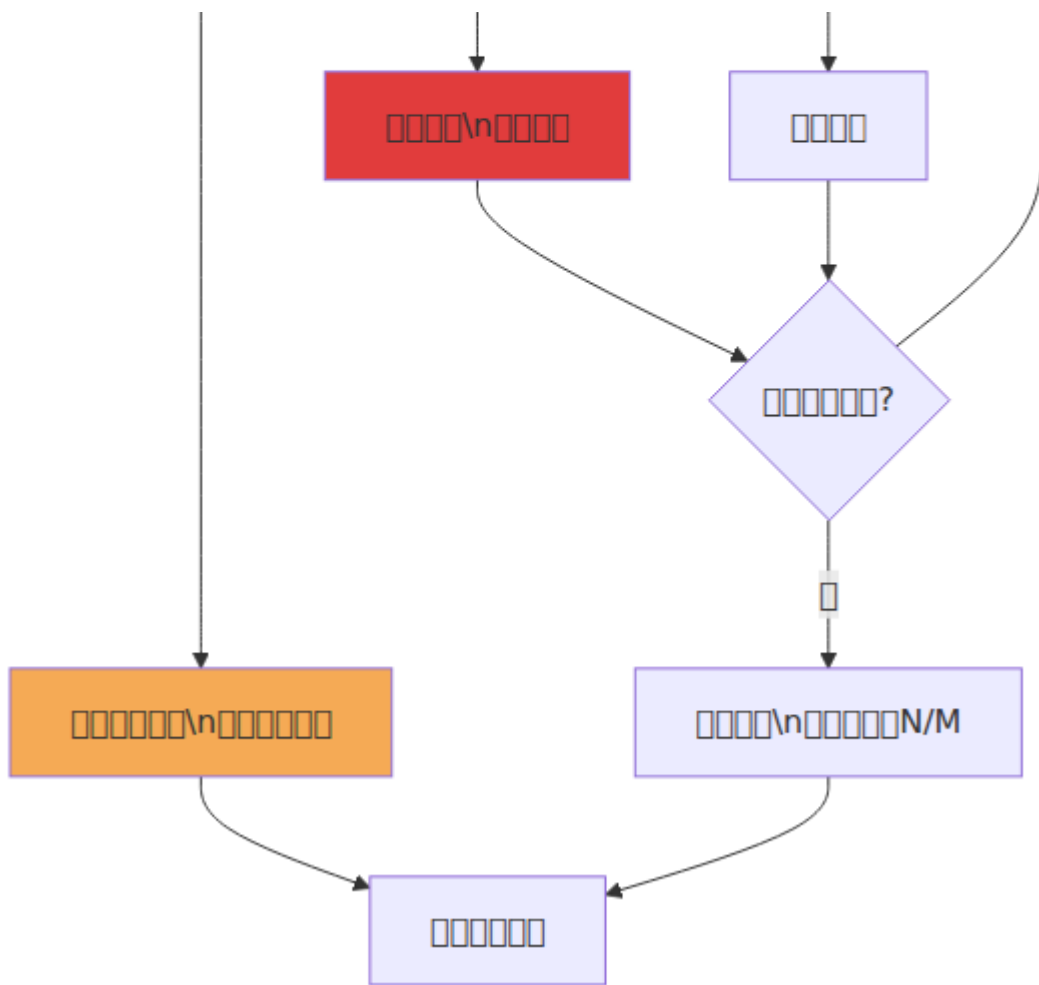
□□□□□□□□□□

□□□□□□

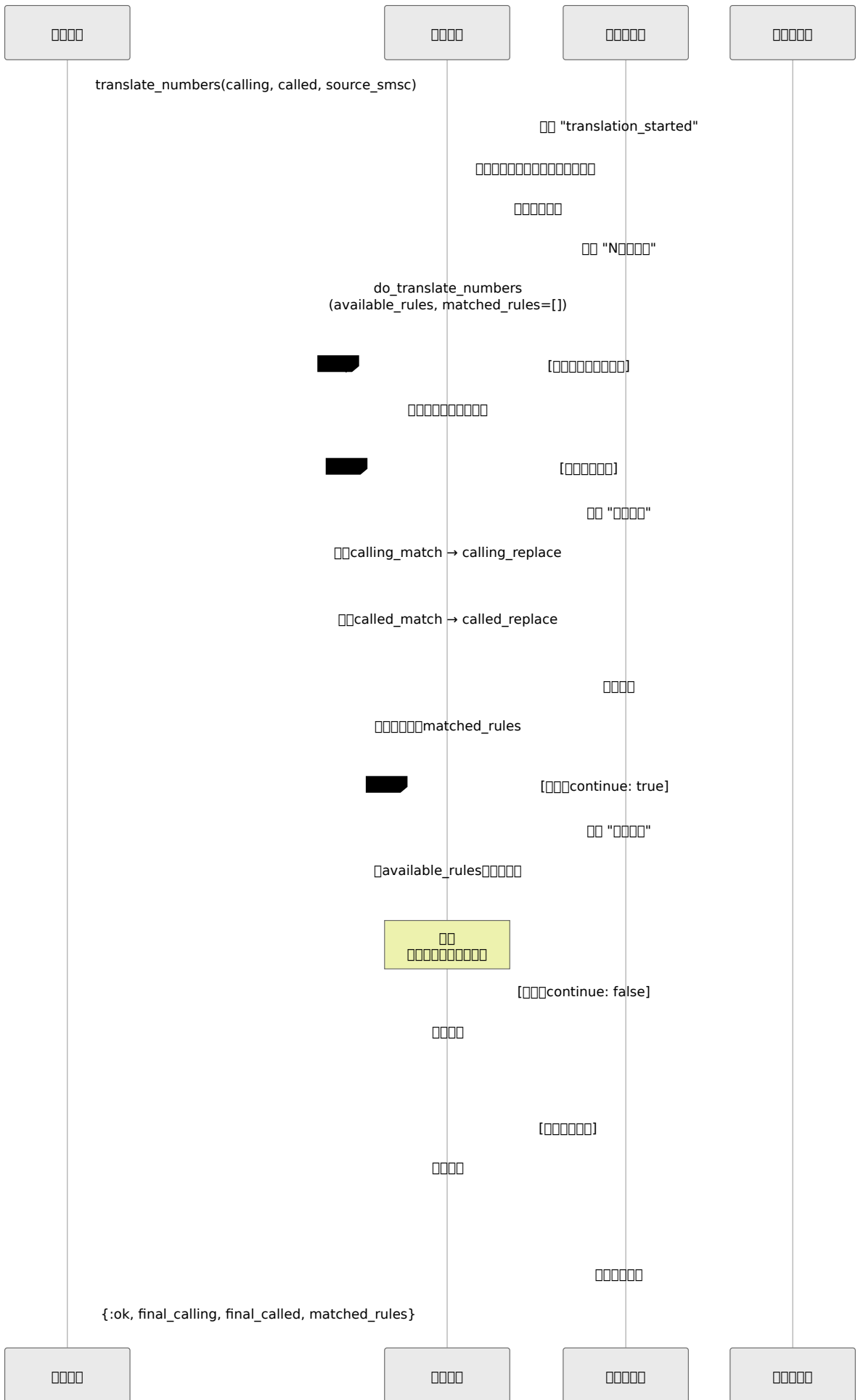
□□?

□□□Mnesia





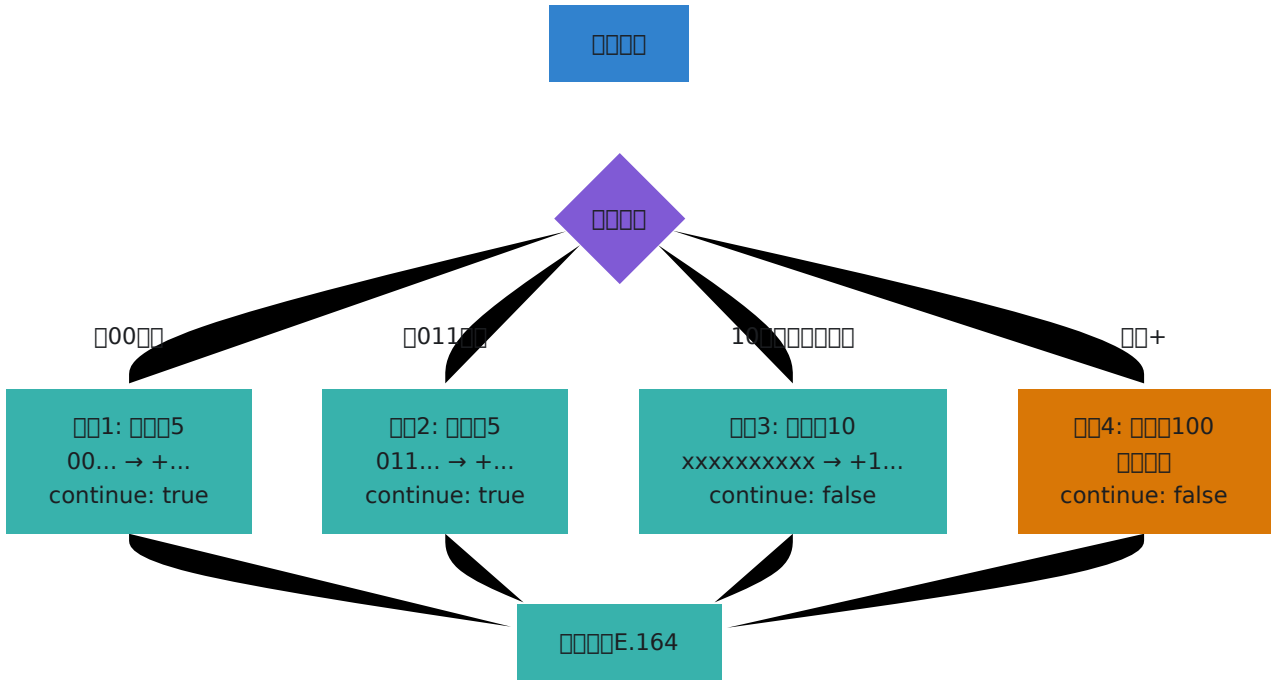
□□□□□□



□□□□

□□□□□□□□

□□□□□□□□□□E.164□



□□□□□□□□

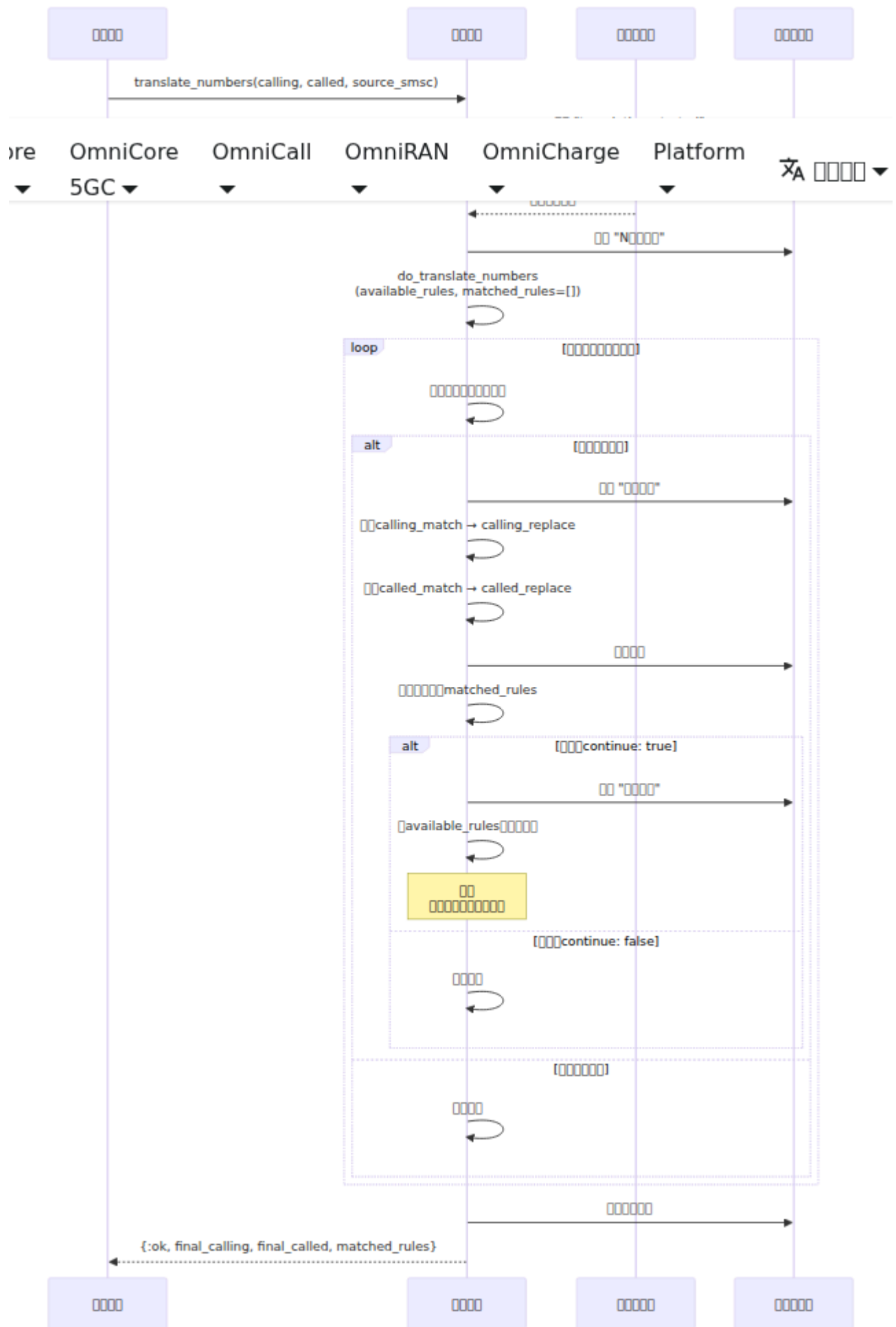
□□□□□□□□□□□□□□□□□□□□

Parse error on line 2: ...chart TD I[□□: "5551234567"] --> S1[-----
 ^ Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND',
 'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND_STOP', 'TAGEND',
 'TRAPEND', 'INVTRAPEND', 'UNICODE_TEXT', 'TEXT', 'TAGSTART', got 'STR'

□□

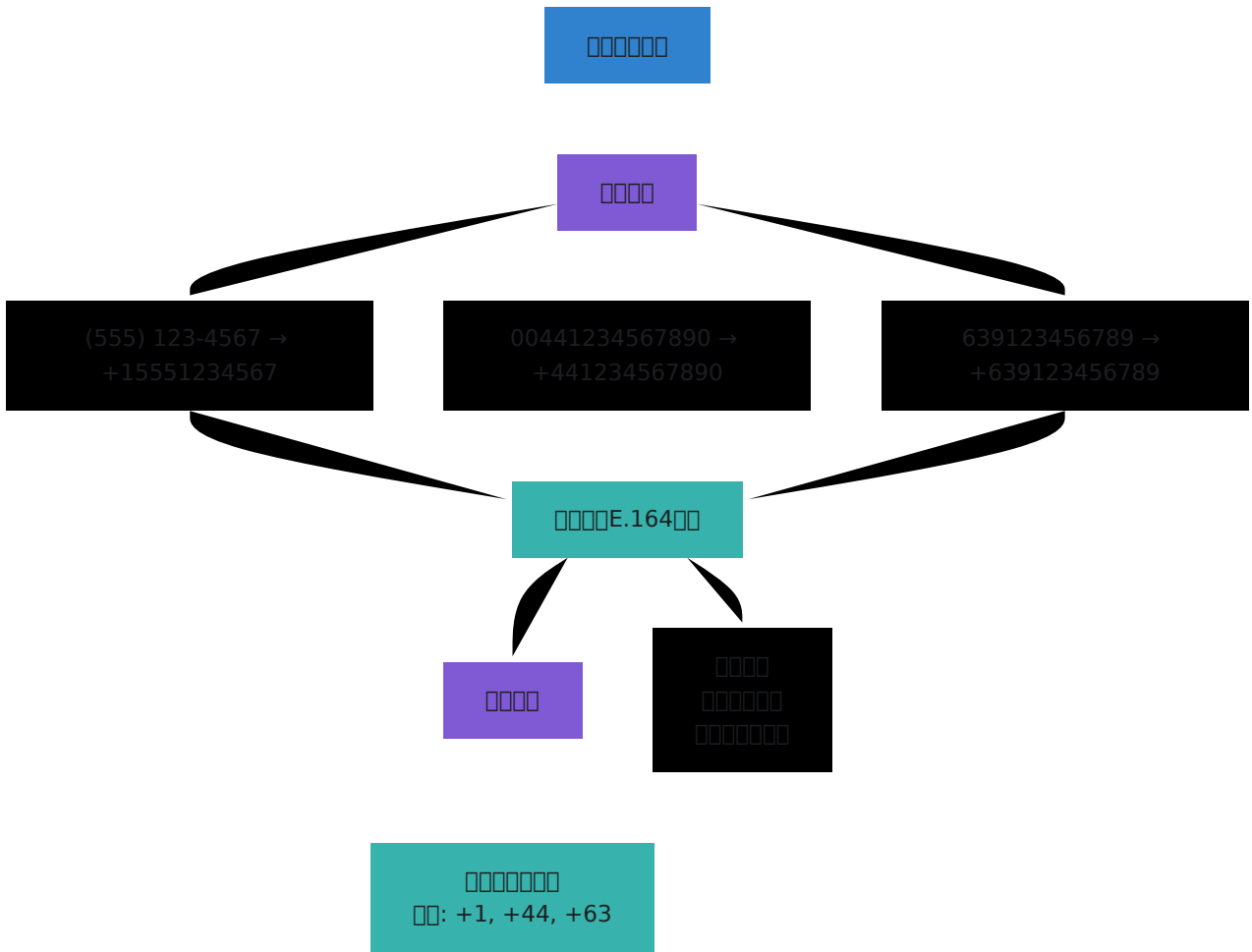
SMSC□□□□

□□□□□□□□□□□□□□



□□□□□□□□

□□□□□□□□□□□□□□□□□□



□□□□□□□□

□□□□□□□□□□□□□□□□

Parse error on line 18: ... style Input fill:#3182CE style R -----^
 Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW',
 'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW',
 'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS',
 'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

□□

Web

UI

`/number_translation`

-
-
-
-
-
- `continue: true`
- JSON

1.
 - "+1" "44"
 - "+639" "1555"
 - SMSC
2.
 -
 -
3. 1-255
4.
 -
 -
- 5.
6. " " "

-
-

- 消息接收方收到消息“↓ 消息”

消息

1. 消息接收方“消息”
2. 消息接收方
3. 消息“消息”

