Configuration Reference

Complete guide to all configuration parameters

Architecture Overview

The OmniMessage SMPP Gateway is a **stateless protocol frontend** that translates SMPP messages to/from OmniMessage. All business logic, routing decisions, and message storage are handled by OmniMessage Core - the gateway simply:

1. **Receives** SMPP PDUs from carriers and clients
2. **Translates** them to OmniMessage format via REST API
3. **Polls** OmniMessage for messages to send
4. **Sends** SMPP PDUs to carriers
5. **Reports** delivery status back to OmniMessage

This is identical to how other OmniMessage frontends (Diameter, MAP, IMS) work - they're all stateless protocol translators that delegate to OmniMessage Core.

Configuration File Location

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

Important: After changing configuration, restart the gateway:

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

Configuration Structure

The configuration file uses Elixir syntax. Basic structure:

```
import Config

# Global settings
config :omnimessage_smpp,
  setting_name: value

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

Global Settings

###API_BASE_URL

OmniMessage Core platform URL

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

Parameter	Type	Required	Default
`api_base_url`	String (URL)	Yes	-

Purpose: URL of the OmniMessage Core platform. The gateway communicates with OmniMessage via REST API for all message processing:

- **Submit Messages**: Send received SMPP messages to OmniMessage for processing
- **Retrieve Messages**: Poll for messages destined for SMPP carriers
- **Report Delivery Status**: Update message delivery status back to OmniMessage
- **System Health**: Periodic health checks

Critical: This is where the gateway gets all its "brains". OmniMessage handles:

- ✓ Message validation and format checking
- ✓ Routing decisions (which carrier to use)
- ✓ Rate limiting and throttling
- ✓ Number validation
- ✓ Message storage and persistence
- ✓ Delivery retry logic
- ✓ Status tracking

The gateway simply translates SMPP ↔ OmniMessage format.

Examples:

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

# HTTPS with hostname
api_base_url: "https://omnimessage-core.company.com:8443"

# HTTP (not recommended for production)
api_base_url: "http://192.168.1.100:8080"
```

Network Requirements:

- Gateway must have network access to OmniMessage Core
- Use HTTPS in production (configure `verify_ssl_peer`)
- Firewall must allow outbound HTTPS on specified port

SMPP_POLL_INTERVAL

Queue check frequency (milliseconds)

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

Parameter	Type	Required	Default
`smpp_poll_interval`	Integer	No	100

Purpose: How often (in milliseconds) each client checks the message queue.

Guidelines:

- **High volume (>100 TPS)**: 100-500ms
- **Medium volume (10-100 TPS)**: 500-1000ms
- **Low volume (<10 TPS)**: 1000-2000ms

Environment variable: `SMPP_POLL_INTERVAL`

VERIFY_SSL_PEER

SSL certificate verification

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

Parameter	Type	Required	Default
`verify_ssl_peer`	Boolean	No	false

Purpose: Whether to verify SSL certificates when connecting to backend API.

Values:

- `true`: Verify certificates (production with valid certs)
- `false`: Skip verification (self-signed certs, testing)

Environment variable: `VERIFY_SSL_PEER`

SMSC_NAME

Gateway identifier for registration

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

Parameter	Type	Required	Default
`smsc_name`	String	No	"smpp_gateway"

Purpose: Identifies this gateway instance in the message queue backend.

Environment variable: `SMSC_NAME`

SMPP Client Bind Configuration

Client binds are **outbound connections** where the gateway acts as an **ESME** (client) connecting to carrier **SMSC** servers. In this mode, the gateway initiates the connection to send and receive messages through external carriers.

Complete Client Bind Example

```
config :omnimessage_smpp, :binds, [
  %{
    # Unique identifier for this connection
    name: "vodafone_uk",

    # Connection mode
    mode: :client,

    # SMPP bind type
    bind_type: :transceiver,

    # Carrier SMPP server address
    host: "smpp.vodafone.co.uk",
    port: 2775,

    # Authentication credentials
    system_id: "your_username",
    password: "your_password",

    # Rate limiting
    tps_limit: 100,

    # Queue check frequency
    queue_check_frequency: 1000
  }
]
```

Client Bind Parameters

name

Unique connection identifier

Type	Required	Example
String	Yes	`"vodafone_uk"`

Purpose: Uniquely identifies this SMPP connection.

- Used in logs and metrics
- Must be unique across all binds
- Use descriptive names (carrier, region, purpose)

Naming conventions:

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

mode

Connection type

Type	Required	Value
Atom	Yes	`:client`

Purpose: Defines this as an outbound connection where the gateway acts as an **ESME** connecting to an external **SMSC**.

Fixed value: Always `:client` for outbound connections.

bind_type

SMPP session type

Type	Required	Allowed Values
Atom	Yes	`:transmitter`, `:receiver`, `:transceiver`

Purpose: Defines message direction capability.

Options:

- `:transmitter` - Send messages only (submit_sm)
- `:receiver` - Receive messages only (deliver_sm)
- `:transceiver` - Send and receive (most common)

Recommendation: Use `:transceiver` unless carrier requires specific type.

host

Carrier SMPP server hostname or IP

Type	Required	Example
String	Yes	`"smpp.carrier.com"` or `"10.5.1.100"`

Purpose: Address of carrier's SMPP server.

Examples:

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

port

SMPP server port

Type	Required	Default	Range
Integer	Yes	2775	1-65535

Purpose: TCP port for SMPP connection.

Standard port: 2775

Examples:

```
port: 2775  # Standard
port: 3000  # Custom
```

system_id

Authentication username

Type	Required	Example
String	Yes	`"company_user"`

Purpose: Username provided by carrier for authentication.

Security: Protect this credential - stored in configuration file.

password

Authentication password

Type	Required	Example
String	Yes	`"secret_password"`

Purpose: Password provided by carrier for authentication.

Security:

- Protect this credential
- Use strong passwords
- Rotate periodically

tps_limit

Transactions per second limit

Type	Required	Default	Range
Integer	Yes	100	1-10000

Purpose: Maximum messages per second to send through this connection.

Guidelines:

- Set to 70-80% of carrier's maximum
- Prevents throttling/disconnection
- Allows headroom for delivery receipts

Examples:

```
tps_limit: 10    # Low volume
tps_limit: 50    # Medium volume
tps_limit: 100   # High volume (most common)
tps_limit: 1000  # Very high volume
```

Calculation:

```
If carrier max = 100 TPS
Set tps_limit = 70-80
Leaves 20-30 TPS headroom
```

queue_check_frequency

Message queue polling interval (milliseconds)

Type	Required	Default	Range
Integer	Yes	1000	100-10000

Purpose: How often to check backend for new messages to send.

Guidelines:

- **High volume (>100 TPS)**: 500-1000ms
- **Medium volume (10-100 TPS)**: 1000-2000ms
- **Low volume (<10 TPS)**: 2000-5000ms

Trade-offs:

- Lower value = faster message pickup, more API load
- Higher value = slower pickup, less API load

Web UI Example:

SMPP Server Bind Configuration

Server binds define **inbound connections** where the gateway acts as an **SMSC** (server) accepting connections from external **ESMEs** (clients). In this mode, partner systems connect to the gateway to send and receive messages.

Complete Server Bind Example

```
config :omnimessage_smpp, :server_binds, [
  %{
    # Unique identifier for this client
    name: "partner_acme",

    # Expected credentials from client
    system_id: "acme_corp",
    password: "acme_secret",

    # Allowed bind types
    allowed_bind_types: [:transmitter, :receiver, :transceiver],

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

    # Rate limiting
    tps_limit: 50,

    # Queue check frequency
    queue_check_frequency: 1000
  }
]
```

Server Bind Parameters

name

Client identifier

Type	Required	Example
String	Yes	`"partner_acme"`

Purpose: Identifies the external client connecting to you.

