

README

Benchee の SMS-C 実装のベンチマーク

インストール

1. SMS ベンチマーク (raw_sms_bench.exs)

submit_message_raw API の SMS PDU

を実行

- SMS PDU の PDU を @sample_pdus
-
- HTML

を実行

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

benchmarks/output/raw_sms_benchmark.html

2. API ベンチマーク (message_api_bench.exs)

API の

を実行

- insert_message
- get_messages_for_smsc
- list_message_queues
-

を実行

```
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 OpenAPI API

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- **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	受信番号
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SMSC
dest_smsc	string / null	受信SMSC
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	

項目	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>` - SMSC ノード名
- `include-unrouted: true|1` - 未ルーティングのメッセージを含めるかどうか `false|0`
- `x-decode: true|1` - 文字コードをデコードするかどうか

クエリパラメータ `smsc` の例:

- `status` - ステータス
- `limit` - 取得件数 100 から 1000
- `offset` - オフセット 0 から

レスポンスの例 `limit` が 100 の場合

200 OK - JSON

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
```

22

```
{
  "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

□□□□□ SMS PDU□□□□□□□□□□□□□□□□□□□□ SIP □□□

11

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

11

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

□□:

- `message_body` - 16비트 SMS PDU의 TPDU 데이터
- `source_msisdn` - 발신자 번호
- `source_smsc` / `smsc_node_name` - 발신자 TPDU의 SMS 센터 이름
- `rp_originator_address`, `rp_destination_address` - 발신자 TPDU의 발신자/수신자 주소

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)

□ □ □ □ □ □ □ □ □ □ □ □

예:

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

요청 본문에 `dest_smsc`, `deliver_after`, `status` 필드를 포함합니다.

응답 (200 OK): `404` 또는 `422` 오류가 발생할 수 있습니다.

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

예외

요청 본문에 `backoff` 필드를 포함합니다.

요청 본문

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

`PUT /api/messages/:id` 또는 `PATCH /api/messages/:id` 요청에 `keep_route=true`와 `dest_smsc` 필드를 포함합니다.

응답 (200 OK): `200 OK`

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

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

예외

요청 본문

```
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 0000000

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

00000000

0000000000000000000000 `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 4040000000000000000000 4220

000000

0000000000000000000000 null0000000000000000000000

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 0000000000000000000000

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": "..."}
  ]
}
```

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、`var1..var4` の変数があり、JSON の場合は CSV のように変数を JSON の中に埋め込むことができます。MSISDN の場合は `var1..var4` の変数があります。

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

CSV の場合は `msisdn / destination_msisdn / number / phone` のように MSISDN の後に `var1..var4` の変数を埋め込むことができます。

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

`PATCH` で更新します。

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

API

```
GET    /api/campaigns      # 取得全体的キャンペーン
GET    /api/campaigns/:id # 取得指定IDのキャンペーン
POST   /api/campaigns      # 新規キャンペーンの作成
DELETE /api/campaigns/:id # 指定IDのキャンペーンの削除
```

リクエスト:

```
{
  "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
}
```

レスポンス:

- `name` `message` `source_msisdn` `source_smsc` - `message` 送信メッセージの本文
例: `{{ var1 }}`..`{{ var4 }}` `{{ msisdn }}`
- `list_id` - 送信対象のリストID
- `audience` - `active_only` (アクティブのみ) `all` (全登録者) `list_id` 指定
- `drip_tps` - 配信ペース (1分あたりのメッセージ数)
- `validity_hours` - 有効期間 (単位: 時間) `expires` 指定
- `deliver_after` - 配信開始時刻 (ISO 8601 形式) `expires` 指定

DELETE 指定IDのキャンペーンを削除 (200 OK) TTL あり

取得 (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

路由

```
GET    /api/routes          # 获取路由列表
GET    /api/routes/:id     # 获取路由详情
POST   /api/routes        # 创建路由
PATCH /api/routes/:id     # 更新路由
PUT    /api/routes/:id     # 更新路由
DELETE /api/routes/:id     # 删除路由
GET    /api/routes/export  # 导出路由
POST   /api/routes/import  # 导入路由
POST   /api/routes/simulate # 模拟路由
```

短信 API

翻译

```
GET    /api/translations    # 获取翻译列表
GET    /api/translations/:id # 获取翻译详情
POST   /api/translations    # 创建翻译
PATCH /api/translations/:id # 更新翻译
DELETE /api/translations/:id # 删除翻译
GET    /api/translations/export # 导出翻译
POST   /api/translations/import # 导入翻译
POST   /api/translations/simulate # 模拟翻译
```

短信 API

□□

```
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 状態

ステータスコード	ステータス名	説明
200	OK	正常に完了
201	Created	新規作成
202	Accepted	受け入れられた
204	No Content	空の内容
400	Bad Request	不正なリクエスト ID 指定
404	Not Found	見つからない
422	Unprocessable Entity	処理不可
500	Internal Server Error	内部サーバーエラー

エラー

エラーメッセージ `error` 形式

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

エラーメッセージの応答形式

```
{ "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](#)

ANSSI R226

OmniMessage SMS R226-3 R226-7 ANSSI R226

ANSI

R226 -

1.

1.1

OmniMessage SMS

REST API HTTPS SMPP IMS SS7/MAP

Elixir/Erlang Phoenix Mnesia MySQL/PostgreSQL

- SMS REST API
- SMPP IMS SS7/MAP
-
-
- ???
- CDR
- 1,750 / 1.5

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

安裝

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

安裝

- SMS 安裝
- 安裝/安裝
- SMSC 安裝
- 安裝
- 安裝
- ENUM E.164 安裝 DNS
- 安裝
- Diameter Ro 安裝

安裝 **README.md**

1.2 規格

1.2.1 規格

SMS 規格

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

規格

- CDR 規格
 - ID
 - MSISDN
 - MSISDN
 - PDU
 - TP-DCS
 - GSM7 UCS-2 8 Latin-1
 - UDH
 - SMSC
 - SMSC
 - Erlang
 - UDH

CDR MySQL/PostgreSQL

- CDR 資料をデータベースに保存
- CDR 資料を SQL データベース MySQL / PostgreSQL に保存
- CDR 資料をデータベースに保存 CDR
 - 保存
 - 検索
 - 更新
 - 削除
- CDR 資料をデータベースに保存
- CDR 資料をデータベースに保存
- CDR 資料をデータベースに保存
- CDR SQL データ REST API データ CSV/JSON データ

CDR 資料

1. CDR 資料をデータベースに SQL データベースに保存
2. CDR 資料 Mnesia データ 1,750 データベースに SQL データ
3. CDR 資料 disc_copies データベースに保存
4. CDR 資料 CDR データベースに保存
5. CDR 資料 データ  データベースに保存

CDR 資料

1. CDR 資料 → CDR Mnesia データ RAM + データベース
2. CDR 資料 → Mnesia データベース
3. CDR 資料/データ → CDR データ SQL データ
4. 24 データ → CDR Mnesia データベース
5. CDR データ SQL → データベースに保存

CDR 資料

- CDR 資料をデータベースに保存
- CDR 資料をデータベースに PDU データ Mnesia データ CDR データ
- CDR 資料をデータベースに CDR データベース
- CDR CDR データベースに保存

목차

- SMSC 및 SMPP/IMS/MAP 구성
- 데이터베이스
- 데이터베이스/파일
- 인증/권한
- IP 주소 할당
- 데이터베이스

1.2.3 구성

목차

- Web UI 구성
 - 데이터베이스
 - 데이터베이스
 - 데이터베이스
 - 데이터베이스
 - 데이터베이스
- Prometheus 데이터베이스
- 데이터베이스

이 문서는 **OPERATIONS_GUIDE.md**

이 문서는 **METRICS.md**

목차

- CDR 데이터베이스
 - 데이터
 - 데이터/데이터
 - 데이터
 - SMSC 데이터
 - 데이터
 - 데이터베이스
- 데이터베이스
 - 데이터/데이터

- 國際電訊/電報
- 國際電報
- 國際電報
- 國際電報

國際電報

- 國際電報MSISDN國際電報
- 國際 IMSI 國際電報 IMS/MAP 國際電報
- 國際電報
- 國際電報
- 國際電報國際電報國際電報

國際電報

- 國際電報
- 國際電報國際電報
- 國際電報
- 國際電報國際電報國際電報
- 國際電報
- 國際電報

國際電報

- E.164 國際電報
- 國際電報國際電報/國際電報
- 國際電報國際電報
- ENUM DNS 國際電報國際電報
- 國際電報國際電報

國際電報 [number_translation_guide.md](#)

國際電報 [sms_routing_guide.md](#)

1.3 國際電報

1.3.1 國際電報

API

- HTTPS/TLS 與 REST API
- 認證與授權
- 傳輸層安全 TLS
- 安全協議與加密

UI

- Web UI
- API
- 認證與授權
- 安全協議與加密

數據庫

- 數據庫設計
- 數據庫/表設計
- 數據庫安全
- 數據庫性能
- 數據庫備份與恢復

1.3.2 數據庫安全

數據庫安全

- 數據庫安全設計
- 數據庫 UI 安全設計
- 數據庫安全協議與加密
- CDR 數據庫安全設計 NULL 數據庫

數據庫安全

- MySQL 數據庫安全 ENCRYPTION='Y'
- PostgreSQL 數據庫安全
- 數據庫安全設計
- 數據庫安全設計

- 消息的持久化

消息

- 消息的持久化
- TLS 加密
- 消息的压缩
- 消息的加密
- 使用 Erlang 消息的持久化

1.4 消息的持久化 Mnesia + SQL 消息

消息

OmniMessage SMS 消息的持久化

消息的持久化 Mnesia 消息

消息 Mnesia

- 使用 Erlang/OTP 消息的持久化
- 消息的持久化
- ACID 消息
- 消息的持久化

消息的持久化 disc_copies

- 消息的持久化 RAM 消息
 - 消息的持久化/消息的持久化
 - 消息的持久化/消息的持久化 I/O
 - 消息 1,750 消息的持久化/消息的持久化
- 消息的持久化 Mnesia 消息 RAM 消息
 - 消息的持久化
 - 消息的持久化
 - 消息的持久化
 - 消息的持久化 Mnesia.*/ 消息

Mnesia 環境構築

1. 既存 REST API へ → Mnesia RAM + 追加
2. 既存 Mnesia → 追加
3. 追加 → Mnesia 追加
4. 追加 → Mnesia 追加 + 追加
5. 追加/追加 → 追加
6. 追加 24 時間 → Mnesia 追加

環境構築

- 追加/追加 **SQL** 追加
- SQL 追加
- 追加 SMS-C 追加 I/O

追加 CDR 追加/追加 SQL 追加

追加 CDR 追加

- 追加
- 追加 MySQL 追加 PostgreSQL 追加
- 追加

追加 CDR 追加 追加 CDR 追加

- 追加
- 追加
- 追加
- 追加

追加 CDR

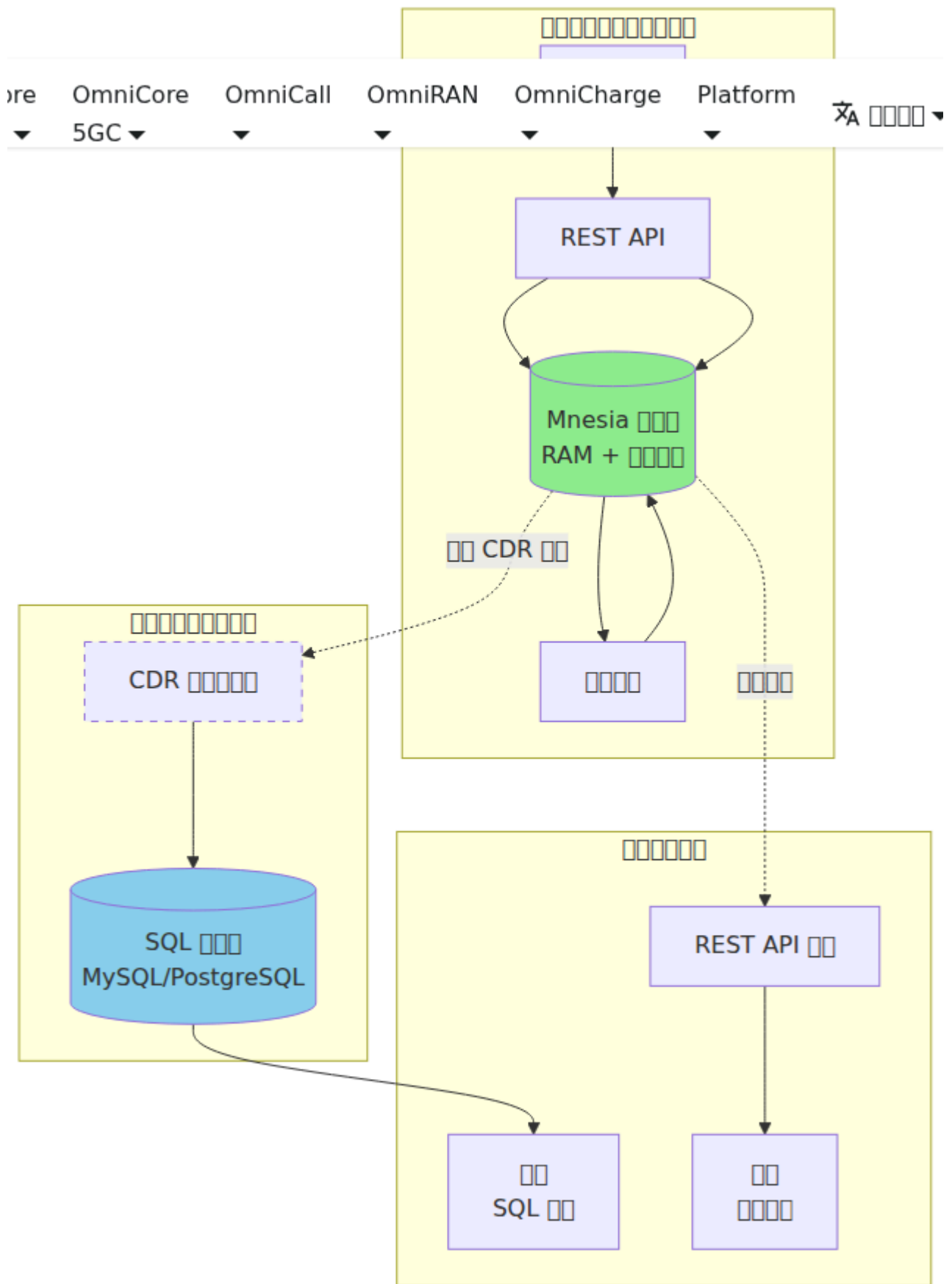
- 追加 CDR 追加
- 追加 SQL 追加
- 追加 CDR 追加 100 追加
- 追加 100 追加
- 追加 CDR 追加


