

README

Benchee の SMS-C 実装のベンチマーク

インストール

1. SMS 実装 (raw_sms_bench.exs)

`submit_message_raw` API の SMS PDU

を実行

- SMS PDU の PDU を `@sample_pdus` から取得
- 実行結果を `raw_sms_benchmark.html` に出力
- HTML を表示

実行

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

結果は `benchmarks/output/raw_sms_benchmark.html` に出力されます。

2. API 実装 (message_api_bench.exs)

API の実装をベンチマーク

を実行

- `insert_message` の実行
- `get_messages_for_smsc` の実行
- `list_message_queues` の実行
- 実行結果を `message_api_benchmark.html` に出力

実行

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

📄 `benchmarks/output/message_api_benchmark.html`

📄

📄📄📄📄📄 Benchee📄📄📄📄📄

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

📄📄

HTML 📄📄📄📄📄📄 `benchmarks/output/` 📄📄📄📄

- 📄📄📄📄📄
- 📄📄📄
- 📄📄📄📄
- 📄📄📄

SMS-C API

← ☐ ☐ ☐ ☐ ☐ ☐

SMS-C REST API 設定ファイル (config/config.exs) の設定項目

□□□□□□□□□□□□ OpenAPI □□□□□□ Swagger UI□□□□□□□□□□□□□□□□

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

11

- 00
- 00
- 0000
- 00
- 00
- 000000
- 00
- 00
- 00
- 00
- 000000
- HTTP 000
- 0000
- 0000

11

□□ URL

https://<host>:8443/api

```

8443 (config :api_ex, api: %{port: ...} ) : enable_tls
true HTTP /api

```

--	--	--	--

□□□□□□ JSON□

Content-Type: application/json

URL API /api/schema

11

API 00000000000000000000 OpenAPI 00000000 API 00000000000000000000

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- **TLS:** `enable_tls` `true` `tls_cert_path` / `tls_key_path` `HTTPS`
- **代理:** `API` `IP` `X-Forwarded-For`

API 0000000000000000

--	--	--	--

data

□□□□

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

□□□□ JSON □□□□

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

□□□□□□□□□□ error □□□□

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

□□□□□□□□□□□□□□□□□□□□

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

□□

□□□□□□□□□□□□□□□□

□□:

```
GET /api/status
```

□□ (200 OK):

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

00:

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

00

0000000000000000 **payload_hex** TP-User-Data 0000 **dcx** 0000000000000000
udh_hex 0000000000000000 0000 000000000000DCS 00000000

0000

000000000000 **ApiEx.Schemas.MessageSchema** 00

欄名	型別	説明
id	integer	ID
source_msisdn	string	発信番号
destination_msisdn	string	着信番号
payload_hex	string	TP-User-Data の16進数表現
dcs	string	00=GSM7, 08=UCS2, 04=8bit, C8=MWI
udh_hex	string / null	UDHの16進数表現
message_text	string / null	メッセージ本文 (UTF-8 変換済み)
source_smsc	string	発信 SMS 送信元
dest_smsc	string / null	着信 SMS 宛先
status	string	queued, backoff, sent, delivered, expired, dropped, auto_replied, balance_rejected, deleted
send_time	datetime	
deliver_time	datetime / null	
expires	datetime	
delivery_attempts	integer	

名前	型	説明
deadletter	boolean	
message_parts	integer / null	
message_part_number	integer / null	
smsc_node_name	string	SMSCノード名

リクエスト

例:

```
GET /api/messages
```

パラメータ:

- `smsc: <frontend_name>` - 検索する SMSC の名前
- `include-unrouted: true|1` - 未ルーティングのメッセージを含める (デフォルトは `false`)
- `x-decode: true|1` - メッセージをデコードする

クエリパラメータ `smsc` の値:

- `status` - ステータス (デフォルトは `all`)
- `limit` - 取得するメッセージの数 (100 から 1000 まで)
- `offset` - オフセット (0 から)

レスポンスは `limit` パラメータで指定された数 (デフォルトは 100) のメッセージを含む

200 OK - JSON 形式

例:

送信

```
curl -H "smc: uk_gateway" https://host:8443/api/messages
```

送信履歴

```
curl -H "smc: uk_gateway" -H "include-unrouted: true"
https://host:8443/api/messages
```

送信履歴

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

送信

URL:

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

ヘッダ `x-decode: true` 送信履歴

レスポンス (200 OK): 送信履歴 **404** ステータス `{"error": "Message not found"}` ID 送信履歴

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

送信

送信履歴 `payload_hex` 送信履歴 422

URL:

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

URL:

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

□□:

- `source_msisdn`, `destination_msisdn` - □□□
- `payload_hex` - □□□TP-User-Data □□□□□□□□
- `dcs` - □□□□□□□□□□□□□□□□□□□□
- `udh_hex` - □□□□□□□□□□□□□□
- `source_smsc` - □□□□□□□□
- `source_type` - □□□□□□□□

□□ (201 Created): □□□□□□□□□□□□ 422 □ `error` □□□□□□□□□□ `payload_hex`□□

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

□□□□

□□□□ UTF-8 □□□□□ SMS-C □□□□□ GSM7 □ UCS2□□□□□ DCS □□□□□□□□□□

□□:

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

11

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

```

message text body source_msisdn destination_msisdn

```

□□ (201 Created):

- [] [] [] [] [] [] [] [] [] []
- [] [] [] { "parts": <n>, "messages": [...] } []

```
curl -X POST https://host:8443/api/messages_plaintext \
-H "Content-Type: application/json" \
-d '{
    "source_msisdn": "15555551234",
    "destination_msisdn": "1555555678",
    "message": "Hello World",
    "source_smsc": "api_client"
}'
```

□□□□ **PDU**

```

000000 SMS PDU0000000000000000000000 SIP 000

```

11

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

11

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

□□:

- `message_body` - 160 octets SMS PDU TPDU
- `source_msisdn` - 16 octets
- `source_smsc` / `smsc_node_name` - 16 octets TPDU
- `rp_originator_address`, `rp_destination_address` - 16 octets TPDU RP

```
IP X-Forwarded-For
```

11

- 201 Created [] { "status": "success", "response": { "id": <id> } }
[] [] rp_ack [] []
- 200 OK [] { "status": "success", "response": { "status":
"delivery_report_received" } } [] PDU [] [] [] [] [] [] [] [] [] []
- 422 [] [] [] [] [] [] [] [] [] [] error [] []

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

000000 PDU 3GPP TS 23.040 000000/00/DCS 00000000000000000000000000000000
 00

□□□□ (PATCH)

□ □ □ □ □ □ □ □ □ □ □ □

##:

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

dest_smsc deliver_after status

(200 OK): #####404 #####422

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

#####

backoff

#####

```
PUT /api/messages/:id # #####PUT = ##
POST /api/messages_increment_attempt (id #####)
PUT /api/messages_increment_attempt/:id (id # URL #)
```

PUT /api/messages/:id # PATCH /api/messages/:id #####PATCH #####
PUT ##### keep_route=true ##### dest_smsc #

(200 OK):

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

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

#####

#####

```
POST /api/messages_mark_delivered      (id 0000)
PUT  /api/messages_mark_delivered/:id  (id 0 URL 0)
```

0000 `smsc` 000000 SMSC0000000000000000

00 (200 OK): 0000000000404 00000000

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

00000000

00000000000000000000 `dest_smsc`0

00:

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

00: { "id": 12345 }

00 (200 OK):

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

000000000000 40400000000000000000 4220

000000

00000000000000000000 null00000000000000000000

00:

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

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

`field` `delivery_time` `dest_smsc` `deliver_after` 400

(200 OK):

:

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

(204 No Content)404

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

MWI 3GPP TS 23.038 MWI PDU
`payload_hex` `dcs`

:

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

:

```
{
  "source_msisdn": "12345",
  "destination_msisdn": "15555555678",
  "indication_type": "voicemail",
  "indication_active": true,
  "message_body": "You have 1 new voicemail",
  "source_smsc": "IMS_SMSC"
}
```

□□:

- indication_type - voicemail □□□□ fax □ email □ other □
- indication_active - true □□□□□ false □□□□□ true □□
- message_body - □□□□□□□ GSM7 □□□□
- source_msisdn □ destination_msisdn □ source_smsc □ dest_smsc □

□□ (201 Created): □□□□□□□ 422 □□□□□

□□

□□□□□□□□□□□□□□□□ REST □□□ :id □□□□ IMSI□

```
GET    /api/locations          # □□□□□□?limit=&offset=□x-total-
count □□
GET    /api/locations/:id   # □ IMSI □□
POST   /api/locations        # □□□□□□□ IMSI □□□
PATCH /api/locations/:id   # □□□□
PUT    /api/locations/:id   # □□
DELETE /api/locations/:id   # □□
```

□□/□□□□:

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

imsi msisdn 200/201 422

SMSC active stats history register

```
GET /api/frontends #
GET /api/frontends/:id #
POST /api/frontends #
```

(POST /api/frontends):

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

`frontend_name` 0000000000 `ip_address`000000 IP (`X-Forwarded-For` 0000) 0000
 { `frontend_name` 000000000000000000000000000000 90 00000000000000000000

201 Created): 000000

00

00000000000000000000 ETS 000000 TTL0

00000000

00:

GET /api/events/:id

`:id` 000000 ID0

200 OK):

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

00 ID 00000000 4000

000000

00:

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

{}:

```
{  
  "message_id": 12345,  
  "name": "external_delivery_confirmed",  
  "description": "Confirmed by downstream system"  
}
```

message_id name description IP

{} (201 Created):

```
{  
  "status": "success",  
  "message": "Event recorded successfully",  
  "event": {  
    "message_id": 12345,  
    "name": "external_delivery_confirmed",  
    "description": "Confirmed by downstream system",  
    "event_source": "smpp-gateway-1",  
    "event_epoch": 1698672010  
  }  
}
```

{} {}

{} / {}
{} {}

API

```
GET    /api/campaign_lists      # 取得
GET    /api/campaign_lists/:id # 取得詳細 "recipients"
POST   /api/campaign_lists    # 作成
PATCH /api/campaign_lists/:id # 更新
DELETE /api/campaign_lists/:id # 削除
```

API には、変数 `var1..var4` を利用して、JSON、CSV、JSON 形式のデータを生成できます。MSISDN 形式の変数 `var1..var4` は、

```
{
  "name": "VIP customers",
  "description": "High value subscribers",
  "recipients": [
    { "msisdn": "12025550101", "var1": "Alice", "var2": "Gold" },
    "12025550102"
  ]
}
```

CSV 形式は `msisdn / destination_msisdn / number / phone` の形式で MSISDN のデータを生成します。変数 `var1..var4` は、

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

`PATCH` 更新

API (201 Created): { "id": 1, "name": "VIP customers", "recipient_count": 2 }

11

```
GET      /api/campaigns           # [] [] [] [] [] [] [] [] [] []
GET      /api/campaigns/:id       # [] [] [] [] [] [] [] []
POST     /api/campaigns           # [] [] [] [] [] [] [] [] [] []
DELETE   /api/campaigns/:id       # [] [] [] [] [] [] [] []
```

□□□□:

```
{
  "name": "Maintenance notice",
  "message": "Planned maintenance tonight 02:00-03:00.",
  "source_msisdn": "12345",
  "source_smsc": "IMS_SMSC",
  "list_id": 1,
  "audience": "active_only",
  "drip_tps": 50
}
```

11

- name message source_msisdn source_smsc - message
{{ var1 }}..{{ var4 }}{{ msisdn }}
- list_id -
- audience - active_only all list_id
- drip_tps -
- validity_hours - expires
- deliver_after - ISO 8601

DELETE [200] TTL []

```
␣␣ (GET /api/campaigns/:id):
```

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

□□□□

□□:

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

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

action □ start □ pause □ resume □ cancel □□□□□□□□□□200□□□□□□□□□□ 404 □□□
□□□□□□□□□□□□ 422 □

□□□□□□□□

```
GET /api/campaign_targets/:id # :id □□□ ID
GET /api/campaign_targets/:id?state=failed # □□□□□□□
```

□□ limit/offset □□□□□□□□□□ state □□□□□ x-total-count □□□□□□□□□□□□□□
state □ pending / queued / delivered / failed / skipped_inactive □□

API

API 文档

API

```
GET    /api/routes          # 获取所有路由
GET    /api/routes/:id     # 获取指定 ID 的路由
POST   /api/routes        # 创建新路由
PATCH /api/routes/:id     # 更新指定 ID 的路由
PUT    /api/routes/:id     # 更新指定 ID 的路由
DELETE /api/routes/:id     # 删除指定 ID 的路由
GET    /api/routes/export  # 导出所有路由
POST   /api/routes/import  # 导入路由
POST   /api/routes/simulate # 模拟路由
```

API SMS 接口

API

```
GET    /api/translations  # 获取所有翻译
GET    /api/translations/:id # 获取指定 ID 的翻译
POST   /api/translations  # 创建新翻译
PATCH /api/translations/:id # 更新指定 ID 的翻译
DELETE /api/translations/:id # 删除指定 ID 的翻译
GET    /api/translations/export # 导出所有翻译
POST   /api/translations/import # 导入翻译
POST   /api/translations/simulate # 模拟翻译
```

API 接口

□□

```
GET /api/operations # □□□□
POST /api/operations/clear_messages # □□□□□□
POST /api/operations/clear_routes # □□□□
POST /api/operations/clear_translations # □□□□
POST /api/operations/clear_frontends # □□□□□□
```

□□□□□□□□□□□□□□□□ API □□□

□□

```
GET /api/federation/status # □□□□
GET /api/federation/identity # □□□□□□□□
POST /api/federation/health # □□□□□□
POST /api/federation/registry # □□□□
POST /api/federation/notify # □□□□□□
POST /api/federation/delivery_status # □□□□□□
GET /api/federation_messages # □□□□□□
```

□□□ □□□□□

Diameter□□□□ DNS

```
GET /api/diameter # Diameter Sh □□□/ dip □□
GET /api/logs # □□□□□□□□
POST /api/naptr # NAPTR □□□□
```

Diameter Sh □□□□□ HSS □□□□□□□ □□□□

HTTP 状态码

状态码	描述	中文描述
200	OK	成功
201	Created	创建成功
202	Accepted	接受成功
204	No Content	无内容
400	Bad Request	请求无效，ID 不存在
404	Not Found	未找到
422	Unprocessable Entity	无法处理
500	Internal Server Error	服务器内部错误

JSON 响应

返回 JSON 响应时，`error` 属性

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

返回 JSON 响应时，`status` 和 `response` 属性

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

API

API

1. API: TPDU `payload_hex` `/api/messages` `/api/messages_plaintext` SMS-C
2. `source_smsc`
3. **E.164**
4. **422** `error`

API

1. ~**90** `POST /api/frontends`
2. `smc` `include-unrouted`
3. `/api/messages_mark_delivered`
4. `PUT /api/messages/:id` `/api/messages_increment_attempt`
- 5.

API

1. TLS API
2. HTTP
3. Prometheus [METRICS.md](#)

ANSI R226 项目建议书

本项目旨在开发一个符合 R226-3 和 R226-7 规范的 ANSI R226 项目建议书。项目将使用 OmniMessage 和 SMS 库。

项目目标

项目将使用 ANSI 规范

项目 R226 - 项目建议书

1. 项目概述

1.1 项目背景

本项目旨在开发一个符合 ANSI R226 规范的 SMS 库。

项目将使用 OmniMessage 库。

项目将使用 SMS 库。

项目将使用 REST API、HTTPS、SMPP、IMS、SS7/MAP 等协议。

项目将使用 Elixir/Erlang 语言。

项目将使用 Phoenix 框架、Mnesia 数据库、MySQL/PostgreSQL 数据库。

项目将使用

项目将使用

- 项目将使用 REST API
- 项目将使用 SMPP、IMS、SS7/MAP 等协议
- 项目将使用 OmniMessage 库
- 项目将使用 SMS 库
- 项目将使用 Elixir/Erlang 语言
- 项目将使用 Phoenix 框架
- 项目将使用 Mnesia 数据库
- 项目将使用 MySQL/PostgreSQL 数据库
- 项目将使用 1,750 个端口/150 个端口/150 个端口

项目将使用

- 安裝 Mnesia 資料庫
 - 安裝 RAM 資料庫
 - 設定 `disc_copies` 資料庫
 - 安裝 24 個資料庫
- 安裝 **CDR** 資料庫 MySQL/PostgreSQL
 - 安裝 CDR 資料庫
 - SQL 資料庫 CDR 表/資料庫
 - 安裝 CDR 資料庫
- 安裝
 - 安裝 1,750 個 SQL 表
 - CDR 表/資料庫
 - SQL
- 安裝 Mnesia 資料庫

安裝

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

安裝

- 安裝
- 安裝/表
- SMSC
- 安裝
- 安裝
- ENUM E.164 DNS
- 安裝
- Diameter Ro OCS

安裝 **README.md**

1.2 環境

OmniMessage 000 ETSI TS 103 221 00000X1X2X300OmniMessage 0000000000
0000000000 X1 00000OmniMessage 00 X2 3GPP TS 33.128 xIRI 000000000000
000000000000 00000X1/X2/X300

1.2.1 背景

□ □ □ □ □ □ □

- 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

--	--	--	--	--	--	--	--

- 國際行動通訊系統 CDR 資料庫儲存項目
 - 行動 ID 號碼
 - 國際行動電話號碼 MSISDN
 - 國際行動電話號碼 MSISDN
 - 國際行動電話號碼
 - 國際行動電話號碼
 - 國際行動電話號碼
 - 國際行動電話號碼
 - 國際行動電話號碼
 - 國際行動電話號碼
 - 國際行動電話號碼/國際 SMS
 - 國際 SMSC 號碼

- 消息 SMSC 發送
- 每個 Erlang 用戶都收到
- 每個用戶都收到
- 每個用戶都收到

CDR CDR_SCHEMA.md

□□□□□□□

- 消息队列
- REST API 消息队列
- 消息队列消息队列
- 消息队列
 - 消息队列/消息
 - SMSC 消息
 - 消息
 - 消息
 - 消息

API API_REFERENCE.md

1.2.2 实验目的

□ □ □ □ □ □ □ □ □ □ □ □

SMSc

1 Mnesia

- 1000 0000000000000000
- 1000 Erlang Mnesia 00000000
- 000000 00000000 disc_copies 00
 - 000000 RAM 0000000000
 - 000000000000000000
 - 0000000000000000
- 0000 0000/0000
- 0000 000000 24 00000000

- 所有 数据库 CDR 数据都存储在 Mnesia 中
- 所有 数据库数据都存储在数据库中
- 数据库 数据库/数据库数据 SQL 数据

2. CDR 数据 MySQL/PostgreSQL

- 所有 数据库数据都存储在数据库中
- 所有 数据库 SQL 数据 MySQL 或 PostgreSQL
- 数据库 数据库数据都存储在 CDR 中
 - 数据库
 - 数据库数据
 - 数据库
 - 数据库数据
- 数据库 数据库数据都存储在数据库中
- 所有 数据库数据都存储在数据库中
- 所有 数据库数据都存储在数据库中
- 所有 SQL 数据 REST API 数据库 CSV/JSON 数据

数据库

1. 所有 数据库数据都存储在 SQL 数据库中
2. 数据库 Mnesia 所有 SQL 数据都存储在 1,750+ 数据库/中
3. 数据库 disc_copies 数据库数据都存储在数据库中
4. 数据库 CDR 数据库数据都存储在数据库中
5. 数据库 数据库数据都存储在数据库中

数据库

1. 数据库 → 数据库 Mnesia RAM + 数据库
2. 数据库 → Mnesia 数据库
3. 数据库/中 → CDR 数据库 SQL 数据库
4. 24 数据库 → 数据库 Mnesia 数据库数据
5. CDR 数据库 SQL 数据库 → 数据库数据都存储在数据库中

数据库

- 数据库数据都存储在数据库中

- 0000000000 PDU 000 Mnesia 0 CDR 00
- 0000000000 CDR 00000000
- 00 CDR 000000000000

000000

- 0000000 SMSC 000SMPP0IMS0MAP 000
- 0000000000
- 0000000000/0000
- 00000/00000
- IP 0000000000
- 000000000000

1.2.3 00000

000000

- Web UI 00000000
 - 00000000
 - 0000000000
 - 000000000000
 - 00000000
 - 0000000000
- Prometheus 000000000000
- 00000000000000000000

0 00000000 **OPERATIONS_GUIDE.md**

0 000000 **METRICS.md**

000000

- CDR 0000000000000000
 - 000000
 - 00/00000000
 - 000000
 - SMSC 00
 - 000000

- 國際行動電話號碼
- 國際號碼
 - 國際/國/國際
 - 國際號碼/國際
 - 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

- E.164 國際號碼
- 國際號碼國際號碼/國際
- 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 [number_translation_guide.md](#)

國際號碼 [sms_routing_guide.md](#)

1.3 安全

1.3.1 安全策略

安全策略

- HTTPS/TLS 保护 REST API 接口
- 数据加密存储
- 数据备份和恢复 TLS
- 安全审计和日志记录

安全策略

- Web UI 安全
- API 安全
- 数据加密
- 安全审计

安全策略

- 数据加密
- 数据备份/恢复
- 安全审计
- 安全审计
- 安全审计和日志记录

1.3.2 安全策略

安全策略

- 数据加密
- 数据加密 UI 接口
- 数据加密
- CDR 数据加密 NULL 数据

安全策略

- MySQL 数据加密 ENCRYPTION='Y'

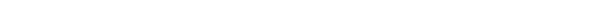
- PostgreSQL 数据库
- 数据库
- 数据库
- 数据库

□□□□□

- 消息队列
- TLS 加密
- 负载均衡
- 消息队列
- 消息队列
- 消息队列 Erlang 消息队列

1.4 Mnesia + SQL

11

OmniMessage SMSc 

1 Mnesia

□□□ **Mnesia**□

- Erlang/OTP 架構
- 分散式系統
- ACID 資料庫
- 網路服務

disc_copies

- **ARMv7-M 32-bit Cortex-M3** RAM
 - 128KB/256KB
 - 16MB/32MB I/O
 - 1,750+ pins/pinpacks
- **ARMv7-M 32-bit Cortex-M3** RAM
 - 128KB/256KB
 - 16MB/32MB I/O

- 000000000000000000000000
- 0000000000 Mnesia.* / 00

Mnesia 0000000000

1. 0000 REST API 00 → 000 Mnesia RAM + 0000
2. 000000 Mnesia → 0000000000
3. 0000000000 → Mnesia 0000000000
4. 0000000000 → Mnesia 000000 + 000
5. 00/0000 → 0000000000
6. 0000000000 24 0000 → 000 Mnesia 0000

0000000000

- 00000000/000000000000 **SQL** 0000
- SQL 000000000000000000000000
- 00000000 SMS-C 00000000 I/O0

0 2000 **CDR** 00/0000 **SQL** 0000

0000 **CDR**000000000000

- 000000000000000000000000
- 00 MySQL 0 PostgreSQL 0000
- 000000000000000000000000

000000 **CDR**0 000000000000000000 CDR 0000

- 0 00000000
- 0 000000000000000000000000
- 0 000000000000000000000000
- 0 000000000000000000000000

000000 **CDR**0

- 00000000 CDR 0000000000000000
- 00000 000000000000 SQL 00
- 000000 00 CDR 000000 1000000000

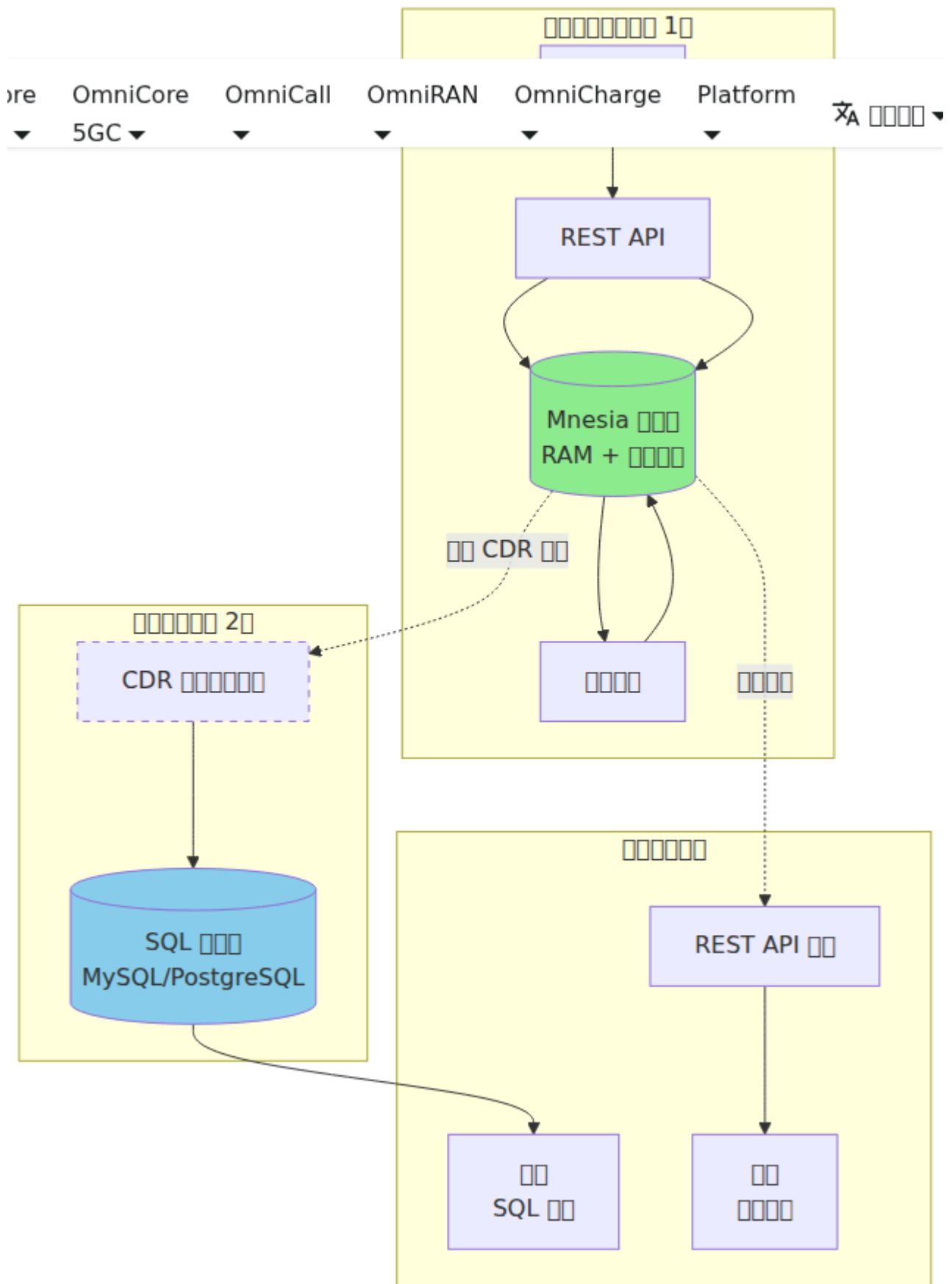
- 数据库 每 100ms 同步一次
- 数据库 同步 CDR 数据

```
# 运行 config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,      # CDR 数据同步
  batch_insert_flush_interval_ms: 100 # 间隔
```

SQL 数据库

- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据
- 数据库 同步 CDR 数据

SQL



SQL

- SQL MySQL/PostgreSQL

- 資料庫設計
- 資料庫建置
- 資料庫測試 SQL

資料庫

資料庫 < 24 資料庫

- 資料庫 Mnesia 資料庫 REST API 資料庫
- 資料庫
- 資料庫設計
- 資料庫建置

資料庫 > 24 資料庫

- 資料庫 SQL 資料庫 CDR 資料庫
- 資料庫 SQL 資料庫
- 資料庫設計
- 資料庫建置

資料庫

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

1.5 資料庫

OmniMessage SMS 資料庫 REST API 資料庫

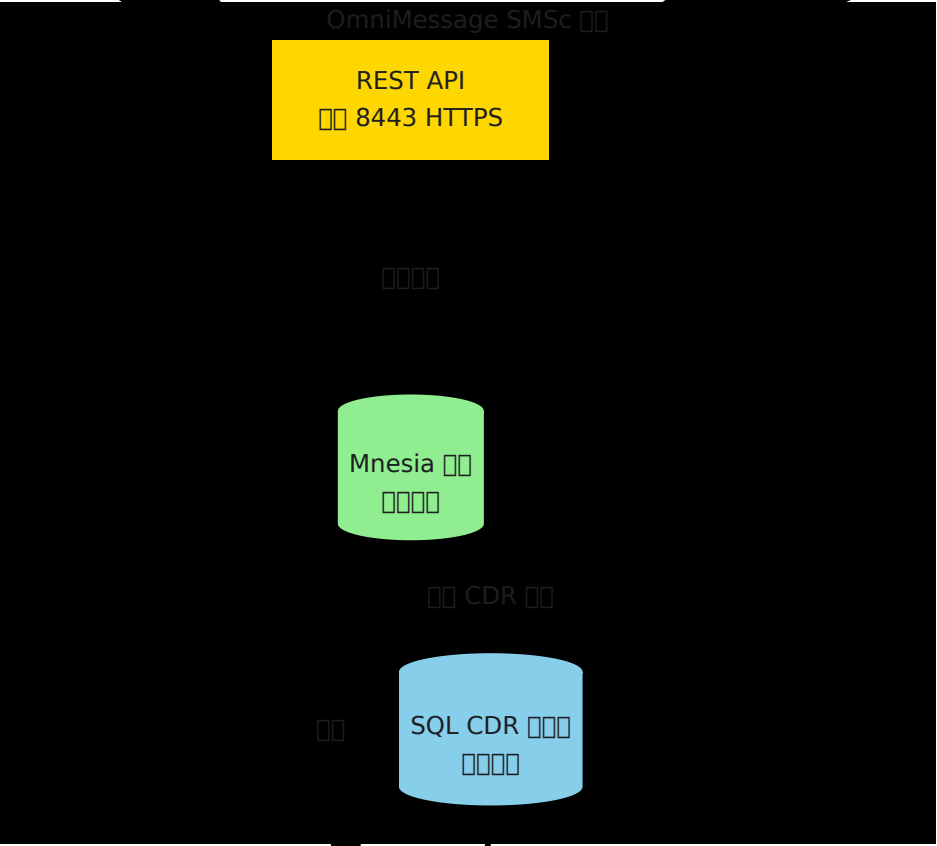
資料庫



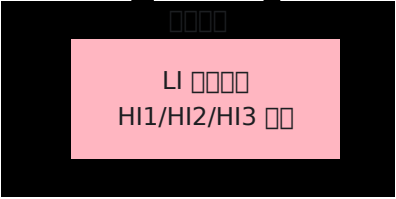
HTTPS POST/GET

HTTPS POST/GET

HTTPS POST/GET



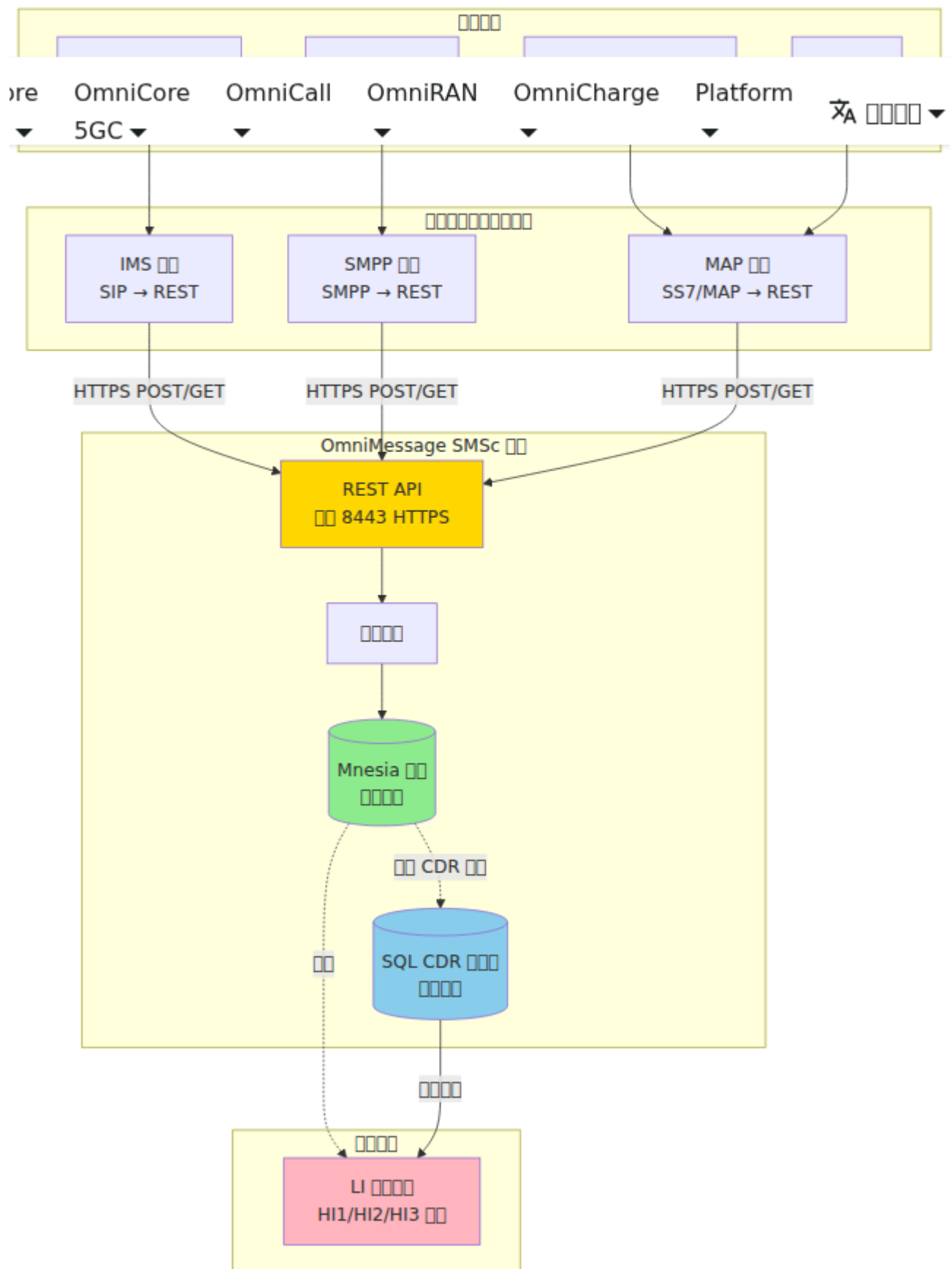
LI



Database

1. IMS/SIP

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



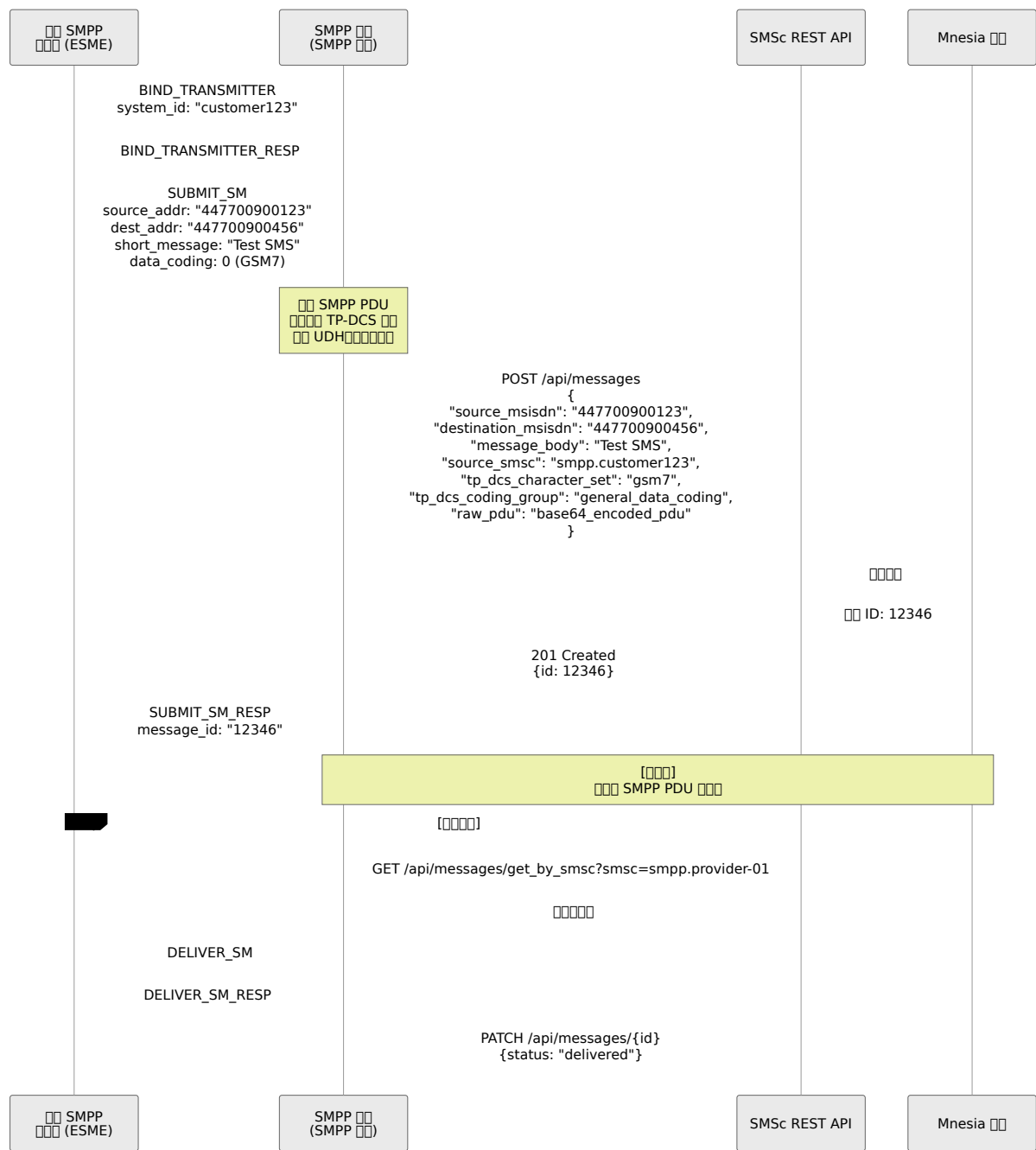
IMS

- IMS (Internet Multimedia Subsystem)

