



Omnitouch IMS Application Server

IMS Application server providing comprehensive telephony application services including call routing, supplementary services, online charging, and subscriber management.

Quick Links

Operations & Monitoring

- [Operations Guide](#) - Control Panel features, subscriber monitoring, CDR viewing, active calls, testing tools
- [Metrics Reference](#) - Complete Prometheus metrics documentation with query examples and dashboards
- [IMS Conference Server](#) - Multi-party conferencing with 3GPP compliance
- [HLR & Call Simulator](#) - Testing tools for HLR integration and call simulation

Configuration

- [Configuration Guide](#) - Complete configuration reference for Event Socket, Control Panel, API, SIP Trunks, and Diameter
- [Sh Interface](#) - Subscriber data retrieval from HSS/Repository
- [Online Charging \(Ro\)](#) - OCS integration and credit control
- [SS7 MAP](#) - HLR queries for roaming and call forwarding

Call Routing & Services

- [Dialplan Configuration](#) - XML dialplan fundamentals, variables, and call routing logic
- [Number Translation](#) - E.164 normalization and country-specific translation
- [Supplementary Services](#) - Call forwarding, CLI blocking, and emergency calling
- [Voicemail](#) - Voicemail service with SMS notifications
- [TTS Prompts](#) - Text-to-Speech prompt configuration

Compliance

- [ANSSI R226 Compliance](#) - French market regulatory compliance documentation

Call Processing Overview

Features Overview

Core Functionality

- **IMS-Compliant Call Processing** - Full 3GPP IMS Application Server functionality
- **Diameter Integration** - Sh (subscriber data) and Ro (online charging) interfaces
- **SIP Trunk Management** - Multiple gateway support with monitoring and failover
- **Call Routing** - Flexible XML-based dialplan with comprehensive variable support
- **Number Translation** - E.164 normalization for multiple countries (AU, US, PF, extendable)

Supplementary Services

- **Call Forwarding** - All variants (CFA, CFB, CFNRy, CFNRc) with MMTel-Config support
- **CLI Blocking** - Prefix-based and SIP header-based caller ID withholding
- **Emergency Calling** - Dedicated routing for emergency services with OCS bypass
- **Voicemail** - Full voicemail service with deposit, retrieval, and SMS notifications
- **IMS Conferencing** - 3GPP-compliant multi-party conferencing (RFC 4579, RFC 4575)

Monitoring & Operations

- **Control Panel Web UI** - Comprehensive monitoring interface (HTTPS on port 443)
 - Real-time subscriber registrations with cell tower locations
 - Call Detail Records with advanced search and filtering
 - Active call monitoring
 - Gateway and Diameter peer status
 - System logs and metrics
 - Testing tools (Sh, OCS, HLR, Number Translation, Call Simulator)
- **REST API** - OpenAPI/Swagger-documented API for programmatic access (port 8444)
- **Prometheus Metrics** - Extensive metrics across 3 endpoints (8080, 9090, 9093)

Integration Points

- **HSS/Repository (Sh)** - Subscriber profile and MMTel-Config retrieval
- **OCS (Ro)** - Real-time credit control with periodic reauthorization
- **HLR (SS7 MAP)** - MSRN retrieval for roaming subscribers
- **SMSc** - Voicemail and missed call notifications
- **OpenAI TTS** - Dynamic prompt generation
- **SBCs/Gateways** - Multiple SIP trunk support

Quick Start

Configuration Structure

Configuration Files

All configuration is in `config/runtime.exs`:

```
# Event Socket - Call control interface
config :tas,
  fs_event_socket: %{
    host: "127.0.0.1",
    port: 8021,
    secret: "YourSecretPassword"
  }

# Allowed source IPs for call routing
config :tas,
  allowed_sbc_source_ips: ["10.5.198.200"],
  allowed_cscf_ips: ["10.8.3.34"]

# Number translation
config :tas,
  number_translate: %{
    country_code: :AU, # :AU, :US, :PF
    localAreaCode: "617"
  }

# Online charging
config :tas,
  online_charging: %{
    enabled: true,
    periodic_ccr_time_seconds: 10,
    schedule_hangup_auth: false,
    skipped_regex: ["^2222", "^911"] # Bypass OCS for these
destinations
  }
```

```
# Emergency calling
config :tas,
  emergency_call_codes: ["911", "112", "000", "sos"]

# Voicemail
config :tas,
  voicemail: %{
    timezone: "Pacific/Tahiti",
    smsc: %{
      smsc_url: "http://10.8.81.215",
      smsc_api_key: "your_api_key",
      source_msisdn: "2222"
    }
  }
}
```

See the [Configuration Guide](#) for complete parameter reference.

Monitoring & Access Architecture

Access Points

- **Control Panel:** <https://your-server-ip/> (configured in config/runtime.exs)
- **REST API:** <https://your-server-ip:8444/api/swaggerui>
- **Prometheus Metrics:**
 - <http://your-server-ip:8080/metrics> (TAS engine, Diameter, OCS, HLR)
 - <http://your-server-ip:9090/metrics> (System, gateways, telephony core)
 - <http://your-server-ip:9093/esl?module=default> (RTP/RTCP media quality)

Dialplan Templates

Dialplan XML templates are located in priv/templates/:

- mo_dialplan.xml - Mobile Originated calls
- mt_dialplan.xml - Mobile Terminated calls
- mo_emergency_dialplan.xml - Emergency calls

See [Dialplan Configuration](#) for detailed dialplan documentation.

Documentation Structure

This README provides a high-level overview. Detailed documentation is organized by topic:

By Role

Operators & Administrators:

1. Start with [Operations Guide](#) for monitoring and day-to-day tasks
2. Review [Metrics Reference](#) for setting up monitoring and alerts
3. Use Control Panel at <https://your-server-ip/> for operational tasks

System Integrators & Engineers:

1. Start with [Configuration Guide](#) for system setup
2. Review [Dialplan Configuration](#) for call routing logic
3. Configure interfaces: [Sh Interface](#), [Online Charging](#), [SS7 MAP](#)
4. Implement services: [Supplementary Services](#), [Voicemail](#)

Developers:

1. Review [REST API documentation](#) (OpenAPI/Swagger)
2. See [Metrics Reference](#) for Prometheus integration
3. Check [Configuration Guide](#) for API configuration

By Topic

Call Processing:

- [Dialplan Configuration](#) - XML dialplan, variables, routing logic
- [Number Translation](#) - E.164 normalization
- [Supplementary Services](#) - Call forwarding, CLI blocking, emergency

Subscriber Data:

- [Sh Interface](#) - HSS/Repository integration
- [Online Charging](#) - Real-time credit control (Ro/OCS)
- [SS7 MAP](#) - HLR integration for roaming

System Configuration:

- [Configuration Guide](#) - Complete configuration reference
- [Operations Guide](#) - Control Panel and operational features
- [Metrics Reference](#) - Monitoring and observability

Value-Added Services:

- [Voicemail](#) - Voicemail with SMS notifications
- [TTS Prompts](#) - Dynamic prompt generation
- [IMS Conference Server](#) - Multi-party conferencing

System Requirements

Requirements

- Elixir/OTP runtime
- PostgreSQL or SQLite for CDR storage
- Access to IMS Core (S-CSCF)
- Network connectivity to:
 - DRA/HSS (Diameter: Sh and optionally Ro)
 - SIP Gateways/SBCs
 - Optional: SS7 MAP Gateway, SMSc, OpenAI TTS API

Support & Documentation

- **Issues & Feature Requests:** Contact your integration engineer
- **Metrics & Monitoring:** See [doc/metrics.md](#)
- **API Documentation:** Access Swagger UI at <https://your-server-ip:8444/api/swaggerui>
- **Control Panel:** Access at <https://your-server-ip/>

License

Proprietary - Omnitouch Systems



Prometheus Metrics and Monitoring Guide

Overview

OmniTAS exports comprehensive operational metrics in Prometheus format for monitoring, alerting, and observability. This guide covers all available metrics, their usage, troubleshooting, and monitoring best practices.

Metrics Endpoint

All metrics are exposed at: `http://<tas-ip>:8080/metrics`

Important: Metric Time Unit Configuration

All duration metrics in this system use `duration_unit: false` in their Histogram declarations. This is **critical** because:

1. The Prometheus Elixir library automatically detects metric names ending in `_milliseconds`
2. By default, it converts native Erlang time units to milliseconds automatically
3. Our code already converts time to milliseconds using `System.convert_time_unit/3`
4. Without `duration_unit: false`, the library would convert milliseconds to nanoseconds (dividing by $\sim 1,000,000$)

Example:

```
# Correct configuration
Histogram.declare(
  name: :http_dialplan_request_duration_milliseconds,
  help: "Duration of HTTP dialplan requests in milliseconds",
  labels: [:call_type],
  buckets: [100, 250, 500, 750, 1000, 1500, 2000, 3000, 5000],
  duration_unit: false # REQUIRED to prevent double conversion
)

# Measuring time correctly
start_time = System.monotonic_time()
# ... do work ...
end_time = System.monotonic_time()
duration_ms = System.convert_time_unit(end_time - start_time, :native, :millisecond)
Histogram.observe([name: :http_dialplan_request_duration_milliseconds], duration_ms)
```

Complete Metric Reference

Diameter Metrics

diameter_response_duration_milliseconds

Type: Histogram **Labels:** application (ro, sh), command (ccr, cca, etc), result (success, error, timeout) **Buckets:** 10, 50, 100, 250, 500, 1000, 2500, 5000, 10000 ms

Description: Duration of Diameter requests in milliseconds

Usage:

```
# Average Diameter Response Time
rate(diameter_response_duration_milliseconds_sum[5m]) /
rate(diameter_response_duration_milliseconds_count[5m])

# P95 Diameter latency
histogram_quantile(0.95,
rate(diameter_response_duration_milliseconds_bucket[5m]))
```

Alert When:

- P95 > 1000ms - Slow Diameter responses

diameter_requests_total

Type: Counter **Labels:** application (ro, sh), command (ccr, udr, etc) **Description:** Total number of Diameter requests sent

Usage:

```
# Request rate
rate(diameter_requests_total[5m])
```

diameter_responses_total

Type: Counter **Labels:** application (ro, sh), command (ccr, udr, etc), result_code (2001, 3002, 5xxx, etc) **Description:** Total number of Diameter responses received

Usage:

```
# Success rate
rate(diameter_responses_total{result_code="2001"}[5m]) /
rate(diameter_responses_total[5m]) * 100
```

diameter_peer_state

Type: Gauge **Labels:** peer_host, peer_realm, application (ro, sh) **Description:** State of Diameter peers (1=up, 0=down) **Update interval:** Every 10 seconds

Usage:


```
# Check for down peers
diameter_peer_state == 0
```

Alert When:

- Any peer down for > 1 minute

Dialplan Generation Metrics

1. HTTP Request Metrics

http_dialplan_request_duration_milliseconds

Type: Histogram **Labels:** call_type (mt, mo, emergency, unknown) **Description:** **End-to-end HTTP request duration** from when the dialplan HTTP request is received to when the response is sent. This includes all processing: parameter parsing, authorization, Diameter lookups (Sh/Ro), HLR lookups (SS7 MAP), and XML generation.

Usage:

```
# Average end-to-end HTTP request time
rate(http_dialplan_request_duration_milliseconds_sum[5m]) /
rate(http_dialplan_request_duration_milliseconds_count[5m])

# P95 by call type
histogram_quantile(0.95,
  rate(http_dialplan_request_duration_milliseconds_bucket[5m])
) by (call_type)

# Compare MT vs MO performance
histogram_quantile(0.95,

rate(http_dialplan_request_duration_milliseconds_bucket{call_type="mt"}[5m])
)
vs
histogram_quantile(0.95,

rate(http_dialplan_request_duration_milliseconds_bucket{call_type="mo"}[5m])
)
```

Alert When:

- P95 > 2000ms - Slow HTTP response times
- P95 > 3000ms - Critical performance issue
- P99 > 5000ms - Severe performance degradation
- Any requests showing call_type="unknown" - Call type detection failure

Insights:

- This is the **most important metric** for understanding user-facing latency
- Typical values: P50: 100-500ms, P95: 500-2000ms, P99: 1000-3000ms
- Includes all component timings (Sh + HLR + OCS + processing)

- If this is slow, drill down into component metrics (subscriber_data, hlr_data, ocs_authorization)
- Expected range: 100ms (fast local calls) to 5000ms (slow with retries/timeouts)

Important Notes:

- Replaces the older dialplan_generation_duration_milliseconds metric which only measured XML generation
- Accurately reflects what FreeSWITCH/SBC experiences
- Use this for SLA monitoring and capacity planning

2. Subscriber Data Metrics

subscriber_data_duration_milliseconds

Type: Histogram **Labels:** result (success, error) **Description:** Time taken to retrieve subscriber data from Sh interface (HSS)

Usage:

```
# Average Sh lookup time
rate(subscriber_data_duration_milliseconds_sum[5m]) /
rate(subscriber_data_duration_milliseconds_count[5m])

# 95th percentile Sh lookup time
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m])
)
```

Alert When:

- P95 > 100ms - Slow HSS responses
- P95 > 500ms - Critical HSS performance issue

subscriber_data_lookups_total

Type: Counter **Labels:** result (success, error) **Description:** Total number of subscriber data lookups

Usage:

```
# Sh lookup rate
rate(subscriber_data_lookups_total[5m])

# Sh error rate
rate(subscriber_data_lookups_total{result="error"}[5m])

# Sh success rate percentage
(rate(subscriber_data_lookups_total{result="success"}[5m]) /
  rate(subscriber_data_lookups_total[5m])) * 100
```

Alert When:

- Error rate > 5% - HSS connectivity issues
- Error rate > 20% - Critical HSS failure

2. HLR Data Metrics

hlr_data_duration_milliseconds

Type: Histogram **Labels:** result (success, error) **Description:** Time taken to retrieve HLR data via SS7 MAP

Usage:

```
# Average HLR lookup time
rate(hlr_data_duration_milliseconds_sum[5m]) /
rate(hlr_data_duration_milliseconds_count[5m])

# 95th percentile HLR lookup time
histogram_quantile(0.95,
  rate(hlr_data_duration_milliseconds_bucket[5m])
)
```

Alert When:

- P95 > 500ms - Slow SS7 MAP responses
- P95 > 2000ms - Critical SS7 MAP issue

hlr_lookups_total

Type: Counter **Labels:** result_type (msrn, forwarding, error, unknown) **Description:** Total HLR lookups by result type

Usage:

```
# HLR lookup rate by type
rate(hlr_lookups_total[5m])

# MSRN discovery rate (roaming subscribers)
rate(hlr_lookups_total{result_type="msrn"}[5m])

# Call forwarding discovery rate
rate(hlr_lookups_total{result_type="forwarding"}[5m])

# HLR error rate
rate(hlr_lookups_total{result_type="error"}[5m])
```

Alert When:

- Error rate > 10% - SS7 MAP issues
- Sudden drop in MSRN rate - Possible roaming issue

Insights:

- High MSRN rate indicates many roaming subscribers
- High forwarding rate indicates many forwarded calls
- Compare to call volume for roaming percentage

3. OCS Authorization Metrics

ocs_authorization_duration_milliseconds

Type: Histogram **Labels:** result (success, error) **Description:** Time taken for OCS authorization

Usage:

```
# Average OCS auth time
rate(ocs_authorization_duration_milliseconds_sum[5m]) /
rate(ocs_authorization_duration_milliseconds_count[5m])

# 95th percentile OCS auth time
histogram_quantile(0.95,
  rate(ocs_authorization_duration_milliseconds_bucket[5m])
)
```

Alert When:

- P95 > 1000ms - Slow OCS responses
- P95 > 5000ms - Critical OCS performance issue

ocs_authorization_attempts_total

Type: Counter **Labels:** result (success, error), skipped (yes, no) **Description:** Total OCS authorization attempts

Usage:

```
# OCS authorization rate
rate(ocs_authorization_attempts_total{skipped="no"}[5m])

# OCS error rate
rate(ocs_authorization_attempts_total{result="error",skipped="no"}[5m])

# OCS skip rate (emergency, voicemail, etc.)
rate(ocs_authorization_attempts_total{skipped="yes"}[5m])

# OCS success rate percentage
(rate(ocs_authorization_attempts_total{result="success",skipped="no"}[5m]) /
  rate(ocs_authorization_attempts_total{skipped="no"}[5m])) * 100
```

Alert When:

- Error rate > 5% - OCS connectivity issues
- Success rate < 95% - OCS declining too many calls

Insights:

- High skip rate indicates many emergency/free calls
- Error rate spikes indicate OCS outages
- Compare success rate to business expectations

4. Call Processing Metrics

call_param_errors_total

Type: Counter **Labels:** error_type (parse_failed, missing_required_params)

Description: Call parameter parsing errors

Usage:

```
# Parameter error rate
rate(call_param_errors_total[5m])

# Errors by type
rate(call_param_errors_total[5m]) by (error_type)
```

Alert When:

- Any errors > 0 - Indicates malformed call parameter requests
- Errors > 1% of call volume - Critical issue

authorization_decisions_total

Type: Counter **Labels:** disposition (mt, mo, emergency, unauthorized), result (success, error) **Description:** Authorization decisions by call type

Usage:

```
# Authorization rate by disposition
rate(authorization_decisions_total[5m]) by (disposition)

# MT call rate
rate(authorization_decisions_total{disposition="mt"}[5m])

# MO call rate
rate(authorization_decisions_total{disposition="mo"}[5m])

# Emergency call rate
rate(authorization_decisions_total{disposition="emergency"}[5m])

# Unauthorized call rate
rate(authorization_decisions_total{disposition="unauthorized"}[5m])
```

Alert When:

- Unauthorized rate > 1% - Possible attack or misconfiguration
- Sudden spike in emergency calls - Possible emergency event

- Unexpected change in MT/MO ratio - Possible issue

Insights:

- MT/MO ratio indicates traffic patterns
- Emergency call rate indicates service usage
- Unauthorized rate indicates security posture

freeswitch_variable_set_duration_milliseconds

Type: Histogram **Labels:** batch_size (1, 5, 10, 25, 50, 100) **Description:** Time to set Dialplan Variables

Usage:

```
# Average variable set time
rate(freeswitch_variable_set_duration_milliseconds_sum[5m]) /
rate(freeswitch_variable_set_duration_milliseconds_count[5m])

# Variable set time by batch size
histogram_quantile(0.95,
  rate(freeswitch_variable_set_duration_milliseconds_bucket[5m])
) by (batch_size)
```

Alert When:

- P95 > 100ms - Slow variable set performance
- Growing trend - Possible system performance issue

5. Module Processing Metrics

dialplan_module_duration_milliseconds

Type: Histogram **Labels:** module (MT, MO, Emergency, CallParams, etc.), call_type
Description: Processing time for each dialplan module

Usage:

```
# Processing time by module
histogram_quantile(0.95,
  rate(dialplan_module_duration_milliseconds_bucket[5m])
) by (module)

# MT module processing time
histogram_quantile(0.95,
  rate(dialplan_module_duration_milliseconds_bucket{module="MT"}[5m])
)
```

Alert When:

- Any module P95 > 500ms - Performance issue
- Growing trend in any module - Potential leak or issue

Insights:

- Identify which module is slowest
- Optimize the slowest modules first
- Compare module times across call types

6. Call Volume Metrics

call_attempts_total

Type: Counter **Labels:** call_type (mt, mo, emergency, unauthorized), result (success, rejected) **Description:** Total call attempts

Usage:

```
# Call attempt rate
rate(call_attempts_total[5m])

# Success rate by call type
(rate(call_attempts_total{result="success"}[5m]) /
 rate(call_attempts_total[5m])) * 100 by (call_type)

# Rejected call rate
rate(call_attempts_total{result="rejected"}[5m])
```

Alert When:

- Rejected rate > 5% - Possible issue
- Sudden drop in call volume - Service outage
- Sudden spike in call volume - Possible attack

active_calls

Type: Gauge **Labels:** call_type (mt, mo, emergency) **Description:** Currently active calls

Usage:

```
# Current active calls
active_calls

# Active calls by type
active_calls by (call_type)

# Peak active calls (last hour)
max_over_time(active_calls[1h])
```

Alert When:

- Active calls > capacity - Overload
- Active calls = 0 for extended time - Service down

7. Simulation Metrics

call_simulations_total

Type: Counter **Labels:** call_type (mt, mo, emergency, unauthorized), source (web, api)
Description: Call simulations run

Usage:

```
# Simulation rate
rate(call_simulations_total[5m])

# Simulations by type
rate(call_simulations_total[5m]) by (call_type)
```

Insights:

- Track diagnostic tool usage
- Identify heavy users
- Correlate with troubleshooting activity

8. SS7 MAP Metrics

ss7_map_http_duration_milliseconds

Type: Histogram **Labels:** operation (sri, prn), result (success, error, timeout) **Buckets:** 10, 50, 100, 250, 500, 1000, 2500, 5000, 10000 ms **Description:** Duration of SS7 MAP HTTP requests in milliseconds

Usage:

```
# SS7 MAP Error Rate
rate(ss7_map_operations_total{result="error"}[5m]) /
rate(ss7_map_operations_total[5m]) * 100
```

Alert When:

- P95 > 500ms - Slow SS7 MAP responses
- Error rate > 50% - Critical SS7 MAP issue

ss7_map_operations_total

Type: Counter **Labels:** operation (sri, prn), result (success, error) **Description:** Total number of SS7 MAP operations

9. Online Charging Metrics

online_charging_events_total

Type: Counter **Labels:** event_type (authorize, answer, reauth, hangup), result (success, nocredit, error, timeout) **Description:** Total number of online charging events

Usage:

```
# OCS Credit Failures
rate(online_charging_events_total{result="nocredit"}[5m])
```

Alert When:

- High rate of credit failures

10. System State Metrics

tracked_registrations

Type: Gauge **Description:** Number of currently active SIP registrations (from FreeSWITCH Sofia registration database) **Update interval:** Every 10 seconds

Notes:

- Automatically decrements when registrations expire (FreeSWITCH manages expiration)

tracked_call_sessions

Type: Gauge **Description:** Number of currently tracked call sessions in ETS **Update interval:** Every 10 seconds

11. HTTP Request Metrics

http_requests_total

Type: Counter **Labels:** endpoint (dialplan, call_event, directory, voicemail, sms_ccr, metrics), status_code (200, 400, 500, etc) **Description:** Total number of HTTP requests by endpoint

Usage:

```
# HTTP Error Rate
rate(http_requests_total{status_code=~"5.."}[5m]) /
rate(http_requests_total[5m]) * 100
```

Alert When:

- HTTP 5xx error rate > 10%

12. Call Rejection Metrics

call_rejections_total

Type: Counter **Labels:** call_type (mo, mt, emergency, unknown), reason (nocredit, unauthorized, parse_failed, missing_params, hlr_error, etc) **Description:** Total number of call rejections by reason

Usage:

```
# Call Rejection Rate by Reason
sum by (reason) (rate(call_rejections_total[5m]))
```

Alert When:

- Rejection rate > 1/sec - Investigation needed

13. Event Socket Connection Metrics

event_socket_connected

Type: Gauge **Labels:** connection_type (main, log_listener) **Description:** Event Socket connection state (1=connected, 0=disconnected) **Update interval:** Real-time on connection state changes

Usage:

```
# Event Socket Connection Status
event_socket_connected
```

Alert When:

- Connection down for > 30 seconds

event_socket_reconnections_total

Type: Counter **Labels:** connection_type (main, log_listener), result (attempting, success, failed) **Description:** Total number of Event Socket reconnection attempts

Grafana Dashboard Integration

The metrics can be visualized in Grafana using the Prometheus data source.
Recommended panels:

Dashboard 1: Call Volume

- Active calls gauge
- Call attempts rate by type (MO/MT/Emergency)
- Call rejection rate

Dashboard 2: Diameter Performance

- Response time heatmap
- Request/response rates
- Peer status table
- Error rate by result code

Dashboard 3: Online Charging Health

- Credit authorization success rate
- "No credit" event rate
- OCS timeout rate

Dashboard 4: System Performance

- Dialplan generation latency (P50/P95/P99)
- SS7 MAP response times
- Overall system availability

Recommended Grafana Dashboard Layout

Row 1: Call Volume

- Call attempts rate (by type)
- Active calls gauge
- Success rate percentage

Row 2: Performance

- P95 HTTP dialplan request time (by call type) - **PRIMARY METRIC**
- P95 Sh lookup time
- P95 HLR lookup time
- P95 OCS authorization time
- P95 dialplan module processing time (by module)

Row 3: Success Rates

- Sh lookup success rate
- HLR lookup success rate
- OCS authorization success rate
- Call attempt success rate

Row 4: Module Performance

- P95 processing time by module
- Module call counts

Row 5: Errors

- Parameter errors
- Unauthorized attempts
- Sh errors
- HLR errors
- OCS errors

Critical Alerts

Priority 1 (Page immediately):

```
# Dialplan completely down
rate(call_attempts_total[5m]) == 0

# HSS completely down
rate(subscriber_data_lookups_total{result="error"}[5m]) /
rate(subscriber_data_lookups_total[5m]) > 0.9

# OCS completely down
rate(ocs_authorization_attempts_total{result="error"}[5m]) /
rate(ocs_authorization_attempts_total[5m]) > 0.9
```

Priority 2 (Alert):

```
# Slow dialplan generation
histogram_quantile(0.95,
  rate(dialplan_generation_duration_milliseconds_bucket[5m])
) > 1000

# High HSS error rate
rate(subscriber_data_lookups_total{result="error"}[5m]) /
rate(subscriber_data_lookups_total[5m]) > 0.2

# High OCS error rate
rate(ocs_authorization_attempts_total{result="error"}[5m]) /
rate(ocs_authorization_attempts_total[5m]) > 0.1
```

Priority 3 (Warning):

```
# Elevated HSS latency
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m])
) > 100

# Elevated OCS latency
histogram_quantile(0.95,
  rate(ocs_authorization_duration_milliseconds_bucket[5m])
) > 1000

# Moderate error rate
rate(call_attempts_total{result="rejected"}[5m]) /
rate(call_attempts_total[5m]) > 0.05
```

Alerting Examples

Diameter Peer Down

```
alert: DiameterPeerDown
expr: diameter_peer_state == 0
for: 1m
annotations:
  summary: "Diameter peer {{ $labels.peer_host }} is down"
```

High Diameter Latency

```
alert: HighDiameterLatency
expr: histogram_quantile(0.95,
rate(diameter_response_duration_milliseconds_bucket[5m])) > 1000
for: 5m
annotations:
  summary: "Diameter P95 latency above 1s"
```

OCS Credit Failures

```
alert: HighOCSCreditFailures
expr: rate(online_charging_events_total{result="nocredit"}[5m]) > 0.1
for: 2m
annotations:
  summary: "High rate of OCS credit failures"
```

SS7 MAP Gateway Errors

```
alert: SS7MapErrors
expr: rate(ss7_map_operations_total{result="error"}[5m]) /
rate(ss7_map_operations_total[5m]) > 0.5
for: 3m
annotations:
  summary: "SS7 MAP error rate above 50%"
```

Event Socket Disconnected

```
alert: EventSocketDown
expr: event_socket_connected == 0
for: 30s
annotations:
  summary: "Event Socket {{ $labels.connection_type }} disconnected"
```

High Call Rejection Rate

```
alert: HighCallRejectionRate
expr: rate(call_rejections_total[5m]) > 1
for: 2m
annotations:
  summary: "High call rejection rate: {{ $value }} rejections/sec"
```

HTTP Error Rate High

```
alert: HighHTTPErrorRate
expr: rate(http_requests_total{status_code=~"5.."}[5m]) /
rate(http_requests_total[5m]) > 0.1
for: 3m
annotations:
  summary: "HTTP 5xx error rate above 10%"
```

Troubleshooting with Metrics

Problem: Metrics showing unrealistic values (nanoseconds instead of milliseconds)

Symptoms:

- Histogram `_sum` values are extremely small (e.g., 0.000315 instead of 315)
- All requests showing in the lowest bucket (< 5ms) when they should be slower
- Values appear to be 1,000,000x smaller than expected

Root Cause: The Prometheus Elixir library automatically converts time units when metric names end in `_milliseconds`, `_seconds`, etc. If `duration_unit: false` is not set, the library will convert your already-converted milliseconds into nanoseconds.

Investigation:

1. Check the metric declaration in [lib/metrics.ex](#)
2. Verify `duration_unit: false` is present:

```
Histogram.declare(  
  name: :some_duration_milliseconds,  
  help: "...",  
  buckets: [...],  
  duration_unit: false # Must be present!  
)
```

3. Check the measurement code uses proper time conversion:

```
start = System.monotonic_time()  
# ... work ...  
duration_ms = System.convert_time_unit(  
  System.monotonic_time() - start,  
  :native,  
  :millisecond  
)  
Histogram.observe([name: :some_duration_milliseconds], duration_ms)
```

Resolution:

1. Add `duration_unit: false` to the histogram declaration
2. Restart the application (required for metric declarations to reload)
3. Verify metrics show realistic values after the fix

Example Fix:

```
# Before (WRONG - will show nanoseconds)  
Histogram.declare(  
  name: :http_dialplan_request_duration_milliseconds,  
  buckets: [5, 10, 25, 50, 100, 250, 500, 1000, 2500]  
)
```

```
# After (CORRECT - will show milliseconds)
Histogram.declare(
  name: :http_dialplan_request_duration_milliseconds,
  buckets: [100, 250, 500, 750, 1000, 1500, 2000, 3000, 5000],
  duration_unit: false
)
```

Problem: Call type showing as "unknown"

Symptoms:

- All metrics show call_type="unknown" instead of mt, mo, or emergency
- Cannot differentiate performance between call types

Root Cause: The call type extraction is failing or not being properly passed through the processing pipeline.