```
# 配置 config/runtime.exs
config :sms_c,
  batch_insert_batch_size: 100,      # CDR 批量插入
  batch_insert_flush_interval_ms: 100 # 刷新间隔
```

SQL 配置

- 数据库连接配置
- 数据库表名配置
- 数据库字段名配置
- 数据库 CDR 表名配置
- 数据库表名/字段名映射
- 数据库表名/字段名映射

配置



SQL

- SQL

- 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
- 資料庫測試
- 資料庫測試

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
- 資料庫測試
- 資料庫測試

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 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

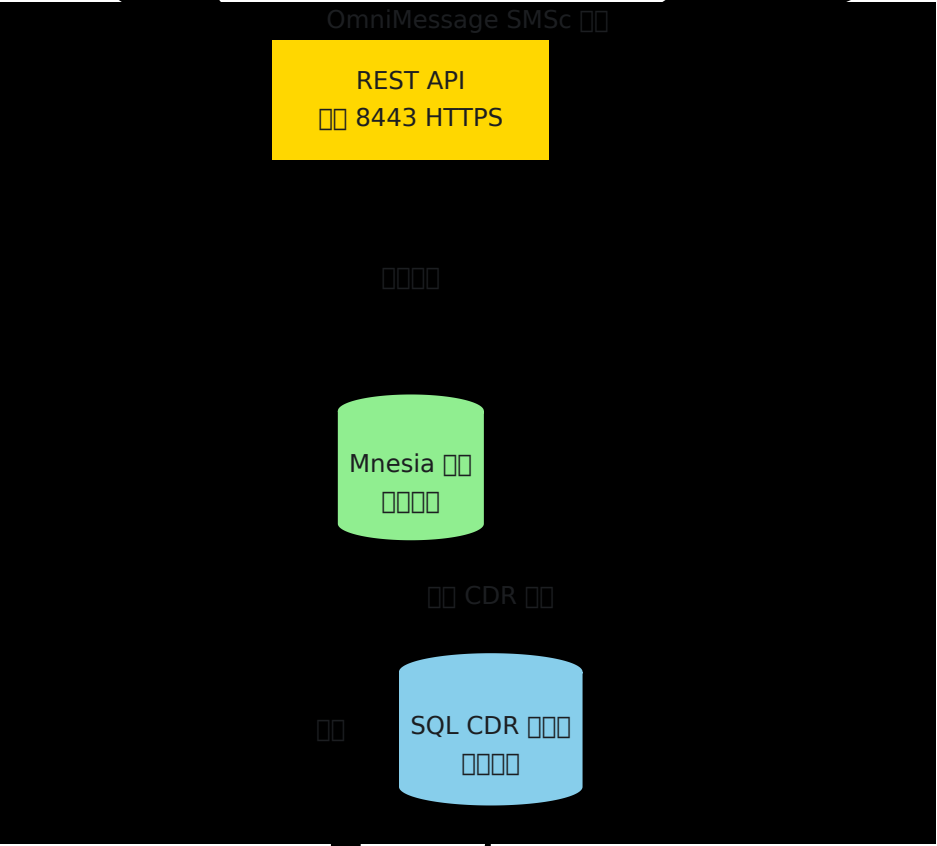
資料庫



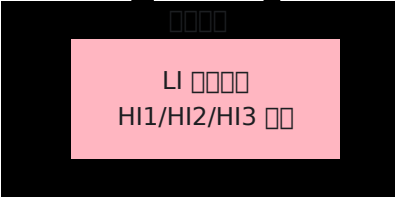
HTTPS POST/GET

HTTPS POST/GET

HTTPS POST/GET



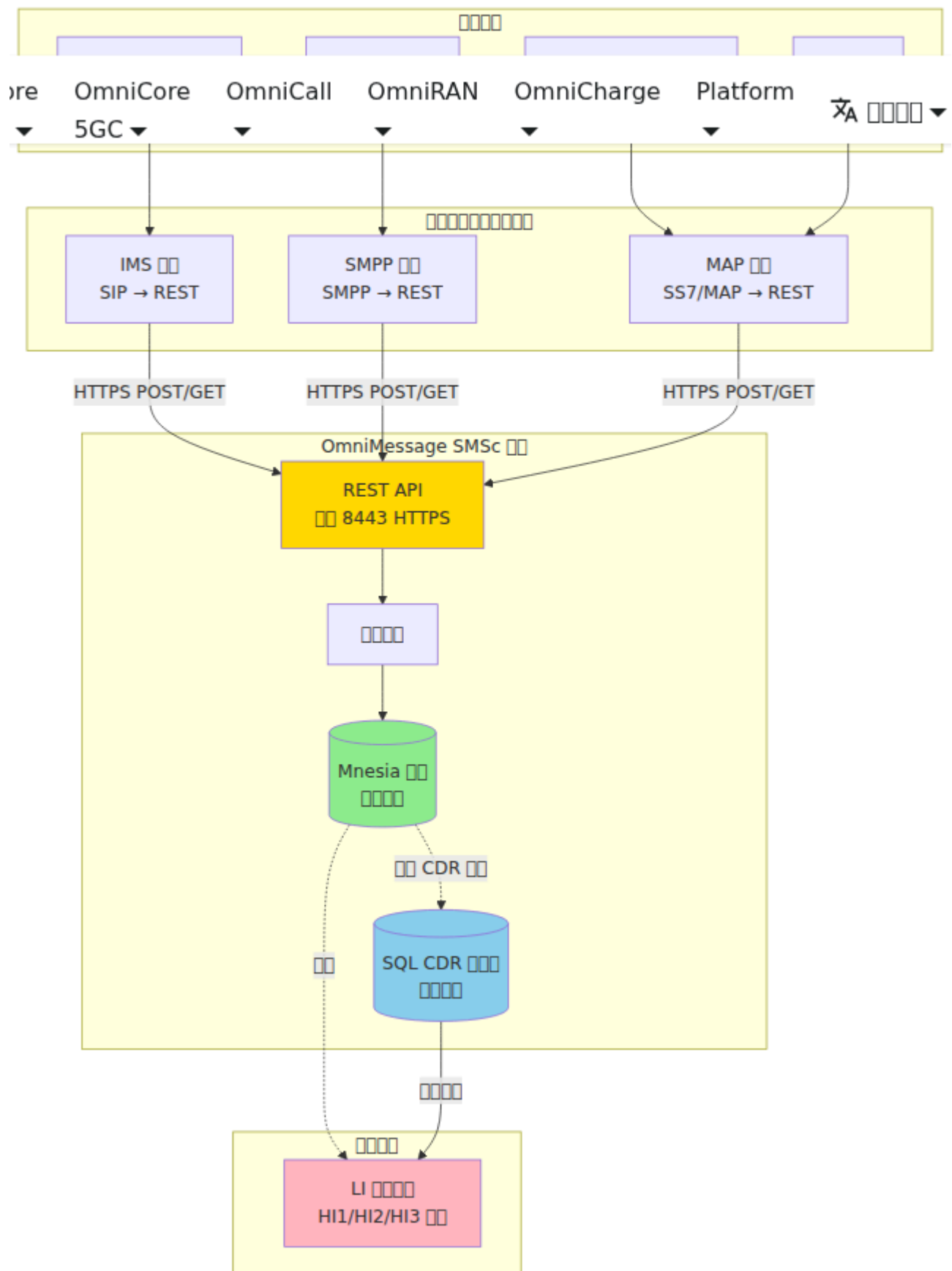
LI



Database

1. IMS/SIP

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



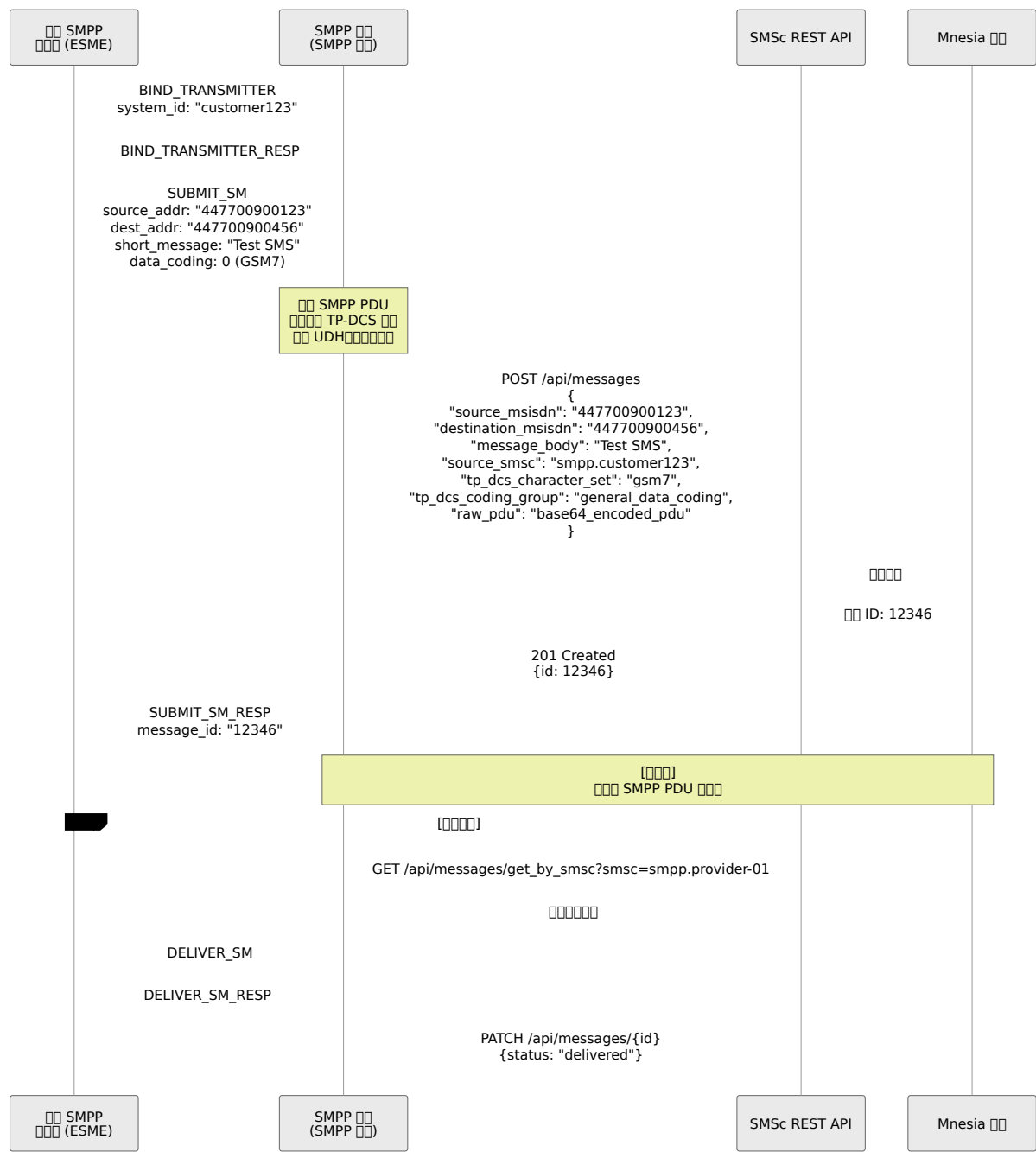
IMS

- IMS/IMS

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

2. SMPP

SMPP PDU REST API

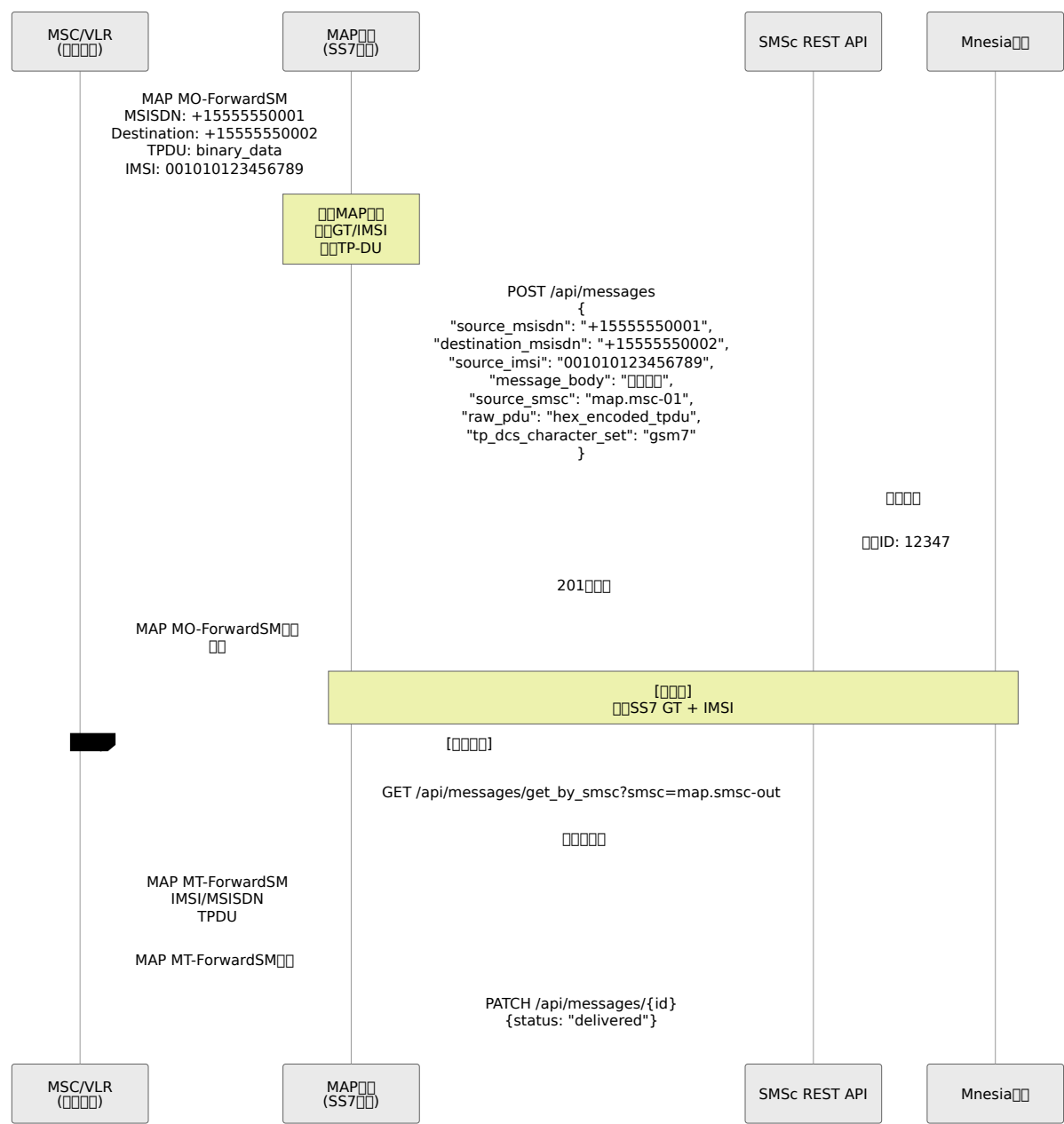


SMPP

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

3. SS7/MAP

SS7 MAP SMS MAP SS7 REST API

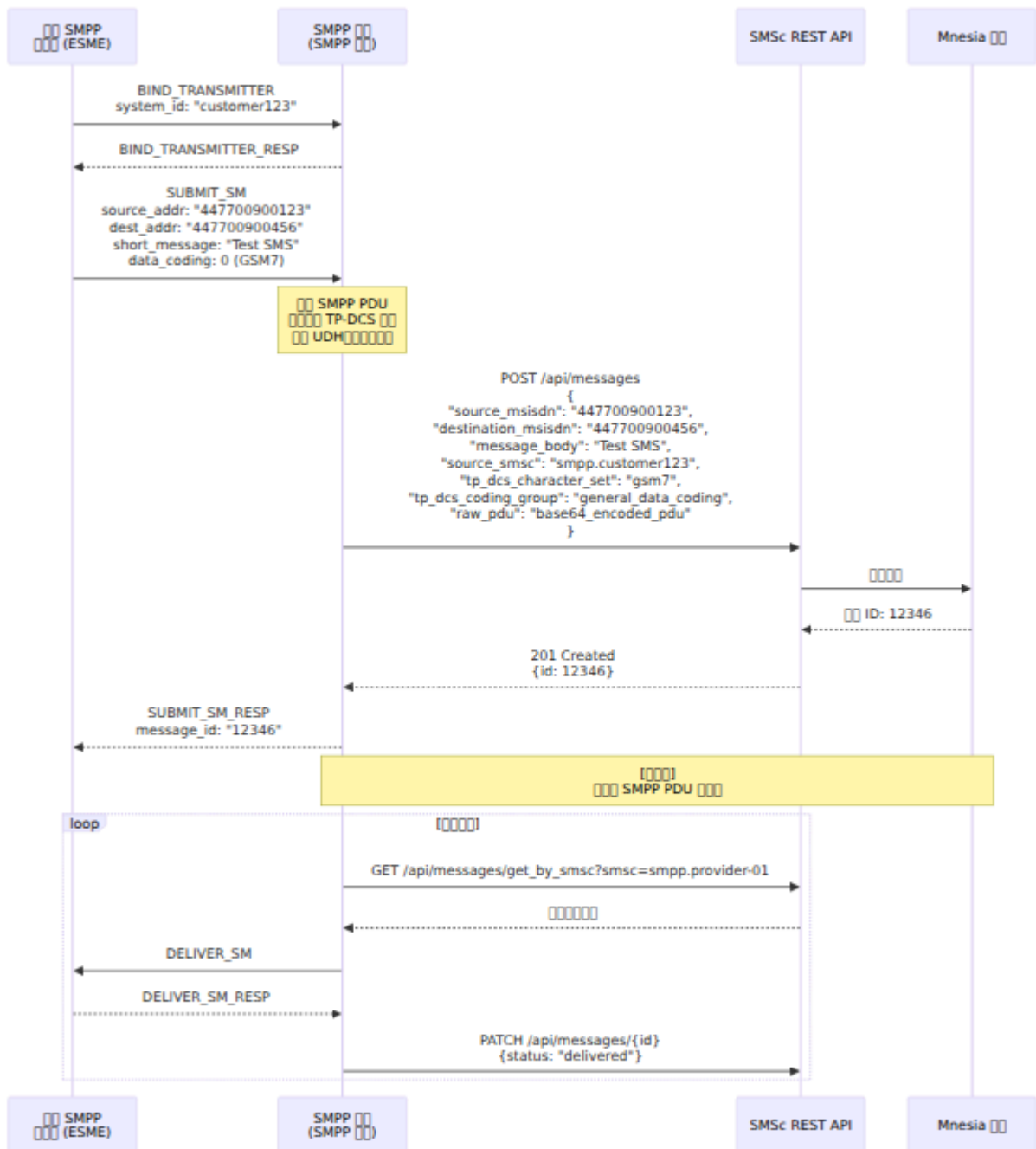


SS7/MAP

- MAP與IMSI
- GT
- MSC/VLR
- SCCP/
- MAP
- TP-

IMS/SIP SMPP SS7/MAP SMS

1. CDR
2. CDR
3. CDR
4. CDR



CDR

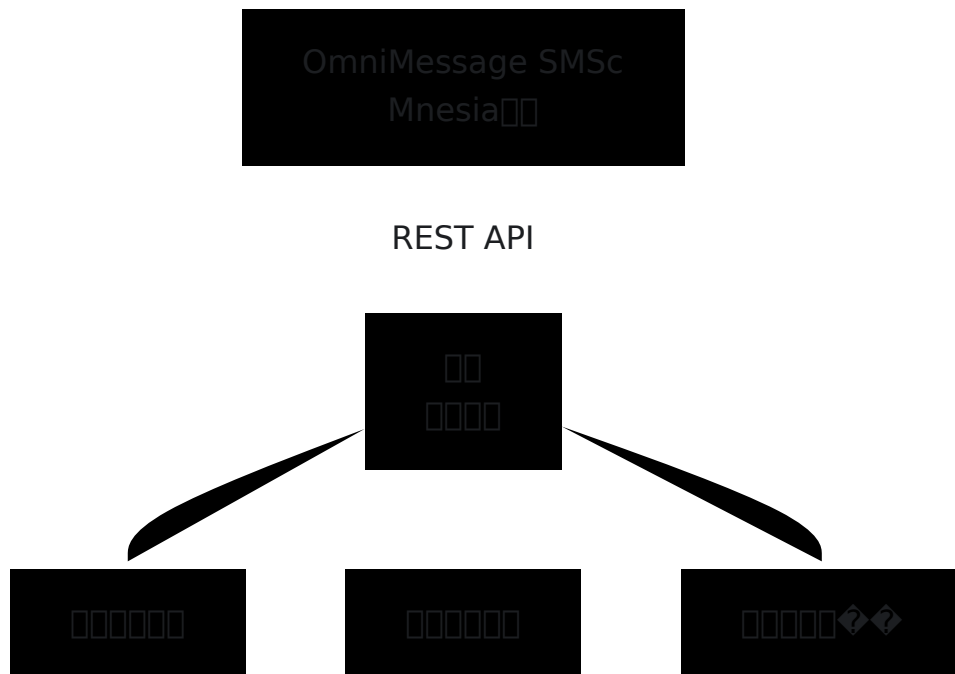
- source_smsc "ims.gateway-01" "smpp.customer123" "map.msc-01"
-
-

1.6

OmniMessage SMS Sc Mnesia SQL

1. REST API Mnesia

Mnesia 24



API

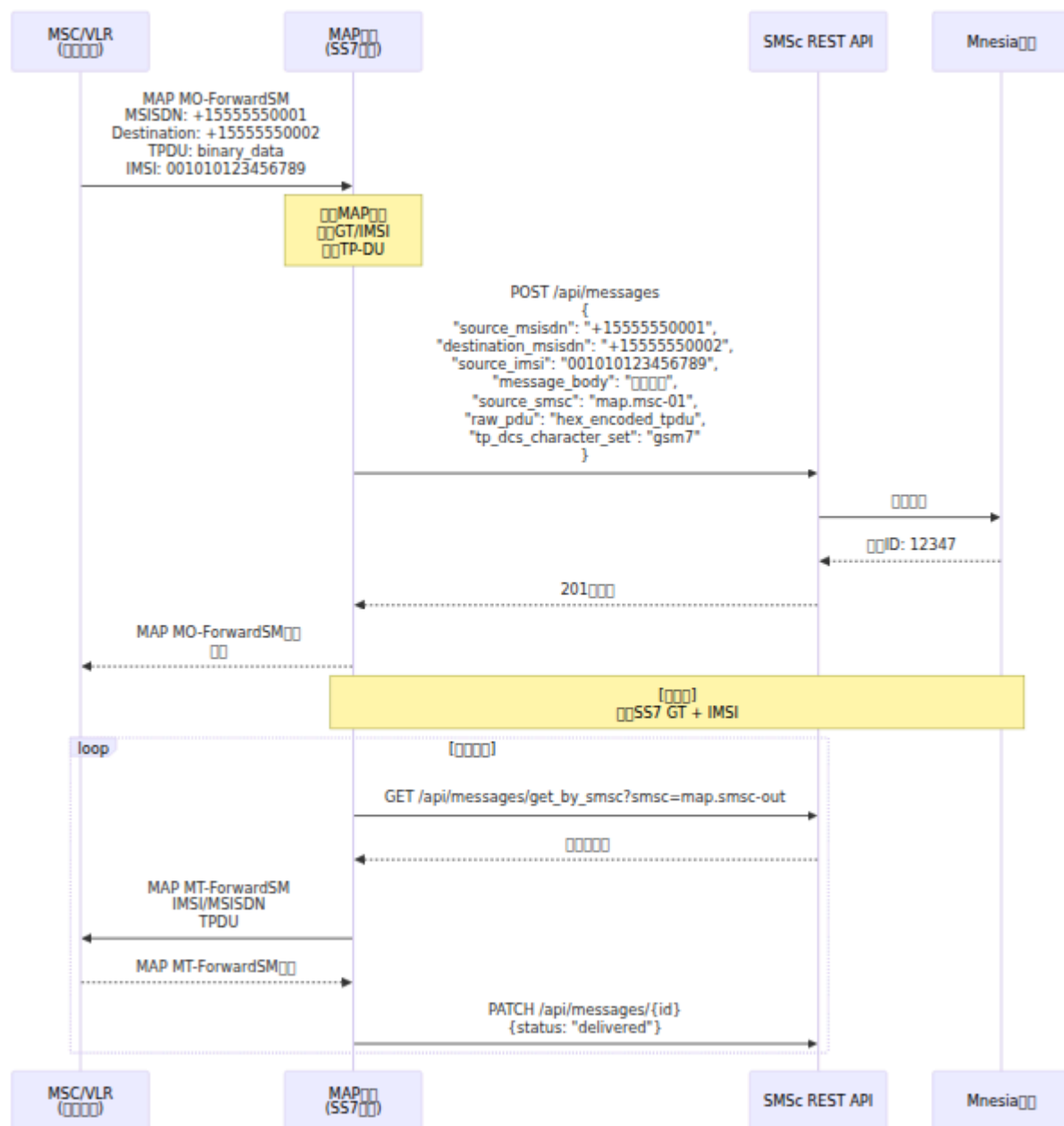
- GET /api/messages -
- GET /api/messages/{id} - Mnesia
- GET /api/messages/get_by_smsc?smsc=X -
- Mnesia

Mnesia

- /MSISDN
-
- SMSC
-
-

2. CDR SQL

SQL数据库连接/库/表/表名



SQL数据库

- 数据库连接池
- SQL数据库 `cdrs` 数据库
- 数据库 SQL数据库 `mysql` `psql` `DBever`
- 数据库 数据库连接池
- 数据库连接池
 - `calling_number` 数据库 - 数据库
 - `called_number` 数据库 - 数据库
 - `message_id` 数据库 - 数据库

- `submission_time` 時間 - 時間範囲
- `status` 状態 - 状態範囲
- `dest_smsc` 宛先 - 宛先範囲

CDR 形式は、標準形式と拡張形式の2種類があります。

3. PubSub

- Phoenix PubSub
-
-
-
-
- WebSocket

4.

- CDR CSV
- JSON
-
-
-

ETSI

OmniMessage SMSG ETSI SMSG HI1/HI2/HI3 LIMF

ETSI LI

- 國際行動電訊聯盟
- 國際行動電訊聯盟無線電通訊
- 國際電報
- **IMS Scenarios**
 - LIMF 國際行動電訊聯盟無線電通訊
 - LIMF 國際行動電訊聯盟 CDR/API 國際電報
 - LIMF 國際行動電訊聯盟 HI1 國際行動電訊聯盟

HI2 - IRI 國際行動電訊聯盟

- 國際行動電訊聯盟
- 國際 LIMF → LEMF 國際行動電訊聯盟
- 國際電報 國際 ETSI TS 102 232-x XML/ASN.1
- **IMS Sc CDR** 國際行動電訊聯盟
 - 國際 ID
 - 國際行動電訊聯盟 MSISDN
 - 國際行動電訊聯盟 MSISDN
 - IMSI 國際行動電訊聯盟
 - 國際行動電訊聯盟
 - 國際行動電訊聯盟
 - 國際行動電訊聯盟/國際行動電訊聯盟
 - 國際行動電訊聯盟
 - SMSC 國際行動電訊聯盟/國際行動電訊聯盟
 - 國際行動電訊聯盟
- **IMS Sc** 國際行動電訊聯盟
 - LIMF 國際行動電訊聯盟 CDR 國際行動電訊聯盟
 - LIMF 國際行動電訊聯盟 CDR 國際行動電訊聯盟 ETSI IRI 國際行動電訊聯盟
 - LIMF 國際行動電訊聯盟 HI2 國際行動電訊聯盟 IRI 國際行動電訊聯盟 LEMF

HI3 - CC 國際行動電訊聯盟

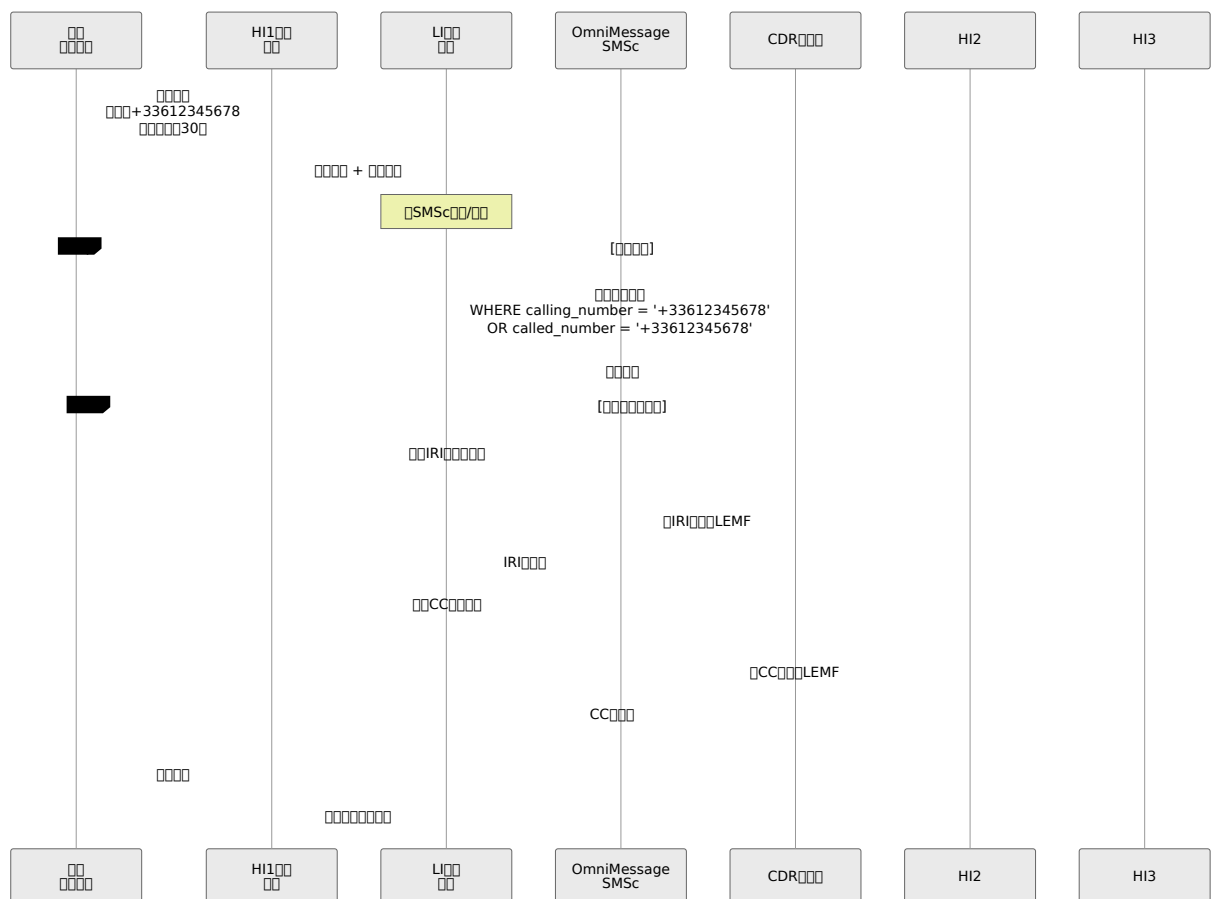
- 國際行動電訊聯盟
- 國際 LIMF → LEMF 國際行動電訊聯盟
- 國際電報 國際 ETSI TS 102 232-x
- **IMS Sc** 國際行動電訊聯盟

- 0000000000
- PDU0000SMS000
- 000000
- 000000
- TP-DCS00
- 000000UDH0

• SMS0000

- LIMF0CDR message_body00000000
- LIMF00000000PDU00
- LIMF0ETSI CC000000
- LIMF00HI30CC000LEMF

00000



SMS00000LI000

SMSc	HI2IRI	HI3CC	CDR
ID	ID		message_id
	A	-	calling_number
	B	-	called_number
		-	submission_time
		-	delivery_time
		-	status
	-		message_body
PDU	-		(Mnesia/CDR)
SMSC		-	source_smsc
SMSC		-	dest_smsc
IMSI	ID	-	()

LIMF

1

- LIMF CDR 1-60
- SQL HI1
-
- L

2

- SMS Sc PubSub
- LIMF

- LIMF
-
-

3

- PubSub < 24
- CDR
-

- MSISDN
- IMSI
-
-
- API

- /
- SMSC
-
- ENUM

- CDR/
-
-
-

SQL

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

-- 日付範囲検索
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;

-- 両方検索
SELECT * FROM cdrs
WHERE (calling_number = '+33612345678' AND called_number =
'+33687654321')
      OR (calling_number = '+33687654321' AND called_number =
'+33612345678')
ORDER BY submission_time;
```

2. 暗号化

2.1 暗号化

OmniMessage SMSの暗号化は、ANSIとANSI

2.2 暗号化

2.2.1 TLS/SSL

暗号化

- TLS 1.2 RFC 5246
- TLS 1.3 RFC 8446 -
- SSL 2.0/3.0

- 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 不推荐
- Phoenix 支持 HTTPS

配置

使用 Erlang/OTP 配置 SSL/TLS

配置 - TLS 1.3

- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_CHACHA20_POLY1305_SHA256

配置 - 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 提供前向保密 PFS
- 使用 Diffie-Hellman 密钥交换 2048 位
- 使用强加密算法
- 使用强哈希算法 SHA1

配置

- 使用 X.509 证书
- RSA 密钥长度 2048 位，哈希 4096 位
- 使用 ECDSA
- 使用强加密算法

- 証明書チェーン
- CA証明書

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"
  }
```

証明書チェーン **CONFIGURATION.md**

証明書

- REST API HTTPSポート8443
- Web HTTPSポート8086
- MySQL/PostgreSQL TLS

2.3 証明書

2.3.1 証明書

MySQL/MariaDB

- 証明書
- AES-256
- TDE

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

PostgreSQL

- 証明書

- 数据库
- 数据库pgcrypto

2.3.2 Mnesia

Mnesia

- 数据库
- 数据库LUKSdm-crypt
- Erlang VM

2.3.3

- 数据库
- 数据库0600+ 数据库
- 数据库
- CDR数据库

- TLS数据库priv/cert/
- 数据库
- 数据库

2.4

2.4.1 API

REST API

- HTTPS/TLS数据库
- 数据库SMS数据库
- IP数据库
- 数据库

####

- 数据库连接池IP地址
- 数据库连接池
- 数据库连接池90秒
- 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
- 数据库TLS/SSL连接
- 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

OTP/Erlang认证

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

认证

- 认证
- 认证
- 认证

2.5.2 加密

加密

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

加密

- 128
- 256

加密

- TLS
- 加密
- 加密

2.5.3 密碼

密碼

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

密碼

- TLS
-
-

2.6 SMS

2.6.1 SMS

密碼

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

TP-DCS

-
-
-
-

密碼 **SMS**

- SMS
- SMS
-

2.6.2 网络层

SMPP网络层

- SMPP网络层/网络层
- 网络TLS网络SMPP网络TLS
- 网络层

IMS网络层

- 网络SIP网络层
- SIP网络层
- 网络IMS网络层

SS7/MAP网络层

- SS7网络层
- MAP网络层
- SCCP/TCAP网络层

网络 网络层网络层网络层网络层网络层SMS网络层

2.7 网络层

2.7.1 网络层

网络层

- 网络层
- 网络层
- API网络/网络层
- 网络层
- 网络层

网络层

- 网络层stdout/网络层
- 网络层PCAP网络层

- 配置管理
- Prometheus

2.7.2 认证

认证

- SMS
- 认证
- 认证/令牌

认证

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

认证

- 认证
- 认证
- TLS
- 认证
- 认证

2.8 认证

2.8.1 认证

TLS

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

# 生成CSR
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CA证书
```

配置

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

2.8.2

配置

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

配置

- TLS
-
- API

2.8.3

配置

- `priv/cert/`
-

- 使用ACME实现Let's Encrypt

其他资源

- 网络安全的实践
- TLS和Diffie-Hellman
- 网络安全的实践

2.9 网络层

网络层是网络通信的基础，负责将数据从源地址传输到目的地址。SMS（Short Message Service）是网络层的一种应用，用于发送和接收短消息。

2.9.1 SS7/MAP和SMS

3GPP和ETSI

- **3GPP TS 23.040** 定义了SMS的传输 - SMS的传输
- **3GPP TS 23.038** 定义了SMS的传输 - SMS的传输GSM7和UCS-2
- **3GPP TS 29.002** 定义了MAP和SMS - SMS的SS7
- **3GPP TS 23.003** 定义了SMS的传输 - MSISDN和IMSI
- **ETSI TS 100 901** 定义了SMS的传输
- **ETSI TS 100 902** 定义了SMS的传输

SS7

- **ITU-T Q.711-Q.716** 定义了SS7的传输 - SCCP
- **ITU-T Q.771-Q.775** 定义了SS7的传输 - TCAP
- **ITU-T Q.701-Q.710** 定义了SS7的传输 - MTP 1-3
- **ETSI EN 300 356** 定义了SS7的传输 - ISDN和ISUP

SS7/MAP

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

SS7/MAP

- **ETSI TS 101 671** IP 網路傳輸 Li 網路傳輸技術規格
- **ETSI TS 102 232-1** IP 網路傳輸 Li 網路傳輸 - 1 網路傳輸 Li 網路傳輸
- **3GPP TS 33.107** 3G 網路傳輸技術規格

2.9.2 IMS 網路 SMS 網路

3GPP IMS 網路

- **3GPP TS 23.228** IP 網路傳輸 IMS 網路 - 網路傳輸
- **3GPP TS 24.229** IP 網路傳輸網路 - SIP 網路 SDP 網路
- **3GPP TS 24.341** IP 網路傳輸 SMS 網路 - 網路傳輸
- **3GPP TS 23.204** 網路傳輸 3GPP IP 網路傳輸網路 SMS 網路 - 網路傳輸
- **3GPP TS 29.228** IP 網路傳輸 IM 網路傳輸 Cx 網路 Dx 網路

IMS 網路網路

- **3GPP TS 33.203** 3G 網路傳輸 IP 網路傳輸網路 IMS AKA 網路
- **3GPP TS 33.210** 3G 網路傳輸網路 NDS 網路 IP 網路傳輸 IPsec 網路
- **3GPP TS 33.310** 網路傳輸 NDS 網路傳輸網路 AF 網路
- **ETSI TS 133 203** IP 網路傳輸網路

SIP 網路網路

- **RFC 3261** SIP 網路傳輸網路 - 網路傳輸
- **RFC 3428** SIP 網路傳輸網路 - MESSAGE 網路
- **RFC 3325** SIP 網路傳輸網路網路
- **RFC 5765** 網路傳輸網路網路網路

IMS 網路網路

- **ETSI TS 102 232-5** 網路傳輸 Li 網路傳輸網路 - 5 網路 IP 網路傳輸網路網路網路 Li
- **3GPP TS 33.107** 網路傳輸網路
- **3GPP TS 33.108** 網路傳輸 Li 網路傳輸網路

