□置参考

所有□置参数的完整指南

架构概述

OmniMessage SMPP □关是一个 无状态协议前端，用于将 SMPP 消息转换为 OmniMessage 及其反向操作。所有业务逻辑、路由决策和消息存储均由 OmniMessage Core 处理 - □关仅仅：

1. 接收 来自运营商和客户的 SMPP PDU
2. 通过 REST API 转换 为 OmniMessage 格式
3. 轮询 OmniMessage 以获取待发送的消息
4. 发送 SMPP PDU 到运营商
5. 报告 交付状态回 OmniMessage

这与其他 OmniMessage 前端（Diameter、MAP、IMS）的工作方式完全相同 - 它们都是无状态的协议转换器，委托给 OmniMessage Core。

□置文件位置

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

重要：更改配置后，重启□关：

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

□置结构

配置文件使用 Elixir 语法。基本结构：

```
import Config

# □局设置
config :omnimessage_smpp,
  setting_name: value

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

全局设置

API_BASE_URL

OmniMessage Core 平台 URL

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

参数	类型	必需	默认值
`api_base_url`	字符串 (URL)	是	-

目的: OmniMessage Core 平台的 URL。□关通过 REST API 与 OmniMessage 进行所有消息处理:

- 提交消息: 将接收到的 SMPP 消息发送到 OmniMessage 进行处理
- 检索消息: 轮询待发送到 SMPP 运营商的消息
- 报告交付状态: 将消息交付状态更新回 OmniMessage
- 系统健康: 定期健康检查

关键: 这是□关获取所有“智能”的地方。OmniMessage 处理:

- ✓ 消息验证和格式检查
- ✓ 路由决策（使用哪个运营商）
- ✓ 速率限制和流量控制
- ✓ 号码验证
- ✓ 消息存储和持久性
- ✓ 交付重试逻辑
- ✓ 状态跟踪

□关仅仅是将 SMPP ↔ OmniMessage 格式进行转换。

示例：

```
# 使用 IP 的 HTTPS
api_base_url: "https://192.168.1.100:8443"

# 使用主机名的 HTTPS
api_base_url: "https://omnimessage-core.company.com:8443"

# HTTP（不推荐用于生产）
api_base_url: "http://192.168.1.100:8080"
```

网络要求：

- □关必须能够访问 OmniMessage Core
- 在生产环境中使用 HTTPS（配置 `verify_ssl_peer`）
- 防火墙必须允许指定端口的出站 HTTPS

SMPP_POLL_INTERVAL

队列检查频率（毫秒）

```
config :omnimessage_smpp,
  smpp_poll_interval: 100
```

参数	类型	必需	默认值
`smpp_poll_interval`	整数	否	100

目的：每个客户端检查消息队列的频率（以毫秒为单位）。

指导原则：

- 高流量（**>100 TPS**）：100-500ms
- 中等流量（**10-100 TPS**）：500-1000ms
- 低流量（**<10 TPS**）：1000-2000ms

环境变量：`SMPP_POLL_INTERVAL`

VERIFY_SSL_PEER

SSL 证书验证

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

参数	类型	必需	默认值
`verify_ssl_peer`	布尔值	否	false

目的：在连接到后端 API 时是否验证 SSL 证书。

值：

- `true`：验证证书（生产环境中使用有效证书）
- `false`：跳过验证（自签名证书，测试）

环境变量：`VERIFY_SSL_PEER`

SMSC_NAME

注册的网关标识符

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

参数	类型	必需	默认值
`smsc_name`	字符串	否	"smpp_gateway"

目的：在消息队列后端中标识此□关实例。

环境变量：`SMSC_NAME`

SMPP 客户端绑定□置

客户端绑定是 出站连接，□关作为 **ESME**（客户端）连接到运营商 **SMSC** 服务器。在这种模式下，□关发起连接以通过外部运营商发送和接收消息。

完整的客户端绑定示例

```
config :omnimessage_smpp, :binds, [
  %{
    # 此连接的唯一标识符
    name: "vodafone_uk",

    # 连接模式
    mode: :client,

    # SMPP 绑定类型
    bind_type: :transceiver,

    # 运营商 SMPP 服务器地址
    host: "smpp.vodafone.co.uk",
    port: 2775,

    # 身份验证凭据
    system_id: "your_username",
    password: "your_password",

    # 速率限制
    tps_limit: 100,

    # 队列检查频率
    queue_check_frequency: 1000
  }
]
```

客户端绑定参数

name

唯一连接标识符

类型	必需	示例
字符串	是	`"vodafone_uk"`

目的：唯一标识此 SMPP 连接。

- 用于日志和指标
- 在所有绑定中必须唯一
- 使用描述性名称（运营商、地区、用途）

命名约定：

- `carrier_region`：`"vodafone_uk"`，`"att_us"`
- `purpose_number`：`"marketing_1"`，`"alerts_primary"`

mode

连接类型

类型	必需	值
原子	是	`:client`

目的：定义为出站连接，□关作为 **ESME** 连接到外部 **SMSC**。

固定值：出站连接始终为 `:client`。

bind_type

SMPP 会话类型

类型	必需	允许的值
原子	是	`:transmitter`，`:receiver`，`:transceiver`

目的：定义消息方向能力。

选项：

- `:transmitter` - 仅发送消息（submit_sm）
- `:receiver` - 仅接收消息（deliver_sm）
- `:transceiver` - 发送和接收（最常见）

建议：除非运营商要求特定类型，否则使用 `:transceiver`。

host

运营商 **SMPP** 服务器主机名或 **IP**

类型	必需	示例
字符串	是	`"smpp.carrier.com"` 或 `"10.5.1.100"`

目的：运营商的 SMPP 服务器地址。

示例：

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

port

SMPP 服务器端口

类型	必需	默认值	范围
整数	是	2775	1-65535

目的：SMPP 连接的 TCP 端口。

标准端口：2775

示例：

```
port: 2775  # 标准
port: 3000  # 自定义
```

system_id

身份验证用户名

类型	必需	示例
字符串	是	`"company_user"`

目的：运营商提供的用于身份验证的用户名。

安全性：保护此凭据 - 存储在配置文件中。

password

身份验证密码

类型	必需	示例
字符串	是	`"secret_password"`

目的：运营商提供的用于身份验证的密码。

安全性：

- 保护此凭据
- 使用强密码
- 定期更换

tps_limit

每秒事务限制

类型	必需	默认值	范围
整数	是	100	1-10000

目的：通过此连接发送的最大消息数。

指导原则：

- 设置为运营商最大值的 70-80%
- 防止限流/断开连接

- 允许交付回执的余量

示例：

```
tps_limit: 10    # 低流量
tps_limit: 50    # 中等流量
tps_limit: 100   # 高流量（最常见）
tps_limit: 1000  # 非常高的流量
```

计算：

```
如果运营商最大 = 100 TPS
设置 tps_limit = 70-80
留下 20-30 TPS 的余量
```

queue_check_frequency

消息队列轮询间隔（毫秒）

类型	必需	默认值	范围
整数	是	1000	100-10000

目的：多长时间检查后端以获取要发送的新消息。

指导原则：

- 高流量（**>100 TPS**）：500-1000ms
- 中等流量（**10-100 TPS**）：1000-2000ms
- 低流量（**<10 TPS**）：2000-5000ms

权衡：

- 较低的值 = 更快的消息获取，更多的 API 负载
- 较高的值 = 较慢的获取，较少的 API 负载

Web UI 示例：

SMPP 服务器绑定□置

服务器绑定定义 入站连接，□关作为 **SMSC**（服务器）接受来自外部 **ESME**（客户端）的连接。在这种模式下，合作伙伴系统连接到□关以发送和接收消息。

完整的服务器绑定示例