Investigation:

1. Check logs for "HTTP dialplan request" messages - they should show the correct call type
2. Verify process_call/1 returns {xml, call_type} tuple, not just xml
3. Verify fsapi_conn/1 extracts call type from the tuple: {xml, call_type} = process_call(body)

Resolution: Ensure the dialplan processing pipeline properly threads call type through all functions.

Problem: Calls are slow

Investigation:

1. Check http_dialplan_request_duration_milliseconds P95 - **START HERE**
2. If high, check component timings:
 - Check subscriber_data_duration_milliseconds for Sh delays
 - Check hlr_data_duration_milliseconds for HLR delays
 - Check ocs_authorization_duration_milliseconds for OCS delays
 - Check dialplan_module_duration_milliseconds for module-specific delays
3. Check if call_type="unknown" - indicates call type detection failure
4. Compare MT vs MO vs Emergency processing times
5. Correlate with system logs for detailed error messages

Resolution: Optimize the slowest component

Problem: Calls are failing

Investigation:

1. Check call_attempts_total{result="rejected"} rate
2. Check subscriber_data_lookups_total{result="error"} for Sh issues
3. Check hlr_lookups_total{result_type="error"} for HLR issues
4. Check ocs_authorization_attempts_total{result="error"} for OCS issues

5. Check `authorization_decisions_total{disposition="unauthorized"}` for auth issues

Resolution: Fix the failing component

Problem: High load

Investigation:

1. Check `active_calls` current value
2. Check `call_attempts_total` rate
3. Check if rate matches expected traffic
4. Compare MT vs MO ratio
5. Check for unusual patterns (spikes, steady growth)

Resolution: Scale up or investigate unusual traffic

Problem: Roaming issues

Investigation:

1. Check `hlr_lookups_total{result_type="msrn"}` rate
2. Check `hlr_data_duration_milliseconds` for delays
3. Use HLR Lookup tool for specific subscribers
4. Check if MSRN is being retrieved correctly

Resolution: Fix HLR connectivity or configuration

Performance Baselines

Typical Values (Well-Tuned System)

- **HTTP dialplan request (end-to-end):** P50: 100-500ms, P95: 500-2000ms, P99: 1000-3000ms
- **Sh lookup time:** P50: 15ms, P95: 50ms, P99: 100ms
- **HLR lookup time:** P50: 100ms, P95: 300ms, P99: 800ms
- **OCS auth time:** P50: 150ms, P95: 500ms, P99: 1500ms
- **Dialplan module processing:** P50: 1-5ms, P95: 10-25ms, P99: 50ms
- **Sh success rate:** > 99%
- **HLR success rate:** > 95% (lower is normal due to offline subscribers)
- **OCS success rate:** > 98%
- **Call success rate:** > 99%

Note: HTTP dialplan request time is the sum of all component times plus overhead. It should roughly equal: Sh lookup + HLR lookup + OCS auth + dialplan module processing + network/parsing overhead. Minimum expected time is ~100ms (when only Sh lookup is needed), maximum typical time is ~2000ms (with all lookups and retries).

Capacity Planning

Monitor these trends:

- Growth in `call_attempts_total` rate
- Growth in `active_calls` peak
- Stable or improving P95 latencies
- Stable or improving success rates

Plan for scaling when:

- Active calls approaching 80% of capacity
- P95 latencies growing despite stable load
- Success rates declining despite stable external systems

Integration with Logging

Correlate metrics with logs:

1. High error rate in metrics → Search logs for ERROR messages
2. Slow response times → Search logs for WARNING messages about timeouts
3. Specific call issues → Search logs by call ID or phone number
4. Use simulation tool to reproduce and debug

Best Practices

1. **Set up dashboards before issues occur**
2. **Define alert thresholds based on your baseline**
3. **Test alerts by using Call Simulator**
4. **Review metrics weekly to identify trends**
5. **Correlate metrics with business events** (campaigns, outages, etc.)
6. **Use metrics to justify infrastructure investments**
7. **Share dashboards with operations team**
8. **Document your alert response procedures**

Configuration

Metrics collection is automatically enabled when the application starts. The metrics endpoint is exposed on the same port as the API (default: 8080).

To configure Prometheus to scrape metrics, add this job to your `prometheus.yml`:

```
scrape_configs:
  - job_name: 'omnitas'
    static_configs:
      - targets: ['<tas-ip>:8080']
    metrics_path: '/metrics'
    scrape_interval: 10s
```

Metric Cardinality

The metrics are designed with controlled cardinality to avoid overwhelming Prometheus:

- **Peer labels:** Limited to configured peers only

- **Call types:** Fixed set (mo, mt, emergency, unauthorized)
- **Result codes:** Limited to actual Diameter/OCS result codes received
- **Operations:** Fixed set per interface (sri/prn for MAP, ccr/cca for Diameter)

Total estimated time series: ~200-500 depending on number of configured peers and active result codes.

Metric Retention

Recommended retention periods:

- **Raw metrics:** 30 days (high resolution)
- **5-minute aggregates:** 90 days
- **1-hour aggregates:** 1 year
- **Daily aggregates:** 5 years

This supports:

- Real-time troubleshooting (raw metrics)
- Weekly/monthly analysis (5-min/1-hour aggregates)
- Capacity planning (daily aggregates)
- Historical comparison (yearly aggregates)



HLR Lookup and Call Simulator - User Guide

Overview

Two new diagnostic tools have been added to help operations staff troubleshoot call routing issues without affecting live traffic.

HLR Lookup Tool

Purpose

The HLR Lookup tool queries the Home Location Register (HLR) via SS7 MAP protocol to retrieve real-time subscriber routing information.

Access

Navigate to `/hlr` or click "HLR" in the navigation menu.

What It Shows

For any phone number, the HLR Lookup displays:

1. MSRN (Mobile Station Roaming Number)

- Temporary routing number assigned when subscriber roams to 2G/3G network
- Only present if subscriber is currently roaming
- Used by the dialplan to route calls to roaming subscriber's current location

2. Call Forwarding Settings

- Real-time call forwarding configuration from HLR
- Types: Unconditional, Busy, No-Reply, Not-Reachable
- Shows forwarding destination number
- Shows if notification is enabled

3. Dialplan Variables

- Exactly which channel variables will be set
- Variables match those used in actual call processing

- Shows how HLR data overrides Sh data

Use Cases

Diagnosing Roaming Issues

Scenario: Incoming call to roaming subscriber fails or routes incorrectly

Steps:

1. Open HLR Lookup page
2. Enter the subscriber's phone number
3. Click "Lookup HLR Data"
4. Check for MSRN in results
5. If MSRN present: Subscriber is roaming, verify MSRN is valid
6. If no MSRN: Subscriber may be in LTE/VoLTE (no MSRN needed)

Verifying Call Forwarding

Scenario: Call forwarding not working as expected

Steps:

1. Open HLR Lookup page
2. Enter the subscriber's phone number
3. Click "Lookup HLR Data"
4. Look for "Call Forwarding" in results
5. Verify forwarding type (Unconditional, Busy, etc.)
6. Verify forwarding destination number
7. Note: HLR data overrides any Sh/HSS data

Testing HLR Connectivity

Scenario: Verify SS7 MAP gateway is working

Steps:

1. Open HLR Lookup page
2. Enter any known subscriber number
3. Click "Lookup HLR Data"
4. Check for "Error" in results
5. If error: Check SS7 MAP gateway connectivity
6. Common errors:
 - "SS7 MAP is disabled" - Check configuration
 - "Timeout" - Network issue to HLR
 - "No VLR Number" - Subscriber offline or doesn't exist

Information Box

The HLR Lookup page includes educational information explaining:

- What MSRN is and when it's used
- How call forwarding works in HLR
- How this integrates with call processing
- SS7 MAP protocol basics

Call Simulator Tool

Purpose

The Call Simulator allows you to simulate complete call routing without actually placing a call or affecting live traffic.

Access

Navigate to /simulator or click "Simulator" in the navigation menu.

Features

Input Parameters

1. Source Number (Caller)

- Phone number of calling party
- For MT calls: Can be any number
- For MO calls: Must be provisioned subscriber

2. Destination Number (Called Party)

- Phone number of called party
- For MT calls: Must be provisioned subscriber
- For MO calls: Can be any number
- For Emergency: Use "urn:service:sos" or similar

3. Source IP Address

- IP address of SIP signaling source
- Must be in allowed_sbc_source_ips (for MT) or allowed_cscf_ips (for MO)
- Determines call disposition (MT vs MO)

4. Force Disposition

- Auto: Determine from IP address (normal behavior)

- MT: Force Mobile Terminating (incoming)
- MO: Force Mobile Originating (outgoing)
- Emergency: Force emergency call processing

5. Options

- Skip OCS Authorization: Bypass online charging (faster simulation)
- Skip HLR Lookup: Bypass SS7 MAP query (faster simulation)

Output

The simulator shows comprehensive results:

1. Call Type Banner

- MT, MO, or Emergency
- Color-coded for quick identification
- Shows source and destination numbers

2. Processing Steps (Left Column)

- **Subscriber Data:** Results from Sh interface (HSS)
- **HLR Data:** Results from SS7 MAP lookup (MT only)
- **OCS Authorization:** Results from online charging (MO only)
- **On-Net Status:** Whether destination is on your network (MO only)

3. Dialplan Variables (Right Column)

- Every variable that would be set on the channel
- Sorted alphabetically for easy reading
- Color-coded values (green for normal, red for errors)

4. Processing Notes

- Step-by-step explanation of what happened
- Describes data flow and decision points
- Helps understand why certain variables were set

Use Cases

Pre-Flight Testing

Scenario: Testing configuration change before deploying to production

Steps:

1. Make configuration change in dev/test environment
2. Open Call Simulator

3. Test multiple scenarios:
 - MT call from your SBC
 - MO call from your CSCF
 - Emergency call
 - On-net destination
 - Off-net destination
4. Verify all variables are correct
5. Check processing notes for any issues
6. Deploy to production with confidence

Debugging MT Call Issues

Scenario: Incoming calls to subscriber failing

Steps:

1. Open Call Simulator
2. Enter destination as the problem subscriber
3. Enter source as test number
4. Set source IP to your SBC IP
5. Leave Force Disposition as "Auto"
6. Click "Simulate Call"
7. Check Subscriber Data section for Sh lookup success
8. Check HLR Data section for MSRN or forwarding
9. Check Final Variables for hangup_case
10. If hangup_case is "UNALLOCATED_NUMBER": Subscriber not provisioned
11. If variables look correct: Issue may be in dialplan template

Debugging MO Call Issues

Scenario: Outgoing calls from subscriber failing

Steps:

1. Open Call Simulator
2. Enter source as the problem subscriber
3. Enter destination as test number
4. Set source IP to your CSCF IP
5. Uncheck "Skip OCS Authorization" if testing charging
6. Click "Simulate Call"
7. Check Caller Data section for Sh lookup success
8. Check OCS Authorization section for success/failure
9. Check On-Net Status to verify correct routing
10. Check Final Variables for allocated_time or hangup_case
11. If hangup_case is "OUTGOING_CALL_BARRED": OCS denied the call

Testing Emergency Call Handling

Scenario: Verify emergency calls work correctly

Steps:

1. Open Call Simulator
2. Enter source as test subscriber
3. Enter destination as "urn:service:sos"
4. Set any source IP (emergency calls bypass IP auth)
5. Click "Simulate Call"
6. Verify Call Type shows "Emergency (SOS)"
7. Verify hangup_case is "none" (emergency calls always proceed)
8. Check that OCS and HLR were bypassed
9. Verify caller data was retrieved for location info

Training Staff

Scenario: Teaching operations staff how call routing works

Steps:

1. Open Call Simulator
2. Run various scenarios and explain each section:
 - Show MT call and explain Sh + HLR lookups
 - Show MO call and explain OCS authorization
 - Show Emergency call and explain bypass behavior
 - Show unauthorized IP and explain rejection
3. Have staff try different combinations
4. Use Processing Notes to explain each decision
5. Compare variables between different scenarios

Comparing Sh vs HLR Data

Scenario: Understanding how HLR overrides Sh data

Steps:

1. Open Call Simulator for MT call
2. Uncheck "Skip HLR Lookup"
3. Click "Simulate Call"
4. Compare Subscriber Data variables vs HLR Data variables
5. Check Final Variables to see which values won
6. Note: HLR data always takes precedence for:
 - MSRN
 - call_forward_all_destination
 - call_forward_not_reachable_destination

Tips

- Use "Skip OCS Authorization" and "Skip HLR Lookup" for faster simulations when testing other aspects
- Copy/paste phone numbers from logs into simulator for quick testing
- Use "Force Disposition" to test specific call types regardless of IP
- Check Processing Notes if you're unsure why certain variables were set
- Run simulation multiple times to verify consistency
- Compare simulation results to actual call logs

Limitations

The simulator:

- Does NOT actually place calls
- Does NOT affect the call routing system
- Does NOT consume OCS quota (even if OCS is queried)
- Does NOT generate CDRs
- Is safe to use on production systems

The simulator DOES:

- Query actual Sh interface (HSS) if not skipped
- Query actual HLR via SS7 MAP if not skipped
- Query actual OCS if not skipped
- Show exactly what would happen in real call
- Use real configuration values

Integration with Monitoring

Both tools integrate with Prometheus metrics:

- HLR lookups via the tool are counted in `hlr_lookups_total`
- Call simulations are counted in `call_simulations_total{call_type, source}`
- Processing times are tracked in respective duration metrics

This helps:

- Track diagnostic tool usage
- Monitor performance of diagnostic queries
- Identify heavy users of diagnostic tools

For complete metrics documentation: See [metrics.md](#) for all available metrics, query examples, and monitoring setup.

Best Practices

1. Use Call Simulator First

- Before making configuration changes
- When troubleshooting subscriber-specific issues
- To understand call flow for training

2. Use HLR Lookup For

- Quick check of roaming status
- Verifying call forwarding from HLR
- Testing SS7 MAP connectivity

3. Document Findings

- Take screenshots of simulator results
- Note any unexpected behavior
- Share results with team for analysis

4. Compare to Logs

- Run simulation with same parameters as failed call
- Compare simulator variables to actual call logs
- Identify discrepancies

5. Regular Testing

- Weekly spot checks with simulator
- Test each call type (MT/MO/Emergency)
- Verify OCS and HLR integration

Troubleshooting the Tools

HLR Lookup Issues

Tool shows "SS7 MAP is disabled"

- Check `config/runtime.exs` for `ss7_map.enabled`
- Restart application after config change

Tool shows timeout errors

- Check SS7 MAP gateway is reachable
- Check network connectivity to HLR
- Check `ss7_map.timeout_ms` in configuration

Tool shows "No VLR Number"

- Subscriber is offline or doesn't exist in HLR
- Normal for subscribers who are powered off
- Normal for non-existent numbers

Call Simulator Issues

Simulator shows "No Sh data"

- Subscriber not provisioned in HSS
- HSS is unreachable
- Check `diameter.sh_application` configuration

Simulator shows "Source IP is not authorized"

- IP not in `allowed_sbc_source_ips` or `allowed_cscf_ips`
- Use "Force Disposition" to override IP-based auth

Simulator shows "Missing required parameters"

- All fields are required except options
- Enter valid phone numbers
- Enter valid IP address

Simulator takes too long

- Uncheck "Skip OCS Authorization" if not testing OCS
- Uncheck "Skip HLR Lookup" if not testing HLR
- Check actual system performance (Sh/HLR/OCS response times)

Support

For issues with these tools:

1. Check application logs for errors
2. Verify configuration (Sh, HLR, OCS)
3. Test connectivity to external systems
4. Contact support team with screenshots and error messages



IMS Conference Server - User Guide

Overview

The IMS Conference Server provides multi-party conferencing capabilities compliant with the 3GPP IMS Conference Framework (RFC 4579, RFC 4575, TS 24.147). It enables subscribers to create and manage audio/video conferences through the IMS Application Server.

Architecture

The IMS Conference Server is an integrated component of OmniTAS that provides:

- **Conference Factory URI:** SIP URI for creating new conferences
- **Conference Focus:** Manages conference state and participants
- **Conference Policy Control:** Enforces participant roles and permissions
- **Media Mixing:** Handles audio/video mixing for conference participants

IMS Conference Factory Architecture

The TAS implements the 3GPP Conference Factory pattern as defined in TS 24.147 and RFC 4579:

Conference Creation Flow (RFC 4579 Factory Pattern)

This diagram shows how a user creates a new conference through the Conference Factory URI:

Participant Join Flow

This diagram shows how additional participants join an existing conference:

Conference Event Package (RFC 4575)

The conference server sends conference state notifications to all participants:

Conference Management Operations

Operations performed through Web UI or OmniTAS Console:

Access

Web Interface

Navigate to /conference or click "Conference" in the navigation menu to access the Conference Management interface.

OmniTAS Console

Access the conference server from the OmniTAS console using the `ims_conference` command.

Features

Conference Management Interface

The web interface provides real-time monitoring and management of active IMS conferences:

Statistics Dashboard

Displays high-level conference server statistics:

- **Active Conferences:** Total number of ongoing conferences
- **Total Participants:** Combined participant count across all conferences
- **Video Conferences:** Number of conferences with video enabled

- **Locked Conferences:** Number of conferences locked to new participants

The dashboard also shows server configuration:

- **Domain:** Conference server domain (e.g., conference.ims.local)
- **Factory URI:** SIP URI for conference creation requests
- **MNC/MCC:** Mobile Network Code and Country Code
- **Access Network:** Network type (e.g., 3GPP-E-UTRAN-FDD)
- **Default Max Participants:** Maximum participants per conference
- **Video by Default:** Whether video is enabled by default
- **Recording Enabled:** Whether conference recording is available

Conference List

Shows all active conferences with:

- **Conference ID:** Unique identifier for the conference
- **URI:** SIP URI of the conference
- **Participants:** Current number of participants
- **Creator:** Phone number/URI of the conference creator

Click on any conference to expand and view detailed information.

Conference Details

Expanding a conference shows:

Conference Information:

- ID and URI
- Room name
- Creator identity
- Conference state
- Participant count (current/max and minimum)
- Video status (Enabled/Disabled)
- Lock status (Locked/Unlocked)
- Recording status (Active/Inactive)

Participant List:

- SIP URI of each participant
- Session UUID
- Participant state
- Role (0 = participant, 1 = moderator)
- Video status

Conference Actions:

- Lock/Unlock conference
- Enable/Disable video
- (Additional actions available via CLI)

Auto-Refresh

The interface automatically refreshes every 5 seconds to show real-time conference status. You can toggle auto-refresh on/off or manually refresh using the "Refresh" button.

OmniTAS Console Commands

All conference management operations are available through the `ims_conference` command in the OmniTAS console.

Command Syntax

```
ims_conference <command> [arguments]
```

Available Commands

list

Lists all active IMS conferences.

```
omnitas@server> ims_conference list
IMS Conferences:
Conference ID      Conference URI      Participants Creator
=====
1-1765699908      sip:conf-1-1765699908@conference.ims.local 3      19078720151

Total: 1 conferences
```

info

Shows detailed information about a specific conference.

Syntax: `ims_conference info <conf_id>`

Important: Use the Conference ID (e.g., 1-1765699908), not the conference name with prefix.

```
omnitas@server> ims_conference info 1-1765699908
Conference Information:
  ID: 1-1765699908
  URI: sip:conf-1-1765699908@conference.ims.local
  Room: ims-conf-1-1765699908
  Creator: 19078720151
  State: 1
  Participants: 3/10 (min: 2)
  Video: Enabled
  Locked: No
  Recording: Inactive

Participants:
- sip:1235;phone-
context=ims.mnc380.mcc313.3gppnetwork.org@ims.mnc380.mcc313.3gppnetwork.org;user=phone;method=INVITE
(342d50e0-9f67-4cc5-9179-4acae6f65f34)
  State: 3, Role: 0, Video: 0n
- sip:1235;phone-
context=ims.mnc380.mcc313.3gppnetwork.org@ims.mnc380.mcc313.3gppnetwork.org;user=phone;method=INVITE
(bd98ca37-64fd-4618-b2db-aaba108c73e2)
  State: 3, Role: 0, Video: 0n
- 19078720151 (6270da85-9b94-4285-8130-8769b11d0aa2)
  State: 3, Role: 1, Video: 0n
```

stats

Displays overall conference server statistics and configuration.

```
omnitas@server> ims_conference stats
IMS Conference Server Statistics:
=====
Active conferences: 1
Total participants: 3
Video conferences: 1
Locked conferences: 0

Configuration:
  Domain: conference.ims.local
  Factory URI: sip:conference-factory@conf-factory.ims.mnc380.mcc313.3gppnetwork.org
  MNC/MCC: 380/313
  Access Network: 3GPP-E-UTRAN-FDD
  Default max participants: 10
  Allow anonymous: Yes
  Video by default: Yes
  Recording enabled: Yes
  Announcements: Join=0n, Leave=0n, Count=0n
```

create

Creates a new conference.

Syntax: `ims_conference create <creator_uri>`

```
omnitas@server> ims_conference create sip:19078720151@ims.local
Conference created: 1-1765699909
Conference URI: sip:conf-1-1765699909@conference.ims.local
```

destroy

Terminates a conference and disconnects all participants.

Syntax: `ims_conference destroy <conf_id>`

```
omnitas@server> ims_conference destroy 1-1765699908
Conference 1-1765699908 destroyed
```

add

Adds a participant to an existing conference.

Syntax: `ims_conference add <conf_id> <sip_uri>`

```
omnitas@server> ims_conference add 1-1765699908 sip:19078720152@ims.local
Adding participant sip:19078720152@ims.local to conference 1-1765699908
```

remove

Removes a participant from a conference.

Syntax: `ims_conference remove <conf_id> <uuid>`

Note: Use the participant's session UUID from the info command output.

```
omnitas@server> ims_conference remove 1-1765699908 342d50e0-9f67-4cc5-9179-4acae6f65f34
Removed participant from conference 1-1765699908
```

lock

Locks a conference to prevent new participants from joining.

Syntax: `ims_conference lock <conf_id>`

```
omnitas@server> ims_conference lock 1-1765699908
Conference 1-1765699908 locked
```

unlock

Unlocks a conference to allow new participants.

Syntax: `ims_conference unlock <conf_id>`

```
omnitas@server> ims_conference unlock 1-1765699908
Conference 1-1765699908 unlocked
```

video

Controls video for a conference.

Syntax: `ims_conference video <conf_id> on|off`

```
omnitas@server> ims_conference video 1-1765699908 on
Video enabled for conference 1-1765699908
```

```
omnitas@server> ims_conference video 1-1765699908 off
Video disabled for conference 1-1765699908
```

record

Controls conference recording.

Syntax: `ims_conference record <conf_id> start|stop`

```
omnitas@server> ims_conference record 1-1765699908 start
Recording started for conference 1-1765699908
```

```
omnitas@server> ims_conference record 1-1765699908 stop
Recording stopped for conference 1-1765699908
```

announce

Plays an announcement to all conference participants.

Syntax: `ims_conference announce <conf_id> <message>`

```
omnitas@server> ims_conference announce 1-1765699908 "This conference will end in 5 minutes"
Announcement sent to conference 1-1765699908
```

subscribers

Lists all subscribers currently in a conference (alternative view to info).

Syntax: `ims_conference subscribers <conf_id>`

```
omnitas@server> ims_conference subscribers 1-1765699908
Subscribers in conference 1-1765699908:
- sip:1235;phone-
context=ims.mnc380.mcc313.3gppnetwork.org@ims.mnc380.mcc313.3gppnetwork.org;user=phone;method=INVITE
- 19078720151
```

Conference States

Conferences and participants have numeric state values:

Conference States

- **0:** Initializing
- **1:** Active
- **2:** Terminating
- **3:** Terminated

Participant States

- **0:** Invited
- **1:** Dialing
- **2:** Alerting
- **3:** Connected
- **4:** Disconnecting
- **5:** Disconnected

Participant Roles

- **0:** Regular participant
- **1:** Moderator/Creator

Use Cases

Monitoring Active Conferences

Scenario: Operations team needs to see how many conferences are active

Steps:

1. Open Conference Management interface (/conference)

2. View the Statistics Dashboard for high-level metrics
3. Review the conference list for specific conferences
4. Use auto-refresh to monitor in real-time

CLI Alternative:

```
omnitas@server> ims_conference stats  
omnitas@server> ims_conference list
```

Troubleshooting Conference Issues

Scenario: User reports they cannot join a conference

Steps:

1. Get the conference ID from the user
2. Run `ims_conference info <conf_id>` to check conference state
3. Check if conference is locked (Locked: Yes)
4. Check current participant count vs. maximum
5. Review participant list for any connection issues
6. Check OmniTAS logs for SIP invite failures

Common Issues:

- Conference locked: `ims_conference unlock <conf_id>`
- Maximum participants reached: Check `default_max_participants` config
- Network issues: Check SIP connectivity and firewall rules

Managing Conference Bandwidth

Scenario: Need to reduce bandwidth usage during network congestion

Steps:

1. Identify conferences with video enabled
2. For non-critical conferences, disable video:

```
ims_conference video <conf_id> off
```

3. Monitor bandwidth usage
4. Re-enable video when congestion clears

Handling Disruptive Participants

Scenario: A participant is being disruptive in a conference

Steps:

1. Get the conference ID and participant's session UUID
2. Remove the participant:

```
ims_conference remove <conf_id> <participant_uuid>
```

3. Lock the conference to prevent them from rejoining:

```
ims_conference lock <conf_id>
```

4. Add legitimate participants manually if needed:

```
ims_conference add <conf_id> <sip_uri>
```

Recording Important Conferences

Scenario: Need to record a conference for compliance or documentation

Steps:

1. Identify the conference ID
2. Start recording:

```
ims_conference record <conf_id> start
```

3. Monitor that recording is active (Recording: Active in info output)
4. Stop recording when complete:

```
ims_conference record <conf_id> stop
```

5. Recording files are stored in the OmniTAS recordings directory

Emergency Conference Termination

Scenario: Need to immediately terminate a conference

Steps:

1. Optionally announce to participants:

```
ims_conference announce <conf_id> "This conference is being terminated"
```

2. Wait a few seconds for announcement to play
3. Destroy the conference:

```
ims_conference destroy <conf_id>
```

4. All participants will be disconnected immediately

Integration with IMS Network

Conference Creation Flow

1. Subscriber sends SIP INVITE to conference factory URI
2. IMS Application Server receives request
3. Conference Server creates new conference instance
4. Conference ID and URI are generated
5. Conference policy is initialized based on creator
6. Creator is added as first participant with moderator role
7. Conference URI is returned to creator
8. Other participants can now join via the conference URI

Participant Roles

Moderator (Role: 1)

- Can lock/unlock conference
- Can remove other participants
- Can control video settings
- Receives conference notifications

Participant (Role: 0)

- Can join/leave conference
- Can speak and listen
- Can enable/disable own video
- Subject to conference policies

3GPP Compliance

The IMS Conference Server implements key 3GPP specifications:

- **TS 24.147:** Conferencing using IP Multimedia (IM) Core Network (CN) subsystem
- **RFC 4579:** Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents
- **RFC 4575:** A Session Initiation Protocol (SIP) Event Package for Conference State
- **RFC 5239:** A Framework for Centralized Conferencing

Network Elements Integration

- **P-CSCF:** Handles initial SIP signaling from UE
- **S-CSCF:** Routes conference requests to Application Server
- **OmniTAS:** Hosts the Conference Server functionality and provides media mixing

- **HSS:** Provides subscriber authentication and authorization

Configuration

Conference server configuration is managed through OmniTAS configuration files:

Key Parameters:

- **domain:** Conference server domain
- **factory_uri:** SIP URI for conference creation
- **mnc_mcc:** Mobile network identifiers
- **access_network:** Network access type
- **default_max_participants:** Default maximum participants per conference
- **allow_anonymous:** Whether to allow anonymous participants
- **video_by_default:** Default video setting for new conferences
- **recording_enabled:** Whether recording feature is available
- **announce_join:** Play tone when participant joins
- **announce_leave:** Play tone when participant leaves
- **announce_count:** Announce participant count

Best Practices

Capacity Planning

- Monitor active conference count and participant counts
- Plan for peak usage (e.g., business hours)
- Allocate sufficient CPU/memory for media mixing
- Consider video vs. audio-only for bandwidth management

Security

- Ensure conference URIs are not easily guessable
- Use conference locking for private conferences
- Monitor for unauthorized access attempts
- Implement maximum participant limits
- Review conference recordings access controls

Operational Monitoring

- Set up alerts for conference server errors
- Monitor conference creation/destruction rates
- Track average conference duration
- Review participant connection failures
- Monitor media quality metrics

For detailed metrics documentation: See [metrics.md](#) for:

- RTP/RTCP media quality metrics (Port 9093)
- Active call and session metrics (Port 9090)
- System and Erlang VM metrics (Port 8080)
- Prometheus query examples

Troubleshooting

- Check OmniTAS logs for conference-related errors
- Verify SIP connectivity between participants and conference server
- Monitor RTP media streams for packet loss
- Verify network bandwidth availability
- Check participant device compatibility

Limitations

- Maximum participants per conference: Configurable (default: 10)
- Maximum concurrent conferences: Limited by server resources
- Video quality: Depends on network bandwidth and participant devices
- Recording format: Determined by OmniTAS configuration
- Conference ID format: Auto-generated, cannot be customized via web interface

Support

For issues or questions about the IMS Conference Server:

1. Check OmniTAS logs for error messages
2. Verify conference server configuration
3. Review network connectivity and firewall rules
4. Contact Omnitouch support with conference ID and timestamps



ANSSI R226 Interception Compliance Documentation

Document Purpose: This document provides technical specifications required for ANSSI R226 authorization under Articles R226-3 and R226-7 of the French Penal Code for the OmniTAS IMS Application Server.

