

安裝

安裝說明書

安裝

OmniMessage SMPP 服務器 安裝說明書 SMPP 服務器 OmniMessage 安裝說明書
安裝說明書 OmniMessage Core 安裝 - 安裝

1. 安裝 安裝說明書 SMPP PDU
2. 安裝 **REST API** 安裝 OmniMessage 安裝
3. 安裝 OmniMessage 安裝說明書
4. 安裝 SMPP PDU 安裝
5. 安裝 安裝 OmniMessage

安裝 OmniMessage 安裝DiameterMAPIMS安裝說明書 - 安裝說明書
OmniMessage Core

安裝

```
/opt/omnimessage-smpp/config/runtime.exs
```

安裝說明書

```
sudo systemctl restart omnimessage-smpp
```

安裝

安裝 Elixir 安裝說明書

```
import Config

# 配置
config :omnimessage_smpp,
  setting_name: value

# SMPP 配置
config :omnimessage_smpp, :binds, [
  %{
    name: "bind_name",
    # ... 配置
  }
]
```

配置

API_BASE_URL

OmniMessage Core 配置 URL

```
config :omnimessage_smpp,
  api_base_url: "https://omnimessage-core.example.com:8443"
```

配置	配置	配置	配置
api_base_url	URL (URL)		-

OmniMessage Core 配置 URL 配置 REST API 配置 OmniMessage 配置

- 配置 SMPP 配置 OmniMessage 配置
- 配置 SMPP 配置
- 配置 OmniMessage
- 配置

配置“配置”配置 OmniMessage 配置

- ✓ 設定済みのポート
- ✓ 設定済みのポート番号
- ✓ 設定済みのポート
- ✓ 設定済みのポート
- ✓ 設定済みのポート
- ✓ 設定済みのポート
- ✓ 設定済みのポート

設定済みの SMPP ↔ OmniMessage 設定済みの

設定

```
# IP 上 HTTPS
api_base_url: "https://192.168.1.100:8443"

# 設定済みの HTTPS
api_base_url: "https://omnimessage-core.company.com:8443"

# HTTP 設定済みの
api_base_url: "http://192.168.1.100:8080"
```

設定済みの

- 設定済みの OmniMessage Core
- 設定済みの HTTPS 設定済みの `verify_ssl_peer`
- 設定済みの HTTPS

OAM/設定 API 設定 (`api_ex`)

設定済みの OAM/設定 API 設定

設定済みの `api_ex` 設定済みの OAM/設定 API (`/status`, `/smpp/client_peers`, `/smpp/server_peers`, `/smpp/binds`, `/smpp/summary`) 設定済みの `8443` (`config/config.exs`) 設定済みの `api_base_url` 設定済みの OmniMessage Core 設定済みの 設定済みの

設定済みの

項目	項目	項目	項目
api_base_url	API → Core (URL)	...:8443	API/メッセージ
api_ex port	API (ポート)	8443	API OAM API

OmniMessage Core api_ex 8443 API SMPP Core 8443 - :eaddrinuse /api/messages 404

config/runtime.exs /etc/omnmessage_smpp/runtime.exs api_ex OAM API Core api_base_url Core 8443

```
# Core :8443 API OAM API :8444
config :api_ex,
  api:
    Application.get_env(:api_ex, :api, %{})
    |> Map.put(:port, 8444)
    |> Map.put(:listen_ip, "0.0.0.0")
```

8443Core

SMPP_POLL_INTERVAL

```
config :omnmessage_smpp,
  smpp_poll_interval: 100
```

項目	項目	項目	項目
smpp_poll_interval	項目	項目	100

時間

- 高負荷 (>100 TPS) 100-500ms
- 中負荷 (10-100 TPS) 500-1000ms
- 低負荷 (<10 TPS) 1000-2000ms

環境変数 `SMPP_POLL_INTERVAL`

VERIFY_SSL_PEER

SSL 検証

```
config :omnimessage_smpp,  
  verify_ssl_peer: false
```

項目	デフォルト	型	説明
<code>verify_ssl_peer</code>	false	Boolean	SSL 接続時のピア検証を有効にするかどうか

API 呼び出し時に SSL 接続を使用するかどうか

値

- `true` SSL 接続を有効にする
- `false` SSL 接続を無効にする

環境変数 `VERIFY_SSL_PEER`

SMSC_NAME

説明

```
config :omnimessage_smpp,  
  smsc_name: "smpp_gateway"
```


配置示例

```
config :omnimessage_smpp, :binds, [
  %{
    # 名称
    name: "vodafone_uk",

    # 模式
    mode: :client,

    # SMPP 绑定
    bind_type: :transceiver,

    # SMPP 主机
    host: "smpp.vodafone.co.uk",
    port: 2775,

    # 绑定源 IP/端口
    # 绑定源 IP/端口
    bind_source_ip: "10.20.30.40",
    bind_source_port: 5000,

    # 系统 ID/密码
    system_id: "your_username",
    password: "your_password",

    # SMPP 系统类型
    system_type: "",
    addr_ton: 0,
    addr_npi: 0,
    address_range: "",

    # TPS 限制
    tps_limit: 100,

    # 队列名称
    # 队列名称
    queue: "vodafone_uk",

    # 队列检查频率
    queue_check_frequency: 1000,

    # 队列大小
    queue_size: 1000
```

```
    enquire_link_interval: 60,  
  
    # 配置缓存  
    cache_enabled: false,  
    cache_max_size: 10000,  
    cache_retry_interval: 60  
  }  
]
```

配置名称

name

配置名称

名称	值	说明
名称		"vodafone_uk"

配置名称 SMPP 配置

- 配置名称
- 配置名称
- 配置名称

配置名称

- carrier_region "vodafone_uk" "att_us"
- purpose_number "marketing_1" "alerts_primary"

mode

配置名称

名称	值	说明
名称		:client

SMSC ESME

:client

bind_type

SMPP

		:transmitter, :receiver, :transceiver

- :transmitter - (submit_sm)
- :receiver - (deliver_sm)
- :transceiver -

:transceiver

host

SMPP IP

		"smpp.carrier.com" "10.5.1.100"

SMPP

```
host: "smpp.vodafone.co.uk"
host: "10.20.30.40"
host: "smpp-primary.carrier.net"
```

port

SMPP □□□□□

00	00	00	00
00	0	2775	1-65535

□□□SMPP □□□ TCP □□□

□□□□□2775

111

```
port: 2775 # []
port: 3000 # []
```

bind_source_ip

□□□□□□□ IP

□□	□□	□□	□□
□□ 	□	(□□□□□□)	"10.20.30.40"

TCP/IP 프로토콜 스택

TCP/IP 프로토콜 스택은 네트워킹을 위한 데이터 전송의 기본 단위를 정의한다.

```

:binds Web UI / Mnesia

```

bind_source_port

[illegible]

00	00	00	00
00	0	(00)	1-65535

TCP 0 IP 0 SO_REUSEADDR
TIME_WAIT :eaddrinuse

bind_source_ip Web UI

system_id

□ □ □ □ □ □ □

□□	□□	□□
□□□	□	"company_user"

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

□□□□□□□□ - □□□□□□□□

password

□ □ □ ◆ ? ◆ ? ◆ ? □ □

00	00	00
000	0	"secret_password"

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

□□□□

- 
- 
- 

tps_limit

□ □ □ □ □ □

項目	単位	値	範囲
tps_limit	個/秒	100	1-10000

tps_limit の設定方法

設定

- tps_limit を 70-80%
- 単位/秒
- 範囲

設定

```
tps_limit: 10    # 単位
tps_limit: 50    # 範囲
tps_limit: 100   # 範囲
tps_limit: 1000  # 範囲
```

設定

```
tps_limit = 100 TPS
tps_limit = 70-80
20-30 TPS 範囲
```

queue

設定

項目	単位	値	範囲
queue	個	name	"sinch.smp"

queue の設定方法 name 範囲 queue 範囲
Sinch 範囲 "sinch.smp" 範囲/範囲

設定

```
# 配置示例
%{name: "sinch_1", queue: "sinch.smpp", ...}
%{name: "sinch_2", queue: "sinch.smpp", ...}
```

queue_check_frequency

配置示例

名称	单位	默认值	范围
queue_check_frequency	秒	1000	100-10000

配置说明

说明

- 高流量 (>100 TPS) 500-1000ms
- 中流量 (10-100 TPS) 1000-2000ms
- 低流量 (<10 TPS) 2000-5000ms

配置

- 配置项 = 配置项名称 API 名称
- 配置项 = 配置项名称 API 名称

enqueue_link_interval

SMPP 配置示例

名称	单位	默认值	范围
enqueue_link_interval	秒	60	0-3600

配置说明 SMPP enqueue_link PDU 配置项 enqueue_link_resp 配置

配置

- **60** 分钟
- **15-30** 分钟
- **120-300** 分钟
- **0** 分钟 `enquire_link`

111

```
enquire_link_interval: 60 # 001000
enquire_link_interval: 30 # 000000
enquire_link_interval: 0  # 00
```

system_type

SMPP □□□□□□

□□	□□	□□	□□
□□□	□	""	"ОТР"

SMPP

addr_ton

□ □ □ □ □ □ □

□□	□□	□□	□□
□□	□	0	0-6

□□□□□□□□□□ SMPP □□□□□□□□□□□□□□□□

1111

- 0 -
- 1 -
- 2 -
- 5 -

Web UI

SMPP

SMSC ESME

配置参数

```
config :omnimessage_smpp, :server_binds, [
  %{
    # 合作伙伴名称
    name: "partner_acme",

    # 系统ID
    system_id: "acme_corp",
    password: "acme_secret",

    # 允许的绑定类型
    allowed_bind_types: [:transmitter, :receiver, :transceiver],

    # IP 白名单
    ip_whitelist: ["192.168.1.0/24", "10.50.1.100"],

    # 源地址白名单 = 源地址
    source_address_whitelist: [],

    # TPS 限制
    tps_limit: 50,

    # 队列检查频率
    queue_check_frequency: 1000,

    # 链路轮询间隔
    enquire_link_interval: 60,

    # 缓存配置
    cache_enabled: false,
    cache_max_size: 10000,
    cache_retry_interval: 60,

    # 是否发送DLR回执
    deliver_receipts: false
  }
]
```

XXXXXXXXXX

name

XXXXXXXXXX

XX	XX	XX
XXXX	X	"partner_acme"

XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXX/XXXXXXXXXXXX

system_id

XXXXXXXXXX

XX	XX	XX
XXXX	X	"acme_corp"

XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX

password

XXXXXXXXXX

XX	XX	XX
XXXX	X	"secure_password"

XXXXXXXXXXXXXXXXXXXX

XXXX

- 接收器
- 发射器
- 收发器

allowed_bind_types

接收器

名称	类型	默认值
allowed_bind_types	字符串	-

接收器、发射器、收发器

示例

```
allowed_bind_types: [:transceiver] # 收发器
allowed_bind_types: [:transmitter, :receiver] # 发射器、接收器
allowed_bind_types: [:transmitter, :receiver, :transceiver] # 收发器、发射器、接收器
```

接收器、发射器、收发器

ip_whitelist

IP 地址列表

名称	类型	默认值	描述
ip_whitelist	数组	[]	IP 地址 CIDR 列表

接收器 - 接收器 IP 地址

示例

- IP "192.168.1.100" /32
- CIDR "192.168.1.0/24" "10.0.0.0/8"
- ["192.168.1.0/24", "10.50.1.100"]

000

```
# 0000 IP000000
ip_whitelist: []

# 00 IP
ip_whitelist: ["203.0.113.50"]

# 00 IP
ip_whitelist: ["203.0.113.50", "203.0.113.51"]

# 00
ip_whitelist: ["192.168.1.0/24"]

# 00
ip_whitelist: ["192.168.1.0/24", "10.50.1.100", "10.60.0.0/16"]
```