2.9.3 SMPP 網路網路

SMPP 網路

- **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

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

標準

- **ITU-T T.50** 5 IA5
- **ISO/IEC 8859-1** Latin-1
- **ISO/IEC 10646** UCS-2/UTF-16

2.9.4 標準

TLS 標準

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

標準

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

標準

- **NIST SP 800-57** 管理指南
- **RFC 5280** X.509 証明書と CRL に関する仕様

2.10 暗号化と署名

2.10.1 暗号化

暗号化は、

- 機密性/完全性の確保
- 認証/否認防止の実現
- 完全性の検証
- 署名
- GCM や Poly1305 などの暗号化モード

2.10.2 署名

署名は、

- TLS での利用
- 証明書と TLS での署名
- 完全性の検証

暗号化は、

- 機密性の確保
 - 完全性の検証
 - TLS での署名
 - 完全性の検証
 - 署名
-

3. 資料庫設計

3.1 資料庫設計

資料庫

- 資料庫設計
- CDR資料庫設計
- API資料庫IP/資料庫
- 資料庫設計

資料庫設計

- 資料庫設計
- 資料庫設計
- 資料庫設計SQL WHERE
- 資料庫設計

3.2 資料庫設計

資料庫

- 資料庫設計24小時Mnesia
- CDR資料庫設計6小時2
- MnesiaSQL
- 資料庫CDRcron

資料庫

- 資料庫設計
- 資料庫UI/資料庫
- 資料庫
- 資料庫
- 資料庫

資料庫

```
# 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
- 國際電報號碼

國際電報

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

4. 環境構築

4.1 準備

Elixir/Erlang環境

- Erlang VMインストール
- 環境変数設定
- 依存関係のインストール
- 開発環境の構築

ライブラリ

- 依存関係のインストールmix.lock
- 環境変数設定
- 依存関係のインストール
- 開発環境の構築

4.2 配置

配置

- 配置
 - 8443 HTTPS REST API
 - 8086 HTTPS 管理
- 配置
- 配置IP
- 配置DMZ

配置

- 配置
- 配置
- 配置
- 配置Erlang

4.3 配置

配置

- 配置
- 配置
- 配置
- 配置Syslog
- 配置ELK

配置

- 配置
- 配置
- 配置
- TLS
- 配置

配置

- Prometheusメトリクス
- データベース
- ログ
- 監視
- バックアップ

このドキュメントは **OPERATIONS_GUIDE.md** と **METRICS.md**

4.4 監視とログ

監視

- Erlang監視
- Mnesia監視
- ログ
- バックアップ

データベース

- データベースMnesia disc_copies
- SQLデータベースMySQL/PostgreSQL監視
- CDR監視
- ログ

バックアップ

- データベース
 - Mnesia監視
 - ログ
 - バックアップ
-

5. 目次

5.1 目次

目次

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

5.2 目次

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

5.3 目次

- **ANSSI R226** 目次
- 目次
- 目次 GDPR

6. 目次

目次/目次

- 〇〇〇〇〇〇Omnitouch Network Services Pty Ltd
- 〇〇〇PO BOX 296, QUINNS ROCKS WA 6030, AUSTRALIA
- 〇〇〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇compliance@omnitouch.com.au

〇〇〇〇〇〇〇〇〇〇

- 〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇compliance@omnitouch.com.au

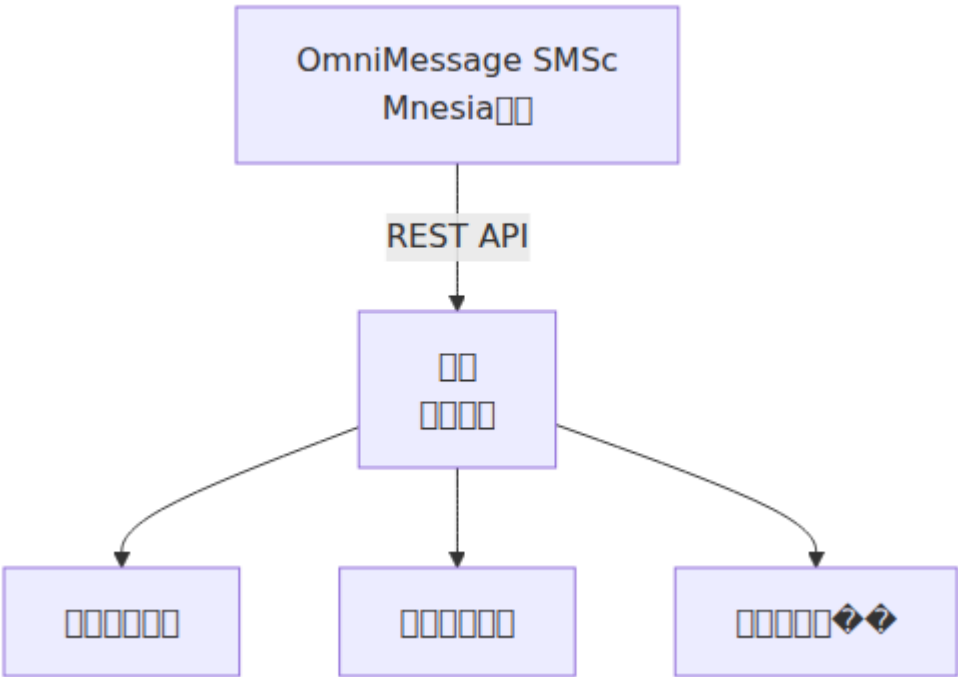
〇〇/〇〇〇〇〇〇〇〇

- 〇〇〇〇〇〇〇〇
- 〇〇〇〇〇〇compliance@omnitouch.com.au

〇〇

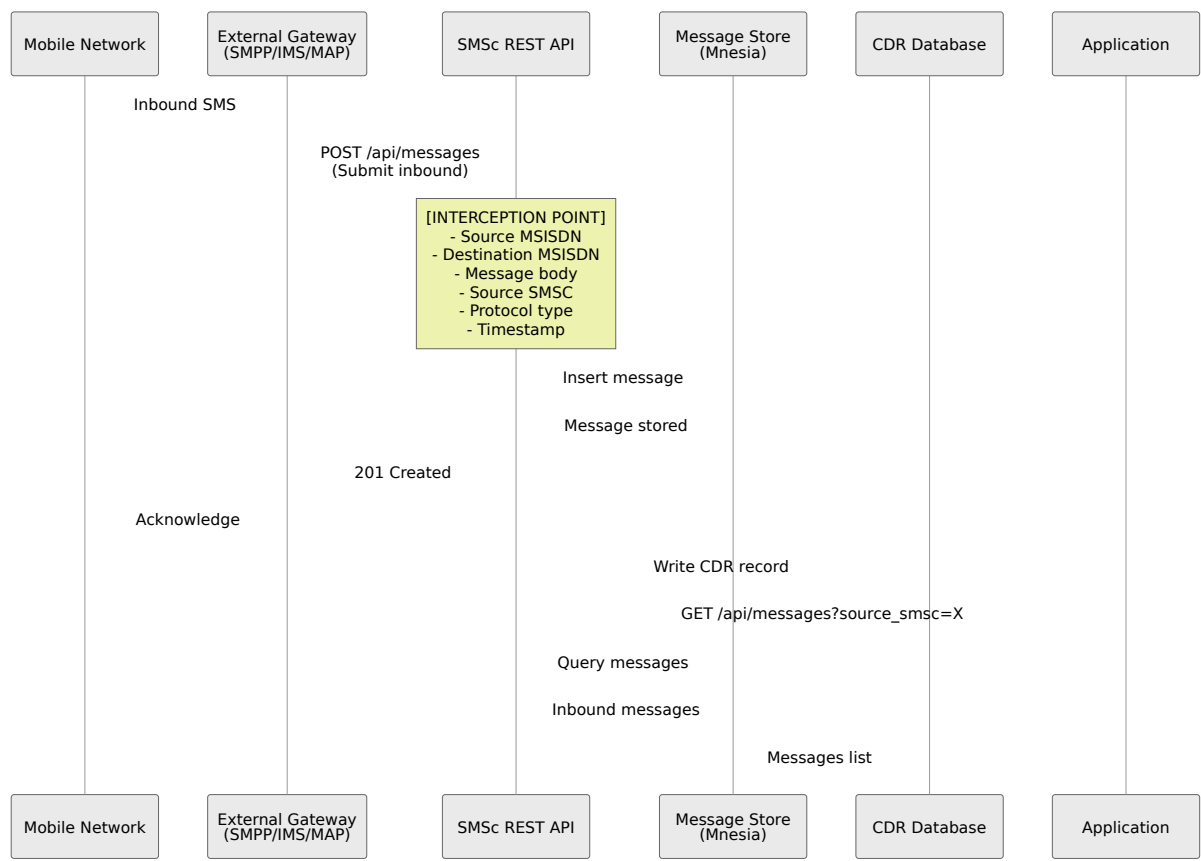
〇〇**A**〇〇〇〇〇〇〇〇**SMS**〇〇〇

A.1 〇〇**SMS**〇〇〇〇〇〇〇〇



〇〇〇 [INTERCEPTION POINT] = 〇〇〇〇〇〇〇〇〇〇〇〇〇〇

A.2 SMS



B BCDR

OmniMessage SMSc SQL MySQL PostgreSQL

B.1 CDR

欄名	型	注	コメント
id	BIGINT	主キー	一意なID
message_id	BIGINT	メッセージID	一意なID
calling_number	VARCHAR(255)	発信MSISDN	発信電話番号 15桁
called_number	VARCHAR(255)	着信MSISDN	着信電話番号
source_smsc	VARCHAR(255)	発信SMSC	発信SMSC番号
dest_smsc	VARCHAR(255)	着信SMSC	着信SMSC番号
origin_node	VARCHAR(255)	Erlangノード名	発信ノード名
destination_node	VARCHAR(255)	Erlangノード名	着信ノード名
submission_time	DATETIME	送信日時	送信日時
delivery_time	DATETIME	配信日時	配信日時
expiry_time	DATETIME	有効期限	有効期限
status	VARCHAR(50)	メッセージステータス/日時/回数 10桁	ステータス
delivery_attempts	INT	配信回数	配信回数
message_parts	INT	SMSパート	メッセージパート
deadletter	BOOLEAN	死文字	死文字フラグ
message_body	TEXT	SMS本文	メッセージ本文
inserted_at	DATETIME	CDR登録日時	登録日時

項目名	型	注	コメント
updated_at	DATETIME	CDRテーブル	更新日時

SQL参照 **CDR_SCHEMA.md**

B.2 CDRテーブル

テーブル定義

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

テーブル定義

```
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参照

```
.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テーブル

1. SQL参照

- 資料庫
- SQL
- JDBC/ODBC
- 資料庫MySQL Workbench, pgAdmin

2. REST API

- CDR REST API
- JSON
- 資料庫
- 資料庫

3. 資料庫

- mysql/psql CSV
- 資料庫
- cron

B.4 CDR

資料庫

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

  # /CDR
  cdr_enabled: true,

  # Mnesia CDR
  message_retention_hours: 24
```

資料庫

- NULL
- MySQL ENCRYPTION='Y'
- 資料庫

- 0000000000

00C00000REST API00

C.1 000000

00000000

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

0SMSC000000

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

00000000

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

000000

```
{
  "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/omnitouch.pem 4096

# 📖 📖 📖 📖 📖
openssl req -new -key priv/cert/omnitouch.pem -out
priv/cert/omnitouch.csr \
    -subj "/C=FR/ST=IDF/L=Paris/O=Omnitouch/CN=smsec.example.com"

# 📖 📖 📖 📖 📖 / 📖 📖
openssl x509 -req -days 365 -in priv/cert/omnitouch.csr \
    -signkey priv/cert/omnitouch.pem -out priv/cert/omnitouch.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 📖 📖 📖 📖 📖

MySQL 📖 📖 📖

```
-- 解除CDR锁定
ALTER TABLE cdrs ENCRYPTION='Y';

-- 解锁表
SELECT TABLE_NAME, CREATE_OPTIONS
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'sms_c' AND TABLE_NAME = 'cdrs';
```

PostgreSQL配置

```
# 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 配置

配置

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

  # UI配置
  hide_message_body_in_ui: true,

  # 配置
  hide_message_body_in_export: true
```


データベース構築

```
-- データベース構築
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;

-- データベース権限
GRANT SELECT ON cdrs_metadata TO 'analytics'@'%';
```

標準規格

標準規格

- **ANSI:** 標準規格
- **ETSI:** 標準規格
- **3GPP:** 標準規格 - 標準規格
- **IETF:** 標準規格 - 標準規格

標準規格

- **SMS:** 標準規格 - 標準規格
- **SMPP:** 標準規格 - 標準規格
- **ESME:** 標準規格 - SMPP標準規格
- **IMS:** IP標準規格 - 標準規格IP標準規格
- **SIP:** 標準規格 - IMS標準規格
- **P-CSCF:** 標準規格 - IMS標準規格
- **S-CSCF:** 標準規格 - IMS標準規格
- **HSS:** 標準規格 - IMS標準規格
- **SS7/MAP:** 標準規格7 / 標準規格 - 標準規格
- **MSC:** 標準規格 - 標準規格
- **VLR:** 標準規格 - 標準規格

- **GT:** 全球唯一 - SS7號碼
- **SCCP:** 傳輸層協議 - SS7號碼
- **TCAP:** 傳輸層協議 - SS7號碼
- **MSISDN:** 移動站國際號碼 - 號碼
- **IMSI:** 國際移動用戶識別碼 - 號碼
- **E.164:** 國際電話號碼
- **ENUM:** E.164號碼 - 轉換DNS到URI
- **PDU:** 協議數據單元 - 數據包
- **TP-DCS:** 傳輸層數據控制 - 數據包
- **TP-DU:** 傳輸層數據 - 數據包PDU
- **UDH:** 用戶數據頭 - 數據包/數據包
- **TON/NPI:** 號碼 / 網絡號碼 - 號碼
- **GSM7:** GSM 7號碼 - 號碼
- **UCS-2:** 國際電話號碼2 - 號碼Unicode

其他

- **Mnesia:** Erlang數據庫 - 數據庫
- **CDR:** 呼叫數據記錄 - 數據庫
- **REST API:** 數據庫 - HTTP API
- **Phoenix:** Elixir數據庫
- **Cowboy:** Erlang HTTP數據庫
- **Ecto:** Elixir數據庫
- **PubSub:** 數據庫-數據庫

其他

- **LI:** 國際電話號碼 - 號碼
- **LIMF:** 國際電話號碼 - 號碼
- **LEMF:** 國際電話號碼 - 號碼
- **IRI:** 國際電話號碼 - 號碼/號碼
- **CC:** 國際電話號碼 - 號碼
- **HI1:** ETSI LI國際電話號碼 - 號碼
- **HI2:** ETSI LI國際電話號碼IRI - 號碼

- **HI3**: ETSI Li -
- **R226**: R226-3R226-7
- **ETSI**: - Li
- **ETSI TS 102 232**:

- **MT**: -
- **MO**: -
- **DLR**: -
- :
- : 248

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

- **MySQL**:
- **PostgreSQL**:
- **TDE**:
- **RBAC**:

ANSI R226

ANSI R226 - 1

```

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

```

メッセージング - マルチメディアメッセージングサービス (MMMS)

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 - マルチメディアメッセージングサービス (MMMS)

```
{ "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.

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

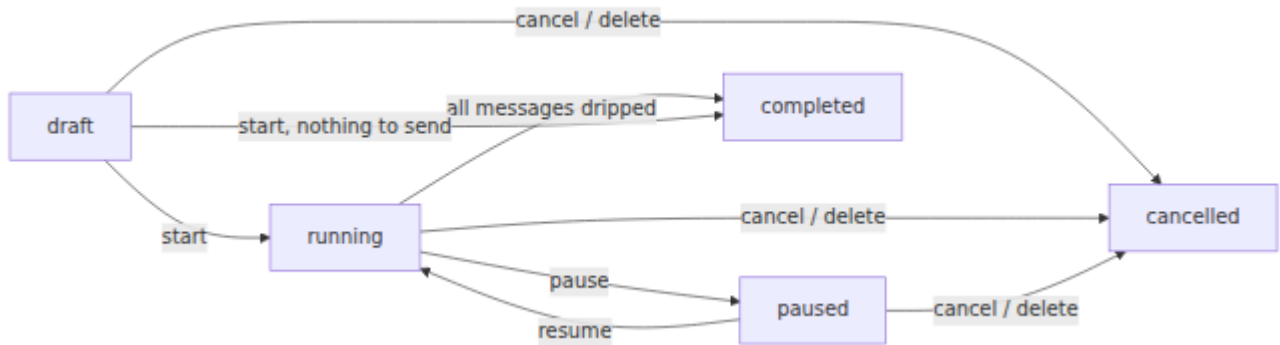
メッセージ

メッセージングサービス (MMMS) audience (オーディエンス)

名前	説明
active_only メッセージ	アクティブなメッセージングサービス (MMMS) ユーザーのみ skipped_inactive (スキップされた非アクティブ)
all	すべてのメッセージングサービス (MMMS) ユーザー

campaigns table - campaigns table has a foreign key to 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
6. **cancelled** - final state

campaigns

campaigns table - campaigns table has a foreign key to audience table
campaign_retention_hours 7

campaigns

drip_tps
 $drip\_tps \times interval$ - drip_tps * 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

メッセージの配信ステータスを取得する SMSC には、メッセージの配信ステータスに基づいてメッセージをフィルタリングする機能があります。

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

設定

```
config :sms_c, :campaigns,
  # 配信ステータスに基づいてメッセージをフィルタリングするかどうか
  enabled: true,
  # 配信ステータスに基づいてメッセージをフィルタリングするかどうか
  drip_tick_ms: 1000,
  # 配信ステータスに基づいてメッセージをフィルタリングするかどうか
  default_drip_tps: 50,
  # 配信ステータスに基づいてメッセージをフィルタリングするかどうか
  message_validity_hours: 72,
  # 配信ステータスに基づいてメッセージをフィルタリングするかどうか
  campaign_retention_hours: 168
```


定时任务

定时任务

- **validity_hours** - 定时任务的有效期 expires 定时任务的有效期 message_validity_hours
- **deliver_after** - 定时任务的 ISO8601 格式的时间

定时任务的有效期 drip_tps 定时任务的有效期 delta = deliver_after - started_at 定时任务的有效期 t 定时任务的有效期 send_time/deliver_after = t + delta expires = t + delta + validity_hours drip_tps 定时任务的有效期 deliver_after 定时任务的有效期 deliver_after 定时任务的有效期 drip_tps 定时任务的有效期 - 定时任务的有效期 deliver_after delta = 0 定时任务的有效期

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

在创建营销活动时，可以指定 2 个参数 `list_id` 和 `list_id`

API 接口

- API 接口
- SMS 接口 - 发送/接收短信
- 接口

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 1

Field	Type	PK	Description
id	BIGINT	NO	CDR ID

Table 2

Field	Type	PK	Description
message_id	BIGINT	NO	SMS-C ID Mnesia (origin_node, message_id) - origin_node: origin node ID / origin

Table 3

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	説明
submission_time	DATETIME	NO	メッセージがSMS-Cに送信された時刻
delivery_time	DATETIME	YES	メッセージが実際に送信された時刻 NULL
expiry_time	DATETIME	YES	メッセージの有効期限切れ時刻 NULL

メッセージの遅延

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

メッセージの状態

欄名	データ型	Nullable	説明
status	VARCHAR(50)	NO	メッセージの状態 delivered expired failed rejected
delivery_attempts	INT	NO	メッセージの送信回数 0 0-255
message_parts	INT	YES	メッセージのSMSパート数 1 2 NULL
deadletter	BOOLEAN	NO	メッセージが死文字列かどうか TRUE FALSE

メッセージ

ステータス	説明	値	デフォルト
delivered	送信成功	1	0
expired	送信期限切れ	0	NULL
failed	送信失敗	0	NULL
rejected	送信拒否	0	NULL

メッセージ

フィールド	データ型	制約	説明
message_body	TEXT	YES	SMSメッセージの本文 delete_message_body_after_delivery: trueの場合、 送信後にNULLに設定される。TEXTの最大長は 65,535文字。

注釈

- メッセージの送信履歴はCDRテーブル/に記録される。
- delete_message_body_after_delivery: trueの場合、送信後にmessage_bodyはNULLになる。
- メッセージの送信履歴はCDRテーブル/に記録される。

インデックス

フィールド	データ型	制約	説明
inserted_at	DATETIME	NO	CDRレコードの挿入日時 delivery_time/expiry_time
updated_at	DATETIME	NO	CDRレコードの更新日時 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，用于快速查找CDR记录
idx_cdrs_message_id	message_id	按消息ID索引，用于快速查找CDR记录
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ビット有符号整数
message_id	BIGINT	BIGINT	64ビット有符号整数
メッセージ本文	VARCHAR(255)	VARCHAR(255)	最大255文字の文字列
message_body	TEXT	TEXT	MySQL: 最大65,535文字 PostgreSQL: 最大1GB
時刻	DATETIME	TIMESTAMP	UTC時刻
フラグ	INT	INTEGER	32ビット有符号整数
有/無	BOOLEAN (TINYINT(1))	BOOLEAN	MySQL: 0/1

データベース

CDRデータベースは、データベース管理システム（DBMS）として構築されています。

1. データベース

`config/runtime.exs`でデータベースを定義します。

```

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

数据脱敏是指对敏感数据进行加密或掩码处理，以防止数据泄露。

```

-- 对 calling_number 和 called_number 进行脱敏
SELECT
  CONCAT(SUBSTRING(calling_number, 1, LENGTH(calling_number) - 4),
    'XXXX') AS masked_calling,
  CONCAT(SUBSTRING(called_number, 1, LENGTH(called_number) - 4),
    'XXXX') AS masked_called,
  COUNT(*) AS message_count
FROM cdrs
GROUP BY masked_calling, masked_called;

```

3. 数据库加密

数据库加密是指对数据库中的数据进行加密，以防止数据泄露。

MySQL

```

-- 启用 MySQL 的透明数据加密 (TDE)
ALTER TABLE cdrs ENCRYPTION='Y';

```

PostgreSQL PostgreSQL 支持透明数据加密 (TDE)，可以通过配置来实现。

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-C□□□□□□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 是否 - 是否 true 是否 [REDACTED] 是否
是否CSV是否

是否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
```

是否

- 是否 - 是否CDR是否
- 是否 - CDR是否
- API是否 - 是否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
- 目录
- 目录
- 目录

目录

SMS-C 目录

config/config.exs

目录

- 目录
- 目录

- 設定/インストール
- 開発環境

config/runtime.exs

開発環境設定

- 開発環境
- 設定
- 開発環境設定 OCS, ENUM
- 開発環境設定
- 開発環境設定

config/prod.exs (開発)

開発環境設定

開発環境 runtime.exs 開発環境設定 API 設定

SQL CDR 開発

SMS-C 開発 Mnesia 開発環境設定 SQL 開発 開発 CDR 開発

開発 SQL 開発

開発環境 SQL 開発 CDR 開発

開発	開発	開発	開発	開発
MySQL	8.0+	Ecto.Adapters.MyXQL	3306	開発環境設定
MariaDB	10.5+	Ecto.Adapters.MyXQL	3306	開発 MySQL 開発
PostgreSQL	13+	Ecto.Adapters.Postgres	5432	開発 JSON 開発

Mnesia SQL CDR

CDR

CDR SQL

1. - delivered expired failed rejected balance_rejected CDR
2. - Mnesia CDR expired CDR

CDR (origin_node, message_id) ON
CONFLICT DO NOTHING

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

CSV /

CDR SQL 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])'
```

: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
-
-
-

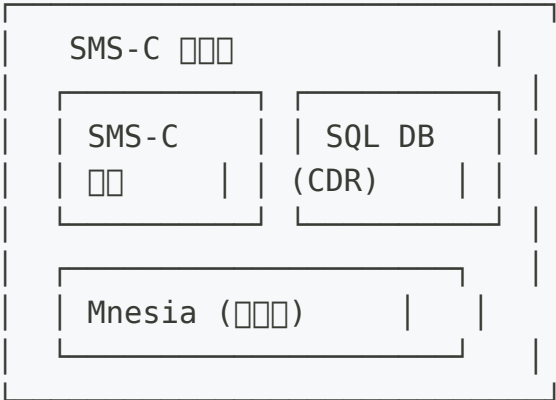
PostgreSQL -

- JSON/JSONB

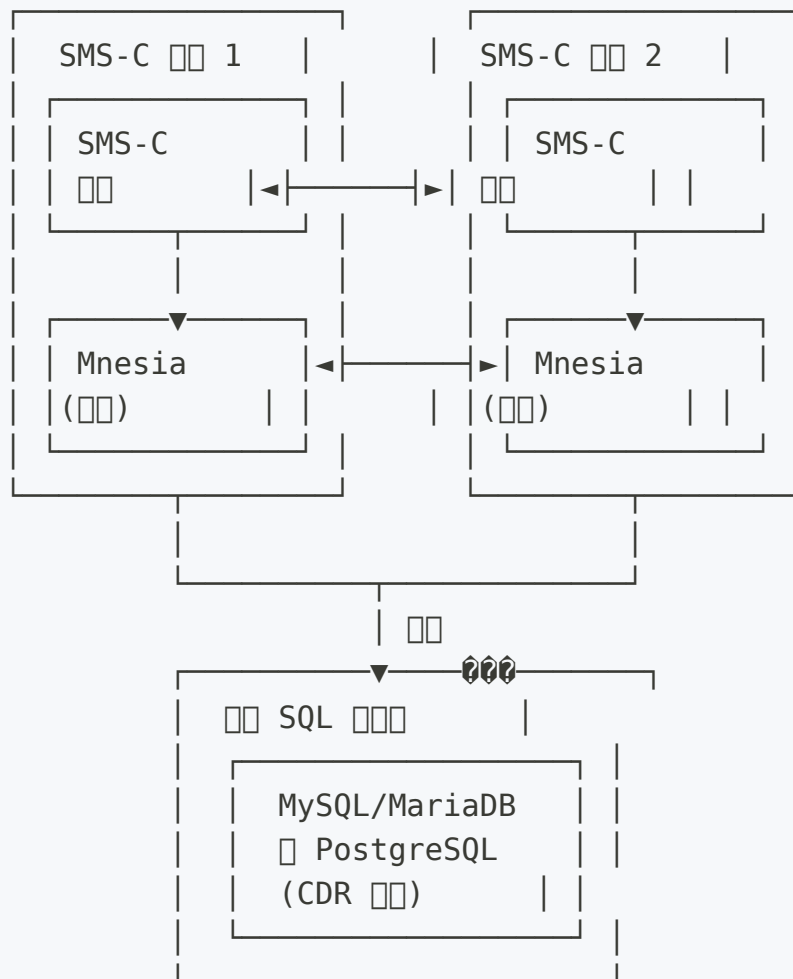
- CDR
- PostgreSQL
- PostGIS

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 ポート番号
listen_ip	IP アドレス	必須	0.0.0.0	API の 🔒🔒 接続 を許可する IP アドレス 0.0.0.0 はすべての IP アドレスを許可
enable_tls	ブール値	必須	true	HTTPS を API に 有効にする
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 のファイルパス
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 のファイルパス

TLS/SSL 設定

以下の TLS 設定を行います

```
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"
  }
```

启用 TLS

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

API 证书

生成证书和私钥

```
# 创建目录
mkdir -p priv/cert

# 生成私钥
openssl genrsa -out priv/cert/server.key 2048

# 生成证书请求
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"

# 生成证书 365 天
openssl x509 -req -days 365 -in priv/cert/server.csr \
  -signkey priv/cert/server.key -out priv/cert/server.crt

# 设置权限
chmod 600 priv/cert/server.key
chmod 644 priv/cert/server.crt
```

使用 Let's Encrypt CA 生成证书

API 证书

IP 地址

```
# 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

Option	Type	Required	Value	Description
port	Integer	Yes	443	HTTPS port
keyfile	String	Yes	priv/cert/omnitouch.pem	TLS keyfile
certfile	String	Yes	priv/cert/omnitouch.crt	TLS certfile

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"  
  ]
```

Reverse proxy

Use Nginx or Apache as reverse proxy

Nginx config

```

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;

    # 短信系统管理
    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;
    }

    # LiveView 与 WebSocket 支持
    location /live {
        proxy_pass http://sms_web;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

```

```
    proxy_read_timeout 86400;
  }
}
```

□□□□

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

□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□

□□□□

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

配置参数

参数	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 服务 Erlang 端口 4369 + 9100-9200

```
# 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 (MB)
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

設定例

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

パラメータ

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

設定

この設定は **Diameter Ro** を有効にするための RFC 4006 に基づく SMS-C の OCS 設定です。
この設定は **CC-Request-Type 4** を有効にするための OCS 設定です。
Diameter の設定は **Diameter Sh** の **:ro** の設定です。

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 設定 Diameter 拒否
  # 拒否
  # :allow - 許可
  # :reject - 拒否
  on_charging_error: :allow,
  # OCS 拒否 CCR
  # Service-Context-Id 拒否
  rating_profile: "sms@sms-c"
```

この設定は Diameter の **:ro** の設定です。IETF の **000**

```

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
      }
    ]
  }
}

```

| 項目                     | 型    | デフォルト | 説明                                                                               |
|------------------------|------|-------|----------------------------------------------------------------------------------|
| diameter_enabled       | bool | false | ディアンター HSS の dip 接続を有効にするかどうか。:on_net/:off_net のいずれかを指定する必要があります。false の場合は無効です。 |
| hss_cache_ttl_seconds  | int  | 7200  | ディアンター HSS のキャッシュ TTL (秒)。                                                       |
| mock_sh                | bool | false | ディアンター HSS のシミュレーションを有効にするかどうか。Diameter の接続をテストするためのオプションです。                     |
| mock_sh_on_net_numbers | list | []    | ディアンター HSS のシミュレーションで使用する MSISDN のリスト。                                           |

ディアンター HSS の接続を有効にするには、HSS の接続を有効にする必要があります。

# ENUM 項目

ENUM の DNS の E.164 の接続を有効にする

## ENUM 項目

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

## ENUM DNS 1 ENUM

```
config :sms_c,  
  enum_enabled: true,  
  enum_domains: ["e164.arpa", "e164.org"],  
  enum_dns_servers: [], # 1 DNS  
  enum_timeout: 5000 # 5 s
```

## ENUM DNS 2 ENUM

```
config :sms_c,  
  enum_enabled: true,  
  enum_domains: ["e164.internal.example.com", "e164.arpa"],  
  enum_dns_servers: [  
    {"10.0.1.53", 53}, # 1 DNS 1  
    {"8.8.8.8", 53}, # Google 1 DNS  
    {"1.1.1.1", 53} # Cloudflare DNS  
  ],  
  enum_timeout: 3000 # 3 s
```

## ENUM 1

1

```
config :sms_c,  
  enum_domains: [  
    "e164.internal.example.com", # 1  
    "e164.carrier.net", # 1  
    "e164.arpa" # 1  
  ]
```

## ENUM 2

2

```
enum_timeout: 2000 # 2 s
```



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+)$",
  calling_replace: "+\1",
  called_match: "^00(\d+)$",
  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

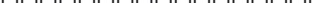
00000000

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

XXXXXX

XXXXXXXXXX??XXXXXX = XXXXXXXXX

1

 SMS 

## 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
}
]

```

メッセージ送信

```
# 簡易なWeb UI
config :sms_c,
  sms_routes: []
```

メッセージ送信の処理

メッセージ送信

メッセージ送信

```
# config/config.exs
config :sms_c,
  batch_insert_batch_size: 100,      # バッチサイズ
  batch_insert_flush_interval_ms: 100 # フlush間隔
```

メッセージ送信

| メッセージ | バッチサイズ | 間隔    | 送信速度           | 送信間隔        |
|-------|--------|-------|----------------|-------------|
| メッセージ | 200    | 200ms | ~5,000 msg/sec | メッセージ 200ms |
| メッセージ | 100    | 100ms | ~4,500 msg/sec | メッセージ 100ms |
| メッセージ | 50     | 20ms  | ~3,000 msg/sec | メッセージ 20ms  |
| メッセージ | 10     | 10ms  | ~1,500 msg/sec | メッセージ 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

```
# 数据库
config :sms_c, SmsC.Repo,
  pool_size: 20

# 数据库连接池
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

# 数据库 - 数据库 Diameter Ro 数据库
# 数据库 :charging + :diameter_ex 数据库
config :sms_c,
  default_charging_enabled: true
```

数据库

数据库

```
# 1 数据库
export DB_NAME=sms_c_tenant1
export NODE_NAME=sms_tenant1@node1.example.com
# 数据库 OCS 数据库 :charging rating_profile 数据库

# 2 数据库
export DB_NAME=sms_c_tenant2
export NODE_NAME=sms_tenant2@node1.example.com
```

数据库

数据库

```
# 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" # 配置短信节点
  ],
  smsc_node_name: "us-east-1a"
```

