Guia da API REST

← Voltar para a Documentação Principal

Este guia fornece documentação abrangente para a **API REST** e **Swagger UI** do OmniSS7.

Índice

Visão Geral

O OmniSS7 fornece uma API REST para acesso programático às operações MAP (Mobile Application Part). A API permite que você:

- Envie requisições MAP (SRI, SRI-for-SM, UpdateLocation, etc.)
- Recupere respostas MAP
- Monitore métricas do sistema via Prometheus

Arquitetura da API

Configuração do Servidor HTTP

Detalhes do Servidor

Parâmetro	Valor	Configurável
Protocolo	HTTP	Não
Endereço IP	0.0.0.0 (todas as interfaces)	Apenas via código
Porta	8080	Apenas via código
Transporte	Plug.Cowboy	Não

URL de Acesso: `http://[server-ip]:8080`

Habilitando/Desabilitando o Servidor HTTP

Controle se o servidor HTTP inicia:

```
config :omniss7,
  start_http_server: true  # Defina como false para desabilitar
```

Padrão: `true` (habilitado)

Quando Desabilitado: O servidor HTTP não será iniciado, e a API REST/Swagger UI estarão indisponíveis.

Swagger UI

A API inclui um **Swagger UI** para documentação interativa da API e testes.

Acessando o Swagger UI

URL: `http://[server-ip]:8080/swagger`

Recursos:

- Documentação interativa da API
- Funcionalidade de teste para endpoints
- Esquemas de requisição/resposta
- Exemplos de payloads

Swagger JSON

A especificação OpenAPI está disponível em:

URL: `http://[server-ip]:8080/swagger.json`

Casos de Uso:

- Importar para Postman ou outros clientes de API
- Gerar bibliotecas de cliente
- Automação da documentação da API

Endpoints da API

Todos os endpoints de operação MAP seguem o padrão: `POST /api/{operation}`

Resumo dos Endpoints

Endpoint	Método	Propósito	Timeout
`/api/sri`	POST	Enviar Informações de Roteamento	10s
`/api/sri-for-sm`	POST	Enviar Informações de Roteamento para SM	10s
`/api/send-auth-info`	POST	Enviar Informações de Autenticação	10s
`/api/MT-forwardSM`	POST	Encaminhar SM Terminado para Móvel	10s
`/api/forwardSM`	POST	Encaminhar SM	10s
`/api/updateLocation`	POST	Atualizar Localização	10s
`/api/prn`	POST	Fornecer Número de Roaming	10s
`/api/ussd/send`	POST	Push USSD originado na rede	10s
`/metrics`	GET	Métricas Prometheus	N/A
`/swagger`	GET	Swagger UI	N/A
`/swagger.json`	GET	Especificação OpenAPI	N/A

Nota: Todas as requisições MAP têm um **timeout de 10 segundos codificado**.

SendRoutingInfo (SRI)

Recupere informações de roteamento para estabelecer uma chamada a um assinante móvel.

Endpoint: `POST /api/sri`

Corpo da Requisição:

```
{
  "msisdn": "1234567890",
  "gmsc": "5551234567"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`msisdn`	String	Sim	MSISDN da parte chamada
`gmsc`	String	Sim	Título Global do MSC Gateway

Resposta (200 OK):

```
{
  "result": {
    "imsi": "001001234567890",
    "msrn": "5551234999",
    "vlr_number": "5551234800",
    ...
  }
}
```

Erro (504 Gateway Timeout):

```
{
  "error": "timeout"
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/sri \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "1234567890",
    "gmsc": "5551234567"
  }'
```

SendRoutingInfoForSM (SRI-for-SM)

Recupere informações de roteamento para entregar um SMS a um assinante móvel.

Endpoint: `POST /api/sri-for-sm`

Corpo da Requisição:

```
{
  "msisdn": "1234567890",
  "service_center": "5551234567"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`msisdn`	String	Sim	MSISDN de destino
`service_center`	String	Sim	Título Global do Centro de Serviço

Resposta (200 OK):

```
{
  "result": {
    "imsi": "001001234567890",
    "msc_number": "5551234800",
    "location_info": {...},
    ...
  }
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "1234567890",
    "service_center": "5551234567"
  }'
```

SendAuthenticationInfo

Solicite vetores de autenticação para um assinante.

Endpoint: `POST /api/send-auth-info`

Corpo da Requisição:

```
{
  "imsi": "001001234567890",
  "vectors": 3
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`imsi`	String	Sim	IMSI do assinante
`vectors`	Integer	Sim	Número de vetores de autenticação a serem gerados

Resposta (200 OK):

```
{
  "result": {
    "authentication_sets": [
      {
        "rand": "0123456789ABCDEF...",
        "xres": "...",
        "ck": "...",
        "ik": "...",
        "autn": "..."
      }
    ],
    ...
  }
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/send-auth-info \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vectors": 3
  }'
```

MT-ForwardSM

Entregue um SMS Terminado para Móvel a um assinante.

Endpoint: `POST /api/MT-forwardSM`

Corpo da Requisição:

```
{
  "imsi": "001001234567890",
  "destination_service_centre": "5551234567",
  "originating_service_center": "5551234568",
  "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`imsi`	String	Sim	IMSI do assinante de destino
`destination_service_centre`	String	Sim	GT do centro de serviço de destino
`originating_service_center`	String	Sim	GT do centro de serviço de origem
`smsPDU`	String	Sim	SMS TPDU em formato hexadecimal

Nota: `smsPDU` deve ser uma string codificada em hex (maiúsculas ou minúsculas).

Resposta (200 OK):

```
{
  "result": {
    "delivery_status": "success",
    ...
  }
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/MT-forwardSM \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "destination_service_centre": "5551234567",
    "originating_service_center": "5551234568",
    "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
  }'
```

ForwardSM

Encaminhe uma mensagem SMS (MO-SMS do assinante).

Endpoint: `POST /api/forwardSM`

Corpo da Requisição: Mesmo que MT-ForwardSM

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/forwardSM \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "destination_service_centre": "5551234567",
    "originating_service_center": "5551234568",
    "smsPDU": "0001000A8121436587F900001C48656C6C6F20576F726C64"
  }'
```

UpdateLocation

Notifique o HLR sobre a mudança de localização do assinante (registro VLR).

Endpoint: `POST /api/updateLocation`

Corpo da Requisição:

```
{
  "imsi": "001001234567890",
  "vlr": "5551234800"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`imsi`	String	Sim	IMSI do assinante
`vlr`	String	Sim	Endereço do Título Global do VLR

Resposta (200 OK):

```
{
  "result": {
    "hlr_number": "5551234567",
    "subscriber_data": {...},
    ...
  }
}
```

Nota: No modo HLR, isso aciona a sequência InsertSubscriberData (ISD) com timeout de 10 segundos por ISD.

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/updateLocation \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vlr": "5551234800"
  }'
```

ProvideRoamingNumber (PRN)

Solicite MSRN (Número de Roaming da Estação Móvel) para roteamento de chamadas para assinantes em roaming.

Endpoint: `POST /api/prn`

Corpo da Requisição:

```
{
  "msisdn": "1234567890",
  "gmsc": "5551234567",
  "msc_number": "5551234800",
  "imsi": "001001234567890"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`msisdn`	String	Sim	MSISDN do assinante
`gmsc`	String	Sim	GT do MSC Gateway
`msc_number`	String	Sim	Número do MSC para o assinante
`imsi`	String	Sim	IMSI do assinante

Resposta (200 OK):

```
{
  "result": {
    "msrn": "5551234999",
    ...
  }
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/prn \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "1234567890",
    "gmsc": "5551234567",
    "msc_number": "5551234800",
    "imsi": "001001234567890"
  }'
```

USSD Send (Originado na Rede)

Envie uma mensagem USSD para um assinante. Requer `ussd_gateway_enabled: true`. Para a documentação completa do USSD Gateway, incluindo protocolo de callback e ciclo de vida da sessão, consulte o Guia do USSD Gateway.

Endpoint: `POST /api/ussd/send`

Corpo da Requisição:

```
{
  "msisdn": "+254712345678",
  "text": "Você tem uma fatura pendente. Responda 1 para pagar.",
  "callback_url": "http://billing-app:9000/ussd"
}
```

Parâmetros:

Campo	Tipo	Necessário	Descrição
`msisdn`	String	Sim	MSISDN do assinante de destino
`text`	String	Sim	Texto USSD a ser exibido (apenas caracteres GSM de 7 bits)
`callback_url`	String	Sim	URL para receber respostas do assinante via HTTP POST

Resposta (200 OK):

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "status": "sent"
}
```

Erro (503 Serviço Indisponível):

```
{
  "error": "USSD gateway not enabled"
}
```

Exemplo cURL:

```
curl -X POST http://localhost:8080/api/ussd/send \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "+254712345678",
    "text": "Você tem uma fatura pendente. Responda 1 para
pagar.",
    "callback_url": "http://billing-app:9000/ussd"
  }'
```

Autenticação

Status Atual: A API **não requer autenticação**.

Considerações de Segurança:

- A API é destinada ao uso em rede interna/confiável
- Considere usar regras de firewall para restringir o acesso
- Para implantações em produção, considere implementar middleware de autenticação

---

Formatos de Resposta

Todas as respostas usam o formato **JSON**.

Resposta de Sucesso

Status HTTP: 200 OK

Estrutura:

```
{
  "result": {
    // Dados de resposta específicos da operação
  }
}
```

Resposta de Erro

Status HTTP:

- 400 Bad Request - Corpo da requisição inválido
- 504 Gateway Timeout - Timeout da requisição MAP (10 segundos)
- 404 Not Found - Endpoint inválido

Estrutura:

```
{
  "error": "timeout"
}
```

ou

```
{
  "error": "invalid request"
}
```

Tratamento de Erros

Erros Comuns

Erro	Código HTTP	Descrição	Solução
JSON Inválido	400	O corpo da requisição não é um JSON válido	Verifique a sintaxe do JSON
Campos Ausentes	400	Campos obrigatórios ausentes	Inclua todos os parâmetros obrigatórios
Timeout	504	A requisição MAP excedeu o timeout de 10s	Verifique a conectividade M3UA, disponibilidade do HLR/VLR
Não Encontrado	404	Endpoint inválido	Verifique a URL do endpoint

Comportamento de Timeout

Todas as requisições MAP têm um **timeout de 10 segundos codificado**:

1. Requisição enviada ao MapClient GenServer
2. Aguarda resposta por até 10 segundos
3. Se não houver resposta → retorna 504 Gateway Timeout
4. Se resposta recebida → retorna 200 OK com resultado

Solução de Problemas de Timeout:

- Verifique o status da conexão M3UA (Web UI → página M3UA)
- Verifique se o elemento de rede (HLR/VLR/MSC) é acessível
- Verifique a configuração de roteamento
- Revise os logs de eventos SS7 em busca de erros

Métricas (Prometheus)

A API expõe métricas Prometheus para monitoramento.

Endpoint de Métricas

URL: `http://[server-ip]:8080/metrics`

Formato: Formato de texto Prometheus

Exemplo de Saída:

```
# HELP map_requests_total Total de requisições MAP
# TYPE map_requests_total counter
map_requests_total{operation="sri"} 42
map_requests_total{operation="sri_for_sm"} 158
map_requests_total{operation="updateLocation"} 23

# HELP cap_requests_total Total de requisições CAP
# TYPE cap_requests_total counter
cap_requests_total{operation="initialDP"} 87
cap_requests_total{operation="requestReportBCSMEvent"} 91

# HELP map_request_duration_milliseconds Duração das
requisições/respostas MAP em ms
# TYPE map_request_duration_milliseconds histogram
map_request_duration_milliseconds_bucket{operation="sri",le="10"}
5
map_request_duration_milliseconds_bucket{operation="sri",le="50"}
12
map_request_duration_milliseconds_bucket{operation="sri",le="100"}
35
...

# HELP map_pending_requests Número de requisições MAP pendentes
# TYPE map_pending_requests gauge
map_pending_requests 3

# HELP omniss7_license_status Status atual da licença (1 = válido,
0 = inválido)
# TYPE omniss7_license_status gauge
omniss7_license_status 1
```

Métricas Disponíveis

Métrica	Tipo	Rótulos	Descri
`map_requests_total`	Counter	`operation`	Total de requisiçõe MAP por t de operaç
`cap_requests_total`	Counter	`operation`	Total de requisiçõe CAP por ti de operaç
`map_request_duration_milliseconds`	Histogram	`operation`	Duração d requisiçã milissegu
`map_pending_requests`	Gauge	-	Número d transaçõe MAP pendente
`ussd_requests_total`	Counter	`direction`	Total de requisiçõe USSD (entrada/s
`ussd_active_sessions`	Gauge	-	Número d sessões U ativas
`omniss7_license_status`	Gauge	-	Status atu licença (1 válido, 0 = inválido)

Configuração do Prometheus

Adicione ao seu `prometheus.yml`:

```yaml
scrape_configs:
  - job_name: 'omniss7'
    static_configs:
      - targets: ['server-ip:8080']
    metrics_path: '/metrics'
    scrape_interval: 15s
```

Exemplos de Requisições

Exemplo em Python

```python
import requests
import json

# Requisição SRI-for-SM
url = "http://localhost:8080/api/sri-for-sm"
payload = {
    "msisdn": "1234567890",
    "service_center": "5551234567"
}

response = requests.post(url, json=payload, timeout=15)

if response.status_code == 200:
    result = response.json()
    print(f"Sucesso: {result}")
elif response.status_code == 504:
    print("Timeout - sem resposta da rede")
else:
    print(f"Erro: {response.status_code} - {response.text}")
```

Exemplo em JavaScript

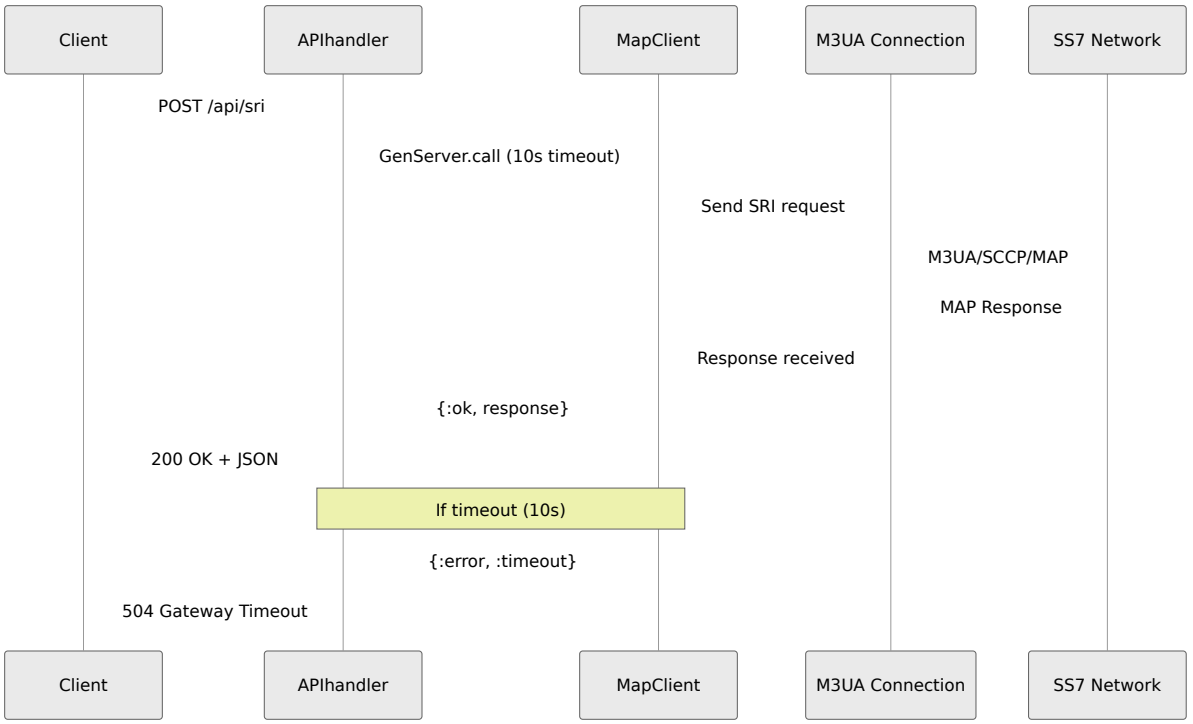
```
const axios = require('axios');

async function sendSRI() {
  try {
    const response = await
axios.post('http://localhost:8080/api/sri', {
      msisdn: '1234567890',
      gmsc: '5551234567'
    }, {
      timeout: 15000
    });

    console.log('Sucesso:', response.data);
  } catch (error) {
    if (error.code === 'ECONNABORTED') {
      console.error('Timeout - sem resposta da rede');
    } else {
      console.error('Erro:', error.response?.data ||
error.message);
    }
  }
}

sendSRI();
```

Exemplo em Bash/cURL

```
#!/bin/bash

# Requisição UpdateLocation
response=$(curl -s -w "\n%{http_code}" -X POST
http://localhost:8080/api/updateLocation \
  -H "Content-Type: application/json" \
  -d '{
    "imsi": "001001234567890",
    "vlr": "5551234800"
  }')

http_code=$(echo "$response" | tail -n 1)
body=$(echo "$response" | sed '$d')

if [ "$http_code" -eq 200 ]; then
  echo "Sucesso: $body"
elif [ "$http_code" -eq 504 ]; then
  echo "Timeout - sem resposta da rede"
else
  echo "Erro $http_code: $body"
fi
```

Diagramas de Fluxo

Fluxo de Requisição da API



Resumo

A API REST do OmniSS7 fornece:

 Operações MAP - Suporte completo para SRI, SRI-for-SM, UpdateLocation, entrega de SMS, autenticação
 Swagger UI - Documentação interativa da API e testes
 Métricas Prometheus - Monitoramento e observabilidade
 Timeouts Codificados - Timeout de 10 segundos para todas as requisições MAP
 Servidor HTTP - Executa na porta 8080 (configurável via `start_http_server`)

Para acesso à Web UI, consulte o Guia da Web UI.

Para detalhes de configuração, consulte a Referência de Configuração.

Referência Técnica (Apêndice)

← Voltar para a Documentação Principal

Referência técnica para protocolos SS7 e implementação do OmniSS7.

Pilha de Protocolos SS7

Códigos de Operação MAP

Operação	Opcode	Propósito
updateLocation	2	Registrar localização do assinante
cancelLocation	3	Cancelar registro no VLR
provideRoamingNumber	4	Solicitar MSRN
sendRoutingInfo	22	Consultar roteamento de chamadas
mt-forwardSM	44	Entregar SMS ao assinante
sendRoutingInfoForSM	45	Consultar roteamento de SMS
mo-forwardSM	46	Encaminhar SMS do assinante
sendAuthenticationInfo	56	Solicitar vetores de autenticação

Tipos de Mensagens TCAP

- **BEGIN** - Iniciar transação
- **CONTINUE** - Meio da transação
- **END** - Resposta final
- **ABORT** - Cancelar transação

Endereçamento SCCP

Formatos de Título Global

- **E.164** - Número de telefone internacional (ex.: 447712345678)
- **E.212** - Formato IMSI (ex.: 234509876543210)
- **E.214** - Formato de código de ponto

Números de Subsistema (SSN)

- **SSN 6**: HLR
- **SSN 7**: VLR
- **SSN 8**: MSC/SMSC
- **SSN 9**: GMLC
- **SSN 10**: SGSN

SMS TPDU

Tipos de Mensagens

- **SMS-DELIVER** (MT) - Rede para móvel
- **SMS-SUBMIT** (MO) - Móvel para rede

- **SMS-STATUS-REPORT** - Status de entrega
- **SMS-COMMAND** - Comando remoto

Codificações de Caracteres

- **GSM7** - Alfabeto GSM de 7 bits (160 caracteres por SMS)
- **UCS2** - Unicode de 16 bits (70 caracteres por SMS)
- **8-bit** - Dados binários (140 bytes por SMS)

Estados M3UA

- **DOWN** - Sem conexão SCTP
- **CONNECTING** - SCTP conectando
- **ASPUP_SENT** - Aguardando ACK de ASPUP
- **INACTIVE** - ASP ativo, mas não em uso
- **ASPAC_SENT** - Aguardando ACK de ASPAC
- **ACTIVE** - Pronto para tráfego

Códigos de Ponto Comum SS7

Os códigos de ponto são tipicamente valores de 14 bits (ITU) ou 24 bits (ANSI).

Formato de Exemplo (ITU):

- Rede: 3 bits
- Cluster: 8 bits
- Membro: 3 bits

Códigos de Erro SCCP

- **0** - Sem tradução para o endereço

- **1** - Sem tradução para endereço específico
- **2** - Congestionamento do subsistema
- **3** - Falha do subsistema
- **4** - Usuário não equipado
- **5** - Falha do MTP
- **6** - Congestionamento da rede
- **7** - Não qualificado
- **8** - Erro no transporte da mensagem

Códigos de Erro MAP

Código	Erro	Descrição
1	unknownSubscriber	Assinante não encontrado no HLR
27	absentSubscriber	Assinante não acessível
34	systemFailure	Falha na rede
35	dataMissing	Dados necessários não disponíveis
36	unexpectedDataValue	Valor de parâmetro inválido

Documentação Relacionada

- ← Voltar para a Documentação Principal
- Guia STP
- Guia do Cliente MAP
- Guia do Centro de SMS
- Guia HLR
- Recursos Comuns

Guia de Configuração do CAMEL Gateway

Visão Geral

O **modo CAMEL Gateway (CAMELGW)** transforma o OmniSS7 em uma plataforma de Rede Inteligente (IN) que fornece serviços de controle de chamadas e cobrança em tempo real usando o protocolo CAMEL Application Part (CAP).

O que é CAMEL?

CAMEL (Aplicações Personalizadas para Lógica Aprimorada de Redes Móveis) é um conjunto de padrões projetados para funcionar em uma rede central GSM ou rede UMTS. Ele permite que os operadores forneçam serviços que requerem controle em tempo real das chamadas, como:

- **Chamadas pré-pagas** - Verificação e cobrança de saldo em tempo real
- **Serviços de tarifa premium** - Cobrança especial para serviços de valor agregado
- **Controle de roteamento de chamadas** - Roteamento dinâmico de destinos com base em tempo/localização
- **Redes privadas virtuais** - Planos de numeração corporativa
- **Filtragem de chamadas** - Permitir/bloquear chamadas com base em critérios

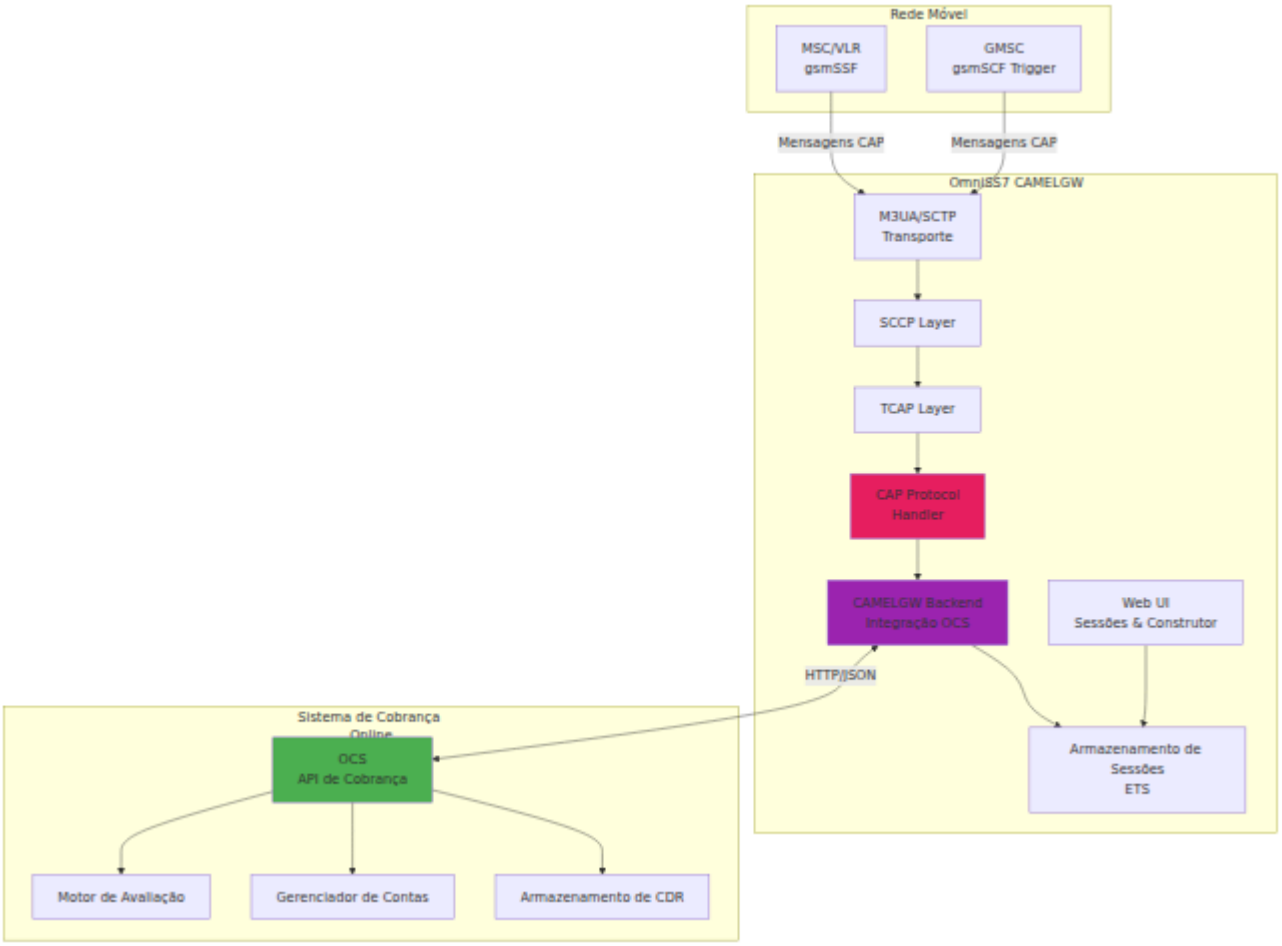
Versões do Protocolo CAP

O OmniSS7 CAMELGW suporta várias versões do CAP:

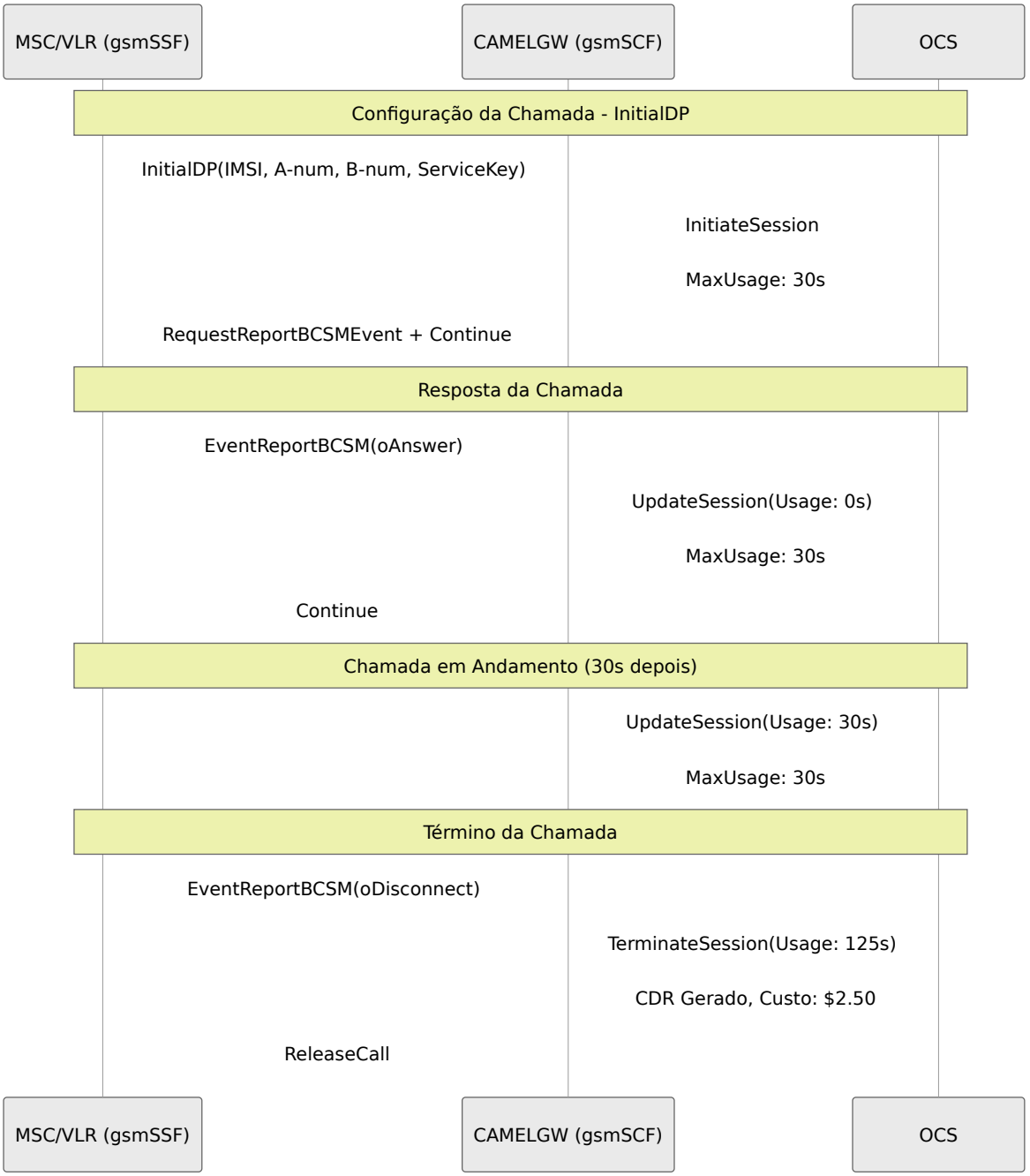
Versão	Fase	Recursos
CAP v1	CAMEL Fase 1	Controle básico de chamadas, operações limitadas
CAP v2	CAMEL Fase 2	Operações aprimoradas, suporte a SMS
CAP v3	CAMEL Fase 3	Suporte a GPRS, operações adicionais
CAP v4	CAMEL Fase 4	Recursos avançados, suporte a multimídia

Padrão: CAP v2 (mais amplamente implantado)

Arquitetura



Exemplo de Fluxo de Chamadas



Configuração

Pré-requisitos

- OmniSS7 instalado e em execução
- Conectividade M3UA com MSC/GMSC (gsmSSF)

- Sistema de Cobrança Online (OCS) com endpoint API (opcional, para cobrança em tempo real)

Habilitar o Modo CAMEL Gateway

Edite `config/runtime.exs` e configure a seção do CAMEL Gateway:

```
config :omniss7,
  # Flags de modo - Habilitar recursos CAP/CAMEL
  cap_client_enabled: true,
  camelgw_mode_enabled: true,

  # Desabilitar outros modos
  map_client_enabled: false,
  hlr_mode_enabled: false,
  smsc_mode_enabled: false,

  # Configuração da Versão CAP/CAMEL
  # Determina qual versão CAP usar para solicitações e diálogos de
saída
  # Opções: :v1, :v2, :v3, :v4
  cap_version: :v2,

  # Integração OCS (para cobrança em tempo real)
  ocs_enabled: true,
  ocs_url: "http://your-ocs-server/api/charging",
  ocs_timeout: 5000,  # milissegundos
  ocs_auth_token: "your-api-token"  # Opcional, se OCS requer
autenticação

  # Configuração de Conexão M3UA para CAMEL
  # Conectar como ASP (Application Server Process) para operações
CAP
  cap_client_m3ua: %{
    mode: "ASP",
    callback: {CapClient, :handle_payload, []},
    process_name: :camelgw_client_asp,

    # Endpoint local (sistema CAMELGW)
    local_ip: {10, 179, 4, 13},
    local_port: 2905,

    # Endpoint remoto (MSC/GMSC - gsmSSF)
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,

    # Parâmetros M3UA
    routing_context: 1,
    network_appearance: 0,
```

```
    asp_identifier: 13
  }
```

Configurar Páginas da Web UI

A Web UI inclui páginas especializadas para operações CAMEL:

```
config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Eventos SS7"},
    {SS7.Web.TestClientLive, "/client", "Cliente SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
    {SS7.Web.CAMELSessionsLive, "/camel_sessions", "Sessões CAP"},
    {SS7.Web.CAMELRequestLive, "/camel_request", "Solicitações
CAP"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/camel_sessions",
               "/camel_request", "/application", "/configuration"]
```

Operações CAP Suportadas

Operações de Entrada (de gsmSSF → gsmSCF)

Operação	Opcode	Descrição	Man
InitialDP	0	Ponto de Detecção Inicial - notificação de configuração de chamada	`handle_initial_d`
EventReportBCSM	6	Evento do Modelo de Estado de Chamada básico (resposta, desconexão, etc.)	`handle_event_rep`
ApplyChargingReport	71	Relatório de cobrança do gsmSSF	`handle_apply_cha`
AssistRequestInstructions	16	Solicitação de assistência do gsmSRF	`handle_assist_re`

Operações de Saída (de gsmSCF → gsmSSF)

Operação	Opcode	Descrição	
Connect	20	Conectar chamada ao número de destino	`CapRequestGer`
Continue	31	Continuar o processamento da chamada sem modificação	`CapRequestGer`
ReleaseCall	22	Liberar/terminar a chamada	`CapRequestGer`
RequestReportBCSMEvent	23	Solicitar notificação de eventos de chamada	`CapRequestGer`
ApplyCharging	35	Aplicar cobrança à chamada	`CapRequestGer`

Recursos da Web UI

Página de Sessões CAMEL

URL: `http://localhost/camel_sessions`

Monitoramento em tempo real de sessões de chamadas CAMEL ativas:

Recursos:

- **Lista de sessões ao vivo** - Atualiza automaticamente a cada 2 segundos
- **Detalhes da sessão** - OTID, ID da Chamada, Estado, Duração
- **Versão CAP** - Exibe a versão do protocolo (CAP v1/v2/v3/v4) detectada a partir do InitialDP
- **Informações da chamada** - IMSI, A-número, B-número, Chave de Serviço
- **Rastreamento de estado** - Iniciada, Respondida, Terminada
- **Cronômetro de duração** - Exibição em tempo real da duração da chamada

Colunas da Tabela:

- ID da Chamada, Estado, Versão, IMSI, Número Chamante, Número Chamado, Chave de Serviço, Duração, Hora de Início, OTID

Estados da Sessão:

- □ **Iniciada** - InitialDP recebido, aguardando resposta
- □ **Respondida** - Chamada respondida, cobrança em andamento
- □ **Terminada** - Chamada encerrada, CDR gerado

Detecção da Versão CAP: O sistema detecta automaticamente a versão do protocolo CAP a partir da parte do diálogo do InitialDP e a exibe na coluna Versão. Isso ajuda a identificar qual versão CAP cada MSC está usando.

Construtor de Solicitações CAMEL

URL: `http://localhost/camel_request`

Ferramenta interativa para construir e enviar solicitações CAP:

Recursos:

- **Selector de tipo de solicitação** - InitialDP, Connect, ReleaseCall, etc.
- **Campos de formulário dinâmicos** - Adapta-se ao tipo de solicitação selecionado
- **Opções SCCP/M3UA** - Configuração avançada de endereçamento

- **Histórico de solicitações** - Últimas 20 solicitações com status
- **Rastreamento de sessão** - Mantém OTID para solicitações de acompanhamento
- **Feedback em tempo real** - Mensagens de sucesso/erro

Tipos de Solicitações:

1. **InitialDP** - Iniciar nova sessão de chamada
    - Chave de Serviço (inteiro)
    - Número Chamante (A-party)
    - Número Chamado (B-party)
2. **Connect** - Roteamento da chamada para o destino
    - Número de Destino
3. **ReleaseCall** - Terminar chamada
    - Código de Causa (16=Normal, 17=Ocupado, 31=Não Especificado)
4. **RequestReportBCSMEvent** - Solicitar notificações de eventos
    - Eventos: oAnswer, oDisconnect, tAnswer, tDisconnect
5. **Continue** - Continuar chamada sem modificação
    - Nenhum parâmetro necessário
6. **ApplyCharging** - Aplicar limites de duração da chamada
    - Duração (segundos, 1-864000)
    - Liberar em Timeout (booleano)
    - Veja Guia do Construtor de Solicitações CAMEL para uso detalhado

Opções SCCP Avançadas:

- Título Global da Parte Chamado
- Título Global da Parte Chamante
- SSN Chamado (padrão: 146 = gsmSSF)

- SSN Chamante (padrão: 146)

Opções M3UA:

- OPC (Código de Ponto de Origem, padrão: 5013)
- DPC (Código de Ponto de Destino, padrão: 5011)

Integração com OCS

Ciclo de Vida da Chamada com Cobrança

1. Iniciação da Chamada (InitialDP)

Quando o MSC envia o InitialDP, o CAMELGW:

1. **Detecta a versão CAP** - Examina a parte do diálogo para identificar CAP v1/v2/v3/v4
2. **Decodifica a mensagem CAP** - Extrai IMSI, números chamantes/chamados
3. **Chama o OCS** - API `InitiateSession`
4. **Recebe autorização** - MaxUsage (por exemplo, 30 segundos)
5. **Armazena a sessão** - No SessionStore (tabela ETS) com a versão CAP
6. **Responde ao MSC** - RequestReportBCSMEvent + Continue (usando a mesma versão CAP)

Exemplo:

```
# Dados decodificados do InitialDP
%{
  imsi: "310150123456789",
  calling_party_number: "14155551234",
  called_party_number: "14155556789",
  service_key: 1,
  msc_address: "19216800123",
  cap_version: :v2  # Detectado a partir do diálogo
}

# Resposta do OCS
{:ok, %{max_usage: 30}}  # 30 segundos autorizados

# Entrada no SessionStore
%{
  call_id: "CAMEL-4B000173",
  initial_dp_data: %{...},
  cap_version: :v2,  # Armazenado para geração de resposta
  start_time: 1730246400,
  state: :initiated
}
```

2. Resposta da Chamada (EventReportBCSM - oAnswer)

Quando a chamada é respondida:

1. **Recebe o evento oAnswer** - Do MSC
2. **Atualiza o OCS** - `UpdateSession` com uso=0
3. **Inicia o loop de débito** - OCS começa a cobrança
4. **Atualiza o estado da sessão** - `:answered` no SessionStore
5. **Continua a chamada** - Envia Continue para o MSC

3. Atualizações Periódicas (Opcional)

Para chamadas longas, solicitar crédito adicional:

```
# A cada 30 segundos
OCS.Client.update_session(call_id, %{}, current_usage)
```

Se MaxUsage retornar 0, o assinante não tem crédito → Enviar ReleaseCall

4. Término da Chamada (EventReportBCSM - oDisconnect)

Quando a chamada termina:

1. **Recebe o evento oDisconnect** - Do MSC
2. **Calcula a duração total** - A partir da hora de início da sessão
3. **Termina a sessão OCS** - API `TerminateSession`
4. **CDR gerado** - Pelo OCS com custo final
5. **Limpa a sessão** - Remove do SessionStore
6. **Envia ReleaseCall** - Confirma a terminação para o MSC

Análise de CDR

Os CDRs são gerados pelo seu OCS e normalmente incluem:

Campos de CDR do CAMEL:

- `Account` - IMSI ou número chamante
- `Destination` - Número da parte chamada
- `OriginID` - Identificador único da chamada (CAMEL-OTID)
- `Usage` - Duração total da chamada (segundos)
- `Cost` - Custo calculado
- `IMSI` - IMSI do assinante
- `CallingPartyNumber` - A-party
- `CalledPartyNumber` - B-party
- `MSCAddress` - Código de ponto MSC atendente
- `ServiceKey` - Chave de serviço CAMEL

Testes

Teste Manual com o Construtor de Solicitações

1. **Navegue até o Construtor de Solicitações:**

   ```
   http://localhost/camel_request
   ```

2. **Envie InitialDP:**

   - Selecione "InitialDP" no dropdown
   - Chave de Serviço: `100`
   - Número Chamante: `14155551234`
   - Número Chamado: `14155556789`
   - Clique em "Enviar Solicitação InitialDP"
   - Anote o OTID gerado

3. **Monitore a Sessão:**

   - Abra uma nova aba: `http://localhost/camel_sessions`
   - Veja a sessão ativa com estado "Iniciada"

4. **Simule a Resposta da Chamada:**

   - Volte ao Construtor de Solicitações
   - Selecione "EventReportBCSM"
   - Tipo de Evento: `oAnswer`
   - Clique em "Enviar Solicitação EventReportBCSM"
   - O estado da sessão muda para "Respondida"

5. **Finalizar Chamada:**

   - Selecione "ReleaseCall"
   - Código de Causa: `16` (Normal)
   - Clique em "Enviar Solicitação ReleaseCall"
   - O estado da sessão muda para "Terminada"

Teste com MSC Real

Configure o Serviço CAMEL no MSC

No seu MSC/VLR, configure o serviço CAMEL:

```
# Exemplo de configuração do MSC Huawei
ADD CAMELSERVICE:
  SERVICEID=1,
  SERVICEKEY=100,
  GSMSCFADDR="55512341234",  # Título Global do CAMELGW
  DEFAULTCALLHANDLING=CONTINUE;

ADD CAMELSUBSCRIBER:
  IMSI="310150123456789",
  SERVICEID=1,
  TRIGGERTYPE=TERMCALL;
```

Monitorar Logs

Assista aos logs do CAMELGW para mensagens CAP recebidas:

```
# Ver logs em tempo real
tail -f /var/log/omniss7/omniss7.log

# Filtrar eventos CAP
grep "CAP:" /var/log/omniss7/omniss7.log

# Ver log de eventos (formatado em JSON)
curl http://localhost/api/events | jq '.[] | select(.map_event |
startswith("CAP:"))'
```

Teste de Carga

Use o Construtor de Solicitações em um loop para teste de carga:

```
# Enviar 100 solicitações InitialDP
for i in {1..100}; do
  curl -X POST http://localhost/api/camel/initial_dp \
    -H "Content-Type: application/json" \
    -d '{
      "service_key": 100,
      "calling_number": "1415555'$i'",
      "called_number": "14155556789"
    }'
  sleep 0.1
done
```

Monitoramento & Operações

Métricas do Prometheus

O CAMELGW expõe métricas em `http://localhost:8080/metrics`:

Métricas específicas do CAP:

- `cap_requests_total{operation}` - Total de solicitações CAP por tipo de operação (por exemplo, initialDP, requestReportBCSMEvent)

Métricas adicionais de MAP/API:

- `map_requests_total{operation}` - Total de solicitações MAP por tipo de operação
- `map_request_duration_milliseconds{operation}` - Histograma de duração das solicitações
- `map_pending_requests` - Número de transações MAP pendentes

Métricas STP do M3UA (se o modo STP estiver habilitado):

- `m3ua_stp_messages_received_total{peer_name,point_code}` - Mensagens recebidas de pares

- `m3ua_stp_messages_sent_total{peer_name,point_code}` - Mensagens enviadas para pares
- `m3ua_stp_routing_failures_total{reason}` - Falhas de roteamento por motivo

Consultas de exemplo:

```
# Solicitações CAP
curl http://localhost:8080/metrics | grep cap_requests_total

# Total de InitialDP recebidos
curl http://localhost:8080/metrics | grep
'cap_requests_total{operation="initialDP"}'

# Solicitações MAP pendentes
curl http://localhost:8080/metrics | grep map_pending_requests
```

Verificações de Saúde

```
# Verificar conectividade M3UA
curl http://localhost/api/m3ua-status

# Verificar conectividade OCS
curl http://localhost/api/ocs-status

# Verificar sessões ativas
curl http://localhost/api/camel/sessions/count
```

Configuração de Logs

Ajuste o nível de log em `config/runtime.exs`:

```
config :logger,
  level: :info  # Opções: :debug, :info, :warning, :error

# Habilitar log de depuração CAP
config :logger, :console,
  metadata: [:cap_operation, :otid, :call_id]
```

Solução de Problemas

Problema: Nenhuma mensagem CAP recebida

Sintomas: O Construtor de Solicitações funciona, mas o MSC não envia InitialDP

Verifique:

1. Status do link M3UA: `curl http://localhost/api/m3ua-status`
2. Configuração do serviço CAMEL no MSC (Chave de Serviço, endereço gsmSCF)
3. Roteamento SCCP (Título Global deve roteá para o CAMELGW)
4. Regras de firewall (permitir porta SCTP 2905)

Solução:

```
# Verificar conectividade M3UA
tcpdump -i eth0 sctp

# Verificar se o MSC pode alcançar o CAMELGW
ss -tuln | grep 2905
```

Problema: Erros no OCS

Sintomas: `INSUFFICIENT_CREDIT` ou erros de timeout

Verifique:

1. O OCS é acessível: `curl http://your-ocs-server/api/health`
2. A conta tem saldo no OCS
3. Plano de avaliação configurado no OCS
4. Conectividade de rede com o OCS
5. O token de autenticação é válido (se necessário)

Solução:

- Verifique a configuração da URL do OCS em `runtime.exs`
- Verifique os logs do OCS para erros
- Teste a API do OCS manualmente com curl
- Verifique se as regras de firewall permitem conectividade

Problema: Sessão não encontrada

Sintomas: EventReportBCSM falha com "Sessão não encontrada"

Causa: Desvio de OTID ou sessão expirada

Solução:

1. Verifique o OTID nos logs
2. Verifique o tempo limite da sessão (padrão: sem expiração)
3. Certifique-se de que o DTID corresponda ao OTID nas mensagens Continue/End

```
# Verificar sessões ativas
iex> CAMELGW.SessionStore.list_sessions()
```

Problema: Erros de decodificação

Sintomas: `Falha ao decodificar InitialDP` nos logs

Causa: Desvio de versão CAP ou mensagem malformada

Solução:

1. Verifique se a configuração da versão CAP corresponde ao MSC
2. Verifique se a codificação ASN.1 está correta
3. Capture PCAP e analise com Wireshark

```
# Capturar mensagens CAP
tcpdump -i eth0 -w cap_trace.pcap sctp port 2905

# Analisar com Wireshark (filtro: m3ua)
wireshark cap_trace.pcap
```

Configuração Avançada

Múltiplas Versões CAP

Suporte a diferentes versões CAP por chave de serviço:

```
config :omniss7,
  cap_version_map: %{
    100 => :v2,  # Chave de Serviço 100 usa CAP v2
    200 => :v3,  # Chave de Serviço 200 usa CAP v3
    300 => :v4   # Chave de Serviço 300 usa CAP v4
  },
  cap_version: :v2  # Padrão
```

Resumo

O modo CAMEL Gateway permite que o OmniSS7 funcione como uma plataforma completa de Rede Inteligente com:

 Suporte completo ao protocolo CAP (v1/v2/v3/v4)
 Cobrança em tempo real via integração OCS
 Operações de controle de chamadas (Conectar, Liberar, Continuar)
 Gerenciamento de sessões com armazenamento ETS

 Teste interativo via Construtor de Solicitações da Web UI
 Monitoramento ao vivo de sessões de chamadas ativas
 Geração de CDR para faturamento e análises
 Desempenho e confiabilidade prontos para produção

Para informações adicionais:

- Documentação do Construtor de Solicitações CAMEL
- Referência Técnica - Operações CAP

Produto: OmniSS7 CAMEL Gateway
Versão da Documentação: 1.0
Última Atualização: 2025-10-26

CAMEL Request Builder - Resumo da Implementação

Visão Geral

Um novo componente LiveView foi criado para construir e enviar solicitações CAMEL/CAP para fins de teste. Isso fornece uma interface de usuário interativa para criar InitialDP e outras operações CAMEL.

Novos Componentes

1. CAMEL Request Builder LiveView

Recursos:

- Interface de usuário interativa baseada em formulário para construir solicitações CAMEL
- Suporte para múltiplos tipos de solicitação:
  - **InitialDP** - Ponto de Detecção Inicial (notificação de configuração de chamada)
  - **Connect** - Conectar chamada ao destino
  - **ReleaseCall** - Liberar/terminar chamada
  - **RequestReportBCSMEvent** - Solicitar notificações de eventos
  - **Continue** - Continuar o processamento da chamada
  - **ApplyCharging** - Aplicar limites de cobrança/duração às chamadas

Principais Capacidades:

- Menu suspenso para seleção do tipo de solicitação
- Campos de formulário dinâmicos com base no tipo de solicitaç��o selecionado
- Opções avançadas de SCCP/M3UA (seção colapsável)
  - Títulos Globais da Parte Chamadora/Chamado
  - Configuração de SSN (Número do Subsistema)
  - Configurações de OPC/DPC (Código de Ponto)
- Histórico de solicitações em tempo real (últimas 20 solicitações)
- Rastreamento de sessão via OTID
- Feedback de sucesso/erro
- Rastreamento do tamanho da solicitação

Rota: `/camel_request`

2. EventLog Aprimorado com Suporte a CAMEL

Novas Funções:

- `paklog_camel/2` - Registro dedicado de mensagens CAMEL/CAP
- `lookup_cap_opcode_name/1` - Pesquisa de código de operação CAP
- `find_cap_opcode/1` - Extrair opcode CAP de JSON
- `extract_cap_tids/1` - Extrair OTID/DTID de mensagens CAP
- `format_cap_to_json/1` - Converter PDUs CAP para formato JSON

Códigos de Operação CAP Suportados:

```
0  => "initialDP"
5  => "connect"
6  => "releaseCall"
7  => "requestReportBCSMEvent"
8  => "eventReportBCSM"
10 => "continue"
13 => "furnishChargingInformation"
35 => "applyCharging"
... (47 operações no total)
```

Recursos:

- Registro em JSON de todas as solicitações/respostas CAMEL
- Detecção automática de ação TCAP (Begin/Continue/End/Abort)
- Extração de endereçamento SCCP
- Tratamento de erros para mensagens malformadas
- Processamento de tarefas em segundo plano (não bloqueante)
- Evento prefixado com "CAP:" para fácil filtragem

3. CapClient Atualizado

Mudanças:

- Chamadas `paklog_camel/2` adicionadas para mensagens de entrada e saída

- Registro duplo: Tanto MAP (`paklog`) quanto CAP (`paklog_camel`) para compatibilidade
- Mensagens de saída registradas em `sccp_m3ua_maker/2`
- Mensagens de entrada registradas em `handle_payload/1`

Configuração

As novas páginas LiveView foram adicionadas à configuração em tempo de execução:

```
# Arquivo: config/runtime.exs

config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "SS7 Events"},
    {SS7.Web.TestClientLive, "/client", "SS7 Client"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
    {SS7.Web.HlrLinksLive, "/hlr_links", "HLR Links"},
    {SS7.Web.CAMELSessionsLive, "/camel_sessions", "CAMEL
Sessions"},
    {SS7.Web.CAMELRequestLive, "/camel_request", "CAMEL Request
Builder"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/hlr_links",
               "/camel_sessions", "/camel_request",
               "/application", "/configuration"]
```

Uso

Acessando o Construtor de Solicitações

1. Navegue até: `https://your-server:8087/camel_request`
2. Selecione o tipo de solicita��ão no menu suspenso
3. Preencha os parâmetros obrigatórios
4. Opcionalmente, expanda "Opções Avançadas de SCCP/M3UA" para ajustes finos

5. Clique em "Enviar Solicitação [RequestType]"

Fluxo de Solicitação

InitialDP (Nova Chamada)

1. Defina a Chave de Serviço (por exemplo, 100)
2. Defina o Número Chamador (A-Party)
3. Defina o Número Chamado (B-Party)
4. Envie a solicitação → Gera novo OTID
5. OTID armazenado na sessão para solicitações de acompanhamento

Solicitações de Acompanhamento (Connect, ReleaseCall, etc.)

1. Deve ter um OTID ativo do InitialDP
2. A solicitação usa automaticamente o OTID armazenado
3. Aviso exibido se não houver OTID ativo

Parâmetros da Solicitação

InitialDP:

- Chave de Serviço (inteiro)
- Número Chamador (formato ISDN)
- Número Chamado (formato ISDN)

Connect:

- Número de Destino (para onde direcionar a chamada)

ReleaseCall:

- Código de Causa (16 = Normal, 17 = Ocupado, 31 = Não Especificado)

RequestReportBCSMEvent:

- Eventos BCSM (separados por vírgula: oAnswer, oDisconnect, etc.)

Continue:

- Sem parâmetros (usa OTID ativo)

ApplyCharging:

- Duração (segundos, 1-864000) - Duração máxima da chamada antes da ação
- Liberar em Timeout (booleano) - Se deve liberar a chamada quando a duração expirar

Opções Avançadas

Endereçamento SCCP:

- GT da Parte Chamadora (Título Global)
- GT da Parte Chamado
- SSN Chamado (padrão 146 = gsmSSF)
- SSN Chamador (padrão 146)

Códigos de Ponto M3UA:

- OPC (Código de Ponto de Origem, padrão 5013)
- DPC (Código de Ponto de Destino, padrão 5011)

Registro em JSON

Todas as mensagens CAMEL agora são registradas em formato JSON no log de eventos com:

- **Direção:** entrada/saída
- **Ação TCAP:** Begin/Continue/End/Abort
- **Operação CAP:** e.g., "CAP:initialDP", "CAP:connect"
- **Endereçamento SCCP:** Informações da Parte Chamadora/Chamado
- **TIDs:** OTID/DTID para correlação
- **Mensagem Completa:** PDU CAP codificado em JSON

Exemplo de Entrada de Log


```
{
  "map_event": "CAP:initialDP",
  "direction": "outgoing",
  "tcap_action": "Begin",
  "otid": "A1B2C3D4",
  "sccp_called": {
    "SSN": 146,
    "GlobalTitle": {
      "Digits": "55512341234",
      "NumberingPlan": "isdn_tele",
      "NatureOfAddress_Indicator": "international"
    }
  },
  "event_message": "{ ... full CAP PDU ... }"
}
```


Histórico de Solicitações

A interface exibe as últimas 20 solicitações com:

- Timestamp
- Tipo de solicitação (com badge colorido)
- OTID (primeiros 8 caracteres hexadecimais)
- Status (enviado/erro)
- Tamanho da mensagem em bytes

Rastreamento de Sessão

Painel de Informações da Sessão Atual:

- Exibe OTID ativo
- Mostra o tamanho da última solicitação em bytes
- Visível apenas quando a sessão está ativa

Fluxo de Trabalho de Teste

1. **Iniciar Nova Chamada:**
    - Enviar InitialDP → Obter OTID
    - O sistema cria a sessão
2. **Controlar Chamada:**
    - Enviar RequestReportBCSMEvent → Solicitar notificações
    - Enviar ApplyCharging → Definir limite de duração da chamada (por exemplo, 290 segundos)
    - Enviar Connect → Roteamento para o destino
    - OU Enviar ReleaseCall → Terminar
3. **Ver Resultados:**
    - Verificar histórico de solicitações
    - Monitorar a página de Sessões CAMEL
    - Revisar logs de eventos com prefixo "CAP:"

ApplyCharging - Controle de Duração da Chamada

Visão Geral

A operação ApplyCharging permite definir uma duração máxima da chamada e, opcionalmente, liberar a chamada quando essa duração expirar. Isso é tipicamente usado para cenários de cobrança pré-paga ou para impor limites de tempo em chamadas.

Casos de Uso

- **Cobrança Pré-Paga**: Limitar a duração da chamada com base no saldo do assinante
- **Cobrança Baseada em Tempo**: Impor intervalos de cobrança periódicos

- **Gerenciamento de Recursos**: Impedir que chamadas sejam executadas indefinidamente
- **Integração OCS**: Coordenar com Sistemas de Cobrança Online para controle de crédito em tempo real

Parâmetros

Duração (maxCallPeriodDuration)

- **Tipo**: Inteiro (1-864000 segundos)
- **Descrição**: Número máximo de segundos que a chamada pode durar antes que o temporizador expire
- **Exemplos**:
  - `60` = 1 minuto
  - `290` = 4 minutos e 50 segundos (valor de teste comum)
  - `3600` = 1 hora
  - `86400` = 24 horas

Liberar em Timeout (releaseIfDurationExceeded)

- **Tipo**: Booleano (true/false)
- **Padrão**: true
- **Descrição**: O que acontece quando a duração expira:
  - `true`: Liberar/desconectar automaticamente a chamada
  - `false`: Enviar notificação, mas manter a chamada ativa (permite que gsmSCF tome uma ação)

Estrutura da Mensagem

A mensagem ApplyCharging é codificada como um TCAP Continue com:

- **TCAP**: Mensagem Continue (usa a transação existente)
- **Opcode**: 35 (applyCharging)
- **Parâmetros**: ApplyChargingArg contendo:
  - `aChBillingChargingCharacteristics`: Informações de cobrança baseadas em tempo

        - `timeDurationCharging`: Duração máxima e sinalizador de liberação
    - `partyToCharge`: Qual parte é cobrada (padrão: sendingSideID)

Exemplo de Uso

Cenário: Chamada pré-paga com limite de 5 minutos

1. Enviar **InitialDP** para iniciar o monitoramento da chamada

```
Chave de Serviço: 100
Chamador: 447700900123
Chamado: 447700900456
→ OTID: A1B2C3D4
```

2. Enviar **ApplyCharging** para definir o limite de 5 minutos

```
Duração: 300 (segundos)
Liberar em Timeout: true
→ Usa OTID: A1B2C3D4
```

3. Enviar **Connect** para completar a chamada

```
Destino: 447700900456
→ Usa OTID: A1B2C3D4
```

4. Após 5 minutos (300 segundos):
    - Chamada liberada automaticamente pela rede
    - gsmSCF recebe notificação de desconexão

Melhores Práticas

1. **Sempre envie ApplyCharging ANTES de Connect**
    - Garante que a cobrança esteja ativa quando a chamada se conecta
    - Impede segmentos de chamadas não cobrados

2. **Use com RequestReportBCSMEvent**
    - Solicitar eventos `oAnswer` e `oDisconnect`
    - Permite rastrear a duração real da chamada
    - Permite reaplicação da cobrança, se necessário
3. **Defina durações razoáveis**
    - Muito curtas: Operações de cobrança frequentes, má experiência do usuário
    - Muito longas: Risco de perda de receita em chamadas pré-pagas
    - Típico: 60-300 segundos para pré-pago, mais longo para pós-pago
4. **Lide com o timeout de forma elegante**
    - Se `release=false`, esteja preparado para lidar com notificações de expiração do temporizador
    - Implemente lógica para estender a duração ou liberar a chamada

Tratamento de Erros

Problemas comuns:

- **Nenhum OTID ativo**: Deve enviar InitialDP primeiro
- **Duração inválida**: Deve ser de 1-864000 segundos
- **Suporte da rede**: Algumas implementações de SSF podem não suportar ApplyCharging
- **Precisão do temporizador**: A resolução do temporizador da rede é tipicamente de 1 segundo, mas pode variar

Monitoramento

Rastreie operações ApplyCharging via:

- **Histórico de Solicitações**: Mostra solicitações ApplyCharging enviadas
- **Log de Eventos**: Pesquisar por "CAP:applyCharging"
- **Sessões CAMEL**: Monitorar sessões ativas com cobrança aplicada

- **Trace TCAP**: Depurar problemas de codificação/decodificação

Detalhes da Implementação

Gerenciamento de Estado

- LiveView atribui o estado do formulário de rastreamento
- OTID armazenado no socket atribui
- Histórico de solicitações limitado a 20 entradas
- Atualização automática desativada (somente envio manual)

Geração de Solicitações

- Usa o módulo existente `CapRequestGenerator`
- Constrói estruturas TCAP/CAP apropriadas
- Codifica com o codec `:TCAPMessages`
- Envolve em SCCP via `CapClient.sccp_m3ua_maker/2`

Mecanismo de Envio

- Envia via M3UA para `:camelgw_client_asp`
- Usa contexto de roteamento 1
- Encapsulamento automático SCCP/M3UA

Tratamento de Erros

- Validação de formulário com feedback ao usuário
- Tratamento elegante de OTID ausente
- Erros de análise exibidos na interface
- Falhas de codificação registradas

Melhorias Futuras

Adições potenciais:

1. Modelos/presets de solicitação
2. Correlação e exibição de respostas
3. Visualização do fluxo de chamadas
4. Detalhamento de sessões
5. Exportar histórico de solicitações
6. Teste de carga (solicitações em massa)
7. Exportação PCAP de mensagens geradas
8. Validação de parâmetros CAP

Notas de Integração

- Compatível com registro MAP existente (`paklog`)
- Compartilha banco de dados de log de eventos com eventos MAP
- Usa a mesma infraestrutura SCCP/M3UA
- Funciona com CAMELSessionsLive para monitoramento
- Integra-se com roteamento M3UA existente

Arquivos Modificados

- `config/runtime.exs` - ATUALIZADO

Dependências

- CapRequestGenerator existente
- CapClient para envio M3UA
- M3UA.Server para transmissão de pacotes
- EventLog para registro de mensagens
- Framework Phoenix LiveView

- Painel de Controle para infraestrutura da interface

Guia de Recursos Comuns

← Voltar para a Documentação Principal

Este guia cobre recursos comuns a todos os modos de operação do OmniSS7.

Índice

Visão Geral da Interface Web

A Interface Web é acessível através do endereço do servidor web configurado.

Navegação Principal

- **Eventos** - Eventos de sinalização SS7 em tempo real e logs de mensagens
- **Aplicação** - Status da aplicação e informações de tempo de execução
- **Configuração** - Visualizador de configuração do sistema
- **Status M3UA** - Conexões de par M3UA (modo STP)
- **Fila de SMS** - Mensagens SMS de saída (modo SMSc)

Acessando a Interface Web

1. Abra seu navegador web
2. Navegue até o nome do host configurado (ex: `http://localhost`)
3. Visualize o painel de status do sistema

Documentação da API Swagger

Documentação interativa da API:

```
http://your-server/swagger
```

Configuração da Interface Web

Configure em `config/runtime.exs`:

```
config :control_panel,
  # Ordem das páginas no menu de navegação
  page_order: ["/events", "/application", "/configuration"],

  # Configurações do servidor web
  web: %{
    listen_ip: "0.0.0.0",    # IP para vincular (0.0.0.0 para
todas as interfaces)
    port: 80,                 # Porta HTTP (443 para HTTPS)
    hostname: "localhost",    # Nome do host do servidor para
geração de URL
    enable_tls: false,        # Defina como verdadeiro para
habilitar HTTPS
    tls_cert: "cert.pem",    # Caminho para o arquivo de
certificado TLS
    tls_key: "key.pem"       # Caminho para o arquivo de chave
privada TLS
  }
```

Parâmetros de Configuração:

Parâmetro	Tipo	Padrão	Descrição
`page_order`	Lista	`["/events", "/application", "/configuration"]`	Ordem das páginas no menu de navegação
`listen_ip`	String	`"0.0.0.0"`	Endereço IP para vincular o servidor web
`port`	Inteiro	`80`	Porta HTTP (use 443 para HTTPS)
`hostname`	String	`"localhost"`	Nome do host do servidor para geração de URL
`enable_tls`	Booleano	`false`	Habilitar HTTPS com TLS
`tls_cert`	String	`"cert.pem"`	Caminho para o certificado TLS (quando TLS habilitado)
`tls_key`	String	`"key.pem"`	Caminho para a chave privada TLS (quando TLS habilitado)

Configuração do Logger

Configure o nível de log em `config/runtime.exs`:

```
config :logger,
  level: :debug  # Opções: :debug, :info, :warning, :error
```

Níveis de Log:

- `:debug` - Informações detalhadas de depuração
- `:info` - Mensagens informativas gerais
- `:warning` - Mensagens de aviso para problemas potenciais
- `:error` - Apenas mensagens de erro

Documentação da API

URL Base da API

```
http://your-server/api
```

Códigos de Resposta

- **200** - Sucesso
- **400** - Solicitação Inválida
- **504** - Tempo Limite do Gateway

Especificação OpenAPI

```
http://your-server/swagger.json
```

Monitoramento e Métricas

Endpoint de Métricas Prometheus

```
http://your-server/metrics
```

Principais Categorias de Métricas

Métricas M3UA/SCTP:

- Mudanças de estado da associação SCTP
- Transições de estado ASP M3UA
- Unidades de dados de protocolo enviadas/recebidas

Métricas M2PA:

- Transições de estado do link (DOWN → ALIGNMENT → PROVING → READY)
- Mensagens e bytes enviados/recebidos por link
- Erros específicos do link (decodificar, codificar, SCTP)

Métricas STP:

- Mensagens recebidas/enviadas por par
- Falhas de roteamento por motivo
- Distribuição de tráfego entre pares

Métricas do Cliente MAP:

- Solicitações MAP por tipo de operação
- Histogramas de duração das solicitações
- Medidor de transações pendentes

Métricas CAP:

- Solicitações CAP por tipo de operação
- Operações de gateway CAMEL

Métricas SMSc:

- Profundidade da fila
- Taxas de entrega
- Mensagens falhadas

Integração com Grafana

As métricas do OmniSS7 são compatíveis com Prometheus e Grafana.

Melhores Práticas

Recomendações de Segurança

1. **Isolamento de Rede**
    - Implantar em VLAN dedicada
    - Regras de firewall para restringir o acesso
    - Permitir SCTP apenas de endereços conhecidos
2. **Segurança da Interface Web**
    - Habilitar TLS para produção
    - Usar proxy reverso com autenticação
    - Restringir a IPs de gerenciamento
3. **Segurança da API**
    - Implementar limitação de taxa
    - Usar chaves de API ou OAuth
    - Registrar todas as solicitações para auditoria

Ajuste de Desempenho

1. **Limites de TPS**
    - Configurar TPS apropriado
    - Monitorar carga do sistema
    - Ajustar buffers SCTP
2. **Otimização de Banco de Dados**

- Adicionar índices
- Arquivar mensagens antigas
- Monitorar pool de conexões

3. **Ajuste M3UA**
    - Ajustar intervalos de heartbeat SCTP
    - Configurar valores de tempo limite
    - Usar múltiplos links para redundância

Multihoming SCTP para Redundância de Rede

O que é Multihoming SCTP?

Multihoming SCTP é um recurso embutido do protocolo SCTP que permite que uma única conexão M3UA se vincule a múltiplos endereços IP na mesma interface de rede ou em diferentes interfaces de rede. Isso proporciona failover automático e redundância na camada de transporte.

Principais Benefícios:

- **Failover Automático**: Se um caminho de rede falhar, o SCTP muda automaticamente para um caminho alternativo sem desconectar a conexão
- **Failover Sem Configuração**: Nenhuma lógica em nível de aplicação necessária - o SCTP gerencia o monitoramento de caminhos e o failover
- **Maior Confiabilidade**: Sobrevive a falhas de rede, falhas de switch ou falhas de NIC
- **Balanceamento de Carga**: O SCTP pode distribuir tráfego entre múltiplos caminhos (dependente da implementação)

Como Funciona

Quando você configura múltiplos endereços IP para uma conexão M3UA, o SCTP:

1. **Vincula a todos os IPs**: O socket se vincula a todos os endereços IP configurados simultaneamente
2. **Monitora caminhos**: O SCTP envia continuamente pacotes de heartbeat em todos os caminhos para monitorar sua saúde
3. **Detecta falhas**: Se os heartbeats falharem no caminho primário, o SCTP o marca como inacessível
4. **Failover automático**: O tráfego muda imediatamente para um caminho de backup sem intervenção da aplicação
5. **Recuperação de caminho**: Quando o caminho falhado se recupera, o SCTP o detecta e o marca como disponível novamente

Configuração

O multihoming SCTP é configurado fornecendo uma **lista de endereços IP** em vez de uma única tupla IP.

IP Único (Tradicional)

```
# IP único - sem multihoming
local_ip: {10, 179, 4, 10}
```

Múltiplos IPs (Multihoming Habilitado)

```
# Múltiplos IPs - multihoming habilitado
# O primeiro IP é primário, os IPs subsequentes são caminhos de
backup
local_ip: [{10, 179, 4, 10}, {10, 179, 4, 11}]
```

Exemplos de Configuração

> **Nota Importante para o Papel do Servidor (Conexões de Entrada):**
>
> Ao configurar pares com `role: :server` (aceitando conexões de entrada de pares multihomed), você deve especificar `remote_ip` como uma **única tupla** - o endereço IP que o par remoto usa para iniciar a conexão SCTP (SCTP INIT). NÃO use uma lista.

> **Por quê?** O STP combina conexões de entrada com base no IP de origem do pacote SCTP INIT. O SCTP descobrirá automaticamente os outros endereços IP multihomed do par durante o handshake de associação. O formato de lista para `remote_ip` é válido apenas para **`role: :client`** (conexões de saída).
>
> **Exemplos:**
>
> ```
> # ✓ CORRETO - Papel de servidor com remote_ip único
> %{
>   role: :server,
>   remote_ip: {10, 0, 2, 100},  # Tupla única
>   # ...
> }
>
> # ✗ ERRADO - Papel de servidor com lista NÃO combinará conexões
> de entrada
> %{
>   role: :server,
>   remote_ip: [{10, 0, 2, 100}, {10, 0, 2,
> 101}],  # Lista não funcionará
>   # ...
> }
> ```

Exemplo 1: Par STP com Multihoming (Papel do Cliente - Saída)

```
# Configuração do par em modo STP (conexão de SAÍDA)
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "Partner_STP_Redundant",
      role: :client,  # Saída - iniciamos a conexão
      # Multihoming: vincular a dois IPs locais para redundância
      local_ip: [{213, 57, 23, 200}, {213, 57, 23, 201}],
      local_port: 0,
      # O par remoto também suporta multihoming - a lista é OK
para o papel de cliente
      remote_ip: [{213, 57, 23, 100}, {213, 57, 23, 101}],
      remote_port: 2905,
      routing_context: 1,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

Exemplo 2: Cliente MAP com Multihoming

```
# Modo cliente MAP com multihoming
config :omniss7,
  map_client_enabled: true,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    # Multihoming: dois IPs locais para failover
    local_ip: [{10, 0, 0, 100}, {10, 0, 0, 101}],
    local_port: 2905,
    # STP remoto com suporte a multihoming
    remote_ip: [{10, 0, 0, 1}, {10, 0, 0, 2}],
    remote_port: 2905,
    routing_context: 1
  }
```

Exemplo 3: Listener STP com Multihoming

```
# Servidor STP autônomo com multihoming
config :omniss7,
  sctp_handler: %{
    enabled: true,
    # Ouvir em múltiplos IPs para conexões de entrada
    local_ip: [{172, 16, 0, 10}, {172, 16, 0, 11}],
    local_port: 2905,
    point_code: 100
  }
```

Exemplo 4: Papel de Servidor - Aceitando Entrada de Par Multihomed

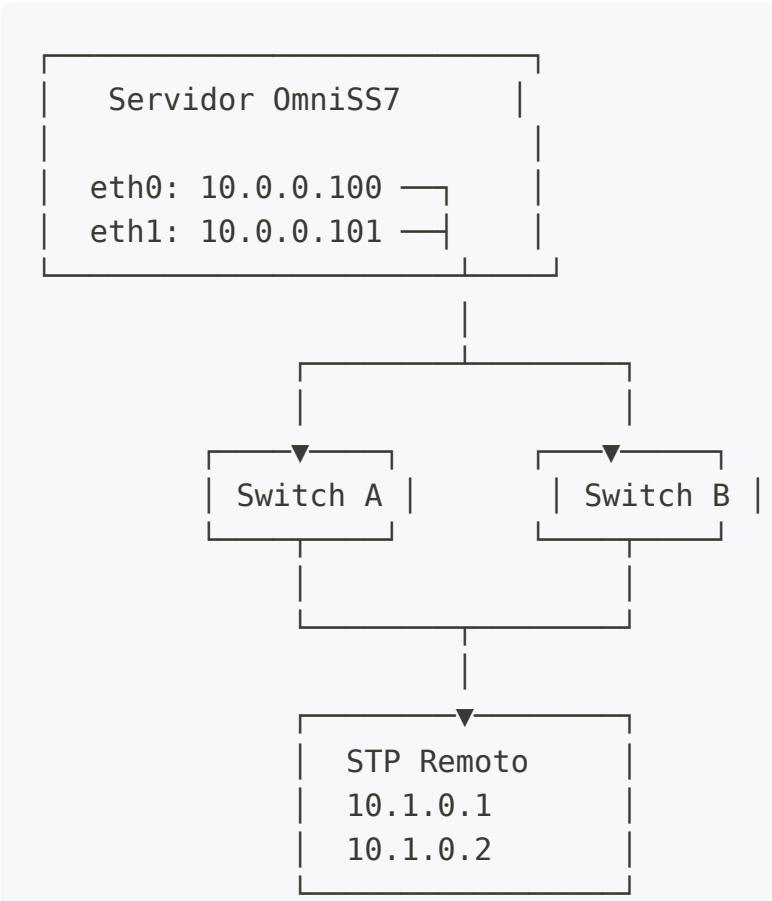
```
# Aceitando conexão de entrada de um HLR multihomed
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "HLR",
      role: :server,  # ENTRADA - HLR se conecta a nós
      # Multihoming: nossos IPs locais (a lista é OK)
      local_ip: [{10, 0, 1, 10}, {10, 0, 1, 11}],
      local_port: 2905,
      # IMPORTANTE: Apenas tupla única - o IP que o HLR usa para
iniciar SCTP INIT
      # O SCTP descobrirá automaticamente os outros IPs do HLR
durante o handshake
      remote_ip: {10, 0, 2, 100},  # Apenas IP primário (NÃO uma
lista!)
      remote_port: 0,  # Aceitar de qualquer porta de origem
      routing_context: 1,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

Exemplo 5: Configuração Mista (Compatível com Versões Anteriores)

```
# Mistura de pares com IP único e multihomed
config :omniss7,
  m3ua_peers: [
    # Par legado - IP único
    %{
      peer_id: 1,
      name: "Legacy_STP",
      role: :client,
      local_ip: {10, 0, 0, 1},      # Tupla de IP único
      local_port: 0,
      remote_ip: {10, 0, 0, 10},
      remote_port: 2905,
      routing_context: 1,
      point_code: 100
    },
    # Novo par - multihoming
    %{
      peer_id: 2,
      name: "Redundant_STP",
      role: :client,
      local_ip: [{10, 0, 0, 2}, {10, 0, 0, 3}],  # Lista de IPs
      local_port: 0,
      remote_ip: [{10, 0, 0, 20}, {10, 0, 0, 21}],
      remote_port: 2905,
      routing_context: 2,
      point_code: 200
    }
  ]
```

Cenários de Topologia de Rede

Cenário 1: Dual NICs (Implantação Comum)



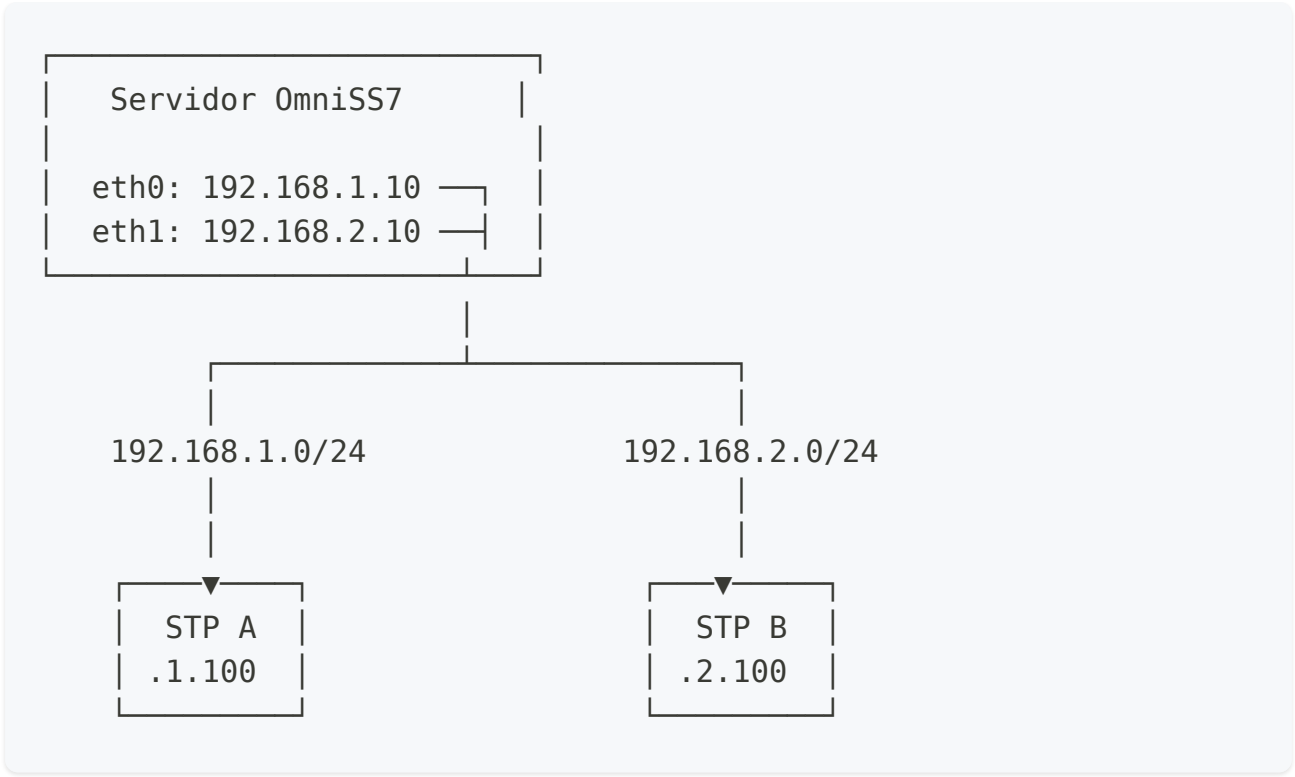
Configuração:

```
local_ip: [{10, 0, 0, 100}, {10, 0, 0, 101}]   # Ambos os NICs
remote_ip: [{10, 1, 0, 1}, {10, 1, 0, 2}]      # Par remoto
```

Benefícios:

- Sobrevive à falha de um NIC
- Sobrevive à falha de um switch
- Failover automático em <1 segundo

Cenário 2: Múltiplas Sub-redes



Configuração:

```
local_ip: [{192, 168, 1, 10}, {192, 168, 2, 10}]
remote_ip: [{192, 168, 1, 100}, {192, 168, 2, 100}]
```

Benefícios:

- Sobrevive à falha de sub-rede
- Redundância geográfica possível
- Caminhos de roteamento independentes

Monitoramento e Registro

Quando o multihoming está habilitado, você verá mensagens de log indicando a configuração:

Multihoming Bem-Sucedido

```
[info] Multihoming do cliente SCTP: vinculado a 2 IPs locais
[info] Multihoming do listener STP habilitado: 2 IPs locais
vinculados
```

Eventos de Failover de Caminho

```
[warning] [MULTIHOMING] Caminho 10.0.0.100 está INACESSÍVEL para o
par Partner_STP (assoc_id=1)
[info] [MULTIHOMING] Caminho 10.0.0.101 agora é o PRINCIPAL para o
par Partner_STP (assoc_id=1)
[info] [MULTIHOMING] Caminho 10.0.0.100 agora está DISPONÍVEL para
o par Partner_STP (assoc_id=1)
```

Exibição na Interface Web

A Interface Web exibe automaticamente informações de multihoming:

Página de Status M3UA:

- **IP Único**: Mostra como `10.0.0.100`
- **Múltiplos IPs**: Mostra como `10.0.0.100 (+1)` ou `10.0.0.100 (+2)`
- **Visualização de Detalhes**: Mostra todos os IPs com rótulos primário/backup

Melhores Práticas

1. Design de Rede

- **Use NICs diferentes** para máxima redundância
- **Diferentes switches** para sobreviver a falhas de switch
- **Diferentes sub-redes** se possível para diversidade de roteamento
- **Mesmo datacenter inicialmente** - teste antes da separação geográfica

2. Planejamento de Endereço IP

- **Primeiro IP é primário** - certifique-se de que está no caminho mais confiável
- **A ordem importa** - liste os IPs na ordem de preferência
- **Endereçamento consistente** - use esquemas de endereçamento semelhantes para solução de problemas

3. Testando Failover

```
# Desativar a interface primária para testar o failover
sudo ip link set eth0 down

# Monitorar logs para failover
tail -f /var/log/omniss7.log | grep MULTIHOMING

# Reativar a interface
sudo ip link set eth0 up
```

4. Ambos os Lados Devem Suportar Multihoming

- **Ótimo**: Tanto o local quanto o remoto usam múltiplos IPs
- **Aceitável**: Apenas um lado usa multihoming
- **Nota**: A redundância é melhor quando ambos os pontos finais a suportam

5. Configuração de Firewall

```
# Permitir SCTP em todos os IPs de multihoming
iptables -A INPUT -p sctp --dport 2905 -s 10.0.0.0/24 -j ACCEPT
iptables -A INPUT -p sctp --dport 2905 -s 10.1.0.0/24 -j ACCEPT
```

Solução de Problemas

Problema: Multihoming Não Funciona

Sintomas: Apenas o IP primário é usado, sem failover

Verificações:

1. Verifique o suporte SCTP do Erlang: `erl -eval 'gen_sctp:open(9999, [binary, {ip, {127,0,0,1}}]).'`
2. Verifique o módulo SCTP do kernel: `lsmod | grep sctp`
3. Carregue o SCTP se necessário: `sudo modprobe sctp`
4. Verifique se ambos os IPs estão configurados no sistema: `ip addr show`

Problema: Caminho Não Está Falhando

Sintomas: Caminho primário marcado como inativo, mas o tráfego não está mudando

Verificações:

1. Verifique as configurações de heartbeat SCTP
2. Verifique se a tabela de roteamento tem rotas para todos os caminhos
3. Verifique se o firewall permite SCTP em todos os IPs
4. Revise os logs de monitoramento de caminhos SCTP

Problema: Flutuação Frequente de Caminhos

Sintomas: Caminhos constantemente alternando entre ATIVO e INATIVO

Verificações:

1. Instabilidade na rede - verifique os links físicos
2. Heartbeat SCTP muito agressivo - pode precisar de ajuste
3. Firewall bloqueando heartbeats SCTP
4. Problemas de MTU em um caminho

Considerações de Desempenho

- **Sobrecarga mínima**: Os heartbeats SCTP são pequenos e infrequentes
- **Sem alterações na aplicação**: O multihoming é transparente para a camada de aplicação
- **Failover rápido**: Tipicamente <1 segundo de detecção e failover
- **Recuperação automática**: Nenhuma intervenção manual necessária

Compatibilidade

- **Compatível com versões anteriores**: O formato de tupla de IP único ainda funciona
- **Implantações mistas**: É possível misturar pares de IP único e multihomed
- **Todos os modos suportados**: Funciona nos modos STP, HLR, SMSc e Cliente MAP
- **Requisito Erlang**: Requer Erlang com suporte a SCTP compilado

Monitoramento e Alerta

Métricas Principais:

- Estado da conexão M3UA
- Taxa de sucesso de solicitações MAP
- Tempos de resposta da API
- Profundidade da fila de mensagens

Limiares de Alerta:

- M3UA inativo > 1 minuto
- Taxa de tempo limite MAP > 10%
- Profundidade da fila > 1000
- Taxa de erro da API > 5%

Referência Completa de Configuração

Todos os Parâmetros de Configuração

Esta seção fornece uma referência completa de todos os parâmetros de configuração disponíveis em todos os modos de operação.

Configuração do Logger (`:logger`)

```
config :logger,
  level: :debug  # :debug | :info | :warning | :error
```

Configuração da Interface Web (`:control_panel`)

```
config :control_panel,
  page_order: ["/events", "/application", "/configuration"],
  web: %{
    listen_ip: "0.0.0.0",
    port: 80,
    hostname: "localhost",
    enable_tls: false,
    tls_cert: "cert.pem",
    tls_key: "key.pem"
  }
```

Parâmetro	Tipo	Obrigatório	Padrão	Descri
`page_order`	Lista de Strings	Não	`["/events", "/application", "/configuration"]`	Ordem páginas menu d navega
`web.listen_ip`	String	Sim	`"0.0.0.0"`	Endere para vincula servido web
`web.port`	Inteiro	Sim	`80`	Númer porta HTTP/H
`web.hostname`	String	Sim	`"localhost"`	Nome d host do servido
`web.enable_tls`	Booleano	Não	`false`	Habilita HTTPS
`web.tls_cert`	String	Se TLS habilitado	`"cert.pem"`	Caminh certifica TLS
`web.tls_key`	String	Se TLS habilitado	`"key.pem"`	Caminh chave privada

Configuração do SocketHandler SCTP (`:omniss7`)

```
config :omniss7,
  sctp_handler: %{
    enabled: false,
    local_ip: {127, 0, 0, 1},
    local_port: 2905
  },
  enable_gt_routing: true,
  m3ua_peers: [...],
  m3ua_routes: [...],
  m3ua_gt_routes: [...]
```

Parâmetro	Tipo	Obrigatório	Padrão	Descriç
`sctp_handler.enabled`	Booleano	Sim	`false`	Habilitar modo STI na inicializaç
`sctp_handler.local_ip`	Tupla	Sim	`{127, 0, 0, 1}`	IP para vincular para M3U de entrad
`sctp_handler.local_port`	Inteiro	Sim	`2905`	Porta SCT para M3U
`enable_gt_routing`	Booleano	Não	`false`	Habilitar roteamer de Título Global

Parâmetros do Par M3UA:

Parâmetro	Tipo	Obrigatório	Descrição
`peer_id`	Inteiro	Sim	Identificador único do par
`name`	String	Sim	Nome descritivo do par
`role`	Átomo	Sim	`:client` ou `:server`
`local_ip`	Tupla ou Lista	Se `:client`	IP(s) local(is) para vincular. Único: `{10, 0, 0, 1}` ou Lista: `[{10, 0, 0, 1}, {10, 0, 0, 2}]`
`local_port`	Inteiro	Se `:client`	Porta local (0 para dinâmica)
`remote_ip`	Tupla ou Lista	Sim	IP(s) do par remoto. Único: `{10, 0, 0, 10}` ou Lista: `[{10, 0, 0, 10}, {10, 0, 0, 11}]`
`remote_port`	Inteiro	Se `:client`	Porta do par remoto
`routing_context`	Inteiro	Sim	Contexto de roteamento M3UA
`point_code`	Inteiro	Sim	Código de ponto SS7
`network_indicator`	Átomo	Não	`:international` ou `:national`

Parâmetros de Rota M3UA:

Parâmetro	Tipo	Obrigatório	Descrição
`dest_pc`	Inteiro	Sim	Código de ponto de destino
`peer_id`	Inteiro	Sim	Par pelo qual rotear
`priority`	Inteiro	Sim	Prioridade da rota (menor = maior prioridade)
`network_indicator`	Átomo	Não	`:international` ou `:national`

Parâmetros de Rota GT M3UA:

Parâmetro	Tipo	Obrigatório	Descrição
`gt_prefix`	String	Sim	Prefixo de Título Global a ser correspondido
`peer_id`	Inteiro	Sim	Par de destino
`priority`	Inteiro	Sim	Prioridade da rota
`description`	String	Não	Descrição da rota para registro
`source_ssn`	Inteiro	Não	Correspondência apenas se o SSN de origem corresponder
`dest_ssn`	Inteiro	Não	Reescrever o SSN de destino para este valor

Configuração do Cliente MAP (`:omniss7`)

```
config :omniss7,
  map_client_enabled: false,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :map_client_asp,
    local_ip: {10, 0, 0, 100},
    local_port: 2905,
    remote_ip: {10, 0, 0, 1},
    remote_port: 2905,
    routing_context: 1
  }
```

Parâmetro	Tipo	Obrigatório	Padrã
`map_client_enabled`	Booleano	Sim	`false`
`map_client_m3ua.mode`	String	Sim	`"ASP"`
`map_client_m3ua.callback`	Tupla	Sim	`{MapClient :handle_pa []}`
`map_client_m3ua.process_name`	Átomo	Sim	`:map_clie`
`map_client_m3ua.local_ip`	Tupla	Sim	-
`map_client_m3ua.local_port`	Inteiro	Sim	`2905`
`map_client_m3ua.remote_ip`	Tupla	Sim	-
`map_client_m3ua.remote_port`	Inteiro	Sim	`2905`
`map_client_m3ua.routing_context`	Inteiro	Sim	-

Configuração do Centro de SMS (`:omniss7`)

```
config :omniss7,
  auto_flush_enabled: false,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: nil,
  auto_flush_tps: 10
```

Parâmetro	Tipo	Obrigatório	Padrão	Descriçã
`auto_flush_enabled`	Booleano	Não	`false`	Habilitar aut flush da fila SMS
`auto_flush_interval`	Inteiro	Não	`10000`	Intervalo de polling da fil (milissegund
`auto_flush_dest_smsc`	String/nil	Não	`nil`	Filtrar por SMSC de destino (nil = todos)
`auto_flush_tps`	Inteiro	Não	`10`	Máximo de transações p segundo

Configuração da API HTTP (`:omniss7`)

O backend de SMS agora usa a API HTTP em vez de conexões diretas ao banco de dados.

```
config :omniss7,
  smsc_api_base_url: "https://10.5.198.200:8443",
  frontend_name: "omni-smsc01"  # Opcional: padrão para
hostname_SMSc
```

Parâmetros da API:

Parâmetro	Tipo	Obrigatório	Padrão
`smsc_api_base_url`	String	Sim	`"https://10.5.198.200:8443"`
`frontend_name`	String	Não	`"{hostname}_SMSc"`

Endpoints da API Utilizados:

- `POST /api/frontends` - Registrar esta instância de frontend com o backend
- `POST /api/messages_raw` - Inserir novas mensagens SMS
- `GET /api/messages` - Recuperar fila de mensagens (com cabeçalho `smsc`)
- `PATCH /api/messages/{id}` - Marcar mensagem como entregue
- `PUT /api/messages/{id}` - Atualizar status da mensagem
- `POST /api/events` - Adicionar rastreamento de eventos
- `GET /api/status` - Endpoint de verificação de saúde

Registro do Frontend:

O sistema registra automaticamente a si mesmo com a API do backend na inicialização e re-registra a cada 5 minutos. O registro inclui:

- Nome e tipo do frontend (SMSc)
- Nome do host

- Tempo de atividade em segundos
- Detalhes da configuração (formato JSON)

Notas de Configuração:

- A verificação SSL está desabilitada por padrão para certificados autoassinados
- As solicitações HTTP têm tempo limite após 5 segundos
- Todos os timestamps estão no formato ISO 8601
- A API usa JSON para corpos de solicitação/resposta

Documentação Relacionada

- ← Voltar para a Documentação Principal
- Guia STP
- Guia do Cliente MAP
- Guia do Centro de SMS
- Guia HLR

OmniSS7 por Omnitouch Network Services

Referência de Configuração

← Voltar para a Documentação Principal

Este documento fornece uma referência abrangente para todos os parâmetros de configuração do OmniSS7.

Índice

Visão Geral

A configuração do OmniSS7 é gerenciada através de `config/runtime.exs`. O sistema suporta quatro modos operacionais:

- **Modo STP** - Ponto de Transferência de Sinal para roteamento
- **Modo HLR** - Registro de Localização do Usuário para gerenciamento de assinantes
- **Modo SMSc** - Centro de SMS para entrega de mensagens

- **Modo CAMEL GW** - Gateway CAMEL para controle inteligente de chamadas

Arquivo de Configuração: `config/runtime.exs`

Flags de Modo Operacional

Controle quais recursos estão habilitados.

Parâmetro	Tipo	Padrão	Descrição	Modos
`map_client_enabled`	Boolean	`false`	Habilitar cliente MAP e conectividade M3UA	Todos
`hlr_mode_enabled`	Boolean	`false`	Habilitar recursos específicos do HLR	HLR
`smsc_mode_enabled`	Boolean	`false`	Habilitar recursos específicos do SMSc	SMSc
`cap_client_enabled`	Boolean	`false`	Habilitar cliente CAP para operações CAMEL	CAMEL GW
`camelgw_mode_enabled`	Boolean	`false`	Habilitar recursos do Gateway CAMEL	CAMEL GW
`ussd_gateway_enabled`	Boolean	`false`	Habilitar Gateway USSD (ponte HTTP/JSON)	USSD GW

Exemplo:

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false
```

Parâmetros do Modo HLR

Configuração para o modo HLR (Registro de Localização do Usuário).

Configuração da API HLR

Parâmetro	Tipo	Padrão	Obrigatório	
`hlr_api_base_url`	String	-	**Sim**	UR da ba
`hlr_api_verify_ssl`	Boolean	`false`	Não	Ha ve ce pa
`hlr_service_center_gt_address`	String	-	**Sim**	En Gl re re Up
`smsc_service_center_gt_address`	String	-	**Sim**	En SM na fo

Exemplo:

```
config :omniss7,
  hlr_api_base_url: "https://10.180.2.140:8443",
  hlr_api_verify_ssl: false,
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112"
```

Configuração do AlertServiceCenter

Quando um assinante realiza UpdateLocation, o HLR envia mensagens alertServiceCenter para os GTs do SMSc configurados para indicar que o assinante agora está acessível.

Parâmetro	Tipo	Padrão	Obrigatório
`hlr_smsc_alert_gts`	Lista de Strings	`[]`	Não
`hlr_alert_location_expiry_seconds`	Inteiro	`172800`	Não

Exemplo:

```
config :omniss7,
  hlr_smsc_alert_gts: [
    "15559876543",
    "15559876544"
  ],
  hlr_alert_location_expiry_seconds: 172800  # 48 horas
```

Mapeamento MSISDN ↔ IMSI

Configuração para geração sintética de IMSI a partir de MSISDNs. Para uma explicação técnica detalhada do algoritmo de mapeamento, consulte Mapeamento MSISDN ↔ IMSI no Guia HLR.

Parâmetro	Tipo	Padrão	Obrigatório	Descriçã
`hlr_imsi_plmn_prefix`	String	`"50557"`	Não	Prefixo PLM (MCC+MNC para geraçã sintética de IMSI
`hlr_msisdn_country_code`	String	`"61"`	Não	Prefixo do código do país para mapeamen reverso IMSI→MSISD
`hlr_msisdn_nsn_offset`	Inteiro	`0`	Não	Deslocame no MSISDN onde o NSN começa (tipicament o comprimen do código d país)
`hlr_msisdn_nsn_length`	Inteiro	`9`	Não	Comprimen do Número Nacional do Assinante a ser extraíd do MSISDN

Exemplo (código do país de 2 dígitos):

```
config :omniss7,
  hlr_imsi_plmn_prefix: "50557",      # MCC 505 + MNC 57
  hlr_msisdn_country_code: "99",      # Exemplo de código do país
de 2 dígitos
  hlr_msisdn_nsn_offset: 2,           # Ignorar código do país de
2 dígitos
  hlr_msisdn_nsn_length: 9            # Extrair NSN de 9 dígitos
```

Exemplo (código do país de 3 dígitos):

```
config :omniss7,
  hlr_imsi_plmn_prefix: "50557",      # MCC 505 + MNC 57
  hlr_msisdn_country_code: "999",     # Exemplo de código do país
de 3 dígitos
  hlr_msisdn_nsn_offset: 3,           # Ignorar código do país de
3 dígitos
  hlr_msisdn_nsn_length: 8            # Extrair NSN de 8 dígitos
```

Importante: Defina `nsn_offset` como o comprimento do seu código do país para extrair corretamente o NSN. Por exemplo:

- Código do país "9" (1 dígito) → `nsn_offset: 1`
- Código do país "99" (2 dígitos) → `nsn_offset: 2`
- Código do país "999" (3 dígitos) → `nsn_offset: 3`

Configuração do InsertSubscriberData (ISD)

Configuração para dados de provisionamento de assinantes enviados para VLRs durante UpdateLocation. Para uma explicação detalhada da sequência ISD e do fluxo de mensagens, consulte Configuração do InsertSubscriberData no Guia HLR.

Parâmetro	Tipo	Padrão	Obrigatório
`isd_network_access_mode`	Atom	`:packetAndCircuit`	Não
`isd_send_ss_data`	Boolean	`true`	Não
`isd_send_call_barring`	Boolean	`true`	Não

Exemplo:

```
config :omniss7,
  isd_network_access_mode: :packetAndCircuit,
  isd_send_ss_data: true,
  isd_send_call_barring: true
```

Configuração CAMEL

Configuração para roteamento inteligente de chamadas baseado em CAMEL. Para uma explicação detalhada da integração CAMEL e das chaves de serviço, consulte Integração CAMEL no Guia HLR.

Parâmetro	Tipo	Padrão	Obr
`camel_service_key`	Inteiro	`11_110`	Não
`camel_trigger_detection_point`	Atom	`:termAttemptAuthorized`	Não
`camel_gsmscf_gt_address`	String	(usa GT chamado)	Não

Exemplo:

```
config :omniss7,
  camel_service_key: 11_110,
  camel_trigger_detection_point: :termAttemptAuthorized
```

Prefixos VLR Domésticos

Configuração para distinguir assinantes domésticos de roaming. Para uma explicação detalhada da detecção de doméstico/roaming e operações PRN, consulte Tratamento de Assinantes Roaming no Guia HLR.

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`home_vlr_prefixes`	Lista	`["5551231"]`	Não	Prefixos GT do VLR considerados rede "doméstica"

Exemplo:

```
config :omniss7,
  home_vlr_prefixes: ["5551231", "5551234"]
```

Parâmetros do Modo SMSc

Configuração para o modo Centro de SMS.

Configuração da API SMSc

Parâmetro	Tipo	Padrão	Obriga
`smsc_api_base_url`	String	-	**Sim**
`smsc_api_verify_ssl`	Boolean	`false`	Não
`smsc_name`	String	`"{hostname}_SMSc"`	Não
`smsc_service_center_gt_address`	String	-	**Sim**

Exemplo:

```
config :omniss7,
  smsc_api_base_url: "https://10.179.3.219:8443",
  smsc_api_verify_ssl: false,
  smsc_name: "ipsmgw",
  smsc_service_center_gt_address: "55512341112"
```

Nota: O registro do frontend ocorre a cada **5 minutos** (hardcoded) através do módulo `SMS.FrontendRegistry`.

Configuração de Auto-Flush

Parâmetro	Tipo	Padrão	Obrigatório	Descriç
`auto_flush_enabled`	Boolean	`true`	Não	Habilitar processament automático da SMS
`auto_flush_interval`	Inteiro	`10_000`	Não	Intervalo de processament fila em milisse
`auto_flush_dest_smsc`	String	-	**Sim**	Nome do SMS destino para a flush
`auto_flush_tps`	Inteiro	`10`	Não	Taxa de processament mensagens (transações/s

Exemplo:

```
config :omniss7,
  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10
```

Parâmetros do Modo STP

Configuração para o modo Ponto de Transferência de Sinal M3UA. Para configuração de roteamento detalhada e exemplos, consulte o Guia de Configuração STP.

Servidor STP Autônomo

Parâmetro	Tipo	Padrão	Obrigatório	Descriç
`sctp_handler.enabled`	Boolean	`false`	Não	Habilitar servidor S SocketHan autônomo
`sctp_handler.local_ip`	Tupla ou Lista	`{127, 0, 0, 1}`	Não	Endereço(para escut conexões. único: `{10 0, 0, 1}` Múltiplos I para multihomi SCTP: `[{1 0, 0, 1}, {10, 0, 0 2}]`
`sctp_handler.local_port`	Inteiro	`2905`	Não	Porta para escutar
`sctp_handler.point_code`	Inteiro	-	**Sim** (se habilitado)	Código de ponto SS7 deste STP

Exemplo (IP Único):

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    point_code: 100
  }
```

Exemplo (Multihoming SCTP):

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    # Múltiplos IPs para redundância
    local_ip: [{10, 179, 4, 10}, {10, 179, 4, 11}],
    local_port: 2905,
    point_code: 100
  }
```

> **Nota**: Para informações detalhadas sobre configuração e benefícios do multihoming SCTP, consulte Multihoming SCTP no Guia Comum.

Roteamento de Título Global

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`enable_gt_routing`	Boolean	`false`	Não	Habilitar roteamento GT além do roteamento PC

Exemplo:

```
config :omniss7,
  enable_gt_routing: true
```

Configuração de Peer M3UA/M2PA

Os peers são configurados através da lista `m3ua_peers` (suporta os protocolos M3UA e M2PA). Para exemplos completos de configuração, consulte o Guia de Configuração STP e Suporte ao Protocolo M2PA.

Parâmetros Comuns de Peer:

Parâmetro	Tipo	Padrão	Obrigatório	Des
`peer_id`	Inteiro	-	**Sim**	Identif único
`name`	String	-	**Sim**	Nome do pee
`protocol`	Atom	`:m3ua`	Não	Tipo d protoc `:m3ua`

`role`	Atom	`:client`	Não	Papel
				conex
				`:clie`
				`:serv`
				ou `:sg`

`local_ip`	Tupla ou Lista	-	**Sim**	Endere local(i vincul
`local_port`	Inteiro	-	**Sim**	Porta S (M3UA M2PA:
`remote_ip`	Tupla ou Lista	-	**Sim**	Endere remot
`remote_port`	Inteiro	-	**Sim**	Porta S remot
`routing_context`	Inteiro	-	Não	Identif contex rotean M3UA M3UA)

Parâmetro	Tipo	Padrão	Obrigatório	Des
`point_code`	Inteiro	-	**Sim**	Código local
`network_indicator`	Atom	`:international`	Não	Indica rede: `:inte` ou `:na`
`initiate_connection`	Boolean	`true`	Não	Se dev conex

Parâmetros Específicos do M2PA:

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`adjacent_point_code`	Inteiro	-	**Sim** (M2PA)	Código de ponto do peer adjacente

Gerenciamento de Sockets:

O M2PA usa automaticamente o SCTP.SocketHandler para gerenciamento de socket compartilhado. Todos os peers M2PA usam o socket compartilhado, o que permite que múltiplos peers compartilhem eficientemente a mesma porta SCTP. Para configuração detalhada, consulte Requisitos de Socket M2PA.