```
config :omnimessage_smpp, :server_binds, [
  %{
    # 此客户端的唯一标识符
    name: "partner_acme",

    # 客户端预期的凭据
    system_id: "acme_corp",
    password: "acme_secret",

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

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

    # 速率限制
    tps_limit: 50,

    # 队列检查频率
    queue_check_frequency: 1000
  }
]
```

服务器绑定参数

name

客户端标识符

类型	必需	示例
字符串	是	`"partner_acme"`

目的：标识连接到您的外部客户端。

命名约定：使用合作伙伴/客户端名称以便于识别。

system_id

客户端预期的用户名

类型	必需	示例
字符串	是	`"acme_corp"`

目的：外部客户端必须提供以进行身份验证的用户名。

提供给客户端：与您的合作伙伴共享此凭据。

password

客户端预期的密码

类型	必需	示例
字符串	是	`"secure_password"`

目的：外部客户端必须提供以进行身份验证的密码。

安全性：

- 使用强密码
- 每个客户端唯一
- 安全地与合作伙伴共享

allowed_bind_types

允许的会话类型

类型	必需	默认值
原子列表	是	-

目的：限制客户端可以使用的绑定类型。

选项：

```
allowed_bind_types: [:transceiver]  # 仅 transceiver
allowed_bind_types: [:transmitter, :receiver]  # TX 或 RX
allowed_bind_types: [:transmitter, :receiver, :transceiver]  # 任何
```

建议：除非需要限制，否则允许所有三种类型。

ip_whitelist

允许的客户端 **IP** 地址

类型	必需	默认值	格式
字符串列表	是	`[]`	IP 或 CIDR 表示法

目的：安全性 - 仅允许来自已知 IP 的连接。

格式：

- 单个 IP：`"192.168.1.100"`（自动 `/32`）
- CIDR 子□：`"192.168.1.0/24"`，`"10.0.0.0/8"`
- 混合：`["192.168.1.0/24", "10.50.1.100"]`

示例：

```
# 允许任何 IP（不推荐）
ip_whitelist: []

# 单个 IP
ip_whitelist: ["203.0.113.50"]

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

# 子网
ip_whitelist: ["192.168.1.0/24"]

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

常见子网：

- `/32` - 单个 IP（没有掩码的 IP 自动）
- `/24` - 256 个地址（例如，192.168.1.0-255）
- `/16` - 65,536 个地址（例如，10.50.0.0-255.255）
- `/8` - 16,777,216 个地址（例如，10.0.0.0-255.255.255.255）

tps_limit

每秒消息限制

与客户端绑定的 `tps_limit` 相同 - 控制出站 deliver_sm 速率。

queue_check_frequency

队列轮询间隔

与客户端绑定的 `queue_check_frequency` 相同 - 多长时间检查是否有要交付给此客户端的消息。

Web UI 示例：

服务器监听置

配置服务器绑定后，关监听传入连接。

完整的监听示例

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

监听参数

host

要绑定的 **IP** 地址

类型	必需	默认值	常见值
字符串	否	`"0.0.0.0"`	`"0.0.0.0"`, `"127.0.0.1"`

目的：监听哪个□络接口。

值：

- `"0.0.0.0"` - 在所有接口上监听（推荐）
- `"127.0.0.1"` - 仅在本地主机上监听（测试）
- `"192.168.1.10"` - 在特定 IP 上监听

port

要监听的 **TCP** 端口

类型	必需	默认值	范围
整数	否	2775	1-65535

目的：用于传入 SMPP 连接的端口。

标准：2775

max_connections

最大并发连接数

类型	必需	默认值	范围
整数	否	100	1-10000

目的：限制同时客户端连接的���数。

指导原则：

- 根据预期客户端设置
- 较高的值使用更多内存
- 典型：10-100 个连接

完整□置示例

示例 1：单个运营商连接

```
import Config

config :omnimessage_smpp,
  api_base_url: "https://smsc.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://smsc.company.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
  }
]
```

示例 **3**：具有服务器绑定的网关

```
import Config

config :omnimessage_smpp,
  api_base_url: "https://smsc.company.com:8443"

# 出站连接
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
  }
]

# 入站客户端定义
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
  }
]
```

```
# 服务器监听
config :omnimessage_smpp, :listen, %{
  host: "0.0.0.0",
  port: 2775,
  max_connections: 100
}
```

□置验证

编辑配置后，重启前进行验证：

语法检查

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

如果语法无效，您将看到错误。在重启之前修复。

测试□置

```
# 在前台重启以查看错误
sudo -u omnimessage-smpp /opt/omnimessage-smpp/bin/omnimessage-
smpp console
```

按 `Ctrl+C` 两次退出。

安全最佳实践

1. 保护□置文件：

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

2. 使用强密码：
   - 至少 12 个字符
   - 混合字母、数字、符号
   - 每个连接唯一
3. 使用 **IP** 白名单：
   - 始终为服务器绑定配置 `ip_whitelist`
   - 在生产中永远不要使用空列表 `[]`
4. 启用 **SSL** 验证：
   - 设置 `verify_ssl_peer: true` 并使用有效证书
5. 定期更换凭据：
   - 每季度更改密码
   - 与运营商/合作伙伴协调

下一步

- 查看 MONITORING.md 以获取指标配置
- 阅读 OPERATIONS.md 以管理连接
- 查看 TROUBLESHOOTING.md 以获取常见问题
- 返回 README.md 以获取概述

术语表

术语和定义

A

API（应用程序编程接口）

用于与消息队列后端系统通信的接口。

自动滚动

Web UI 日志标签中的功能，自动滚动以显示最新的日志条目。

B

后端

SMPP 关连接以检索和存储消息的消息队列系统。

绑定

两个系统之间的 SMPP 连接。可以是发射器、接收器或收发器。

绑定类型

SMPP 会话的类型：

- 发射器：仅发送消息
- 接收器：仅接收消息
- 收发器：发送和接收消息

绑定失败

当 SMPP 身份验证尝试失败时，通常是由于凭据不正确或 IP 限制。

C

CIDR（无类域间路由）

指定 IP 地址范围的符号（例如，`192.168.1.0/24` 表示 256 个 IP 地址）。

客户端绑定

出站 SMPP 连接，其中□关作为 **ESME** 连接到外部 **SMSC**（通常是运营商的 SMPP 服务器）。在此模式下，□关是客户端。

连接状态

SMPP 绑定的当前状态：

- 已连接：活动且可操作
- 未连接：未连接
- 重新连接：尝试建立连接

计数���

仅增加的指标（在服务重启时重置），用于发送的消息等总数。

D

数据编码

指定消息字符编码的 SMPP 字段（GSM-7、UCS-2 等）。

交付失败

当消息无法送达时，由运营商的错误响应指示。

交付回执（**DLR**）

运营商关于消息交付状态的确认。

dest_smsc

消息队列中的字段，指示哪个 SMPP 连接应处理该消息。

断开连接

当活动的 SMPP 连接被终止时，无论是故意还是由于错误。

E

查询链接

定期发送的 SMPP 保持活动消息，以验证连接是否处于活动状态。

ESM 类

指示消息类型和特性的 SMPP 字段。

ESME（外部短消息实体）

在 SMPP 术语中，连接到 SMSC 以发送或接收消息的客户端应用程序。当□关在 客户端模式 下运行时，它充当连接到运营商 SMSC 的 ESME。当它在 服务器模式 下运行时，它接受来自外部 ESME 的连接。