配置短信

配置短信节点

```
# 编译
mix compile

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

# 配置 Diameter Ro 和 OCS 消息
# :diameter.service_info(:omnimessage, :connections)

# 启动服务器
iex -S mix phx.server
```

配置短信

配置短信节点

| 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する
- API リスナーのインストール
- データベースの接続設定
- Web UI のインストール
- 設定ファイルの編集
- サービスの起動
- ログの確認
- ヘルプの表示
- 再起動

10. 配置数据库

数据库配置

| 配置项     | 配置值        | 说明                           |
|---------|------------|------------------------------|
| 数据库名称   | 数据库名称      | 数据库名称                        |
| 数据库用户   | 用户/密码      | DB_* 用户                      |
| API 端口  | 用户/IP 端口   | API_PORT  listen_ip          |
| 数据库连接   | Cookie 配置  | ERLANG_COOKIE 4369-9100-9200 |
| 数据库     | OCS/DRA 配置 | Diameter Ro 配置               |
| ENUM 配置 | DNS 配置     | DNS 配置                       |
| 配置      | 配置         | 配置                           |
| 配置      | 配置         | sms_routes Web UI            |

数据库配置

数据库Mnesia

配置

Mnesia 配置

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

  # []
  retention_check_interval_minutes: 60
```

111

- 24-72 小时
- 4-8 小时
- 12-24 小时

□□□□□

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

**CDR**

□□□□□□□□□□ CDR □□□□□□□□ Ecto □□□□□□□□□□□□□□□□□□

```
config :sms_c,  
  # 00/00 CDR 00000000false  
  cdr_enabled: true
```

**CDR** ☐ ☐ ☐ ☐ ☐

- ID号码/号码
- / 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 データベース  
  delete_message_body_after_delivery: false,  
  
  # Web UI  
  hide_message_body_in_ui: false,  
  
  # CSV  
  hide_message_body_in_export: false
```

設定

| 項目                                                    | 説明            |
|-------------------------------------------------------|---------------|
| <code>delete_message_body_after_delivery: true</code> | Mnesia データベース |
| <code>hide_message_body_in_ui: true</code>            |               |
| <code>hide_message_body_in_export: true</code>        |               |

実行

実行コマンド

```
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

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

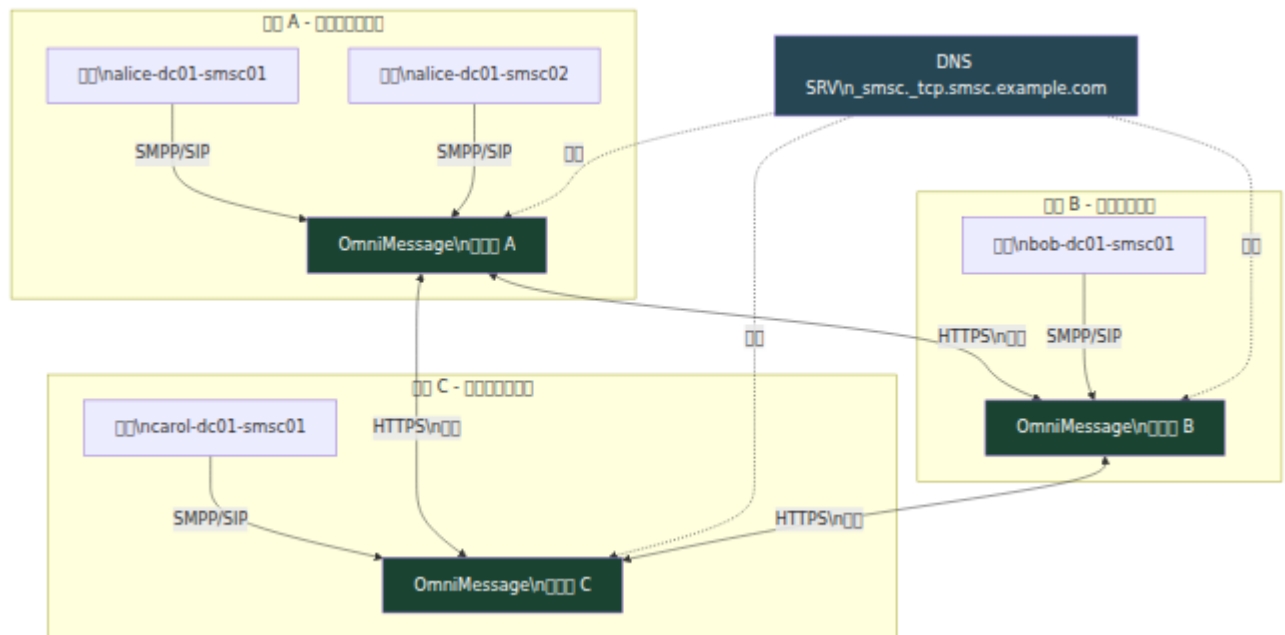
← □□□□□□ | □□□□ | □□□□

OmniMessage 可以透过 **HTTP** 或 **HTTPS** 连接到远端服务器 - 透过 DNS SRV 记录或 HTTPS 记录。

**FederationPoller**

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

|             |                               |
|-------------|-------------------------------|
| IP          | IP                            |
| Port        | Port WAN Port (Internet Port) |
| Protocol    | Protocol - TCP/UDP            |
| Source      | Source (IP Address)           |
| Destination | Destination (IP Address)      |
| Service     | Service HTTPS or TLS          |
| Port        | Port HTTPS Port (8443)        |
| Time        | 5-20 seconds                  |
| Result      | Result - Success              |

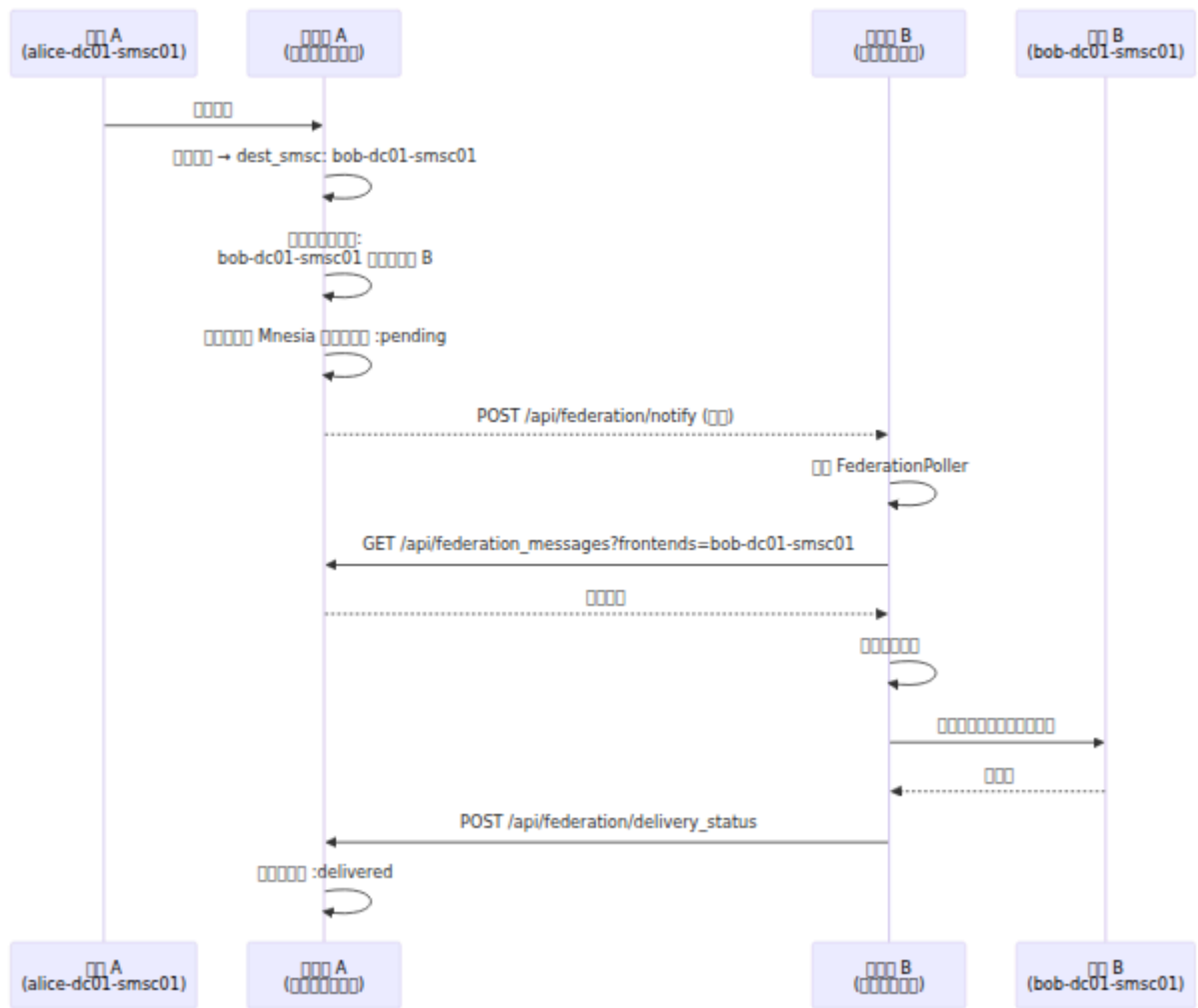


- 0000000000000000000000000000000000000000000000000000000
- 0000000000000000000000000000000000
- 000000000000000000000000
- 0000000000000000000000000000000000
- 000DNS SRV 000000000000000000 DNS 00000

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

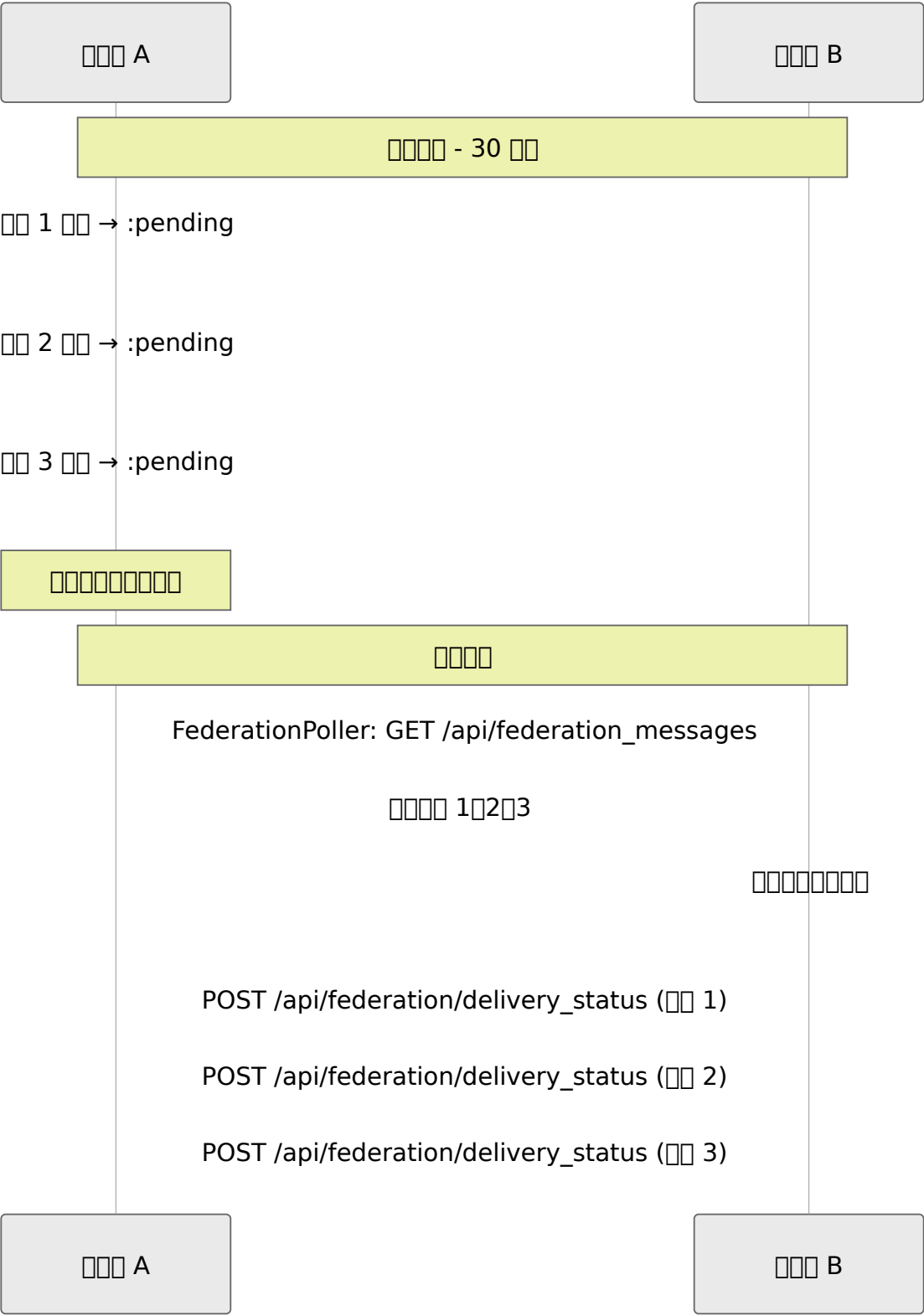
1. `Mnesia` `:pending`
2.
3. `FederationPoller` `GET /api/federation_messages`
4.
5.

```
FederationPoller :pending 5
```



消息 - 消息ID

消息ID - FederationPoller 消息 - 消息ID

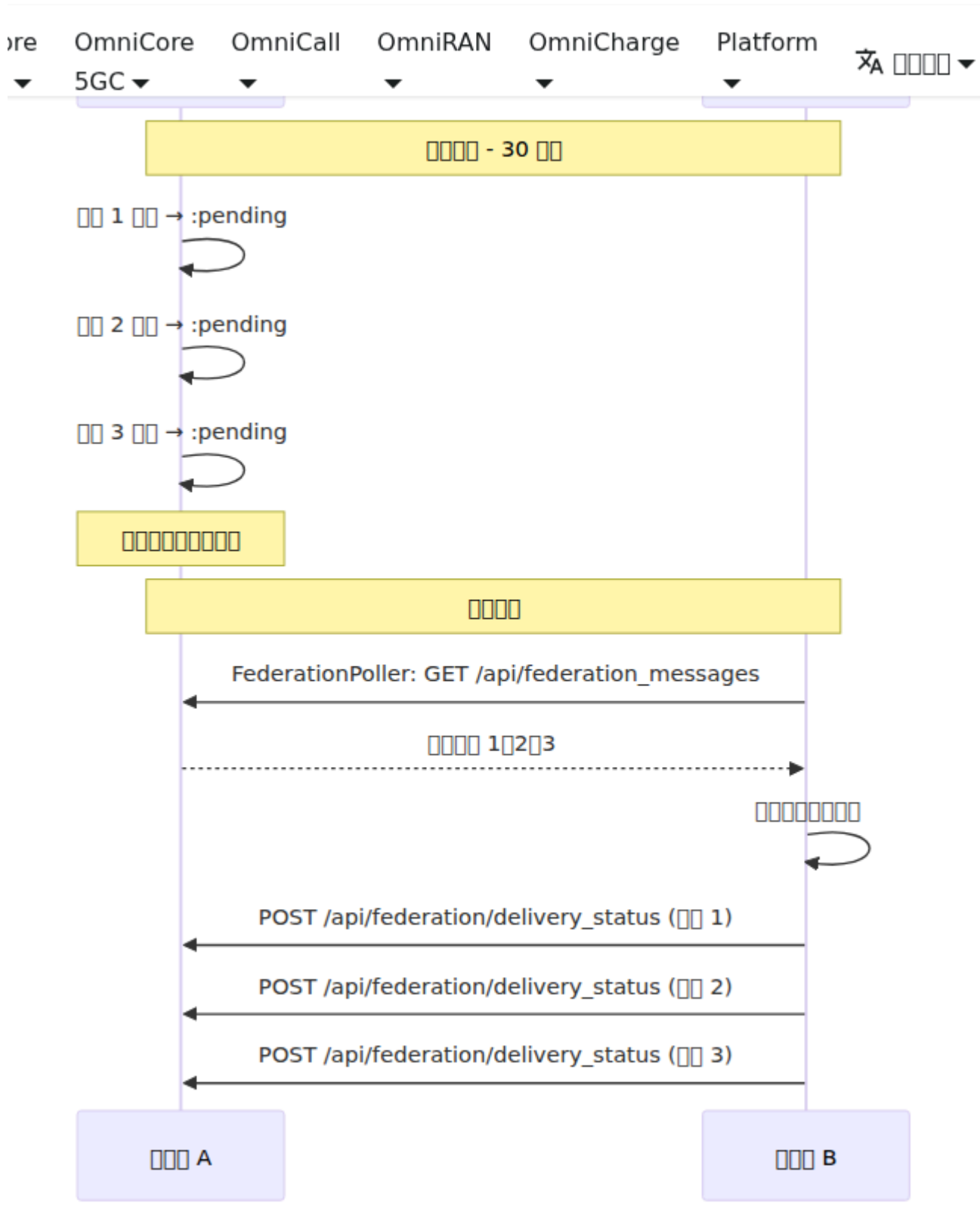


Message 1

Message 2

```
sequenceDiagram
    participant Carol as Carol (C)
    participant Bob as Bob (B)
    Carol->>Bob: POST /api/federation/notify
    Bob-->>Carol: 
    Carol->>Bob: GET /api/federation/messages?frontends=carol-dc01-sm-sc01
    Bob-->>Carol: 
    Carol->>Bob: POST /api/federation/delivery_status
    Bob-->>Carol: 
    Carol->>Bob: POST /api/federation/notify
    Bob-->>Carol: 
    Carol->>Bob: GET /api/federation/messages?frontends=bob-dc01-sm-sc01
    Bob-->>Carol: 
    Carol->>Bob: POST /api/federation/delivery_status
    Bob-->>Carol: 
```





Sequence diagram illustrating the interaction between the 5GC and the Platform.



## 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: "_smc._tcp.smc.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
  http_timeout_ms: 5_000,

  # DNS SRV 记录 URL 对应的 API 端口
  api_port: 8443,

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

配置

| 配置                                     | 数据类型 | 是否必填 | 默认值                | 说明                                                                                        |
|----------------------------------------|------|------|--------------------|-------------------------------------------------------------------------------------------|
| <code>enabled</code>                   | 布尔   | 否    | <code>false</code> | 是否启用 <code>false</code> 则不启用<br>静态对等节点                                                    |
| <code>dns_srv_domain</code>            | 字符串  | 否    | <code>""</code>    | 静态对等节点的 DNS SRV 记录<br>所属域 <code>static_peers</code>                                       |
| <code>dns_poll_interval_ms</code>      | 整数   | 否    | <code>30000</code> | DNS SRV 记录刷新间隔<br>静态对等节点的 DNS 记录                                                          |
| <code>health_check_interval_ms</code>  | 整数   | 否    | <code>10000</code> | 健康检查间隔<br>静态对等节点                                                                          |
| <code>registry_sync_interval_ms</code> | 整数   | 否    | <code>15000</code> | 注册表同步间隔<br>静态对等节点                                                                         |
| <code>poll_interval_ms</code>          | 整数   | 否    | <code>5000</code>  | FederationPoller 刷新间隔<br>静态对等节点                                                           |
| <code>http_timeout_ms</code>           | 整数   | 否    | <code>5000</code>  | HTTP 请求超时时间<br>静态对等节点                                                                     |
| <code>api_port</code>                  | 整数   | 否    | <code>8443</code>  | 静态对等节点的 DNS SRV 记录 URL<br>所属 API 名称 <code>config</code><br><code>:api_ex</code> 所属 API 名称 |
| <code>static_peers</code>              | 数组   | 否    | <code>[]</code>    | 静态对等节点的 DNS SRV 记录<br>静态对等节点的 <code>host</code> 和<br><code>port</code> 列表                 |

# 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

```
; 示例 DNS 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.

; 示例 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      | 状态 | 描述                |
|------|---------|----|-------------------|
| 8443 | TCP/TLS | 监听 | HTTPS 端口          |
| 53   | UDP/TCP | 监听 | DNS SRV 记录 DNS 记录 |

配置防火墙

```
# 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 the X-Federation-Node header

## Example

### 1. DNS SRV

Use the DNS SRV record to find the service

```
# Example A - (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 会定期轮询该地址。