Classification: Regulatory Compliance Documentation

Target Authority: Agence nationale de la sécurité des systèmes d'information (ANSSI)

Regulation: R226 - Protection of Correspondence Privacy and Lawful Interception

1. DETAILED TECHNICAL SPECIFICATIONS

1.1 Commercial Technical Datasheet

Product Name: OmniTAS IMS Application Server **Product Type:** Telecommunications Application Server (TAS) **Primary Function:** IMS (IP Multimedia Subsystem) call processing and session management **Network Protocols:** SIP, Diameter, HTTP/HTTPS, SS7/MAP **Deployment Model:** On-premises server application

Core Capabilities

Call Processing:

- Session Initiation Protocol (SIP) proxy and B2BUA functionality
- IMS Initial Filter Criteria (iFC) processing
- Session routing and call control
- Emergency call handling (E.164 PSAP routing)
- Call Detail Record (CDR) generation

Network Interfaces:

- **Northbound:** IMS S-CSCF interface (SIP over TCP/UDP)
- **Southbound:** SBC/Gateway interface (SIP trunking)
- **Diameter:** Sh (subscriber data), Ro (online charging)
- **SS7:** MAP gateway interface for HLR/MSC interworking

- **HTTP/HTTPS:** External service integration (SMS, TTS, MAP gateway)

Storage and Processing:

- Real-time session state management
- CDR storage and retrieval
- Subscriber registration database (Sofia SIP)
- Configuration database (SQLite)

1.2 Interception Capabilities

1.2.1 Signal Acquisition

SIP Signaling Capture:

- The OmniTAS processes all SIP signaling messages between IMS subscribers and external networks
- Full access to SIP headers including:
 - Calling party identification (From, P-Asserted-Identity)
 - Called party identification (To, Request-URI)
 - Contact URIs and network location
 - Call routing information
 - Session description (SDP) including media codecs and endpoints

Call Metadata Acquisition:

- Complete Call Detail Records (CDR) stored in database with:
 - Timestamp (start, answer, end times)
 - Caller and callee identifiers (MSISDN, IMSI, SIP URI)
 - Call direction (mobile originating/terminating)
 - Call result (answered, busy, failed, etc.)
 - Duration and charging information
 - Network location data (cell tower information when available)

Session Recording Interface (SIPREC):

- SIPREC protocol support for lawful interception
- Capability to replicate SIP signaling to external recording servers
- Configurable session recording policies
- **Licensing Control:** SIPREC functionality requires explicit licensing authorization
- **Access Control:** SIPREC configuration restricted to authorized administrators

1.2.2 Media Processing Capabilities

Media Plane:

- B2BUA with RTP media relay capabilities
- RTP streams pass through the server
- Access to media flows for interception purposes
- SDP parsing for media endpoint and codec information

Signaling Plane:

- SIP message parsing and analysis
- Diameter message encoding/decoding (Sh, Ro interfaces)
- HTTP/HTTPS request/response processing

1.2.3 Analysis Capabilities

Real-Time Call Monitoring:

- Web UI dashboard showing active calls with:
 - Call state (trying, ringing, active, terminated)
 - Caller/callee information
 - Call duration
 - Media codec information
 - Network endpoints

Historical Analysis:

- CDR database queryable by:
 - Time range
 - Calling/called party number
 - Call type (voice, emergency, etc.)
 - Call result/disposition
 - Duration thresholds

Subscriber Tracking:

- Active registration monitoring
- Subscriber location tracking via:
 - IMS registration contact URI
 - P-Access-Network-Info header (cell tower identification)
 - IP address and port information
- Historical registration records

Network Analytics:

- Call volume metrics (Prometheus integration)
- Gateway status and connectivity
- Diameter peer connectivity
- System performance metrics

For comprehensive metrics documentation: See [metrics.md](#) for detailed

monitoring, alerting, and observability configuration.

Location Intelligence:

- Cell tower database integration
- E.164 number to geographic location mapping (North American Numbering Plan)
- Emergency services routing (PSAP mapping)

1.3 Countermeasure Capabilities

1.3.1 Privacy Protection Mechanisms

Communication Confidentiality:

- Diameter TLS transport security
- HTTPS for web interfaces and APIs
- Database encryption at rest (configurable)

Access Control:

- Role-based access control (RBAC) for web UI
- Password hashing with SHA-512 and salt (65,532 iterations)

Audit Logging:

- Complete audit trail of administrative actions
- Configuration change logging
- Authentication event logging
- Tamper-evident log storage

1.3.2 Anti-Interception Features

Secure Communications:

- Mandatory TLS for external interfaces (configurable)
- Certificate-based authentication
- Perfect Forward Secrecy (PFS) cipher suites

Data Protection:

- Automatic CDR retention policies
- Secure data deletion capabilities
- Database access controls
- Network segmentation support (separate management/signaling/media networks)

System Hardening:

- Boot parameter protection
- Integrity verification mechanisms
- Minimal attack surface (only required services enabled)

1.4 Technical Architecture for Lawful Interception

Lawful Interception Integration Points

1. SIPREC Interface (Session Recording Protocol - RFC 7866):

2. CDR Export Interface:

- CDR export to external systems
- Standard formats (CSV, JSON)
- Secure transfer (HTTPS)

3. Direct Database Access:

- Read-only database credentials for authorized systems
- SQL query access to CDR tables
- Subscriber registration data access
- Audit log access

4. API Integration:

- RESTful API for call monitoring
- Real-time active call queries
- Historical CDR retrieval
- Subscriber registration status

Interception Triggering Mechanisms

Target-Based Interception:

- Subscriber identifier matching (MSISDN, IMSI, SIP URI)
- Configurable interception rules in application logic
- SIPREC session forking based on caller/callee identity

Event-Based Interception:

- Emergency call detection and recording
- Specific destination number monitoring
- Geographic area-based triggering (cell tower location)

Time-Based Interception:

- Scheduled recording windows
- Retention period enforcement

- Automatic expiration of interception warrants
-

2. ENCRYPTION AND CRYPTANALYSIS CAPABILITIES

2.1 Cryptographic Capabilities Overview

The OmniTAS IMS Application Server implements cryptographic mechanisms for securing communications and protecting sensitive data. This section documents all cryptographic capabilities in accordance with ANSSI requirements.

2.2 Transport Layer Encryption

2.2.1 TLS/SSL Implementation

Supported Protocols:

- TLS 1.2 (RFC 5246)
- TLS 1.3 (RFC 8446)
- SSL 2.0/3.0: DISABLED (known vulnerabilities)
- TLS 1.0/1.1: DEPRECATED (configurable, disabled by default)

Cipher Suites (Configurable Priority List):

Preferred - TLS 1.3:

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256

Supported - TLS 1.2:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

Security Features:

- Perfect Forward Secrecy (PFS) required
- Strong Diffie-Hellman groups (2048-bit minimum)
- Elliptic Curve Cryptography: NIST P-256, P-384, P-521
- Server Name Indication (SNI) support
- OCSP stapling for certificate validation

Certificate Management:

- X.509 certificate support
- RSA key sizes: 2048-bit minimum, 4096-bit recommended
- ECDSA support (P-256, P-384)
- Certificate chain validation
- CRL and OCSP revocation checking
- Self-signed certificates (development only)
- External CA integration

Applications:

- HTTPS for web UI and API access
- Diameter over TLS

2.3 Data Encryption at Rest

2.3.1 Database Encryption

SQLite Encryption:

- SQLCipher integration support
- AES-256 encryption
- Encrypted storage for sensitive data (CDR, subscriber data)

2.3.2 File System Encryption

Sensitive Data Storage:

- CDR files: AES-256 encryption (optional)
- Configuration files: Encrypted storage for credentials
- Private keys: Encrypted keystores (PKCS#12, PEM with passphrase)
- Log files: Encryption support for archived logs

Key Storage:

- File-based keystores with passphrase protection
- Secure key rotation mechanisms

2.4 Authentication and Password Cryptography

2.4.1 Password Hashing

Algorithm: SHA-512 with salt **Configuration:**

- Randomly generated salt (128-bit minimum)
- 65,532 iteration rounds (configurable)

- Salt stored alongside hash
- Resistant to rainbow table attacks

Storage Format:

```
$6$rounds=65532$<salt>$<hash>
```

Applications:

- Web UI user authentication
- API token generation
- Administrator password storage
- Database user credentials

2.4.2 SSH Key Authentication

Supported Key Types:

- RSA: 1024-4096 bits (2048-bit minimum recommended)
- DSA: 1024-4096 bits (deprecated, RSA preferred)
- ECDSA: P-256, P-384, P-521 curves
- Ed25519: 256-bit (preferred for new deployments)

Key Management:

- External key generation support
- Public key import for client authentication
- Server host key management
- Individual key revocation
- Key rotation procedures

SSH Protocol:

- SSH-2 protocol only (SSH-1 disabled)
- Strong MAC algorithms (HMAC-SHA2-256, HMAC-SHA2-512)
- Key exchange: curve25519-sha256, ecdh-sha2-nistp256, diffie-hellman-group14-sha256

2.5 Diameter Protocol Security

2.5.1 Diameter Security Mechanisms

Transport Security:

- TLS over TCP for Diameter peer connections
- Mutual certificate authentication

Application-Level Security:

- Peer authentication via Origin-Host/Origin-Realm validation
- Shared secret configuration (legacy, deprecated)
- AVP (Attribute-Value Pair) encryption for sensitive data
- End-to-end security with CMS (Cryptographic Message Syntax)

2.6 SIP Identity Mechanisms

P-Asserted-Identity:

- Trusted network assertion
- Identity validation and translation
- Privacy header support

Note: Subscriber authentication is performed by the IMS Core (P-CSCF/S-CSCF), not by the TAS.

2.7 Cryptanalysis and Security Assessment Capabilities

2.7.1 Protocol Analysis Tools

Built-in Debugging Capabilities:

- SIP message tracing with full header/body capture
- Diameter message logging (AVP decoding)
- TLS handshake debugging
- Certificate chain validation logging

External Integration:

- Wireshark/tcpdump packet capture support
- SSLKEYLOGFILE export for TLS decryption (development only)
- PCAP export for offline analysis

2.7.2 Vulnerability Assessment Considerations

Known Cryptographic Weaknesses:

- Legacy MD5 support in SIP Digest (maintained for backward compatibility)
- Configurable weak cipher suites (disabled by default)
- Self-signed certificate support (development/testing only)

Security Testing:

- Regular security audits recommended
- Penetration testing support
- Cipher suite strength validation
- Certificate expiration monitoring

2.8 Key Management Infrastructure

2.8.1 Key Generation

Internal Key Generation:

- RSA key generation: OpenSSL library (FIPS 140-2 compliant algorithms)
- Random number generation: /dev/urandom (Linux kernel CSPRNG)
- Entropy sources: Hardware RNG, system entropy pool

2.8.2 Key Storage and Protection

Private Key Storage:

- File system with restricted permissions (0600)
- Encrypted PEM format with passphrase
- Secure deletion on key rotation

Key Backup:

- Encrypted backup procedures
- Split-key recovery mechanisms
- Secure key escrow (if required by regulation)

2.8.3 Key Distribution

Certificate Distribution:

- Manual import via web UI
- Automated provisioning via API
- ACME protocol support (Let's Encrypt, future enhancement)

Symmetric Key Distribution:

- Out-of-band key exchange for Diameter peers
- Diffie-Hellman key agreement in TLS
- No cleartext key transmission

2.9 Compliance and Standards

Cryptographic Standards Compliance:

- NIST SP 800-52: TLS guidelines
- NIST SP 800-131A: Cryptographic algorithm transitions
- RFC 7525: TLS recommendations
- ETSI TS 133 310: IMS network security
- 3GPP TS 33.203: IMS access security

French Cryptography Regulations:

- Cryptographic means declaration (if applicable)
- ANSSI cryptographic product certification (if required)
- No export-restricted cryptography (all standard algorithms)

2.10 Cryptanalysis Resistance

2.10.1 Design Principles

Defense Against Cryptanalysis:

- No custom/proprietary cryptographic algorithms
- Industry-standard, peer-reviewed algorithms only
- Regular security updates for cryptographic libraries
- Deprecation of weak algorithms

2.10.2 Operational Security

Key Rotation:

- TLS certificate renewal (annually recommended)
- Session key rotation (per-session for TLS)
- Password expiration policies (configurable)

Monitoring and Detection:

- Failed authentication attempt logging
 - Certificate expiration alerts
 - Cipher suite negotiation logging
 - Anomaly detection for encryption failures
-

3. INTERCEPTION CONTROL AND AUTHORIZATION

3.1 Access Control for Lawful Interception

Administrative Authorization:

- Lawful interception features require administrator-level privileges
- SIPREC configuration access: Super-admin role only
- CDR access: Configurable role-based permissions
- Audit logging of all interception-related actions

Legal Framework Integration:

- Interception warrant tracking (external system integration)
- Target identifier authorization lists
- Time-limited interception activation
- Automatic deactivation on warrant expiration

3.2 Data Retention and Privacy

Retention Policies:

- CDR retention: Configurable (default 90 days, regulatory requirement 1 year)
- Registration logs: Configurable retention
- Audit logs: Minimum 1 year retention
- Automatic purging of expired data

Privacy Protections:

- Minimal data collection principle
- Purpose limitation (telecommunications service provision)
- Access logging and monitoring

3.3 Handover Interfaces for Law Enforcement

Standard Lawful Interception Interfaces:

- ETSI LI (Lawful Interception) interface support (via external mediation device)
- SIPREC to LI gateway integration
- X1, X2, X3 interface support (external system)

Delivery Formats:

- IRI (Intercept Related Information): CDR metadata
- CC (Content of Communication): SIP signaling + media (via MRF)
- Structured reporting: XML, JSON formats

4. SYSTEM SECURITY AND INTEGRITY

4.1 Boot Security

Secure Boot Mechanisms:

- Bootparameter protection (ANSSI R226 requirement)
- Configuration integrity verification
- Tamper detection on startup
- Secure configuration loading

4.2 Network Security

Network Security:

- Minimal exposed ports (SIP, Diameter, HTTPS only)
- Port-based access control
- IP whitelisting/blacklisting

4.3 Intrusion Detection

Monitoring Capabilities:

- Failed authentication monitoring
 - Unusual call pattern detection
 - Anomalous Diameter traffic detection
 - Security event alerting (SIEM integration)
-

5. DOCUMENTATION REFERENCES

5.1 Technical Manuals

Available documentation in the project repository:

- **README.md:** System overview, architecture, and operational features
- **doc/deployment_guide.md:** Deployment instructions (if available)
- **doc/configuration.md:** Configuration reference (if available)

5.2 Security Certifications

- **Penetration Test Reports:** [To be provided upon request]
- **Security Audit Reports:** [To be provided upon request]
- **Cryptographic Module Validation:** OpenSSL FIPS 140-2 compliance

5.3 Compliance Documentation

- **ANSSI R226 Authorization Request:** This document
 - **Lawful Interception Compliance:** As required by French telecommunications regulations
-

6. CONTACT INFORMATION

Vendor/Operator Information:

- Company Name: Omnitouch Network Services Pty Ltd

- Address: PO BOX 296, QUINNS ROCKS WA 6030, AUSTRALIA
- Contact Person: Compliance Team
- Email: compliance@omnitouch.com.au

Technical Security Contact:

- Name: Compliance Team
- Email: compliance@omnitouch.com.au

Legal/Compliance Contact:

- Name: Compliance Team
 - Email: compliance@omnitouch.com.au
-

APPENDICES

Appendix A: SIP Message Flow Examples

A.1 Mobile Originating Call Flow with Interception Points

Legend: [INTERCEPTION] = Points where lawful interception data is captured

A.2 Emergency Call with Location Tracking

A.3 SIPREC Recording Session Establishment

Appendix B: CDR Schema

The OmniTAS system stores Call Detail Records in a SQLite database (FreeSWITCH CDR format) located at `/etc/freeswitch/db/cdr.db`.

B.1 Key CDR Fields for Lawful Interception

Field Name	Type	Description	Interception Relevance
uuid	TEXT	Unique call identifier	Session correlation
caller_id_number	TEXT	Calling party number (MSISDN)	Primary identifier for target tracking
caller_id_name	TEXT	Calling party display name	Identity verification
destination_number	TEXT	Called party number	Target destination tracking
start_stamp	DATETIME	Call start timestamp	Event timeline

Field Name	Type	Description	Interception Relevance
answer_stamp	DATETIME	Call answer timestamp	Call establishment time
end_stamp	DATETIME	Call end timestamp	Session duration calculation
duration	INTEGER	Total call duration (seconds)	Session length
billsec	INTEGER	Billable seconds (answered time)	Actual conversation duration
hangup_cause	TEXT	Call termination reason	Call outcome analysis
sip_hangup_disposition	TEXT	SIP termination details	Protocol-level termination
network_addr	TEXT	Network IP address	Source location tracking
sip_from_user	TEXT	SIP From header user part	Original SIP identity
sip_to_user	TEXT	SIP To header user part	SIP destination
sip_call_id	TEXT	SIP Call-ID header	SIP session correlation

B.2 CDR Query Examples for Lawful Interception

Query calls by target number:

```
SELECT * FROM cdr
WHERE caller_id_number = '+33612345678'
      OR destination_number = '+33612345678'
ORDER BY start_stamp DESC;
```

Query calls within time window:

```
SELECT * FROM cdr
WHERE start_stamp BETWEEN '2025-11-01 00:00:00' AND '2025-11-30 23:59:59'
      AND (caller_id_number = '+33612345678' OR destination_number = '+33612345678')
ORDER BY start_stamp DESC;
```

Export to CSV for law enforcement:

```
.mode csv
.output /tmp/interception_report.csv
SELECT caller_id_number, destination_number, start_stamp, end_stamp,
duration, hangup_cause
```

```
FROM cdr
WHERE caller_id_number = '+33612345678'
ORDER BY start_stamp DESC;
.output stdout
```

B.3 CDR API Access

The TAS provides programmatic access via the `Tas.Cdr` module:

```
# Get all calls for a specific number
Tas.Cdr.get_records_by(:caller_id_number, "+33612345678")

# Get calls in date range
Tas.Cdr.get_records_by_date_range("2025-11-01 00:00:00", "2025-11-30
23:59:59")

# Search with advanced filtering
Tas.Cdr.get_filtered_records(search: "33612345678", limit: 1000)

# Get statistics
Tas.Cdr.get_statistics()
```

B.4 CDR Retention

- Default retention: Configurable (typically 90 days to 1 year)
- Automatic purging: Supported
- Manual export: Via Web UI at `/cdr` or API
- Format: SQLite database, exportable to CSV/JSON

Appendix C: SIPREC Configuration Examples

SIPREC (Session Initiation Protocol Recording) enables the OmniTAS to send both call signaling and media to external Session Recording Servers for lawful interception.

C.1 SIPREC Architecture

C.2 Triggering SIPREC Recording

Recording can be triggered based on:

Target-based:

- Caller phone number (`caller_id_number`)
- Called phone number (`destination_number`)
- SIP URI matching

Event-based:

- All emergency calls (911, 112, etc.)
- Calls to/from specific destinations
- Time-window based recording

Geographic:

- Cell tower location (via P-Access-Network-Info header)
- IP address ranges

C.3 SIPREC Session Content

The SIPREC session sends to the SRS:

Signaling Metadata:

- Complete SIP headers (From, To, P-Asserted-Identity)
- Call-ID and session identifiers
- Timestamps (start, answer, end)
- Caller/callee information

Media Streams:

- Participant 1 RTP stream (caller audio)
- Participant 2 RTP stream (callee audio)
- Codec information
- DTMF tones

C.4 Integration with Law Enforcement

The Session Recording Server provides:

- **X1 Interface:** Administrative function (warrant management)
- **X2 Interface:** Intercept Related Information (IRI) - call metadata
- **X3 Interface:** Content of Communication (CC) - actual media

The OmniTAS serves as the Session Recording Client (SRC) and delivers both IRI and CC to the SRS for handover to law enforcement via standardized interfaces.

Appendix D: Encryption Configuration Guide

D.1 Certificate Generation

Generate TLS Certificate:

```
# Generate private key  
openssl genrsa -out server.key 4096
```

```
# Generate certificate signing request
openssl req -new -key server.key -out server.csr

# Self-signed certificate (for testing)
openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt

# Production: Obtain certificate from trusted CA
```

Note: SIP signaling to/from IMS does not use TLS. SIP communication is unencrypted TCP/UDP.

D.2 HTTPS Configuration for Web UI

API/Web Server TLS (config/runtime.exs):

```
config :api_ex,
  api: %{
    enable_tls: true,
    tls_cert_path: "priv/cert/server.crt",
    tls_key_path: "priv/cert/server.key",
    tls_versions: [:"tlsv1.2", ::"tlsv1.3"],
    ciphers: [
      "ECDHE-RSA-AES256-GCM-SHA384",
      "ECDHE-RSA-AES128-GCM-SHA256",
      "TLS_AES_256_GCM_SHA384",
      "TLS_AES_128_GCM_SHA256"
    ]
  }
}
```

D.3 SIP Configuration

SIP interfaces use unencrypted TCP/UDP transport. No TLS configuration required.

FreeSWITCH SIP Profile:

```
<!-- SIP profile uses TCP/UDP only -->
<profile name="external">
  <settings>
    <param name="sip-port" value="5060"/>
    <param name="context" value="public"/>
  </settings>
</profile>
```

D.4 Diameter TLS Configuration

Diameter Peer TLS:

```
# Enable TLS for Diameter connections
config :diameter_ex,
  peers: [
    %{
      host: "dra.example.com",
      port: 3868,
      transport: :tls,
      tls_opts: [
        certfile: "priv/cert/diameter.crt",
        keyfile: "priv/cert/diameter.key",
        cacertfile: "priv/cert/ca.crt",
        verify: :verify_peer
      ]
    }
  ]
]
```

D.5 Database Encryption

SQLite Encryption with SQLCipher:

```
# config/runtime.exs
config :exqlite,
  encryption: true,
  encryption_key: System.get_env("DB_ENCRYPTION_KEY")
```

Note: Database encryption is optional. For lawful interception purposes, physical access controls and database access logging may be sufficient.

D.6 Password Security Configuration

Password hashing is automatically configured with SHA-512 and salt:

```
# Default password hashing configuration
config :pbkdf2_elixir,
  rounds: 65_532,
  salt_len: 16
```

No additional configuration required - secure by default.

Appendix E: Glossary

Regulatory and Standards Bodies

- **ANSSI:** Agence nationale de la sécurité des systèmes d'information - French National Cybersecurity Agency
- **ETSI:** European Telecommunications Standards Institute
- **3GPP:** 3rd Generation Partnership Project - Mobile telecommunications standards organization
- **IETF:** Internet Engineering Task Force - Internet standards body

IMS Network Components

- **IMS:** IP Multimedia Subsystem - All-IP network architecture for multimedia services
- **CSCF:** Call Session Control Function - SIP server in IMS core
 - **P-CSCF:** Proxy-CSCF - First contact point for UE, SIP proxy
 - **I-CSCF:** Interrogating-CSCF - Entry point to operator's network
 - **S-CSCF:** Serving-CSCF - Session control and service triggering
- **HSS:** Home Subscriber Server - Subscriber database
- **TAS:** Telephony/Telecommunications Application Server - Service logic execution

Protocols and Signaling

- **SIP:** Session Initiation Protocol (RFC 3261) - Signaling protocol for voice/video calls
- **SDP:** Session Description Protocol (RFC 4566) - Media session parameters
- **RTP:** Real-time Transport Protocol (RFC 3550) - Media stream transport
- **RTCP:** RTP Control Protocol - Quality monitoring for RTP
- **SRTP:** Secure RTP (RFC 3711) - Encrypted media streams
- **Diameter:** AAA protocol used in IMS (authentication, authorization, accounting)
 - **Sh:** Diameter interface for subscriber data access
 - **Ro:** Diameter interface for online charging
- **SIPREC:** Session Initiation Protocol Recording (RFC 7866) - Call recording protocol

Telecommunications Equipment

- **SBC:** Session Border Controller - Network edge security and media gateway
- **MRF:** Media Resource Function - Media processing (transcoding, mixing, recording)
- **UE:** User Equipment - Mobile handset or device
- **PSAP:** Public Safety Answering Point - Emergency services call center
- **DRA:** Diameter Routing Agent - Diameter message routing

Lawful Interception

- **LI:** Lawful Interception - Legal monitoring of telecommunications
- **IRI:** Intercept Related Information - Call metadata for law enforcement
- **CC:** Content of Communication - Actual voice/media content
- **SRC:** Session Recording Client - SIPREC client (OmniTAS role)
- **SRS:** Session Recording Server - SIPREC server for recording storage
- **X1 Interface:** LI administrative interface (warrant provisioning)
- **X2 Interface:** LI interface for IRI delivery
- **X3 Interface:** LI interface for CC delivery
- **R226:** Articles R226-3 and R226-7 of French Penal Code governing interception equipment

Call Processing

- **CDR:** Call Detail Record - Billing and logging record for each call
- **B2BUA:** Back-to-Back User Agent - SIP element that acts as both client and server
- **DTMF:** Dual-Tone Multi-Frequency - Touch-tone signals
- **MSISDN:** Mobile Station International Subscriber Directory Number - Phone number
- **IMSI:** International Mobile Subscriber Identity - Unique subscriber identifier
- **E.164:** International numbering plan for telephone numbers

Security and Encryption

- **TLS:** Transport Layer Security (RFC 5246, RFC 8446) - Encryption protocol
- **PFS:** Perfect Forward Secrecy - Cryptographic property ensuring session key security
- **SHA-512:** Secure Hash Algorithm with 512-bit output
- **AES:** Advanced Encryption Standard
- **RSA:** Rivest-Shamir-Adleman - Public key cryptography algorithm
- **ECDSA:** Elliptic Curve Digital Signature Algorithm
- **PKI:** Public Key Infrastructure - Certificate management system
- **CA:** Certificate Authority - Issues digital certificates
- **CRL:** Certificate Revocation List
- **OCSP:** Online Certificate Status Protocol

Network and Location

- **MAP:** Mobile Application Part - SS7 protocol for mobile networks
- **HLR:** Home Location Register - Subscriber location database (legacy)
- **SS7:** Signaling System No. 7 - Legacy telephony signaling
- **NANP:** North American Numbering Plan
- **Cell Tower/Cell ID:** Mobile network base station identifier for location tracking

Data Formats and Storage

- **SQLite:** Embedded relational database
- **SQLCipher:** SQLite extension with encryption support
- **CSV:** Comma-Separated Values - Export format
- **JSON:** JavaScript Object Notation - Data interchange format
- **XML:** eXtensible Markup Language - Structured data format

Application Components

- **API:** Application Programming Interface - Programmatic access
- **UI:** User Interface - Web-based control panel
- **RBAC:** Role-Based Access Control - Permission system
- **UUID:** Universally Unique Identifier - Session tracking

Document Version: 1.0 **Date:** 2025-11-29 **Prepared for:** ANSSI R226
Authorization Application **Document Classification:** Regulatory Compliance -
Confidential



Configuration Guide

◇ [Back to Main Documentation](#)

This document provides comprehensive configuration reference for the TAS Application Server.

Related Documentation

Core Configuration

- ◇ [Main README](#) - Overview and quick start
- ◇ [Operations Guide](#) - Monitoring and operational tasks
- ◇ [Metrics Reference](#) - Prometheus metrics and monitoring

Integration Interfaces

- ◇ [Sh Interface](#) - Subscriber data retrieval from HSS/Repository
- ◇ [Online Charging \(Ro\)](#) - OCS integration and credit control
- ◇ [SS7 MAP](#) - HLR queries for roaming and call forwarding

Call Processing

- ◇ [Dialplan Configuration](#) - XML dialplan and call routing logic
- ◇ [Number Translation](#) - E.164 normalization rules
- ⚙ [Supplementary Services](#) - Call forwarding, CLI blocking, emergency

Value-Added Services

- ◇ [Voicemail](#) - Voicemail service with SMS notifications
- ◇ [TTS Prompts](#) - Text-to-Speech prompt configuration
- ◇ [IMS Conference Server](#) - Multi-party conferencing

Testing & Compliance

- ◇ [HLR & Call Simulator](#) - Testing tools
- ◇ [ANSSI R226 Compliance](#) - French market compliance

Config

The Application Server needs:

- To connect to SIP Trunks / SBCs for calls to/from off-net
- Connect to the DRA or HSS to get the Sh
- Optionally connect to DRA or OCS for Ro online charging
- Dialplan Config
- Configuration around the dialing rules / number translation
- Voicemail config
- Prompts
- Tests
- Metrics (Prometheus)

Event Socket Configuration

The Event Socket is used for call control, monitoring active calls, and interacting with the telephony engine. This connection allows the TAS to control call routing, retrieve channel variables, and manage active sessions.