指数退避

重试策略，每次失败后等待时间加倍（1分钟、2分钟、4分钟、8分钟……）。

F

防火墙

控制进出□络流量的□络安全系统。

G

网关

连接消息队列和移动□络的 SMPP □关应用程序。

仪表

可以增加或减少的指��，表示当前值（例如，连接状态）。

Grafana

用于在仪表板中显示 Prometheus 指标的流行可视化工具。

GSM-7

用于 SMS 的标准 7 位字符编码，支持每条消息最多 160 个字符。

H

HTTP/HTTPS

用于□络通信的协议。HTTPS 是加密版本。

I

IP 白名单

可以连接到□关的允许 IP 地址列表（安全功能）。

ISDN（综合业务数字网）

常用于电话号码的编号计划。

J

(□□□)

K

保持活动

定期发送的消息（enquire_link），以维护连接并检测故障。

KPI（关键绩效指标）

指示系统性能的可测量值（例如，交付成功率）。

L

标签

在 Prometheus 中，附加到指标的键值对，用于标识（例如，`bind_name="vodafone_uk"`）。

实时视图

用于实时 Web UI 更新的 Phoenix 框架技术。

M

消息队列

存储等待发送或接收的消息的后端系统。

指标

系统性能的定量测量，以 Prometheus 格式公开。

MO（移动发起）

从手机发送到□关的消息（入站）。

MT（移动终止）

从□关发送到手机的消息（出站）。

MSISDN（移动台国际用户目录号码）

移动电话号码的标准格式。

N

NPI（编号计划指示符）

指定编号方案的 SMPP 字段（例如，ISDN）。

O

出站

从□关流向移动□络的消息。

入站

从移动□络流向□关的消息。

P

PDU（协议数据单元）

单个 SMPP 消息包（例如，submit_sm、deliver_sm）。

Prometheus

收集和存储时间序列指标的开源监控系统。

Q

队列

等待处理或发送的消息列表。

队列检查频率

□关多长时间（以毫秒为单位）轮询后端以获取新消息。

队列工作者

从队列中检索消息并通过 SMPP 发送的组件。

R

速率限制

控制消息吞吐量以遵守运营商限制。见 TPS。

接收器

仅接收消息的 SMPP 绑定类型（deliver_sm）。

重新连接

重新建立断开的 SMPP 连接。

重试

尝试再次发送失败的消息，通常使用指数退避。

S

服务器绑定

允许外部 **ESMEs**（客户端）连接到□关的配置。在此模式下，□关充当接受来自合作伙伴系统的入站连接的 **SMSC**（服务器）。

会话

两个系统之间的活动 SMPP 连接。

SMPP（短消息对等网络）

用于在系统之间交换 SMS 消息的行业标准协议。

SMSC（短消息服务中心）

在 SMPP 术语中，接受来自 ESME（客户端）连接并处理 SMS 消息路由和交付的服务器组件。当□关在 服务器模式 下运行时，它充当接受来自外部 ESME 的连接的 SMSC。

SSL/TLS

用于安全通信的加密协议。

Submit_SM

用于提交消息以进行交付的 SMPP PDU。

Submit_SM_Resp

对 submit_sm 的 SMPP 响应，指示成功或失败。

系统 **ID**

用于 SMPP 身份验证的用户名。

T

遥测

系统指标的自动收集和传输。

TON（号码类型）

指定号码格式的 SMPP 字段（例如，国际、国家）。

TPS（每秒事务数）

通过连接的最大消息每秒速率限制。

收发器

可以同时发送和接收消息的 SMPP 绑定类型（最常见）。

发射器

仅发送消息的 SMPP 绑定类型（submit_sm）。

吞吐量

消息处理速率，通常以每秒消息数来衡量。

U

UCS-2

用于 SMS 的 16 位 Unicode 字符编码，支持每条消息最多 70 个字符。

正常运行时间

连接或服务持续运行的时间。

V

有效期

在过期之前尝试交付消息的时间限制。

W

Web 仪表板

用于监控和管理□关的基于浏览器的用户界面。

白名单

见 IP 白名单。

X

(□□□)

Y

(□□□)

Z

(□□□)

缩略语快速参考

缩略语	全称
API	应用程序编程接口
CIDR	无类域间路由
DLR	交付回执
ESME	外部短消息实体
GSM	全球移动通信系统
HTTP	超文本传输协议
HTTPS	安全超文本传输协议
IP	互联□协议
ISDN	综合业务数字□
KPI	关键绩效指标
MO	移动发起
MSISDN	移动台国际用户目录号码
MT	移动终止
NPI	编号计划指示符
PDU	协议数据单元
SMPP	短消息对等□络
SMSC	短消息服务中心

缩略语	全称
SMS	短消息服务
SSL	安全套接字层
TLS	传输层安全
TON	号码类型
TPS	每秒事务数
UCS	通用编码字符集
UI	用户界面
URL	统一资源定位符

相关文档

- **README.md** - 系统概述和入门
- **CONFIGURATION.md** - 配置参数说明
- **OPERATIONS.md** - 日常操作
- **MONITORING.md** - 指标和监控
- **TROUBLESHOOTING.md** - 问题解决

监控和指标指南

SMPP网关监控的完整参考

概述

SMPP□关以Prometheus格式公开指标，用于监控连接健康、消息吞吐量和系统性能。

关键：由于□关是无状态的，并依赖于OmniMessage Core，**OmniMessage**连接性是最重要的监控指标。请监控以下两个方面：

1. **SMPP**网关指标 - 协议级健康
2. **OmniMessage API**指标 - 后端连接性和健康

指标端点

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

格式： Prometheus文本格式

访问：默认情况下对localhost开放（配置防火墙以进行远程访问）

快速测试

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

可用指标

所有指标以 `smpp_` 为前缀，并包含用于识别的标签。

许可证指标

omnimessage_smpp_license_status

类型：Gauge

描述：当前许可证状态

值：

- `1` = 有效许可证
- `0` = 无效/过期许可证

标签：无

示例：

```
omnimessage_smpp_license_status 1
```

用途��

- 当值为0时发出警报（无效许可证）
- 当许可证无效时，出站队列处理停止，但SMPP绑定保持连接
- Web UI仍可访问以进行故障排除

产品名称：`omnimessage_smpp`

注意：

- 当许可证无效（`license_status == 0`）时，□关停止处理出站队列
- SMPP绑定（客户端和服务器）保持连接并接受绑定请求
- 入站消息仍然被接收但不被处理
- 无论许可证状态如何，UI和监控仍然可访问

警报示例：

```
- alert: SMPP_License_Invalid
  expr: omnimessage_smpp_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "SMPP网关许可证无效或过期"
    description: "许可证状态无效 - 出站消息处理被阻止"
```

连接状态指标

smpp_connection_status

类型: Gauge

描述: 当前SMPP绑定的连接状态

值:

- `1` = 已连接
- `0` = 已断开

标签:

- `bind_name` - 连接名称（例如，“vodafone_uk”）
- `mode` - 连接类型（“client”或“server”）
- `host` - 远程主机（仅客户端模式）
- `port` - 远程端口（仅客户端模式）
- `bind_type` - SMPP绑定类型（仅客户端模式）
- `system_id` - 使用的系统ID

示例:

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

用途:

- 当值为0时发出警报（已断开）
- 跟踪连接正常运行时间百分比
- 监控重新连接频率

消息计数器

smpp_messages_sent_total

类型：Counter

描述：通过SMPP绑定发送的消息总数

单位：消息

标签：与connection_status相同

示例：

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

用途：

- 计算消息速率（消息/秒）
- 跟踪每日/每月的消息量
- 比较实际与预期的吞吐量

smpp_messages_received_total

类型：Counter

描述：通过SMPP绑定接收的消息总数

单位：消息

标签：与connection_status相同

示例：

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

用途:

- 监控入站消息量
- 跟踪移动发起（MO）流量
- 对意外的流量变化发出警报

投递指标

smpp_delivery_failures_total

类型: Counter

描述: 消息投递失败的总数

单位: 失败

标签: 与connection_status相同

示例:

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

用途:

- 计算投递成功率
- 对高失败率发出警报
- 识别问题连接

成功率计算:

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

绑定操作指标

smpp_bind_success_total

类型： Counter

描述：成功绑定操作的总数

单位：绑定尝试

示例：

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

用途：

- 跟踪绑定稳定性
- 监控身份验证成功率

smpp_bind_failures_total

类型： Counter

描述：失败绑定操作的总数

单位：绑定尝试

示例：

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

用途：

- 对身份验证失败发出警报
- 识别凭据问题
- 跟踪运营商连接问题

连接事件指标

smpp_connection_attempts_total

类型：Counter

描述：连接尝试的总数

单位：尝试

示例：

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

用途：

- 跟踪连接流失
- 监控重新连接频率

smpp_disconnection_total

类型：Counter

描述：断开的总数

单位：断开

示例：

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

用途：

- 对频繁断开发出警报
- 识别□络问题
- 跟踪连接稳定性

正常运行时间指标

smpp_uptime_seconds

类型：Gauge

描述：当前SMPP绑定的正常运行时间（以秒为单位）

单位：秒

示例:

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

用途:

- 跟踪连接稳定性
- 计算正常运行时间百分比
- 对最近的重启发出警报

OmniMessage API健康指标

虽然□关本身公开与SMPP相关的指标，但**OmniMessage API**的健康至关重要。您还应该监控:

来自**OmniMessage**的指标（如果可用）

- `omnimessage_api_requests_total` - 从□关发出的API请求总数
- `omnimessage_api_request_duration_seconds` - API响应时间
- `omnimessage_queue_depth` - OmniMessage队列中待处理的消息

来自网关日志（如果未公开指标）

查找以下模式以检测API问题:

- "api.*connection refused" - 无法连接到OmniMessage
- "api.*timeout" - OmniMessage未响应
- "api.*http 503" - OmniMessage暂时不可用
- "api.*parse error" - 响应格式问题

Prometheus□置

基本抓取□置

添加到 `/etc/prometheus/prometheus.yml`���

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

多个网关

```yaml
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'
```

服务发现

使用基于文件的发现：

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

文件 `/etc/prometheus/targets/smpp-production.json`：

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

Grafana仪表板

示例仪表板面板

连接状态面板

查询：

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

可视化：Stat

阈值：

- 红色：值 < 1（已断开）
- 绿色：值 == 1（已连接）

消息速率面板

查询：

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

可视化：Graph

单位：消息/秒

图例：`{{bind_name}}`

投递成功率面板

查询:

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

可视化: Gauge
单位: 百分比 (0-100)
阈值:

- 红色: < 95%
- 黄色: 95-98%
- 绿色: > 98%

连接正常运行时间面板

查询:

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

可视化: Stat
单位: 小时

警报规则

Prometheus警报规则

保存到 `/etc/prometheus/rules/smpp-alerts.yml`:

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

      # 高失败率
      - alert: SMPPHighFailureRate
        expr: |
          (
            rate(smpp_delivery_failures_total[5m])
            /
            rate(smpp_messages_sent_total[5m])
          ) > 0.05
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "在 {{ $labels.bind_name }} 上高投递失败率"
          description: "投递失败率在 {{ $labels.bind_name }} 上为 {{
$value | humanizePercentage }}。"

      # 绑定失败
      - alert: SMPPBindFailures
        expr: increase(smpp_bind_failures_total[10m]) > 3
        labels:
          severity: warning
        annotations:
          summary: "在 {{ $labels.bind_name }} 上多次绑定失败"
          description: "{{ $labels.bind_name }} 在过去10分钟内绑定失败 {{
$value }} 次。"

      # 没有发送消息（预期时）
      - alert: SMPPNoTraffic
        expr: rate(smpp_messages_sent_total[10m]) == 0
```

```
        for: 30m
        labels:
          severity: warning
        annotations:
          summary: "在 {{ $labels.bind_name }} 上没有发送消息"
          description: "{{ $labels.bind_name }} 在过去30分钟内没有发送任何消息。"

      # 频繁断开
      - alert: SMPPFrequentDisconnections
        expr: increase(smpp_disconnection_total[1h]) > 5
        labels:
          severity: warning
        annotations:
          summary: "在 {{ $labels.bind_name }} 上频繁断开"
          description: "{{ $labels.bind_name }} 在过去一小时内断开 {{
$value }} 次。"

      # OmniMessage API不可达
      - alert: OmniMessageAPIUnreachable
        expr: |
          count(count_over_time({job="omnimessage-smpp"} |=
"api.*connection refused"[5m])) > 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "OmniMessage API不可达"
          description: "SMPP网关无法连接到OmniMessage API。检查API_BASE_URL配
置和网络连接。"

      # OmniMessage API超时
      - alert: OmniMessageAPITimeout
        expr: |
          count(count_over_time({job="omnimessage-smpp"} |=
"api.*timeout"[5m])) > 5
        for: 2m
        labels:
          severity: warning
        annotations:
          summary: "OmniMessage API超时"
          description: "检测到多个API超时。OmniMessage可能很慢或超载。"

      # 没有消息流（API问题）
      - 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: "未检测到消息流 - 检查OmniMessage连接性"
          description: "在过去30分钟内没有发送或接收消息。检查OmniMessage API连接性和队
状态。"
```

在 `prometheus.yml` 中加载规则:

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

Web仪表板监控

内置的Web UI提供实时监控，无需Prometheus。

访问

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

实时状态页面

导航: SMPP → 实时状态

功能:

- 实时连接状态
- 消息计数器
- 连接正常运行时间
- 手动重连/断开控制
- 每5秒自动刷新

用途:

- 快速状态检查
- 手动干预
- 实时故障排除

仪表板显示：

- 总绑定：所有客户端和服务器连接的总计数
- 客户端绑定：到运营商的出站连接（显示已连接/已断开计数）
- 服务器绑定：来自合作伙伴的入站连接（显示活动/等待计数）
- 服务器监听：入站服务器套接字的配置（主机、端口、最大连接数）

日志监控

系统日志

查看日志：

```
# 实时跟踪日志
sudo journalctl -u omnimessage-smpp -f

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

# 从特定时间开始
sudo journalctl -u omnimessage-smpp --since "1 hour ago"

# 按级别过滤
sudo journalctl -u omnimessage-smpp -p err
```

Web UI日志

导航：Web UI中的日志选项卡

功能：

- 实时日志流
- 按级别过滤（调试、信息、警告、错误）
- 搜索日志
- 暂停/恢复
- 清除日志

日志视图允许您：

- □□□□□□□□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□/□□□□□□□□□□□□□
- □□/□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□

□□□□□□□KPI□

□□□□

□□□ □□□□□□□□□□□

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

□□□ > 99.9%

□□□□□

□□□ □□□□□□□□□

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

□□□ □□□□□□□□□

□□□□□

□□□ □□□□□□□□□

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

□□□ > 98%

绑定稳定性

指标：每小时的绑定尝试

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

目标：< 10次每小时（表示连接稳定）

监控最佳实践

1. 设置警报

- 为关键指标配置Prometheus警报
- 使用PagerDuty/OpsGenie进行24/7警报
- 定期测试警报

2. 创建仪表板

- 为每个□关构建Grafana仪表板
- 在一个仪表板上包含所有连接
- 添加容量规划面板

3. 定期审查

- 每周审查指标
- 识别趋势和模式
- 计划容量调整

4. 记录基线

- 记录正常的消息量
- 记录预期的TPS速率
- 注意高峰时间/天

5. 与后端关联

- 监控后端API指标
- 跟踪端到端消息流
- 识别瓶颈

使用指标进行故障排除

连接问题

检查: `smpp_connection_status`

- 值为0 = 审查日志，检查□络，验证凭据
- 频繁变化 = □络不稳定

投递率低

检查: `smpp_delivery_failures_total`

- 高比率 = 检查运营商状态，审查消息格式
- 跨连接比较 = 识别问题运营商

吞吐量低

检查: `smpp_messages_sent_total` 速率

- 低于预期 = 检查TPS限制，队列可用性
- 检查后端API指标

绑定问题

检查: `smpp_bind_failures_total`

- 增加 = 身份验证问题，凭据问题
- 检查配置中的system_id和密码

相关文档

- **CONFIGURATION.md** - 配置监控设置
- **OPERATIONS.md** - 操作程序
- **TROUBLESHOOTING.md** - 解决问题
- **README.md** - 概述和快速入门

操作指南

日常操作程序

关键依赖: **OmniMessage Core**

重要: OmniMessage SMPP □关在没有访问 OmniMessage Core 的情况下无法运行。所有消息处理都发生在 OmniMessage 中 - □关只是一个协议转换器。

如果 OmniMessage 无法使用:

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

检查 **OmniMessage** 健康状况:

```
# 测试 API 连接性
curl -k https://omnimessage-
core.example.com:8443/api/system/health

# 检查日志中的配置 API URL
grep api_base_url /opt/omnimessage-smpp/config/runtime.exs
```

日常操作

早晨健康检查

在每天开始时执行以下检查:

1. 访问 **Web** 仪表板
    - URL: `https://your-server:8087`

- 检查仪表板是否正确加载

2. 检查连接状态

   - 导航到：SMPP → 实时状态
   - 验证所有连接显示“已连接”（绿色）
   - 注意任何断开的绑定

3. 查看消息指标

   - 导航到：队列标签
   - 检查消息计数是否合理
   - 验证没有意外的队列积压

4. 检查系统日志

   - 导航到：日志标签
   - 查找错误消息（红色）
   - 注意任何警告模式

5. 查看 **Prometheus** 指标

   - `curl http://localhost:4000/metrics`
   - 或检查 Grafana 仪表板
   - 验证消息速率是否正常

持续监控

设置警报以监控：

- 连接失败（> 2 分钟宕机）
- 高交付失败率（> 5%）
- 长时间没有流量
- 频繁断开连接

请参见 MONITORING.md 以获取警报配置。

管理 SMPP 连接

SMPP 对等体的□置方式

SMPP 连接（对等体）可以使用 两种方法 进行配置：

方法 1: Web UI（推荐）

- 优点：更改立即生效，无需重启
- 位置：SMPP → 客户端对等体 / 服务器对等体标签
- 操作：添加、编辑、删除对等体
- 持久性：存储在 Mnesia 数据库中
- 最佳用途：日常操作、测试、快速更改

方法 2: □置文件

- 优点：代码形式的配置，版本控制
- 位置：`/opt/omnimessage-smpp/config/runtime.exs`
- 操作：在 Elixir 配置中定义对等体
- 持久性：基于文件，重启后仍然有效
- 要求：更改后需要重启服务
- 最佳用途：初始设置、基础设施作为代码

注意：Web UI 更改单独存储并覆盖配置文件设置。

请参见 CONFIGURATION.md 以获取配置文件参考。

添加新的客户端连接

目的：配置□关作为连接到运营商的 **ESME**（客户端）连接到 **SMSC**（服务器）

准备：收集运营商的信息：

- SMPP 服务器主机名/IP
- 端口号（通常为 2775）
- 系统 ID（用户名）
- 密码
- 绑定类型（通常为双向）
- TPS 限制

选择以下方法之一：

选项 **A**：通过 **Web UI**（推荐）

优点：立即生效，无需重启

步骤：

1. 导航到客户端对等体：
    - 打开 Web UI：`https://your-server:8087`
    - 导航到：SMPP → 客户端对等体
2. 添加新对等体：
    - 点击“添加新客户端对等体”
    - 填写表单：
        - 名称：`vodafone_uk`（唯一标识符）
        - 主机：`smpp.vodafone.co.uk`
        - 端口：`2775`
        - 系统 **ID**：`your_username`
        - 密码：`your_password`
        - 绑定类型：`Transceiver`

        - **TPS** 限制：`100`
        - 队列检查频率：`1000`
    - 点击“保存”

3. 连接自动建立：
    - 关立即尝试连接
    - 导航到：SMPP → 实时状态
    - 状态应在 10-30 秒内更改为“已连接”（绿色）
    - 检查日志标签以获取成功绑定消息
4. 测试消息流：
    - 导航到：队列标签
    - 提交测试消息，`dest_smsc` 匹配绑定名称
    - 在实时状态中监控传输
    - 验证交付确认

选项 **B**：通过置文件

优点：基础设施作为代码，版本控制

步骤：

1. 编辑置文件：

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

2. 将新绑定添加到□置中:

```
config :omnimessage_smpp, :binds, [
  # 现有绑定...

  # 添加新绑定
  %{
    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
  }
]
```

3. 保存并重启服务:

```
# 保存文件（在 nano 中按 Ctrl+X, Y, Enter）

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

4. 验证连接:

   - 导航到: SMPP → 实时状态
   - 找到新连接
   - 状态应为“已连接”（绿色）
   - 检查日志以获取成功绑定

5. 测试消息流:

   - 导航到: 队列标签

- 提交测试消息，`dest_smsc` 匹配新绑定名称
- 在实时状态中监控传输
- 验证交付确认

添加服务器绑定

目的：配置□关作为 **SMSC**（服务器），接受来自外部 **ESMEs**（合作伙伴客户端）的连接

准备：

1. 生成凭据：
   - 创建唯一的系统 ID：`partner_name`
   - 创建强密码
   - 安全地记录并与合作伙伴共享
2. 获取合作伙伴信息：
   - 合作伙伴的源 IP 地址
   - 预期消息量（用于 TPS 限制）
   - 所需的绑定类型

选择以下方法之一：

选项 **A**：通过 **Web UI**（推荐）

优点：立即生效，无需重启

步骤：

1. 导航到服务器对等体：
   - 打开 Web UI：`https://your-server:8087`
   - 导航到：SMPP → 服务器对等体
2. 添加新服务器对等体：
   - 点击“添加新服务器对等体”
   - 填写表单：
     - 名称：`partner_acme`（唯一标识符）

        - 系统 **ID**：`acme_corp`
        - 密码：`secure_password_123`
        - 允许的绑定类型：选择所有（发送者、接收者、双向）
        - **IP** 白名单：`203.0.113.0/24`（多个用逗号分隔）
        - **TPS** 限制：`50`
        - 队列检查频率：`1000`
    - 点击“保存”

3. 网关准备好连接：
    - 服务器对等体现在处于活动状态，等待合作伙伴连接
    - 无需重启
4. 与合作伙伴共享信息：
    - 关 IP 地址
    - 端口：`2775`
    - 系统 ID：`acme_corp`
    - 密码：`secure_password_123`
    - 绑定类型：如配置
5. 等待合作伙伴连接：
    - 导航到：SMPP → 实时状态
    - 观察传入连接

- 验证身份验证成功
- 检查 IP 是否匹配白名单

选项 **B**：通过□置文件

优点：基础设施作为代码，版本控制

步骤：

1. 编辑□置文件：

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

2. 添加服务器绑定和监听□置：

```
# 添加到 server_binds 列表
config :omnimessage_smpp, :server_binds, [
  # 现有服务器绑定...

  # 添加新服务器绑定
  %{
    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
  }
]

# 确保存在监听配置（仅需一次）
config :omnimessage_smpp, :listen, %{
  host: "0.0.0.0",
  port: 2775,
  max_connections: 100
}
```