消息

- 消息/消息接收方
- ↓ 消息接收方
- 消息接收方
- 消息接收方

消息

消息接收方 `/translation_simulator`

消息

- 消息接收方
- 消息接收方
- 消息接收方
- 消息接收方
- 消息接收方
- 消息接收方10消息

消息

1. 消息接收方
 - 消息接收方
 - 消息接收方
 - 消息接收方
2. 消息“消息”
3. 消息接收方
 - 消息接收方

- 國際貿易與國際金融
- 國際貿易與國際金融
 - 國際貿易與國際金融
 - 國際貿易與國際金融
 - 國際貿易與國際金融
 - 國際貿易與國際金融“↓”
 - 國際貿易與國際金融
 - 國際貿易與國際金融

4. 國際貿易與國際金融

5. 國際貿易與國際金融

國際貿易

□□□□

□□□□: 5551234567 → +1-555-123-4567

□□□□: 9078720155 → +1-907-872-0155

✓ □3□□□□□□

□□□□

00

□□1

□□ #1 (□□□10)	↓ □□
□10□□□□□□□□	
□□: 9078720155 → +19078720155	

□□2

□□ #2 (□□□20)	↓ □□
□□□□□□□□□□	
□□: +19078720155 → +1-907-8720155	

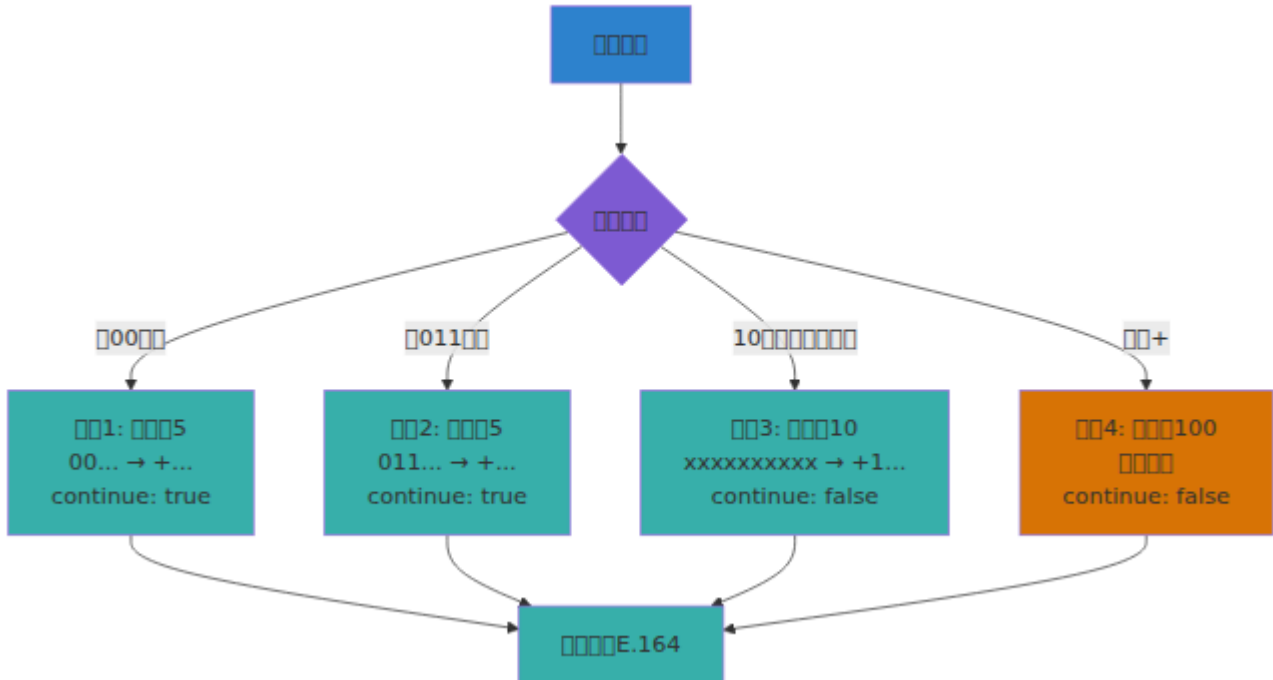
□□3

□□ #3 (□□□30)	
□□□□□□□□	
□□: +1-907-8720155 → +1-907-872-0155	

000

API

000000



0000

translate_numbers

- calling_number
- called_number
- source_smsc
- message_id

000

- {ok, translated_calling, translated_called, [rules_applied]} -
-
-
-

```
# []
{:ok, new_calling, new_called, rules} =
  NumberTranslation.translate_numbers(
    calling_number: "5551234567",
    called_number: "9078720155",
    source_smsc: "domestic_gateway",
    message_id: "msg_123"
  )

# []
if rules != [] do
  Logger.info("[] #{length(rules)} []")
  Enum.each(rules, fn rule ->
    Logger.info(" - [] ##{rule.rule_id}: #{rule.description}")
  end)
end
```

□□□□□□

```
# □□□□□
{: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: "□10□□□□□+1",
  enabled: true,
  continue: false
})

# □□□□
{:ok, updated_rule} = NumberTranslation.update_rule(rule_id, %{
  enabled: false,
  description: "□□□□□□□"
})

# □□□□
:ok = NumberTranslation.delete_rule(rule_id)

# □□□□□□
rule = NumberTranslation.get_rule(rule_id)

# □□□□□□
all_rules = NumberTranslation.list_rules()

# □□□□□□□□□□□□□□□□
enabled_rules = NumberTranslation.list_enabled_rules()
```

📄/📄📄📄

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

# 📄📄JSON📄📄
json = Jason.encode!(backup, pretty: true)
File.write!("translation_rules_backup.json", json)

# 📄📄📄📄📄📄📄📄📄
{:ok, %{imported: 3, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :merge)

# 📄📄📄📄📄📄📄📄📄
{:ok, %{imported: 5, failed: 0}} =
  NumberTranslation.import_rules(backup, mode: :replace)
```

📄📄📄📄

📄📄📄📄

1. 📄📄📄📄📄
 - **1-10**📄📄📄📄📄📄📄📄📄📄📄📄
 - **11-50**📄📄📄📄📄📄
 - **51-100**📄📄📄📄
 - **101+**📄📄📄📄📄📄📄
2. 📄📄📄📄**continue**
 - 📄📄📄📄📄📄📄📄📄📄📄📄 `continue: true`
 - 📄📄📄📄📄📄📄📄 `continue: false`

- 000000003-4000000000

3. 000000

- 0000000000
- 00000000000000“5551234567 → +15551234567”
- 0000000000/00

4. 00000000

- 0000000000000000
- 0000000\1\2000000000
- 0000000000000000

00

1. 00000000

- 0000000000
- 0000000000000000
- 000000000000

2. 00000000

- 0000000000000000
- 0000000000
- 0000000000

3. 0000000000

- 0005000000000000
- 0000000000000000
- 00Telemetry00000000

00

1. 0000000000

- 0000000000000000

- `continue`
- `continue`

2. `continue`

- `continue`
- `continue`
- `continue`

3. `continue`

- `message_id`
- `message_id`
- `message_id`

4. `continue`

- `continue`
- `continue`
- `continue`
- `continue`

正则表达式

1. 正则表达式

- `10` `^(\d{10})$`
- `+` `^\+(\d+)$`
- `0+` `^0+(\.+)$`
- `(\d{3})(\d{3})(\d{4})` `^(\d{3})(\d{3})(\d{4})$` → `\1-\2-\3`

2. 正则表达式

- `(\d{3})(\d{7})` `^(\d{3})(\d{7})$`
- `+1\1\2` `+1\1\2`
- `(\d{1,3})(\d+)` `^\+(\d{1,3})(\d+)$` → `00\1\2`

3. 正则表达式

□□□□

1. □□□□□□□□□□
2. □□□□□□□□□□
3. □□□□□□□□
4. □□□□□□□□□□
5. □□□□□□□□continue□□

□□□□/□□□□

□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□

- □□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□

□□□□

1. □□□□□□□□□□□□
2. □□□□continue: true□□□□
3. □□□□□□□□
4. □□□□□□□□□□
5. □□□□□□□□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□

□□□□

- □□□□□□□□□□□□continue: true
- □□□□□□□□□□□□
- □□□□□□□□□□□□

□□□□

1. □□□□□□□□□□
2. □□□□□□continue□□
3. □□□□□□□□
4. □□□□□□□□continue: false

□□□□□□□□□□

□□□□□□□□□□□□

□□□□

- □□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□

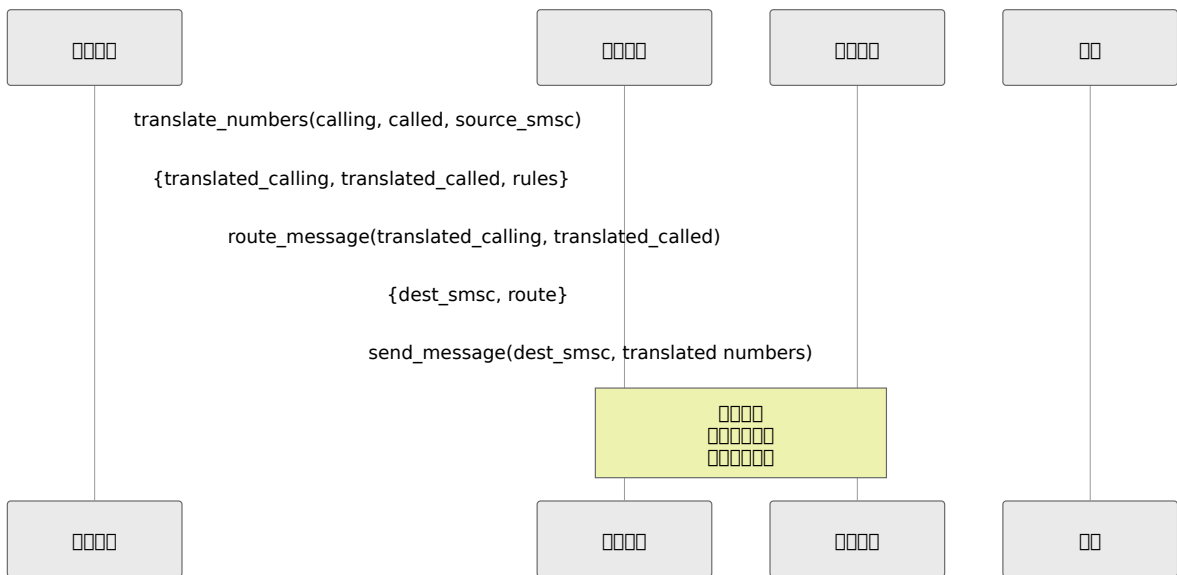
□□□□

1. □□□□□□□□□□□□□□□□
2. □□□□□□□□
3. □□□□□□□□□□□□
4. □□□□□□□□□□translate_numbers

□□□□

□□□□□□

□□□□□□□□□□□□□□□□



0000

EventLogger

- translation_started: 0000
- translation_candidates: 00000000
- translation_matched: 00000000
- translation_calling: 00000000
- translation_called: 00000000
- translation_continue: 0000 continue=true 0000
- translation_none: 000000

message_id translate_numbers/1 0000

Telemetry

Telemetry

```
:telemetry.attach(  
  "number-translation-handler",  
  [:sms_c, :number_translation, :translate, :stop],  
  fn _event_name, measurements, metadata, _config ->  
    # measurements: %{duration: []}  
    # metadata: %{rules_applied: [], ...}  
  end,  
  nil  
)
```

□□□□□□□□□□

- □□□□□□□□p50□p95□p99□
- □□□□□□□□□□
- □□□□□□□□□□□□
- continue□□□□□□□□

□□

Mnesia□□□□□□□□□□□□□□□□□□□□□□□□□□□□

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

□□

□□□□

□□□□□□□□□□

□□□□□

1. □□□□□□

2. □□□□□□□

□□□□□□□?

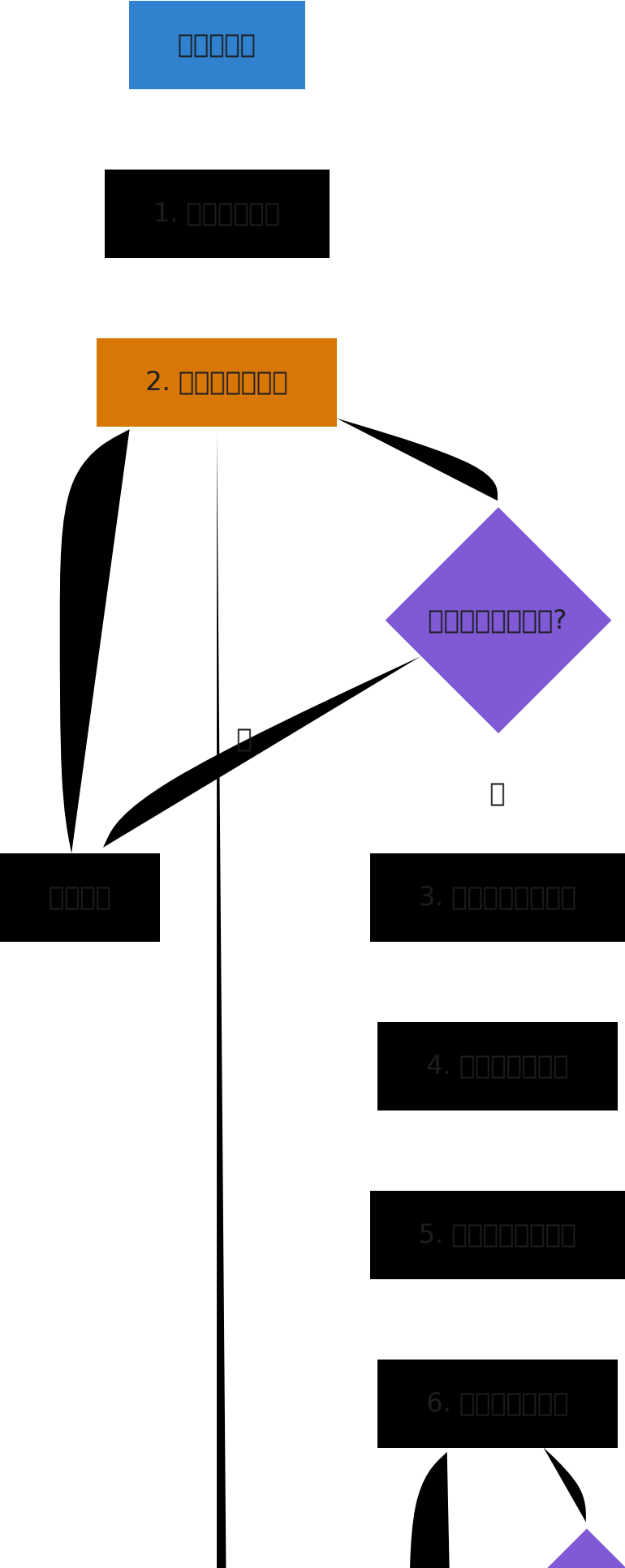
□□□□

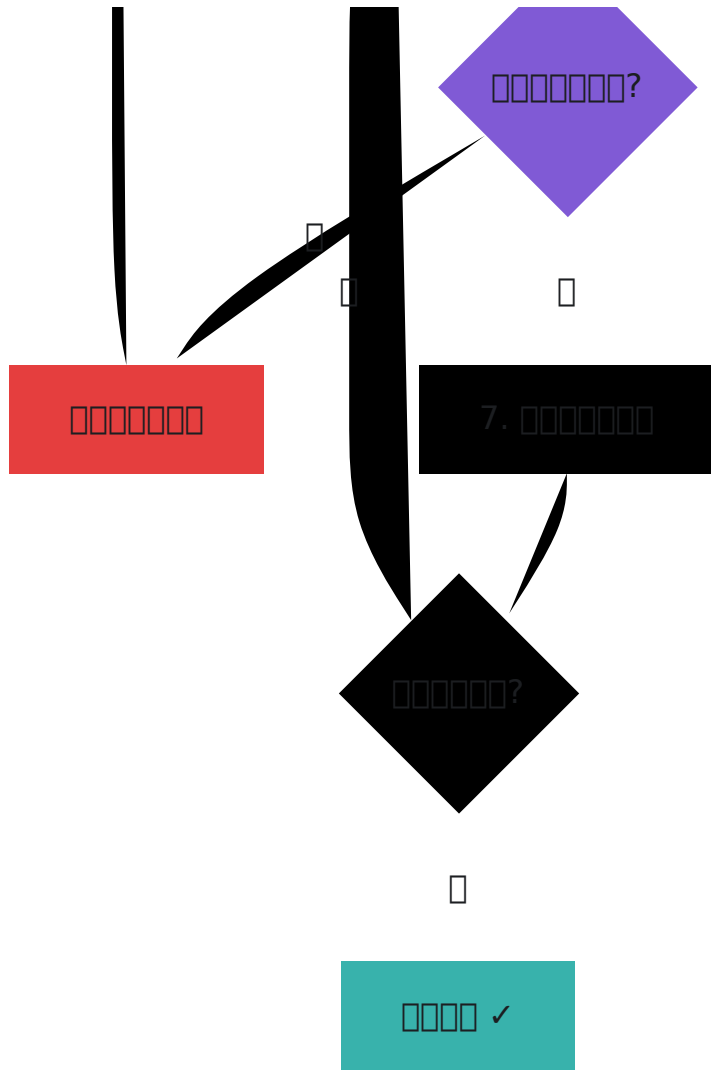
3. □□□□□□□

4. □□□□□□□

5. □□□□□□□

6. □□□□□□□





□□

□□**1**□□**?****?****?**□□□□□

□□□□□□□□□□□□□□□□E.164 (+1XXXXXXXXXX)

```

# 00100000000000000000
%{
  calling_match: "^(\\d{10})$",
  calling_replace: "+1\\1",
  called_match: "^(\\d{10})$",
  called_replace: "+1\\1",
  priority: 5,
  description: "0010000000+1",
  enabled: true,
  continue: false
}

# 00201 + 10000000000000000000
%{
  calling_match: "^1(\\d{10})$",
  calling_replace: "+1\\1",
  called_match: "^1(\\d{10})$",
  called_replace: "+1\\1",
  priority: 10,
  description: "1XXXXXXXXXXXX000+1XXXXXXXXXXXX",
  enabled: true,
  continue: false
}

# 00000
# "5551234567" → "+15551234567"00010
# "15551234567" → "+15551234567"00020
# "+15551234567" → "+15551234567"0000000000

```

00**2**00000000000000000000

0000000000000000+000000000000