00000

- /32 - 00 IP000000000000 IP
- /24 - 256 0000000192.168.1.0-255
- /16 - 65,536 000000010.50.0.0-255.255
- /8 - 16,777,216 000000010.0.0.0-255.255.255.255

source_address_whitelist

0000000

00	00	00	00
000000	0	[]	00000000

0000000000000000000000000000 ID0000000000000000000000000000

000

```
# 默认配置
source_address_whitelist: []

# 品牌白名单
source_address_whitelist: ["^MyBrand$", "^AlertService$"]

# 国家
source_address_whitelist: ["^614", "^\\+61"]

# 手机号
source_address_whitelist: ["^61[45]\\d{8}$"]

# 完整配置
source_address_whitelist: ["^MyBrand$", "^614", "^\\+614000000001$"]
```

配置项 `ESME_RINVSRCADR` 在 `SOURCE_ADDRESS_WHITELIST.md`

tps_limit

配置项

配置项 `tps_limit` 用于 - 限制 outbound deliver_sm 速率

queue_check_frequency

配置项

配置项 `queue_check_frequency` 用于 - 检查队列的频率

enquire_link_interval

SMPP 配置项

配置项 `enquire_link_interval` 用于 - 检查 enquire_link PDU 的频率

enabled

配置项

submit_sm ID submit_sm_resp
OmniMessage deliver_sm
ID stat:DELIVRD stat:EXPIRED stat:UNDELIV stat:REJECTD

	stat
delivered	DELIVRD
expired	EXPIRED
dropped	UNDELIV
rejected	REJECTD
balance_rejected	REJECTD

USAGE.md MONITORING.md

Web UI

配置

配置文件

配置

```
config :omnimessage_smpp, :listen, %{  
  host: "0.0.0.0",  
  port: 2775,  
  max_connections: 100  
}
```

配置

host

配置 IP

配置	配置	配置	配置
配置	配置	"0.0.0.0"	"0.0.0.0" 和 "127.0.0.1"

配置

配置

- "0.0.0.0" - 配置
- "127.0.0.1" - 配置
- "192.168.1.10" - 配置 IP

port

配置 TCP

IP	Port	Protocol	Service
10.10.10.1	2775	TCP	1-65535

SMPP connection

2775

max_connections

10000

IP	Port	Protocol	Service
10.10.10.1	100	TCP	1-10000

10000

100

- 10000
- 10000
- 10-100

□□□□□□

□□ **1**□□□□□□□□

```
import Config

config :omnimessage_smpp,
  api_base_url: "https://smcsc.company.com:8443",
  verify_ssl_peer: true,
  smsc_name: "smpp_prod"

config :omnimessage_smpp, :binds, [
  %{
    name: "att_primary",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.att.com",
    port: 2775,
    system_id: "company_user",
    password: "secure_pass_123",
    tps_limit: 100,
    queue_check_frequency: 1000
  }
]
```

2

```
import Config

config :omnimessage_smpp,
  api_base_url: "https://smcsc.companay.com:8443"

config :omnimessage_smpp, :binds, [
  #
  %{
    name: "att_us",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.att.com",
    port: 2775,
    system_id: "att_username",
    password: "att_password",
    tps_limit: 100,
    queue_check_frequency: 1000
  },

  #
  %{
    name: "vodafone_uk",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.vodafone.co.uk",
    port: 2775,
    system_id: "voda_username",
    password: "voda_password",
    tps_limit: 50,
    queue_check_frequency: 1000
  }
]
```

□□ **3** □□□□□□□□

```
import Config

config :omnimessage_smpp,
  api_base_url: "https://smcsc.com:8443"

# 客户端
config :omnimessage_smpp, :binds, [
  %{
    name: "upstream_carrier",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.carrier.com",
    port: 2775,
    system_id: "my_username",
    password: "my_password",
    tps_limit: 100,
    queue_check_frequency: 1000
  }
]

# 服务器
config :omnimessage_smpp, :server_binds, [
  %{
    name: "partner_alpha",
    system_id: "alpha_corp",
    password: "alpha_secret",
    allowed_bind_types: [:transmitter, :receiver, :transceiver],
    ip_whitelist: ["203.0.113.0/24"],
    tps_limit: 50,
    queue_check_frequency: 1000
  },
  %{
    name: "partner_beta",
    system_id: "beta_inc",
    password: "beta_password",
    allowed_bind_types: [:transceiver],
    ip_whitelist: ["198.51.100.50"],
    tps_limit: 25,
    queue_check_frequency: 2000
  }
]
```

```
# 配置
config :omnimessage_smpp, :listen, %{
  host: "0.0.0.0",
  port: 2775,
  max_connections: 100
}
```

第4章 配置与运行

本章将介绍如何配置和运行 `sinch.smpp`。首先，我们将配置 `IP` 地址和端口，然后介绍如何运行 `IP` 地址和端口。

```

import Config

config :omnimessage_smpp,
  api_base_url: "https://smcsc.com:8443"

config :omnimessage_smpp, :binds, [
  %{
    name: "sinch_1",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.sinch.com",
    port: 2775,
    system_id: "sinch_user",
    password: "sinch_pass",
    tps_limit: 100,
    queue_check_frequency: 1000,

    # 消息队列
    queue: "sinch.smpp",

    # 本地IP/本地端口
    bind_source_ip: "10.20.30.40",
    bind_source_port: 5001
  },
  %{
    name: "sinch_2",
    mode: :client,
    bind_type: :transceiver,
    host: "smpp.sinch.com",
    port: 2775,
    system_id: "sinch_user",
    password: "sinch_pass",
    tps_limit: 100,
    queue_check_frequency: 1000,

    # 与 sinch_1 共享消息队列
    queue: "sinch.smpp",

    # 本地 IP
    bind_source_ip: "10.20.30.40",
    bind_source_port: 5002
  }
]

```

```
}  
]
```

```
sinch.smpd dest_smsc: "sinch.smpd"  
IP SO_REUSEADDR  
queue bind_source_ip bind_source_port
```

~~~~~

~~~~~

~~~~~

```
# Elixir  
/opt/omnimessage-smpd/bin/omnimessage-smpd eval "File.read!  
( 'config/runtime.exs' )"
```

~~~~~

~~~~~

```
#  
sudo -u omnimessage-smpd /opt/omnimessage-smpd/bin/omnimessage-  
smpd console
```

Ctrl+C

---

~~~~~

systemd shell

環境変数	設定項目	設定値
API_BASE_URL	api_base_url	"https://10.17
SMSC_NAME	smsc_name	"smpp_gateway"
SMPP_POLL_INTERVAL	smpp_poll_interval	100
VERIFY_SSL_PEER	verify_ssl_peer	false
CACHE_FLUSH_INTERVAL	cache_flush_interval	10000
CACHE_MAX_RETRY_ATTEMPTS	cache_max_retry_attempts	10
CACHE_BACKOFF_MULTIPLIER	cache_backoff_multiplier	2
MNESIA_STORAGE_TYPE	mnesia_storage_type	disc_copies

systemd の設定

```
sudo systemctl edit omnimessage-smpp
```

```
[Service]
Environment="API_BASE_URL=https://omnimessage-
core.company.com:8443"
Environment="SMSC_NAME=smpp_prod_01"
Environment="VERIFY_SSL_PEER=true"
```

インストール

1. パッケージのインストール

```
sudo chmod 600 /opt/omnimessage-smpp/config/runtime.exs
sudo chown omnimessage-smpp:omnimessage-smpp /opt/omnimessage-smpp/config/runtime.exs
```

2. 設定ファイルの編集

- 12 行目
- 13 行目
- 14 行目

3. IP の設定

- 15 行目 `ip_whitelist`
- 16 行目 `[]`

4. SSL の設定

- 17 行目 `verify_ssl_peer: true`

5. ログの設定

- 18 行目
- 19 行目

README

- [MONITORING.md](#) 監視
- [USAGE.md](#) 使用方法
- [TROUBLESHOOTING.md](#) トラブルシューティング
- [README.md](#) 本 README



□□□□

A

API (□□□□□□□□) □□□□□□□□□□□□□□□□

□□□□ Web UI □□□□□□□□□□□□□□□□□□□□

B

□□ SMPP □□□□□□□□□□□□□□□□

□□ □□□□□□ SMPP □□□□□□□□□□□□□□□□

□□□□ SMPP □□□□□□

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

□□□□ SMPP □□□□□□□□□□□□□□□□ IP □□□□

C

CIDR (□□□□□□□□) □□ IP □□□□□□□□□□□□□□□□ **192.168.1.0/24** □□ 256 □ IP □□□□

□□□□□□ □□ SMPP □□□□□□□□□□ **ESME** □□□□□□ **SMSC**□□□□□□□□□□ SMPP □□□□□□□□□□□□□□□□□□□□

□□□□ SMPP □□□□□□□□□□

- □□□□□□□□□□
- □□□□□□□□

- □ □ □ □ □ □ □ □ □ □ □ □

□ □ □ □

D

SMPP GSM-7 UCS-2

Deliver_SM 0 SMSC 00000000 SMPP PDU 000000000000 ESME 0000000000000000
0000000000000000

□ □ □ □ □

□□□□ (DLR) □□□□□□□□□□□□□□□□

```
deliver_time ████████████████████████ OmniMessage Core ████████████████████████  
████████████████████████████████████████
```

dest smsc 000000000000 SMPP 00000000000000000000000000000000 0000 000000

[illegible]

E

SMPP

ESM □ **SMPP** □□□□□□□□□□□□□□□□

ESME (ESME) は SMPP 経由で SMSC へメッセージを送信する際に、メッセージの送信先 SMSC へ ESME 番号を送信する必要がある。ESME 番号は

☐ 1 ☐ 2 ☐ 4 ☐ 8 ☐ ☐

F

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

透過 SMPP 與 OmniMessage Core 連接後 60 秒內發送訊息失敗時，會自動重試 90 次。OmniMessage 會將訊息放入 `dest_smsc` 或 `queue` 佇列，直到成功為止。

G

透過 `SMPP` 發送訊息。

透過 `SMPP` 接收訊息。

Grafana 與 `Prometheus` 監控系統。

GSM-7 的 SMS 訊息 7 個字元，160 個字元。

H

HTTP/HTTPS 的 Web 服務，`HTTPS` 服務。

I

IP 地址，IP 地址。

ISDN (數位式網路) 數位式網路。

J

(空)

K

透過 `KPI` 發送訊息。

KPI (關鍵績效指標) 關鍵績效指標。

L

📄 Prometheus 配置参数 bind_name="vodafone_uk" 📄

📄 配置 Web UI 📄 Phoenix 📄

M

📄 配置参数

📄 Prometheus 配置

MO (配置) 配置参数

MT (配置) 配置参数

MSISDN (配置) 配置参数

N

NPI (配置) SMPP 配置参数 ISDN

O

📄 配置参数

📄 配置参数

P

PDU (配置) SMPP 配置参数 submit_sm deliver_sm

Prometheus 配置参数 📄 📄

Q

如何 实现 消息 的 持久化

如何 实现 消息 的 持久化 和 消息 的 持久化

如何 实现 消息 的 持久化 SMPP 消息 的 持久化

R

如何 实现 消息 的 持久化 TPS

如何 实现 消息 的 持久化 SMPP 消息 的 持久化 deliver_sm

如何 实现 消息 的 持久化 SMPP 消息

如何 实现 消息 的 持久化 消息 的 持久化

如何 实现 消息 的 持久化 OmniMessage Core 消息 的 持久化 dest_smsg 消息 的 持久化 queue 消息 的 持久化 queue 消息 的 持久化