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

2. SMPP

SMPP SMS SMPP PDU SMPP REST API

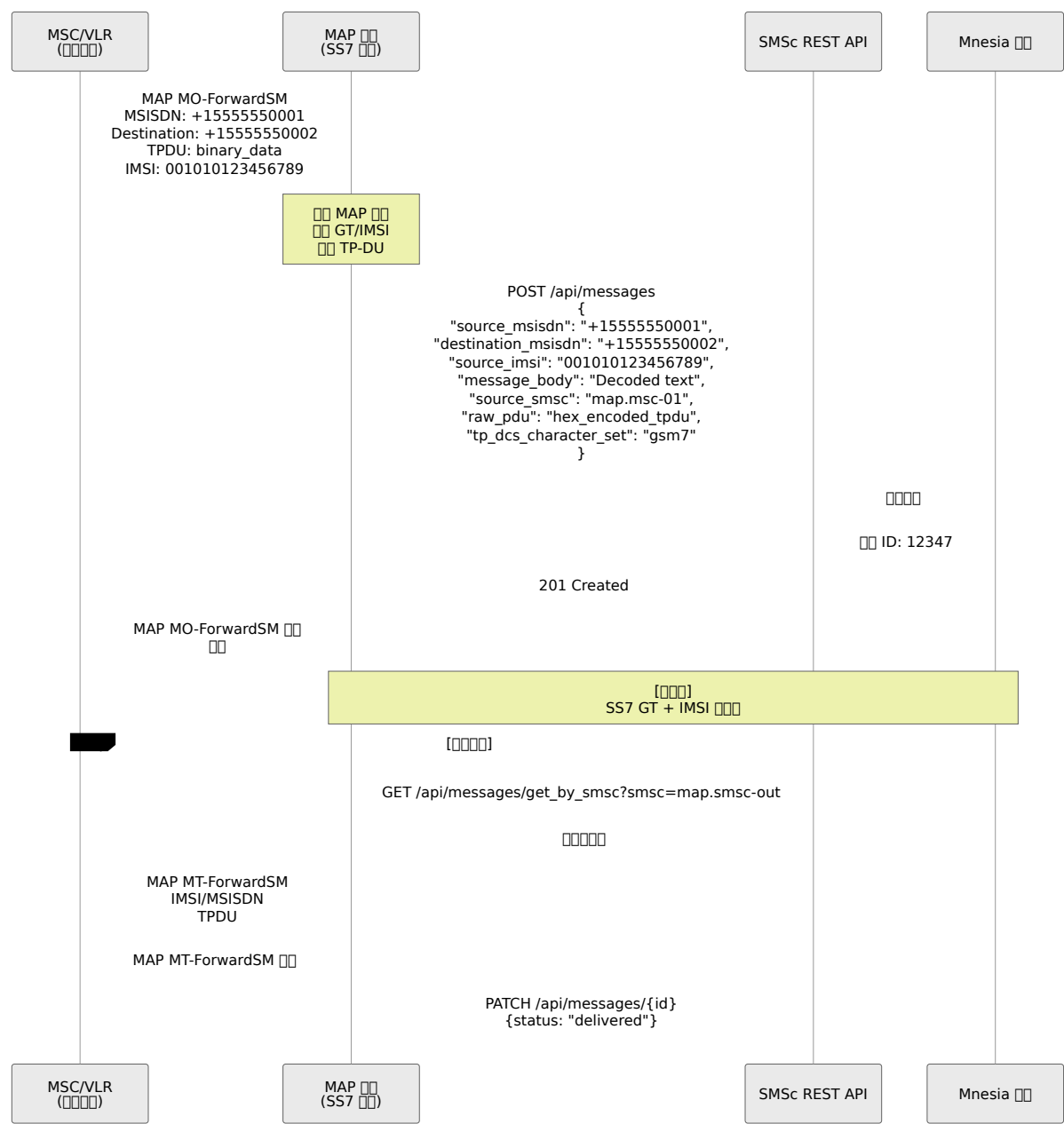


SMPP

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

3. SS7/MAP

SS7 MAP SMS MAP SS7 REST API



SS7/MAP

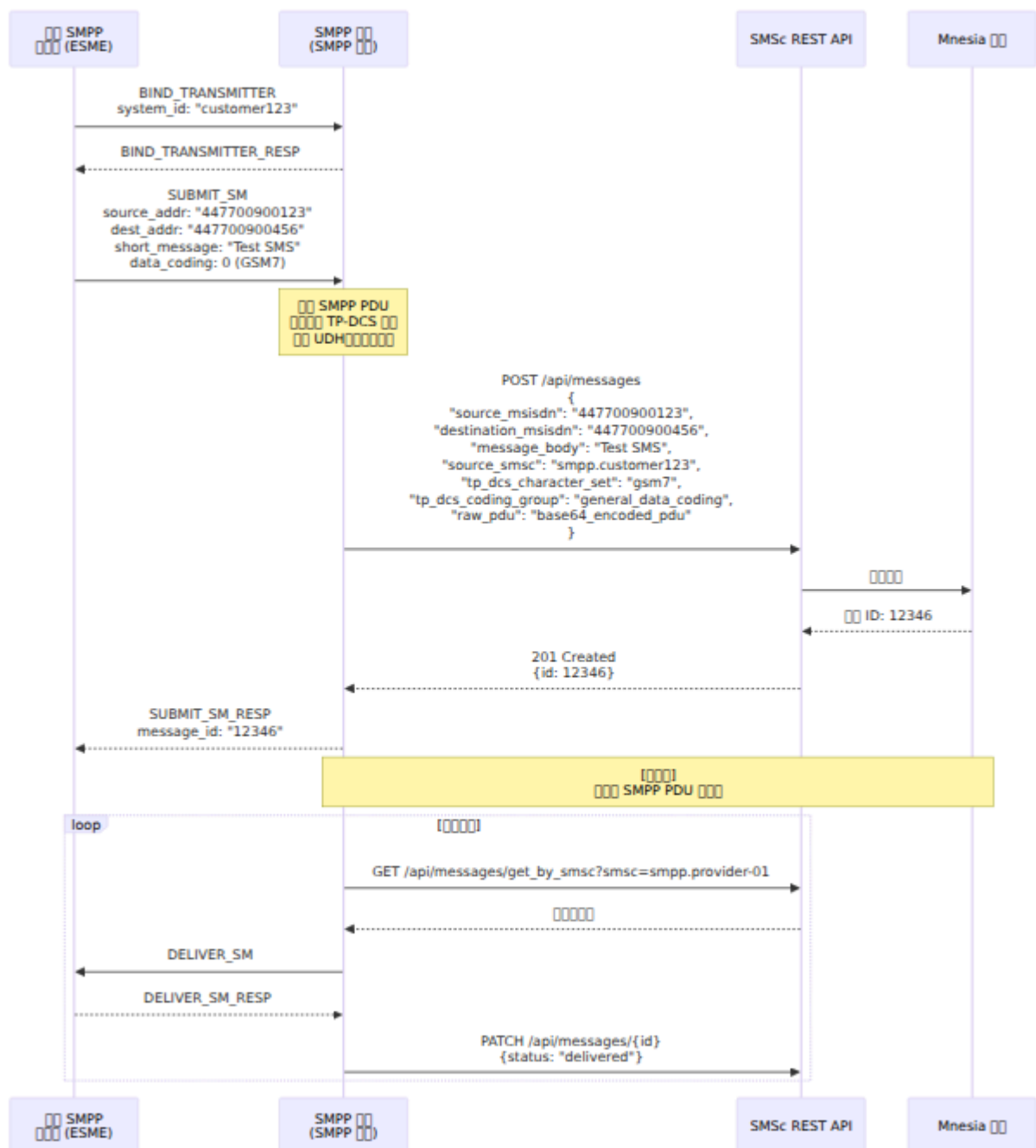
- MAP 網路 IMSI
- 網路GT網路
- MSC/VLR 網路網路網路
- SCCP 網路/網路
- MAP 網路
- TP-網路網路網路

網路網路網路

網路網路網路 網路網路網路IMS/SIP網路SMPP 網路 SS7/MAP網路網路網路 SMSc 網路網路網路網路
網路網路網路

1. 網路網路網路 網路網路網路???
2. 網路 **CDR** 網路 網路網路網路 CDR 網路
3. 網路網路 網路網路網路
4. 網路網路網路 網路網路網路 CDR 網路

網路網路



CDR

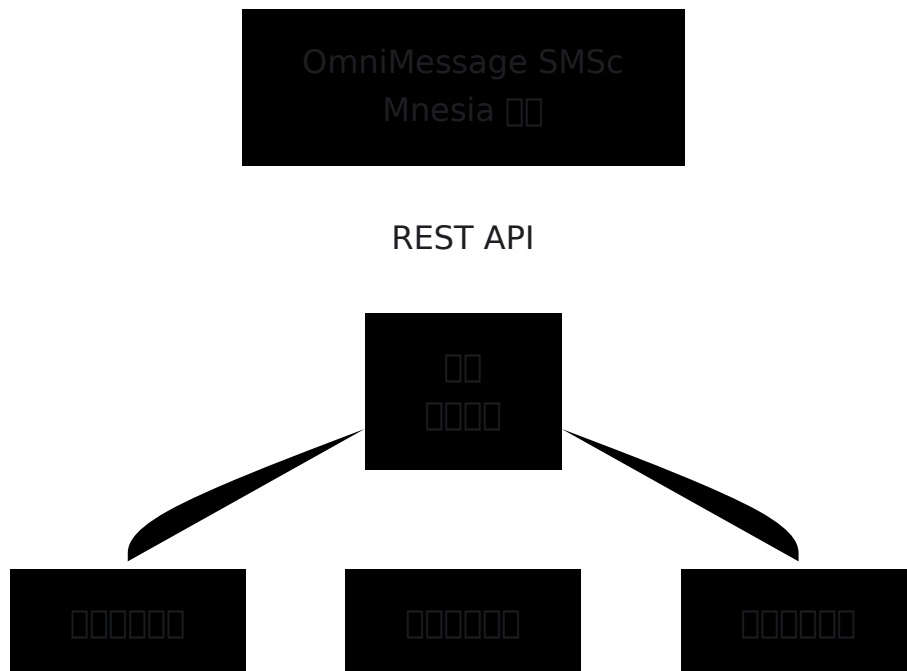
- source_smsc "ims.gateway-01" "smpp.customer123" "map.msc-01"
-
-

1.6

OmniMessage SMS Sc Mnesia SQL

1. REST API Mnesia

Mnesia 24



API

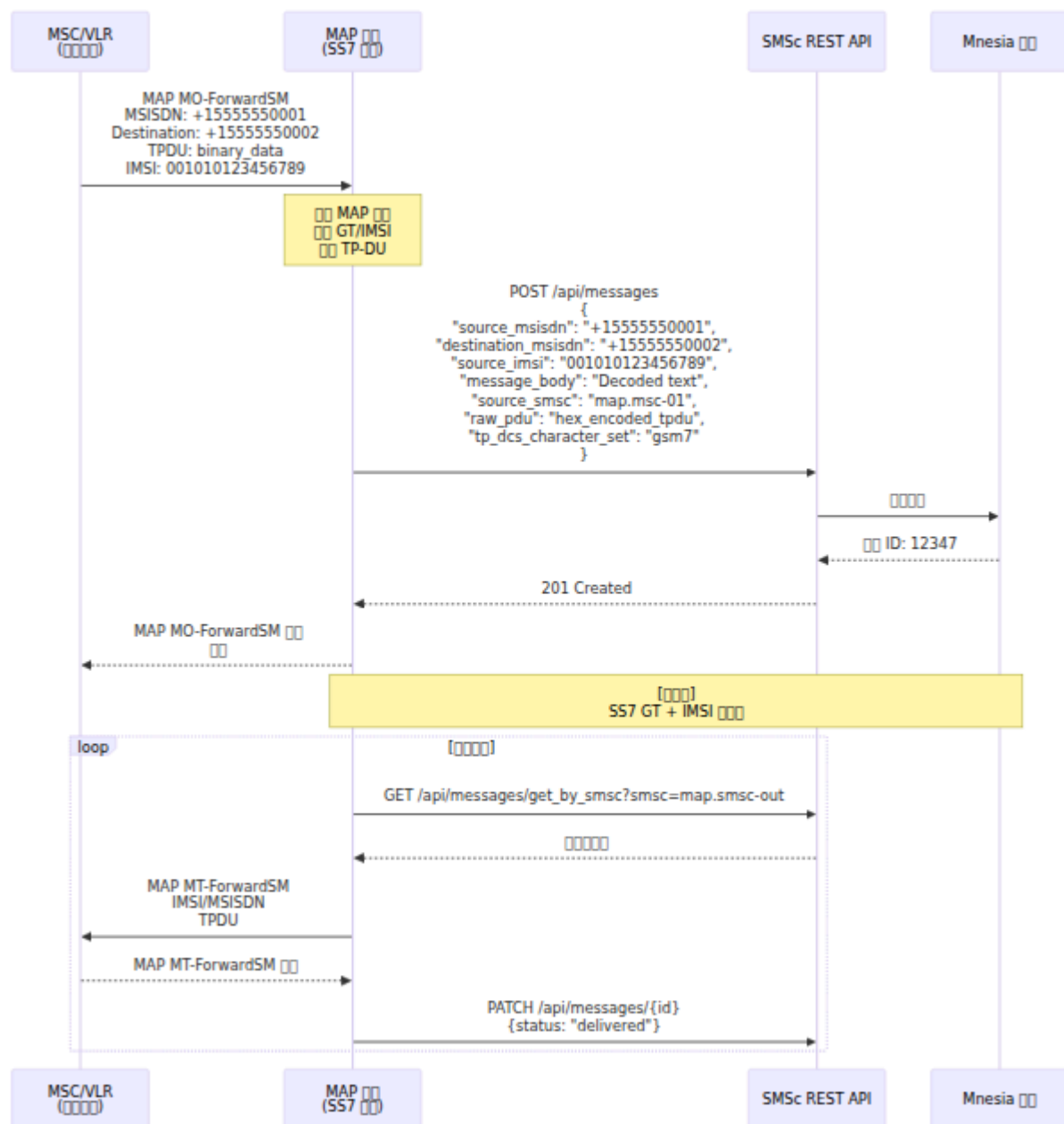
- GET /api/messages -
- GET /api/messages/{id} - Mnesia
- GET /api/messages/get_by_smsc?smsc=X -
- Mnesia

Mnesia

- MSISDN
-
- SMSC
-
-

2. CDR SQL

SQL



SQL

-
- SQL `cdrs`
- SQL `mysql`/`psql`/`DBBeaver`
-
- - `calling_number` -
 - `called_number` -
 - `message_id` -

- `submission_time` 時間 - 送信時刻
- `status` 状態 - 送信状態
- `dest_smsc` 宛先 - 宛先番号

CDR 形式 送信時刻/送信状態/宛先番号

3. PubSub

- Phoenix PubSub 形式
- 送信時刻
- 送信状態
- 宛先番号
- 送信時刻/送信状態
- WebSocket 形式

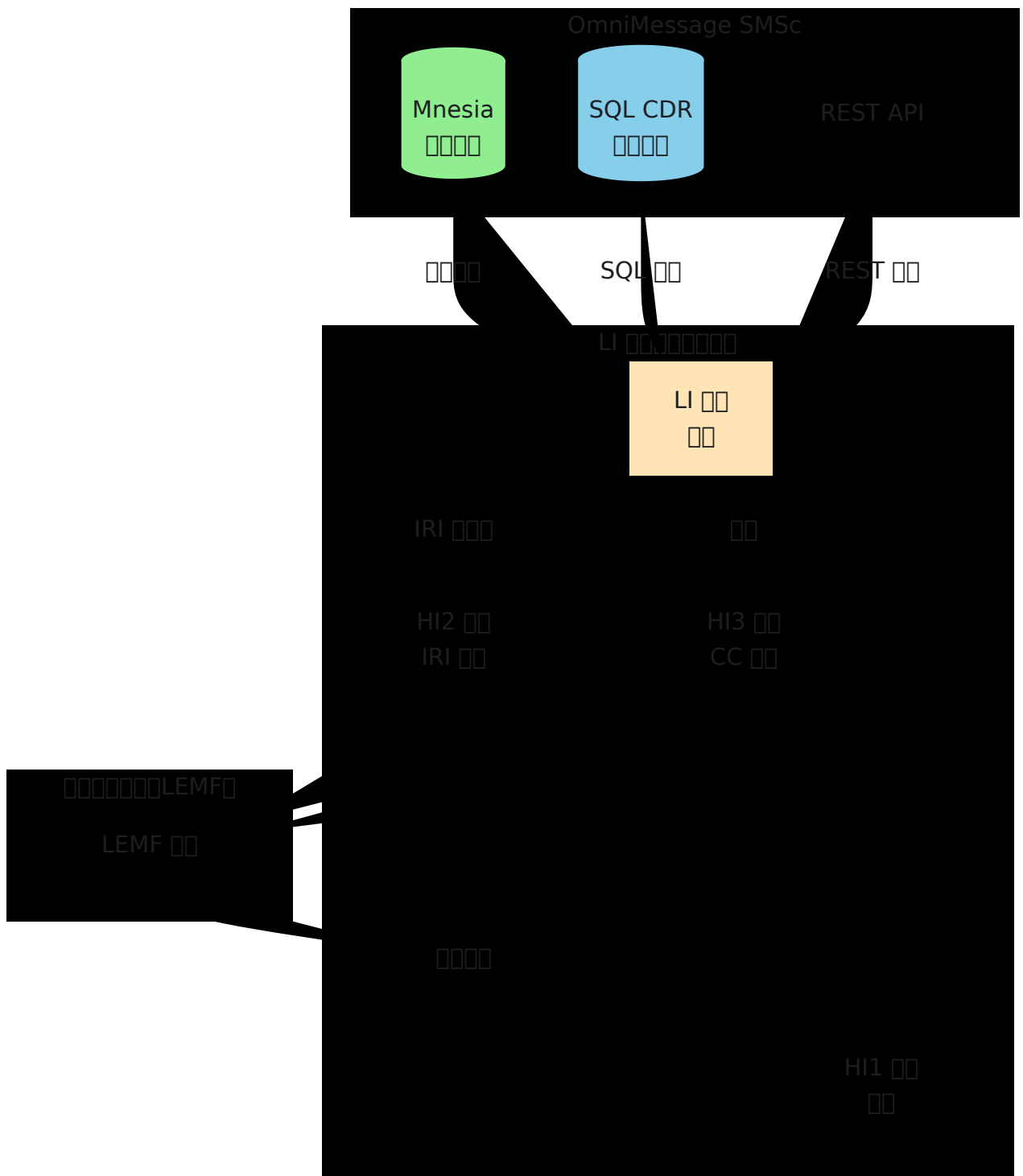
4. 形式

- CDR 形式 CSV
- 送信時刻 JSON
- 送信状態
- 宛先番号
- 送信時刻/送信状態

ETSI 形式

OmniMessage SMSc 形式 ETSI 形式 SMSc 形式 HI1/HI2/HI3 形式
送信時刻/送信状態/宛先番号

ETSI LI 形式



CDR

HI1 CDR - CDR

- CDR CDR
- CDR LEMF → LIMF
- CDR
 - CDR/MSISDNIMSI

- 國際電訊聯盟
- 國際電訊聯盟
- 國際電訊聯盟
- **SMSc** 國際電訊聯盟
 - LIMF 國際電訊聯盟
 - LIMF 國際電訊聯盟 SMSc CDR/API 國際電訊聯盟
 - LIMF 國際電訊聯盟 HI1 國際電訊聯盟

HI2 國際電訊聯盟 - IRI 國際電訊聯盟

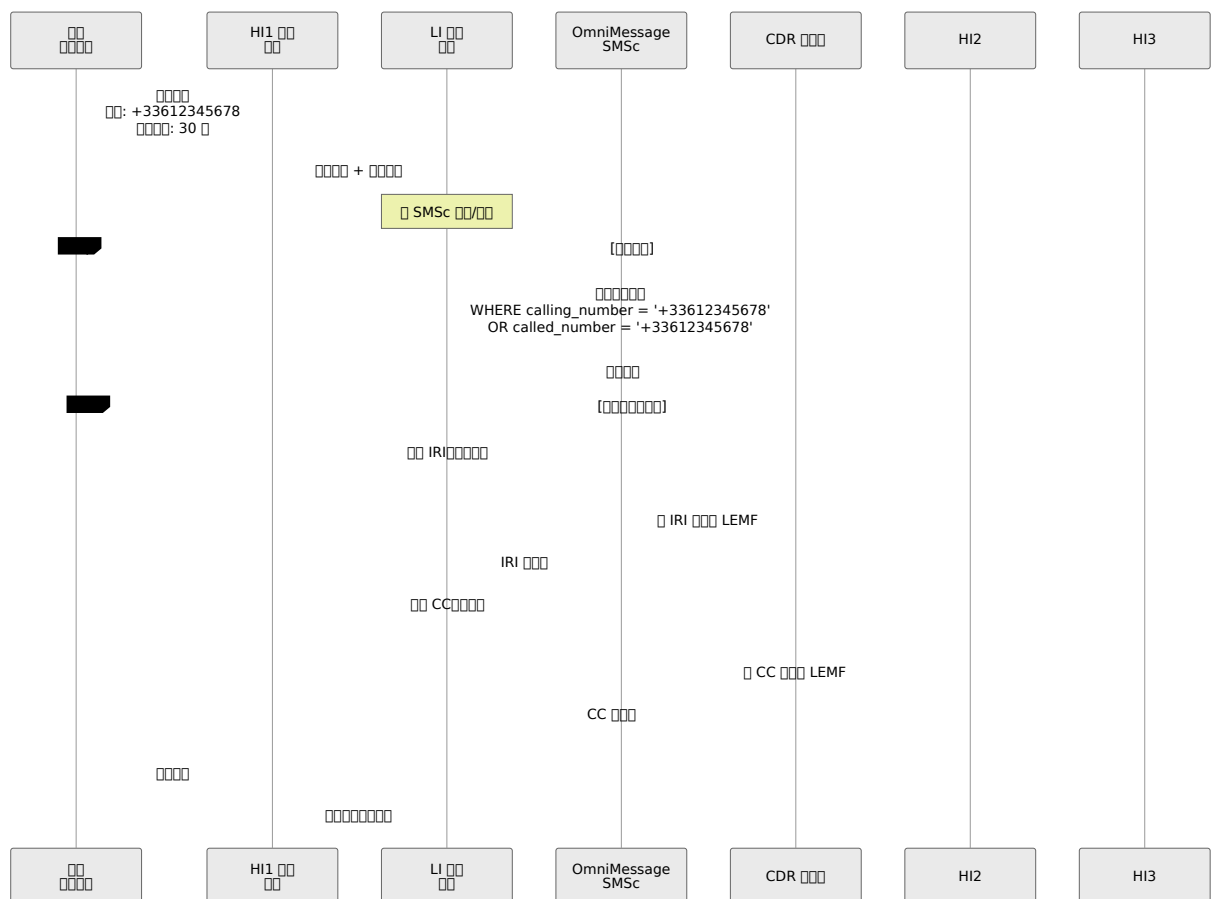
- 國際電訊聯盟
- 國際電訊聯盟 LIMF → LEMF 國際電訊聯盟
- 國際電訊聯盟 國際電訊聯盟 ETSI TS 102 232-x 國際電訊聯盟 XML/ASN.1
- 國際電訊聯盟 **SMSc CDR** 國際電訊聯盟
 - 國際電訊聯盟 ID
 - 國際電訊聯盟 MSISDN 國際電訊聯盟
 - 國際電訊聯盟 MSISDN 國際電訊聯盟
 - IMSI 國際電訊聯盟
 - 國際電訊聯盟
 - 國際電訊聯盟
 - 國際電訊聯盟/國際電訊聯盟/國際電訊聯盟
 - 國際電訊聯盟
 - SMSC 國際電訊聯盟/國際電訊聯盟
 - 國際電訊聯盟
- **SMSc** 國際電訊聯盟
 - LIMF 國際電訊聯盟 CDR 國際電訊聯盟
 - LIMF 國際電訊聯盟 CDR 國際電訊聯盟 ETSI IRI 國際電訊聯盟
 - LIMF 國際電訊聯盟 HI2 國際電訊聯盟 IRI 國際電訊聯盟 LEMF

HI3 國際電訊聯盟 - CC 國際電訊聯盟

- 國際電訊聯盟
- 國際電訊聯盟 LIMF → LEMF 國際電訊聯盟
- 國際電訊聯盟 國際電訊聯盟 ETSI TS 102 232-x
- **SMSc** 國際電訊聯盟

- 000000000000
- 00 PDU0000 SMS 000
- 0000000
- 0000000
- TP-DCS 00
- 000000UDH0
- 0 **SMS** 0000
 - LIMF 0 CDR message_body 00000000
 - LIMF 000000000 PDU 0000
 - LIMF 000000 ETSI CC 00
 - LIMF 00 HI3 0 CC 000 LEMF

000000



SMS 000000 LI 000

SMSc 項目	HI2 (IRI)	HI3 (CC)	CDR 項目
メッセージ ID	発信者 ID	着信者	message_id
発信番号	発信者 A	-	calling_number
着信番号	着信者 B	-	called_number
送信時刻	送信時刻	-	submission_time
配信時刻	配信時刻	-	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)
送信 SMS	送信 SMS	-	source_smsc
受信 SMS	受信 SMS	-	dest_smsc
IMSI	発信者 ID	-	(発信者)

LIMF 項目

1 項目

- LIMF 項目 CDR 項目 1-60 項目
- SQL 項目 HI1 項目
- 項目
- 項目 LI 項目

2 項目

- SMSc PubSub 項目
- LIMF 項目

- LIMF 認證服務
- 認證服務
- 認證服務

第 3 節

- 認證服務 PubSub 認證 < 24 小時
- 認證服務 CDR 認證
- 認證服務認證

認證服務

認證服務

- 認證服務 MSISDN
- 認證 IMSI 認證服務
- 認證服務
- 認證服務 認證
- 認證服務 API 認證

認證服務

- 認證/認證服務
- 認證 SMSC 認證
- 認證服務認證服務
- 認證服務 ENUM 認證

認證服務

- CDR 認證/認證
- 認證
- 認證
- 認證

認證服務 SQL 認證

```
-- 電話番号検索
SELECT * FROM cdrs
WHERE calling_number = '+33612345678'
      OR called_number = '+33612345678'
ORDER BY submission_time DESC;

-- 通話履歴検索
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;

-- 通話履歴検索
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' AND called_number =
'+33687654321')
      OR (calling_number = '+33687654321' AND called_number =
'+33612345678')
ORDER BY submission_time;
```

2. 通話履歴検索

2.1 通話履歴検索

OmniMessage SMSc 通話履歴検索 ANSSI 通話履歴検索

2.2 通話履歴

2.2.1 TLS/SSL 通話

通話履歴

- TLS 1.2 (RFC 5246)
- TLS 1.3 (RFC 8446) - 通話
- SSL 2.0/3.0: 通話履歴

- TLS 1.0/1.1: 不推荐使用

应用

- Erlang/OTP SSL/TLS 不推荐使用
- 使用 TLS 的 Cowboy Web 应用
- Phoenix 使用 HTTPS 应用

配置

配置 Erlang/OTP 不推荐使用

应用 - **TLS 1.3**

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256

应用 - **TLS 1.2**

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

配置

- 使用 ECDHE/DHE 不推荐使用 PFS
- 使用 Diffie-Hellman 使用 2048 位密钥
- 使用椭圆曲线
- 使用 SNI

配置

- 使用 X.509 证书
- RSA 使用 2048 位密钥 4096 位密钥
- 使用 ECDSA
- 使用 SHA-256

- 証明書を作成
- CA 証明

TLS 証明書

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

証明書と **CONFIGURATION.md**

証明書

- REST API 証明書 HTTPS 8443
- Web 証明書 HTTPS 8086
- MySQL/PostgreSQL 証明書 TLS

2.3 証明書

2.3.1 証明書

MySQL/MariaDB 証明書

- 証明書
- AES-256 証明書
- 証明書 TDE

```
-- 証明書 CDR 証明書
ALTER TABLE cdrs ENCRYPTION='Y';
```

PostgreSQL 証明書

- 証明書

- 証明書管理
- 証明書pgcrypto 管理

2.3.2 Mnesia 管理

Mnesia 管理

- 証明書管理
- 証明書管理LUKSdm-crypt
- Erlang VM 証明書管理

2.3.3 証明書管理

証明書管理

- 証明書管理
- 証明書管理0600+ 証明書管理
- 証明書管理
- CDR 証明書管理

証明書管理

- TLS 証明書管理 `priv/cert/`
- 証明書管理
- 証明書管理

2.4 証明書管理

2.4.1 API 管理

REST API 管理

- 証明書 HTTPS/TLS 管理
- 証明書管理SMSc 証明書管理
- IP 証明書管理???
- 証明書管理

証明書管理

- 数据库连接池IP地址
- 数据库连接池
- 数据库连接池90 端口
- 数据库连接池

2.4.2 数据库连接池

数据库连接池

- 数据库/数据库连接池
- TLS/SSL 连接池
- 数据库 IP 地址
- 数据库连接池RBAC

数据库

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

数据库连接池

```
-- 数据库连接池
CREATE USER 'li_readonly'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c.cdrcs TO 'li_readonly'@'%';

-- 数据库连接池
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.cdrcs TO 'analytics'@'%';
```

2.5 安全协议

2.5.1 认证

1 Erlang/OTP 认证

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

2 认证

- 认证
- 认证
- 认证

2.5.2 加密

1 加密

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

2 加密

- 128 加密
- 256 加密

3 加密

- TLS 加密
- 加密
- 加密

2.5.3 密碼

密碼

- RSA 2048 或 4096 位元
- ECDSA
 - P-256 或 P-384 或 P-521
- Ed25519 或 EdDSA

密碼

- TLS
-
-

2.6 SMS

2.6.1 SMS

密碼

- GSM 7 位元 SMS
- UCS-2 或 Unicode 16 位元
- 8 位元
- Latin-1

TP-DCS

-
-
-
-

密碼 **SMS**

- SMS
- SMS 或 SMS
-

2.6.2 〇〇〇〇〇〇

SMPP ☐☐☐ ☒ ☒ ☒ ☐☐☐☐☐

- SMPP 消息/消息
- 通过 TLS 实现 SMPP over TLS
- 消息

IMS

- SIP 100000
- SIP 10000
- IMS 1000000000

SS7/MAP □□□□□□□□

- SS7 □□□□
- MAP □□□□
- SCCP/TCAP □□□

[illegible]

2.7 □□□□□□□□□□

2.7.1 背景

□ □ □ □ □ □ □

- 五五五五五
- 五五五五
- API 五五/五五五
- 五五五五五五
- 五五五五五五

□□□□□

- `stdout/stderr`
- PCAP files

- 監視ツール
- Prometheus

2.7.2 認証

認証

- SMS
- 認証
- 認証/トークン

認証

- TLS
- 認証
- Vault/AWS Secrets Manager
- Erlang/OTP
- API
- IP

認証

- 認証
- 認証
- TLS
- 認証
- 認証

2.8 認証

2.8.1 認証

TLS

```
# 生成RSA 4096 密钥
openssl genrsa -out omnitouch.pem 4096

# 生成证书请求
openssl req -new -key omnitouch.pem -out omnitouch.csr

# 生成证书
openssl x509 -req -days 365 -in omnitouch.csr -signkey
omnitouch.pem -out omnitouch.crt

# 生成证书链 CA 证书
```

配置

- Erlang/OTP CSPRNG 随机数生成器
- `/dev/urandom`
- ID 标识符

2.8.2 配置

配置

- 证书有效期 0600
- 证书 `priv/cert/` 目录
- PEM 格式
- 证书链

配置

- TLS 配置
- 证书链
- API 配置

2.8.3 配置

配置

- 证书 `priv/cert/` 目录
- 证书链

- 使用 ACME 协议 Let's Encrypt

其他功能

- 支持多种加密算法
- TLS 支持 Diffie-Hellman 密钥交换
- 支持多种证书格式

2.9 其他功能

其他功能   支持多种加密算法 SMS 支持多种加密算法

2.9.1 SMS 支持 SS7/MAP 协议

3GPP 和 ETSI 标准

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

SS7 支持

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

SS7/MAP 支持

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

SS7/MAP 支持

- ETSI TS 102 232-1 1000000000 - 1 0000000000

2.9.2 SMS 和 IMS 数据库

3GPP IMS ☐☐☐

- **3GPP TS 24.229** IP **VoLTE** - SIP & SDP

IMS

- **3GPP TS 33.210** 3G ☐ NDS ☐ IP ☐ IPsec ☐

SIP

- **RFC 3428** SIP MESSAGE

IMS □□□□□□

- **3GPP TS 33.107**

2.9.3 SMPP ☐☐☐☐☐

SMPP ☐ ☐ ☐

- **SMPP v3.4** 標準化規格 - 公開
- **SMPP v5.0** 標準化規格 SMPP 5

SMPP 拡張

- **SMPP over TLS** SMPP over TLS
- **SMPP ID** ID
- **IP** IP

標準規格

- **GSM 03.40** ETSI TS 100 901 SMS
- **GSM 03.38** ETSI TS 100 900
- **GSM 04.11** ETSI TS 100 942 SMS

標準規格

- **ITU-T T.50** No. 5
- **ISO/IEC 8859-1** Latin-1
- **ISO/IEC 10646** UCS-2/UTF-16

2.9.4 標準規格

TLS 標準規格

- **NIST SP 800-52** TLS
- **NIST SP 800-131A**
- **RFC 7525** TLS DTLS
- **RFC 8446** TLS 1.3

標準規格

- **FIPS 197** AES
- **FIPS 180-4** SHA-2
- **NIST SP 800-38D** GCM
- **RFC 7539** IETF ChaCha20 Poly1305

標準規格

- **NIST SP 800-57** 管理指南
- **RFC 5280** 定义 X.509 证书和 CRL 格式

2.10 应用

2.10.1 应用

应用

- 认证/授权
- 数据完整性/机密性
- 不可否认性
- 审计
- 使用 GCM 或 Poly1305

2.10.2 应用

应用

- TLS 连接
- 使用 TLS 的 API
- 应用层

应用

- 认证
 - 授权
 - TLS 连接
 - 数据完整性/机密性
 - 审计
-

3. 專案目標

3.1 系統目標

系統目標

- 系統穩定可靠
- CDR 準確無誤
- API 接口 IP/埠
- 系統安全

系統功能

- 系統部署與安裝
- 系統配置與管理
- 系統數據庫 SQL WHERE 查詢
- 系統監控與告警

3.2 系統架構

系統架構

- 系統部署與安裝 24 小時 Mnesia 監控
- CDR 數據庫 6 個月 2 年
- Mnesia 與 SQL 數據庫
- 系統 CDR 數據庫 cron

系統架構

- 系統部署與安裝
- 系統 UI/管理
- 系統數據庫
- 系統數據庫
- 系統數據庫

系統

```
# 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

- 國際電訊聯盟
- 國際電報聯盟

國際電報

IRI國際電報號碼

- CDR 國際電報號碼
 - 國際 ID
 - 國際/國際
 - 國際電報號碼
 - 國際
 - 國際
 - SMSC 國際
 - 國際電報號碼

CC國際電報號碼

- 國際電報號碼
- 國際 PDU 國際
- 國際
- 國際電報號碼

國際電報

```
# 生成 CSV 文件
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 設定

設定

- 設定
 - 8443でHTTPS REST API
 - 8086でHTTPS 管理
- 設定
- 設定 IP 設定
- 設定 DMZ 設定

設定

- 設定
- 設定
- 設定
- 設定Erlang 設定

4.3 設定

設定

- 設定
- 設定
- 設定
- Syslog 設定
- 設定 ELK 設定

設定

- 設定
- 設定
- 設定
- TLS 設定
- 設定

設定

- Prometheus [リンク](#)
- [リンク](#)
- [リンク](#)
- [リンク](#)
- [リンク](#)

[リンク](#)
[リンク](#)
[OPERATIONS_GUIDE.md](#)
[リンク](#)
[METRICS.md](#)

4.4 [リンク](#)

[リンク](#)

- Erlang [リンク](#)
- Mnesia [リンク](#)
- [リンク](#)
- [リンク](#)

[リンク](#)

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

[リンク](#)

- [リンク](#)
 - Mnesia [リンク](#)
 - [リンク](#)
 - [リンク](#)
-

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** - SMS 目次
- **number_translation_guide.md** - 目次
- **METRICS.md** - Prometheus 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

- **ANSSI R226** 目次 [目次]
- 目次 [目次]
- 目次 [目次] GDPR 目次

6. 目次

目次/目次

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

□□□□□□□□

- 00000000
- 00000compliance@omnitouch.com.au

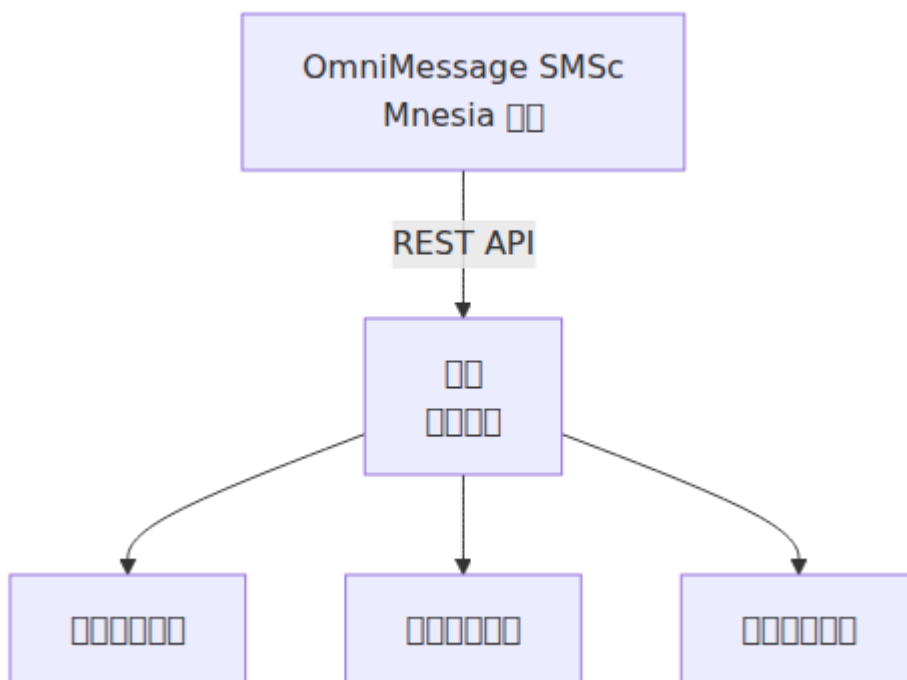
00/000000

- [omnitouch.com.au](#)
- compliance@omnitouch.com.au

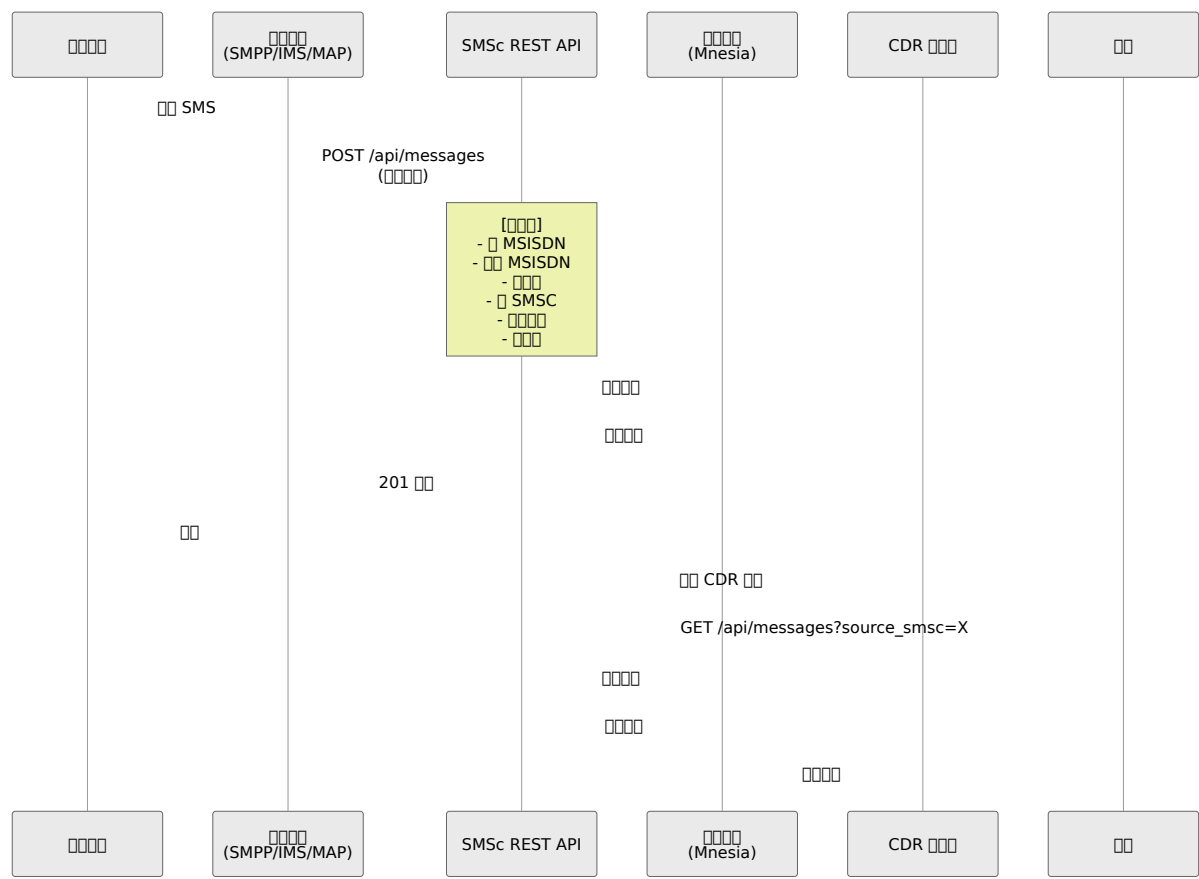
11

Android SMS

A.1 SMS 消息

[illegible]

A.2 SMS



B CDR

OmniMessage SMSc SQL MySQL PostgreSQL

B.1 CDR

欄名	型	注	コメント
id	BIGINT	主キー	自動インクリメント
message_id	BIGINT	メッセージID	自動インクリメント
calling_number	VARCHAR(255)	発信MSISDN	15桁以内
called_number	VARCHAR(255)	着信MSISDN	15桁以内
source_smsc	VARCHAR(255)	発信SMSセンター	15桁以内
dest_smsc	VARCHAR(255)	着信SMSセンター	15桁以内
origin_node	VARCHAR(255)	Erlang ノード名	15桁以内
destination_node	VARCHAR(255)	Erlang ノード名	15桁以内
submission_time	DATETIME	送信日時	YYYY-MM-DD HH:MM:SS
delivery_time	DATETIME	配信日時	YYYY-MM-DD HH:MM:SS
expiry_time	DATETIME	有効期限日時	YYYY-MM-DD HH:MM:SS
status	VARCHAR(50)	送信状態/日時/回数	15桁以内
delivery_attempts	INT	配信回数	15桁以内
message_parts	INT	SMS 分割数	15桁以内
deadletter	BOOLEAN	失敗フラグ	15桁以内
message_body	TEXT	SMS 本文	16000文字以内
inserted_at	DATETIME	CDR 登録日時	YYYY-MM-DD HH:MM:SS
updated_at	DATETIME	CDR 更新日時	YYYY-MM-DD HH:MM:SS

■ ■■■■■■■■ SQL ■■■ CDR_SCHEMA.md

B.2 CDR ■■■■■■■■■■

■■■■■■■■■■■■■■■■■■■■

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

■■■■■■■■■■■■■■■■■■■■

```
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;
```

■■■ CSV ■■■■■■■■

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

B.3 CDR ■■■■■■■■

1. ■■ SQL ■■■

- ■■■■■■■■
- ■■ SQL ■■
- JDBC/ODBC ■■
- ■■■■■■■■■■MySQL Workbench■pgAdmin■

2. REST API

- CDR REST API
- JSON
-
-

3.