Configuration Location: `config/runtime.exs`

```
config :tas,
  fs_event_socket: %{
    host: "127.0.0.1",
    port: 8021,
    secret: "YourSecretPassword"
  }
```

Configuration Parameters:

- **host** (string, required): Hostname or IP address of the Event Socket server
 - Default: "127.0.0.1" (localhost)
 - Use localhost if the telephony engine runs on the same server as TAS
 - Use remote IP for distributed deployments
 - Example: "10.8.82.60" for remote connection
- **port** (integer, required): TCP port for Event Socket connections
 - Default: 8021
 - Standard Event Socket port is 8021
 - Must match the Event Socket configuration in your telephony engine
 - Example: 8021
- **secret** (string, required): Authentication password for Event Socket
 - Must match the password configured in your telephony engine
 - Used for authenticating ESL connections
 - **Security Note:** Use a strong random password and keep it secure
 - Example: "cd463RZ8qMk9AHMMDGT3V"

Use Cases:

- Real-time call control and routing
- Retrieving active call information for the /calls view in Control Panel
- Executing dialplan applications programmatically
- Monitoring call state changes and events
- Managing conference calls

Connection Behavior:

- TAS establishes persistent connections to the Event Socket
- Automatically reconnects on connection failure
- Used for both inbound (receiving events) and outbound (controlling calls) modes
- Connection timeouts and retry logic are built-in

Security Considerations:

- Always use a strong, unique password for the secret parameter
- If using remote connections, ensure firewall rules allow only trusted TAS servers
- Consider using localhost-only connections when TAS and telephony engine are co-located
- Do not expose Event Socket port to public networks

Troubleshooting:

- **Connection Refused:** Verify the telephony engine is running and Event Socket is enabled
- **Authentication Failed:** Check that secret matches the telephony engine configuration
- **Timeout Errors:** Verify network connectivity and firewall rules
- **Cannot Control Calls:** Ensure TAS has connected successfully (check logs)

Control Panel Configuration

The Control Panel provides a web-based interface for monitoring and managing the TAS system. This includes viewing subscribers, CDRs, active calls, Diameter peers, gateways, and system configuration.

Configuration Location: config/runtime.exs

```
config :control_panel,
  page_order: ["/application", "/configuration"]

config :control_panel, ControlPanelWeb.Endpoint,
  url: [host: "0.0.0.0", path: "/"],
  https: [
```

```
port: 443,  
keyfile: "priv/cert/server.key",  
certfile: "priv/cert/server.crt"  
]
```

Configuration Parameters:

Page Order Configuration

- **page_order** (list of strings): Controls the display order of configuration pages in the Control Panel
 - Specifies which pages appear in navigation and their order
 - Example: ["/application", "/configuration"]
 - Default: If not set, pages appear in default alphabetical order

Web Endpoint Configuration

- **url** (map): Public URL configuration for the Control Panel
 - **host**: Hostname for generating URLs (e.g., "tas.example.com" or "0.0.0.0")
 - **path**: Base path for all Control Panel routes (default: "/")
 - Used for generating absolute URLs in redirects and links
- **https** (map): HTTPS/TLS configuration for secure access
 - **port** (integer): HTTPS port number (standard is 443)
 - **keyfile** (string): Path to TLS private key file (PEM format)
 - **certfile** (string): Path to TLS certificate file (PEM format)
 - Both files must be readable by the TAS application

Certificate Management:

The Control Panel requires valid TLS certificates for HTTPS access:

1. Self-Signed Certificates (Development/Testing):

```
openssl req -x509 -newkey rsa:4096 -keyout priv/cert/server.key  
\  
-out priv/cert/server.crt -days 365 -nodes
```

2. Production Certificates:

- Use certificates from a trusted Certificate Authority (CA)
- Common providers: Let's Encrypt (free), commercial CAs
- Ensure certificates include full chain for browser trust
- Keep private keys secure with appropriate file permissions (chmod 600)

Access Control:

The Control Panel provides access to sensitive operational data:

- **Subscriber Information:** Registration details, call history, locations
- **Call Detail Records:** Complete call records with MSISDN data
- **System Configuration:** Diameter peers, gateways, routing
- **Active Calls:** Real-time monitoring of ongoing sessions

Recommended Security Measures:

- Deploy behind firewall or VPN for production environments
- Use strong TLS certificates from trusted CAs
- Implement network-level access controls (IP whitelisting)
- Consider additional authentication layers if exposing externally
- Regularly audit access logs
- Use HTTPS only - never serve over plain HTTP

Common Deployment Patterns:

1. Internal-Only Access:

```
url: [host: "10.8.82.60", path: "/"] # Internal network only
```

2. External Access with Domain:

```
url: [host: "tas.operator.com", path: "/"]  
https: [port: 443, ...]
```

3. Behind Reverse Proxy:

```
url: [host: "tas.internal", path: "/panel"] # Nginx/Apache  
forwards to this
```

Troubleshooting:

- **Certificate Errors:** Verify paths to keyfile and certfile are correct and files are readable
- **Port Already in Use:** Check if another service is using port 443, or change to another port
- **Cannot Access UI:** Verify firewall rules allow access to the configured HTTPS port
- **SSL Handshake Failures:** Ensure certificate and key match and are in PEM format

API Configuration

The TAS includes a REST API for programmatic access to system functions,

subscriber management, and operational data. The API supports OpenAPI/Swagger documentation and is secured with TLS.

Configuration Location: config/runtime.exs

```
config :api_ex,
  api: %{
    port: 8444,
    listen_ip: "0.0.0.0",
    product_name: "OmniTAS",
    title: "API - OmniTAS",
    hostname: "localhost",
    enable_tls: true,
    tls_cert_path: "priv/cert/server.crt",
    tls_key_path: "priv/cert/server.key"
  }
```

Configuration Parameters:

- **port** (integer, required): TCP port for the API server
 - Default: 8444
 - Choose a port that doesn't conflict with other services
 - Standard HTTPS port is 443, but custom ports are common for APIs
 - Example: 8444, 8443, 9443
- **listen_ip** (string, required): IP address to bind the API server
 - "0.0.0.0": Listen on all network interfaces (external access)
 - "127.0.0.1": Listen only on localhost (internal access only)
 - Specific IP: Bind to a particular interface (e.g., "10.8.82.60")
 - **Security:** Use "127.0.0.1" if API only needed internally
- **product_name** (string): Product identifier for API metadata
 - Used in API responses and documentation
 - Example: "OmniTAS", "MyOperator-IMS"
- **title** (string): Human-readable title for API documentation
 - Displayed in OpenAPI/Swagger UI header
 - Example: "API - OmniTAS", "IMS Application Server API"
- **hostname** (string): Hostname for API server in documentation
 - Used in OpenAPI spec for generating example URLs
 - Should match how clients access the API
 - Examples: "localhost", "api.operator.com", "10.8.82.60"

- **enable_tls** (boolean): Enable or disable TLS/HTTPS for API
 - true: Serve API over HTTPS (recommended for production)
 - false: Serve API over HTTP (only for testing/development)
 - **Security:** Always use true in production environments
- **tls_cert_path** (string): Path to TLS certificate file (PEM format)
 - Required when enable_tls: true
 - Must be readable by the TAS application
 - Example: "priv/cert/server.crt"
- **tls_key_path** (string): Path to TLS private key file (PEM format)
 - Required when enable_tls: true
 - Must be readable by the TAS application
 - **Security:** Protect with file permissions (chmod 600)
 - Example: "priv/cert/server.key"

API Features:

The REST API provides programmatic access to:

- Subscriber management and provisioning
- Call Detail Records (CDR) queries
- System status and health checks
- Diameter peer status
- Gateway status and statistics
- Active call monitoring
- Configuration management

OpenAPI/Swagger Documentation:

The API includes built-in OpenAPI (Swagger) documentation:

- Access Swagger UI at: <https://hostname:port/api/swaggerui>
- OpenAPI JSON spec at: <https://hostname:port/api/openapi>
- Interactive API testing directly from the browser
- Complete endpoint documentation with request/response schemas

Security Considerations:

- **Authentication:** Implement API authentication based on your security requirements
- **Network Access:** Use firewall rules to restrict API access to authorized clients
- **TLS Required:** Always enable TLS in production (enable_tls: true)
- **Certificate Validation:** Use trusted certificates for production APIs
- **Rate Limiting:** Consider implementing rate limiting for public-facing APIs

- **Access Logs:** Monitor API access logs for suspicious activity

Example Usage:

```
# Query API with curl (replace with actual endpoint)
curl -k https://localhost:8444/api/health

# Access Swagger documentation
https://localhost:8444/api/swaggerui
```

Common Deployment Scenarios:

1. Internal API Only:

```
listen_ip: "127.0.0.1" # Only accessible from localhost
enable_tls: false      # HTTP for internal testing
```

2. Production API with TLS:

```
listen_ip: "0.0.0.0"   # Accessible from network
enable_tls: true        # HTTPS required
hostname: "api.operator.com"
```

3. Development/Testing:

```
listen_ip: "0.0.0.0"
enable_tls: false      # HTTP for easier testing
port: 8080              # Non-privileged port
```

Troubleshooting:

- **Port Binding Failed:** Verify port is not in use by another service, or run as root for ports < 1024
- **TLS Errors:** Check that certificate and key paths are correct and files are readable
- **Cannot Connect:** Verify firewall allows access to the configured port
- **Certificate Mismatch:** Ensure hostname matches the certificate Common Name (CN) or SAN
- **API Returns 404:** Check that the API application started successfully in logs

SIP Trunk Config

Ansible is responsible for creating the XML config for each outgoing gateway, visible in the Gateways tab, which are used for outgoing calls.

CSCF addresses and Gateway addresses have to be included in the that are visible in the runtime config, so we know what IPs to allow calls from, we do this

in the `allowed_sbc_source_ips` for Gateways / SBCs (sources that will send MT traffic towards the network) and `allowed_cscf_ips` for CSCFs (sources that MO traffic will originate from).

Note - If you will route calls from your TAS to itself (ie a MO call to an on-net subscriber routes back into the MT dialplan) then your TAS IP must also be in the allowed source IPs list.

```
config :tas,  
  allowed_sbc_source_ips: ["10.5.198.200", "103.26.174.36"],  
  allowed_cscf_ips: ["10.8.3.34"],
```

From the Web UI we can see the state of each gateway, and:

- SIP Registration status (if register is enabled)
- SIP Realm
- SIP Proxy Address (if used)
- Username
- Ping Time (Average SIP OPTIONS response time (if SIP OPTIONS enabled))
- Uptime (Seconds since the profile was restarted or came up)
- Calls in / Calls Out / Failed Calls In / Failed Calls Out
- Last SIP OPTIONS ping time (Epoch)
- SIP OPTIONS ping frequency
- More info in the **detail** button

Gateway Configuration Reference

Gateways are configured in XML format. Each gateway represents a SIP trunk connection to an external SBC, carrier, or PSTN gateway.

Basic Gateway Example:

```
<include>  
  <gateway name="carrier_trunk">  
    <param name="proxy" value="203.0.113.50;transport=tcp"/>  
    <param name="register" value="true"/>  
    <param name="caller-id-in-from" value="true"/>  
    <param name="username" value="trunk_user"/>  
    <param name="password" value="secure_password"/>  
    <param name="register-transport" value="tcp"/>  
    <param name="retry-seconds" value="30"/>  
    <param name="ping" value="25"/>  
  </gateway>  
</include>
```

Gateway without Registration:

```
<include>
```

```
<gateway name="sbc_static">
  <param name="proxy" value="198.51.100.10"/>
  <param name="register" value="false"/>
  <param name="caller-id-in-from" value="true"/>
</gateway>
</include>
```

Gateway Parameters

Required Parameters

name (gateway attribute)

- The unique name identifier for this gateway
- Used in dialplan to reference the gateway: sofia/gateway/name/destination
- Example: <gateway name="my_trunk">

proxy

- SIP proxy/gateway IP address or hostname
- Can include port and transport protocol
- Examples:
 - value="203.0.113.50" (default port 5060, UDP)
 - value="203.0.113.50:5061" (custom port)
 - value="203.0.113.50;transport=tcp" (TCP transport)
 - value="203.0.113.50:5061;transport=tls" (TLS on port 5061)

register

- Whether to send SIP REGISTER to the gateway
- Values: true | false
- Set to true if the trunk requires registration
- Set to false for static IP-based trunks

Authentication Parameters

username

- SIP authentication username
- Used in REGISTER and for digest authentication
- Required if register="true"
- Example: value="trunk_account_123"

password

- SIP authentication password
- Used for digest authentication challenges

- Required if register="true"
- Example: value="MySecureP@ssw0rd"

realm

- SIP realm for authentication
- Optional - usually auto-detected from challenge
- Example: value="sip.carrier.com"

auth-username

- Alternative username for authentication (if different from username)
- Rarely needed - only if carrier requires different user in auth vs From header
- Example: value="auth_user_456"

Registration Parameters

register-transport

- Transport protocol for REGISTER messages
- Values: udp | tcp | tls
- Must match transport specified in proxy parameter
- Example: value="tcp"

register-proxy

- Alternative proxy address for REGISTER (if different from call routing)
- Useful when registration server differs from call routing server
- Example: value="register.carrier.com:5060"

retry-seconds

- Seconds to wait before retrying failed registration
- Default: 30
- Range: 5 to 3600
- Example: value="30"

expire-seconds

- Registration expiry time in seconds
- Default: 3600 (1 hour)
- The gateway will re-register before expiry
- Example: value="1800" (30 minutes)

caller-id-in-from

- Include caller ID in SIP From header
- Values: true | false

- `true`: From header includes actual caller number (required by most carriers)
- `false`: From header uses gateway username
- **Recommendation:** Set to `true` for most deployments
- Example: `value="true"`

Monitoring Parameters

ping

- Send SIP OPTIONS ping every N seconds
- Monitors gateway availability and measures latency
- Disabled if not specified or set to 0
- Typical values: 15 to 60 seconds
- Visible in Gateway Status UI as "Ping Time"
- Example: `value="25"`

ping-max

- Maximum time (seconds) to retry pings before marking gateway down
- Default: Calculated from ping interval
- Example: `value="3"`

Call Routing Parameters

extension

- Fixed destination number to always dial on this gateway
- Rarely used - usually destination comes from dialplan
- Example: `value="+12125551234"`

extension-in-contact

- Include extension in Contact header
- Values: `true` | `false`
- Default: `false`
- Example: `value="false"`

contact-params

- Additional parameters to append to Contact header
- Useful for carrier-specific requirements
- Example: `value="line=1;isup=true"`

Advanced Parameters

from-user

- Override username in From header
- Default: Uses calling number or gateway username
- Example: value="trunk_pilot"

from-domain

- Override domain in From header
- Default: Uses proxy domain
- Example: value="my-domain.com"

outbound-proxy

- Outbound proxy for all SIP messages
- Different from proxy - used as Route header target
- Example: value="edge-proxy.carrier.com:5060"

context

- Dialplan context for incoming calls from this gateway
- Default: public
- Allows different incoming call routing per gateway
- Example: value="from-carrier"

channels

- Maximum concurrent calls on this gateway
- Default: Unlimited
- Used for capacity management
- Example: value="100"

dtmf-type

- DTMF transmission method
- Values: rfc2833 | info | inband | auto
- Default: rfc2833 (recommended)
- rfc2833: RTP telephone events (most common)
- info: SIP INFO messages
- inband: Audio tones
- Example: value="rfc2833"

codec-prefs

- Preferred codec list for this gateway
- Comma-separated list in preference order
- Example: value="PCMU,PCMA,G729"
- Common codecs: PCMU, PCMA, G729, AMR, AMR-WB, G722, OPUS

rtp-timeout-sec

- Hangup call if no RTP received for N seconds
- Default: 0 (disabled)
- Useful for detecting dead calls
- Example: value="120"

rtp-hold-timeout-sec

- Timeout for calls on hold with no RTP
- Default: 0 (disabled)
- Example: value="1800" (30 minutes)

SIP Signaling Options

sip-port

- Local SIP port to use for this gateway
- Default: Profile's port
- Rarely needed
- Example: value="5060"

rtp-ip

- Local IP address for RTP media
- Default: Profile's RTP IP
- Example: value="10.0.0.5"

register-proxy-port

- Port for registration proxy
- Only needed if different from proxy port
- Example: value="5061"

contact-host

- Override host portion of Contact header
- Useful for NAT scenarios
- Example: value="public-ip.example.com"

distinct-to

- Use distinct To header (different from Request-URI)
- Values: true | false
- Carrier-specific requirement
- Example: value="false"

cid-type

- Caller ID type in Remote-Party-ID or P-Asserted-Identity headers
- Values: rpid | pid | none

- rpid: Remote-Party-ID header
- pid: P-Asserted-Identity header
- Example: value="pid"

extension-in-contact

- Add extension parameter to Contact URI
- Values: true | false
- Example: value="true"

Transport Security

transport (in proxy parameter)

- Transport protocol
- Values: udp | tcp | tls | ws | wss
- Specified as part of proxy value
- Example: proxy="203.0.113.50;transport=tcp"

For TLS connections, additional certificate configuration may be required in the SIP profile.

Complete Example with Common Options

```
<include>
  <gateway name="primary_carrier">
    <!-- Required: Basic connection -->
    <param name="proxy" value="sbc.carrier.com:5060;transport=tcp"/>
    <param name="register" value="true"/>

    <!-- Authentication -->
    <param name="username" value="customer_trunk_01"/>
    <param name="password" value="SecurePassword123"/>

    <!-- Registration -->
    <param name="register-transport" value="tcp"/>
    <param name="expire-seconds" value="1800"/>
    <param name="retry-seconds" value="30"/>

    <!-- Caller ID -->
    <param name="caller-id-in-from" value="true"/>

    <!-- Monitoring -->
    <param name="ping" value="30"/>

    <!-- Media -->
    <param name="codec-prefs" value="PCMU,PCMA,G729"/>
    <param name="dtmf-type" value="rfc2833"/>
  </gateway>
</include>
```

```

<!-- Call limits -->
<param name="channels" value="100"/>

<!-- RTP timeouts -->
<param name="rtp-timeout-sec" value="300"/>
</gateway>
</include>

```

Gateway Usage in Dialplan

Reference gateways in your dialplan using the `sofia/gateway/name/destination` format:

```

<!-- Route to specific gateway -->
<action application="bridge" data="sofia/gateway/
primary_carrier/+12125551234"/>

<!-- Route using variable -->
<action application="bridge" data="sofia/gateway/
primary_carrier/${tas_destination_number}"/>

<!-- Route with custom SIP headers -->
<action application="bridge" data="{sip_h_X-Custom=Value}sofia/
gateway/primary_carrier/${tas_destination_number}"/>

<!-- Failover between gateways -->
<action application="bridge" data="sofia/gateway/
primary_carrier/${tas_destination_number}|sofia/gateway/
backup_carrier/${tas_destination_number}"/>

```

Troubleshooting Gateway Issues

Gateway Won't Register:

- Verify username and password are correct
- Check proxy address is reachable
- Confirm register-transport matches carrier requirements
- Review logs for authentication failures

Calls Fail:

- Check gateway status in Web UI (/gw)
- Verify caller-id-in-from setting matches carrier requirement
- Confirm codec compatibility with codec-prefs
- Check firewall allows SIP and RTP traffic

Poor Call Quality:

- Review ping times in Gateway Status
- Check rtp-timeout-sec isn't too aggressive
- Verify codec preferences match network capabilities
- Monitor network latency and packet loss

Diameter Peer Config

Diameter peers must be defined in the runtime config.

This config is largely boilerplate.

The Ro interface does not need to be included in the Applications if Ro is not used in your deployment.

```
config :diameter_ex,
  diameter: %{
    service_name: :omnitouch_tas,
    listen_ip: "10.8.82.60",
    listen_port: 3868,
    decode_format: :map,
    host: "example-dc01-as01",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    product_name: "OmniTAS",
    request_timeout: 5000,
    peer_selection_algorithm: :random,
    allow_undefined_peers_to_connect: true,
    log_unauthorized_peer_connection_attempts: true,
    control_module: Tas.Control.Diameter,
    processor_module: DiameterEx.Processor,
    auth_application_ids: [],
    acct_application_ids: [],
    vendor_id: 10415,
    supported_vendor_ids: [10415],
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        vendor_specific_application_ids: [
          %{
            vendor_id: 10415,
            auth_application_id: 16_777_217,
            acct_application_id: nil
          }
        ]
      },
      %{
        application_name: :ro,
        application_dictionary: :diameter_gen_3gpp_ro,
```

```

    vendor_specific_application_ids: [
      %{
        vendor_id: 0,
        auth_application_id: 4,
        acct_application_id: nil
      }
    ]
  },
],
peers: [
  %{
    port: 3868,
    host: "example-dc01-dra01.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "1.2.3.4",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  },
  %{
    port: 3869,
    host: "example-dc01-dra02.epc.mnc001.mcc001.3gppnetwork.org",
    ip: "1.2.3.44",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    tls: false,
    transport: :diameter_tcp,
    initiate_connection: true
  }
]
}

```

You can check the status of Diameter peers from the **Diameter** tab on the Web UI.

You can also test retrieving Sh data from the **Sh** tab on the Web UI to try to fetch any of the data from Sh.



Dialplan Configuration & Call Routing

◇ [Back to Main Documentation](#)

Comprehensive guide to XML dialplan configuration, call routing logic, and dialplan variables.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - SIP trunk and gateway configuration
- ◇ [Operations Guide](#) - Dialplan testing and templates viewer

Call Processing Flow

- ◇ [Number Translation](#) - E.164 normalization (happens before dialplan)
- ◇ [Sh Interface](#) - Subscriber data retrieved for dialplan variables
- ◇ [SS7 MAP](#) - MSRN/HLR data in dialplan variables
- ◇ [Online Charging](#) - OCS authorization in call flow

Services Implementation

- ⚙ [Supplementary Services](#) - Implementing call forwarding, CLI blocking in dialplan
- ◇ [Voicemail](#) - Voicemail routing and deposit/retrieval in dialplan
- ◇ [TTS Prompts](#) - Using prompts in dialplan with playback

Monitoring

- ◇ [Dialplan Metrics](#) - Dialplan-specific metrics and monitoring
- ◇ [Metrics Reference](#) - General system metrics

Dialplan Config / Call Routing

The TAS uses XML dialplans with a schema compatible with standard telecom XML dialplan formats, with variables populated by the TAS. This means you can define your own dialplan as needed, with the business logic for the operator, but have all the required data such as Repository Data, SS7 routing info, IMPPI / IMPU identities, dialplan normalization, etc, etc.

Dialplans are written into priv/templates and take the form:

- mo_dialplan.xml - Mobile Originated Call Dialplan
- mo_emergency_dialplan.xml - Mobile Originated Emergency Call Dialplan
- mt_dialplan.xml - Mobile Terminated Call Dialplan

You can view the Dialplans from inside the Web UI.

Various variables are set by the TAS before the XML gets parsed, these variables are printed to the log at the start of the call with their current values and are very helpful when defining your own call logic.

FreeSWITCH XML Dialplan Fundamentals

OmniTAS uses the same XML call routing system as the FreeSWITCH project, which allows for flexible call routing to meet your needs.

This section explains the core concepts and provides practical examples.

Basic Structure

A dialplan consists of **extensions** containing **conditions** and **actions**:

```
<extension name="description-of-what-this-does">
  <condition field="${variable}" expression="regex-pattern">
    <action application="app_name" data="parameters"/>
    <anti-action application="app_name" data="parameters"/>
  </condition>
</extension>
```

Extensions are evaluated in order from top to bottom. When a condition matches, its actions execute.

Conditions and Regex Matching

Conditions test variables against regular expressions. If the regex matches, actions execute; if not, anti-actions execute.

Basic exact match:

```
<condition field="${tas_destination_number}" expression="2222">
  <action application="log" data="INFO Calling voicemail access number"/>
</condition>
```

Multiple number match:

```
<condition field="${tas_destination_number}" expression="^(2222|3444|3445)$">
  <action application="log" data="INFO Calling special service"/>
</condition>
```

Pattern matching with capture groups:

```
<condition field="${tas_destination_number}" expression="^1(8[0-9]{9})$">
  <!-- Matches 1 followed by 8 and 9 more digits -->
  <action application="log" data="INFO Matched toll-free: $1"/>
  <action application="bridge" data="sofia/gateway/trunk/${tas_destination_number}"/>
</condition>
```

Prefix matching:

```
<condition field="${tas_destination_number}" expression="^00">
  <!-- Matches any number starting with 00 (international) -->
  <action application="log" data="INFO International call detected"/>
</condition>
```

Range matching:

```
<condition field="${msisdn}" expression="^5551241[0-9]{4}$">
  <!-- Matches 55512410000 through 55512419999 -->
  <action application="log" data="INFO Subscriber in range"/>
</condition>
```

Actions vs Anti-Actions

Actions execute when a condition matches. **Anti-actions** execute when a condition does NOT match.

```
<condition field="${cli_withheld}" expression="true">
  <!-- Executes if CLI is withheld -->
  <action application="set" data="effective_caller_id_number=anonymous"/>
  <action application="set" data="origination_privacy=hide_number"/>

  <!-- Executes if CLI is NOT withheld -->
  <anti-action application="log" data="DEBUG CLI is normal"/>
  <anti-action application="set" data="effective_caller_id_number=${msisdn}"/>
</condition>
```

The continue="true" Attribute

By default, when an extension's condition matches, the dialplan stops processing further extensions. The continue="true" attribute allows processing to continue to the next extension.

Without continue (default behavior):

```

<extension name="First-Check">
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="Log" data="INFO Processing call"/>
  </condition>
</extension>

<extension name="Never-Reached">
  <!-- This NEVER executes because the previous extension matched -->
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="Log" data="INFO This won't print"/>
  </condition>
</extension>

```

With continue="true":

```

<extension name="Print-Vars" continue="true">
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="Info" data=""/>
  </condition>
</extension>

<extension name="Check-Balance" continue="true">
  <condition field="${hangup_case}" expression="OUTGOING_CALL_BARRED">
    <action application="log" data="ERROR Insufficient balance"/>
    <action application="hangup" data="${hangup_case}"/>
  </condition>
</extension>

<extension name="Route-Call">
  <!-- This extension still gets evaluated -->
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="bridge" data="sofia/gateway/trunk/${tas_destination_number}"/>
  </condition>
</extension>

```

Use continue="true" for:

- Logging/debugging extensions
- Setting variables that apply to multiple scenarios
- Validation checks that don't route the call

Common Applications

call control

answer - Answer the call (send 200 OK)

```
<action application="answer" data=""/>
```

hangup - Terminate the call with a specific cause

```

<action application="hangup" data="NORMAL_CLEARING"/>
<action application="hangup" data="USER_BUSY"/>
<action application="hangup" data="NO_ANSWER"/>

```

bridge - Connect the call to another destination

```

<!-- Bridge to external gateway -->
<action application="bridge" data="sofia/gateway/trunk/+12125551234"/>

<!-- Bridge to internal extension with codec preferences -->
<action application="bridge" data="{absolute_codec_string=AMR-WB,AMR,PCMA}sofia/internal/
sip:user@domain.com"/>

<!-- Bridge with timeout -->
<action application="bridge" data="{originate_timeout=30}sofia/gateway/
trunk/${tas_destination_number}"/>

```

Variables and Channel Data

set - Set a channel variable

```
<action application="set" data="my_variable=my_value"/>
<action application="set" data="sip_h_X-Custom-Header=CustomValue"/>
<action application="set" data="effective_caller_id_number=anonymous"/>
```

unset - Remove a channel variable

```
<action application="unset" data="sip_h_P-Asserted-Identity"/>
```

export - Set variable and export to B-leg (bridged call)

```
<action application="export" data="sip_h_X-Account-Code=ABC123"/>
```

Media and Prompts

playback - Play an audio file

```
<action application="playback" data="/sounds/en/us/callie/misc/8000/out_of_credit.wav"/>
<action application="playback" data="${base_dir}/sounds/custom_prompt.wav"/>
```

sleep - Pause for specified milliseconds

```
<action application="sleep" data="1000"/> <!-- Sleep for 1 second -->
```

echo - Echo audio back to caller (testing)

```
<action application="echo" data=""/>
```

conference - Place call into conference

```
<action application="conference" data="room-${destination_number}@wideband"/>
```

voicemail

voicemail - Access voicemail system

```
<!-- Leave voicemail for mailbox -->
<action application="voicemail" data="default default ${msisdn}"/>

<!-- Check voicemail with auth -->
<action application="voicemail" data="check auth default default ${msisdn}"/>
```

Logging and Debugging

log - Write to log file

```
<action application="log" data="INFO Processing call from ${msisdn}"/>
<action application="log" data="DEBUG Destination: ${tas_destination_number}"/>
<action application="log" data="ERROR Call failed with cause: ${hangup_cause}"/>
```

info - Dump all channel variables to log

```
<action application="info" data=""/>
```

Misc Applications

say - Text-to-speech number reading

```
<action application="say" data="en number iterated ${tas_destination_number}"/>
```

send_dtmf - Send DTMF tones

```
<action application="send_dtmf" data="1234#"/>
```


Practical Examples

Emergency Services Routing:

```
<extension name="Emergency-911">
  <condition field="${tas_destination_number}" expression="^(911|112)$">
    <action application="log" data="ALERT Emergency call from ${msisdn}"/>
    <action application="answer" data=""/>
    <action application="playback" data="/sounds/emergency_services_transfer.wav"/>
    <action application="bridge" data="sofia/gateway/emergency_gw/${tas_destination_number}"/>
  </condition>
</extension>
```

Conditional Routing Based on Balance:

```
<extension name="Check-Credit">
  <condition field="${hangup_case}" expression="OUTGOING_CALL_BARRED">
    <action application="answer" data=""/>
    <action application="playback" data="/sounds/out_of_credit.wav"/>
    <action application="hangup" data="CALL_REJECTED"/>
  </condition>
</extension>
```

On-Net vs Off-Net Routing:

```
<extension name="Route-Decision">
  <condition field="${on_net_status}" expression="true">
    <!-- On-net: route back through TAS -->
    <action application="log" data="INFO Routing to on-net subscriber"/>
    <action application="bridge" data="sofia/internal/${tas_destination_number}@10.179.3.60"/>
    <anti-action application="log" data="INFO Routing off-net"/>
    <anti-action application="bridge" data="sofia/gateway/trunk/${tas_destination_number}"/>
  </condition>
</extension>
```

Anonymous Caller ID Handling:

```
<extension name="CLI-Privacy" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <action application="set" data="effective_caller_id_name=anonymous"/>
    <action application="set" data="effective_caller_id_number=anonymous"/>
    <action application="set" data="origination_privacy=hide_number"/>
  </condition>
</extension>
```

Voicemail on No Answer:

```
<extension name="Try-Bridge-Then-VM">
  <condition field="${tas_destination_number}" expression="^(555124115\d{2})$">
    <action application="set" data="call_timeout=30"/>
    <action application="bridge" data="sofia/internal/${tas_destination_number}@domain.com"/>

    <!-- If bridge fails, go to voicemail -->
    <action application="log" data="INFO Bridge failed, routing to voicemail"/>
    <action application="answer" data=""/>
    <action application="voicemail" data="default default ${tas_destination_number}"/>
  </condition>
</extension>
```

Number Range Routing:

```
<extension name="Local-Numbers">
  <condition field="${tas_destination_number}" expression="^([2-9]\d{2})$">
    <!-- 3-digit local extensions 200-999 -->
    <action application="log" data="INFO Local extension: $1"/>
    <action application="bridge" data="sofia/internal/$1@pbx.local"/>
  </condition>
</extension>
```

```

<extension name="National-Numbers">
  <condition field="${tas_destination_number}" expression="^555\d{7}$">
    <!-- National mobile numbers -->
    <action application="log" data="INFO National mobile call"/>
    <action application="bridge" data="sofia/gateway/national_trunk/${tas_destination_number}"/>
  </condition>
</extension>

<extension name="International">
  <condition field="${tas_destination_number}" expression="^00\d+$">
    <!-- International calls starting with 00 -->
    <action application="log" data="INFO International call"/>
    <action application="bridge" data="sofia/gateway/intl_trunk/${tas_destination_number}"/>
  </condition>
</extension>

```

Further Documentation

For complete details on each application:

- **FreeSWITCH Dialplan Documentation:** <https://freeswitch.org/confluence/display/FREESWITCH/Dialplan>
- **FreeSWITCH mod_dptools:** https://freeswitch.org/confluence/display/FREESWITCH/mod_dptools
(complete application reference)
- **Regular Expression Reference:** <https://freeswitch.org/confluence/display/FREESWITCH/Regular+Expression>
- **Channel Variables:** <https://freeswitch.org/confluence/display/FREESWITCH/Channel+Variables>

The FreeSWITCH wiki contains detailed documentation for every dialplan application, including all parameters and use cases.