```

# 001000000000+000000000000
%{
  calling_match: "^00(.+)$",
  calling_replace: "+\1",
  called_match: "^00(.+)$",
  called_replace: "+\1",
  priority: 5,
  description: "000000000000+",
  enabled: true,
  continue: true # 000000
}

# 002000000000000000000000
%{
  calling_match: "^\\+(\\d+)$",
  calling_replace: "00\1",
  called_match: "^\\+(\\d+)$",
  called_replace: "00\1",
  priority: 10,
  description: "+00000000000000",
  enabled: true,
  continue: false # 00
}

# 000000
# 001"00441234567890" → "+441234567890"00010000
# 002"+441234567890" → "00441234567890"00020000
# 000"00441234567890"
# 000000[0010002]

```

003 SMSC 00000

000000SMSC00000000

```

# 00100000SMSC - 0000000050
%{
  source_smsc: "trusted_gateway",
  calling_match: nil, # 000
  calling_replace: nil,
  called_match: nil,
  called_replace: nil,
  priority: 5,
  description: "000000000000",
  enabled: true,
  continue: false
}

# 00200000SMSC - 00000000100
%{
  source_smsc: "untrusted_gateway",
  calling_match: "^(.*)$",
  calling_replace: "+VALIDATE\1",
  called_match: "^(.*)$",
  called_replace: "+VALIDATE\1",
  priority: 10,
  description: "00000000000000",
  enabled: true,
  continue: false
}

# 003000SMSC00000000001000
%{
  source_smsc: nil, # 000
  calling_match: "^(\\d{10})$",
  calling_replace: "+1\1",
  called_match: "^(\\d{10})$",
  called_replace: "+1\1",
  priority: 100,
  description: "000010000000+1",
  enabled: true,
  continue: false
}

```

□□**4**□□□□□□□□

□□□□□□ → □□□□□□ → □□□□□□□□

```

# 010000000000
%{
  calling_match: "^0+(.)$",
  calling_replace: "\\1",
  called_match: "^0+(.)$",
  called_replace: "\\1",
  priority: 5,
  description: "00000",
  enabled: true,
  continue: true
}

# 00200000000000000000
%{
  calling_match: "^(\\d{10})$",
  calling_replace: "+1\\1",
  called_match: "^(\\d{10})$",
  called_replace: "+1\\1",
  priority: 10,
  description: "01000000+1",
  enabled: true,
  continue: true
}

# 003000000000000000
%{
  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: "0000+1-XXX-XXX-XXXX",
  enabled: true,
  continue: false
}

# 00000
# 000"005551234567"
# 001"005551234567" → "5551234567"00010000
# 002"5551234567" → "+15551234567"00020000
# 003"+15551234567" → "+1-555-123-4567"00030000

```

```
# "+1-555-123-4567"  
# [123]
```

□□

□□□□□□□□

- □□□□□□ test/sms_c/messaging/number_translation_test.exe □□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□Mnesia□□□□ :mnesia.table_info(:translation_rule, :size)
- □□Telemetry□□□□□□□□□□

SMS-C

← | [README](#)

SMS-C

-
-
-
-
-
-
-
-
-
-

1.

```
# API  
curl https://api.example.com:8443/api/status  
  
#  
# {"status":"ok","application":"OmniMessage","timestamp":"2025-10-30T08:00:00Z"}
```


2. Prometheus

Prometheus

- 24
- $< 1\%$
- < 1000
- $> 95\%$
-

3.

Web UI: https://sms-admin.example.com/message_queue

-
- < 5
- > 3
-

4.

Web UI: https://sms-admin.example.com/frontend_status

-
-
- 24

5.

Web UI: <https://sms-admin.example.com/logs>

-
-

- 接收消息
- 消息接收成功
- 消息接收失败

接收消息

接收消息成功

接收消息失败 Prometheus 告警

```
# 接收消息成功
increase(sms_c_message_received_count[1h])

# 接收消息失败
increase(sms_c_delivery_succeeded_count[1h])

# 接收消息失败率
rate(sms_c_delivery_succeeded_count[1h]) /
rate(sms_c_message_received_count[1h])
```

接收消息失败

- 接收消息失败率 > 95%

接收消息失败

- 接收消息失败率 > 50% 告警
- 接收消息失败率 > 200% 告警
- 接收消息失败率 90% 告警

接收消息

接收消息成功

接收消息失败

短信接收 (sms_c_message_received_count)

- 短信接收速率
- 短信接收速率
- 短信 `rate(sms_c_message_received_count[5m])`

短信处理停止时长 (sms_c_message_processing_stop_duration)

- 短信处理停止时长
- 短信 p95 > 1000ms
- 短信 `histogram_quantile(0.95, sms_c_message_processing_stop_duration)`

路由

路由失败 (sms_c_routing_failed_count)

- 路由失败
- 路由失败 > 0
- 短信 `increase(sms_c_routing_failed_count[5m])`

路由匹配 (sms_c_routing_route_matched_count)

- 路由匹配
- 路由匹配
- 短信 `sms_c_routing_route_matched_count`

投递

投递成功率

- 投递成功率
- 投递成功率 < 95%
- 短信 `rate(sms_c_delivery_succeeded_count[5m]) / rate(sms_c_delivery_queued_count[5m])`

投递尝试 (sms_c_delivery_succeeded_attempt_count)

- 投递尝试

- `p95 > 2`
- `histogram_quantile(0.95, sms_c_delivery_succeeded_attempt_count)`

🔍🔍

`(sms_c_queue_size_size)`

- `> 10,000`
- `sms_c_queue_size_size`

`(sms_c_queue_oldest_message_age_seconds)`

- `> 300`
- `sms_c_queue_oldest_message_age_seconds`

📌📌

📌📌📌

1. 📌📌📌

- 📌📌📌 5 📌📌
- 📌📌📌 5 📌📌
- 📌📌📌 24 📌📌

2. 📌📌📌📌

- 📌📌📌📌
- 📌📌📌📌
- 📌📌📌📌

3. 📌📌📌📌📌

- 📌📌📌📌📌
- 📌📌📌📌📌

- 24 時間

4. 認証

- ID
- パスワード
- SMSC
- 名前

5. 権限

- 権限
- 権限/ロール
- 権限
- 権限

6. 設定

- API 設定p95
- 権限設定p95
- ENUM 設定p95

設定

権限設定

```
# 短信 - 路由失败
- alert: RoutingFailures
  expr: increase(sms_c_routing_failed_count[5m]) > 0
  severity: critical
  description: "{{ $value }}" 5 分钟内"

# 短信 - 队列积压
- alert: QueueBacklog
  expr: sms_c_queue_size_pending > 10000
  severity: critical
  description: "队列积压 {{ $value }}"

# 短信 - 队列老化
- alert: OldMessagesInQueue
  expr: sms_c_queue_oldest_message_age_seconds > 300
  severity: critical
  description: "队列老化 {{ $value }}"

# 短信 - 前端断开
- alert: FrontendDisconnected
  expr: sms_c_frontend_status_count{status="disconnected"} > 0
  severity: critical
  description: "{{ $value }}"
```

短信系统告警配置

```

# 消息投递率低
- alert: LowDeliveryRate
  expr: rate(sms_c_delivery_succeeded_count[10m]) /
rate(sms_c_delivery_queued_count[10m]) < 0.90
  severity: warning
  description: "消息投递率低 {{ $value }}"

# 消息重试率高
- alert: HighRetryRate
  expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count) > 2
  severity: warning
  description: "95 分位数消息重试率 {{ $value }}"

# ENUM 枚举查找慢
- alert: SlowEnumLookups
  expr: histogram_quantile(0.95, sms_c_enum_lookup_stop_duration)
> 5000
  severity: warning
  description: "ENUM 查找慢 > 5 s"

# ENUM 枚举缓存命中率低
- 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: "ENUM 缓存命中率 {{ $value }}"

```

消息队列

消息队列 ID

消息 ID

1. **Web UI** 查看 `/message_queue`

2. 查看消息队列 ID

3. 查看消息 ID

API

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

1. **Web UI** `/message_queue`
- 2.
- 3.

1. **Web UI** “”
2. **API** `GET /api/events/12345`

1. `message_inserted` -
- ↓
2. `number_translated` -
- ↓
3. `message_routed` -
- ↓
4. `charging_attempted` -
- ↓
5. `message_delivered` -

1. message_inserted
↓
2. message_routed
↓
3. delivery_attempt_1 - 1회 시도
↓
4. delivery_attempt_2 - 2회 시도
↓
5. delivery_attempt_3 - 4회 시도
↓
6. message_dead_letter - 실패

1. 메시지 생성

1.1. 메시지 생성

- "sms" 메시지
- deliver_after: 1000 (1초)
- delivery_attempts: 0 (1회 시도)

1.2. 메시지 전송

- "sms" 메시지
- deliver_time: 1000 (1초)
- dest_smsc: 1234567890

1.3. 메시지 수신

- "sms" 메시지 delivery_attempts
- deadletter: true (실패)
- destination: 1234567890

2. 메시지 처리

SMS-C (Short Message Service Center) 메시지 처리

2.1. 메시지 처리

get_messages_for_smsc(smsc_name)

1. dest_smsc
2.
 - dest_smsc null
 - destination_msisdn
 - location
 -

MSISDN +447700900123 uk_gateway

```
#  
POST /api/locations  
{  
  "msisdn": "+447700900123",  
  "imsi": "234150123456789",  
  "location": "uk_gateway",  
  "expires": "2025-11-01T12:00:00Z"  
}
```

```
# dest_smsc  
POST /api/messages  
{  
  "source_msisdn": "+15551234567",  
  "destination_msisdn": "+447700900123",  
  "message_body": "Hello",  
  "source_smsc": "api"  
  # dest_smsc null  
}
```

uk_gateway

```
# 消息队列
GET /api/messages/queue?smsc=uk_gateway
```

```
# 消息队列 dest_smsc 为空
# 消息队列 uk_gateway 消息
```

消息队列

消息队列消息

- `locations` 消息队列 `destination_msisdn` 消息
- `location` 消息队列 SMSC 消息
- `expires` 消息队列

消息队列消息

消息队列

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

```
# 消息队列
# expires 消息 > 消息
```

消息队列

- 消息队列消息
- 消息队列 `location` 消息队列消息
-  消息队列消息

消息队列

消息队列

```
# [] delivery_attempts [] 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"
  }'
```

□□□□□

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

□□□□□□□□

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

□□□□

□□□□□□

Web UI □□□□ `/sms_routing`

□□ **API** □

```
# □□□□□□
curl https://api.example.com:8443/api/routes
```




□□□□□□□□

Prometheus □□□

```
# 增加过去1小时匹配的短信路由次数  
increase(sms_c_routing_route_matched_count[1h])
```

配置项

Web UI

1. 访问 `/sms_routing`
2. 输入“配置项”
3. 配置项
 - 配置项名称
 - 配置项描述
 - 选择 **SMSC** 配置项   
 - 选择 **SMSC** 配置项/配置项名称
 - 配置项名称包含 Diameter/HSS
 - 配置项名称包含 1-255
 - 配置项名称包含 1-100
 - 配置项名称
 - 配置项名称
4. 输入“配置项”

配置项

- 配置项 `+44`
- 选择 **SMSC** `uk_gateway`
- 配置项 50
- 配置项 100
- 配置项 "配置项"

配置项

配置项

配置项 1

- 配置项 `+44`

- SMSC `uk_primary`
- 50
- 70
- "70%"

2

- +44
- SMSC `uk_backup`
- 50
- 30
- "30%"

- SMSC `carrier_smp_bind`
- SMSC `local_msc`
-
- 50
- 100
- "X —"

Diameter/HSS HSS

1. `/simulator`
2.
 - `+15551234567`
 - `+447700900000`
 - SMSC
 -

3. 設定“送信先”

4. 送信先

- 送信先を指定する
- 送信先を指定する
- 送信先を指定する

送信先

- 送信先を指定する
- 送信先を指定する
- 送信先を指定する
- 送信先を指定する

送信先

Web UI

1. 送信先 `/sms_routing`
2. 送信先
3. 設定“送信先”
4. 送信先
5. 設定“送信先”

送信先

- 送信先を指定する“送信先”
- 送信先を指定する
- 送信先を指定する
- 送信先を指定する SMSC

送信先

Web UI

1. 送信先 `/sms_routing`
2. 送信先
3. 設定“送信先”

4. 配置

配置路由和中间件

路由/中间件

配置路由

1. 配置 `/sms_routing`
2. 配置“路由”
3. 配置 JSON 路由

配置

1. 配置 `/sms_routing`
2. 配置“路由”
3. 配置 JSON 路由
4. 配置中间件
 - 配置中间件
 - 配置中间件

配置

- 配置中间件
- 配置中间件
- 配置
- 配置

配置

配置

Web UI `/frontend_status`

配置

- 配置中间件“路由”

- 000000000000 < 90 00
- 0000000000

00 API

```
# 00000000
curl https://api.example.com:8443/api/frontends/active

# 00000000
curl https://api.example.com:8443/api/frontends/stats
```

00000000

000000

1. 000000000000
2. 000 SMS-C 000000
3. 0000000000
4. 000000000000 60 000000

00000000

1. 0000000000 POST /api/frontends/register
2. 00 API 0000000000
3. 00 JSON 00000000
4. 00 curl 00000000

00000000

```
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"
}'
```

□□□□□□

Web UI

1. □□□ `/frontend_status`
2. □□□□□□□□
3. □□“□□”
4. □□□□□□□□

□□ API

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

□□□□□

- □□□□□□□
- □□□□□□□□□□□□
- □□□□□□

□□□□□□□

□□□□□□□□ `config/runtime.exs` □□□□□□□□□□□□□□□□

□□□□□□□□

□□□□□□□

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

□□□□□□□

□□□□□□□□□□□□

□□ `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: "☐ 10 ☐☐☐☐☐☐ +1",
  enabled: true
}
```

☐☐☐☐☐☐☐☐

```
%{
  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: "☐ 00 ☐☐☐☐☐ +",
  enabled: true
}
```

☐☐☐☐☐☐☐☐☐☐

```
%{
  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: "A",
  enabled: true
}
```

- 1.
- 2.
3. `number_translated`
- 4.

`enabled: false`

```
%{
  ...
  enabled: false
}
```

データベース

データベース

データベース

データベース CDR データ

- **MySQL/MariaDB** `information_schema.tables` データ
- **PostgreSQL** `pg_database_size()` データ `psql` `\l+` データ

データベース CDR データ

CDR データ

- データ 30-90 データ
- データ
- データ

データベース

データベース

- **MySQL/MariaDB** `OPTIMIZE TABLE` データ
- **PostgreSQL** `VACUUM ANALYZE` データ `autovacuum`

データベース

Mnesia データ

データベース Mnesia データ

```
# データ IEx データ
:mnesia.table_info(:sms_route, :size)
:mnesia.table_info(:translation_rule, :size)
```

データベース Mnesia データ

```
# 安装Web UI
# 安装 /sms_routing
# 安装“Mnesia”

# 安装 Mnesia 备份
:mnesia.backup("/var/backups/sms_c/mnesia_backup.bup")
```

安装 Mnesia

```
# 安装 Web UI 包
# 安装 Mnesia
:mnesia.restore("/var/backups/sms_c/mnesia_backup.bup", [])
```

配置

配置 logrotate

```
# /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
}
```

验证

验证安装和配置

```
# 0000000000
systemctl restart sms_c
```

```
# 0000000000
# 0000000000
```

000000000000

```
systemctl restart sms_c
```

0000

- 000000000000
- 00 Prometheus 0000000000
- 000000000000
- 0000000000

00000000

0000

1. 000000

- config/runtime.exs
- config/config.exs
- config/prod.exs 000000

2. 0000Mnesia00

- 00 Web UI 00
- 0 Mnesia 0000

3. SQL CDR 0000

- 0000
- 000000000000

4. TLS 备份

- `priv/cert/*.crt`
- `priv/cert/*.key`

备份脚本

备份目录

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

BACKUP_DIR="/var/backups/sms_c/$(date +%Y%m%d)"
mkdir -p $BACKUP_DIR

# 备份配置
cp -r /opt/sms_c/config $BACKUP_DIR/

# 备份证书
cp -r /opt/sms_c/priv/cert $BACKUP_DIR/

# 设置权限
chmod 600 $BACKUP_DIR/cert/*

echo "备份完成: $BACKUP_DIR"
```

备份脚本


```
#!/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

# SQL CDR
# MySQL/MariaDB mysqldump --single-transaction
# PostgreSQL pg_dump -F c

# 
# - mysqldump pg_dump
# - 
# - 
# - 30

# 
find $BACKUP_DIR -name "sms_c_*.gz" -mtime +30 -delete

echo "sms_c_${DATE}"
```

□□□□□

```
#!/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

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

echo "routes_${DATE}.json"
```

□□□□crontab□

```
# 0000 2 0
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
```

0000

0000

```
# 000000
systemctl stop sms_c

# 000000
cp -r /var/backups/sms_c/20251030/config/* /opt/sms_c/config/

# 0000
cp -r /var/backups/sms_c/20251030/cert/* /opt/sms_c/priv/cert/

# 000000
systemctl start sms_c
```

00 **SQL CDR** 0000

0000000000000000

- **MySQL/MariaDB**000000 mysql 00000000
- **PostgreSQL**000 pg_restore 00000000000000

00000000000000 SMS-C 00000000000000

000000

1. 000 Web UI `/sms_routing`
2. 00“0000”
3. 0000 JSON 00
4. 00“00”00
5. 0000

□□□□

□□□□□□

□□□□□□

Prometheus □□□30 □□□□□

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

□□□□□□