Exemplo (Peer M3UA):

```
config :omniss7,
  m3ua_peers: [
    %{
      peer_id: 1,
      name: "HLR_East",
      protocol: :m3ua,
      role: :sgp,
      local_ip: {10, 179, 4, 10},
      local_port: 2905,
      remote_ip: {10, 179, 4, 20},
      remote_port: 2905,
      point_code: 100,
      network_indicator: :international
    }
  ]
```

Exemplo (Peer M2PA):

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,
    point_code: 100
  },
  m3ua_peers: [
    %{
      peer_id: 2,
      name: "M2PA_Link_STP_West",
      protocol: :m2pa,
      role: :client,
      local_ip: {10, 179, 4, 10},
      local_port: 3565,
      remote_ip: {10, 179, 4, 30},
      remote_port: 3565,
      point_code: 100,
      adjacent_point_code: 200
    }
  ]
```

Rotas de Código de Ponto M3UA

Configuração de roteamento de Código de Ponto. As rotas definem qual peer usar para alcançar códigos de ponto de destino específicos.

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`m3ua_routes`	Lista de Mapas	`[]`	Não	Lista de rotas de código de ponto. Se não especificado, as rotas são geradas automaticamente a partir dos códigos de ponto dos peers.

Formato da Rota: Cada rota em `m3ua_routes` deve ser um mapa com:

- `dest_pc`: Código de ponto de destino (Inteiro)
- `peer_id`: ID do peer para roteamento (Inteiro)
- `priority`: Prioridade da rota - valor mais baixo = maior prioridade (Inteiro)
- `network_indicator`: Indicador de rede (Atom): `:international` ou `:national`

Exemplo:

```
config :omniss7,
  m3ua_routes: [
    # Rota para PC 100 via peer 1 (maior prioridade)
    %{dest_pc: 100, peer_id: 1, priority: 1, network_indicator:
:international},
    # Rota para PC 200 via peer 2
    %{dest_pc: 200, peer_id: 2, priority: 1, network_indicator:
:international},
    # Balanceamento de carga: mesmo dest_pc com prioridades
diferentes
    %{dest_pc: 300, peer_id: 3, priority: 1, network_indicator:
:international},
    %{dest_pc: 300, peer_id: 4, priority: 2, network_indicator:
:international}
  ]
```

Rotas de Título Global M3UA

Roteamento baseado em prefixo de Título Global com transformação avançada de parâmetros SCCP. O maior prefixo correspondente é usado primeiro, depois a prioridade.

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`m3ua_gt_routes`	Lista de Mapas	`[]`	Não	Lista de regras de roteamento de Título Global com transformações opcionais

Formato da Rota: Cada rota em `m3ua_gt_routes` deve ser um mapa com:

Parâmetros Básicos:

- `gt_prefix`: Prefixo de Título Global a ser correspondido (String) - string vazia corresponde a todos
- `peer_id`: ID do peer para roteamento (Inteiro) - use 0 para DESCARTAR tráfego

- `priority`: Prioridade da rota - valor mais baixo = maior prioridade (Inteiro)
- `description`: Descrição legível por humanos (String)

Parâmetros de Correspondência Opcionais:

- `source_ssn`: Correspondência do Número de SubSistema de origem (Inteiro)
- `source_tt`: Correspondência do Tipo de Tradução de origem (Inteiro)
- `source_npi`: Correspondência do Indicador de Plano de Numeração de origem (Inteiro)
- `source_nai`: Correspondência do Indicador de Natureza de Endereço de origem (Inteiro)

Parâmetros de Transformação Opcionais:

- `dest_ssn`: Transformar SSN na mensagem encaminhada (Inteiro)
- `dest_tt`: Transformar Tipo de Tradução na mensagem encaminhada (Inteiro)
- `dest_npi`: Transformar Indicador de Plano de Numeração na mensagem encaminhada (Inteiro)
- `dest_nai`: Transformar Indicador de Natureza de Endereço na mensagem encaminhada (Inteiro)

Valores Comuns:

- **Tipo de Tradução (TT):** 0=Desconhecido, 1=Internacional, 2=Nacional, 3=Específico da Rede
- **Plano de Numeração (NPI):** 0=Desconhecido, 1=ISDN(E.164), 6=Móvel(E.212)
- **Natureza de Endereço (NAI):** 0=Desconhecido, 1=Assinante, 3=Nacional, 4=Internacional
- **Número de SubSistema (SSN):** 6=HLR, 7=VLR, 8=MSC, 9=EIR, etc.

Exemplo:

```
config :omniss7,
  m3ua_gt_routes: [
    # Roteamento básico por prefixo
    %{gt_prefix: "1234", peer_id: 1, priority: 1, description:
"Números dos EUA"},
    %{gt_prefix: "44", peer_id: 2, priority: 1, description:
"Números do Reino Unido"},

    # Transformação de Tipo de Tradução
    %{
      gt_prefix: "61",
      peer_id: 3,
      priority: 1,
      description: "Números australianos: transformação TT 0→1",
      source_tt: 0,  # Correspondência TT=0 (Desconhecido)
      dest_tt: 1     # Transformar para TT=1 (Internacional)
    },

    # Transformação de NPI
    %{
      gt_prefix: "49",
      peer_id: 1,
      priority: 1,
      description: "Números alemães: conversão de Móvel→ISDN NPI",
      source_npi: 6,  # Correspondência NPI=6 (Móvel/E.212)
      dest_npi: 1     # Transformar para NPI=1 (ISDN/E.164)
    },

    # Transformação combinada com roteamento SSN
    %{
      gt_prefix: "86",
      source_ssn: 8,   # Correspondência SSN=8 (MSC)
      peer_id: 3,
      dest_ssn: 6,     # Reescrever para SSN=6 (HLR)
      priority: 1,
      description: "Tráfego chinês: normalização completa",
      source_tt: 0,
      dest_tt: 2,
      source_npi: 6,
      dest_npi: 1,
      source_nai: 4,
      dest_nai: 3
    },
```

```
    # Rota padrão/backup
    %{
      gt_prefix: "",
      peer_id: 1,
      priority: 99,
      description: "Rota de fallback padrão"
    }
  ]
```

Parâmetros do Modo CAMEL Gateway

Configuração para o modo Gateway CAMEL (protocolo CAP).

Flags do Modo CAMEL

Habilitar recursos CAMEL/CAP (defina `cap_client_enabled: true` e `camelgw_mode_enabled: true` nas Flags de Modo Operacional).

Configuração do Protocolo CAP

Parâmetro	Tipo	Padrão	Obrigatório	Descriçã
`cap_version`	Atom	`:v2`	Não	Versão do protocolo CAP: `:v1`, `:v2`, `:v3` ou `:v4`
`camel_gsmscf_gt_address`	String	(usa GT chamado)	Não	Título Global gsmSCF padrão para respostas CAMEL

Mapeamento de Versão CAP:

- `:v1` → OID do Contexto de Aplicação: 0.4.0.0.1.0.50.0
- `:v2` → OID do Contexto de Aplicação: 0.4.0.0.1.0.50.1 (padrão - mais amplamente suportado)
- `:v3` → OID do Contexto de Aplicação: 0.4.0.0.1.21.3.4
- `:v4` → OID do Contexto de Aplicação: 0.4.0.0.1.23.3.4

Nota: As solicitações de entrada são detectadas automaticamente a partir de seu OID de contexto de aplicação e as respostas correspondem à versão da solicitação.

Exemplo:

```
config :omniss7,
  cap_client_enabled: true,
  camelgw_mode_enabled: true,
  cap_version: :v2,
  camel_gsmscf_gt_address: "68988411553"
```

Integração CGrateS

Integração de cobrança em tempo real com CGrateS para faturamento pré-pago/pós-pago.

Parâmetro	Tipo	Padrão	Obrigatório	De
`cgrates_enabled`	Boolean	`false`	Não	Habilit integra CGrate
`cgrates_url`	String	-	**Sim** (se habilitado)	URL d JSON-R CGrate
`cgrates_tenant`	String	`"cgrates.org"`	Não	Identif inquili
`cgrates_request_type`	String	`"*prepaid"`	Não	Tipo d `"*pre` `"*pos` `"*pse`
`cgrates_timeout`	Inteiro	`5000`	Não	Timeo solicita CGrate milisse

Exemplo:

```
config :omniss7,
  cgrates_enabled: true,
  cgrates_url: "http://localhost:2080/jsonrpc",
  cgrates_tenant: "cgrates.org",
  cgrates_request_type: "*prepaid",
  cgrates_timeout: 5000
```

Conexão M3UA CAP

O Gateway CAMEL usa uma conexão M3UA separada para operações CAP.

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`cap_client_m3ua`	Mapa	-	**Sim**	Configuração de conexão M3UA CAP (mesma estrutura que `map_client_m3ua`, incluindo parâmetros opcionais `opc` e `dpc`)

Exemplo:

```
config :omniss7,
  cap_client_m3ua: %{
    mode: "ASP",
    callback: {CapClient, :handle_payload, []},
    process_name: :camelgw_client_asp,
    local_ip: {10, 5, 198, 200},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 4,
    opc: 5013,    # Código de Ponto de Origem
    dpc: 5011     # Código de Ponto de Destino
  }
```

Parâmetros de NAT de Título Global

A Tradução de Endereço de Rede de Título Global permite diferentes GTs de resposta com base no prefixo da parte chamadora, prefixo da parte chamada ou ambos. As regras são correspondidas por peso (menor = maior prioridade), depois pela especificidade do prefixo (prefixo combinado mais longo = mais específico). Para explicação detalhada e exemplos, consulte o Guia de NAT de Título Global.

Parâmetro	Tipo	Padrão	Obrigatório	Descrição
`gt_nat_enabled`	Boolean	`false`	Não	Habilitar/desabilitar recurso de NAT GT
`gt_nat_rules`	Lista de Mapas	`[]`	**Sim** (se habilitado)	Lista de regras de NAT GT com correspondência de prefixo

Formato da Regra: Cada regra em `gt_nat_rules` deve ser um mapa com:

- `calling_prefix`: Prefixo de string para corresponder ao GT chamador (opcional)
- `called_prefix`: Prefixo de string para corresponder ao GT chamado (opcional)
- `weight`: Prioridade inteira (menor = maior prioridade) - padrão: 100
- `response_gt`: Título Global a ser usado nas respostas (obrigatório)

Prioridade de Correspondência:

1. As regras são correspondidas pelo `weight` (valor mais baixo = maior prioridade)
2. Se os pesos forem iguais, o comprimento do prefixo combinado mais longo vence

3. Ambos `calling_prefix` e `called_prefix` podem ser usados juntos para correspondência precisa

Exemplo:

```
config :omniss7,
  gt_nat_enabled: true,
  gt_nat_rules: [
    # Alta prioridade: Correspondência de ambos chamando de "8772"
E chamado para "555"
    %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

    # Prioridade média: Correspondência apenas chamando de "8772"
    %{calling_prefix: "8772", weight: 10, response_gt:
"68988411553"},

    # Prioridade média: Correspondência apenas chamado para "555"
    %{called_prefix: "555", weight: 10, response_gt:
"68988411554"},

    # Correspondência apenas chamando de "8773"
    %{calling_prefix: "8773", weight: 10, response_gt:
"68988411554"},

    # Regra de fallback wildcard (corresponde a tudo, maior peso)
    %{weight: 100, response_gt: "68988411555"}
  ]
```

Veja Também: Guia de NAT GT para uso detalhado e exemplos.

Parâmetros de Conexão M3UA

Configuração de conexão M3UA para modo cliente MAP. Para uso detalhado e exemplos, consulte o Guia do Cliente MAP.

Parâmetro	Tipo	Padrão	Obrigatór
`map_client_m3ua.mode`	String	-	**Sim**
`map_client_m3ua.callback`	Tupla	-	**Sim**
`map_client_m3ua.process_name`	Atom	-	**Sim**
`map_client_m3ua.local_ip`	Tupla ou Lista	-	**Sim**
`map_client_m3ua.local_port`	Inteiro	`2905`	**Sim**
`map_client_m3ua.remote_ip`	Tupla ou Lista	-	**Sim**
`map_client_m3ua.remote_port`	Inteiro	`2905`	**Sim**
`map_client_m3ua.routing_context`	Inteiro	-	**Sim**
`map_client_m3ua.opc`	Inteiro	`5013`	Não

Parâmetro	Tipo	Padrão	Obrigatór
`map_client_m3ua.dpc`	Inteiro	`5011`	Não
`map_client_m3ua.receive_watchdog`	Boolean	`true`	Não
`map_client_m3ua.receive_watchdog_idle`	Inteiro	`15`	Não

Exemplo (IP Único):

```
config :omniss7,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1,
    opc: 5013,    # Código de Ponto de Origem (2-114-5)
    dpc: 5011     # Código de Ponto de Destino (2-114-3)
  }
```

> **Formato do Código de Ponto**: Códigos de ponto no formato `X-Y-Z` são convertidos em inteiros como: `(X * 2048) + (Y * 8) + Z`. Por exemplo, `2-114-5` = `(2 * 2048) + (114 * 8) + 5` = `5013`.

Exemplo (Multihoming SCTP):

```
config :omniss7,
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    # Múltiplos IPs locais para redundância
    local_ip: [{10, 179, 4, 11}, {10, 179, 4, 12}],
    local_port: 2905,
    # Múltiplos IPs remotos para redundância do STP
    remote_ip: [{10, 179, 4, 10}, {10, 179, 4, 20}],
    remote_port: 2905,
    routing_context: 1
  }
```

> **Nota**: Para informações detalhadas sobre configuração e benefícios do multihoming SCTP, consulte Multihoming SCTP no Guia Comum.

Receive Watchdog

O **watchdog de recebimento** monitora conexões SCTP em busca de sockets zumbis — associações que permanecem em estado ESTABELECIDO no nível do OS, mas onde a extremidade remota parou silenciosamente de enviar dados. Sem o watchdog, uma conexão morta pode não ser detectada até que a próxima tentativa de envio falhe.

Qualquer payload SCTP recebido redefine o temporizador ocioso, incluindo respostas M3UA BEAT, mensagens NOTIFY e dados de aplicação. De acordo com RFC 4666 §3.8, M3UA BEAT é opcional — SCTP já realiza heartbeating obrigatório na camada de transporte. Em ambientes onde o SG remoto não envia tráfego de camada de aplicação periódico (e não envia M3UA BEATs), desabilitar o watchdog evita ciclos de reconexão desnecessários.

Parâmetro	Tipo	Padrão	Descrição
`receive_watchdog`	Boolean	`true`	Habilitar ou desabilitar o watchdog de recebimento. Quando `false`, conexões ociosas nunca são encerradas pelo watchdog; o heartbeating da camada de transporte SCTP ainda opera normalmente.
`receive_watchdog_idle`	Inteiro	`15`	Segundos de inatividade antes que o watchdog encerre a conexão e acione uma reconexão. Não tem efeito quando `receive_watchdog: false`.

Exemplo — Desabilitando o watchdog para um cliente ASP:

```
config :omniss7,
  map_client_m3ua: %{
    mode: "ASP",
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1,
    receive_watchdog: false        # Desabilitar — SG remoto não
envia tráfego periódico
  }
```

Exemplo — Personalizando o limite ocioso:

```
config :omniss7,
  map_client_m3ua: %{
    mode: "ASP",
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1,
    receive_watchdog: true,
    receive_watchdog_idle: 30      # Encerrar após 30 s de
silêncio (padrão: 15 s)
  }
```

Parâmetros de Infraestrutura

Configuração para componentes de infraestrutura do sistema, incluindo licenciamento, interface web, servidor API e registro.

Configuração de Licença

Parâmetro	Tipo	Padrão	Obrigatór
`license_client.license_server_api_urls`	Lista de Strings	-	**Sim**
`license_client.licensee`	String	-	**Sim**

Exemplo:

```
config :license_client,
  license_server_api_urls: ["https://localhost:10443/api"],
  licensee: "Omnitouch Network Services Pty. Ltd."
```

Interface Web do Painel de Controle

Parâmetro	Tipo	P
`control_panel.parent_application_readable_name`	String	`"OmniSS7 St`
`control_panel.use_additional_pages`	Lista de Tuplas	`[]`
`control_panel.page_order`	Lista de Strings	`[]`
`ControlPanelWeb.Endpoint.url.host`	String	`"0.0.0.0"`
`ControlPanelWeb.Endpoint.https.port`	Inteiro	`8087`

Parâmetro	Tipo	P
`ControlPanelWeb.Endpoint.https.keyfile`	String	`"priv/cert/`
`ControlPanelWeb.Endpoint.https.certfile`	String	`"priv/cert/`

Exemplo:

```
config :control_panel, ControlPanelWeb.Endpoint,
  url: [host: "0.0.0.0", path: "/"],
  https: [
    port: 8087,
    keyfile: "priv/cert/omnitouch.pem",
    certfile: "priv/cert/omnitouch.crt"
  ],
  parent_application_readable_name: "OmniSS7 Stack STP"

config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Eventos SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "Peers"}
  ],
  page_order: ["/events", "/m3ua", "/application",
"/configuration"]
```

Servidor API REST

Parâmetro	Tipo	Padrão	Obr
`start_http_server`	Boolean	`true`	Não
`api_ex.api.port`	Inteiro	`8445`	Não
`api_ex.api.listen_ip`	String	`"0.0.0.0"`	Não
`api_ex.api.product_name`	String	`"OmniSS7"`	Não
`api_ex.api.title`	String	`"API - OmniSS7"`	Não
`api_ex.api.hostname`	String	`"localhost"`	Não
`api_ex.api.enable_tls`	Boolean	`true`	Não
`api_ex.api.tls_cert_path`	String	`"priv/cert/omnitouch.crt"`	Não
`api_ex.api.tls_key_path`	String	`"priv/cert/omnitouch.pem"`	Não

Exemplo:

```
config :omniss7,
  start_http_server: true

config :api_ex,
  api: %{
    port: 8445,
    listen_ip: "0.0.0.0",
    product_name: "OmniSS7",
    title: "API - OmniSS7",
    hostname: "localhost",
    enable_tls: true,
    tls_cert_path: "priv/cert/omnitouch.crt",
    tls_key_path: "priv/cert/omnitouch.pem"
  }
```

Endpoints da API:

- API REST: `https://[server-ip]:8445/api/*`
- Swagger UI: `http://[server-ip]:8080/swagger`
- Métricas do Prometheus: `http://[server-ip]:8080/metrics`

Configuração de Registro

Parâmetro	Tipo	Padrão	Obriga
`logger.level`	Atom	`:debug`	Não
`logger.backends`	Lista	`[:console]`	Não
`logger.default_formatter.format`	String	-	Não
`logger.default_formatter.metadata`	Lista	`[]`	Não
`logger.default_formatter.truncate`	Atom/Inteiro	`:infinity`	Não

Exemplo:

```
config :logger,
  level: :debug,
  backends: [:console, SS7.Web.LoggerBackend]

config :logger, :default_formatter,
  format: "[$date] [$time] [$level] $message\n",
  metadata: [:error_code, :file],
  truncate: :infinity
```

Parâmetros de Banco de Dados

Configuração para persistência do banco de dados Mnesia.

Parâmetro	Tipo	Padrão	Obrigatório	Descriç
`mnesia_storage_type`	Atom	`:disc_copies`	Não	Tipo de armazenam do Mnesia: `:disc_copi` ou `:ram_co`

Exemplo:

```
config :omniss7,
  mnesia_storage_type: :disc_copies  # Produção
  # mnesia_storage_type: :ram_copies  # Apenas para teste
```

Tipos de Armazenamento:

- `:disc_copies` - Armazenamento persistente em disco (sobrevive a reinicializações) - **Recomendado para produção**
- `:ram_copies` - Apenas em memória (perdido na reinicialização) - Apenas para teste

Tabelas Mnesia:

- `m3ua_peer` - Conexões de peer M3UA
- `m3ua_route` - Rotas de Código de Ponto
- `m3ua_gt_route` - Rotas de Título Global

Localização: diretório `Mnesia.{node_name}/`

Valores Hardcoded

Os seguintes valores são **hardcoded no código-fonte** e não podem ser alterados via configuração.

Timeouts

Valor	Impacto	Solução Alternativa
Timeout de solicitação MAP: **10 segundos**	Todas as operações MAP expiram após 10s	Modificar o código-fonte
Timeout de ISD: **10 segundos**	Cada mensagem ISD expira após 10s	Modificar o código-fonte

Servidor HTTP

Valor	Impacto	Solução Alternativa
IP HTTP: **0.0.0.0**	Servidor de métricas/Swagger escuta em todas as interfaces	Modificar o código-fonte
Porta HTTP: **8080**	Endpoint de métricas/Swagger executa na porta 8080	Modificar o código-fonte

Intervalos de Registro

Valor	Impacto	Solução Alternativa
Registro do frontend: **5 minutos**	SMSc registra com o backend a cada 5 min	Modificar o código-fonte

Atualização Automática da Interface Web

Página	Intervalo
Gerenciamento de Roteamento	5 segundos
Assinantes Ativos	2 segundos

Exemplos de Configuração

Configuração Mínima do HLR

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false,

  hlr_api_base_url: "https://10.180.2.140:8443",
  hlr_service_center_gt_address: "55512341111",
  smsc_service_center_gt_address: "55512341112",

  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

Configuração Mínima do SMSc

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: true,

  smsc_api_base_url: "https://10.179.3.219:8443",
  smsc_name: "ipsmgw",
  smsc_service_center_gt_address: "55512341112",

  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10,

  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    local_ip: {10, 179, 4, 12},
    local_port: 2905,
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

STP com Servidor Autônomo

```
config :omniss7,
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: false,

  enable_gt_routing: true,
  mnesia_storage_type: :disc_copies,

  sctp_handler: %{
    enabled: true,
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    point_code: 100
  },

  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    local_ip: {10, 179, 4, 10},
    local_port: 2906,
    remote_ip: {10, 179, 4, 11},
    remote_port: 2905,
    routing_context: 1
  }
```

Resumo

Total de Parâmetros de Configuração: 75+

Por Categoria:

- Modo Operacional: 5 parâmetros
- Modo HLR: 17 parâmetros
- Modo SMSc: 8 parâmetros

- Modo STP: 5+ parâmetros (mais listas m3ua_peers, m3ua_routes, m3ua_gt_routes)
- Modo CAMEL GW: 14 parâmetros
- NAT de Título Global: 2 parâmetros
- Conexão M3UA: 8 parâmetros
- Infraestrutura (Licença, Web, API, Registro): 23 parâmetros
- Banco de Dados: 1 parâmetro

Parâmetros Obrigatórios por Modo:

Modo HLR:

- `hlr_api_base_url`
- `hlr_service_center_gt_address`
- `smsc_service_center_gt_address`
- Todos os parâmetros `map_client_m3ua.*` (8)

Modo SMSc:

- `smsc_api_base_url`
- `smsc_service_center_gt_address`
- `auto_flush_dest_smsc` (se auto-flush habilitado)
- Todos os parâmetros `map_client_m3ua.*` (8)

Modo STP:

- `sctp_handler.point_code` (se manipulador SCTP habilitado)
- `sctp_handler.local_ip`
- `sctp_handler.local_port`

Modo CAMEL GW:

- `cgrates_url` (se CGrateS habilitado)
- Todos os parâmetros `cap_client_m3ua.*` (8)

Infraestrutura:

- `license_client.license_server_api_urls`

- `license_client.licensee`

Documentação Relacionada

- **Guia HLR** - Configuração específica do HLR
- **Guia SMSc** - Configuração específica do SMSc
- **Guia STP** - Configuração de roteamento STP
- **Guia API** - Referência da API REST
- **Guia de Gateway USSD** - Configuração do Gateway USSD e protocolo de callback HTTP
- **Guia da Interface Web** - Documentação da interface web

Guia de NAT de Título Global

Visão Geral

A Tradução de Endereço de Título Global (GT NAT) é um recurso que permite que o OmniSS7 responda com diferentes endereços de Título Global com base no prefixo GT da parte chamadora, no prefixo GT da parte chamada ou em uma combinação de ambos. Isso é essencial ao operar com múltiplos Títulos Globais e precisar garantir que as respostas usem o GT correto com base em qual rede ou par está chamando e/ou qual GT eles chamaram.

Novidades (GT NAT Aprimorado)

O recurso GT NAT foi aprimorado com novas capacidades poderosas:

Novos Recursos

1. **Correspondência de Prefixo da Parte Chamado**: As regras agora podem corresponder ao `called_prefix` além do `calling_prefix`
2. **Correspondência Combinada**: As regras podem corresponder a ambos os prefixos chamadores E chamados simultaneamente
3. **Priorização Baseada em Peso**: As regras agora usam um campo `weight` (menor = maior prioridade) em vez de apenas o comprimento do prefixo
4. **Correspondência Flexível**: Agora você pode criar regras com:
   - Apenas prefixo chamador
   - Apenas prefixo chamado
   - Ambos os prefixos chamador e chamado
   - Nenhum (regra de curinga/fallback)

Novo Formato de Regra

Campos obrigatórios:

- `weight`: Prioridade inteira (menor = maior prioridade)
- `response_gt`: O GT para responder

Campos opcionais (pelo menos um recomendado para correspondência específica):

- `calling_prefix`: Correspondência no prefixo GT da parte chamadora
- `called_prefix`: Correspondência no prefixo GT da parte chamada

Exemplo:

```
gt_nat_rules: [
  # Regra específica com ambos os prefixos - maior prioridade
  %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

  # Regras específicas - prioridade média
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},
  %{called_prefix: "555", weight: 10, response_gt: "333333"},

  # Fallback de curinga - menor prioridade
  %{weight: 100, response_gt: "999999"}
]
```

Casos de Uso

Operação Multi-Rede

Quando você tem várias redes parceiras e cada uma espera respostas de um GT específico:

- **Rede A** chama seu GT `111111` e espera respostas de `111111`
- **Rede B** chama seu GT `222222` e espera respostas de `222222`

Sem o GT NAT, você precisaria de instâncias separadas ou roteamento complexo. Com o GT NAT, uma única instância do OmniSS7 pode lidar com isso de forma inteligente.

Cenários de Roaming

Ao operar como um HLR ou SMSc com acordos de roaming:

- Assinantes da **rede doméstica** usam GT `555000`
- **Parceiro de roaming 1** usa GT `555001`
- **Parceiro de roaming 2** usa GT `555002`

O GT NAT garante que cada parceiro receba respostas do GT correto para o qual estão configurados para rotear.

Testes e Migração

Durante migrações de rede ou testes:

- Migre gradualmente o tráfego do GT antigo para o GT novo
- Mantenha ambos os GTs durante o período de transição
- Roteie respostas com base em qual GT o chamador usou

Como Funciona

Fluxo de Tradução de Endereço

1. **Solicitação de Entrada**: O OmniSS7 recebe uma mensagem SCCP com:
    - GT da Parte Chamado: `55512341112` (seu GT)
    - GT da Parte Chamadora: `877234567` (seu GT)
2. **Consulta GT NAT**: O sistema verifica o GT chamador `877234567` em relação às regras de prefixo configuradas
3. **Correspondência de Prefixo**: Encontra o prefixo correspondente mais longo (por exemplo, `8772` corresponde a `877234567`)

4. **Seleção do GT de Resposta**: Usa `response_gt` da regra correspondente (por exemplo, `55512341112`)

5. **Resposta Enviada**: A resposta SCCP usa:

    - GT da Parte Chamado: `877234567` (invertido - seu GT)
    - GT da Parte Chamadora: `55512341112` (GT NAT'd)

Tipos de Resposta Afetados

O GT NAT se aplica a múltiplas camadas da pilha SS7:

Camada SCCP (Todas as Respostas)

- Endereços GT Chamado/Chamador SCCP em todas as mensagens de resposta
- Reconhecimentos de ISD (InsertSubscriberData)
- Respostas de UpdateLocation
- Respostas de erro

Camada MAP (Operação Específica)

- **Respostas SRI-for-SM**: `networkNode-Number` (endereço GT do SMSc)
- **UpdateLocation**: `hlr-Number` nas respostas
- **InsertSubscriberData**: GT HLR nas mensagens ISD

Configuração

Configuração Básica

Adicione ao `config/runtime.exs`:

```
config :omniss7,
  # Habilitar GT NAT
  gt_nat_enabled: true,

  # Definir regras de GT NAT
  gt_nat_rules: [
    # Regra 1: Chamadas do prefixo "8772" recebem resposta do
"55512341112"
    %{calling_prefix: "8772", response_gt: "55512341112"},

    # Regra 2: Chamadas do prefixo "8773" recebem resposta do
"55512341111"
    %{calling_prefix: "8773", response_gt: "55512341111"},

    # Regra padrão (prefixo vazio corresponde a tudo)
    %{calling_prefix: "", response_gt: "55512311555"}
  ]
```

Parâmetros de Configuração

Para referência completa de configuração, consulte Parâmetros de NAT de Título Global na Referência de Configuração.

Parâmetro	Tipo	Obrigatório	Descrição
`gt_nat_enabled`	Booleano	Sim	Habilitar/desabilitar recurso GT NAT
`gt_nat_rules`	Lista de Mapas	Sim (se habilitado)	Lista de regras de correspondência de prefixo

Formato da Regra

Cada regra é um mapa com as seguintes chaves:

```
%{
  calling_prefix: "8772",      # (Opcional) Prefixo para
corresponder ao GT chamador
  called_prefix: "555",        # (Opcional) Prefixo para
corresponder ao GT chamado
  weight: 10,                  # (Obrigatório) Valor de prioridade
(menor = maior prioridade)
  response_gt: "55512341112"   # (Obrigatório) GT a ser usado nas
respostas
}
```

Campos da Regra:

- **`calling_prefix`** (Opcional): Prefixo de string para corresponder ao GT chamador de entrada
  - A correspondência é feita por `String.starts_with?/2`
  - A string vazia `""` ou `nil` atua como curinga (corresponde a qualquer GT chamador)
  - Pode ser omitido para corresponder a qualquer GT chamador
- **`called_prefix`** (Opcional): Prefixo de string para corresponder ao GT chamado de entrada
  - A correspondência é feita por `String.starts_with?/2`
  - A string vazia `""` ou `nil` atua como curinga (corresponde a qualquer GT chamado)
  - Pode ser omitido para corresponder a qualquer GT chamado
- **`weight`** (Obrigatório): Valor de prioridade inteiro
  - Peso menor = maior prioridade (processado primeiro)
  - Deve ser >= 0
  - Usado como critério de ordenação primário para regras de correspondência
- **`response_gt`** (Obrigatório): O endereço de Título Global a ser usado nas respostas

- Deve ser uma string de número E.164 válida
- Deve corresponder a um dos seus GTs configurados

Pelo menos um dos `calling_prefix` ou `called_prefix` deve ser especificado para roteamento específico. Ambos podem ser omitidos para uma regra de curinga/fallback.

Lógica de Correspondência de Regras

As regras são avaliadas por **peso primeiro (crescente), depois pela especificidade combinada do prefixo**:

Algoritmo de Correspondência:

1. Filtrar regras onde todos os prefixos especificados correspondem
   - Se `calling_prefix` estiver definido, deve corresponder ao GT chamador
   - Se `called_prefix` estiver definido, deve corresponder ao GT chamado
   - Se ambos estiverem definidos, ambos devem corresponder
   - Se nenhum estiver definido, a regra atua como um curinga
2. Classificar regras correspondentes por:
   - **Primário**: Peso (crescente - valores menores primeiro)
   - **Secundário**: Comprimento combinado do prefixo (decrescente - mais longo = mais específico)
3. Retornar a primeira regra correspondente

Exemplos:

```
# Regras de exemplo
gt_nat_rules: [
  # Peso 1: Maior prioridade - corresponde a ambos os prefixos
  %{calling_prefix: "8772", called_prefix: "555", weight: 1,
response_gt: "111111"},

  # Peso 10: Prioridade média - regras específicas
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},  #
Apenas chamador
  %{called_prefix: "555", weight: 10, response_gt: "333333"},    #
Apenas chamado

  # Peso 100: Menor prioridade - curinga fallback
  %{weight: 100, response_gt: "444444"}  # Corresponde a tudo
]

# Exemplos de correspondência:
# Chamando: "877234567", Chamado: "555123" -> "111111" (peso 1,
ambos correspondem)
# Chamando: "877234567", Chamado: "999999" -> "222222" (peso 10,
apenas chamador)
# Chamando: "999999999", Chamado: "555123" -> "333333" (peso 10,
apenas chamado)
# Chamando: "999999999", Chamado: "888888" -> "444444" (peso 100,
curinga)
```

Exemplos

Exemplo 1: Dois Parceiros de Rede

Cenário: Você opera um SMSc com dois parceiros de rede. Cada um espera respostas de um GT diferente.

```
config :omniss7,
  gt_nat_enabled: true,

  # GT padrão do SMSc (usado quando o GT NAT está desabilitado ou
nenhuma regra corresponde)
  smsc_service_center_gt_address: "5551000",

  # Regras de GT NAT para parceiros
  gt_nat_rules: [
    # Parceiro A (prefixo 4412) espera respostas do GT 5551001
    %{calling_prefix: "4412", weight: 10, response_gt: "5551001"},

    # Parceiro B (prefixo 4413) espera respostas do GT 5551002
    %{calling_prefix: "4413", weight: 10, response_gt: "5551002"},

    # Padrão: usar GT padrão do SMSc (fallback de curinga)
    %{weight: 100, response_gt: "5551000"}
  ]
```

Fluxo de Tráfego:

```
SRI-for-SM de entrada de 44121234567:
  GT Chamado: 5551001 (seu GT que o Parceiro A usa)
  GT Chamador: 44121234567 (GT do Parceiro A)

Consulta GT NAT:
  "44121234567" corresponde ao prefixo "4412"
  GT de resposta selecionado: "5551001"

Resposta SRI-for-SM para 44121234567:
  GT Chamado: 44121234567 (invertido)
  GT Chamador: 5551001 (NAT'd)
  networkNode-Number: 5551001 (na resposta MAP)
```

Exemplo 2: HLR com GTs Regionais

Cenário: HLR nacional com diferentes GTs por região.

```
config :omniss7,
  gt_nat_enabled: true,
  hlr_service_center_gt_address: "555000",  # GT padrão do HLR

  gt_nat_rules: [
    # VLRs da região Norte (prefixo 5551)
    %{calling_prefix: "5551", weight: 10, response_gt: "555100"},

    # VLRs da região Sul (prefixo 5552)
    %{calling_prefix: "5552", weight: 10, response_gt: "555200"},

    # VLRs da região Oeste (prefixo 5553)
    %{calling_prefix: "5553", weight: 10, response_gt: "555300"},

    # Padrão para outras regiões (curinga)
    %{weight: 100, response_gt: "555000"}
  ]
```

Exemplo 3: Cenário de Migração

Cenário: Migrando gradualmente do GT antigo para o novo GT.

```
config :omniss7,
  gt_nat_enabled: true,
  hlr_service_center_gt_address: "123456789",  # GT antigo
(padrão)

  gt_nat_rules: [
    # Redes migradas (já atualizaram suas configurações)
    %{calling_prefix: "555", weight: 10, response_gt:
"987654321"},  # Novo GT
    %{calling_prefix: "666", weight: 10, response_gt:
"987654321"},  # Novo GT

    # Todos os outros ainda usam o GT antigo (curinga)
    %{weight: 100, response_gt: "123456789"}  # GT antigo
  ]
```

Exemplo 4: Correspondência de Prefixo da Parte Chamado (NOVO)

Cenário: Você tem múltiplos GTs para diferentes serviços e deseja responder com o GT correto com base em qual GT foi chamado.

```
config :omniss7,
  gt_nat_enabled: true,

  gt_nat_rules: [
    # Quando chamam seu GT de SMS (5551xxx), responda com esse GT
    %{called_prefix: "5551", weight: 10, response_gt: "555100"},

    # Quando chamam seu GT de Voz (5552xxx), responda com esse GT
    %{called_prefix: "5552", weight: 10, response_gt: "555200"},

    # Quando chamam seu GT de Dados (5553xxx), responda com esse
GT
    %{called_prefix: "5553", weight: 10, response_gt: "555300"},

    # Fallback padrão
    %{weight: 100, response_gt: "555000"}
  ]
```

Fluxo de Tráfego:

```
Solicitação de entrada para GT Chamado: 555100 (seu GT de SMS)
GT Chamador: 441234567 (qualquer chamador)

Consulta GT NAT:
  GT Chamado "555100" corresponde ao prefixo "5551"
  GT de resposta selecionado: "555100"

A resposta usa GT Chamador: 555100 (corresponde ao que eles
chamaram)
```

Exemplo 5: Correspondência Combinada de Chamador + Chamado (AVANÇADO)

Cenário: Diferentes parceiros chamam diferentes GTs, e você deseja controle detalhado.

```
config :omniss7,
  gt_nat_enabled: true,

  gt_nat_rules: [
    # Parceiro A chamando seu GT de SMS - maior prioridade (peso
1)
    %{calling_prefix: "4412", called_prefix: "5551", weight: 1,
response_gt: "555101"},

    # Parceiro B chamando seu GT de SMS - maior prioridade (peso
1)
    %{calling_prefix: "4413", called_prefix: "5551", weight: 1,
response_gt: "555102"},

    # Qualquer um chamando seu GT de SMS - prioridade média (peso
10)
    %{called_prefix: "5551", weight: 10, response_gt: "555100"},

    # Parceiro A chamando qualquer GT - prioridade média (peso 10)
    %{calling_prefix: "4412", weight: 10, response_gt: "555200"},

    # Fallback padrão - baixa prioridade (peso 100)
    %{weight: 100, response_gt: "555000"}
  ]
```

Exemplos de Correspondência:

```
# Parceiro A chama GT de SMS
Chamando: "441234567", Chamado: "555100"
→ Corresponde à regra de peso 1 (ambos os prefixos) → "555101"

# Parceiro A chama GT de Voz
Chamando: "441234567", Chamado: "555200"
→ Corresponde à regra de peso 10 (apenas chamador) → "555200"

# Chamador desconhecido chama GT de SMS
Chamando: "999999999", Chamado: "555100"
→ Corresponde à regra de peso 10 (apenas chamado) → "555100"

# Chamador desconhecido chama GT de Voz
Chamando: "999999999", Chamado: "555200"
→ Corresponde ao curinga de peso 100 → "555000"
```

Modos Operacionais

O GT NAT funciona em todos os modos operacionais do OmniSS7:

Modo HLR

O GT NAT afeta:

- Respostas de UpdateLocation (GT HLR na resposta)
- Mensagens InsertSubscriberData (GT HLR como parte chamadora)
- Respostas de SendAuthenticationInfo
- Respostas de Cancel Location

Para mais informações sobre operações HLR, consulte o Guia de Configuração do HLR.

Configuração:

```
config :omniss7,
  hlr_mode_enabled: true,
  hlr_service_center_gt_address: "5551234567",  # GT padrão do HLR

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "331", weight: 10, response_gt:
"5551234568"},  # França
    %{calling_prefix: "44", weight: 10, response_gt:
"5551234569"},   # Reino Unido
    %{weight: 100, response_gt: "5551234567"}  # Curinga padrão
  ]
```

Modo SMSc

O GT NAT afeta:

- Respostas SRI-for-SM (campo `networkNode-Number`) - veja Detalhes SRI-for-SM
- Reconhecimentos de MT-ForwardSM

Para mais informações sobre operações SMSc, consulte o Guia de Configuração do SMSc.

Configuração:

```
config :omniss7,
  smsc_mode_enabled: true,
  smsc_service_center_gt_address: "5559999",  # GT padrão do SMSc

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "1", weight: 10, response_gt: "5559991"},
# América do Norte
    %{calling_prefix: "44", weight: 10, response_gt: "5559992"},
# Reino Unido
    %{calling_prefix: "86", weight: 10, response_gt: "5559993"},
# China
    %{weight: 100, response_gt: "5559999"}  # Curinga padrão
  ]
```

Modo Gateway CAMEL

O GT NAT afeta:

- Todas as respostas em nível SCCP (GT gsmSCF como Parte Chamadora)
- Respostas de operações CAMEL/CAP (InitialDP, EventReportBCSM, etc.)
- Reconhecimentos de RequestReportBCSMEvent
- Respostas de ApplyCharging
- Respostas de Continue

Configuração:

```
config :omniss7,
  camelgw_mode_enabled: true,
  camel_gsmscf_gt_address: "55512341112",  # GT padrão gsmSCF

  gt_nat_enabled: true,
  gt_nat_rules: [
    %{calling_prefix: "555", weight: 10, response_gt:
"55512341111"},  # Rede A
    %{calling_prefix: "666", weight: 10, response_gt:
"55512311555"},  # Rede B
    %{weight: 100, response_gt: "55512341112"}  # Curinga padrão
  ]
```

Caso de Uso: Ao operar como um gsmSCF (Função de Controle de Serviço) para várias redes, cada gsmSSF da rede pode esperar respostas de um GT gsmSCF específico. O GT NAT garante que o GT correto seja usado com base em qual gsmSSF está chamando.

Registro e Depuração

Habilitar Registro do GT NAT

O GT NAT inclui registro automático de todas as traduções:

```
# Nos logs, você verá:
[info] GT NAT [resposta SRI-for-SM]: GT Chamador 877234567 -> GT
de Resposta 55512341112
[info] GT NAT [UpdateLocation ISD]: GT Chamador 331234567 -> GT de
Resposta 55512341111
[info] GT NAT [resposta MAP BEGIN]: GT Chamador 441234567 -> GT de
Resposta 55512311555
```

O campo de contexto mostra onde o NAT foi aplicado:

- `"resposta SRI-for-SM"` - No manipulador SRI-for-SM
- `"UpdateLocation ISD"` - Nas mensagens InsertSubscriberData
- `"UpdateLocation END"` - Na resposta UpdateLocation END
- `"resposta MAP BEGIN"` - Respostas MAP BEGIN genéricas
- `"ISD ACK"` - Reconhecimento ISD
- `"resposta de erro HLR"` - Resposta de erro do HLR
- `"resposta CAMEL"` - Respostas de operações CAMEL/CAP (gsmSCF)

Validação

O sistema valida a configuração do GT NAT na inicialização:

```
# Verifique a configuração do GT NAT
iex> GtNat.validate_config()
{:ok, [
  %{calling_prefix: "8772", weight: 10, response_gt:
"55512341112"},
  %{calling_prefix: "8773", weight: 10, response_gt:
"55512341111"}
]}

# Verifique se está habilitado
iex> GtNat.enabled?()
true

# Obtenha todas as regras
iex> GtNat.get_rules()
[
  %{calling_prefix: "8772", weight: 10, response_gt:
"55512341112"},
  %{calling_prefix: "8773", weight: 10, response_gt:
"55512341111"}
]
```

Testando o GT NAT

Teste a lógica do GT NAT programaticamente:

```
# Teste a tradução com GT chamador apenas (called_gt é nil)
iex> GtNat.translate_response_gt("877234567", nil, "default_gt")
"55512341112"

# Teste a tradução com ambos os GTs chamador e chamado
iex> GtNat.translate_response_gt("877234567", "555123",
"default_gt")
"55512341112"

# Teste com registro (GT chamado nil)
iex> GtNat.translate_response_gt_with_logging("877234567", nil,
"default_gt", "test")
# Logs: GT NAT [test]: GT Chamador 877234567 -> GT de Resposta
55512341112
"55512341112"

# Teste com registro (ambos os GTs)
iex> GtNat.translate_response_gt_with_logging("877234567",
"555123", "default_gt", "test")
# Logs: GT NAT [test]: GT Chamador 877234567, GT Chamado 555123 ->
GT de Resposta 55512341112
"55512341112"

# Teste sem correspondência (retorna padrão)
iex> GtNat.translate_response_gt("999999999", "888888",
"default_gt")
"default_gt"
```

Resolução de Problemas

Problema: GT NAT Não Funcionando

Verifique 1: Está habilitado?

```
iex> Application.get_env(:omniss7, :gt_nat_enabled)
true  # Deve ser true
```

Verifique 2: As regras estão configuradas?