3. 保存并重启服务：

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

4. 与合作伙伴共享信息：
   - □关 IP 地址
   - 端口：`2775`
   - 系统 ID：`acme_corp`
   - 密码：`secure_password_123`
   - 绑定类型：如配置
5. 等待合作伙伴连接：
   - 导航到：SMPP → 实时状态
   - 观察传入连接
   - 验证身份验证成功
   - 检查 IP 是否匹配白名单

修改现有连接

目的：更新连接参数（TPS 限制、密码、IP 白名单等）

选择以下方法之一：

选项 **A**：通过 **Web UI**（推荐）

优点：立即生效，无需重启

步骤：

1. 导航到对等体：
   - 打开 Web UI：`https://your-server:8087`
   - 对于客户端连接：SMPP → 客户端对等体
   - 对于服务器连接：SMPP → 服务器对等体
2. 编辑对等体：
   - 找到要修改的对等体

- 点击“编辑”按钮
- 更新所需参数：
  - 常见更改：TPS 限制、密码、IP 白名单、主机/端口
- 点击“保存”

3. 更改立即生效：
   - 连接会自动使用新设置重新连接
   - 无需重启服务
   - 导航到：SMPP → 实时状态以验证
4. 验证更改：
   - 检查连接是否成功建立
   - 监控日志标签以查找错误
   - 如果适用，测试消息流

选项 **B**：通过□置文件

优点：基础设施作为代码，版本控制

步骤：

1. 编辑□置文件：

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

2. 修改绑定参数：
   - 在 `:binds` 或 `:server_binds` 列表中找到绑定
   - 更新所需参数：
     - 常见更改：TPS 限制、密码、IP 白名单、主机/端口
   - 示例：

```
%{
  name: "vodafone_uk",
  # ... 其他参数
  tps_limit: 150,  # 从 100 更改
  password: "new_password"  # 更新密码
}
```

3. 保存并重启服务：

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

4. 验证更改：
   - 导航到：SMPP → 实时状态
   - 检查连接是否成功建立
   - 监控日志以查找错误
   - 测试消息流

移除连接

目的：退役一个 SMPP 连接

步骤：

1. 通知利益相关者：
   - 通知运营商/合作伙伴
   - 协调停机窗口
2. 通过 **Web UI** 断开连接：
   - 导航到：SMPP → 实时状态
   - 找到连接
   - 点击“断开连接”
   - 确认操作
3. 移除□置：

- 导航到：SMPP → 客户端/服务器对等体
- 找到连接
- 点击“删除”
- 确认移除

4. 验证移除：

- 检查实时状态 - 连接应消失
- 检查日志以确保干净关闭

管理消息流

检查消息队列

目的：监控待处理消息

步骤：

1. 访问队列：

- 导航到：队列标签
- 查看待处理消息列表

2. 检查消息详细信息：

- 点击消息行
- 查看：
  - 目标号码
  - 消息正文
  - 目标 SMSC（`dest_smsc`）
  - 交付尝试
  - 状态

3. 搜索特定消息：

   - 使用搜索过滤器
   - 按目标、内容或 SMSC 过滤

故障排除卡住的消息

症状：消息未被交付

步骤：

1. 检查连接状态：

   - 导航到：SMPP → 实时状态
   - 验证目标连接是否已连接
   - 如果断开，请参见 重新连接

2. 检查消息详细信息：

   - 导航到：队列标签
   - 找到卡住的消息
   - 检查 `dest_smsc` 字段是否与连接名称匹配
   - 检查 `deliver_after` 时间戳（重试调度）

3. 检查交付尝试：

   - 高尝试 = 重复失败
   - 检查日志以获取错误消息
   - 可能表示格式无效或运营商拒绝

4. 手动干预（如有必要）：
    - 联系运营商以验证问题
    - 可能需要取消并重新提交消息
    - 与后端团队检查队列问题

连接故障排除

重新连接绑定

症状：连接显示“已断开”（红色）

步骤：

1. 检查网络连接：

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

2. 检查日志中的错误：
    - 导航到：日志标签
    - 过滤：错误级别
    - 查找身份验证失败、□络超时

3. 验证凭据：
    - 导航到：SMPP → 客户端/服务器对等体
    - 检查系统 ID 和密码是否正确
    - 如果不确定，请联系运营商
4. 手动重新连接：
    - 导航到：SMPP → 实时状态
    - 找到断开的绑定
    - 点击“重新连接”按钮
    - 等待 10-30 秒
    - 检查状态是否更改为“已连接”
5. 如果���新连接失败：
    - 检查防火墙规则
    - 验证运营商服务器是否正常运行
    - 联系运营商支持
    - 请参见 TROUBLESHOOTING.md

处理身份验证失败

症状：日志中重复的绑定失败

原因：

- 用户名/密码不正确
- IP 未在运营商处列入白名单
- 账户已暂停/过期

步骤：

1. 验证凭据：
    - 导航到：SMPP → 客户端对等体
    - 仔细检查系统 ID 和密码
    - 与运营商确认
2. 检查 **IP** 白名单：
    - 确认您的□关 IP 与运营商一致
    - 请求运营商验证 IP 白名单
3. 检查账户状态：
    - 验证账户是否处于活动状态
    - 检查合同是否过期
    - 联系运营商账单部门
4. 更新□置：
    - 如果凭据更改，请在 Web UI 中更新
    - 点击“重新连接”以使用新凭据重试

监控和警报

检查 **Prometheus** 指标

快速检查：

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

预期输出：

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

所有值应为 `1`（已连接）。

响应警报

连接中断警报：

1. 检查 Web UI → SMPP → 实时状态
2. ��试手动重新连接
3. 检查日志中的错误
4. 如果长时间中断，请联系运营商
5. 请参见 TROUBLESHOOTING.md

高失败率警报：

1. 检查日志中的错误模式
2. 查看最近的配置更改
3. 联系运营商以了解拒绝情况
4. 检查消息格式合规性

无流量警报：

1. 检查后端队列是否有消息
2. 验证 `dest_smsc` 路由是否正确
3. 检查 TPS 限制是否过于严格
4. 查看 `queue_check_frequency` 设置

维护程序

定期维护

每月执行：

1. 查看指标：
    - 分析消息量趋势
    - 检查交付成功率
    - 确定优化机会
2. 更新文档：
    - 记录任何配置更改
    - 更新联系信息
    - 记录运营商维护窗口
3. 凭据审计：
    - 审查所有 SMPP 密码
    - 计划凭据轮换
    - 验证 IP 白名单是否最新
4. 容量规划：
    - 查看高峰消息速率
    - 检查与 TPS 限制的比较
    - 规划增长

服务重启

何时需要：

- 在配置文件更改后
- 在系统更新后
- 在故障排除期间

步骤：

```
# 检查当前状态
sudo systemctl status omnimessage-smpp

# 重启服务
sudo systemctl restart omnimessage-smpp

# 验证重启
sudo systemctl status omnimessage-smpp

# 检查日志
sudo journalctl -u omnimessage-smpp -n 50
```