- mysql/psql CSV
-
- cron

B.4 CDR

```
# config/runtime.exs
config :sms_c,
  #
  delete_message_body_after_delivery: false, # true

  # / CDR
  cdr_enabled: true,

  # Mnesia CDR
  message_retention_hours: 24
```

- NULL
- MySQL ENCRYPTION='Y'
-
-

C REST API

C.1

□□□□□□

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

□ **SMSC** □□□□

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

□□□□□□

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

□□□□

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

□ □□ **API** □□ **API_REFERENCE.md**

C.2 □□□□

□□□□□

- `source_smsc` - □□□□□□
- `dest_smsc` - □□□□□□□
- □□□□□□□□ CDR □□□□□□
- □□□□□□□ CDR □□□□□□

□□□□□

- □□ CDR □□□ REST API
- □□□□□□□□□□
- □□□□□□
- □□□□□□□□□□

□□ **D**□□□□□□

D.1 TLS □□□□

□□ **TLS** □□□

```
# □□ 4096 □ RSA □□
openssl genrsa -out priv/cert/omnitech.pem 4096

# □□□□□□□□□□
openssl req -new -key priv/cert/omnitech.pem -out
priv/cert/omnitech.csr \
    -subj "/C=FR/ST=IDF/L=Paris/O=Omnitech/CN=smsc.example.com"

# □□□□□□□□□□/□□□
openssl x509 -req -days 365 -in priv/cert/omnitech.csr \
    -signkey priv/cert/omnitech.pem -out priv/cert/omnitech.crt

# □□□□ CSR □□□□□□□□ CA □□□□
```

□□□□□□□□

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

D.2 数据库

MySQL 数据库

```
-- 对 CDR 表加密
ALTER TABLE cdrs ENCRYPTION='Y';

-- 查看表信息
SELECT TABLE_NAME, CREATE_OPTIONS
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'sms_c' AND TABLE_NAME = 'cdrs';
```

PostgreSQL 数据库

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

D.3 部署

部署环境

```
# config/runtime.exs
config :sms_c,
  # 删除消息体在投递后
  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
```

创建视图

```
-- 创建视图
CREATE VIEW cdrs_metadata AS
SELECT
  id, message_id, calling_number, called_number,
  source_smsc, dest_smsc, origin_node, destination_node,
  submission_time, delivery_time, expiry_time,
  status, delivery_attempts, message_parts, deadletter,
  inserted_at, updated_at
FROM cdrs;

-- 授予权限
GRANT SELECT ON cdrs_metadata TO 'analytics'@'%';
```

配置 ETSI

配置 ETSI

- **ANSSI:** 法国国家网络安全局
- **ETSI:** 欧洲电信标准协会
- **3GPP:** 第三代合作伙伴计划 - 移动通信标准
- **IETF:** 互联网工程任务组 - 互联网标准

配置

- **SMSc:** 短信中心 - 负责发送和接收短信
- **SMPP:** 短信点对点协议 - 用于短信网关之间的通信

- **ESME:** 應用層 - SMPP 應用層
- **IMS:** IP 網路層 - 應用層 IP 網路
- **SIP:** 應用層 - IMS 應用層
- **P-CSCF:** 代理客戶端 - IMS 代理客戶端
- **S-CSCF:** 服務客戶端 - IMS 服務客戶端
- **HSS:** 用戶數據庫 - IMS 用戶數據庫
- **SS7/MAP:** 用戶數據庫 7 / 應用層 - 應用層
- **MSC:** 用戶數據庫 - 應用層
- **VLR:** 用戶數據庫 - 應用層
- **GT:** 用戶數據庫 - SS7 用戶數據庫
- **SCCP:** 用戶數據庫 - SS7 用戶數據庫
- **TCAP:** 用戶數據庫 - SS7 用戶數據庫
- **MSISDN:** 用戶數據庫 - 用戶數據庫
- **IMSI:** 用戶數據庫 - 用戶數據庫
- **E.164:** 用戶數據庫
- **ENUM:** E.164 用戶數據庫 - 用戶數據庫 DNS 用戶數據庫 URI 用戶數據庫
- **PDU:** 用戶數據庫 - 用戶數據庫
- **TP-DCS:** 用戶數據庫 - 用戶數據庫
- **TP-DU:** 用戶數據庫 - 用戶數據庫 PDU 用戶數據庫
- **UDH:** 用戶數據庫 - 用戶數據庫/用戶數據庫
- **TON/NPI:** 用戶數據庫 / 用戶數據庫 - 用戶數據庫
- **GSM7:** GSM 7 用戶數據庫 - 用戶數據庫
- **UCS-2:** 用戶數據庫 2 用戶數據庫 - 用戶數據庫 Unicode 用戶數據庫

應用層

- **Mnesia:** Erlang 用戶數據庫 - 用戶數據庫
- **CDR:** 用戶數據庫 - 用戶數據庫
- **REST API:** 用戶數據庫 - 用戶數據庫 HTTP 用戶數據庫 API 用戶數據庫
- **Phoenix:** Elixir 用戶數據庫
- **Cowboy:** Erlang HTTP 用戶數據庫
- **Ecto:** Elixir 用戶數據庫
- **PubSub:** 用戶數據庫-用戶數據庫

標準

- **LI:** 標準 - 標準化
- **LIMF:** 標準 - 標準化
- **LEMF:** 標準 - 標準化
- **IRI:** 標準 - 標準化/標準化
- **CC:** 標準 - 標準化
- **HI1 標準:** ETSI LI 標準 - 標準化
- **HI2 標準:** ETSI LI 標準 IRI 標準 - 標準化
- **HI3 標準:** ETSI LI 標準 CC 標準 - 標準化
- **R226:** 標準 R226-3 標準 R226-7 標準
- **ETSI:** 標準 - 標準 LI 標準
- **ETSI TS 102 232:** 標準

標準

- **MT:** 標準 - 標準化
- **MO:** 標準 - 標準化
- **DLR:** 標準 - 標準化
- 標準: 標準
- 標準: 標準2標準4標準8標準

標準

- **TLS:** 標準 - 標準
- **PFS:** 標準 - 標準化
- **AES:** 標準
- **RSA:** Rivest-Shamir-Adleman - 標準
- **ECDSA:** 標準
- **SHA:** 標準
- **X.509:** 標準
- **CA:** 標準
- **CSPRNG:** 標準

標準

- **MySQL:** 数据库
 - **PostgreSQL:** 数据库
 - **TDE:** 透明数据加密
 - **RBAC:** 基于角色的访问控制
-

版本 1.0
日期 2025-11-29
符合 ANSSI R226 标准
数据库 名称 - 表

API /

←

SMSC

	MSISDNJSON CSV
	+

HTTP API Mnesia SMSC

- (list_id) - MSISDN
- list_id - MSISDN

var1, var2, var3, var4 - {{ var1 }}...{{ var4 }} msisdn

メッセージング - 個人化されたメッセージを送信するための方法

CSV ファイル - MSISDN (msisdn/destination_msisdn/number/phone) と変数 (var1...var4) を含む

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

JSON ファイル - 個人化されたメッセージを送信するための方法

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

メッセージ

Hi {{ var1 }}, your {{ var2 }} plan renews soon. Reply STOP to opt out.

12025550101 var1=Alice, var2=Gold Hi Alice, your Gold plan renews soon. Reply STOP to opt out.

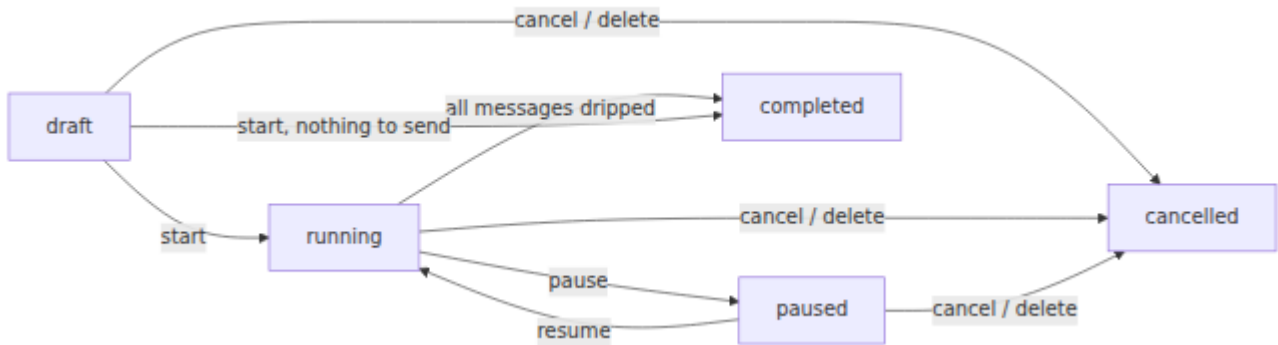
メッセージ

audience (オーディエンス) を指定する

メッセージ	オーディエンス
active_only (アクティブのみ) メッセージ	メッセージを送信するオーディエンス skipped_inactive (スキップされた非アクティブ)
all (すべて)	すべてのオーディエンス

campaigns table - campaigns table has an audience_id foreign key to the audience table
active_only

campaigns



1. **draft** - initial state
2. **start** - transition to **running** state
3. **running** - active state
4. **pause / resume** - toggle between running and paused
5. **completed** - final state after all messages are sent
6. **cancelled** - final state after cancellation

campaigns

campaigns table - campaigns table has an audience_id foreign key to the audience table
campaign_retention_hours 7

campaigns

drip_tps - drip rate per second
drip_tps × interval - drip rate per second multiplied by interval

campaigns

stats GET /api/campaigns/:id

項目	説明
total	メッセージの総数
pending	待機中メッセージ
queued	キューイング中メッセージ
delivered	配信済みメッセージ
failed	配信失敗メッセージ/割合
skipped_inactive	スキップされた無効なメッセージ
dispatched	queued + delivered + failed
progress_percent	pending メッセージの割合
delivery_percent	配信率

メッセージの配信状況を確認する SMSC には、メッセージの配信状況を確認するための API が用意されています。

GET /api/campaign_targets/:id?state=failed

設定

```
config :sms_c, :campaigns,
  # 配信機能の有効化/無効化
  enabled: true,
  # 配信機能の起動
  drip_tick_ms: 1000,
  # 配信機能の drip_tps のデフォルト値
  default_drip_tps: 50,
  # 配信機能の validity_hours のデフォルト値
  message_validity_hours: 72,
  # 配信機能の campaign_retention_hours のデフォルト値
  campaign_retention_hours: 168
```

定时任务

定时任务

- **validity_hours** - 消息的有效性时间 expires 消息的有效性时间 message_validity_hours
- **deliver_after** - 消息的 ISO8601 时间戳

定时任务 drip_tps 在 // 消息的有效性时间 delta = deliver_after - started_at 时间 t 消息的有效性时间 send_time/deliver_after = t + delta expires = t + delta + validity_hours drip_tps 消息的有效性时间 deliver_after 消息 deliver_after 消息 drip_tps 消息 - 消息的有效性时间 deliver_after delta = 0 消息的有效性时间

API 操作

```
# 1. 创建营销活动列表 (CSV 转 JSON)
curl -k -X POST https://smc:8443/api/campaign_lists \
  -H 'Content-Type: application/json' \
  -d '{"name": "VIP customers",

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

# 2. 创建营销活动
curl -k -X POST https://smc:8443/api/campaigns \
  -H 'Content-Type: application/json' \
  -d '{"name": "Renewal notice", "message": "Hi {{ var1 }}, your {{ var2
renews soon.",
      "source_msisdn": "12345", "source_smc": "IMS_SMC",
      "list_id": 1, "audience": "active_only", "drip_tps": 50}'
# => {"id": 7, "status": "draft", ...}

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

# 4. 获取 / 更新
curl -k https://smc:8443/api/campaigns/7
# => {... "stats":
{"total": ..., "delivered": ..., "failed": ..., "progress_percent": ...}}
```

在 2 个 `list_id` 中

API 操作

- API 操作
- SMS 操作 - 发送/接收
- 操作

CDR

←

CDR

-
-
-
- SQL
-
-
-
-
-

cdrs SMS CDR delivered expired failed rejected balance_rejected rejected Mnesia CDR

CDR Mnesia

-
-
-
- Mnesia



MySQL / MariaDB

```
CREATE TABLE cdrs (  
  id BIGINT AUTO_INCREMENT 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  
  source_smsc VARCHAR(255),  
  dest_smsc VARCHAR(255),  
  
  -- 节点  
  origin_node VARCHAR(255),  
  destination_node VARCHAR(255),  
  
  -- 时间  
  submission_time DATETIME NOT NULL,  
  delivery_time DATETIME,  
  expiry_time DATETIME,  
  
  -- 状态  
  status VARCHAR(50) NOT NULL,  
  delivery_attempts INT DEFAULT 0,  
  message_parts INT,  
  deadletter BOOLEAN DEFAULT FALSE,  
  
  -- 消息内容  
  message_body TEXT,  
  
  -- 时间戳  
  inserted_at DATETIME NOT NULL,  
  updated_at DATETIME NOT NULL,  
  
  -- 索引  
  UNIQUE KEY cdrs_origin_node_message_id_index (origin_node,
```

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

(origin_node, message_id) PRIMARY KEY, CDR PRIMARY KEY, CDR
ON CONFLICT DO NOTHING

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  
    source_smsc VARCHAR(255),  
    dest_smsc VARCHAR(255),  
  
    -- 节点  
    origin_node VARCHAR(255),  
    destination_node VARCHAR(255),  
  
    -- 时间  
    submission_time TIMESTAMP NOT NULL,  
    delivery_time TIMESTAMP,  
    expiry_time TIMESTAMP,  
  
    -- 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,  
  
    -- 时间戳  
    inserted_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);  
  
-- 索引  
CREATE UNIQUE INDEX cdrs_origin_node_message_id_index ON  
cdrs(origin_node, message_id);  
CREATE INDEX idx_cdrs_message_id ON cdrs(message_id);
```

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

Table 1

Table 2

Field	Type	PK	Description
id	BIGINT	NO	CDR ID

Table 3

Field	Type	PK	Description
message_id	BIGINT	NO	SMS-C Mnesia (origin_node, message_id) - SMS-C Mnesia

Table 4

Field	Type	PK	Description
calling_number	VARCHAR(255)	NO	MSISDN E.164+15551234567
called_number	VARCHAR(255)	NO	MSISDN E.164+15551234567

SMSC

source_smsc	VARCHAR(255)	YES	SMSC API SMSC NULL
dest_smsc	VARCHAR(255)	YES	SMSC NULL

origin_node	VARCHAR(255)	YES	Erlang "sms@node1.example.com"
destination_node	VARCHAR(255)	YES	Erlang NULL

UTC

欄名	データ型	Nullable	コメント
submission_time	DATETIME	NO	SMS-C送信日時
delivery_time	DATETIME	YES	SMS-C送信完了日時 NULL
expiry_time	DATETIME	YES	SMS-C送信有効期限 NULL

計算式

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

フィールド

欄名	データ型	Nullable	コメント
status	VARCHAR(50)	NO	送信状態 delivered expired failed rejected
delivery_attempts	INT	NO	送信回数 0 0-255
message_parts	INT	YES	SMSメッセージ部分数 1 2 NULL
deadletter	BOOLEAN	NO	死文字列フラグ TRUE FALSE

備考

ステータス	説明	データ型	Nullable
delivered	正常に配信された	boolean	NO
expired	有効期限が切れた	boolean	NULL
failed	配信に失敗した	boolean	NULL
rejected	拒否された	boolean	NULL

メッセージ

フィールド	データ型	Nullable	説明
message_body	TEXT	YES	SMSメッセージの本文 delete_message_body_after_delivery: trueの場合、 メッセージはNULLで保存され、TEXTフィールドは 65,535文字に制限される

メッセージID

- メッセージIDはCDRメッセージ/メッセージID
- delete_message_body_after_delivery: trueの場合、メッセージIDはNULL
- メッセージIDは一意である

メッセージ時刻

フィールド	データ型	Nullable	説明
inserted_at	DATETIME	NO	CDRメッセージがデータベースに挿入された時刻 delivery_time/expiry_time
updated_at	DATETIME	NO	CDRメッセージがデータベースで更新された時刻 inserted_at

SQL

練習

データベースからCDR

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

データベースから

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

データベースから

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

練習

データベースからSMSC

```

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;

```

データベースからCDRを削除

```

-- データベースからCDRを削除
SELECT COUNT(*) FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR);

-- データベースからCDRを削除
DELETE FROM cdrs
WHERE submission_time < DATE_SUB(NOW(), INTERVAL 2 YEAR)
LIMIT 10000; -- データベースからCDRを削除

```

データベース

データベースからCDRを削除

```
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;
```

索引

索引名称

索引名称	索引	说明
PRIMARY	id	主键
cdrs_origin_node_message_id_index	origin_node, message_id	按节点和消息ID索引，用于快速检索CDR记录
idx_cdrs_message_id	message_id	按消息ID索引，用于快速检索CDR
idx_cdrs_calling_number	calling_number	按主叫号码索引
idx_cdrs_called_number	called_number	按被叫号码索引
idx_cdrs_status	status	按状态索引
idx_cdrs_submission_time	submission_time	按提交时间索引
idx_cdrs_dest_smsc	dest_smsc	按目的地短消息服务中心索引

创建索引

创建索引

创建索引

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

创建索引

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

创建索引

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

创建索引MySQL

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

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

创建索引

创建索引

欄名	MySQL/MariaDB	PostgreSQL	説明
id	BIGINT AUTO_INCREMENT	BIGSERIAL	64ビット整数
message_id	BIGINT	BIGINT	64ビット
メッセージ	VARCHAR(255)	VARCHAR(255)	最大255文字 の文字列
message_body	TEXT	TEXT	MySQLは 65,535文字 PostgreSQLは 最大
時刻	DATETIME	TIMESTAMP	UTC時刻
フラグ	INT	INTEGER	32ビット整数
有無	BOOLEAN (TINYINT(1))	BOOLEAN	MySQLは0/1

インストール

CDRデータベースをインストールする

1. インストール

config/runtime.exsで設定する

```

config :sms_c,
  # 删除消息体
  delete_message_body_after_delivery: true,

  # UI隐藏消息体
  hide_message_body_in_ui: true,

  # 导出隐藏消息体
  hide_message_body_in_export: true

```

2. 数据脱敏

数据脱敏需求

```

-- 数据脱敏需求4
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. 数据加密

数据加密需求

MySQL

```

-- 数据加密
ALTER TABLE cdrs ENCRYPTION='Y';

```

PostgreSQL PostgreSQL支持透明数据加密(TDE)

4. 数据库

数据库CDR数据库

```
-- 数据库用户
CREATE USER 'billing_ro'@'%' IDENTIFIED BY 'secure_password';
GRANT SELECT ON sms_c.cdcs TO 'billing_ro'@'%';

-- 数据库表
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.cdcs TO 'analytics'@'%';
```

数据库

数据库

数据库

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

数据库

1. 数据库MySQL 8.0+ PostgreSQL 11+

```
-- MySQL
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
);

-- 
ALTER TABLE cdrs DROP PARTITION p202510;
```

2.

```
-- CDR
CREATE TABLE cdrs_archive LIKE cdrs;

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

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

3.

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

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

#
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

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

```
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)
);
```

```
-- 测试数据
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

CDR000000000000

```
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□□□□□□□□

SMS-CDR RFC 4180 CSV SQL

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

```

[[:since[:until]]]DateTime[:status[:delivered,
:failed]][:limit] {[:ok, {path, row count}]} message body

```

hide_message_body_in_export 是 - 是否 true 是 [REDACTED] 是否
是否CSV是否

是SQL是否

```
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
```

是

- 是 - 是CDR是否
- 是 - CDR是否
- API是 - 是REST API是否CDR

SMS-C 目录

← 目录

SMS-C 目录

目录

- 目录
- 目录
- API 目录
- Web UI 目录
- 目录
- 目录
- 目录
- Diameter Sh (HSS) 目录
- ENUM 目录
- 目录
- 目录 (X1/X2/X3)
- 目录
- 目录
- 目录
- 目录

目录

SMS-C 目录

config/config.exs

目录

- 目录

- 資料庫
- 表/欄位
- 查詢

config/runtime.exs

資料庫設定

- 資料庫
- 表
- 欄位 (OCS, ENUM)
- 查詢
- 設定

config/prod.exs (測試)

測試設定

測試時 `runtime.exs` 會自動啟用 API 測試

SQL CDR 測試

SMS-C 測試 **Mnesia** 資料庫測試 **SQL** 測試 CDR 測試

測試 SQL 測試

測試 SQL 測試 CDR 測試

DB	Version	Adapter	Count	Notes
MySQL	8.0+	Ecto.Adapters.MyXQL	3306	Default database
MariaDB	10.5+	Ecto.Adapters.MyXQL	3306	MySQL compatible
PostgreSQL	13+	Ecto.Adapters.Postgres	5432	Supports JSON

Mnesia database is used for storing CDR records in SQL format

CDR Export

CDR Export SQL Query

1. Filter by status - `delivered`, `expired`, `failed`, `rejected`, `balance_rejected`
2. Filter by Mnesia CDR status - `expired`, `failed`, `rejected`

CDR Export SQL Query (origin_node, message_id) ON CONFLICT DO NOTHING

message_id Mnesia message_id SMS-C CDR (origin_node, message_id) CDR enabled: false

CSV / Export

CDR Export SQL Query RFC 4180 CSV mix

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

from :since until DateTime :status :limit
message_body hide_message_body_in_export [REDACTED]

MySQL / MariaDB

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

PostgreSQL

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

SQL

MySQL/MariaDB -

- CDR
-
-
-

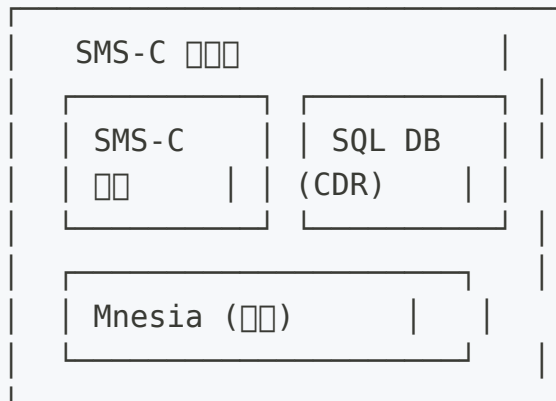
PostgreSQL - 1000000

- 空間データを JSON/JSONB で
- 1 CDR を保存する
- 空間 PostgreSQL を使う
- 空間データを PostGIS

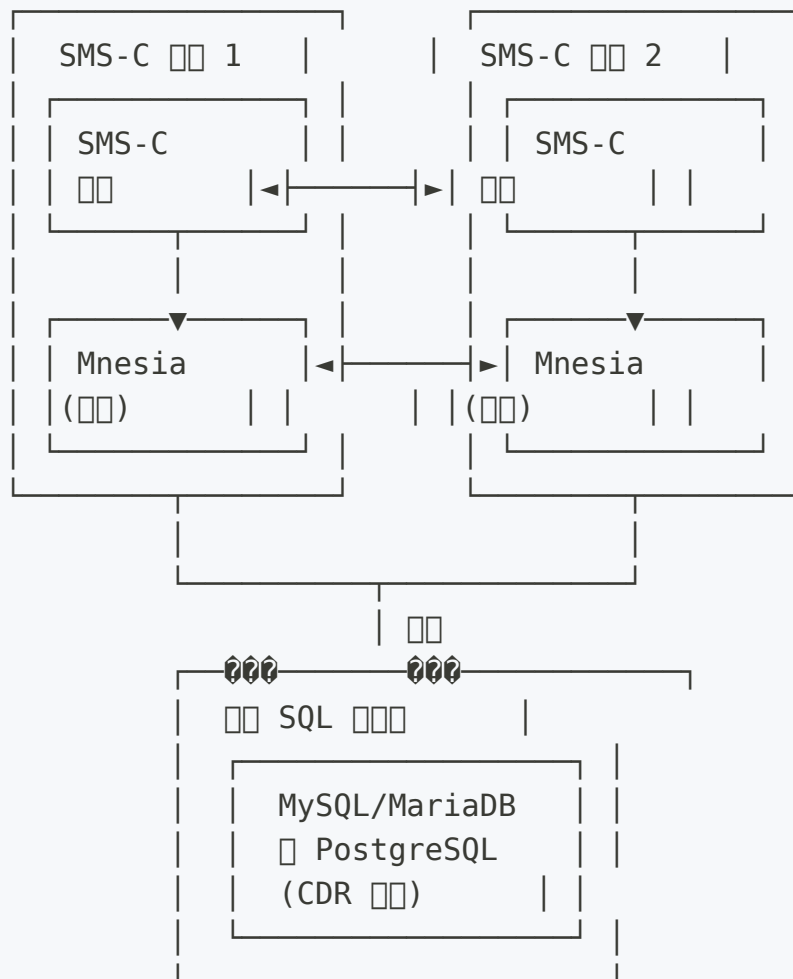
□ □ □ □

SQL CDR SMS-C

□□□□□□□□/□□□□



□□◆?◆□□□□□ - □□□□



SQL 数据库

- 数据库 CDR 数据库
- 数据库
- 数据库 SMS-C 数据库
- 数据库 SMS-C 数据库 CDR 数据库
- 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
テスト (< 100 msg/sec)	10-15	テスト
本番 (100-1000 msg/sec)	20-30	本番
本番 (> 1000 msg/sec)	40-100	本番

接続数 `pool_size = (データベース 接続数) * 1.5`

データベース接続

データベース接続

```
# データベース接続
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
```

データベース接続

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

API □□□□ `:api_ex` □□□□□□ `:api` □□□□□□□□□□ `routes:` □□ `config/config.exs` □□□□□□
□□□□□□□□□□□□

項目	型	必須	デフォルト値	説明
port	整数	必須	8443	REST API の TCP ポート
listen_ip	IP アドレス	必須	0.0.0.0	API リッスンする IP アドレス 0.0.0.0 はすべての IP アドレスをリッスンします
enable_tls	ブール値	必須	true	TLS を有効にするかどうか HTTPS 経由で API を呼び出す
tls_cert_path	ファイルパス	enable_tls が true の場合必須	priv/cert/omnitouch.crt	TLS 証明書ファイルのパス
tls_key_path	ファイルパス	enable_tls が true の場合必須	priv/cert/omnitouch.pem	TLS 私鍵ファイルのパス

TLS/SSL 設定

以下は TLS 設定の例です。

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

🔒 TLS 🚀🚀🚀

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

API 🚀🚀

🔑🔑🔑🔑🔑🔑🔑

```
# 🔑🔑🔑🔑  
mkdir -p priv/cert  
  
# 🔑🔑  
openssl genrsa -out priv/cert/server.key 2048  
  
# 🔑🔑🔑🔑🔑  
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"  
  
# 🔑🔑🔑🔑🔑🔑🔑 365 📅  
openssl x509 -req -days 365 -in priv/cert/server.csr \  
  -signkey priv/cert/server.key -out priv/cert/server.crt  
  
# 🔑🔑  
chmod 600 priv/cert/server.key  
chmod 644 priv/cert/server.crt
```

🔑🔑🔑🔑🔑🔑🔑 CA🔑Let's Encrypt🔑🔑 CA 🔑🔑🔑🔑🔑

API 🚀🚀

IP 🔑🔑🔑🔑🔑🔑🔑

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

Web UI Phoenix TLS https:

tls_cert_path/tls_key_path config/config.exs force_ssl HTTPS

項目	種類	単位	値	説明
port	整数	ポート	443	HTTPS 接続ポート
keyfile	文字列	ファイル名	priv/cert/omnitouch.pem	TLS 秘密鍵ファイル
certfile	文字列	ファイル名	priv/cert/omnitouch.crt	TLS 証明書ファイル

Web UI

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

インストール

Ngix Apache 環境構築

Ngix 設定

```

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 支持 HTTP 和 DNS SRV 以及 HTTPS 连接

--	--	--	--

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

配置参数

参数	类型	默认值	描述
enabled	布尔	false	是否启用
dns_srv_domain	字符串	" "	用于 DNS SRV 记录的域名
dns_poll_interval_ms	整数	30000	DNS 轮询间隔 (毫秒)
health_check_interval_ms	整数	10000	健康检查间隔 (毫秒)
registry_sync_interval_ms	整数	15000	注册表同步间隔 (毫秒)
forward_retry_interval_ms	整数	5000	转发重试间隔 (毫秒)
forward_max_retries	整数	50	转发最大重试次数
http_timeout_ms	整数	5000	HTTP 请求超时 (毫秒)
api_port	整数	8443	API 服务的 URL 端口
static_peers	列表	[]	静态对等节点的 DNS SRV 记录

防火墙

在 8443 端口上启用 HTTPS 监听 Erlang 的 4369 + 9100-9200 端口

```
# 配置 iptables 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

部署步骤

□□□□

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

□□□□

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

□□□□□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□

□□□□ = $2^{(\square\square\square\square)}$ □□

階数	必要メモリ
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` 10分

設定方法

```
# 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

設定

ここでは **Diameter Ro** を実装するための RFC 4006 を実装するための SMS-C の OCS を実装するための **CC-Request-Type 4** を実装するための OCS を実装するための Diameter を実装するための **Diameter Sh** - **:ro** を実装するための

設定

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

config :sms_c, :charging,
  # 設定
  # :reject - 拒否 :balance_rejected 拒否
  # :auto_reply - SMS out_of_balance_reply_text 拒否
  out_of_balance_action: :reject,
  out_of_balance_reply_text: "拒否",
  # OCS 拒否 Diameter 拒否
  # 拒否
  # :allow - 拒否
  # :reject - 拒否
  on_charging_error: :allow,
  # OCS 拒否 CCR
  # Service-Context-Id 拒否
  rating_profile: "sms@sms-c"
```

ここでは Diameter を実装するための **:ro** を実装するための IETF を実装するための 0

```

config :sms_c,
  diameter_enabled: true

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

```

~~~~~

~~~~~ **charged** ~~~~~

| charged | ?? |
|----------------|------------------------------|
| :yes | ~~~~~ OCS ~~~~~ |
| :no | ~~~~~ |
| :default | ~~~ default_charging_enabled |

~~~~~

```

config :sms_c,
  default_charging_enabled: false

```

default OCS charged: :yes

config :sms\_c, :charging - out\_of\_balance\_action:  
:reject rating\_profile: "sms@sms-c"  
default\_charging\_enabled false

on\_charging\_error: :allow HTTP

- Diameter
- OCS

on\_charging\_error: :reject - OCS  
out\_of\_balance\_action

OCS out\_of\_balance\_action  
:balance\_rejected

## 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: "smsc01",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{vendor_id: 10415, auth_application_id: 16_777_217,
acct_application_id: nil}
        ]
      }
    ],
    peers: [
      %{
        host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
        ip: "10.0.0.1",
        port: 3868,
        realm: "epc.mnc001.mcc001.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
}

```





ENUM/ENUM

```
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+)$",          # 00 00
  calling_replace: "+\1",              # 000 +
  called_match: "^00(\d+)$",
  called_replace: "+\1",
  priority: 10,
  description: "00 00000000 +",
  enabled: true
}
```

00000000

```
%{
  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: "000000000000",
  enabled: true
}
```

00000000

00000000

```
%{
  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

XXXXXXXXXXXXXXXXXXXX



Table 1

Interface	Protocol	OmniMessage Protocol	Port	Security
X1	HTTP	HTTP	HTTPS or TLS	Optional
X2	HTTP	HTTP	TCP or TLS	Optional
X3	HTTP	HTTP	TCP	Optional SMS or Optional

Table 2

Interface	Protocol
ETSI TS 103 221-1	Optional X1 or Optional
ETSI TS 103 221-2	Optional X2/X3 or Optional
3GPP TS 33.128	Optional or Optional

Figure 1



workload size, see the topology, and get pricing — no sales call needed.

[Size my network](#)

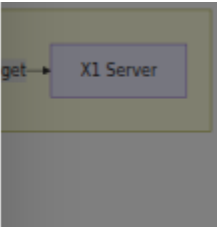


Figure 2

config :sms\_c, :li config/runtime.exs  
enabled: true

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

□□□□

項目	型別	デフォルト値	説明
enabled	boolean	false	メッセージングが有効かどうか。falseの場合はメッセージングが行われません。
ne_identifier	string	"sm-sc-01"	TS 103 221-2 の 18桁の OmniMessage の X2 の識別子。LEMF の IP の識別子。
x1	array	[]	X1 の X1 の識別子。
x2	array	[]	X2 の X2 の識別子。
x3	array	[]	OmniMessage の X3 の識別子。[enabled: false]

X1

OmniMessage X1 LEMF TLS TS 103 221-1 7.2.2.1 allow\_insecure: true

enabled			false	X1
port			8443	X1 TCP
tls			-	X1 TLS X1 TLS
allow_insecure			false	HTTP X1

X1 TLS

certfile			-	PEM
keyfile			-	PEM
cacertfile			-	TLS TLS

X2

OmniMessage X2 LEMF SMS TS 33.128 xIRI SMS

項目	項目	項目	項目	項目
<code>enabled</code>	項目	項目	<code>false</code>	項目 X2 項目
<code>address</code>	項目	項目	-	LEMF X2 項目 IP 項目
<code>port</code>	項目	項目	-	LEMF X2 項目 TCP 項目
<code>tls</code>	項目 項目	項目	<code>false</code>	項目 <code>true</code> 項目 TLS 項目 TLS 項目 X2項目 <code>false</code> 項目 TCP

項目

OmniMessage 項目  
OmniMessage 項目OmniMessage 項目

項目

項目 **ANSI R226** 項目

項目

項目 **SMS** 項目



## 配置短信

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

    # 国际短信
    %{
      calling_regex: nil,
      called_regex: ~r/^\+44/,
      source_smsc: nil,
      dest_smsc: "uk_gateway_1",
      source_type: nil,
      enum_domain: nil,
      enum_result_domain: nil,
      action: :route,
      auto_reply_message: nil,
      charged: :default,
      originating_network: :any,
      terminating_network: :any,
      weight: 70,
      priority: 50,
      description: "国际短信70%",
      enabled: true
    }
  ]
```

```

},
%{
  calling_regex: nil,
  called_regex: ~r/^\+44/,
  source_smsc: nil,
  dest_smsc: "uk_gateway_2",
  source_type: nil,
  enum_domain: nil,
  enum_result_domain: nil,
  action: :route,
  auto_reply_message: nil,
  charged: :default,
  originating_network: :any,
  terminating_network: :any,
  weight: 30,
  priority: 50,
  description: "☐☐☐☐☐☐☐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,
  enum_result_domain: nil,
  action: :route,
  auto_reply_message: nil,
  charged: :default,
  originating_network: :any,
  terminating_network: :on_net,
  weight: 100,
  priority: 50,
  description: "☐☐☐ X - ☐☐☐☐☐☐☐",
  enabled: true
}
]

```

## ❓❓ 確認事項

```
# 確認事項 Web UI 確認
config :sms_c,
  sms_routes: []
```

確認事項

確認事項

確認事項

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

確認事項

確認事項	確認事項	確認	確認	確認
確認	200	200ms	~5,000 msg/sec	確認 200ms
確認	100	100ms	~4,500 msg/sec	確認 100ms
確認	50	20ms	~3,000 msg/sec	確認 20ms
確認	10	10ms	~1,500 msg/sec	確認 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 omnimessage
    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
```

批量插入

批量插入 < 20ms

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

# 数据库配置
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

# 配置 Diameter Ro 接口
# 配置 :charging + :diameter_ex 接口
config :sms_c,
  default_charging_enabled: true
```

配置数据库

配置租户

```
# 租户 1 配置
export DB_NAME=sms_c_tenant1
export NODE_NAME=sms_tenant1@node1.example.com
# 配置 OCS 接口 :charging rating_profile 接口

# 租户 2 配置
export DB_NAME=sms_c_tenant2
export NODE_NAME=sms_tenant2@node1.example.com
```

配置接口

配置接口

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

部署

部署到生产环境

```
# 编译
mix compile

# 创建数据库
mix ecto.create
mix ecto.migrate

# 配置 Diameter Ro 与 OCS 交互
# :diameter.service_info(:omnimessage, :connections)

# 启动服务
iex -S mix phx.server
```

部署到生产环境

部署到生产环境



項目	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

## インストール手順

- 依存パッケージをインストール
- API リスナーを設定
- データベース接続を設定
- Web UI を設定
- ノード名を設定
- Erlang Cookie を設定
- サービスを開始
- ログを確認
- ヘルプを表示
- 再起動

10. 配置 数据库

数据库配置

配置	数据库	配置
数据库名称	数据库名称	数据库名称
数据库用户	用户/密码	用户 DB_* 密码
API 配置	用户/IP 配置	用户 API_PORT 用户 listen_ip
数据库配置	Cookie 配置	用户 ERLANG_COOKIE配置 4369, 9100-9200
配置	OCS/DRA 配置	用户 Diameter Ro 配置
ENUM 配置	DNS 配置	用户 DNS 配置
配置	配置	配置
配置	配置	用户 sms_routes 用户 Web UI

数据库配置 数据库

数据库配置Mnesia

配置

配置 Mnesia 配置

```

config :sms_c,
  # [] Mnesia []
  # [] 24[] runtime.exs [] 168[] 
  message_retention_hours: 168,

  # []
  retention_check_interval_minutes: 60

```

[]

- [] 24-72 []
- [] 4-8 []
- [] 12-24 []

[]

- []~1KB
- 10,000 []~10MB
- 100,000 []~100MB

## CDR[]

[]CDR [] Ecto []

```

config :sms_c,
  # []/[] CDR []false
  cdr_enabled: true

```

## CDR []

- [] ID[]/[]
- []/[] SMSC
- []/[]
- []
- []
- []

環境構築

- 必要なパッケージ CDR
- 必要な依存関係

インストール

必要なパッケージをインストール

```
config :sms_c,  
  # Mnesia 関連  
  delete_message_body_after_delivery: false,  
  
  # Web UI 関連  
  hide_message_body_in_ui: false,  
  
  # CSV 関連  
  hide_message_body_in_export: false
```

確認

項目	内容
delete_message_body_after_delivery: true	Mnesia 関連
hide_message_body_in_ui: true	Web UI 関連
hide_message_body_in_export: true	CSV 関連

確認

インストール完了

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

配置项

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

配置项

配置项