Naming conventions: Use partner/client name for easy identification.

system_id

Expected username from client

Type	Required	Example
String	Yes	`"acme_corp"`

Purpose: Username that external client must provide to authenticate.

Provide to client: Share this credential with your partner.

password

Expected password from client

Type	Required	Example
String	Yes	`"secure_password"`

Purpose: Password that external client must provide to authenticate.

Security:

- Use strong passwords
- Unique per client
- Share securely with partner

allowed_bind_types

Permitted session types

Type	Required	Default
List of Atoms	Yes	-

Purpose: Restricts what bind types the client can use.

Options:

```
allowed_bind_types: [:transceiver]  # Only transceiver
allowed_bind_types: [:transmitter, :receiver]  # TX or RX
allowed_bind_types: [:transmitter, :receiver, :transceiver]  # Any
```

Recommendation: Allow all three unless you need restrictions.

ip_whitelist

Allowed client IP addresses

Type	Required	Default	Format
List of Strings	Yes	`[]`	IPs or CIDR notation

Purpose: Security - only allow connections from known IPs.

Formats:

- Single IP: `"192.168.1.100"` (automatically `/32`)
- CIDR subnet: `"192.168.1.0/24"`, `"10.0.0.0/8"`
- Mix both: `["192.168.1.0/24", "10.50.1.100"]`

Examples:

```
# Allow any IP (not recommended)
ip_whitelist: []

# Single IP
ip_whitelist: ["203.0.113.50"]

# Multiple IPs
ip_whitelist: ["203.0.113.50", "203.0.113.51"]

# Subnet
ip_whitelist: ["192.168.1.0/24"]

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

Common subnets:

- `/32` - Single IP (automatic for IPs without mask)
- `/24` - 256 addresses (e.g., 192.168.1.0-255)
- `/16` - 65,536 addresses (e.g., 10.50.0.0-255.255)
- `/8` - 16,777,216 addresses (e.g., 10.0.0.0-255.255.255.255)

tps_limit

Messages per second limit

Same as client bind `tps_limit` - controls outbound deliver_sm rate.

queue_check_frequency

Queue polling interval

Same as client bind `queue_check_frequency` - how often to check for messages to deliver to this client.

Web UI Example:

Server Listen Configuration

When server binds are configured, gateway listens for incoming connections.

Complete Listen Example

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

Listen Parameters

host

IP address to bind to

Type	Required	Default	Common Values
String	No	`"0.0.0.0"`	`"0.0.0.0"`, `"127.0.0.1"`

Purpose: Which network interface to listen on.

Values:

- `"0.0.0.0"` - Listen on all interfaces (recommended)
- `"127.0.0.1"` - Listen on localhost only (testing)
- `"192.168.1.10"` - Listen on specific IP

port

TCP port to listen on

Type	Required	Default	Range
Integer	No	2775	1-65535

Purpose: Port for incoming SMPP connections.

Standard: 2775

max_connections

Maximum concurrent connections

Type	Required	Default	Range
Integer	No	100	1-10000

Purpose: Limits total number of simultaneous client connections.

Guidelines:

- Set based on expected clients
- Higher values use more memory
- Typical: 10-100 connections

Complete Configuration Examples

Example 1: Single Carrier Connection

```
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
  }
]
```

Example 2: Multiple Carriers

```
import Config

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

config :omnimessage_smpp, :binds, [
  # North America
  %{
    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
  },

  # Europe
  %{
    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
  }
]
```

Example 3: Gateway with Server Binds

```
import Config

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

# Outbound connections
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
  }
]

# Inbound client definitions
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
  }
]
```

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

Configuration Validation

After editing configuration, validate before restarting:

Syntax Check

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

If syntax is invalid, you'll see an error. Fix before restarting.

Test Configuration