S

如何 实现 消息 的 持久化 SMPP PDU 消息 的 持久化 submit_sm 消息 的 持久化 submit_sm_resp 消息 的 持久化

如何 实现 消息 的 持久化 ESMEs 消息 的 持久化 SMSC 消息 的 持久化

如何 实现 消息 的 持久化 SMPP 消息

如何 实现 消息 的 持久化 消息 的 持久化 queue 消息 的 持久化 "sinch.smp" 消息 的 持久化

如何 实现 消息 的 持久化 SMS 消息 的 持久化 5-6 消息 的 持久化 deliver_sm 消息 的 持久化 TON 消息 的 持久化

000 00 bind_source_ip 0 bind_source_port 0000000000000000 IP 000/0 TCP
 000000000000 IP 0000000000000000 CONFIGURATION.md

source_smsc **deliver_sm**

SMPP (Short Message Peer-to-Peer) SMS 送信

SMSC (送信元) → SMPP 送信元 → ESME(送信元) → SMS 送信元(送信元) →
→ 送信元 → 送信元 → ESME 送信元 → SMSC

SSL/TLS □□□□□□□□□□□□

Submit_SM SMPP PDU

Submit_SM_Resp □ submit_sm □ SMPP □□□□□□□□□□

□□ **ID** □□ SMPP □□□□□□□□

T

□ □ □ □ □ □ □ □ □ □ □ □ □ □

TON () SMPP

TPS ()

□□□□ □□□□□□□□□□□□□□ SMPP □□□□□□□□□□□□□□

SMPP submit_sm

[illegible]

U

UCS-2 □ SMS □ 16 □ Unicode □□□□□□□□□□ 70 □□□□

[illegible]

V

○○○ ○○○○○○○○○○○○○○○○○○○

W

Web ○○○ ○○○○○○○○○○○○○○○○○○○

○○○ ○ IP ○○○○○○○○○○○○○

X

(○○○)

Y

(○○○)

Z

(○○○)

□□□□□□□□

□□□	□□
API	□□□□□□□□
CIDR	□□□□□□
DLR	□□□□
ESME	□□□□□□□□
GSM	□□□□□□□□
HTTP	□□□□□□□□
HTTPS	□□□□□□□□□□
IP	□□□□□□
ISDN	□□□□□□□□
KPI	□□□□□□□□
MO	□□□□□
MSISDN	□□□□□□□□□□□□
MT	□□□□□
NPI	□□□□□□□□
PDU	□□□□□□□□
SMPP	□□□□□□□□
SMSC	□□□□□□□□

項目	説明
SMS	短メッセージサービス
SSL	セキュア・ソケット・レイヤー
TLS	トランスポート・レイヤー・セキュリティ
TON	トーン
TPS	トランザクション・プログラム・セクション
UCS	ユーザー・コンタクト・システム
UI	ユーザー・インターフェース
URL	ユニーク・リソース・Locator

目次

- **README.md** - 概要
 - **CONFIGURATION.md** - 設定
 - **USAGE.md** - 使い方
 - **MONITORING.md** - 監視
 - **TROUBLESHOOTING.md** - トラブルシューティング
-

SMPP 架构图

简介

SMPP 架构图展示了 SMPP 客户端 API 与 Mnesia 数据库的交互流程。API 通过 Mnesia 数据库与后端 API 进行交互。

组件

- SMPP 客户端 - 通过 API 与 SMPP 服务器交互
- Mnesia 数据库 - 存储 Mnesia 的 `:disc_copies` 数据
- 缓存 - 用于存储 SMPP 数据
- SMPP 服务器 - 接收/发送 SMPP 消息
- 队列 - 用于存储 SMPP 消息的 FIFO 队列
- 缓存 - 用于存储 SMPP 消息
- 缓存 - 用于存储 SMPP 消息的 LiveView 视图
- **Prometheus** 指标 - 用于监控 SMPP 系统

流程

提交

提交流程

SMPP Client → submit_sm → Gateway Server

↓

Cache in Mnesia (immediate response)

↓

CacheFlushWorker (background)

↓

Backend API

□□□□

SMPP Client → submit_sm → Gateway Server

↓

Backend API (direct)

□□

1. MessageCache □□

- □□□□□□
- □□□□
- □□ LiveView □□□□□□□□

2. CacheFlushWorker

- □□□□□□□□□□ GenServer
- □□□□□□□□□□
- □□□□□□
- □□□□□□□□□□

3. Mnesia □ (:smpp_message_cache)

- □□ :disc_copies □□□□□
- □ bind_name□next_retry_at □ status □□
- □□□□□□□□□□

□□

□□□□

□□ config/runtime.exs:

```

config :omnimessage_smpp,
  # 配置缓存
  cache_flush_interval: 10_000,

  # 配置重试
  cache_max_retry_attempts: 10,

  # 配置退避
  cache_backoff_multiplier: 2,

  # Mnesia 存储 (:disc_copies 或 :ram_copies)
  mnesia_storage_type: :disc_copies

```

配置 SMPP

配置 SMPP 连接参数

```

config :omnimessage_smpp, :binds, [
  %{
    name: "my_smpp_bind",
    mode: :server,
    system_id: "username",
    password: "password",

    # 配置缓存
    cache_enabled: true,           # 是否启用缓存
    cache_max_size: 10_000,       # 缓存最大大小
    cache_retry_interval: 60     # 缓存重试间隔
  }
]

```

环境变量

```

# 配置缓存
CACHE_FLUSH_INTERVAL=10000           # 是否启用缓存
CACHE_MAX_RETRY_ATTEMPTS=10          # 缓存最大大小
CACHE_BACKOFF_MULTIPLIER=2           # 缓存重试间隔
MNESIA_STORAGE_TYPE=disc_copies      # Mnesia 存储

```

メッセージ

キュー

メッセージキューの仕組み

メッセージ: 60 件

メッセージ: 2

メッセージ 0: 60s (1 件)

メッセージ 1: 120s (2 件)

メッセージ 2: 240s (4 件)

メッセージ 3: 480s (8 件)

メッセージ 4: 960s (16 件)

メッセージ 5: 1920s (32 件)

...

メッセージ 9: 30,720s (8.5 日)

メッセージ

メッセージ 10 の状態は `failed_permanent` である

- メッセージの寿命が尽きた
- メッセージが破損した
- メッセージが“メッセージ”として扱えない
- メッセージが破損した

メッセージ

```
:pending → :delivering → SUCCESS (メッセージ)
               → FAILURE → :pending (メッセージ)
                           → :failed_permanent (メッセージ)
```



LiveView 簡介

透過網址 `http://your-server:4000/smpp` → “網頁”顯示

顯示內容

- 狀態 - 連線狀態
- 訊息 - 訊息狀態
- 錯誤 - 錯誤訊息

顯示內容

- 訊息
- 訊息狀態
- 訊息 / 訊息
- 訊息
- 訊息狀態

Prometheus 簡介

```
# 訊息狀態
smpp_cache_size{bind_name="my_bind",mode="server"} 42

# 訊息狀態
smpp_cache_delivered_total{bind_name="my_bind"} 1234

# 訊息狀態
smpp_cache_retry_total{bind_name="my_bind"} 56

# 訊息狀態
smpp_cache_permanent_failures_total{bind_name="my_bind"} 2

# 訊息狀態
smpp_cache_overflow_total{bind_name="my_bind"} 0
```

□□□□

```
INFO  Message -123456789 cached for my_smpp_bind
INFO  Successfully delivered cached message -123456789, API ID:
99999
WARN  Failed to deliver message -123456789 (retry 3/10), next
retry at 2026-02-01 12:34:56Z
ERROR Message -123456789 exceeded max retries (10), marking as
failed_permanent
WARN  Cache overflow for my_smpp_bind: deleted oldest message
```

□□

□□/□□□□

□□ **LiveView UI**

1. □□□ `http://your-server:4000/smpp`
2. □□“□□□□□”□“□□□□□”□□□
3. □□□□
4. □□“□□□□□”□□□
5. □□□□

□□ **IEx** □□□

```
# 绑定成功
SmsC.SMPPConfig.update_server_peer("my_bind", "username",
"password",
  cache_enabled: true,
  cache_max_size: 10_000,
  cache_retry_interval: 60
)

# 绑定失败
SmsC.SMPPConfig.update_server_peer("my_bind", "username",
"password",
  cache_enabled: false
)
```

消息缓存

```
# 查看缓存
SmsC.SMPP.MessageCache.get_cache_summary()
# => %{total_cached: 42, pending: 40, failed: 2}

# 查看指定绑定的缓存
SmsC.SMPP.MessageCache.get_cache_by_bind()
# => [
#   %{bind_name: "bind1", total: 30, pending: 28, failed: 2,
#     max_size: 10_000},
#   %{bind_name: "bind2", total: 12, pending: 12, failed: 0,
#     max_size: 10_000}
# ]

# 查看指定绑定的缓存消息数量
SmsC.SMPP.MessageCache.count_cached_messages("my_bind")
# => 42
```

消息队列

消息队列

```
# 删除失败消息
{:atomic, failed_messages} = :mnesia.transaction(fn ->
  :mnesia.match_object({:smpp_message_cache, :_, "my_bind", :_,
    :_, :_, :_, :_, :_, :failed_permanent})
end)

# 遍历
Enum.each(failed_messages, fn {_, {bind_name, msg_id}, _, _, _, _,
  _, _, _, _} ->
  SmsC.SMPP.MessageCache.delete_cache_record(bind_name, msg_id)
end)
```

删除失败消息

```
# 将 failed_permanent 消息改为 pending
SmsC.SMPP.MessageCache.update_cache_record("my_bind", -123456, %{
  status: :pending,
  retry_count: 0,
  next_retry_at: DateTime.utc_now(),
  last_error: nil
})
```

配置

配置 / 配置

配置: `cache_overflow_total` 配置

配置: 配置

配置:

1. 配置 `cache_max_size`
2. 配置 API 配置 API 配置
3. 配置
4. 配置

□□□□□□

□□: □□□□ :pending □□

□□□□:

1. API □□□

- □□ API □□□
- □□□□ API □□
- □□□□□□

2. next_retry_at □□□

- □□□□□□ next_retry_at ⚠️⚠️□□
- □□□□□□□□

3. □□□□□□□□

```
# □□□□□□□□□□  
Supervisor.which_children(SmsC.SMPP.Supervisor)
```

4. □□□□□

- □□□□□□□□ cache_enabled: true

□□□□□□

□□: □□□□□□□□ retry_count □

□□:

```
# 消息队列实现
{:atomic, messages} = :mnesia.transaction(fn ->
  :mnesia.match_object({:smpp_message_cache, :_, "my_bind", :_,
    :_, :_, :_, :_, :_, :_})
end)

high_retry = Enum.filter(messages, fn {_ , _ , _ , _ , _ , _ ,
  retry_count, _ , _ , _} ->
  retry_count >= 5
end)

# 记录最后错误
Enum.each(high_retry, fn {_ , _ , _ , msg_id, _ , _ , retry_count, _ ,
  last_error, _} ->
  IO.puts("Message #{msg_id}: #{retry_count} retries, error: #
    {inspect(last_error)}")
end)
```

Mnesia 简介

简介: 消息队列

简介 **Mnesia** 简介:

```
ls -lh Mnesia.*
du -sh Mnesia.*
```

简介:

1. 消息队列实现
2. 消息队列 `cache_max_size`
3. 消息队列实现 FIFO 队列

メモリ

メモリ

- メモリ容量 500-1000 MB程度
- 10,000 項目 $\approx$ 5-10 MB 程度
- `:disc_copies` 項目

CPU

- プロセッサ 10 程度
- プロセッサ 100 程度
- プロセッサ 10 程度 API 程度

I/O

- `:disc_copies` 項目
- `>1000` 項目/項目
 - `:ram_copies` 項目
 - 項目
 - 項目

メモリ

項目	cache_max_size	cache_flush_interval
項目 (<100 項目/項目)	10,000	10,000ms
項目 (100-500 項目/項目)	50,000	5,000ms
項目 (>500 項目/項目)	100,000	3,000ms

□□□□

□□□□

1. Mnesia □□□□□□ □ `:disc_copies` □
2. □□□□□□□□
3. □□□□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□

1. □□□□□□□□□□□□□□□□
2. □□□□ `cache_enabled: false` □ `cache_max_size: 10,000` □
`cache_retry_interval: 60`
3. □□□□□
4. □□□□□□□□□□□□

API □□□□

1. □□□□□□□□□□□□□□
2. □ API □□□□□□□□□□□□□□
3. □□□□□□□□□□FIFO□
4. □□□□□□□□□□ API □□

□□□□

1. □□□□□□ - □□□□□□□□□□□□
2. □□□□ - □□□□ `cache_permanent_failures_total` □ `cache_overflow_total` □□□□
3. □□□□□□ - □□□□□□□□□□□□□□□□ `cache_max_size`
4. □□□□□□ - □□□□□□□□□□□□□□□□□□□
5. □□□□□□ - □□ API □□□□□□□□□□
6. □□□□□□ - □□ API □□□□□□□□□□□□
7. □□□□□□ - □□□□□□□□ `mnesia_storage_type: disc_copies`



- Four small orange-outlined squares arranged horizontally.
- Five small orange-outlined squares arranged horizontally.
- Four small orange-outlined squares arranged horizontally.