Dialplan Variables

Variables set by the TAS in the XML dialplan logic:

Common Variables (All Call Types)

Initial Setup:

- destination_number - translated destination number
- tas_destination_number - translated destination number
- effective_caller_id_number - translated source number

Emergency Calls

- hangup_case - "none"
- ims_private_identity - private user identity
- ims_public_identity - public user identity
- msisdn - subscriber number (stripped of +)
- imsi - IMSI from private identity
- ims_domain - domain from private identity

MT Calls (Mobile Terminated)

- ims_private_identity - private user identity
- ims_public_identity - public user identity
- msisdn - subscriber number (stripped of +)
- imsi - IMSI from private identity
- ims_domain - domain from private identity
- call_forward_all_destination - CFA destination or "none"
- call_forward_not_reachable_destination - CFNRc destination
- scscf_address - S-CSCF address or "none"
- scscf_domain - S-CSCF domain or "none"
- no_reply_timer - timeout for no reply
- hangup_case - "none" or "UNALLOCATED_NUMBER"
- msrn - MSRN from PRN (if roaming) or forwarded number from SRI (if call forwarding active)
- tas_destination_number - Routing destination override (set to MSRN or forwarded number)

MO Calls (Mobile Originated)

- `hangup_case` - "none", "OUTGOING_CALL_BARRED", or "UNALLOCATED_NUMBER"
- `ims_private_identity` - private user identity
- `ims_public_identity` - public user identity
- `msisdn` - subscriber number (stripped of +)
- `imsi` - IMSI from private identity
- `ims_domain` - domain from private identity
- `allocated_time` - time allocated by OCS (if online charging enabled)
- `cli_withheld` - "true" or "false" string
- `on_net_status` - "true" or "false" string (whether destination is on-net)
- `msrn` - MSRN for roaming subscribers (if applicable)
- `tas_destination_number` - MSRN override (if roaming)

Emergency Calling

Emergency calling is controlled through the `emergency_call_codes` configuration parameter and is automatically detected during call authorization.

Configuration

Configure the emergency call codes in your `config/runtime.exs` file:

```
config :tas,  
  emergency_call_codes: ["911", "912", "913", "sos"],  
  # ... other config
```

Configuration Details:

- `emergency_call_codes` (required): List of strings representing emergency service numbers
- These codes are checked in addition to SIP emergency URNs (e.g., `<urn:service:sos>`)
- The system performs **exact match** comparison against the destination number
- Common codes include: "911" (US), "112" (EU), "000" (AU), "999" (UK), "sos"

How Emergency Detection Works

The `Tas.Dialplan.Authorization.is_emergency_call?/2` function checks two conditions:

1. **SIP URI Emergency Service URN:** Detects `<urn:service:sos>` or any URI containing "service:sos"
2. **Destination Number Match:** Compares Caller-Destination-Number against configured `emergency_call_codes`

If **either condition** is true, the call is classified as emergency.

Code Reference: See `lib/dialplan/authorization.ex`

Processing Flow

Call Flow Details:

1. Call arrives at TAS
2. Authorization module checks destination against emergency patterns
3. If emergency detected:
 - Call type is set to `:emergency`
 - `mo_emergency_dialplan.xml` template is used
 - OCS authorization is typically bypassed
 - Call is routed to PSAP gateway
4. Metrics are recorded with `call_type: emergency` label

Dialplan Routing

Define the routing for emergency calls in `priv/templates/mo_emergency_dialplan.xml`. This template determines how calls are routed to your PSAP (Public Safety Answering Point) gateway or SIP URI based on your market requirements.

Example emergency dialplan:

```

<extension name="Emergency-SOS">
  <condition field="${destination_number}" expression="^(911|912|913|sos)$">
    <action application="log" data="ALERT Emergency call from ${msisdn}"/>
    <action application="answer" data=""/>
    <action application="bridge" data="sofia/gateway/psap_gw/${destination_number}"/>
  </condition>
</extension>

```

Best Practices

- **Always include "sos"** in your emergency codes list for SIP URN compatibility
- **Include all local emergency numbers** for your jurisdiction (e.g., 911, 112, 000, 999)
- **Test emergency routing** regularly using the Call Simulator
- **Bypass OCS** for emergency calls to ensure they always connect (configured via `skipped_regex`)
- **Configure PSAP gateway** with high availability and redundancy
- **Monitor emergency call metrics** to ensure system reliability

On-Net Mobile Originated call to an On-Net Mobile-Terminating Subscriber

You can route from your dialplan to your dialplan through something like this:

```

<action application="bridge" data="{absolute_codec_string=AMR-
WB,AMR,PCMA,PCMU,originate_retries=1,originate_timeout=60,sip_invite_call_id=${sip_call_id}}sofia/
internal/${tas_destination_number}@10.179.3.60" />

```

Where 10.179.3.60 is the TAS IP (It's routing the call back to the TAS) - Just make sure the TAS IP it's in your `allowed_sbc_source_ips` list.



Metrics Documentation

This document describes the Prometheus metrics exposed by the IMS Application Server components.

Table of Contents

- [Metrics Endpoints](#)
- [Port 9090 - System Metrics](#)
 - [Call and Session Metrics](#)
 - [System Resource Metrics](#)
 - [Memory Metrics](#)
 - [Codec Status Metrics](#)
 - [Endpoint Status Metrics](#)
 - [Module Status Metrics](#)
 - [Registration Metrics](#)
 - [Sofia Gateway Metrics](#)
 - [Exporter Health Metrics](#)
- [Port 8080 - TAS Engine Metrics](#)
 - [Application Call Metrics](#)
 - [Diameter Protocol Metrics](#)
 - [Telephony Operations Metrics](#)
 - [Online Charging System \(OCS\) Metrics](#)
 - [Dialplan & Processing Metrics](#)
 - [Event Socket Metrics](#)
 - [Erlang Mnesia Database Metrics](#)
 - [Erlang VM Memory Metrics](#)
 - [Erlang VM Statistics](#)
 - [Erlang VM System Information](#)
 - [Erlang VM Microstate Accounting \(MSACC\)](#)
 - [Erlang VM Allocators](#)
- [Port 9093 - Media & Call Quality Metrics](#)
 - [RTP Audio - Byte Counters](#)
 - [RTP Audio - Packet Counters](#)
 - [RTP Audio - Special Packet Types](#)
 - [RTP Audio - Jitter & Quality Metrics](#)
 - [RTCP Metrics](#)
- [Go Runtime Metrics](#)
- [Process Metrics](#)
- [Prometheus HTTP Metrics](#)
- [Metric Types](#)
- [Usage](#)
- [Example Queries](#)
- [Metric Time Unit Configuration](#)
- [Grafana Dashboard Integration](#)

- [Alerting Examples](#)
- [Troubleshooting with Metrics](#)
- [Performance Baselines](#)
- [Best Practices](#)

Metrics Endpoints

Port	Endpoint	Purpose	Jump to Section
9090	/metrics	System, gateway, and core telephony metrics	Port 9090 →
8080	/metrics	TAS engine, Diameter, HLR, OCS, and Erlang VM metrics	Port 8080 →
9093	/esl?module=default	RTP/RTCP media quality and call statistics	Port 9093 →

Port 9090 - System Metrics

Call and Session Metrics

Metric Name	Port	Description
freeswitch_bridged_calls	9090	Number of bridged calls currently active
freeswitch_detailed_bridged_calls	9090	Number of detailed bridged calls active
freeswitch_current_calls	9090	Number of calls currently active
freeswitch_detailed_calls	9090	Number of detailed calls active
freeswitch_current_channels	9090	Number of channels currently active
freeswitch_current_sessions	9090	Number of sessions currently active
freeswitch_current_sessions_peak	9090	Peak number of sessions since startup
freeswitch_current_sessions_peak_last_5min	9090	Peak number of sessions in the last 5 minutes
freeswitch_sessions_total	9090	Total number of sessions since startup (counter)
freeswitch_current_sps	9090	Current sessions per second
freeswitch_current_sps_peak	9090	Peak sessions per second since startup
freeswitch_current_sps_peak_last_5min	9090	Peak sessions per second in the last 5 minutes
freeswitch_max_sessions	9090	Maximum number of

Metric Name	Port	Description
freeswitch_max_sps	9090	sessions allowed Maximum sessions per second allowed

System Resource Metrics

Metric Name	Port	Description
freeswitch_current_idle_cpu	9090	Current CPU idle percentage
freeswitch_min_idle_cpu	9090	Minimum CPU idle percentage recorded
freeswitch_uptime_seconds	9090	Uptime in seconds
freeswitch_time_synced	9090	Whether system time is in sync with exporter host time (1=synced, 0=not synced)

Memory Metrics

Metric Name	Port	Description
freeswitch_memory_arena	9090	Total non-mmapped bytes (malloc arena)
freeswitch_memory_ordblks	9090	Number of free chunks
freeswitch_memory_smblocks	9090	Number of free fastbin blocks
freeswitch_memory_hblocks	9090	Number of mapped regions
freeswitch_memory_hblockhd	9090	Bytes in mapped regions
freeswitch_memory_usmblocks	9090	Maximum total allocated space
freeswitch_memory_fsmblocks	9090	Free bytes held in fastbins
freeswitch_memory_uordblks	9090	Total allocated space
freeswitch_memory_fordblks	9090	Total free space
freeswitch_memory_keepcost	9090	Topmost releasable block

Codec Status Metrics

Metric Name	Port	Description
freeswitch_codec_status	9090	Codec status with labels: ikey (module), name (codec name), type (codec). Value=1 indicates codec is available

Available Codecs Include:

- G.711 alaw/ulaw
- PROXY PASS-THROUGH
- PROXY VIDEO PASS-THROUGH
- RAW Signed Linear (16 bit)
- Speex
- VP8/VP9 Video
- AMR variants
- B64
- G.723.1, G.729, G.722, G.726 variants

- OPUS
- MP3
- ADPCM, GSM, LPC-10

Endpoint Status Metrics

Metric Name	Port	Description
freeswitch_endpoint_status	9090	Endpoint status with labels: ikey (module), name (endpoint name), type (endpoint). Value=1 indicates endpoint is available

Available Endpoints Include:

- error, group, pickup, user (mod_dptools)
- loopback, null (mod_loopback)
- rtc (mod_rtc)
- rtp, sofia (mod_sofia)
- modem (mod_spandsp)

Module Status Metrics

Metric Name	Port	Description
freeswitch_load_module	9090	Module load status (1=loaded, 0=not loaded) with label: module

Key Modules Monitored:

- mod_sofia (SIP)
- mod_conference, mod_conference_ims
- mod_opus, mod_g729, mod_amr, etc.
- mod_event_socket
- mod_dptools
- mod_python3
- mod_rtc
- And many more...

Registration Metrics

Metric Name	Port	Description
freeswitch_registrations	9090	Total number of active registrations
freeswitch_registration_details	9090	Detailed registration information with labels: expires, hostname, network_ip, network_port, network_proto, realm, reg_user, token, url

Sofia Gateway Metrics

Metric Name	Port	Description
freeswitch_sofia_gateway_status	9090	Gateway status with labels: context, name, profile, proxy, scheme, status (UP/DOWN)
freeswitch_sofia_gateway_call_in	9090	Number of inbound calls through gateway
freeswitch_sofia_gateway_call_out	9090	Number of outbound calls through gateway
freeswitch_sofia_gateway_failed_call_in	9090	Number of failed inbound calls
freeswitch_sofia_gateway_failed_call_out	9090	Number of failed outbound calls
freeswitch_sofia_gateway_ping	9090	Last ping timestamp (Unix epoch)
freeswitch_sofia_gateway_pingtime	9090	Last ping time in milliseconds
freeswitch_sofia_gateway_pingfreq	9090	Ping frequency in seconds
freeswitch_sofia_gateway_pingcount	9090	Number of pings sent
freeswitch_sofia_gateway_pingmin	9090	Minimum ping time recorded
freeswitch_sofia_gateway_pingmax	9090	Maximum ping time recorded

Exporter Health Metrics

Metric Name	Port	Description
freeswitch_up	9090	Whether the last scrape was successful (1=success, 0=failure)
freeswitch_exporter_total_scrapes	9090	Total number of scrapes performed (counter)
freeswitch_exporter_failed_scrapes	9090	Total number of failed scrapes (counter)

[↑ Back to top](#)

Port 8080 - TAS Engine Metrics

These metrics are exposed by the Telephony Application Server engine and provide insight into call processing, database operations, and Erlang VM performance.

Application Call Metrics

Metric Name	Port	Description
call_simulations_total	8080	Total number of call simulations (counter)
call_attempts_total	8080	Total number of call attempts (counter)
call_rejections_total	8080	Total number of call rejections by reason

Metric Name	Port	Description
		(counter)
call_param_errors_total	8080	Total number of call parameter parsing errors (counter)
active_calls	8080	Number of currently active calls with labels: call_type (mo/mt/emergency)
tracked_call_sessions	8080	Number of currently tracked call sessions in ETS

Diameter Protocol Metrics

Metric Name	Port	Description
diameter_peer_state	8080	State of Diameter peers (1=up, 0=down) with labels: peer_host, peer_realm, application
diameter_requests_total	8080	Total number of Diameter requests (counter)
diameter_responses_total	8080	Total number of Diameter responses (counter)
diameter_response_duration_milliseconds	8080	Duration of Diameter requests in milliseconds (histogram)

Telephony Operations Metrics

Metric Name	Port	Description
hlr_lookups_total	8080	Total number of HLR lookups (counter)
hlr_data_duration_milliseconds	8080	Duration of HLR data retrieval in milliseconds (histogram)
subscriber_data_lookups_total	8080	Total number of subscriber data lookups (counter)
subscriber_data_duration_milliseconds	8080	Duration of Sh subscriber data retrieval in milliseconds (histogram)
ss7_map_operations_total	8080	Total number of SS7 MAP operations (counter)
ss7_map_http_duration_milliseconds	8080	Duration of SS7 MAP HTTP requests in milliseconds (histogram)
tracked_registrations	8080	Number of currently tracked SIP registrations

Online Charging System (OCS) Metrics

Metric Name	Port	Description
ocs_authorization_attempts_total	8080	Total number of OCS

Metric Name	Port	Description
		authorization attempts (counter)
ocs_authorization_duration_milliseconds	8080	Duration of OCS authorization in milliseconds (histogram)
online_charging_events_total	8080	Total number of online charging events (counter)
authorization_decisions_total	8080	Total number of authorization decisions (counter)

Dialplan & Processing Metrics

Metric Name	Port	Description
http_requests_total	8080	Total number of HTTP requests with labels: endpoint, status_code (counter)
http_dialplan_request_duration_milliseconds	8080	Duration of HTTP dialplan requests in milliseconds (histogram)
dialplan_module_duration_milliseconds	8080	Duration of individual dialplan module processing (histogram)
freeswitch_variable_set_duration_milliseconds	8080	Duration of variable setting operations (histogram)

Event Socket Metrics

Metric Name	Port	Description
event_socket_connected	8080	Event Socket connection state (1=connected, 0=disconnected) with label: connection_type
event_socket_reconnections_total	8080	Total number of Event Socket reconnection attempts (counter)

Erlang Mnesia Database Metrics

Metric Name	Port	Description
erlang_mnesia_held_locks	8080	Number of held locks
erlang_mnesia_lock_queue	8080	Number of transactions waiting for a lock
erlang_mnesia_transaction_participants	8080	Number of participant transactions
erlang_mnesia_transaction_coordinators	8080	Number of coordinator transactions

Metric Name	Port	Description
erlang_mnesia_failed_transactions	8080	Number of failed (aborted) transactions (counter)
erlang_mnesia_committed_transactions	8080	Number of committed transactions (counter)
erlang_mnesia_logged_transactions	8080	Number of transactions logged (counter)
erlang_mnesia_restarted_transactions	8080	Total number of transaction restarts (counter)
erlang_mnesia_memory_usage_bytes	8080	Total bytes allocated by all mnesia tables
erlang_mnesia_tablewise_memory_usage_bytes	8080	Bytes allocated per mnesia table with label: table
erlang_mnesia_tablewise_size	8080	Number of rows per table with label: table

Erlang VM Memory Metrics

Metric Name	Port	Description
erlang_vm_memory_atom_bytes_total	8080	Memory allocated for atoms with label: usage (used/free)
erlang_vm_memory_bytes_total	8080	Total memory allocated with label: kind (system/processes)
erlang_vm_memory_dets_tables	8080	DETS tables count
erlang_vm_memory_ets_tables	8080	ETS tables count
erlang_vm_memory_processes_bytes_total	8080	Memory allocated for processes with label: usage (used/free)
erlang_vm_memory_system_bytes_total	8080	Memory for emulator (not process-related) with label: usage (atom/binary/code/ets/other)

Erlang VM Statistics

Metric Name	Port	Description
erlang_vm_statistics_bytes_output_total	8080	Total bytes output to ports (counter)
erlang_vm_statistics_bytes_received_total	8080	Total bytes received through ports (counter)
erlang_vm_statistics_context_switches	8080	Total context

Metric Name	Port	Description
		switches since startup (counter)
erlang_vm_statistics_dirty_cpu_run_queue_length	8080	Length of dirty CPU run-queue
erlang_vm_statistics_dirty_io_run_queue_length	8080	Length of dirty IO run-queue
erlang_vm_statistics_garbage_collection_number_of_gcs	8080	Number of garbage collections (counter)
erlang_vm_statistics_garbage_collection_bytes_reclaimed	8080	Bytes reclaimed by GC (counter)
erlang_vm_statistics_garbage_collection_words_reclaimed	8080	Words reclaimed by GC (counter)
erlang_vm_statistics_reductions_total	8080	Total reductions (counter)
erlang_vm_statistics_run_queues_length	8080	Length of normal run-queues
erlang_vm_statistics_runtime_milliseconds	8080	Sum of runtime for all threads (counter)
erlang_vm_statistics_wallclock_time_milliseconds	8080	Real time measured (counter)

Erlang VM System Information

Metric Name	Port	Description
erlang_vm_dirty_cpu_schedulers	8080	Number of dirty CPU scheduler threads
erlang_vm_dirty_cpu_schedulers_online	8080	Number of dirty CPU schedulers online
erlang_vm_dirty_io_schedulers	8080	Number of dirty I/O scheduler threads
erlang_vm_ets_limit	8080	Maximum number of ETS tables

Metric Name	Port	Description
erlang_vm_logical_processors	8080	allowed Number of logical processors configured
erlang_vm_logical_processors_available	8080	Number of logical processors available
erlang_vm_logical_processors_online	8080	Number of logical processors online
erlang_vm_port_count	8080	Number of ports currently existing
erlang_vm_port_limit	8080	Maximum number of ports allowed
erlang_vm_process_count	8080	Number of processes currently existing
erlang_vm_process_limit	8080	Maximum number of processes allowed
erlang_vm_schedulers	8080	Number of scheduler threads
erlang_vm_schedulers_online	8080	Number of schedulers online
erlang_vm_smp_support	8080	1 if compiled with SMP support, 0 otherwise
erlang_vm_threads	8080	1 if compiled with thread support, 0 otherwise
erlang_vm_thread_pool_size	8080	Number of async threads in pool
erlang_vm_time_correction	8080	1 if time correction enabled, 0 otherwise
erlang_vm_wordsize_bytes	8080	Size of Erlang term words in bytes
erlang_vm_atom_count	8080	Number of atoms currently existing
erlang_vm_atom_limit	8080	Maximum number of atoms allowed

Erlang VM Microstate Accounting (MSACC)

Detailed time tracking for scheduler activities with labels: type, id

Metric Name	Port	Description
erlang_vm_msacc_aux_seconds_total	8080	Time spent handling auxiliary jobs (counter)
erlang_vm_msacc_check_io_seconds_total	8080	Time spent checking for new I/O events (counter)
erlang_vm_msacc_emulator_seconds_total	8080	Time spent executing Erlang processes (counter)
erlang_vm_msacc_gc_seconds_total	8080	Time spent in garbage collection (counter)
erlang_vm_msacc_other_seconds_total	8080	Time spent on unaccounted

Metric Name	Port	Description
		activities (counter)
erlang_vm_msacc_port_seconds_total	8080	Time spent executing ports (counter)
erlang_vm_msacc_sleep_seconds_total	8080	Time spent sleeping (counter)
erlang_vm_msacc_alloc_seconds_total	8080	Time spent managing memory (counter)
erlang_vm_msacc_bif_seconds_total	8080	Time spent in BIFs (counter)
erlang_vm_msacc_busy_wait_seconds_total	8080	Time spent busy waiting (counter)
erlang_vm_msacc_ets_seconds_total	8080	Time spent in ETS BIFs (counter)
erlang_vm_msacc_gc_full_seconds_total	8080	Time spent in fullsweep GC (counter)
erlang_vm_msacc_nif_seconds_total	8080	Time spent in NIFs (counter)
erlang_vm_msacc_send_seconds_total	8080	Time spent sending messages (counter)
erlang_vm_msacc_timers_seconds_total	8080	Time spent managing timers (counter)

Erlang VM Allocators

Detailed memory allocator metrics with labels: alloc, instance_no, kind, usage

Metric Name	Port	Description
erlang_vm_allocators	8080	Allocated (carriers_size) and used (blocks_size) memory for different allocators. See erts_alloc(3).

Allocator types include: temp_alloc, sl_alloc, std_alloc, ll_alloc, eheap_alloc, ets_alloc, fix_alloc, literal_alloc, binary_alloc, driver_alloc

[↑ Back to top](#)

Port 9093 - Media & Call Quality Metrics

These metrics provide real-time RTP/RTCP statistics and call quality information per channel.

Metric Name	Port	Description
freeswitch_info	9093	System info with label: version
freeswitch_up	9093	Ready status (1=ready, 0=not ready)
freeswitch_stack_bytes	9093	Stack size in bytes
freeswitch_session_total	9093	Total number of sessions
freeswitch_session_active	9093	Active number of sessions
freeswitch_session_limit	9093	Session limit

Metric Name	Port	Description
rtp_channel_info	9093	RTP channel info with labels for channel details

RTP Audio - Byte Counters

Metric Name	Port	Description
rtp_audio_in_raw_bytes_total	9093	Total bytes received (including headers)
rtp_audio_out_raw_bytes_total	9093	Total bytes sent (including headers)
rtp_audio_in_media_bytes_total	9093	Total media bytes received (payload only)
rtp_audio_out_media_bytes_total	9093	Total media bytes sent (payload only)

RTP Audio - Packet Counters

Metric Name	Port	Description
rtp_audio_in_packets_total	9093	Total packets received
rtp_audio_out_packets_total	9093	Total packets sent
rtp_audio_in_media_packets_total	9093	Total media packets received
rtp_audio_out_media_packets_total	9093	Total media packets sent
rtp_audio_in_skip_packets_total	9093	Inbound packets discarded
rtp_audio_out_skip_packets_total	9093	Outbound packets discarded

RTP Audio - Special Packet Types

Metric Name	Port	Description
rtp_audio_in_jitter_packets_total	9093	Jitter buffer packets received
rtp_audio_in_dtmf_packets_total	9093	DTMF packets received
rtp_audio_out_dtmf_packets_total	9093	DTMF packets sent
rtp_audio_in_cng_packets_total	9093	Comfort Noise Generation packets received
rtp_audio_out_cng_packets_total	9093	Comfort Noise Generation packets sent
rtp_audio_in_flush_packets_total	9093	Flushed packets (buffer resets)

RTP Audio - Jitter & Quality Metrics

Metric Name	Port	Description
rtp_audio_in_jitter_buffer_bytes_max	9093	Largest jitter buffer size in bytes
rtp_audio_in_jitter_seconds_min	9093	Minimum jitter in seconds
rtp_audio_in_jitter_seconds_max	9093	Maximum jitter in seconds
rtp_audio_in_jitter_loss_rate	9093	Packet loss rate due to jitter (ratio)
rtp_audio_in_jitter_burst_rate	9093	Packet burst rate due to jitter (ratio)
rtp_audio_in_mean_interval_seconds	9093	Mean interval between inbound

Metric Name	Port	Description
rtp_audio_in_flaw_total	9093	packets Total audio flaws detected (glitches, artifacts)
rtp_audio_in_quality_percent	9093	Audio quality as percentage (0-100)
rtp_audio_in_quality_mos	9093	Mean Opinion Score (1-5, where 5 is best)

RTCP Metrics

Metric Name	Port	Description
rtcp_audio_bytes_total	9093	Total RTCP bytes
rtcp_audio_packets_total	9093	Total RTCP packets

Go Runtime Metrics

Metric Name	Port	Description
go_goroutines	9090	Number of goroutines currently running
go_threads	9090	Number of OS threads created
go_info	9090	Information about the Go environment (with version label)
go_gc_duration_seconds	9090	Pause duration of garbage collection cycles (summary)
go_memstats_alloc_bytes	9090	Number of bytes allocated and still in use
go_memstats_alloc_bytes_total	9090	Total number of bytes allocated (counter)
go_memstats_heap_alloc_bytes	9090	Heap bytes allocated and still in use
go_memstats_heap_idle_bytes	9090	Heap bytes waiting to be used
go_memstats_heap_inuse_bytes	9090	Heap bytes currently in use
go_memstats_heap_objects	9090	Number of allocated heap objects
go_memstats_heap_released_bytes	9090	Heap bytes released to OS
go_memstats_heap_sys_bytes	9090	Heap bytes obtained from system
go_memstats_sys_bytes	9090	Total bytes obtained from system

Process Metrics

Metric Name	Port	Description
process_cpu_seconds_total	9090	Total user and system CPU time spent (counter)
process_max_fds	9090	Maximum number of open file descriptors
process_open_fds	9090	Current number of open file descriptors
process_resident_memory_bytes	9090	Resident memory size in bytes

Metric Name	Port	Description
process_virtual_memory_bytes	9090	Virtual memory size in bytes
process_virtual_memory_max_bytes	9090	Maximum amount of virtual memory available
process_start_time_seconds	9090	Process start time since Unix epoch

Prometheus HTTP Metrics

Metric Name	Port	Description
promhttp_metric_handler_requests_in_flight	9090	Current number of scrapes being served
promhttp_metric_handler_requests_total	9090	Total number of scrapes by HTTP status code (counter)

[↑ Back to top](#)

Metric Types

- **gauge**: A metric that can go up or down (e.g., `current_calls`, `cpu_idle`)
 - **counter**: A metric that only increases (e.g., `sessions_total`, `failed_scrapes`)
 - **summary**: A metric that tracks quantiles over a sliding time window (e.g., `gc_duration_seconds`)
-

[↑ Back to top](#)

Usage

To scrape these metrics, configure your Prometheus server to scrape all three endpoints:

```
scrape_configs:
  - job_name: 'ims_as_system'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'ims_as_engine'
    static_configs:
      - targets: ['localhost:8080']
    metrics_path: '/metrics'

  - job_name: 'ims_as_media'
    static_configs:
      - targets: ['localhost:9093']
    metrics_path: '/esl'
    params:
      module: ['default']
```

[↑ Back to top](#)

Example Queries

Quick Links:

- [General Metrics \(Port 9090\)](#)
- [Media Quality Metrics \(Port 9093\)](#)
- [TAS Engine Metrics \(Port 8080\)](#)

General Metrics

Current call volume:

```
freeswitch_current_calls
```

Gateway health:

```
freeswitch_sofia_gateway_status{status="UP"}
```

Average ping time to gateways:

```
avg(freeswitch_sofia_gateway_pingtime)
```

Sessions per second rate:

```
freeswitch_current_sps
```

Memory usage:

```
freeswitch_memory_uordblks
```

Media Quality Metrics

Call quality (MOS score):

```
rtp_audio_in_quality_mos
```

Audio quality percentage:

```
rtp_audio_in_quality_percent
```

Jitter rate:

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

Packet loss rate:

```
rtp_audio_in_jitter_loss_rate
```

Average jitter:

```
avg(rtp_audio_in_jitter_seconds_max - rtp_audio_in_jitter_seconds_min)
```

RTP bandwidth (inbound):

```
rate(rtp_audio_in_media_bytes_total[1m]) * 8
```

Audio flaws detected:

```
increase(rtp_audio_in_flaw_total[5m])
```

TAS Engine Metrics

Active calls by type:

```
active_calls
```

Diameter peer health:

```
diameter_peer_state{application="sh"}
```

Call attempt rate:

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

HLR lookup latency (95th percentile):

```
histogram_quantile(0.95, hlr_data_duration_milliseconds)
```

OCS authorization latency:

```
histogram_quantile(0.99, ocs_authorization_duration_milliseconds)
```

Subscriber data lookup rate:

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

Diameter request success rate:

```
rate(diameter_responses_total[5m]) / rate(diameter_requests_total[5m])
```

Event Socket connection status:

```
event_socket_connected
```

Mnesia transaction performance:

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

Mnesia failed transaction rate:

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

Erlang VM process count:

```
erlang_vm_process_count
```

Erlang VM memory usage:

```
erlang_vm_memory_bytes_total
```

Garbage collection rate:

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

Scheduler run queue length:

```
erlang_vm_statistics_run_queues_length
```

ETS table count:

```
erlang_vm_memory_ets_tables
```

HTTP dialplan request duration (median):

```
histogram_quantile(0.5, http_dialplan_request_duration_milliseconds)
```

[↑ Back to top](#)

Metric Time Unit Configuration

Important for Developers:

All duration metrics in this system use `duration_unit: false` in their Histogram declarations. This is **critical** because:

1. The Prometheus Elixir library automatically detects metric names ending in `_milliseconds`
2. By default, it converts native Erlang time units to milliseconds automatically
3. Our code already converts time to milliseconds using `System.convert_time_unit/3`
4. Without `duration_unit: false`, the library would convert milliseconds to nanoseconds (dividing by $\sim 1,000,000$)

Example:

```
# Correct configuration
Histogram.declare(
  name: :http_dialplan_request_duration_milliseconds,
  help: "Duration of HTTP dialplan requests in milliseconds",
  labels: [:call_type],
  buckets: [100, 250, 500, 750, 1000, 1500, 2000, 3000, 5000],
  duration_unit: false # REQUIRED to prevent double conversion
)

# Measuring time correctly
start_time = System.monotonic_time()
# ... do work ...
end_time = System.monotonic_time()
duration_ms = System.convert_time_unit(end_time - start_time, :native,
:millisecond)
Histogram.observe([name:
:http_dialplan_request_duration_milliseconds], duration_ms)
```

[↑ Back to top](#)

Grafana Dashboard Integration

The metrics can be visualized in Grafana using the Prometheus data source.