```
# Restart in foreground to see errors
sudo -u omnimessage-smpp /opt/omnimessage-smpp/bin/omnimessage-smpp console
```

Press `Ctrl+C` twice to exit.

Security Best Practices

1. **Protect configuration file**:

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

2. **Use strong passwords**:
   - Minimum 12 characters
   - Mix letters, numbers, symbols
   - Unique per connection
3. **Use IP whitelists**:
   - Always configure `ip_whitelist` for server binds
   - Never use empty list `[]` in production
4. **Enable SSL verification**:
   - Set `verify_ssl_peer: true` with valid certificates
5. **Regular credential rotation**:
   - Change passwords quarterly
   - Coordinate with carriers/partners

Next Steps

- Review MONITORING.md for metrics configuration
- Read OPERATIONS.md for managing connections
- See TROUBLESHOOTING.md for common issues
- Return to README.md for overview

Glossary

Terms and Definitions

A

API (Application Programming Interface) Interface used to communicate with the message queue backend system.

Auto-Scroll Feature in the web UI Logs tab that automatically scrolls to show newest log entries.

B

Backend The message queue system that the SMPP Gateway connects to for retrieving and storing messages.

Bind An SMPP connection between two systems. Can be transmitter, receiver, or transceiver.

Bind Type The type of SMPP session:

- **Transmitter**: Send messages only
- **Receiver**: Receive messages only
- **Transceiver**: Send and receive messages

Bind Failure When an SMPP authentication attempt fails, usually due to incorrect credentials or IP restrictions.

C

CIDR (Classless Inter-Domain Routing) Notation for specifying IP address ranges (e.g., `192.168.1.0/24` represents 256 IP addresses).

Client Bind An outbound SMPP connection where the gateway acts as an **ESME** connecting to an external **SMSC** (typically a carrier's SMPP server). In this mode, the gateway is the client.

Connection Status Current state of an SMPP bind:

- **Connected**: Active and operational
- **Disconnected**: Not connected
- **Reconnecting**: Attempting to establish connection

Counter A metric that only increases (resets on service restart), used for totals like messages sent.

D

Data Coding SMPP field specifying message character encoding (GSM-7, UCS-2, etc.).

Delivery Failure When a message cannot be delivered, indicated by an error response from the carrier.

Delivery Receipt (DLR) Confirmation from the carrier about message delivery status.

dest_smsc Field in message queue indicating which SMPP connection should handle the message.

Disconnection When an active SMPP connection is terminated, either intentionally or due to error.

E

Enquire Link SMPP keepalive message sent periodically to verify connection is active.

ESM Class SMPP field indicating message type and features.

ESME (External Short Message Entity) In SMPP terminology, the client application that connects to an SMSC to send or receive messages. When the gateway operates in **Client mode**, it acts as an ESME connecting to carrier SMSCs. When it operates in **Server mode**, it accepts connections from external ESMEs.

Exponential Backoff Retry strategy where wait time doubles after each failure (1min, 2min, 4min, 8min...).

F

Firewall Network security system that controls incoming and outgoing network traffic.

G

Gateway The SMPP Gateway application that bridges between message queue and mobile networks.

Gauge A metric that can increase or decrease, representing current value (e.g., connection status).

Grafana Popular visualization tool for displaying Prometheus metrics in dashboards.

GSM-7 Standard 7-bit character encoding for SMS, supporting up to 160 characters per message.

H

HTTP/HTTPS Protocols used for web communication. HTTPS is encrypted version.

I

IP Whitelist List of allowed IP addresses that can connect to the gateway (security feature).

ISDN (Integrated Services Digital Network) Numbering plan commonly used for telephone numbers.

J

(No terms)

K

Keepalive Periodic messages (enquire_link) sent to maintain connection and detect failures.

KPI (Key Performance Indicator) Measurable value indicating system performance (e.g., delivery success rate).

L

Label In Prometheus, key-value pairs attached to metrics for identification (e.g., `bind_name="vodafone_uk"`).

LiveView Phoenix framework technology used for real-time web UI updates.

M

Message Queue Backend system that stores messages waiting to be sent or received.

Metrics Quantitative measurements of system performance, exposed in Prometheus format.

MO (Mobile Originated) Messages sent from mobile phones to the gateway (inbound).

MT (Mobile Terminated) Messages sent from the gateway to mobile phones (outbound).

MSISDN (Mobile Station International Subscriber Directory Number) Standard format for mobile phone numbers.

N

NPI (Numbering Plan Indicator) SMPP field specifying the numbering scheme (e.g., ISDN).

O

Outbound Messages flowing from gateway to mobile networks.

Inbound Messages flowing from mobile networks to gateway.

P

PDU (Protocol Data Unit) Individual SMPP message packet (e.g., submit_sm, deliver_sm).

Prometheus Open-source monitoring system that collects and stores time-series metrics.

Q

Queue List of messages waiting to be processed or sent.

Queue Check Frequency How often (in milliseconds) the gateway polls the backend for new messages.

Queue Worker Component that retrieves messages from queue and sends via SMPP.

R

Rate Limiting Controlling message throughput to comply with carrier restrictions. See TPS.

Receiver SMPP bind type that only receives messages (deliver_sm).

Reconnect Re-establishing a disconnected SMPP connection.

Retry Attempting to send a failed message again, usually with exponential backoff.

S

Server Bind Configuration that allows external **ESMEs** (clients) to connect to the gateway. In this mode, the gateway acts as an **SMSC** (server) accepting inbound connections from partner systems.

Session Active SMPP connection between two systems.

SMPP (Short Message Peer-to-Peer) Industry-standard protocol for exchanging SMS messages between systems.

SMSC (Short Message Service Center) In SMPP terminology, the server component that accepts connections from ESMEs (clients) and handles SMS message routing and delivery. When the gateway operates in **Server mode**, it acts as an SMSC accepting connections from external ESMEs.

SSL/TLS Encryption protocols for secure communication.

Submit_SM SMPP PDU for submitting a message for delivery.

Submit_SM_Resp SMPP response to submit_sm, indicating success or failure.

System ID Username used for SMPP authentication.

T

Telemetry Automated collection and transmission of system metrics.

TON (Type of Number) SMPP field specifying number format (e.g., international, national).

TPS (Transactions Per Second) Rate limit for maximum messages per second through a connection.

Transceiver SMPP bind type that can both send and receive messages (most common).

Transmitter SMPP bind type that only sends messages (submit_sm).

Throughput Message processing rate, typically measured in messages per second.

U

UCS-2 16-bit Unicode character encoding for SMS, supporting up to 70 characters per message.

Uptime Duration that a connection or service has been continuously operational.

V

Validity Period Time limit for message delivery attempt before expiration.

W

Web Dashboard Browser-based user interface for monitoring and managing the gateway.

Whitelist See IP Whitelist.

X

(No terms)

Y

(No terms)

Z

(No terms)

Acronym Quick Reference

Acronym	Full Term
API	Application Programming Interface
CIDR	Classless Inter-Domain Routing
DLR	Delivery Receipt
ESME	External Short Message Entity
GSM	Global System for Mobile Communications
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
ISDN	Integrated Services Digital Network
KPI	Key Performance Indicator
MO	Mobile Originated
MSISDN	Mobile Station International Subscriber Directory Number
MT	Mobile Terminated
NPI	Numbering Plan Indicator
PDU	Protocol Data Unit
SMPP	Short Message Peer-to-Peer
SMSC	Short Message Service Center

Acronym	Full Term
SMS	Short Message Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TON	Type of Number
TPS	Transactions Per Second
UCS	Universal Coded Character Set
UI	User Interface
URL	Uniform Resource Locator

Related Documentation

- **README.md** - System overview and getting started
- **CONFIGURATION.md** - Configuration parameters explained
- **OPERATIONS.md** - Day-to-day operations
- **MONITORING.md** - Metrics and monitoring
- **TROUBLESHOOTING.md** - Problem resolution

Monitoring and Metrics Guide

Complete reference for monitoring the SMPP Gateway

Overview

The SMPP Gateway exposes metrics in Prometheus format for monitoring connection health, message throughput, and system performance.

Critical: Since the gateway is stateless and depends on OmniMessage Core, **OmniMessage connectivity is the most important metric to monitor**. Monitor both:

1. **SMPP Gateway metrics** - Protocol-level health
2. **OmniMessage API metrics** - Backend connectivity and health

Metrics Endpoint

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

Format: Prometheus text format

Access: Open to localhost by default (configure firewall for remote access)

Quick Test

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

Available Metrics

All metrics are prefixed with `smpp_` and include labels for identification.

License Metrics

omnimessage_smpp_license_status

Type: Gauge **Description**: Current license status **Values**:

- `1` = Valid license
- `0` = Invalid/expired license

Labels: None

Example:

```
omnimessage_smpp_license_status 1
```

Use:

- Alert when value is 0 (invalid license)
- When license is invalid, outbound queue processing stops but SMPP binds remain connected
- Web UI remains accessible for troubleshooting

Product Name: `omnimessage_smpp`

Notes:

- When license is invalid (`license_status == 0`), the gateway stops processing outbound queues
- SMPP binds (both client and server) remain connected and accept bind requests
- Inbound messages are still received but not processed
- UI and monitoring remain accessible regardless of license status

Alerting Example:

```
- alert: SMPP_License_Invalid
  expr: omnimessage_smpp_license_status == 0
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: "SMPP Gateway license invalid or expired"
    description: "License status is invalid - outbound message processing is blocked"
```

Connection Status Metrics

smpp_connection_status

Type: Gauge **Description**: Current connection status of SMPP bind **Values**:

- `1` = Connected
- `0` = Disconnected

Labels:

- `bind_name` - Connection name (e.g., "vodafone_uk")
- `mode` - Connection type ("client" or "server")
- `host` - Remote host (client mode only)
- `port` - Remote port (client mode only)
- `bind_type` - SMPP bind type (client mode only)
- `system_id` - System ID used

Example:

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

Use:

- Alert when value is 0 (disconnected)
- Track connection uptime percentage
- Monitor reconnection frequency

Message Counters

smpp_messages_sent_total

Type: Counter **Description**: Total number of messages sent through SMPP bind **Unit**: Messages

Labels: Same as connection_status

Example:

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

Use:

- Calculate message rate (messages/second)
- Track daily/monthly volume
- Compare actual vs expected throughput

smpp_messages_received_total

Type: Counter **Description**: Total number of messages received through SMPP bind **Unit**: Messages

Labels: Same as connection_status

Example:

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

Use:

- Monitor inbound message volume
- Track mobile-originated (MO) traffic
- Alert on unexpected volume changes

Delivery Metrics

smpp_delivery_failures_total

Type: Counter **Description**: Total number of message delivery failures **Unit**: Failures

Labels: Same as connection_status

Example:

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

Use:

- Calculate delivery success rate
- Alert on high failure rates
- Identify problematic connections

Success Rate Calculation:

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

Bind Operation Metrics

smpp_bind_success_total

Type: Counter **Description**: Total number of successful bind operations **Unit**: Bind attempts

Example:

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

Use:

- Track bind stability
- Monitor authentication success

smpp_bind_failures_total

Type: Counter **Description**: Total number of failed bind operations **Unit**: Bind attempts

Example:

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

Use:

- Alert on authentication failures
- Identify credential issues
- Track carrier connection problems

Connection Event Metrics

smpp_connection_attempts_total

Type: Counter **Description**: Total number of connection attempts **Unit**: Attempts

Example:

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

Use:

- Track connection churn
- Monitor reconnection frequency

smpp_disconnection_total

Type: Counter **Description**: Total number of disconnections **Unit**: Disconnections

Example:

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

Use:

- Alert on frequent disconnections
- Identify network issues
- Track connection stability

Uptime Metrics

smpp_uptime_seconds

Type: Gauge **Description**: Current uptime of SMPP bind in seconds **Unit**: Seconds

Example:

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

Use:

- Track connection stability
- Calculate uptime percentage
- Alert on recent restarts

OmniMessage API Health Metrics

While the gateway itself exposes SMPP-related metrics, **OmniMessage API health is critical**. You should also monitor:

From OmniMessage Metrics (if available)

- `omnimessage_api_requests_total` - Total API requests from gateway
- `omnimessage_api_request_duration_seconds` - API response times
- `omnimessage_queue_depth` - Messages pending in OmniMessage queue

From Gateway Logs (if metrics not exposed)

Look for these patterns to detect API issues:

- "api.*connection refused" - Cannot reach OmniMessage
- "api.*timeout" - OmniMessage not responding
- "api.*http 503" - OmniMessage temporarily down
- "api.*parse error" - Response format issue

Prometheus Configuration

Basic Scrape Config

Add to `/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'
```

Multiple Gateways

```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'
```

Service Discovery

Using file-based discovery:

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

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

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

Grafana Dashboards

Sample Dashboard Panels

Connection Status Panel

Query:

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

Visualization: Stat **Thresholds**:

- Red: value < 1 (disconnected)
- Green: value == 1 (connected)

Message Rate Panel

Query:

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

Visualization: Graph **Unit**: messages/second **Legend**: `{{bind_name}}`

Delivery Success Rate Panel

Query:

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

Visualization: Gauge **Unit**: Percent (0-100) **Thresholds**:

- Red: < 95%
- Yellow: 95-98%

- Green: > 98%

Connection Uptime Panel

Query:

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

Visualization: Stat **Unit**: Hours

Alerting Rules

Prometheus Alert Rules

Save to `/etc/prometheus/rules/smpp-alerts.yml`:

```yaml
groups:
  - name: smpp_gateway
    interval: 30s
    rules:
      # Connection down
      - alert: SMPPConnectionDown
        expr: smpp_connection_status == 0
        for: 2m
        labels:
          severity: critical
        annotations:
          summary: "SMPP connection {{ $labels.bind_name }} is down"
          description: "Connection {{ $labels.bind_name }} has been disconnected for more than 2 minutes."

      # High failure rate
      - alert: SMPPHighFailureRate
        expr: |
          (
            rate(smpp_delivery_failures_total[5m])
            /
            rate(smpp_messages_sent_total[5m])
          ) > 0.05
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "High delivery failure rate on {{ $labels.bind_name }}"
          description: "Delivery failure rate is {{ $value | humanizePercentage }} on {{ $labels.bind_name }}."

      # Bind failures
      - alert: SMPPBindFailures
        expr: increase(smpp_bind_failures_total[10m]) > 3
        labels:
          severity: warning
        annotations:
          summary: "Multiple bind failures on {{ $labels.bind_name }}"
          description: "{{ $labels.bind_name }} has failed to bind {{ $value }} times in the last 10 minutes."
      # No messages sent (when expected)
      - alert: SMPPNoTraffic
        expr: rate(smpp_messages_sent_total[10m]) == 0
        for: 30m
        labels:
          severity: warning
        annotations:
          summary: "No messages sent on {{ $labels.bind_name }}"
          description: "{{ $labels.bind_name }} has not sent any messages for 30 minutes."

      # Frequent disconnections
      - alert: SMPPFrequentDisconnections
        expr: increase(smpp_disconnection_total[1h]) > 5
        labels:
          severity: warning
        annotations:
          summary: "Frequent disconnections on {{ $labels.bind_name }}"
          description: "{{ $labels.bind_name }} has disconnected {{ $value }} times in the last hour."

      # OmniMessage API unreachable
      - alert: OmniMessageAPIUnreachable
        expr: |
          count(count_over_time({job="omnimessage-smpp"} |= "api.*connection refused"[5m])) > 0
        for: 1m
        labels:
          severity: critical
        annotations:
          summary: "OmniMessage API is unreachable"
          description: "The SMPP Gateway cannot reach OmniMessage API. Check API_BASE_URL configuration and network connectivity."

      # OmniMessage API timeouts
      - alert: OmniMessageAPITimeout
        expr: |
          count(count_over_time({job="omnimessage-smpp"} |= "api.*timeout"[5m])) > 5
        for: 2m
        labels:
          severity: warning
        annotations:
          summary: "OmniMessage API is timing out"
          description: "Multiple API timeouts detected. OmniMessage may be slow or overloaded."

      # No message flow (API issue)
      - 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: "No message flow detected - check OmniMessage connectivity"
          description: "No messages sent or received for 30 minutes. Check OmniMessage API connectivity and queue status."
```

Load rules in `prometheus.yml`:

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

Web Dashboard Monitoring

The built-in web UI provides real-time monitoring without Prometheus.

Access

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

Live Status Page

Navigation: SMPP → Live Status

Features:

- Real-time connection status
- Message counters
- Connection uptime
- Manual reconnect/disconnect controls
- Auto-refresh every 5 seconds

Use:

- Quick status check
- Manual intervention
- Real-time troubleshooting

The dashboard displays:

- **Total Binds**: Combined count of all client and server connections
- **Client Binds**: Outbound connections to carriers (showing connected/disconnected count)
- **Server Binds**: Inbound connections from partners (showing active/waiting count)
- **Server Listening**: Configuration of the inbound server socket (host, port, max connections)

Log Monitoring

System Logs

View logs:

```
# Follow logs in real-time
sudo journalctl -u omnimessage-smpp -f

# Last 100 lines
sudo journalctl -u omnimessage-smpp -n 100

# Since specific time
sudo journalctl -u omnimessage-smpp --since "1 hour ago"

# Filter by level
sudo journalctl -u omnimessage-smpp -p err
```

Web UI Logs

Navigation: Logs tab in web UI

Features:

- Real-time log streaming
- Filter by level (debug, info, warning, error)
- Search logs
- Pause/resume
- Clear logs

The logs view allows you to:

- **Level Filter**: Select log level (All, Debug, Info, Warning, Error)
- **Search**: Find specific log entries by text content
- **Auto-scroll**: Enable/disable automatic scrolling as new logs arrive
- **Pause/Resume**: Pause log updates to review specific entries
- **Clear**: Clear all displayed logs

Key Performance Indicators (KPIs)

Connection Health

Metric: Connection uptime percentage

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

Target: > 99.9%

Message Delivery Rate

Metric: Messages delivered per second

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

Target: Matches expected volume

Delivery Success Rate

Metric: Percentage of successful deliveries

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

Target: > 98%

Bind Stability

Metric: Bind attempts per hour

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

Target: < 10 per hour (indicates stable connection)