如何集成

SMPP 指标

简介

SMPP 指标 Prometheus 指标名称

指标名称 OmniMessage Core **OmniMessage** 指标名称

1. **SMPP** 指标 - 指标名称
2. **OmniMessage API** 指标 - 指标名称

配置

URL: `http://your-server:4000/metrics`

指标: Prometheus 指标

指标: 指标名称 localhost 指标名称

命令

```
curl http://localhost:4000/metrics
```

指标

指标名称 `smpp_` 指标名称

□□□□□

omnmessage_smpp_license_status

□□: Gauge

□□: □□□□□□□

□:

- 1 = □□□□□
- 0 = □□/□□□□□

□□: □

□□:

```
omnmessage_smpp_license_status 1
```

□□:

- □□□0□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□SMPP□□□□□□
- Web UI□□□□□□□□□□□□

□□□□: omnmessage_smpp

□□:

- □□□□□□□ license_status == 0 □□□□□□□□□□□□
- SMPP□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□

□□□□:

```
- alert: SMPP_License_Invalid
  expr: omnimessage_smpp_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "SMPP许可证过期"
    description: "许可证过期 - 请及时续费"
```

配置项

smpp_connection_status

类型: Gauge

单位: SMPP许可证数量

值:

- 1 = 许可证已过期
- 0 = 许可证有效

配置:

- bind_name - 绑定名称 "vodafone_uk"
- mode - 模式 "client" 或 "server"
- host - 主机地址
- port - 端口
- bind_type - SMPP绑定类型
- system_id - 系统ID

示例:

```
smpp_connection_status{bind_name="vodafone_uk",mode="client",host="sn
1
```

备注:

- 0000000000000000
- 0000000000000000
- 00000000

000000

smpp_messages_sent_total

00: Counter

00: 00SMPP0000000000

00: 00

00: 0connection_status00

00:

```
smpp_messages_sent_total{bind_name="vodafone_uk",mode="client",...}  
150234
```

00:

- 0000000000/00
- 0000/0000
- 0000000000000000

smpp_messages_received_total

00: Counter

00: 00SMPP0000000000

00: 00

00: 0connection_status00

00:

```
smpp_messages_received_total{bind_name="partner_acme",mode="server",...
45123
```

📄:

- 📄📄📄📄
- 📄📄📄📄MO📄📄
- 📄📄📄📄📄📄

📄📄📄

smpp_delivery_failures_total

📄📄: Counter

📄📄: 📄📄📄📄📄📄

📄📄: 📄📄

📄📄: 📄connection_status📄📄

📄📄:

```
smpp_delivery_failures_total{bind_name="vodafone_uk",mode="client",...
234
```

📄📄:

- 📄📄📄📄📄
- 📄📄📄📄📄📄
- 📄📄📄📄

📄📄📄📄:

```
success_rate = (messages_sent - delivery_failures) / messages_sent
* 100
```

概要

smpp_bind_success_total

単位: Counter

項目: 結合成功回数

単位: 回

説明:

```
smpp_bind_success_total{bind_name="vodafone_uk",...} 45
```

タグ:

- 結合成功
- 結合失敗

smpp_bind_failures_total

単位: Counter

項目: 結合失敗回数

単位: 回

説明:

```
smpp_bind_failures_total{bind_name="vodafone_uk",...} 3
```

タグ:

- 結合成功
- 結合失敗
- 結合失敗

概要

smpp_connection_attempts_total

Counter

Counter

Counter

Counter

```
smpp_connection_attempts_total{bind_name="vodafone_uk",...} 48
```

Counter

- Counter
- Counter

smpp_disconnection_total

Counter

Counter

Counter

Counter

```
smpp_disconnection_total{bind_name="vodafone_uk",...} 3
```

Counter

- Counter
- Counter
- Counter

Counter

smpp_enquire_link_sent_total

Counter

Counter PDU

Counter PDU

📄: 📄connection_status📄

📄:

```
smpp_enquire_link_sent_total{bind_name="vodafone_uk",mode="client",...
1440
```

📄:

- 📄📄📄📄📄📄
- 📄📄📄📄📄📄📄📄📄📄

smpp_enquire_link_received_total

📄: Counter

📄: 📄📄📄📄📄📄📄📄📄PDU📄

📄: PDU

📄: 📄connection_status📄

📄:

```
smpp_enquire_link_received_total{bind_name="vodafone_uk",mode="client
1438
```

📄:

- 📄📄📄📄📄📄📄 >> 📄📄
- 📄📄📄📄📄📄📄📄

📄📄📄📄📄📄

smpp_uptime_seconds

📄: Gauge

📄: SMPP📄📄📄📄📄📄📄📄📄📄

📄: 📄

11

```
smpp_uptime_seconds{bind_name="vodafone_uk",...} 86400
```

11

- 
- 
- 

666666

MESSAGE_CACHE.md

smpp_cache_size

□□: Gauge

□□ : □□□□□□□□□□□□□□□□

□□ : □□

11

- `bind_name` - 字符串
- `mode` - 字符串“client”或“server”

11

```
smpp_cache_size{bind_name="partner_acme",mode="server"} 42
```

11

- 00000000
- 000 cache_max_size 00000000

smpp_cache_delivered_total

Counter

API

```
smpp_cache_delivered_total{bind_name="partner_acme"} 1234
```

smpp_cache_retry_total

Counter

```
smpp_cache_retry_total{bind_name="partner_acme"} 56
```

smpp_cache_permanent_failures_total

Counter

```
smpp_cache_permanent_failures_total{bind_name="partner_acme"} 2
```

- > 0

smpp_cache_overflow_total

Counter

👤:

```
smpp_cache_overflow_total{bind_name="partner_acme"} 0
```

👤:

- 📄📄📄📄📄📄📄📄API📄📄📄📄

📄📄📄📄**DLR**📄📄

📄📄📄📄📄📄`deliver_receipts: true`📄📄📄📄📄📄📄📄📄📄
[CONFIGURATION.md](#)📄[USAGE.md](#)📄📄📄📄📄

smpp_dlr_tracked_total

👤: Counter

👤: 📄📄📄📄📄📄📄📄📄📄

👤: 📄

👤:

- `bind_name` - 📄📄📄
- `mode` - 📄📄📄📄“client”📄“server”📄

👤:

```
smpp_dlr_tracked_total{bind_name="masorange",mode="server"} 10234
```

smpp_dlr_sent_total

👤: Counter

👤: 📄📄📄📄📄📄📄📄📄📄

👤: 📄

👤:

```
smpd_dlr_sent_total{bind_name="masorange",mode="server"} 10180
```

22

- `smpp_dlr_tracked_total` `XXXXXXXXXXXX`

smpp_dlr_dropped_total

□□: Counter

□□: □□□□□□□□□□□□□□**◆◆** □□□□□□□□□□**24** □□□□□□□□□□□□□□

□□ : □□

□□:

```
smpp_dlr_dropped_total{bind_name="masorange",mode="server"} 12
```

11

- [illegible]

smpp_dlr_pending

□□: Gauge

□□ : □□□□□□□□□□□□□□□□

□□ : □□

22

```
smpp_dlr_pending{bind_name="masorange",mode="server"} 37
```

11

- 
- 

□□□□

```
# 5분간 평균값
sum(rate(smpp_dlr_sent_total[5m]))
  / sum(rate(smpp_dlr_tracked_total[5m]))

# 분당 값
rate(smpp_dlr_dropped_total[5m])

# 분당 대기 중인 값
smpp_dlr_pending
```

OmniMessage API

이 섹션에서는 SMPP와 관련된 **OmniMessage API**에 대해 설명합니다.

이 **OmniMessage** 메트릭은 다음과 같습니다.

- `omnmessage_api_requests_total` - 전체 API 요청 횟수
- `omnmessage_api_request_duration_seconds` - API 요청 소요 시간
- `omnmessage_queue_depth` - OmniMessage 대기 중인 메시지 수

이 메트릭은 다음과 같이 설정됩니다.

이 메트릭은 API 요청에 대해 다음과 같이 설정됩니다.

- "api.*connection refused" - OmniMessage 연결 거부
- "api.*timeout" - OmniMessage 시간 초과
- "api.*http 503" - OmniMessage 503 에러
- "api.*parse error" - OmniMessage 파싱 오류

Prometheus

이 섹션에서는 Prometheus에 대해 설명합니다.

이 메트릭은 `/etc/prometheus/prometheus.yml`에 설정됩니다.

```

scrape_configs:
  - job_name: 'omnimessage-smpp'
    scrape_interval: 15s
    static_configs:
      - targets: ['your-server:4000']
        labels:
          environment: 'production'
          service: 'omnimessage-smpp'

```

□□□□

```

scrape_configs:
  - job_name: 'omnimessage-smpp-instances'
    scrape_interval: 15s
    static_configs:
      - targets:
          - 'smpp-gw-1:4000'
          - 'smpp-gw-2:4000'
          - 'smpp-gw-3:4000'
        labels:
          environment: 'production'

```

□□□□

□□□□□□□□□□

```

scrape_configs:
  - job_name: 'omnimessage-smpp-instances'
    file_sd_configs:
      - files:
          - '/etc/prometheus/targets/smpp-*.json'

```

□□/etc/prometheus/targets/smpp-production.json□

```
[
  {
    "targets": ["smpp-gw-1:4000", "smpp-gw-2:4000"],
    "labels": {
      "environment": "production",
      "datacenter": "us-east"
    }
  }
]
```

Grafana

:

```
smpp_connection_status{job="omnimessage-smpp"}
```

: Stat

:

- : < 1
- : == 1

:

```
rate(smpp_messages_sent_total{job="omnimessage-smpp"}[5m])
```

: Graph

: /

: {{bind_name}}

报警规则

规则:

```
100 * (1 - (
    rate(smpp_delivery_failures_total{job="omnimessage-smpp"}[5m])
    /
    rate(smpp_messages_sent_total{job="omnimessage-smpp"}[5m])
))
```

类型: Gauge

范围: 0-100

阈值:

- 阈值: < 95%
- 阈值: 95-98%
- 阈值: > 98%

报警规则

规则:

```
smpp_uptime_seconds{job="omnimessage-smpp"} / 3600
```

类型: Stat

范围: 0

报警规则

Prometheus报警规则

规则文件: `/etc/prometheus/rules/smpp-alerts.yml`

```

groups:
- name: smpp_gateway
  interval: 30s
  rules:
    # 连接
    - alert: SMPPConnectionDown
      expr: smpp_connection_status == 0
      for: 2m
      labels:
        severity: critical
      annotations:
        summary: "SMPP连接{{ $labels.bind_name }}异常"
        description: "连接{{ $labels.bind_name }}已断开2分钟"

    # 失败率
    - alert: SMPPHighFailureRate
      expr: |
        (
          rate(smpp_delivery_failures_total[5m])
          /
          rate(smpp_messages_sent_total[5m])
        ) > 0.05
      for: 5m
      labels:
        severity: warning
      annotations:
        summary: "连接{{ $labels.bind_name }}失败率过高"
        description: "连接{{ $labels.bind_name }}失败率{{ $value | humanizePercentage }}%持续5分钟"

    # 绑定失败
    - alert: SMPPBindFailures
      expr: increase(smpp_bind_failures_total[10m]) > 3
      labels:
        severity: warning
      annotations:
        summary: "连接{{ $labels.bind_name }}绑定失败"
        description: "连接{{ $labels.bind_name }}在10分钟内发生{{ $value }}次绑定失败"

    # 无流量
    - alert: SMPPNoTraffic
      expr: rate(smpp_messages_sent_total[10m]) == 0