```
-- □□□□□□  
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;
```

□□□□

CPU □□□□

- □□□ < 50% □□
- □□ > 70% □□
- □□□ > 90%

□□□□□□

- □□□ < 70% □□
- □□ > 80%
- □□□ > 90%

□□□□□□

- CPU < 60%
- CPU > 75%
- CPU > 85%

Network

- CPU < 1000
- CPU > 5000
- CPU > 10,000

System

System Configuration

- CPU > 70%
- Memory > 80%
- Disk I/O

System Performance

- CPU > 50%
- CPU > 5000 msg/sec
- Memory
- Disk I/O

System Health

- System Configuration
- System Performance
- System Health
- System Security

□□□□

□□□□□

□□□□□□□□

- □□□□□□
- □□□□□□
- □□□□□□
- API □□□□

□□1 □□□□□□□□

- □□□□□ < 80%
- □□□□□□
- □□□□□ > 10%
- □□□□□□

□□4 □□□□□□□□

- □□□□□□
- □□□□□□ 80-95%
- □□□□□□
- ENUM □□□□

□□24 □□□□□□□□

- □□□□□□
- □□□□□□
- □□□□□□□□

□□□□□□□

1. □□□□□□

- □□ Prometheus □□
- □□□□□□□□

- 00000000
- 000000

2. 00000

- 00000000
- 000000
- 00000000OCSDNS
- 0000000000

3. 00000

- 000000000000
- 00000000000000000000
- 0000000000000000
- 00000000

4. 000

- 000000
- 000000000000
- 00000000
- 000000

5. 00000

- 0000
- 00000000
- 000000
- 0000

6. 0000

- 000000
- 0000/00
- 000000
- 000000

□□□□

□□□□□□

1. □□□□□□□□
2. □□□□□□□□□□□□
3. □□□□□□□□
4. □□ Prometheus □□□□□□
5. □□□□□□□□□□/□□

□□□□□□

1. □□□□□□□□
2. □□□□□□□□□□□□
3. □□□□□□□□□□
4. □□□□□□□□□□□□□□
5. □□□□□□□□□□□□□□□□

□□□□□□

1. □□□□□□□□□□
2. □□□□□□□□
3. □□□□□□□□
4. □□□□ API □□□□
5. □□□□□□□□

□□□□□□□□

1. □□□□□□□□□□□□
2. □□□□□□□□□□□□
3. □□□□□□□□□□CPU/□□□□
4. □□ ENUM □□□□□□
5. □□□□□□□□□□

□□□□□□□□□□□□□□ □□□□□□□□

README

[← 概要](#) | [README](#)

このリポジトリは SMS-C のソースコードを公開しています

概要

SMS-C は Mnesia を利用して SQL を実行し CDR を生成し **1,750** 行/秒 を処理します

環境

Intel i7-8650U @ 1.90GHz (8 コア) を使用

項目	値	単位 (秒)	備考
生成 (1000)	1,750 行/秒	0.58 秒	SQL 21 行
生成 (100)	1,750 行/秒	0.57 秒	SQL 21 行
100 SMSC 処理	800 行/秒	1.25 秒	
メモリ使用量	62 KB	-	50%

処理速度は約 1.5 秒あたりです

インストール

- インストール
- Mnesia のインストール
- CDR の生成
- 実行
- デバッグ

□□□□□□

SMS-C □□□□□□□□□□□□□□

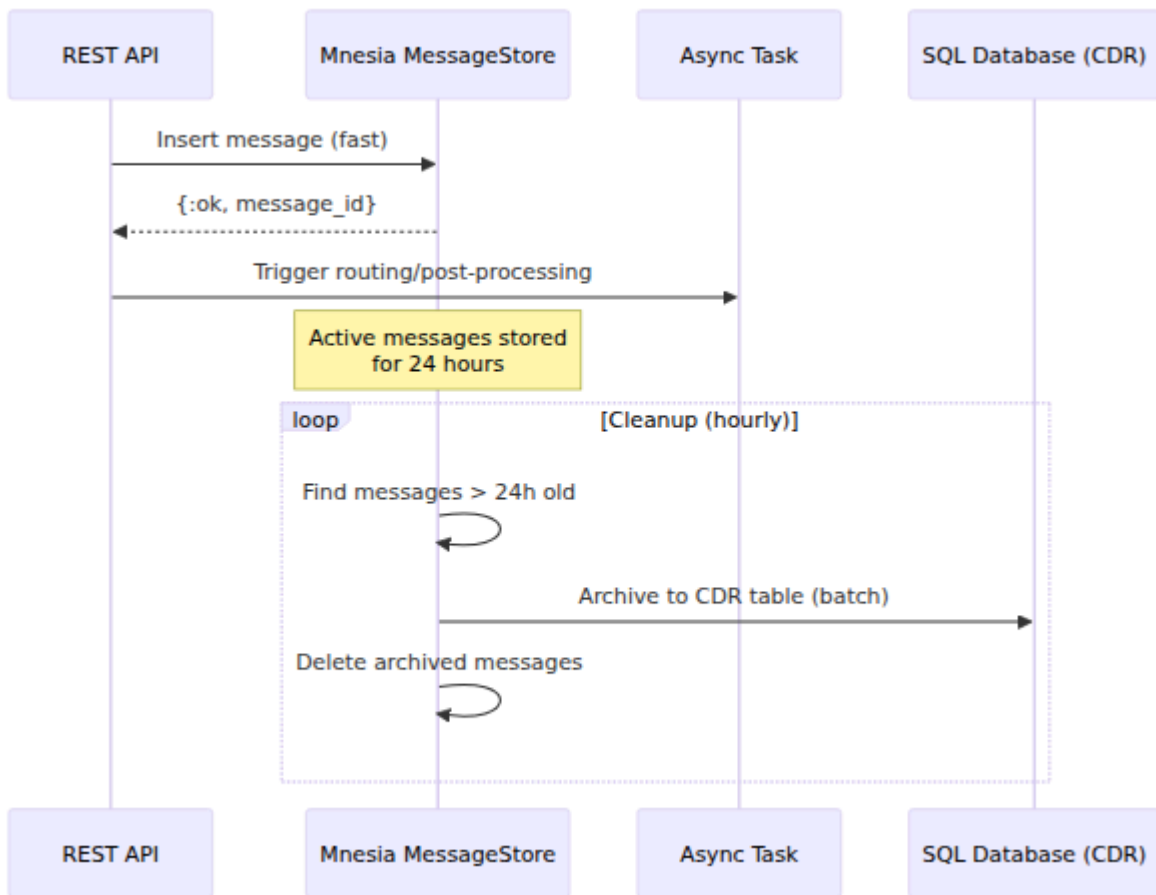
□□□□□□ (Mnesia)

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□ (disc_copies)
- □□□1,750 □□/□□□□□□□0.58 □□□□
- □□□□□□ (□□□24 □□)
- □□□□□□□□ Mnesia □□□□□□□□

CDR □□ (SQL □□□)

- □□□□□□□□□□□□
- □□□SQL □□□ (MySQL/MariaDB □ PostgreSQL) □□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□ (□□□□□□□□□□)
- □□□□□□□□□□□□

□□□



Mnesia □□

□□□□□□

```
# config/runtime.exs
config :sms_c,
  message_retention_hours: 24 # □□□24 □□
```

□□□□□

- □□□ (>1M □□/□)□12-24 □□□□
 - □□□ Mnesia □□□
 - □□□□□
 - □□□□□□□ MySQL

- 消息 (100K-1M 条/天) 24-48 小时
 - 消息接收
 - 消息存储
- 消息 (<100K 条/天) 48-168 小时
 - 消息接收
 - 消息存储

Mnesia 消息

MessageStore 消息

- `status` - 消息接收/存储
- `dest_smsc` - 消息 SMSC 消息
- `expires` - 消息
- `destination_msisdn` - 消息
- `source_msisdn` - 消息

Mnesia 消息

消息 `disc_copies` 消息

- 消息
- 消息
- 消息
- 消息

CDR 消息

`BatchInsertWorker` 消息 CDR 消息 MySQL

```

# config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,          # CDR 1000
  batch_insert_flush_interval_ms: 100   # 1000ms

```

CDR 1000

1000ms

```

batch_insert_batch_size: 200
batch_insert_flush_interval_ms: 200

```

- 3秒内MySQL 200
- CDR 1000ms flush

1000ms

```

batch_insert_batch_size: 100
batch_insert_flush_interval_ms: 100

```

- 1000ms flush
- CDR 100ms flush

CDR 100

```

batch_insert_batch_size: 20
batch_insert_flush_interval_ms: 20

```

- CDR 100ms flush
- MySQL 20ms flush

□□□□

□□□□ **Mnesia** □□

□□□□□□□□□□□□□□

```
# □□□□□□□□□□  
MessageStore.list(status: :pending)  
MessageStore.list(dest_smsc: "gateway-1")  
Messaging.get_messages_for_smsc("gateway-1")  
  
# □□□□□□□□□□  
MessageStore.list(limit: :infinity) # □□□□□□
```

MySQL □□□

□□ CDR □□□□□□□□ MySQL □□□□

```
# config/runtime.exs  
config :sms_c, SmsC.Repo,  
  pool_size: 10 # □□□□ CDR □□□□
```

□□□

- □□□□ pool_size: 10
- □□ CDR □□□ pool_size: 20-30
- □□□□ pool_size: 5

□□□□

□□□□□□□

□□□□□□□ Benchee □□□□□□□□□□□□□□

```
# SMS API
mix run benchmarks/raw_sms_bench.exs
```

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

0000

0000

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

0000

- **ips**
- **average**
- **median**
- **99th %** 99 SLA

0000

Intel i7-8650U 8

項目	insert_message (Mnesia)	項目 (MySQL)
行数 (合計)	1,750 行/分	83 行/分
行数 (平均)	1,750 行/分	89 行/分
行数 (最大)	0.58 行	16 行
行数 (p99)	<5 行	30 行
行数 (最大)	62 KB	121 KB
行数	約 21 行	-

原因

- 行数が非常に多い
- 行数が非常に多い
- Mnesia が MySQL と I/O
- 50% 程度

原因

原因

原因

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

原因

```
%{
  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
}
```

□□□□□□

1. □□□□□ `current_queue_size` - □□□□□□□□ `batch_size`
2. □□□□□□□□ `last_flush_duration_ms` - □□ `batch_size=100` □ < 100 □□
3. □□□□□□ `flush_errors` - □□ 0 □□□□
4. □□□□□ `total_flushed / uptime` - □□□□□□□□□□

□□

□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□ > 0 □□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□□□□

□□□□□□□□

□□□□□

1. □□□□□□□□□□□□□□□□ `pool_size`
2. □□□□□□□□□□□□□□□□□□

3. `batch_insert_batch_size`
4. `batch_insert_batch_size`

□□□□□□

□□□□□

1. `batch_insert_flush_interval_ms`
2. `batch_insert_batch_size`
3. I/O□□□□
4. API□□□□□□□□

□□□□□□□

□□□□□

1. □□□□□□□□□□□□□□□□
2. `batch_insert_batch_size`
3. `flush_errors`
4. `Supervisor.terminate_child/2` □□□

□□□□□

1. 100/100ms□□□□□□□□□□□□□□□□
2. 1 □□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□□□□□□□
4. □□□□□□□□□□□□
5. □□□□□□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□
7. □□□□ - □□□□□□□□□□

□□□□

□□□□□□□□