```
iex> Application.get_env(:omniss7, :gt_nat_rules)
[%{calling_prefix: "8772", response_gt: "55512341112"}, ...]  #
Deve retornar lista
```

Verifique 3: Verifique os logs Procure por "GT NAT" nos logs para ver se as traduções estão acontecendo.

Problema: GT Errado nas Respostas

Sintoma: Respostas usam endereço GT inesperado

Causa: A correspondência de prefixo da regra pode ser muito ampla ou a regra padrão está capturando o tráfego

Solução: Revise os pesos e prefixos das regras:

```
# RUIM: Curinga com peso baixo (captura tudo primeiro)
gt_nat_rules: [
  %{weight: 1, response_gt: "111111"},              # Isso
corresponde a tudo primeiro!
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"}  #
Nunca alcançado
]

# BOM: Regras específicas com peso mais baixo, curinga com peso
mais alto
gt_nat_rules: [
  %{calling_prefix: "8772", weight: 10, response_gt: "222222"},  #
Específico, peso baixo
  %{weight: 100, response_gt: "111111"}  # Curinga, peso alto
(fallback)
]
```

Problema: GT NAT Não Aplicado a Tipo de Mensagem Específico

Sintoma: Algumas respostas usam GT NAT'd, outras não

Cobertura Atual:

- □ GT Chamador SCCP (todas as respostas)
- □ Respostas SRI-for-SM (networkNode-Number)
- □ Mensagens UpdateLocation ISD (GT HLR)
- □ Respostas UpdateLocation END
- □ Reconhecimentos ISD
- □ Respostas MAP BEGIN

Se um tipo de mensagem específico não estiver usando GT NAT, pode não estar implementado ainda. Verifique o código-fonte ou entre em contato com o suporte.

Considerações de Desempenho

Desempenho de Consulta

O GT NAT usa correspondência de prefixo simples com complexidade O(n), onde n é o número de regras.

Dicas de desempenho:

- Mantenha a contagem de regras abaixo de 100 para melhor desempenho
- Use prefixos específicos para reduzir a contagem de regras
- A regra padrão (prefixo vazio) deve ser a última

Benchmark (sistema típico):

- 10 regras: < 1µs por consulta
- 50 regras: < 5µs por consulta
- 100 regras: < 10µs por consulta

Uso de Memória

Cada regra requer ~100 bytes de memória:

- 10 regras ≈ 1 KB
- 100 regras ≈ 10 KB

Melhores Práticas

1. Sempre Inclua uma Regra de Fallback de Curinga

```
gt_nat_rules: [
  %{calling_prefix: "8772", weight: 10, response_gt: "111111"},
  %{calling_prefix: "8773", weight: 10, response_gt: "222222"},
  %{weight: 100, response_gt: "default_gt"}  # Sempre tenha um
curinga com peso alto
]
```

2. Use Prefixos e Pesos Significativos

```
# BOM: Prefixos claros e específicos com pesos apropriados
%{calling_prefix: "331", weight: 10, response_gt: "..."}  # França
%{calling_prefix: "44", weight: 10, response_gt: "..."}   # Reino
Unido

# RUIM: Prefixos excessivamente amplos ou pesos confusos
%{calling_prefix: "3", weight: 5, response_gt: "..."}     # Muitos
países
%{calling_prefix: "331", weight: 100, response_gt: "..."}  # Peso
deve ser menor para regras específicas
```

3. Documente Suas Regras

```
gt_nat_rules: [
  # Parceiro XYZ - rede do Reino Unido (faixa de GT: 4412xxxxxxx)
  # Peso 10: Prioridade padrão do parceiro
  %{calling_prefix: "4412", weight: 10, response_gt: "5551001"},

  # Parceiro ABC - rede da França (faixa de GT: 33123xxxxxx)
  # Peso 10: Prioridade padrão do parceiro
  %{calling_prefix: "33123", weight: 10, response_gt: "5551002"}
]
```

4. Teste Antes da Implantação

```
# Teste no iex antes de implantar
iex> GtNat.translate_response_gt("44121234567", nil, "default")
"5551001"  # Resultado esperado

# Teste com GT chamado
iex> GtNat.translate_response_gt("44121234567", "555123",
"default")
"5551001"  # Resultado esperado
```

5. Monitore os Logs

Habilite o registro em nível INFO para ver todas as traduções do GT NAT em produção.

Integração com Outros Recursos

Modo STP

O GT NAT funciona independentemente do roteamento STP. O STP roteia com base em códigos de ponto e GTs de destino, enquanto o GT NAT lida com endereçamento de resposta.

Para mais informações sobre roteamento STP, consulte o Guia de Configuração do STP.

Integração CAMEL

O GT NAT está **totalmente integrado** com operações CAMEL/CAP:

Camada SCCP:

- GT da Parte Chamadora em todas as respostas CAMEL
- Aplicado automaticamente com base no GT gsmSSF de entrada

Configuração:

- `camel_gsmscf_gt_address` - GT padrão gsmSCF (opcional)
- Se não configurado, usa o GT Chamado da solicitação de entrada
- As regras do GT NAT substituem o padrão com base no prefixo da parte chamadora

Exemplo:

```
# Quando gsmSSF 555123456 chama seu gsmSCF
# Entrada: Chamado=55512341112, Chamando=555123456
# Consulta GT NAT: "555" -> response_gt="55512341111"
# Resposta: Chamado=555123456, Chamando=55512341111
```

Balanceamento de Carga

O GT NAT pode ser combinado com balanceamento de carga M3UA para gerenciamento avançado de tráfego.

Guia de Migração

Habilitando o GT NAT em Sistema Existente

1. **Prepare a Configuração**

```
# Adicione ao runtime.exs (mantenha desabilitado inicialmente)
config :omniss7,
  gt_nat_enabled: false,  # Comece desabilitado
  gt_nat_rules: [
    # Suas regras aqui com pesos
    %{calling_prefix: "877", weight: 10, response_gt:
"111111"},
    %{weight: 100, response_gt: "999999"}  # Fallback de
curinga
  ]
```

2. **Teste a Configuração**

```
# Valide se a configuração compila
mix compile

# Teste no iex
iex -S mix
iex> GtNat.validate_config()
```

3. **Habilite em Staging**

```
gt_nat_enabled: true  # Altere para true
```

4. **Monitore os Logs**

```
tail -f log/omniss7.log | grep "GT NAT"
```

5. **Implante em Produção**

   - Implemente durante a janela de manutenção
   - Monitore as primeiras 24 horas de perto
   - Tenha um plano de reversão pronto (defina `gt_nat_enabled: false`)

Suporte

Para problemas ou perguntas:

- Verifique os logs para mensagens "GT NAT"
- Valide a configuração com `GtNat.validate_config()`
- Revise a seção de resolução de problemas deste guia
- Entre em contato com o suporte do OmniSS7 com trechos de log

Guia de Configuração do HLR

← Voltar à Documentação Principal

Este guia fornece a configuração para usar o OmniSS7 como um **Home Location Register (HLR/HSS)** com **OmniHSS** como o banco de dados de assinantes de backend.

Integração do OmniHSS

O modo HLR do OmniSS7 funciona como uma interface de sinalização SS7 que se conecta ao **OmniHSS**, um servidor de assinantes doméstico (HSS) completo. Esta arquitetura separa as preocupações:

- **OmniSS7 (Frontend HLR)**: Gerencia todo o sinal de protocolo SS7/MAP, roteamento SCCP e comunicação de rede
- **OmniHSS (Backend HSS)**: Gerencia dados de assinantes, autenticação, provisionamento e recursos avançados

Por que o OmniHSS?

O OmniHSS fornece gerenciamento de assinantes de nível de operadora com recursos incluindo:

- **Suporte Multi-IMSI**: Cada assinante pode ter múltiplos IMSIs associados a um único MSISDN para roaming internacional, troca de rede e provisionamento de eSIM
- **Autenticação Flexível**: Suporte para algoritmos de autenticação Milenage (3G/4G/5G) e COMP128 (2G)
- **Rastreamento de Sessão de Circuito e Pacote**: Rastreamento independente de registros de rede CS (comutação de circuito) e PS (comutação de pacotes)

- **Provisionamento Avançado**: Perfis de serviço personalizáveis, serviços suplementares e dados de assinatura CAMEL
- **Design API-First**: API RESTful HTTP para integração com sistemas de cobrança, CRM e provisionamento
- **Atualizações em Tempo Real**: Rastreamento de localização, gerenciamento de sessões e geração de vetores de autenticação

Todos os dados de assinantes, credenciais de autenticação e configurações de serviço são armazenados e gerenciados no OmniHSS. O OmniSS7 consulta o OmniHSS por meio de chamadas de API HTTPS para responder a operações MAP como UpdateLocation, SendAuthenticationInfo e SendRoutingInfo.

Importante: O modo HLR do OmniSS7 é um **frontend de sinalização apenas**. Toda a lógica de gerenciamento de assinantes, algoritmos de autenticação, regras de provisionamento e operações de banco de dados são tratadas pelo OmniHSS. Este guia cobre a configuração do protocolo SS7/MAP no OmniSS7. Para informações sobre provisionamento de assinantes, configuração de autenticação, perfis de serviço e operações administrativas, **consulte a documentação do OmniHSS**.

Suporte Multi-IMSI

O OmniHSS suporta nativamente configurações Multi-IMSI, permitindo que um único assinante (identificado por MSISDN) tenha múltiplos IMSIs. Isso possibilita:

- **Perfis de Roaming Internacional**: Diferentes IMSIs para diferentes regiões para reduzir custos de roaming
- **eSIM Multi-Perfil**: Múltiplos perfis de rede em um único dispositivo compatível com eSIM
- **Troca de Rede**: Troca sem interrupções entre redes sem mudar o MSISDN
- **Coordenação de Dual SIM**: Coordenação entre múltiplos SIMs físicos ou virtuais
- **Teste e Desenvolvimento**: Múltiplos IMSIs de teste apontando para o mesmo assinante

Como funciona:

- Cada IMSI tem suas próprias credenciais de autenticação (Ki, OPc, algoritmo)
- Cada IMSI pode ter registros de sessão de circuito e pacote independentes
- Serviços e perfis de assinantes podem ser compartilhados ou personalizados por IMSI
- O OmniSS7 consulta o OmniHSS por IMSI, e o OmniHSS retorna os dados de assinante apropriados
- Sistemas de cobrança podem rastrear o uso por IMSI enquanto associam todos os IMSIs a uma única conta

Exemplo de cenário Multi-IMSI:

```
Assinante MSISDN: +1-555-123-4567
├─ IMSI 1: 310260123456789 (Rede Doméstica dos EUA - autenticação
Milenage)
├─ IMSI 2: 208011234567890 (Perfil de Roaming na França -
autenticação Milenage)
└─ IMSI 3: 440201234567891 (Perfil de Roaming no Reino Unido -
autenticação COMP128)
```

Todos os três IMSIs podem ser usados independentemente para registro na rede, mas todos pertencem à mesma conta de assinante. O OmniHSS gerencia o mapeamento IMSI-para-assinante e garante a autenticação e provisionamento adequados para cada IMSI.

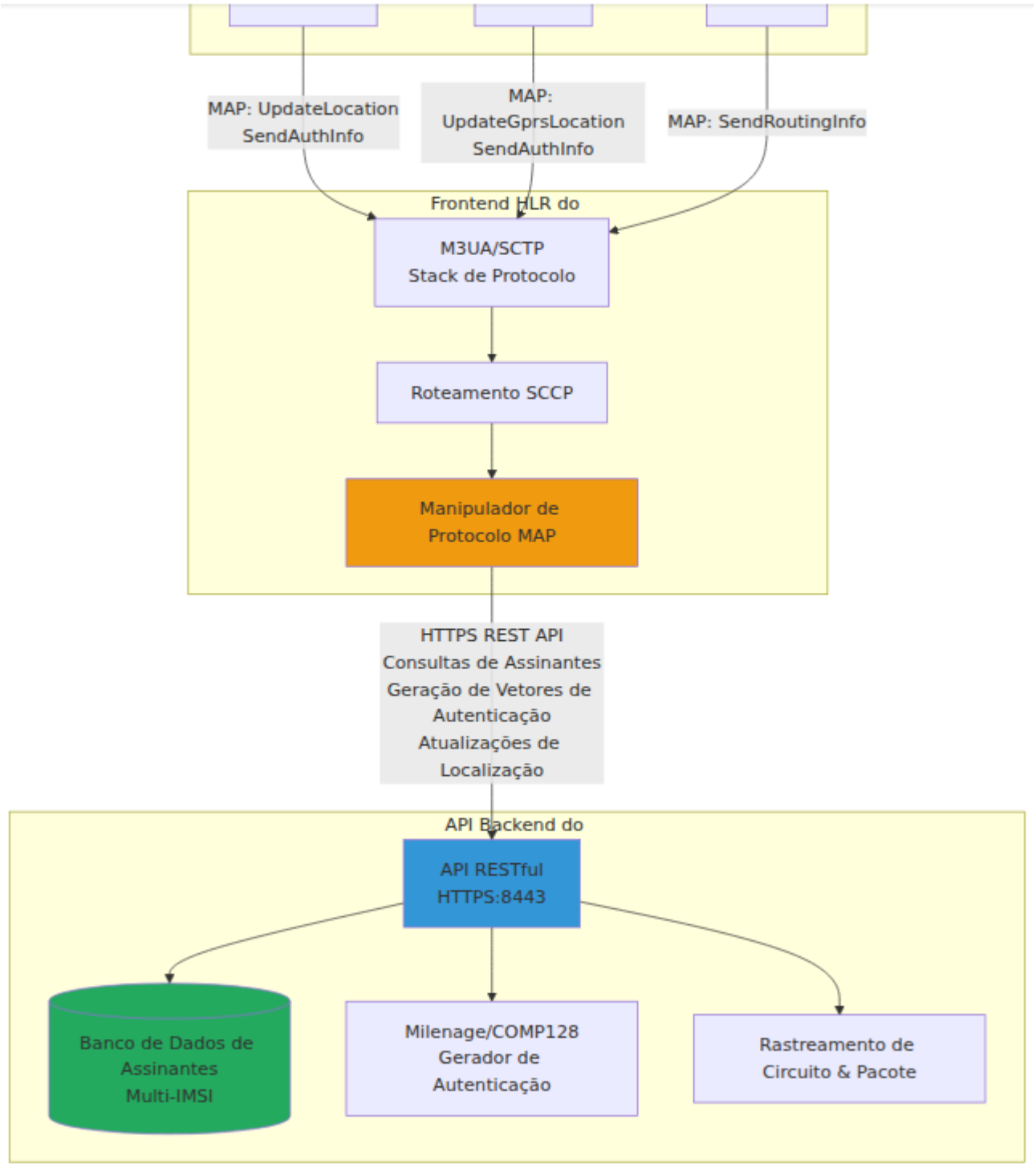
Índice

O que é o Modo HLR?

Modo HLR permite que o OmniSS7 funcione como um Home Location Register para:

- **Gerenciamento de Assinantes**: Armazenar e gerenciar dados de assinantes
- **Autenticação**: Gerar vetores de autenticação para acesso à rede
- **Rastreamento de Localização**: Processar atualizações de localização de VLRs
- **Informações de Roteamento**: Fornecer informações de roteamento para chamadas e SMS

Arquitetura HLR



Habilitando o Modo HLR

O OmniSS7 pode operar em diferentes modos. Para usá-lo como um HLR, você precisa habilitar o modo HLR na configuração.

Mudando para o Modo HLR

O `config/runtime.exs` do OmniSS7 contém três modos operacionais pré-configurados. Para habilitar o modo HLR:

1. **Abra** `config/runtime.exs`
2. **Encontre** as três seções de configuração (linhas 53-174):
   - Configuração 1: Modo STP (linhas 53-85)
   - Configuração 2: Modo HLR (linhas 87-123)
   - Configuração 3: Modo SMSc (linhas 125-174)
3. **Comente** a configuração atualmente ativa (adicione `#` a cada linha)
4. **Descomente** a configuração HLR (remova `#` das linhas 87-123)
5. **Personalize** os parâmetros de configuração conforme necessário
6. **Reinicie** o aplicativo: `iex -S mix`

Configuração do Modo HLR

A configuração completa do HLR fica assim:

```
config :omniss7,
  # Flags de modo - Habilitar recursos HLR apenas
  map_client_enabled: true,
  hlr_mode_enabled: true,
  smsc_mode_enabled: false,

  # Configuração da API Backend do OmniHSS
  hlr_api_base_url: "https://10.180.2.140:8443",

  # Endereço GT do Centro de Serviço HLR para operações de SMS
  hlr_service_center_gt_address: "1234567890",

  # Configuração de Mapeamento MSISDN ↔ IMSI
  # Veja: seção de Mapeamento MSISDN ↔ IMSI para detalhes
  hlr_imsi_plmn_prefix: "50557",
  hlr_msisdn_country_code: "61",
  hlr_msisdn_nsn_offset: 0,
  hlr_msisdn_nsn_length: 9,

  # Configuração do InsertSubscriberData
  # Modo de Acesso à Rede: :packetAndCircuit, :packetOnly, ou
:circuitOnly
  isd_network_access_mode: :packetAndCircuit,

  # Enviar ISD #2 (dados de Serviços Suplementares)
  isd_send_ss_data: true,

  # Enviar ISD #3 (dados de Bloqueio de Chamadas)
  isd_send_call_barring: true,

  # Configuração CAMEL (para respostas SendRoutingInfo)
  # Chave de Serviço para iniciação de serviço CAMEL
  camel_service_key: 11_110,

  # Ponto de Detecção de Gatilho CAMEL
  # Opções: :termAttemptAuthorized, :tBusy, :tNoAnswer, :tAnswer
  camel_trigger_detection_point: :termAttemptAuthorized,

  # Prefixos VLR Domésticos
  # Lista de prefixos de endereço VLR que são considerados "rede"
doméstica
  # Se o VLR do assinante começar com um desses prefixos, use a
resposta SRI padrão
```

```
  # Caso contrário, o assinante está em roaming e precisamos
enviar PRN para obter MSRN
  home_vlr_prefixes: ["123456"],

  # Configuração de Conexão M3UA
  # Conectar como ASP para receber operações MAP (UpdateLocation,
SendAuthInfo, etc.)
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :hlr_client_asp,
    # Endpoint local (sistema HLR)
    local_ip: {10, 179, 4, 11},
    local_port: 2905,
    # Endpoint remoto STP
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }
```

Parâmetros de Configuração a Personalizar

Para uma referência completa de todos os parâmetros de configuração, consulte a Referência de Configuração.

Parâmetro	Tipo	Padrão
`hlr_api_base_url`	String	*Obrigatório*
`hlr_service_center_gt_address`	String	*Obrigatório*
`smsc_service_center_gt_address`	String	*Obrigatório*
`hlr_smsc_alert_gts`	Lista	`[]`
`hlr_alert_location_expiry_seconds`	Inteiro	`172800`
`hlr_imsi_plmn_prefix`	String	`"50557"`
`hlr_msisdn_country_code`	String	`"61"`

Parâmetro	Tipo	Padrão
`hlr_msisdn_nsn_offset`	Inteiro	`0`
`hlr_msisdn_nsn_length`	Inteiro	`9`
`isd_network_access_mode`	Átomo	`:packetAndCircuit`
`isd_send_ss_data`	Booleano	`true`
`isd_send_call_barring`	Booleano	`true`
`camel_service_key`	Inteiro	`11_110`
`camel_trigger_detection_point`	Átomo	`:termAttemptAuthorized`

Parâmetro	Tipo	Padrão
`home_vlr_prefixes`	Lista	`["5551231"]`
`local_ip`	Tupla	*Obrigatório*
`local_port`	Inteiro	`2905`
`remote_ip`	Tupla	*Obrigatório*
`remote_port`	Inteiro	`2905`
`routing_context`	Inteiro	`1`

O que Acontece Quando o Modo HLR Está Habilitado

Quando `hlr_mode_enabled: true`, a interface web mostrará:

- **Eventos SS7** - Registro de eventos
- **Cliente SS7** - Teste de operações MAP
- **M3UA** - Status da conexão
- **Links HLR** - Status da API HLR + gerenciamento de assinantes ← *Específico do HLR*
- **Recursos** - Monitoramento do sistema
- **Configuração** - Visualizador de configuração

As abas **Roteamento**, **Teste de Roteamento** e **Links SMSc** estarão ocultas.

Notas Importantes

- **Configuração Obrigatória**: O parâmetro `hlr_service_center_gt_address` é **obrigatório**. O aplicativo não iniciará se não estiver configurado.
- **Backend OmniHSS**: A API do OmniHSS deve ser acessível no `hlr_api_base_url` configurado
- **Timeout de Requisição da API**: Todas as requisições da API do OmniHSS têm um **timeout fixo de 5 segundos**
- **Timeout de Requisição MAP**: Todas as requisições MAP (SRI, UpdateLocation, SendAuthInfo, etc.) têm um **timeout fixo de 10 segundos**
- **Timeout de ISD**: Cada mensagem InsertSubscriberData (ISD) em uma sequência UpdateLocation tem um **timeout fixo de 10 segundos**
- A conexão M3UA com o STP é necessária para receber operações MAP
- Após mudar de modos, você deve reiniciar o aplicativo para que as alterações tenham efeito
- **Interface Web**: Consulte o Guia da Interface Web para informações sobre como usar a interface web
- **Acesso à API**: Consulte o Guia da API para documentação da API REST e acesso ao Swagger UI

Banco de Dados de Assinantes

O OmniHSS gerencia todos os dados de assinantes incluindo identidades, credenciais de autenticação, perfis de serviço e informações de localização. O OmniSS7 recupera esses dados por meio de chamadas de API RESTful.

Modelo de Assinante do OmniHSS

O OmniHSS armazena informações abrangentes sobre assinantes:

- **Múltiplos IMSIs por assinante**: Suporte para configurações Multi-IMSI (eSIM, perfis de roaming, troca de rede)

- **Credenciais de autenticação**: Ki, OPc e seleção de algoritmo (Milenage ou COMP128)
- **Perfis de serviço**: Categoria de assinante, serviços permitidos, parâmetros de QoS
- **Rastreamento de localização**: Rastreamento independente da VLR/MSC atual (sessão de circuito) e SGSN/GGSN (sessão de pacote)
- **Dados de assinatura CAMEL**: Chaves de serviço, pontos de gatilho e endereços gsmSCF
- **Serviços suplementares**: Encaminhamento de chamadas, bloqueio, espera, configurações CLIP/CLIR
- **Estado administrativo**: Habilitado/desabilitado, restrições de serviço, datas de expiração

---

Vetores de Autenticação

Gerar Vetores de Autenticação

O OmniHSS gera vetores de autenticação usando os algoritmos Milenage ou COMP128 com base no método de autenticação configurado para cada assinante. Quando o OmniSS7 recebe requisições MAP **sendAuthenticationInfo**:

1. O OmniSS7 extrai o IMSI da requisição MAP
2. O OmniSS7 chama a API do OmniHSS para gerar vetores de autenticação
3. O OmniHSS recupera as credenciais Ki e OPc do assinante
4. O OmniHSS gera o número solicitado de vetores (RAND, XRES, CK, IK, AUTN)
5. O OmniSS7 codifica os vetores no formato MAP e os retorna ao VLR/SGSN solicitante

Integração da API OmniHSS

O OmniSS7 se comunica com o OmniHSS via API REST HTTPS para recuperar informações de assinantes, atualizar dados de localização e gerar vetores de

autenticação:

```
config :omniss7,
  hlr_api_base_url: "https://omnihss-server:8443"
```

Quando o OmniSS7 recebe operações MAP da rede SS7, ele consulta o OmniHSS para:

- **Recuperar dados de assinante** por IMSI ou MSISDN
- **Gerar vetores de autenticação** usando credenciais Ki/OPc armazenadas
- **Atualizar a localização da sessão de circuito** quando os assinantes realizam UpdateLocation
- **Verificar o status do assinante** e direitos de serviço

Atualizações de Localização

Processamento de Atualização de Localização

Ao receber requisições MAP **updateLocation**, o OmniSS7 coordena com o OmniHSS para registrar o assinante em um novo VLR:

1. **Extrair informações de localização** da requisição UpdateLocation (IMSI, novo GT VLR, novo GT MSC)
2. **Consultar o OmniHSS** para verificar se o assinante existe e está habilitado
3. **Atualizar a sessão de circuito** no OmniHSS com a nova localização VLR/MSC
4. **Enviar mensagens InsertSubscriberData (ISD)** para provisionar o assinante no novo VLR
5. **Retornar a resposta UpdateLocation** ao VLR (inclui GT HLR do `hlr_service_center_gt_address`)
6. **Enviar alertServiceCenter** para os GTs do SMSc configurados (se `hlr_smsc_alert_gts` estiver preenchido)

Nota: O parâmetro de configuração `hlr_service_center_gt_address` especifica o Título Global do HLR que é retornado nas respostas UpdateLocation. Isso permite que o VLR/MSC identifique e roteie mensagens de volta para este HLR.

Integração do Centro de Serviço de Alerta

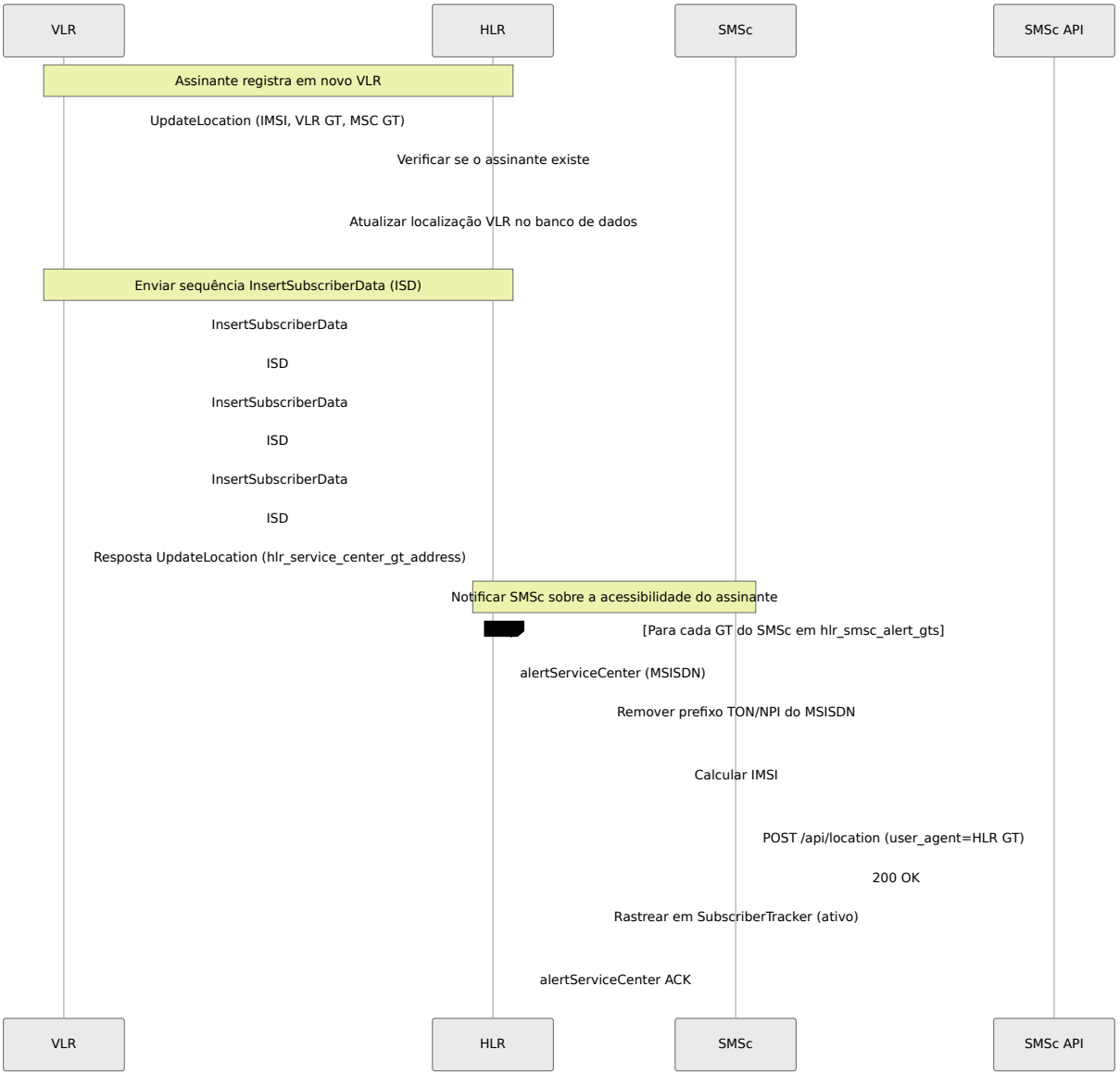
Após uma atualização de localização bem-sucedida, o HLR pode notificar automaticamente os sistemas SMSc de que um assinante agora está acessível, enviando mensagens **alertServiceCenter** (opcode MAP 64). Para informações sobre como o SMSc lida com esses alertas, consulte Tratamento de Centro de Serviço de Alerta no Guia SMSc.

Configuração

Configure a lista de Títulos Globais do SMSc a serem notificados:

```
config :omniss7,
  # Lista de GTs do SMSc para enviar alertServiceCenter após
UpdateLocation
  hlr_smsc_alert_gts: [
    "15559876543",
    "15559876544"
  ],

  # Tempo de expiração de localização quando o SMSc recebe
alertServiceCenter (padrão: 48 horas)
  hlr_alert_location_expiry_seconds: 172800
```

Diagrama de Fluxo



Comportamento

Quando um assinante realiza UpdateLocation:

1. O HLR envia alertServiceCenter para **cada** GT do SMSc na lista `hlr_smsc_alert_gts`
2. A mensagem inclui o MSISDN do assinante
3. O HLR usa `hlr_service_center_gt_address` como o GT da parte chamadora
4. Endereçamento SCCP: SSN chamador=6 (HLR), SSN chamado=8 (SMSc)

O SMSc recebe o alerta e:

- **Remove o prefixo TON/NPI** do MSISDN (por exemplo, "19123123213" → "123123213")
- Marca o assinante como acessível em seu banco de dados de localização (via POST para /api/location)
- **Define o campo** `user_agent` para o GT do HLR ao chamar a API (para rastrear qual HLR enviou o alerta)
- Define o tempo de expiração da localização com base em `hlr_alert_location_expiry_seconds`
- Rastreia o assinante no Tracker de Assinantes do SMSc para monitoramento

Testando

Use a página **Assinantes Ativos** na interface web para enviar manualmente mensagens alertServiceCenter para testes:

1. Navegue até a aba "Assinantes Ativos"
2. Encontre a seção "Testar Centro de Serviço de Alerta"
3. Insira MSISDN, GT do SMSc e GT do HLR (os padrões são preenchidos a partir da configuração)
   - GT do SMSc padrão é a primeira entrada em `hlr_smsc_alert_gts`
   - GT do HLR padrão é `hlr_service_center_gt_address`
4. Clique em "Enviar alertServiceCenter"

Isso é útil para testar o tratamento de alertas do SMSc sem exigir um fluxo completo de UpdateLocation. O formulário usa validação `phx-blur` para evitar mostrar erros enquanto digita.

Configuração do InsertSubscriberData (ISD)

Após uma atualização de localização bem-sucedida, o HLR envia dados de provisionamento do assinante para o VLR usando mensagens **InsertSubscriberData** (ISD). A configuração do ISD permite que você personalize quais dados são enviados e como.

Para referência de parâmetros de configuração, consulte Configuração do ISD na Referência de Configuração.

Sequência ISD

O HLR pode enviar até 3 mensagens ISD sequenciais:

1. **ISD #1** (Sempre enviado) - Dados básicos do assinante:
    - IMSI
    - MSISDN
    - Categoria do assinante
    - Status do assinante (serviceGranted)
    - Lista de serviços de portadora
    - Lista de teleserviços
    - Modo de acesso à rede
2. **ISD #2** (Opcional) - Dados de Serviços Suplementares (SS):
    - Configurações de encaminhamento de chamadas (incondicional, ocupado, sem resposta, não acessível)
    - Espera de chamada
    - Retenção de chamada
    - Serviço de múltiplos participantes
    - Status e recursos de serviços suplementares
3. **ISD #3** (Opcional) - Dados de Bloqueio de Chamadas:
    - Bloqueio de todas as chamadas de saída (BAOC)
    - Bloqueio de chamadas internacionais de saída (BOIC)
    - Dados de restrição de acesso

Opções de Configuração

```
# Configuração do InsertSubscriberData
# Modo de Acesso à Rede: :packetAndCircuit, :packetOnly, ou
:circuitOnly
isd_network_access_mode: :packetAndCircuit,

# Enviar ISD #2 (dados de Serviços Suplementares)
isd_send_ss_data: true,

# Enviar ISD #3 (dados de Bloqueio de Chamadas)
isd_send_call_barring: true,
```

Modo de Acesso à Rede

O parâmetro `isd_network_access_mode` controla que tipo de acesso à rede o assinante está autorizado:

Valor	Descrição	Caso de Uso
`:packetAndCircuit`	Tanto comutação de pacotes (GPRS/LTE) quanto comutação de circuitos (voz)	Padrão - Assinantes de serviço completo
`:packetOnly`	Apenas comutação de pacotes (dados/LTE)	Cartões SIM apenas para dados, dispositivos IoT
`:circuitOnly`	Apenas comutação de circuitos (voz/SMS)	Dispositivos legados, planos apenas de voz

Controlando Mensagens ISD

Você pode controlar quais mensagens ISD são enviadas com base em suas necessidades de rede:

Enviar todos os ISDs (Padrão - Conjunto completo de recursos):

```
isd_send_ss_data: true,
isd_send_call_barring: true,
```

Enviar apenas dados básicos do assinante (Provisionamento mínimo):

```
isd_send_ss_data: false,
isd_send_call_barring: false,
```

Enviar básico + serviços suplementares (Sem bloqueio de chamadas):

```
isd_send_ss_data: true,
isd_send_call_barring: false,
```

Exemplo de Fluxo ISD

Quando UpdateLocation é recebido:

```
VLR → HLR: UpdateLocation (INÍCIO)
HLR → VLR: InsertSubscriberData #1 (CONTINUAR) - Dados básicos
VLR → HLR: ISD #1 ACK (CONTINUAR)
HLR → VLR: InsertSubscriberData #2 (CONTINUAR) - Dados SS [se
habilitado]
VLR → HLR: ISD #2 ACK (CONTINUAR)
HLR → VLR: InsertSubscriberData #3 (CONTINUAR) - Bloqueio de
chamadas [se habilitado]
VLR ��� HLR: ISD #3 ACK (CONTINUAR)
HLR → VLR: Resposta UpdateLocation (FIM)
```

Se `isd_send_ss_data` ou `isd_send_call_barring` estiverem definidos como `false`, essas mensagens ISD são puladas, e o FIM de UpdateLocation é enviado mais cedo.

Melhores Práticas

- **Configuração Padrão**: Use `:packetAndCircuit` e habilite todos os ISDs para máxima compatibilidade

- **IoT/M2M**: Use `:packetOnly` e desabilite dados SS/bloqueio de chamadas para dispositivos apenas de dados
- **Interoperabilidade**: Alguns VLRs mais antigos podem não suportar todos os serviços suplementares - desabilite `isd_send_ss_data` se encontrar problemas
- **Desempenho**: Desabilitar ISDs não utilizados reduz a sobrecarga de mensagens e acelera as atualizações de localização

---

Integração CAMEL

Configuração CAMEL para SendRoutingInfo

Ao responder a requisições **SendRoutingInfo** (SRI) de um GMSC (Gateway MSC), o HLR pode instruir o GMSC a invocar serviços CAMEL para roteamento inteligente de chamadas e controle de serviços.

Para referência de parâmetros de configuração, consulte Configuração CAMEL na Referência de Configuração.

O que é CAMEL?

CAMEL (Aplicações Personalizadas para Lógica Aprimorada de Redes Móveis) é um protocolo que permite serviços de rede inteligentes em redes GSM/UMTS. Ele permite que os operadores de rede implementem serviços de valor agregado como:

- Cobrança pré-paga
- Triagem e bloqueio de chamadas
- Redes Privadas Virtuais (VPN)
- Serviços de tarifa premium
- Encaminhamento de chamadas com lógica personalizada
- Serviços baseados em localização

Opções de Configuração

```
# Configuração CAMEL (para respostas SendRoutingInfo)
# Chave de Serviço para iniciação de serviço CAMEL
camel_service_key: 11_110,

# Ponto de Detecção de Gatilho CAMEL
# Opções: :termAttemptAuthorized, :tBusy, :tNoAnswer, :tAnswer
camel_trigger_detection_point: :termAttemptAuthorized,
```

Chave de Serviço

A `camel_service_key` identifica qual serviço CAMEL deve ser invocado no gsmSCF (Função de Controle de Serviço). Este é um identificador numérico configurado em sua rede:

Chave de Serviço	Caso de Uso Típico
`11_110`	Controle de chamada pré-paga (padrão)
`100`	Serviço pré-pago de origem
`200`	Encaminhamento de chamadas com lógica personalizada
`300`	Serviço de Rede Privada Virtual (VPN)
Personalizado	Serviços específicos do operador

Exemplo de Configuração:

```
# Para controle de chamada pré-paga
camel_service_key: 11_110,

# Para serviço VPN
camel_service_key: 300,
```

Ponto de Detecção de Gatilho

O `camel_trigger_detection_point` especifica quando o serviço CAMEL deve ser acionado durante a configuração da chamada:

Ponto de Detecção	Descrição	Quando Acionado
`:termAttemptAuthorized`	Tentativa de chamada autorizada (padrão)	Antes da chamada ser roteada para o assinante
`:tBusy`	Ocupado na terminação	Quando o assinante está ocupado
`:tNoAnswer`	Sem resposta na terminação	Quando o assinante não atende
`:tAnswer`	Resposta na terminação	Quando o assinante atende a chamada

Exemplos de Configuração:

Controle padrão pré-pago (acionar antes do roteamento):

```
camel_trigger_detection_point: :termAttemptAuthorized,
```

Tratamento personalizado de ocupado (acionar quando ocupado):

```
camel_trigger_detection_point: :tBusy,
```

Cobrança baseada em resposta (acionar ao atender):

```
camel_trigger_detection_point: :tAnswer,
```

Resposta SRI com CAMEL

Quando configurado, as respostas SendRoutingInfo incluem informações de assinatura CAMEL:

```
GMSC → HLR: SendRoutingInfo (INÍCIO)
HLR → GMSC: Resposta SRI (FIM) com:
  - IMSI
  - Número VLR
  - Estado do assinante
  - Informações de roteamento CAMEL:
    * Chave de Serviço: 11_110
    * Endereço gsmSCF: <endereço configurado>
    * Ponto de Detecção de Gatilho: termAttemptAuthorized
    * Tratamento de Chamada Padrão: continueCall

GMSC contata gsmSCF no ponto de gatilho para executar o serviço
CAMEL
```

Melhores Práticas

- **Redes de Produção**: Use chaves de serviço padronizadas acordadas com seu provedor gsmSCF
- **Teste**: Use `:termAttemptAuthorized` para testes mais abrangentes
- **Serviços Pré-pagos**: A chave de serviço `11_110` é um padrão comum da indústria para chamadas pré-pagas de terminação
- **Tratamento de Fallback**: `defaultCallHandling: :continueCall` garante que as chamadas prossigam se o gsmSCF estiver inacessível

---

Tratamento de Assinantes em Roaming

Detecção de VLR Doméstico vs VLR em Roaming

Quando o HLR recebe uma requisição **SendRoutingInfo** (SRI), ele precisa determinar se o assinante está em um VLR "doméstico" (dentro de sua rede) ou em um VLR em roaming (visitando outra rede). O comportamento difere com base nessa determinação:

Para referência de parâmetros de configuração, consulte Prefixos VLR Domésticos na Referência de Configuração.

- **VLR Doméstico**: Retornar resposta SRI padrão com informações de roteamento CAMEL
- **VLR em Roaming**: Enviar uma solicitação Provide Roaming Number (PRN) para obter um MSRN, e então retorná-lo na resposta SRI

Configuração

```
# Prefixos VLR Domésticos
# Lista de prefixos de endereço VLR que são considerados "rede"
doméstica
# Se o endereço VLR do assinante começar com um desses prefixos,
use a resposta SRI padrão
# Caso contrário, o assinante está em roaming e precisamos enviar
PRN para obter MSRN
home_vlr_prefixes: ["555123"],
```

Exemplo de Configuração:

```
# Operador de rede doméstica única
home_vlr_prefixes: ["555123"],

# Múltiplas redes domésticas (por exemplo, diferentes regiões ou
subsidiárias)
home_vlr_prefixes: ["555123", "555124", "555125"],
```

Como Funciona

1. Fluxo de Assinante Doméstico (Padrão)

Quando o endereço VLR do assinante começa com um prefixo doméstico configurado:

```
GMSC → HLR: SendRoutingInfo (MSISDN: "1234567890")
HLR consulta a API de backend para dados do assinante
HLR verifica endereço VLR: "5551234567"
HLR determina: VLR começa com "555123" → Rede doméstica
HLR → GMSC: Resposta SRI com informações de roteamento CAMEL:
  - IMSI
  - Número VLR: "5551234567"
  - Endereço gsmSCF (MSC): "5551234501"
  - Chave de serviço CAMEL: 11_110
  - Ponto de detecção de gatilho: termAttemptAuthorized
```

2. Fluxo de Assinante em Roaming (PRN Necessário)

Quando o endereço VLR do assinante NÃO corresponde a nenhum prefixo doméstico:

```
GMSC → HLR: SendRoutingInfo (MSISDN: "1234567890")
HLR consulta a API de backend para dados do assinante
HLR verifica endereço VLR: "49170123456"
HLR determina: VLR não começa com "555123" → Roaming
HLR → MSC: ProvideRoamingNumber (PRN):
  - MSISDN: "1234567890"
  - IMSI: "999999876543210"
  - Número MSC: "49170123456"
  - Endereço GMSC: "5551234501"
MSC → HLR: Resposta PRN com MSRN: "49170999888777"
HLR → GMSC: Resposta SRI com informações de roteamento:
  - IMSI
  - Número VLR: "49170123456"
  - Número de Roaming (MSRN): "49170999888777"
```

Diferenças na Estrutura de Resposta

Resposta SRI de Assinante Doméstico

```
%{
  imsi: "999999876543210",
  extendedRoutingInfo: {
    :camelRoutingInfo, %{
      gmscCamelSubscriptionInfo: %{
        "t-CSI": %{
          serviceKey: 11_110,
          "gsmSCF-Address": "5551234501",
          defaultCallHandling: :continueCall,
          "t-BcsmTriggerDetectionPoint": :termAttemptAuthorized
        }
      }
    }
  },
  subscriberInfo: %{
    locationInformation: %{"vlr-number": "5551234567"},
    subscriberState: {:notProvidedFromVLR, :NULL}
  }
}
```

Resposta SRI de Assinante em Roaming

```
%{
  imsi: "999999876543210",
  extendedRoutingInfo: {
    :routingInfo, %{
      roamingNumber: "49170999888777"  # MSRN do PRN
    }
  },
  subscriberInfo: %{
    locationInformation: %{"vlr-number": "49170123456"},
    subscriberState: {:notProvidedFromVLR, :NULL}
  }
}
```

Operação Provide Roaming Number (PRN)

Estrutura da Solicitação PRN

A solicitação PRN enviada ao MSC/VLR contém:

Campo	Fonte	Descrição
MSISDN	Solicitação SRI	Número de telefone do assinante
IMSI	API HLR	IMSI do assinante
Número MSC	API HLR	MSC atendendo o assinante em roaming (`serving_msc`)
Endereço GMSC	Solicitação SRI	GMSC fazendo a solicitação SRI original
Número de Referência de Chamada	Estático	Identificador de referência de chamada
Fases CAMEL Suportadas	Estático	Fases CAMEL suportadas pelo GMSC

Tratamento da Resposta PRN

O HLR espera uma resposta PRN contendo:

- **MSRN** (Número de Roaming da Estação Móvel): Um número temporário alocado pela rede visitada para roteamento da chamada

Tratamento de Erros:

- Se o PRN expirar → Retorna erro 27 (Assinante Ausente) na resposta SRI
- Se o PRN falhar → Retorna erro 27 (Assinante Ausente) na resposta SRI
- Se o MSRN não puder ser extraído → Retorna erro 27 (Assinante Ausente) na resposta SRI

Exemplos de Configuração

Operador de Rede Doméstica Única

```
# Todos os endereços VLR começando com "555123" são considerados
domésticos
home_vlr_prefixes: ["555123"],
```

- VLR `5551234567` → Doméstico (resposta CAMEL)
- VLR `5551235001` → Doméstico (resposta CAMEL)
- VLR `49170123456` → Roaming (PRN + resposta MSRN)

Operador Multi-Região

```
# Múltiplas redes domésticas em diferentes regiões
home_vlr_prefixes: ["555123", "555124", "555125"],
```

- VLR `5551234567` → Doméstico (região 1)
- VLR `5552341234` → Doméstico (região 2)
- VLR `5553411111` → Doméstico (região 3)
- VLR `44201234567` → Roaming (internacional)

Configuração de Teste

Para testar a funcionalidade PRN, defina uma lista vazia para tratar todos os VLRs como roaming:

```
# Todos os VLRs são tratados como roaming (para testar o fluxo
PRN)
home_vlr_prefixes: [],
```

Melhores Práticas

- **Seleção de Prefixo**: Use o prefixo único mais curto que identifique os VLRs da sua rede (por exemplo, código do país + código da rede)
- **Múltiplos Prefixos**: Inclua todos os prefixos de VLR na sua rede, incluindo diferentes regiões e subsidiárias
- **Acordos de Roaming**: Certifique-se de que o PRN seja adequadamente suportado pelas redes parceiras de roaming

- **Teste**: Teste minuciosamente os cenários domésticos e de roaming antes da implantação em produção
- **Monitoramento**: Monitore as taxas de tempo limite do PRN para identificar problemas de conectividade com parceiros de roaming

Solução de Problemas

Sintoma: Todos os assinantes tratados como roaming

- **Causa**: `home_vlr_prefixes` não configurado ou prefixos não correspondem aos endereços VLR
- **Solução**: Verifique os endereços VLR em seu banco de dados e atualize os prefixos conforme necessário

Sintoma: Solicitações PRN expirando

- **Causa**: Problemas de conectividade de rede com o MSC/VLR parceiro de roaming
- **Solução**: Verifique o roteamento M3UA/SCCP para endereços MSC remotos

Sintoma: MSRN inválido na resposta SRI

- **Causa**: Formato da resposta PRN do parceiro de roaming não corresponde à estrutura esperada
- **Solução**: Revise os logs de resposta PRN e ajuste `extract_msrn_from_prn/1` se necessário

---

Operações HLR

Operações MAP Suportadas

- `updateLocation` (Opcode 2) - Registrar localização VLR
- `sendAuthenticationInfo` (Opcode 56) - Gerar vetores de autenticação
- `sendRoutingInfo` (Opcode 22) - Fornecer MSRN para chamadas com suporte CAMEL

- `sendRoutingInfoForSM` (Opcode 45) - Fornecer GT do MSC para SMS
- `cancelLocation` (Opcode 3) - Desregistrar do antigo VLR
- `insertSubscriberData` (Opcode 7) - Enviar perfil do assinante

Mapeamento de Campos de Resposta

Esta seção detalha de onde cada campo nas respostas HLR vem.

Resposta SendRoutingInfo (SRI)

Propósito: Fornece informações de roteamento para chamadas recebidas para um assinante.

O HLR fornece dois tipos diferentes de resposta com base em se o assinante está em um VLR doméstico ou em roaming:

Resposta de Assinante Doméstico (Roteamento CAMEL)

Usado quando o endereço VLR do assinante começa com um valor configurado em `home_vlr_prefixes`.

Estrutura da Resposta:

Campo	Fonte	Descrição
IMSI	API OmniHSS	IMSI do assinante do banco de dados OmniHSS
Número VLR	API OmniHSS	VLR atual atendendo o assinante (`circuit_session.assigned_vlr`)
Estado do Assinante	Estático	Sempre `notProvidedFromVLR`
extendedRoutingInfo	-	Tipo: `camelRoutingInfo`
Endereço gsmSCF	API OmniHSS	MSC atendendo o assinante (`circuit_session.assigned_msc`)
Chave de Serviço	runtime.exs	Identificador de serviço CAMEL (`camel_service_key`)
Ponto de Detecção de Gatilho	runtime.exs	Quando acionar CAMEL (`camel_trigger_detection_point`)
Tratamento de Capacidade CAMEL	Estático	Nível de suporte à fase CAMEL
Tratamento de Chamada Padrão	Estático	Fallback se gsmSCF inacessível

Resposta de Assinante em Roaming (Roteamento MSRN)

Usado quando o endereço VLR do assinante NÃO corresponde a nenhum valor configurado em `home_vlr_prefixes`.

Estrutura da Resposta:

Campo	Fonte	Descrição	
IMSI	API OmniHSS	IMSI do assinante do banco de dados OmniHSS	`"9`
Número VLR	API OmniHSS	VLR atual atendendo o assinante (`circuit_session.assigned_vlr`)	`"4`
Estado do Assinante	Estático	Sempre `notProvidedFromVLR`	`:n`
extendedRoutingInfo	-	Tipo: `routingInfo`	-
Número de Roaming (MSRN)	Resposta PRN	MSRN obtido da solicitação ProvideRoamingNumber	`"4`

Lógica de Decisão de Roteamento:

```
1. OmniSS7 recebe solicitação SendRoutingInfo
2. OmniSS7 consulta dados do assinante da API OmniHSS
3. OmniSS7 verifica endereço VLR contra home_vlr_prefixes:

   Se VLR começa com prefixo doméstico:
     → Retornar informações de roteamento CAMEL (fluxo de
assinante doméstico)

   Se VLR NÃO corresponder a nenhum prefixo doméstico:
     → Enviar ProvideRoamingNumber (PRN) para MSC
     → Extrair MSRN da resposta PRN
     → Retornar informações de roteamento com MSRN (fluxo de
assinante em roaming)
```

Fluxo de Dados:

- OmniSS7 consulta o OmniHSS para informações sobre o assinante
- OmniHSS retorna IMSI, localização atual VLR/MSC e estado do assinante
- OmniSS7 usa esses dados para construir a resposta MAP

Requisitos de Configuração:

```
# Em runtime.exs
home_vlr_prefixes: ["555123"],  # Lista de prefixos VLR domésticos
```

Respostas de Erro:

- Se `serving_vlr` e `serving_msc` forem `null`: Retorna erro 27 (Assinante Ausente)
- Se assinante não for encontrado: Retorna erro 1 (Assinante Desconhecido)
- Se a solicitação PRN expirar (caso de roaming): Retorna erro 27 (Assinante Ausente)
- Se a resposta PRN for inválida (caso de roaming): Retorna erro 27 (Assinante Ausente)

Resposta UpdateLocation com InsertSubscriberData

Propósito: Registra o assinante em um novo VLR e provisiona os dados do assinante.

Resposta UpdateLocation FIM

Campo	Fonte	Descrição	Exemp
Número HLR	runtime.exs	Título Global deste HLR (`hlr_service_center_gt_address`)	`"5551234`
Tipo de Mensagem TCAP	Estático	Resposta final após todos os ISDs	`FIM`

InsertSubscriberData #1 (Dados Básicos do Assinante)

Campo	Fonte	Descrição	Exemplo
IMSI	Solicitação	Da solicitação UpdateLocation	`"999999876543`
MSISDN	API OmniHSS	Número de telefone do assinante do OmniHSS	`"555123456"`
Categoria	Estático	Categoria do assinante	`"\n"` (0x0A)
Status do Assinante	Estático	Status do serviço	`:serviceGrant`
Lista de Serviços de Portadora	Estático	Serviços de portadora suportados	`[<<31>>]`
Lista de Teleserviços	Estático	Teleserviços suportados	`[<<17>>, "` `"\""]`
Modo de Acesso à Rede	runtime.exs	Acesso de pacote/circuito (`isd_network_access_mode`)	`:packetAndCir`

InsertSubscriberData #2 (Serviços Suplementares) - Opcional

Campo	Fonte	Descrição	Controlado Por
SS Provisionados	Estático	Dados de serviços suplementares	`isd_send_ss_data: true`
Encaminhamento de Chamadas	Estático	Status de configurações de encaminhamento (incondicional, ocupado, sem resposta, não acessível)	Config habilitada
Espera de Chamadas	Estático	Status do serviço de espera de chamadas	Config habilitada
Serviço de Múltiplos Participantes	Estático	Suporte a chamadas de conferência	Config habilitada

ISD #2 inclui:

- Encaminhamento de chamadas incondicional (código SS 21)
- Encaminhamento de chamadas em ocupado (código SS 41)
- Encaminhamento de chamadas sem resposta (código SS 42)
- Encaminhamento de chamadas não acessível (código SS 62)
- Espera de chamadas (código SS 43)
- Serviço de múltiplos participantes (código SS 51)
- Serviços CLIP/CLIR

InsertSubscriberData #3 (Bloqueio de Chamadas) - Opcional

Campo	Fonte	Descrição	Controlado Por
Informações de Bloqueio de Chamadas	Estático	Configurações de bloqueio de chamadas	`isd_send_call_barring: true`
BAOC	Estático	Bloqueio de Todas as Chamadas de Saída (código SS 146)	Config habilitada
BOIC	Estático	Bloqueio de Chamadas Internacionais de Saída (código SS 147)	Config habilitada
Dados de Restrição de Acesso	Estático	Restrições de acesso à rede	Config habilitada

Controle da Sequência ISD:

- ISD #1: **Sempre enviado** - Contém dados essenciais do assinante
- ISD #2: Enviado apenas se `isd_send_ss_data: true` em runtime.exs
- ISD #3: Enviado apenas se `isd_send_call_barring: true` em runtime.exs

Resposta SendRoutingInfoForSM (SRI-for-SM)

Propósito: Fornece informações de roteamento MSC/SMSC para entrega de SMS. Quando um SMSc precisa entregar um SMS a um assinante, ele envia uma solicitação SRI-for-SM ao HLR para determinar onde roteá-lo.

Estrutura da Resposta:

Campo	Fonte	Descrição	Como Gerado
IMSI	Calculado	IMSI sintético derivado do MSISDN	`PLMN_PREFIX + zero_padded_MSISDN`
Número do Nó da Rede	runtime.exs	Endereço GT do SMSC para roteamento de SMS	`smsc_service_center_gt_address`

Parâmetros de Configuração (do `runtime.exs`):

```
# Endereço do Centro de Serviço GT (retornado nas respostas SRI-
for-SM)
# Isso informa ao SMSc solicitante onde enviar mensagens MT-
ForwardSM
smsc_service_center_gt_address: "5551234567",  # Obrigatório

# Configuração de Mapeamento MSISDN ↔ IMSI
# Prefixo PLMN: MCC (001 = Rede de Teste) + MNC (01 = Operador de
Teste)
hlr_imsi_plmn_prefix: "001001",                # Único parâmetro
de configuração necessário!
```

Mapeamento MSISDN ↔ IMSI

Parâmetros de Configuração:

Esses parâmetros controlam como o OmniSS7 gera IMSIs sintéticos a partir de MSISDNs para respostas SRI-for-SM:

- **`hlr_imsi_plmn_prefix`**: O prefixo MCC+MNC a ser usado ao construir IMSIs sintéticos (por exemplo, "50557" para MCC=505, MNC=57)

- `hlr_msisdn_country_code`: Código do país a ser precedido ao fazer o mapeamento reverso IMSI→MSISDN (por exemplo, "61" para Austrália, "1" para EUA/Canadá)
- `hlr_msisdn_nsn_offset`: Posição do caractere onde o Número Nacional do Assinante (NSN) começa dentro do MSISDN (normalmente 0 se o MSISDN não incluir o código do país, ou comprimento do código do país se incluir)
- `hlr_msisdn_nsn_length`: Número de dígitos a serem extraídos do MSISDN como o NSN

Para detalhes adicionais de configuração, consulte Mapeamento MSISDN ↔ IMSI na Referência de Configuração.

Por que o Mapeamento MSISDN para IMSI é Necessário?

O protocolo MAP para **SendRoutingInfoForSM** (SRI-for-SM) exige que o HLR retorne um **IMSI** (Identidade Internacional de Assinante Móvel) em sua resposta. No entanto, o SMSc solicitante só conhece o **MSISDN** do assinante (número de telefone).

Em uma rede tradicional:

- O SMSc envia SRI-for-SM com o MSISDN de destino (por exemplo, "5551234567")
- O HLR deve procurar o assinante em seu banco de dados para encontrar seu IMSI
- O HLR retorna o IMSI na resposta SRI-for-SM
- O SMSc então usa esse IMSI ao enviar MT-ForwardSM para o MSC/VLR

Abordagem do OmniSS7 - IMSIs Sintéticos:

Em vez de manter um banco de dados completo de assinantes com mapeamentos MSISDN-para-IMSI, o OmniSS7 usa um esquema de codificação simples para **calcular** IMSIs sintéticos diretamente a partir do MSISDN. Essa abordagem fornece dois benefícios principais:

1. **Privacidade**: Os reais IMSIs de assinantes armazenados no banco de dados HLR nunca são expostos nas respostas SRI-for-SM enviadas pela rede SS7

2. **Simplicidade**: Não há necessidade de consultar o banco de dados HLR para buscas de IMSI durante operações SRI-for-SM - o IMSI é calculado em tempo real a partir do MSISDN

Como Funciona:

Os MSISDNs são codificados diretamente na parte do assinante do IMSI (os dígitos após MCC+MNC):

```
IMSI = PLMN_PREFIX + zero_padded_MSISDN
```

Onde:

- **PLMN_PREFIX**: MCC + MNC (por exemplo, "001001" para Rede de Teste)
- **MSISDN**: Todos os dígitos numéricos do número de telefone
- **Zero Padding**: Preenchido à esquerda com zeros para preencher o IMSI exatamente em 15 dígitos

Exemplo Passo a Passo:

```
# Configuração
plmn_prefix = "001001"  # MCC 001 + MNC 01

# Entrada: MSISDN da solicitação SRI-for-SM (decodificado TBCD)
msisdn = "555123456"    # 9 dígitos

# Passo 1: Calcular espaço disponível para o número do assinante
subscriber_digits = 15 - String.length("001001")  # = 9 dígitos