Recommended Dashboard Layout

Row 1: Call Volume & Health

- Active calls gauge (active_calls)
- Call attempts rate by type (rate(call_attempts_total[5m]))
- Call rejection rate (rate(call_rejections_total[5m]))
- Gateway health (freeswitch_sofia_gateway_status)

Row 2: Performance (Latency Percentiles)

- P95 HTTP dialplan request time by call type
- P95 Sh subscriber data lookup time
- P95 HLR lookup time
- P95 OCS authorization time
- P95 Diameter response time by application

Row 3: Success Rates

- Subscriber data lookup success rate
- HLR lookup success rate
- OCS authorization success rate
- Diameter peer state

Row 4: Media Quality

- Call quality MOS score (rtp_audio_in_quality_mos)
- Audio quality percentage (rtp_audio_in_quality_percent)
- Jitter statistics
- Packet loss rate

Row 5: System Resources

- Erlang VM process count
- Erlang VM memory usage
- ETS table count
- Scheduler run queue length
- Garbage collection rate

Row 6: Error Tracking

- Call parameter errors
- Authorization failures
- Event Socket connection status
- Mnesia transaction failures

Example Panel Queries

Active Calls by Type:

```
sum by (call_type) (active_calls)
```

P95 Dialplan Generation Latency:

```
histogram_quantile(0.95,  
  rate(http_dialplan_request_duration_milliseconds_bucket[5m])  
)
```

Diameter Success Rate:

```
rate(diameter_responses_total{result="success"}[5m]) /  
rate(diameter_requests_total[5m]) * 100
```

Media Quality - Average MOS:

```
avg(rtp_audio_in_quality_mos)
```

[↑ Back to top](#)

Alerting Examples

Critical Alerts (Page Immediately)

System Down - No Call Attempts:

```
alert: SystemDown
expr: rate(call_attempts_total[5m]) == 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "TAS system appears down - no call attempts"
  description: "No call attempts detected for 2 minutes"
```

Diameter Peer Down:

```
alert: DiameterPeerDown
expr: diameter_peer_state == 0
for: 1m
labels:
  severity: critical
annotations:
  summary: "Diameter peer {{ $labels.peer_host }} is down"
  description: "Peer for {{ $labels.application }} application is unavailable"
```

Event Socket Disconnected:

```
alert: EventSocketDisconnected
expr: event_socket_connected == 0
for: 30s
labels:
  severity: critical
annotations:
  summary: "Event Socket {{ $labels.connection_type }} disconnected"
  description: "Critical communication channel down"
```

High Severity Alerts

High Diameter Latency:

```
alert: HighDiameterLatency
expr: |
  histogram_quantile(0.95,
    rate(diameter_response_duration_milliseconds_bucket[5m])
  ) > 1000
for: 5m
```



```
labels:
  severity: high
annotations:
  summary: "High Diameter latency detected"
  description: "P95 latency is {{ $value }}ms"
```

OCS Authorization Failures:

```
alert: OCSAuthFailures
expr: |
  rate(ocs_authorization_attempts_total{result="no_credit"}[5m]) /
  rate(ocs_authorization_attempts_total[5m]) > 0.1
for: 5m
labels:
  severity: high
annotations:
  summary: "High rate of OCS no-credit responses"
  description: "{{ $value | humanizePercentage }}" of requests denied
  credit"
```

High Call Rejection Rate:

```
alert: HighCallRejectionRate
expr: |
  rate(call_rejections_total[5m]) /
  rate(call_attempts_total[5m]) > 0.05
for: 5m
labels:
  severity: high
annotations:
  summary: "Call rejection rate above 5%"
  description: "{{ $value | humanizePercentage }}" of calls rejected"
```

Poor Media Quality:

```
alert: PoorMediaQuality
expr: avg(rtp_audio_in_quality_mos) < 3.5
for: 3m
labels:
  severity: high
annotations:
  summary: "Poor call quality detected"
  description: "Average MOS score is {{ $value }}"
```

Warning Alerts

High Memory Usage:

```
alert: HighMemoryUsage
```

```
expr: |
    erlang_vm_memory_bytes_total{kind="processes"} /
    (erlang_vm_process_limit * 1000000) > 0.8
for: 10m
labels:
    severity: warning
annotations:
    summary: "Erlang VM memory usage high"
    description: "Process memory at {{ $value | humanizePercentage }}"
```

High Scheduler Run Queue:

```
alert: HighSchedulerRunQueue
expr: erlang_vm_statistics_run_queues_length > 10
for: 5m
labels:
    severity: warning
annotations:
    summary: "High scheduler run queue length"
    description: "Run queue length is {{ $value }}"
```

Mnesia Transaction Failures:

```
alert: MnesiaTransactionFailures
expr: rate(erlang_mnesia_failed_transactions[5m]) > 1
for: 5m
labels:
    severity: warning
annotations:
    summary: "Mnesia transaction failures detected"
    description: "{{ $value }} failures per second"
```

[↑ Back to top](#)

Troubleshooting with Metrics

Problem: Metrics showing unrealistic values (nanoseconds instead of milliseconds)

Symptoms:

- Histogram values in the billions
- Latency metrics showing microsecond/nanosecond values

Cause: Missing `duration_unit: false` in Histogram declaration

Solution: Add `duration_unit: false` to all duration histogram declarations:

```
Histogram.declare(  
  name: :my_metric_duration_milliseconds,  
  # ... other options ...  
  duration_unit: false  
)
```

Problem: Calls are slow

Investigation Steps:

1. Check overall dialplan generation time:

```
histogram_quantile(0.95,  
  rate(http_dialplan_request_duration_milliseconds_bucket[5m]))
```

2. Break down by component:

```
# Subscriber data lookup  
histogram_quantile(0.95,  
  rate(subscriber_data_duration_milliseconds_bucket[5m]))  
  
# HLR lookup  
histogram_quantile(0.95,  
  rate(hlr_data_duration_milliseconds_bucket[5m]))  
  
# OCS authorization  
histogram_quantile(0.95,  
  rate(ocs_authorization_duration_milliseconds_bucket[5m]))
```

3. Check module-specific delays:

```
histogram_quantile(0.95,  
  rate(dialplan_module_duration_milliseconds_bucket[5m])  
) by (module)
```

Common Causes:

- External system latency (HSS, HLR, OCS)
- Network issues
- Database contention
- High system load

Problem: Calls are failing

Investigation Steps:

1. Check call rejection reasons:

```
sum by (reason) (rate(call_rejections_total[5m]))
```

2. Check authorization decisions:

```
sum by (decision) (rate(authorization_decisions_total[5m]))
```

3. Check Diameter peer health:

```
diameter_peer_state
```

4. Check Event Socket connection:

```
event_socket_connected
```

Problem: High load

Investigation Steps:

1. Check call volume:

```
rate(call_attempts_total[5m])  
active_calls
```

2. Check Erlang VM resources:

```
erlang_vm_process_count  
erlang_vm_statistics_run_queues_length  
erlang_vm_memory_bytes_total
```

3. Check garbage collection:

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

Problem: Poor Media Quality

Investigation Steps:

1. Check MOS scores:

```
rtp_audio_in_quality_mos  
rtp_audio_in_quality_percent
```

2. Check jitter:

```
rtp_audio_in_jitter_seconds_max  
rtp_audio_in_jitter_loss_rate
```

3. Check packet loss:

```
rtp_audio_in_skip_packets_total  
rtp_audio_in_flaw_total
```

4. Check bandwidth usage:

```
rate(rtp_audio_in_media_bytes_total[1m]) * 8
```

[↑ Back to top](#)

Performance Baselines

Typical Values (Well-Tuned System)

Latency (P95):

- HTTP dialplan request: 200-500ms
- Subscriber data (Sh) lookup: 50-150ms
- HLR data lookup: 100-300ms
- OCS authorization: 100-250ms
- Diameter requests: 50-200ms
- Dialplan module processing: 10-50ms per module

Success Rates:

- Call completion: >95%
- Subscriber data lookups: >99%
- HLR lookups: >98%
- OCS authorizations: >99% (excluding legitimate no-credit)
- Diameter peer uptime: >99.9%

Media Quality:

- MOS score: >4.0
- Audio quality percentage: >80%
- Jitter: <30ms
- Packet loss rate: <1%

System Resources:

- Erlang process count: <50% of limit
- Erlang memory usage: <70% of available
- Scheduler run queue: <5
- ETS tables: <1000

Capacity Planning

Per-Server Capacity (recommended maximums):

- Concurrent calls: 500-1000 (depends on hardware)
- Calls per second: 20-50 CPS

- Registered subscribers: 10,000-50,000

Scaling Indicators (add capacity when):

- Active calls consistently >70% of capacity
 - Erlang process count >70% of limit
 - P95 latency degrading
 - Scheduler run queues consistently >10
-

[↑ Back to top](#)

Best Practices

Monitoring Strategy

1. Set up dashboards for different audiences:

- Operations dashboard: Call volume, success rates, system health
- Engineering dashboard: Latency percentiles, error rates, resource usage
- Executive dashboard: High-level KPIs, uptime, cost metrics

2. Configure alerts at multiple levels:

- Critical: Page on-call (system down, major outage)
- High: Alert during business hours (degraded performance)
- Warning: Track in ticket system (potential issues)

3. Use appropriate time ranges:

- Real-time monitoring: 5-minute windows
- Troubleshooting: 15-minute to 1-hour windows
- Capacity planning: Daily/weekly aggregates

4. Focus on user impact:

- Prioritize end-to-end latency metrics
- Track success rates over individual error counters
- Monitor media quality for user experience

Query Performance

1. Use recording rules for frequently-used queries:

```
groups:  
- name: ims_as_aggregations  
  interval: 30s
```

```
rules:
  - record: job:call_attempts:rate5m
    expr: rate(call_attempts_total[5m])

  - record: job:dialplan_latency:p95
    expr: histogram_quantile(0.95,
rate(http_dialplan_request_duration_milliseconds_bucket[5m]))
```

2. **Avoid high-cardinality labels** in queries (e.g., don't group by phone number)

3. **Use appropriate rate intervals:**

- Short-term trends: [5m]
- Medium-term trends: [1h]
- Long-term trends: [1d]

Metric Cardinality

Monitor cardinality to prevent Prometheus performance issues:

```
# Check metric cardinality
count by (__name__) ({__name__=~".+"})
```

High-cardinality risks:

- Labels with unique values per call (phone numbers, call IDs)
- Unbounded label values
- Labels with >1000 unique values

Solution:

- Use labels for categories, not unique identifiers
- Aggregate high-cardinality data in external systems
- Use recording rules to pre-aggregate

[↑ Back to top](#)



Number Translation

◇ [Back to Main Documentation](#)

Number translation converts phone numbers between different formats to ensure consistent E.164 formatting throughout the system.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - Number translation configuration (number_translate)
- ◇ [Operations Guide](#) - Number translation testing in Control Panel

Call Processing Flow

- ◇ [Dialplan Configuration](#) - Using translated numbers in dialplan (translation happens first)
- ◇ [Sh Interface](#) - Sh lookup uses translated numbers
- ◇ [Online Charging](#) - OCS receives translated numbers
- ◇ [SS7 MAP](#) - HLR queries use translated numbers

Related Services

- ⚙ [Supplementary Services](#) - CLI blocking prefix stripping during translation
- ◇ [Voicemail](#) - Voicemail numbers in translation

Monitoring

- ◇ [Metrics Reference](#) - Number translation metrics

Number Translation

Number translation converts phone numbers between different formats (local, national, international) to ensure consistent E.164 formatting throughout the system.

What is Number Translation?

Number translation normalizes phone numbers to E.164 format (international standard) before call processing. This ensures:

- Consistent numbering throughout the system
- Proper routing to on-net and off-net destinations
- Compatibility with international SIP trunks and IMS networks

E.164 Format: [Country Code][National Number] (no + prefix, no spaces)

- Example: 61403123456 (Australia mobile)
- Example: 16505551234 (US number)

When Translation Occurs

Translation happens:

- **Before** Sh lookups
- **Before** HLR lookups
- **Before** OCS authorization
- **Before** dialplan XML is generated

For MT Calls: Translate destination number (called party) **For MO Calls:** Translate both source and destination numbers

Configuration

```
config :tas,
  number_translate: %{
    country_code: :PF,          # ISO 3166-1 alpha-2 country code
    localAreaCode: "617"       # Default area code for short numbers
  }
```

Parameters:

- **country_code:** ISO country code as atom (e.g., :AU, :US, :PF)
- **localAreaCode:** Area code prepended to short local numbers

Supported Country Codes

The TAS includes translation logic for these countries:

Country Code	Country	Supported Formats	E.164 Prefix
:AU	Australia	0NSN (10-digit), SN (8-digit), E.164	61
:US	United States	NPANXXXXXX (10-digit), 1+NPANXXXXXX, E.164	1

Country Code	Country	Supported Formats	E.164 Prefix
:PF	French Polynesia	Local (6-digit), National (8-digit), E.164	689

Adding New Country Codes: Country-specific translation logic must be implemented in `lib/numbertranslation.ex`. Contact your integration engineer to add new countries.

Special Translation Behaviors

1. CLI Blocking Prefix Stripping

Before format translation, CLI blocking prefixes are removed:

```
Input:  *67555123456
Step 1: Strip *67 → 555123456
Step 2: Translate → 1555123456 (if US)
```

2. SIP Parameter Stripping

Parameters after semicolons are removed:

```
Input:  61403123456;npdi;rn=+614000000000
Step 1: Strip ;npdi;rn=... → 61403123456
Step 2: Translate → 61403123456
```

3. Non-Digit Character Removal

All non-digit characters (except +) are stripped:

```
Input:  +61 (403) 123-456
Step 1: Strip formatting → +61403123456
Step 2: Translate → 61403123456
```

Variables Set After Translation

Variable	Value	Description
<code>destination_number</code>	E.164 format	Normalized destination number
<code>tas_destination_number</code>	E.164 format	Same as <code>destination_number</code> (both set for compatibility)
<code>effective_caller_id_number</code>	E.164 format	Normalized source number (MO calls)

What Happens When Translation Fails

Scenario: Undefined Country Code

```
config :tas, number_translate: %{country_code: :XX} # Invalid
```

Result: {:error, "Undefined Country Code"} - call rejected

Scenario: Invalid Number Format

Input: "abc123" (contains letters)

Step 1: Strip non-digits → "123"

Step 2: Too short, cannot match any pattern

Result: May pass through as-is or reject based on dialplan logic

Best Practice: Always validate subscriber provisioning with correct E.164 numbers in HSS.

Testing Number Translation

Web UI Translation Tester (/translate):

1. Navigate to /translate in Control Panel
2. Select country code from dropdown
3. Enter test number in any format
4. View translated E.164 output
5. Test multiple formats to validate

Common Test Scenarios:

- Local short codes → E.164
- National format (0NSN) → E.164
- International format (+CC) → E.164
- Numbers with CLI prefixes → stripped and translated
- Numbers with formatting (spaces, dashes) → clean E.164

Troubleshooting Number Translation

Problem: Calls failing with "UNALLOCATED_NUMBER"

1. Check translated number format:

- Use /translate tool to test number
- Verify output matches expected E.164 format
- Confirm country code and area code are correct

2. Check Sh lookup:

- Translated number is used for Sh query
- Use /sh_test with translated number
- Verify subscriber exists with that MSISDN

3. Check dialplan variables:

- Review logs for destination_number value
- Confirm translation occurred before dialplan

Problem: Wrong area code applied

```
# Configuration
config :tas, number_translate: %{
  country_code: :AU,
  localAreaCode: "617" # Wrong for your region
}

# Input: 12345678 (8-digit local)
# Output: 6161712345678 (incorrect - double area code)
# Fix: Set correct localAreaCode for your deployment
```

Problem: International numbers not recognized

Check if number includes country code:

- ☞ +61403123456 or 61403123456 → Recognized
- ☞ 0403123456 in wrong country_code config → Misrouted

MO vs MT Translation Behavior

MT (Mobile Terminated) Calls:

- Only destination number (called party) is translated
- Source number (caller) passed through as-is from SIP
- Destination used for Sh lookup of called subscriber

MO (Mobile Originating) Calls:

- Destination number (called party) translated
- Source number (calling party) also translated
- Source used for Sh lookup of calling subscriber
- Both numbers normalized for consistent logging/CDR

Best Practices

1. Use Correct Country Code:

- Set country_code to match your deployment region

- Test thoroughly before production

2. Configure Appropriate Local Area Code:

- `localAreaCode` should match your network's default area
- Used for short numbers without area code

3. Test All Number Formats:

- Local (short codes)
- National (0NSN format)
- International (+CC format)
- Special service numbers (emergency, voicemail)

4. Monitor Translation Logs:

- Check for "Undefined Country Code" errors
- Watch for unexpected number formats
- Validate E.164 output matches expectations

5. Document Your Numbering Plan:

- Define which formats subscribers will use
- Test each format in `/translate` tool
- Train operations staff on expected formats



Online Charging (Ro Interface)

◇ [Back to Main Documentation](#)

Configuration and operational details for the Ro (Online Charging) Diameter interface.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - Diameter peer configuration and OCS settings
- ◇ [Operations Guide](#) - OCS testing in Control Panel

Call Processing Integration

- ◇ [Dialplan Configuration](#) - Call flow with OCS authorization
- ⚙ [Supplementary Services](#) - Emergency calls bypass OCS, skipped_regex
- ◇ [Number Translation](#) - Number normalization before OCS request

Related Interfaces

- ◇ [Sh Interface](#) - Subscriber data interface (also uses Diameter)
- ◇ [SS7 MAP](#) - Alternative for subscriber routing

Monitoring

- ◇ [Metrics Reference](#) - OCS/Ro metrics, authorization duration, credit events
- ◇ [Dialplan Metrics](#) - Call-level OCS metrics

Ro Configuration

If Ro (Online Charging) is to be used it must be enabled in the config.

You will of course need a connected Diameter peer that can handle Ro or a DRA that has one.

`periodic_ccr_time_seconds` sets how often during a session a Credit-Control-Request type 2 (Update) is sent. This will probably need to be adjusted based on the needs of the OCS.

If the Credit-Control-Request type 1 (Initial) returns the max usage seconds for all the

balance, we can set a scheduled hangup for that time. This is controlled by the `schedule_hangup_auth` flag which if enabled will get the seconds allocated in the CCR-I and set the call to hangup at that time.

`skipped_regex` (optional) allows you to specify regular expression patterns to match destination numbers that should bypass online charging. Calls to destinations matching these patterns will be authorized without contacting the OCS. This is useful for internal services, voicemail, emergency numbers, or other destinations that should not be charged. If omitted or set to an empty list, all calls will go through online charging when enabled. Patterns use Elixir regex syntax (e.g., `"^2222"` matches numbers starting with 2222, `"^344[4-6]"` matches numbers starting with 3444, 3445, or 3446).

```
config :tas,
  ...
  online_charging: %{
    enabled: true,
    periodic_ccr_time_seconds: 10,
    schedule_hangup_auth: true,
    skipped_regex: ["^2222", "^2223", "^8999", "^344[4-6]"] # Optional
  },
```

Online Charging Operational Behavior

How Periodic Charging Works:

When a mobile-originated (MO) call is answered:

1. The system sends an initial CCR-I (Initial Credit Control Request) to authorize the call
2. If credit is available, the OCS returns allocated time (e.g., 120 seconds)
3. A background process starts that runs every `periodic_ccr_time_seconds` to reauthorize the call
4. This process sends CCR-U (Update) requests to check if the subscriber still has credit
5. Each successful CCR-U extends the call and grants additional time
6. When the call ends, a CCR-T (Terminate) is sent for final accounting

Request Type Values:

- 1 = Initial (CCR-I) - sent when call is first authorized
- 2 = Update (CCR-U) - sent on answer and periodically during the call
- 3 = Terminate (CCR-T) - sent when call ends
- 4 = Event - used for SMS or one-time event charging

What Happens When Credit Runs Out:

If a periodic CCR-U returns zero seconds or an error:

- The periodic charging process stops automatically
- If `schedule_hangup_auth` is enabled, the call will terminate when allocated

time expires

- The subscriber's call ends and they receive appropriate notification

Tuning periodic_ccr_time_seconds:

Choose the interval based on your operational needs:

- **Lower values (5-10 seconds):** More responsive to credit exhaustion, catches out-of-credit situations quickly, but creates higher load on OCS
- **Higher values (30-60 seconds):** Lower OCS load and network traffic, but subscribers may consume slightly more credit than available between checks
- **Recommended:** Start with 10-30 seconds based on your minimum credit increment and OCS capacity

Monitoring Periodic Charging:

Watch these metrics to ensure periodic charging is working correctly:

- `online_charging_events_total{event_type="reauth"}` - tracks periodic CCR-U requests
- `online_charging_events_total{event_type="reauth",result="success"}` - successful reauthorizations
- `online_charging_events_total{event_type="reauth",result="nocredit"}` - subscribers running out of credit mid-call
- High reauth failure rates indicate OCS connectivity or performance issues

Troubleshooting Periodic Charging:

If calls are not being reauthorized properly:

1. Check logs for "Sending periodic CCR-U for <call_id>" messages - confirms the background process is running
2. Look for "Periodic CCR failed - Stopping job" warnings - indicates credit exhaustion or OCS errors
3. Verify `ocs_authorization_duration_milliseconds` metric shows reasonable response times
4. Confirm OCS is returning valid CCA-U responses with granted time units
5. Check that `periodic_ccr_time_seconds` is not set too aggressively for your OCS capacity



Operations Guide

◇ [Back to Main Documentation](#)

This document covers operational monitoring and management features available in the Control Panel.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - System configuration reference
- ◇ [Metrics Reference](#) - Prometheus metrics and monitoring

Monitoring & Testing Tools

- ◇ [HLR & Call Simulator](#) - Testing tools for HLR and call simulation
- ◇ [IMS Conference Server](#) - Conference management and monitoring
- ◇ [Dialplan Metrics](#) - Dialplan-specific metrics

Call Processing & Services

- ◇ [Dialplan Configuration](#) - Call routing and dialplan reference
- ◇ [Sh Interface](#) - Subscriber data testing
- ◇ [Online Charging](#) - OCS testing
- ◇ [Number Translation](#) - Number translation testing
- ◇ [Voicemail](#) - Voicemail management

Integration Interfaces

- ◇ [SS7 MAP](#) - HLR/MAP testing
- ⚙ [Supplementary Services](#) - Emergency calling, call forwarding

Operations

This section covers operational monitoring and management features available in the OmniTAS Control Panel.

Table of Contents

- [Subscribers View](#)
- [Call Detail Records \(CDR\)](#)
- [Active Calls Monitoring](#)
- [IMS Conference Server](#)
- [Gateway Status](#)
- [Diameter Peer Status](#)
- [Logs Viewer](#)
- [Cell Tower Database](#)
- [Call Simulator](#)
- [HLR/MAP Testing](#)
- [Other Views](#)

Subscribers View

The Subscribers view provides real-time monitoring of IMS subscriber registrations stored in the Sofia SIP registration database.

Access: Navigate to /subscribers in the Control Panel

Features

- **Registration List:** View all active subscriber registrations
- **Registration Details:** Click on any registration to view complete details including:
 - SIP User and Realm
 - Contact URI
 - Registration status and expiration
 - Network information (IP, port, hostname)
 - Authentication details
 - Cell tower location (when available via P-Access-Network-Info)
 - MCC/MNC, Radio Type, TAC/LAC, Cell ID
 - Geographic coordinates and coverage range
 - Interactive map view powered by OpenStreetMap and OpenCellID data

Data Source

Registration data is queried directly from the Sofia registration database, providing real-time visibility into subscriber attachment status. Cell tower locations are resolved using the OpenCellID database when subscribers provide P-Access-Network-Info headers in their SIP REGISTER messages.

Use Cases

- Monitor active subscriber registrations

- Verify subscriber attachment status
 - Troubleshoot registration issues
 - Audit subscriber connectivity
-

Call Detail Records (CDR)

The CDR view provides access to call detail records stored by TAS for billing, troubleshooting, and analytics purposes.