```

```

    for: 30m
    labels:
      severity: warning
    annotations:
      summary: "⚠️ {{ $labels.bind_name }} 连接异常"
      description: "{{ $labels.bind_name }} 30分钟内连接异常"

# 连接异常
- alert: SMPPFrequentDisconnections
  expr: increase(smpp_disconnection_total[1h]) > 5
  labels:
    severity: warning
  annotations:
    summary: "⚠️ {{ $labels.bind_name }} 连接异常"
    description: "{{ $labels.bind_name }} 连接异常, 当前值: {{ $value }}"

# OmniMessage API 连接异常
- alert: OmniMessageAPIUnreachable
  expr: |
    count(count_over_time({job="omnmessage-smpp"} |=
"api.*connection refused"[5m])) > 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "OmniMessage API 连接异常"
    description: "SMPP 连接异常, OmniMessage API 连接异常, API_BASE_URL: {{ $labels.api_base_url }}"

# OmniMessage API 超时
- alert: OmniMessageAPITimeout
  expr: |
    count(count_over_time({job="omnmessage-smpp"} |=
"api.*timeout"[5m])) > 5
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: "OmniMessage API 超时"
    description: "OmniMessage API 超时, 当前值: {{ $value }}"

# 无消息流
- alert: NoMessageFlow

```

```
    expr: rate(smpp_messages_sent_total[10m]) == 0 and
rate(smpp_messages_received_total[10m]) == 0
    for: 30m
    labels:
      severity: warning
    annotations:
      summary: "短信发送/接收 - 0条OmniMessage"
      description: "30秒内短信发送/接收量为0条OmniMessage API接口异常"
```

在 prometheus.yml 中添加

```
rule_files:
  - '/etc/prometheus/rules/smpp-alerts.yml'
```

Web 访问

Web UI 访问 Prometheus

地址

URL: `https://your-server:8087`

功能

支持: SMPP → 短信

支持:

- 短信发送
- 短信接收
- 短信状态
- 短信发送/接收
- 5G 短信

支持:

- 
- 
- 

□ □ □ □ □ □

- 0000: 0000000000000000
- 000000: 0000000000000000/000000
- 0000000: 0000000000000000/0000000
- 00000000: 00000000000000000000000000000000

□ □ □ □

--	--	--	--

□□□□:

```
# 確認
sudo journalctl -u omnimessage-smpp -f

# 100行
sudo journalctl -u omnimessage-smpp -n 100

# 1時間前以降
sudo journalctl -u omnimessage-smpp --since "1 hour ago"

# エラー
sudo journalctl -u omnimessage-smpp -p err
```

Web UI

Web UI

:

-
-
-
-
-

- 目標値: 100%
- 単位: 割合
- 計算式: $\frac{\text{成功メッセージ数}}{\text{送信メッセージ数}} \times 100$
- 入力/出力: 送信メッセージ数, 成功メッセージ数
- 備考: 送信失敗メッセージ数は除外

送信遅延率KPI

概要

目的: 送信遅延率を監視

```
avg_over_time(smpp_connection_status[24h]) * 100
```

目標: > 99.9%

計算式

単位: 割合

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

備考: 送信失敗メッセージは除外

計算式

単位: 割合

```
100 * (1 - rate(smpp_delivery_failures_total[5m]) /  
rate(smpp_messages_sent_total[5m]))
```

目標: > 98%

📌 📌 📌 📌

📌: 📌 📌 📌 📌 📌

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

📌: < 10 📌 📌 📌 📌 📌 📌 📌

📌 📌 📌 📌 📌

1. 📌 📌

- 📌 📌 📌 Prometheus 📌
- 📌 PagerDuty/OpsGenie 📌 24/7 📌
- 📌 📌 📌

2. 📌 📌 📌

- 📌 📌 📌 Grafana 📌
- 📌 📌 📌 📌 📌 📌
- 📌 📌 📌 📌

3. 📌 📌

- 📌 📌 📌
- 📌 📌 📌 📌
- 📌 📌 📌

4. 📌 📌

- 📌 📌 📌 📌
- 📌 📌 TPS 📌
- 📌 📌 📌 / 📌

5. 配置

- 配置API
 - 配置参数
 - 配置
-

配置参数

配置

配置: `smpp_connection_status`

- 0 = 配置
- 配置 = 配置

配置

配置: `smpp_delivery_failures_total`

- 配置 = 配置
- 配置 = 配置

配置

配置: `smpp_messages_sent_total` 配置

- 配置 = 配置 TPS 配置
- 配置 API 配置

配置

配置: `smpp_bind_failures_total`

- 配置 = 配置
- 配置 system_id 配置

□□□□

- **CONFIGURATION.md** - □□□□□□
 - **USAGE.md** - □□□□
 - **TROUBLESHOOTING.md** - □□□□
 - **README.md** - □□□□□□□□
-

Regex	Match	Result
<code>^61[45]\d{8}\$</code>	614/615 11	61400000001, 61512345678
<code>^(MyBrand AlertSvc)\$</code>		MyBrand, AlertSvc
<code>(?i)^mybrand\$</code>		MyBrand, mybrand, MYBRAND
<code>^(?!Test).+</code>	Test	MyBrand, TestBrand

Regex

Regex is a tool for matching patterns in text.

Regex is a tool for matching patterns in text.

Input	Regex	Result
MyBrand	<code>^MyBrand\$</code>	Match
61412345678	<code>^614</code>	Match
+61400000001	<code>^\+61400000001\$</code>	Match
OtherBrand		Match
61500000001		Match

Regex

Regex is a tool for matching patterns in text.

項目	値
PDU	submit_sm_resp
送信先	0x00000000A
送信元	ESME_RINVSRCADR送信元アドレス
送信ID	

送信元アドレスがホワイトリストに登録されていないため拒否されました

```
SMPP Server: Rejected submit_sm from partner_acme - source_addr 'UnauthorisedBrand' not in whitelist
```

送信元アドレスがホワイトリストに登録されていないため拒否されました

??

Web UI

- 1. 送信元 SMPP > Server Peers
- 2. 送信元アドレス Edit 送信元 Add New Server Peer
- 3. 送信元 Source Address Whitelist 送信元IPアドレス
- 4. 送信元アドレスを登録する

```
^MyBrand$,^614,^\+61400000001$
```

- 5. 送信元 Save

送信元アドレスを登録する submit_sm PDU

?? 送信元

送信元 source_address_whitelist 送信元 runtime.exs 送信元

```
config :omnimessage_smpp, :server_binds, [
  %{
    name: "partner_acme",
    system_id: "acme_corp",
    password: "secure_password",
    allowed_bind_types: [:transmitter, :receiver, :transceiver],
    ip_whitelist: ["203.0.113.0/24"],
    source_address_whitelist: ["^MyBrand$", "^614",
    "^\\+61400000001$"],
    tps_limit: 50,
    queue_check_frequency: 1000
  }
]
```

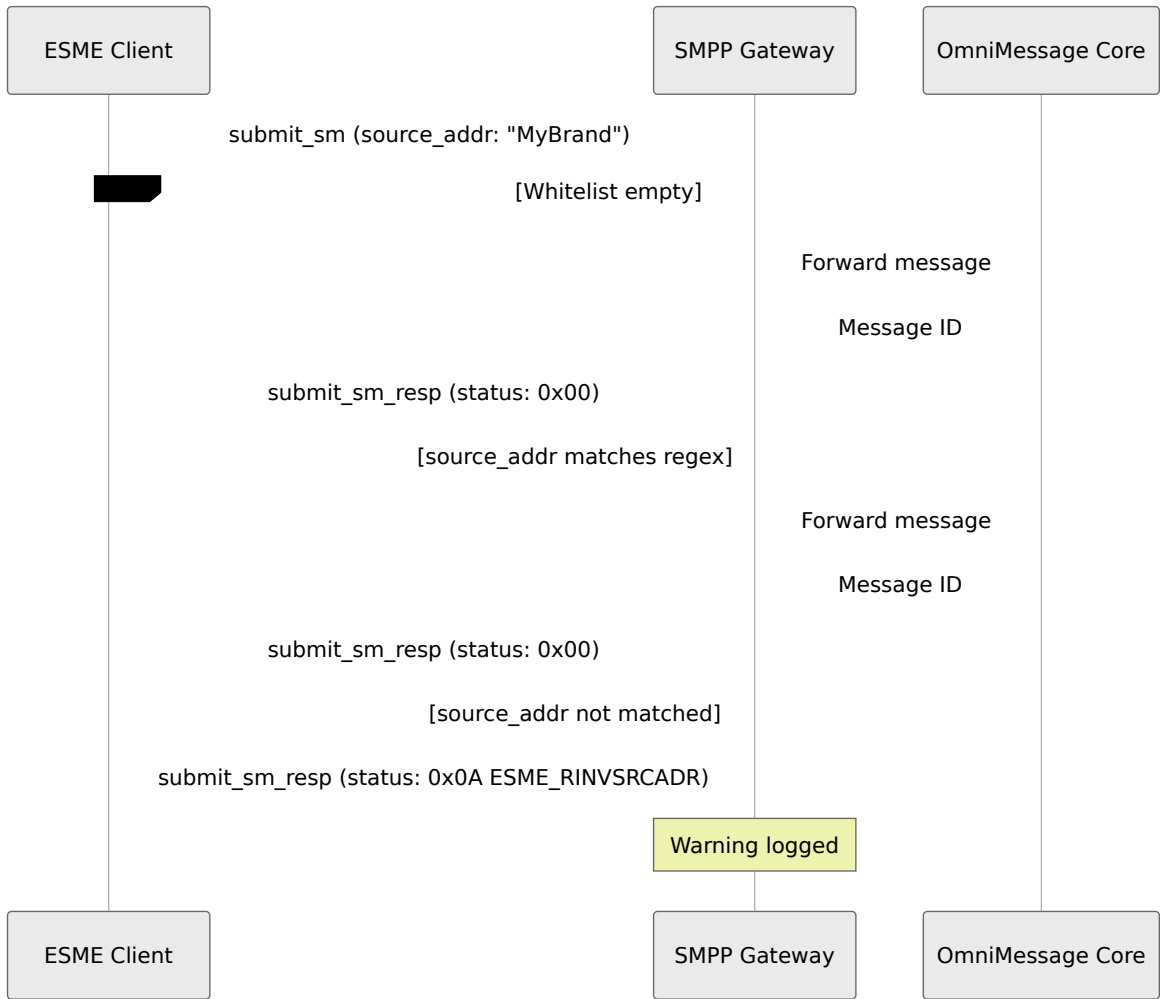
Table

Field	Type	Default	Options	Description
source_address_whitelist	Array	[""]	[""]	Array of source addresses that are allowed to connect to the server. The array can contain regular expressions or IP addresses. If empty, all source addresses are allowed.

Table

Array of source addresses that are allowed to connect to the server. The array can contain regular expressions or IP addresses. If empty, all source addresses are allowed.