## 配置 3 个节点

配置中指定了 3 个节点。

```
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 秒。

## 配置

配置 Prometheus 的 9568 端口。

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

- `origin_node` - 节点名称

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

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

同僚ノードの監視

| 項目            | 監視内容                                             | 閾値              |
|---------------|--------------------------------------------------|-----------------|
| 同僚ノードのステータス   | <code>GET /api/federation/status</code> の Web UI | 同僚ノードのステータスが正常  |
| 同僚ノードの識別子     | <code>GET /api/federation/identity</code> のレスポンス | 同僚ノードの識別子が 5 以上 |
| 同僚ノードのヘルスチェック | 同僚ノードからのヘルスチェック結果                                | > 2 同僚ノードが正常    |

同僚ノードの監視

同僚ノードの監視

例: 同僚ノードのステータスを確認する `cluster_status` のメトリクス

例:

- DNS SRV レコードを確認
- DNS レコードを確認
- SRV レコードを確認
- `enabled: false` の状態を確認

步骤:

1. 使用 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` 静态对等节点/地址

## 配置

配置: 配置静态对等节点“静态对等节点”

步骤:

- 配置静态对等节点
- 配置静态对等节点 8443 端口
- TLS 配置
- 配置静态对等节点

步骤:

1. 配置静态对等节点 `curl -k https://<peer-host>:8443/api/federation/identity`
2. 配置静态对等节点 IP 为 8443 端口
3. 配置静态对等节点 TLS 配置
4. 配置静态对等节点

## 配置

配置: 配置静态对等节点 `:pending` 静态对等节点 0

步骤:

- 配置静态对等节点 FederationPoller 静态对等节点
- FederationPoller 配置
- 配置静态对等节点静态对等节点

步骤:

1. 安装依赖包
2. 运行 FederationPoller 服务并查看日志输出“Federation poller started”
3. 使用 curl 命令 GET /api/federation/status - 查看状态
4. 使用 curl 命令 federation\_messages 并查看输出  
`curl -k 'https://<origin>: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 ##### :any#####  
##### 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

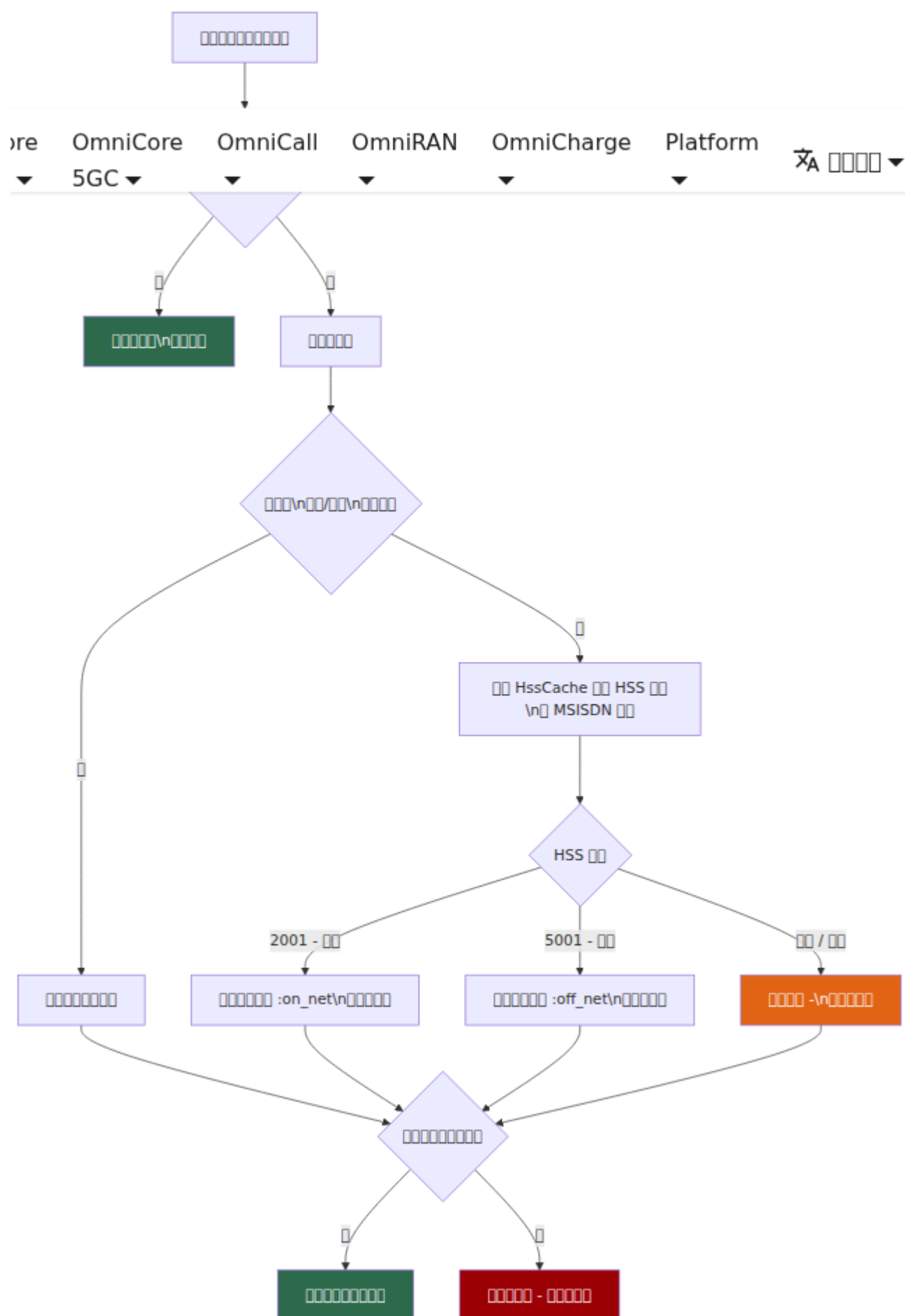
ENUM - HSS ENUM HSS (Sh) “ENUM” HSS “ENUM”/ENUM  
ENUM (DNS NAPTR) “ENUM”/ENUM HSS  
Sh ENUM SMS → ENUM

□ □ □ □

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

**HSS** □□□□□□□□

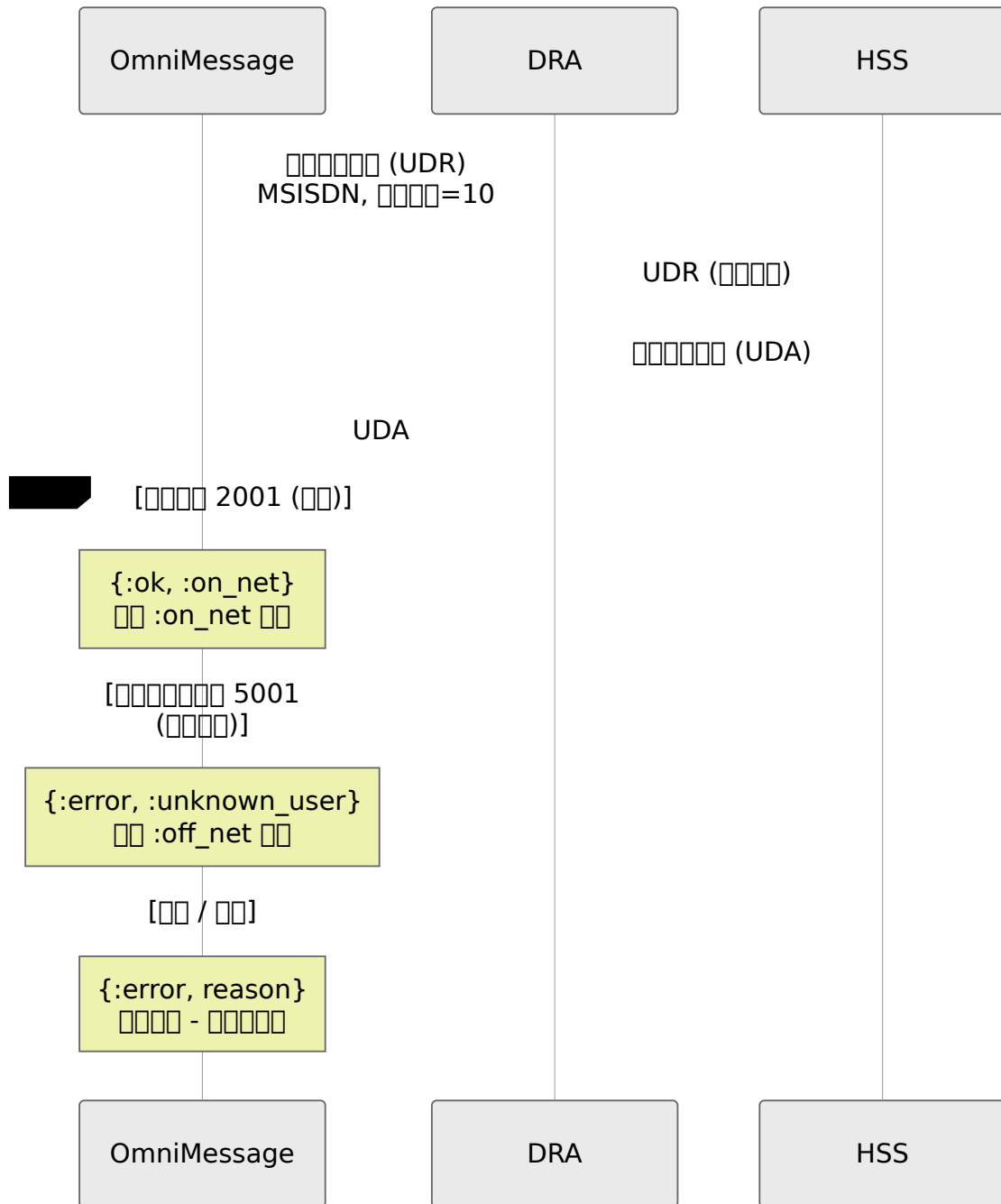
OmniMessage □□□□□□□□







# Diameter Sh 消息



# UDR

| AVP                            |                               |                         |
|--------------------------------|-------------------------------|-------------------------|
| Session-Id                     |                               |                         |
| Auth-Session-State             | 1 (NO_STATE_MAINTAINED)       |                         |
| Data-Reference                 | 10                            |                         |
| Origin-Host                    | +                             | OmniMessage<br>Diameter |
| Origin-Realm                   |                               | OmniMessage<br>Diameter |
| Destination-Realm              |                               | HSS DRA                 |
| User-Identity                  | TBCD MSISDN                   |                         |
| Server-Name                    | "OmniMessage"                 |                         |
| Vendor-Specific-Application-Id | Auth: 16777217, Vendor: 10415 | 3GPP TS 29.329<br>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<br>:off_net |
| 5001               | 項目項目項目<br>項目 | {:error,<br>:unknown_user} | 項目 - 項目 :off_net項目項目<br>:on_net |
| 項目 / 項目 / 項目<br>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       | デフォルト | 説明                                                                         |
|------------------------|---------|-------|----------------------------------------------------------------------------|
| diameter_enabled       | boolean | false | HSS 接続を有効にするかどうか。<br>false の場合は、 <code>:on_net/:off_net</code> のいずれも機能しない。 |
| hss_cache_ttl_seconds  | integer | 7200  | HSS 接続の TTL。<br><code>runtime.exec</code> の YAML ファイルで指定可能。                |
| mock_sh                | boolean | false | HSS 接続をシミュレートする。<br>Diameter 接続をシミュレートする。                                  |
| mock_sh_on_net_numbers | array   | []    | <code>mock_sh</code> が true の場合、<br>MSISDN のリスト。                           |

# 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<br>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<br>: round-robin |

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

アプリケーション

applications   3GPP Diameter HSS Sh

| 項目                              | 型      | 必須 | 注 | 説明                                                 |
|---------------------------------|--------|----|---|----------------------------------------------------|
| application_name                | String | 必須 | - | HSS アプリケーション名<br>:sh                               |
| application_dictionary          | String | 必須 | - | Erlang Diameter アプリケーション名<br>: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<br>項目 FQDN 項目 項目 項目<br>項目 項目 項目                                   |
| ip                  | 項目       | 項目 | -             | 項目 TCP 項目 IP 項目                                                                  |
| port                | 項目<br>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<br>項目 :diameter_tcp<br>項目<br>:diameter_sctp 項目                                |
| initiate_connection | 項目       | 項目 | true          | 項目 true 項目<br>OmniMessage 項目<br>項目 TCP 項目<br>項目 false 項目<br>OmniMessage 項目<br>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

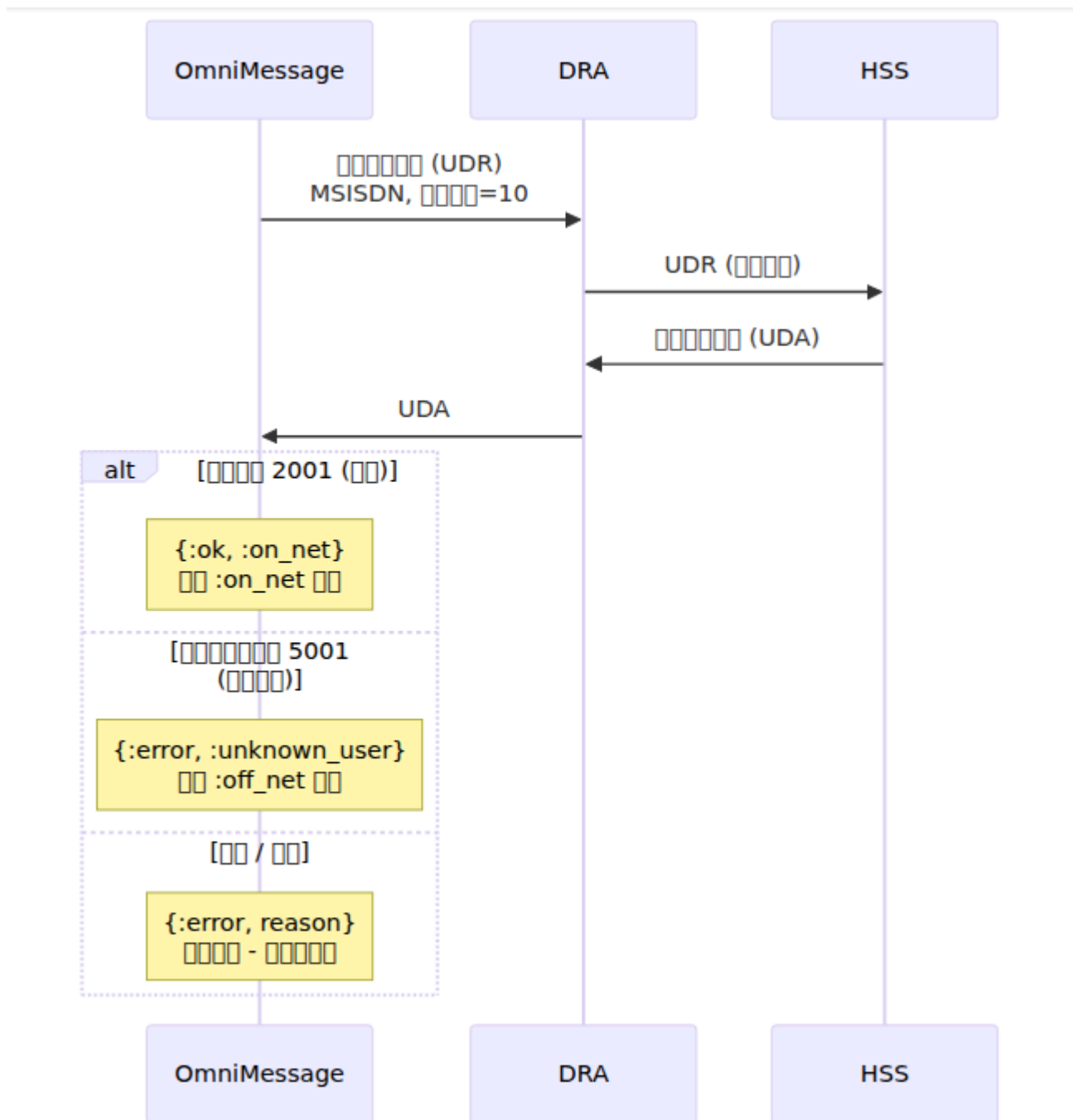
- **TTL** `hss_cache_ttl_seconds` 7200 = 2 小时
- 配置 Diameter 的 MSISDN 配置
- 配置 Diameter 的 MSISDN 配置
- 配置 Diameter 的 `{:ok, :on_net}` 配置 `{:error, :unknown_user}` 配置 Diameter 配置 `nil` 配置 - 配置 Diameter 配置

## 配置

`:saf` 配置 配置 HSS 配置 `action: :saf` 配置 `: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



000000

0000

OmniMessage 000000 Diameter Sh 000

- 000Diameter 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`

## API

HSS `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`

## API

`expires` HSS

## /

| 項目                  | 値                         | 単位     | 説明                         |
|---------------------|---------------------------|--------|----------------------------|
| originating_network | :any   :on_net   :off_net | 桁<br>桁 | 発信 MSISDN の HSS 番号<br>範囲指定 |
| terminating_network | :any   :on_net   :off_net | 桁<br>桁 | 着信 MSISDN の HSS 番号<br>範囲指定 |

範囲指定 :any の HSS 番号 :on\_net 2001 :off\_net 5001  
の範囲指定

## 配置

```
config :sms_c, :sms_routes, [
  # 本地网关 - 本地网关
  %{
    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: "本地网关 - 本地网关",
    originating_network: :any,
    terminating_network: :on_net,
    enabled: true
  },

  # 本地网关 - 外部网关
  %{
    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: "本地网关 - 外部网关",
    originating_network: :any,
    terminating_network: :any,
    enabled: true
  }
]
```

## 配置说明

1. 本地网关 - 本地网关
2. 本地网关 - 外部网关

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. `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 `: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` - 接続済み

タグ:

メトリック: `omnimessage`

注: 接続数が増加すると、接続数が増加する。 "NOLICENCE" 接続数が増加する。

接続数:

- 接続数が増加する
- 接続数 ( `dest_smsc` ) が増加する **"NOLICENCE"**
- 接続数が増加する
- UI 接続数が増加する
- 接続数が増加する

注:

```
- alert: SMS_C_License_Invalid
  expr: sms_c_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "SMS-C ██████████"
    description: "██████████ - ████████ NOLICENCE"
```

## ❏ Prometheus ❏:

```
# 0000000000000000
sms_c_license_status == 1

# 0000000000000000
sms_c_license_status == 0

# 000000 NOLICENCE 00000000000000000000
sms_c_routing_route_matched_count{dest_smsc="NOLICENCE"}
```

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

## sms\_c\_message\_received\_count

□□: Counter

□□: SMS-C □□□□□□□□□□□□□□□□

□□:

- `source_smsc`: 短信中心地址 (SMSC 地址)
- `source_type`: 网络类型 (ims, circuit\_switched, smpp)
- `message_type`: 消息类型 (sms, mms)

00 : 0000000000000000000000000000000000000000000000000

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/短信发送成功/短信发送成功

短信: 短信发送失败/短信发送成功/短信发送成功/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/短信发送成功/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/短信发送成功/短信发送成功

短信: 短信 p95 或 p99 短信 SLA 短信发送成功

---

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

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

□□: Counter

□□ : □□□□□□□    □□□□□□□□□□□□□□

11

- `route_id`: 0000000000
- `dest_smsc`: 00000000 SMSC
- `priority`: 0000000000

:

:

## sms\_c\_routing\_failed\_count

□□: Counter

□□: □□□□□□□□□□□□□□□□

11

- reason: no\_route\_found, validation\_failed

:

:

## sms\_c\_routing\_action\_count

□□: Counter

□□ : □□□□□□□□□□

短信:

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

短信: 短信发送失败原因

短信: 短信发送失败原因

---

## sms\_c\_routing\_stop\_duration

短信: Histogram

短信: 短信发送失败原因

短信: 短信

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

短信:

- `dest_smsc`: 短信接收方 SMSC

短信: 短信发送失败原因

短信: 短信发送失败原因 p95 > 50ms

---

## ENUM/NAPTR 短信

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

短信: Counter

短信: 短信接收方 ENUM 短信接收方 DNS 短信

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 □□□□□□

□□:



短信: 短信发送失败次数统计

短信:

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

短信: 短信发送失败次数统计

短信: 短信发送失败次数统计

---

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

短信: Counter

短信: 短信发送失败次数统计

短信:

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

短信: 短信发送失败次数统计

短信: 短信发送失败次数统计

---

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

短信: Histogram

短信: 短信发送成功次数统计

短信: 1, 2, 3, 5, 10

短信:

- `dest_smsc`: 短信 SMSC 地址

短信: 短信发送成功次数统计

例: 電話番号と 2 桁の送信回数

---

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

例: Histogram

例: 電話番号

例: 1, 2, 3, 5, 10

例:

- `dest_smsc`: 宛 SMS 番号

例: 電話番号

---

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

例: Histogram

例: 電話番号と SMS-C 送信時間差 (ms) のヒストグラム

例: 例

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

例:

- `source_smsc`: 送信元 SMS 番号
- `dest_smsc`: 宛 SMS 番号
- `delivery_attempts`: 送信回数 (0 = 未送信)

例: 宛 SMS 番号と SMS-C 送信時間差 (ms) のヒストグラム

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])))
)

# Top 10 SMSCs - Latency 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

**00:** 00000000000000000000000000000000 SMSC 0000

```
sms_c_delivery_time_delta_delta_ms -> 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: 東京証券取引所 SLA 平均値 > 300 日

□ □ □ □

0000 **Diameter Ro** 0000000000 account 0000000000 MSISDN000000000000  
000000 reason: :no\_credit00 OCS 0000/0000000000 charging\_failed 0000  
0 :no\_credit 0000000000 :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`: 理由

単位: 回

単位: 回

---

## `sms_c_charging_succeeded_duration`

概要: Histogram

単位: 回

単位: ms

バケット: 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

指标: 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 00000000000000

□□□□:

```
# 000000
rate(sms_c_location_registered_count[1m])

# IMS 00 IMS 0000
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

□□: □□□□□□□□□□□□ IMS □□□□□□□□□□□□□□□□□□□ SMSC □□□□□□□□

□□ : □□□□□□□□

- 
- 
- 

## ❏ Prometheus ❏:

```

# 总和
sum(sms_c_location_active_registrations_count)

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

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

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

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

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

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

```

## HTTP API 接口

### phoenix\_endpoint\_stop\_duration

接口: Distribution (Histogram)

接口: HTTP 接口   接口 

接口:

- **method**: HTTP 接口 **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

例:

```
# 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

例: Counter

例: HTTP 異常率/異常率

例:

- route: API 異常率
- kind: error, exit, throw

例: 異常率/異常率

例: 異常率/異常率

例:

```
# 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 指标

SMS-C 通过 **Diameter Sh** 接口 3GPP TS 29.329 与 HSS 交互，用于鉴权、计费、位置更新等。  
Diameter 指标

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

类型: Gauge

范围: Diameter 对等体状态 1 到 0

指标:

- `origin_host`: Diameter Origin-Host
- `ip`: IP 地址

单位:

```
# 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: 短信联邦健康检查消息数量

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% 以上のメモリ使用率

---

## vm\_memory\_processes

⚠: Gauge

⚠: Erlang プロセスのメモリ使用率

⚠: 80% 以上のメモリ使用率

⚠: 80% 以上のメモリ使用率

---

## 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

⚠:  $10 \times \text{CPU}$  の実行キューの長さ

---

## vm\_system\_counts\_process\_count

⚠: Gauge

⚠: VM のプロセス数

⚠: 262,144 プロセス

⚠: 262,144 プロセス

---

## ENUM 表

表 10 ENUM 表

- ENUM 表
- Mnesia 表
- ENUM 表

ENUM 表

## ENUM 表

### ENUM 表

ENUM 表 SMSC 表

表:  $(\text{sms\_c\_delivery\_succeeded\_count}) / (\text{sms\_c\_delivery\_queued\_count})$

表: 表 > 95% 表

---

### ENUM 表

ENUM 表

表:

- $\text{sms\_c\_message\_processing\_stop\_duration}$  (表)
- $\text{sms\_c\_delivery\_time\_delta\_delta\_ms}$  (表)

表: 表

---

## ENUM 表

表 ENUM 表



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

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)

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

111

101010101010

```
00: sms_c_delivery_succeeded_attempt_count 00000000
```

[illegible]

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> > 10                        | 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> > 100                         | Warning  | 消息队列积压，请检查消息队列配置。      |
| 消息队列老化    | <code>queue_oldest_message_age_seconds</code> > 300           | Critical | 消息队列老化，请检查消息队列配置。      |
| 消息投递失败    | <code>delivery_failed_count</code> > 10                       | High     | 消息投递失败，请检查消息队列配置。      |
| 消息投递死信    | <code>delivery_dead_letter_count</code> > 0                   | High     | 消息投递死信，请检查消息队列配置。      |
| ENUM 查找失败 | <code>enum_lookup_stop_duration</code> p95 > 5000ms           | Warning  | DNS 查找失败，请检查 DNS 配置。   |
| ENUM 可用性  | ENUM 可用性 < 0.7                                                | Warning  | ENUM 可用性低，请检查 ENUM 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> > 10                       | High     | 消息充电失败，请检查消息队列配置。      |
| 消息处理超时    | <code>message_processing_stop_duration</code> p95 > 1000ms    | Warning  | 消息处理超时，请检查消息队列配置。      |

## □□□□□

### □□□□□

□□: □□□□□□□□

□□:

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

### □□□□□

□□: □□□□□□

□□:

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

### □□□□□

□□: □□□□□□□□

□□:

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

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

[illegible]

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

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` 秒 - 短信处理时间
  2. `sms_c_routing_stop_duration` - 短信路由时间
  3. `sms_c_enum_lookup_stop_duration` - ENUM 短信处理时间
  4. `sms_c_charging_succeeded_duration` - 短信计费时间
  5. 短信计费失败时间
- 

## 短信队列大小

指标名称

1. `sms_c_queue_size_pending` 秒 - 短信队列大小
  2. `sms_c_delivery_attempted_count` - 短信尝试发送次数
  3. `sms_c_delivery_failed_count` - 短信发送失败次数
  4. `sms_c_delivery_time_delta_delta_ms` - 短信发送时间差
  5. `dest_smsc` 短信发送目的地
- 

# Prometheus 配置

## 配置项

短信接收速率5分钟:

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

短信接收速率1小时:

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

增加计数:

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

按源类型求和:

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

按 SMSC 求和:

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

成功率

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分間のp95:

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

直近5分間のp99:

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

キュー

キューのサイズ:

```
sms_c_queue_size_pending
```

失敗したキューのサイズ:

```
sms_c_queue_size_failed
```

キューの最古メッセージの年齢:

```
sms_c_queue_oldest_message_age_seconds / 60
```

キューのサイズの変化率:

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

キューの長さ:

```
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:

```
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 **p95**:

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

指标 **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: `{{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%
- 指标:
  - 指标: < 90
  - 指标: 90-95
  - 指标: > 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
- 指标:
  - 指标: < 5
  - 指标: 5-10
  - 指标: > 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
- 指标: 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
- 指标: > 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 轴范围

## 5. 指标范围

- 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

- 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

- 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
- 单位: 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 平均
- 单位: 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%

## 可用性

### 可用性

可用性:

```
alert: RoutingFailuresDetected
expr: increase(sms_c_routing_failed_count[5m]) > 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "{ $value } 5 分間"
  description: "SMS 送信失敗"
```

可用性:

```
alert: MessageQueueBacklog
expr: sms_c_queue_size_pending > 10000
for: 5m
labels:
  severity: critical
annotations:
  summary: "SMS 送信キューサイズ { $value } 以上"
  description: "SMS 送信キューサイズ超過"
```