Access: Navigate to /cdr in the Control Panel

Features

- **Paginated View:** Browse through call records (100 per page with Previous/Next controls)
- **Advanced Search:** Powerful search with support for exact match, inverse/exclude, and multiple terms
- **Column Selection:** Customize which fields to display
 - Click "**Columns**" button to open column picker modal
 - Select/deselect individual columns
 - **Select All / Deselect All** quick actions
 - Selection persists across sessions (saved to browser localStorage)
 - Shows "X / Y columns" counter
- **Sortable Columns:** Click any column header to sort (ascending/descending)
 - Visual indicators (▲ ascending, ▼ descending)
 - Sorted column highlighted in blue
 - Resets to page 1 when sorting changes
- **Multiple Filter Options:**
 - **Text Search:** Search across all fields with advanced operators
 - **Date Range Filter:** Filter by start/end date and time (datetime picker)
 - **Field-Specific Filter:** Filter by exact field value (hangup cause, caller ID, destination, context)
 - **Active Filter Display:** Visual chips show currently active filters
 - **Clear All:** One-click removal of all active filters
- **Detailed Information:** Click on any CDR row to expand and view all fields:
 - Call parties (caller ID name/number, destination number)
 - Timestamps (start, answer, end)
 - Duration and billed seconds
 - Hangup cause (color-coded: green=normal, yellow=cancel, red=error)
 - Call UUIDs (A-leg and B-leg)
 - Context and account code
 - All available database fields in alphabetical order
- **Color-Coded Hangup Causes:**

- ◇ Green: NORMAL_CLEARING
- ◇ Yellow: Cancelled calls
- ◇ Red: Error conditions
- **Total Count:** Real-time display of total matching records
- **Responsive Layout:** Filters wrap appropriately on smaller screens

How to Use

1. Basic Viewing:

- Page loads with latest 100 CDR records (sorted by start_stamp descending)
- Total record count shown in top-right
- Use **Previous** / **Next** buttons to navigate pages
- Click any row to expand and see all fields

2. Customize Columns:

- Click **"Columns"** button in top-right
- Modal shows all available fields
- Check/uncheck fields to show/hide columns
- Use **"Select All"** or **"Deselect All"** for quick selection
- Settings automatically saved to browser
- Close modal to apply changes

3. Sort Data:

- Click any column header to sort by that field
- First click: Descending (▼)
- Second click: Ascending (▲)
- Third click: Back to descending
- Sorted column highlighted in blue

4. Search Records:

- Enter search query in **"Search"** box
- Supports advanced operators (see Search Syntax below)
- Searches across multiple fields: caller_id_number, destination_number, uuid, caller_id_name, hangup_cause
- Click **"Apply"** to execute search

5. Filter by Date Range:

- Use **"Start Date"** and **"End Date"** datetime pickers
- Both dates required for date filtering
- Supports date and time selection
- Click **"Apply"** to filter

6. Filter by Specific Field:

- Select field from "**Select Field to Filter**" dropdown:
 - Hangup Cause
 - Caller ID
 - Destination
 - Context
- Enter exact value in "**Enter Filter Value**"
- Click "**Apply**" to filter

7. Combine Filters:

- All filters can be used simultaneously:
 - Text search + Date range + Field filter all work together
- Active filters shown as chips below the filter form
- Click "**Clear All**" to remove all filters at once

8. View Details:

- Click any CDR row to expand
- Shows all database fields in a grid layout
- Fields displayed in alphabetical order
- Hangup cause color-coded for quick identification
- Click row again to collapse

Advanced Search Syntax

The search box supports powerful query syntax for precise record filtering across multiple fields simultaneously.

How Search Works:

The search engine checks **all searchable fields** in each CDR record. A record is included in results when it matches your search criteria in **any** of these fields:

- caller_id_number
- destination_number
- uuid
- caller_id_name
- hangup_cause

Search Operators (can be combined):

1. Contains Search (default):

- Syntax: term (no quotes)
- Matches: Records where **any field contains** the term anywhere within it

- SQL: Uses LIKE '%term%' across all searchable fields joined with OR
- Example: 61480 matches "61480123456", "55561480999", etc.

2. **Exact Match:**

- Syntax: "term" (with double quotes)
- Matches: Records where **any field exactly equals** the term
- SQL: Uses = 'term' across all searchable fields joined with OR
- Example: "911" matches only exactly "911", not "9115" or "1911"

3. **Inverse/Exclude:**

- Syntax: !term (exclamation mark prefix, no quotes)
- Matches: Records where **no field contains** the term
- SQL: Uses NOT LIKE '%term%' across all searchable fields joined with AND
- Example: !NORMAL excludes any record with "NORMAL" in any field

4. **Exact Inverse/Exclude:**

- Syntax: !"term" (exclamation mark + double quotes)
- Matches: Records where **no field exactly equals** the term
- SQL: Uses != 'term' across all searchable fields joined with AND
- Example: !"NORMAL_CLEARING" excludes records where any field is exactly "NORMAL_CLEARING"

5. **Multiple Terms with AND:**

- Syntax: term1 AND term2 (case-insensitive AND)
- Matches: Records matching **all terms** (each term can match different fields)
- Each term is processed with its own operator (quotes, !, etc.)
- Terms are combined with AND in SQL
- Example: "911" AND "12345" finds records with "911" in one field AND "12345" in another

Search Execution Logic:

For each CDR record:

For normal search (no !):

- Check if ANY field contains/equals the term → Include if TRUE
- SQL: field1 LIKE '%term%' OR field2 LIKE '%term%' OR ...

For inverse search (!):

- Check if ALL fields do NOT contain/equal the term → Include if TRUE
- SQL: field1 NOT LIKE '%term%' AND field2 NOT LIKE '%term%' AND ...

For AND searches:

- Each term is evaluated separately
- All term conditions must be TRUE → Include if TRUE
- SQL: (term1_conditions) AND (term2_conditions) AND ...

Complex Search Examples:

Query	How It Works	Result
61480	Contains search across all fields	All records with "61480" anywhere (caller, destination, UUID, etc.)
"911"	Exact match across all fields	Records where any field is exactly "911"
!NORMAL_CLEARING	Inverse contains search	Excludes records with "NORMAL_CLEARING" in ANY field (failed calls)
!"NORMAL_CLEARING"	Exact inverse	Excludes records where any field exactly equals "NORMAL_CLEARING"
"911" AND "12345"	Exact "911" AND exact "12345"	Records with both values (e.g., caller="12345", destination="911")
!NORMAL AND 61480	Inverse contains "NORMAL" AND contains "61480"	Non-normal calls involving "61480"
!"ANSWER" AND !NORMAL	Exact inverse "ANSWER" AND inverse contains "NORMAL"	Exclude answered calls and anything with "NORMAL"
61480 AND !NORMAL_CLEARING	Contains "61480" AND inverse contains "NORMAL_CLEARING"	Failed calls involving "61480"

Practical Use Cases:

- **Find specific number:** 61480123456 - Contains search finds partial matches
- **Find exact emergency calls:** "911" - Only calls to exactly "911"
- **All failed calls:** !NORMAL_CLEARING - Exclude successful calls
- **Specific caller's failed calls:** "61480123456" AND !NORMAL - Combine exact caller with inverse
- **Exclude test numbers:** !test AND !demo - Multiple inverse searches
- **Complex debugging:** 61480 AND !"ANSWER" AND !CANCEL - Contains one term, exclude exact and partial others

Data Source

CDR data is queried directly from the TAS CDR SQLite database.

The schema may vary between deployments based on specific requirements.

CDR Export Options

Important: CDR records can be exported in various formats to support integration with billing systems, analytics platforms, and reporting tools.

The CDR database schema and export formats are deployment-specific. When setting up your system, **please request the specific CDR output formats you need from your integration engineer**. Common export formats include:

- CSV (Comma-Separated Values)
- JSON (for API integration)
- XML
- Direct database access
- Custom formatted exports

Your integration engineer can configure CDR export mechanisms tailored to your operational and billing requirements.

Use Cases

- **Call Troubleshooting:** Search for specific calls by number or UUID to debug issues
- **Billing Reconciliation:** Filter by date range to match billing periods
- **Quality Analysis:** Filter by hangup cause to identify problem patterns
- **Emergency Call Auditing:** Search for "911" to verify emergency call handling
- **Customer Support:** Look up specific customer calls by caller ID or destination
- **Pattern Analysis:** Sort by duration or timestamps to identify anomalies
- **Compliance & Record Keeping:** Date range filters for regulatory reporting
- **Failed Call Analysis:** Use !NORMAL_CLEARING to find all failed calls
- **Context-Based Reports:** Filter by context to analyze specific call flows

Configuration

Default Visible Columns

You can configure which CDR fields are **shown by default** in the LiveView by setting `cdrs_field_list` in your `config/runtime.exs`:

```
config :tas,  
  cdrs_field_list: [  
    "caller_id_number",  
    "destination_number",  
    "start_stamp",
```



```
"duration",  
"hangup_cause"  
]
```

Behavior:

- If `cdrs_field_list` is **not set**: All available CDR fields are shown by default
- If `cdrs_field_list` **is set**: Only the specified fields are shown by default, but **all other fields remain available** in the column picker
- If a field in the list doesn't exist in the CDR data, it will be automatically skipped
- Field names can be specified as strings or atoms
- Users can manually select additional columns from the column picker at any time

Use Cases:

- Set a clean default view with only essential fields visible
- Reduce information overload for new users
- Standardize the initial column layout across all users
- Keep advanced fields hidden by default but still accessible

Example Configuration:

```
# Show only essential call information by default  
cdrs_field_list: [  
  "start_stamp",  
  "caller_id_number",  
  "destination_number",  
  "duration",  
  "billsec",  
  "hangup_cause"  
]
```

Note: This configuration sets the *default visible* columns. All CDR fields remain available in the "Columns" picker - users can manually show/hide any field they need.

API / Programmatic Access

For automated CDR analysis, the underlying Elixir API supports all search features:

```
# Elixir API examples  
Tas.Cdr.get_filtered_records(search: "911")  
Tas.Cdr.get_filtered_records(search: "\"911\"")  
Tas.Cdr.get_filtered_records(search: "!NORMAL_CLEARING")
```

```
Tas.Cdr.get_filtered_records(search: "\"911\" AND \"12345\"")
```

See `lib/cdr/cdr.ex` for full API documentation.

Troubleshooting

No Results Found

1. Check for typos in search terms
2. Try removing quotes for broader search
3. Verify the term exists in searchable fields
4. Check date range isn't too restrictive

Too Many Results

1. Add more AND terms to narrow
2. Use exact match with quotes
3. Apply date range filters
4. Use field-specific filters

Unexpected Results

1. Remember search applies to ALL searchable fields
2. Check if term appears in unexpected field (like UUID)
3. Use exact match to avoid partial matches
4. Verify inverse logic (AND vs OR)

Tips

- **Column Selection:** Hide unused columns to focus on relevant data and improve performance
- **Combine Filters:** Use search + date range + field filter together for precise queries
- **Date Range Performance:** Narrow date ranges return results faster for large databases
- **Sort for Analysis:** Sort by duration to find long/short calls, or by timestamp to see call patterns
- **Active Filter Chips:** Use visual chips to verify which filters are currently active
- **Persistent Settings:** Column selections are saved per browser, useful for different analysis tasks
- **Color Coding:** Quickly scan hangup causes - green is good, red needs investigation
- **Expandable Details:** Click rows to see all fields without cluttering the main view
- **Search Operators:** Master the search syntax for powerful filtering:
 - Use quotes for exact matches: "911"

- Use ! to exclude: !NORMAL_CLEARING
 - Combine with AND: "61480" AND !NORMAL
 - **Pagination:** Remember filters persist across pages - use pagination to review large result sets
-

Active Calls Monitoring

The Active Calls view shows real-time information about ongoing calls through the system.

Access: Navigate to /calls in the Control Panel

Features

- **Real-time Status:** Live view of active call sessions
 - **Call Details:** View channel variables and call state information
 - **UUID Tracking:** Monitor both A-leg and B-leg call identifiers
-

IMS Conference Server

The IMS Conference Server provides multi-party conferencing capabilities compliant with 3GPP IMS standards (RFC 4579, RFC 4575, TS 24.147).

Access: Navigate to /conference in the Control Panel

Documentation: See [IMS Conference Server User Guide](#) for detailed documentation

Features

- **Real-time Monitoring:** Live view of active conferences and participants
- **Conference Statistics Dashboard:**
 - Active conference count
 - Total participants across all conferences
 - Video conference count
 - Locked conference count
 - Server configuration details (domain, MNC/MCC, max participants)
- **Conference List:** View all active conferences with:
 - Conference ID and SIP URI
 - Current participant count
 - Conference creator identity
- **Conference Details:** Click any conference to expand and view:
 - Full conference information (state, video status, locked status, recording status)
 - Complete participant list with roles and states

- Participant video status
- **Conference Control Actions:**
 - Lock/Unlock conferences to control access
 - Enable/Disable video for conferences
 - Real-time status updates with action feedback
- **Auto-Refresh:** Configurable auto-refresh (default: 5 seconds) for real-time monitoring

OmniTAS Console Management

All conference operations are also available through the OmniTAS console using the `ims_conference` command:

```
ims_conference list           # List all active conferences
ims_conference info <conf_id> # Show conference details
ims_conference stats          # Show server statistics
ims_conference lock <conf_id> # Lock a conference
ims_conference unlock <conf_id> # Unlock a conference
ims_conference video <conf_id> on|off # Control video
ims_conference record <conf_id> start|stop # Control recording
ims_conference add <conf_id> <sip_uri> # Add participant
ims_conference remove <conf_id> <uuid> # Remove participant
ims_conference destroy <conf_id> # Terminate conference
```

Use Cases

- **Operational Monitoring:** Real-time visibility into active conferences and resource usage
- **Capacity Management:** Monitor participant counts and video usage to manage bandwidth
- **Troubleshooting:** Diagnose conference access issues, participant connection problems
- **Conference Control:** Lock conferences for privacy, manage video to control bandwidth
- **Compliance:** Monitor and record conferences for regulatory compliance

3GPP Compliance

The conference server implements key 3GPP IMS conferencing specifications:

- **TS 24.147:** Conferencing using IM Core Network subsystem
 - **RFC 4579:** SIP Call Control - Conferencing for User Agents
 - **RFC 4575:** SIP Event Package for Conference State
 - **RFC 5239:** Framework for Centralized Conferencing
-

Gateway Status

Monitor the status and health of SIP gateways/trunks connected to the TAS.

Access: Navigate to /gw in the Control Panel

Features

- **Registration Status:** View gateway registration state
- **Call Statistics:** Track incoming/outgoing calls and failures
- **Ping Monitoring:** SIP OPTIONS ping times and reachability
- **Gateway Details:** Complete configuration and status information

Monitored Metrics

- SIP Registration status
 - Ping time (average SIP OPTIONS response time)
 - Uptime (seconds since profile restart)
 - Calls In / Calls Out
 - Failed Calls In / Failed Calls Out
 - Last ping time and frequency
-

Diameter Peer Status

Monitor Diameter peer connectivity for Sh and Ro interfaces.

Access: Navigate to /diameter in the Control Panel

Features

- **Peer Status:** Connection state for each configured peer
 - **Application Support:** View supported Diameter applications (Sh, Ro)
 - **Watchdog Status:** Diameter watchdog monitoring
-

System Logs Viewer

Real-time unified log viewer for both TAS Backend (Elixir) and TAS Call Processing (FreeSWITCH) logs.

Access: Navigate to /logs in the Control Panel

Features

- **Unified Log Stream:** View logs from both TAS Backend and Call Processing in one interface

- **Real-time Updates:** Live streaming of log messages as they occur (auto-refresh every 1 second)
- **Color-Coded Log Levels:**
 - **Console** - Console messages (purple/magenta)
 - **Alert/Critical** - Urgent issues requiring immediate attention (red)
 - **Error** - Error conditions (light red)
 - **Warning** - Warning messages (yellow)
 - **Notice** - Notable informational messages (cyan)
 - **Info** - General informational messages (blue)
 - **Debug** - Debug/verbose logging (gray)
- **Source Badges:**
 - **TAS Backend** - Elixir application logs (blue badge)
 - **TAS Call Processing** - FreeSWITCH logs (purple badge)
- **Left Border Indicators:** Color-coded left border matching log level for quick visual scanning
- **Multiple Filters:**
 - **Source Filter:** All Sources / TAS Backend / TAS Call Processing
 - **Level Filter:** All / Console / Alert / Critical / Error / Warning / Notice / Info / Debug
 - **Text Search:** Real-time keyword search across log messages
- **Pause/Resume:** Freeze log streaming to analyze specific entries without losing context
- **Clear Logs:** Remove all current log entries from display
- **Log Counter:** Shows filtered logs vs total logs (e.g., "Showing 150 of 500 logs")
- **Tail Behavior:** Maintains last 500 log entries for performance
- **Metadata Display:** File name and line number for source code references (when available)
- **Scrollable View:** Fixed-height container with auto-scroll for latest logs

How to Use

1. Basic Viewing:

- Page loads with latest 500 log entries from both sources
- Logs appear in real-time as they're generated
- Most recent logs appear at the top
- Auto-refreshes every 1 second

2. Filter by Source:

- Select from "**Source**" dropdown:
 - **All Sources** - Show both TAS Backend and Call Processing logs
 - **TAS Backend** - Only Elixir application logs
 - **TAS Call Processing** - Only FreeSWITCH/dialplan logs
- Filter applies immediately

3. Filter by Log Level:

- Select from "**Level**" dropdown:
 - **All** - Show all log levels
 - **Console** through **Debug** - Show only that specific level
- Useful for focusing on errors or debugging specific issues

4. Search for Keywords:

- Type in the "**Search logs...**" box
- Case-insensitive search across log messages
- Filters in real-time as you type
- Combines with source and level filters

5. Pause/Resume Stream:

- Click "**Pause**" button (orange) to freeze log updates
- "PAUSED" indicator appears in header
- Review specific log entries without new logs interrupting
- Click "**Resume**" button (green) to restart live streaming

6. Clear Logs:

- Click "**Clear**" button (red) to remove all displayed logs
- Clears both TAS Backend and Call Processing logs
- Fresh logs will appear as they're generated

7. Read Log Entries:

- **Timestamp:** Shows time in HH:MM:SS.milliseconds format
- **Source Badge:** Indicates TAS Backend (blue) or Call Processing (purple)
- **Log Level:** Color-coded level in brackets [ERROR], [INFO], etc.
- **File/Line:** Source code location (when available)
- **Message:** The actual log message content

Log Levels Explained

Level	Color	When Used	Example
Console	Purple	Console-specific messages	High-priority FreeSWITCH console output
Alert	Red	Immediate action required	System component failure
Critical	Red	Critical conditions	Database connection lost
Error	Light Red	Error conditions	Failed to process call, invalid configuration
Warning	Yellow	Warning conditions	Deprecated function used, retry attempt

Level	Color	When Used	Example
Notice	Cyan	Notable normal events	Configuration reloaded, service started
Info	Blue	Informational messages	Call connected, Diameter request sent
Debug	Gray	Debug-level messages	Function entry/exit, variable values

Use Cases

- **Real-time Troubleshooting:** Monitor logs during active call to debug issues
- **Error Investigation:** Filter by Error/Critical levels to find problems
- **Call Flow Analysis:** Search for Call-ID or phone number to trace call path
- **Performance Monitoring:** Watch for warnings and errors during load testing
- **Integration Debugging:** Filter TAS Backend to see Diameter/Sh/Ro messages
- **Dialplan Debugging:** Filter TAS Call Processing to see FreeSWITCH call routing
- **System Health Monitoring:** Keep logs open to watch for anomalies
- **Development & Testing:** Use Debug level to see verbose application behavior

Tips

- **Combine Filters:** Use Source + Level + Search together for precise filtering
 - Example: Source="TAS Backend" + Level="Error" + Search="Diameter" → Find Diameter errors
- **Pause Before Searching:** Pause stream before typing search query to avoid logs scrolling
- **Use Debug Wisely:** Debug level is verbose - filter to specific source to reduce noise
- **Color Scanning:** Quickly scan left borders - red borders indicate problems
- **Source Badges:** Blue badges (Backend) for app logic, Purple badges (Call Processing) for calls
- **Timestamp Precision:** Millisecond timestamps help correlate events across systems
- **File References:** Click/note file:line references to jump to source code
- **Clear Regularly:** Clear logs when switching investigation contexts for clarity
- **Search for UUIDs:** Search for Call-ID/UUID to follow a specific call through entire system
- **Emergency Search:** Search "911" or "emergency" to quickly find emergency call handling

Technical Details

- **Log Limit:** Maximum 500 logs displayed (oldest discarded when limit reached)
 - **Refresh Rate:** Auto-refresh every 1000ms (1 second)
 - **Search:** Case-insensitive substring matching on message field only
 - **Empty Filtering:** Automatically filters out empty/placeholder log messages
 - **Source Detection:** Logs tagged with `:elixir` or `:freeswitch` source
 - **Sorting:** Logs sorted by timestamp descending (newest first)
 - **PubSub:** Elixir logs delivered via Phoenix PubSub for real-time updates
 - **FreeSWITCH Logs:** Collected via Event Socket Interface (ESI) log listener
-

Cell Tower Database

Manage and query the OpenCellID cell tower location database for emergency services and location-based features.

Access: Navigate to `/cell_towers` in the Control Panel

Features

- **Database Statistics:** View total records, coverage by country/network
- **Search & Query:**
 - Search by MCC (Mobile Country Code)
 - Search by MNC (Mobile Network Code)
 - Search by radio type (GSM, UMTS, LTE)
 - Search by location string
- **Database Management:**
 - Import cell tower data
 - Re-download latest dataset from OpenCellID
 - View import status and progress
- **Location Resolution:** Resolve cell IDs to geographic coordinates

Use Cases

- Emergency call location determination
- Subscriber location tracking (with consent)
- Network coverage analysis
- Troubleshooting roaming location issues
- Cell tower database maintenance

Data Source

Cell tower data is sourced from OpenCellID (<https://opencellid.org/>), a collaborative community project to create a free database of cell tower locations worldwide.

Call Simulator

Interactive call simulation tool for testing dialplan logic without making real calls.

Access: Navigate to /simulator in the Control Panel

Detailed Documentation: See [doc/HLR_AND_CALL_SIMULATOR.md](#)

Features

- **Simulate Call Types:** Test MO, MT, and Emergency calls
- **Configurable Parameters:**
 - Source and destination numbers
 - Source IP address (to simulate SBC/CSCF)
 - Force specific call disposition
 - Skip OCS authorization for faster testing
- **Comprehensive Results:**
 - Complete dialplan variable output
 - Sh/HLR lookup results
 - OCS authorization result
 - SS7 MAP query results (if applicable)
 - Generated dialplan XML
- **Step-by-Step Processing:** View each stage of call processing

Use Cases

- Test dialplan changes before deployment
- Verify subscriber provisioning
- Debug call routing issues
- Train staff on call flow
- Validate OCS/HLR integration
- Test emergency call handling

HLR/MAP Testing

Test SS7 MAP operations including Send Routing Info (SRI) and Provide Roaming Number (PRN) queries.

Access: Navigate to /hlr in the Control Panel

Detailed Documentation: See [doc/HLR_AND_CALL_SIMULATOR.md](#)

Features

- **SRI Query:** Test Send Routing Info for call routing

- **PRN Query:** Test Provide Roaming Number for roaming subscribers
- **Real Results:** Actual queries to configured MAP gateway
- **Response Display:** View MSRN, MSC address, and forwarding status
- **Error Handling:** Clear display of MAP errors and timeouts

Use Cases

- Verify HLR connectivity
- Test roaming number allocation
- Debug call routing to roaming subscribers
- Validate MAP gateway configuration
- Troubleshoot call forwarding issues

OCS Testing

Test Diameter Ro (Online Charging) Credit-Control-Request (CCR) operations directly against your OCS.

Access: Navigate to /ocs_test in the Control Panel

Features

- **Flexible CCR Types:** Send INITIAL, UPDATE, TERMINATION, or EVENT requests
- **Session Simulation:** Reuse the same Call ID to simulate a complete session lifecycle
- **Event Type Selection:** Test both SMS (event-based) and Call (session-based) charging
- **Direction Control:** Test both outgoing (MO) and incoming (MT) scenarios
- **Optional Parameters:** Specify Destination-Host and Username for advanced testing
- **Real-time Results:** View complete CCA (Credit-Control-Answer) responses

How to Use

1. Enter Test Parameters:

- **Called MSISDN:** The destination number (e.g., 61400123456)
- **Calling MSISDN:** The originating number (e.g., 61400987654)
- **Event Type:** Choose sms or call
 - SMS defaults to EVENT_REQUEST (type 4)
 - Call defaults to INITIAL_REQUEST (type 1)
- **Direction:** out for MO or in for MT

2. Configure CCR Type:

- **Request-Type:** Select the CCR type:
 - 1 – INITIAL_REQUEST - Start a new session
 - 2 – UPDATE_REQUEST - Mid-session re-authorization
 - 3 – TERMINATION_REQUEST - End session and report usage
 - 4 – EVENT_REQUEST - One-time event (SMS, immediate event)
- **Request-Number:** Starts at 1, increment for each request in the same session

3. Session Testing:

- **Call ID:** Auto-generated unique identifier for correlation
- Click "**New ID**" to generate a fresh Call ID for a new test session
- **Keep the same Call ID** to simulate a complete session:
 - First request: INITIAL_REQUEST (type 1, number 1)
 - Mid-session: UPDATE_REQUEST (type 2, number 2, 3, 4...)
 - Final request: TERMINATION_REQUEST (type 3, number N+1)

4. Advanced Options:

- **Destination-Host:** Target a specific OCS node (optional)
- **Username:** Override the subscriber identifier (optional)

5. Run and Review:

- Click "**Run CCR**" to send the request
- View the complete CCA response with all AVPs
- Check result code, granted units, and validity time
- Last run timestamp shown in top-right corner

Use Cases

- **OCS Connectivity Testing:** Verify Diameter Ro connection and authentication
- **Credit Control Logic:** Test credit allocation, consumption, and exhaustion scenarios
- **Session Flow Testing:** Simulate complete call lifecycle (INITIAL → UPDATE → TERMINATION)
- **Rating Validation:** Verify correct charging rates for different number ranges
- **Failover Testing:** Test OCS redundancy by targeting specific Destination-Host
- **Integration Debugging:** Troubleshoot OCS integration issues with detailed AVP inspection
- **Load Testing Preparation:** Validate OCS behavior before load testing
- **Emergency Number Bypass:** Verify that emergency numbers bypass charging correctly

Tips

- Use the same Call ID with incrementing Request-Numbers to test session continuity
 - Monitor OCS logs simultaneously to correlate test requests
 - Test UPDATE requests to verify mid-session re-authorization logic
 - Verify that TERMINATION requests properly close sessions and prevent leaks
 - Test credit exhaustion by sending UPDATE requests after consuming granted units
-

Sh Interface Testing

Test Diameter Sh User-Data-Request (UDR) operations to retrieve subscriber profile data from the HSS.

Access: Navigate to /sh_test in the Control Panel

Features

- **Multiple Data References:** Query over 20 different subscriber data types
- **Real HSS Queries:** Live Diameter Sh requests to your configured HSS
- **Complete Response Display:** View full XML subscriber data and AVPs
- **Session Tracking:** Shows HSS hostname, realm, and session ID
- **Error Handling:** Clear display of Diameter result codes and error conditions

How to Use

1. Enter Public Identity:

- **Public Identity:** The subscriber's IMS Public Identity
- Format: sip:61400123456@ims.mncXXX.mccXXX.3gppnetwork.org
- Can also use tel:+61400123456 format

2. Select Data Reference: Choose the type of subscriber data to retrieve:

- **RepositoryData (0):** Complete subscriber profile
- **IMSPublicIdentity (10):** List of public identities
- **IMSUserState (11):** Registration state
- **S-CSCFName (12):** Assigned S-CSCF
- **InitialFilterCriteria (13):** iFC triggers for application servers
- **LocationInformation (14):** Current location
- **ChargingInformation (16):** P-Charging addresses
- **MSISDN (17):** Phone number
- **IMSI (32):** International Mobile Subscriber Identity

- **IMSPrivateUserIdentity (33):** Private user identity
- And many more...

3. Run and Review:

- Click "**Fetch SH Data**" to send the UDR request
- View the complete User-Data-Answer (UDA) response
- Check subscriber profile XML, service data, and iFC rules
- Session metadata shows which HSS responded

Use Cases

- **Subscriber Verification:** Confirm subscriber is provisioned in HSS
- **iFC Debugging:** Review Initial Filter Criteria and trigger points
- **Registration Troubleshooting:** Check user state and S-CSCF assignment
- **Charging Configuration:** Verify P-Charging-Function-Addresses
- **HSS Connectivity Testing:** Validate Diameter Sh connection
- **Profile Validation:** Ensure correct service profile is assigned
- **Integration Testing:** Test HSS integration after provisioning changes
- **Roaming Analysis:** Check location information and serving network

Tips

- Use **IMSPublicIdentity (10)** to see all aliases for a subscriber
- Use **RepositoryData (0)** to get the complete subscriber profile in one query
- Check **IMSUserState (11)** to verify if a subscriber is registered
- **InitialFilterCriteria (13)** shows which application servers will be triggered
- The session ID can be used to correlate queries in HSS logs
- Error responses include Diameter result codes (e.g., 5001 = User Unknown)

Number Translation Testing

Test number translation rules and formatting without making actual calls.

Access: Navigate to /translate in the Control Panel

Features

- **Real-time Translation:** Auto-translates as you type
- **Country Code Support:** Test different country code contexts
- **Disposition-Aware:** Apply different rules based on call disposition
- **Live Results:** Immediate feedback with translated number
- **Debug Information:** View raw return values for troubleshooting

How to Use

1. Configure Parameters:

- **Country Code:** The dialing context (e.g., AU, US, NZ)
 - Defaults to the value configured in `Tas.Config.number_translate()[:country_code]`
 - Accepts formats: AU, :AU, au
- **Phone Number:** The number to translate
 - Examples: +61400111222, 0400111222, 61400111222
- **Disposition:** (Optional) Call context for conditional rules
 - Examples: originate, route, emergency

2. Test Translation:

- Enter values in the form
- Translation runs automatically as you type
- Or click "**Translate**" to manually trigger
- View the translated result immediately

3. Review Results:

- **Translated:** Shows the formatted output number
- **Error:** Displays validation errors or translation failures
- **Raw return (debug):** Shows the complete Elixir tuple for debugging

Use Cases

- **Dialplan Development:** Test number formatting rules before deployment
- **Format Validation:** Verify E.164 conversion is working correctly
- **Country Code Testing:** Ensure correct handling of international prefixes
- **Emergency Number Detection:** Verify emergency numbers are properly identified
- **Short Code Handling:** Test special service codes (voicemail, etc.)
- **Trunk Preparation:** Format numbers correctly for SIP trunk requirements
- **Disposition Logic:** Test different rules for MO vs MT scenarios
- **Debugging Translation Issues:** Troubleshoot why specific numbers fail routing

Tips

- Test both local format (0400111222) and international format (+61400111222)
- Verify emergency numbers (000, 112) are detected correctly
- Use disposition field to test different call scenarios (MO, MT, emergency)
- Check that short codes and internal numbers are handled appropriately
- The debug output shows the raw return value - useful for investigating

issues

- Test edge cases like leading zeros, international prefixes, and special characters

Voicemail Management

Manage and listen to voicemail messages stored in the system.