```
[info] 配置项Mnesia配置24h  
[info] CDR 配置  
[info] 配置项  
[info] OCS 配置url: http://..., tenant: ...
```

配置项

--	--	--	--

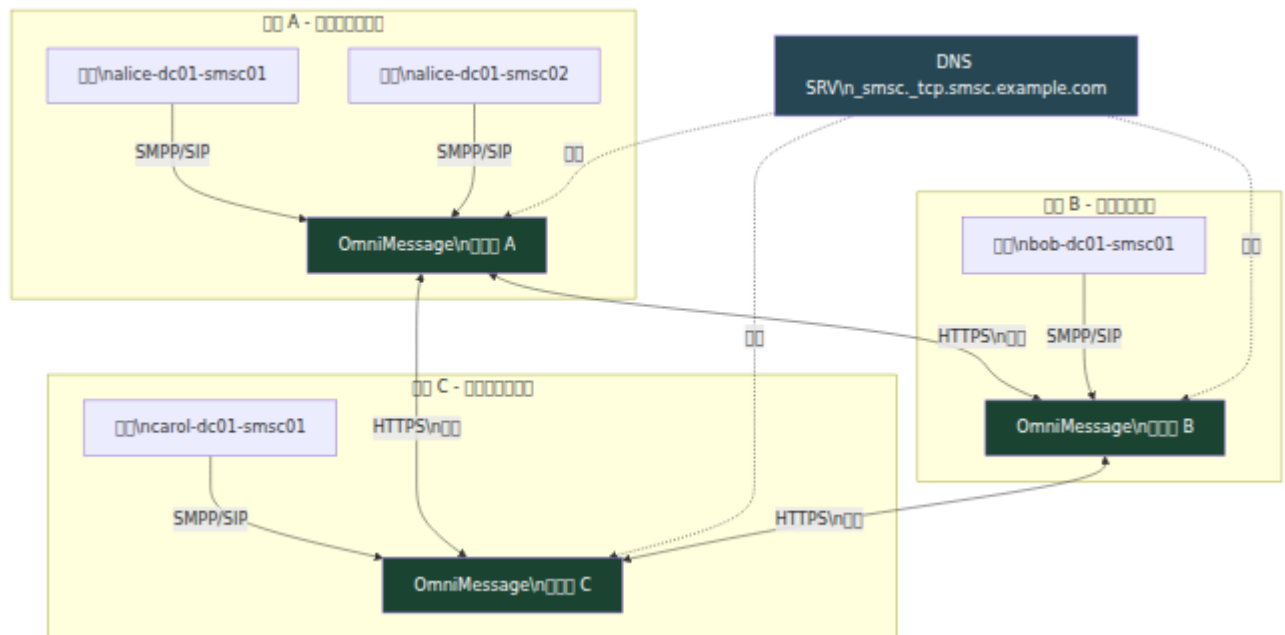
←      |     |

11

OmniMessage 通过 HTTP 接口与消息收发器通信 - 通过 DNS SRV 记录发现 HTTPS 接口

**FederationPoller**

00	00
00	000 WAN 00000000000000
0000	?? - 0000000000
00000	000000000000000000
000	00 HTTPS 0 TLS
00	00 HTTPS 00084430
0000	5-20 0000
0000	00000 - 0000



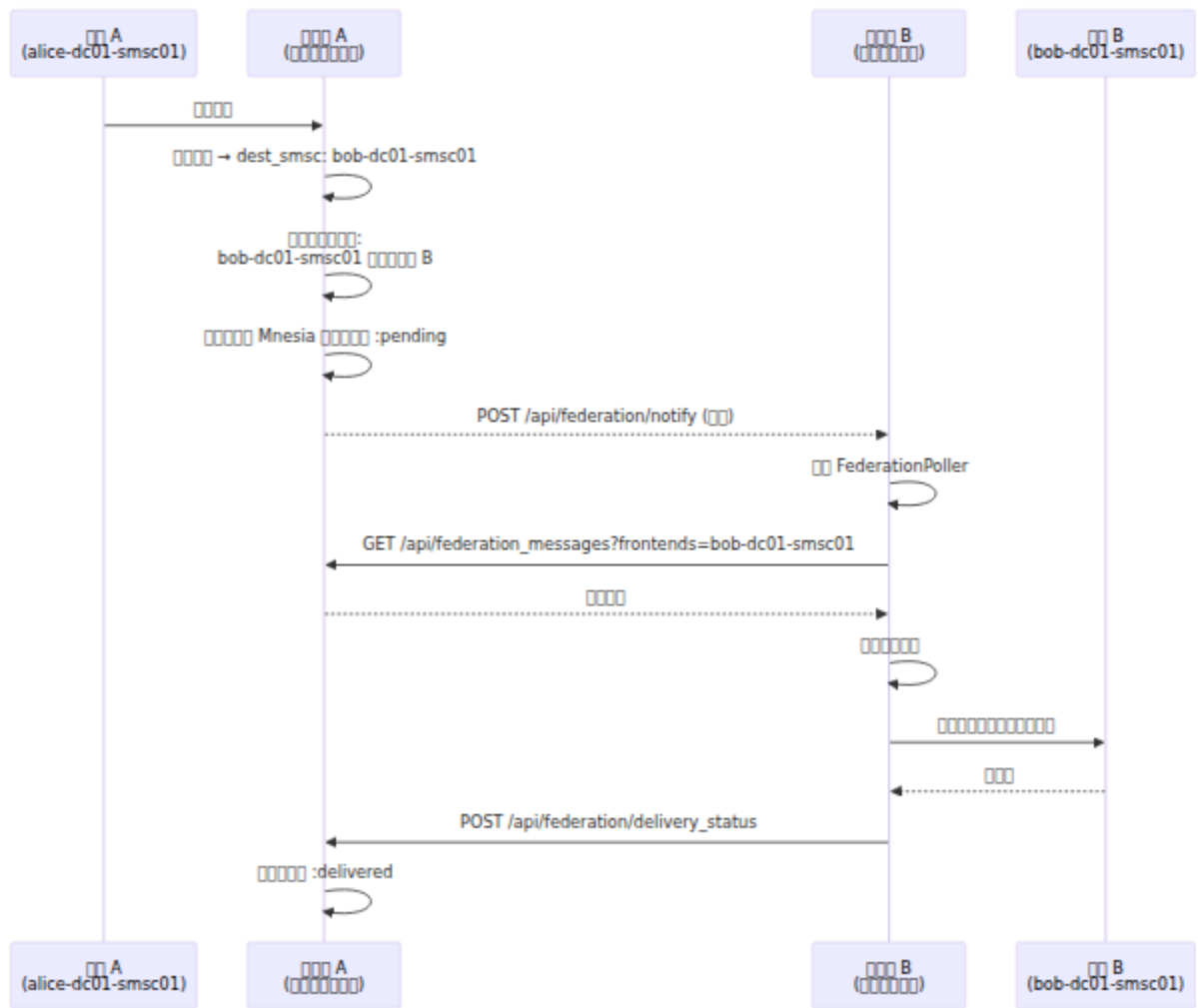
- 0000000000000000000000000000000000000000000000000000000
- 0000000000000000000000000000000000
- 000000000000000000000000
- 0000000000000000000000000000000000
- 000DNS SRV 00000000000000000000 DNS 00000

[illegible]

1. `Mnesia` `:pending`
2.
3. `FederationPoller` `GET /api/federation_messages`
4.
5.

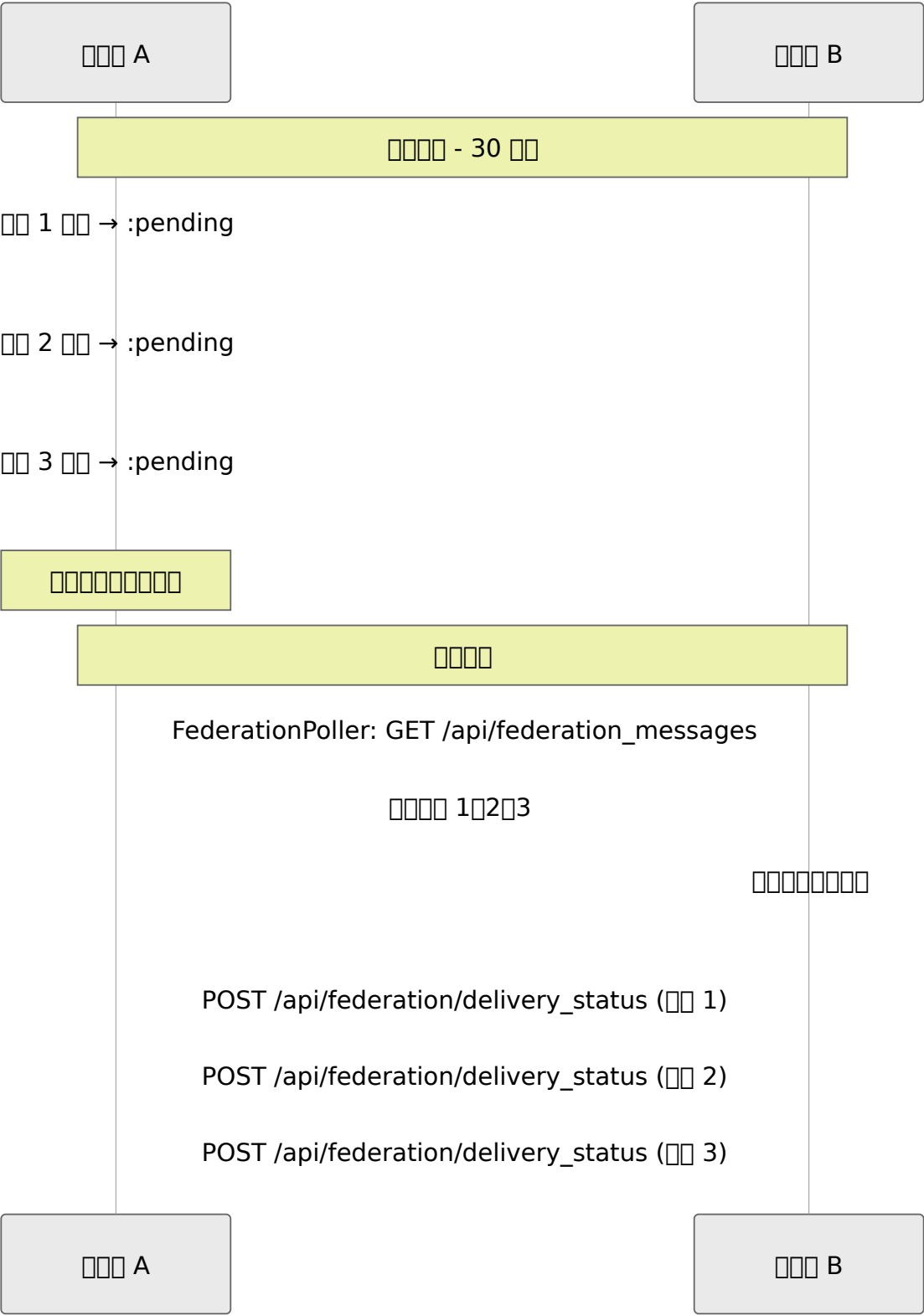
```
FederationPoller :pending 5
```





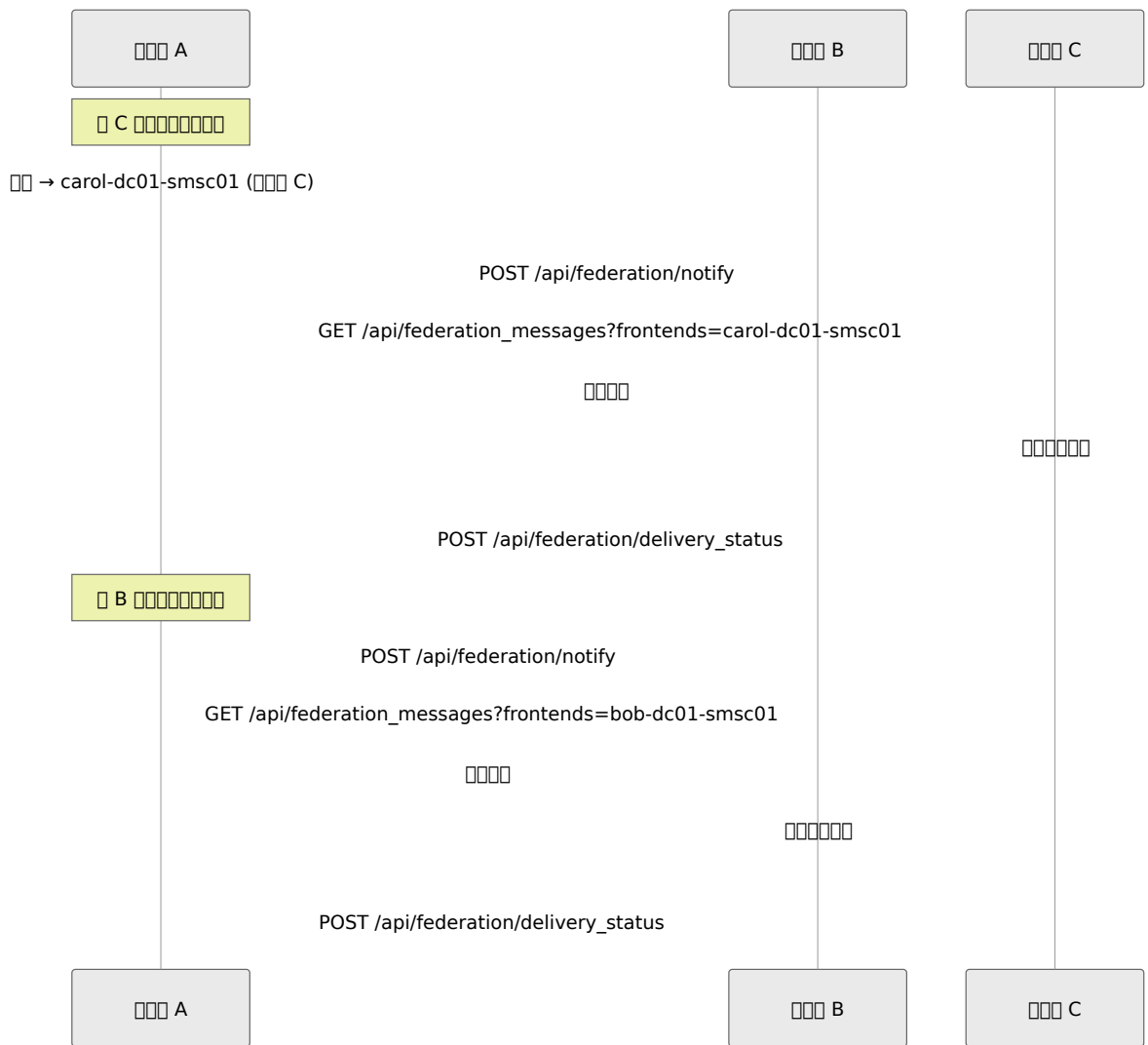
Message - FederationPoller

FederationPoller Message - FederationPoller



1 2 3

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30



POST /api/federation/notify

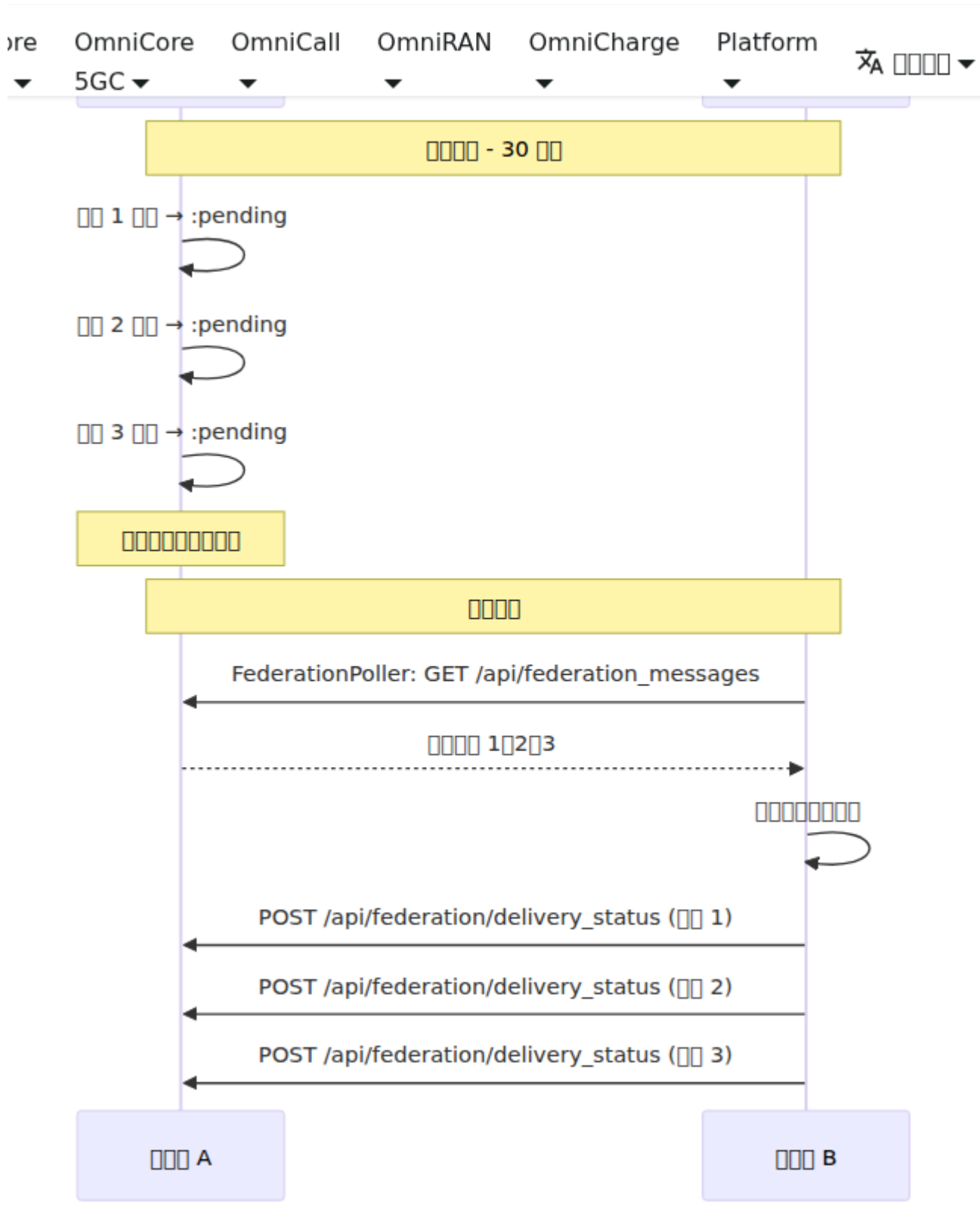
GET /api/federation/messages?frontends=carol-dc01-smisc01

POST /api/federation/delivery\_status

10 POST /api/federation/health - 3

POST /api/federation/registry

15 POST /api/federation/registry -



11111111111111111111



## 配置文件

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

  # 静态 DNS SRV 记录
  dns_srv_domain: "_smsc._tcp.smsc.example.com",

  # DNS SRV 轮询间隔
  dns_poll_interval_ms: 30_000,

  # 健康检查间隔
  health_check_interval_ms: 10_000,

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

  # FederationPoller 轮询间隔
  poll_interval_ms: 5_000,

  # API 端 HTTP 超时时间
  http_timeout_ms: 5_000,

  # 通过 DNS SRV 记录获取 URL 的 API 端口
  api_port: 8443,

  # 静态 DNS SRV 记录 - 默认为 dns_srv_domain
  static_peers: []
```

配置

項目	型	デフォルト	説明
<code>enabled</code>	boolean	<code>false</code>	この機能を有効にするかどうか。true なら有効、false なら無効。
<code>dns_srv_domain</code>	string	<code>""</code>	DNS SRV レコードを検索するドメイン名。static_peers に指定されたドメイン名と一致する必要があります。
<code>dns_poll_interval_ms</code>	int	<code>30000</code>	DNS SRV レコードを検索する間隔。単位はミリ秒。DNS レコードが変更されたときにのみ検索される。
<code>health_check_interval_ms</code>	int	<code>10000</code>	ピアの健康状態を確認する間隔。単位はミリ秒。
<code>registry_sync_interval_ms</code>	int	<code>15000</code>	ピアの登録情報を同期する間隔。単位はミリ秒。
<code>poll_interval_ms</code>	int	<code>5000</code>	FederationPoller がピアの登録情報を取得する間隔。単位はミリ秒。
<code>http_timeout_ms</code>	int	<code>5000</code>	HTTP/HTTPS リクエストのタイムアウト時間。単位はミリ秒。
<code>api_port</code>	int	<code>8443</code>	DNS SRV レコードで指定された URL のポート番号。API のポート番号は config :api_ex で指定されたポート番号と一致する必要があります。
<code>static_peers</code>	list	<code>[]</code>	DNS SRV レコードで指定されたピアのリスト。各ピアは host と port で指定されます。

# 配置

配置 DNS SRV 记录

```
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}
  ]
```

键	值	类型	备注
host	10.0.1.2	字符串	节点的 IP 地址
port	8443	整数	节点的 HTTPS API 端口

## DNS SRV 记录

DNS SRV 记录用于指定服务的名称和端口。格式如下：

### DNS 记录

```
; 定义 SRV 记录
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
alice.smsg.example.com.
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
bob.smsg.example.com.
_smsg._tcp.smsg.example.com. 86400 IN SRV 10 100 8443 smsg-
carol.smsg.example.com.

; 定义 A 记录
smsg-alice.smsg.example.com. 86400 IN A 10.0.1.2
smsg-bob.smsg.example.com. 86400 IN A 10.0.2.2
smsg-carol.smsg.example.com. 86400 IN A 10.0.3.2
```



配置 DNS SRV 记录 dns\_poll\_interval\_ms 默认为 30 秒

配置 DNS SRV 记录

## 配置

配置 HTTPS 端口

端口	协议	配置	说明
8443	TCP/TLS	配置	配置 HTTPS 端口
53	UDP/TCP	配置	DNS SRV 记录 DNS 记录

配置

```
# 配置 IP 地址
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 接口

API 接口 8443 端口 /api/federation/ 配置

URI	Method	Description
/api/federation/identity	GET	Get the identity information
/api/federation/health	POST	Health check - returns the status of the service
/api/federation/registry	POST	Register a new node
/api/federation/notify	POST	Notify the service of a change
/api/federation/delivery_status	POST	Report the delivery status of a message
/api/federation_messages	GET	Get the list of messages from the FederationPoller

API returns JSON with X-Federation-Node header

## Configuration

### 1. DNS SRV Configuration

Use DNS SRV records to find the service

```
# Example A - Configuration (config/runtime.exs)
config :sms_c,
  smsc_node_name: "alice-dc01-smsc01"

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

```
# 节点 B - 配置 (config/runtime.exs)
config :sms_c,
  smsc_node_name: "bob-dc01-smsc01"

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

```
# 节点 C - 配置 (config/runtime.exs)
config :sms_c,
  smsc_node_name: "carol-dc01-smsc01"

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

节点 B 配置了 SRV 记录，节点 C 配置了 B 的 FederationPoller 配置。A 配置了 B 的 FederationPoller 配置。

## 2. 配置 DNS SRV 记录

配置 DNS SRV 记录

```
# 配置 (config/runtime.exs)
config :sms_c,
  smsc_node_name: "primary-smc01"

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

```
# 配置 (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}
  ]
```

配置中指定了 IP 地址，FederationPoller 将连接到该 IP 地址。

## 配置 3 个静态对等节点

配置中指定了 3 个静态对等节点。

```
config :sms_c, :federation,
  enabled: true,
  dns_srv_domain: "_smc.tcp.cluster.internal",
  health_check_interval_ms: 5_000,
  registry_sync_interval_ms: 5_000,
  poll_interval_ms: 1_000,
  http_timeout_ms: 2_000
```

配置中指定了 RTT 时间为 10 毫秒。

## 配置

配置 Prometheus 端口 9568。

配置 `sms_c_federation_message_received_count` 指标：名称、单位、帮助文本。

- `origin_node` - 源节点

配置 `sms_c_federation_health_check_count` 指标：名称、单位、帮助文本。

- `peer` - 同僚ノード
- `result` - `ok` または `failed`

例:

```
# 同僚ノードからのメッセージ受信率
rate(sms_c_federation_message_received_count[5m]) * 60

# 同僚ノードのヘルスチェック失敗率
sum by (peer)
(rate(sms_c_federation_health_check_count{result="failed"}[5m]))
```

同僚ノードのヘルスチェック

項目	操作	条件
同僚ノードのヘルスチェック	<code>GET /api/federation/status</code> の Web UI	同僚ノードのヘルスチェック
同僚ノードのヘルスチェック	<code>GET /api/federation/identity</code> のヘルスチェック	同僚ノードのヘルスチェック > 5 分
同僚ノードのヘルスチェック	同僚ノード	> 2 同僚ノード

同僚ノード

同僚ノード

例: 同僚ノードのヘルスチェック `cluster_status` のヘルスチェック

例:

- DNS SRV のヘルスチェック
- DNS のヘルスチェック
- SRV のヘルスチェック
- 同僚ノード `enabled: false`

步骤:

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` 配置对等节点/对等端

## 配置对等端

注意: 配置对等端时, 必须配置“对等端”

步骤:

- 配置对等端
- 配置对等端 8443 端口
- TLS 配置
- 配置对等端

步骤:

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

## 配置对等端

注意: 配置对等端 `:pending` 配置对等端 0

步骤:

- 配置对等端 FederationPoller 配置对等端
- FederationPoller 配置
- 配置对等端配置对等端

步骤:

1. `docker-compose up -d`
2. `docker logs -f FederationPoller` `docker-compose up -d` “Federation poller started”
3. `curl -X GET http://localhost:8443/api/federation/status` - `docker-compose up -d`
4. `curl -X GET http://localhost:8443/api/federation_messages?frontends=<frontend_name>&include_unrouted=false`

# HSS □□□□□ (Diameter Sh)

← □□□□□□ | **SMS** □□□□

11

OmniMessage 通过 Diameter Sh 接口与 HSS 交互 MSISDN 信息

```
0000 000 - 0000 //originating_network 0  
terminating_network 000 :on_net 0 :off_net 00000000 :any 00000000  
000000000000 HSS 000
```

OmniMessage 送信 送信先 (UDR) 送信元番号 HSS 送信元番号 2001 送信元番号  
送信先 HSS 番号 5001 送信元番号 送信元番号 送信元番号 送信元番号  
SMS 送信 → 送信 / 送信

ENUM - HSS ENUM HSS (Sh) “ENUM HSS ENUM”ENUM (DNS NAPTR) “ENUM HSS ShENUM SMS ENUM → ENUM

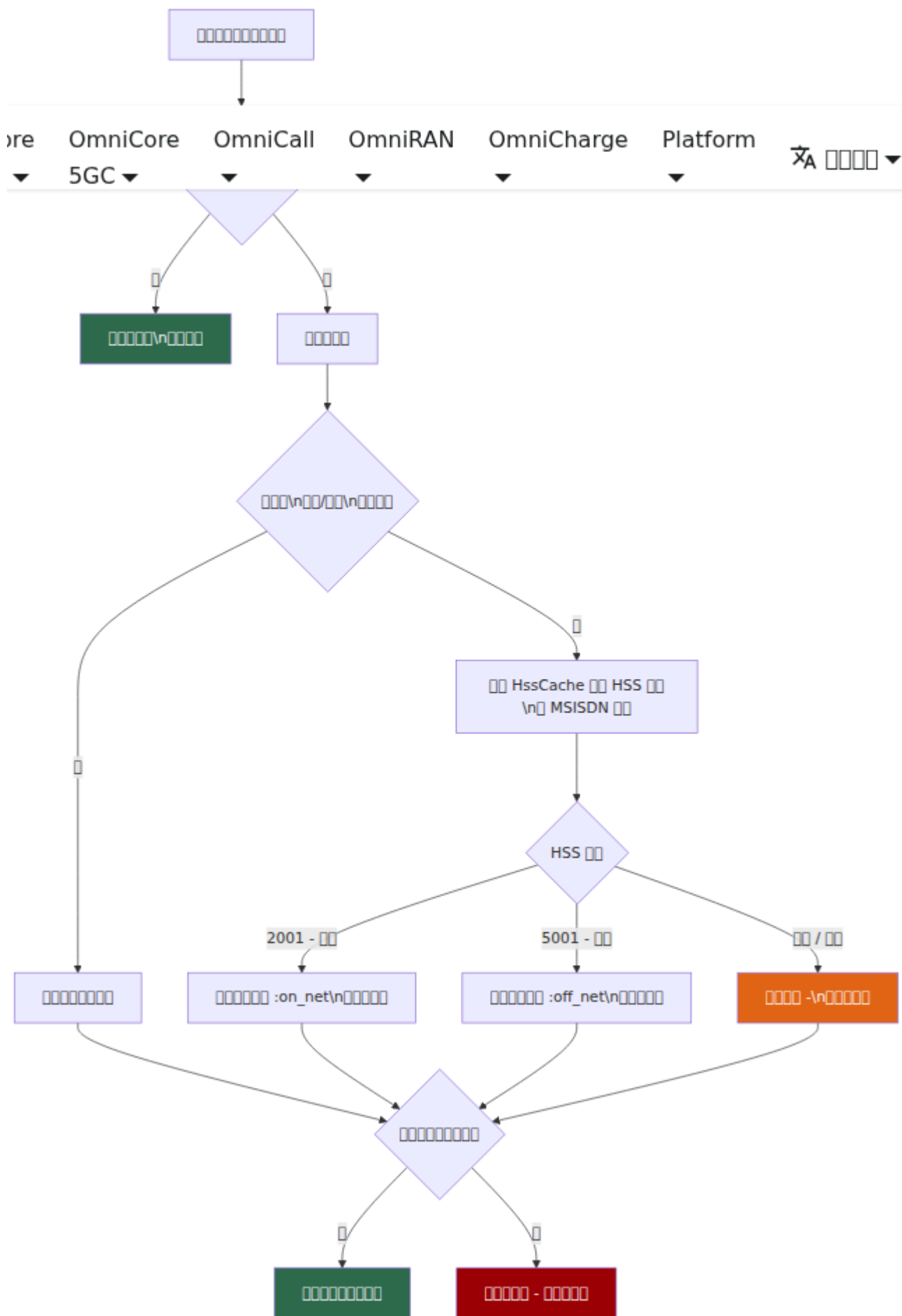
□ □ □ □

標準	標準名
3GPP TS 29.329	Diameter Sh
3GPP TS 29.328	IMS Sh -
RFC 6733	Diameter



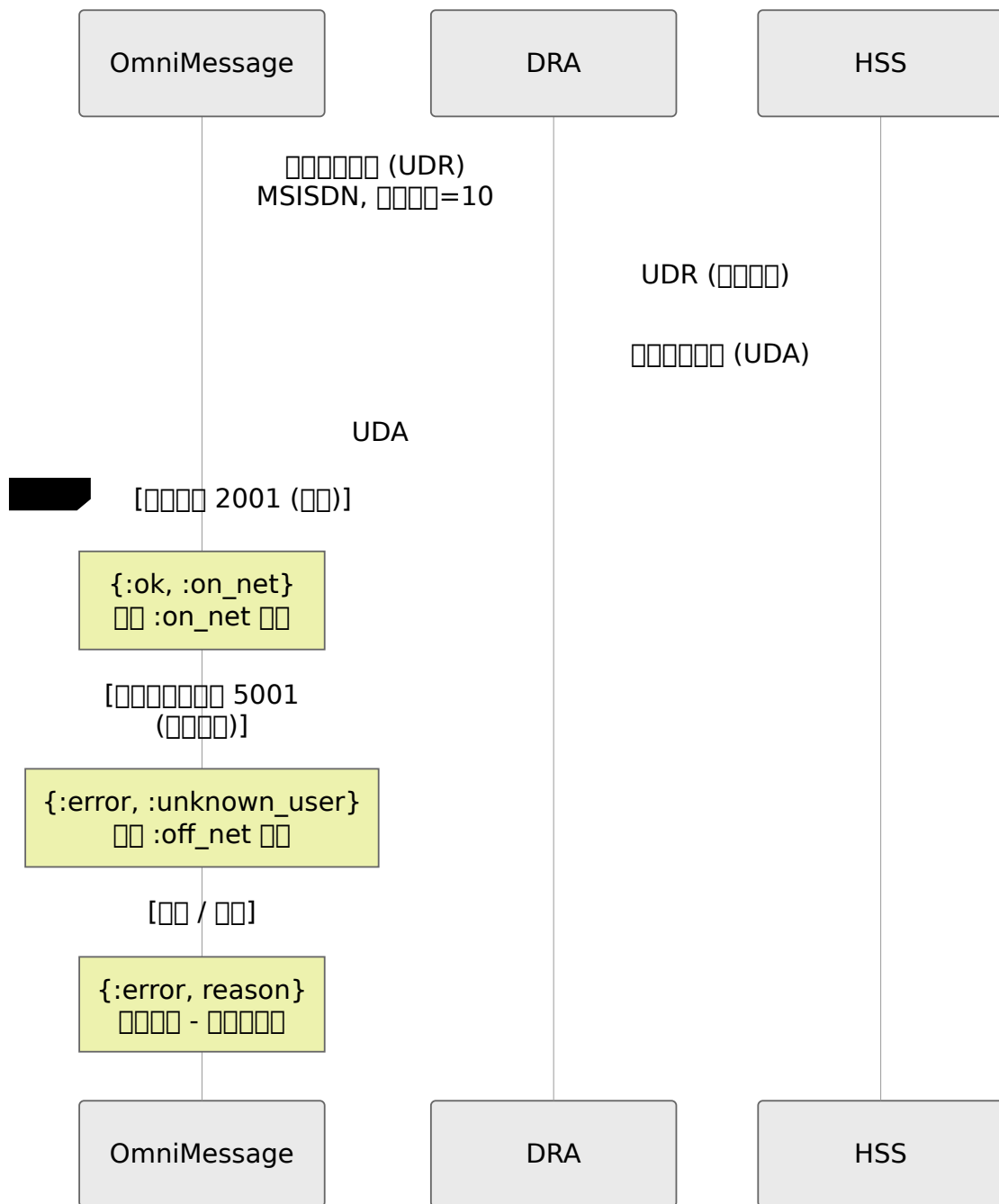
**HSS** □□□□□□□□

OmniMessage □□□□□□□□





# Diameter Sh 消息



# UDR

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

# UDA

UDA

項目	項目	項目	項目項目項目
2001	項目項目	{:ok, :on_net}	項目 - 項目 :on_net項目項目 :off_net
5001	項目項目項目 項目	{:error, :unknown_user}	項目 - 項目 :off_net項目項目 :on_net
項目 / 項目 / 項目 項目	-	{:error, reason}	項目項目 - 項目項目項目項目項目項目

項目

## 項目 HSS 項目

config/runtime.exs 項目項目項目項目 HSS 項目

- :sms\_c 項目 diameter\_enabled 項目 - 項目項目項目項目項目項目 HSS 項目項目 false 項目項目  
項目 :on\_net/:off\_net 項目項目項目項目項目項目項目項目
- :diameter\_ex 項目 - 項目 Diameter 項目項目項目項目項目項目項目項目

□□□□

```
# □□□□□□□□ 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.mnc001.mcc001.3gppnetwork.org",
    product_name: "OmniMessage",
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: SmsC.Diameter.Control,
    processor_module: DiameterEx.Processor,
    auth_application_ids: [],
    acct_application_ids: [],
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{
            vendor_id: 10415,
            auth_application_id: 16_777_217,
            acct_application_id: nil
          }
        ]
      }
    ],
    peers: [
      %{
        port: 3868,
        host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
```

```

        ip: "10.0.0.1",
        realm: "epc.mnc001.mcc001.3gppnetwork.org",
        tls: false,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }

```

## OmniMessage

項目	型別	デフォルト値	説明
diameter_enabled	Boolean	false	HSS と接続するかどうか。 false の場合は、 :on_net/:off_net を指定する必要がある。 指定しない場合は、 :on_net が適用される。
hss_cache_ttl_seconds	Integer	7200	HSS のキャッシュ TTL。 runtime.exs で YAML ファイルで指定する。
mock_sh	Boolean	false	HSS をシミュレートする。 Diameter のメッセージは 生成されない。
mock_sh_on_net_numbers	List	[]	mock_sh が true の場合に MSISDN を指定する。



# Diameter 参数

参数	数据类型	取值范围	默认值	说明
service_name	字符串		-	Diameter 服务名称
listen_ip	IP 地址		"0.0.0.0"	监听 IP 地址
listen_port	端口		3868	监听端口 6733
decode_format	字符串		-	Diameter 解码格式
host	字符串		-	Omni 主机名称
realm	字符串		-	Omni 3GPP epc.realm
product_name	字符串		-	产品名称
request_timeout	整数		5000	请求超时时间 (ms)
peer_selection_algorithm	字符串		: random	对等选择算法 : random

項目	型	型	値	説明
<code>allow_undefined_peers_to_connect</code>	bool	bool	true	未定義のピアに接続を許可するかどうか
<code>log_unauthorized_peer_connection_attempts</code>	bool	bool	true	未承認のピア接続試行をログに記録するかどうか
<code>control_module</code>	string	string	-	Diameter SMC
<code>processor_module</code>	string	string	-	Diameter Diameter
<code>auth_application_ids</code>	list	list	[]	認証アプリケーションID
<code>acct_application_ids</code>	list	list	[]	課金アプリケーションID
<code>vendor_id</code>	string	string	-	3GPP
<code>supported_vendor_ids</code>	list	list	-	サポートされているベンダーID

アプリケーション

`applications` 2つのDiameterアプリケーションID HSSアプリケーションID ShアプリケーションID

項目	型	必須	注	説明
application_name	String	必須	-	HSS アプリケーション名 :sh
application_dictionary	String	必須	-	Erlang Diameter アプリケーション名 :diameter_gen_3gpp_sh
vendor_specific_application_ids	String	必須	-	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	String	10415	3GPP ID
auth_application_id	String	16_777_217	3GPP TS 29.329 Sh ID
acct_application_id	-	nil	Sh ID

接続

peers 接続 OmniMessage 接続 Diameter 接続 DRA 接続 HSS 接続

項目	項目	項目	項目	項目
host	項目	項目	-	項目 Diameter 項目 項目 FQDN 項目 項目 項目 項目 項目 項目
ip	項目	項目	-	項目 TCP 項目 IP 項目
port	項目 項目	項目	3868	項目 Diameter 項目
realm	項目	項目	-	項目 Diameter 項目
tls	項目	項目	false	項目 TLS 項目
transport	項目	項目	:diameter_tcp	項目 項目 :diameter_tcp 項目 :diameter_sctp 項目
initiate_connection	項目	項目	true	項目 true 項目 OmniMessage 項目 項目 TCP 項目 項目 false 項目 OmniMessage 項目 項目

項目項目

項目項目項目項目 OmniMessage 項目 peer\_selection\_algorithm 項目項目項目項目項目

```

peers: [
  %{
    port: 3868,
    host: "dra01.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.0.0.1",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  },
  %{
    port: 3868,
    host: "dra02.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.0.0.2",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  }
]

```

□□□□

□□□□□□ HSS □□□□□□□□□□□□

```

config :sms_c,
  diameter_enabled: true,
  mock_sh: true,
  mock_sh_on_net_numbers: [
    "15555550001",
    "15555550002"
  ]

```

□□□□□□ mock\_sh\_on\_net\_numbers □□□□□□ MSISDN □□□□ { :ok, :on\_net } □□□□□□  
 □□□ 2001□□□□□□ MSISDN □□□□ { :error, :unknown\_user } □□□□□□□□ 5001□□□□□□  
 □ Diameter □□□□□□□□□□□□□□

# HSS 配置

HSS 配置 `SmsC.Diameter.HssCache` 与 MSISDN 配置 `Diameter` 配置

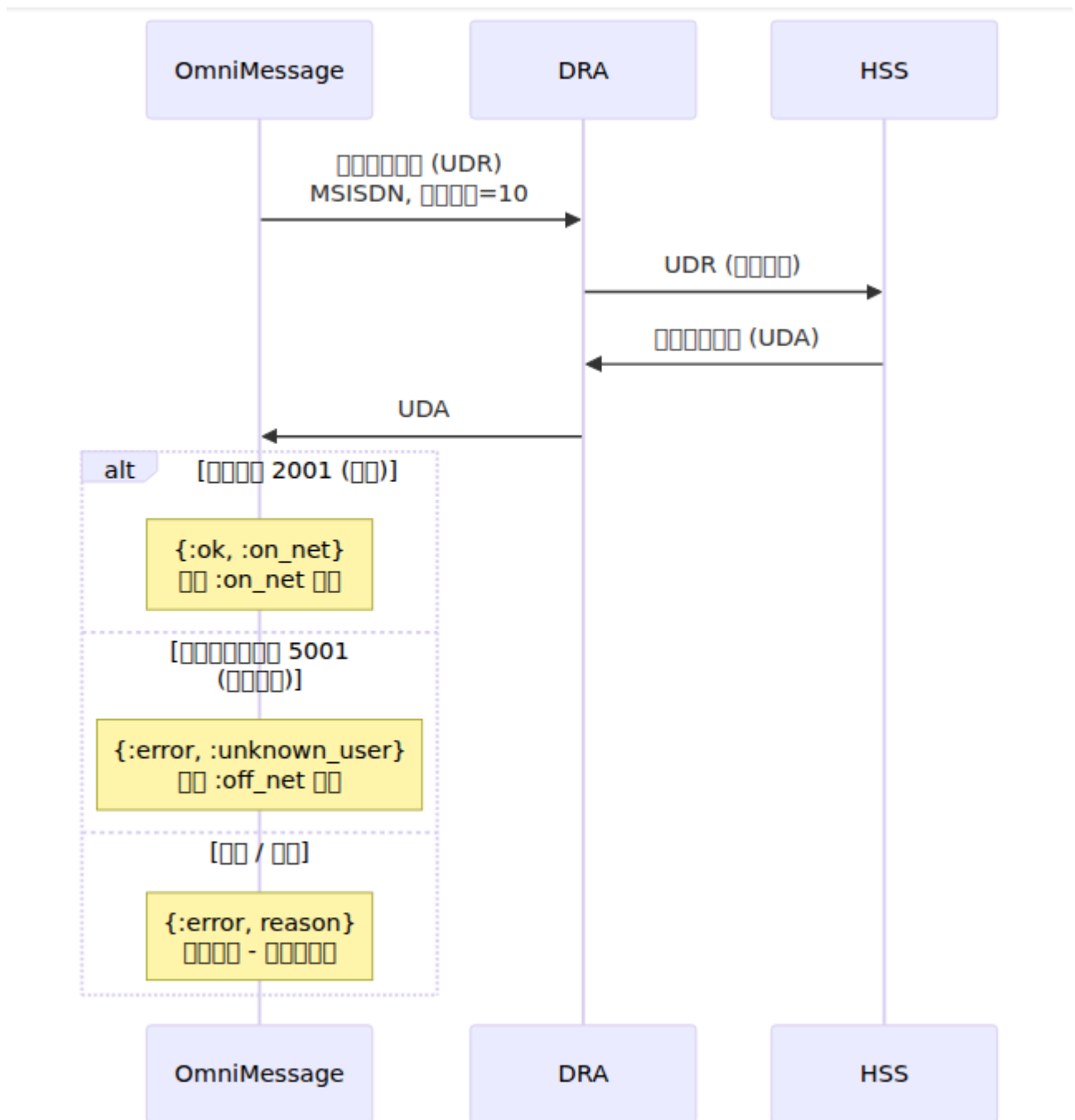
- **TTL** `hss_cache_ttl_seconds` 配置 7200 = 2 小时
- 配置 `MSISDN` 配置
- 配置 `MSISDN` 配置
- 配置 `{:ok, :on_net}` 配置 `{:error, :unknown_user}` 配置 `Diameter` 配置 `nil` 配置 - 配置

## 配置

`:saf` 配置 `action: :saf` 配置 `terminating_network: :on_net` 配置 `SMS` 配置  
→ `saf` 配置

配置 `SAF` 配置 `Mnesia` 配置 `dest_smsc` 配置 `MSISDN` 配置

配置 `dead_letter_time_minutes` 配置



length

length

OmniMessage length Diameter Sh length

- length Diameter Sh (HSS): ENABLED (host: smsc01, realm: epc.mnc001.mcc001.3gppnetwork.org, peers: 2)

- `Diameter Sh (HSS): DISABLED`

## Overview

• HSS `subscriber_exists?/1` `{:error, reason}` `:on_net` `:off_net` `:any`

• `OmniMessage` `terminating_network:` `:any`

• `Diameter` `diameter_enabled: false` `nil` `:on_net` `:off_net` `:any`

## Details

HSS `route_evaluated` `term_network: ✓`

- `route_evaluated` `term_network: ✓` `(terminating requires on_net, +15551234 is on-net)` `x (... is off-net)` API
- **Diameter Sh** `SmsC.Telemetry` `METRICS.md`
- `HSS dip: <msisdn> is ON-NET (2001) / is NOT in HSS (5001)` `SmsC.Diameter.Sh`

## Summary

• `expires` HSS

• `SmsC.Diameter.Sh`

## Conclusion / Summary

• `SmsC.Diameter.Sh`



項目	値	単位	説明
originating_network	:any   :on_net   :off_net	桁 桁	発信 MSISDN の HSS 桁 桁を指定する
terminating_network	:any   :on_net   :off_net	桁 桁	着信 MSISDN の HSS 桁 桁を指定する

桁桁桁桁 :any 桁 HSS 桁桁 :on\_net 桁桁桁桁 2001桁 :off\_net 桁 5001桁桁桁桁  
桁???

## 配置