通过 **Web UI** 验证：

1. 访问仪表板（可能需要 30-60 秒才能上线）
2. 导航到：SMPP → 实时状态
3. 等待所有连接建立（1-2 分钟）
4. 检查日志以获取错误

置备份

在更改之前备份关键文件：

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

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

如有需要，恢复：

```
# 恢复配置
sudo cp /opt/omnimessage-smpp/config/runtime.exs.backup.YYYYMMDD \
  /opt/omnimessage-smpp/config/runtime.exs

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

紧急程序

完全服务中断

步骤：

1. 检查服务状态：

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

2. 如果服务停止，启动它：

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

3. 检查日志以获取崩溃原因：

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

4. 如果无法启动：

   - 检查配置语法错误
   - 验证 SSL 证书是否存在
   - 检查磁盘空间：`df -h`
   - 检查内存：`free -h`

5. 如果无法解决，请联系支持

运营商请求紧急断开

步骤：

1. 立即断开连接：
    - 导航到：SMPP → 实时状态
    - 找到受影响的连接
    - 点击“断开连接”
2. 记录原因：
    - 记录运营商名称
    - 记录时间和原因
    - 保存通信记录
3. 调查问题：
    - 检查最近的消息模式
    - 查看日志以获取错误
    - 确定根本原因
4. 协调解决方案：
    - 与运营商合作
    - 实施修复
    - 在重新连接之前进行测试

高流量峰值

症状：意外的高消息流量

步骤：

1. 检查 **TPS** 限制：
    - 导航到：SMPP → 实时状态
    - 验证连接是否未被限制
    - 可能需要临时增加 TPS 限制

2. 监控运营商稳定性：
    - 观察断开连接情况
    - 检查交付成功率
3. 与后端协调：
    - 验证消息源是否合法
    - 可能需要在上游实施速率限制
4. 如有需要，进行扩展：
    - 可能需要额外的□关实例
    - 联系支持以获取扩展建议

最佳实践

每日检查清单

- [ ] 检查所有 SMPP 连接是否已连接
- [ ] 查看错误日志以查找任何问题
- [ ] 监控消息队列是否积压
- [ ] 检查 Prometheus/Grafana 仪表板
- [ ] 验证交付成功率 > 98%

每周任务

- [ ] 查看指标趋势
- [ ] 检查模式异常
- [ ] 测试灾难恢复程序
- [ ] 根据需要更新文档
- [ ] 查看并确认警报

每月任务

- [ ] 凭据审计
- [ ] 容量规划审查
- [ ] 更新运营商联系人
- [ ] 查看并优化 TPS 设置
- [ ] 备份配置文件

相关文档

- **CONFIGURATION.md** - 配置连接和设置
- **MONITORING.md** - 设置 Prometheus 警报
- **TROUBLESHOOTING.md** - 解决常见问题
- **README.md** - 系统概述

故障排除指南

常见问题及解决方案

OmniMessage 连接问题

由于 SMPP 网关是无状态的，并且完全依赖于 OmniMessage Core，因此与 OmniMessage 的连接问题是最关键的问题。

OmniMessage 断开连接的症状

- 没有外发消息：队列积压，消息未发送
- 没有入站消息：合作伙伴无法提交消息
- 超时：API 调用超时或挂起
- 日志显示： "Connection refused", "Timeout", "HTTP 503", "Connection reset"

诊断

1. 检查 **OmniMessage** 可用性：

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

# 从网关主机专门测试
ssh gateway-server 'curl -k https://omnimessage-
core.example.com:8443/api/system/health'
```

2. 检查配置的 **API URL**：

```
# 查看配置
grep -A1 'api_base_url' /opt/omnimessage-smpp/config/runtime.exs

# 检查网络连接
ping omnimessage-core.example.com
nc -zv omnimessage-core.example.com 8443
```

3. 检查网关日志中的 **API** 错误:

```
# 查找与 API 相关的错误
sudo journalctl -u omnimessage-smpp -f | grep -i
'api\|omnimessage\|connect'

# 搜索日志中的最近错误
sudo journalctl -u omnimessage-smpp -n 200 | grep -i error
```

解决方案

如果 **OmniMessage** 停止工作:

1. 联系 OmniMessage 运营团队
2. 待处理的消息将积累在队列中
3. 关将继续重试（请参见 `SMPP_POLL_INTERVAL`）
4. 检查 OmniMessage 状态页面或监控

如果 **OmniMessage** 正常但网关无法连接:

1. 检查防火墙规则允许外发 HTTPS
2. 检查 DNS 解析: `nslookup omnimessage-core.example.com`
3. 检查络路由: `traceroute omnimessage-core.example.com`
4. 验证 SSL 证书（如果使用 HTTPS）

如果 **API URL** 置错误:

1. 编辑 `/opt/omnimessage-smpp/config/runtime.exs`
2. 验证 `api_base_url` 是否正确（生产环境必须为 HTTPS）
3. 重启关: `sudo systemctl restart omnimessage-smpp`

连接问题

连接无法建立

症状:

- 状态显示 "Disconnected" (红色)
- 日志中没有成功绑定
- 重复连接尝试

可能的原因和解决方案:

1. 网络连接问题

检查:

```
# 测试 DNS 解析
nslookup smpp.carrier.com

# 测试连接
ping -c 3 smpp.carrier.com

# 测试端口
telnet smpp.carrier.com 2775
# 或
nc -zv smpp.carrier.com 2775
```

解决方案:

- 如果 DNS 失败: 在配置中使用 IP 地址而不是主机名
- 如果 ping 失败: 检查防火墙规则，联系运营商
- 如果端口失败: 验证正确的端口号，检查防火墙

2. 凭证错误

检查:

- 日志显示 "bind failed" 或 "authentication error"
- Web UI: SMPP → 客户端对等体 → 验证 system_id 和密码

解决方案:

- 与运营商确认凭证
- 检查拼写错误（区分大小写）
- 更新配置并重新连接

3. IP 未列入白名单

检查:

- 连接立即被拒绝
- 运营商日志显示未经授权的 IP

解决方案:

- 确认您的□关公共 IP:

  ```
  curl ifconfig.me
  ```

- 请求运营商将 IP 添加到白名单
- 验证 IP 是否已更改（动态 IP）

4. 防火墙阻止

检查:

```
# 检查端口是否开放
sudo iptables -L -n | grep 2775

# 检查 UFW（Ubuntu/Debian）
sudo ufw status | grep 2775

# 检查 firewalld（RHEL/CentOS）
sudo firewall-cmd --list-ports | grep 2775
```

解决方案:

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

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

连接持续掉线

症状：

- 连接已建立但频繁断开
- `smpp_disconnection_total` 指标增加
- 日志显示重复重连

可能的原因和解决方案：

1. 网络不稳定

检查：

```
# 监控丢包
ping -c 100 smpp.carrier.com | grep loss

# 检查网络错误
netstat -s | grep -i error
```

解决方案：

- 联系运营商关于网络问题
- 如果在您这边，检查 ISP
- 考虑备用连接/路由

2. 查询链路超时

检查：

- 日志显示 "enquire_link timeout"
- 连接在不活动期间掉线

解决方案:

- 默认超时为 30 秒
- 验证□络允许保持活动数据包
- 检查是否有激进的防火墙超时空闲连接

3. 超过 TPS 限制

检查:

- 断开连接时消息速率高
- 运营商限制消息

解决方案:

- 查看 `tps_limit` 设置
- 将 TPS 降低到运营商最大值的 70-80%
- 在多个绑定之间分散流量

4. 运营商服务器问题

检查:

- 检查运营商服务状态
- 联系运营商支持

解决方案:

- 等待运营商解决
- 如果可用，配置备用运营商

消息投递问题

消息未发送

症状：

- 消息卡在队列中
- `smpp_messages_sent_total` 不增加
- 连接显示已连接

可能的原因和解决方案：

1. 错误的 dest_smsc 路���

检查：

- Web UI → 队列 → 检查消息 `dest_smsc` 字段
- 与 SMPP → 实时状态中的连接名称进行比较

解决方案：

- 消息根据 `dest_smsc` 字段路由
- 验证后端是否设置正确的 `dest_smsc`
- 如果 `dest_smsc` 为 NULL，检查默认路由

2. 消息计划在未来发送

检查：

- Web UI → 队列 → 检查 `deliver_after` 字段
- 带有未来时间戳的消息尚未发送

说明：

- 重试系统为失败的消息设置 `deliver_after`
- 消息将在该时间之前等待重试

解决方案：

- 等待计划时间
- 如果紧急，联系后端团队重置时间戳

3. TPS 限制过低

检查:

- 大量队列积压
- 消息发送非常缓慢

解决方案:

- 增加配置中的 `tps_limit`
- 验证运营商是否能处理更高的速率
- 查看 CONFIGURATION.md

4. 队列工作者未运行

检查:

- 服务状态
- 日志中的错误

解决方案:

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

# 检查日志
sudo journalctl -u omnimessage-smpp -f
```

高投递失败率

症状:

- `smpp_delivery_failures_total` 增加
- 日志显示 "submit_sm_resp" 带有错误状态

- 消息未到达收件人

可能的原因和解决方案：

1. 无效的目的地号码

检查：

- 日志中的特定错误代码
- 审查消息目的地格式

常见错误代码：

- `0x0000000B` - 无效目的地
- `0x00000001` - 无效消息长度
- `0x00000003` - 无效命令

解决方案：

- 验证号码格式（推荐使用 E.164）
- 检查号码是否包含国家代码
- 验证运营商要求

2. 无效的消息内容

检查：

- 消息长度
- 特殊字符
- 编码

解决方案：

- GSM-7：最大 160 个字符
- UCS-2：最大 70 个字符
- 删除不支持的字符
- 检查编码设置

3. 运营商拒绝

检查：

- 运营商的特定错误代码
- 被拒绝消息的模式

解决方案：

- 联系运营商以获取拒绝原因
- 可能需要内容过滤
- 检查是否存在垃圾邮件/滥用模式

4. 消息过期

检查：

- 消息 `expires` 时间戳
- 投递尝试的时间

解决方案：

- 增加消息有效期
- 减少时间敏感消息的重试延迟

Web UI 问题

无法访问 Web 仪表板

症状：

- 浏览器无法连接到 https://your-server:8087
- 超时或连接被拒绝

可能的原因和解决方案：

1. 服务未运行

检查：

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

解决方案���

```
# 如果停止，启动它
sudo systemctl start omnimessage-smpp

# 检查日志中的错误
sudo journalctl -u omnimessage-smpp -n 50
```

2. 防火墙阻止端口 8087

检查:

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

解决方案:

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

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

3. SSL 证书问题

检查:

- 浏览器显示安全警告
- 证书过期或无效

解决方案:

- 接受安全例外（如果是自签名）

- 安装有效的 SSL 证书
- 检查证书文件是否存在:

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

4. 错误的 URL

检查:

- 验证使用 HTTPS（而不是 HTTP）
- 验证正确的服务器 IP/主机名
- 验证端口 8087

Web UI 显示错误

症状:

- 页面加载但显示错误
- 功能无法正常工作
- 数据未显示

解决方案:

1. 清除浏览器缓存:
    - Ctrl+F5（强制刷新）
    - 清除浏览器缓存和 cookies
2. 检查浏览器控制台:
    - 按 F12
    - 检查控制台选项卡中的 JavaScript 错误
    - 如果发现错误，报告给支持团队
3. 尝试不同的浏览��:
    - 在 Chrome、Firefox、Edge 中测试
    - 隔离浏览器特定问题

4. 检查服务日志:

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

指标问题

Prometheus 指标不可用

症状:

- `curl http://localhost:4000/metrics` 失败
- Prometheus 无法抓取指标
- 空响应或错误响应

可能的原因和解决方案:

1. 服务未运行

检查:

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

解决方案:

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

2. 端口不可访问

检查:

```
# 本地测试
curl http://localhost:4000/metrics

# 远程测试
curl http://your-server-ip:4000/metrics
```

解决方案：

- 如果本地有效但远程无效：检查防火墙
- 在防火墙中为 Prometheus 服务器打开端口 4000

3. 错误的端点

验证：

- 端点为 `/metrics`（而不是 `/prometheus` 或 `/stats`）
- 端口为 4000（而不是 8087）

指标显示意外值

症状：

- 计数器重置为零
- 指标显示错误值
- 某些绑定缺少指标

解决方案：

1. 服务重启重置计数器：
    - 服务重启时计数器重置
    - 这是正常行为
    - 在 Prometheus 查询中使用 `increase()` 或 `rate()`
2. 新绑定未显示：
    - 指标仅在第一次事件后出现

- 发送测试消息以填充指标
- 检查绑定是否启用并已连接

3. 过期指标：

- 旧绑定可能仍在指标中显示
- 重启服务以清除过期条目
- 或使用 Prometheus 重标记进行过滤

性能问题

CPU 使用率高

检查：

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

可能的原因：

- 消息量非常大
- 连接过多
- 配置问题

解决方案：

- 检查消息速率是否在容量范围内
- 审查 TPS 限制
- 如果持续高 CPU，联系支持

内存使用率高

检查：

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

可能的原因：

- 内存中队列消息过大
- 内存泄漏（罕见）

解决方案：

- 重启服务以清除内存
- 检查消息队列大小
- 如果内存持续增长，联系支持

消息处理缓慢

症状：

- 消息发送时间长
- 队列积压
- 消息速率低

检查：

1. TPS 限制 - 可能过于严格
2. `queue_check_frequency` - 可能过高
3. 后端 API 响应时间 - 可能较慢
4. 到运营商的网络延迟

解决方案：

- 如果运营商允许，增加 TPS
- 减少 `queue_check_frequency` 以加快轮询
- 优化后端 API
- 检查网络延迟

□置问题

□置文件语法错误

症状:

- 配置更改后服务无法启动
- 日志显示 "syntax error" 或 "parse error"

检查:

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

常见错误:

- map 条目之间缺少逗号
- 引号不匹配（" vs '）
- 括号或大括号不匹配
- 顶部缺少 `import Config`

解决方案:

- 从备份恢复
- 仔细检查语法
- 使用具有 Elixir 语法高亮的文本编辑器

更改未生效

症状:

- 修改配置但行为没有变化
- 旧设置仍然有效

解决方案:

```
# 配置更改需要重启
sudo systemctl restart omnimessage-smpp

# 验证重启是否成功
sudo systemctl status omnimessage-smpp

# 检查日志中的错误
sudo journalctl -u omnimessage-smpp -n 50
```

紧急恢复

完全系统故障

步骤:

1. 检查基本系统健康:

   ```
   # 磁盘空间
   df -h

   # 内存
   free -h

   # CPU 负载
   uptime
   ```

2. 检查服务状态:

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

3. 查看最近日志:

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

4. 尝试重启服务：

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

5. 如果重启失败：
   - 检查配置语法
   - 验证 SSL 证书是否存在
   - 检查文件权限
   - 查看日志以获取具体错误
6. 从备份恢复（如有需要）：

```
# 恢复配置
sudo cp /opt/omnimessage-smpp/config/runtime.exs.backup \
  /opt/omnimessage-smpp/config/runtime.exs

# 重启
sudo systemctl restart omnimessage-smpp
```

7. 如果未解决，联系支持

获取帮助

收集信息

在联系支持之前，请收集：

1. 版本：`cat /opt/omnimessage-smpp/VERSION`
2. 最近日志：

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

3. □置（清除密码）：

```
sudo cp /opt/omnimessage-smpp/config/runtime.exs
/tmp/config.exs
# 编辑 /tmp/config.exs 以在发送之前删除密码
```

4. 指标输出：

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

5. 系统信息：

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

联系支持

- 电子邮件：support@omnitouch.com
- 电话：+61 XXXX XXXX（24/7）
- 包括：上述所有信息

相关文档

- **OPERATIONS.md** - 正常操作程序
- **CONFIGURATION.md** - 配置参考
- **MONITORING.md** - 监控和指标
- **README.md** - 系统概述