Monitoring Best Practices

1. Set Up Alerts

- Configure Prometheus alerts for critical metrics
- Use PagerDuty/OpsGenie for 24/7 alerting
- Test alerts regularly

2. Create Dashboards

- Build Grafana dashboards for each gateway

- Include all connections on one dashboard
- Add capacity planning panels

3. Regular Reviews

- Review metrics weekly
- Identify trends and patterns
- Plan capacity adjustments

4. Document Baselines

- Record normal message volumes
- Document expected TPS rates
- Note peak times/days

5. Correlate with Backend

- Monitor backend API metrics
- Track end-to-end message flow
- Identify bottlenecks

Troubleshooting with Metrics

Connection Issues

Check: `smpp_connection_status`

- Value 0 = Review logs, check network, verify credentials
- Frequent changes = Network instability

Poor Delivery Rates

Check: `smpp_delivery_failures_total`

- High rate = Check carrier status, review message format
- Compare across connections = Identify problem carrier

Low Throughput

Check: `smpp_messages_sent_total` rate

- Below expected = Check TPS limits, queue availability
- Check backend API metrics

Bind Problems

Check: `smpp_bind_failures_total`

- Increasing = Authentication issues, credential problems
- Check system_id and password in config

Related Documentation

- **CONFIGURATION.md** - Configure monitoring settings
- **OPERATIONS.md** - Operational procedures
- **TROUBLESHOOTING.md** - Resolve issues
- **README.md** - Overview and quickstart

Operations Guide

Day-to-day operational procedures

Critical Dependency: OmniMessage Core

IMPORTANT: The OmniMessage SMPP Gateway cannot function without access to OmniMessage Core. All message processing happens in OmniMessage - the gateway is just a protocol translator.

If OmniMessage becomes unavailable:

- □ New messages cannot be submitted
- □ Pending messages cannot be retrieved
- □ Delivery status cannot be reported
- □ System appears to hang or timeout

Check OmniMessage Health:

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

# Check configured API URL in logs
grep api_base_url /opt/omnimessage-smpp/config/runtime.exs
```

Daily Operations

Morning Health Check

Perform these checks at the start of each day:

1. **Access Web Dashboard**

   - URL: `https://your-server:8087`
   - Check if dashboard loads properly

2. **Check Connection Status**

   - Navigate to: SMPP → Live Status
   - Verify all connections show "Connected" (green)
   - Note any disconnected binds

3. **Review Message Metrics**

   - Navigate to: Queue tab
   - Check message counts are reasonable
   - Verify no unexpected queue buildup

4. **Check System Logs**

   - Navigate to: Logs tab
   - Look for error messages (red)
   - Note any warning patterns

5. **Review Prometheus Metrics**

   - `curl http://localhost:4000/metrics`
   - Or check Grafana dashboards
   - Verify message rates are normal

Continuous Monitoring

Set up alerts for:

- Connection failures (> 2 minutes down)
- High delivery failure rates (> 5%)
- No traffic for extended periods
- Frequent disconnections

See MONITORING.md for alert configuration.

Managing SMPP Connections

How SMPP Peers Are Configured

SMPP connections (peers) can be configured using **two methods**:

Method 1: Web UI (Recommended)

- **Advantage**: Changes take effect immediately, no restart required
- **Location**: SMPP → Client Peers / Server Peers tabs
- **Operations**: Add, edit, delete peers
- **Persistence**: Stored in Mnesia database
- **Best for**: Day-to-day operations, testing, quick changes

Method 2: Configuration File

- **Advantage**: Configuration as code, version control
- **Location**: `/opt/omnimessage-smpp/config/runtime.exs`
- **Operations**: Define peers in Elixir configuration
- **Persistence**: File-based, survives restarts
- **Requires**: Service restart after changes
- **Best for**: Initial setup, infrastructure as code

Note: Web UI changes are stored separately and override configuration file settings.

See CONFIGURATION.md for configuration file reference.

Adding a New Client Connection

Purpose: Configure the gateway to act as an **ESME** (client) connecting to a carrier's **SMSC** (server)

Preparation: Gather information from carrier:

- SMPP server hostname/IP
- Port number (usually 2775)
- System ID (username)
- Password
- Bind type (usually transceiver)
- TPS limit

Choose one of the following methods:

Option A: Via Web UI (Recommended)

Advantages: Immediate effect, no restart required

Steps:

1. **Navigate to Client Peers**:
    - Open Web UI: `https://your-server:8087`
    - Navigate to: SMPP → Client Peers
2. **Add New Peer**:
    - Click "Add New Client Peer"
    - Fill in the form:
        - **Name**: `vodafone_uk` (unique identifier)
        - **Host**: `smpp.vodafone.co.uk`
        - **Port**: `2775`

        - **System ID**: `your_username`
        - **Password**: `your_password`
        - **Bind Type**: `Transceiver`
        - **TPS Limit**: `100`
        - **Queue Check Frequency**: `1000`
    - Click "Save"

3. **Connection Establishes Automatically**:

    - Gateway immediately attempts connection
    - Navigate to: SMPP → Live Status
    - Status should change to "Connected" (green) within 10-30 seconds
    - Check Logs tab for successful bind message

4. **Test Message Flow**:

    - Navigate to: Queue tab
    - Submit test message with dest_smsc matching bind name
    - Monitor in Live Status for transmission
    - Verify delivery confirmation

Option B: Via Configuration File

Advantages: Infrastructure as code, version control

Steps:

1. **Edit Configuration File**:

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

2. **Add New Bind to Configuration**:

   ```
   config :omnimessage_smpp, :binds, [
     # Existing binds...

     # Add new bind
     %{
       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. **Save and Restart Service**:

   ```
   # Save file (Ctrl+X, Y, Enter in nano)

   # Restart service
   sudo systemctl restart omnimessage-smpp
   ```

4. **Verify Connection**:

   - Navigate to: SMPP → Live Status
   - Find new connection
   - Status should be "Connected" (green)
   - Check logs for successful bind

5. **Test Message Flow**:

    - Navigate to: Queue tab
    - Submit test message with dest_smsc matching new bind name
    - Monitor in Live Status for transmission
    - Verify delivery confirmation

Adding a Server Bind

Purpose: Configure the gateway to act as an **SMSC** (server) accepting connections from external **ESMEs** (partner clients)

Preparation:

1. **Generate Credentials**:

    - Create unique system_id: `partner_name`
    - Create strong password
    - Document and share securely with partner

2. **Get Partner Information**:

    - Partner's source IP addresses
    - Expected message volume (for TPS limit)
    - Required bind types

Choose one of the following methods:

Option A: Via Web UI (Recommended)

Advantages: Immediate effect, no restart required

Steps:

1. **Navigate to Server Peers**:

    - Open Web UI: `https://your-server:8087`
    - Navigate to: SMPP → Server Peers

2. **Add New Server Peer**:

    - Click "Add New Server Peer"

   - Fill in the form:
      - **Name**: `partner_acme` (unique identifier)
      - **System ID**: `acme_corp`
      - **Password**: `secure_password_123`
      - **Allowed Bind Types**: Select all (Transmitter, Receiver, Transceiver)
      - **IP Whitelist**: `203.0.113.0/24` (comma-separated for multiple)
      - **TPS Limit**: `50`
      - **Queue Check Frequency**: `1000`
   - Click "Save"

3. **Gateway Ready for Connection**:

   - Server peer is now active and waiting for partner connection
   - No restart required

4. **Share Information with Partner**:

   - Gateway IP address
   - Port: `2775`
   - System ID: `acme_corp`
   - Password: `secure_password_123`
   - Bind Type: As configured

5. **Wait for Partner Connection**:

    - Navigate to: SMPP → Live Status
    - Watch for incoming connection
    - Verify authentication success
    - Check IP matches whitelist

Option B: Via Configuration File

Advantages: Infrastructure as code, version control

Steps:

1. **Edit Configuration File**:

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

2. **Add Server Bind and Listen Configuration**:

```
# Add to server_binds list
config :omnimessage_smpp, :server_binds, [
  # Existing server binds...

  # Add new server bind
  %{
    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
  }
]

# Ensure listen configuration exists (only needed once)
config :omnimessage_smpp, :listen, %{
  host: "0.0.0.0",
  port: 2775,
  max_connections: 100
}
```

3. **Save and Restart Service**:

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

4. **Share Information with Partner**:

   - Gateway IP address
   - Port: `2775`
   - System ID: `acme_corp`
   - Password: `secure_password_123`
   - Bind Type: As configured

5. **Wait for Partner Connection**:

   - Navigate to: SMPP → Live Status

- Watch for incoming connection
- Verify authentication success
- Check IP matches whitelist

Modifying Existing Connection

Purpose: Update connection parameters (TPS limits, passwords, IP whitelist, etc.)

Choose one of the following methods:

Option A: Via Web UI (Recommended)

Advantages: Immediate effect, no restart required

Steps:

1. **Navigate to Peers**:

    - Open Web UI: `https://your-server:8087`
    - For client connections: SMPP → Client Peers
    - For server connections: SMPP → Server Peers

2. **Edit Peer**:

    - Find the peer to modify
    - Click "Edit" button
    - Update desired parameters:
        - Common changes: TPS limit, password, IP whitelist, host/port
    - Click "Save"

3. **Changes Apply Immediately**:

    - Connection automatically reconnects with new settings
    - No service restart required
    - Navigate to: SMPP → Live Status to verify

4. **Verify Changes**:

    - Check connection establishes successfully
    - Monitor Logs tab for errors
    - Test message flow if applicable

Option B: Via Configuration File

Advantages: Infrastructure as code, version control

Steps:

1. **Edit Configuration File**:

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

2. **Modify Bind Parameters**:

    - Find the bind in the `:binds` or `:server_binds` list
    - Update desired parameters:
        - Common changes: TPS limit, passwords, IP whitelist, host/port
    - Example:

        ```
        %{
          name: "vodafone_uk",
          # ... other params
          tps_limit: 150,  # Changed from 100
          password: "new_password"  # Updated password
        }
        ```

3. **Save and Restart Service**:

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

4. **Verify Changes**:

    - Navigate to: SMPP → Live Status
    - Check connection establishes successfully
    - Monitor logs for errors

    - Test message flow

Removing a Connection

Purpose: Decommission an SMPP connection

Steps:

1. **Notify Stakeholders**:
    - Inform carrier/partner
    - Coordinate downtime window
2. **Disconnect via Web UI**:
    - Navigate to: SMPP → Live Status
    - Find connection
    - Click "Drop Connection"
    - Confirm action
3. **Remove Configuration**:
    - Navigate to: SMPP → Client/Server Peers
    - Find connection
    - Click "Delete"
    - Confirm removal
4. **Verify Removal**:
    - Check Live Status - connection should be gone
    - Review logs for clean shutdown

Managing Message Flow

Checking Message Queue

Purpose: Monitor pending messages

Steps:

1. **Access Queue**:
    - Navigate to: Queue tab
    - View list of pending messages

2. **Check Message Details**:
    - Click on message row
    - Review:
        - Destination number
        - Message body
        - Target SMSC (dest_smsc)
        - Delivery attempts
        - Status

3. **Search for Specific Message**:

  - Use search filter
  - Filter by destination, content, or SMSC

Troubleshooting Stuck Messages

Symptoms: Messages not being delivered

Steps:

1. **Check Connection Status**:
   - Navigate to: SMPP → Live Status
   - Verify target connection is connected
   - If disconnected, see Reconnecting
2. **Check Message Details**:
   - Navigate to: Queue tab
   - Find stuck message
   - Check `dest_smsc` field matches connection name
   - Check `deliver_after` timestamp (retry scheduling)
3. **Check Delivery Attempts**:
   - High attempts = repeated failures
   - Check logs for error messages
   - May indicate invalid format or carrier rejection
4. **Manual Intervention** (if needed):
   - Contact carrier to verify issue
   - May need to cancel and resubmit message
   - Check with backend team for queue issues

Connection Troubleshooting

Reconnecting a Bind

Symptoms: Connection shows "Disconnected" (red)

Steps:

1. **Check Network Connectivity**:

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

2. **Check Logs for Errors**:

   - Navigate to: Logs tab
   - Filter: Error level
   - Look for authentication failures, network timeouts

3. **Verify Credentials**:

   - Navigate to: SMPP → Client/Server Peers
   - Check system_id and password are correct
   - Contact carrier if unsure

4. **Manual Reconnect**:

    - Navigate to: SMPP → Live Status
    - Find disconnected bind
    - Click "Reconnect" button
    - Wait 10-30 seconds
    - Check if status changes to "Connected"

5. **If Reconnect Fails**:

    - Check firewall rules
    - Verify carrier server is operational
    - Contact carrier support
    - See TROUBLESHOOTING.md

Handling Authentication Failures

Symptoms: Repeated bind failures in logs

Causes:

- Incorrect username/password
- IP not whitelisted at carrier
- Account suspended/expired

Steps:

1. **Verify Credentials**:

    - Navigate to: SMPP → Client Peers
    - Double-check system_id and password
    - Confirm with carrier

2. **Check IP Whitelisting**:

    - Confirm your gateway IP with carrier
    - Request carrier verify IP whitelist

3. **Check Account Status**:
   - Verify account is active
   - Check for expired contracts
   - Contact carrier billing
4. **Update Configuration**:
   - If credentials changed, update in Web UI
   - Click "Reconnect" to retry with new credentials

Monitoring and Alerting

Checking Prometheus Metrics

Quick check:

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

Expected output:

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

All values should be `1` (connected).

Responding to Alerts

Connection Down Alert:

1. Check Web UI → SMPP → Live Status
2. Attempt manual reconnect
3. Check logs for errors
4. Contact carrier if prolonged outage

5. See TROUBLESHOOTING.md

High Failure Rate Alert:

1. Check logs for error patterns
2. Review recent configuration changes
3. Contact carrier about rejections
4. Check message format compliance

No Traffic Alert:

1. Check backend queue has messages
2. Verify `dest_smsc` routing is correct
3. Check TPS limits aren't too restrictive
4. Review `queue_check_frequency` setting

Maintenance Procedures

Routine Maintenance

Perform monthly:

1. **Review Metrics**:
   - Analyze message volume trends
   - Check delivery success rates
   - Identify optimization opportunities
2. **Update Documentation**:
   - Document any configuration changes
   - Update contact information
   - Note carrier maintenance windows
3. **Credential Audit**:
   - Review all SMPP passwords

    - Plan credential rotation
    - Verify IP whitelists are current

4. **Capacity Planning**:

    - Review peak message rates
    - Check against TPS limits
    - Plan for growth

Service Restart

When needed:

- After configuration file changes
- After system updates
- During troubleshooting

Steps:

```
# Check current status
sudo systemctl status omnimessage-smpp

# Restart service
sudo systemctl restart omnimessage-smpp

# Verify restart
sudo systemctl status omnimessage-smpp

# Check logs
sudo journalctl -u omnimessage-smpp -n 50
```