```
config :sms_c, :sms_routes, [
  # 1555 - 1555
  %{
    calling_regex: nil,
    called_regex: ~r/^1555/,
    source_smsc: nil,
    dest_smsc: "local-gateway",
    source_type: nil,
    action: :route,
    charged: :default,
    weight: 100,
    priority: 10,
    description: "1555 - 1555",
    originating_network: :any,
    terminating_network: :on_net,
    enabled: true
  },

  # 1555 - 1555
  %{
    calling_regex: nil,
    called_regex: nil,
    source_smsc: nil,
    dest_smsc: "external-gateway",
    source_type: :ims,
    action: :route,
    charged: :default,
    weight: 100,
    priority: 255,
    description: "1555 - 1555",
    originating_network: :any,
    terminating_network: :any,
    enabled: true
  }
]
```

## 配置说明

1. 1555 - 1555
2. 1555 - 1555

3. P10 "local-gateway" 配置 `terminating_network: :on_net` 配置 HSS 配置 HSS 号 2001 配置 `local-gateway` 配置 5001 配置
4. P255 "external-gateway" 配置 `terminating_network: :any` 配置 HSS 配置

配置 配置 HSS 配置 // 配置 配置 P10 配置 `action` 配置 `:saf`

配置

配置

配置/配置

配置

- `diameter_enabled` `false` - `:on_net` 配置
- 配置 Diameter 配置
- HSS 配置 Sh 配置 "HSS dip" 配置
- 配置 HSS 配置

配置

1. 配置 `runtime.exs` 配置 `diameter_enabled: true`
2. 配置 Diameter 配置 `Diameter Sh (HSS): ENABLED (... peers: N)`
3. 配置 `HSS dip`: 配置 Sh 配置
4. 配置 HSS 配置 E.164 配置 + 配置

配置 ? ? 配置

配置 `:saf` 配置

配置

- 配置 `OmniMessage`

- 設定するMSISDNを指定する `destination_msisdn`
- 設定するメッセージ

設定

1. 設定/メッセージメッセージメッセージ
2. 設定メッセージメッセージMSISDN
3. 設定 `dead_letter_time_minutes` - 設定メッセージメッセージ

## Diameter

設定するDiameter

設定

- 設定するIP
- 設定する3868
- 設定するメッセージ `host` 設定するDiameter
- OmniMessage

設定

1. 設定するIP
2. 設定するメッセージメッセージTCP
3. 設定する `host` 設定するメッセージメッセージ
4. 設定OmniMessage 設定 `realm`

# SMS-C Prometheus



←



SMS-C Prometheus



Prometheus

`http://localhost:9568/metrics`

Prometheus 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
- `vm` - Erlang VM 接続

管理画面

## sms\_c\_license\_status

種別: Gauge

単位: OmniMessage SMS-C 接続数

説明:

- `1` - 接続中
- `0` - 接続失敗

タグ: 種別

メトリック名: `omnimessage`

注: 接続失敗時に "NOLICENCE" と表示されます

接続失敗時の原因:

- 接続失敗 原因
- 接続失敗 (`dest_smsc`) 接続失敗 **"NOLICENCE"**
- 接続失敗 原因
- UI 接続失敗 原因
- 接続失敗 原因

注:

```
- 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 - NOLICENCE"
```

Prometheus:

```
# SMS-C 异常
sms_c_license_status == 1

# SMS-C 正常
sms_c_license_status == 0

# NOLICENCE 异常
sms_c_routing_route_matched_count{dest_smsc="NOLICENCE"}
```

短信接收计数

## sms\_c\_message\_received\_count

Counter

SMS-C 接收计数

标签:

- source\_smsc: 短信 SMSC
- source\_type: 短信类型 (ims, circuit\_switched, smpp)
- message\_type: 消息类型 (sms, mms)

说明: 短信接收计数

短信: 短信发送失败/短信发送成功

---

## sms\_c\_message\_validated\_count

短信: Counter

短信: 短信发送成功

短信:

- `valid`: 短信发送成功 `true` 或 `false`

短信: 短信发送失败/短信发送成功

短信: 短信发送失败/短信发送成功 > 5% 短信

---

## sms\_c\_message\_processing\_stop\_duration

短信: Histogram

短信: 短信发送失败/短信发送成功

短信: 短信

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

短信:

- `success`: 短信发送成功 `true` 或 `false`

短信: 短信发送失败/短信发送成功

短信: p95 或 p99 短信 SLA 短信发送成功

---



计数器

## sms\_c\_routing\_route\_matched\_count

计数器: Counter

计数器: 统计短信路由匹配成功的次数

计数器:

- route\_id: 路由ID
- dest\_smsc: 目标SMSC
- priority: 优先级

计数器: 统计短信路由匹配成功的次数

计数器: 统计短信路由匹配成功的次数

---

## sms\_c\_routing\_failed\_count

计数器: Counter

计数器: 统计短信路由失败的原因

计数器:

- reason: 失败原因, no\_route\_found, validation\_failed

计数器: 统计短信路由失败的原因

计数器: 统计短信路由失败的原因

---

## sms\_c\_routing\_action\_count

计数器: Counter

计数器: 统计短信路由动作的次数

短信:

- `action`: 短信动作drop, auto\_reply, forward
- `route_id`: 短信路由ID

短信: 短信发送失败次数统计

短信: 短信发送失败原因统计

---

## sms\_c\_routing\_stop\_duration

短信: Histogram

短信: 短信路由停止时间分布

短信: 单位

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

短信:

- `dest_smsc`: 短信接收方 SMSC

短信: 短信路由停止时间分布

短信: 短信路由停止时间分布p95 > 50ms

---

## ENUM/NAPTR 短信

### sms\_c\_enum\_cache\_hit\_count

短信: Counter

短信: 短信ENUM缓存命中DNS次数

短信:

- `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

{}:

- 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

{}:

- domain: [] ENUM []

{}: [] ENUM [] [] [] [] [] [] [] [] 1-3 [] [] []

{}: [] [] [] [] 0 [] [] [] [] [] DNS [] [] [] [] []

---

{} {} {} {}

## sms\_c\_delivery\_queued\_count

{}: Counter

{}: [] [] [] [] [] [] SMSC [] [] [] [] []

{}:

- `dest_smsc`: 短信中心号码

:

00: 00000/0000000000000000

**sms\_c\_delivery\_attempted\_count**

□□: Counter

□□ : □□□□□□□□□□□□□□□□

11

- `dest_smsc`: 短信中心号码

**00:** 00000000000000000000000000000000

**sms\_c\_delivery\_succeeded\_count**

□□: Counter

□□: □□□□□□□ SMSC □□□□□□

11

- `dest_smsc`: 短信中心号码

:

00: 0000000 SLA 00000000

```
##: success_rate = succeeded / queued
```

**sms\_c\_delivery\_failed\_count**

□□: Counter

短信: 短信发送失败次数

短信:

- `dest_smsc`: 短信 SMSC 地址
- `reason`: 失败原因

短信: 短信发送失败次数

短信: 短信发送失败次数

---

## sms\_c\_delivery\_dead\_letter\_count

短信: Counter

短信: 短信发送失败次数

短信:

- `reason`: 失败原因 `max_retries_exceeded`, `expired`

短信: 短信发送失败次数

短信: 短信发送失败次数

---

## sms\_c\_delivery\_succeeded\_attempt\_count

短信: Histogram

短信: 短信发送成功次数

短信: 1, 2, 3, 5, 10

短信:

- `dest_smsc`: 短信 SMSC 地址

短信: 短信发送成功次数

例: 電話番号 2 桁を指定する

---

## sms\_c\_delivery\_failed\_attempt\_count

例: Histogram

例: 電話番号を指定する

例: 1, 2, 3, 5, 10

例:

- `dest_smsc`: 宛 SMS 番号

例: 電話番号を指定する

---

## sms\_c\_delivery\_time\_delta\_delta\_ms

例: Histogram

例: 電話番号を指定する SMS-C 番号を指定する

例: 例

例: 50, 100, 250, 500, 1000, 2000, 5000, 10000, 20000, 60000, 600000 ms  
(50ms 10 例)

例:

- `source_smsc`: 送信 SMS
- `dest_smsc`: 宛 SMS
- `delivery_attempts`: 送信回数 0 = 送信失敗

例: 例-SMS-C 番号を指定する SMS 番号を指定する SLA 番号を指定する

例:

- p95 時間/時間 SLA 時間
- 時間時間時間時間時間

Prometheus :



```

# 95th Percentile SMSC Latency P95
histogram_quantile(0.95,
  sum by (source_smsc, dest_smsc, le) (
    rate(sms_c_delivery_time_delta_delta_ms_bucket[5m])
  )
)

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

# Avg SMSC Latency
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])))

# Latency Percentiles by Delivery Attempts
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])))
)

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

# Top 2 Latencies
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

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

## sms\_c\_delivery\_time\_by\_smsc\_delta\_ms

□□: Histogram

[illegible]

```
sms_c_delivery_time_delta_delta_ms - sms_c_delivery_attempts  
sms_c_delivery_time_delta_delta_ms / sms_c_delivery_attempts  
[:sms_c, :delivery,  
:time_delta]
```

00:00

□: 50, 100, 250, 500, 1000, 2000, 5000, 10000, 20000, 60000, 600000 ms  
(50ms □ 10 □□)

11

- `source_smsc`: 5 digit SMSC
- `dest_smsc`: 5 digit SMSC

□□: □□□□□□/□□□□□□□□□□□□□□/SLA □□□□□□□□□□

```

sms_c_delivery_time_delta_delta_ms  _delta

```

## ❏ Prometheus ❏:

```
# 平均/秒 SMS 送信時間
sum by (source_smsc, dest_smsc)
(rate(sms_c_delivery_time_by_smsc_delta_ms_sum[5m])) /
sum by (source_smsc, dest_smsc)
(rate(sms_c_delivery_time_by_smsc_delta_ms_count[5m]))

# 送信時間
sum(rate(sms_c_delivery_time_by_smsc_delta_ms_sum[5m])) /
sum(rate(sms_c_delivery_time_by_smsc_delta_ms_count[5m]))

# 平均/秒 SMS 送信 P95 時間
histogram_quantile(0.95,
  sum by (source_smsc, dest_smsc, le) (
    rate(sms_c_delivery_time_by_smsc_delta_ms_bucket[5m])
  )
)
```

送信時間

## sms\_c\_queue\_size\_size

送信: Gauge

送信: 送信待ちメッセージの総数

送信:

- `queue_type`: `message_queue`, `dead_letter`

送信: 送信待ちメッセージの総数

送信: 送信待ちメッセージの総数

## sms\_c\_queue\_size\_pending

送信: Gauge

短信: 短信队列大小

短信:

- queue\_type: 短信

短信: 短信队列大小

短信: 短信队列大小

---

## sms\_c\_queue\_size\_failed

短信: Gauge

短信: 短信队列大小

短信:

- queue\_type: 短信

短信: 短信队列大小

短信: 短信队列大小

---

## sms\_c\_queue\_size\_delivered

短信: Gauge

短信: 短信队列大小/短信队列大小

短信:

- queue\_type: 短信

短信: 短信队列大小

短信: 短信队列大小

---

## sms\_c\_queue\_oldest\_message\_age\_seconds

□□: Gauge

:

11

- `queue_type`: □□□□

00: 0000000000000000 SLA 00000000

例: 顧客のSLA 達成率 > 300 日

□ □ □ □

0000 **Diameter Ro** 0000000000 account 0000000000 MSISDN000000000000  
 00000000 reason: :no\_credit00 OCS 00000/000000000000 charging\_failed 0000  
 0 :no\_credit 000000000000 :balance\_rejected00OCS 0000000000000000

## sms\_c\_charging\_requested\_count

□□: Counter

□□: □□□ OCS □□□□□□□□Diameter Ro CCR □□□□□□

11

- account: 000000 MSISDN00000000

:

**sms\_c\_charging\_succeeded\_count**

□□: Counter

□□ : □□□□□□□□

概要:

- `account`: 口座番号

単位: 回

計算式: `success_rate = succeeded / requested`

---

## sms\_c\_charging\_failed\_count

概要: Counter

単位: 回

概要:

- `account`: 口座番号
- `reason`: 失敗理由

単位: 回

計算式: `success_rate = succeeded / requested`

---

## sms\_c\_charging\_succeeded\_duration

概要: Histogram

単位: 秒

単位: 秒

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

概要:

- `account`: 口座番号

単位: 回

📄: 📄 p95 📄📄📄📄📄📄📄📄📄

---

📄📄📄📄📄

## **sms\_c\_mnesia\_table\_size\_record\_count**

📄: Gauge

📄: 📄 Mnesia 📄📄📄📄📄📄📄📄 10 📄📄📄📄

📄:

- `table`: 📄

📄📄:

- `sms_route` - SMS 📄📄
- `message_store` - 📄📄📄📄📄📄📄📄📄📄
- `location_store` - 📄📄/📄📄
- `frontend_store` - 📄/SMSC 📄
- `translation_rule` - 📄📄📄
- `cell_tower_store` - 📄📄📄
- `message_events` - 📄📄📄

📄: 📄📄📄📄📄📄📄📄📄📄📄📄

📄: 📄📄📄📄📄📄📄📄📄📄📄📄

📄 **Prometheus** 📄:

```
# 短信记录
sms_c_mnesia_table_size_record_count

# 短信记录
sms_c_mnesia_table_size_record_count{table="message_store"}

# 短信记录
sms_c_mnesia_table_size_record_count{table="location_store"}

# 短信记录
sms_c_mnesia_table_size_record_count{table="sms_route"}
```

---

## sms\_c\_frontend\_status\_count

指标: Gauge

单位: 无

维度:

- `frontend_name`: 前端名称
- `status`: 连接状态connected, disconnected

指标: 无

单位: 无

---

## sms\_c\_location\_registered\_count

指标: Counter

单位: 无/秒

维度:

- `location`: 位置/SMSC 名称
- `ims_capable`: 是否支持IMS true/false



00: 0000000000 IMS 00 IMS 000000000000

□□ : □□□□□□□□

- 0000000000000000
- 0000000
- 0 IMS 0000000000000000

□□□□:

```
# 1000000
rate(sms_c_location_registered_count[1m])

# IMS 100 IMS 1000000
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

例: 00000000000000000000/SMSC IMS 00000000 10 00000000000000000000

11

- location: 00000000/SMSC 00
- ims\_capable: 00000000 IMS true/false

00: 0000000000000000 IMS 00000000000000000000 SMSC 00000000

□□ : □□□□□□□□

- 
- 
- 

## ❏ Prometheus ❏:

```

# 總和
sum(sms_c_location_active_registrations_count)

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

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

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

# 按 IMS 支援分組
sum by (ims_capable) (sms_c_location_active_registrations_count)

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

# 按位置分組
sum by (location) (sms_c_location_active_registrations_count) /
sum(sms_c_location_active_registrations_count) * 100

```

## HTTP API 說明

### phoenix\_endpoint\_stop\_duration

圖表: Distribution (Histogram)

說明: HTTP 請求響應時間分布

方法:

- method: HTTP 請求方法 GET, POST

- `path`: `xxxxxxxxxxxxxUUIDxxxx IDxxxx /:idxxxx/api/messages, /api/messages/:id`

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

`:` `API` `xxxxxxxxxxxxxxxxxxxx` SLA

`:` `xxxxxxxxxx`

- `xxxxxx P95` `xx > 500ms`
- `xxxxxx P99` `xx > 1s`
- `xxxxxx`

`xxxx:`

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

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

## phoenix\_endpoint\_stop\_count

`:` Counter

`:` `xxxxxxxxxx HTTP` `xxxxxxxxxxxxxx HTTP` `xxxxxx`

`:`

- `method`: HTTP `xxxxxx` `GET`, `POST`
- `path`: `xxxxxxxxxxxxxxxxxx /:id`
- `status`: HTTP `xxxxxx` 200, 201, 400, 404, 500 `xx`

`:` `API` `xxxxxxxxxxxxxxxxxxxx`

`:` `xxxxxxxxxx`

- 異常率 > 5%
- 異常率 5xx 以上
- 異常率

SQL:

```
# 異常率
sum by (path) (rate(phoenix_endpoint_stop_count[5m]))

# 異常率
sum by (path) (rate(phoenix_endpoint_stop_count{status=~"5.."}[5m])) /
sum by (path) (rate(phoenix_endpoint_stop_count[5m]))

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

## phoenix\_router\_dispatch\_exception\_count

Counter

HTTP ステータスコード/例外

:

- `route`: API 名前
- `kind`: `error`, `exit`, `throw`

: 例外発生回数

: 例外発生率

:

```
# 5m 异常率
rate(phoenix_router_dispatch_exception_count[5m])

# 1h 异常数增加
increase(phoenix_router_dispatch_exception_count[1h])
```

---

## Diameter Sh 简介

SMS-C 通过 **Diameter Sh** 接口 3GPP TS 29.329 与 HSS 交互，用于鉴权、计费、策略控制等。  
Diameter 简介

### sms\_c\_diameter\_peer\_status

类型: Gauge

范围: Diameter 对等体 1 到 0

属性:

- origin\_host**: Diameter Origin-Host
- ip**: IP 地址

单位:

```
# 检查对等体状态
sms_c_diameter_peer_status == 0
```

---

### sms\_c\_diameter\_sh\_request\_count

类型: Counter

范围: Diameter Sh 与 HSS 交互的 UDR 数量

---

## sms\_c\_diameter\_sh\_response\_count

📊: Counter

📄: 📄📄📄📄 Diameter Sh 📄📄📄📄📄UDA📄📄

📄📄:

- `result`: 📄📄📄📄📄📄 `on_net`, `unknown_user`, `error`📄
- `result_code`: Diameter Experimental-Result / Result-Code📄📄📄 `2001`, `5001`📄

📄📄📄📄:

```
# Sh 📄📄📄📄📄  
sum(rate(sms_c_diameter_sh_response_count{result="error"}[5m]))  
  / sum(rate(sms_c_diameter_sh_response_count[5m]))
```

---

## sms\_c\_diameter\_sh\_response\_duration\_ms

📊: Distribution (Histogram)

📄📄: Diameter Sh 📄📄📄📄📄📄📄📄📄

📄📄:

- `result`: 📄📄📄📄

📄: 5, 10, 25, 50, 100, 250, 500, 1000, 2500 ms

📄📄📄📄:

```
# P95 Sh 📄📄📄📄  
histogram_quantile(0.95,  
  rate(sms_c_diameter_sh_response_duration_ms_bucket[5m]))
```

## sms\_c\_diameter\_sh\_error\_count

Counter

Diameter Sh / UDA

:

- reason:

SMS-C

## sms\_c\_federation\_forward\_attempted\_count

Counter

:

:

- dest\_controller:

## sms\_c\_federation\_forward\_completed\_count

Counter

:

:

- dest\_controller:

## sms\_c\_federation\_forward\_queued\_count

短信: Counter

短信: 短信转发队列中的消息数量

短信:

- `dest_controller`: 目标控制器
- 

## sms\_c\_federation\_forward\_failed\_count

短信: Counter

短信: 短信转发失败的消息数量

短信:

- `dest_controller`: 目标控制器
- 

## sms\_c\_federation\_message\_received\_count

短信: Counter

短信: 接收到的短信消息数量

短信:

- `origin_node`: 源节点
- 

## sms\_c\_federation\_health\_check\_count

短信: Counter

短信: 健康检查消息数量

短信:



- `peer`: `peer`
- `result`: `peer` `healthy`, `unhealthy`

## sms\_c\_federation\_peer\_status\_count

`peer`: Gauge

`peer`: `peer`

`peer`:

- `status`: `peer` `healthy`, `degraded`, `unhealthy`

## sms\_c\_federation\_queue\_depth\_count

`peer`: Gauge

`peer`: `peer`

`peer`:

```
# peer
sms_c_federation_queue_depth_count > 1000
```

## Erlang VM

### vm\_memory\_total

`peer`: Gauge

`peer`: Erlang VM `peer`

`peer`: `peer`

⚠: 80% 以上を超過する

---

## vm\_memory\_processes

⚠: Gauge

⚠: Erlang プロセスのメモリ使用量

⚠: 80% 以上を超過する

⚠: 80% 以上を超過する

---

## vm\_total\_run\_queue\_lengths\_total

⚠: Gauge

⚠: CPU の実行キューの長さ

⚠: CPU の実行キューの長さ

⚠:  $10 * \text{CPU}$  を超過する

---

## vm\_system\_counts\_process\_count

⚠: Gauge

⚠: VM のプロセス数

⚠: 80% 以上を超過する

⚠: 262,144 を超過する

---



$$\square\square: (\text{sms\_c\_enum\_cache\_hit\_count}) / (\text{sms\_c\_enum\_cache\_hit\_count} + \text{sms\_c\_enum\_cache\_miss\_count})$$

例:  $80\%$  の確率で TTL を更新する

--	--	--	--	--

□ □ □ □ □ □ □ □ □ □

🔍🔍: sms\_c\_routing\_route\_matched\_count | route\_id |

□□	:	□□□□□□□□□□□□□□□□
----	---	------------------

□ □ □ □ □ □

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

11

- sms\_c\_queue\_size\_pending (int)
- sms\_c\_queue\_oldest\_message\_age\_seconds (int)

$$\square\square : \square\square\square\square\square\square\square + \square\square\square\square\square = \square\square\square\square\square$$

1111

10101010101010101010

```
00: sms_c_delivery_succeeded_attempt_count 00000000
```

**□□:** □□ 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</code> > 100	Critical	消息路由失败，请检查消息队列配置。
消息队列积压	<code>queue_size_pending</code> > 1000	Warning	消息队列积压，请检查消息处理速度。
消息队列老化	<code>queue_oldest_message_age_seconds</code> > 300	Critical	消息队列老化，请检查消息处理速度。
消息投递失败	<code>delivery_failed_count</code> > 100	High	消息投递失败，请检查消息接收端配置。
消息投递死信	<code>delivery_dead_letter_count</code> > 0	High	消息投递死信，请检查消息接收端配置。
ENUM 查询失败	<code>enum_lookup_stop_duration</code> p95 > 5000ms	Warning	DNS 查询失败，请检查 DNS 配置。
ENUM 查询成功率	ENUM 查询成功率 < 0.7	Warning	ENUM 查询成功率低，请检查 DNS 配置。
前端状态异常	<code>frontend_status_count{status="disconnected"}</code> > 0	High	前端状态异常，请检查前端配置。
消息充电失败	<code>charging_failed_count</code> > 100	High	消息充电失败，请检查消息接收端配置。
消息处理超时	<code>message_processing_stop_duration</code> p95 > 1000ms	Warning	消息处理超时，请检查消息处理速度。

## □□□□□

### □□□□□

□□: □□□□□□□□

□□:

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

### □□□□□

□□: □□□□□□

□□:

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

### □□□□□

□□: □□□□□□□□

□□:

- |  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

- □□□□: 15 □
- **5** □□□□: 90 □
- **1** □□□□: 2 □



--	--	--	--	--	--	--	--

1. `sms_c_message_received_count` - 00000000
2. `sms_c_routing_failed_count` - 00000000
3. `sms_c_delivery_queued_count` - 00000000
4. `sms_c_delivery_failed_count` - 00000000
5. `dest_smsc` 00000000

## 短信处理时间

指标名称

1. `sms_c_message_processing_stop_duration` 消息 - 处理时间
  2. `sms_c_routing_stop_duration` - 路由时间
  3. `sms_c_enum_lookup_stop_duration` - ENUM 查找时间
  4. `sms_c_charging_succeeded_duration` - 计费时间
  5. 其他时间
- 

## 短信队列大小

指标名称

1. `sms_c_queue_size_pending` 消息 - 队列大小
  2. `sms_c_delivery_attempted_count` - 尝试发送次数
  3. `sms_c_delivery_failed_count` - 发送失败次数
  4. `sms_c_delivery_time_delta_delta_ms` - 发送时间差
  5. `dest_smsc` 目标短信中心
- 

# Prometheus 配置

配置项

配置项名称5 配置项:

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

配置项名称1 配置项:

```
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

按源类型求和:

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

按源类型求和:

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

按源类型求和p95:

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

按目的地求和:

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

原因:

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

直近5分p95:

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

直近5分p99:

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

待機

待機原因:

```
sms_c_queue_size_pending
```

失敗原因:

```
sms_c_queue_size_failed
```

待機時間:

```
sms_c_queue_oldest_message_age_seconds / 60
```

待機率:

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

待機率:

```
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]))
```

📊📊📊

📊📊📊:

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

📊📊📊📊:

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

📊📊📊**p50, p95, p99**:

```
histogram_quantile(0.50, sms_c_routing_stop_duration_bucket)  
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)  
histogram_quantile(0.99, sms_c_routing_stop_duration_bucket)
```

📊📊📊📊:

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

📌📌📌📌📌:

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

📌📌📌📌📌📌:

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

## ENUM 📌

**ENUM** 📌📌📌:

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

**ENUM** 📌📌📌📌:

```
(rate(sms_c_enum_cache_hit_count[5m]) /  
(rate(sms_c_enum_cache_hit_count[5m]) +  
rate(sms_c_enum_cache_miss_count[5m]))) * 100
```

**ENUM** 📌📌📌p95:

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

📌 **ENUM** 📌📌📌📌📌📌:

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

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

指标 **NAPTR** 指标名称:

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

**ENUM** 指标:

```
sms_c_enum_cache_size_size
```

指标

指标名称 **p95**:

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

指标名称 **p99**:

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

指标:

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

指标:

```
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"})
```

查询语句:

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

查询语句

**Mnesia** 查询:

```
sms_c_mnesia_table_size_record_count
```

查询语句:

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

查询语句:

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

## Grafana 配置

配置 1: 配置

配置: 配置

配置:

1. 配置

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

## 2. 消息投递成功率

- 公式:  $\frac{\text{rate}(\text{sms\_c\_delivery\_succeeded\_count}[5\text{m}])}{\text{rate}(\text{sms\_c\_delivery\_queued\_count}[5\text{m}])} * 100$
- 范围: 0-100%
- 阈值:
  - 阈值: < 90
  - 阈值: 90-95
  - 阈值: > 95

## 3. 消息队列积压量

- 公式:  $\text{sms\_c\_queue\_size\_pending}$
- 公式:  $\text{sms\_c\_queue\_size\_failed}$
- 单位: 条
- 维度: {{queue\_type}}

## 4. 消息队列最老消息年龄

- 公式:  $\text{sms\_c\_queue\_oldest\_message\_age\_seconds} / 60$
- 单位: 分钟
- 阈值:
  - 阈值: < 5
  - 阈值: 5-10
  - 阈值: > 10

## 5. 消息前端连接数

- 公式:  $\text{sum}(\text{sms\_c\_frontend\_status\_count}\{\text{status}=\text{"connected"}\})$
- 单位: 条
- 单位: 条

## 6. 消息路由失败率

- 公式:  $\text{rate}(\text{sms\_c\_routing\_failed\_count}[5\text{m}]) * 60$
- 单位: 条/秒
- 阈值: > 0



## Table 2: ???

Table: ?????????

Table:

### 1. ?????

- Table: `histogram_quantile(0.50, sms_c_message_processing_stop_duration_bucket)` (p50)
- Table: `histogram_quantile(0.95, sms_c_message_processing_stop_duration_bucket)` (p95)
- Table: `histogram_quantile(0.99, sms_c_message_processing_stop_duration_bucket)` (p99)
- Table: Table
- Table: Table

### 2. ?????

- Table: `histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)`
- ENUM: `histogram_quantile(0.95, sms_c_enum_lookup_stop_duration_bucket)`
- Table: `histogram_quantile(0.95, sms_c_charging_succeeded_duration_bucket)`
- Table: `histogram_quantile(0.95, sms_c_delivery_time_delta_delta_ms_bucket)`
- Table: Table
- Table

### 3. ?????

- Table: `sms_c_delivery_succeeded_attempt_count_bucket`
- ?????
- Table: Table 1 ?????

### 4. ENUM ?????

- 指标: `rate(sms_c_enum_cache_hit_count[5m]) / (rate(sms_c_enum_cache_hit_count[5m]) + rate(sms_c_enum_cache_miss_count[5m]))`
- 指标: `sms_c_enum_cache_size_size`
- 指标 Y 轴单位: 无

## 5. 指标说明

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

## 指标 3: 指标

指标: 指标

指标:

### 1. 指标说明

- 指标: `sum by (source_type) (increase(sms_c_message_received_count[1h]))`
- 指标: IMS 和 CS 和 SMPP

### 2. 指标 SMSC 指标

- 指标: `sum by (source_smsc) (rate(sms_c_message_received_count[1h]))`
- 指标 10 指标
- 指标

### 3. 指标说明

- 统计
  - 统计 ID
  - 统计 SMSC
  - 统计 1h: `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
- 统计: `{{dest_smsc}}`

#### 5. 统计/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: 统计 7
- 统计

### 统计 4: 统计

统计: 统计

统计:

#### 1. 统计

- 指标: sms\_c\_queue\_size\_size
- 单位: 消息队列大小
- 数据类型: 整数

## 2. 消息队列大小

- 指标: sms\_c\_mnesia\_table\_size\_record\_count{table="sms\_route"}
- 指标: sms\_c\_mnesia\_table\_size\_record\_count{table="translation\_rule"}
- 单位: 30 秒

## 3. 消息队列大小

- 指标:  $\text{rate}(\text{sms\_c\_delivery\_queued\_count}[5\text{m}]) - (\text{rate}(\text{sms\_c\_delivery\_succeeded\_count}[5\text{m}]) + \text{rate}(\text{sms\_c\_delivery\_failed\_count}[5\text{m}]))$
- 单位: 消息/秒
- 指标: 消息/秒

## 4. 消息队列大小

- 指标:  $\text{max\_over\_time}(\text{rate}(\text{sms\_c\_message\_received\_count}[5\text{m}])[24\text{h}:])$
- 单位: 24 小时内 5m 内
- 指标: 消息/秒

## 5. 消息队列大小

- 指标:  $(\text{rate}(\text{sms\_c\_message\_received\_count}[5\text{m}]) / \text{MAX\_CAPACITY}) * 100$
- 指标: MAX\_CAPACITY 消息队列大小
- 单位: 消息/秒
- 指标:
  - 指标: < 70
  - 指标: 70-85
  - 指标: > 85

## 📄 5: SLA 📄

📄: 📄 SLA 📄📄📄📄

📄:

### 1. SLA 📄📄📄📄

- 📄📄: `(rate(sms_c_delivery_succeeded_count[1h]) / rate(sms_c_delivery_queued_count[1h])) * 100`
- 📄📄 99%
- 📄:
  - 📄: < 95
  - 📄: 95-99
  - 📄: >= 99

### 2. 📄 SLA 📄📄📄📄📄

- 📄: `count(sms_c_delivery_time_delta_delta_ms_bucket{le="5000"}) / count(sms_c_delivery_time_delta_delta_ms_bucket)`
- 📄 5 📄📄📄📄📄
- 📄: 📄

### 3. SLA 📄📄📄📄

- 📄 5 📄📄📄: `increase(sms_c_queue_oldest_message_age_seconds{} > 300)[24h:]`
- 📄 0

### 4. 📄📄📄📄📄📄

- 📄: `up{job="sms-c"}`
- 📄: 1 = 📄0 = 📄
- 📄📄📄

### 5. 📄📄📄📄📄📄

- 📄: `avg_over_time((rate(sms_c_delivery_succeeded_count[1h]) / rate(sms_c_delivery_queued_count[1h]))[24h:1h])`

- 可用性: 99.99%
- SLA 99%

## 报警规则

### 报警规则

报警规则:

```
alert: RoutingFailuresDetected
expr: increase(sms_c_routing_failed_count[5m]) > 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "{ $value } 路由失败 5 分钟"
  description: "路由失败 5 分钟"
```

报警规则:

```
alert: MessageQueueBacklog
expr: sms_c_queue_size_pending > 10000
for: 5m
labels:
  severity: critical
annotations:
  summary: "消息队列 backlog 大小为 { $value }"
  description: "消息队列 backlog 大小为 { $value }"
```

报警规则:

```
alert: OldMessagesInQueue
expr: sms_c_queue_oldest_message_age_seconds > 300
for: 2m
labels:
  severity: critical
annotations:
  summary: "队列消息过老 {{ $value }} 秒"
  description: "队列消息过老"
```

队列消息过老:

```
alert: NoActiveFrontends
expr: sum(sms_c_frontend_status_count{status="connected"}) == 0
for: 1m
labels:
  severity: critical
annotations:
  summary: "无活跃前端"
  description: "无活跃前端"
```

无活跃前端:

```
alert: DeadLetterMessagesIncreasing
expr: rate(sms_c_delivery_dead_letter_count[10m]) > 0
for: 5m
labels:
  severity: critical
annotations:
  summary: "{{ $value }} 死信消息增加"
  description: "死信消息增加"
```

死信消息增加

死信消息增加:

```
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: "短信发送成功率: {{ $value | humanizePercentage }}"
  description: "短信发送成功率 95%以下"
```

配置:

```
alert: HighDeliveryRetryRate
expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count_bucket) > 2
for: 15m
labels:
  severity: warning
annotations:
  summary: "95th 短信发送成功率: {{ $value }}"
  description: "短信发送成功率 95%以上"
```

配置:

```
alert: SlowMessageProcessing
expr: histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket) > 1000
for: 10m
labels:
  severity: warning
annotations:
  summary: "95th 短信发送延迟: {{ $value }}ms"
  description: "短信发送延迟 1000ms以上"
```

**ENUM** 配置:



```

alert: HighEnumFailureRate
expr: rate(sms_c_enum_lookup_stop_duration_count{success="false"}
[10m]) > 0.1
for: 10m
labels:
  severity: warning
annotations:
  summary: "ENUM 异常: {{ $value }}"
  description: "DNS 异常 DNS 异常"

```

■ **ENUM 异常:**

```

alert: LowEnumCacheHitRate
expr: rate(sms_c_enum_cache_hit_count[10m]) /
(rate(sms_c_enum_cache_hit_count[10m]) +
rate(sms_c_enum_cache_miss_count[10m])) < 0.70
for: 30m
labels:
  severity: warning
annotations:
  summary: "ENUM 异常: {{ $value | humanizePercentage }}"
  description: "异常异常异常异常异常"

```

异常:

```

alert: ChargingFailuresDetected
expr: rate(sms_c_charging_failed_count[10m]) > 0.05
for: 10m
labels:
  severity: warning
annotations:
  summary: "异常: {{ $value }}"
  description: "异常异常 OCS 异常"

```

异常

- 异常异常异常异常异常异常异常异常异常异常

- 了解 Prometheus 的 `rate()` 与 `increase()`
- Gauge 的使用
- Histogram 的 p50, p95, p99 百分位数
- Prometheus 的查询语言
- 了解 5m, 1h, 24h+ 的时间范围
- Prometheus 的报警系统
- Grafana 的报警配置 `dest_smsc` 与 `source_smsc`

# MMS 簡介 (MMSc)

← 消息服務 | 消息 | **API 簡介**

## 簡介

- 簡介
- 消息
- 消息服務
- 消息
- **MMS 簡介**
- 消息/消息
- 消息
- 消息
- 消息
- 消息
- 消息

## 簡介

OmniMessage 消息服務 **MMS 簡介 (MMSc)** 消息服務 3GPP TS 23.140 消息 MMS 消息 **MM1** 消息 (HTTP/WAP) 消息 **MM4** 消息 (SMTP)

MMSc 消息 OmniMessage 消息 Web 消息 MMS 消息 MMSc

1. 消息 MMS 消息 Mnesia 消息
2. 消息 **MMS** 消息 SMS 消息 SMS 消息
3. 消息 URL 消息 MMS

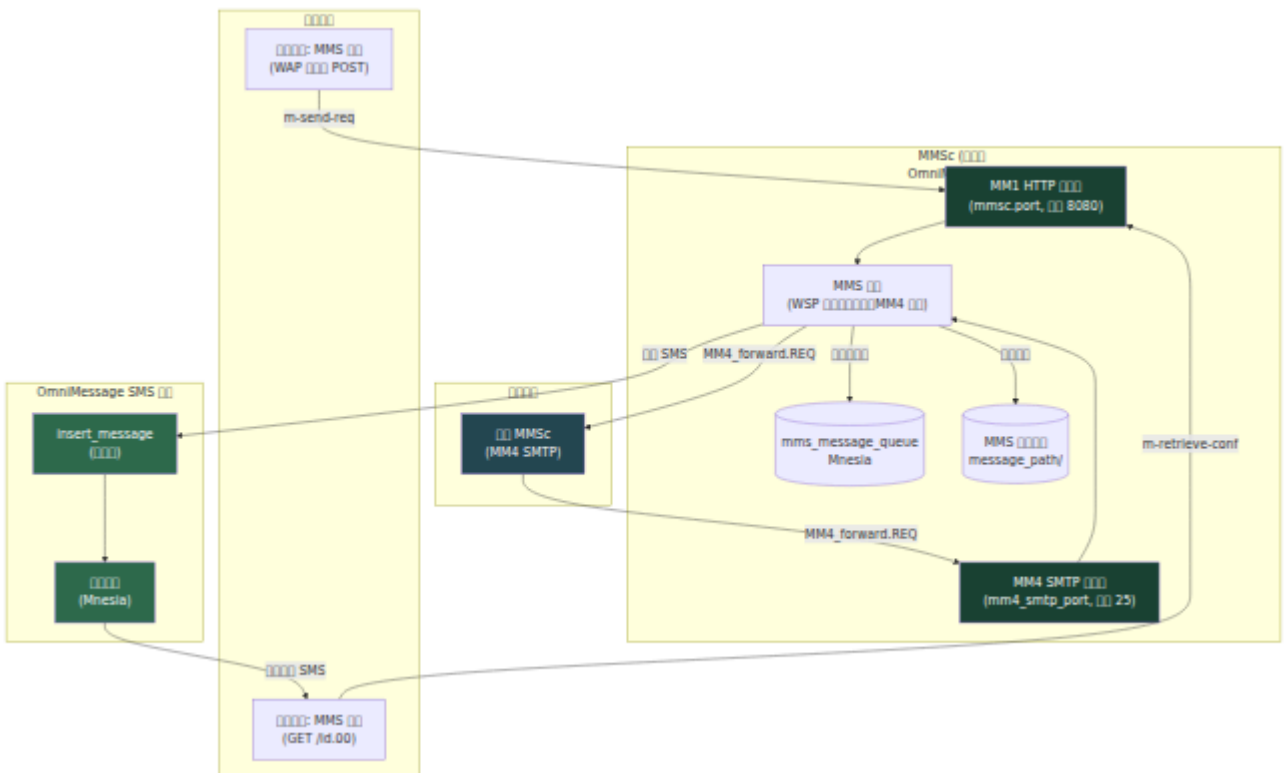
消息 SMS 消息 MMS 消息 SMS 消息 HTTP 消息

--	--	--	--

標準	標準
3GPP TS 23.140	IPマルチキャスト (MMS) 標準
3GPP TS 26.140	MMS 標準
OMA MMS 標準	MM1 WAP 標準

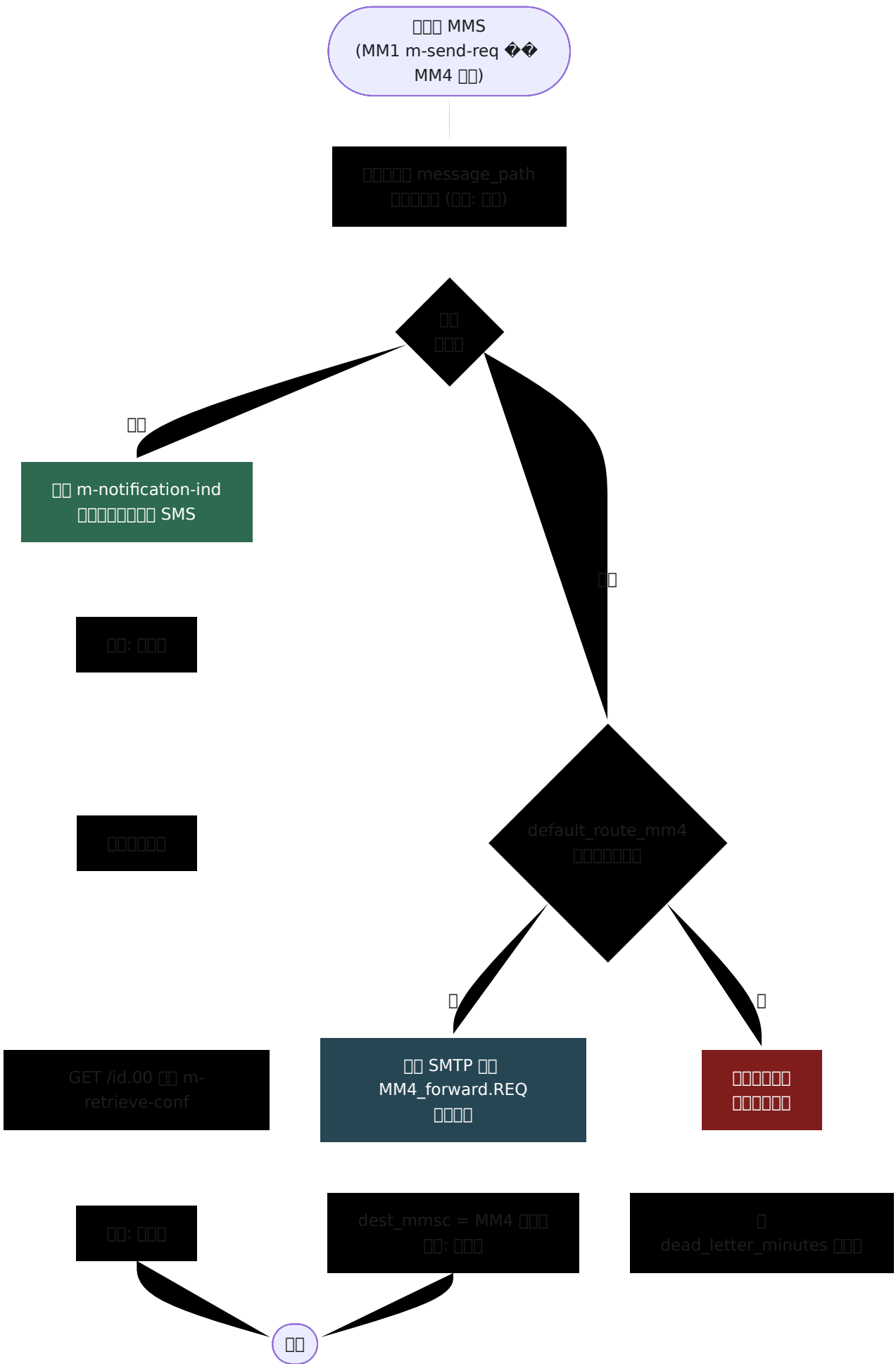
11

MM1 (利用 HTTP) と MM4 (利用 SMTP) を通じて OmniMessage API を通じて MMS を利用する Mnesia を通じて SMS を利用する MMSc を通じて MMS を利用する。利用する。



□□□□□□

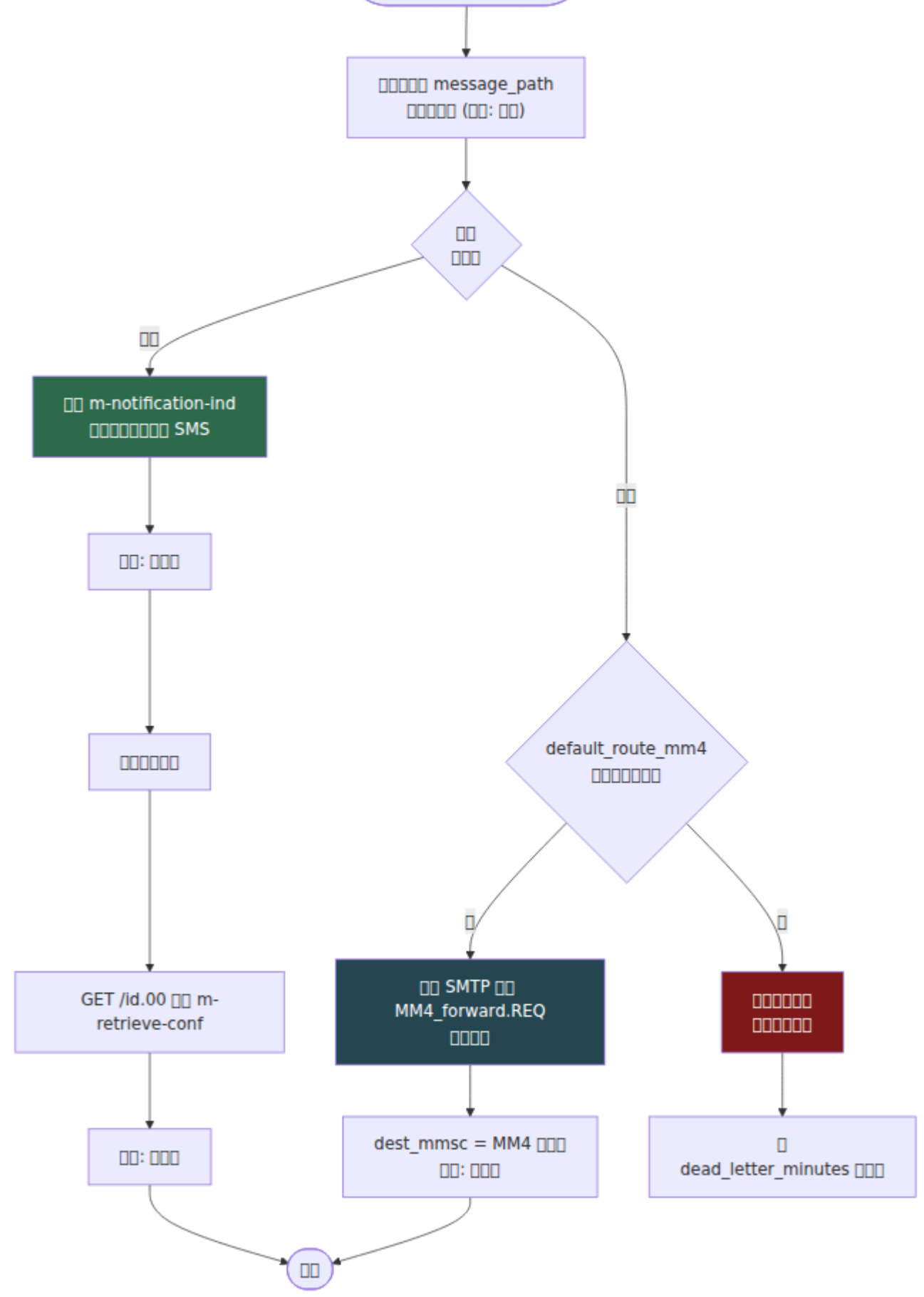
□□□□ MMS□□□□□□ MM1□□□□□□□□ MM4□□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□



□□□□

□□□□ (**MM1**)

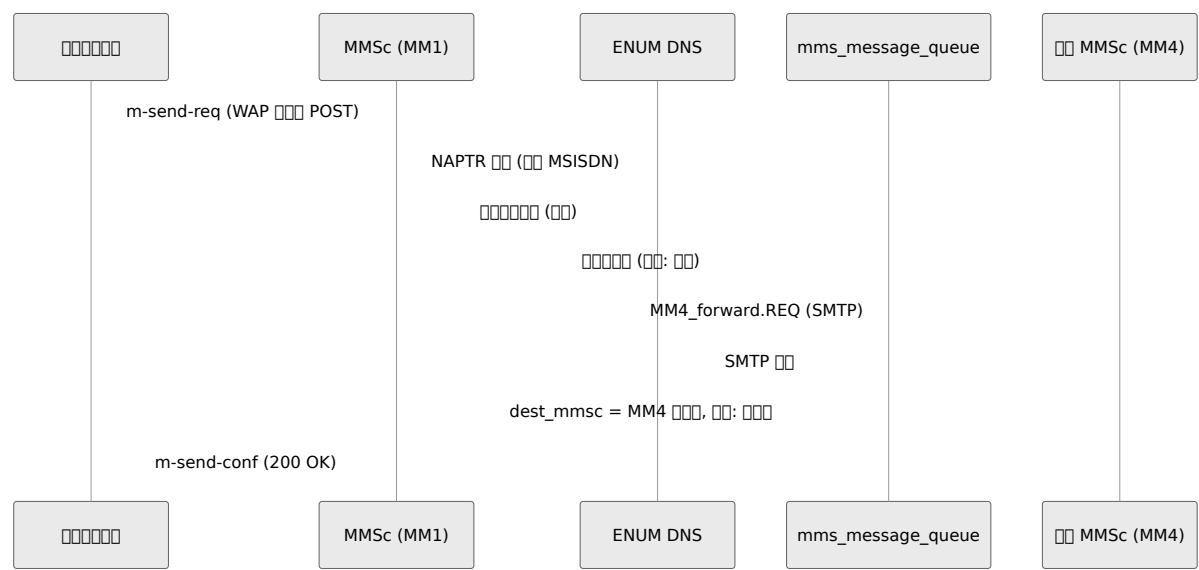
□□□□□□□□□□ MMS□





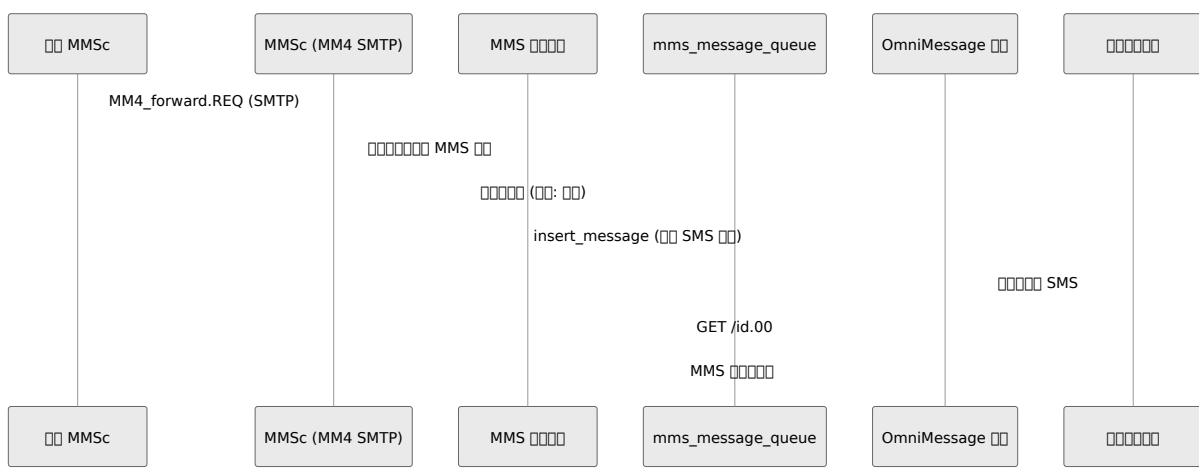
# MM4

MMS



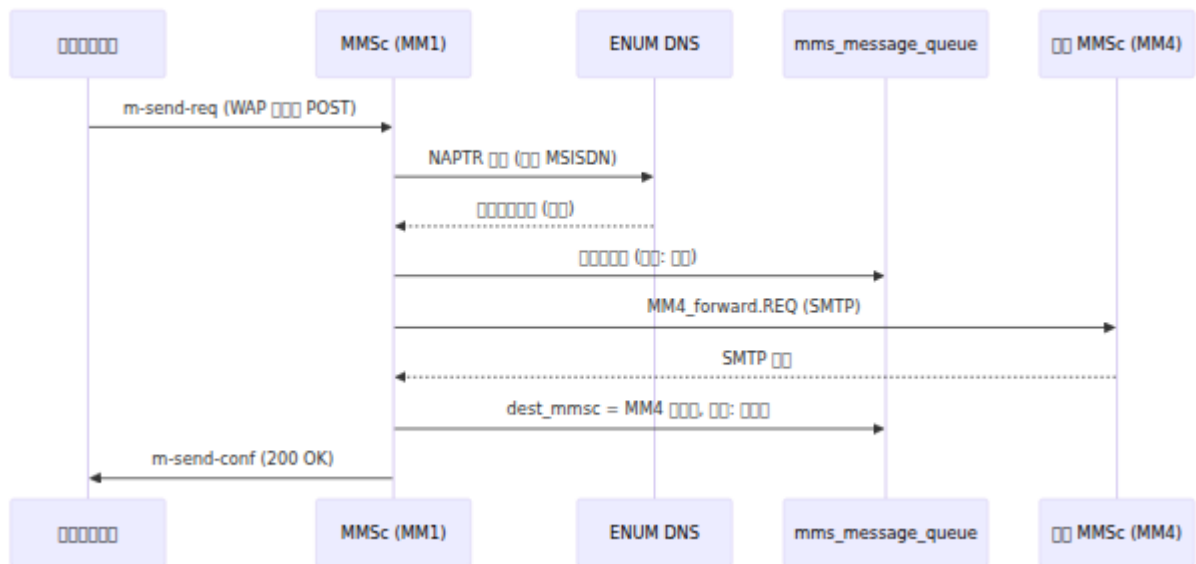
# MM4 ( )

MMS



# MMS

MMS WSP m-notification-ind PDU SMS 8 (DCS 0x44) SMS MT SMS SMS SMS



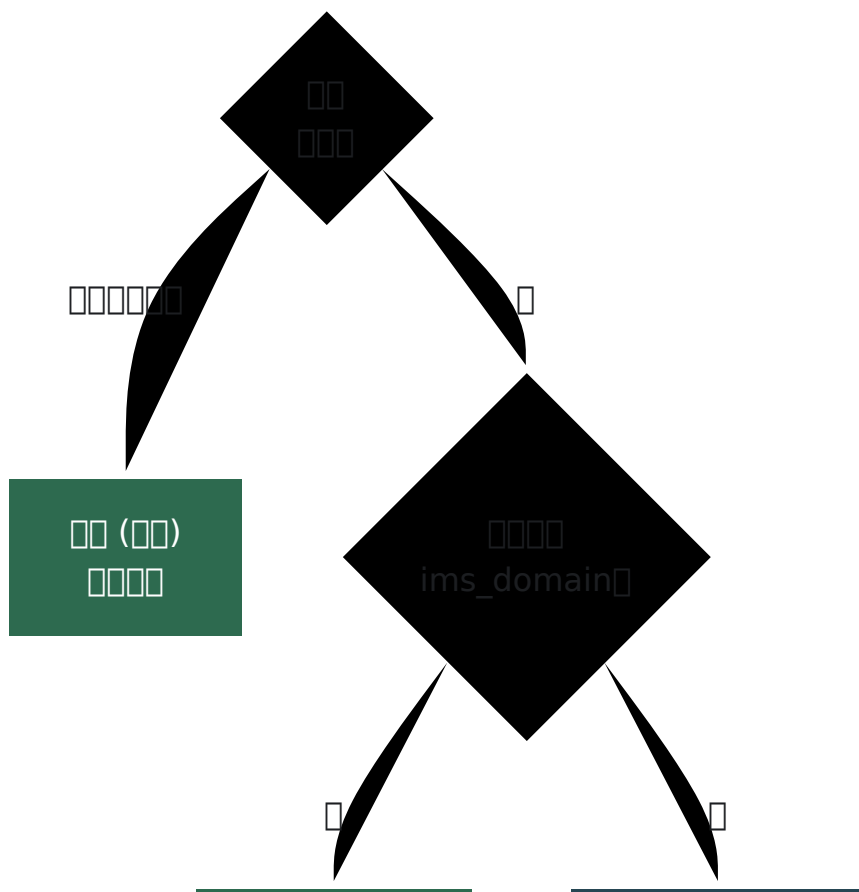
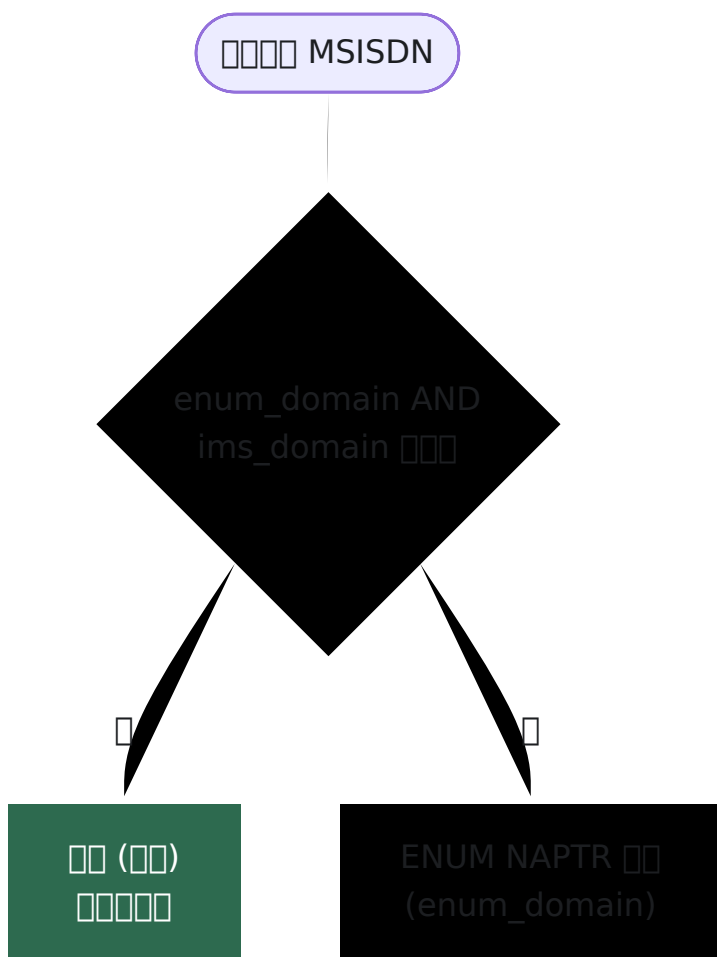
[http://<mms\\_domain>/<unique\\_id>.00](http://<mms_domain>/<unique_id>.00)

- `mms_domain` 为 `mmsc.mms_domain`
- `<unique_id>.00` 为 MM1 分配的 .00 标识符

MM1 的 URL 为 MMS 的 `m-retrieve-conf`

## /

MMSc 为 MM4 的 SMS-C 的 ENUM (NAPTR) 标识符  
/ 标识符 `ims_domain` 的 ENUM 标识符



MMSc  
SMS

MM4  
MM4

enum	enum	enum
ims_domain enum_domain	enum (enum)	enum SMS
ENUM NAPTR enum ims_domain	enum	enum SMS
ENUM NAPTR enum ims_domain	enum	enum MM4 enum
ENUM enum	enum (enum)	enum

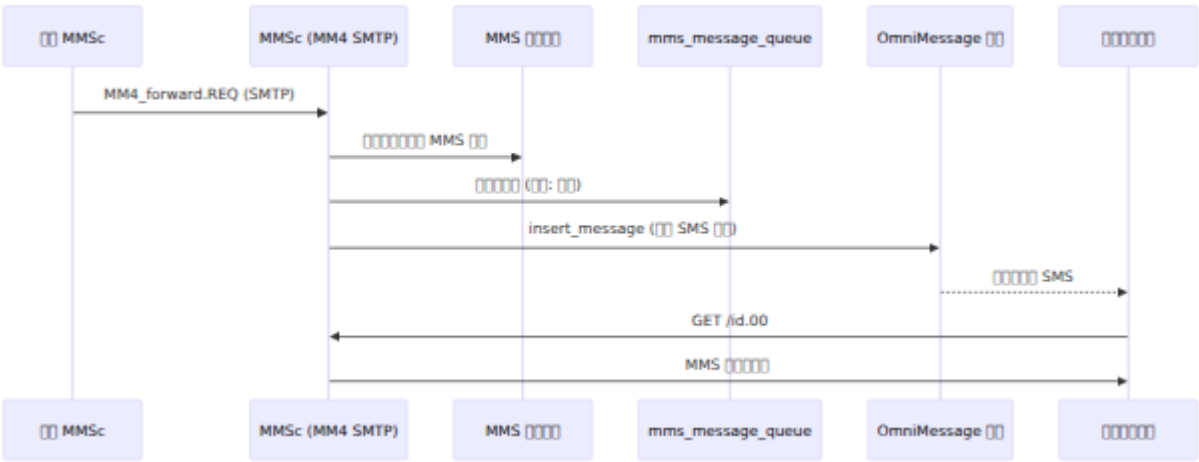
ENUM enum (enum) enum MM4 enum

enum

enum runtime.exs enum :sms\_c enum :mmsc enum MMSc enum :mmsc enum  
enum MMSc enum

enum

enum MMSc enum



□□□□

```
# config/runtime.exs
config :sms_c, :mmsc,
  # □□
  enabled: true,
  port: 8080,
  host: "0.0.0.0",
  mms_domain: "mms.example.com",
  message_path: "/var/lib/omnimessage/stored_mms",
  notify_source_msisdn: "6675",
  dead_letter_minutes: 1440,

  # □□/□□ (ENUM) □□
  enum_domain: "e164.arpa",
  ims_domain: "ims.mnc001.mcc001.3gppnetwork.org",

  # □□□□□□□□ (MM1 □□□ IP □ MSISDN)
  pcrf_api_url: nil,
  hss_api_url: nil,

  # □□/□□ MM4 (□□□□)
  mm4_enabled: false,
  mm4_smtp_port: 25,
  mm4_domain: "mms.example.com",
  default_route_mm4: nil
```

**:mmsc** 配置

配置项	数据类型	默认值	说明
enabled	boolean	false	是否启用MMSC服务
port	integer	8080	MMSC服务端口
host	string	0.0.0.0	MMSC服务监听地址
mms_domain	string	-	MMSC域名
message_path	string	/var/lib/omnimessage/stored_mms	MMSC消息存储路径
notify_source_msisdn	integer	6675	MMSC通知源号码
dead_letter_minutes	integer	1440	MMSC死信保留时间

項目	型	単位	初期値	説明
enum_domain	enum	なし	-	ENUMドメインの識別子。ENUMドメインの識別子は、ENUMドメインの識別子と一致する必要がある。
ims_domain	string	なし	""	IMSドメインの識別子。IMSドメインの識別子は、IMSドメインの識別子と一致する必要がある。
pcrf_api_url	string	なし	nil	PCRF APIのURL。PCRF APIのURLは、PCRF APIのURLと一致する必要がある。MS
hss_api_url	string	なし	nil	HSS APIのURL。HSS APIのURLは、HSS APIのURLと一致する必要がある。ID (pci) 識別子
mm4_enabled	boolean	なし	false	MM4が有効かどうか。MM4が有効かどうか。MM4
mm4_smtp_port	integer	なし	25	MM4 SMTPポート番号。MM4 SMTPポート番号
mm4_domain	string	なし	mm4_domain	MM4ドメインの識別子。MM4ドメインの識別子は、MM4ドメインの識別子と一致する必要がある。MM4
default_route_mm4	string	なし	nil	MM4デフォルトルート。MM4デフォルトルート

MM4	MM4	MM4	MM4	MM4
				: mr MM

MM4

MM4

```
# config/runtime.exs
config :sms_c, :mm4_peers, %{
  "carrier-a" => [
    %{host: "mm4.carrier-a.example.net", port: 25, mms_domain:
      "mms.carrier-a.example.net"}
  ]
}
```

mm4.default\_route\_mm4

MM4

MM4	MM4	MM4	MM4	MM4
host			-	MM4 SMTP IP SMTP HELO
port			25	MM4 SMTP
mms_domain			" "	MMS RCPT (+ <msisdn>/TYPE=PLMN<mms_domain>) dest_mmsc





MMS 消息 通过 Mnesia 的 `:mms_message_queue` 消息队列接收 SMS 消息并存储在  
消息数据库中 MMS 消息 通过 `.mms` 消息 `message_path` 消息队列接收并存储在

## mms\_message\_queue

項目	説明
mms_id	Mnesia 識別子
unique_id	URL 識別子 (<unique_id>.00)
source_msisdn	送信元 MSISDN
destination_msisdn	宛先 MSISDN
recipient_list	MMS 宛先リスト To/Cc (CSV)
subject	MMS 件名
file_path	.mms ファイル message_path 参照
source_mmsc	送信元 MMSc 識別子 MM4 参照
dest_mmsc	宛先 MMSc 識別子 MM4_<peer> 参照
on_net	MMSc 接続状態
notify_message_ids	通知 SMS ID SMS 参照
send_time	MMSc 送信 MMS 時刻
notify_time	通知 SMS 時刻
retrieve_time	MMS 取得時刻
expires	MMS 有効期限
status	状態
deadletter	死文字列

状態	説明
<code>inserted_at</code> / <code>updated_at</code>	タイムスタンプ

メッセージ



stored

通知 SMS

notified

MM4 状態 / `dead_letter_minutes` 属性

取得 MMS

状態 / `dead_letter_minutes`

retrieved

forwarded

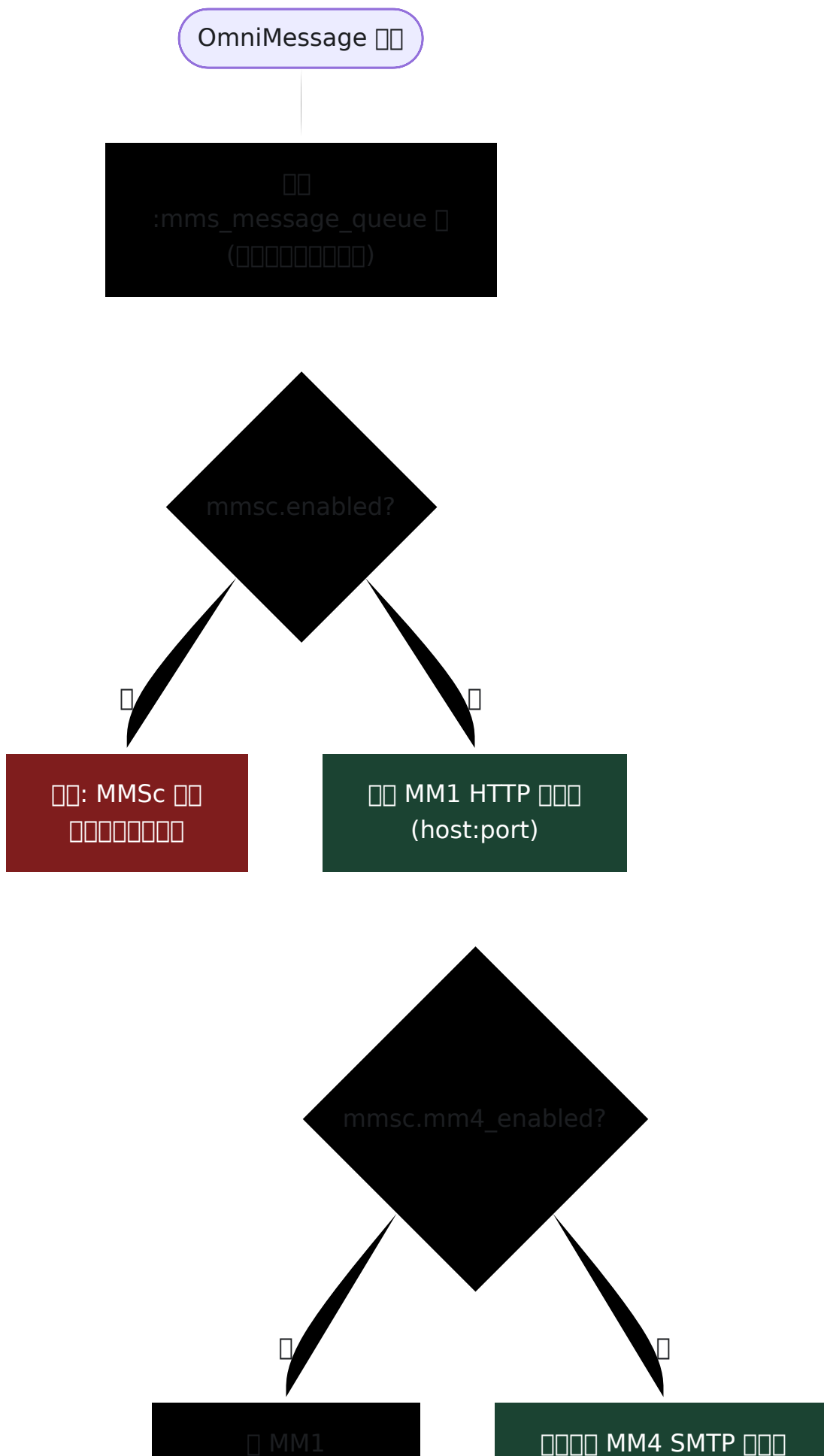
expired

failed



状態	説明
<code>stored</code>	メッセージがデータベースに保存された状態
<code>notified</code>	SMS 通知が送信された状態
<code>retrieved</code>	MM1 状態から MMS 状態に移行した
<code>forwarded</code>	MM4 状態に移行した
<code>expired</code>	<code>dead_letter_minutes</code> 属性が満了した状態
<code>failed</code>	メッセージが失敗した状態





(MM4 設定)

(mm4\_smtp\_port)

- **MM1 HTTP** 設定は `mmsc.enabled` を `true` に設定し、`mmsc.host:mmsc.port` に接続し、`m-send-req`、`m-notifyresp-ind`、`m-acknowledge-ind` のリクエストを `GET /<unique_id>.00` として送信する。
- **MM4 SMTP** 設定は `mmsc.enabled` を `mmsc.mm4_enabled` に設定し、`mmsc.mm4_smtp_port` に接続し、`MM4_forward.REQ` のリクエストを送信する。
- **MM4** は、SMTP 接続を保持する。
- 設定された `MMSc` のメッセージキューは `:mms_message_queue` に設定されている。

MM4 (25) は `OmniMessage` を送信する。  
(`CAP_NET_BIND_SERVICE`) は `mm4_smtp_port` の `MM1` 接続を 8080 にバインドする。

**TLS** は `MM1` 接続を `HTTP` から `TLS` にアップグレードする。  
`mmsc.port` は `TLS` 接続のポート番号である。

設定

`MMSc` は `MMS Prometheus` と `SMS` の接続を確認する。  
`OmniMessage` は `MMS` の `CDR` と `SMS` の接続を確認する。

設定された `status`、`send_time`、`notify_time`、`retrieve_time` の値を確認する。

設定

## MMSc 設定

設定された `MM1` の `MMSc: DISABLED` の状態を確認する。

設定

- `:mmsc` は `runtime.exs` に設定されている。
- `mmsc.enabled` は `false` に設定されている。

設定

1. 设置 `:mmsc` 属性为 `enabled: true`
2. 检查配置是否正确 `MMSc: ENABLED`

## 配置 MMS

配置 SMS 属性

配置

- `mms_domain` 设置 MMS APN 属性
- 设置 `mmsc.port`
- 设置 `.mms` 属性 `message_path`

配置

1. 设置 URL 属性 (`mms_domain`) 设置 MMS APN 属性
2. 设置 `mmsc.host:mmsc.port` 属性 TLS 属性
3. 设置 `status` 属性 `.mms` 属性 `file_path`

## 配置 MMS 属性

配置 MMS 属性 `status` 属性 `stored`

配置

- `default_route_mm4` 属性
- `:mmsc_mm4_peers` 属性
- 设置 `ims_domain/enum_domain` 属性

配置

1. 设置 `default_route_mm4` 属性 `:mmsc_mm4_peers` 属性
2. 设置 `host/port` 属性 SMTP
3. 设置 `enum_domain` 属性 `ims_domain` 属性

## MM4

MM4

- `mm4_enabled` `false` SMTP
- 25
- SMTP

1. `mm4_enabled: true`
2. `CAP_NET_BIND_SERVICE` `mm4_smtp_port`
3. `mm4_smtp_port`

- `pcrf_api_url` / `hss_api_url` MSISDN
- PCRF/HSS 200

1. `pcrf_api_url` `hss_api_url`
2. /MSISDN



# SMS-C

←

SMS-C is a system for managing SMS messages. It is built using Mnesia for database storage and Erlang for the backend logic.

- SMS-C is a system for managing SMS messages.
- It is built using Mnesia for database storage and Erlang for the backend logic.
- **SMSC** is the main component for sending and receiving SMS messages.
- It supports sending SMS messages to phone numbers in the range 1-255.
- It also supports receiving SMS messages from phone numbers.
- The system is designed to be scalable and reliable.
- It uses a `runtime.exs` file for configuration.
- The system is built using Erlang and Mnesia.
- **Web UI** is used for managing the system.
- It includes a REST API for managing SMS messages.
- It also includes a command-line interface for managing the system.
- The system is designed to be easy to use and integrate with other systems.

項目	型	説明	注
rule_id	int	ルールID	必須
calling_prefix	string/nil	発信番号表示番号nil = 未設定	○
called_prefix	string/nil	着信番号表示番号nil = 未設定	○
source_smsc	string/nil	送信SMS番号nil = 未設定	○
calling_match	string/nil	発信番号表示番号とTON/NPIの一致	○
calling_replace	string/nil	発信番号表示番号の置換	○
called_match	string/nil	着信番号表示番号の一致	○
called_replace	string/nil	着信番号表示番号の置換	○
source_ton	string/nil	送信番号表示番号/TON = 未設定	○
source_npi	string/nil	送信番号表示番号/NPI = 未設定	○
priority	int	優先度1-255	○
description	string	説明	○
enabled	bool	有効/無効	○
continue	bool	続行false	○

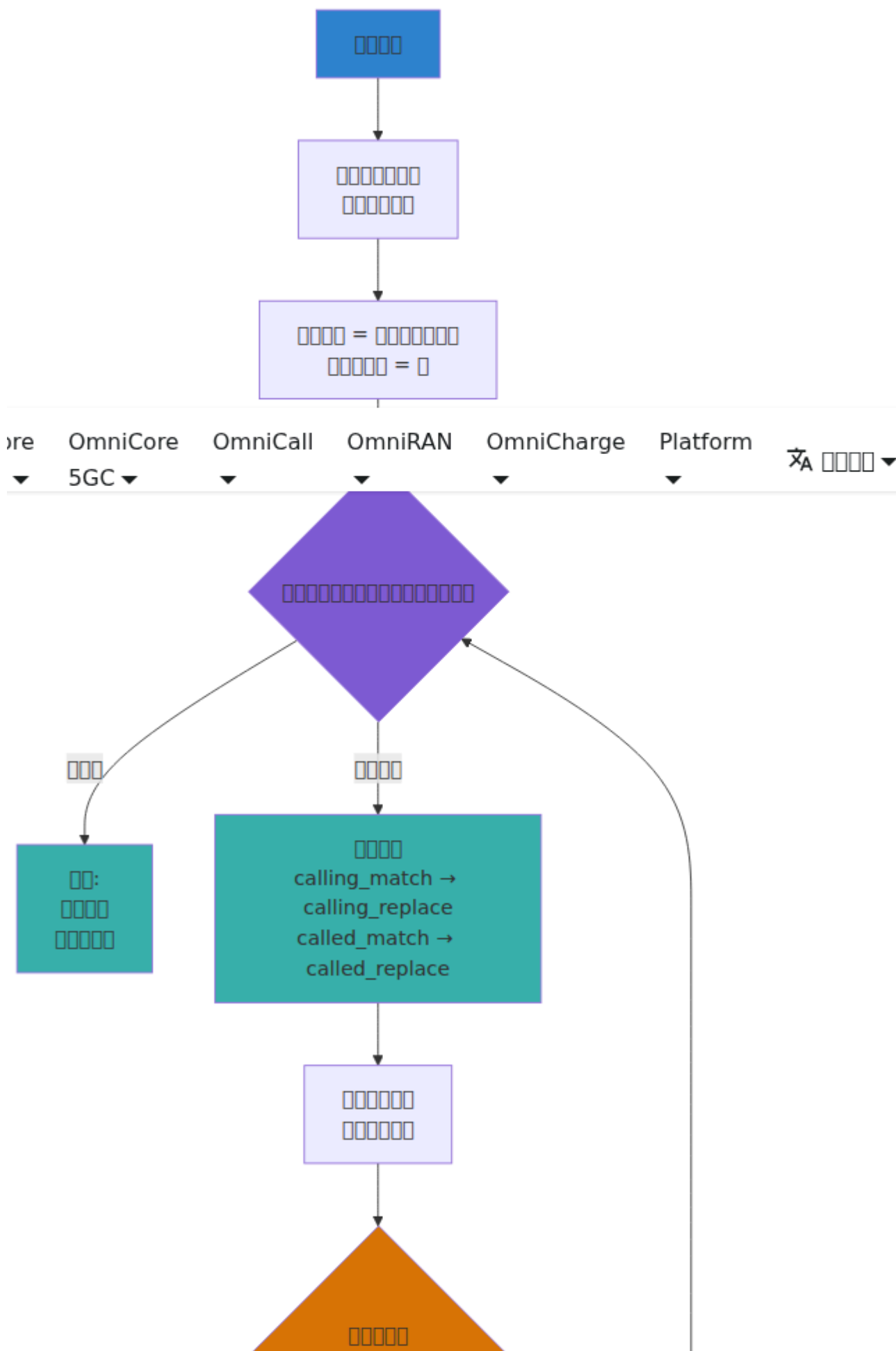
送信番号表示番号とTON/NPIの一致

送信番号表示番号と送信番号表示番号/TONと送信番号表示番号/NPI

必須

送信番号表示番号







# TON/NPI

TON/NPI unknown/unknown

TON/NPI

source_ton		SMS-DELIVER TPDU
source_npi		SMS-DELIVER TPDU

= -

source\_ton/source\_npi TPDU IMS/SS7 SMS-DELIVER TPDU - TON/NPI SMS-DELIVER TPDU

TON

種別	説明
unknown	不明
international	国際標準E.164に準拠
national	国内標準
network_specific	ネットワーク固有
subscriber	ダイヤル番号
alphanumeric	英数字ID
abbreviated	略称

国際標準NPI

種別	説明
unknown	不明
isdn	ISDN/E.164
data	X.121
telex	Telex
land_mobile	E.212
national	国内標準
private	私用
ermes	ERMES

国際標準NPI





continue: true 当TON/NPI 为国际或 ISDN 号码时，继续处理下一个号码。  
否则，返回 false。

说明

TPDU 中的 TON 和 NPI 字段

长度	TON	NPI
长度 ≥ 5	international	isdn
长度 < 5	national	isdn
号码为 Acme	alphanumeric	unknown

当 TON 和 NPI 均为国际或 ISDN 号码时，继续处理下一个号码。

## TON/NPI

说明

- Web UI - 在 TON/NPI 翻译规则中，将“”替换为。
- REST API - /api/translations 中的 source\_ton 和 source\_npi 字段。
- 配置 - config :sms\_c, :translation\_rules 中的 translation\_rules 字段。

示例

3-6 中的 TON/NPI 为 unknown/unknown 时，calling\_match 为 true，calling\_replace 为 false。

config/runtime.exs

```
config :sms_c, :translation_rules, [
  %{
    calling_match: "^\d{3,6}$",
    calling_replace: nil,
    source_ton: "unknown",
    source_npi: "unknown",
    priority: 5,
    continue: true,
    description: "□□□□3-6□□=> unknown/unknown TON/NPI",
    enabled: true
  }
]
```

```

calling_prefix[]called_prefix[]source_smsc[]
calling_match[]^\\d{3,6}$[]source_ton/source_npi[]3
[]6[]unknown/unknown[]calling_replace[]nil[]
continue: true[]

```

national 4 international 5-6

## REST API - POST /api/translations