```
# 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
```

□□□□□□□□□□

```
# 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
```

□□□□□/□□

```
# 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
```

□□□□□

- Ecto □□□□

- Benchee []
- Phoenix [] [] []

SMS-C

← | [README](#)

简介

SMS-C 是一个基于 Mnesia 实现的 SMS 网关，支持 SMSC、SMS、Mnesia 等。

特性

- 支持 SMSC、SMS、Mnesia 等。
- 支持 **SMSC** 接口。
- 支持 IMS、SMPP 等。
- 支持 Mnesia 数据库。
- 支持...
- 支持...
- 支持...
- 支持...
- 支持...
- 支持 `runtime.exs` 脚本。
- 支持...
- **Web UI** 支持 CRUD 操作。
- 支持...
- 支持...
- **ENUM** 支持 DNS 查询。

安装

安装

...

Field	Type	Description	Default
<code>route_id</code>	integer	Route ID	0
<code>calling_prefix</code>	string/nil	Calling prefix, nil = empty	
<code>called_prefix</code>	string/nil	Called prefix, nil = empty	
<code>source_smsc</code>	string/nil	Source SMSC, nil = empty	
<code>dest_smsc</code>	string/nil	Destination SMSC, auto_reply, drop, true	0
<code>source_type</code>	atom/nil	Source type: :ims, :circuit_switched, :smpp, nil	
<code>enum_domain</code>	string/nil	ENUM domain	
<code>auto_reply</code>	boolean	Auto reply flag	0
<code>auto_reply_message</code>	string/nil	Auto reply message, auto_reply true	0
<code>drop</code>	boolean	Drop flag	0
<code>charged</code>	atom	Charged status: :yes, :no, :default	0
<code>on_net_only</code>	boolean	On net only flag, Diameter/HSS	0
<code>weight</code>	integer	Weight, 1-100, 100	
<code>priority</code>	integer	Priority, 1-255	
<code>description</code>	string	Description	

enabled	boolean	/	

1. auto_reply=false drop=false dest_smsc
2. auto_reply=true auto_reply_message
3. drop=true

on_net_only true HSS Diameter Sh

source_smsc

-
- **SMPP**
-

- **HSS 2001** dest_smsc
- **HSS 5001**
- **HSS**
- **Diameter**



1

1. MSISDN
- 2.

3. 3. 3. 3. 3. 3. 3. 3. 3. 3.

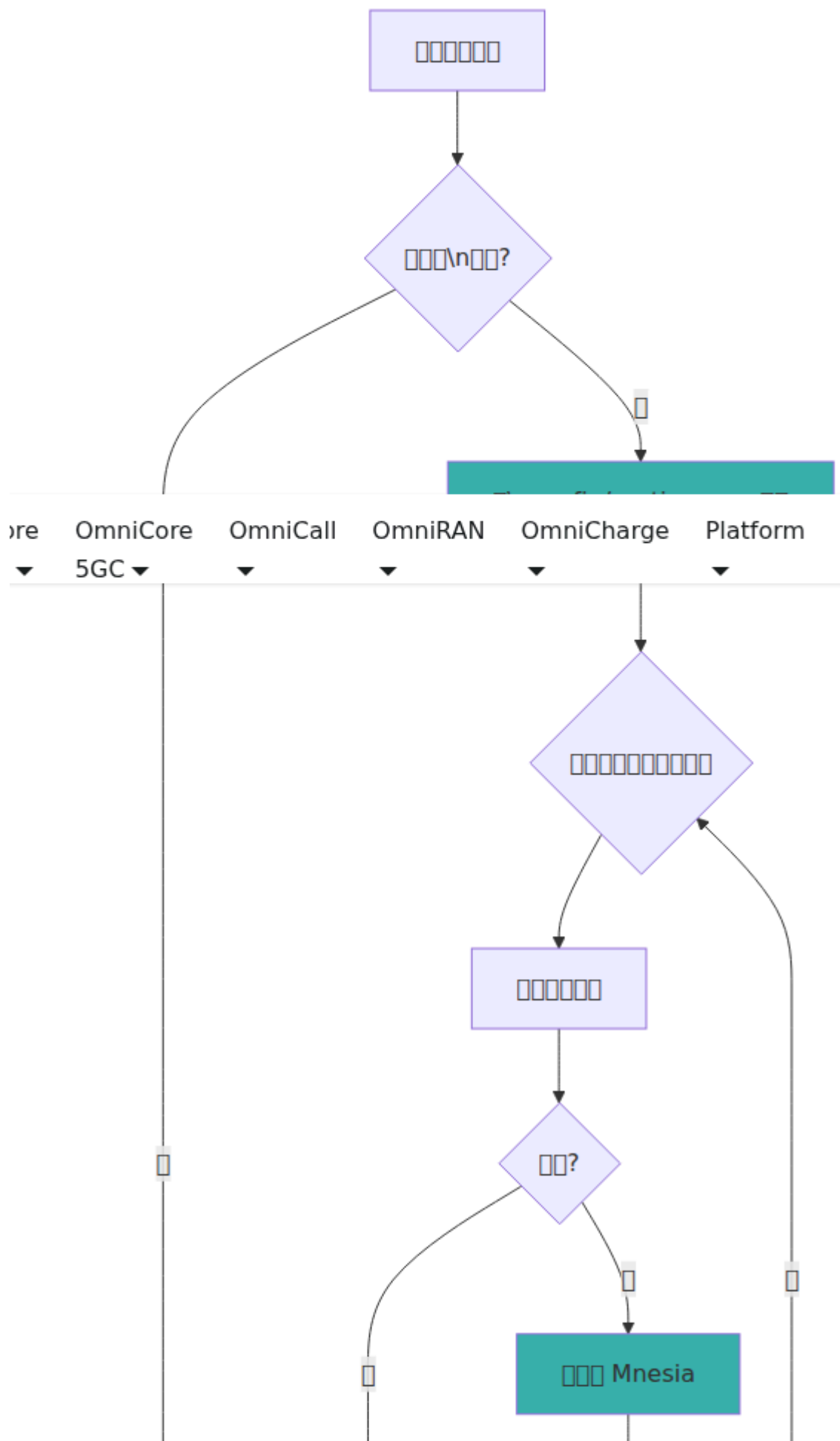
3. 2 HSS Diameter Sh

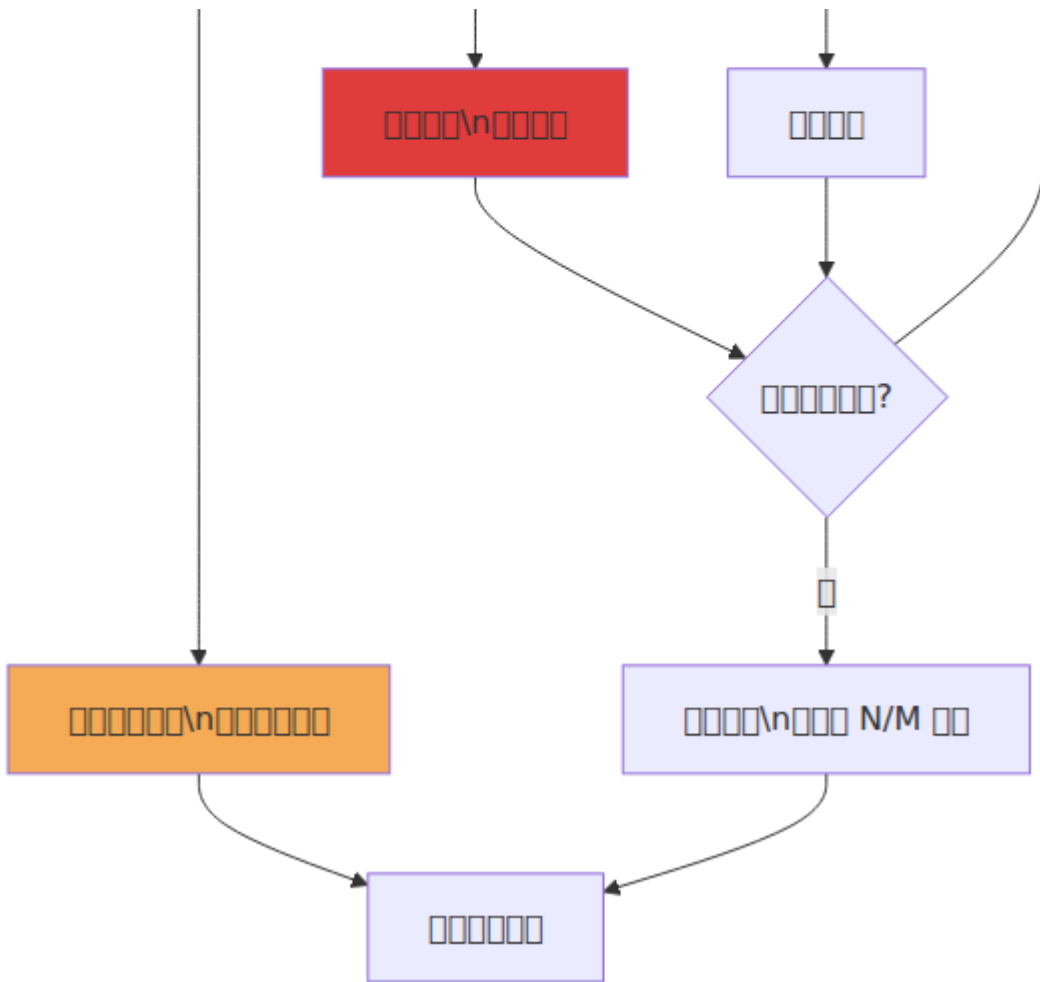
1. Diameter Sh HSS MSISDN
2. 2001
3. 5001
4. HSS
5. HSS

3. 3 Diameter

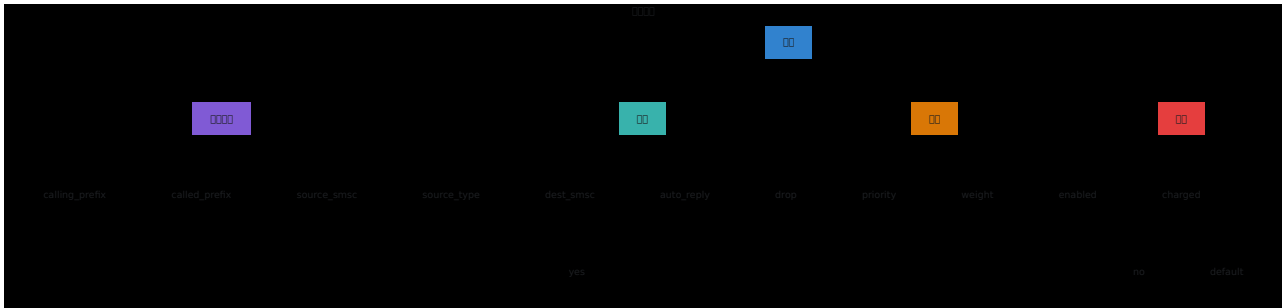
- 1.
2.
 - o
 - o
 - o SMSC = +25
 - o ENUM = +15
 - o = +10
 - o ENUM = +5
3. =
- 4.
5.
 - o HSS HSS
 - o SMSC
 - o
 - o

- nil
-





源短信中心

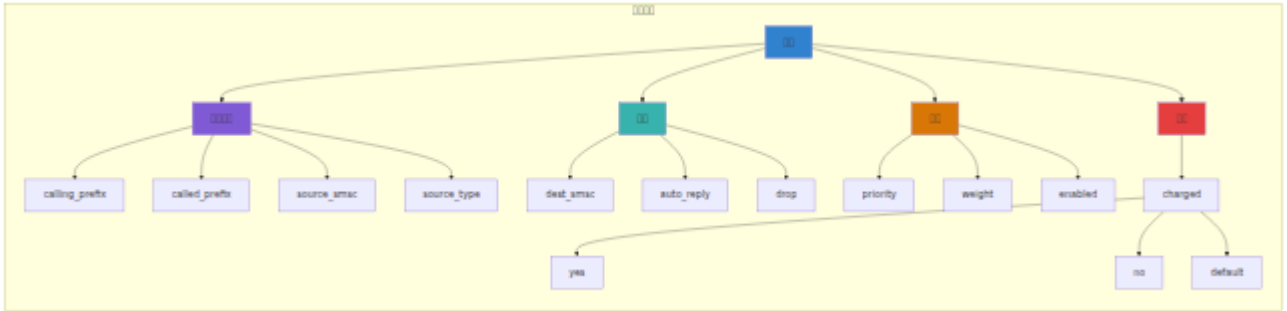


配置 `config/runtime.exe` 和 `config/sms_routes.example.exe` 源短信中心

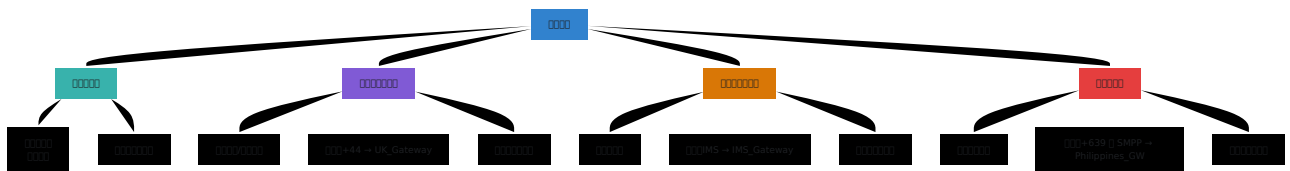
- 源短信中心
- 被叫前缀
- 源短信中心
- 源短信中心
- 源短信中心
- 源短信中心

□□

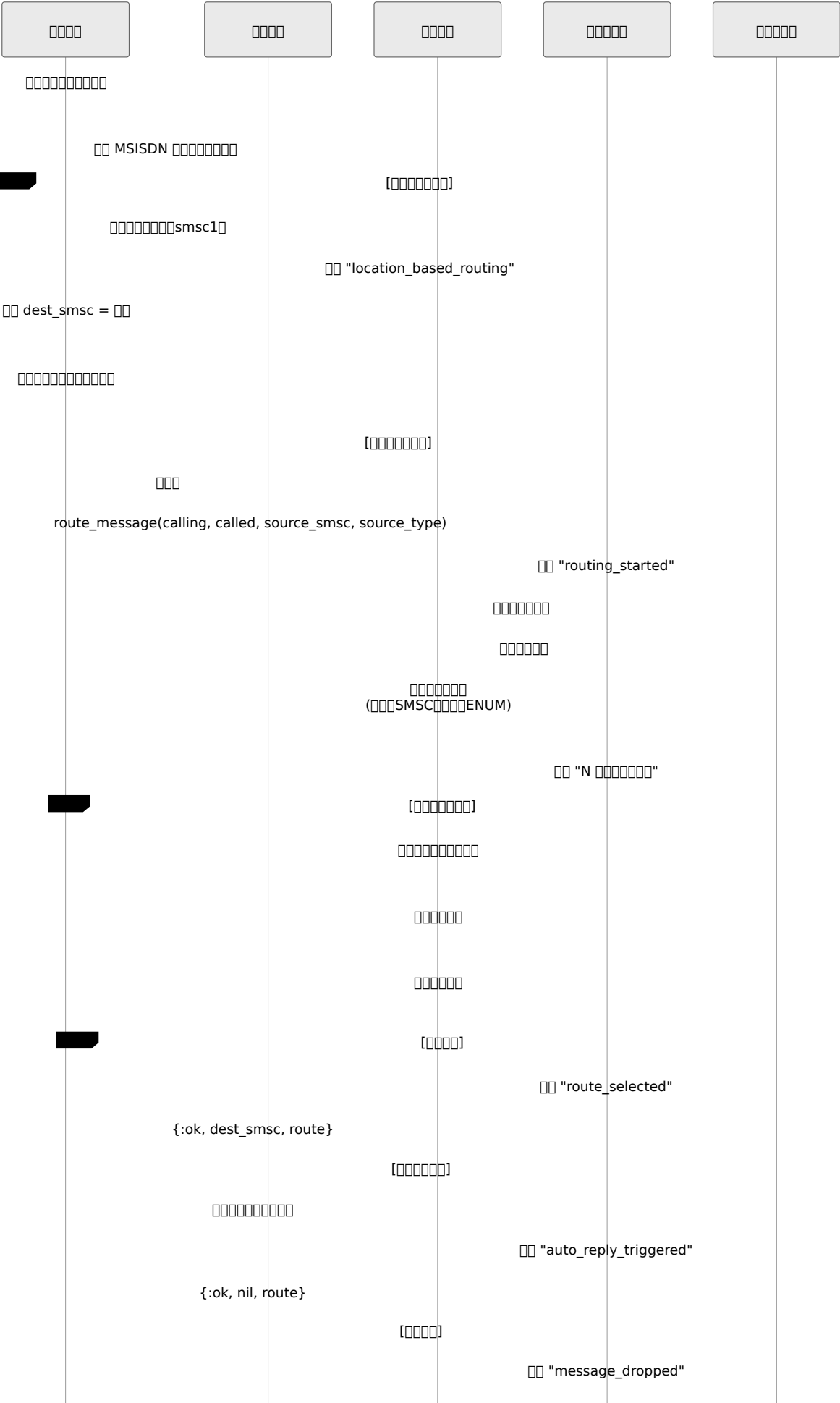