□□□□



□□

□□□□□□□□

□□□□□□ID□AcmeCorp□□□□

^AcmeCorp\$

□□□□□□□□□□

□□□□□□□□□□□□614□□□□

^614

□□□□□□□□

□□□□□614□615□□□□□□11□□□□□

^61[45]\d{8}\$

□□□□□□□□

□□□□□□□□□□□□

^AcmeCorp\$,^614,^\+61290000001\$

□□□□□□□□□□

□□□□□□□□□□□□

(?i)^acmecorp\$

□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□

□□□□□□□□□□**ESME_RINVSRCADR**

□□□□□□□□ submit_sm_resp □□□□□□ 0x0A □

□□□□□

- 000000000000000000000000
- 000000000000000000000000
- 000000^/\$0000000000000000
- 000000000000000000000000 (?i) 0000000000000000

000000

1. 00Web UI0000000000000000
2. 000000000000000000000000
3. 00Elixir0000000000000000 Regex.match?(~r/^pattern\$/, "address")
4. 0000000000000000
5. 0000000000000000000000 +000000 \+

00000000

000000000000000000000000

000000

- 0000000000000000
- 0000000000000000000000 614 00 X614Y
- ESME0000000000000000
- 0000000000000000000000

000000

1. 00Web UI0000000000000000
2. 0000000000000000000000 ^0 \$
3. 00ESME0000000000000000000000
4. 0000000000000000000000

000000

- 00000 - 0000000000000000
- 00000 - 00SMPP

- 0000 - 00000000

OmniMessage Core

インストール

インストール / 確認

OmniMessage Core 8443 :eaddrinuse

確認

- systemctl show <unit> -p NRestarts
-

```
Running ApiEx.Endpoint with Bandit ... at https failed, port
8443 already in use
... failed to start child: ApiEx.Supervisor ... ** (EXIT)
:eaddrinuse
```

- curl -sk https://<host>:8443/api/messages 404

SMPP OmniMessage Core 8443 api_ex API

OAM/API Core Core 8443

/etc/omnmessage_smpp/runtime.exs api_ex

api_base_url Core API

```
# /etc/omnmessage_smpp/runtime.exs
config :api_ex,
  api:
    Application.get_env(:api_ex, :api, %{})
    |> Map.put(:port, 8444)
    |> Map.put(:listen_ip, "0.0.0.0")
```

インストール

```
sudo systemctl restart omnimessage-smpp
sudo systemctl restart omnimessage # Core 8443
```

Core 8443

```
ss -lntp | grep -E ':8443|:8444'
curl -sk -o /dev/null -w '%{http_code}\n'
https://127.0.0.1:8443/api/messages # 404
```

CONFIGURATION.md → OAM/API

OmniMessage

SMPP OmniMessage Core OmniMessage

OmniMessage

-
-
- API
- "HTTP 503"

1. OmniMessage

```
#
curl -k -v https://omnimessage-
core.example.com:8443/api/system/health

#
ssh gateway-server 'curl -k https://omnimessage-
core.example.com:8443/api/system/health'
```

2. 確認 API URL

```
# 確認
grep -Al 'api_base_url' /opt/omnimessage-smpp/config/runtime.exs

# 確認接続
ping omnimessage-core.example.com
nc -zv omnimessage-core.example.com 8443
```

3. 確認 API ログ

```
# API ログを確認
sudo journalctl -u omnimessage-smpp -f | grep -i
'api\|omnimessage\|connect'

# エラーログを確認
sudo journalctl -u omnimessage-smpp -n 200 | grep -i error
```

確認

OmniMessage 確認

- OmniMessage 確認
- 確認
- 確認 `SMPP_POLL_INTERVAL`
- OmniMessage 確認

OmniMessage 確認

- 確認 HTTPS
- DNS 確認 `nslookup omnimessage-core.example.com`
- 確認 `traceroute omnimessage-core.example.com`
- SSL 確認 HTTPS

API URL 確認

- `/opt/omnimessage-smpp/config/runtime.exs`
- `api_base_url` 確認 HTTPS

3. 確認 `sudo systemctl restart omnimessage-smpp`

確認

確認

確認

- 確認“確認”確認
- 確認確認
- 確認確認

確認確認

1. 確認

確認

```
# 確認 DNS 確認
nslookup smpp.carrier.com

# 確認
ping -c 3 smpp.carrier.com

# 確認
telnet smpp.carrier.com 2775
# 確認
nc -zv smpp.carrier.com 2775
```

確認

- 確認 DNS 確認確認 IP 確認確認
- 確認 ping 確認確認確認確認
- 確認確認確認確認確認確認

2. 確認

📌

- 📄 “📄” “📄”
- Web UI → SMPP → 📄 system_id 📄

📌

- 📄
- 📄
- 📄

3. IP 📄

📌

- 📄
- 📄 IP

📌

- 📄 IP

```
curl ifconfig.me
```

- 📄 IP 📄
- 📄 IP 📄 IP

4. 📄

📌

```
# 📄
sudo iptables -L -n | grep 2775

# 📄 UFW📄Ubuntu/Debian📄
sudo ufw status | grep 2775

# 📄 firewalld📄RHEL/CentOS📄
sudo firewall-cmd --list-ports | grep 2775
```

□ □ □ □ □

```
# Ubuntu/Debian
sudo ufw allow out 2775/tcp

# RHEL/CentOS
sudo firewall-cmd --permanent --add-port=2775/tcp
sudo firewall-cmd --reload
```

5. □□□□□/□□□□□

bind_source_ip bind_source_port
CONFIGURATION.md

111

- `bind_source_ip` `bind_source_port`

□ □ □ □ □

- `bind_source_ip` `ip addr show`
- `bind_source_ip` 选项——指定源 IP 地址
- 选项 `SO_REUSEADDR` 选项 `TIME_WAIT` 选项
`sudo ss -ltnp | grep <port>`

□ □ □ □ □ □

111

- 0000000000
- `smpp_disconnection_total` 0000
- 0000000000

□ □ □ □ □ □ □ □ □ □

1. 网络测试

网络

```
# 测试
ping -c 100 smpp.carrier.com | grep loss

# 测试丢包
netstat -s | grep -i error
```

网络

- 网络延迟/抖动
- 网络带宽/ISP
- 网络拥塞/丢包

2. 网络配置

网络

- 配置“enquire_link timeout”
- 配置网络参数

网络

- 配置 30 秒
- 配置网络参数
- 配置网络参数

3. 配置 TPS 参数

网络

- 配置网络参数
- 配置网络参数

网络

- 配置 `tps_limit` 参数

- TPS 測試時間約 70-80%
- 測試時間約 70-80%

4. 測試時間

測試

- 測試時間
- 測試時間

測試

- 測試時間
- 測試時間

測試時間

測試時間

測試

- 測試時間
- `smpp_messages_sent_total` 測試
- 測試時間

測試時間

1. 測試 `dest_smsc` 測試

測試

- Web UI → 測試 → 測試 `dest_smsc` 測試
- SMPP → 測試時間

測試

- 測試 `dest_smsc` 測試

- 00000000000000000000 `dest_smsc`
- 00 `dest_smsc` 0 NULL000000000000

2. 000000000000

0000

- Web UI → 00 → 00 `deliver_after` 00
- 00000000000000000000

0000

- 00000000000000000000 `deliver_after`
- 00000000000000000000

000000

- 00000000
- 000000000000000000000000

3. TPS 00000

0000

- 00000000
- 000000000000

000000

- 00000000 `tps_limit`
- 000000000000000000000000
- 00 **CONFIGURATION.md**

4. 000000000000

0000

- 000000
- 00000000

5/5

```
# 0000
sudo systemctl restart omnimessage-smpp

# 0000
sudo journalctl -u omnimessage-smpp -f
```

5. □□□□□□□□□□

[illegible]

- `queue` `queue`
-

111

- `Successfully registered SMPP peer frontend '<name>'` `<name>` `dest_smsc`
- `queue`

□□□□□

- 配置 `dest_smsc` 和 `queue` 为 `{hostname}_{peer}` 格式
- 配置 `queue` 为 `queue`
- 配置 `queue` 为 `queue`

□ □ □ □ □ □

111

- `smpp_delivery_failures_total` □□
- □□□□“submit_sm_resp”□□□□□□
- □□□□□□□□

□□□□□□□□□□

1. □□□□□□□□

□□□

- □□□□□□□□□□
- □□□□□□□□□□

□□□□□□□□

- 0x0000000B - □□□□□
- 0x00000001 - □□□□□□
- 0x00000003 - □□□□

□□□□□

- □□□□□□  E.164□
- □□□□□□□□□□□□
- □□□□□□□□

2. □□□□□□□□

□□□

- □□□□
- □□□□
- □□

□□□□□□

- GSM-7□□□ 160 □□□
- UCS-2□□□ 70 □□□
- □□□□□□□□□□
- □□□□□□□□

3. □□□□□□

□□□

- 環境変数
- 設定ファイル

確認

- 環境変数
- 設定ファイル
- 環境変数/設定ファイル

4. 確認

確認

- 環境変数 `expires` 設定
- 設定ファイル

確認

- 環境変数
- 設定ファイル

Web UI 確認

環境変数 Web 確認

確認

- 環境変数 `https://your-server:8087`
- 設定ファイル

環境変数/設定ファイル

1. 確認

確認

```
sudo systemctl status omnimessage-smpp
```

□□□□□

```
# □□□□□□□□□□
sudo systemctl start omnimessage-smpp

# □□□□□□□□□□
sudo journalctl -u omnimessage-smpp -n 50
```

2. □□□□□□□ 8087

□□□

```
sudo ufw status | grep 8087
# □
sudo firewall-cmd --list-ports | grep 8087
```

□❓❓□□□

```
# Ubuntu/Debian
sudo ufw allow 8087/tcp

# RHEL/CentOS
sudo firewall-cmd --permanent --add-port=8087/tcp
sudo firewall-cmd --reload
```

3. SSL □□□□

□□□

- □□□□□□□□□□
- □□□□□□□□

□□□□□

- □□□□□□□□□□□□□□□□

- 証明書 SSL 用
- 証明書インストール

```
ls -l /opt/omnimessage-smpp/priv/cert/
```

4. URL 用

用

- 証明書 HTTPS 用 HTTP
- 証明書 IP/用
- 証明書 8087

Web UI 用

用

- 証明書
- 証明書
- 証明書

用

1. 証明書

- Ctrl+F5 用
- 証明書 Cookie

2. 証明書

- F12
- 証明書 JavaScript 用
- 証明書

3. 証明書

- Chrome Firefox Edge 用
- 証明書

4. 確認

```
sudo journalctl -u omnimessage-smpp -f
```

確認

Prometheus 確認

確認

- `curl http://localhost:4000/metrics` 確認
- Prometheus 確認
- 確認

確認

1. 確認

確認

```
sudo systemctl status omnimessage-smpp
```

確認

```
sudo systemctl start omnimessage-smpp
```

2. 確認

確認

```
# macOS
curl http://localhost:4000/metrics

# Linux
curl http://your-server-ip:4000/metrics
```

確認

- 確認コマンドを実行
- Prometheus がポート 4000

3. 確認

確認

- `/metrics` `/prometheus` `/stats`
- 4000ポート 8087

確認

確認

- 確認
- 確認
- 確認

確認

1. 確認

- 確認
- 確認
- Prometheus `increase()` `rate()`

2. 確認

- 確認

- 消息队列
- 消息队列

3. 消息队列

- 消息队列
- 消息队列
- 消息 Prometheus 消息队列



CPU 消息

消息

```
top -p $(pgrep -f omnimessage-smpp)
```

消息

- 消息
- 消息
- 消息

消息

- 消息队列
- 消息 TPS 消息
- 消息 CPU 消息

消息

消息

```
ps aux | grep omnimessage-smpp
```

測試環境

- 測試環境配置
- 測試數據生成

測試結果

- 測試結果分析
- 測試結果總結
- 測試結果建議

測試總結

測試

- 測試環境
- 測試數據
- 測試結果

測試

1. TPS 測試 - 測試環境
2. `queue_check_frequency` - 測試數據
3. 測試 API 測試 - 測試結果
4. 測試結果分析

測試結果

- 測試結果分析 TPS
 - 測試 `queue_check_frequency` 測試數據
 - 測試 API
 - 測試結果
-

環境構築

インストール

前提

- 64bit OS
- 64bit CPU

手順

```
# Elixir インストール
/opt/omnimessage-smpp/bin/omnimessage-smpp eval "File.read!
('config/runtime.exs')"
```

確認

- 64bit OS
- 64bit CPU
- 64bit CPU
- Elixir インストール `import Config`

確認

- Elixir
- Elixir
- Elixir インストール

インストール

前提

- 64bit OS
- 64bit CPU

確認

```
# 重启服务
sudo systemctl restart omnimessage-smpp

# 查看状态
sudo systemctl status omnimessage-smpp

# 查看日志
sudo journalctl -u omnimessage-smpp -n 50
```