# Passo 2: Preencher MSISDN à esquerda com zeros para preencher a
parte do assinante
padded_msisdn = String.pad_leading("555123456", 9, "0")  # =
"555123456" (sem preenchimento necessário)

# Passo 3: Concatenar prefixo PLMN + MSISDN preenchido
imsi = "001001" <> "555123456"  # = "001001555123456" (exatamente
15 dígitos)
```

Exemplos Completos:

MSISDN de Entrada	Prefixo PLMN	Dígitos do Assinante Disponíveis	MSISDN Preenchido	IMSI Fina
`"555123456"`	`"001001"` (6)	9	`"555123456"`	`"001001555123`
`"99"`	`"001001"` (6)	9	`"000000099"`	`"001001000000`
`"999999999"`	`"001001"` (6)	9	`"999999999"`	`"001001999999`
`"91123456789"`	`"001001"` (6)	9	`"555123456"`	`"001001555123`

Tratamento de Casos Limite:

- **MSISDNs Curtos**: Preenchidos à esquerda com zeros (por exemplo, `"99"` → `"000000099"`)
- **MSISDNs Longos**: Mantidos os dígitos mais à direita, os dígitos mais à esquerda são truncados (por exemplo, `"91123456789"` → `"555123456"`)
- **Comprimento do IMSI**: Sempre exatamente 15 dígitos

Tratamento Reverso (IMSI → MSISDN):

O SMSc pode reverter esse mapeamento para converter IMSIs de volta para MSISDNs:

```
# Entrada: IMSI da resposta SRI-for-SM
imsi = "001001555123456"

# Passo 1: Remover prefixo PLMN
plmn_prefix = "001001"
subscriber_portion = String.slice(imsi, 6, 9)  # = "555123456"

# Passo 2: Remover zeros à esquerda para obter o MSISDN real
msisdn = String.replace_leading(subscriber_portion, "0", "")  # =
"555123456"
```

Exemplos de Mapeamento Reverso:

IMSI de Entrada	Prefixo PLMN	Porção do Assinante	Remover Zeros à Esquerda	MSISDI Final
`"001001555123456"`	`"001001"`	`"555123456"`	`"555123456"`	`"5551234`
`"001001000000099"`	`"001001"`	`"000000099"`	`"99"`	`"99"`
`"001001999999999"`	`"001001"`	`"999999999"`	`"999999999"`	`"9999999`

Propriedades deste Mapeamento:

- **Determinístico**: O mesmo MSISDN sempre produz o mesmo IMSI
- **Reversível**: Pode converter de IMSI para MSISDN
- **Configuração Mínima**: Apenas requer `hlr_imsi_plmn_prefix`
- **Privacidade Preservada**: Reais IMSIs nunca expostos
- **Sem Consulta ao Banco de Dados**: Cálculo rápido, sem chamadas à API necessárias
- **Sempre 15 Dígitos**: IMSI é sempre exatamente 15 dígitos

Tratamento de Entrada MSISDN:

Quando o HLR recebe uma solicitação SRI-for-SM, o MSISDN passa por decodificação TBCD:

1. **Decodificação TBCD**: Converter TBCD binário para string (pode incluir prefixo TON/NPI como "91")
2. **Extrair Dígitos**: Manter apenas dígitos numéricos, remover quaisquer caracteres não numéricos
3. **Normalizar**: Se mais longo que o espaço disponível, pegar os dígitos mais à direita; se mais curto, preencher à esquerda com zeros
4. **Codificar**: Concatenar prefixo PLMN + MSISDN normalizado

Considerações de Segurança:

Os IMSIs sintéticos retornados nas respostas SRI-for-SM são puramente para fins de roteamento. Eles NÃO são os reais IMSIs armazenados no banco de dados de assinantes HLR. Isso fornece uma camada adicional de proteção à privacidade, pois os reais IMSIs de assinantes são expostos apenas quando absolutamente necessário (por exemplo, durante operações UpdateLocation ou SendAuthenticationInfo que requerem vetores de autenticação reais).

Fluxo de Resposta:

```
1. SMSc → HLR: Solicitação SRI-for-SM
   - MSISDN (TBCD): "91123456789" (inclui TON/NPI)

2. Processamento do HLR:
   - Decodificação TBCD: "91123456789"
   - Extrair dígitos: "91123456789" (11 dígitos)
   - Ajustar para 9 dígitos: "555123456" (9 mais à direita)
   - Adicionar PLMN: "001001" + "555123456" = "001001555123456"
   - Obter GT do SMSC da configuração: "5551234567"

3. HLR → SMSc: Resposta SRI-for-SM
   - IMSI: "001001555123456" (sintético, sempre 15 dígitos)
   - Número do Nó da Rede: "5551234567" (onde enviar MT-ForwardSM)

4. SMSc envia MT-ForwardSM para "5551234567" com IMSI
"001001555123456"
```

Configuração:

Os seguintes parâmetros são usados em `runtime.exs`:

```
# Prefixo PLMN: MCC (001 = Rede de Teste) + MNC (01 = Operador de
Teste)
hlr_imsi_plmn_prefix: "001001",

# Extração de NSN (se MSISDNs incluírem código do país)
hlr_msisdn_country_code: "1",      # Usado para mapeamento reverso
(IMSI→MSISDN)
hlr_msisdn_nsn_offset: 1,          # Ignorar 1 dígito do código do
país
hlr_msisdn_nsn_length: 10          # Extrair 10 dígitos de NSN
```

Configuração de Extração de NSN:

Se seus MSISDNs incluírem o código do país (por exemplo, `"68988000088"` em vez de apenas `"88000088"`), você deve configurar a extração de NSN:

- **`hlr_msisdn_nsn_offset`**: Posição onde o NSN começa (normalmente o comprimento do seu código do país)
- **`hlr_msisdn_nsn_length`**: Número de dígitos no NSN

Exemplos:

Exemplo	Código do País	MSISDN Exemplo	nsn_offset	nsn_length	N Extr
CC de 1 dígito	`"9"`	`"95551234567"`	`1`	`10`	`"5551`
CC de 2 dígitos	`"99"`	`"99412345678"`	`2`	`9`	`"4123`
CC de 3 dígitos	`"999"`	`"99988000088"`	`3`	`8`	`"8800`

Como Funciona:

1. **MSISDN → IMSI**: Extrair NSN do MSISDN, preencher com zeros à esquerda, concatenar com prefixo PLMN

```
MSISDN: "99988000088"
NSN: String.slice("99988000088", 3, 8) = "88000088"
MSISDN Preenchido: "088000088" (9 dígitos)
IMSI: "547050" + "088000088" = "547050088000088"
```

2. **IMSI → MSISDN**: Remover prefixo PLMN, remover zeros à esquerda, preceder código do país

```
IMSI: "547050088000088"
Porção do Assinante: "088000088"
Remover zeros: "88000088"
MSISDN: "+999" + "88000088" = "+99988000088"
```

Requisitos da API: Nenhum - SRI-for-SM usa valores calculados e configuração apenas. Nenhuma

chamada à API de backend é necessária.

Resumo da Fonte dos Campos

Tipo de Fonte	Descrição	Exemplos
API OmniHSS	Dados dinâmicos do banco de dados de assinantes OmniHSS	IMSI, MSISDN, VLR/MSC atendendo a partir de circuit_session
runtime.exs	Parâmetros de configuração do OmniSS7	`smsc_service_center_gt_address`, `camel_service_key`, `isd_network_access_mode`
Estático	Valores codificados na geração de resposta	Status do assinante, serviços de portadora, códigos SS
Solicitação	Campos extraídos da requisição MAP de entrada	IMSI da UpdateLocation, MSISDN da SRI
Calculado	Valores derivados usando lógica	IMSI sintético em SRI-for-SM (hlr_imsi_prefix + NSN)

Dependências de Configuração

Obrigatório em runtime.exs:

- `hlr_service_center_gt_address` - Usado nas respostas UpdateLocation
- `smsc_service_center_gt_address` - Usado nas respostas SRI-for-SM (onde enviar MT-ForwardSM)

Opcional em runtime.exs (com padrões):

- `camel_service_key` - Padrão: `11_110`
- `camel_trigger_detection_point` - Padrão: `:termAttemptAuthorized`
- `isd_network_access_mode` - Padrão: `:packetAndCircuit`
- `isd_send_ss_data` - Padrão: `true`
- `isd_send_call_barring` - Padrão: `true`
- `hlr_imsi_plmn_prefix` - Padrão: `"001001"` (prefixo PLMN para mapeamento MSISDN↔IMSI)

Obrigatório do OmniHSS:

O OmniHSS deve fornecer endpoints da API REST para:

- Consulta de assinante por IMSI e MSISDN
- Atualizações de localização de sessão de circuito (atribuição VLR/MSC)
- Geração de vetores de autenticação
- Consultas de status de assinante e perfil de serviço

Documentação Relacionada

Documentação do OmniSS7:

- ← Voltar à Documentação Principal
- Guia de Recursos Comuns
- Guia do Cliente MAP
- Referência Técnica
- Referência de Configuração

Documentação do OmniHSS: Para gerenciamento de assinantes, provisionamento, configuração de autenticação e operações administrativas,

consulte a **documentação do produto OmniHSS**. O OmniHSS contém toda a lógica do banco de dados de assinantes, algoritmos de autenticação, regras de provisionamento de serviços e capacidades de gerenciamento Multi-IMSI.

OmniSS7 por Omnitouch Network Services

Guia de Configuração do Cliente MAP

← Voltar à Documentação Principal

Este guia fornece configuração detalhada para usar o OmniSS7 como um **Cliente MAP** para enviar solicitações do protocolo MAP para elementos de rede.

Índice

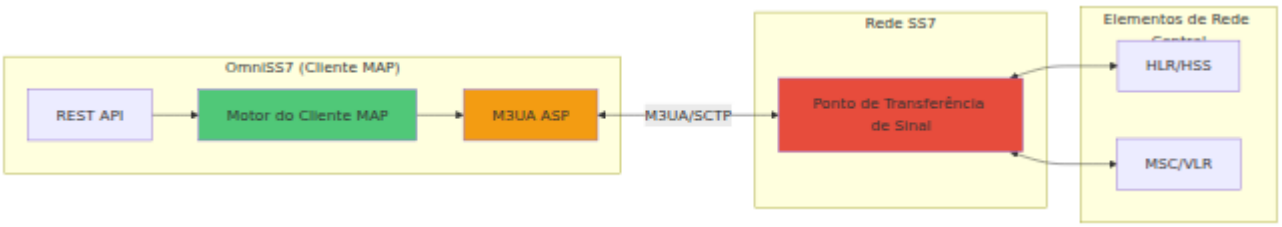
6. Solução de Problemas

O que é o Modo Cliente MAP?

O **Modo Cliente MAP** permite que o OmniSS7 se conecte como um **Processo de Servidor de Aplicação (ASP)** a um par M3UA (STP ou SGP) e envie/receba mensagens **MAP (Mobile Application Part)** para serviços como:

- **Consultas HLR**: SRI (Send Routing Info), SRI-for-SM, Authentication Info
- **Atualizações de Localização**: Update Location, Cancel Location
- **Gerenciamento de Assinantes**: Provide Roaming Number (PRN), Insert Subscriber Data

Arquitetura da Rede



Habilitando o Modo Cliente MAP

Edite `config/runtime.exs` e configure as configurações do cliente MAP. Para referência completa de configuração, consulte Parâmetros de Conexão M3UA na Referência de Configuração.

Configuração Básica

```
config :omniss7,
  # Habilitar modo Cliente MAP
  map_client_enabled: true,

  # Conexão M3UA para Cliente MAP (conecta como ASP ao STP/SGP
remoto)
  map_client_m3ua: %{
    mode: "ASP",                      # Modo M3UA: "ASP" (cliente)
ou "SGP" (servidor)
    callback: {MapClient, :handle_payload, []},  # Callback para
mensagens recebidas
    process_name: :map_client_asp,    # Nome do processo
registrado
    local_ip: {10, 0, 0, 100},       # Endereço IP local
    local_port: 2905,                 # Porta SCTP local
    remote_ip: {10, 0, 0, 1},        # IP STP/SGP remoto
    remote_port: 2905,                # Porta STP/SGP remota
    routing_context: 1                # Contexto de roteamento
M3UA
  }
```

Exemplo de Configuração para Produção

```
config :omniss7,
  # Habilitar Cliente MAP para produção
  map_client_enabled: true,

  # Conexão M3UA de produção
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :map_client_asp,
    local_ip: {10, 0, 0, 100},
    local_port: 2905,
    remote_ip: {10, 0, 0, 1},        # IP STP de produção
    remote_port: 2905,
    routing_context: 1
  }

config :control_panel,
  web: %{
    listen_ip: "0.0.0.0",
    port: 443,
    hostname: "ss7-gateway.example.com",
    enable_tls: true,
    tls_cert: "/etc/ssl/certs/gateway.crt",
    tls_key: "/etc/ssl/private/gateway.key"
  }
```

Operações MAP Disponíveis

1. Enviar Informações de Roteamento para SM (SRI-for-SM)

Consulta o HLR para determinar o MSC de atendimento para entrega de SMS. Para informações detalhadas sobre como o HLR processa solicitações SRI-for-SM, consulte SRI-for-SM no Guia HLR.

Endpoint da API: `POST /api/sri-for-sm`

Solicitação:

```
{
  "msisdn": "447712345678",
  "serviceCenter": "447999123456"
}
```

Resposta:

```
{
  "result": {
    "imsi": "234509876543210",
    "locationInfoWithLMSI": {
      "networkNode-Number": "447999555111"
    }
  }
}
```

Exemplo de cURL:

```
curl -X POST http://localhost/api/sri-for-sm \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "447712345678",
    "serviceCenter": "447999123456"
  }'
```

---

2. Enviar Informações de Roteamento (SRI)

Consulta o HLR para informações de roteamento de chamadas de voz.

Endpoint da API: `POST /api/sri`

Solicitação:

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456"
}
```

Resposta:

```
{
  "result": {
    "imsi": "234509876543210",
    "extendedRoutingInfo": {
      "routingInfo": {
        "roamingNumber": "447999555222"
      }
    }
  }
}
```

---

3. Fornecer Número de Roaming (PRN)

Solicita um número de roaming temporário (MSRN) do MSC de atendimento.

Endpoint da API: `POST /api/prn`

Solicitação:

```
{
  "msisdn": "447712345678",
  "gmsc": "447999123456",
  "msc_number": "447999555111",
  "imsi": "234509876543210"
}
```

---

4. Enviar Informações de Autenticação

Solicita vetores de autenticação do HLR para autenticação de assinantes.

Endpoint da API: `POST /api/send-auth-info`

Solicitação:

```
{
  "imsi": "234509876543210",
  "vectors": 5
}
```

Resposta:

```
{
  "result": {
    "authenticationSetList": [
      {
        "rand": "0123456789ABCDEF0123456789ABCDEF",
        "xres": "ABCDEF0123456789",
        "ck": "0123456789ABCDEF0123456789ABCDEF",
        "ik": "FEDCBA9876543210FEDCBA9876543210",
        "autn": "0123456789ABCDEF0123456789ABCDEF"
      }
    ]
  }
}
```

---

5. Atualizar Localização

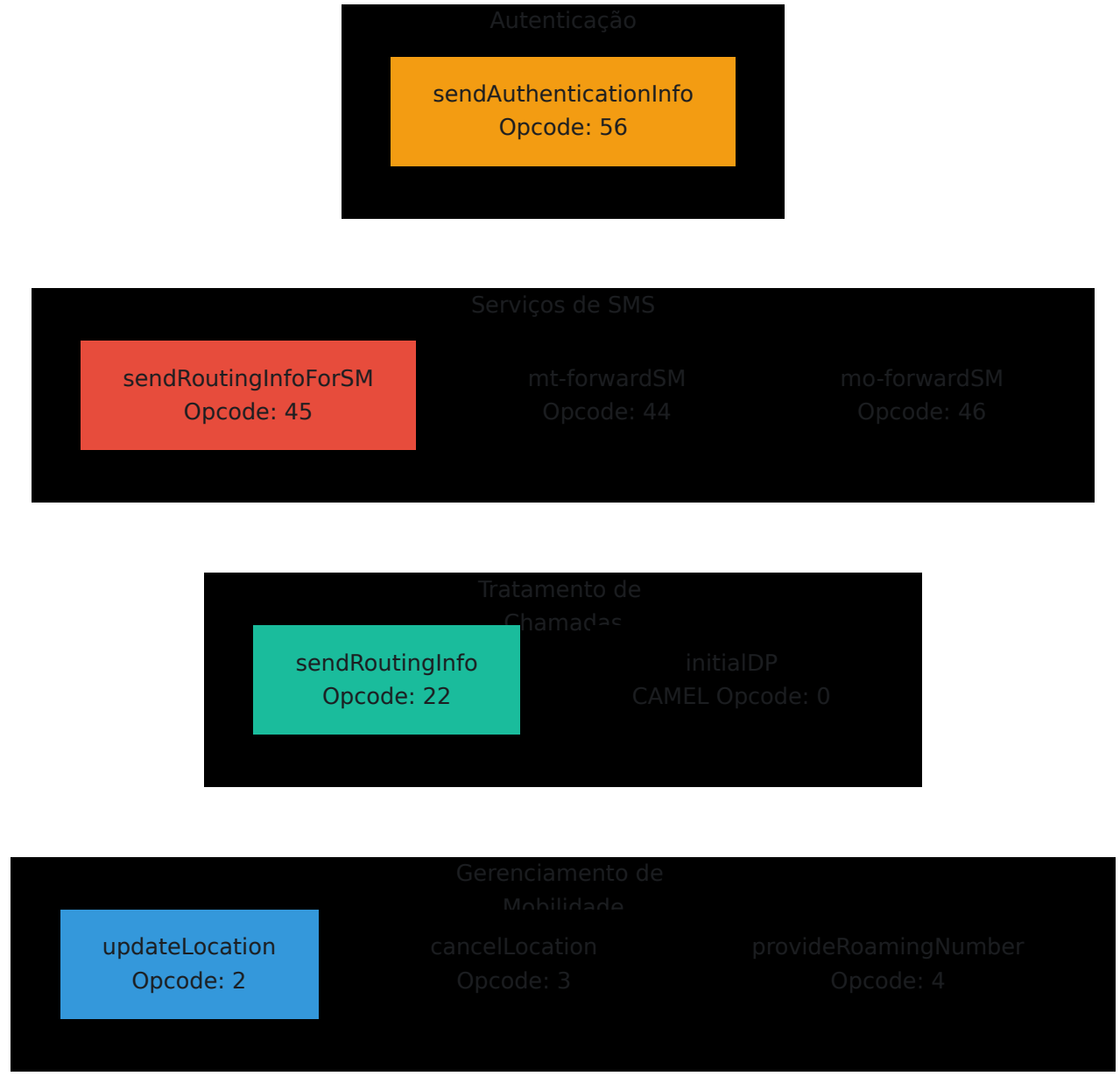
Registra a localização atual de um assinante no HLR. Para informações detalhadas sobre o processamento de UpdateLocation e sequências de InsertSubscriberData, consulte Atualizações de Localização no Guia HLR.

Endpoint da API: `POST /api/updateLocation`

Solicitação:

```
{
  "imsi": "234509876543210",
  "vlr": "447999555111"
}
```

Resumo das Operações MAP



Enviando Solicitações via API

Usando Swagger UI

A Swagger UI fornece uma interface interativa para enviar solicitações SS7.

Acessar Swagger UI:

1. Navegue até `http://your-server/swagger`
2. Navegue pelos endpoints da API disponíveis
3. Clique em qualquer endpoint para expandir seus detalhes

Enviando uma Solicitação:

1. Clique no endpoint que deseja usar (por exemplo, `/api/sri-for-sm`)
2. Clique no botão "Try it out"
3. Preencha os parâmetros necessários no corpo da solicitação
4. Clique em "Execute"
5. Veja a resposta abaixo

Códigos de Resposta da API

- **200** - Sucesso, resultado retornado no corpo da resposta
- **400** - Solicitação Inválida, parâmetros inválidos
- **504** - Tempo Limite do Gateway, sem resposta da rede SS7 dentro de 10 segundos

---

Métricas do Cliente MAP

Métricas Disponíveis

Métricas de Solicitação:

- `map_requests_total` - Número total de solicitações MAP enviadas
  - Rótulos: `operation` (valores: `sri`, `sri_for_sm`, `prn`, `authentication_info`, etc.)
- `map_request_errors_total` - Número total de erros de solicitação MAP
  - Rótulos: `operation`
- `map_request_duration_milliseconds` - Histograma das durações das solicitações MAP

    - Rótulos: `operation`

- `map_pending_requests` - Número atual de solicitações MAP pendentes (gauge)

Exemplos de Consultas Prometheus

```
# Total de solicitações SRI-for-SM na última hora
increase(map_requests_total{operation="sri_for_sm"}[1h])

# Tempo médio de resposta para solicitações SRI
rate(map_request_duration_milliseconds_sum{operation="sri"}[5m]) /
rate(map_request_duration_milliseconds_count{operation="sri"}[5m])

# Taxa de erro para todas as operações MAP
sum(rate(map_request_errors_total[5m])) by (operation)

# Solicitações pendentes atuais
map_pending_requests
```

Solução de Problemas do Cliente MAP

Problema: Tempo Limite das Solicitações

Sintomas:

- A API retorna 504 Gateway Timeout
- Sem resposta do HLR/MSC

Verificações:

1. Verifique se a conexão M3UA está ATIVA:

```
# No console IEx
:sys.get_state(:map_client_asp)
```

2. Verifique a conectividade de rede com o STP
3. Verifique o contexto de roteamento e endereçamento SCCP
4. Verifique os logs em busca de erros SCCP

Problema: Erros SCCP

Sintomas:

- A API retorna respostas de erro SCCP
- Os logs mostram mensagens de "serviço unitdata SCCP"

Códigos de Erro SCCP Comuns:

- **Sem Tradução**: Título Global não encontrado na tabela de roteamento do STP
- **Falha de Subsistema**: Subsistema de destino (HLR SSN 6) está indisponível
- **Falha de Rede**: Congestionamento ou falha na rede

Soluções:

- Contate o administrador do STP para verificar a configuração de roteamento
- Verifique se o Título Global de destino é acessível
- Verifique se o subsistema de destino está operacional

Documentação Relacionada

- ← Voltar à Documentação Principal
- Guia de Recursos Comuns - Interface Web, API, Monitoramento

- Guia STP - Configuração de roteamento
- Guia do Centro de SMS - Entrega de SMS
- Referência Técnica - Especificações do protocolo

OmniSS7 por Omnitouch Network Services

Guia de Configuração do Centro de SMS (SMSc)

← Voltar à Documentação Principal

Este guia fornece a configuração detalhada para usar o OmniSS7 como um **Centro de SMS (SMSc)** na frente com o **OmniMessage** como a plataforma de armazenamento e entrega de mensagens no backend.

Integração com OmniMessage

O modo SMSc do OmniSS7 funciona como uma frente de sinalização SS7 que se conecta ao **OmniMessage**, uma plataforma de SMS de nível de operadora. Esta arquitetura separa as preocupações:

- **OmniSS7 (Frontend SMSc)**: Lida com toda a sinalização do protocolo SS7/MAP, roteamento SCCP e comunicação de rede
- **OmniMessage (Backend SMS)**: Gerencia armazenamento de mensagens, enfileiramento, lógica de tentativas, rastreamento de entrega e decisões de roteamento

Por que OmniMessage?

O OmniMessage fornece capacidades de mensagens SMS de nível de operadora com recursos incluindo:

- **Gerenciamento de Fila de Mensagens**: Armazenamento persistente com lógica de tentativas configurável e enfileiramento por prioridade
- **Rastreamento de Entrega**: Status de entrega em tempo real, relatórios de entrega (DLR) e rastreamento de motivos de falha
- **Suporte a Múltiplos SMSc**: Várias instâncias de frontend podem se conectar a um único backend OmniMessage para balanceamento de carga

e redundância

- **Inteligência de Roteamento**: Regras de roteamento avançadas com base no destino, remetente, conteúdo da mensagem e hora do dia
- **Limitação de Taxa**: Controles TPS (transações por segundo) por rota para evitar congestionamento na rede
- **Design API-First**: API HTTP RESTful para integração com sistemas de cobrança, portais de clientes e aplicativos de terceiros
- **Análise e Relatórios**: Estatísticas de volume de mensagens, taxas de sucesso de entrega e métricas de desempenho

Todos os dados de mensagens, estado de entrega e configurações de roteamento são armazenados e gerenciados no OmniMessage. O OmniSS7 consulta o OmniMessage por meio de chamadas de API HTTPS para recuperar mensagens pendentes, atualizar status de entrega e registrar-se como um frontend ativo.

Importante: O modo SMSc do OmniSS7 é um **frontend de sinalização apenas**. Toda a lógica de roteamento de mensagens, gerenciamento de filas, algoritmos de tentativas, rastreamento de entrega e regras de negócios são tratados pelo OmniMessage. Este guia cobre a configuração do protocolo SS7/MAP no OmniSS7. Para informações sobre roteamento de mensagens, configuração de filas, relatórios de entrega, limitação de taxa e análises, **consulte a documentação do OmniMessage**.

Índice

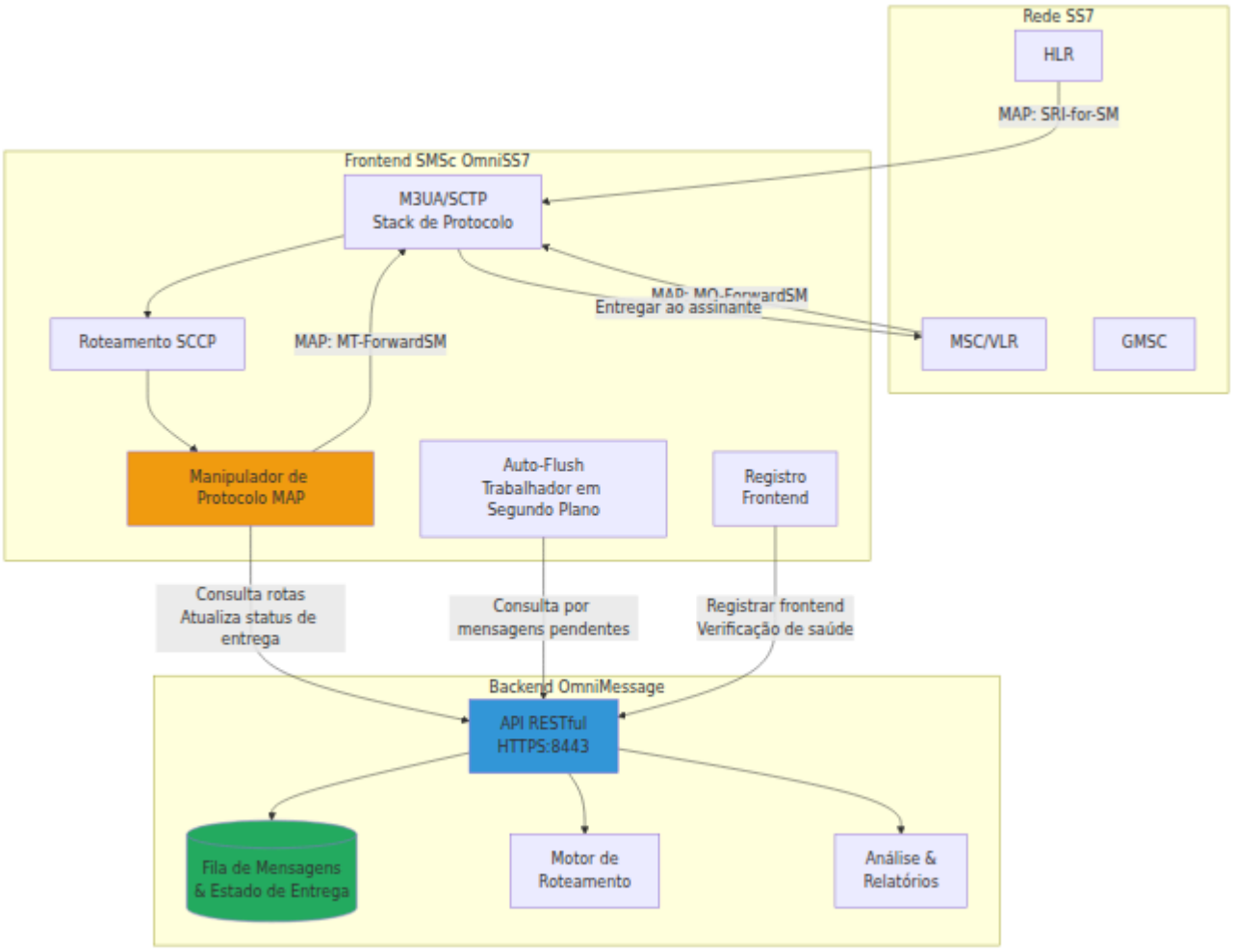
O que é o Modo Centro de SMS?

Nota: Esta seção cobre apenas a configuração de sinalização SS7 do OmniSS7. Para regras de roteamento de mensagens, gerenciamento de filas, rastreamento de entrega e configuração de lógica de negócios, consulte a **documentação do produto OmniMessage**.

Modo Centro de SMS permite que o OmniSS7 funcione como um SMSc para:

- **Entrega MT-SMS**: Entrega de SMS Terminados em Móvel para assinantes
- **Tratamento MO-SMS**: Recepção e roteamento de SMS Originados em Móvel
- **Enfileiramento de Mensagens**: Fila de mensagens com suporte a banco de dados e lógica de tentativas
- **Auto-Flush**: Entrega automática de SMS da fila
- **Relatórios de Entrega**: Rastrear status de entrega de mensagens

Arquitetura do Centro de SMS



Habilitando o Modo SMSc

O OmniSS7 pode operar em diferentes modos. Para usá-lo como um SMSc, você precisa habilitar o modo SMSc na configuração.

Mudando para o Modo SMSc

O `config/runtime.exs` do OmniSS7 contém três modos operacionais pré-configurados. Para habilitar o modo SMSc:

1. **Abra** `config/runtime.exs`
2. **Encontre** as três seções de configuração (linhas 53-204):
   - Configuração 1: Modo STP (linhas 53-95)

    - Configuração 2: Modo HLR (linhas 97-142)
    - Configuração 3: Modo SMSc (linhas 144-204)
3. **Comente** qualquer outra configuração ativa (adicione `#` a cada linha)
4. **Descomente** a configuração SMSc (remova `#` das linhas 144-204)
5. **Personalize** os parâmetros de configuração conforme necessário
6. **Reinicie** a aplicação: `iex -S mix`

Configuração do Modo SMSc

A configuração completa do SMSc se parece com isso:

```
config :omniss7,
  # Flags de modo - Habilitar recursos STP + SMSc
  # Nota: map_client_enabled é verdadeiro porque o SMSc precisa de
capacidades de roteamento
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: true,

  # Configuração da API do Backend OmniMessage
  smsc_api_base_url: "https://10.179.3.219:8443",
  # Identificaç��o do SMSc para registro com o backend
  smsc_name: "ipsmgw",
  # Endereço GT do Centro de Serviço para operações de SMS
  smsc_service_center_gt_address: "5551234567",

  # Configura��ão de Auto Flush (processamento de fila de SMS em
segundo plano)
  auto_flush_enabled: true,
  auto_flush_interval: 10_000,
  auto_flush_dest_smsc: "ipsmgw",
  auto_flush_tps: 10,

  # Configuração de Conexão M3UA
  # Conectar como ASP para enviar/receber operações MAP SMS
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    # Endpoint local (sistema SMSc)
    local_ip: {10, 179, 4, 12},
    local_port: 2905,
    # Endpoint STP remoto
    remote_ip: {10, 179, 4, 10},
    remote_port: 2905,
    routing_context: 1
  }

config :control_panel,
  use_additional_pages: [
    {SS7.Web.EventsLive, "/events", "Eventos SS7"},
    {SS7.Web.TestClientLive, "/client", "Cliente SS7"},
    {SS7.Web.M3UAStatusLive, "/m3ua", "M3UA"},
    {SS7.Web.RoutingLive, "/routing", "Roteamento"},
```

```
    {SS7.Web.RoutingTestLive, "/routing_test", "Teste de
Roteamento"},
    {SS7.Web.SmscLinksLive, "/smsc_links", "Links SMSc"}
  ],
  page_order: ["/events", "/client", "/m3ua", "/routing",
"/routing_test", "/smsc_links", "/application", "/configuration"]
```

Parâmetros de Configuração a Personalizar

Para uma referência completa de todos os parâmetros de configuração, consulte a Referência de Configuraç��o.

Parâmetro	Tipo	Padrão	D
`smsc_api_base_url`	String	*Obrigatório*	**Endpo** backer OmniM
`smsc_name`	String	`" {hostname}_SMSc"`	Seu id SMSc p
`smsc_service_center_gt_address`	String	*Obrigatório*	Título Centro
`auto_flush_enabled`	Booleano	`true`	Habilit proces autom
`auto_flush_interval`	Inteiro	`10_000`	Interva proces fila em
`auto_flush_dest_smsc`	String	*Obrigatório*	Nome destin flush
`auto_flush_tps`	Inteiro	`10`	Taxa d proces mensa (transa
`local_ip`	Tupla	*Obrigatório*	Endere sistem
`local_port`	Inteiro	`2905`	Porta S
`remote_ip`	Tupla	*Obrigatório*	Endere para c SS7

Parâmetro	Tipo	Padrão	D
`remote_port`	Inteiro	`2905`	Porta S
`routing_context`	Inteiro	`1`	ID do c roteam

O que Acontece Quando o Modo SMSc é Habilitado

Quando `smsc_mode_enabled: true` e `map_client_enabled: true`, a interface da web mostrará:

- **Eventos SS7** - Registro de eventos
- **Cliente SS7** - Teste de operações MAP
- **M3UA** - Status da conexão
- **Roteamento** - Gerenciamento da tabela de rotas (STP habilitado)
- **Teste de Roteamento** - Teste de rotas (STP habilitado)
- **Links SMSc** - Status da API SMSc + gerenciamento de fila de SMS ← *específico do SMSc*
- **Recursos** - Monitoramento do sistema
- **Configuração** - Visualizador de configuração

A aba **Links HLR** será ocultada.

Notas Importantes

- O modo SMSc requer `map_client_enabled: true` para capacidades de roteamento
- **Backend OmniMessage**: A API do OmniMessage deve ser acessível no `smsc_api_base_url` configurado
- **Registro do Frontend**: O sistema registra automaticamente no OmniMessage a cada **5 minutos** através do módulo `SMS.FrontendRegistry`

- **Timeout de Requisição da API**: Todas as requisições da API OmniMessage têm um **timeout fixo de 5 segundos**
- **Timeout de Requisição MAP**: SRI-for-SM tem um timeout de diálogo de 10 segundos. MT-ForwardSM tem um **timeout de diálogo de 30 segundos** para acomodar respostas lentas do VLR em redes 2G/CS
- O auto-flush processa automaticamente a fila de SMS em segundo plano
- A conexão M3UA com o STP é necessária para enviar/receber operações MAP SMS
- Após mudar de modos, você deve reiniciar a aplicação para que as mudanças tenham efeito
- **Interface Web**: Consulte o Guia da Interface Web para informações sobre o uso da interface web
- **Acesso à API**: Consulte o Guia da API para documentação da API REST e acesso à Swagger UI

Configuração da API HTTP

Configuração do Backend OmniMessage

O OmniSS7 se comunica com o OmniMessage via API REST HTTPS para gerenciar a entrega de mensagens, rastrear o estado do assinante e registrar-se como um frontend ativo:

```
config :omniss7,
  # URL base da API OmniMessage
  smsc_api_base_url: "https://10.5.198.200:8443",
  # Identificador do nome do SMSc para registro (padrão para
hostname_SMSc se vazio)
  smsc_name: "omni-smsc01",
  # Endereço GT do Centro de Serviço para operações de SMS
  smsc_service_center_gt_address: "5551234567"
```

Parâmetros de Configuração:

Parâmetro	Tipo	Obrigatório	Padr
`smsc_api_base_url`	String	Sim	`"https://loca`
`smsc_name`	String	Não	`""` (usa `"{hostname}_SMS`
`smsc_service_center_gt_address`	String	Não	`"5551234567"`

Registro do Frontend

O sistema registra automaticamente com o OmniMessage na inicialização e **re-registra a cada 5 minutos** através do módulo `SMS.FrontendRegistry`. Isso permite que o OmniMessage:

- Rastreie frontends ativos para balanceamento de carga
- Monitore tempo de atividade e status de saúde

- Colete informações de configuração
- Gerencie roteamento de SMS distribuído entre múltiplos frontends

Detalhes da Implementação:

- **Intervalo de Registro**: 5 minutos (fixo)
- **Processo**: Iniciado automaticamente quando `smsc_mode_enabled: true`

Carga Única de Registro:

```
{
  "frontend_name": "omni-smsc01",
  "configuration": "{...}",
  "frontend_type": "SS7",
  "hostname": "smsc-server01",
  "uptime_seconds": 12345
}
```

Nota: O nome do frontend é retirado do parâmetro de configuração `smsc_name`. Se não estiver definido, o padrão é `"{hostname}_SMSc"`.

Comunicação da API OmniMessage

Quando o OmniSS7 recebe operações MAP da rede SS7 ou processa a fila de mensagens, ele se comunica com o OmniMessage para:

- **Registrar-se como um frontend ativo** e relatar status de saúde
- **Enviar mensagens originadas em móvel (MO)** recebidas de assinantes
- **Recuperar mensagens terminadas em móvel (MT)** da fila para entrega
- **Atualizar status de entrega** com relatórios de sucesso/falha
- **Consultar informações de roteamento** para encaminhamento de mensagens

Endpoint	Método	Propósito	Corpo da Requisição
`/api/frontends`	POST	Registrar instância de frontend	`{"frontend_name": "...", "frontend_type": "SMSc", "hostname": "...", "uptime_seconds": ...}`
`/api/messages_raw`	POST	Inserir nova mensagem SMS	`{"source_msisdn": "...", "source_smsc": "...", "message_body": "..."}`
`/api/messages`	GET	Obter fila de mensagens	Cabeçalho: `smsc: <smsc_name>`
`/api/messages/{id}`	PATCH	Marcar mensagem como entregue	`{"deliver_time": "...", "dest_smsc": "..."}`
`/api/messages/{id}`	PUT	Atualizar status da mensagem	`{"dest_smsc": null}`
`/api/locations`	POST	Inserir/atualizar localização do assinante	`{"msisdn": "...", "imsi": "...", "location": "...", "ims_capable": true, "csfb": false, "expires": "...", "user_agent": "...",`

Endpoint	Método	Propósito	Corpo da Requisição
			`"ran_location": "...", "imei": "...", "registered": "..."}`
`/api/events`	POST	Adicionar rastreamento de eventos	`{"message_id": ..., "name": "...", "description": "..."}`
`/api/status`	GET	Verificação de saúde	-

Formato de Resposta da API

Todas as respostas da API usam o formato JSON com as seguintes convenções:

- **Respostas de sucesso**: HTTP 200-201 com corpo JSON contendo dados de resultado
- **Respostas de erro**: HTTP 4xx/5xx com detalhes de erro no corpo da resposta
- **Carimbos de data/hora**: Formato ISO 8601 (ex.: `"2025-10-21T12:34:56Z"`)
- **IDs de Mensagem**: Identificadores inteiros ou de string

Módulos de Cliente da API

O sistema SMS consiste em três módulos principais:

1. SMSc.APIClient

Módulo principal do cliente da API que fornece toda a comunicação da API HTTP com o OmniMessage:

- `frontend_register/4` - Registrar frontend com o OmniMessage
- `insert_message/3` - Inserir mensagem SMS bruta (versão de 3 parâmetros compatível com Python)
- `insert_location/9` - Inserir/atualizar dados de localização do assinante
- `get_message_queue/2` - Recuperar mensagens pendentes da fila
- `mark_dest_smsc/3` - Marcar mensagem como entregue ou falhada
- `add_event/3` - Adicionar rastreamento de eventos para mensagens
- `flush_queue/2` - Processar mensagens pendentes (SRI-for-SM + MT-forwardSM)
- `auto_flush/2` - Loop de processamento contínuo da fila

2. SMS.FrontendRegistry

Gerencia o registro periódico do frontend com o backend:

- Registra automaticamente na inicialização
- Re-registra a cada 5 minutos
- Usa `smsc_name` da configuração (retorna ao hostname se não estiver definido)
- Coleta informações de configuração e tempo de atividade do sistema

3. SMS.Utils

Funções utilitárias para operações SMS:

- `generate_tp_scts/0` - Gerar timestamp de SMS no formato TPDU

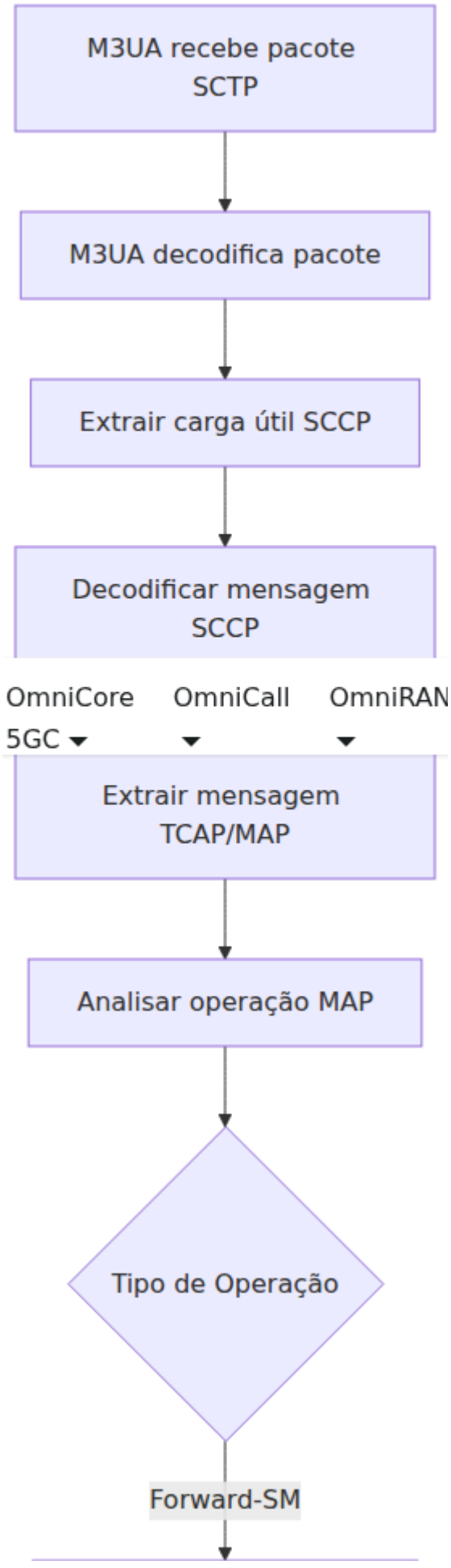
Fluxos de Mensagens SMS

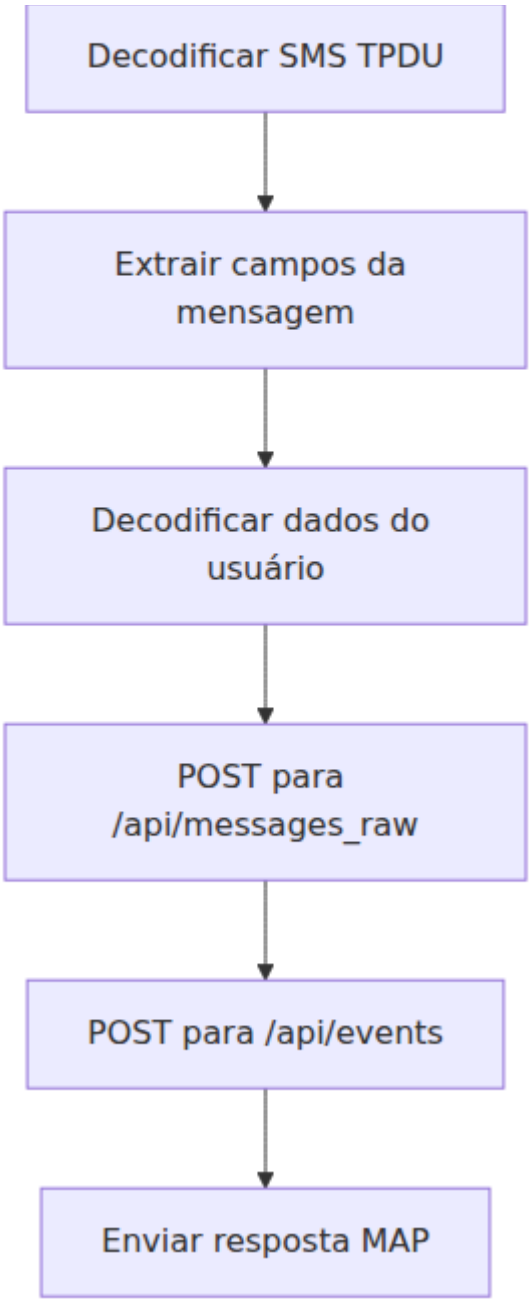
Fluxo de SMS de Entrada (Originado em Móvel)


M3UA recebe pacote SCTP
M3UA decodifica pacote
Extrair carga útil SCCP
Decodificar mensagem SCCP
Extrair mensagem TCAP/MAP
Analisar operação MAP
Tipo de Operação
Forward-SM

Fluxo de SMS de Saída (Terminado em Móvel)

Quando uma API HSS/HLR é configurada (`hlr_api_base_url`), o IP-SM-GW pode entregar MT-SMS para **assinantes on-net** sem uma rodada SRI-for-SM, consultando diretamente a API HSS para o IMSI real do assinante e o VLR que atende. Um **cache VLR** no Rastreador de Assinantes evita consultas repetidas ao HSS para o mesmo assinante.


M3UA recebe pacote SCTP
M3UA decodifica pacote
Extrair carga útil SCCP
Decodificar mensagem SCCP
OmniCore 5GC
OmniCall
OmniRAN
Extrair mensagem TCAP/MAP
Analisar operação MAP
Tipo de Operação
Forward-SM



Principais Etapas Explicadas:

- **Verificação de Cache VLR**: Antes de consultar a API HSS, o sistema verifica o cache VLR do Rastreador de Assinantes para uma entrada válida (IMSI + endereço VLR dentro do TTL). Um hit de cache ignora completamente a API HSS. Consulte Cache de Endereços VLR para detalhes.

- **Entrega Direta da API HSS (on-net)**: Quando `hlr_api_base_url` está configurado, o IP-SM-GW consulta a API HSS com o MSISDN de destino para obter o **IMSI real** do assinante e o **endereço VLR que atende**. Se ambos

estiverem disponíveis, o MT-ForwardSM é enviado diretamente para o VLR — ignorando completamente o SRI-for-SM.

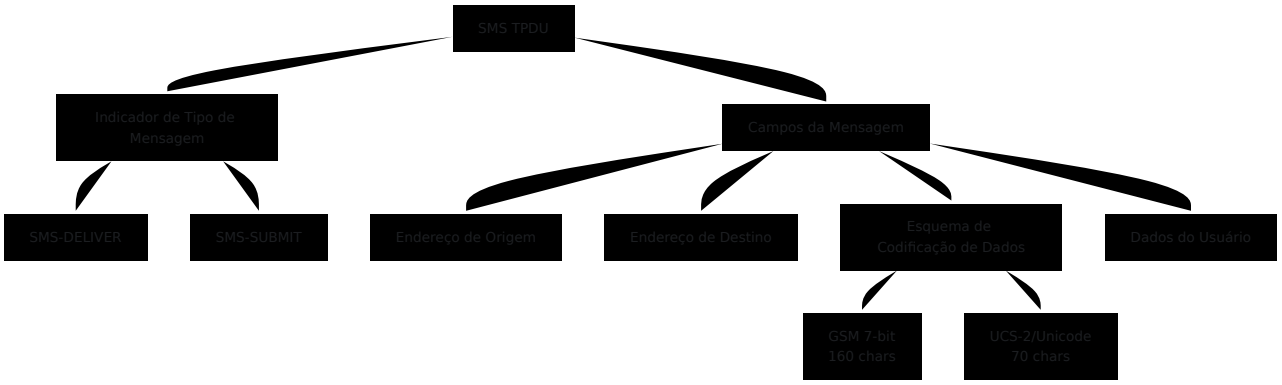
- **Assinante Ausente (on-net)**: Se a API HSS encontrar o assinante, mas nenhum VLR de atendimento estiver atribuído, a mensagem é marcada como falhada com uma entrada de localização expirada. As tentativas de entrega são feitas quando o HLR envia um alertServiceCenter após o assinante se registrar novamente.

- **Fallback SRI-for-SM (off-net)**: Quando a API HSS não está configurada ou o assinante não é encontrado no HSS, o fluxo padrão SRI-for-SM é utilizado. O HLR responde com um IMSI sintético e número de nó de rede. Consulte SRI-for-SM no Guia HLR.

- **Limpeza de VLR em Caso de Falha**: Quando a entrega do MT-ForwardSM falha (timeout, erro SCCP, etc.), o endereço VLR em cache é limpo para que a próxima tentativa consulte novamente a API HSS em busca de um novo VLR.

- **Timeout do MT-ForwardSM**: O diálogo MT-ForwardSM tem um **timeout de 30 segundos** para acomodar respostas mais lentas de VLR em 2G/CS.

Estrutura do SMS TPDU



Tratamento de Centro de Serviço de Alerta

O SMSc pode receber mensagens **alertServiceCenter** do HLR para rastrear o status de alcançabilidade do assinante.

Para informações sobre como o HLR envia mensagens alertServiceCenter, consulte Integração do Centro de Serviço de Alerta no Guia HLR.

O que é alertServiceCenter?

Quando um assinante realiza uma Atualização de Localização no HLR (ou seja, registra-se com um novo VLR/MSC), o HLR pode notificar sistemas SMSc que o assinante agora está acessível enviando uma mensagem **alertServiceCenter** (opcode MAP 64).