可用性:

```
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: "95th 短信发送成功率 2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: "95th 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

- 監視可能な Prometheus 関数 `rate()` と `increase()`
- Gauge 監視
- Histogram 監視 (p50, p95, p99)
- Prometheus 監視
- 5m, 1h, 24h+ 監視
- Prometheus 監視
- Grafana 監視 (dest\_smsc, source\_smsc)

# MMS 簡介 (MMSc)

← 消息中心 | 消息 | **API 簡介**

## 簡介

- 簡介
- 消息
- 消息中心
- 消息
- **MMS 簡介**
- 消息/消息
- 消息
- 消息
- 消息
- 消息
- 消息

## 簡介

OmniMessage 消息中心 **MMS 簡介 (MMSc)** 消息中心消息中心消息中心 3GPP TS 23.140 消息中心 MMS 消息中心 **MM1** 消息中心 (HTTP/WAP) 消息中心 **MM4** 消息中心 (SMTP)

MMSc 消息中心 OmniMessage 消息中心消息中心消息中心 Web 消息中心消息中心消息中心 MMS 消息中心 MMSc

1. 消息 MMS 消息中心消息中心消息中心 Mnesia 消息
2. 消息 **MMS** 消息中心消息中心 SMS 消息中心消息中心消息中心 SMS 消息中心消息中心消息中心
3. 消息中心消息中心 URL 消息中心消息中心 MMS

消息中心消息中心 SMS 消息中心 MMS 消息中心 SMS 消息中心消息中心消息中心 HTTP 消息

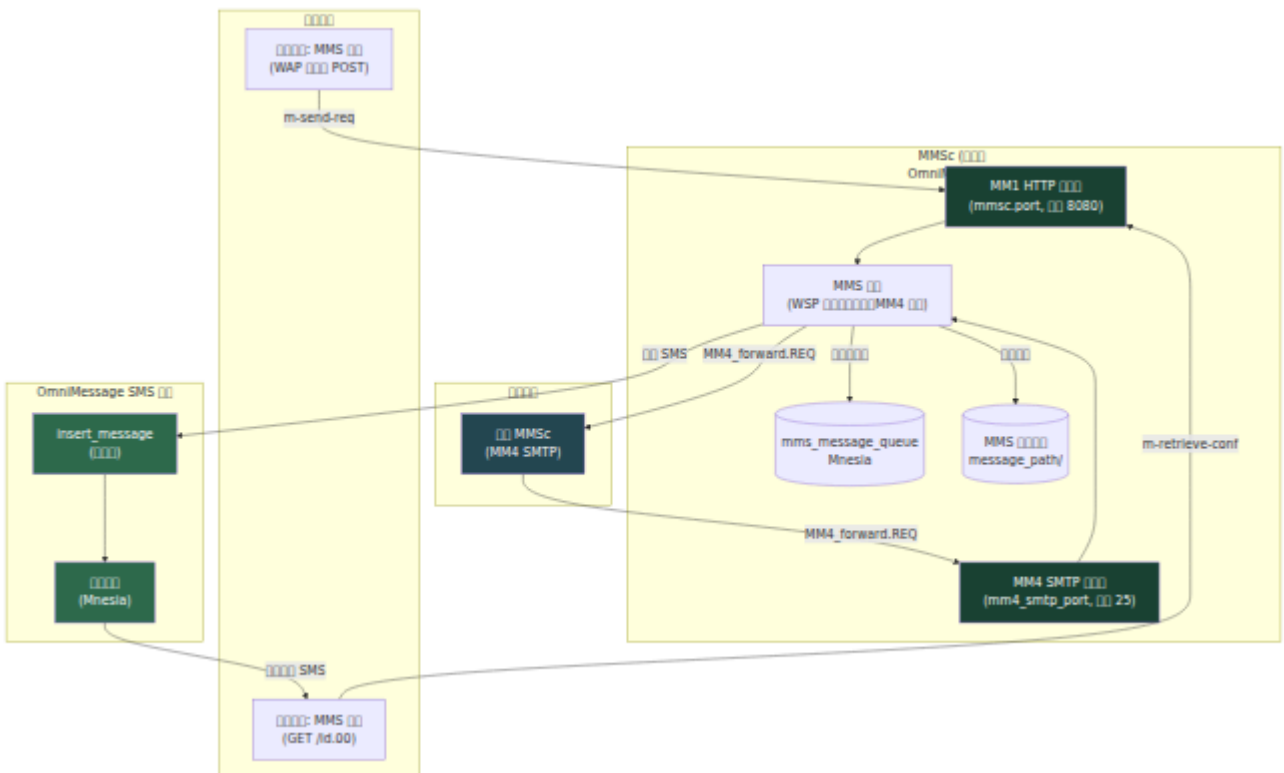


|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

| 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         |

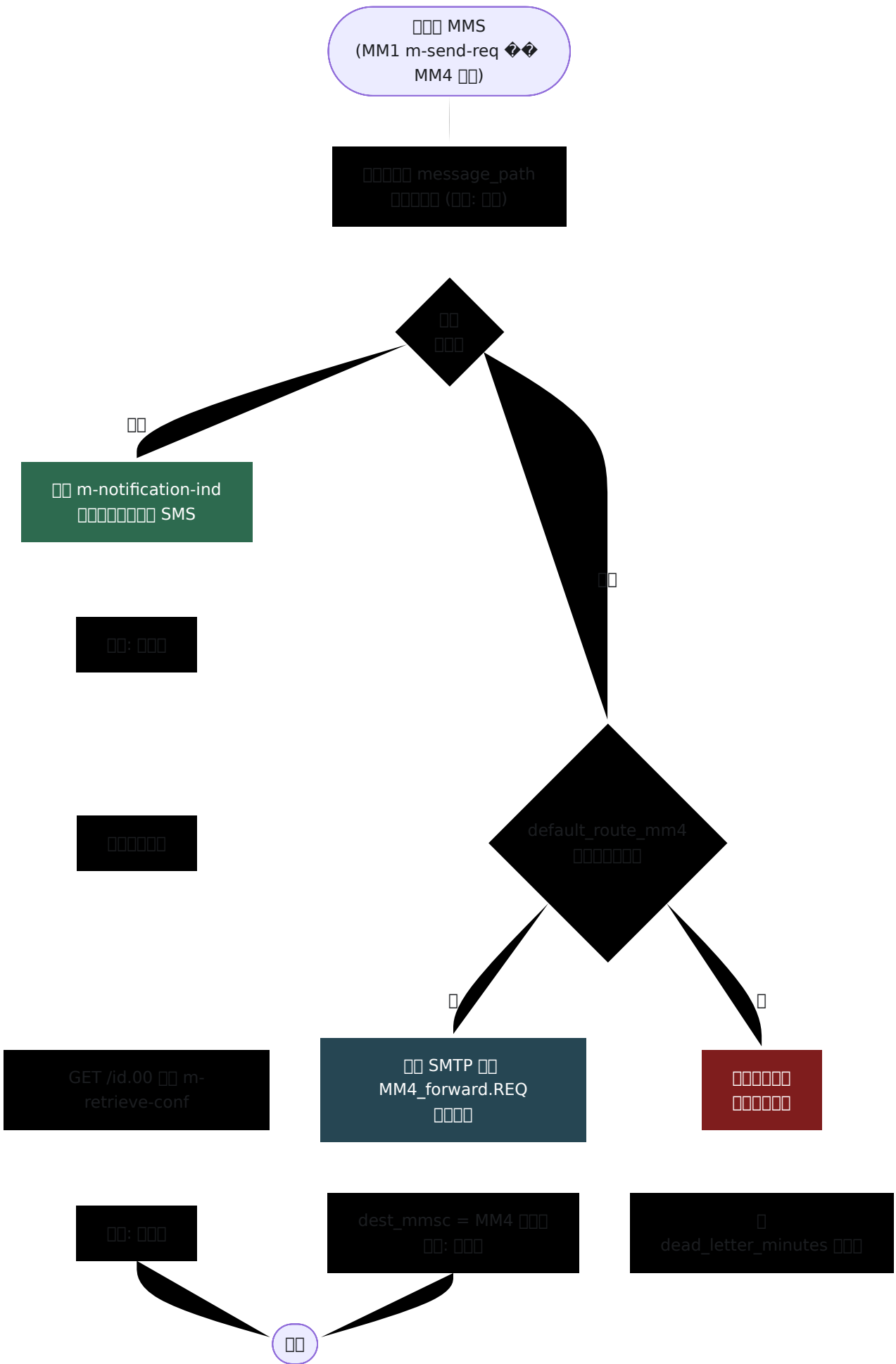
11

MM1 (利用 HTTP) と MM4 (利用 SMTP) を通じて OmniMessage API を利用して MMS を送信する Mnesia を利用した SMS 送信機能 MMSc を利用して MMS を送信する機能を実装する。



□□□□□□

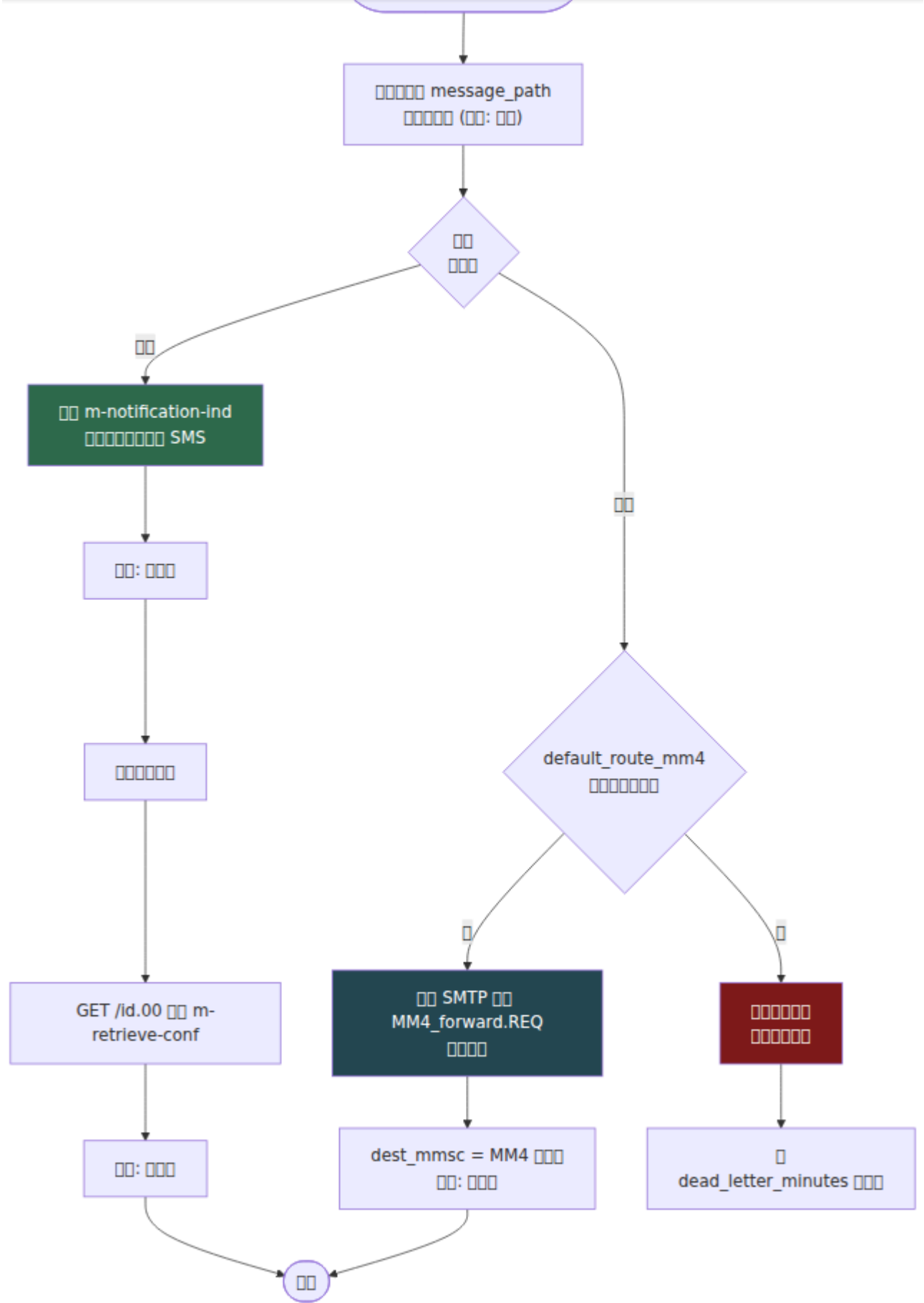
□□□□ MMS□□□□□□ MM1□□□□□□□□ MM4□□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□



□□□□

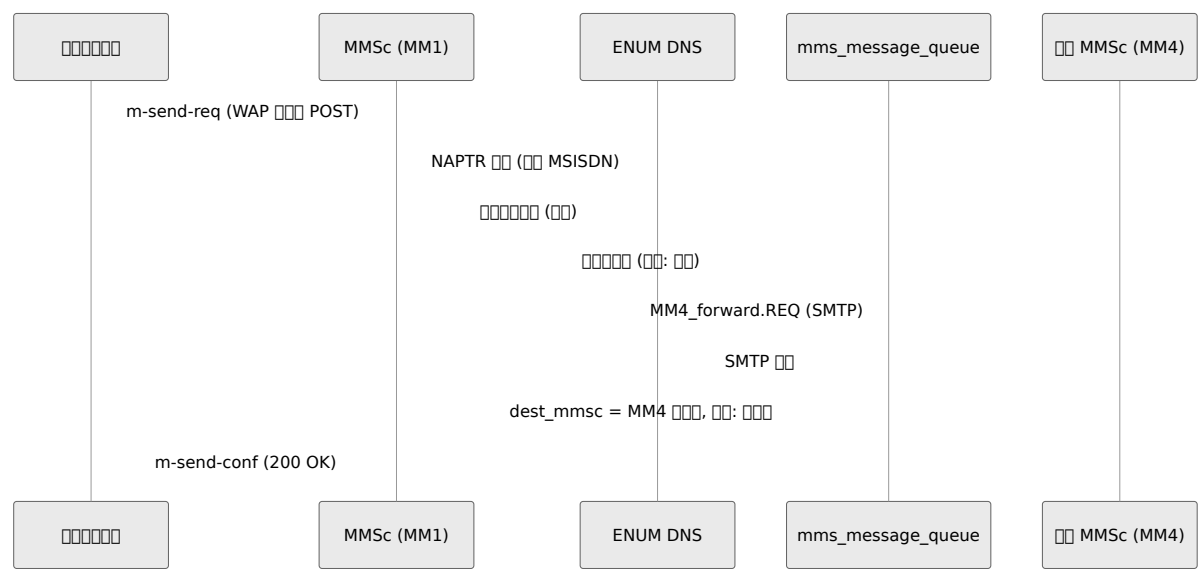
□□□□ (MM1)

□□□□□□□□□□ MMS□



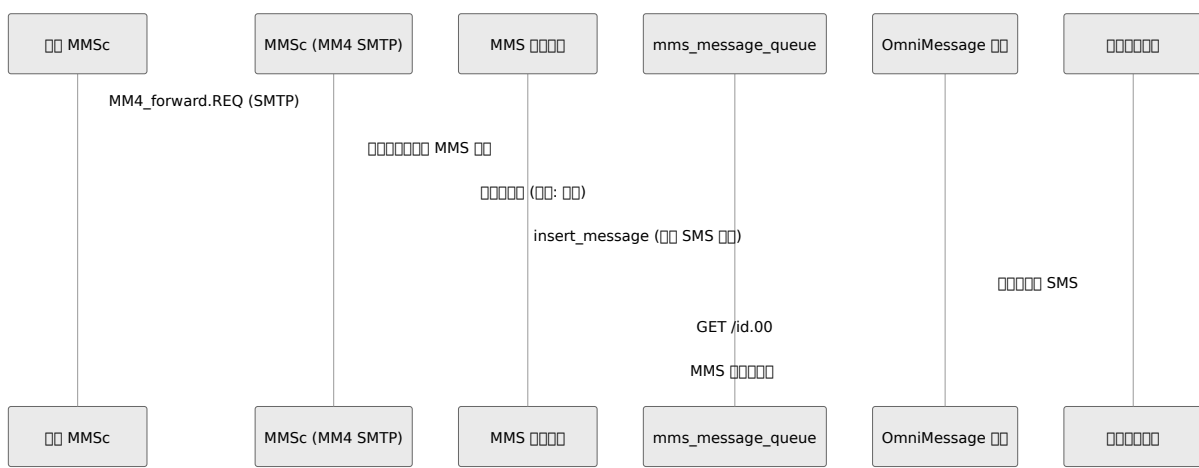
# MM4

MMS



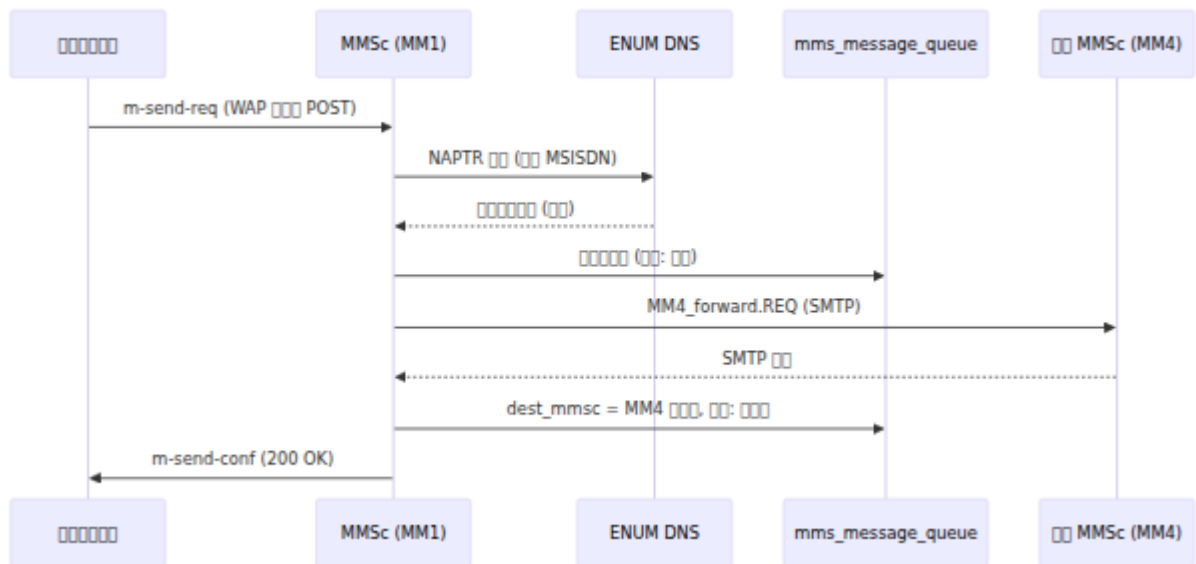
# MM4 ( )

MMS



# MMS

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



Sequence diagram

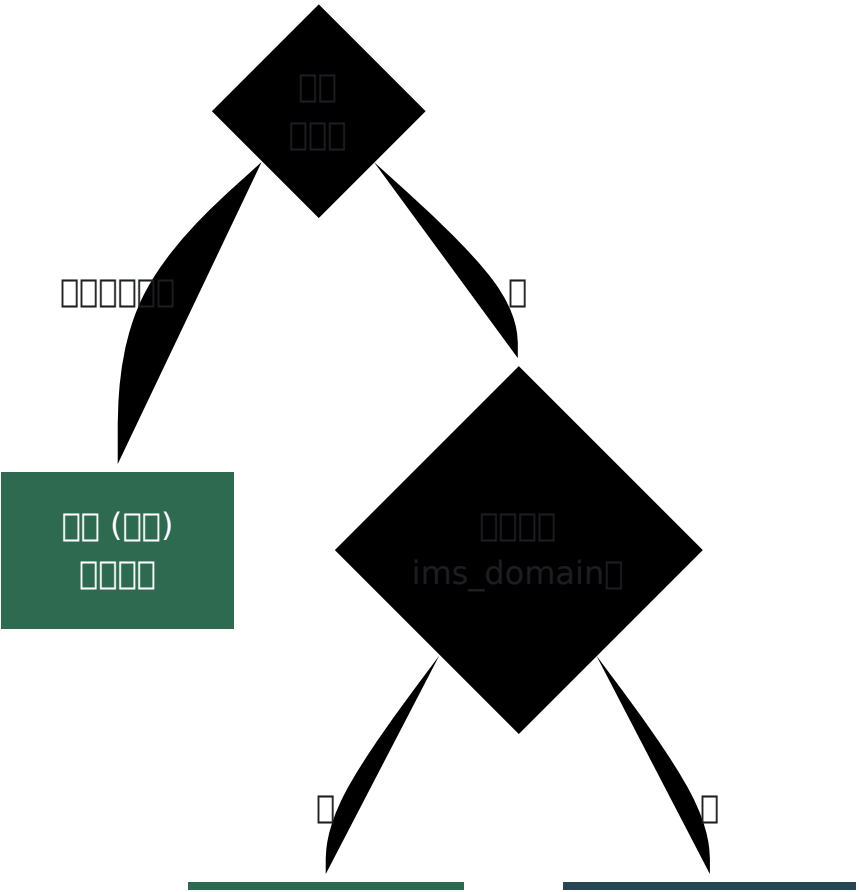
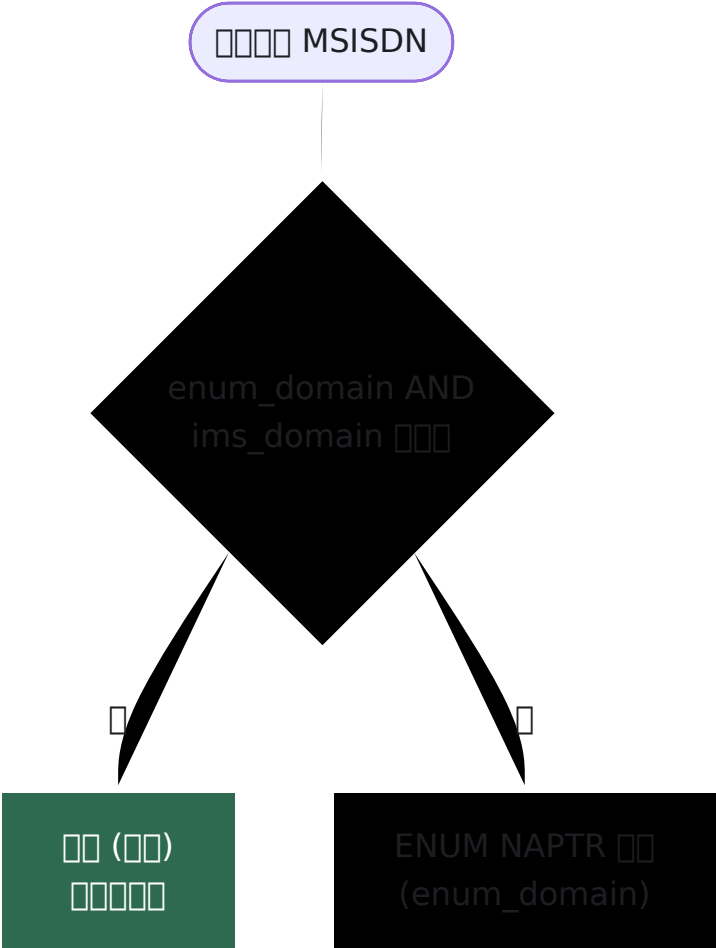
`http://<mms_domain>/<unique_id>.00`

- `mms_domain` is `mmsc.mms_domain`
- `<unique_id>.00` is the unique ID of MM1 .00

MM1 URL is `m-retrieve-conf`

## MM1/MM4

MMSc MM4 SMS-C ENUM (NAPTR) `ims_domain` ENUM





MMSc  
SMS

MMSc  
MM4

| enum                         | enum        | enum            |
|------------------------------|-------------|-----------------|
| ims_domain 或 enum_domain     | enum (enum) | enum SMS        |
| ENUM NAPTR 或 enum_ims_domain | enum        | enum SMS        |
| ENUM NAPTR 或 enum_ims_domain | enum        | enum MM4 或 enum |
| ENUM 或 enum                  | enum (enum) | enum            |

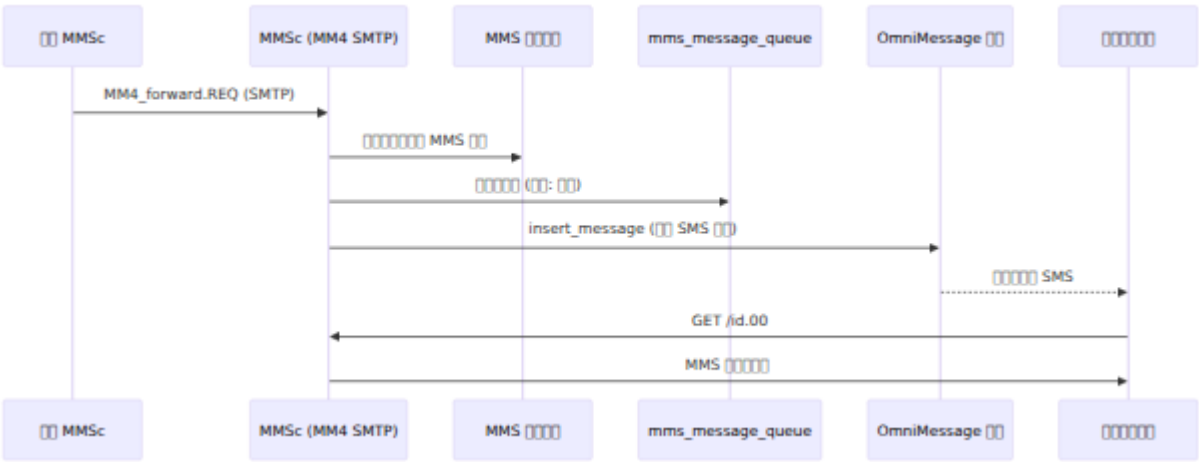
ENUM 或 enum (enum) 或 enum MM4 或 enum

enum

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

enum

enum MMSc 或 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服务域名   |
| 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<br>0/0<br>000<br>000           |
| ims_domain        | enum    | なし | ""             | IMSドメイン<br>000<br>000                   |
| pcrf_api_url      | string  | なし | nil            | PCRF API URL<br>MM<br>0 ID<br>000<br>MS |
| hss_api_url       | string  | なし | nil            | HSS API URL<br>ID (PCI)<br>pci<br>000   |
| mm4_enabled       | boolean | なし | false          | MM4が有効かどうか<br>000<br>000<br>000<br>MM   |
| mm4_smtp_port     | integer | なし | 25             | MM4 SMTPポート<br>00<br>000                |
| mm4_domain        | enum    | なし | 000 mms_domain | MM4ドメイン<br>0 M<br>MM<br>0 M             |
| default_route_mm4 | string  | なし | nil            | MM4デフォルトルート<br>000<br>000               |





MMS 消息 通过 Mnesia 的 `:mms_message_queue` 消息队列接收 SMS 消息并存储在  
消息队列中 MMS 消息 格式 `.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 固有識別子               |
| unique_id          | URL 固有識別子 (<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 message_path 固有識別子    |
| source_mmsc        | 送信元 MMSc MM4 固有識別子         |
| dest_mmsc          | 宛先 MMSc MM4_<peer> 固有識別子   |
| on_net             | MMSc ネットワーク上               |
| notify_message_ids | SMS 通知 ID SMS 固有識別子        |
| send_time          | MMSc から MMS 送信時刻           |
| notify_time        | SMS 通知時刻                   |
| retrieve_time      | MMS 受信時刻                   |
| expires            | MMS 有効期限                   |
| status             | ステータス                      |
| deadletter         | 死文字列                       |

| 項目                                                 | 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 通知 / `dead_letter_minutes` 設定

メッセージ MMS

通知 / `dead_letter_minutes`

retrieved

forwarded

expired

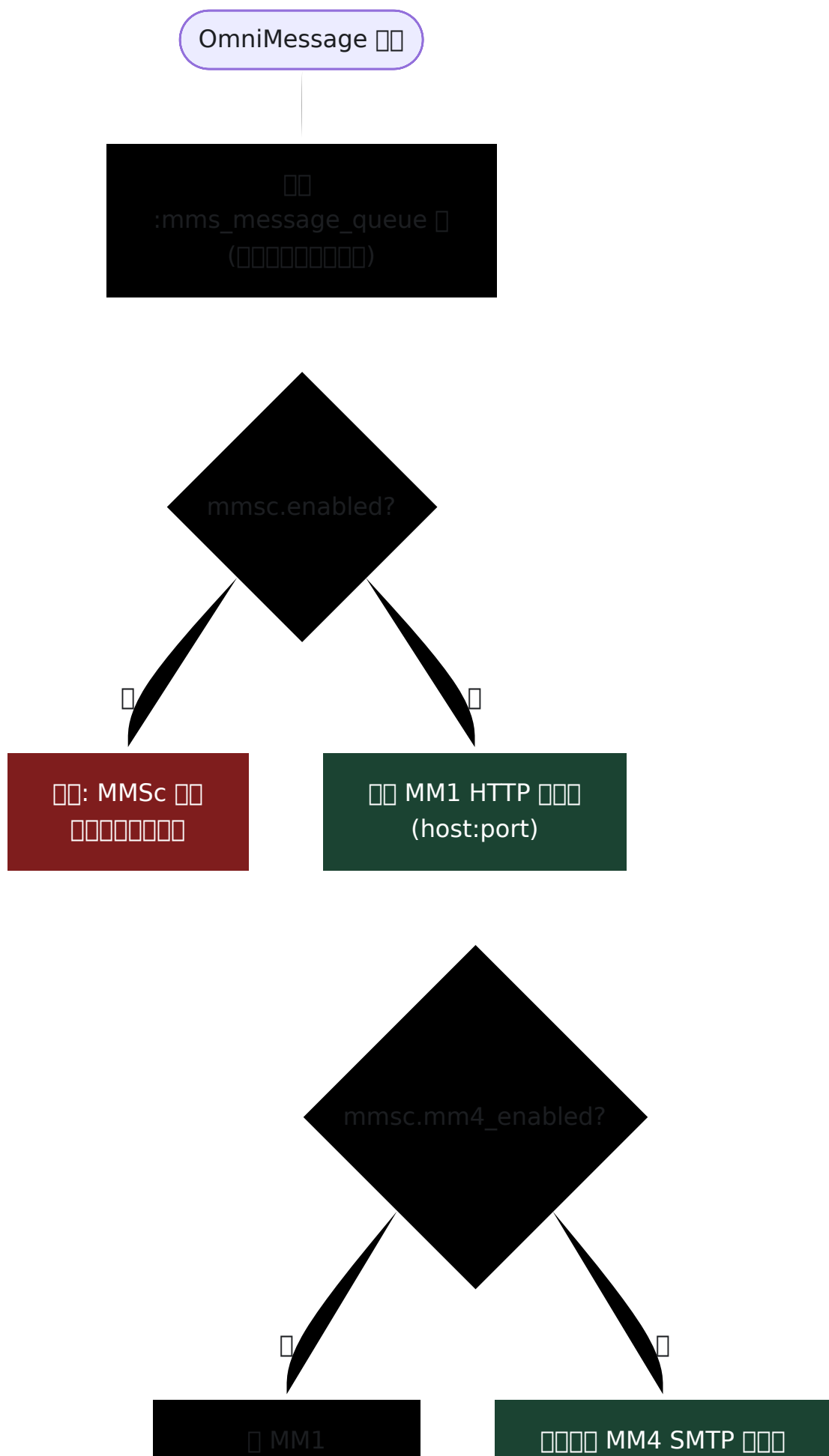
failed



| 項目        | 説明                                                       |
|-----------|----------------------------------------------------------|
| stored    | メッセージがデータベースに保存された状態                                     |
| notified  | SMS 通知が送信された状態                                           |
| retrieved | MM1 から MMS に取得された状態                                      |
| forwarded | MM4 通知が送信された状態                                           |
| expired   | メッセージが <code>dead_letter_minutes</code> 設定のタイムアウトで失われた状態 |
| failed    | 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` に設定したホストとポートに `m-send-req`、`m-notifyresp-ind`、`m-acknowledge-ind` のリクエストを送信し、`GET /<unique_id>.00` を取得する。
- **MM4 SMTP** 設定は `mmsc.enabled` を `true` に設定し、`mmsc.mm4_enabled` を `true` に設定し、`mmsc.mm4_smtp_port` に設定したポートに `MM4_forward.REQ` を送信する。
- **MM4** は、設定した `MM4_forward.REQ` のポートに SMTP のリクエストを送信する。
- 設定した `MMSc` のポートに `:mms_message_queue` を送信する。

MM4 (25) は `OmniMessage` を送信する。  
(`CAP_NET_BIND_SERVICE`) は `mm4_smtp_port` の `MM1` のポートに `8080` を設定する。

**TLS** は `MM1` のポートに `HTTP` を送信し、`mmsc.port` のポートに `TLS` を送信する。

設定

`MMSc` は `MMS Prometheus` を送信し、`SMS` は送信する。  
`OmniMessage` は `MMS` を送信し、`CDR` は `SMS` を送信する。

設定は `status`、`send_time`、`notify_time`、`retrieve_time` を送信する。

設定

## MMSc 設定

設定は `MM1` のポートに `MMSc: DISABLED` を送信する。

設定

- `:mmsc` は `runtime.exs` を送信する。
- `mmsc.enabled` は `false` を送信する。

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` 设置 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` 设置
- `:mmsc_mm4_peers` 设置
- 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  
Mnesia

- 
- 
- SMSC**
- 1-255
- 
- 
- `runtime.exe`
- 
- Web UI** CRUD
- 
- 
-

| 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一致するかどうかTON/NPI | ○  |
| calling_replace | string/nil | 発信番号表示番号の置換              | ○  |
| called_match    | string/nil | 着信番号表示番号と一致するかどうか        | ○  |
| called_replace  | string/nil | 着信番号表示番号の置換              | ○  |
| source_ton      | string/nil | 送信番号表示番号のTON/NI = 未設定    | ○  |
| source_npi      | string/nil | 送信番号表示番号のNPI/NI = 未設定    | ○  |
| priority        | int        | 優先度1-255                 | ○  |
| description     | string     | 説明                       | ○  |
| enabled         | bool       | 有効/無効                    | ○  |
| continue        | bool       | 続行かどうかfalse              | ○  |

送信番号表示番号と一致するかどうかTON/NPI

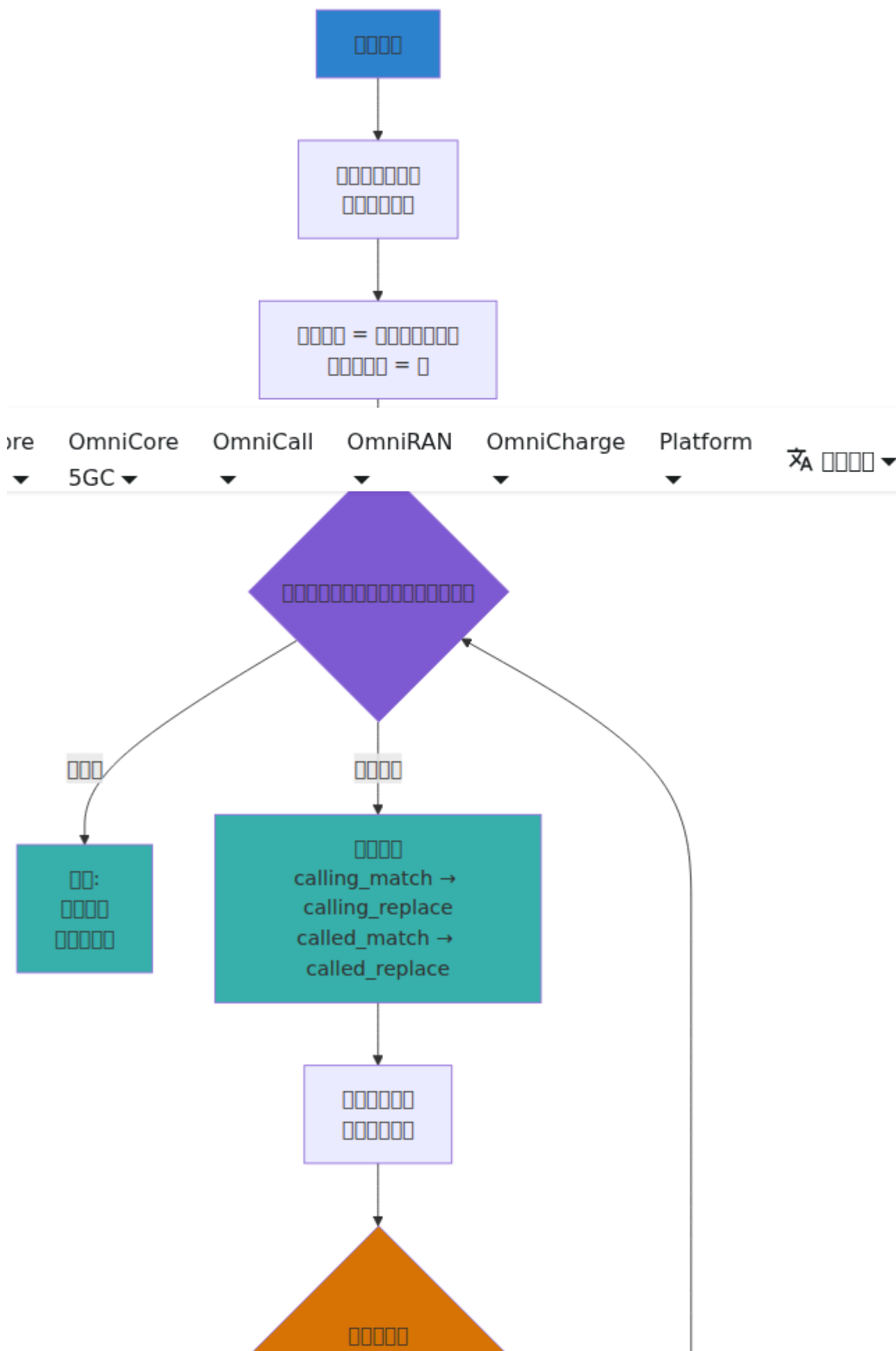
送信番号表示番号と一致するかどうかTON/NPI

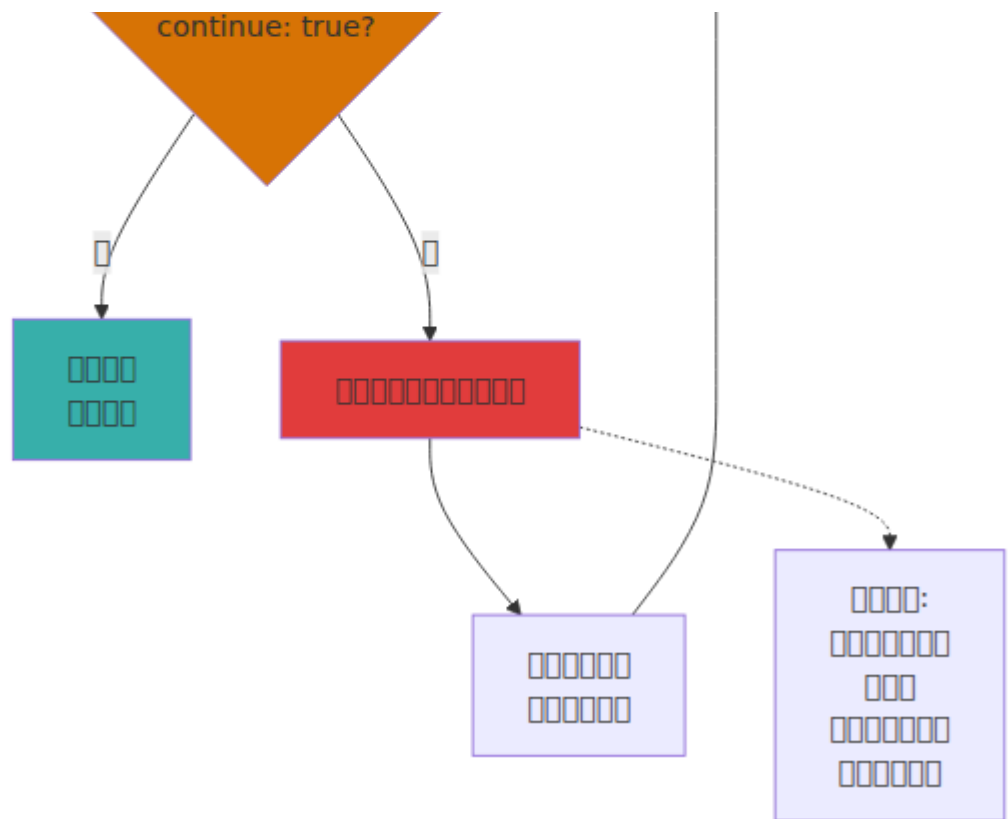
送信番号

送信番号表示番号









111

- nil
- 
- nil/

□ □ □ □ □ □ □ □ □ □

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



# 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E.164番号 |
| national         | 国内番号      |
| network_specific | ネットワーク固有  |
| subscriber       | ダイヤル番号    |
| alphanumeric     | アルファベットID |
| abbreviated      | 略称        |

国際番号NPI

| 種別          | 説明         |
|-------------|------------|
| unknown     | 不明         |
| isdn        | ISDN/E.164 |
| data        | X.121      |
| telex       | Telex      |
| land_mobile | E.212      |
| national    | 国内番号       |
| private     | 私用番号       |
| ermes       | ERMES      |

国際番号NPI



continue: true 当 TON/NPI 为空时，继续处理下一个号码。  
否则，返回 false。

说明

TPDU 中的 TON/NPI 字段

| 长度          | TON           | NPI     |
|-------------|---------------|---------|
| 长度 ≥ 5      | international | isdn    |
| 长度 < 5      | national      | isdn    |
| 号码以 Acme 开头 | alphanumeric  | unknown |

当 TON/NPI 为空时，返回 false。当 TON/NPI 不为空时，返回 true。

## TON/NPI 转换

说明

- Web UI** - 在 TON/NPI 转换规则中，使用“\*”表示任意字符。
- REST API** - 在 `/api/translations` 接口中，使用 `source_ton` 和 `source_npi` 字段。  
示例：  
`config :sms_c, :translation_rules`

示例

3-6 位的 TON/NPI 未知号码，使用 `calling_match` 和 `calling_replace` 字段。

在 `config/runtime.exs` 中