```
{
  "calling_match": "^\\d{3,6}$",
  "calling_replace": null,
  "source_ton": "unknown",
  "source_npi": "unknown",
  "priority": 5,
  "continue": true,
  "description": "□□□□□□3-6□□=> unknown/unknown TON/NPI",
  "enabled": true
}
```

API rule\_id TON/NPI

[illegible]

national/isdn

config/runtime.exs

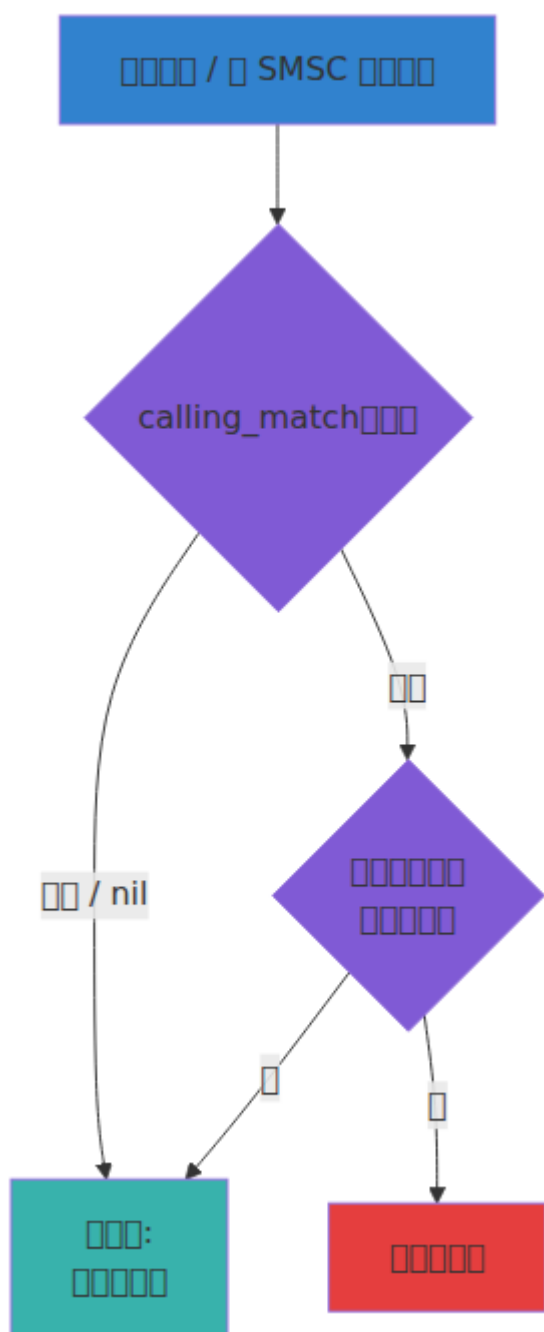
```
%{
  calling_prefix: "07",
  calling_match: "^07\d{8}$",
  calling_replace: nil,
  source_ton: "national",
  source_npi: "isdn",
  priority: 20,
  continue: false,
  description: "07xxxxxxxx => national/isdn",
  enabled: true
}
```

calling\_prefix: "07" 07 calling\_match  
07 8 national/isdn international/isdn  
≥ 5

national TON

config/runtime.exs

Web UI



□□□□

```

# 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
  },

  # 000000000000
  %{
    calling_prefix: "00",
    called_prefix: nil,
    source_smsc: nil,
    calling_match: "^00(.+)$",
    calling_replace: "+\\1",
    called_match: nil,
    called_replace: nil,
    priority: 5,
    description: "000000000000+",
    enabled: true,
    continue: true # 00000000
  },

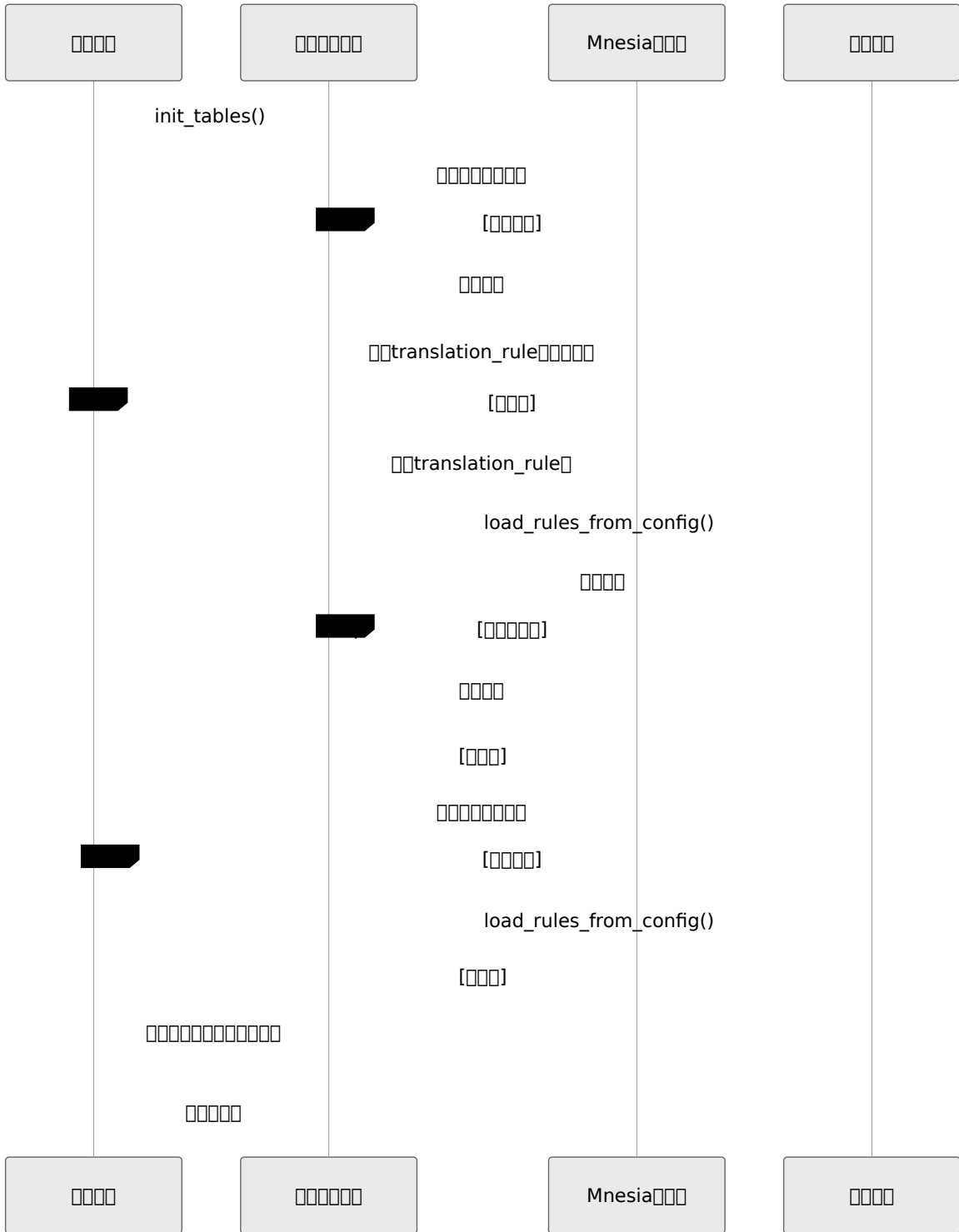
  # 0000000000unknown/unknown TON/NPI0000000000
  %{
    calling_match: "^(\\d{3,6})$",
    calling_replace: nil,
    source_ton: "unknown",
    source_npi: "unknown",
    priority: 6,
    description: "00003-600=> unknown/unknown TON/NPI",
    enabled: true,
    continue: true
  },

```

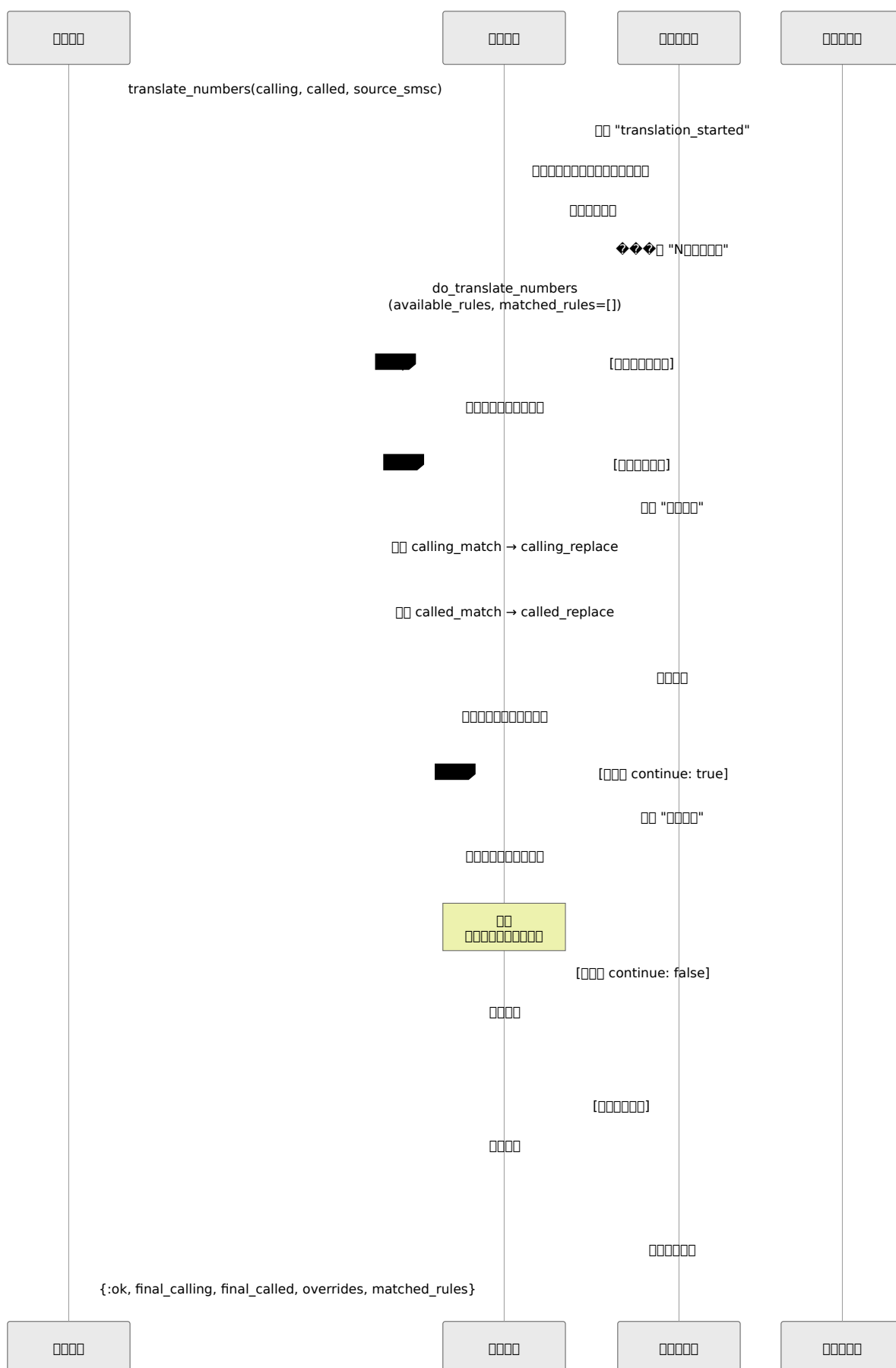
```
# [][][][][][]
%{
  calling_prefix: "+44",
  called_prefix: "+44",
  source_smsc: nil,
  calling_match: "^\\+44(.*?)$",
  calling_replace: "0044\\1",
  called_match: "^\\+44(.*?)$",
  called_replace: "0044\\1",
  priority: 20,
  description: "[][][][][][]",
  enabled: true,
  continue: false
}
]
```

□□

□□□□□



--	--	--	--	--	--

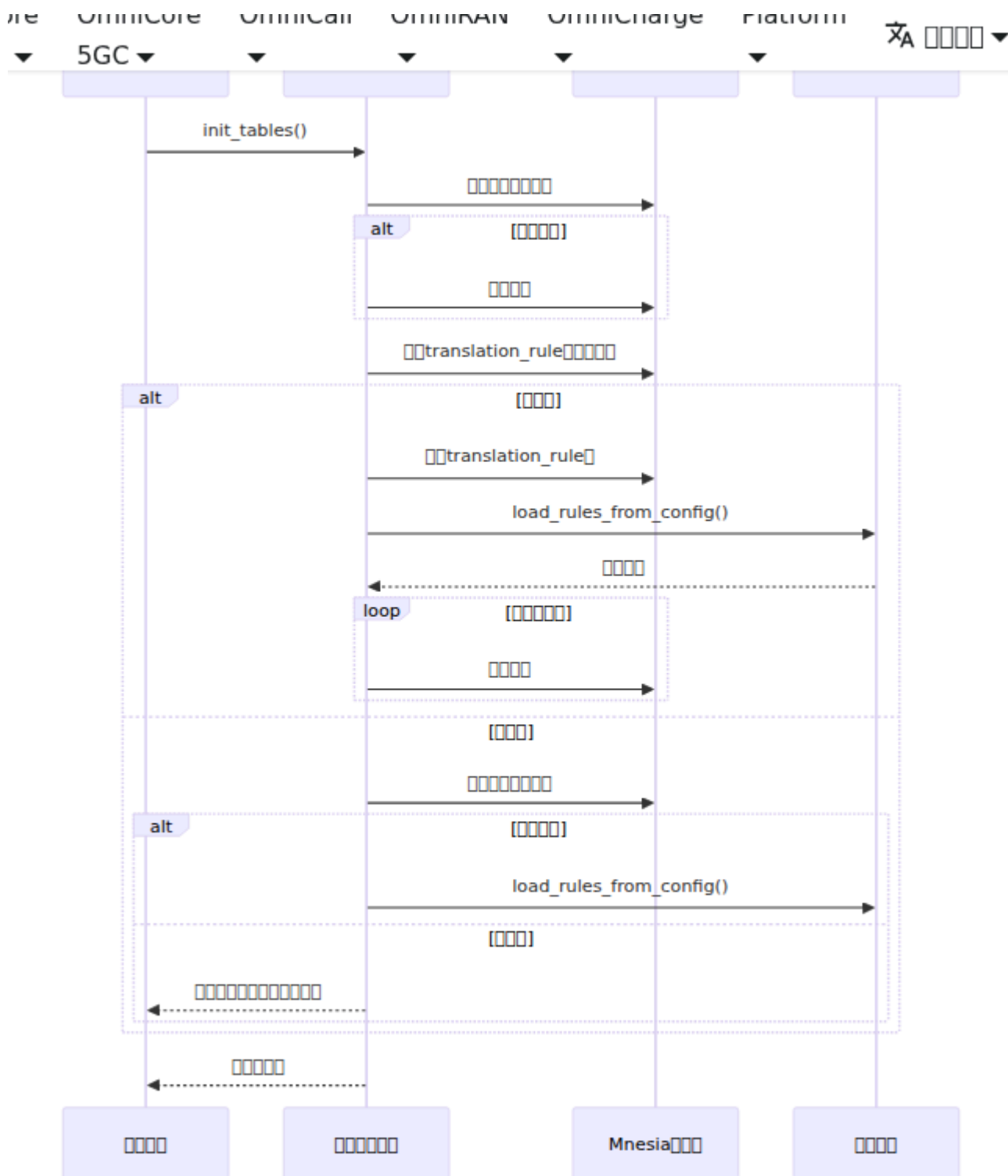




□□□□

□□□□□□□

□□□□□□□□□□E.164□



□□□□□□□

□□□□□□□□□□□□□□□□□□

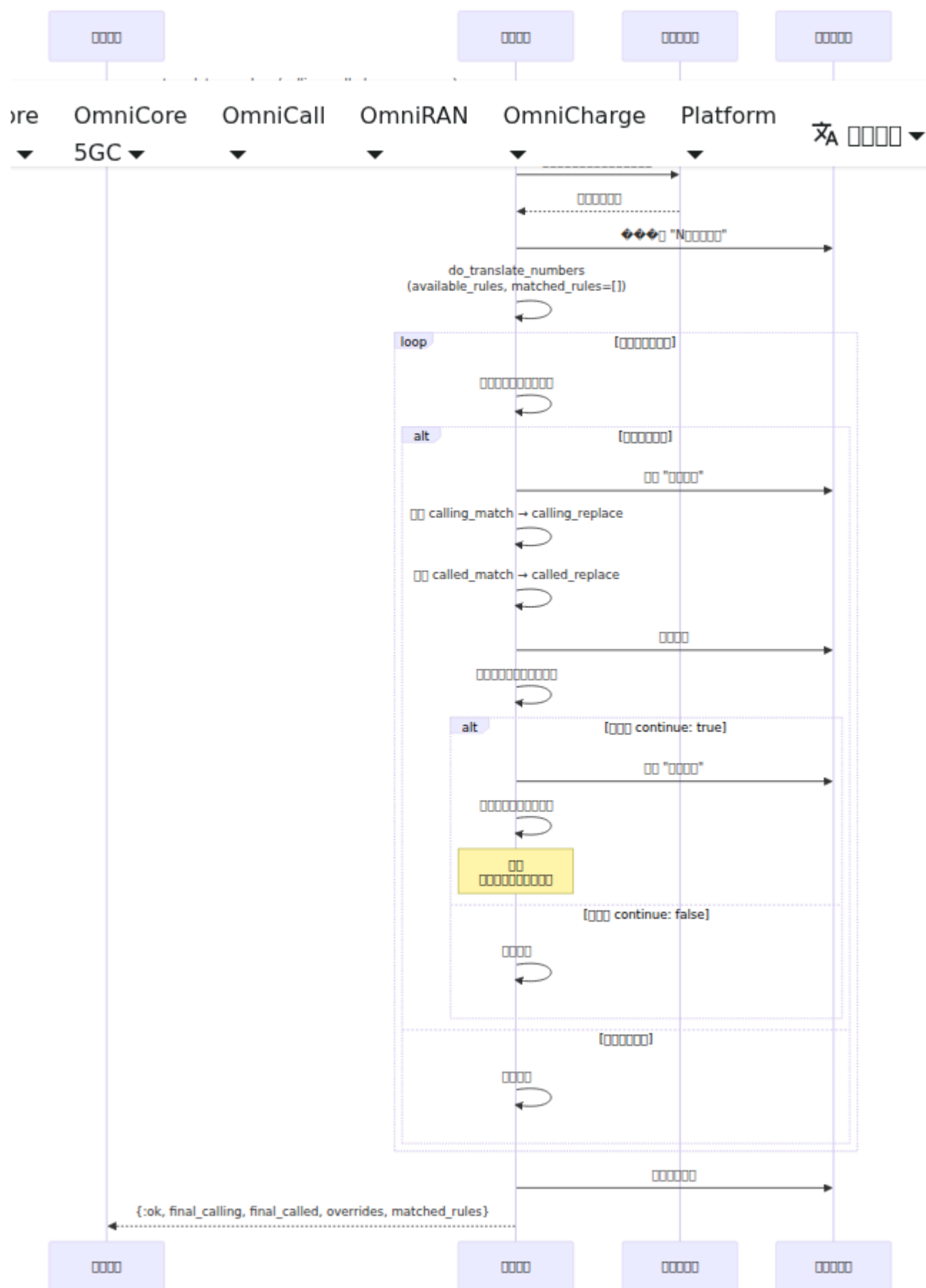
Parse error on line 2: ...chart TD I[□□: "5551234567"] --> S1[ -----  
^ Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND',  
'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND\_STOP', 'TAGEND',  
'TRAPEND', 'INVTRAPEND', 'UNICODE\_TEXT', 'TEXT', 'TAGSTART', got 'STR'



**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. □□□□□□□□□□□□
  - □□□□□□□□□□□□



- 國際標準化組織ISO
- 國際電訊聯盟ITU
- 國際標準化組織ISO
- 國際標準化組織ISO
- 國際標準化組織ISO 100000

國際標準

#### 1. 國際標準

- 國際標準
- 國際標準
- 國際標準

#### 2. 國際標準

#### 3. 國際標準

- 國際標準
- 國際標準
- 國際標準
  - 國際標準
  - 國際標準
  - 國際標準
  - 國際標準
  - 國際標準
  - 國際標準

#### 4. 國際標準

#### 5. 國際標準

國際標準

□□□□

□□□□: 5551234567 → +1-555-123-4567  
□□□□: 5558720155 → +1-555-872-0155  
✓ □3□□□□□□

□□□□

┌ □□1 ───────────────────┐  
│ □□ #1 (□□□□10)            ↓ □□ │  
│ □10□□□□□□□□□□            │  
│  
│ □□: 5558720155 → +15558720155    │  
└──────────────────────────┘

┌ □□2 ───────────────────┐  
│ □□ #2 (□□□□20)            ↓ □□ │  
│ □□□□□□□□□□            │  
│  
│ □□: +15558720155 → +1-555-8720155    │  
└──────────────────────────┘

┌ □□3 ───────────────────┐  
│ □□ #3 (□□□□30)            │  
│ □□□□□□□□            │  
│  
│ □□: +1-555-8720155 → +1-555-872-0155 │  
└──────────────────────────┘



# API

## API



## API

### translate\_numbers

- calling\_number
- called\_number
- source\_smsc
- message\_id

## API

- {:ok, translated\_calling, translated\_called, overrides, [rules\_applied]} -
- overrides %{:source\_ton:, source\_npi:} nil
- 
- 
-

```

# []
{:ok, new_calling, new_called, _overrides, rules} =
  NumberTranslation.translate_numbers(
    calling_number: "5551234567",
    called_number: "5558720155",
    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
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()
```



- 00000003-4000000000

### 3. 00000

- 000000000
- 00000000000000“5551234567 → +15551234567”
- 0000000000/00

### 4. 0000000

- 000000000000000
- 000000\1\20000000
- 0000000000000000

## 00

### 1. 00000000

- 000000000
- 0000000000000
- 00000000000

### 2. 0000000

- 000000000000000
- 00000000
- 00000000

### 3. 000000000

- 000500000000000
- 00000000000000
- 00Telemetry00???

## ???

### 1. 000000

- 0000000000

- 000000000000000000
- 00000000

## 2. 00000

- 0000000000000000
- 0000000000
- 000000000000

## 3. 00000

- 0000ID00000000
- 0000000000000000
- 00000000

## 4. 00000

- 000000000000
- 0000000000
- 00000
- 000000000

## 00000

### 1. 00000

- 10000000000<sup>^(\d{10})\$</sup>
- 00000<sup>^+(\d+)\$</sup>
- 000000<sup>^0+(.+)\$</sup>
- 000000<sup>^(\d{3})(\d{3})(\d{4})\$</sup> → \1-\2-\3

### 2. 0000

- 0000000000<sup>^(\d{3})(\d{7})\$</sup>
- 00000000<sup>+1\1\2</sup>
- 00000<sup>^+(\d{1,3})(\d+)\$</sup> → 00\1\2

### 3. 00000000

- 匹配 \.
- 匹配 \+
- 匹配 \(\| \)

匹配

匹配

匹配

匹配

- 匹配
- 匹配SMSC
- 匹配
- 匹配
- 匹配continue: false

匹配

1. 匹配
2. 匹配/
3. 匹配
4. 匹配
5. 匹配

匹配

匹配

匹配

- 匹配
- 匹配
- 匹配1\2

□□□□

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

□□□□/□□□□

□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□

□□□□

- □□□□□□□□
- □□□□□□□□
- □□□□□□

□□□□

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

□□□□□□□□□□

□□□□□□□□□□□□

□□□□

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



□□□□

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

□□□□□□□□□□

□□□□□□□□□□□□

□□□□

- □□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□

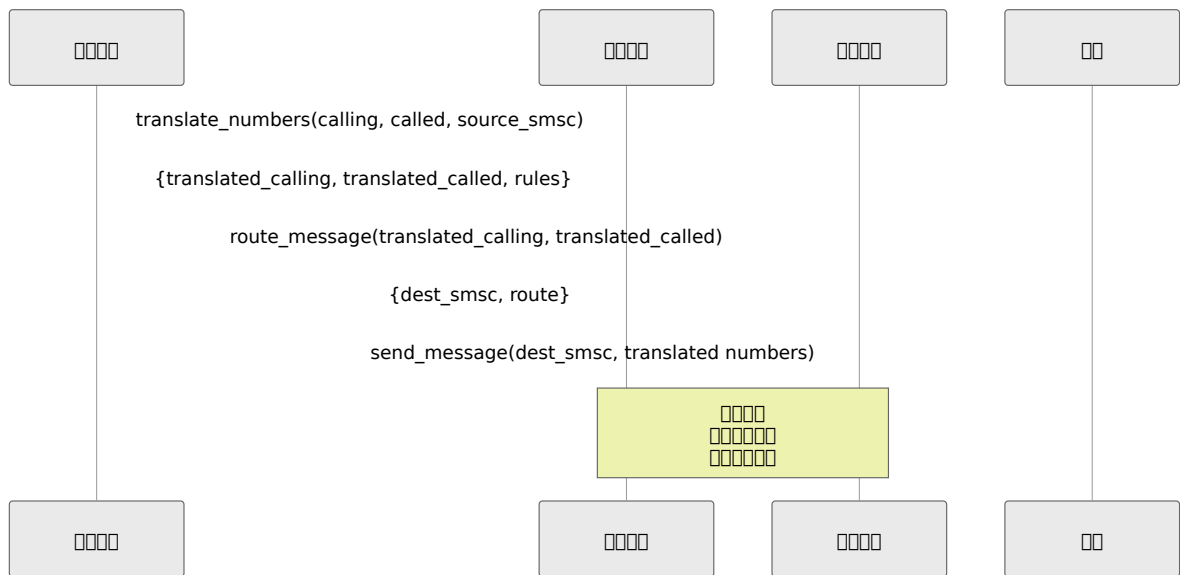
□□□□

1. □□□□□□□□□□□□□□□□
2. □□□□□□□□
3. □□□□□□□□□□□□
4. □□□□□□□□□□translate\_numbers

□□□□

□□□□□□

□□□□□□□□□□□□□□□□



## EventLogger

EventLogger

- translation\_started:
- translation\_candidates:
- translation\_matched:
- translation\_calling:
- translation\_called:
- translation\_continue: continue=true
- translation\_none:

message\_id translate\_numbers/1

## 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
)
```

□□□□□□

- 00000000p500p950p990
- 0000000000
- 00000000000000
- continue00000000

□ □

Mnesia

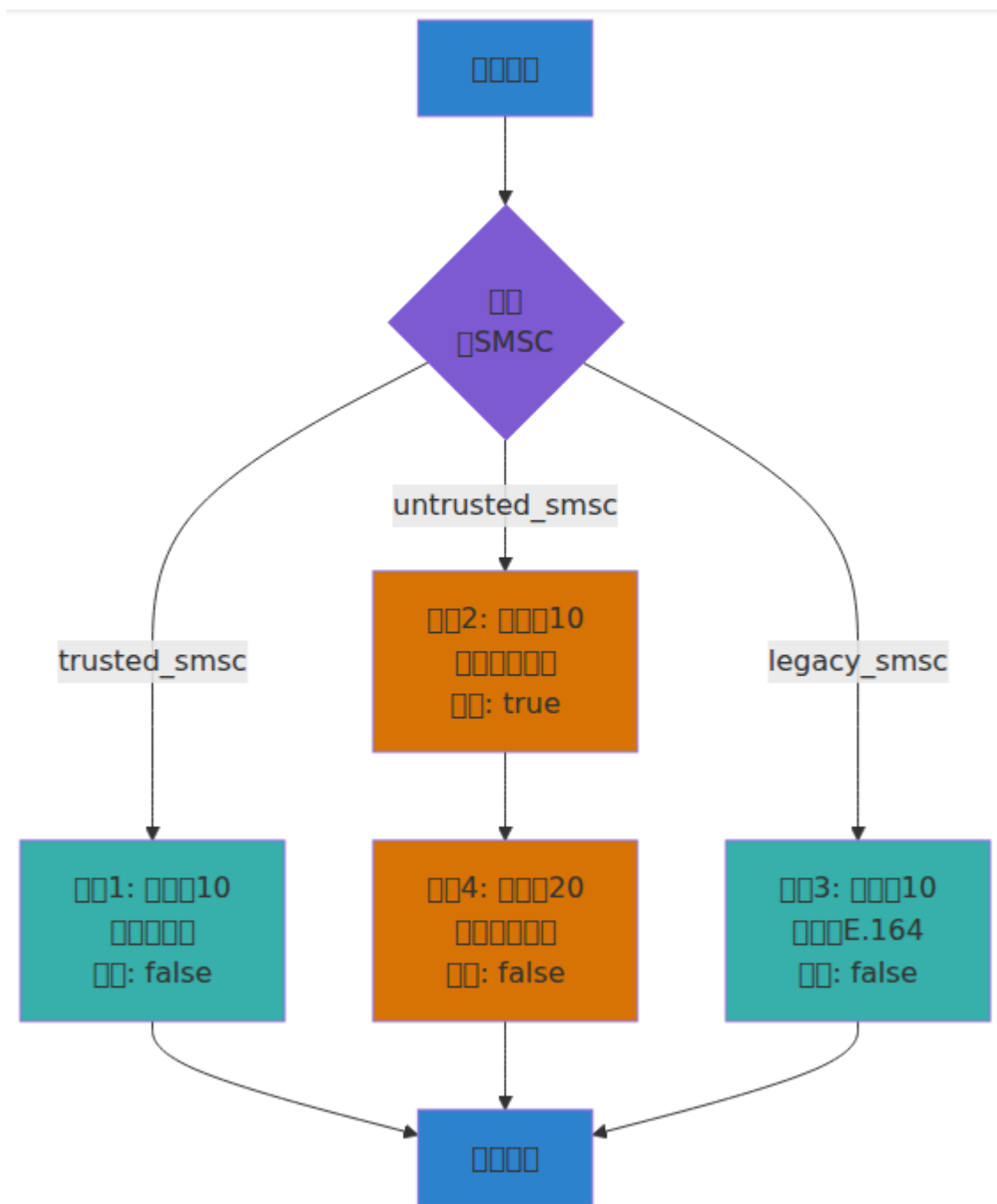
---

```
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'
```



□ □ □ □

[illegible]



00

00**1**000000000

0000000000000000E.164 (+1XXXXXXXXXX)

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

# 21 + 10
%{
  calling_match: "^1(\\d{10})$",
  calling_replace: "+1\\1",
  called_match: "^1(\\d{10})$",
  called_replace: "+1\\1",
  priority: 10,
  description: "1XXXXXXXXXX+1XXXXXXXXXX",
  enabled: true,
  continue: false
}

# 
# "5551234567" → "+15551234567"1
# "15551234567" → "+15551234567"2
# "+15551234567" → "+15551234567"0000000000
```

2

0000000000+0000000000

```
# 00100000000+000000000000
%{
    calling_match: "^00(.+)\$",
    calling_replace: "+\\1",
    called_match: "^00(.+)\$",
    called_replace: "+\\1",
    priority: 5,
    description: "0000000000+",
    enabled: true,
    continue: true # 00000
}

# 002000000000000000000000
%{
    calling_match: "^\\+(\\d+)\$",
    calling_replace: "00\\1",
    called_match: "^\\+(\\d+)\$",
    called_replace: "00\\1",
    priority: 10,
    description: "[+0000000000000000]",
    enabled: true,
    continue: false # 00
}

# 000000
# 001: "00441234567890" → "+441234
# 002: "+441234567890" → "00441234
# 00: "00441234567890"
# 00000: [001, 002]
```

### 3 SMSC

□□□□□SMSC□□□□□□□□

```

# 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
}

# 003000SMSC0000000000100
%{
  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
}

# 020000000000000000
%{
  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
}

# 03000000000000
%{
  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
# 00: "005551234567"
# 001: "005551234567" → "5551234567"00010000
# 002: "5551234567" → "+15551234567"00020000
# 003: "+15551234567" → "+1-555-123-4567"00030000

```

```
# 電話: "+1-555-123-4567"  
# 電話番号: [001, 002, 003]
```

00

00000000

- 00000000000000000000
- 0000000000000000
- 00Mnesia0000:mnesia.table\_info(:translation\_rule, :size)
- 00Telemetry0000000000

# SMS-C 繁體中文

← 繁體中文

SMS-C 繁體中文說明書

## 目錄

- 簡介
- 安裝
- 配置
- 使用
- 維護
- 故障排除
- 安全
- 性能
- 兼容性
- 聯繫我們

## 快速入門

### 安裝與配置

本節將指導您完成安裝和配置。

#### 1. 系統要求

```
# API 地址
curl https://api.example.com:8443/api/status

# 請求頭
# {"status":"ok","application":"OmniMessage","timestamp":"2025-10-30T08:00:00Z"}
```

## 2. Prometheus 指标

Prometheus 指标列表

- 指标名称 24 小时
- 指标名称 < 1%
- 指标名称 < 1000 个
- 指标名称 > 95%
- 指标名称 1000 个

## 3. 配置项

Web UI: [https://sms-admin.example.com/message\\_queue](https://sms-admin.example.com/message_queue)

配置项

- 配置项名称
- 配置项名称 < 5 个
- 配置项名称 > 3 个
- 配置项名称 1000 个

## 4. 配置项

Web UI: [https://sms-admin.example.com/frontend\\_status](https://sms-admin.example.com/frontend_status)

配置项

- 配置项名称
- 配置项名称
- 配置项名称 24 小时

## 5. 配置项

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

配置项

- 配置项名称
- 配置项名称

- 消息
- 消息接收
- 消息发送

消息接收

消息发送

通过 Prometheus 监控

```
# 消息接收
increase(sms_c_message_received_count[1h])

# 消息发送
increase(sms_c_delivery_succeeded_count[1h])

# 消息成功率
rate(sms_c_delivery_succeeded_count[1h]) /
rate(sms_c_message_received_count[1h])
```

消息接收

- 消息接收成功率
- 消息接收/发送成功率
- 消息接收成功率 > 95%

消息发送

- 消息发送成功率 > 50% 消息
- 消息发送成功率 > 200% 消息
- 消息发送成功率 > 90% 消息

消息

消息接收

消息发送

短信接收 (sms\_c\_message\_received\_count)

- 短信接收量
- 短信接收量
- 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)`

短信

短信路由失败 (sms\_c\_routing\_failed\_count)

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

短信路由匹配 (sms\_c\_routing\_route\_matched\_count)

- 短信路由匹配量
- 短信路由匹配量
- 短信 `rate(sms_c_routing_route_matched_count)`

短信

短信

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

短信成功尝试 (sms\_c\_delivery\_succeeded\_attempt\_count)

- 短信成功尝试量

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

过滤

过滤 (`sms_c_queue_size_size`)

- 过滤
- `size > 10,000`
- `sms_c_queue_size_size`

过滤 (`sms_c_queue_oldest_message_age_seconds`)

- 过滤
- `age > 300`
- `sms_c_queue_oldest_message_age_seconds`

聚合

聚合

### 1. 聚合

- 聚合 5 分钟
- 聚合 5 分钟
- 聚合 24 小时

### 2. 聚合

- 聚合
- 聚合
- 聚合

### 3. 聚合

- 聚合
- 聚合

- 00000000 24 00

#### 4. 00000000

- 00 ID
- 000000000000
- 00 SMSC
- 000

#### 5. 00000000

- 0000
- 000000/000
- 0000
- 000000000000

#### 6. 0000000000

- API 00000p95
- 00000000p95
- ENUM 00000p95

0000

00000000000000



```

# 🚨 - 
- alert: RoutingFailures
  expr: increase(sms_c_routing_failed_count[5m]) > 0
  severity: critical
  description: "{{ $value }}" 5 
# - 
- alert: QueueBacklog
  expr: sms_c_queue_size_pending > 10000
  severity: critical
  description: " {{ }}"
# - 
- alert: OldMessagesInQueue
  expr: sms_c_queue_oldest_message_age_seconds > 300
  severity: critical
  description: " {{ }}"
# - 
- alert: FrontendDisconnected
  expr: sms_c_frontend_status_count{status="disconnected"} > 0
  severity: critical
  description: "{{ $value }}"

```

```

# 配送遅延
- alert: LowDeliveryRate
  expr: rate(sms_c_delivery_succeeded_count[10m]) /
rate(sms_c_delivery_queued_count[10m]) < 0.90
  severity: warning
  description: "配送遅延 {{ $value }}"

# 再送率
- alert: HighRetryRate
  expr: histogram_quantile(0.95,
sms_c_delivery_succeeded_attempt_count) > 2
  severity: warning
  description: "95th 再送率 {{ $value }}"

# ENUM 検索遅延
- alert: SlowEnumLookups
  expr: histogram_quantile(0.95, sms_c_enum_lookup_stop_duration)
> 5000
  severity: warning
  description: "ENUM 検索遅延 > 5 秒"

# ENUM キャッシュヒット率
- 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 キャッシュヒット率 {{ $value }}"

```

確認

確認

確認 ID

1. **Web UI** `/message_queue`
2. 確認 ID
3. 確認

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 - 1回目
  ↓
4. delivery_attempt_2 - 2回目
  ↓
5. delivery_attempt_3 - 3回目
  ↓
6. message_dead_letter - 失敗

```

## メッセージの状態

### メッセージの状態

- “queued” → “backoff”
- deliver\_after: queued, nil, backoff
- delivery\_attempts: queued (0), backoff

### メッセージの状態

- “delivered” → “sent” → DLR
- deliver\_time
- dest\_smsc

### メッセージの状態

- “backoff” → delivery\_attempts - 30回 → expires
- “expired” → “dropped” → “balance\_rejected” → “auto\_replied”
- ...

## メッセージの状態

SMS-C ...