Configuração

O tempo de expiração da localização é configurado no HLR:

```
config :omniss7,
  # Tempo de expiração da localização quando o SMSc recebe
alertServiceCenter (padrão: 48 horas)
  hlr_alert_location_expiry_seconds: 172800
```

Comportamento

Quando o SMSc recebe uma mensagem alertServiceCenter:

1. **Decodificar MSISDN**: Extrair o MSISDN do assinante da mensagem (formato TBCD)
2. **Remover prefixo TON/NPI**: Remover prefixos comuns como "19", "11", "91" (ex.: "19123123213" → "123123213")
3. **Resolver IMSI**: Se `hlr_api_base_url` estiver configurado, consultar a API HSS para resolver o **IMSI real** do assinante. Se a consulta HSS falhar ou nenhuma API estiver configurada, retornar à geração de um IMSI sintético usando o mesmo mapeamento que SRI-for-SM. Usar o IMSI real é crítico para entrega MT para assinantes 2G/CS — o MSC/VLR requer o IMSI correto no MT-ForwardSM.
4. **POST para /api/location**: Atualizar o banco de dados de localização com:
   - `msisdn`: Número de telefone do assinante (limpo)
   - `imsi`: IMSI sintético
   - `location`: Nome do SMSc (ex.: "ipsmgw")
   - `expires`: Hora atual + `hlr_alert_location_expiry_seconds`
   - `csfb`: true (assinante acessível via Circuit-Switched Fallback)
   - `ims_capable`: false (isto é registro 2G/3G CS, não IMS/VoLTE)
   - `user_agent`: GT do HLR que enviou o alerta (para rastreamento)
   - `ran_location`: "SS7"
5. **Rastrear no Rastreador de Assinantes SMSc**: Registrar o assinante com GT do HLR, status=ativo, contadores de mensagens em 0
6. **Enviar ACK**: Responder ao HLR com reconhecimento de alertServiceCenter

Tratamento de Assinante Ausente

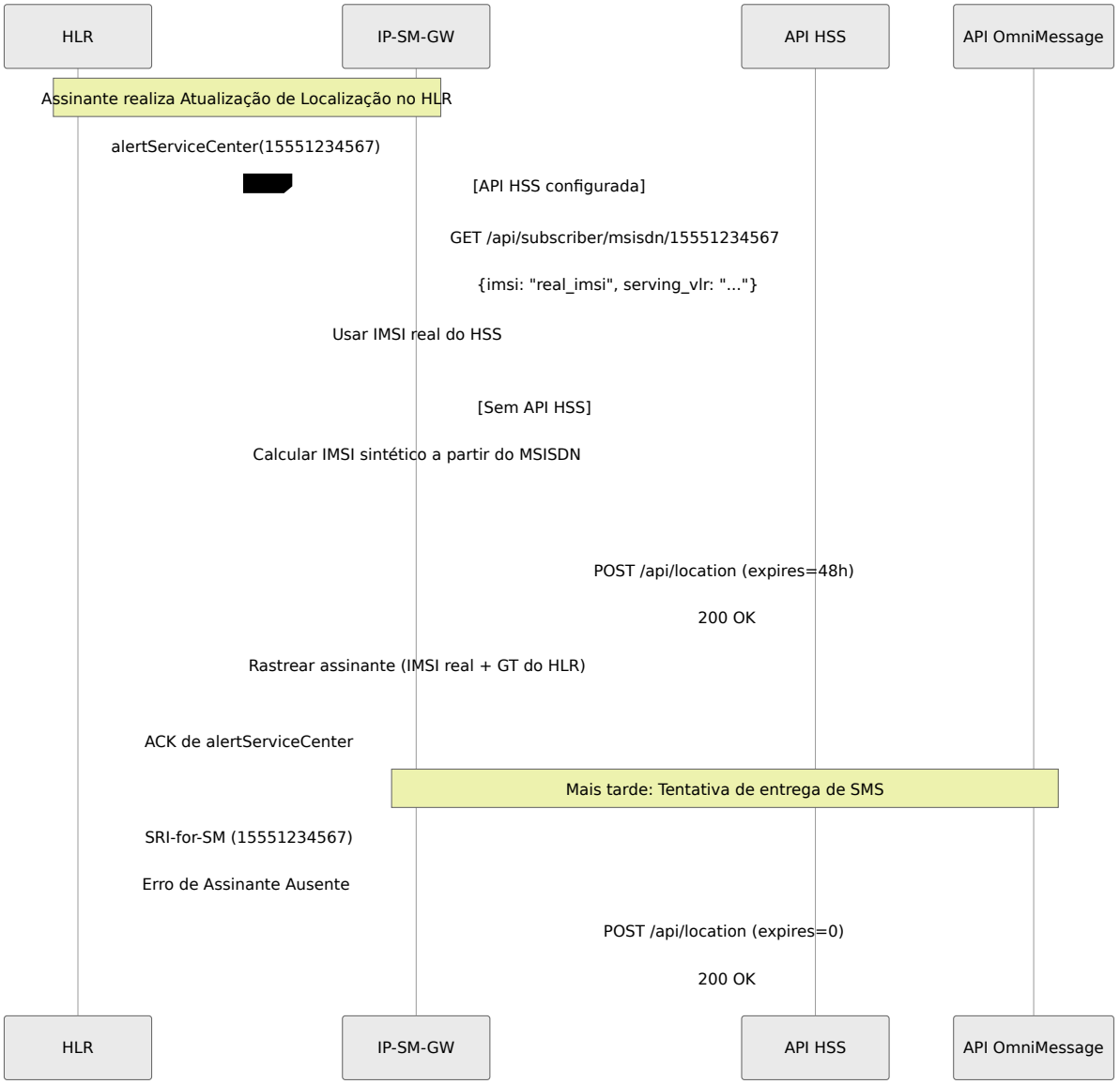
Quando o SMSc tenta entregar uma mensagem e recebe um erro de "assinante ausente" durante SRI-for-SM (para mais informações sobre SRI-for-SM, consulte SRI-for-SM no Guia HLR):

1. **Detectar ausência**: SRI-for-SM retorna erro `absentSubscriberDiagnosticSM`

2. **Expirar localização**: POST para /api/location com `expires=0` para marcar o assinante como inacessível
3. **User agent**: Definir como "SS7_AbsentSubscriber" para identificar a origem
4. **Atualizar rastreador**: Marcar o assinante como `falhado` no Rastreador de Assinantes SMSc

Isso garante que o banco de dados de localização e o rastreador reflitam com precisão o status de alcançabilidade do assinante.

Diagrama de Fluxo



Endpoint da API

POST /api/location

```
{
  "msisdn": "15551234567",
  "imsi": "001010123456789",
  "location": "ipsmgw",
  "ims_capable": false,
  "csfb": true,
  "expires": "2025-11-01T12:00:00Z",
  "user_agent": "15551111111",
  "ran_location": "SS7",
  "imei": "",
  "registered": "2025-10-30T12:00:00Z"
}
```

Nota: O campo `user_agent` contém o GT do HLR que enviou o alertServiceCenter, permitindo que o SMSc rastreie qual HLR está fornecendo atualizações de localização.

Para assinantes ausentes, `expires` é definido como o horário atual (expiração imediata).

Prevenção de Loop

O SMSc implementa **prevenção automática de loop** para evitar loops infinitos de roteamento de mensagens quando as mensagens se originam de redes SS7, permitindo ainda a entrega legítima MO→MT entre assinantes on-net.

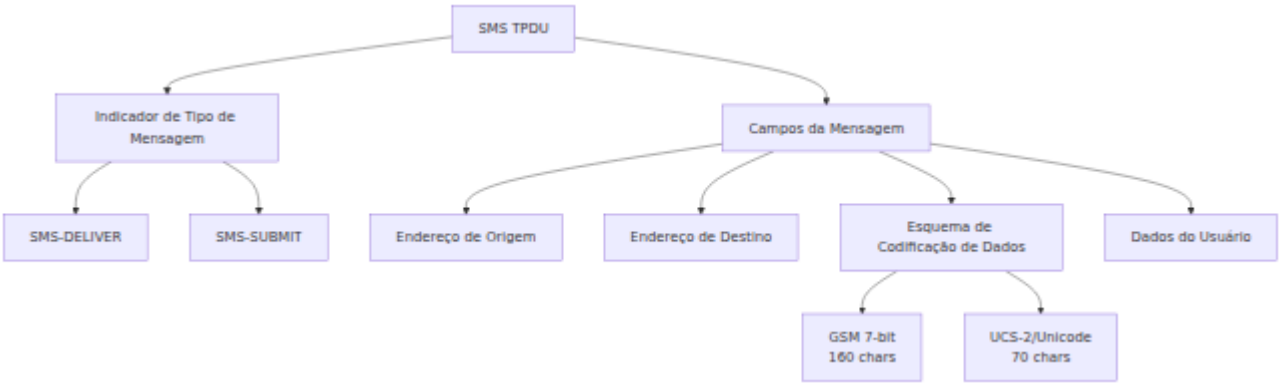
Por que a Prevenção de Loop é Importante

Quando o SMSc recebe mensagens SMS originadas em móvel (MO) da rede SS7, ele as insere na fila de mensagens com um campo `source_smsc` identificando sua origem (ex.: `"SS7_MO_15551234567"`). Sem a prevenção de loop, essas mensagens poderiam ser:

1. Recebidas da rede SS7 → Enfileiradas com `source_smsc` contendo "SS7"
2. Recuperadas da fila → Processadas para entrega
3. Enviadas de volta para a rede SS7 → Criando um loop

Como Funciona

A lógica de prevenção de loop distingue entre destinos **on-net** e **off-net**. Mensagens da SS7 são bloqueadas apenas quando o assinante de destino **não está on-net** — isso evita loops de roteamento de trânsito enquanto permite a entrega legítima MO→MT entre assinantes na mesma rede (ex.: MSC para MSC via IP-SM-GW).



Detecção On-Net

Quando uma mensagem se origina da SS7, o sistema verifica se o assinante de destino está on-net antes de aplicar a prevenção de loop:

1. **Cache VLR** (mais rápido): Verificar se o assinante tem um endereço VLR em cache com IMSI válido no Rastreador de Assinantes
2. **API HSS** (autoritativa): Se `hlr_api_base_url` estiver configurado e o cache falhar, consultar a API HSS por MSISDN. Se o assinante existir no HSS, ele está on-net.

Se nenhuma verificação encontrar o assinante, o destino é considerado **off-net** e a mensagem é bloqueada.

Implementação

Ao processar mensagens da fila, o SMSc verifica o campo `source_smsc`:

- **Se `source_smsc` contém "SS7" e o destino é off-net**:
  - Mensagem é ignorada
  - Evento adicionado: `"Prevenção de Loop"` com descrição informando que o destino é off-net
  - Mensagem marcada como falhada via requisição PUT
  - Registrada com nível de aviso
- **Se `source_smsc` contém "SS7" e o destino é on-net**:
  - Mensagem processada normalmente — este é um fluxo legítimo MO→MT
  - A entrega prossegue via caminho da API HSS / cache VLR
- **Se `source_smsc` não contém "SS7"**:
  - Mensagem processada normalmente (ex.: mensagens da API web, SMPP ou outras fontes não-SS7)

Valores de SMSC de Origem

As mensagens podem ter vários valores de `source_smsc`:

Origem	Valor de Exemplo	Ação
Rede SS7 (MO-FSM), destino off-net	`"SS7_MO_15551234567"`	**Ignorado** - Prevenção de Loop
Rede SS7 (MO-FSM), destino on-net	`"SS7_MO_15551234567"`	**Permitido** - Entrega on-net MO→MT
API Externa/SMPP	`"ipsmgw"` ou `"api_gateway"`	Processado normalmente
Outro SMSc	`"smsc-node-01"`	Processado normalmente

Rastreamento de Eventos

Quando uma mensagem é ignorada devido à prevenção de loop, um evento é registrado:

```
{
  "message_id": 12345,
  "name": "Prevenção de Loop",
  "description": "Mensagem ignorada - source_smsc
'SS7_MO_15551234567' contém 'SS7' e o destino é off-net,
prevenindo loop de mensagem"
}
```

Esse evento é visível em:

- **Interface Web**: Página de Eventos SS7 (`/events`)
- **Banco de Dados**: Tabela `events` via API
- **Logs**: Entradas de log de nível de aviso

Configuração

A prevenção de loop está **sempre habilitada** e não pode ser desativada. Este é um recurso crítico de segurança para evitar interrupções na rede devido a loops de mensagens. O bypass on-net requer tanto `smsc_mode_enabled: true` quanto o cache VLR ou `hlr_api_base_url` configurados.

Cenários de Exemplo

Cenário 1: MO-SMS da SS7 para assinante off-net (bloqueado)

```
1. Telefone móvel → MSC/VLR → IP-SM-GW (via MO-ForwardSM)
2. IP-SM-GW recebe MO-FSM, insere na fila: source_smsc =
"SS7_MO_2471900"
3. Auto-flush recupera mensagem da fila
4. Destino não no cache VLR ou HSS → off-net
5. IP-SM-GW detecta "SS7" em source_smsc + destino off-net →
IGNORAR
6. Evento registrado: "Prevenção de Loop"
7. Nenhum SRI-for-SM ou MT-ForwardSM enviado (loop prevenido)
```

Cenário 2: MO-SMS da SS7 para assinante on-net (permitido)

```
1. Telefone móvel → MSC/VLR → IP-SM-GW (via MO-ForwardSM)
2. IP-SM-GW recebe MO-FSM, insere na fila: source_smsc =
"SS7_MO_2471900"
3. Auto-flush recupera mensagem da fila
4. Destino encontrado na API HSS → assinante on-net
5. IP-SM-GW permite entrega apesar da origem SS7
6. HSS retorna IMSI real + VLR → MT-ForwardSM enviado diretamente
para o VLR
```

Rastreamento de Assinantes SMSc

O SMSc inclui um GenServer de **Rastreador de Assinantes** que mantém o estado em tempo real para assinantes com base em mensagens alertServiceCenter e tentativas de entrega de mensagens.

Propósito

O rastreador fornece:

- **Monitoramento de alcançabilidade**: Quais assinantes estão atualmente acessíveis
- **Rastreamento de HLR**: Qual HLR enviou o alertServiceCenter para cada assinante

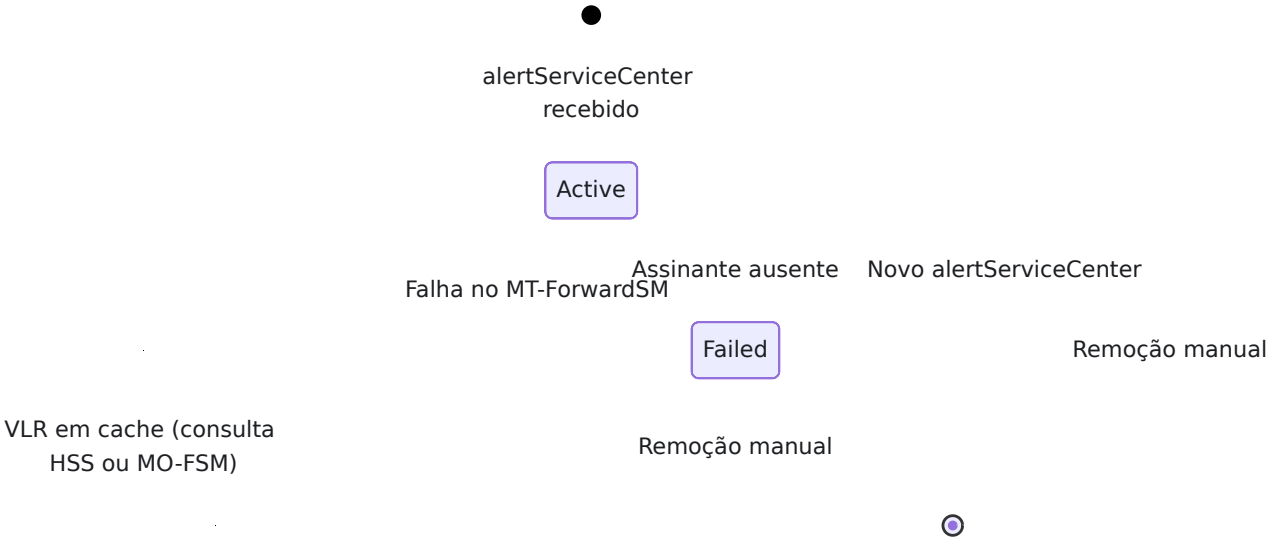
- **Contadores de mensagens**: Número de mensagens enviadas/recebidas por assinante
- **Rastreamento de falhas**: Marcar assinantes como falhados quando as tentativas de entrega falham
- **Visibilidade na Interface Web**: Painel em tempo real mostrando todos os assinantes rastreados

Informações Rastreadas

Para cada assinante, o rastreador armazena:

Campo	Descrição	Exemplo
`msisdn`	Número de telefone do assinante (chave)	"15551234567"
`imsi`	IMSI do assinante (real do HSS se disponível)	"001010123456789"
`hlr_gt`	GT do HLR que enviou alertServiceCenter	"15551111111"
`messages_sent`	Contagem de mensagens MT-FSM enviadas	5
`messages_received`	Contagem de mensagens MO-FSM recebidas	2
`status`	`:active` ou `:failed`	`:active`
`updated_at`	Timestamp Unix da última atualização	1730246400
`vlr_address`	Endereço VLR GT em cache	"14155550100"
`vlr_cached_at`	Timestamp Unix quando o VLR foi armazenado em cache pela última vez	1730246400

Transições de Estado



Comportamento

Quando alertServiceCenter é recebido:

- Criar ou atualizar entrada do assinante
- Definir `status = :active`
- Registrar GT do HLR
- Resolver IMSI real da API HSS (se configurada), caso contrário, usar IMSI sintético
- Preservar cache VLR existente e contadores de mensagens

Quando a API HSS retorna assinante com VLR:

- Armazenar o endereço VLR e IMSI real no rastreador
- Entregas subsequentes usam o VLR em cache sem reconsultar o HSS

Quando MO-ForwardSM é recebido:

- Armazenar o GT da parte chamadora SCCP como o endereço VLR do assinante (o GT chamador em um MO-ForwardSM é o VLR que retransmitiu a mensagem)
- Cria uma entrada no rastreador se uma não existir

Quando a entrega do MT-ForwardSM falha:

- Limpar o endereço VLR em cache para que a próxima tentativa de entrega reconsulte a API HSS

Quando SRI-for-SM é bem-sucedido:

- Incrementar contador `messages_sent`
- Atualizar timestamp `updated_at`

Quando SRI-for-SM falha:

- Definir `status = :failed`
- Manter no rastreador para monitoramento

Quando o assinante é removido:

- Excluir da tabela ETS
- Não aparece mais na Interface Web

Interface Web - Página de Assinantes SMSc

Caminho: `/smsc_subscribers` **Atualização automática**: A cada 2 segundos

Nota: Esta página está disponível apenas quando executada no modo SMSc. Após descomentar a configuração SMSc em `config/runtime.exs`, você deve reiniciar a aplicação para que a rota se torne disponível.

A página **Assinantes SMSc** fornece monitoramento em tempo real de todos os assinantes rastreados:

Recursos

1. **Tabela de Assinantes**
    - MSISDN, IMSI, GT do HLR, GT do VLR (endereço VLR em cache)
    - Contadores de mensagens enviadas/recebidas
    - Insígnia de status (Ativo/Falhado) com codificação de cores
    - Timestamp da última atualização e duração
    - Botão de remoção para assinantes individuais

2. **Estatísticas Resumidas**
    - Total de assinantes rastreados
    - Contagem de assinantes ativos
    - Contagem de assinantes falhados
    - Número de HLRs únicos
3. **Ações**
    - Limpar Tudo: Remover todos os assinantes rastreados
    - Remover: Remover assinante individual

Exemplo de Visualização

```
┌──────────────────────────────────────────────────────────────────────────────
│ Assinantes Rastreados SMSc                                          Total:
│
├──────────────────────────────────────────────────────────────────────────────
│ MSISDN          IMSI              HLR GT         VLR GT         Msg
│                                                                   S/
├──────────────────────────────────────────────────────────────────────────────
│ 15551234567     001010123456789   15551111111    14155550100    5/2
│ 15559876543     001010987654321   15551111111    -              0/0
│ 15551112222     001010111222233   15552222222    14155550200    3/1
└──────────────────────────────────────────────────────────────────────────────

Resumo:  Total: 3  |  Ativos: 2  |  Falhados: 1  |  HLRs Únicos: 2
```

Funções da API

O rastreador expõe essas funções para acesso programático:

```
# Chamado quando alertServiceCenter é recebido
SMSc.SubscriberTracker.alert_received(msisdn, imsi, hlr_gt)

# Incrementar contadores de mensagens
SMSc.SubscriberTracker.message_sent(msisdn)
SMSc.SubscriberTracker.message_received(msisdn)

# Marcar como falhado (falha SRI-for-SM)
SMSc.SubscriberTracker.mark_failed(msisdn)

# Remover do rastreamento
SMSc.SubscriberTracker.remove_subscriber(msisdn)

# Funções de consulta
SMSc.SubscriberTracker.get_active_subscribers()
SMSc.SubscriberTracker.get_subscriber(msisdn)
SMSc.SubscriberTracker.count_subscribers()
SMSc.SubscriberTracker.clear_all()
```

Integração

O rastreador está automaticamente integrado com:

- **Manipulador alertServiceCenter**: Chama `alert_received/3` na atualização de localização bem-sucedida
- **Manipulador SRI-for-SM**: Incrementa `messages_sent` na roteação bem-sucedida
- **Tratamento de assinante ausente**: Chama `mark_failed/1` quando o assinante está ausente
- **Erros de assinante desconhecido**: Chama `mark_failed/1` quando SRI-for-SM falha

---

Cache de Endereços VLR

Quando opera como um IP-SM-GW com uma API HSS configurada, o Rastreador de Assinantes armazena endereços VLR em cache para evitar consultas ao HSS em cada tentativa de entrega MT.

Como Funciona

O cache VLR é preenchido a partir de duas fontes:

1. **Consulta API HSS**: Quando o pipeline de entrega consulta a API HSS e obtém um assinante com um VLR de atendimento, o IMSI e o endereço VLR são armazenados em cache no Rastreador de Assinantes.
2. **MO-ForwardSM**: Quando um MO-ForwardSM chega de um assinante, o GT da parte chamadora SCCP (que é o VLR/MSC que retransmitiu a mensagem) é armazenado como o endereço VLR do assinante.

Em tentativas de entrega MT subsequentes, o cache é verificado primeiro. Se uma entrada válida existir (tanto IMSI quanto VLR presentes, dentro do TTL), a API HSS é completamente ignorada.

Invalidação do Cache

O cache VLR é limpo quando:

- **A entrega do MT-ForwardSM falha** (timeout, erro SCCP, rejeição do VLR) — o VLR em cache pode estar desatualizado
- **TTL expira** — configurável via `vlr_cache_ttl_seconds`, entradas mais antigas que o TTL são tratadas como falhas de cache

O cache **não** é limpo no alertServiceCenter — a entrada VLR existente é preservada, pois o assinante pode ainda estar acessível no mesmo VLR.

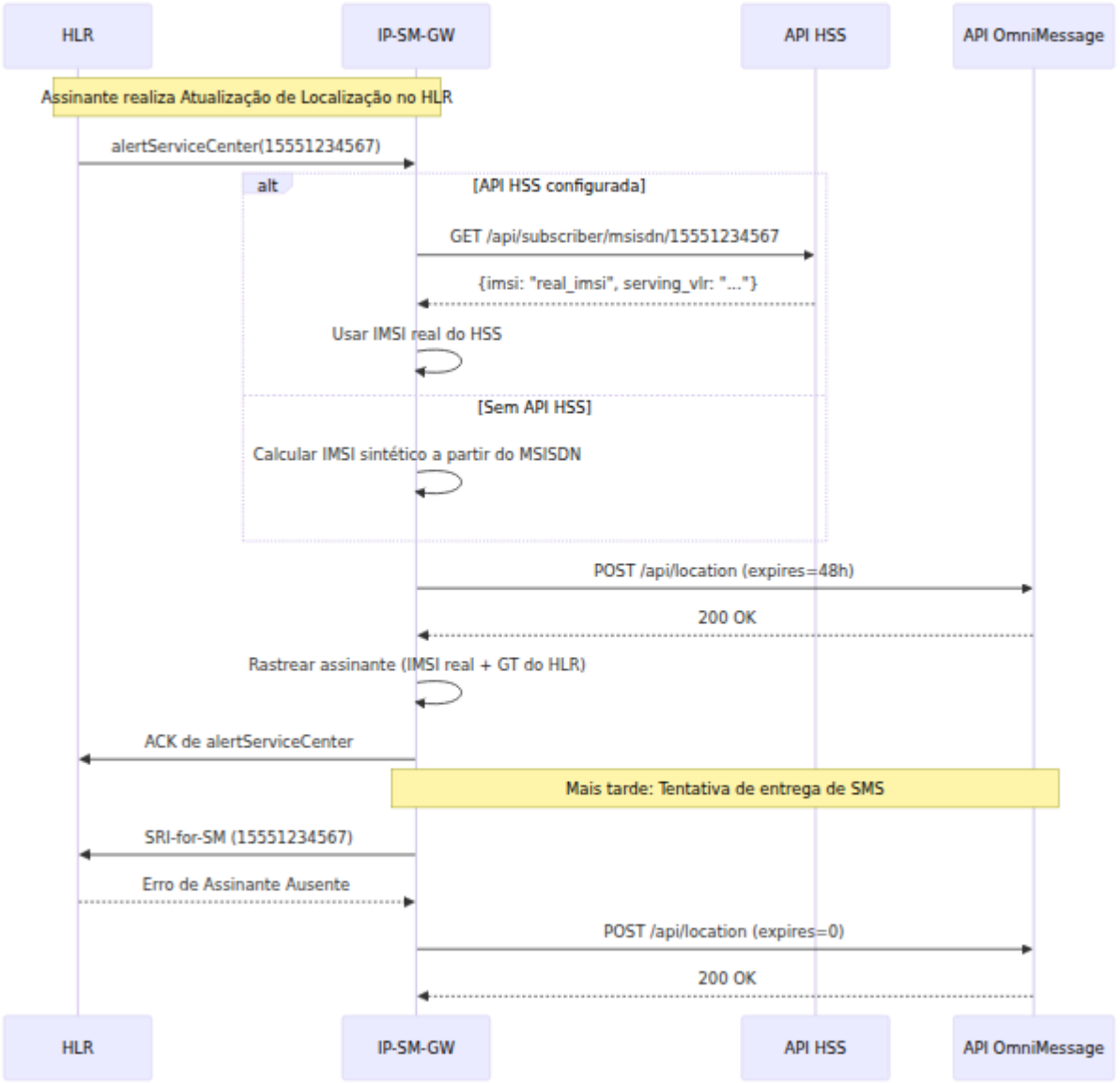
Configuração

```
config :omniss7,
  # Endpoint da API HSS/HLR (obrigatório para cache VLR e entrega
on-net)
  hlr_api_base_url: "https://10.179.2.140:8443",

  # TTL do cache VLR em segundos (padrão: 3600 = 1 hora)
  vlr_cache_ttl_seconds: 3600
```

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
`hlr_api_base_url`	String	Não	`nil`	URL base da API HSS/HLR. Quando configurado, habilita consulta de assinantes on-net, cache VLR e entrega MT direta. Quando `nil`, todas as entregas usam o fluxo SRI-for-SM.
`vlr_cache_ttl_seconds`	Inteiro	Não	`3600`	Idade máxima em segundos para uma entrada VLR em cache ser considerada válida. Após esse período, a próxima tentativa de entrega reconsulta a API HSS.

Parâmetro	Tipo	Obrigatório	Padrão	Descrição
				Valores mais baixos aumentam o tráfego da API HSS, mas melhoram a precisão para assinantes móveis.

Diagrama de Sequência



Configuração de Auto-Flush da Fila de SMS

O serviço **Auto-Flush** processa automaticamente mensagens SMS pendentes.

Para referência de parâmetros de configuração, consulte Configuração de Auto-Flush na Referência de Configuração.

Configuração

```
config :omniss7,
  auto_flush_enabled: true,           # Habilitar/desabilitar
auto-flush
  auto_flush_interval: 10_000,        # Intervalo de consulta em
milissegundos
  auto_flush_dest_smsc: nil,          # Filtro: nil = todos
  auto_flush_tps: 10                  # Máximo de transações por
segundo
```

Como Funciona

1. **Consulta**: A cada `auto_flush_interval` milissegundos, consulta a API por mensagens pendentes
2. **Filtragem**: Opcionalmente filtra por `auto_flush_dest_smsc`
3. **Limitação de Taxa**: Processa até `auto_flush_tps` mensagens por ciclo
4. **Entrega**: Para cada mensagem:
   - Envia **SRI-for-SM** (Enviar Informações de Roteamento para Mensagem Curta) para o HLR para obter informações de roteamento
     - O HLR retorna um IMSI sintético calculado a partir do MSISDN
     - O HLR retorna o endereço GT do SMSC para onde o MT-ForwardSM deve ser enviado
     - Consulte Detalhes SRI-for-SM no Guia HLR para documentação completa
   - Em caso de sucesso, envia **MT-forwardSM** para MSC/VLR
   - Atualiza o status da mensagem via API (entregue/falhado)
   - Adiciona rastreamento de eventos via API

> **Mergulho Técnico**: Para uma explicação completa de como SRI-for-SM funciona, incluindo mapeamento de MSISDN para IMSI, configuração do endereço do centro de serviço e geração de IMSI sintético que preserva a privacidade, consulte a seção SRI-for-SM no Guia de Configuração HLR.

Métricas SMSc

Métricas Disponíveis

Métricas da Fila de SMS:

- `smsc_queue_depth` - Número atual de mensagens pendentes
- `smsc_messages_delivered_total` - Total de mensagens entregues com sucesso
- `smsc_messages_failed_total` - Total de mensagens que falharam na entrega
- `smsc_delivery_duration_milliseconds` - Histograma dos tempos de entrega

Consultas de Exemplo:

```
# Profundidade atual da fila
smsc_queue_depth

# Taxa de sucesso de entrega (últimos 5 minutos)
rate(smsc_messages_delivered_total[5m]) /
(rate(smsc_messages_delivered_total[5m]) +
rate(smsc_messages_failed_total[5m]))

# Tempo médio de entrega
rate(smsc_delivery_duration_milliseconds_sum[5m]) /
rate(smsc_delivery_duration_milliseconds_count[5m])
```

Solução de Problemas SMSc

Problema: Mensagens Não Entregues

Verificações:

1. Verifique se o auto-flush está habilitado

2. Verifique a conexão com o banco de dados
3. Monitore os logs em busca de erros
4. Verifique se a conexão M3UA está ATIVA
5. Verifique os limites de TPS

Problema: Alta Profundidade da Fila

Causas Possíveis:

- Limite de TPS muito baixo
- Problemas de timeout no HLR
- Problemas de conectividade de rede
- Números de destino inválidos

Soluções:

- Aumente `auto_flush_tps`
- Verifique a disponibilidade do HLR
- Revise os logs de mensagens falhadas

API MT-forwardSM

Enviar SMS via API

Endpoint da API: `POST /api/MT-forwardSM`

Requisição:

```
{
  "imsi": "234509876543210",
  "destination_serviceCentre": "447999555111",
  "originating_serviceCenter": "447999123456",
  "smsPDU":
"040B917477218345F60000157030185714 0C0BD4F29C0E9281C4E1F11A"
}
```

Resposta:

```
{
  "result": "success",
  "message_id": "12345"
}
```

Documentação Relacionada

Documentação OmniSS7:

- ← Voltar à Documentação Principal
- Guia de Configuração HLR - Configuração e operações do modo HLR
  - **Detalhes Técnicos SRI-for-SM** - Documentação completa sobre mapeamento de MSISDN para IMSI e configuração do centro de serviço
- Guia de Recursos Comuns - Interface Web, API, Monitoramento
- Guia do Cliente MAP - Operações MAP
- Referência Técnica - Especificações de protocolo

Documentação OmniMessage: Para configuração de roteamento de mensagens, gerenciamento de filas, rastreamento de entrega, limitação de taxa e análises, consulte a **documentação do produto OmniMessage**. O OmniMessage contém toda a lógica de roteamento de mensagens, algoritmos de tentativas de fila, tratamento de relatórios de entrega e motor de regras de negócios.

Guia de Configuração do STP M3UA & M2PA

← Voltar para a Documentação Principal

Este guia fornece uma configuração detalhada para usar o OmniSS7 como um **Ponto de Transferência de Sinalização (STP)**.

Índice

O que é um Ponto de Transferência de Sinalização (STP)?

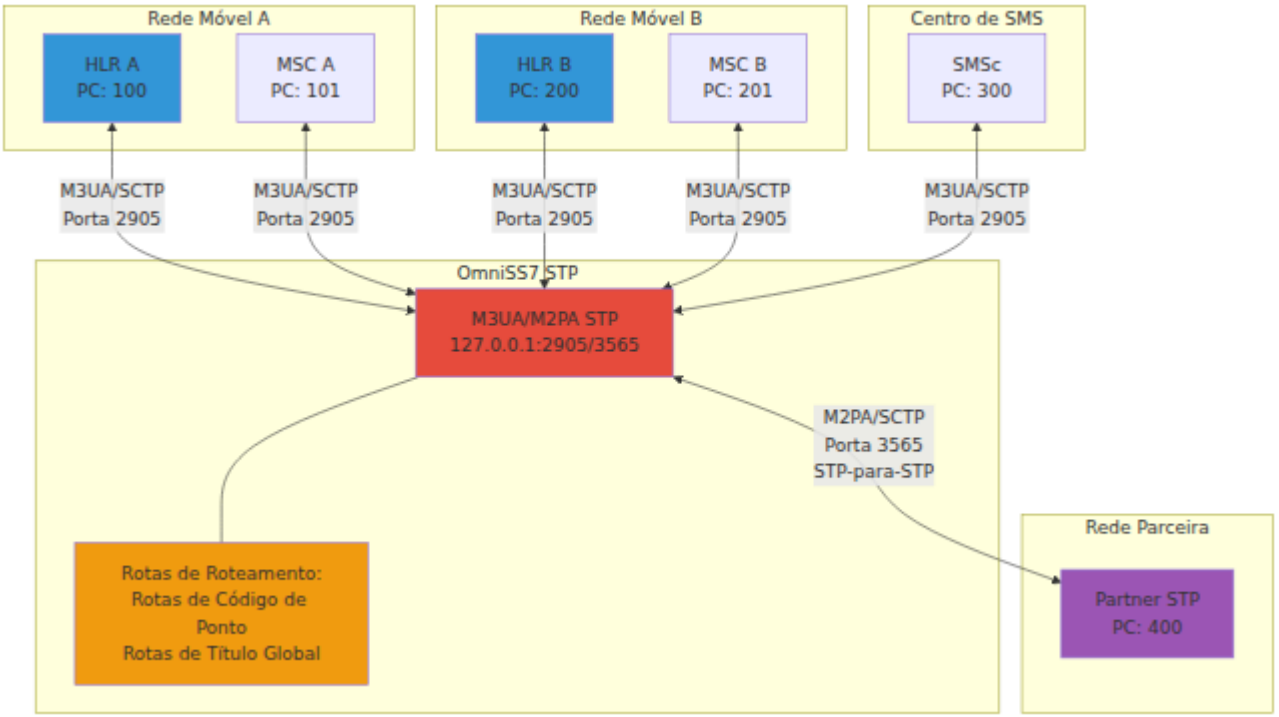
Um **Ponto de Transferência de Sinalização (STP)** é um elemento crítico da rede em redes de sinalização SS7 e baseadas em IP que roteia mensagens de sinalização entre os nós da rede.

Funções do STP

- **Roteamento de Mensagens**: Roteia o tráfego de sinalização SS7 com base no Código de Ponto de Destino (PC) ou Título Global (GT)
- **Tradução de Protocolo**: Conecta redes SS7 tradicionais com redes M3UA/SCTP baseadas em IP
- **Distribuição de Carga**: Distribui o tráfego entre múltiplos destinos usando roteamento baseado em prioridade

- **Gateway de Rede**: Conecta diferentes redes de sinalização e provedores de serviços
- **Ocultação de Topologia**: Pode reescrever endereços para ocultar a topologia interna da rede

Diagrama de Rede do STP



Funções de Rede do STP Explicadas

ASP (Processo de Servidor de Aplicação)

- **Função**: Cliente conectando a um SGP/STP remoto
- **Direção**: Conexão de saída
- **Caso de Uso**: Seu STP conecta-se ao STP de uma rede parceira

SGP (Processo de Gateway de Sinalização)

- **Função**: Servidor aceitando conexões de ASPs

- **Direção**: Conexão de entrada
- **Caso de Uso**: Redes parceiras conectam-se ao seu STP

AS (Servidor de Aplicação)

- **Definição**: Agrupamento lógico de um ou mais ASPs
- **Objetivo**: Fornece redundância e compartilhamento de carga
- **Caso de Uso**: Vários ASPs atendendo ao mesmo destino

Interfaciando com Redes TDM via Portas de Sinalização

OmniSS7 é uma plataforma de sinalização baseada em IP que utiliza protocolos SIGTRAN (M3UA/M2PA sobre SCTP). Para trocar sinalização com redes **SS7 baseadas em TDM** legadas, você precisa de um **Gateway de Sinalização (SGW)** para conectar os dois mundos.

O que é um Gateway de Sinalização?

Um **Gateway de Sinalização (SGW)** é um elemento de rede que converte a sinalização SS7 entre:

- **Lado TDM**: SS7 tradicional usando MTP1/MTP2/MTP3 sobre links E1/T1
- **Lado IP**: SIGTRAN usando M3UA ou M2PA sobre SCTP/IP

O SGW atua como um tradutor de protocolo, permitindo que aplicações baseadas em IP como o OmniSS7 se comuniquem com equipamentos TDM legados (switches, STPs, HLRs) que não suportam sinalização IP.

Arquitetura TDM para IP

Comparação da Pilha de Protocolos

Camada	TDM SS7	Gateway de Sinalização	IP (SIGTRAN)
Parte do Usuário	SCCP/TCAP/MAP	← Transparente →	SCCP/TCAP/MAP
Rede	MTP3	Conversão	M3UA/M2PA
Link	MTP2	Terminação	SCTP
Física	MTP1 (E1/T1)	Terminação	IP/Ethernet

O SGW termina MTP1/MTP2 no lado TDM e apresenta os dados do usuário MTP3 sobre M3UA ou M2PA no lado IP. As camadas superiores (SCCP, TCAP, MAP, CAP) passam de forma transparente.

Conectando o OmniSS7 a um Gateway de Sinalização

OmniSS7 conecta-se a um Gateway de Sinalização como um **ASP (Processo de Servidor de Aplicação)**, enquanto o SGW atua como um **SGP (Processo de Gateway de Sinalização)**.

Exemplo de Configuração

```
config :omniss7,
  # Conectar ao Gateway de Sinalização como ASP
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :sgw_connection,
    # Endpoint local (OmniSS7)
    local_ip: {10, 0, 0, 1},
    local_port: 0,                    # Porta local dinâmica
    # Endpoint do Gateway de Sinalização
    remote_ip: {10, 0, 0, 100},       # Endereço IP do SGW
    remote_port: 2905,                # Porta M3UA do SGW
    routing_context: 1                # Atribuído pelo SGW
  },

  # Ou configurar como um par para roteamento STP
  peers: [
    %{
      peer_id: 1,
      name: "TDM_Gateway",
      role: :client,                  # OmniSS7 inicia a conexão
      local_ip: {10, 0, 0, 1},
      local_port: 0,
      remote_ip: {10, 0, 0, 100},     # Endereço IP do SGW
      remote_port: 2905,
      routing_context: 1,
      point_code: 100,                # Faixa de código de ponto
atrás do SGW
      network_indicator: :international
    }
  ],

  # Roteie os códigos de ponto da rede TDM através do gateway
  m3ua_routes: [
    %{
      dest_pc: 100,                   # Código de ponto do MSC TDM
      peer_id: 1,                     # Roteie via par SGW
      priority: 1,
      network_indicator: :international
    },
    %{
      dest_pc: 200,                   # Código de ponto do HLR TDM
      peer_id: 1,
```

```
      priority: 1,
      network_indicator: :international
    }
  ]
```

Considerações de Configuração do SGW

Ao configurar seu Gateway de Sinalização para trabalhar com o OmniSS7:

Parâmetro	Descrição	Valor Típico
IP Remoto	Endereço IP do OmniSS7	IP do seu servidor
Porta Remota	Porta SCTP do OmniSS7	2905
Contexto de Roteamento	Identificador AS no SGW	Atribuído pelo administrador do SGW
Códigos de Ponto	Códigos de ponto da rede TDM acessíveis via SGW	Específico da rede
Modo de Tráfego	Modo de tratamento de tráfego ASP	Override ou Loadshare

Cenários de Implantação

Cenário 1: Aplicação IP Acessando Rede TDM

OmniSS7 envia consultas MAP (SRI-SM, PRN) para HLRs baseados em TDM via SGW:

```
OmniSS7 (ASP) → SGW (SGP) → Rede TDM → HLR
     M3UA          MTP2          MTP3
```

Cenário 2: OmniSS7 como STP Entre IP e TDM

OmniSS7 roteia tráfego entre elementos de rede baseados em IP e redes TDM:

```
IP SMSC → OmniSS7 STP → SGW → HLR TDM
                ↓
          HLR baseado em IP
```

Cenário 3: SGW Duplo para Redundância

Conecte-se a dois Gateways de Sinalização para alta disponibilidade:

```
peers: [
  %{
    peer_id: 1,
    name: "SGW_Primary",
    role: :client,
    remote_ip: {10, 0, 0, 100},
    remote_port: 2905,
    point_code: 100,
    # ... outra configuração
  },
  %{
    peer_id: 2,
    name: "SGW_Backup",
    role: :client,
    remote_ip: {10, 0, 0, 101},
    remote_port: 2905,
    point_code: 100,
    # ... outra configuração
  }
],

m3ua_routes: [
  # Rota primária via SGW_Primary
  %{dest_pc: 100, peer_id: 1, priority: 1, network_indicator:
:international},
  # Rota de backup via SGW_Backup
  %{dest_pc: 100, peer_id: 2, priority: 2, network_indicator:
:international}
]
```

Habilitando o Modo STP

OmniSS7 pode operar em diferentes modos. Para usá-lo como um STP, você precisa habilitar o modo STP na configuração.

Mudando para o Modo STP

O arquivo `config/runtime.exs` do OmniSS7 contém três modos operacionais pré-configurados. Para habilitar o modo STP:

1. **Abra** `config/runtime.exs`
2. **Encontre** as três seções de configuração (linhas 53-174):
   - Configuração 1: Modo STP (linhas 53-85)
   - Configuração 2: Modo HLR (linhas 87-123)
   - Configuração 3: Modo SMSc (linhas 125-174)
3. **Comente** a configuração atualmente ativa (adicione `#` a cada linha)
4. **Descomente** a configuração STP (remova `#` das linhas 53-85)
5. **Personalize** os parâmetros de configuração conforme necessário
6. **Reinicie** a aplicação: `iex -S mix`

Configuração do Modo STP

A configuração completa do STP fica assim:

```
config :omniss7,
  # Flags de modo - Habilitar recursos STP apenas
  map_client_enabled: true,
  hlr_mode_enabled: false,
  smsc_mode_enabled: false,

  # Configuração de Conexão M3UA
  # Conectar como ASP (Processo de Servidor de Aplicação) ao
STP/SGW remoto
  map_client_m3ua: %{
    mode: "ASP",
    callback: {MapClient, :handle_payload, []},
    process_name: :stp_client_asp,
    # Endpoint local (este sistema)
    local_ip: {10, 179, 4, 10},
    local_port: 2905,
    # Endpoint remoto STP/SGW
    remote_ip: {10, 179, 4, 11},
    remote_port: 2905,
    routing_context: 1
  }
```

Parâmetros de Configuração a Personalizar

Para uma referência completa de todos os parâmetros de configuração, consulte a Referência de Configuração.

Parâmetro	Tipo	Padrão	Descrição	Exemplo
`map_client_enabled`	Booleano	`true`	Habilitar cliente MAP e capacidades de roteamento	`true`
`local_ip`	Tupla ou Lista	*Obrigatório*	Endereço(s) IP do seu sistema. Único: `{10, 0, 0, 1}` ou Lista para multihoming: `[{10, 0, 0, 1}, {10, 0, 0, 2}]`	`{10, 179, 4, 10}`
`local_port`	Inteiro	`2905`	Porta SCTP local	`2905`
`remote_ip`	Tupla ou Lista	*Obrigatório*	Endereço(s) IP do STP/SGW remoto. Único ou Lista para multihoming	`{10, 179, 4, 11}`
`remote_port`	Inteiro	`2905`	Porta SCTP remota	`2905`
`routing_context`	Inteiro	`1`	ID do contexto de	`1`

Parâmetro	Tipo	Padrão	Descrição	Exemplo
			roteamento M3UA	
`enable_gt_routing`	Booleano	`false`	Habilitar roteamento de Título Global (além do roteamento de PC)	`true`

> **Dica**: Use multihoming SCTP fornecendo uma lista de endereços IP para `local_ip` e/ou `remote_ip` para habilitar failover automático. Consulte o Guia de Multihoming SCTP.

O que Acontece Quando o Modo STP é Habilitado

Quando `map_client_enabled: true`, a interface web mostrará:

- **Eventos SS7** - Registro de eventos
- **Cliente SS7** - Testes de operação MAP
- **M3UA** - Status da conexão
- **Roteamento** - Gerenciamento da tabela de rotas ← *Específico do STP*
- **Teste de Roteamento** - Teste de rotas ← *Específico do STP*
- **Recursos** - Monitoramento do sistema
- **Configuração** - Visualizador de configuração

As abas **Links HLR** e **Links SMSc** serão ocultadas.

Notas Importantes

- O protocolo SCTP (protocolo IP 132) deve ser permitido através de firewalls
- A porta M3UA padrão é 2905 (padrão da indústria)

- Garantir recursos de sistema suficientes para lidar com o tráfego de roteamento
- **Persistência de Roteamento**: Todas as rotas configuradas via interface web ou API são armazenadas no **banco de dados Mnesia** e **sobrevivem a reinicializações**
- **Mesclagem de Configuração**: Rotas de `runtime.exs` são carregadas na inicialização e mescladas com rotas Mnesia
- Após alterar modos, você deve reiniciar a aplicação para que as alterações tenham efeito
- **Interface Web**: Consulte o Guia da Interface Web para gerenciar rotas via interface web
- **Acesso API**: Consulte o Guia da API para documentação da API REST e acesso ao Swagger UI

Modo STP Autônomo

Além das capacidades de roteamento STP disponíveis quando `map_client_enabled: true`, você pode executar um **servidor STP M3UA autônomo** que escuta por conexões de entrada.

Habilitando o STP Autônomo

Adicione esta configuração a `config/runtime.exs`:

```
config :omniss7,
  sctp_handler: %{
    enabled: true,
    local_ip: {127, 0, 0, 1},    # Endereço IP para escutar
    local_port: 2905,             # Porta para escutar
    point_code: 100               # Código de ponto deste STP
  }
```

Parâmetros de Configuração do STP

Parâmetro	Tipo	Padrão	Descrição	Exemplo
`enabled`	Booleano	`false`	Habilitar servidor STP autônomo	`true`
`local_ip`	Tupla	`{127, 0, 0, 1}`	Endereço IP para escutar por conexões	`{0, 0, 0, 0}`
`local_port`	Inteiro	`2905`	Porta para escutar	`2905`
`point_code`	Inteiro	*Obrigatório*	Código de ponto SS7 deste STP	`100`

Quando Usar STP Autônomo

- **Roteamento Puro**: Quando você só precisa de roteamento M3UA sem funcionalidade de cliente MAP
- **STP Central**: Para criar um roteador de sinalização central para múltiplos elementos de rede
- **Arquitetura Hub**: Conectar múltiplos HLRs, MSCs e SMSCs através de um STP central

Nota: Você pode habilitar tanto `map_client_m3ua` quanto `sctp_handler` simultaneamente se precisar de conexões de saída e funcionalidade STP de entrada.

Persistência da Tabela de Roteamento (Mnesia)

Todas as tabelas de roteamento (pares, rotas de Código de Ponto e rotas de Título Global) são armazenadas em um **banco de dados Mnesia** para

persistência.

Como Funciona o Roteamento

1. **Rotas de runtime.exs**: Rotas definidas em `config/runtime.exs` sob `peers` (ou legado `m3ua_peers`), `m3ua_routes`, e `m3ua_gt_routes` são carregadas na inicialização da aplicação
2. **Rotas da Interface Web**: Rotas adicionadas via a página de Roteamento da Interface Web são armazenadas no Mnesia
3. **Mesclagem de Rotas**: Na reinicialização, rotas de runtime.exs são mescladas com rotas existentes no Mnesia (sem duplicatas)
4. **Persistência**: Todas as rotas configuradas via Interface Web **sobrevivem a reinicializações da aplicação**

Tipo de Armazenamento Mnesia

Controle como as tabelas de roteamento são armazenadas. Para mais detalhes sobre a configuração do banco de dados, consulte Parâmetros de Banco de Dados na Referência de Configuração.

```
config :omniss7,
  mnesia_storage_type: :disc_copies  # ou :ram_copies para testes
```

Tipo de Armazenamento	Descrição	Persistência	Caso de Uso
`:disc_copies`	Armazenamento em disco (padrão)	**Sobrevive a reinicializações**	Ambientes de produção
`:ram_copies`	Somente em memória	Perdido na reinicialização	Testes, desenvolvimento

Padrão: `:disc_copies`

Localização do Banco de Dados Mnesia

O Mnesia armazena tabelas de roteamento no diretório Mnesia da aplicação:

- **Localização**: `Mnesia.{node_name}/` (ex: `Mnesia.nonode@nohost/`)
- **Tabelas**: `m3ua_peer`, `m3ua_route`, `m3ua_gt_route`

Gerenciando Rotas

Você tem três opções para gerenciar rotas:

1. **Runtime.exs** - Configuração estática carregada na inicialização
2. **Interface Web** - Gerenciamento de rotas interativo (veja Guia da Interface Web)
3. **API REST** - Gerenciamento programático de rotas (veja Guia da API)

Melhor Prática: Use `runtime.exs` para configuração base e a Interface Web para alterações dinâmicas de rotas durante a operação.

Configurando Pares M3UA

Pares representam pontos finais de conexão M3UA (outros STPs, HLRs, MSCs, SMSCs). Adicione pares a `config/runtime.exs`.