检查

磁盘空间

内存

1. 检查磁盘空间

```
# 查看磁盘
df -h

# 查看内存
free -h

# CPU 使用率
uptime
```

2. 检查服务状态

```
sudo systemctl status omnimessage-smpp
```

3. 查看日志

```
sudo journalctl -u omnimessage-smpp -n 200
```

4. 確認

```
sudo systemctl restart omnimessage-smpp
```

5. 確認

- 確認
- SSL 確認
- 確認
- 確認

6. 確認

```
# 確認
sudo cp /opt/omnimessage-smpp/config/runtime.exs.backup \
/opt/omnimessage-smpp/config/runtime.exs

# 確認
sudo systemctl restart omnimessage-smpp
```

7. 確認

確認

確認

確認

1. 確認 `cat /opt/omnimessage-smpp/VERSION`

2. 確認

```
sudo journalctl -u omnimessage-smpp -n 200 > /tmp/smpp-logs.txt
```

3. 確認

```
sudo cp /opt/omnimessage-smpp/config/runtime.exs  
/tmp/config.exs  
# [] /tmp/config.exs []
```

4. []

```
curl http://localhost:4000/metrics > /tmp/metrics.txt
```

5. []

```
uname -a > /tmp/system-info.txt  
free -h >> /tmp/system-info.txt  
df -h >> /tmp/system-info.txt
```

[]

- [] support@omnitouch.com
- [] +61 XXXX XXXX (24/7)
- [] []

[]

- **USAGE.md** - []
- **CONFIGURATION.md** - []
- **MONITORING.md** - []
- **README.md** - []



XXXXXX

XXXXXX OmniMessage Core

XXXX OmniMessage SMPP XXXXXXXX OmniMessage Core XXXXXXXXXXXXXXXXXXXX
OmniMessage - XXXXXXXXXXXXXXXX

XX OmniMessage XXXXX

- XX XXXXXXXX
- XX XXXXXXXXXXXX
- XX XXXXXXXX
- XX XXXXXXXXXXXX

XX **OmniMessage** XXXXX

```
# XX API XXXX
curl -k https://omnimessage-
core.example.com:8443/api/system/health

# XXXXXXXXXXXX API URL
grep api_base_url /opt/omnimessage-smpp/config/runtime.exs
```



XXXXXXXXXX

XXXXXXXXXXXXXXXXXX

1. XX **Web** XXXX

- URL: `https://your-server:8087`

- 配置数据库连接

2. 配置

- 配置SMPP → 配置
- 配置“”
- 配置

3. 配置

- 配置
- 配置
- 配置

4. 配置

- 配置
- 配置
- 配置

5. 配置 Prometheus

- `curl http://localhost:4000/metrics`
- 配置 Grafana
- 配置

○○○○

○○○○○○○○

- ○○○○> 2 ○○○○
- ○○○○○○> 5%
- ○○○○○○
- ○○○○○○

○○○ **MONITORING.md** ○○○○○○○○

○○○○○○

○○○○○○○○○○ OmniMessage Core □ SMPP ○○○○○○○○○

- **dest_smsc** - ○○○○○○○○ ○○○○○○○○ OmniMessage ○○○○○○○ **dest_smsc:**
"vodafone_uk" ○○○○○○ **vodafone_uk** ○○○○○○○○○○○ SMPP **submit_sm** ○○○
- **source_smsc** - ○○○○○○○○ ○○○○○○○○ OmniMessage ○○○○○○○ **source_smsc:**
"partner_acme" ○○○○○○○○ SMPP **deliver_sm** ○○○○○○○○ **partner_acme** ○○○○
○○○○○○○

○○○**???**○○○○○○○○ **submit_sm** PDU○○○○○○○○○○ ESME○○○○○○○○○ **deliver_sm**
PDU○○○○○○○○○ ESME ○○○ SMSC○○

○○○○

○○○○○○○ OmniMessage Core ○○○○○○○○○○○ SMPP ○○○○○○○○○○○

- ○○○○○○ **smsc_name** ○○○○○○○○○ "smpp_gateway" ○○○○○○ **SMSC_NAME** □
- ○○○○ 60 ○○○○○○○○○○○○
- ○○○○○○○○○○○○○○○○○○○ 90 ○○○○
- ○○○○○○○○○○○○○○○○○○○ ○○○ ○○○○○○○○○○○○○○○○○○○○○ **dest_smsc** □
 - ○○○○○○○○○○○○○ **queue** ○○○○○○○○○○○○○○○○○○○○○○○○○○○○○ **queue** ○○○○○○○○
○○○○○○○○○○○○○○○○

- `omnimessage_core`

OmniMessage Core `omnimessage_core`

`omnimessage_core`

1. `frontend_register` `omnimessage_core`
2. `smc_name` `omnimessage_core`
3. `omnimessage_core` `api_base_url`

SMPP

SMPP

SMPP `omnimessage_core` `omnimessage_core`

1 Web UI

- `omnimessage_core`
- `SMPP` → `omnimessage_core` / `omnimessage_core`
- `omnimessage_core`
- `Mnesia` `omnimessage_core`
- `omnimessage_core`

2

- `omnimessage_core`
- `/opt/omnimessage-smpp/config/runtime.exs`
- `Elixir` `omnimessage_core`
- `omnimessage_core`
- `omnimessage_core`
- `omnimessage_core`

Web UI `omnimessage_core`

`CONFIGURATION.md` `omnimessage_core`

環境構築

環境構築には **SMSC** と **ESME** が必要

環境構築

- SMPP 環境構築/IP
- ポート番号 2775
- ID 設定
- 設定
- 環境構築
- TPS 設定

環境構築

環境 **Web UI** 環境

環境  環境

環境

1. 環境構築

- 環境 Web UI `https://your-server:8087`
- 環境 SMPP → 環境

2. 環境構築

- 環境“環境構築”
- 環境
 - 環境 `vodafone_uk` 環境
 - 環境 `smpp.vodafone.co.uk`
 - 環境 `2775`
 - 環境 ID `your_username`
 - 環境 `your_password`
 - 環境 Transceiver
 - **TPS** 環境 `100`
 - 環境 `1000`

- 00“00”

3. 00000000

- 0000000000
- 0000SMPP → 0000
- 0000 10-30 00000“000”0000
- 000000000000000000

4. 0000000

- 0000000000
- 00000000 dest_smsc 00000000
- 000000000000
- 00000000

00 **B**00000000

000000000000000000

000

1. 00000000

```
sudo nano /opt/omnimessage-smpp/config/runtime.exs
```

2. ❶❷❸ 配置配置配置

```
config :omnimessage_smpp, :binds, [  
  # 配置...  
  
  # 配置  
  %{  
    name: "vodafone_uk",  
    mode: :client,  
    bind_type: :transceiver,  
    host: "smpp.vodafone.co.uk",  
    port: 2775,  
    system_id: "your_username",  
    password: "your_password",  
    tps_limit: 100,  
    queue_check_frequency: 1000,  
  
    # 配置配置配置配置配置配置 IP/配置  
    bind_source_ip: "10.20.30.40",  
    bind_source_port: 5000,  
  
    # 配置配置配置配置配置配置配置配置  
    queue: "vodafone_uk"  
  }  
]
```

配置 **CONFIGURATION.md** 配置配置配置配置配置配置 `bind_source_ip`
`bind_source_port` 配置 `queue`

3. 配置配置配置

```
# 配置配置 nano 配置 Ctrl+X, Y, Enter  
  
# 配置  
sudo systemctl restart omnimessage-smpp
```

4. 配置配置

- 配置配置 SMPP → 配置配置
- 配置配置

- 設置“SMS”項目
- 設置發送號碼

5. 設定

- 設定項目
- 設定項目 `dest_smsc` 設定項目
- 設定項目
- 設定項目

設定項目

設定項目 **ESMEs** 設定項目 **SMSC** 設定項目

設定

1. 設定

- 設定項目 ID `partner_name`
- 設定項目
- 設定項目

2. 設定項目

- 設定項目 IP 設定
- 設定項目 TPS 設定
- 設定項目

設定項目

設定 **A** 設定 **Web UI** 設定

設定項目

設定

1. 設定項目

- 設定 Web UI `https://your-server:8087`

- $\text{SMPP} \rightarrow \text{}$

2. □□□□□□□□

- “ ”
- - partner_acme
 - ID acme_corp
 - secure_password_123
 -
 - IP 203.0.113.0/24
 - TPS 50
 - 1000
- “ ”

3. □□□□□□□□

- ☐ 
- ☐ 

4. □□□□□□□□

- IP 10.0.0.1
- Port 2775
- ID acme_corp

- `secure_password_123`
- `secure_password_123`

5. `secure_password_123`

- `secure_password_123` → `secure_password_123`
- `secure_password_123`
- `secure_password_123`
- `secure_password_123` IP `secure_password_123`

`secure_password_123` **B** `secure_password_123`

`secure_password_123` `secure_password_123`

`secure_password_123`

1. `secure_password_123`

```
sudo nano /opt/omnimessage-smpp/config/runtime.exs
```

2. `secure_password_123`

```
# 配置 server_binds 配置
config :omnmessage_smpp, :server_binds, [
  # 配置...

  # 配置
  %{
    name: "partner_acme",
    system_id: "acme_corp",
    password: "secure_password_123",
    allowed_bind_types: [:transmitter, :receiver,
:transceiver],
    ip_whitelist: ["203.0.113.0/24"],
    tps_limit: 50,
    queue_check_frequency: 1000
  }
]

# 配置
config :omnmessage_smpp, :listen, %{
  host: "0.0.0.0",
  port: 2775,
  max_connections: 100
}
```

3. 配置

```
sudo systemctl restart omnmessage-smpp
```

4. 配置

- IP
- 2775
- ID acme_corp
- secure_password_123
-

5. 配置

- SMPP →

- 設定する
- 設定を確認する
- 設定 IP 設定を確認する

設定

設定を確認するTPS 設定を確認IP 設定を確認

設定を確認する

設定 **A** 設定 **Web UI** 設定

設定を確認する

設定

1. 設定

- 設定 Web UI `https://your-server:8087`
- 設定を確認するSMPP → 設定を確認
- 設定を確認するSMPP → 設定を確認

2. 設定

- 設定を確認する
- 設定“設定”
- 設定を確認する
 - 設定を確認するTPS 設定を確認IP 設定を確認/設定
- 設定“設定”

3. 設定

- 設定を確認する
- 設定を確認する
- 設定を確認するSMPP → 設定を確認

4. 設定

- 設定を確認する

- 設定ファイルの作成
- サービスの起動

1. **設定ファイルの作成**

設定ファイルの作成

1. 設定ファイル

1. 設定ファイル

```
sudo nano /opt/omnimessage-smpp/config/runtime.exs
```

2. 設定ファイル

- `:binds` と `:server_binds` の設定
- 設定値
- 設定値

```
%{
  name: "vodafone_uk",
  # ... 設定値
  tps_limit: 150, # 100 以上
  password: "new_password" # 設定値
}
```

3. サービスの再起動

```
sudo systemctl restart omnimessage-smpp
```

4. 確認

- サービスの起動状況を確認
- サービスのログを確認
- サービスの再起動を確認
- サービスの再起動を確認

概要

SMPP のインストール

前提条件

1. インストール

- インストール/アンインストール
- インストール

2. Web UI のインストール

- インストールSMPP → インストール
- インストール
- インストール“インストール”
- インストール

3. インストール

- インストールSMPP → インストール/インストール
- インストール
- インストール“インストール”
- インストール

4. インストール

- インストール - インストール
- インストール/インストール

インストール

インストール/アンインストール

インストール **enabled** インストール/インストール

Web UI

1. インストールSMPP → インストール/インストール

2. □□□□□□□□
3. □□“□□”
4. □□□□“□□”□□□
5. □□“□□”

□ □

□□□□□

- 
- 
- 

--	--	--	--

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

[illegible]