□□□□□□

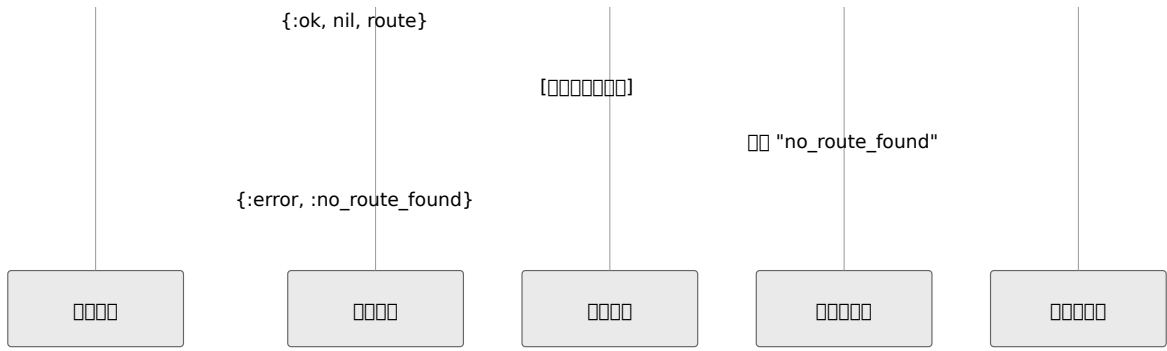


□□□□□□



□□□□□





□□□□

□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□

Parse error on line 4: ... -->|□□◆◆□□
□□ "ims-core-1"| REG[□□□ □-----
 -----^ Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND',
 'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND_STOP', 'TAGEND',
 'TRAPEND', 'INVTRAPEND', 'UNICODE_TEXT', 'TEXT', 'TAGSTART', got 'STR'

□□

□□□□□

1. □□□□□□□□□□
2. □□□□□□□□□□
3. □□□□□□□□□□ MSISDN □□□□□□□□
4. □□□□□□□□□□□□□□□□□□□□
5. □□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□

□□□

- □□□□□□□□□□□□□□□□
- □□□□□□ - □□□□□□□□□□
- □□□□□□ - □□□□□□□□□□
- □□□□□□□□□□ - □□□□□□□□□□

□□□

- IMS Core IMS/VoLTE Core
- MSC Core
- SIP Core

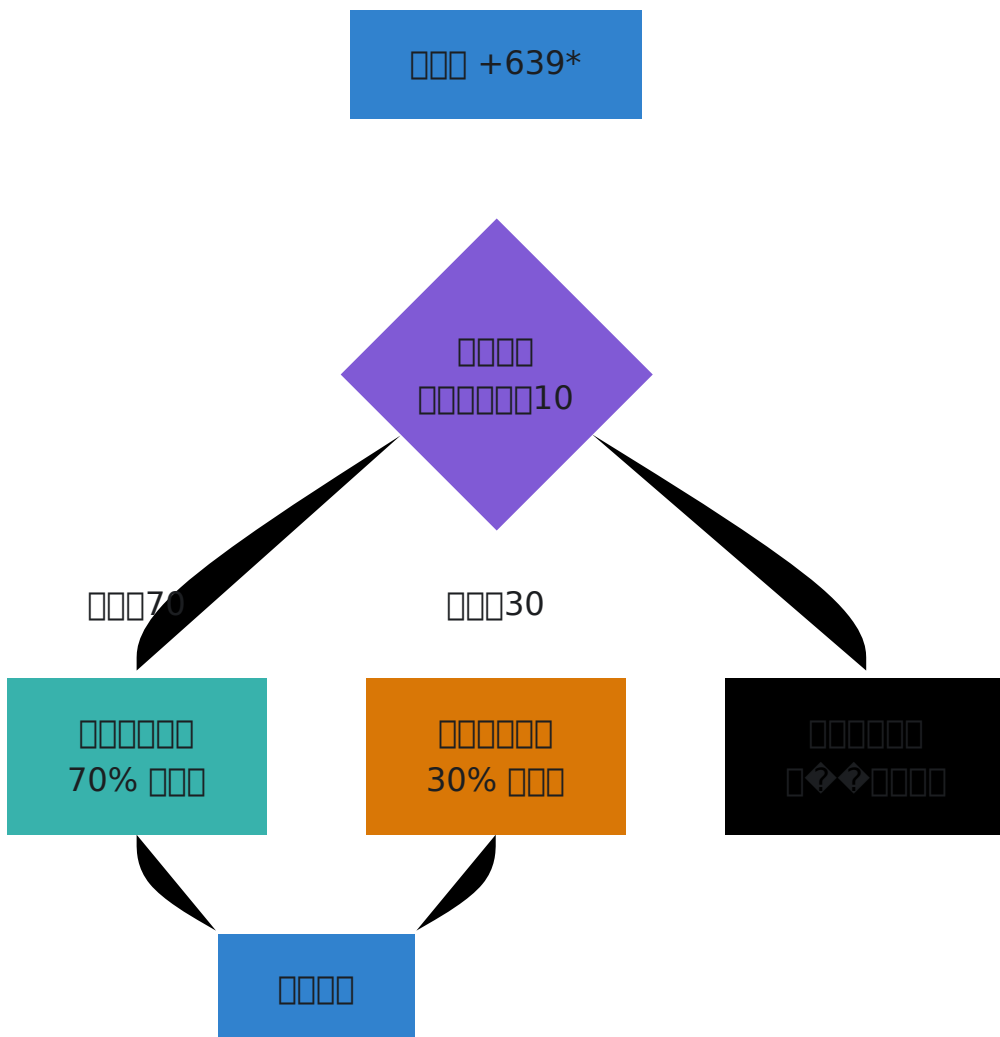
Core

SMSC Core



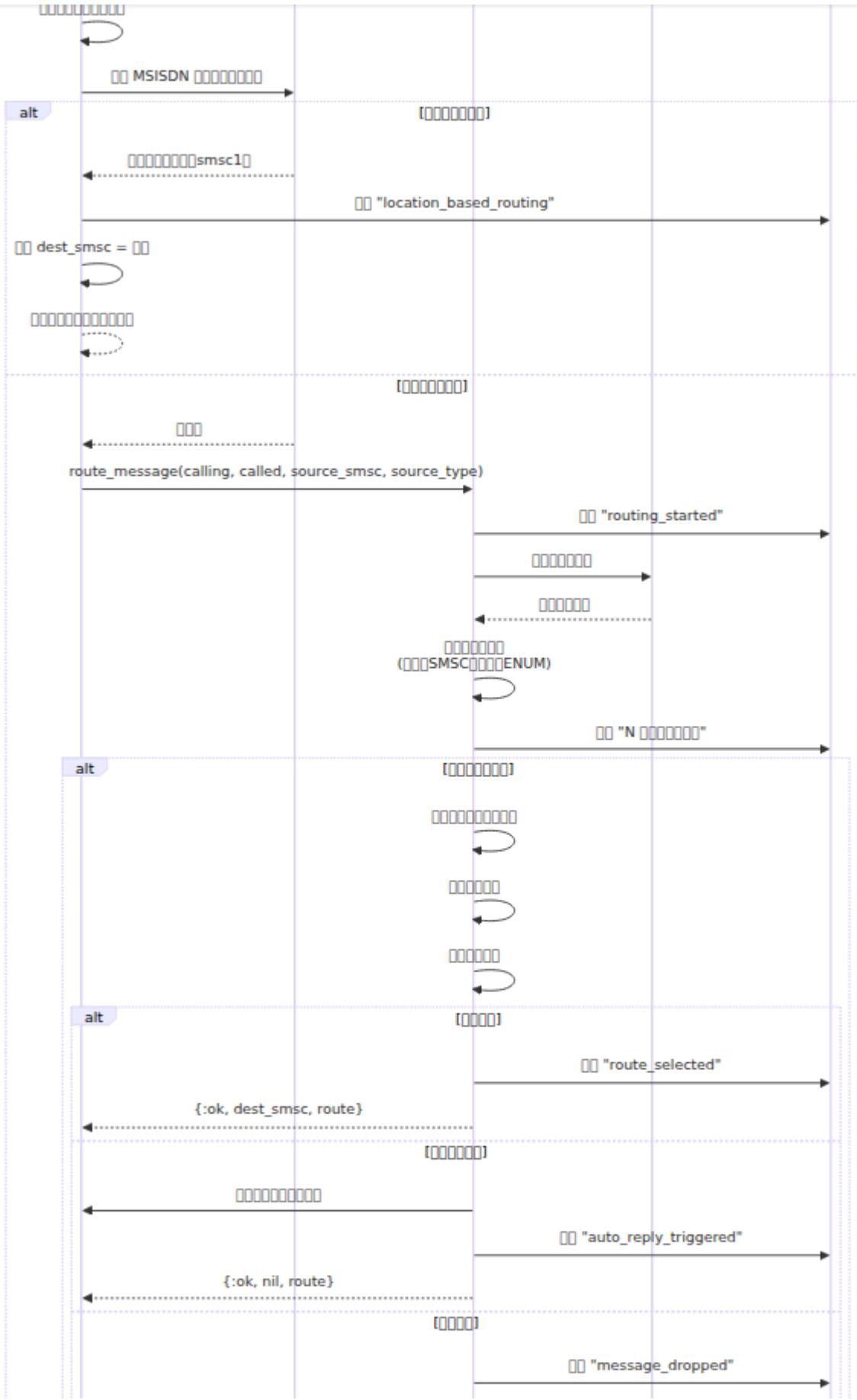
Core

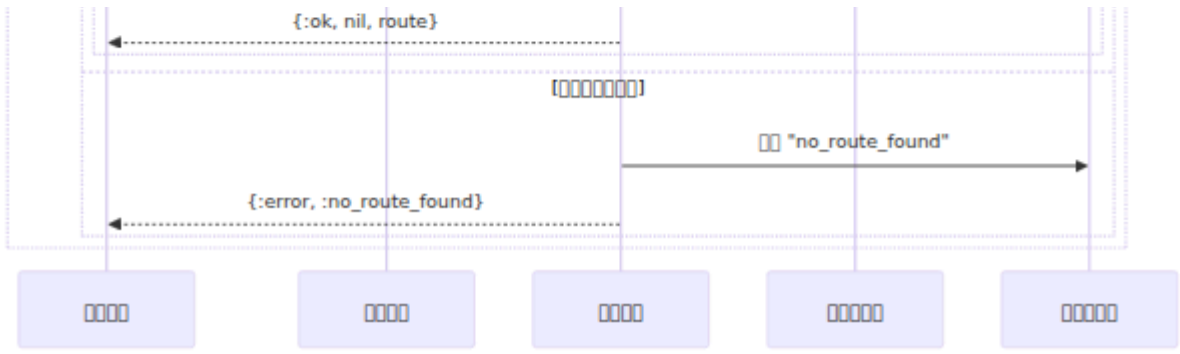
SMSC Core



□□□□□

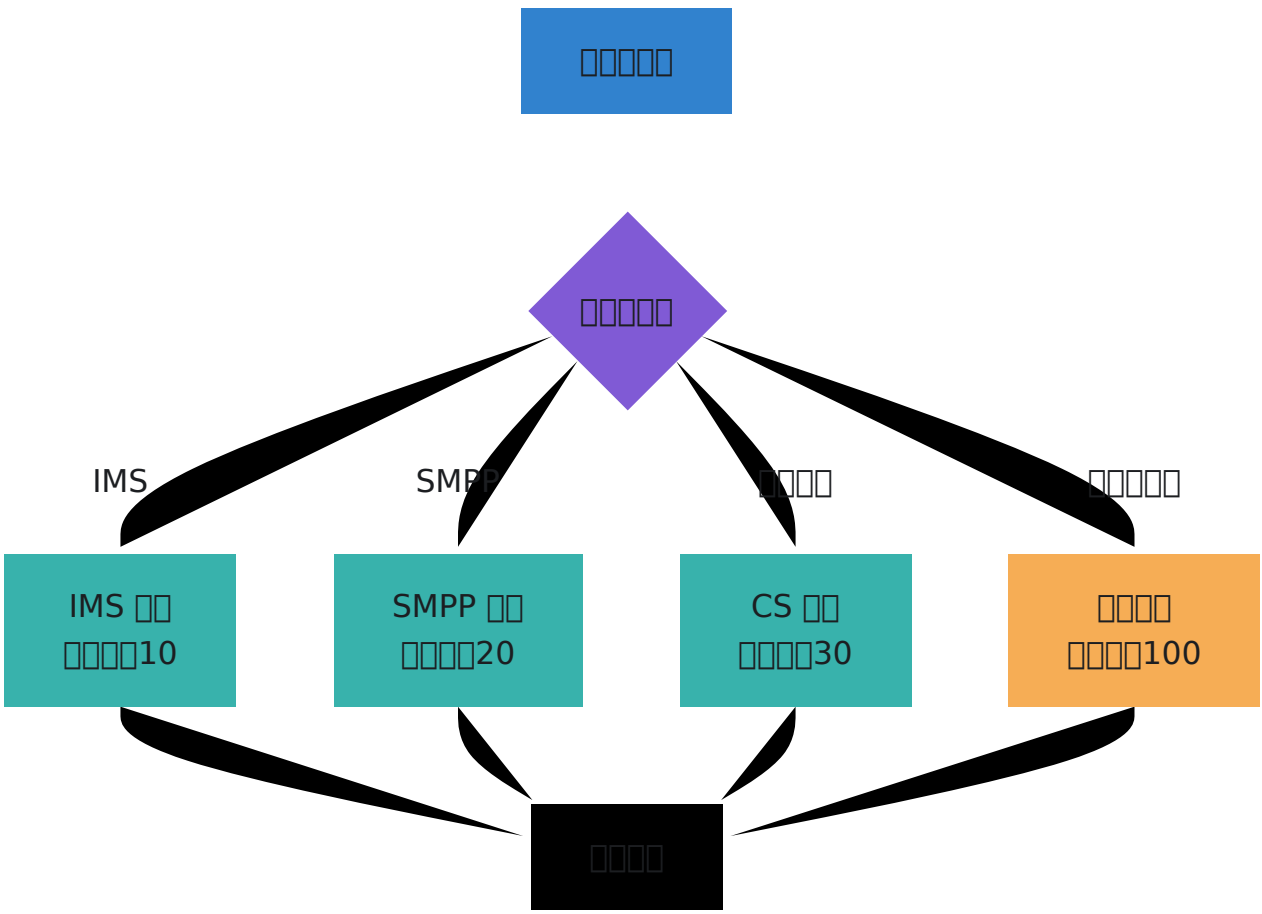
□□□□□□□□□□□□□□□□





□□□□□□□□

□□□□□□□□□□□□



□□□□□

□□□□□□□□□□□□□□□□□□□□

☐☐☐ +639*

☐☐☐☐☐☐☐☐☐

☐☐☐ ☐

☐☐☐☐ 6391 ☐☐☐☐

☐☐☐☐☐☐☐☐☐☐
☐☐☐☐☐☐
• ☐☐☐ → ☐☐☐ 1
• ☐☐☐☐ → ☐☐☐ 50

☐☐☐☐☐☐☐☐☐☐

☐☐☐ 50

☐☐☐☐ 639 ☐☐☐☐

☐☐☐ SMSC
☐☐☐☐

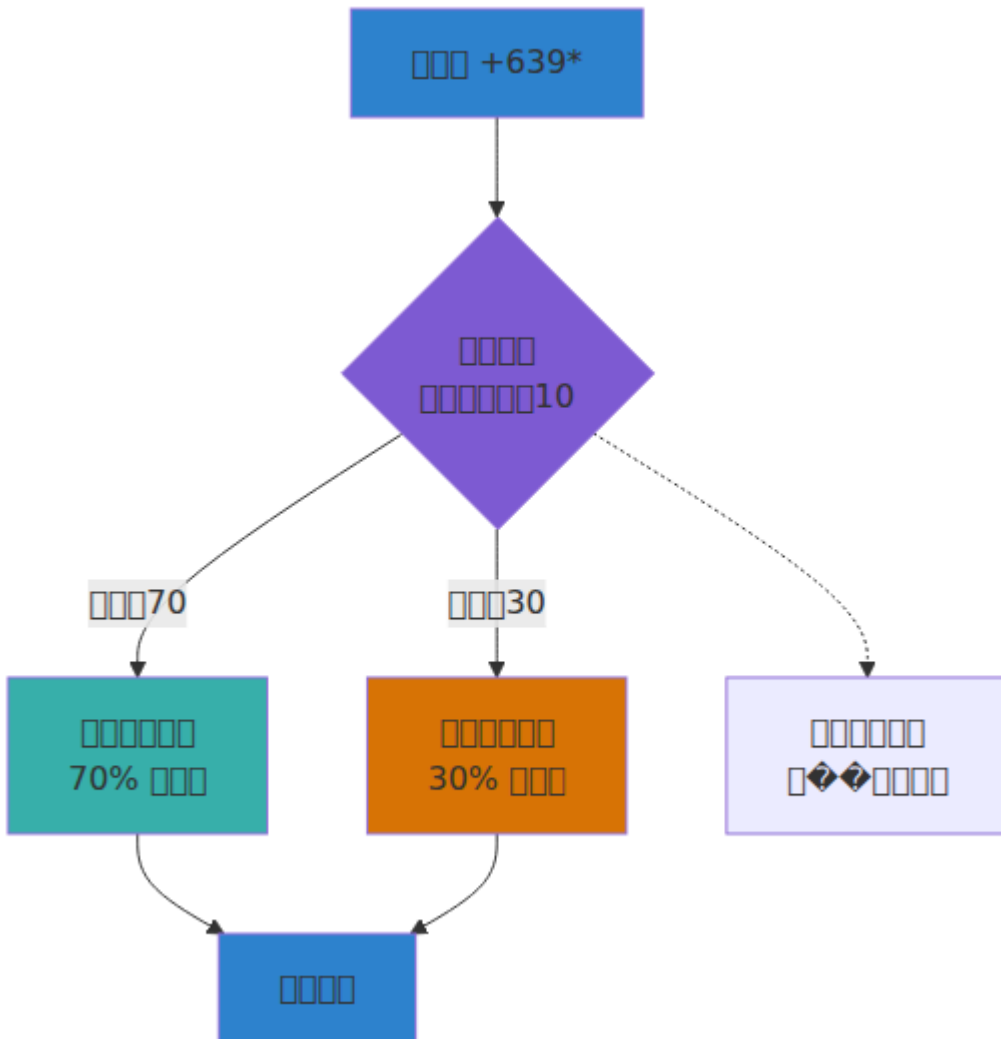
☐☐☐☐ SMSC
☐☐☐☐

☐☐☐☐☐☐

☐☐☐☐☐☐

□□□□□□□□

□□□□□□□□□□□□□□□□



Web □□

□□□□ UI

□□□□□□□□□□ /sms_routing □□□□□□□□□□

□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□
- □□□□□□

- 00/0000000000
- 00000000
- 0000000 5 0000

000000

1. 00 "000000"
2. 000000000000 SMSC 000000
3. 0000001-1000000 10000000001-2550000 1000
4. 00 "00" 000000
5. 00 "000000"

000000

1. 0000000 "00"
2. 0000000000
3. 00 "000000"

000000

- 00 "00" 0000000000
- 00 "00" 000000

000000

00000000 /simulator 0000000000

000

- 00000000000000
- 00000000000000000000000000000000
- 000000000000000000
- 00/000000000000
- 00000000000000
- 000000000000 10 0000

000000

1. 消息类型

- 消息类型
- 消息内容
- 消息 SMSC 地址
- 消息类型/IMS/消息/SMPP

2. 消息 "消息"

3. 消息内容

- 消息内容 "消息"
- 消息内容
 - ✓ 消息 = 消息
 - ✗ 消息 X = 消息
 - 消息/消息
- 消息
 - 消息 + "SELECTED" 消息 = 消息
 - 消息 + "MATCHED" 消息 = 消息
 - 消息 = 消息

4. 消息内容

5. 消息内容

消息内容 消息内容

- 消息 "消息 '1234'" 消息 "消息 '44' 消息"
- 消息 "消息内容" 消息 "消息 '639' 消息"
- 消息 **SMSC** "消息 'smc1'" 消息 "消息 'untrusted_smc' 消息 'none'"
- 消息 "消息内容" 消息 "消息 'smpp' 消息 'IMS'"

API

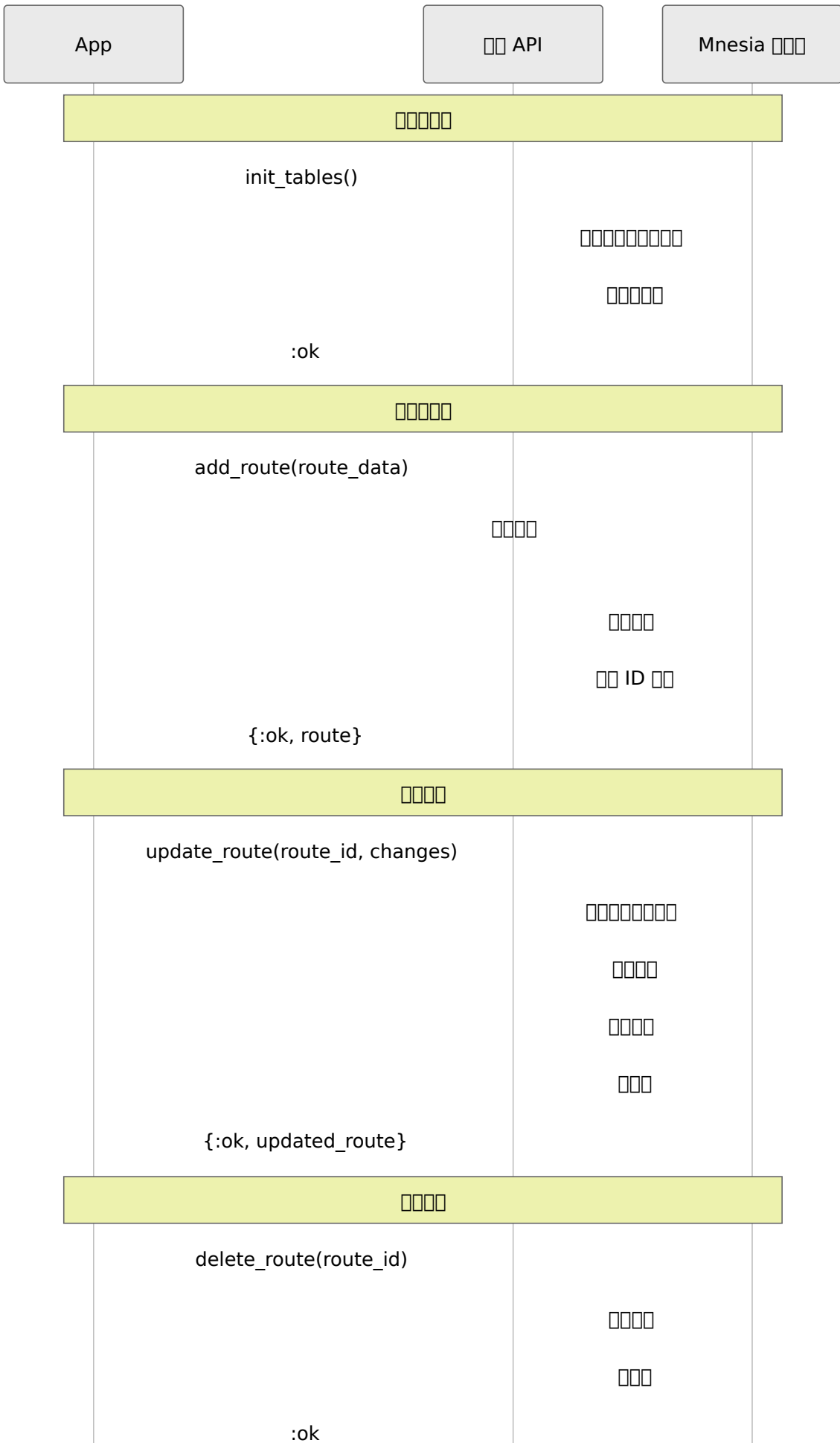
□□□□□□

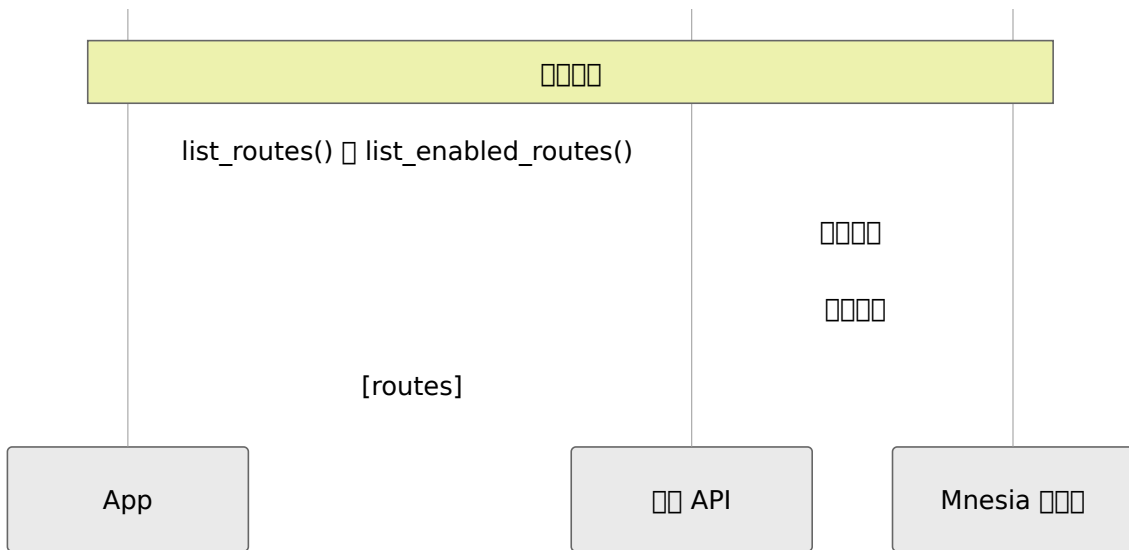
```
export_routes [POST]
import_routes [POST]
merge [POST]
replace [POST]
```