Verify via Web UI:

1. Access dashboard (may take 30-60 seconds to come online)
2. Navigate to: SMPP → Live Status
3. Wait for all connections to establish (1-2 minutes)
4. Check logs for errors

Configuration Backup

Backup critical files before changes:

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

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

Restore if needed:

```
# Restore configuration
sudo cp /opt/omnimessage-smpp/config/runtime.exs.backup.YYYYMMDD \
  /opt/omnimessage-smpp/config/runtime.exs

# Restart service
sudo systemctl restart omnimessage-smpp
```

Emergency Procedures

Complete Service Outage

Steps:

1. **Check service status**:

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

2. **If service stopped, start it**:

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

3. **Check logs for crash reason**:

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

4. **If won't start**:

   - Check configuration syntax errors
   - Verify SSL certificates exist
   - Check disk space: `df -h`
   - Check memory: `free -h`

5. **Contact support** if unresolved

Carrier Requests Emergency Disconnect

Steps:

1. **Drop connection immediately**:

   - Navigate to: SMPP → Live Status
   - Find affected connection
   - Click "Drop Connection"

2. **Document reason**:

   - Note carrier name
   - Record time and reason
   - Save correspondence

3. **Investigate issue**:

   - Check recent message patterns
   - Review logs for errors
   - Identify root cause

4. **Coordinate resolution**:

   - Work with carrier

    - Implement fixes
    - Test before reconnecting

High Volume Spike

Symptoms: Unexpectedly high message traffic

Steps:

1. **Check TPS limits**:
    - Navigate to: SMPP → Live Status
    - Verify connections aren't throttling
    - May need to increase TPS limits temporarily
2. **Monitor carrier stability**:
    - Watch for disconnections
    - Check delivery success rates
3. **Coordinate with backend**:
    - Verify message source is legitimate
    - May need to implement rate limiting upstream
4. **Scale if needed**:
    - May need additional gateway instances
    - Contact support for scaling advice

Best Practices

Daily Checklist

- □ Check all SMPP connections are connected
- □ Review error logs for any issues

- □ Monitor message queue for buildup
- □ Check Prometheus/Grafana dashboards
- □ Verify delivery success rates > 98%

Weekly Tasks

- □ Review metrics trends
- □ Check for pattern anomalies
- □ Test disaster recovery procedures
- □ Update documentation as needed
- □ Review and acknowledge alerts

Monthly Tasks

- □ Credential audit
- □ Capacity planning review
- □ Update carrier contacts
- □ Review and optimize TPS settings
- □ Backup configuration files

Related Documentation

- **CONFIGURATION.md** - Configure connections and settings
- **MONITORING.md** - Set up Prometheus alerting
- **TROUBLESHOOTING.md** - Resolve common issues
- **README.md** - System overview

Troubleshooting Guide

Common issues and solutions

OmniMessage Connectivity Issues

Since the SMPP Gateway is stateless and depends entirely on OmniMessage Core, connectivity problems with OmniMessage are the most critical issues.

Symptoms of OmniMessage Disconnection

- **No outbound messages**: Queue builds up, messages not being sent
- **No inbound messages**: Partners can't submit messages
- **Timeouts**: API calls timing out or hanging
- **Logs show**: "Connection refused", "Timeout", "HTTP 503", "Connection reset"

Diagnosis

1. Check OmniMessage Availability:

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

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

2. Check Configured API URL:

```
# Review the configuration
grep -A1 'api_base_url' /opt/omnimessage-smpp/config/runtime.exs

# Check for network connectivity
ping omnimessage-core.example.com
nc -zv omnimessage-core.example.com 8443
```

3. Check Gateway Logs for API Errors:

```
# Look for API-related errors
sudo journalctl -u omnimessage-smpp -f | grep -i
'api\|omnimessage\|connect'

# Search logs for recent errors
sudo journalctl -u omnimessage-smpp -n 200 | grep -i error
```

Solutions

If OmniMessage is down:

1. Contact OmniMessage operations team
2. Pending messages will accumulate in the queue
3. Gateway will keep retrying (see `SMPP_POLL_INTERVAL`)
4. Check OmniMessage status page or monitoring

If OmniMessage is up but gateway can't reach it:

1. Check firewall rules allow outbound HTTPS
2. Check DNS resolution: `nslookup omnimessage-core.example.com`
3. Check network routing: `traceroute omnimessage-core.example.com`
4. Verify SSL certificates if using HTTPS

If API URL is misconfigured:

1. Edit `/opt/omnimessage-smpp/config/runtime.exs`
2. Verify `api_base_url` is correct (must be HTTPS for production)
3. Restart gateway: `sudo systemctl restart omnimessage-smpp`

Connection Problems

Connection Won't Establish

Symptoms:

- Status shows "Disconnected" (red)
- No successful bind in logs
- Repeated connection attempts

Possible Causes & Solutions:

1. Network Connectivity Issues

Check:

```
# Test DNS resolution
nslookup smpp.carrier.com

# Test connectivity
ping -c 3 smpp.carrier.com

# Test port
telnet smpp.carrier.com 2775
# or
nc -zv smpp.carrier.com 2775
```

Solutions:

- If DNS fails: Use IP address instead of hostname in configuration
- If ping fails: Check firewall rules, contact carrier
- If port fails: Verify correct port number, check firewall

2. Incorrect Credentials

Check:

- Logs show "bind failed" or "authentication error"
- Web UI: SMPP → Client Peers → verify system_id and password

Solutions:

- Confirm credentials with carrier
- Check for typos (case-sensitive)
- Update configuration and reconnect

3. IP Not Whitelisted

Check:

- Connection rejected immediately
- Carrier logs show unauthorized IP

Solutions:

- Confirm your gateway's public IP:
  ```
  curl ifconfig.me
  ```
- Request carrier add IP to whitelist
- Verify IP hasn't changed (dynamic IP)

4. Firewall Blocking

Check:

```
# Check if port is open
sudo iptables -L -n | grep 2775

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

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

Solutions:

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

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

Connection Keeps Dropping

Symptoms:

- Connection established but frequently disconnects
- `smpp_disconnection_total` metric increasing
- Logs show repeated reconnections

Possible Causes & Solutions:

1. Network Instability

Check:

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

# Check network errors
netstat -s | grep -i error
```

Solutions:

- Contact carrier about network issues
- Check with ISP if on your end
- Consider backup connection/route

2. Enquire Link Timeout

Check:

- Logs show "enquire_link timeout"
- Connection drops after periods of inactivity

Solutions:

- Default timeout is 30 seconds
- Verify network allows keepalive packets
- Check for aggressive firewalls timing out idle connections

3. TPS Limit Exceeded

Check:

- High message rate at disconnect time
- Carrier throttling messages

Solutions:

- Review `tps_limit` setting
- Reduce TPS to 70-80% of carrier maximum
- Spread traffic across multiple binds

4. Carrier Server Issues

Check:

- Check carrier service status
- Contact carrier support

Solutions:

- Wait for carrier to resolve
- Configure backup carrier if available

Message Delivery Problems

Messages Not Being Sent

Symptoms:

- Messages stuck in queue
- `smpp_messages_sent_total` not increasing
- Connection shows connected

Possible Causes & Solutions:

1. Wrong dest_smsc Routing

Check:

- Web UI → Queue → Check message `dest_smsc` field
- Compare with connection name in SMPP → Live Status

Solutions:

- Messages route based on `dest_smsc` field
- Verify backend is setting correct `dest_smsc`
- If `dest_smsc` is NULL, check default routing

2. Messages Scheduled for Future

Check:

- Web UI → Queue → Check `deliver_after` field
- Messages with future timestamp won't send yet

Explanation:

- Retry system sets `deliver_after` for failed messages
- Messages wait until that time before retry

Solutions:

- Wait for scheduled time
- If urgent, contact backend team to reset timestamp

3. TPS Limit Too Low

Check:

- Large queue buildup
- Messages sending very slowly

Solutions:

- Increase `tps_limit` in configuration
- Verify carrier can handle higher rate
- See CONFIGURATION.md

4. Queue Worker Not Running

Check:

- Service status
- Logs for errors

Solutions:

```
# Restart service
sudo systemctl restart omnimessage-smpp

# Check logs
sudo journalctl -u omnimessage-smpp -f
```

High Delivery Failure Rate

Symptoms:

- `smpp_delivery_failures_total` increasing
- Logs show "submit_sm_resp" with error status

- Messages not reaching recipients

Possible Causes & Solutions:

1. Invalid Destination Numbers

Check:

- Logs for specific error codes
- Review message destination format

Common Error Codes:

- `0x0000000B` - Invalid destination
- `0x00000001` - Invalid message length
- `0x00000003` - Invalid command

Solutions:

- Validate number format (E.164 recommended)
- Check number includes country code
- Verify with carrier requirements

2. Invalid Message Content

Check:

- Message length
- Special characters
- Encoding

Solutions:

- GSM-7: Max 160 characters
- UCS-2: Max 70 characters
- Remove unsupported characters
- Check encoding settings

3. Carrier Rejection

Check:

- Specific error codes from carrier
- Patterns in rejected messages

Solutions:

- Contact carrier for rejection reason
- May need content filtering
- Check for spam/abuse patterns

4. Expired Messages

Check:

- Message `expires` timestamp
- Delivery attempt timing

Solutions:

- Increase message validity period
- Reduce retry delay for time-sensitive messages

Web UI Problems

Can't Access Web Dashboard

Symptoms:

- Browser can't connect to https://your-server:8087
- Timeout or connection refused

Possible Causes & Solutions:

1. Service Not Running

Check:

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

Solutions:

```
# If stopped, start it
sudo systemctl start omnimessage-smpp

# Check logs for errors
sudo journalctl -u omnimessage-smpp -n 50
```

2. Firewall Blocking Port 8087

Check:

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

Solutions:

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

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

3. SSL Certificate Issues

Check:

- Browser shows security warning
- Certificate expired or invalid

Solutions:

- Accept security exception (if self-signed)

- Install valid SSL certificate
- Check certificate files exist:

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

4. Wrong URL

Check:

- Verify using HTTPS (not HTTP)
- Verify correct server IP/hostname
- Verify port 8087

Web UI Shows Errors

Symptoms:

- Page loads but shows errors
- Functions don't work
- Data not displaying

Solutions:

1. **Clear Browser Cache**:

   - Ctrl+F5 (hard refresh)
   - Clear browser cache and cookies

2. **Check Browser Console**:

   - Press F12
   - Check Console tab for JavaScript errors
   - Report to support if errors found

3. **Try Different Browser**:

   - Test in Chrome, Firefox, Edge
   - Isolate browser-specific issues

4. **Check Service Logs**:

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

Metrics Problems

Prometheus Metrics Not Available

Symptoms:

- `curl http://localhost:4000/metrics` fails
- Prometheus can't scrape metrics
- Empty or error response

Possible Causes & Solutions:

1. Service Not Running

Check:

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

Solutions:

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

2. Port Not Accessible

Check:

```
# Test locally
curl http://localhost:4000/metrics

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

Solutions:

- If local works but remote doesn't: Check firewall
- Open port 4000 in firewall for Prometheus server

3. Wrong Endpoint

Verify:

- Endpoint is `/metrics` (not `/prometheus` or `/stats`)
- Port is 4000 (not 8087)

Metrics Show Unexpected Values

Symptoms:

- Counters reset to zero
- Gauges show wrong values
- Missing metrics for some binds

Solutions:

1. **Service Restart Resets Counters**:

    - Counters reset on service restart
    - This is normal behavior
    - Use `increase()` or `rate()` in Prometheus queries

2. **New Binds Not Showing**:

    - Metrics only appear after first event

    - Send test message to populate metrics
    - Check bind is enabled and connected

3. **Stale Metrics**:

    - Old binds may still show in metrics
    - Restart service to clear stale entries
    - Or use Prometheus relabeling to filter

Performance Problems

High CPU Usage

Check:

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

Possible Causes:

- Very high message volume
- Too many connections
- Configuration issue

Solutions:

- Check message rate is within capacity
- Review TPS limits
- Contact support if sustained high CPU

High Memory Usage

Check:

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

Possible Causes:

- Large message queue in memory
- Memory leak (rare)

Solutions:

- Restart service to clear memory
- Check message queue size
- Contact support if memory grows continuously

Slow Message Processing

Symptoms:

- Messages take long to send
- Queue building up
- Low message rate

Check:

1. TPS limits - may be too restrictive
2. `queue_check_frequency` - may be too high
3. Backend API response time - may be slow
4. Network latency to carrier

Solutions:

- Increase TPS if carrier allows
- Decrease `queue_check_frequency` for faster polling
- Optimize backend API
- Check network latency

Configuration Problems

Configuration File Syntax Errors

Symptoms:

- Service won't start after config change
- Logs show "syntax error" or "parse error"

Check:

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

Common Mistakes:

- Missing comma between map entries
- Mismatched quotes (" vs ')
- Unmatched brackets or braces
- Missing `import Config` at top

Solutions:

- Restore from backup
- Carefully review syntax
- Use text editor with Elixir syntax highlighting

Changes Not Taking Effect

Symptoms:

- Modified configuration but no change in behavior
- Old settings still active

Solutions:

```
# Configuration changes require restart
sudo systemctl restart omnimessage-smpp

# Verify restart succeeded
sudo systemctl status omnimessage-smpp

# Check logs for errors
sudo journalctl -u omnimessage-smpp -n 50
```

Emergency Recovery

Complete System Failure

Steps:

1. **Check basic system health**:

   ```
   # Disk space
   df -h

   # Memory
   free -h

   # CPU load
   uptime
   ```

2. **Check service status**:

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

3. **Review recent logs**:

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

4. **Try service restart**:

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

5. **If restart fails**:

   - Check configuration syntax
   - Verify SSL certificates exist
   - Check file permissions
   - Review logs for specific error

6. **Restore from backup** (if needed):

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

   # Restart
   sudo systemctl restart omnimessage-smpp
   ```

7. **Contact support** if unresolved

Getting Help

Information to Gather

Before contacting support, collect:

1. **Version**: `cat /opt/omnimessage-smpp/VERSION`
2. **Recent Logs**:

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

3. **Configuration** (sanitize passwords):

```
sudo cp /opt/omnimessage-smpp/config/runtime.exs
/tmp/config.exs
# Edit /tmp/config.exs to remove passwords before sending
```

4. **Metrics Output**:

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

5. **System Info**:

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

Contact Support

- **Email**: support@omnitouch.com
- **Phone**: +61 XXXX XXXX (24/7)
- **Include**: All information from above

Related Documentation

- **OPERATIONS.md** - Normal operational procedures
- **CONFIGURATION.md** - Configuration reference
- **MONITORING.md** - Monitoring and metrics
- **README.md** - System overview