- 30 日
- 500 日
- 日

□ □ □ □ □ □ □ □ □ □

000000 SMPP enquire_link PDU 0000000000000000

- 00000060 000000 enquire_link_interval 00000000
- 000000 enquire_link_interval: 000000
- 00000000000000000000 enquire_link00000000000000000000

□ Prometheus □ `smpp_enquire_link_sent_total` □

[illegible]

TPS 限制

限制 TPS 的方法有以下几种：
`tps_limit`

- 设置 1 个限制
- 限制 TPS 的峰值
- 限制 TPS 的均值 100 个限制
- 限制 TPS 的方差

限制 TPS 的方法有以下几种：

1. `tps_limit` 限制 TPS 的峰值
 2. `queue_check_frequency` 限制 TPS 的均值
 3. 限制 TPS 的方差
-

限制 TPS

限制 TPS

限制 TPS

限制

1. 限制
 - 限制 TPS 的峰值
 - 限制 TPS 的均值

2. 送信側

- 送信機
- 送信機
 - 送信機
 - 送信機
 - 送信機 SMSC dest_smsc
 - 送信機
 - 送信機

3. 受信側

- 受信機
- 受信機 SMSC 送信機

送信機

送信機

送信機

1. 送信機

- 送信機 SMPP → 送信機
- 送信機
- 送信機 送信機

2. 設定項目

- 送信先電話番号
- 送信先国番号
- 送信先 `dest_smsc` 電話番号
- 送信先 `deliver_after` 時間

3. 送信方法

- 送信 = 送信
- 送信先電話番号
- 送信先国番号

4. 送信先電話番号

- 送信先電話番号
- 送信先国番号
- 送信先電話番号

送信先 **DLR** 送信

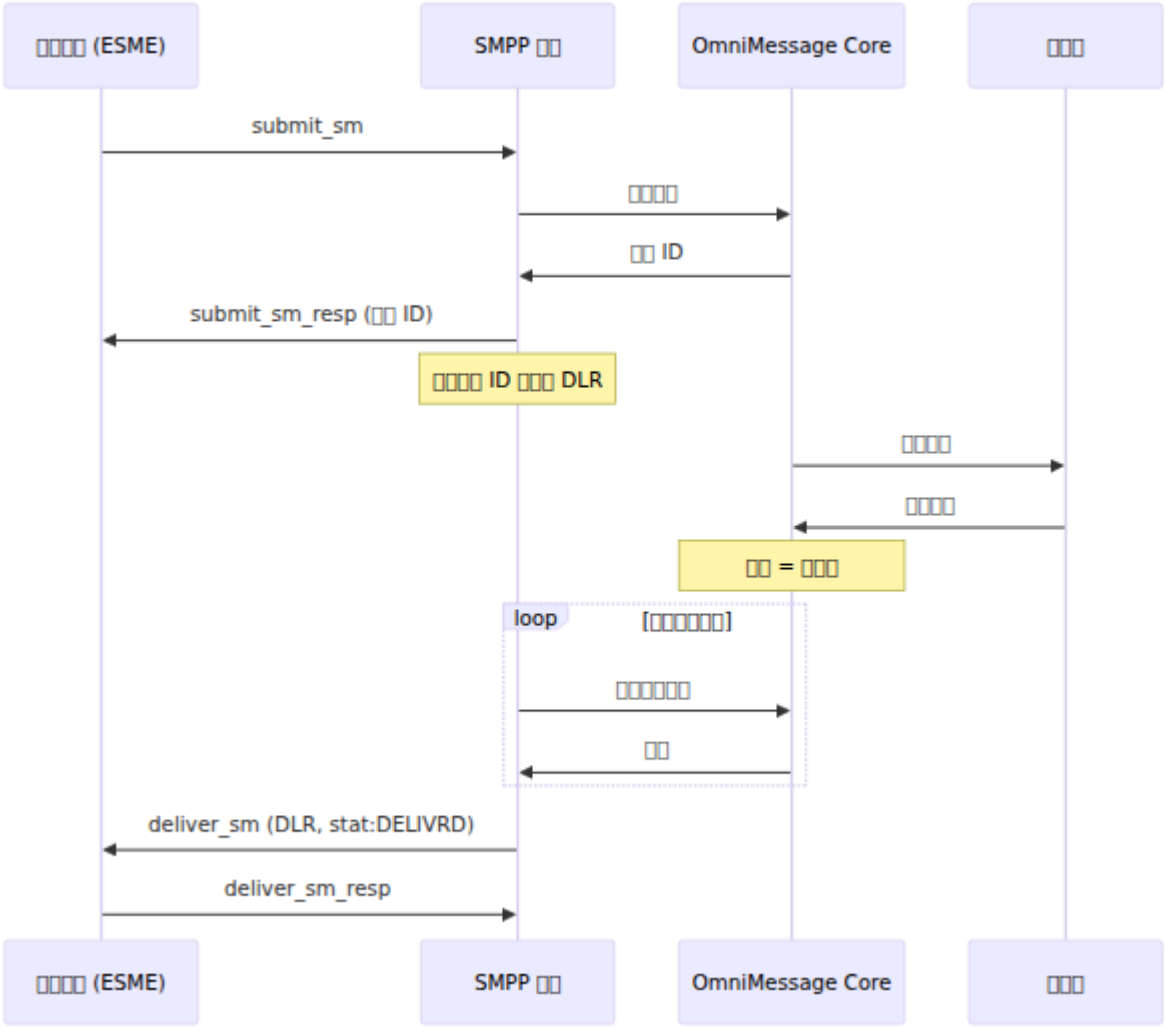
送信先 DLR 送信 SMPP 送信先電話番号  `deliver_receipts: true` 送信
送信 DLR 送信先電話番号

送信先 `deliver_sm` PDU 送信 `esm_class` 送信 `0x04` SMSC 送信先電話番号
`submit_sm_resp` 送信先 ID 送信先電話番号

送信

- 送信先 `deliver_receipts: true` 送信 `CONFIGURATION.md`
- 送信先 `cache_enabled: false` 送信先 ID 送信 ID 送信 ID 送信先電話番号
送信先電話番号

□□□



□□□□□

□□ stat □□□□□□□□□□□□

ステータス	stat	説明
delivered	DELIVRD	正常に配信された
expired	EXPIRED	配信期限が満了した
dropped	UNDELIV	配信失敗した
rejected	REJECTD	拒否された
balance_rejected	REJECTD	残高不足で拒否された

注意事項

- 配信失敗したメッセージは、自動的に再送信される。
- 配信失敗したメッセージは、24 時間以内で再送信される。
- DLR 情報は [MONITORING.md](#) を参照。

接続方法

接続方法

接続方法“接続方法”参照

接続

1. 接続方法

```
ping -c 3 carrier-smpp-server.com
telnet carrier-smpp-server.com 2775
```

2. 接続方法

- 接続方法

- 00000000
- 0000000000000000

3. 000000

- 0000SMPP → 0000/00000000
- 00 system_id 00000000
- 0000000000000000

4. 00000000

- 0000SMPP → 0000
- 00000000
- 00“0000”00
- 00 10-30 0
- 0000000000“0000”

5. 0000000000

- 00000000
- 0000000000000000
- 00000000
- 00 [TROUBLESHOOTING.md](#)

環境構築

必要なソフトウェア

OS

- Windows/Linux
- IP 設定
- 仮想マシン

ツール

1. 仮想マシン

- 仮想マシン SMPP → 仮想マシン
- 仮想マシン system_id 設定
- 仮想マシン

2. IP 設定

- 仮想マシン IP
- 仮想マシン IP 設定

3. 仮想マシン

- 仮想マシン
- 仮想マシン
- 仮想マシン

4. 仮想マシン

- 仮想マシン Web UI 設定
 - 仮想マシン“仮想マシン”
-

□□□□□

□□ Prometheus □□

□□□□□

```
curl http://localhost:4000/metrics | grep smpp_connection_status
```

□□□□□

```
smpp_connection_status{bind_name="vodafone_uk",...} 1
smpp_connection_status{bind_name="att_us",...} 1
```

□□□□□ 1□□□□□□

□□□□□

□□□□□□□

1. □□ Web UI → SMPP → □□□□
2. □□□□□□□□
3. □□□□□□□□
4. □□□□□□□□□□□□□□
5. □□ **TROUBLESHOOTING.md**

□□□□□□□

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

□□□□□□□

1. □□□□□□□□□□□□
2. □□ `dest_smsc` □□□□□□

3. 消息 TPS 限制策略

4. 消息 `queue_check_frequency` 配置

消息队列

消息队列

消息队列

1. 消息队列

- 消息队列
- 消息队列
- 消息队列

2. 消息队列

- 消息队列
- 消息队列
- 消息队列

3. 消息队列

- 消息队列 SMPP 消息
- 消息队列
- 消息 IP 消息队列

4. 消息队列

- 消息队列
- 消息 TPS 限制策略
- 消息队列

消息队列

消息队列

- 確認する
- 確認する
- 確認する

確認

```
# 確認
sudo systemctl status omnimessage-smpp

# 再起動
sudo systemctl restart omnimessage-smpp

# 確認
sudo systemctl status omnimessage-smpp

# 確認
sudo journalctl -u omnimessage-smpp -n 50
```

Web UI 確認

1. 確認する 30-60 分間隔
2. 確認SMPP → 確認
3. 確認する1-2 分間隔
4. 確認🔍🔍🔍

確認

確認する

```
# 確認
sudo cp /opt/omnimessage-smpp/config/runtime.exs \
/opt/omnimessage-smpp/config/runtime.exs.backup.$(date +%Y%m%d)

# 確認
sudo tar -czf /tmp/smpp-certs-$(date +%Y%m%d).tar.gz \
/opt/omnimessage-smpp/priv/cert/
```

確認する

```
# 备份
sudo cp /opt/omnimessage-smpp/config/runtime.exs.backup.YYYMMDD \
/opt/omnimessage-smpp/config/runtime.exs

# 重启
sudo systemctl restart omnimessage-smpp
```

验证

检查服务状态

1. 检查服务是否运行

```
sudo systemctl status omnimessage-smpp
```

2. 检查服务是否启动

```
sudo systemctl start omnimessage-smpp
```

3. 检查服务日志

```
sudo journalctl -u omnimessage-smpp -n 100
```

4. 检查系统资源

- 检查磁盘空间
- 检查 SSL 证书
- 检查磁盘空间 `df -h`
- 检查内存 `free -h`

5. 检查服务配置

SMPP 介紹

簡介

1. 什麼是 SMPP

- 什麼是 SMPP → 通訊協定
- 通訊協定
- 什麼是“通訊協定”

2. 什麼是 SMPP

- 什麼是 SMPP
- 什麼是 SMPP
- 什麼是 SMPP

3. 什麼是 SMPP

- 什麼是 SMPP
- 什麼是 SMPP
- 什麼是 SMPP

4. 什麼是 SMPP

- 什麼是 SMPP
- 什麼是 SMPP
- 什麼是 SMPP

SMPP 介紹

什麼是 SMPP

簡介

1. 什麼是 TPS

- 什麼是 SMPP → 通訊協定
- 什麼是 SMPP
- 什麼是 SMPP TPS

2. 認證與授權

- 認證與授權
- 認證與授權

3. 系統架構

- 系統架構
- 系統架構

4. 系統部署

- 系統部署
- 系統部署

系統部署

系統部署

- ☐ 系統部署 SMPP 系統部署
- ☐ 系統部署系統部署
- ☐ 系統部署系統部署
- ☐ 系統 Prometheus/Grafana 系統
- ☐ 系統部署 > 98%

系統部署

- ☐ 系統部署
- ☐ 系統部署
- ☐ 系統部署
- ☐ 系統部署
- ☐ 系統部署

目录

- ☐ 简介
 - ☐ 快速入门
 - ☐ 部署指南
 - ☐ 配置 TPS 节点
 - ☐ 常见问题
-

附录

- **CONFIGURATION.md** - 配置文件
 - **SOURCE_ADDRESS_WHITELIST.md** - 源地址白名单
 - **MONITORING.md** - Prometheus 监控
 - **TROUBLESHOOTING.md** - 故障排除
 - **README.md** - 说明
-