```
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
  }
]
```

```

0000000000000000 calling_prefix 0000000000000000 called_prefix 0000000000000000 source_smsc 0000000000000000
calling_match 00000000 ^\d{3,6}$ 0000000000000000 source_ton/source_npi 00000000000000003
0600000000000000 unknown/unknown 0000000000000000 calling_replace 0000000000000000 nil 0000000000000000
continue: true 0000000000000000

```

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

□ □ □ □ □ □ □ □ □ □ □

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
}
```

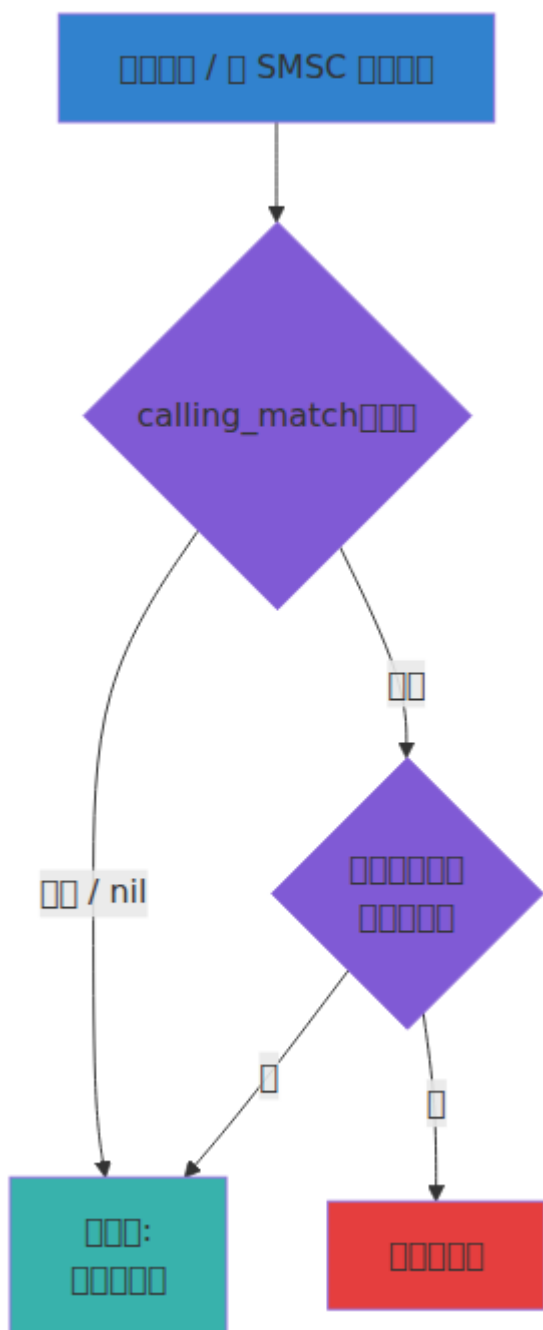
calling\_prefix: "07" 07 calling\_match  
07 8 national/isdn international/isdn  
≥ 5

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: "00000SMSC1000000000+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
  },

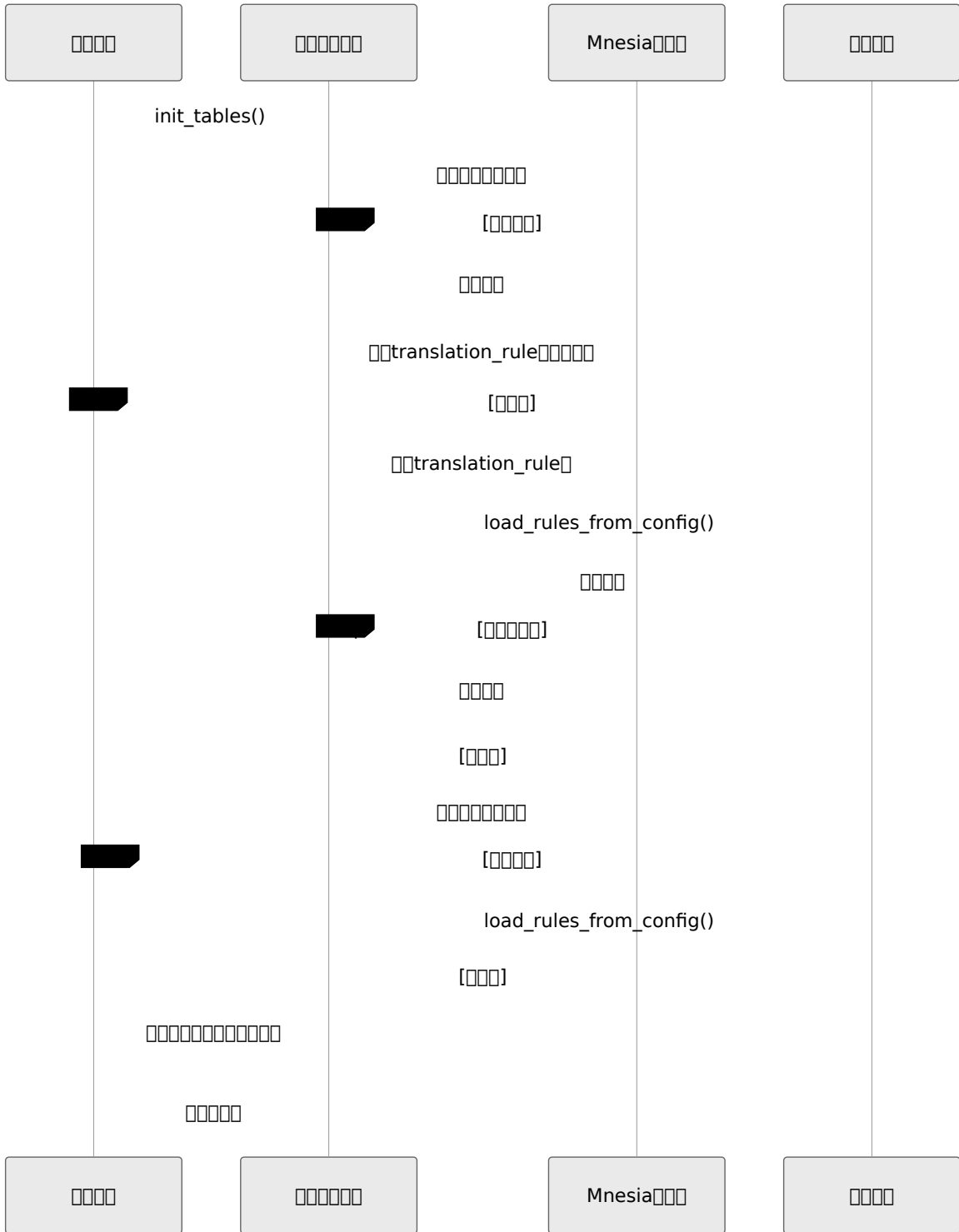
  # 0000000000unknown/unknown TON/NPI0000000000
  %{
    calling_match: "^(\\d{3,6})$",
    calling_replace: nil,
    source_ton: "unknown",
    source_npi: "unknown",
    priority: 6,
    description: "00003-600=> unknown/unknown TON/NPI",
    enabled: true,
    continue: true
  },

```

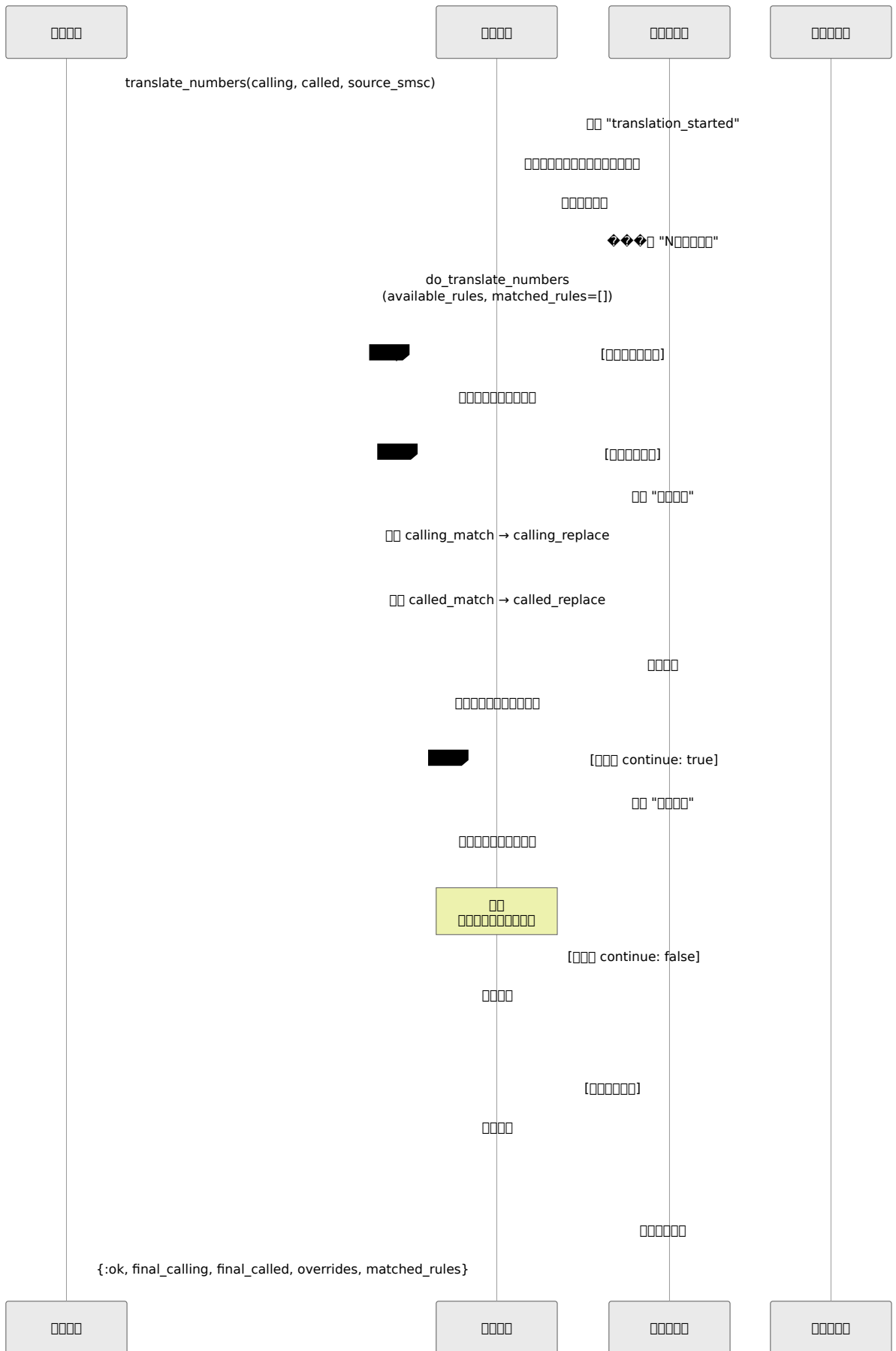
```
# [][][][][][]
%{
  calling_prefix: "+44",
  called_prefix: "+44",
  source_smsc: nil,
  calling_match: "^\\+44(.*)$",
  calling_replace: "0044\\1",
  called_match: "^\\+44(.*)$",
  called_replace: "0044\\1",
  priority: 20,
  description: "[]",
  enabled: true,
  continue: false
}
]
```

□□

□□□□□



□□□□□□



[illegible]

□□□□□□□

□□□□□□□□□□□□□□□□

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'

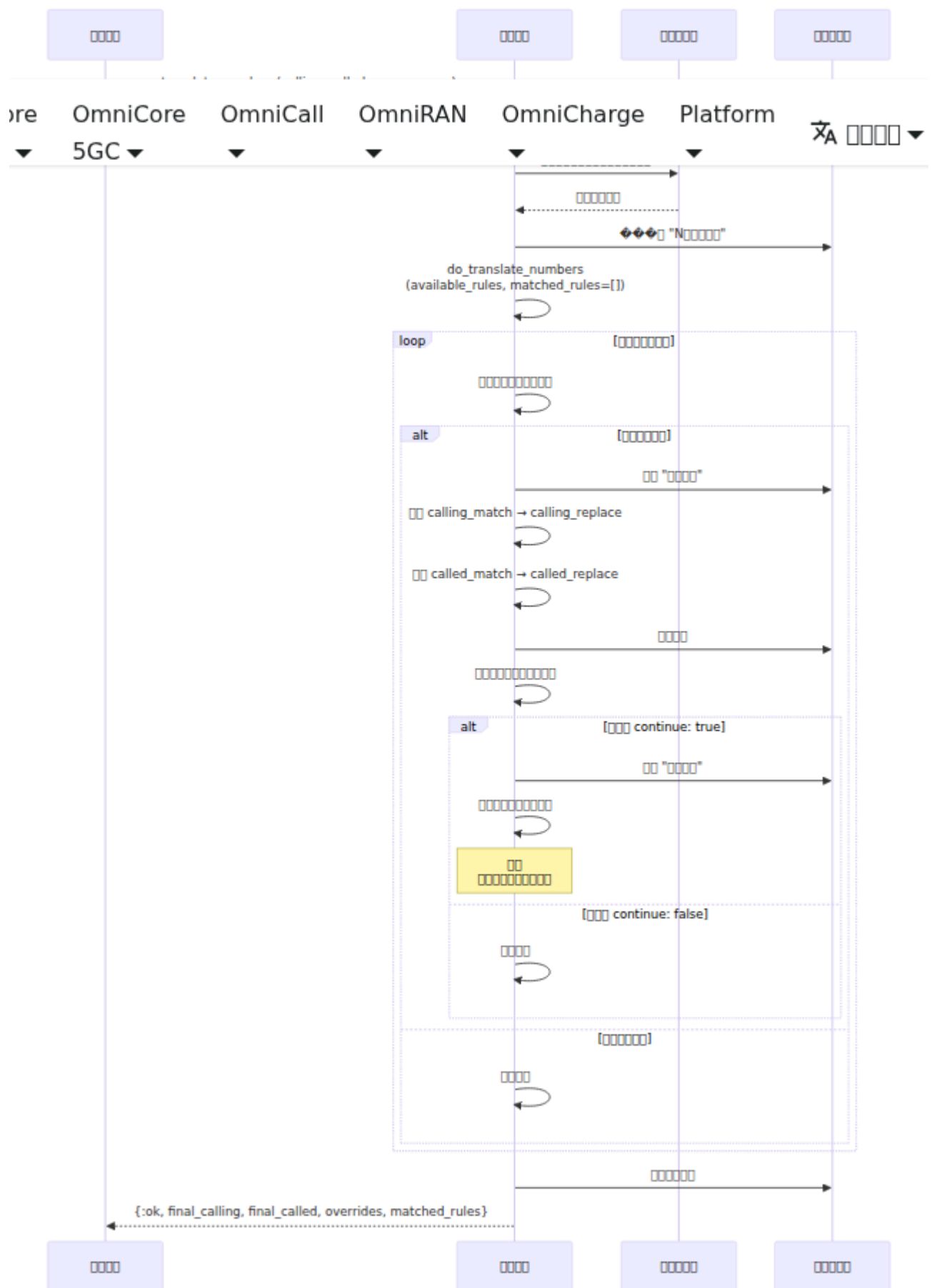


**SMSC**□□□□

□□□□□□□□□□□□











- 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

```

□□□□□□

```
# □□□□□
{: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
- 0000000000000000“5551234567 → +15551234567”0
- 0000000000/00

### 4. 0000000

- 0000000000000000
- 000000\1\200000000
- 0000000000000000

## 00

### 1. 00000000

- 000000000
- 0000000000000
- 00000000000

### 2. 0000000

- 0000000000000000
- 00000000
- 00000000

### 3. 000000000

- 0005000000000000
- 0000000000000000
- 00Telemetry00???

## ???

### 1. 000000

- 0000000000

- 000000000000000000
- 00000000

## 2. 00000

- 0000000000000000
- 0000000000
- 000000000000

## 3. 00000

- 0000ID00000000
- 0000000000000000
- 00000000

## 4. 00000

- 000000000000
- 0000000000
- 00000
- 000000000

## 0000

### 1. 00000

- 1000000000<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

□□□□□□□□□□

□□□□□□□□□□□□

□□□□

- □□□□□□□□
- □□□□□□□□
- □□□□□□□□

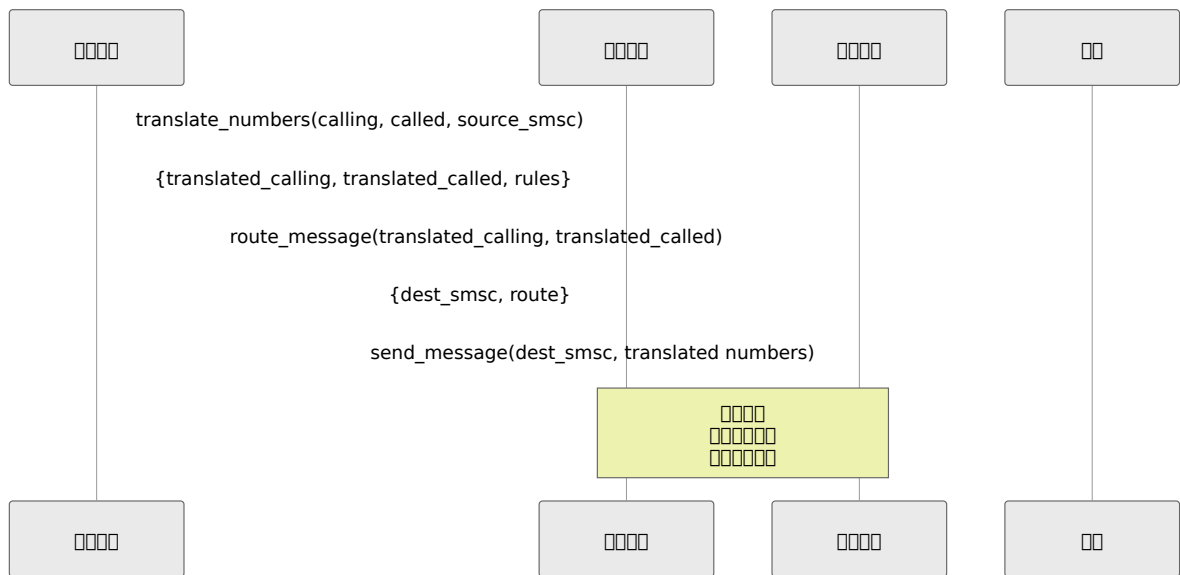
□□□□

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

□□□□

□□□□□□

□□□□□□□□□□□□□□□□



## EventLog

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

□□

[illegible]

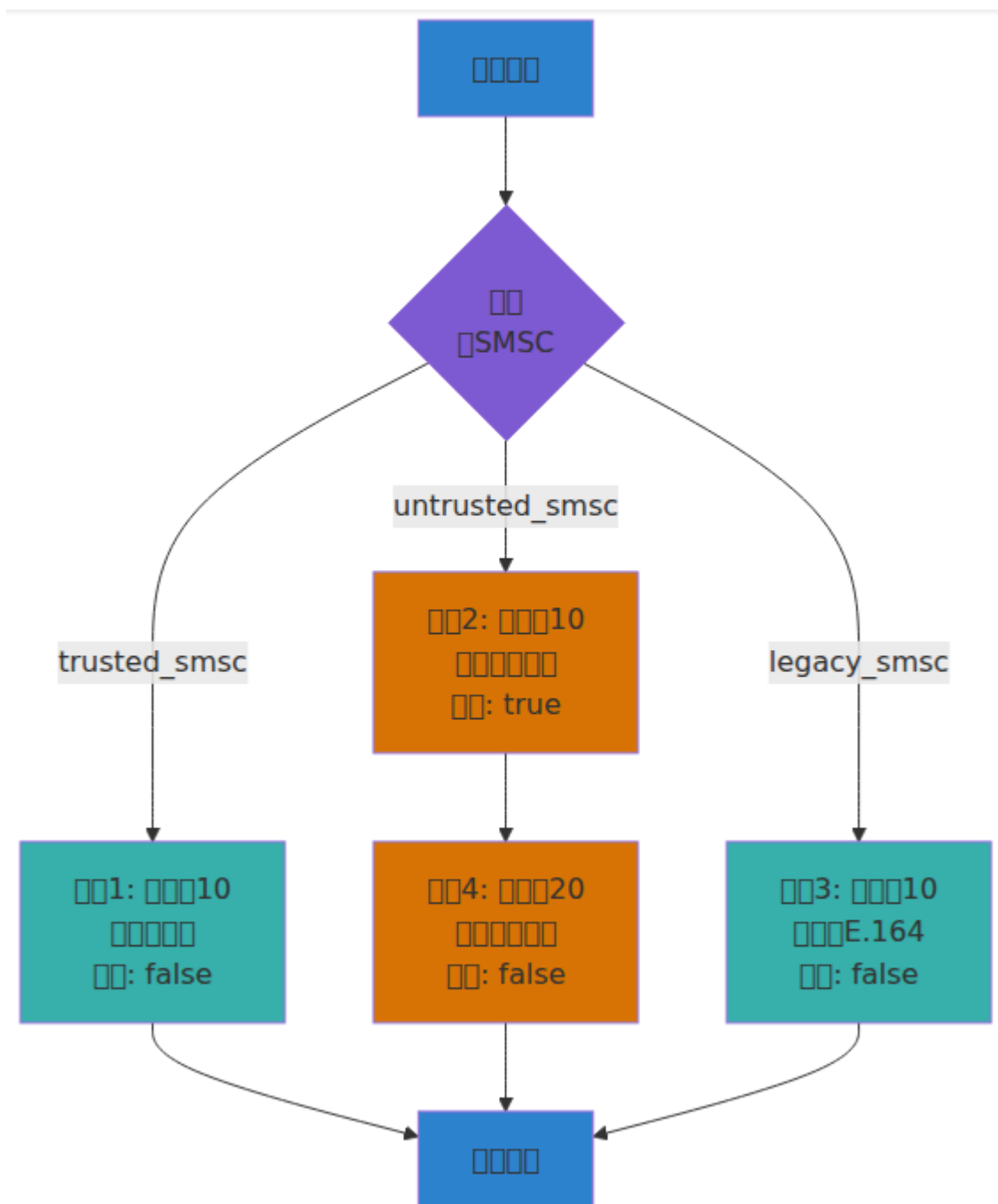
```
Parse error on line 25: ... style New fill:#3182CE style P -----^
Expecting 'SOLID_OPEN_ARROW', 'DOTTED_OPEN_ARROW', 'SOLID_ARROW',
'BIDIRECTIONAL_SOLID_ARROW', 'DOTTED_ARROW',
'BIDIRECTIONAL_DOTTED_ARROW', 'SOLID_CROSS', 'DOTTED_CROSS',
'SOLID_POINT', 'DOTTED_POINT', got 'TXT'
```



□□□□

□□□□□□□□





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

```
# 100+
%{
  calling_match: "^00(.+)$",
  calling_replace: "+\1",
  called_match: "^00(.+)$",
  called_replace: "+\1",
  priority: 5,
  description: "00+ ",
  enabled: true,
  continue: true #
}

# 2
%{
  calling_match: "^\\+(\\d+)$",
  calling_replace: "00\1",
  called_match: "^\\+(\\d+)$",
  called_replace: "00\1",
  priority: 10,
  description: "+00 ",
  enabled: true,
  continue: false #
}

#
# 1: "00441234567890" → "+441234567890"1
# 2: "+441234567890" → "00441234567890"2
# : "00441234567890"
# : [1, 2]
```

## 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
}

# 02000000000000000000
%{
  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
}

# 0300000000000000
%{
  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 端點
curl https://api.example.com:8443/api/status

# 響應示例
# {"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

- 消息接收
- 消息发送
- 消息状态

## 消息接收

消息接收成功

通过 Prometheus 监控

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

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

# 消息状态
rate(sms_c_delivery_succeeded_count[1h]) /
rate(sms_c_message_received_count[1h])
```

消息接收失败

- 消息接收失败
- 消息接收失败/消息接收成功
- 消息接收失败 > 95%

消息接收异常

- 消息接收失败 > 50% 异常
- 消息接收失败 > 200% 异常
- 消息接收失败 > 90% 异常

## 消息发送

消息发送成功

消息发送失败

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` かどうか
- `histogram_quantile(0.95, sms_c_delivery_succeeded_attempt_count)`

~~~~~

~~~~~ (`sms_c_queue_size_size`)

- かどうか
- `> 10,000`
- `sms_c_queue_size_size`

~~~~~ (`sms_c_queue_oldest_message_age_seconds`)

- かどうか
- `> 300`
- `sms_c_queue_oldest_message_age_seconds`

~~~~~

~~~~~

1. かどうか

- 5 かどうか
- 5 かどうか
- 24 かどうか

2. かどうか

- かどうか
- かどうか
- かどうか

3. かどうか

- かどうか
- かどうか

- 00000000 24 00

4. 00000000

- 00 ID
- 000000000000
- 00 SMSC
- 000

5. 00000000

- 0000
- 000000/000
- 0000
- 000000000000

6. 0000000000

- API 00000p950
- 000000000p950
- ENUM 00000p950

0000

00000000000000