```
calling_number [POST]
called_number [POST]
source_area [POST]
source_type [POST]
message_id [POST]
```

```
init_tables [POST]
add_route [POST]
update_route [POST]
delete_route [POST]
get_route [POST]
list_routes [POST]
not_enabled_routes [POST]
```

□□□□□





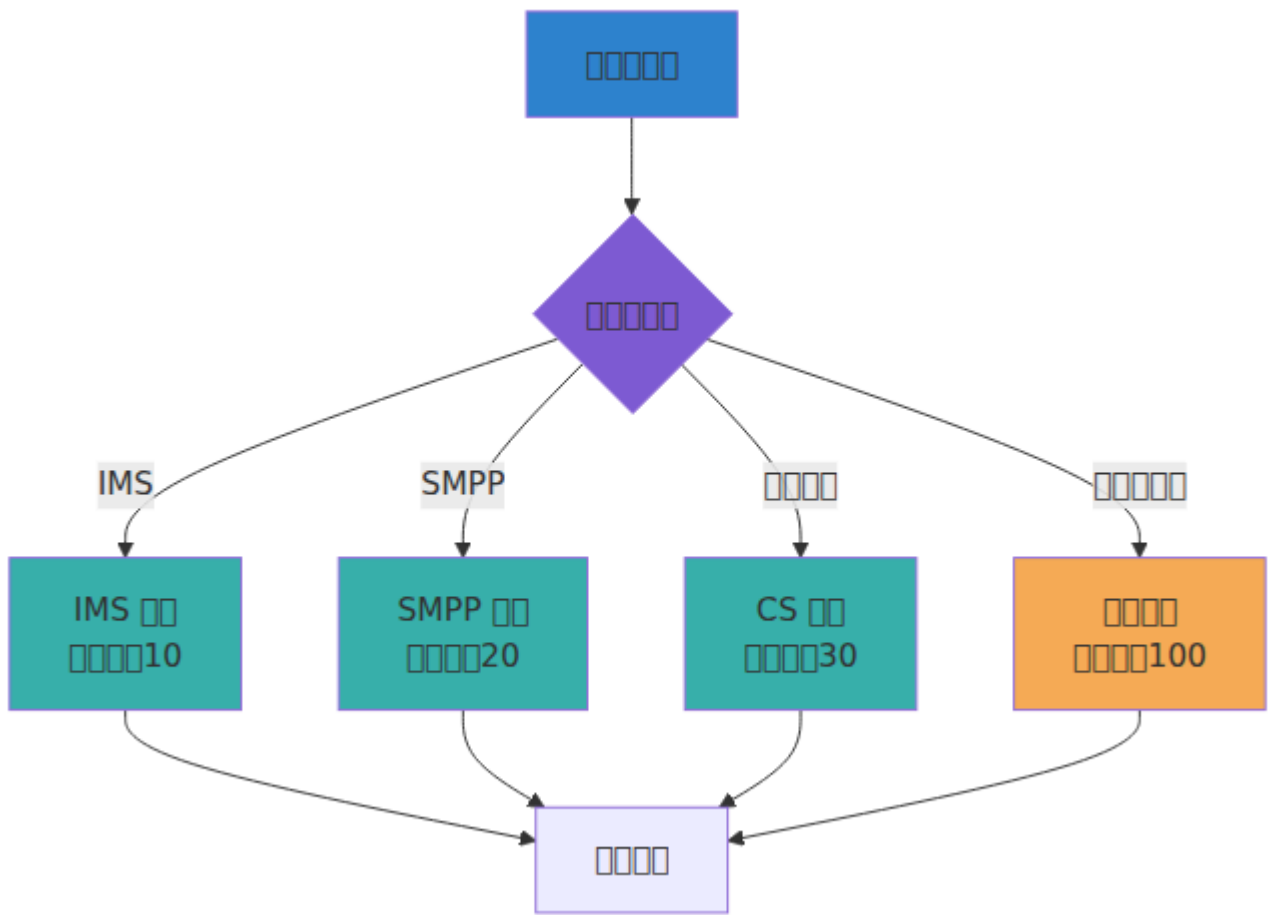
[] [] [] [] []

route_message [] [] [] [] [] []

- calling_number [] [] [] [] [] [] [] [] [] []
- called_number [] [] [] [] [] [] [] [] [] []
- source_smsc [] [] [] [] [] SMSC [] [] []
- source_type [] [] [] [] [] [] [] [] :ims [] :circuit_switched [] :smpp []
- message_id [] [] [] [] [] [] [] [] [] [] [] []

[] [] []

- {:ok, dest_smsc, route} - [] [] [] [] [] [] [] []
- {:error, :no_route_found} - [] [] [] [] [] [] [] []



CS/IMS



CS

CS

1. CS 1-10
2. CS
3. CS
4. CS

□□

1. □□□□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□□□

□□

1. □□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□
4. □□□□□□□□□□□□□□□□□□

□□

1. □□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□

□□□□

□□□□□

□□□□□ `{:error, :no_route_found}`

□□□□□

- □□□□□□
- □□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□

□□□□□

1. □□□□□□□□□□ `SmsRouting.list_enabled_routes()`

2. 000000000000000000

3. 000000000000000000 add_route(%{dest_smsc: "debug_smsc", priority: 255})

4. 000000000000000000

00000000

00000000000000

00000

- 00000000
- 0000000000000000
- 00000000000000
- 0000000000000000

00000

1. 0000000000000000
2. 000000000000000000
3. 00000000000000
4. 0000000000000000

00000

00000000

00000

- 00000000000000
- 00000000
- Mnesia 00000000

00000

1. 00000000
2. 00000000000000

3. Mnesia `init_tables`
- 4.

ENUM/NAPTR

ENUM E.164 DNS NAPTR SMS-C ENUM DNS ENUM

ENUM

ENUM E.164 DNS

- +1-212-555-1234
- **ENUM** 4.3.2.1.5.5.2.1.2.1.e164.arpa
- **DNS** NAPTR
- SIP URI

ENUM `config/runtime.exs`

ENUM

`enum_enabled: true` ENUM DNS ENUM

ENUM

ENUM

ENUM

- `e164.arpa` - IETF ENUM
- `e164.org` - ENUM
- ENUM

DNS 枚举

枚举 DNS 记录 {ip_address, port}

枚举 DNS 记录

枚举 DNS 记录

- Google DNS {"8.8.8.8", 53} {"8.8.4.4", 53}
- Cloudflare DNS {"1.1.1.1", 53} {"1.0.0.1", 53}
- 枚举 DNS {"10.0.0.53", 53}

枚举

枚举 DNS 记录 5000 枚举记录

ENUM 枚举

Parse error on line 37: ... style Router fill:#3182CE style C -----^
Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW',
'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW',
'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS',
'SOLID_POINT', 'DOTTED_POINT', got 'TXT'

枚举

ENUM 枚举

枚举 ENUM 记录 15 枚举记录 DNS 记录

枚举

- 枚举 DNS 记录
- 枚举记录
- 枚举 DNS 记录

枚举

- 枚举 NAPTR 记录

- Prometheus 監視器/代理
- 監視器/代理

代理

- 代理
- 代理
- 代理 15 代理
- 代理 ETS 代理

ENUM

enum_result_domain ENUM 代理

代理

+1-555-0100 ENUM NAPTR 代理

- E2U+sip
- sip:customer@voip-carrier.com
- voip-carrier.com

代理

enum_result_domain: "voip-carrier.com" ENUM 代理

代理

- enum_result_domain: nil - 代理
- enum_result_domain: "specific.com" - ENUM 代理
- ENUM 代理

代理

ENUM 代理 +15 代理

ENUM 代理

NAPTR 代理 /naptr_test 代理

ENUM

- ENUM DNS 記錄
- NAPTR 記錄
- NAPTR 記錄
- 記錄
- 記錄

ENUM 記錄

1. 記錄 + 記錄
2. ENUM 記錄 e164.arpa
3. 記錄 "記錄"
4. 記錄
 - 記錄 NAPTR 記錄
 - 記錄
 - 記錄 E2U+sip E2U+tel 記錄
 - 記錄
 - 記錄
 - 記錄

ENUM 記錄

- 記錄 DNS 記錄 "記錄"
- 記錄
- 記錄
- 記錄

記錄

記錄 NAPTR 記錄

- 記錄
- 記錄
- 記錄 u=記錄 s=記錄
- 記錄 E2U+sip E2U+tel 記錄

- ENUM
- SIP
- VoIP

ENUM

1. VoIP

ENUM SIP/VoIP VoIP

- ENUM SIP URI sip:number@voip-carrier.com
- voip-carrier.com
- enum_result_domain: "voip-carrier.com"
- VoIP

2.

- ENUM
- carrier-a.com
- A
-

3.

- ENUM
-
-

4.

ENUM

- ENUM
-
-

5. ENUM

ENUM 911/112 緊急通報サービス

- ENUM サービス
- 緊急通報サービス
- 国際緊急通報サービス

ENUM サービス

ENUM

1. ENUM サービス 1-10

- ENUM サービス
- VoIP サービス
- 国際緊急通報サービス

2. ENUM サービス 50-100

- ENUM サービス
- ENUM サービス
- ENUM サービス

3. ENUM サービス 200+

- ENUM サービス
- ENUM サービス

ENUM

- ENUM 1 `enum_result_domain: "sip.carrier.com"` → VoIP サービス
- ENUM 10 `enum_result_domain: "tel.carrier.com"` → PSTN サービス
- ENUM 50 `called_prefix: "+1"` → 国際緊急通報サービス
- ENUM 100 `called_prefix: "+"` → 国際緊急通報サービス
- ENUM 200 サービス → 国際緊急通報サービス

ENUM

DNS 性能

ENUM 性能 DNS 性能

- 平均 < 1ms
- 90-100ms DNS 性能

原因

- 性能 DNS 性能
- 性能 5000ms
- 性能 > 80%
- 性能

原因

性能

- 性能
- 性能 ETS 性能
- 性能 TTL 性能
- 性能

原因

ENUM 性能

- DNS 性能 → 性能
- NAPTR 性能 → 性能
- NAPTR 性能 → 性能
- DNS 性能 → 性能

ENUM 性能

Prometheus 性能 ENUM 性能

- sms_c_enum_lookup_stop_duration - 性能
- sms_c_enum_cache_hit_count - 性能
- sms_c_enum_cache_miss_count - 性能

- `sms_c_enum_cache_size_size` - 000000
- `sms_c_enum_naptr_records_record_count` - 00000 NAPTR 000

00000

- 000000000000 > 70%
- 000000 **p95** < 1000ms
- 00000000 DNS 00

000 `docs/METRICS.md` 000000000000

0000 **ENUM**

000000 **NAPTR** 00

- 00 ENUM 000
- 00 DNS 00000
- 000000000000 ENUM 000
- 00000 ENUM 00000e164.org
- 00 NAPTR 000000000

0000 **ENUM** 00000

- 00 DNS 00000
- 0000000
- 0000000000
- 0000000000 DNS 000
- 000000000

0000 **ENUM** 0000000000

- 0000000 `enum_result_domain` 00
- 00000000000000000000
- 0000000000000000
- 000000000000 NAPTR 00000

000000 **ENUM** 00

- `enum_enabled: true` `config/runtime.exs`
- `enum_domains` `enum_domains`
- `enum_domains`
- `enum_domains` ENUM `enum_domains`

`enum_domains`

DNS `enum_domains`

- `enum_domains` DNS `enum_domains`
- `enum_domains` DNSSEC
- `enum_domains` NAPTR `enum_domains`
- `enum_domains`

`enum_domains`

- `enum_domains`
- `enum_domains` DNS `enum_domains`
- `enum_domains`

`enum_domains`

- ENUM `enum_domains` DNS `enum_domains`
- `enum_domains` DNS `enum_domains`
- `enum_domains` VPN/`enum_domains` DNS `enum_domains`

`enum_domains`

`enum_domains` EventLogger `enum_domains`

- `sms_routing_started` `enum_domains`
- `sms_routing_candidates` `enum_domains`
- `sms_routing_matches` `enum_domains`
- `sms_routing_selected` `enum_domains`
- `sms_routing_failed` `enum_domains`

`enum_domains` `message_id` `enum_domains` `route_message/1` `enum_domains`

□□

Mnesia □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

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

□□

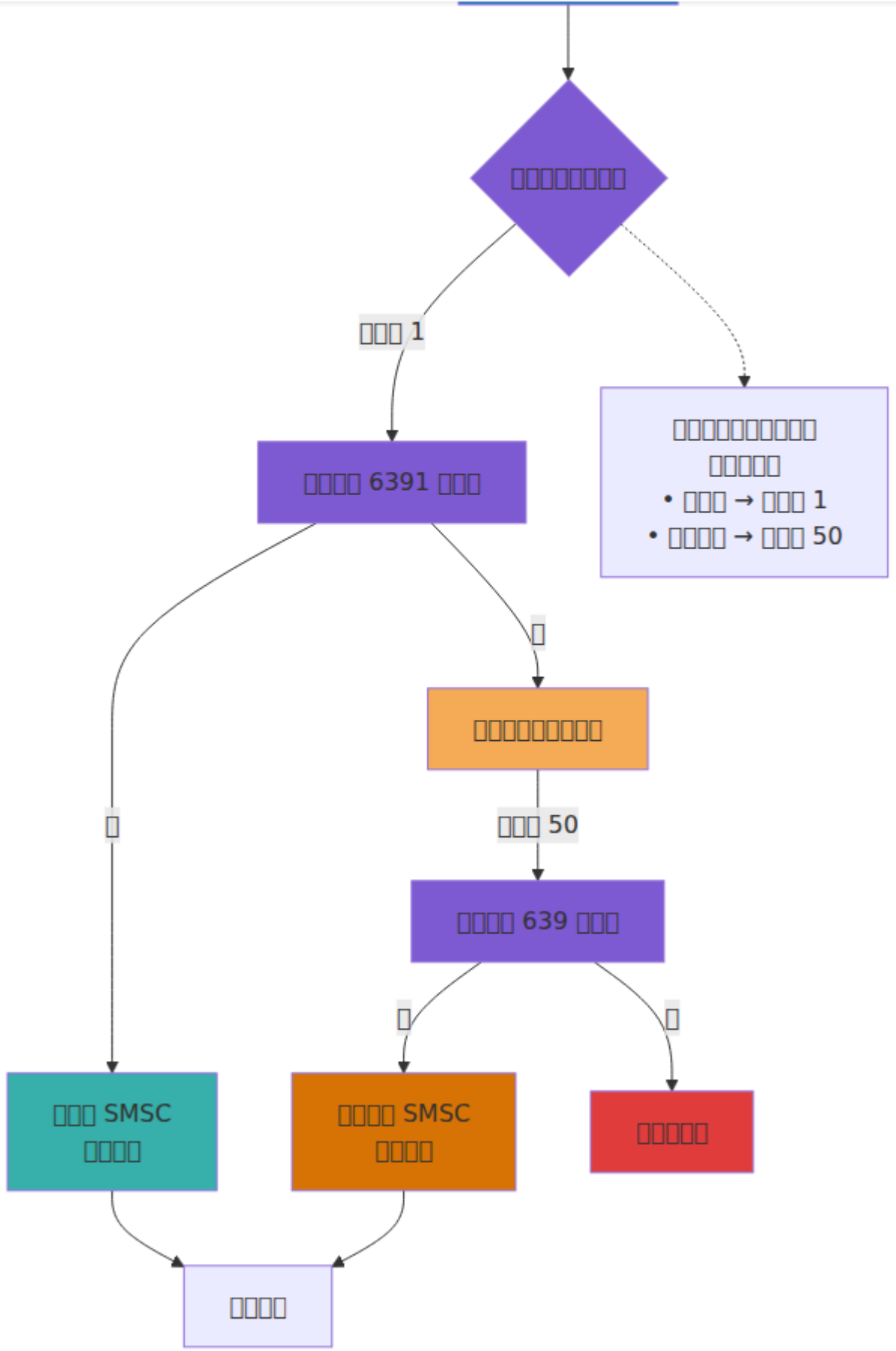
□□

□□□□□□□□ test/sms_c/messaging/sms_routing_test.exs □□□□□□□□□□

- □□□□
- □□□□□□□□
- □□□□□□□□
- □□□□□
- □□□□

□□□ ◊? ◊? ◊? □□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



□□□□□□□□

1. □□□□

- □□ Mnesia □□□
- □□□□□□□□

2. □□□□□

- □□□□ → □□□□□□□□
- □□□□ → □□□□□□□
- □□□□□ → □□□□□□

3. □□□□

- □□□□□□□□
- □□□□□□□
- □□□□□□□

4. □□□□

- □□□□□□□□
- □□ `route_message/1` API
- □□□□□□□

5. □□□□□

- □□□□□□□□
- □□□□□
- □□□□□□□□□□□□□□

6. □□

- □□□□□□□□
- □□□□□□□
- □□□□□

□□

□□□□□□□□

- □□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□□□□□
- □□ Mnesia □□□□ `:mnesia.table_info(:sms_route, :size)`

□□□□

□□□□□□

```
# 1. □□ API □□  
curl https://api.example.com:8443/api/status  
  
# 2. □□ Prometheus □□□□  
curl https://api.example.com:9568/metrics | grep sms_c  
  
# 3. □□□□□□□□  
tail -f /var/log/sms_c/application.log  
  
# 4. □□□□□□  
systemctl status sms_c  
  
# 5. □□ SQL CDR □□□□□□ (MySQL/MariaDB)  
mysql -u sms_user -p -h db.example.com -e "SELECT 1"  
  
# □□ PostgreSQL:  
# psql -U sms_user -h db.example.com -d sms_c_prod -c "SELECT 1"
```

□□□□

□□□□□□□□

```
# □□ 100 □□□□□□□□□□  
tail -1000 /var/log/sms_c/application.log | grep "\[error\  
  
# □□□□□□□□  
grep "routing_failed" /var/log/sms_c/application.log  
  
# □□ SQL □□□□□  
grep -i "database\|sql\|ecto" /var/log/sms_c/application.log |  
grep error
```

□□□□□□□□

```
# 实时监控
tail -f /var/log/sms_c/application.log | grep -E "
(error|warning|critical)"
```

实时监控

实时监控

```
# 接收消息速率
rate(sms_c_message_received_count[5m])

# 消息投递成功率
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])
```

实时监控

```
# 待处理消息队列大小
sms_c_queue_size_pending

# 队列中最老消息的年龄(秒)
sms_c_queue_oldest_message_age_seconds
```

实时监控

```
# 消息处理停止时长 (p95)
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)

# 消息路由停止时长 (p95)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
```

□□□□□□

□□□□□□

□□□

- □□□□□◀◀□□□”□□
- □□□□□□□□
- □□□□□□

□□□□□

1. □□□□□□□

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

□□□□□□□□

□□□□□□□□□□□□□□

2. □□□□□□□

□□ Web UI: `/message_queue`

- □□□□□□“□□□”
- □□ `dest_smsc` □
- □□ `deliver_after` □□□□

3. □□□□□

□□ Web UI: `/simulator`

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□

4. □□□□□□□

□□□□□□□□

- □□□□□□□ `/api/messages` □

- `dest_smsc` 欄

確認

確認

```
# 確認
systemctl status frontend_service

# API
curl -k https://api.example.com:8443/api/status

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

SMSC

- 確認
- 確認
- 確認
- `dest_smsc` 欄

確認

- `deliver_after` 欄
- 確認

```
curl -X PATCH https://api.example.com:8443/api/messages/12345 \
-H "Content-Type: application/json" \
-d '{"deliver_after": "2025-10-30T12:00:00Z"}'
```

□□□□□□

□□□

- `delivery_attempts` □□□□
- □□□□□□□□ > 3□
- □□□□□□

□□□□□

1. □□□□□□□

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

□□□

- □□□□□□
- □□□□
- □□□□□

2. □□□□□□□

- □□□□□□□□□□
- □□□□□
- □□□□□
- □□□□□□□□

□□□□□

□□□□□□□

- □□□□□□□□
- □□□□□□

□□□□□


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

# PATCH
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"}'
```

Queue

- Queue
- Queue
- Queue

Queue

Queue

- Queue `deadletter: true`
- Queue
- Queue "Queue"

Queue

1. Queue

Queue Web UI: `/message_queue`

- Queue
- Queue

2. Queue

- Queue
- Queue
- Queue

□□□□

□□□□□□

```
# □□□□□□ 24 □□  
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}'
```

□□□□

□□□□□

□□□

- □□□ no_route_found
- sms_c_routing_failed_count □□□□
- □□□□□□ "routing_failed"

□□□□□

1. □□□□□□□□□□

□□ Web UI: /sms_routing

- □□□□□□□□□□
- □□□□□□□□□□□□

2. □□□□□

□□ Web UI: /simulator

- □□□□□□□□□□□□□□□□ SMSC□
- □□□□□□
- □□□□□□□□□□

3. □□□□□□□

- `android.permission.SEND_SMS`
- `android.permission.RECEIVE_SMS`
- `android.permission.READ_SMS`

`android.permission.SEND_SMS`

`android.permission.RECEIVE_SMS`

`android.permission.READ_SMS`