Exemplo de Configuração de Par

> **Nota**: A chave de configuração `peers` é o padrão atual. A chave legado `m3ua_peers` ainda é suportada para compatibilidade.

```
config :omniss7,
  peers: [
    # Conexão de saída para o STP Parceiro (função: :client)
    %{
      peer_id: 1,                           # Identificador único
      name: "Partner_STP_West",             # Nome descritivo
      role: :client,                        # :client para saída,
:server para entrada
      local_ip: {10, 0, 0, 1},             # IP local para
vincular
      local_port: 0,                        # 0 = atribuição de
porta dinâmica
      remote_ip: {10, 0, 0, 10},           # IP remoto do par
      remote_port: 2905,                    # Porta remota do par
      routing_context: 1,                   # Contexto de
roteamento M3UA
      point_code: 100,                      # Código de ponto
deste par
      network_indicator: :international     # :international ou
:national
    },

    # Conexão para HLR Local (função: :client)
    %{
      peer_id: 2,
      name: "Local_HLR",
      role: :client,
      local_ip: {10, 0, 0, 1},
      local_port: 0,
      remote_ip: {10, 0, 0, 20},
      remote_port: 2905,
      routing_context: 2,
      point_code: 200,
      network_indicator: :international
    },

    # Conexão de entrada do MSC Remoto (função: :server)
    # Para a função :server, o STP aguarda a conexão de entrada
    %{
      peer_id: 3,
      name: "Remote_MSC",
      role: :server,                        # Aceitar conexão de
entrada
```

```
      remote_ip: {10, 0, 0, 30},           # IP de origem esperado
      remote_port: 2905,                    # Porta de origem
esperada (0 = aceitar de qualquer porta)
      routing_context: 3,
      point_code: 300,
      network_indicator: :international
    },

    # Conexão de entrada com porta de origem dinâmica (sem
filtragem de porta)
    %{
      peer_id: 4,
      name: "Dynamic_Client",
      role: :server,
      remote_ip: {10, 0, 0, 40},           # IP de origem esperado
      remote_port: 0,                       # 0 = aceitar conexões
de qualquer porta de origem
      routing_context: 4,
      point_code: 400,
      network_indicator: :international
    }
  ]
```

Parâmetros de Configuração do Par

Parâmetro	Tipo	Obrigatório	Descrição
`peer_id`	Inteiro	Sim	Identificador numérico único para o par
`name`	String	Sim	Nome legível por humanos para logs e monitoramento
`role`	Átomo	Sim	`:client` (saída) ou `:server` (entrada)
`local_ip`	Tupla ou Lista	Sim (cliente)	Endereço(s) IP local(is) para vincular. Único: `{10, 0, 0, 1}` ou Múltiplos para multihoming SCTP: `[{10, 0, 0, 1}, {10, 0, 0, 2}]`
`local_port`	Inteiro	Sim (cliente)	Porta local (0 para dinâmica)
`remote_ip`	Tupla ou Lista	Sim	**Função cliente**: Tupla única ou lista para par remoto multihomed. **Função servidor**: Apenas tupla única - o IP do qual o par remoto se conecta (veja a nota abaixo)
`remote_port`	Inteiro	Sim	Porta do par remoto (0 para entrada = aceitar de qualquer porta de origem)
`routing_context`	Inteiro	Sim	Identificador do contexto de roteamento M3UA

Parâmetro	Tipo	Obrigatório	Descrição
`point_code`	Inteiro	Sim	Código de ponto SS7 deste par
`network_indicator`	Átomo	Não	`:international` ou `:national`

> **Multihoming SCTP para Função Servidor (Conexões de Entrada)**: Ao aceitar conexões de entrada de um par remoto multihomed, você só precisa especificar o **endereço IP único** que o par remoto usa para iniciar a conexão SCTP (SCTP INIT). Isso deve ser uma **tupla única**, não uma lista. O SCTP descobrirá automaticamente os outros endereços IP multihomed do par durante o handshake de associação. O formato de lista para `remote_ip` é usado apenas para **função cliente** ao conectar-se a um par remoto multihomed.
>
> **Multihoming SCTP para Função Cliente (Conexões de Saída)**: Para redundância de rede ao conectar-se a pares remotos, você pode configurar múltiplos endereços IP para `local_ip` e `remote_ip` usando listas. Isso habilita failover automático se um caminho de rede falhar. Consulte o Guia de Multihoming SCTP para exemplos de configuração detalhados e melhores práticas.

Filtragem de Porta de Origem para Conexões de Entrada

Para **conexões de entrada** (função: `:server`), o parâmetro `remote_port` controla a filtragem da porta de origem:

- **Porta Específica** (ex: `remote_port: 2905`): Aceitar apenas conexões daquela porta de origem exata
  - Fornece segurança adicional validando a porta de origem
  - Use quando o par remoto usar uma porta de origem fixa
- **Qualquer Porta** (`remote_port: 0`): Aceitar conexões de qualquer porta de origem

- Útil quando o par remoto usa portas de origem dinâmicas/epidêmicas
- Apenas valida o endereço IP de origem
- Mais flexível, mas ligeiramente menos seguro

Exemplo:

```
# Aceitar apenas de 10.5.198.200:2905 (porta específica)
%{
  peer_id: 1,
  name: "Strict_Peer",
  role: :server,
  remote_ip: {10, 5, 198, 200},
  remote_port: 2905,
  # ... outra configuração
}

# Aceitar de 10.5.198.200 com qualquer porta de origem
%{
  peer_id: 2,
  name: "Flexible_Peer",
  role: :server,
  remote_ip: {10, 5, 198, 200},
  remote_port: 0,  # Aceitar de qualquer porta de origem
  # ... outra configuração
}
```

Suporte ao Protocolo M2PA

OmniSS7 suporta tanto os protocolos **M3UA** quanto **M2PA** para transporte de sinalização SS7.

O que é M2PA?

M2PA (Camada de Adaptação Peer-to-Peer do Usuário MTP2) é um protocolo padronizado pelo IETF (RFC 4165) para transportar mensagens MTP3 SS7 sobre redes IP usando SCTP.

M3UA vs M2PA: Principais Diferenças

Recurso	M3UA	M2PA
Arquitetura	Cliente/Servidor (ASP/SGW)	Peer-to-Peer
Caso de Uso	Gateway entre SS7 e IP	Links diretos ponto a ponto
Gerenciamento de Estado de Link	Nível de aplicação (ASPUP/ASPAC)	Estilo MTP2 (Alinhamento, Prova, Pronto)
Números de Sequência	Sem sequenciamento inerente	BSN/FSN de 24 bits para entrega ordenada
Implantação Típica	Gateway SS7-para-IP, STP	Links de sinalização diretos entre nós
RFC	RFC 4666	RFC 4165

Orientação sobre Seleção de Protocolo

Recomendação: Use M3UA por padrão. Use M2PA apenas quando especificamente necessário.

Quando Usar M3UA (Recomendado)

M3UA é o protocolo recomendado para a maioria das implantações:

- **Implantações STP**: Implementações padrão de ponto de transferência de sinalização
- **Funções de Gateway**: Conectando redes SS7 com sinalização baseada em IP

- **Conexões de Elementos de Rede**: Conectando HLRs, MSCs, SMSCs e outros elementos de rede ao seu STP
- **Gateway de Sinalização (SGW)**: Gateway central aceitando conexões de múltiplos Servidores de Aplicação
- **Topologias Flexíveis**: Arquiteturas cliente/servidor com controle centralizado
- **Redes de Múltiplos Fornecedores**: Padrão da indústria amplamente suportado (RFC 4666)

Use M3UA para conectar elementos de rede (HLR, MSC, SMSC, VLR, etc.) ao seu STP.

Quando Usar M2PA (Apenas Casos Especiais)

M2PA deve ser usado apenas em cenários específicos:

- **Links STP-para-STP**: Conexões diretas ponto a ponto entre Pontos de Transferência de Sinalização em uma rede multi-STP
- **Substituição de TDM Legado**: Substituindo links tradicionais SS7 TDM quando o sistema remoto requer especificamente M2PA
- **Compatibilidade MTP2 Necessária**: Ao conectar-se a sistemas legados que exigem gerenciamento de estado de link estilo MTP2
- **Requisito de Parceiro**: Quando um parceiro ou interconexão exige especificamente o protocolo M2PA

Importante: Não use M2PA para conectar elementos de rede (HLR, MSC, SMSC) ao seu STP - use M3UA em vez disso. M2PA é projetado para interconexões STP-para-STP onde ambos os lados operam como nós de roteamento.

Configurando Pares M2PA

Pares M2PA são configurados da mesma forma que pares M3UA, com um parâmetro adicional `protocol`.

Configuração de Par M2PA

Adicione pares M2PA à sua configuração `peers` em `config/runtime.exs` (tanto pares M3UA quanto M2PA compartilham a mesma seção de configuração, diferenciados pelo parâmetro `protocol`):

Parâmetros Chave para M2PA:

Parâmetro	Valor	Descrição
`protocol`	`:m2pa`	Especifica o protocolo M2PA (padrão é `:m3ua` se omitido)
`role`	`:client` ou `:server`	Direção da conexão
`local_port`	Inteiro	Porta SCTP local (a porta padrão do M2PA é **3565**)
`remote_port`	Inteiro	Porta SCTP remota (a porta padrão do M2PA é **3565**)
`point_code`	Inteiro	Seu código de ponto
`adjacent_point_code`	Inteiro	Código de ponto do par remoto (específico do M2PA)
`send_sltm`	Booleano	Controla o comportamento SLTM (veja Configuração do SLTM abaixo)
`network_indicator`	Átomo	`:international` ou `:national` - deve corresponder ao par remoto

Nota: M2PA usa **porta 3565** como padrão da indústria (diferente da porta 2905 do M3UA).

Configuração do SLTM

SLTM (Mensagem de Teste de Link de Sinalização) e **SLTA (Reconhecimento de Mensagem de Teste de Link de Sinalização)** são mensagens de manutenção MTP3 usadas para verificar a conectividade de ponta a ponta após um link M2PA atingir o estado PRONTO. De acordo com a ITU-T Q.707, um lado envia SLTM e o outro responde com SLTA para confirmar que o link está operacional.

Comportamento do SLTM

O parâmetro `send_sltm` controla qual lado inicia o teste SLTM:

Valor de `send_sltm`	Comportamento
`true`	Enviamos SLTM quando o link se torna PRONTO, aguardamos SLTA
`false`	Aguardamos o par enviar SLTM, respondemos com SLTA
Não definido (padrão)	Segue Q.707: o respondente SCTP envia SLTM, o iniciador SCTP aguarda

Comportamento Padrão (Conforme Q.707):

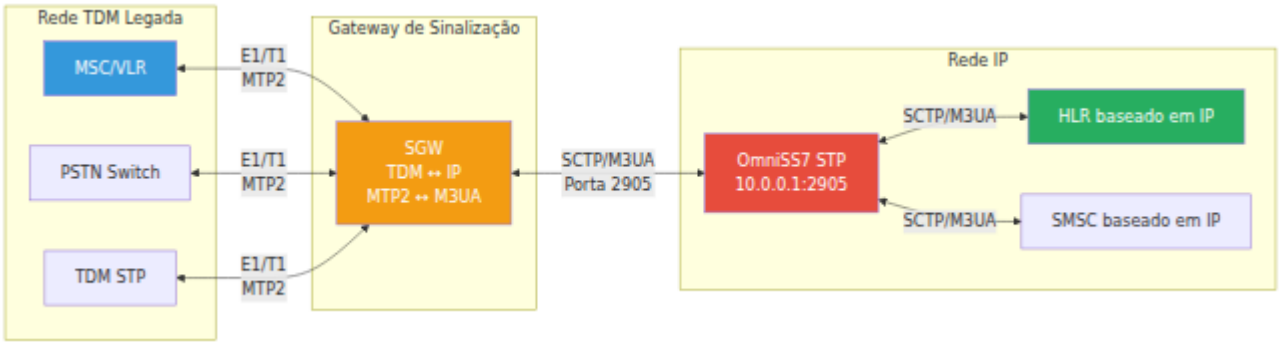
- Se `initiate_connection: true` (iniciador SCTP/client) → Aguardamos o par enviar SLTM
- Se `initiate_connection: false` (respondente SCTP/servidor) → Enviamos SLTM

Quando Substituir os Padrões do SLTM

Substitua o comportamento padrão ao conectar-se a equipamentos que não seguem as convenções Q.707:

```
# Exemplo: Forçar nosso lado a enviar SLTM independentemente do
papel SCTP
%{
  peer_id: 100,
  name: "Partner_STP",
  protocol: :m2pa,
  role: :client,
  local_ip: {10, 0, 0, 1},
  local_port: 3565,
  remote_ip: {10, 0, 0, 2},
  remote_port: 3565,
  point_code: 7415,
  adjacent_point_code: 15528,
  network_indicator: :international,
  send_sltm: true  # Substituir: Enviamos SLTM mesmo sendo o
iniciador SCTP
}
```

Fluxo de Mensagem SLTM



Resolvendo Problemas de SLTM

Sintoma: O link atinge PRONTO, mas nenhum tráfego flui

Ambos os lados podem estar aguardando que o outro envie SLTM. Verifique os logs para:

```
"aguardando o par enviar SLTM"
```

Solução: Configure `send_sltm: true` em um lado para quebrar o impasse.

Sintoma: SLTM enviado, mas SLTA não recebido

Causas possíveis:

1. `adjacent_point_code` está incorreto (SLTM enviado com DPC errado)
2. Desajuste de indicador de rede (`:international` vs `:national`)
3. Par remoto não configurado para responder ao nosso código de ponto

Solução: Verifique se `adjacent_point_code` corresponde ao código de ponto do STP (configurado em `sctp_handler.point_code`).

Estados de Link M2PA

Os links M2PA progridem através de vários estados durante a inicialização:

1. **Desligado** - Nenhuma conexão estabelecida
2. **Alinhamento** - Fase de sincronização inicial (~1 segundo)
3. **Prova** - Verificação da qualidade do link (~2 segundos)
4. **Pronto** - Link ativo e pronto para tráfego

A progressão do estado do link garante sinalização confiável antes que o tráfego seja trocado.

Gerenciando Pares M2PA via Interface Web

A página **Roteamento** na Interface Web fornece suporte total para gerenciar pares M2PA:

1. **Navegue** até a página de Roteamento
2. **Selecione** a aba "Pares"
3. **Clique** em "Adicionar Novo Par"
4. **Escolha** "M2PA (RFC 4165)" no menu suspenso de Protocolo
5. **Preencha** a configuração do par:
   - Nome do Par (identificador descritivo)
   - Protocolo: M2PA
   - Função: cliente ou servidor
   - Código de Ponto (seu PC)

    - Endereços IP local/remoto
    - Portas local/remota (tipicamente 3565 para M2PA)
    - Indicador de Rede (internacional ou nacional)
6. **Clique** em "Salvar Par"

A tabela de pares exibe o tipo de protocolo com codificação de cores:

- **Azul** - pares M3UA
- **Verde** - pares M2PA

Comportamento de Roteamento M2PA

Pares M2PA se integram perfeitamente ao sistema de roteamento do OmniSS7:

- **Rotas de Código de Ponto**: Funcionam de forma idêntica para M2PA e M3UA
- **Rotas de Título Global**: Totalmente suportadas em links M2PA
- **Prioridade de Rota**: Pares M2PA e M3UA podem ser misturados nas mesmas tabelas de roteamento
- **Revezamento de Mensagens**: Mensagens podem chegar em M2PA e ser roteadas para M3UA, e vice-versa

Métricas M2PA

M2PA fornece métricas abrangentes do Prometheus para monitorar a saúde e o tráfego do link:

Métricas de Tráfego:

- `m2pa_messages_sent_total` - Total de mensagens MTP3 enviadas por link
- `m2pa_messages_received_total` - Total de mensagens MTP3 recebidas por link
- `m2pa_bytes_sent_total` - Total de bytes enviados sobre M2PA
- `m2pa_bytes_received_total` - Total de bytes recebidos sobre M2PA

Todas as métricas de tráfego são rotuladas por: `link_name`, `point_code`, `adjacent_pc`

Métricas de Estado do Link:

- `m2pa_link_state_changes_total` - Transições de estado do link (DOWN → ALIGNMENT → PROVING → READY)
  - Rótulos: `link_name`, `from_state`, `to_state`

Métricas de Erro:

- `m2pa_errors_total` - Total de erros por tipo
  - `decode_error` - Falhas de decodificação de mensagens M2PA
  - `encode_error` - Falhas de codificação de mensagens M2PA
  - `sctp_send_error` - Falhas de transmissão SCTP
  - Rótulos: `link_name`, `error_type`

Acessando Métricas:

- Endpoint do Prometheus: `http://seu-servidor:8080/metrics`
- Métricas se registram automaticamente na inicialização da aplicação

Melhores Práticas M2PA

1. **Seleção de Porta**: Use a porta 3565 para M2PA (padrão da indústria)
2. **Monitoramento de Link**: Monitore mudanças de estado do link via métricas
3. **Regras de Firewall**: Garanta que SCTP (protocolo IP 132) seja permitido
4. **Códigos de Ponto**: Verifique se os códigos de ponto adjacentes estão configurados corretamente em ambos os lados
5. **Indicador de Rede**: Deve corresponder entre pares (internacional ou nacional)
6. **Testes**: Use a página de Teste de Roteamento para verificar conectividade após a configuração

Requisitos de Socket M2PA

M2PA usa sockets compartilhados do SCTP.SocketHandler. Todos os pares M2PA usam automaticamente o SCTP.SocketHandler para gerenciamento

de sockets, o que permite que múltiplos pares compartilhem eficientemente a mesma porta SCTP.

Requisitos:

- SCTP.SocketHandler deve estar habilitado e em execução
- O socket compartilhado deve ser configurado antes que os pares M2PA sejam iniciados

Exemplo:

```
# Habilitar SCTP.SocketHandler
sctp_handler: %{
  enabled: true,
  local_ip: {10, 179, 4, 10},
  local_port: 3565,
  point_code: 100
}

# Pares M2PA usam automaticamente o socket compartilhado
peers: [
  %{
    peer_id: 1,
    name: "M2PA_Link_1",
    protocol: :m2pa,
    role: :client,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,
    remote_ip: {10, 179, 4, 20},
    remote_port: 3565,
    point_code: 100,
    adjacent_point_code: 200
  },
  %{
    peer_id: 2,
    name: "M2PA_Link_2",
    protocol: :m2pa,
    role: :client,
    local_ip: {10, 179, 4, 10},
    local_port: 3565,           # Mesma porta compartilhada entre
pares
    remote_ip: {10, 179, 4, 30},
    remote_port: 3565,
    point_code: 100,
    adjacent_point_code: 300
  }
]
```

Requisitos SCTP:

- Entrega ordenada (sem a flag `unordered`)
- PPID 5 (identificador M2PA conforme RFC 4165)

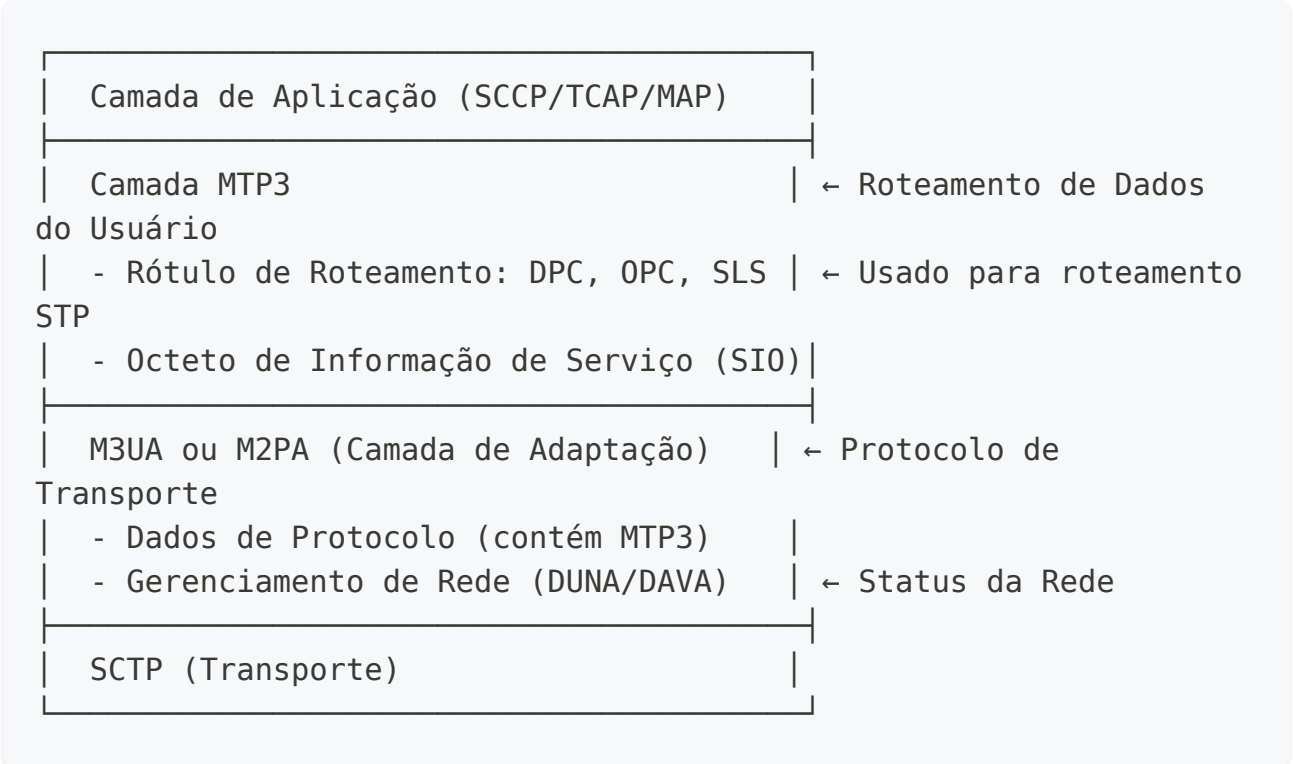
Configurando o Roteamento de Código de Ponto

O roteamento de Código de Ponto direciona mensagens com base no **Código de Ponto de Destino (DPC)** no cabeçalho MTP3.

Compreendendo Códigos de Ponto na Pilha de Protocolos SS7

Os códigos de ponto existem em diferentes camadas da pilha de protocolos SS7. Compreender essa distinção é importante:

Camadas da Pilha de Protocolos:



Dois Tipos de Códigos de Ponto:

1. **Códigos de Ponto da Camada MTP3** (Usados para Roteamento):
    - Localizados no rótulo de roteamento MTP3 (DPC, OPC)
    - Presentes no parâmetro de Dados de Protocolo M3UA (tag 528)
    - Presentes nas mensagens de Dados do Usuário M2PA

    - **O STP usa esses valores DPC para decisões de roteamento**
    - Esses determinam onde a mensagem é finalmente entregue

2. **Códigos de Ponto da Camada M3UA** (Usados para Gerenciamento de Rede):

    - Presentes em mensagens de gerenciamento M3UA (DUNA, DAVA, SCON, DUPU)
    - Indicam códigos de ponto afetados para status de rede
    - Informam os pares quais destinos estão disponíveis/in disponíveis
    - Não são usados para roteamento de dados do usuário

Como Funciona o Roteamento STP:

- **Para mensagens M3UA DATA**: O STP extrai a mensagem MTP3 do parâmetro de Dados de Protocolo (tag 528), que contém o rótulo de roteamento MTP3 (DPC, OPC, SLS). O DPC da camada MTP3 é usado para procurar rotas.
- **Para mensagens de Dados do Usuário M2PA**: O STP extrai a mensagem MTP3 do campo de dados do usuário M2PA, em seguida, lê o DPC do rótulo de roteamento MTP3.
- **Mensagens de gerenciamento M3UA**: Mensagens de gerenciamento de rede (DUNA, DAVA, SCON) contêm códigos de ponto afetados na camada M3UA para sinalizar status de rede entre pares.

Rotas Básicas de Código de Ponto

Adicione rotas a `config/runtime.exs`:

```
config :omniss7,
  m3ua_routes: [
    # Roteie todo o tráfego para PC 100 para o par 1 (STP
Parceiro)
    %{
      dest_pc: 100,                    # Código de ponto de
destino
      peer_id: 1,                      # Par para roteamento
      priority: 1,                     # Prioridade (menor = maior
prioridade)
      network_indicator: :international
      # mask: 14                       # Opcional: padrão é 14
(correspondência exata)
    },

    # Roteie todo o tráfego para PC 200 para o par 2 (HLR Local)
    %{
      dest_pc: 200,
      peer_id: 2,
      priority: 1,
      network_indicator: :international
    },

    # Exemplo de balanceamento de carga: PC 300 com rotas primária
e de backup
    %{
      dest_pc: 300,
      peer_id: 3,                      # Rota primária
      priority: 1,
      network_indicator: :international
    },
    %{
      dest_pc: 300,
      peer_id: 4,                      # Rota de backup (número de
prioridade mais alto)
      priority: 2,
      network_indicator: :international
    }
  ]
```

Nota: O campo `mask` é opcional e padrão para `14` (correspondência exata). Especifique `mask` apenas quando precisar de roteamento baseado em intervalo

(veja a seção sobre Máscaras de Código de Ponto abaixo).

Lógica de Roteamento

1. O STP recebe uma mensagem M3UA DATA ou M2PA Dados do Usuário
2. O STP extrai a **mensagem MTP3** do Dados de Protocolo (M3UA) ou do campo de Dados do Usuário (M2PA)
3. O STP lê o **Código de Ponto de Destino (DPC)** do rótulo de roteamento MTP3
4. Procura na tabela de roteamento por DPC correspondente (considerando máscaras)
5. Se várias rotas existirem, seleciona a rota com **máscara mais específica** (valor de máscara mais alto), então **menor número de prioridade**
6. Envolve a mensagem MTP3 em M3UA DATA ou Dados do Usuário M2PA para o par de destino
7. Roteia a mensagem para o par correspondente
8. Se o par selecionado estiver inativo, tenta a próxima rota de prioridade mais alta

Máscaras de Código de Ponto

Os códigos de ponto são valores de 14 bits (faixa de 0-16383). Por padrão, as rotas correspondem a um único código de ponto exatamente (máscara `/14`). No entanto, você pode usar **máscaras de código de ponto** para criar rotas que correspondem a **intervalos** de códigos de ponto.

Compreendendo Máscaras

A máscara especifica quantos **bits mais significativos** devem corresponder entre o PC de destino da rota e o DPC da mensagem de entrada. Os bits restantes podem ser qualquer valor, criando um intervalo de códigos de ponto correspondentes.

Tabela de Referência de Máscaras:

Máscara	Códigos de Ponto Correspondidos	Caso de Uso
`/14`	1 PC (correspondência exata)	Destino único (padrão)
`/13`	2 PCs	Pequeno intervalo
`/12`	4 PCs	Pequeno intervalo
`/11`	8 PCs	Pequeno intervalo
`/10`	16 PCs	Intervalo médio
`/9`	32 PCs	Intervalo médio
`/8`	64 PCs	Intervalo médio
`/7`	128 PCs	Intervalo médio-grande
`/6`	256 PCs	Grande intervalo
`/5`	512 PCs	Grande intervalo
`/4`	1.024 PCs	Muito grande intervalo
`/3`	2.048 PCs	Muito grande intervalo
`/2`	4.096 PCs	Extremamente grande intervalo
`/1`	8.192 PCs	Metade de todos os PCs
`/0`	16.384 PCs	Todos os PCs (padrão/rota de fallback)

Exemplos de Máscaras de Código de Ponto

Nota: O campo `mask` é **opcional** em todos os exemplos. Se omitido, padrão para `14` (correspondência exata).

Exemplo 1: Código de Ponto Único (Comportamento Padrão)

```
# Sem campo de máscara (recomendado para PC único)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  network_indicator: :international
}
# Máscara padrão para 14 - Corresponde: Apenas PC 1000

# Máscara explícita (mesmo resultado)
%{
  dest_pc: 1000,
  peer_id: 1,
  priority: 1,
  mask: 14,                              # Correspondência exata
explícita
  network_indicator: :international
}
# Corresponde: Apenas PC 1000
```

Exemplo 2: Pequeno Intervalo

```
%{
  dest_pc: 1000,
  peer_id: 2,
  priority: 1,
  mask: 12,                              # Corresponde a 4 PCs
  network_indicator: :international
}
# Corresponde: PC 1000, 1001, 1002, 1003
```

Exemplo 3: Intervalo Médio

```
%{
  dest_pc: 1000,
  peer_id: 3,
  priority: 1,
  mask: 8,                                # Corresponde a 64 PCs
  network_indicator: :international
}
# Corresponde: PC 1000-1063 (64 códigos de ponto consecutivos)
```

Exemplo 4: Rota Padrão/Fallback

```
%{
  dest_pc: 0,
  peer_id: 4,
  priority: 10,                           # Baixa prioridade
(número alto)
  mask: 0,                                # Corresponde a todos os
PCs
  network_indicator: :international
}
# Corresponde: Todos os códigos de ponto (0-16383)
# Use como uma rota catch-all/default com baixa prioridade
```

Combinando Rotas Específicas e Mascaradas

Você pode combinar rotas específicas com rotas mascaradas para um roteamento flexível:

```
config :omniss7,
  m3ua_routes: [
    # Rota específica para PC 1000 (tem prioridade)
    %{
      dest_pc: 1000,
      peer_id: 1,
      priority: 1,
      network_indicator: :international
      # máscara padrão para 14 (correspondência exata)
    },

    # Rota de intervalo para PCs 1000-1063
    %{
      dest_pc: 1000,
      peer_id: 2,
      priority: 1,
      mask: 8,                          # Corresponde a 64 PCs
      network_indicator: :international
    },

    # Rota padrão/fallback para todos os outros PCs
    %{
      dest_pc: 0,
      peer_id: 3,
      priority: 10,                     # Baixa prioridade
      mask: 0,                          # Corresponde a todos os
PCs
      network_indicator: :international
    }
  ]
```

Decisão de Roteamento para DPC 1000:

1. Corresponde à rota de máscara `/14` (PC 1000 exatamente) - **Selecionada** (mais específica)
2. Também corresponde à rota de máscara `/8` (PC 1000-1063) - Ignorada (menos específica)
3. Também corresponde à rota de máscara `/0` (todos os PCs) - Ignorada (menos específica)

Decisão de Roteamento para DPC 1015:

1. Não corresponde à rota de máscara `/14` (PC 1000 apenas)
2. Corresponde à rota de máscara `/8` (PC 1000-1063) - **Selecionada** (mais específica)
3. Também corresponde à rota de máscara `/0` (todos os PCs) - Ignorada (menos específica)

Decisão de Roteamento para DPC 5000:

1. Não corresponde à rota de máscara `/14`
2. Não corresponde à rota de máscara `/8`
3. Corresponde à rota de máscara `/0` (todos os PCs) - **Selecionada** (única correspondência, rota fallback)

Melhores Práticas

1. **Omitir `mask` para Destinações Únicas**: Para correspondências exatas de código de ponto, omita o campo `mask` completamente (padrão para `/14`)
2. **Use `/14` Explicitamente Apenas Quando Necessário**: Especifique `mask: 14` apenas quando precisar deixar claro na documentação ou ao misturar com rotas de intervalo
3. **Use Máscaras de Intervalo para Blocos de Rede**: Roteie segmentos inteiros da rede para pares específicos com máscaras `/0` a `/13`
4. **Use `/0` como Fallback**: Crie uma rota padrão com baixa prioridade para capturar tráfego não correspondido
5. **Mais Específico Vence**: O mecanismo de roteamento sempre seleciona a rota mais específica (valor de máscara mais alto) primeiro
6. **Prioridade como Critério de Desempate**: Se várias rotas tiverem a mesma máscara, o menor número de prioridade vence

---

Configurando o Roteamento de Título Global (GT)

O roteamento de Título Global permite **roteamento baseado em conteúdo** usando números de telefone ou valores IMSI em vez de códigos de ponto. Para

tradução avançada de endereço de Título Global com base em parte chamadora/parte chamada, consulte o Guia de NAT de Título Global.

Pré-requisitos

- Habilitar roteamento GT: `enable_gt_routing: true` em `config/runtime.exs`

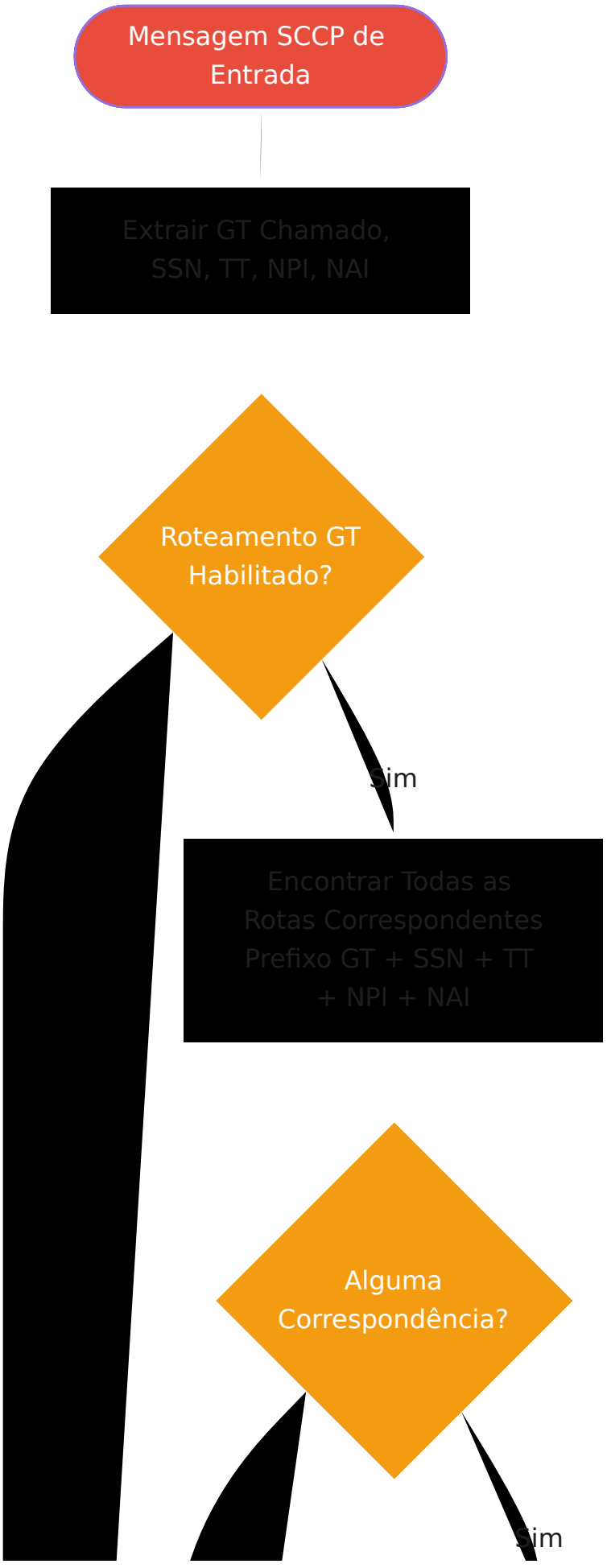
Configuração de Rotas GT

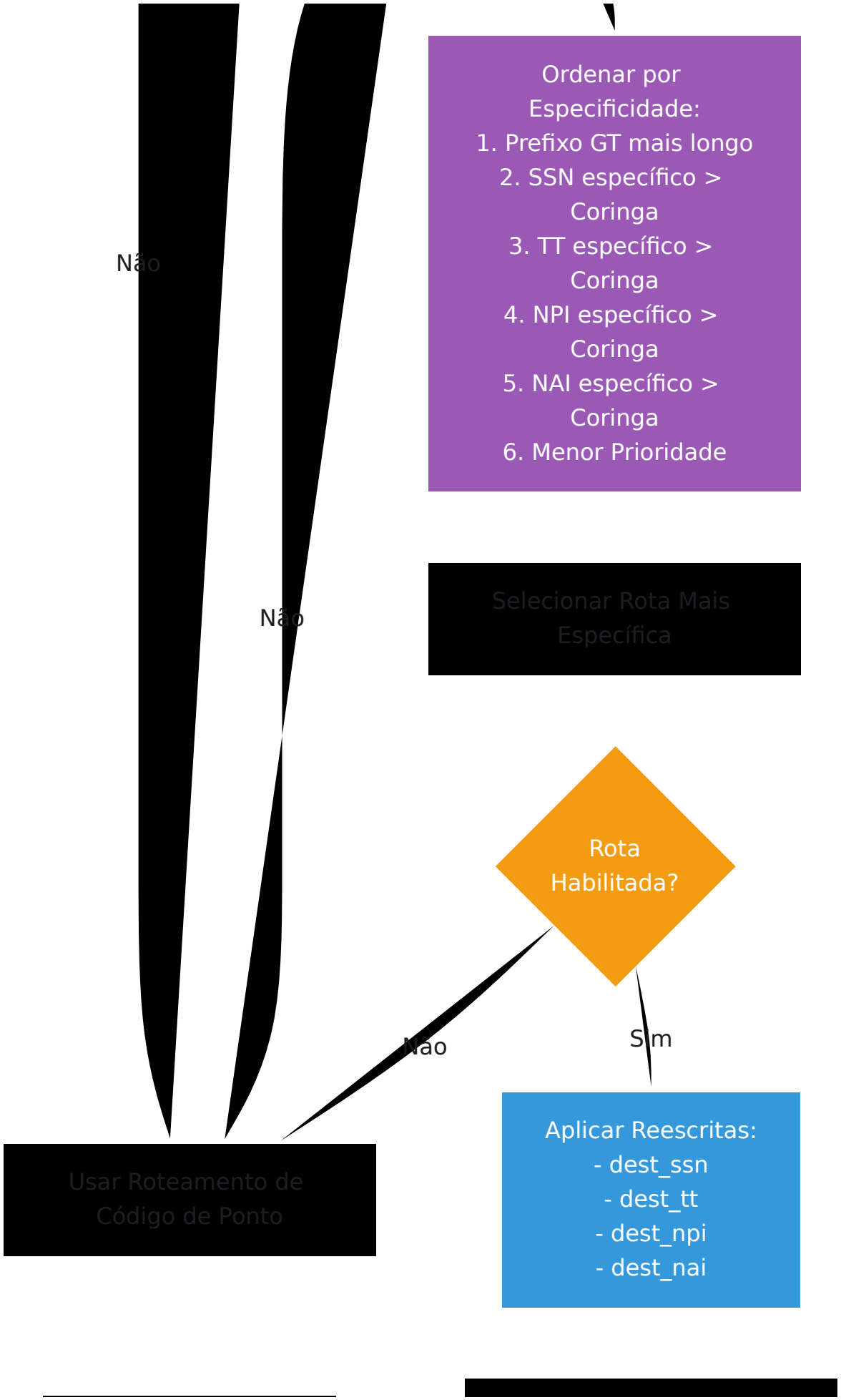
```
config :omniss7,
  # Habilitar roteamento GT
  enable_gt_routing: true,

  m3ua_gt_routes: [
    # Roteie todos os números do Reino Unido (prefixo 44) para o
par 1
    %{
      gt_prefix: "44",                # Prefixo de Título Global
a ser correspondido
      peer_id: 1,                     # Par de destino
      priority: 1,                    # Prioridade (menor =
maior)
      description: "Números do Reino Unido"  # Descrição para
registro
    },

    # Roteie números dos EUA (prefixo 1) para o par 2
    %{
      gt_prefix: "1",
      peer_id: 2,
      priority: 1,
      description: "Números dos EUA"
    },

    # Rota mais específica: números móveis do Reino Unido
começando com 447
    %{
      gt_prefix: "447",               # A correspondência de
prefixo mais longa vence
      peer_id: 3,
      priority: 1,
      description: "Números móveis do Reino Unido"
    },

    # Roteamento específico de SSN (opcional)
    %{
      gt_prefix: "555",
      source_ssn: 8,                  # Correspondência apenas se
SSN de origem = 8 (SMSC)
      peer_id: 4,
```

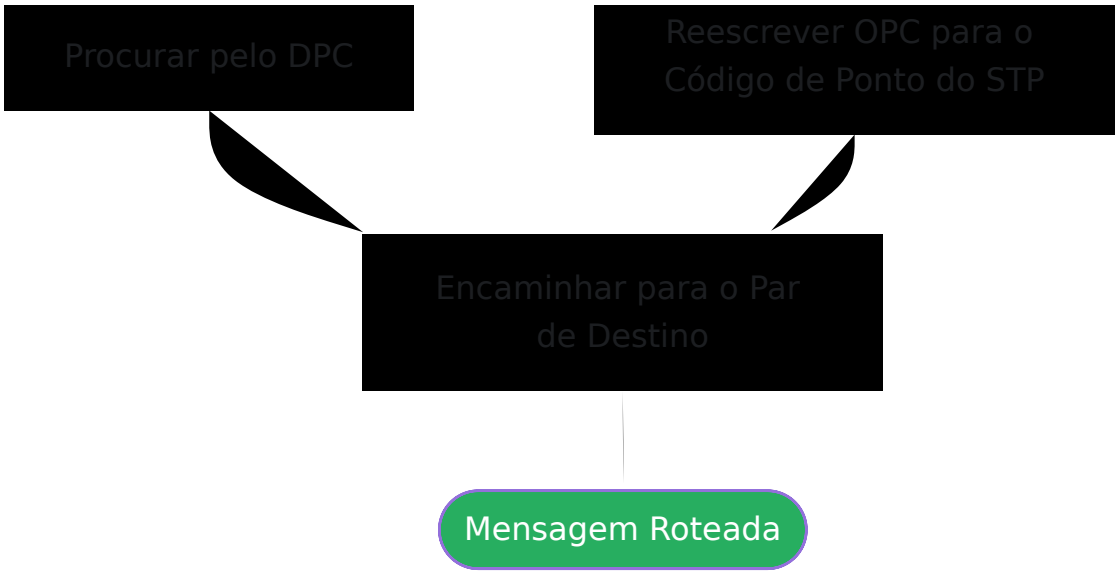
```
      dest_ssn: 6,                    # Reescrever SSN de destino
para 6 (HLR)
      priority: 1,
      description: "Tráfego SMS para prefixo 61"
    }
  ]
```

Lógica de Roteamento GT

O algoritmo de roteamento GT segue este processo de decisão:


Mensagem SCCP de Entrada
Extrair GT Chamado, SSN, TT, NPI, NAI
Roteamento GT Habilitado?
Sim
Encontrar Todas as Rotas Correspondentes Prefixo GT + SSN + TT + NPI + NAI
Alguma Correspondência?
Sim


Não
Não
Ordenar por Especificidade:
1. Prefixo GT mais longo
2. SSN específico > Coringa
3. TT específico > Coringa
4. NPI específico > Coringa
5. NAI específico > Coringa
6. Menor Prioridade
Selecionar Rota Mais Específica
Rota Habilitada?
Não
Sim
Aplicar Reescritas:
- dest_ssn
- dest_tt
- dest_npi
- dest_nai
Usar Roteamento de Código de Ponto



Passos de Roteamento:

1. **Correspondência de Prefixo Mais Longo**: O STP encontra todas as rotas GT onde o prefixo corresponde ao início do Título Global
    - Exemplo: GT "447712345678" corresponde a "44" e "447", mas "447" vence (correspondência mais longa)
2. **Correspondência de SSN** (Opcional):
    - Se `source_ssn` for especificado, a rota só corresponde quando o SSN da Parte Chamadora SCCP iguala esse valor
    - Se `source_ssn` for `nil`, a rota corresponde a qualquer SSN (coringa)
3. **Correspondência de TT/NPI/NAI** (Opcional):
    - Se `source_tt`, `source_npi` ou `source_nai` forem especificados, as rotas devem corresponder a esses indicadores
    - Valores `nil` atuam como coringas (correspondem a qualquer valor)
4. **Seleção Baseada em Especificidade**:
    - Rotas com critérios de correspondência mais específicos vencem sobre coringas
    - Ordem de prioridade: Comprimento do Prefixo GT → SSN → TT → NPI → NAI → Número de Prioridade
5. **Reescrita de Indicadores** (Opcional):

- Se `dest_ssn`, `dest_tt`, `dest_npi` ou `dest_nai` forem especificados, o STP reescreve esses indicadores
- Útil para normalização de protocolo e interconexão de rede

6. **Fallback para Código de Ponto**:
   - Se nenhuma rota GT corresponder, o STP recai no roteamento de Código de Ponto usando o DPC

Exemplos de Roteamento GT

Para uma mensagem de entrada com:

- GT: "447712345678"
- SSN: 8
- TT: 0
- NPI: 1
- NAI: 4

Com estas rotas configuradas:

```
# Rota A: Corresponde e transforma Tipo de Tradução
%{gt_prefix: "447", peer_id: 1, source_tt: 0, dest_tt: 3,
priority: 1, description: "Números do Reino Unido: TT 0→3
transformação"}

# Rota B: Corresponde TT específico
%{gt_prefix: "447", peer_id: 2, source_tt: 0, priority: 1,
description: "Números do Reino Unido: TT específico"}

# Rota C: Corresponde TT específico + NPI
%{gt_prefix: "447", source_tt: 0, source_npi: 1, peer_id: 3,
priority: 1, description: "Números do Reino Unido: TT específico +
NPI"}
```

Resultado: A Rota C é selecionada (mais específica: corresponde a GT + TT + NPI)

A mensagem é encaminhada com indicadores transformados conforme os valores de `dest_tt`, `dest_npi`, `dest_nai` da Rota C.

Roteamento Avançado: Tipo de Tradução, NPI e NAI

Além da correspondência de prefixo GT e SSN, o STP suporta roteamento e transformação com base nos indicadores de Título Global SCCP:

- **Tipo de Tradução (TT)**: Identifica o plano de numeração e tipo de endereço
- **Indicador de Plano de Numeração (NPI)**: Define o plano de numeração (ex: ISDN, Dados, Telex)
- **Indicador de Natureza de Endereço (NAI)**: Especifica o formato do endereço (ex: Internacional, Nacional, Assinante)

Correspondência (Indicadores de Origem)

As rotas podem corresponder aos indicadores das mensagens de entrada:

- `source_tt`: Correspondência de mensagens com Tipo de Tradução específico
- `source_npi`: Correspondência de mensagens com Indicador de Plano de Numeração específico
- `source_nai`: Correspondência de mensagens com Indicador de Natureza de Endereço específico

- Valor `nil` = coringa (corresponde a qualquer valor)

Transformação (Indicadores de Destino)

As rotas podem reescrever indicadores ao encaminhar:

- `dest_tt`: Transformar Tipo de Tradução para novo valor
- `dest_npi`: Transformar Indicador de Plano de Numeração para novo valor
- `dest_nai`: Transformar Indicador de Natureza de Endereço para novo valor
- Valor `nil` = preservar valor original (sem transformação)

Seleção Baseada em Especificidade

Quando várias rotas correspondem, a rota mais específica é selecionada usando esta ordem de prioridade:

1. Correspondência de prefixo GT mais longo
2. SSN específico sobre SSN coringa
3. TT específico sobre TT coringa
4. NPI específico sobre NPI coringa
5. NAI específico sobre NAI coringa
6. Menor número de prioridade

Exemplos de Configuração

```
config :omniss7,
  enable_gt_routing: true,

  m3ua_gt_routes: [
    # Exemplo 1: Correspondência e transformação do Tipo de
Tradução
    %{
      gt_prefix: "44",
      peer_id: 1,
      source_tt: 0,      # Correspondência TT=0 (Desconhecido)
      dest_tt: 3,        # Transformar para TT=3 (Nacional)
      priority: 1,
      description: "Números do Reino Unido: TT 0→3 transformação"
    },

    # Exemplo 2: Correspondência de NPI específico e transformação
de NAI
    %{
      gt_prefix: "1",
      peer_id: 2,
      source_npi: 1,     # Correspondência NPI=1 (ISDN/Telefônico)
      source_nai: 4,     # Correspondência NAI=4 (Internacional)
      dest_nai: 3,       # Transformar para NAI=3 (Nacional)
      priority: 1,
      description: "Números dos EUA: Internacional→Nacional NAI"
    },

    # Exemplo 3: Roteamento combinado de SSN e indicadores
    %{
      gt_prefix: "33",
      source_ssn: 8,     # Correspondência de tráfego SMS
      source_tt: 0,      # Correspondência TT=0
      dest_ssn: 6,       # Reescrever SSN para HLR
      dest_tt: 2,        # Transformar para TT=2
      dest_npi: 1,       # Definir NPI=1 (ISDN)
      dest_nai: 4,       # Definir NAI=4 (Internacional)
      peer_id: 3,
      priority: 1,
      description: "SMS francês: Normalização completa"
    },

    # Exemplo 4: TT coringa, NPI específico
    %{
```

```
      gt_prefix: "49",
      source_tt: nil,    # Correspondência de qualquer TT
(coringa)
      source_npi: 6,     # Correspondência NPI=6 (Dados)
      dest_npi: 1,       # Transformar para NPI=1 (ISDN)
      peer_id: 4,
      priority: 1,
      description: "Normalização da rede de dados alemã"
    }
  ]
```

Valores Comuns de TT/NPI/NAI

Tipo de Tradução (TT):

- `0` = Desconhecido
- `1` = Internacional
- `2` = Nacional
- `3` = Específico da Rede

Indicador de Plano de Numeração (NPI):

- `0` = Desconhecido
- `1` = ISDN/Telefônico (E.164)
- `3` = Dados (X.121)
- `4` = Telex (F.69)
- `6` = Móvel Terrestre (E.212)

Indicador de Natureza de Endereço (NAI):

- `0` = Desconhecido
- `1` = Número de Assinante
- `2` = Reservado para Uso Nacional
- `3` = Número Significativo

Guia do Gateway USSD

← Voltar para a Documentação Principal

Este guia cobre o **Gateway USSD** do OmniSS7, que conecta diálogos USSD SS7/MAP a callbacks HTTP/JSON, permitindo que desenvolvedores de terceiros construam aplicações USSD com um simples endpoint HTTP.

Índice

Visão Geral

O Gateway USSD lida com duas direções de tráfego USSD:

- **Originado pelo Móvel (Entrada)** — Um assinante disca um código curto (por exemplo, `*100#`). O gateway recebe o `processUnstructuredSS-Request` MAP (opcode 59), encaminha para seu callback HTTP e retransmite sua resposta de volta via SS7.
- **Originado pela Rede (Saída)** — Sua aplicação envia uma mensagem USSD para um assinante via a API REST. O gateway envia um

`unstructuredSS-Request` MAP (opcode 60) via SS7 e roteia a resposta do assinante para seu callback.

Ambas as direções suportam **diálogos de múltiplas etapas** — menus interativos onde o assinante responde e recebe prompts de acompanhamento.