```

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

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

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

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

```

短信系统监控配置

```

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

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

# ENUM 検索遅延率
- alert: SlowEnumLookups
  expr: histogram_quantile(0.95, sms_c_enum_lookup_stop_duration)
> 5000
  severity: warning
  description: "ENUM 検索遅延率 > 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回目送信
  ↓
4. delivery_attempt_2 - 2回目送信
  ↓
5. delivery_attempt_3 - 3回目送信
  ↓
6. message_dead_letter - 失敗

```

メッセージの状態

メッセージの状態

- “queued” → “backoff” 状態
- deliver_after 0 なら “queued” → nil / 0 なら “backoff” 状態
- delivery_attempts 0 なら “queued” 0 なら “backoff” 状態

メッセージの状態

- “delivered” → “sent” 状態 DLR
- deliver_time 状態
- dest_smsc 状態

メッセージの状態

- “backoff” → delivery_attempts 0 - 30 回失敗したら “expires” 状態
- “expired” → “dropped” → “balance_rejected” → “auto_replied” - 状態
- 状態

メッセージの状態

SMS-C 状態

メッセージの状態

□□□□ `get_messages_for_smsc(smsc_name)` □□□□□□□□□□◆◆□□□□□□

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

□□□□□□

□□ MSISDN □ `+447700900123` □□□□□□□□ `uk_gateway` □□□

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

□□□□□□□□□□□□□□□□□□□□

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

`uk_gateway` □□□□□□□□□□□□□□□□

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

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

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 SMSC 消息队列
- expires 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
# expires 消息 > 消息
```

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

```
# 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"
  }'
```

完成

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

完成

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

完成

完成

Web UI [/sms_routing](#)

API

```
# 创建路由
curl https://api.example.com:8443/api/routes
```

完成

Prometheus 配置

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


Web UI

1. `/sms_routing`
2. `"XXXXXX"`
3. `XXXXXX`
 - `XXXXXXXXXXXXXXXXXXXX`
 - `XXXXXXXXXXXXXXXXXXXXXXXXXXXX`
 - `X SMSCXXXXXXXXXXXX`
 - `XX SMSCXXXXXXXXXXXX/XXXXXXXXXX`
 - `XXXXXXXXXXXXXXXXXXXXXXXXXXXX Diameter/HSSX`
 - `XXXXXXXXXXXX1-255XXXXXXXXXXXX`
 - `XXXXXXXXXXXX1-100X`
 - `XXXXXXXXXXXX`
 - `XXXXXXXXXXXX`
4. `"XXXXX"`

- 00000 +44
- 00 SMSC uk_gateway
- 0000 50
- 000 100
- 000 "00000"

□□□□□□□□□□

[illegible]

□□ 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

- 送信先を指定する
- 送信先を指定する
- 送信先を指定する

送信先

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

送信先

Web UI

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

送信先

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

送信先

Web UI

1. 送信先 /sms_routing
2. 送信先
3. 設定“送信先”

4. 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
 - 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` 用于配置系统

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資料庫 30-90 查詢
- 資料庫資料庫
- 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

0000

```
# 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
```

SQL CDR 0000

0000000000000000

- **MySQL/MariaDB**000000 mysql 00000000
- **PostgreSQL**0000 pg_restore 000000000000

000000000000 SMS-C 0000000000000000

000000

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;
```

監視項目

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

📌

📌

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

📌

1. 📌
2. 📌
3. 📌
4. 📌
5. 📌🔍🔍🔍📌

📌

1. 📌
2. 📌
3. 📌
4. 📌 API 📌
5. 📌

📌

1. 📌
2. 📌
3. 📌CPU/📌
4. 📌 ENUM 📌
5. 📌

📌 📌.

□□□□

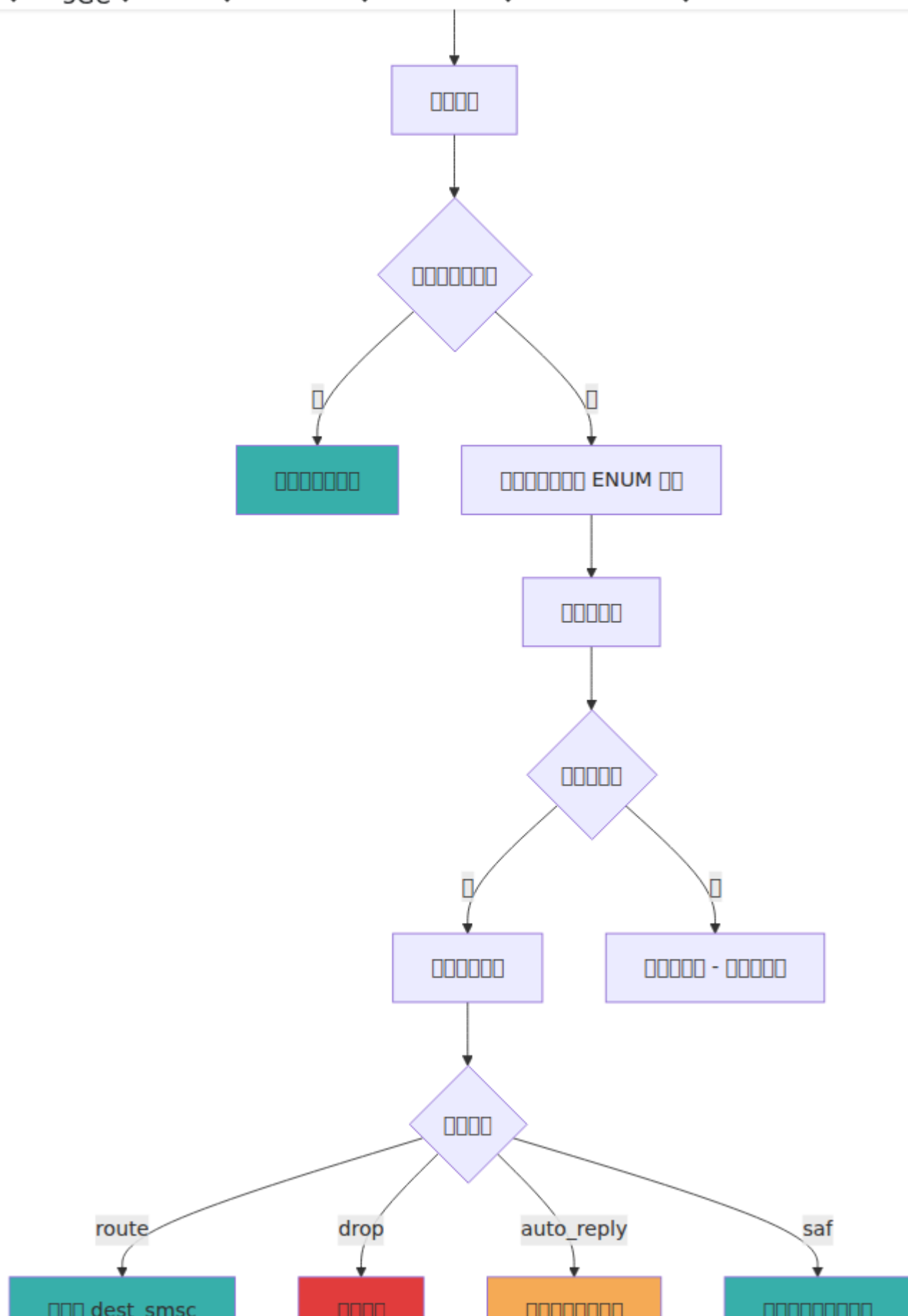
← □□□□□

□□

SMS-C □□□□□□□□□□□□□□□□ Mnesia □□□□□□□□□□ API □□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□ ENUM□DNS NAPTR□□□□□□□□□□ HSS □□□□/□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□





originating_network / terminating_network
- /

1. -
2. - HSS /
3. -

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	着信者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に一致しない
source_type	String	発信者タイプ。ims, circuit_switched, smpp, nil
enum_domain	String	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	String	ENUM NAPTRの結果。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": "AU 配置"
}
```

drop

配置项: dest_smsc

```
{
  "action": "drop",
  "called_regex": "^900",
  "priority": 5,
  "description": "配置"
}
```

auto_reply

配置项: auto_reply_message

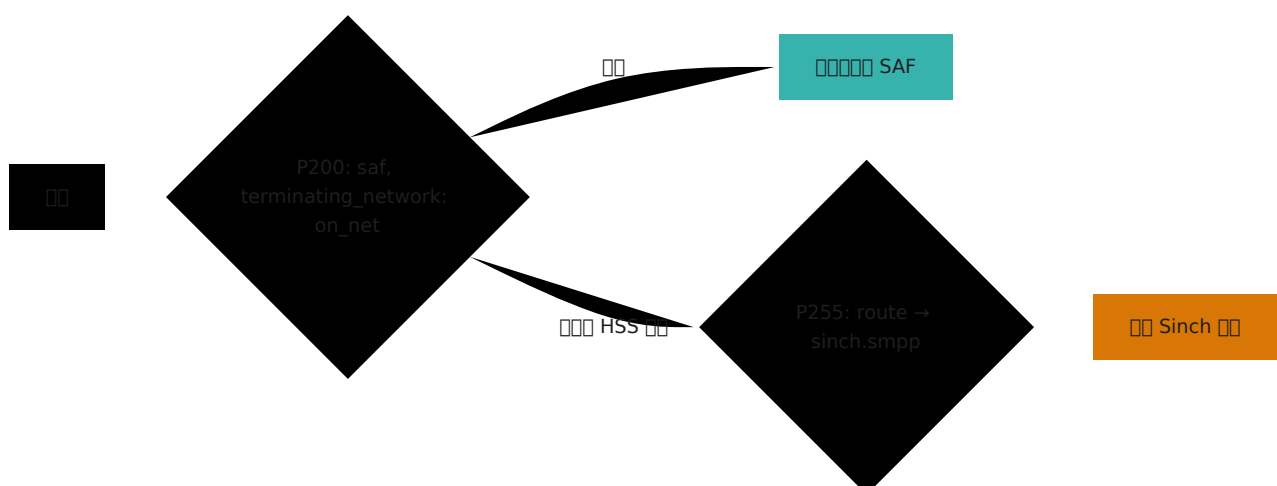
```
{
  "action": "auto_reply",
  "called_regex": "^911$",
  "auto_reply_message": "☎☎☎☎☎☎☎☎ 911☎☎☎☎ 911☎",
  "priority": 1,
  "description": "911 ☎☎☎☎"
}
```

saf

```
00000000  dest_smsc
```

```
saf 00000000 - 0000 HSS 00000000000000000000 saf 0 terminating_network:
:on_net 0000000000000000000000000000
```

```
{
  "action": "saf",
  "terminating_network": "on_net",
  "priority": 200,
  "description": "□□□□□□□□□□"
}
```

$$\square\square\square\square\square\square\square\square + \square\square\square$$


P200 SAF HSS -
P255

ENUM 簡介

ENUM 將電話號碼與 DNS E.164 NAPTR 記錄關聯，以便通過 DNS 查詢 HSS 獲取 IMS 地址。

配置

1. 啟用 ENUM 功能，設置 `enum_enabled: true` 和 `enum_domains` 列表。
2. 將電話號碼 +15551234 轉換為 E.164 格式 4.3.2.1.5.5.5.1.e164.arpa。
3. 通過 NAPTR 記錄中的 `result_domain` 和 `replacement` 字段，將 sip:user@host 轉換為 sip:+15551234@ims.example.com。
4. 設置 `enum_result_domain` 為 NAPTR 記錄中的 `result_domain`。
5. 設置 ETS 值為 15，並設置 `(number, domain)` 列表。

ENUM 數據庫/表

ENUM 數據庫/表 - 存儲 `enum_result_domain` 和 IMS 地址。HSS 通過“表”查詢 HSS 數據庫/表。

ENUM 查詢

- 通過 NAPTR 記錄中的 `enum_result` 字段，獲取 URI 列表 - 通過 `route.dest_smsc` 字段。
- 通過 `enum_domain` 列表，查詢 DNS 記錄中的 `enum_domains` 列表。

配置

ENUM DNS 配置 `enum_enabled` `enum_domains` `enum_dns_servers` `enum_timeout` - 參見 [CONFIGURATION.md](#) → ENUM 配置

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、SMSC/電話番号で指定される。

originating_network / terminating_network

設定例

項目	説明
:any	HSS に登録されている
: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、必ずしも一致する必要がある。
- terminating_network: :on_net は、SMPP の宛先電話番号が originating_network: :on_net に一致している場合にのみ、terminating_network: :off_net に設定される。

設定例

例として :any の IP アドレスと HSS の登録情報を示す。

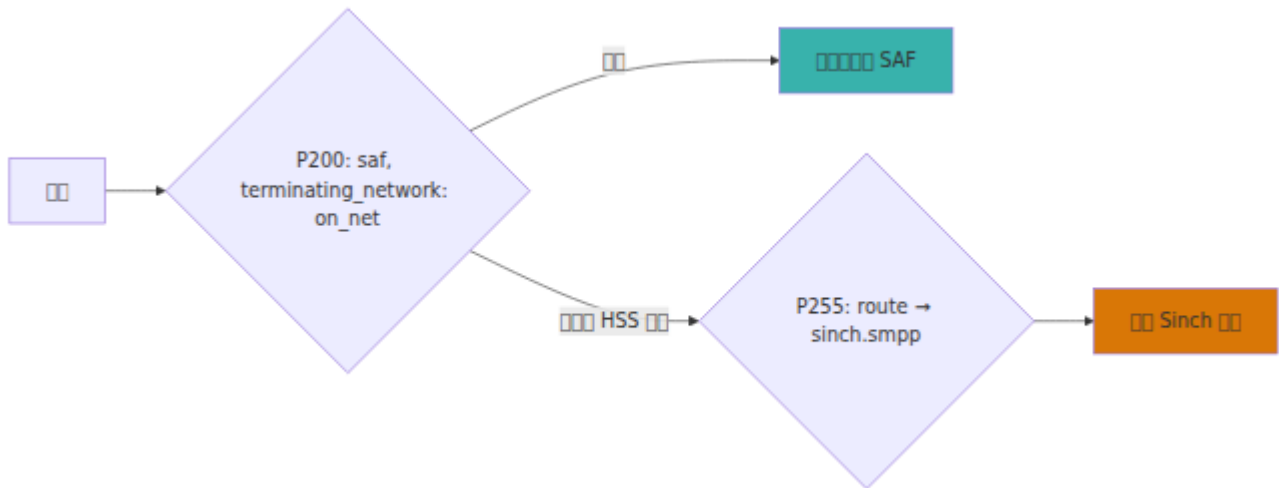
- Diameter IP → IP アドレス
- HSS IP / IP / IP → IP アドレス
- HSS IP / IP / IP → IP アドレス

例として :off_net の IP アドレスと HSS の登録情報を示す。

HSS 設定

HSS の IP アドレスと Diameter の IP アドレス、MSISDN の登録情報。

- 参数 **TTL** 2 参数 `hss_cache_ttl_seconds` 参数
- 参数 `originating_network` 参数 `terminating_network` 参数
`:on_net` 参数 `:off_net` 参数 HSS 参数 `:any` 参数
- 参数 `MSISDN` 参数 HSS 参数
- 参数



参数

参数 `route_evaluated` 参数

- 参数 ID 参数
- 参数 `enum_result` 参数 `orig_network/term_network` HSS 参数
- 参数 `SMSC` 参数 HSS 参数

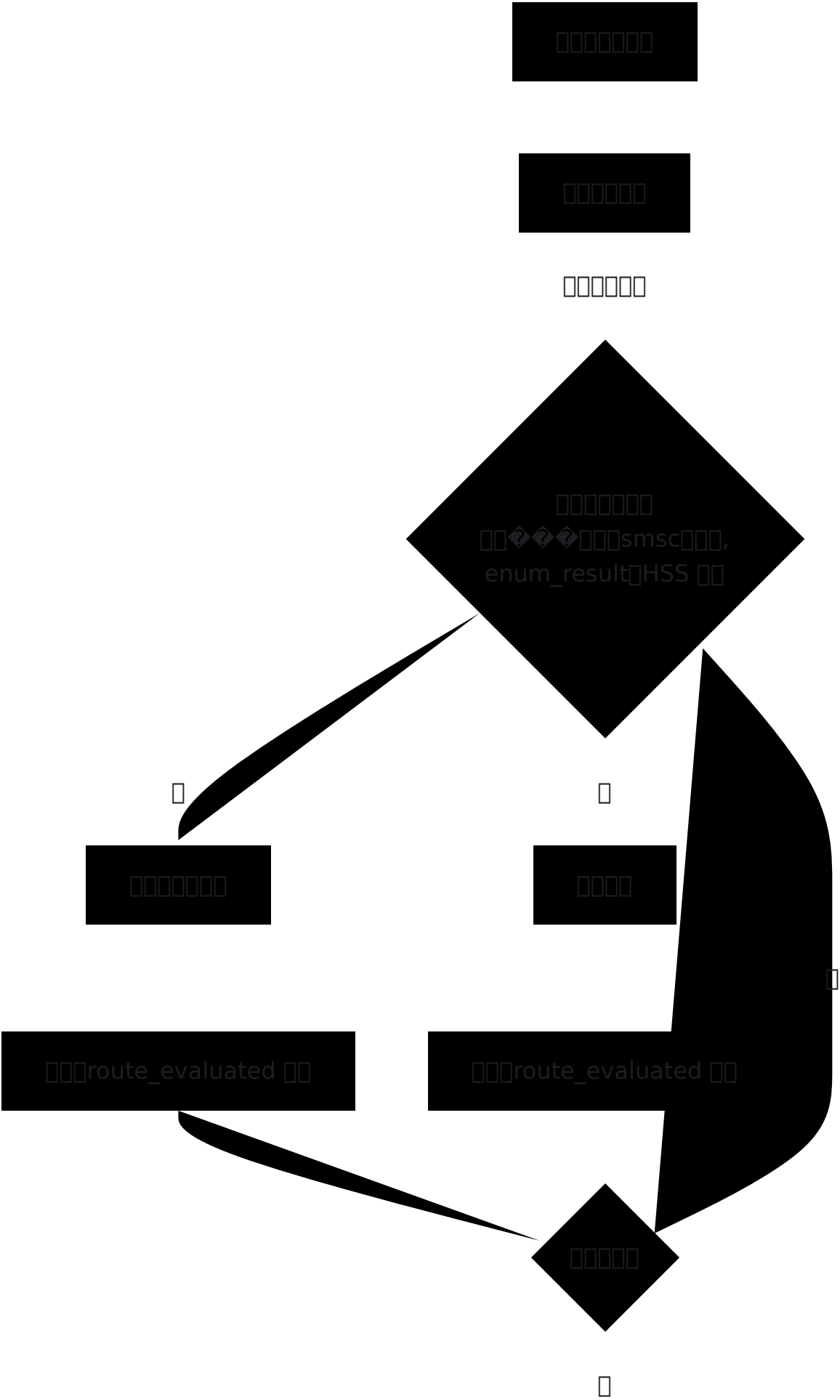
参数

```
#4 [saf] 参数P200 "[]" |
calling_regex: ✓ (参数), called_regex: ✓ (参数),
source_smsc: ✓ (参数), source_type: ✓ (参数),
enum_result: ✓ (参数),
term_network: ✓ (参数+15551234 参数)
```

```
#3 [route] 参数P255 "Sinch 参数" -> sinch.smpp |
calling_regex: ✓ (参数), called_regex: ✓ (参数),
source_smsc: ✓ (参数), source_type: x (ims vs circuit_switched),
enum_result: ✓ (参数), term_network: ✓ (参数)
```

□□□□□□□□□□□□□□□□ API □□□





XXXXXXXXXX

XXXXXXXX

XXXXXX

XXXX

XXXX

HSS 00/XXXXXXXXXXXXXXXXXXXXXXXXXXXX - XXXXXXX HSS 000

XXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

項目	サイズ
<code>called_regex</code> 文字列	文字 × 100
<code>calling_regex</code> 文字列	文字 × 50
<code>source_smsc</code> 文字	+25
<code>enum_result_domain</code> 文字	+15
<code>source_type</code> 文字	+10
<code>enum_domain</code> 文字	+5

文字列の長さは、

文字

文字列

① `config/runtime.exs` 文字列の長さは、API 文字列

[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
charged	boolean	なし	:default	充電済み:yes 未充電: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
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。默认值为 7200 秒。在 <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


```
DELETE /api/routes/:id
```

00/00

```
POST /api/routes/import
GET /api/routes/export
```

merge replace

□ □ □ □

□ □ □ □ □

```
dest_smsc: null sms_routing_failed
```

□□□□□

- 電話番号を指定する
- 電話番号を指定する
- 電話番号を指定する HSS 番号 `terminating_network/originating_network` 番号を指定する HSS 番号を指定する `:any` 番号を指定する

□ □ □ □ □

1. 00000000 - 0000000000000000/0000 HSS 0000
2. 000000000000000000
3. 000000000000 originating_network: :any terminating_network:
:any 000000000000 255

□ □ □ □ □ □ □ □ □ □

[illegible]

□□□□

- □□□□ `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
```

□□□□□□

```
# 实时监控
tail -f /var/log/sms_c/application.log | grep -E "
(error|warning|critical)"
```

配置

配置项

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

# 配置
rate(sms_c_delivery_succeeded_count[5m]) /
rate(sms_c_delivery_queued_count[5m])
```

配置

```
# 配置
sms_c_queue_size_pending

# 配置
sms_c_queue_oldest_message_age_seconds
```

配置

```
# 配置 (p95)
histogram_quantile(0.95,
sms_c_message_processing_stop_duration_bucket)

# 配置 (p95)
histogram_quantile(0.95, sms_c_routing_stop_duration_bucket)
```

API 消息

API 消息

API

- 消息 "queued"/"backoff" 消息
- 消息接收器
- 消息接收器

API

1. 消息接收器

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

API 消息接收器

API 消息接收器

2. 消息接收器

Web UI: /message_queue

- 消息接收器 "queued" 消息 "backoff"
- 消息 dest_smsc 消息
- 消息 deliver_after 消息

3. 消息接收器

Web UI: /simulator

- 消息接收器接收器
- 消息接收器接收器

4. 消息接收器

API 消息接收器

- 消息接收器 /api/messages 消息

- 配置 `smc` 参数

配置

配置

```
# 检查服务状态
systemctl status frontend_service

# 检查 API 接口
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": "smp",
  "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"}'
```

API 仕様

リクエスト

- `delivery_attempts` フィールド
- `delivery_attempts > 3`
- `status`

レスポンス

1. エラーレスポンス

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

エラー

- `400`
- `401`
- `403`

2. 成功レスポンス

- `200`
- `201`
- `202`
- `204`

セキュリティ

認証

- `Basic Auth`
- `API Key`

その他

更新メッセージ

```
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"`メッセージキューのメッセージ `expires`メッセージ

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

000000000

```
0000: +
00 SMSC: international_gateway
000: 200
00: 100
00: ✓
00: 000000
```

000000000

- 00 Web UI 0000000
- 0000000000000

00000000

000

- 000000000000
- 0000000000
- 000000000000

000000

1. 0000000000

00 Web UI: /simulator

- 00000000000000
- 00 "0000" 00
- 00000000000000

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 < 100 msg/sec
- 使用 API 进行测试
- 使用 API 进行测试

测试

1. 性能测试

```
# 性能测试 (iex)
SmsC.Messaging.BatchInsertWorker.stats()
```

测试

- `current_queue_size` 当前队列大小
- `flush_errors` > 0
- `last_flush_duration_ms` 上次 flush 耗时

2. 性能测试

```
# 查询时间
ecto_pools_query_time

# 队列时间
ecto_pools_queue_time
```

配置

配置

配置

```
config :sms_c, SmsC.Repo,
  pool_size: 50 # 20 个
```

配置

配置

```
config :sms_c,
  batch_insert_batch_size: 200, # 批量插入
  batch_insert_flush_interval_ms: 200 # 批量插入间隔
```

配置

```
# 异步创建 /create_async
curl -X POST
https://api.example.com:8443/api/messages/create_async

# NOT: /api/messages (同步)
```

配置

配置

- `sms_c_queue_size_pending` 配置
- 配置

- 0000000000000000

000000

1. 000000000000

```
# 00000
rate(sms_c_message_received_count[5m])

# 00000
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
# 数据库
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

# 数据库 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/网络

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
- 配置

配置

1. 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 空间 (MySQL/MariaDB)
du -sh /var/lib/mysql
```

```
# 检查 PostgreSQL 空间 (PostgreSQL)
du -sh /var/lib/postgresql
```

2. 清理空间

```
# 清理 MySQL (MySQL/MariaDB)
find /var/lib/mysql -type f -exec du -h {} + | sort -rh | head -20
```

```
# 清理 PostgreSQL (PostgreSQL)
find /var/lib/postgresql -type f -exec du -h {} + | sort -rh |
head -20
```

```
# 清理日志
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 计划任务

部署

测试

验证

- 检查日志文件 "expired"
- 检查日志文件
- 检查日志文件

部署

1. 部署

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

2. 部署

- 部署 `/api/frontends/register`
- 部署 API 接口
- 部署 60 秒

3. 部署 API 接口

```
grep "frontend.*register" /var/log/sms_c/application.log | tail -20
```

□□□□□

□□□□□□

□□□□□□□

```
curl -X POST https://api.example.com:8443/api/frontends/register \
-H "Content-Type: application/json" \
-d '{
  "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

□□□□□□□□□□/□□

□□□

- 設定ファイル/ディレクトリ
- 設定ファイル
- 設定ファイル

設定ファイル

1. 設定ファイル

- 設定ファイル
- 設定ファイル
- 設定ファイルCPU/メモリ

2. 設定ファイル

```
# 設定ファイル
ping -c 100 api.example.com

# 設定ファイル
netstat -s | grep -i reset
```

3. 設定ファイル

- 設定ファイル
- 設定ファイル> 90 設定

設定ファイル

設定ファイル

- 設定ファイル
- 設定ファイル
- 設定ファイル

設定ファイル

- 設定ファイル
- 設定ファイル
- 設定ファイル

確認する

確認する

```
REGISTRATION_INTERVAL = 60 # 分
```

確認 **Diameter Ro** 設定が OCS Diameter 設定と一致していることを確認する
:balance_rejected

確認

- sms_c_charging_failed_count
- "charging_failed" "balance_rejected"
- :balance_rejected Diameter 設定

確認

1. **Diameter Ro** 設定が CER/CEA 設定と一致していることを確認する

```
# 確認  
:diameter.service_info(:omnimessage, :connections)
```

2. 設定が :ro 設定と一致していることを確認する

```
grep -E "diameter_enabled|application_name: :ro"  
config/runtime.exs
```

確認 Charging requested but the Diameter Ro stack is not enabled/configured 設定が一致していることを確認する

3. 設定が OCS 設定と一致していることを確認する

```
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

配置策略

配置策略

- 配置策略
- 命中率 < 70% 策略

配置策略

设置 NAPTR 策略

配置策略

- 配置策略
- 配置策略
- 设置 TTL

配置策略

配置策略

配置策略

- 配置策略
- 配置策略

- 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. 環境構築

```
# 127.0.0.1:3000@node1.example.com
ping node2.example.com

# 127.0.0.1
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

```
# 检查节点是否在线
Node.list()
```

2. 安装 Mnesia

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

安装步骤

安装依赖

```
# 安装 Erlang
systemctl stop omnimessage

# 安装 Mnesia
systemctl start omnimessage # 在 node1 上

# 安装 Mnesia 数据库
systemctl start omnimessage # 在 node2 上
systemctl start omnimessage # 在 node3 上
```

Mnesia 配置

- 配置 Mnesia 数据库
- 配置 Mnesia 节点
- 配置 Mnesia 节点
- 配置 Mnesia 节点
- 配置 Mnesia 节点

API 接口

API 接口

接口

- 接口

- 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
```

1. 配置

2. 配置

3. 配置

4. 配置

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

API 500 配置

1. 配置

- 配置
- 500 配置
- 配置

2. 配置

1. 配置

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

2. 配置

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

3. 配置

```
# 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. サービス確認

```
systemctl status omnimessage
```

2. ポート確認

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

3. URL 確認

- 確認
- 確認
- HTTP と HTTPS

確認

確認

確認

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

ポート 80 を 4000 に変更

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";  
};
```

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 📁📁📁📁

-     
-     

-     > 90%
-    
-    

1.    

```
free -h
```

```
ps aux | grep beam.smp
```

2.    

```
sms_c_enum_cache_size_size
```

ENUM   

-    
-   TTL
-                                      

□□□□

- □□□□
- □□□□□□□□

□□□□

- □□□□□□
- □□□□□□□□

□□□□□□□□□□

- □□□□ - □□□□
- □□□□ - □□□□
- □□□□ - □□□□
- □□□□□□ - `/var/log/sms_c/application.log`