```
android.permission.SEND_SMS: (S)
android.permission.RECEIVE_SMS: (S)
android.permission.SEND_SMS: (S)
android.permission.SEND_SMS: default_gateway
android.permission.SEND_SMS: 255
android.permission.SEND_SMS: 100
android.permission.SEND_SMS: ✓
android.permission.SEND_SMS: android.permission.SEND_SMS
```

`android.permission.SEND_SMS`

`android.permission.SEND_SMS`

```
android.permission.SEND_SMS: +
android.permission.SEND_SMS: international_gateway
android.permission.SEND_SMS: 200
android.permission.SEND_SMS: 100
android.permission.SEND_SMS: ✓
android.permission.SEND_SMS: android.permission.SEND_SMS
```

`android.permission.SEND_SMS`

- `android.permission.SEND_SMS` Web UI
- `android.permission.SEND_SMS`

`android.permission.SEND_SMS`

`android.permission.SEND_SMS`

- 000000000000
- 0000000000
- 000000000000

000000

1. 0000000000

Web UI: `/simulator`

- 00000000000000
- 00“0000”00
- 00000000000000

2. 0000000000

- 0000 = 000000
- 000000000000
- 00000000000000

3. 0000000000

000000

- 00000000000000 +100 0
- 00000000000000 +50 0
- 000 SMSC+25 0
- 000000+10 0
- 00 ENUM 00+15 0

000000

000000

000000000000

□□□□:

□□□□: +1555

□□□: 10 (□□□□)

□□□□:

□□□□: +1

□□□: 50 (□□□□)

□□□□□

□□□□□□□□□□

□□□ (70%):

□□: 70

□□ (30%):

□□: 30

□□□□□□□□□□

□□□□□□□□□□□□

□□□□:

□□□□: +15551234

□□ SMSC: dedicated_gateway

□□□: 1

□□□□:

□□□□: +1

□□ SMSC: general_gateway

□□□: 50

□□□□□□□□

□□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□

- 自動返信メッセージ

確認

1. 確認

- `auto_reply: true`
- `auto_reply_message` 設定
- 有効
- 自動返信メッセージ

2. 確認

- 自動返信メッセージ
- 自動返信メッセージ“auto_reply”

3. 確認

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

確認

確認

- 自動返信メッセージ
- 自動返信メッセージ
- 有効

確認

確認

```
ステータス: ✓  
自動返信メッセージ: "自動返信メッセージ"
```

確認

□□□□□□□□□□□□□□□□□□

```
□□□□□□:
  □□□: 10

□□□□□:
  □□□: 50
```

□□□□

□□□□□□□

□□□

- sms_c_message_processing_stop_duration p95 > 1000ms
- API □□□□
- □□□□

□□□□□

1. □□□□□□□

```
# □□□□
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)

# ENUM □□□□
histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)

# □□□□
histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)

# □□□□
histogram_quantile(0.95, sms_c_delivery_succeeded_duration_bucket)
```

2. □□□□□□□

```
# CPU 使用率  
top -b -n 1 | grep sms_c
```

```
# 実行中のプロセス  
ps aux | grep beam.smp
```

確認

確認

- 確認
- 確認
- 確認

ENUM 確認

- DNS 確認
- 確認
- 確認/DNS 確認
- ENUM 確認

確認

- OCS 確認
- OCS 確認
- 確認
- 確認

確認

- 確認
- 確認
- 確認
- 確認

確認


```
# config/config.exs
# 配置短信发送
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# 配置数据库
config :sms_c, SmsC.Repo,
  pool_size: 50
```

配置

配置

- 配置 < 100 msg/sec
- 配置 API 配置
- 配置 API 配置

配置

1. 配置

```
# 配置 (iex)
SmsC.Messaging.BatchInsertWorker.stats()
```

配置

- `current_queue_size` 配置
- `flush_errors` > 0
- `last_flush_duration_ms` 配置

2. 配置

```
# 数据库连接池
ecto_pools_query_time

# 数据库队列
ecto_pools_queue_time
```

数据库

数据库

数据库

```
config :sms_c, SmsC.Repo,
  pool_size: 50 # 20 个
```

数据库

数据库

```
config :sms_c,
  batch_insert_batch_size: 200, # 数据库
  batch_insert_flush_interval_ms: 200 # 数据库
```

数据库

```
# 数据库 /create_async
curl -X POST
https://api.example.com:8443/api/messages/create_async

# NOT: /api/messages (数据库)
```

数据库

数据库

- sms_c_queue_size_pending 数据库
- 数据库

- 接收消息的速率

接收消息

1. 接收消息的速率

```
# 接收消息的速率
rate(sms_c_message_received_count[5m])

# 消息成功送达的速率
rate(sms_c_delivery_succeeded_count[5m])
```

2. 消息成功送达的速率

- 接收消息的速率
- 消息成功送达的速率
- 消息尝试发送的速率

3. 消息成功送达的比率

```
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_attempted_count[5m])
```

接收消息

接收消息

- 接收消息的速率
- 消息成功送达的速率 5-10 秒
- 消息尝试发送的速率

接收消息

- 接收消息的速率
- 消息成功送达的速率
- 消息尝试发送的速率

接收消息

- 数据库
- 数据库
- 数据库

数据库

- 数据库
- 数据库
- 数据库

数据库

数据库

数据库

- 数据库“数据库”
- API 500 数据库
- 数据库

数据库

1. 数据库 SQL CDR 数据库

```
# MySQL/MariaDB
systemctl status mysql

# PostgreSQL
systemctl status postgresql

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

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

2. 数据库

```
# Ping
ping db.example.com

# (MySQL/MariaDB: 3306, PostgreSQL: 5432)
telnet db.example.com 3306
#
telnet db.example.com 5432
```

3.

```
#
echo $DB_USERNAME
echo $DB_HOSTNAME
echo $DB_PORT

# (MySQL/MariaDB)
mysql -u $DB_USERNAME -p$DB_PASSWORD -h $DB_HOSTNAME

# PostgreSQL:
# psql -U $DB_USERNAME -h $DB_HOSTNAME -d sms_c_prod
```

```
# (MySQL/MariaDB)
systemctl start mysql

# (PostgreSQL)
systemctl start postgresql
```

```
export DB_USERNAME=correct_user
export DB_PASSWORD=correct_password
```

```
# 检查配置
systemctl restart sms_c
```

检查配置

- 检查配置
- 检查配置
- 检查 VPN/配置

检查配置

检查配置

```
config :sms_c, SmsC.Repo,
  pool_size: 50 # 检查配置
```

检查配置

检查配置

- 检查配置
- API 检查
- 检查配置

检查配置

1. 检查配置

```

-- MySQL/MariaDB: 检查慢查询
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 1; -- 时间 > 1 秒

-- 查看慢查询 (MySQL/MariaDB)
SELECT * FROM mysql.slow_log ORDER BY query_time DESC LIMIT 10;

-- PostgreSQL: 检查 postgresql.conf 配置
-- log_min_duration_statement = 1000 # 秒
-- 查看 PostgreSQL 配置

```

2. 检查消息队列

```

-- 检查消息队列
SHOW INDEX FROM message_queues;

-- 检查消息队列表结构
-- - source_smsc
-- - dest_smsc
-- - send_time
-- - inserted_at

```

3. 检查表大小

```

-- MySQL (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';

-- PostgreSQL (PostgreSQL)
-- SELECT schemaname, tablename,
--
pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename))
AS size
-- FROM pg_tables WHERE schemaname = 'public';

```

□□□□

□□□□

```
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);
```

□□□□

```
-- MySQL/MariaDB
OPTIMIZE TABLE message_queues;
OPTIMIZE TABLE frontend_registrations;

-- PostgreSQL
-- VACUUM ANALYZE message_queues;
-- VACUUM ANALYZE frontend_registrations;
```

□□□□

□□□□□

```
-- □□□□ 30 □□□□□□□□
DELETE FROM message_queues
WHERE status = 'delivered'
AND deliver_time < DATE_SUB(NOW(), INTERVAL 30 DAY)
LIMIT 10000;
```

□□□□□□

□□□

- □□□“□□□□”
- □□□□□□□□

- 检查磁盘

检查磁盘

1. 检查磁盘使用情况

```
df -h

# MySQL 数据库 (MySQL/MariaDB)
du -sh /var/lib/mysql

# PostgreSQL 数据库 (PostgreSQL)
du -sh /var/lib/postgresql
```

2. 查找大文件

```
# MySQL/MariaDB 数据库
find /var/lib/mysql -type f -exec du -h {} + | sort -rh | head -20

# PostgreSQL 数据库
find /var/lib/postgresql -type f -exec du -h {} + | sort -rh | head -20

# 短信日志
du -sh /var/log/sms_c/*
```

检查数据库

清理数据库

```
-- 清理消息队列
DELETE FROM message_queues
WHERE inserted_at < DATE_SUB(NOW(), INTERVAL 90 DAY)
LIMIT 100000;
```

检查数据库

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

# 清理旧日志
find /var/log/sms_c -name "*.log.*" -mtime +30 -delete
```

配置

- 配置 logrotate
- 配置 find
- 配置 cron

部署

测试

验证

- 验证“配置”
- 验证“部署”
- 验证“测试”

总结

1. 配置

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

2. 部署

- 部署 `/api/frontends/register`
- 部署 API 接口
- 部署 60 秒

3. 部署 **API** 接口

```
grep "frontend.*register" /var/log/sms_c/application.log | tail -20
```

□□□□

□□□□□

□□□□□□

```
curl -X POST https://api.example.com:8443/api/frontends/register \
-H "Content-Type: application/json" \
-d '#123;
  "frontend_name": "uk_gateway",
  "frontend_type": "smpp",
  "ip_address": "10.0.1.50"
  &#125;'
```

□□□□□□□□□□□□□□□□/□□□□

□□□□□

□□□ 90 □□□□□□□□□□ 60 □□□□□□

```
# □□□□ 60 □□□□□
while True:
    register_with_smsc()
    time.sleep(60)
```

□□□□□

- □□□□□ API □□□□□□□
- □□ DNS □□
- □□□□□□□□□□ curl □□

□□□□□□□□□□/□□

□□□

- 网络延迟/丢包率
- 网络带宽
- 网络拥塞

网络工具

1. 网络连通性

- 网络连通性测试
- 网络路由跟踪
- 网络带宽测试

2. 网络性能

```
# 网络延迟测试  
ping -c 100 api.example.com  
  
# 网络流量统计  
netstat -s | grep -i reset
```

3. 网络故障排除

- 网络故障排查
- 网络性能优化

网络配置

网络接口

- 网络接口配置
- 网络接口管理
- 网络接口性能优化

网络路由

- 网络路由配置
- 网络路由管理
- 网络路由性能优化

□□□□□□

□□□□□□

```
REGISTRATION_INTERVAL = 60 # □
```

□□/□□□□

□□□□

□□□

- sms_c_charging_failed_count □□
- □□□□□□“charging_failed”
- □□□□□ charge_failed: true

□□□□□

1. □□ **OCS** □□□□

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

□□□ {"result": "Pong"}

2. □□ **OCS** □□□

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

3. □□□□□

```
# 检查 OCS URL
grep ocs_url config/runtime.exs
```

检查

OCS 安装

```
# 检查 OCS 是否安装
systemctl status ocs

# 启动 OCS
systemctl start ocs
```

检查

检查

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

检查

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

检查

检查

- 检查 OCS 是否安装
- 检查 OCS 配置
- 检查 OCS 启动

□□□□

□□□

- `sms_c_charging_succeeded_duration` p95 > 500ms
- □□□□□□□□□□
- □□□□□

□□□□□

1. □□□□□□□

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

2. □□ **OCS** □□□

```
# 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. □□□□□□□

```
# Ping OCS □□  
ping -c 10 ocs.example.com
```

□□□□□

OCS □□

- □□ OCS □□
- □□ OCS □□
- □□□□□□□□□□

□□□□□

- □ OCS □□□□□□ SMS-C □□□

- 配置项
- 配置项 VPN/配置

配置项

配置项

```
config :sms_c,  
  ocs_timeout: 5000 # 5
```

ENUM 配置

ENUM 配置

配置

- `sms_c_enum_lookup_stop_duration` 配置
- 配置 ENUM 配置
- 配置 `enum_result_domain` 配置

配置

1. 配置 ENUM 配置

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

2. 配置 DNS 配置

```
# 配置 DNS 配置  
dig @8.8.8.8 e164.arpa  
  
# 配置 ENUM 配置  
# 配置 +15551234567:  
dig @8.8.8.8 NAPTR 7.6.5.4.3.2.1.5.5.5.1.e164.arpa
```

3. 配置 DNS 配置


```
# DNS ping
ping 10.0.1.53

# nc
nc -zv 10.0.1.53 53
```

???

DNS

DNS

```
config :sms_c,
  enum_dns_servers: [
    "8.8.8.8", 53, # Google DNS
    "1.1.1.1", 53, # Cloudflare DNS
  ]
```

ENUM

```
config :sms_c,
  enum_domains: ["e164.arpa"] #
```

```
config :sms_c,
  enum_timeout: 10000 # 10
```

ENUM

```
config :sms_c,
  enum_enabled: false
```

ENUM 配置

配置

- 命中率 < 70%
- 命中率
- 命中率

配置

1. 命中率

```
# 命中率  
rate(sms_c_enum_cache_hit_count[5m]) /  
(rate(sms_c_enum_cache_hit_count[5m]) +  
rate(sms_c_enum_cache_miss_count[5m]))  
  
# 命中率  
sms_c_enum_cache_size_size
```

2. 命中率

- 命中率
- TTL 命中率

配置

配置

- 命中率
- 命中率 < 70% 命中率

配置

配置 NAPTR 配置

配置

- 命中率
- 命中率

- 设置 TTL

设置

设置

设置

- 设置
- 设置
- Erlang 设置

设置

1. 设置

```
# 设置 IEx 设置  
Node.self()  
# 设置: :sms@node1.example.com  
  
Node.list()  
# 设置: 设置
```

2. 设置 Erlang Cookie

```
# 设置 cookie 设置  
cat ~/.erlang.cookie  
  
# 设置
```

3. 设置

```
# ping node2.example.com
ping node2.example.com

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

cookie

Cookie

cookie

```
export ERLANG_COOKIE=same_secret_value_here

# ~/.erlang.cookie
echo "same_secret_value_here" > ~/.erlang.cookie
chmod 400 ~/.erlang.cookie
```

iptables

iptables

```
# EPMD
iptables -A INPUT -p tcp --dport 4369 -j ACCEPT

# Erlang
iptables -A INPUT -p tcp --dport 9100:9200 -j ACCEPT
```

DNS

IP

```
config :sms_c,
  cluster_nodes: [
    "sms@10.0.1.10",
    "sms@10.0.1.11"
  ]
```

□□□□□

□□□

- ❓❓❓□□□□□□□□
- □□□□□□□□□□
- Mnesia □□□

□□□□□

1. □□□□□□□□

```
# □□□□□□ (IEx)  
Node.list()
```

2. □□ **Mnesia**□

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

□□□□□

□□□□□□□□

```
# □□□□□□□  
systemctl stop sms_c  
  
# □□□□□□□□□  
systemctl start sms_c # □ node1 □  
  
# □□□□□□□□□□□□□□□□□□  
systemctl start sms_c # □ node2 □  
systemctl start sms_c # □ node3 □
```

Mnesia □□□□

- □□□□□□□□□□
- □□□□□□
- □□ Mnesia □□

- `systemctl`
- `netstat`

API `status`

API `status`

`status`

- `systemctl`
- `netstat`
- `curl`

`status`

1. `API status`

```
# systemctl status sms_c  
  
# netstat -tlnp | grep 8443
```

2. `iptables`

```
# iptables -L -n | grep 8443  
  
# curl -k https://localhost:8443/api/status
```

3. `TLS status`

```
# 查看证书文件
ls -l priv/cert/server.crt priv/cert/server.key

# 查看证书信息
openssl x509 -in priv/cert/server.crt -noout -dates
```

查看

查看

```
systemctl start sms_c
```

查看

```
# 配置 API 端口
iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

查看

查看

查看

查看

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

API 端口 500 配置

配置

- 配置
- 500 端口
- 配置

查看

1. 查看日志

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

2. 数据库连接

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

3. 系统资源

```
# 内存  
free -h  
  
# CPU  
top -b -n 1  
  
# 磁盘  
df -h
```

网络

网络配置

- 网络接口
- 网络配置

防火墙

- 防火墙规则
- 防火墙配置
- 防火墙日志

安全加固

- 系统更新
- 用户管理
- 权限管理

Web UI 問題

問題 Web UI

問題

- 問題
- 404 問題
- 問題

問題

1. 問題

```
systemctl status sms_c
```

2. 問題

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

3. 問題 **URL**

- 問題
- 問題
- HTTP と HTTPS

問題

問題

問題

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

問題 80 と 4000

□□□□□□□□

```
systemctl start sms_c
```

□□□□

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

LiveView □□□

□□□

- □□□□□□□□
- □□□□
- □□□□□□□□ WebSocket □□

□□□□□

1. □□□□□□□□□□

- □□□□□□□□ (F12)
- □□ WebSocket □□
- □□□□□□□□□□□□□□

2. □□□□□□□□

□□□□□□□□□□□□□□□□ WebSocket□

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

□□□□□

WebSocket □□□□

- WebSocket 连接
- 消息接收
- 消息发送

快捷键

- 刷新 (Ctrl+F5)
- 重新加载

性能监控

CPU 使用率

命令

- CPU 使用率 > 80%
- 查看
- 限制

配置

1. 配置

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

2. 配置

```
# 配置  
rate(sms_c_message_received_count[5m])  
  
# 配置  
rate(sms_c_routing_route_matched_count[5m])
```

配置

配置

- 000000000000
- 00000000 CPU

000000

- 000000
- 000000

ENUM 000000

- 00000000
- 0000000000

000000

000

- 0000 > 90%
- 000000
- 000000

000000

1. 000000

```
free -h
```

```
ps aux | grep beam.smp
```

2. 00000000

```
sms_c_enum_cache_size_size
```

000000

ENUM 000000

- 0000

- TTL
- ENUM

ENUM

```
# ENUM (IEx)  
SmsC.Messaging.BatchInsertWorker.stats()
```

ENUM


ENUM

- ENUM
- ENUM

ENUM

- ENUM
- ENUM

ENUM

- ENUM - ENUM
- ENUM - ENUM
- ENUM - ENUM
-  ENUM - `/var/log/sms_c/application.log`