Características Principais

Propriedade	Valor
Transporte	HTTP POST síncrono por turno
Codificação	Alfabeto padrão GSM de 7 bits (DCS 0x0F) conforme 3GPP TS 23.038
Comprimento máximo do texto	182 caracteres (configurável)
Rastreamento de sessão	UUID gerado pelo gateway por diálogo
Autenticação	Nenhuma (confia na rede SS7)
Roteamento	Correspondência de prefixo de código curto aos URLs de callback

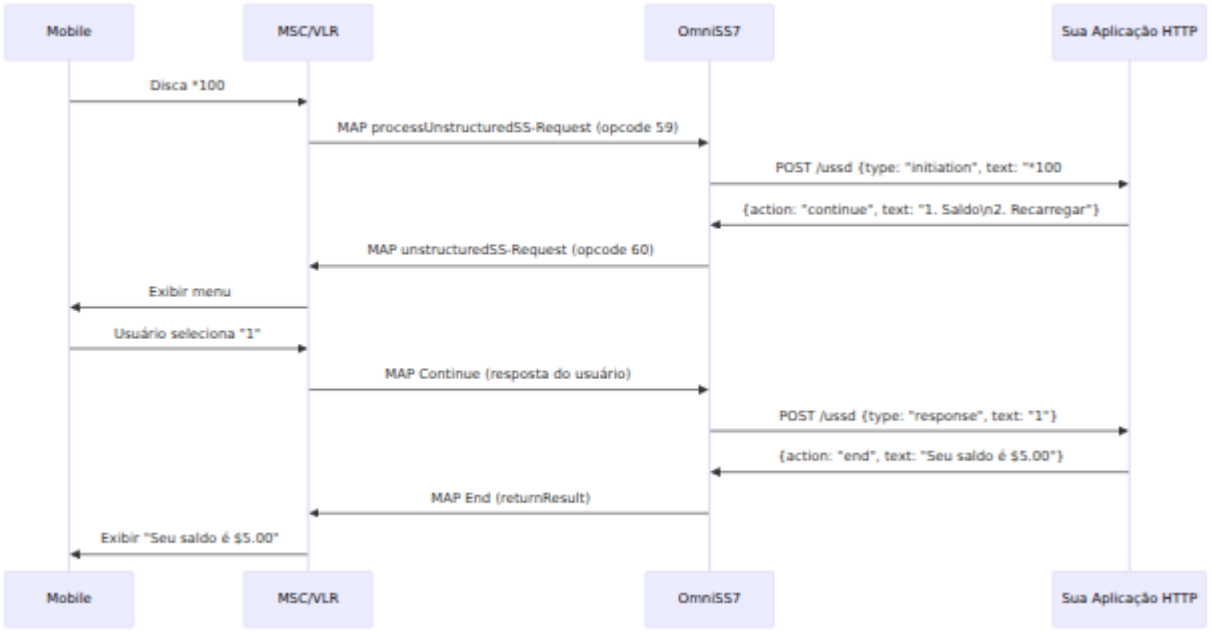
Referências 3GPP

Especificação	Relevância
3GPP TS 23.090	USSD Fase 2 — arquitetura e procedimentos
3GPP TS 24.090	USSD Fase 3 — detalhes do protocolo
3GPP TS 29.002	Protocolo MAP — USSD-Arg, USSD-Res, opcodes 59/60/61
3GPP TS 23.038	Alfabeto padrão GSM de 7 bits e esquema de codificação de dados

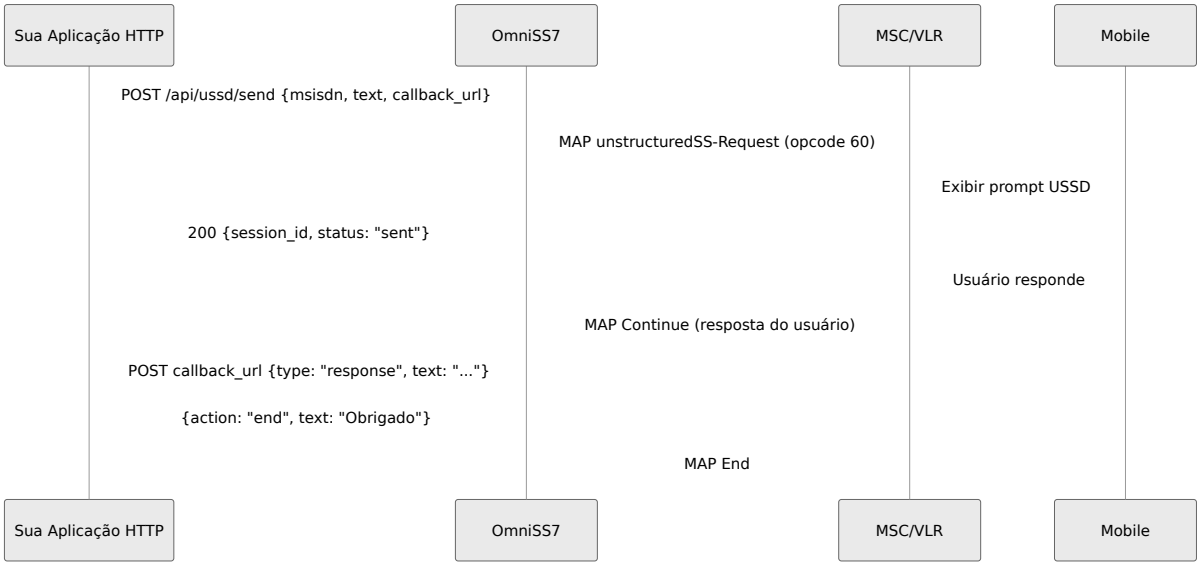
---

Arquitetura

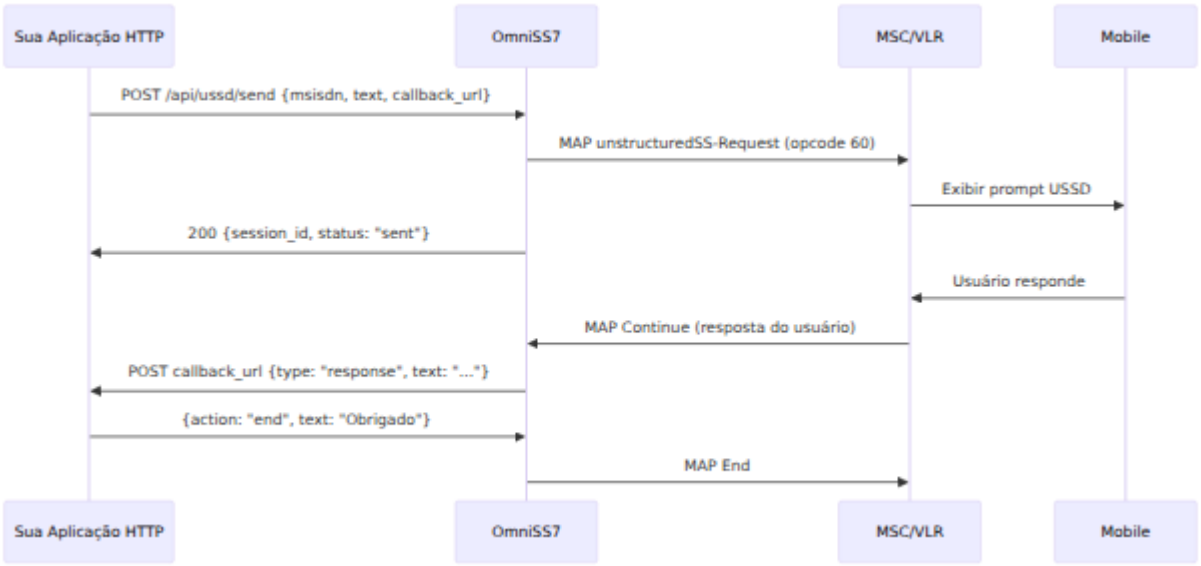
Fluxo Originado pelo Móvel (Entrada)



Fluxo Originado pela Rede (Push Saída)



Visão Geral dos Componentes



Habilitando o Gateway USSD

O Gateway USSD requer que o **modo Cliente MAP** esteja habilitado, além de sua própria flag de recurso.

```
config :omniss7,
  map_client_enabled: true,
  ussd_gateway_enabled: true
```

O gateway também requer uma conexão M3UA funcional (veja o Guia do Cliente MAP para configuração do M3UA).

Configuração

Parâmetros do Gateway USSD

```
config :omniss7,
  ussd_gateway_enabled: true,
  ussd_gateway: %{
    # Roteamento de código curto — correspondência de prefixo mais
longo
    routes: [
      %{pattern: "*100", url: "http://balance-app:9000/ussd"},
      %{pattern: "*200", url: "http://topup-app:9000/ussd"},
      %{pattern: "*",    url: "http://default-app:9000/ussd"}
    ],

    # Timeouts de sessão
    session_timeout_ms: 180_000,    # Duração total da sessão (3
minutos)
    turn_timeout_ms: 30_000,        # Tempo máximo de espera pela
resposta do assinante por turno (30 segundos)

    # Configurações de callback HTTP
    http_timeout_ms: 5_000,         # Timeout para HTTP POST para
seu aplicativo (5 segundos)

    # Limites de texto
    max_text_length: 182            # Máximo de 7 bits GSM (trunca
com aviso se excedido)
  }
```

Referência de Parâmetros

Parâmetro	Tipo	Obrigatório	Padrão
`ussd_gateway_enabled`	Booleano	Sim	`false`
`ussd_gateway.routes`	Lista de Mapas	Sim	`[]`
`ussd_gateway.session_timeout_ms`	Inteiro	Não	`180_000`
`ussd_gateway.turn_timeout_ms`	Inteiro	Não	`30_000`

Parâmetro	Tipo	Obrigatório	Padrão
`ussd_gateway.http_timeout_ms`	Inteiro	Não	`5_000`
`ussd_gateway.max_text_length`	Inteiro	Não	`182`

Parâmetros de Roteamento

Cada entrada na lista `routes` é um mapa:

Parâmetro	Tipo	Obrigatório	Descrição
`pattern`	String	Sim	Prefixo de código curto a ser correspondido. Use `"*"` como um fallback geral. Prefixos mais longos têm prioridade.
`url`	String	Sim	URL do endpoint HTTP para receber POSTs de callback para códigos curtos correspondentes.

Correspondência de Roteamento

As rotas são correspondidas por **prefixo mais longo primeiro**. Para a string de discagem `*100#`:

1. `"*100"` corresponde (comprimento 4) — selecionado
2. `"*10"` corresponde (comprimento 3) — ignorado, mais curto
3. `"*"` corresponde (comprimento 1) — fallback

Se nenhuma rota corresponder, o gateway retorna um erro MAP para o móvel e registra um aviso.

Protocolo de Callback HTTP

Sua aplicação recebe requisições HTTP POST do gateway e responde com instruções JSON.

Requisição do Gateway para Sua Aplicação

Content-Type: `application/json`

Primeiro turno (iniciação da sessão):

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "msisdn": "+254712345678",
  "type": "initiation",
  "text": "*100#",
  "turn": 1
}
```

Turnos subsequentes (assinante respondeu):

```
{
  "session_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "msisdn": "+254712345678",
  "type": "response",
  "text": "1",
  "turn": 2
}
```

Campos da Requisição

Campo	Tipo	Descrição
`session_id`	String	UUID gerado pelo gateway. Único por diálogo USSD. Use isso para correlacionar turnos.
`msisdn`	String	MSISDN do assinante (se disponível na mensagem MAP). Pode estar vazio para algumas redes.
`type`	String	`"initiation"` para o primeiro turno, `"response"` para respostas subsequentes do assinante.
`text`	String	A string de discagem (por exemplo, `*100#`) na iniciação, ou a entrada do assinante (por exemplo, `1`) na resposta.
`turn`	Inteiro	Contador de turnos começando em 1. Incrementa com cada interação do assinante.

Resposta da Sua Aplicação

Sua aplicação deve responder com JSON contendo um `action` e `text`:

Continuar (mostrar menu, aguardar entrada do assinante):

```
{
  "action": "continue",
  "text": "1. Saldo\n2. Recarregar\n3. Transferir"
}
```

Encerrar (mostrar mensagem final, fechar sessão):

```
{
  "action": "end",
  "text": "Seu saldo é $5.00"
}
```

Campos da Resposta

Campo	Tipo	Obrigatório	Descrição
`action`	String	Sim	`"continue"` para manter a sessão aberta e aguardar a entrada do assinante, ou `"end"` para exibir uma mensagem final e fechar a sessão.
`text`	String	Sim	Texto a ser exibido no aparelho do assinante. O comprimento máximo é determinado por `max_text_length` (padrão 182). Use `\n` para quebras de linha.

USSD Originado na Rede (Push API)

Envie uma mensagem USSD para um assinante a partir de sua aplicação.

Endpoint

`POST /api/ussd/send`

Requisição

```
{
  "msisdn": "+254712345678",
  "text": "Você tem uma fatura pendente. Responda 1 para pagar.",
  "callback_url": "http://billing-app:9000/ussd"
}
```

Campos da Requisição

Campo	Tipo	Obrigatório	Descrição
`msisdn`	String	Sim	MSISDN do assinante de destino no formato internacional.
`text`	String	Sim	Texto inicial USSD a ser exibido. Codificado como GSM de 7 bits.
`callback_url`	String	Sim	URL para receber a resposta do assinante via o protocolo de callback padrão.

Resposta

Sucesso (200 OK):

```
{
  "session_id": "xyz-789-abc-123",
  "status": "sent"
}
```

Respostas de erro:

Status HTTP	Corpo	Causa
400	`{"error": "invalid request", "required": ["msisdn", "text", "callback_url"]}`	Campos obrigatórios ausentes
400	`{"error": "gsm7_encode_failed", ...}`	O texto contém caracteres não no alfabeto GSM de 7 bits
500	`{"error": "send_failed", ...}`	Falha ao enviar M3UA (verifique a conectividade)
503	`{"error": "USSD gateway not enabled"}`	`ussd_gateway_enabled` é `false`

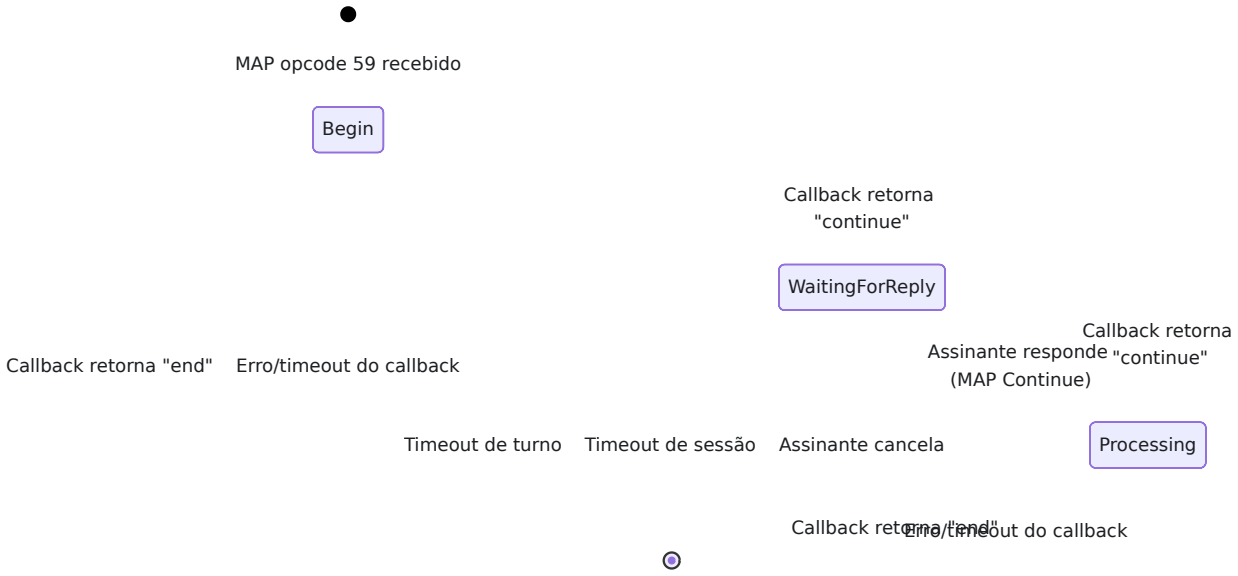
Exemplo de cURL

```
curl -X POST http://localhost:8080/api/ussd/send \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "+254712345678",
    "text": "Você tem uma fatura pendente. Responda 1 para
pagar.",
    "callback_url": "http://billing-app:9000/ussd"
  }'
```

Ciclo de Vida da Sessão

Cada diálogo USSD é rastreado como uma **sessão** com um `session_id` único.

Estados da Sessão



Sessão de Um Turno

Se seu callback retornar `"end"` no primeiro turno, nenhuma sessão persistente é criada. O gateway envia um MAP End com o resultado e retorna imediatamente.

Sessão de Múltiplos Turnos

Se seu callback retornar `"continue"`, o gateway:

1. Cria um **Session GenServer** registrado em `UssdGateway.Registry`
2. Envia um MAP Continue com opcode 60 (unstructuredSS-Request) para o móvel
3. Aguarda a resposta do assinante (até `turn_timeout_ms`)
4. Encaminha a resposta para seu callback
5. Repete até que seu callback retorne `"end"` ou ocorra um timeout

Comportamento de Timeout

Timeout	Padrão	Efeito
Timeout de turno	30 segundos	Se o assinante não responder dentro dessa janela, a sessão é encerrada com um erro MAP.
Timeout de sessão	3 minutos	Duração total da sessão. Encerra a sessão independentemente da atividade.
Timeout de callback HTTP	5 segundos	Se sua aplicação não responder a tempo, o gateway envia um erro MAP para o móvel e encerra a sessão.

Tratamento de Erros

O gateway lida com falhas de forma elegante e sempre tenta enviar uma resposta de erro MAP para o móvel, para que o assinante veja uma mensagem significativa em vez de um timeout de rede.

Cenário	Ação do Gateway	Código de Erro MAP
Timeout de callback HTTP ou 5xx	Encerrar sessão, enviar MAP End com erro	34 (systemFailure)
JSON inválido do callback	Encerrar sessão, enviar MAP End com erro	34 (systemFailure)
Texto USSD excede `max_text_length`	Truncar texto, registrar aviso, continuar normalmente	N/A (truncado, não é um erro)
Timeout do assinante (sem resposta)	Encerrar sessão, enviar MAP End com erro	34 (systemFailure)
Nenhuma rota corresponde ao código curto	Enviar MAP End com erro, registrar aviso	34 (systemFailure)
Falha do Session GenServer	Sessão morre, assinante vê timeout de rede	N/A (saída do processo)
Gateway USSD não habilitado	Retornar Facility Not Supported	21 (facilityNotSupported)

Métricas e Monitoramento

O Gateway USSD expõe métricas Prometheus no endpoint padrão `/metrics` (porta 8080).

Métricas USSD

Métrica: `ussd_requests_total`
Tipo: Contador
Descrição: Total de requisições USSD processadas
Rótulos:

- `direction` — `"inbound"` (originado pelo móvel) ou `"outbound"` (push originado na rede)

Métrica: `ussd_active_sessions`
Tipo: Gauge
Descrição: Número de sessões USSD ativas atualmente

Métrica: `map_request_duration_milliseconds`
Tipo: Histograma
Descrição: Duração das operações de envio USSD em milissegundos
Rótulos:

- `operation` — `"ussd_send"` para requisições de push de saída

Exemplos de Consultas Prometheus

```
# Taxa de requisições USSD por direção
rate(ussd_requests_total[5m])

# Sessões ativas
ussd_active_sessions

# Latência USSD de saída (p95)
histogram_quantile(0.95,
rate(map_request_duration_milliseconds_bucket{operation="ussd_send"}
[5m]))
```

Servidor de Callback de Exemplo

Python (Flask)

```python
from flask import Flask, request, jsonify

app = Flask(__name__)
sessions = {}

@app.route('/ussd', methods=['POST'])
def ussd():
    data = request.json
    session_id = data['session_id']
    text = data['text']
    turn = data['turn']

    if data['type'] == 'initiation':
        sessions[session_id] = {'state': 'main_menu'}
        return jsonify({
            'action': 'continue',
            'text': 'Bem-vindo!\n1. Ver saldo\n2. Comprar
crédito\n3. Transferir'
        })

    state = sessions.get(session_id, {}).get('state')

    if state == 'main_menu':
        if text == '1':
            del sessions[session_id]
            return jsonify({
                'action': 'end',
                'text': 'Seu saldo é $5.00'
            })
        elif text == '2':
            sessions[session_id]['state'] = 'buy_airtime'
            return jsonify({
                'action': 'continue',
                'text': 'Digite o valor:'
            })
        else:
            del sessions[session_id]
            return jsonify({
```

```
                'action': 'end',
                'text': 'Opção inválida. Adeus.'
            })

    elif state == 'buy_airtime':
        del sessions[session_id]
        return jsonify({
            'action': 'end',
            'text': f'Você comprou ${text} de crédito. Obrigado!'
        })

    return jsonify({'action': 'end', 'text': 'Sessão expirada.'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=9000)
```

Node.js (Express)

```
const express = require('express');
const app = express();
app.use(express.json());

const sessions = new Map();

app.post('/ussd', (req, res) => {
  const { session_id, text, type } = req.body;

  if (type === 'initiation') {
    sessions.set(session_id, { state: 'main_menu' });
    return res.json({
      action: 'continue',
      text: 'Bem-vindo!\n1. Ver saldo\n2. Comprar crédito'
    });
  }

  const session = sessions.get(session_id);
  if (!session) {
    return res.json({ action: 'end', text: 'Sessão expirada.' });
  }

  if (session.state === 'main_menu' && text === '1') {
    sessions.delete(session_id);
    return res.json({ action: 'end', text: 'Seu saldo é $5.00' });
  }

  sessions.delete(session_id);
  return res.json({ action: 'end', text: 'Adeus.' });
});

app.listen(9000, () => console.log('Callback USSD na porta
9000'));
```

Solução de Problemas

O Discar USSD Retorna "Serviço Não Disponível"

Sintomas: O assinante disca um código curto e imediatamente recebe um erro de rede.

Possíveis causas:

- `ussd_gateway_enabled` é `false`
- Nenhuma rota corresponde ao código curto discado
- Conexão M3UA está inativa

Resolução:

1. Verifique `ussd_gateway_enabled: true` na configuração
2. Verifique se um padrão de rota corresponde ao código curto (lembre-se de incluir `*` como fallback)
3. Verifique o status do par M3UA na interface Web (página de Pares)

Callback Não Recebendo Requisições

Sintomas: Os logs do gateway mostram início USSD, mas sua aplicação nunca recebe o POST HTTP.

Possíveis causas:

- URL de callback é inacessível a partir do host OmniSS7
- Firewall bloqueando HTTP de saída do OmniSS7
- Aplicação de callback não está em execução

Resolução:

1. Teste a conectividade: `curl -v http://your-app:9000/ussd` a partir do host OmniSS7
2. Verifique as regras do firewall para HTTP de saída

3. Verifique se sua aplicação de callback está escutando na porta configurada

Sessões Expirando Prematuramente

Sintomas: Sessões de múltiplos turnos terminam com "systemFailure" antes que o assinante possa responder.

Possíveis causas:

- `turn_timeout_ms` é muito curto para sua base de assinantes
- `http_timeout_ms` é muito curto para o tempo de processamento da sua aplicação
- Latência de rede entre OmniSS7 e seu servidor de callback

Resolução:

1. Aumente `turn_timeout_ms` (o padrão de 30 segundos deve ser suficiente para a maioria dos casos)
2. Aumente `http_timeout_ms` se sua aplicação precisar de mais tempo de processamento
3. Implemente o servidor de callback próximo ao OmniSS7 para reduzir a latência

Erros de Codificação GSM de 7 bits

Sintomas: Erros `gsm7_encode_failed` nos logs ou respostas 400 de `/api/ussd/send`.

Possíveis causas:

- O texto contém caracteres fora do alfabeto padrão GSM de 7 bits (por exemplo, emoji, caracteres CJK)

Resolução:

- Restrinja o texto USSD ao conjunto básico de caracteres GSM: letras ASCII, dígitos, pontuação comum e alguns caracteres gregos/nórdicos

- Consulte a Seção 6.2.1 do 3GPP TS 23.038 para a tabela completa de caracteres

---

Documentação Relacionada

- **Guia da API** — Referência completa da API REST (todos os endpoints, incluindo `/api/ussd/send`)
- **Guia do Cliente MAP** — Configuração da conexão M3UA necessária para USSD
- **Referência de Configuração** — Todos os parâmetros de configuração
- **Guia de Recursos Comuns** — Interface Web, monitoramento e configuração do Prometheus

Guia da Interface Web

← Voltar para a Documentação Principal

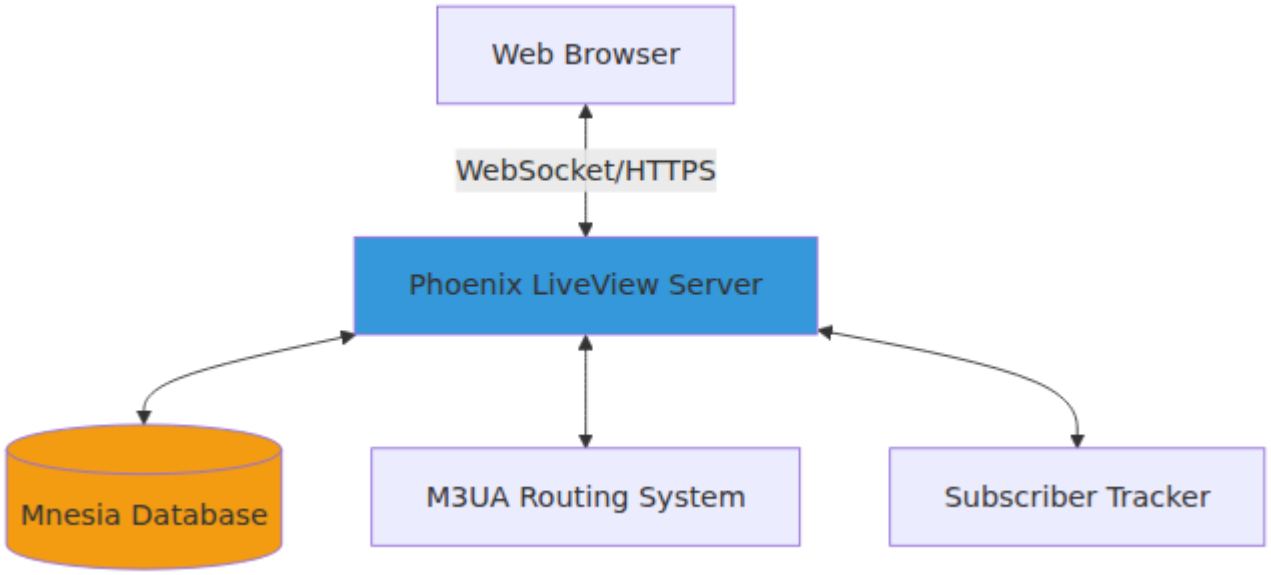
Este guia fornece documentação abrangente para o uso da **Interface Web** do OmniSS7 (interface Phoenix LiveView).

Índice

Visão Geral

A Interface Web do OmniSS7 é um aplicativo **Phoenix LiveView** que fornece capacidades de monitoramento e gerenciamento em tempo real. As páginas disponíveis dependem do modo operacional ativo (STP, HLR ou SMSc).

Arquitetura da Interface Web



Configuração do Servidor

- **Protocolo**: HTTPS
- **Porta**: 443 (configurada em `config/runtime.exs`)
- **IP Padrão**: 0.0.0.0 (escuta em todas as interfaces)
- **Certificados**: Localizados em `priv/cert/`

URL de Acesso: `https://[server-ip]:443`

Acessando a Interface Web

Pré-requisitos

1. **Certificados SSL**: Certifique-se de que certificados SSL válidos estão presentes em `priv/cert/`:
   - `omnitouch.crt` - Arquivo de certificado
   - `omnitouch.pem` - Arquivo de chave privada
2. **Aplicação em Execução**: Inicie a aplicação com `iex -S mix`

3. **Firewall**: Certifique-se de que a porta 443 está aberta para tráfego HTTPS

Páginas Disponíveis por Modo

Página	Modo STP	Modo HLR	Modo SMSc	Descrição
Eventos SS7	✅	✅	✅	Registro de eventos e captura de mensagens SCCP
Cliente SS7	✅	✅	✅	Teste manual de operações MAP
M3UA	✅	✅	✅	Status da conexão M3UA
Roteamento	✅	✅	✅	Gerenciamento da tabela de roteamento M3UA
Teste de Roteamento	✅	✅	✅	Teste e validação de rotas
Links HLR	✅	✅	✅	Status da API HLR e gerenciamento de assinantes
Assinantes Ativos	✅	✅	✅	Rastreamento em tempo real da localização de assinantes (HLR)
Links SMSc	✅	✅	✅	Status da API SMSc e gerenciamento de filas
Assinantes SMSc	✅	✅	✅	Rastreamento em tempo real de assinantes (SMSc)
Aplicação	✅	✅	✅	Recursos do sistema e monitoramento

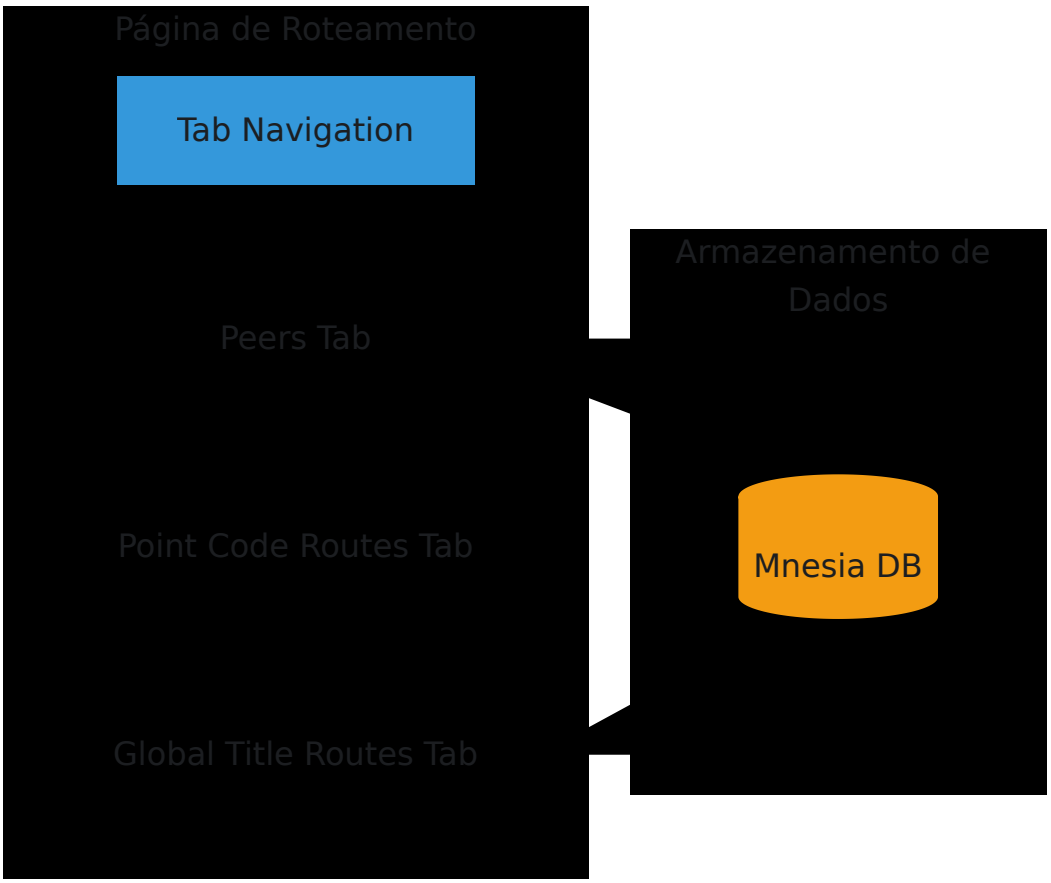
Página	Modo STP	Modo HLR	Modo SMSc	Descrição
Configuração	□	□	□	Visualizador de configuração

Gerenciamento de Roteamento

Página: `/routing`
Modos: STP, SMSc
Atualização Automática: A cada 5 segundos

A página de Gerenciamento de Roteamento fornece uma interface em abas para gerenciar tabelas de roteamento M3UA.

Layout da Página



Aba de Peers

Gerencie conexões de peers M3UA (outros STPs, HLRs, MSCs, SMSCs).

Colunas da Tabela de Peers

Coluna	Descrição	Exemplo
ID	Identificador único do peer	`1`
Nome	Nome legível do peer	`"STP_West"`
Função	Função da conexão	`client`, `server`, `stp`
Point Code	Código de ponto SS7 do peer	`100`
Remoto	IP:Porta remota	`10.0.0.10:2905`
Status	Status da conexão	`active`, `aspup`, `down`
Ações	Botões Editar/Excluir	-

Adicionando um Peer

1. **Clique** na aba Peers
2. **Preencha** os campos do formulário:
   - **Peer ID**: Gerado automaticamente se deixado em branco
   - **Peer Name**: Nome descritivo (obrigatório)
   - **Role**: Selecione `client`, `server` ou `stp`
   - **Point Code**: Código de ponto SS7 (obrigatório)
   - **Local IP**: Endereço IP do seu sistema
   - **Local Port**: 0 para atribuição dinâmica de porta
   - **Remote IP**: Endereço IP do peer
   - **Remote Port**: Porta do peer (tipicamente 2905)
   - **Routing Context**: ID do contexto de roteamento M3UA
   - **Network Indicator**: `international` ou `national`

3. **Clique** em "Add Peer"

Persistência: O peer é imediatamente salvo no Mnesia e sobrevive a reinicializações.

Editando um Peer

1. **Clique** no botão "Edit" na linha do peer
2. **Modifique** os campos do formulário conforme necessário
3. **Clique** em "Update Peer"

Nota: Se você mudar o Peer ID, o peer antigo será excluído e um novo será criado.

Excluindo um Peer

1. **Clique** no botão "Delete" na linha do peer
2. **Confirme** a exclusão (todas as rotas que usam este peer também serão removidas)

Indicadores de Status do Peer

Status	Cor	Descrição
`active`	□ Verde	Peer está conectado e roteando mensagens
`aspup`	□ Amarelo	ASP está ativo, mas ainda não está em operação
`down`	□ Vermelho	Peer está desconectado

Aba de Rotas de Código de Ponto

Configure regras de roteamento com base nos Códigos de Ponto de Destino.

Colunas da Tabela de Rotas

Coluna	Descrição	Exemplo
Destino PC	Código de ponto alvo (formato zone.area.id)	`1.2.3` (100)
Máscara	Máscara de sub-rede para correspondência de PC	`/14` (exato), `/8` (intervalo)
Peer ID	Peer alvo para esta rota	`1`
Peer Name	Nome do peer alvo	`"STP_West"`
Prioridade	Prioridade da rota (1 = mais alta)	`1`
Rede	Indicador de rede	`international`
Ações	Botões Editar/Excluir	-

Adicionando uma Rota de Código de Ponto

1. **Clique** na aba "Point Code Routes"
2. **Preencha** os campos do formulário:
   - **Destino Point Code**: Insira como `zone.area.id` (ex: `1.2.3`) ou inteiro (0-16383)
   - **Máscara**: Selecione máscara `/14` para correspondência exata, valores mais baixos para intervalos
   - **Peer ID**: Selecione o peer alvo no dropdown
   - **Prioridade**: Insira a prioridade (1 = mais alta, padrão)
   - **Network Indicator**: Selecione `international` ou `national`
3. **Clique** em "Add Route"

Formato do Código de Ponto: Você pode inserir códigos de ponto em dois formatos:

- **Formato 3-8-3**: `zone.area.id` (ex: `1.2.3`)
- **Formato Inteiro**: `0-16383` (ex: `1100`)

O sistema converte automaticamente entre os formatos.

Entendendo Máscaras

Os códigos de ponto são valores de 14 bits (0-16383). A máscara especifica quantos bits mais significativos devem corresponder:

Máscara	PCs Correspondidos	Caso de Uso
`/14`	1 (correspondência exata)	Roteamento para destino específico
`/13`	2 PCs	Pequeno intervalo
`/8`	64 PCs	Intervalo médio
`/0`	Todos os 16.384 PCs	**Rota padrão/catch-all**

Exemplos:

- `PC 1000 /14` → Corresponde apenas ao PC 1000
- `PC 1000 /8` → Corresponde ao PC 1000-1063 (64 PCs consecutivos)
- `PC 0 /0` → Corresponde a todos os códigos de ponto (rota padrão)

Cartão de Referência de Máscara de Código de Ponto

A página da web inclui uma referência interativa mostrando todos os valores de máscara e seus intervalos.

Aba de Rotas de Título Global

Configure regras de roteamento com base nos endereços de Título Global SCCP.

Requisito: O roteamento de Título Global deve estar habilitado na configuração:

```
config :omniss7,
  enable_gt_routing: true
```

Colunas da Tabela de Rotas

Coluna	Descrição	Exemplo
GT Prefix	Prefixo GT da parte chamada (vazio = fallback)	`"1234"`, `""`
Source SSN	Correspondência no SSN da parte chamada (opcional)	`6` (HLR), `any`
Peer ID	Peer alvo	`1`
Peer	Nome do peer	`"HLR_West (1)"`
Dest SSN	Reescrever SSN ao encaminhar (opcional)	`6`, `preserve`
Prioridade	Prioridade da rota	`1`
Descrição	Descrição da rota	`"Números dos EUA"`
Ações	Botões Editar/Excluir	-

Adicionando uma Rota de Título Global

1. **Clique** na aba "Global Title Routes"
2. **Preencha** os campos do formulário:
   - **GT Prefix**: Deixe vazio para rota de fallback, ou insira dígitos (ex: `"1234"`)
   - **Source SSN**: Opcional - filtrar pelo SSN da parte chamada
   - **Peer ID**: Selecione o peer alvo
   - **Dest SSN**: Opcional - reescrever SSN ao encaminhar

    - **Prioridade**: Prioridade da rota (1 = mais alta)
    - **Descrição**: Descrição legível
3. **Clique** em "Add Route"

Rotas de Fallback: Se o GT Prefix estiver vazio, a rota atua como um catch-all para GTs que não correspondem a nenhuma outra rota.

Valores Comuns de SSN

A página inclui um cartão de referência com valores comuns de SSN:

SSN	Elemento de Rede
6	HLR (Home Location Register)
7	VLR (Visitor Location Register)
8	MSC (Mobile Switching Center)
9	EIR (Equipment Identity Register)
10	AUC (Authentication Center)
142	RANAP
145	gsmSCF (Service Control Function)
146	SGSN

Reescrita de SSN

- **Source SSN**: Correspondência no SSN da Parte Chamadora em mensagens recebidas
- **Dest SSN**: Se definido, reescreve o SSN da Parte Chamadora ao encaminhar
    - Vazio = preservar SSN original
    - Valor = substituir por este SSN

Caso de Uso: Roteie mensagens com SSN=6 (HLR) para um peer e reescreva para SSN=7 (VLR) no lado de saída.

Persistência da Tabela de Roteamento

Todas as rotas são armazenadas no Mnesia e sobrevivem a reinicializações da aplicação.

Como as Rotas Persistem

1. **Mudanças na Interface Web**: Todas as operações de adicionar/editar/excluir são imediatamente salvas no Mnesia
2. **Reinicialização da Aplicação**: As rotas são carregadas do Mnesia na inicialização
3. **Mesclagem de runtime.exs**: Rotas estáticas de `config/runtime.exs` são mescladas com as rotas do Mnesia (sem duplicatas)

Prioridade da Rota

Quando várias rotas correspondem a um destino:

1. **Mais Específico Primeiro**: Valores de máscara mais altos (mais específicos) têm precedência
2. **Campo de Prioridade**: Números de prioridade mais baixos roteiam primeiro (1 = maior prioridade)
3. **Status do Peer**: Apenas rotas para peers `active` são usadas

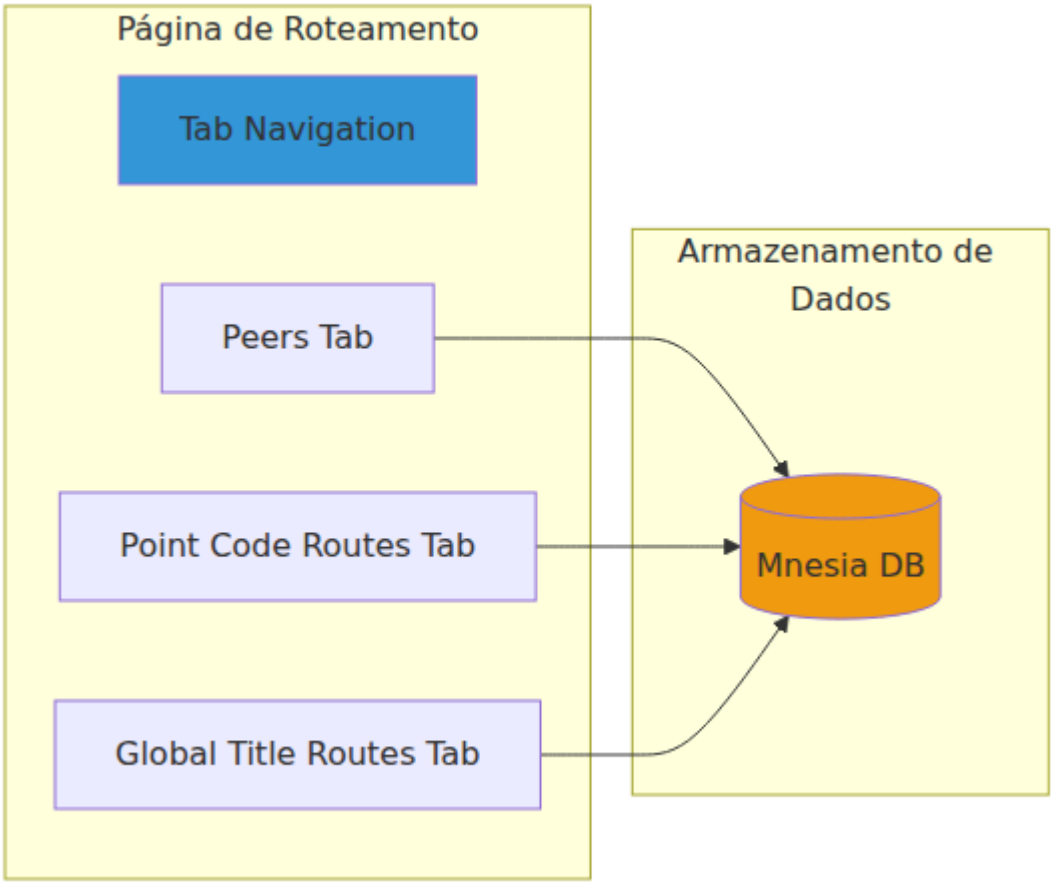
---

Assinantes Ativos

Página: `/subscribers`
Modo: Apenas HLR
Atualização Automática: A cada 2 segundos

Exibe rastreamento em tempo real de assinantes que enviaram solicitações de UpdateLocation.

Recursos da Página



Colunas da Tabela de Assinantes

Coluna	Descrição	Exemplo
IMSI	IMSI do assinante	`"50557123456789"`
Número VLR	Endereço GT VLR atual	`"555123155"`
Número MSC	Endereço GT MSC atual	`"555123155"`
Atualizado Em	Último timestamp de UpdateLocation	`"2025-10-25 14:23:45 UTC"`
Duração	Tempo desde o registro	`"2h 15m 34s"`

Resumo de Estatísticas

Quando assinantes estão presentes, um cartão de resumo exibe:

- **Total Ativo**: Número total de assinantes registrados
- **VLRs Únicos**: Número de endereços VLR distintos
- **MSCs Únicos**: Número de endereços MSC distintos

Limpando Assinantes

Botão Limpar Tudo: Remove todos os assinantes ativos do rastreador.

Confirmação: Requer confirmação antes de limpar (não pode ser desfeito).

Caso de Uso: Limpar registros de assinantes obsoletos após manutenção ou teste de rede.

Atualização Automática

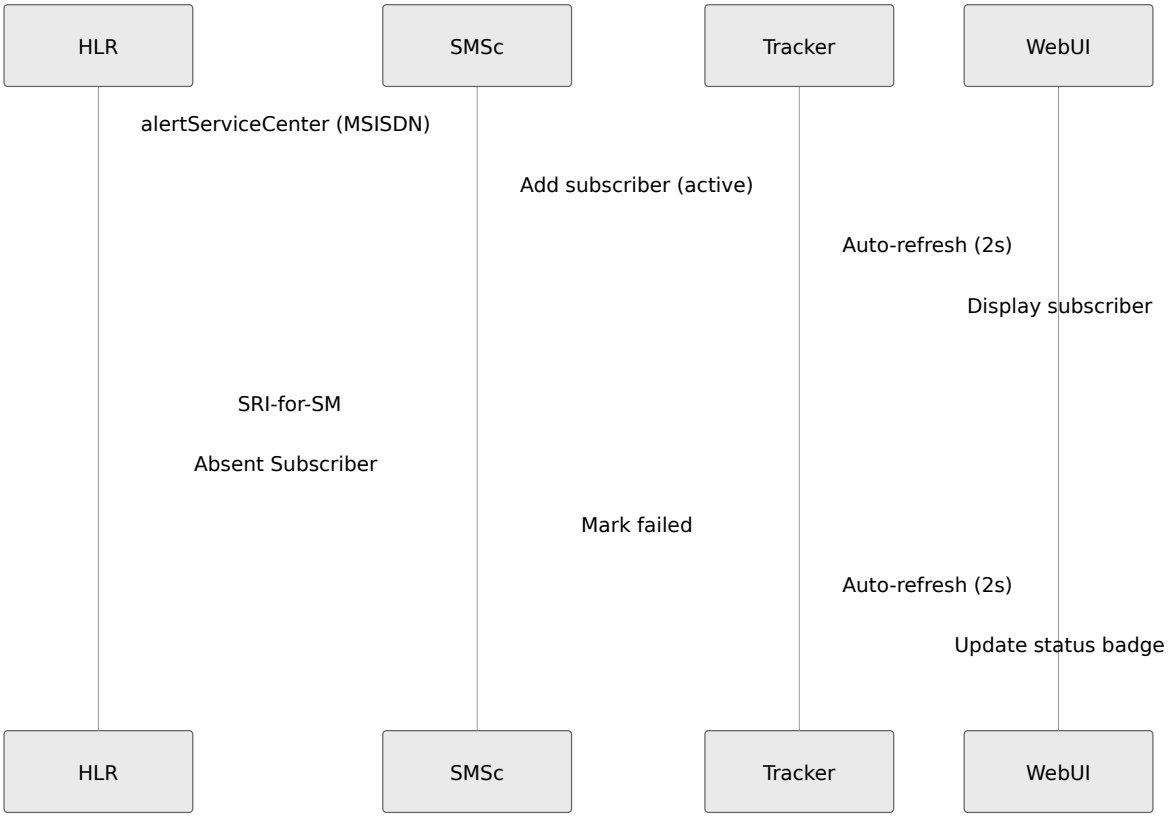
A página atualiza automaticamente a cada **2 segundos** para mostrar atualizações em tempo real dos assinantes.

Assinantes SMSc

Página: `/smsc_subscribers`
Modo: Apenas SMSc
Atualização Automática: A cada 2 segundos

Exibe rastreamento em tempo real de assinantes com base nas mensagens alertServiceCenter recebidas dos HLRs, status de entrega de mensagens e rastreamento de falhas.

Recursos da Página



Colunas da Tabela de Assinantes

Coluna	Descrição	Exemplo
MSISDN	Número de telefone do assinante	`"15551234567"`
IMSI	IMSI do assinante	`"001010123456789"`
HLR GT	HLR GT que enviou alertServiceCenter	`"15551111111"`
Msgs Enviadas	Contagem de mensagens MT-FSM enviadas	`5`
Msgs Recebidas	Contagem de mensagens MO-FSM recebidas	`2`
Status	`Active` ou `Failed` (codificado por cor)	`● Active`
Última Atualização	Timestamp da última atualização	`"2025-10-30 14:23:45 UTC"`
Duração	Tempo desde a última atualiza��ão	`"15m 34s"`

Indicadores de Status

- ● **Active** (Verde): Assinante está acessível, último alertServiceCenter recebido com sucesso
- ○ **Failed** (Vermelho): Última tentativa de entrega falhou (erro SRI-for-SM ou assinante ausente)

Resumo de Estatísticas

Quando assinantes estão presentes, um cartão de resumo exibe:

- **Total Rastreado**: Número total de assinantes rastreados
- **Ativos**: Número de assinantes com status ativo
- **Falhados**: Número de assinantes com status falhado
- **HLRs Únicos**: Número de HLRs distintos enviando alertas

Gerenciando Assinantes

Botão Remover: Remove assinante individual do rastreamento.

Botão Limpar Tudo: Remove todos os assinantes rastreados.

Confirmação: Limpar tudo requer confirmação antes de limpar (não pode ser desfeito).

Caso de Uso:

- Remover entradas obsoletas após problemas de rede
- Limpar dados de teste após desenvolvimento
- Monitorar quais HLRs estão enviando alertas

Contadores de Mensagens

O rastreador incrementa automaticamente os contadores:

- **Mensagens Enviadas**: Incrementado quando SRI-for-SM é bem-sucedido e MT-FSM é enviado
- **Mensagens Recebidas**: Incrementado quando MO-FSM é recebido do assinante

Atualização Automática

A página atualiza automaticamente a cada **2 segundos** para mostrar atualizações em tempo real de assinantes e status.

Operações Comuns

Pesquisa e Filtragem

Atualmente, a Interface Web não inclui funcionalidade de pesquisa/filtragem embutida. Para encontrar rotas específicas:

1. Use a função de busca do seu navegador (Ctrl+F / Cmd+F)
2. Pesquise por nomes de peers, códigos de ponto ou prefixos GT

Operações em Lote

Para realizar alterações em lote nas rotas:

1. **Opção 1**: Use a API REST para acesso programático
2. **Opção 2**: Edite `config/runtime.exs` e reinicie a aplicação
3. **Opção 3**: Use a Interface Web para alterações individuais nas rotas

Exportar/Importar

Nota: A Interface Web não suporta atualmente a exportação ou importação de tabelas de roteamento. As rotas são:

- Armazenadas em arquivos de banco de dados Mnesia
- Configuradas em `config/runtime.exs`

Para fazer backup das rotas:

1. **Mnesia**: Faça backup do diretório `Mnesia.{node_name}/`
2. **Config**: Controle de versão de `config/runtime.exs`

Comportamento de Atualização Automática

Diferentes páginas têm diferentes intervalos de atualização:

Página	Intervalo de Atualização	Razão
Gerenciamento de Roteamento	5 segundos	Mudan��as de rota são infrequentes
Assinantes Ativos	2 segundos	Estado dos assinantes muda frequentemente
Status M3UA	Varia por página	Monitoramento do estado da conexão

Conexão WebSocket: Todas as páginas usam conexões WebSocket do Phoenix LiveView para atualizações em tempo real.

Interrupção de Rede: Se a conexão WebSocket for perdida, a página tentará se reconectar automaticamente.

Solução de Problemas

Página Não Carregando

1. **Verifique o Certificado HTTPS**: Certifique-se de que `priv/cert/omnitouch.crt` e `.pem` estão presentes
2. **Verifique a Porta 443**: Verifique se as regras do firewall permitem tráfego HTTPS
3. **Aplicação em Execução**: Confirme se a aplicação está em execução com `iex -S mix`

4. **Console do Navegador**: Verifique se há erros de certificado SSL (avisos de certificado autoassinado)

Rotas Não Persistindo

1. **Verifique o Armazenamento do Mnesia**: Verifique `mnesia_storage_type: :disc_copies` na configuração
2. **Diretório do Mnesia**: Certifique-se de que o diretório do Mnesia é gravável
3. **Verifique os Logs**: Procure por erros do Mnesia nos logs da aplicação

Atualização Automática Não Funcionando

1. **Conexão WebSocket**: Verifique o console do navegador para erros do WebSocket
2. **Rede**: Verifique a conexão de rede estável
3. **Recarregar a Página**: Tente atualizar a página (F5)

Documentação Relacionada

- **Guia STP** - Configuração detalhada de roteamento
- **Guia HLR** - Gerenciamento de assinantes
- **Guia API** - API REST para acesso programático
- **Referência de Configuração** - Todos os parâmetros de configuração

---

Resumo

A Interface Web do OmniSS7 fornece gerenciamento intuitivo e em tempo real de tabelas de roteamento e rastreamento de assinantes:

 Atualizações em Tempo Real - Atualização automática mantém os dados atuais
 Armazenamento Persistente - Mnesia garante que as rotas sobrevivam a

reinicializações
 Interface Baseada em Função - Páginas se adaptam ao modo operacional (STP/HLR/SMSc)
 Gerenciamento Interativo - Adicione, edite, exclua rotas sem reinicialização
 Monitoramento de Status - Status de conexão e peer ao vivo

Para operações avançadas ou automação, consulte o Guia API.