### メッセージの状態

get\_messages\_for\_smsc(smsc\_name) 返回的JSON数据

1. dest\_smsc 返回的JSON数据
2. 返回的JSON数据 - 返回的JSON数据
  - dest\_smsc 返回的JSON数据
  - destination\_msisdn 返回的JSON数据
  - location 返回的JSON数据
  - 返回的JSON数据

返回的JSON数据

MSISDN +447700900123 返回的JSON数据 uk\_gateway 返回的JSON数据

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

返回的JSON数据

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

uk\_gateway 返回的JSON数据

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

# 消息队列 dest_smsc 为 null
# 消息队列 uk_gateway 消息
```

消息队列

消息队列消息队列消息队列

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

消息队列消息队列

消息队列消息队列

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

# 消息队列消息队列
# expires 消息队列 > 消息队列
```

消息队列

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

消息队列

消息队列消息队列

```
# 更新 delivery_attempts 和 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"
  }'
```

更新成功

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

更新成功

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

删除成功

删除成功

**Web UI** [/sms\\_routing](#)

**API**

```
# 获取路由
curl https://api.example.com:8443/api/routes
```

获取成功

Prometheus 监控

```
# 00000000000000000000
increase(sms_c_routing_route_matched_count[1h])
```

--	--	--	--	--

## Web UI

1. `/sms_routing`
2. `"XXXXXX"`
3. `XXXXXX`
  - `XXXXXXXXXXXXXXXXXXXX`
  - `XXXXXXXXXXXXXXXXXXXXXXXXXXXX`
  - `1 SMSCXXXXXXXXXXXX`
  - `2 SMSCXXXXXXXXXXXX/XXXXXXXXXXXX`
  - `XXXXXXXXXXXXXXXXXXXXXXXXXXXX Diameter/HSS`
  - `XXXXXXXXXXXX1-255XXXXXXXXXXXX`
  - `XXXXXXXXXXXX1-100`
  - `XXXXXXXXXXXX`
  - `XXXXXXXXXXXX`
4. `"XXXX"`

--	--	--	--	--	--	--	--

- 00000 +44
- 00 SMSC uk\_gateway
- 0000 50
- 000 100
- 000 "00000"

□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□ 1□

- +44



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

2

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

- SMSC `carrier_smpp_bind`
- SMSC `local_msc`
- 
- 50
- 100
- "X - "

Diameter/HSS HSS

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

3. 設定“送信先”

4. 送信先設定

- 送信先設定画面から送信先を選択
- 送信先設定画面から送信先を選択
- 送信先設定画面から送信先を選択

送信先設定

- 送信先設定画面から送信先を選択
- 送信先設定画面から送信先を選択
- 送信先設定画面から送信先を選択
- 送信先設定画面から送信先を選択

送信先設定

**Web UI**

1. 送信先 /sms\_routing
2. 送信先設定
3. 設定“送信先”
4. 送信先
5. 設定“送信先”

送信先設定

- 送信先設定画面から“送信先”を選択
- 送信先設定画面から“送信先”を選択
- 送信先設定画面から“送信先”を選択
- 送信先設定画面から“送信先”を選択

送信先設定

**Web UI**

1. 送信先 /sms\_routing
2. 送信先設定
3. 設定“送信先”

#### 4. 消息

消息队列消息队列消息队列消息队列

消息/消息

消息队列消息

1. 消息 `/sms_routing`
2. 消息“消息”
3. 消息 JSON 消息

消息

1. 消息 `/sms_routing`
2. 消息“消息”
3. 消息 JSON 消息
4. 消息队列
  - 消息队列消息
  - 消息队列消息

消息

- 消息队列消息
- 消息队列消息
- 消息
- 消息

消息

消息队列

**Web UI**消息 `/frontend_status`

消息

- 消息队列消息“消息”

- 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. 0000000000 60 000000

### 000000

1. 00000000 POST /api/frontends/register
2. 00 API 0000000000
3. 00 JSON 000000
4. 00 curl 000000

### 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"
}'
```

## API 操作

### Web UI

1. 访问 `/frontend_status`
2. 检查返回结果
3. 查看“配置”
4. 检查返回结果

### API

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

### 返回

- 返回结果
- 返回结果
- 返回结果

## 配置

配置文件 `config/runtime.exs` 用于配置

### 配置

### 配置

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

## 配置

### 配置

配置 `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
}
```

□□□□□□□□□□



## 資料庫

### 資料庫

資料庫❓❓❓

資料庫資料庫 CDR 資料庫

- **MySQL/MariaDB** 查詢 `information_schema.tables` 資料庫
- **PostgreSQL** 查詢 `pg_database_size()` 查詢 `psql` 查詢 `\l+` 查詢

資料庫 **CDR** 查詢

資料庫資料庫 CDR 資料庫資料庫

- 資料庫資料庫資料庫 30-90 查詢
- 資料庫資料庫
- 資料庫資料庫

查詢

資料庫資料庫

- **MySQL/MariaDB** 資料庫 `OPTIMIZE TABLE` 查詢
- **PostgreSQL** 資料庫 `VACUUM ANALYZE` 資料庫 `autovacuum`

資料庫資料庫

## Mnesia 資料庫

查詢 **Mnesia** 查詢

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

查詢 **Mnesia** 查詢



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

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

## 安装 Mnesia

```
# 安装 Web UI 数据库
# 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 omnimessage || true
    endscript
}
```

## 测试

### 测试 logrotate

```
# 重启消息服务
systemctl restart omnimessage
```

```
# 重启消息服务
# 重启消息服务
```

重启消息服务

```
systemctl restart omnimessage
```

重启

- 重启消息服务
- 重启 Prometheus 服务
- 重启消息服务
- 重启消息服务

重启

重启

### 1. 重启

- `config/runtime.exs`
- `config/config.exs`
- `config/prod.exs` 重启

### 2. 重启 Mnesia

- 重启 Web UI
- 重启 Mnesia 服务

### 3. SQL CDR 重启

- 重启
- 重启消息服务

#### 4. TLS 备份

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

备份脚本

脚本内容

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

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

# 设置权限
chmod 600 $BACKUP_DIR/cert/*

echo "备份完成: $BACKUP_DIR"
```

运行脚本

```
#!/bin/bash
# /opt/omnimessage/scripts/backup_database.sh

BACKUP_DIR="/var/backups/sms_c/database"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p $BACKUP_DIR

# 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/omnimessage/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/omnimessage/scripts/backup_config.sh
0 2 * * * /opt/omnimessage/scripts/backup_database.sh
0 2 * * * /opt/omnimessage/scripts/backup_routes.sh
```

0000

00000

```
# 000000
systemctl stop omnimessage

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

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

# 000000
systemctl start omnimessage
```

00 **SQL CDR** 0000

0000000000000000

- **MySQL/MariaDB**000000 mysql 00000000
- **PostgreSQL**0000 pg\_restore 000000000000

00000000000000 SMS-C 0000000000000000

0000000

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

## 5. 監視

### 監視項目

#### メッセージキュー

##### メッセージキュー

Prometheus 監視30 分間隔

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

##### メッセージキュー

```
-- 監視項目
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;
```

### 監視

#### CPU 監視

- CPU < 50% 時
- CPU > 70% 時
- CPU > 90%

##### メッセージキュー

- CPU < 70% 時
- CPU > 80%

- 時間 > 90%

#### 時間

- 時間 < 60% 時間
- 時間 > 75%
- 時間 > 85%

#### 時間

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

#### 時間

#### 時間

- CPU 時間 > 70%
- 時間 > 80%
- 時間

#### 時間

- 時間 CPU > 50%
- 時間 > 5000 msg/sec
- 時間
- 時間

#### 時間

- 時間
- 時間
- 時間
- 時間

## □□□□

## □□□□□

### □□□□□□□□

- □□□□□□
- □□□□□□
- □□□□□□
- API □□□□

### □□1 □□□□□□□□

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

### □□4 □□□□□□□□

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

### □□24 □□□□□□□□

- □□□□□□
- □□□□□□
- □□□□□□□□

## □□□□□□□

### 1. □□□□□□□

- □□ Prometheus □□
- □□□□□□□□



- 0000000000
- 000000

## 2. 000000

- 0000000000
- 00000000
- 00000000OCS0DNS00
- 0000000000

## 3. 000000

- 0000000000
- 000000000000000000
- 00000000000000
- 00000000

## 4. 000

- 000000
- 00000000000
- 00000000
- 000000

## 5. 000000

- 0000
- 00000000
- 000000
- 000000

## 6. 0000

- 000000
- 0000/00
- 000000
- 000000

## 面试题

### 基础问题

1. 进程和线程
2. 进程间通信和线程间通信
3. 进程池和线程池
4. 了解 Prometheus 吗
5. 进程调度和线程调度

### 进阶问题

1. 内存管理
2. 垃圾回收
3. 内存泄漏
4. 进程崩溃和线程崩溃
5. 了解  吗

### 应用问题

1. 性能优化
2. 并发编程
3. 数据库
4. 了解 API 吗
5. 网络编程

### 系统设计

1. 负载均衡
2. 分布式存储
3. 分布式事务CPU/内存
4. 了解 ENUM 吗
5. 数据库设计

面试问题很多，这里只列举了一些 。

□□□□

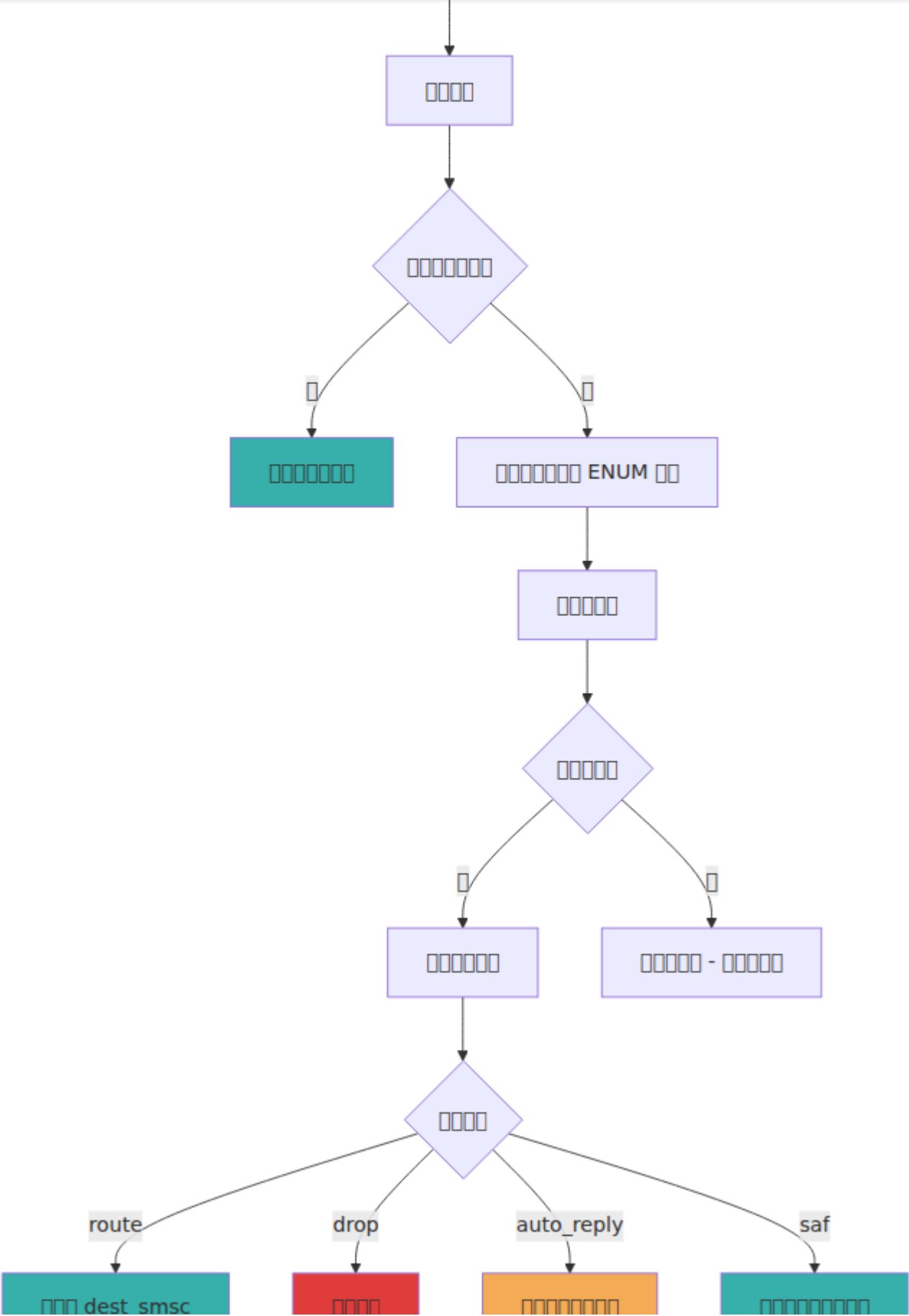
← □□□□□

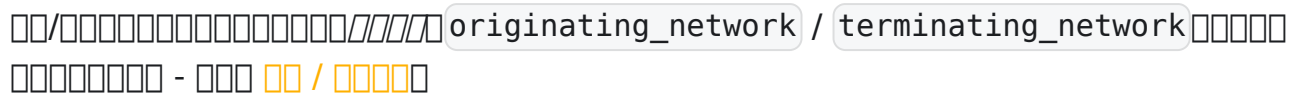
□□

SMS-C □□□□□□□□□□□□□□□□ Mnesia □□□□□□□□□□ API □□□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□ ENUM□DNS NAPTR□□□□□□□□□□ HSS □□□/□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□





1. **အခြေခံအားဖြည့်ဆည်းပေးခြင်း** - အခြေခံအားဖြည့်ဆည်းပေးခြင်း
2. **အခြေခံအားဖြည့်ဆည်းပေးခြင်း** - အခြေခံအားဖြည့်ဆည်းပေးခြင်း/အခြေခံအားဖြည့်ဆည်းပေးခြင်း HSS အ/အခြေခံအားဖြည့်ဆည်းပေးခြင်း
3. **အခြေခံအားဖြည့်ဆည်းပေးခြင်း** - အခြေခံအားဖြည့်ဆည်းပေးခြင်း

## enum\_domains

項目	型	説明
route_id	int	ルートID
calling_regex	Regex	発信者番号に一致する正規表現。nil = 一致しない
called_regex	Regex	被呼番号に一致する正規表現。nil = 一致しない
source_smsc	string	発信者SMS番号。nil = 不明
dest_smsc	string	宛SMS番号。routeに指定されたsafに一致するdrop/auto_replyに一致する
source_type	enum	発信者タイプ。ims, circuit_switched, smpp, nil
enum_domain	enum	ENUMドメイン。nil = 不明。DNSでenum_domainsに一致するENUMドメイン
enum_result_domain	enum	ENUM NAPTRの結果。ENUM NAPTRの結果に一致するnil = 不明

項目	データ型	説明
action	enum	route   drop   auto_reply   saf
auto_reply_message	string	auto_reply のメッセージ
charged	boolean	yes   no   default
weight	integer	1-100
priority	integer	1-255
description	string	
enabled	boolean	
originating_network	string	MSISDN HSS :any   :on_net   :off_net :any
terminating_network	string	MSISDN HSS :any   :on_net   :off_net :any

## 配置

action 配置项: route | drop | auto\_reply | saf

### route

配置项: dest\_smsc

```
{
  "action": "route",
  "called_regex": "^61",
  "dest_smsc": "australia-gw",
  "priority": 100,
  "description": "配置项 AU 配置"
}
```

### drop

配置项: dest\_smsc

```
{
  "action": "drop",
  "called_regex": "^900",
  "priority": 5,
  "description": "配置项"
}
```

### auto\_reply

配置项: auto\_reply\_message



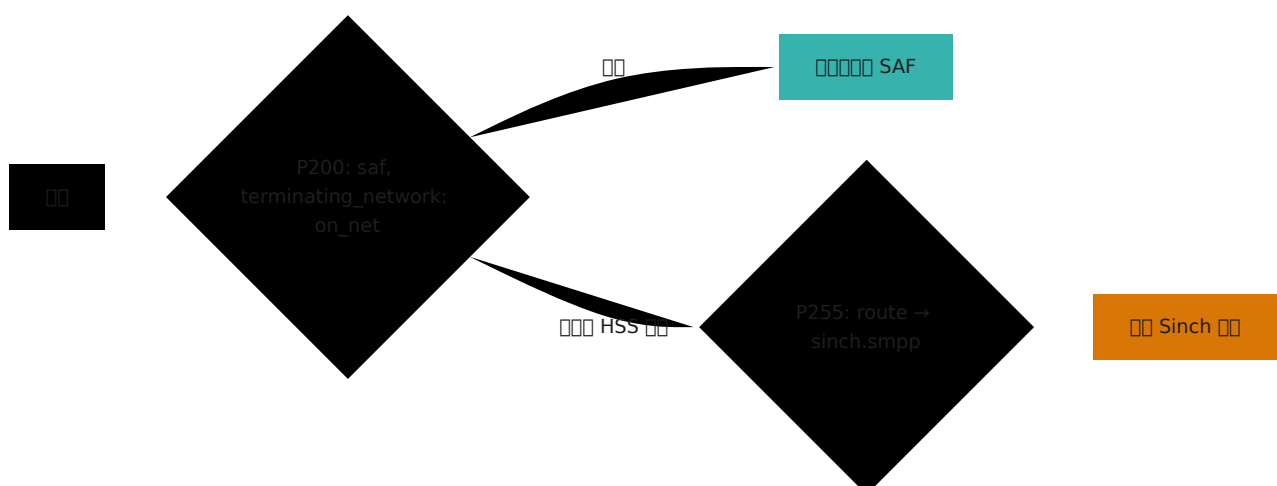
```
{
  "action": "auto_reply",
  "called_regex": "^911$",
  "auto_reply_message": "☎☎☎☎☎☎☎☎ 911☎☎☎☎ 911☎",
  "priority": 1,
  "description": "911 ☎☎☎☎"
}
```

**saf**

dest\_smsc

```
saf 00000000 - 0000 HSS 0000000000000000 saf 0 terminating_network:
:on_net 000000000000000000000000
```

```
{
  "action": "saf",
  "terminating_network": "on_net",
  "priority": 200,
  "description": "XXXXXXXXXX"
}
```

$$\square\square\square\square\square\square\square\square + \square\square\square$$


P200 SAF HSS -  
P255

# ENUM 簡介

ENUM 將電話號碼轉換為 DNS E.164 NAPTR 記錄，以便通過 DNS 查詢 HSS 獲取 SIP 地址。

## 配置

1. 啟用 ENUM 功能，設置 `enum_enabled: true` 和 `enum_domains` 列表。
2. 配置 ENUM 查詢，例如 `+15551234` 轉換為 `4.3.2.1.5.5.5.1.e164.arpa`。
3. 配置 NAPTR 記錄，設置 `result_domain` 和 `replacement` 字段。
4. 配置 ENUM 結果，設置 `enum_result_domain` 和 `result_domain`。
5. 配置 ETS 列表，設置 `(number, domain)` 列表。

## ENUM 查詢/配置

ENUM 查詢/配置 - 通過 `enum_result_domain` 設置 IMS/ENUM 查詢“域”。

## ENUM 配置

- 配置 NAPTR 記錄，設置 `enum_result` 列表。
- 配置 `enum_domain` 列表，設置 `enum_domains` 列表。

## 配置

ENUM DNS 配置 `enum_enabled` `enum_domains` `enum_dns_servers` `enum_timeout` 列表。

## IP / 電話番号

**Diameter Sh dip** の **HSS** には、電話番号とIPアドレスの対応関係が登録されている。  
電話番号は、電話番号/SMSC/電話番号の形式で登録されている。

## originating\_network / terminating\_network

ネットワーク

ネットワーク	説明
:any	任意の HSS に対して
:on_net	MSISDN が HSS に登録されている 2001
:off_net	MSISDN が HSS に登録されていない 5001

- originating\_network と terminating\_network は、ネットワーク
- terminating\_network: :on\_net は、SMPP を通じて  
originating\_network: :on\_net に対して  
terminating\_network: :off\_net

## ネットワーク

:any は、任意の HSS に対して

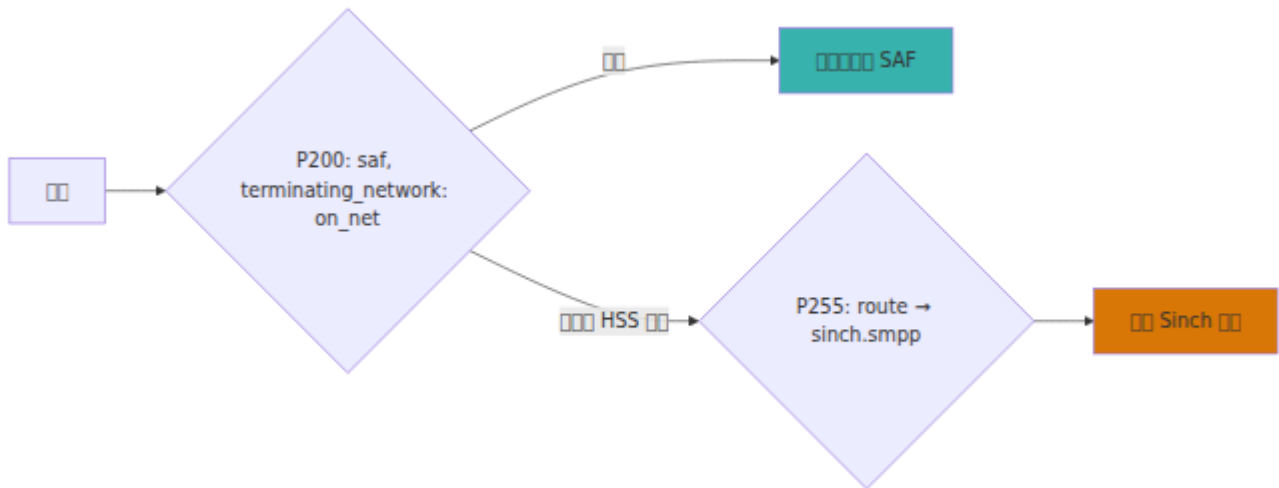
- Diameter IP → 電話番号
- HSS IP / IP / IP → 電話番号
- HSS IP 5001 に対して

:off\_net は、5001 に対して HSS の  
電話番号

## HSS 番号

HSS 番号は、Diameter 番号 MSISDN 番号

- 消息 **TTL** 2 秒 消息头 `hss_cache_ttl_seconds` 消息
- 消息头 `originating_network` 和 `terminating_network` 消息头 `:on_net` 和 `:off_net` 消息头 `HSS` 消息头 `:any` 消息头
- 消息头 `MSISDN` 消息头 `HSS` 消息
- 消息头 `MSISDN` 消息头 `HSS` 消息



## 消息头

消息头 `route_evaluated` 消息头

- 消息 ID 消息头
- 消息头 `enum_result` 和 `orig_network/term_network` HSS 消息
- 消息头 `SMSC` 消息头 `HSS` 消息

消息头

```

消息 #4 [saf] 消息 P200 "消息" |
  calling_regex: ✓ (消息), called_regex: ✓ (消息),
  source_smsc: ✓ (消息), source_type: ✓ (消息),
  enum_result: ✓ (消息),
  term_network: ✓ (消息+15551234 消息)
  
```

```

消息 #3 [route] 消息 P255 "Sinch 消息" -> sinch.smpp |
  calling_regex: ✓ (消息), called_regex: ✓ (消息),
  source_smsc: ✓ (消息), source_type: x (ims vs circuit_switched),
  enum_result: ✓ (消息), term_network: ✓ (消息)
  
```

□□□□□□□□□□□□□□□□ API □□□





XXXXXXXXXX

XXXXXXXX

XXXXXX

XXXX

XXXX

HSS 00/XXXXXXXXXXXXXXXXXXXXXXXXXXXX - XXXXXXX HSS 000

XXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXX



項目	サイズ
called_regex 文字列	文字 × 100
calling_regex 文字列	文字 × 50
source_smsc 文字	+25
enum_result_domain 文字	+15
source_type 文字	+10
enum_domain 文字	+5

文字列の長さは、このように決まっています。

文字

文字列の長さ

config/runtime.exs の中で、このように API を呼び出します。

[illegible]

enum

enum	enum	enum	enum	enum
action	enum	enum	:route	enum enum :route enum :drop enum :auto_reply
calling_regex	enum	enum	nil	enum nil enum
called_regex	enum	enum	nil	enum nil enum
source_smsc	enum	enum	nil	enum SMSC enum nil enum
dest_smsc	enum	enum	nil	enum route enum saf enum enum drop/auto_reply enum
source_type	enum	enum	nil	enum enum :ims enum :circuit_switched enum :
enum_domain	enum	enum	nil	enum ENUM enum ??? enum

項目	型	単位	初期値	説明
enum_result_domain	enum	なし	nil	ENUM NAPTR の結果を返す。r 参照
auto_reply_message	string	なし	nil	自動返信メッセージ。:auto_reply 参照
charged	boolean	なし	:default	課金有無。:yes :no :default
weight	integer	なし	100	重み。1-100
priority	integer	なし	100	優先度。1-255
description	string	なし	""	説明
enabled	boolean	なし	true	有効/無効
originating_network	string	なし	:any	発信ネットワーク。MSISDN HSS 参照 :any :on_net :off_net
terminating_network	string	なし	:any	着信ネットワーク。MSISDN HSS 参照 :any :on_net :off_net

# HSS 配置

配置项	数据类型	默认值	说明
<code>hss_cache_ttl_seconds</code>	整数	7200	配置 HSS 缓存时间，单位秒 <code>runtime.exs</code> 中配置

# API

配置

POST /api/routes

```
{
  "action": "saf",
  "terminating_network": "on_net",
  "priority": 200,
  "description": "配置"
}
```

查询

GET /api/routes

更新

PUT /api/routes/:id



□□□□

- □□□□ `terminating_network: :on_net` □□□□□□□□
- □□□□□□□□/□□□□□□□□□□□□□□
- HSS □□□□□□ `hss_dip_error` / Sh □□□ - □□□□□□□□□□□□□□□□□□
- □□□□□□ Diameter

□□□□

1. □□□□□□□□□□□□□□ `terminating_network: :on_net` □□
2. □□□□□□□□ Diameter □□□□□□HSS □□□□□□□□
3. □□ HSS □□□□□□□□□□□□□□□□

## HSS □□□□

□□□□□□□□□□□□□□□□□□□□/□□□□□□□□

□□□□

1. □□ API □□□□□□ - □□□□□□□□ HSS □□
2. □□□□ TTL □□□□□□ 2 □□□
3. □□□□□□□□□□□□ `hss_cache_ttl_seconds`

# SMS-C 目录

← 目录

目录 SMS-C 目录

目录

- 目录
- 目录
- 目录
- 目录
- 目录
- 目录
- 目录/目录
- ENUM 目录
- 目录
- API 目录
- Web UI 目录
- 目录



## □□□□

### □□□□□□

```
# 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 omnimessage  
  
# 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
```

### □□□□□□□

```
# 監視ログ
tail -f /var/log/sms_c/application.log | grep -E "
(error|warning|critical)"
```

監視

監視項目

```
# 受信メッセージ数
rate(sms_c_message_received_count[5m])

# 配信成功率
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])
```

監視項目

```
# 待ち行列サイズ
sms_c_queue_size_pending

# 待ち行列最古メッセージの年齢
sms_c_queue_oldest_message_age_seconds
```

監視項目

```
# 処理時間 (p95)
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)

# 配信時間 (p95)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
```

# API 消息

## API 消息

### 消息

- 消息 "queued"/"backoff" 消息
- 消息消息消息
- 消息消息

### 消息消息

#### 1. 消息消息消息

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

### 消息消息消息消息

### 消息消息消息消息消息消息消息消息

#### 2. 消息消息消息

消息 Web UI: /message\_queue

- 消息消息消息 "queued" 消息 "backoff"
- 消息 dest\_smsc 消息
- 消息 deliver\_after 消息消息

#### 3. 消息消息

消息 Web UI: /simulator

- 消息消息消息消息消息消息消息消息
- 消息消息消息消息消息消息消息消息

#### 4. 消息消息消息

### 消息消息消息消息消息

- 消息消息消息消息 /api/messages 消息

- 配置 `smc` 参数

配置

配置

```
# 检查服务状态
systemctl status frontend_service

# 通过 API 检查
curl -k https://api.example.com:8443/api/status

# 注册前端
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"
}'
```

配置 **SMSC**

- 配置
- 配置
- 配置
- 配置 `dest_smc` 参数

配置

- 配置 `deliver_after` 参数
- 配置

```
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. □□□□□□□

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

□□□□□

□□□□□□□

- □□□□□□□□
- □□□□□□

□□□□□

# 設定メッセージ

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

# 送信メッセージ

```
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"}'
```

メッセージキュー

- メッセージ
- メッセージキュー
- メッセージキューの管理

メッセージ

メッセージ

- メッセージ `deadletter: true`
- メッセージキュー
- メッセージ `"queued"/"backoff"` メッセージキュー `expires` メッセージキュー

メッセージ

1. メッセージ

Web UI: `/message_queue`

- メッセージ
- メッセージ

2. メッセージ

- メッセージ
- メッセージ
- メッセージ

□□□□□

□□□□□□□

```
# □□□□□□□ 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. □□□□□□□□

- 0000000000
- 0 SMSC 0000000000
- 0000000000

00000

000000

0000000

```
0000: (0)
0000: (0)
0 SMSC: (0)
00 SMSC: default_gateway
000: 255
00: 100
00: ✓
00: 000000
```

0000000

0000000000

```
0000: +
00 SMSC: international_gateway
000: 200
00: 100
00: ✓
00: 000000
```

0000000000

- 00 Web UI 00000000
- 00000000000000

00000000

000



- 000000000000
- 0000000000
- 000000000000

000000

1. 0000000000

00 Web UI: /simulator

- 00000000000000
- 00 "0000" 00
- 000000000000

2. 0000000000

- 0000 = 000000
- 000000000000
- 00000000000000

3. 0000000000

00000000

- 00000000000000 +100 0
- 00000000000000 +50 0
- 0000 SMSC0+25 0
- 00000000+10 0
- 00 ENUM 00+15 0

000000

00000000

00000000000000

□□□□:

□□□□: +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

- `action: :auto_reply`
- `auto_reply_message` 確認
- 確認
- 確認

2. 確認

- 確認
- 確認 `"auto_reply"` 確認

3. 確認

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

確認

確認

- 確認
- 確認
- 確認

確認

?? 確認

```
確認: ✓  
確認: "確認"
```

確認

00000000000000000000

```
000000:
 000: 10

0000:
 000: 50
```

0000

00000000

000

- sms\_c\_message\_processing\_stop\_duration p95 > 1000ms
- API 0000
- 0000

00000

1. 00000000

```
# 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_succeeded_duration_bucket)
```

2. 00000000

```
# 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/確認 DNS 確認
- 確認 ENUM

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
# 配置数据库连接
config :sms_c,
  batch_insert_batch_size: 200,
  batch_insert_flush_interval_ms: 200

# 配置缓存
config :sms_c, SmsC.Repo,
  pool_size: 50
```

## 性能测试

### 测试

- 消息速率 < 100 msg/sec
- 消息 API 调用速率
- 消息 API 调用延迟

### 测试方法

#### 1. 消息速率测试

```
# 消息速率测试 (iex)
SmsC.Messaging.BatchInsertWorker.stats()
```

### 测试

- `current_queue_size` 消息队列大小
- `flush_errors` > 0
- `last_flush_duration_ms` 消息队列清空时间

#### 2. 消息延迟测试

```
# 監視時間
ecto_pools_query_time

# 監視時間
ecto_pools_queue_time
```

監視時間

監視時間

監視時間

```
config :sms_c, SmsC.Repo,
  pool_size: 50 # 20 分
```

監視時間

監視時間

```
config :sms_c,
  batch_insert_batch_size: 200, # 監視時間
  batch_insert_flush_interval_ms: 200 # 監視時間
```

監視時間

```
# 監視時間 /create_async
curl -X POST
https://api.example.com:8443/api/messages/create_async

# NOT: /api/messages (監視時間)
```

監視時間

監視時間

- sms\_c\_queue\_size\_pending 監視時間
- 監視時間

- 0000000000000000

000000

### 1. 000000000000

```
# 0000
rate(sms_c_message_received_count[5m])

# 0000
rate(sms_c_delivery_succeeded_count[5m])
```

### 2. 00000000

- 0000000000000000
- 0000000000000000
- 00000000

### 3. 0000000000

```
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_attempted_count[5m])
```

000000

00000000

- 00000000
- 0000000000 5-10 00
- 00000000

000000

- 0000000000
- 00000000
- 00000000

000000



- 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 "unable to connect to database"
- API 500 错误
- 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 数据库
ping db.example.com

# 数据库 (MySQL/MariaDB: 3306, PostgreSQL: 5432)
telnet db.example.com 3306
# 
telnet db.example.com 5432
```

### 3. 数据库

```
# 数据库
echo $DB_USERNAME
echo $DB_HOSTNAME
echo $DB_PORT

# 数据库 (MySQL/MariaDB)
mysql -u $DB_USERNAME -p$DB_PASSWORD -h $DB_HOSTNAME

# PostgreSQL:
# psql -U $DB_USERNAME -h $DB_HOSTNAME -d sms_c_prod
```

数据库

数据库

```
# 数据库 (MySQL/MariaDB)
systemctl start mysql

# 数据库 (PostgreSQL)
systemctl start postgresql
```

数据库

数据库

```
export DB_USERNAME=correct_user
export DB_PASSWORD=correct_password
```

```
# 重启服务
systemctl restart omnimessage
```

配置

- 数据库配置
- 消息队列配置
- 配置 VPN/代理

配置

配置

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

配置

配置

- 数据库配置
- API 配置
- 消息队列配置

配置

1. 数据库配置

```
-- MySQL/MariaDB: 开启慢查询
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: 在 postgresql.conf 中配置
-- log_min_duration_statement = 1000 # 1秒
-- 重启 PostgreSQL 生效
```

## 2. 查看消息队列

```
-- 查看消息队列
SHOW INDEX FROM message_queues;

-- 查看消息队列表结构
-- - source_smsg
-- - dest_smsg
-- - send_time
-- - inserted_at
```

## 3. 查看表大小

```
-- 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
-- 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;
```

□□□□□□□

□□□

- □□□ "Disk full"
- □□□□□□□□

- 空间满了

空间满了

## 1. 检查磁盘空间

```
df -h
```

```
# 检查 MySQL 数据库 (MySQL/MariaDB)
du -sh /var/lib/mysql
```

```
# 检查 PostgreSQL 数据库 (PostgreSQL)
du -sh /var/lib/postgresql
```

## 2. 清理空间

```
# 清理 MySQL 数据库 (MySQL/MariaDB)
find /var/lib/mysql -type f -exec du -h {} + | sort -rh | head -20
```

```
# 清理 PostgreSQL 数据库 (PostgreSQL)
find /var/lib/postgresql -type f -exec du -h {} + | sort -rh |
head -20
```

```
# 清理日志文件
du -sh /var/log/sms_c/*
```

空间满了

空间满了

```
-- 清理消息队列
```

```
DELETE FROM message_queues
WHERE inserted_at < DATE_SUB(NOW(), INTERVAL 90 DAY)
LIMIT 100000;
```

空间满了

```
# 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 命令
- 配置 cron 任务

## 部署

## 测试

## 验证

- 检查 "expired"
- 检查日志文件
- 检查 cron 任务

## 总结

### 1. 部署

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

### 2. 配置

- 配置 `/api/frontends/register`
- 配置 API 接口
- 配置 60 秒间隔

### 3. 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 '{
  "frontend_name": "uk_gateway",
  "frontend_type": "smpp",
  "ip_address": "10.0.1.50"
}'
```

□□□□□□□□□□□□□□□□/□□□□

□□□□□

□□□ 90 □□□□□□□□□ 60 □□□□□□

```
# □□□□ 60 □□□□□
while True:
    register_with_smsc()
    time.sleep(60)
```

□□□□□

- □□□□□ API □□□□□□□
- □□ DNS □□
- □□□□□□□□ curl

□□□□□□□□□□/□□

□□□



- 網路連線/網路卡
- 網路設定
- 網路測試

網路測試

1. 網路測試

- 網路測試工具
- 網路測試方法
- 網路測試CPU/網路卡

2. 網路測試

```
# 網路測試
ping -c 100 api.example.com

# 網路測試
netstat -s | grep -i reset
```

3. 網路測試

- 網路測試工具
- 網路測試方法 > 90 分

網路測試

網路測試

- 網路測試工具
- 網路測試方法
- 網路測試CPU/網路卡

網路測試

- 網路測試工具
- 網路測試方法
- 網路測試CPU/網路卡

確認する

確認する

```
REGISTRATION_INTERVAL = 60 # 分
```

確認 **Diameter Ro** 設定が OCS 設定と Diameter 設定が一致していることを確認する。 :balance\_rejected

確認

- sms\_c\_charging\_failed\_count 確認
- 確認 "charging\_failed" と "balance\_rejected"
- 確認 :balance\_rejected 設定と Diameter 設定

確認

1. 確認 **Diameter Ro** 設定が CER/CEA 設定と DRA/OCS 設定

```
# 確認  
:diameter.service_info(:omnimessage, :connections)
```

2. 確認 :ro 設定

```
grep -E "diameter_enabled|application_name: :ro"  
config/runtime.exs
```

確認 Charging requested but the Diameter Ro stack is not enabled/configured 設定 - 確認

3. 確認 OCS 設定

```
grep rating_profile config/runtime.exs
```

確認

**OCS / DRA** 設定

- IP 地址 DRA Diameter Origin-Host/Realm
- CCR Service-Context-Id rating\_profile OCS SMS

配置 :sms\_c, :charging

```
config :sms_c, :charging,
  out_of_balance_action: :reject,    # :auto_reply
  out_of_balance_reply_text: "...",
  rating_profile: "sms@sms-c"
```

配置

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

配置

配置

- OCS
- 
- 

配置

配置

- sms\_c\_charging\_succeeded\_duration p95 > 500ms
- 
- 

配置

1. 配置

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

## 2. **OCS / DRA** **Diameter Ro** **配置**

`sms_c_charging_succeeded_duration` **OCS** **配置**/**DRA** **配置**

## 3. **DRA** **配置**

```
# Ping DRA / OCS 配置
ping -c 10 dra01.example.com
```

**配置**

## **OCS** **配置**

- **OCS** **配置**
- **OCS** **配置**
- **配置**

**配置**

- **OCS** **配置** SMS-C
- **配置**
- **配置** VPN/**配置**

**配置**

**配置**

```
config :sms_c,
  ocs_timeout: 5000 # 5 配置
```

# ENUM **配置**

## ENUM **配置**

**配置**

- `sms_c_enum_lookup_stop_duration` **配置**

- 枚举枚举 ENUM 枚举
- 枚举 `enum_result_domain` 枚举

枚举枚举

### 1. 枚举 **ENUM** 枚举

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

### 2. 枚举 **DNS** 枚举

```
# 枚举 DNS 枚举
dig @8.8.8.8 e164.arpa

# 枚举 ENUM 枚举
# 枚举 +15551234567
dig @8.8.8.8 NAPTR 7.6.5.4.3.2.1.5.5.5.1.e164.arpa
```

### 3. 枚举 **DNS** 枚举枚举

```
# 枚举 DNS 枚举枚举
ping 10.0.1.53

# 枚举枚举
nc -zv 10.0.1.53 53
```

枚举枚举

## **DNS** 枚举枚举枚举

枚举枚举 DNS

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

## ENUM 0000

0000

```
config :sms_c,  
  enum_domains: ["e164.arpa"] # 00000
```

00000000

00000

```
config :sms_c,  
  enum_timeout: 10000 # 10 0
```

00 **ENUM**0000000000

```
config :sms_c,  
  enum_enabled: false
```

## ENUM 0000

000

- 00000000 < 70%0
- 000000000
- 00000

00000

1. 00000000

```
# 计算命中率
rate(sms_c_enum_cache_hit_count[5m]) /
(rate(sms_c_enum_cache_hit_count[5m]) +
rate(sms_c_enum_cache_miss_count[5m]))

# 设置缓存大小
sms_c_enum_cache_size_size
```

## 2. 配置策略

- 配置策略
- 设置 TTL 策略

策略

策略配置

- 策略配置
- 策略 < 70% 策略

策略

策略 NAPTR 策略配置

策略

- 策略
- 策略
- 策略 TTL

策略

策略配置

策略

- 策略
- 策略配置

- Erlang 環境

環境

### 1. 環境構築

```
# Erlang 環境
Node.self()
# 127.0.0.1:3000@node1.example.com

Node.list()
# 127.0.0.1:3000@node1.example.com
```

### 2. Erlang Cookie

```
# Erlang cookie
cat ~/.erlang.cookie

# 127.0.0.1:3000@node1.example.com
```

### 3. 接続

```
# 127.0.0.1:3000@node1.example.com
ping node2.example.com

# 127.0.0.1:3000@node1.example.com
nc -zv node2.example.com 4369
nc -zv node2.example.com 9100-9200
```

環境

### Cookie 環境

127.0.0.1:3000@node1.example.com cookie



```
export ERLANG_COOKIE=same_secret_value_here
```

```
# 创建 ~/.erlang.cookie  
echo "same_secret_value_here" > ~/.erlang.cookie  
chmod 400 ~/.erlang.cookie
```

配置完成

配置完成

```
# EPMD  
iptables -A INPUT -p tcp --dport 4369 -j ACCEPT  
  
# Erlang 配置  
iptables -A INPUT -p tcp --dport 9100:9200 -j ACCEPT
```

## DNS 配置

配置 IP 地址

```
config :sms_c,  
  cluster_nodes: [  
    "sms@10.0.1.10",  
    "sms@10.0.1.11"  
  ]
```

配置完成

配置

- 配置完成
- 配置完成
- Mnesia 配置

配置完成

1. 配置完成

```
# 00000000IEx
Node.list()
```

## 2. 00 Mnesia

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

00000

00000000

```
# 000000
systemctl stop omnimessage

# 00000000
systemctl start omnimessage # 0 node1 0

# 000000000000000000
systemctl start omnimessage # 0 node2 0
systemctl start omnimessage # 0 node3 0
```

## Mnesia 0000

- 000000000000
- 0000000
- 00 Mnesia 00
- 0000
- 0000

# API 00

## API 00000

000

- 0000

- 証明書
- 設定

確認

### 1. API 確認

```
# 確認
systemctl status omnimessage

# 確認
netstat -tlnp | grep 8443
```

### 2. 確認

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

# 確認
curl -k https://localhost:8443/api/status
```

### 3. TLS 確認

```
# 確認
ls -l priv/cert/server.crt priv/cert/server.key

# 確認
openssl x509 -in priv/cert/server.crt -noout -dates
```

確認

確認

```
systemctl start omnimessage
```

確認

```
# API 500
iptables -A INPUT -p tcp --dport 8443 -j ACCEPT
```

500

500

500

500

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

## API 500

500

- 500
- 500
- 500

500

1. 500

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

2. 500

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

3. 500

```
# 内存
free -h

# CPU
top -b -n 1

# 磁盘
df -h
```

常用命令

常用命令

- 常用命令
- 常用命令

常用命令

- 常用命令
- 常用命令
- 常用命令

常用命令

- 常用命令
- 常用命令
- 常用命令

# Web UI 常用

## 常用 Web UI

常用

- 常用
- 404 常用
- 常用

確認

1. 確認メッセージ

```
systemctl status omnimessage
```

2. 確認

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

3. 確認 URL

- 確認
- 確認
- HTTP と HTTPS

確認

確認

確認

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

確認メッセージ80 と 4000

確認

```
systemctl start omnimessage
```

確認

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

# LiveView 簡介

簡介

- 簡易易用
- 快速
- 基於 Phoenix 的 WebSocket 協議

安裝

## 1. 安裝 Phoenix

- 安裝 Phoenix F12
- 安裝 WebSocket 協議
- 安裝 Phoenix 依賴

## 2. 安裝 LiveView

安裝 Phoenix 依賴 WebSocket

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

安裝

## WebSocket 簡介

- 基於 Phoenix 的 WebSocket
- 快速
- 簡易易用

安裝

- 安裝 Ctrl+F5
- 安裝 Phoenix

# 📁📁📁📁📁

## CPU 📁📁📁

📁📁

- CPU 📁📁 > 80%
- 📁📁📁
- 📁📁📁📁📁

📁📁📁📁

1. 📁📁📁📁

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

2. 📁📁📁📁

```
# 📁📁📁📁📁  
rate(sms_c_message_received_count[5m])  
  
# 📁📁📁  
rate(sms_c_routing_route_matched_count[5m])
```

📁📁📁📁

📁📁📁

- 📁📁📁📁📁📁📁
- 📁📁📁📁📁 CPU📁

📁📁📁📁📁

- 📁📁📁📁📁
- 📁📁📁📁📁

## ENUM 📁📁📁📁



-  00000000
- 000000000000

00000000

000

- 0000 > 90%
- 00000000
- 00000000

000000

1. 000000

```
free -h
```

```
ps aux | grep beam.smp
```

2. 00000000

```
sms_c_enum_cache_size_size
```

000000

**ENUM** 000000

- 0000
- 00 TTL
- 0000000000 ENUM

0000000000

```
# 0000000000IEx
SmsC.Messaging.BatchInsertWorker.stats()
```

000000000000000000

□□□□

- □□□□
- □□□□□□□□

□□□□

- □□□□□□
- □□□□□□□□

---

□□□□□□□□□□

- □□□□ - □□□□
- □□□□ - □□□□
- □□□□ - □□□□
- □□□□□□ - `/var/log/sms_c/application.log`