Access: Navigate to /voicemail in the Control Panel

Features

- **Complete Voicemail List:** View all voicemail messages across all mailboxes
- **In-Browser Playback:** Listen to voicemail recordings directly in the web interface
- **Message Details:** View username, UUID, timestamps, file paths, and metadata
- **Delete Functionality:** Remove individual voicemail messages
- **Auto-Refresh:** Refresh button to reload latest voicemail data
- **Dynamic Columns:** Automatically displays all available database fields

How to Use

1. View Voicemail List:

- Page loads automatically with all voicemail records
- Table shows all fields from the voicemail database
- Timestamps are automatically formatted from epoch values
- File paths are shortened for readability

2. Listen to Messages:

- Click "**► Play**" button next to any voicemail
- Audio player appears with controls (play, pause, seek, volume)
- Supports WAV, MP3, and OGG formats
- Click "**Stop**" to close the audio player

3. Delete Messages:

- Click "**Delete**" button to remove a voicemail
- Confirmation prompt prevents accidental deletion
- Page automatically refreshes after successful deletion

4. Refresh Data:

- Click "**Refresh**" button in top-right to reload voicemail list

- Useful after new voicemails are left

Message Details Displayed

The table dynamically shows all available fields, typically including:

- **Username:** Mailbox owner
- **UUID:** Unique message identifier
- **Created Epoch:** When the message was left (auto-formatted to readable date/time)
- **Read Epoch:** When the message was accessed (if applicable)
- **File Path:** Location of the audio file
- Additional metadata from the voicemail database

Use Cases

- **Subscriber Support:** Listen to voicemail messages for troubleshooting
- **Testing Voicemail Delivery:** Verify voicemails are being stored correctly
- **Message Management:** Clean up old or test voicemail messages
- **Troubleshooting Recording Issues:** Check file paths and verify audio files exist
- **Mailbox Maintenance:** Monitor voicemail storage and usage
- **Quality Assurance:** Review recorded messages for audio quality

Tips

- File paths are automatically shortened to show only the relevant portion
- Epoch timestamps are automatically converted to human-readable format
- Empty voicemail database shows "No voicemail records found"
- Audio playback uses HTML5 audio element - supported in all modern browsers
- Delete confirmation prevents accidental removal of important messages

TTS Prompt Management

Manage Text-to-Speech (TTS) generated audio prompts used throughout the system.

Access: Navigate to /prompts in the Control Panel

Features

- **Prompt Settings Display:** View current TTS voice, response format, and instructions
- **Recording Status:** See which prompts exist and which are missing
- **File Details:** View file size, modification time, and path for each prompt

- **In-Browser Playback:** Listen to prompts directly in the web interface
- **Generate Missing:** Automatically create all missing prompt files
- **Re-record Individual:** Regenerate a specific prompt with updated settings
- **Re-record All:** Regenerate all prompts (useful after changing voice or settings)

How to Use

1. Review Prompt Settings:

- **Voice:** TTS voice being used (e.g., alloy, nova, shimmer)
- **Response Format:** Audio format (e.g., wav, mp3, opus)
- **Instructions:** Special instructions passed to TTS engine

2. Check Recording Status:

- **Text:** The prompt text to be spoken
- **Relative Path:** Where the audio file is stored
- **Exists:** Green "Yes" if file exists, Yellow "No" if missing
- **Size:** File size in bytes/KiB/MiB
- **Modified:** Last modification timestamp

3. Generate Prompts:

- **Generate Missing:** Creates only prompts that don't exist yet
 - Useful for initial setup or after adding new prompts
- **Re-record All:** Regenerates all prompts regardless of existence
 - Useful after changing voice, format, or instructions
 - Use with caution as it regenerates everything

4. Manage Individual Prompts:

- ► **Play:** Listen to the prompt (only enabled if file exists)
- ⬮ **Re-record:** Regenerate just this one prompt
 - Useful if one prompt sounds incorrect
 - Uses current voice and settings

5. Listen to Prompts:

- Click "► **Play**" to hear the prompt
- Audio player appears at bottom with full controls
- Click "**Stop**" to close the player

Prompt Configuration

Prompts are configured in your application config:

```
config :tas, :prompts,
```

```
voice: "nova",
response_format: "wav",
instructions: "Speak clearly and professionally.",
recordings: [
  {%path: "/sounds/en/us/callie/voicemail/vm-enter_id.wav",
   text: "Please enter your mailbox ID followed by pound"},
  # ... more prompts
]
```

Use Cases

- **Initial Setup:** Generate all prompts after system installation
- **Voice Changes:** Re-record all prompts with a different TTS voice
- **Quality Improvement:** Fix individual prompts that don't sound right
- **Format Updates:** Regenerate prompts in different audio format (wav → mp3)
- **Text Updates:** Re-record after changing prompt text in config
- **Testing TTS:** Preview how prompts will sound before deployment
- **Troubleshooting Playback:** Verify prompt files exist and are accessible
- **Storage Management:** Check file sizes and manage disk usage

Tips

- Use **"Generate Missing"** for initial setup - it won't overwrite existing prompts
- Use **"Re-record All"** after changing voice or format in config
- Individual **"Re-record"** is useful for iterating on specific prompts
- Listen to prompts before deployment to ensure quality
- Larger response formats (wav) have better quality but use more disk space
- The instructions field can guide TTS engine for tone and pacing
- Re-recording can take time if you have many prompts - be patient
- Prompts are stored in FreeSWITCH sounds directory for easy access

Dialplan XML Templates

View and inspect FreeSWITCH dialplan XML templates used for call routing.

Access: Navigate to /routing in the Control Panel

Features

- **Template List:** View all XML dialplan templates from priv/templates/ directory
- **File Details:** See filename and last modified timestamp for each template
- **Syntax Highlighting:** Color-coded XML display for easy reading
 - Tags in teal

- Attributes in light blue
- Values in orange/tan
- Comments in green
- **Expandable View:** Click any template to view its full XML content
- **Read-Only View:** Safe inspection without risk of accidental modification
- **Scrollable Content:** Large templates scroll within fixed-height container (max 600px)

How to Use

1. View Template List:

- Page loads with all .xml files from the templates directory
- Sorted alphabetically by filename
- Shows modification timestamp for each file

2. Inspect Template:

- Click any row to expand and view the XML content
- Template displays with syntax highlighting
- Click again to collapse

3. Read XML Content:

- **Tags** (teal): XML element names like <extension>, <condition>
- **Attributes** (light blue): Attribute names like name=, field=
- **Values** (orange): Attribute values like "public", "destination_number"
- **Comments** (green): XML comments <!-- ... -->

Use Cases

- **Review Dialplan Logic:** Inspect routing rules and call flow templates
- **Troubleshoot Call Routing:** Understand which templates are used for different call types
- **Verify Template Syntax:** Check XML structure before deployment
- **Training & Documentation:** Share template contents with team members
- **Change Auditing:** Compare modification timestamps to track updates
- **Template Development:** Reference existing templates when creating new ones

Tips

- Templates are loaded from priv/templates/ within the TAS application
- Only .xml files are displayed
- Templates are read-only through the web interface
- Modification timestamps help identify recent changes
- Use this view to verify templates match your dialplan expectations

- Syntax highlighting makes complex XML easier to parse visually
- Combine with /logs view to correlate routing behavior with templates

Technical Details

- **Location:** Templates stored in priv/templates/ directory
 - **Format:** FreeSWITCH XML dialplan format
 - **File Extension:** Only .xml files listed
 - **Sorting:** Alphabetical by filename
 - **Syntax Highlighting:** Client-side colorization using regex patterns
 - **Max Display Height:** 600px with scroll for large files
-

ESL Command Runner

Execute FreeSWITCH Event Socket Layer (ESL) commands directly from the web interface.

Access: Navigate to /command in the Control Panel

Features

- **Command Execution:** Run any ESL/FreeSWITCH API command
- **Live Output:** See command results in real-time
- **Command History:** Recent commands dropdown (last 10 commands)
- **Auto-Complete Ready:** Monospace input for precise command entry
- **Error Handling:** Clear display of command errors and exceptions
- **No Auto-Execute:** Selecting history fills input but requires explicit "Run" click

How to Use

1. Enter Command:

- Type ESL command in the input box
- Examples:
 - status - Show FreeSWITCH status
 - show channels - List active calls
 - uuid_dump <uuid> - Dump all variables for a call
 - sofia status - Show SIP profile status
 - reloadxml - Reload XML dialplan
 - version - Show FreeSWITCH version

2. Run Command:

- Click **"Run"** button to execute
- Button shows "Running..." while executing

- Cannot run multiple commands simultaneously

3. View Output:

- Results appear in the "Output" section below
- Successful commands show raw response
- Errors prefixed with "ERROR:"
- Output is scrollable with max height of 600px
- Monospace font for aligned data

4. Use Command History:

- Recent commands appear in dropdown after first execution
- Select from "Recent:" dropdown to fill input field
- History maintains last 10 unique commands
- Most recent command at top
- Selecting history does NOT auto-execute (safety feature)

Common Commands

Command	Description	Example Output
status	System status and uptime	FreeSWITCH running info
show channels	List all active calls	Channel list or "0 total"
show calls	Summary of active calls	Call count summary
uuid_dump <uuid>	All variables for a call	Complete variable dump
uuid_kill <uuid>	Hangup specific call	" +OK" or error
sofia status	SIP profile status	Profile list and states
sofia status profile <name>	Specific profile details	Registration count, etc
reloadxml	Reload dialplan XML	" +OK" confirmation
version	FreeSWITCH version info	Version string
global_getvar <var>	Get global variable	Variable value
api help	List available commands	Command reference

Use Cases

- **Call Debugging:** Get detailed info about active call with `uuid_dump`
 - **System Status:** Check FreeSWITCH health with `status` and `show calls`
 - **SIP Troubleshooting:** Inspect SIP profiles with `sofia status`
 - **Dialplan Reload:** Apply config changes with `reloadxml`
 - **Emergency Actions:** Kill stuck calls with `uuid_kill`
 - **Variable Inspection:** Check global or channel variables
-

Troubleshooting

Subscribers Not Showing

- Verify OmniTAS is running
- Check Sofia profile is active: `sofia status profile internal`
- Verify database path in configuration matches actual database location

CDR Records Not Appearing

- Confirm OmniTAS CDR module is loaded
- Check CDR database exists at configured path
- Verify CDR module configuration in OmniTAS

Performance Considerations

- Large CDR databases (>1M records) may require additional indexing for optimal performance
- Consider archiving old CDR records periodically
- Subscriber registration queries are typically fast as the registration database is small

Configuration

Access Control

The Control Panel should be deployed behind appropriate access controls (firewall, VPN, authentication) as it provides visibility into subscriber activity and call records.

TTS Prompt Configuration

◇ [Back to Main Documentation](#)

Configuration for Text-to-Speech (TTS) prompts using OpenAI's TTS engine.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - TTS prompts configuration (voice, instructions, recordings)
- ◇ [Operations Guide](#) - TTS prompt management in Control Panel

Integration & Usage

- ◇ [Dialplan Configuration](#) - Using prompts in dialplan with playback application
- ◇ [Voicemail](#) - Voicemail greeting and instruction prompts
- ⚙ [Supplementary Services](#) - Service announcement prompts
- ◇ [Online Charging](#) - Out-of-credit prompts

Prompt Configuration

You can define prompts in the config that are then generated with Text to Speech.

You can then use these in your dialplan with the playback commands.

For the prompts we can define "instructions" for tone, language, accent, etc, and pick the voice. The TTS engine uses OpenAI's text to speech engine, which you can test from openai.fm

```
config :tas,  
...  
prompts: %{  
  voice: "alloy",  
  instructions: "Speak with a prim, British accent.",  
  response_format: "wav",  
  recordings: [  
    %{  
      text:
```



```
        "You do not have sufficient credit to make that call,  
please topup your service and then try again ",  
        path: "/sounds/en/us/callie/misc/8000/out_of_credit.wav"  
    },  
    %{  
        text: "The destination you have called is unable to be  
reached",  
        path: "/sounds/en/us/callie/misc/8000/  
unable_to_be_reached.wav"  
    },  
    %{  
        text: "Your call is being transferred to emergency services",  
        path: "/sounds/en/us/callie/misc/8000/  
emergency_services_transfer.wav"  
    }  
]  
}
```



Sh Interface (Subscriber Data Retrieval)

◊ [Back to Main Documentation](#)

The Sh interface provides access to subscriber profile data from the HSS/Repository via Diameter.

Related Documentation

Core Documentation

- ◊ [Main README](#) - Overview and quick start
- ◊ [Configuration Guide](#) - Diameter peer configuration
- ◊ [Operations Guide](#) - Sh interface testing in Control Panel

Call Processing Integration

- ◊ [Dialplan Configuration](#) - Using Sh data in dialplan variables
- ⚙ [Supplementary Services](#) - MMTel-Config for call forwarding
- ◊ [SS7 MAP](#) - HLR data vs Sh data priority

Related Interfaces

- ◊ [Online Charging](#) - Ro interface (also uses Diameter)
- ◊ [Number Translation](#) - Number normalization before Sh lookup

Monitoring

- ◊ [Metrics Reference](#) - Sh interface metrics and monitoring

Sh Interface (Subscriber Data Retrieval)

The Sh interface is used to retrieve subscriber profile data from the HSS/Repository before processing calls. This data includes subscriber identities, services, and MMTel configuration.

What is the Sh Interface?

The Sh interface is a 3GPP-standardized Diameter interface between the TAS and HSS/Repository (Repo). It provides real-time access to:

- IMS subscriber identities (IMPI/IMPU)
- Call forwarding settings (MMTel-Config)
- Subscriber service authorization
- S-CSCF assignment

When Sh Lookups Occur

Sh Lookups Happen On:

- **MT Calls:** Lookup called party (destination subscriber)
- **MO Calls:** Lookup calling party (source subscriber)
- **Emergency Calls:** Lookup calling party (for location/identity)

Data Retrieved from Sh Interface

The TAS queries for Sh-User-Data which returns an XML document containing:

1. IMS Identities:

- **IMPI (Private Identity):** username@domain - used for authentication
 - Format: {IMSI}@ims.mnc{MNC}.mcc{MCC}.3gppnetwork.org
 - Example: 505014001234567@ims.mnc001.mcc505.3gppnetwork.org
- **IMPU (Public Identity):** sip:+number@domain - used for routing
 - Format: sip:+{MSISDN}@ims.mnc{MNC}.mcc{MCC}.3gppnetwork.org
 - Example: sip:+61403123456@ims.mnc001.mcc505.3gppnetwork.org

2. S-CSCF Assignment:

- S-CSCF server name and domain where subscriber is registered
- Used for routing on-net calls back to IMS core

3. MMTel Services (Multimedia Telephony Configuration):

- **Call Forward All (CFA):** Unconditional forwarding to another number
- **Call Forward Busy (CFB):** Forward when subscriber is busy
- **Call Forward No Reply (CFNRy):** Forward after timeout (includes timer value)
- **Call Forward Not Reachable (CFNRc):** Forward when subscriber is offline/unregistered

What is MMTel-Config?

MMTel-Config is the subscriber's Multimedia Telephony service configuration stored in the HSS/Repository. It contains:

```
<MMTelSS>
  <CDIV>
    <SS-ActivationState>active</SS-ActivationState>
    <Ruleset>
      <Rule>
        <RuleCondition>communication-diverted</RuleCondition>
        <ForwardTo>+61403555123</ForwardTo>
        <NotificationType>notify</NotificationType>
      </Rule>
    </Ruleset>
  </CDIV>
</MMTelSS>
```

Common MMTel Services:

- **CDIV (Communication Diversion):** Call forwarding rules
- **OIP (Originating Identity Presentation):** Caller ID presentation rules
- **TIP (Terminating Identity Presentation):** Called party number rules

Dialplan Variables Set from Sh Data

After a successful Sh lookup, these variables are populated:

Variable	Source	Example Value	Description
ims_private_identity	IMPI	505014001234567@ims.domain	Private user identity for authentication
ims_public_identity	IMPU	sip:+61403123456@ims.domain	Public user identity for routing
msisdn	IMPU (parsed)	61403123456	Subscriber number (+ stripped)
imsi	IMPI (parsed)	505014001234567	IMSI from private identity
ims_domain	IMPI/	ims.mnc001.mcc505.3gppnetwork.org	IMS domain

Variable	Source	Example Value	Description
	IMPU		
scscf_address	S-CSCF name	scscf01.ims.domain or "none"	S-CSCF server address
scscf_domain	S-CSCF domain	ims.domain or "none"	S-CSCF domain
call_forward_all_destination	MMTel CDIV	61403555123 or "none"	CFA destination number
call_forward_not_reachable_destination	MMTel CDIV	2222 or config default	CFNRc destination (voicemail)
no_reply_timer	MMTel CDIV	30 or config default	Timeout before CFNRy activates

Priority: Sh Data vs Configuration Defaults

The TAS uses this priority for call forwarding data:

1. **MMTel-Config from Sh** (highest priority - subscriber-specific settings)
2. **HLR Data from SS7 MAP** (overrides Sh for MT calls if roaming/forwarding active)
3. **Configuration Defaults** (lowest priority - used when no Sh data available)

Example:

```
# Configuration defaults (used only if Sh returns no MMTel-Config)
config :tas,
  call_forward_not_reachable_destination: "2222", # Voicemail
  default_no_reply_timer: 30
```

What Happens When Sh Lookup Fails

Failure Scenarios:

1. **Subscriber Not Provisioned in HSS:**
 - Sh returns "User Unknown" error
 - hangup_case variable set to "UNALLOCATED_NUMBER"
 - Call rejected with appropriate SIP response
2. **HSS Unreachable / Timeout:**
 - Sh request times out (default: 5000ms)
 - Error logged and metric recorded
 - Call may be rejected or routed with defaults (deployment-specific)
3. **No MMTel-Config in Response:**
 - Subscriber exists but has no call forwarding configured
 - Configuration defaults are used for call_forward_not_reachable_destination and no_reply_timer
 - Call proceeds normally with default values

Monitoring Sh Interface

Key Metrics:

```
# Sh lookup success rate
rate(subscriber_data_lookups_total{result="success"}[5m]) /
rate(subscriber_data_lookups_total[5m]) * 100
```

```
# Sh lookup latency (P95)
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m]))

# Sh error rate
rate(subscriber_data_lookups_total{result="error"}[5m])
```

Alert Thresholds:

- P95 latency > 100ms: Slow HSS responses
- Error rate > 5%: HSS connectivity issues
- Error rate > 20%: Critical HSS failure

Troubleshooting:

1. Check Diameter peer status in Web UI (/diameter)
2. Test Sh lookup in Web UI (/sh_test) with known subscriber
3. Review logs for "Subscriber Data" errors
4. Verify HSS/Repository is reachable from TAS
5. Check subscriber_data_lookups_total metric for patterns

Testing Sh Interface

Use the Web UI Sh Test tool (/sh_test):

1. Navigate to /sh_test in Control Panel
2. Enter subscriber MSISDN (e.g., +61403123456)
3. Click "Query Sh"
4. Review returned data:
 - IMPI/IMPU identities
 - S-CSCF assignment
 - MMTel services
 - Call forwarding configuration

Common Test Scenarios:

- Verify newly provisioned subscribers are in HSS
- Check call forwarding settings for specific subscriber
- Validate S-CSCF assignment after IMS registration
- Test HSS connectivity and response times



SS7 MAP / Gateway-MSC Configuration

◇ [Back to Main Documentation](#)

Configuration for HLR queries to retrieve MSRN (roaming numbers) and call forwarding information via SS7 MAP.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - SS7 MAP configuration (ss7_map parameters)
- ◇ [Operations Guide](#) - HLR/MAP testing in Control Panel

Call Processing Integration

- ◇ [Dialplan Configuration](#) - Using MSRN and forwarded_to_number in dialplan routing
- ⚙ [Supplementary Services](#) - HLR-based call forwarding (alternative to Sh/MMTel)
- ◇ [Sh Interface](#) - Sh vs MAP data priority
- ◇ [Number Translation](#) - Number format for HLR queries

Testing & Monitoring

- ◇ [HLR & Call Simulator](#) - Testing HLR/MAP integration
- ◇ [Metrics Reference](#) - HLR/MAP query metrics

Gateway-MSC Configuration

The TAS can query an HLR to retrieve the roaming number (MSRN) or MSC when a subscriber is roaming on 2G/3G networks, and can also retrieve call forwarding information.

This will set the msrn or forwarded_to_number dialplan variables which can then be used to route the call appropriately.

Configuration Parameters:

- **enabled** - Enable/disable SS7 MAP functionality
- **http_map_server_url_base** - Base URL of the MAP gateway HTTP API
- **gmsc** - Gateway MSC address used for SRI/PRN queries
- **timeout_ms** - HTTP timeout for MAP operations in milliseconds (default: 5000)

```
config :tas,
  ...
  ss7_map: %{
    enabled: true,
    http_map_server_url_base: "http://10.5.1.216:8080",
    gmsc: "55512411506",
    timeout_ms: 5000 # Optional, defaults to 5000ms
  },
```

Functionality: The TAS performs SRI (Send Routing Information) and handles routing based on the following priority:

1. **Call Forwarding Active** - If the SRI response contains a forwarded number, it is treated as an MSRN (no PRN is performed). The forwarded number is set in the msrn variable and used for routing.
2. **Roaming (2G/3G)** - If the subscriber is roaming (VLR present) and no call forwarding is active, performs PRN (Provide Roaming Number) to get the MSRN for routing to the V-MSC
3. **Normal** - If neither forwarding nor roaming applies, the call proceeds with standard routing

The msrn and tas_destination_number dialplan variables are set appropriately for routing (either from PRN or from the forwarded number)



Supplementary Services

◇ [Back to Main Documentation](#)

Configuration and implementation of call forwarding, CLI blocking, and emergency calling services.

Related Documentation

Core Documentation

- ◇ [Main README](#) - Overview and quick start
- ◇ [Configuration Guide](#) - Service configuration parameters (emergency codes, CLI blocking, default call forward)
- ◇ [Operations Guide](#) - Testing supplementary services

Call Processing & Data Sources

- ◇ [Dialplan Configuration](#) - Implementing services in dialplan logic
- ◇ [Sh Interface](#) - MMTel-Config for call forwarding settings
- ◇ [SS7 MAP](#) - HLR-based call forwarding (alternative to Sh)
- ◇ [Number Translation](#) - CLI blocking prefix handling

Service Interactions

- ◇ [Online Charging](#) - Emergency calls bypass OCS
- ◇ [Voicemail](#) - Call forward on busy/no-answer routes to voicemail

Monitoring

- ◇ [Metrics Reference](#) - Call forwarding and service metrics
- ◇ [Dialplan Metrics](#) - Service usage metrics

Supplementary Services (Call Forward / Blocked CLI / Emergency Codes)

Config for blocked CLI prefixes, emergency call codes, and default Call Forward data (Call Forward / No Reply data is only used when no MMTel-Config data is returned from the Repository on Sh).

```
config :tas,
...
blocked_cli_prefix: ["*67"],
call_forward_not_reachable_destination: "2222",
default_no_reply_timer: 30,
emergency_call_codes: ["911", "912", "913", "sos"],
...
```


Configuration Parameters:

- **blocked_cli_prefix** (list of strings): Prefixes that trigger CLI (Calling Line ID) withholding
 - Example: ["*67"] - dialing *67 before a number hides caller ID
 - Used in dialplan to set cli_withheld variable
- **call_forward_not_reachable_destination** (string): Default destination for Call Forward Not Reachable (CFNRc)
 - Only used when no MMTel-Config is returned from Sh interface
 - Example: "2222" - forwards to voicemail
- **default_no_reply_timer** (integer): Default timeout in seconds before CFNRc activates
 - Only used when no MMTel-Config is returned from Sh interface
 - Example: 30 - rings for 30 seconds before forwarding
- **emergency_call_codes** (list of strings): Emergency service numbers for your jurisdiction
 - Checked during call authorization to detect emergency calls
 - SIP emergency URNs (e.g., <urn:service:sos>) are always checked in addition to these codes
 - Common examples: ["911", "112", "000", "999", "sos"]
 - See [Emergency Calling](#) section for detailed usage

How Caller ID Blocking Works

The TAS supports two methods for blocking caller ID (CLI withholding), both of which set the cli_withheld dialplan variable to "true":

Method 1: Prefix-Based Blocking

When a subscriber dials a destination number prefixed with a code from blocked_cli_prefix:

1. The number translation module detects the prefix (e.g., caller dials *67555123456)
2. The prefix is **stripped** from the destination number (becomes 555123456)
3. The cli_withheld variable is set to "true"
4. The dialplan can then use this variable to hide the caller's identity

Example configuration:

```
blocked_cli_prefix: ["*67"]           # US-style blocking
blocked_cli_prefix: ["#31#"]          # European GSM-style blocking
blocked_cli_prefix: ["*67", "#31#"]  # Support both
```

Method 2: SIP From Header Detection

When the UE/handset requests privacy via SIP headers:

1. The TAS checks if the SIP From header display name contains "anonymous" (case-

- insensitive)
- 2. If found, the cli_withheld variable is set to "true"
- 3. This honors the subscriber's privacy request set at the device level

Implementing CLI Blocking in Dialplan

The TAS sets the cli_withheld variable, but your **dialplan XML must implement the actual blocking behavior:**

```
<extension name="CLI-Privacy" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <!-- Hide caller identity -->
    <action application="set" data="effective_caller_id_name=anonymous"/>
    <action application="set" data="effective_caller_id_number=anonymous"/>
    <action application="set" data="origination_privacy=hide_number"/>

    <!-- Optionally set P-Asserted-Identity privacy -->
    <action application="set" data="sip_h_Privacy=id"/>
  </condition>
</extension>
```

Variables Set by TAS for CLI Blocking:

The TAS sets these variables before dialplan execution:

Variable	Type	Values	Description
cli_withheld	string	"true" or "false"	Indicates if CLI blocking was requested via prefix OR From header
tas_destination_number	string	normalized number	Destination with blocking prefix removed (e.g., 555123456)
destination_number	string	normalized number	Same as tas_destination_number (both are set)

Variables Your Dialplan Should Set (when cli_withheld="true"):

These variables control how caller identity is presented:

Variable	Recommended Value	Purpose
effective_caller_id_number	"anonymous"	Hides the caller's phone number
effective_caller_id_name	"anonymous"	Hides the caller's display name
origination_privacy	"hide_number"	SIP privacy flag for outbound leg
sip_h_Privacy	"id"	SIP Privacy header (RFC 3323)
sip_h_P-Asserted-Identity	(unset or remove)	Optional: Remove P-Asserted-Identity header

Complete Dialplan Example:

```
<extension name="CLI-Privacy-Handler" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <!-- Log for troubleshooting -->
    <action application="log" data="INFO CLI blocking requested for call to
${tas_destination_number}"/>
  </condition>
</extension>
```

```

<!-- Hide caller identity on outbound call -->
<action application="set" data="effective_caller_id_name=anonymous"/>
<action application="set" data="effective_caller_id_number=anonymous"/>
<action application="set" data="origination_privacy=hide_number"/>

<!-- Set SIP Privacy headers -->
<action application="set" data="sip_h_Privacy=id"/>

<!-- Optional: Remove P-Asserted-Identity if present -->
<action application="unset" data="sip_h_P-Asserted-Identity"/>

<!-- Anti-action runs if cli_withheld is false -->
<anti-action application="log" data="DEBUG Using normal caller ID:
${msisdn}"/>
<anti-action application="set"
data="effective_caller_id_number=${msisdn}"/>
</condition>
</extension>

<!-- This extension continues to the actual call routing -->
<extension name="Route-Outbound-Call">
<condition field="${tas_destination_number}" expression="^(.+)$">
<action application="bridge" data="sofia/gateway/
trunk/${tas_destination_number}"/>
</condition>
</extension>

```

Important Notes:

- Both methods can work simultaneously (prefix OR SIP header triggers blocking)
- The prefix is **always stripped** from the destination number, even if dialplan doesn't implement privacy
- The cli_withheld variable is a **string** ("true" or "false"), not a boolean
- Call Forwarding / Blocked CLI behavior is implemented in your dialplan XML
- The example config includes these features, but if you do not define them in your dialplan, they will not function
- Variables are set during the MO (Mobile Originating) call flow only

How Call Forwarding Works

Call forwarding (also known as Communication Diversion or CDIV) allows subscribers to redirect incoming calls to another destination. The TAS supports multiple types of call forwarding with configurable behavior.

Types of Call Forwarding

1. Call Forward All (CFA) - Unconditional Forwarding

- **Variable:** call_forward_all_destination
- **When Active:** All incoming calls are immediately forwarded
- **Priority:** Checked first (after HLR forwarding)
- **Common Use:** Subscriber wants all calls sent to another number
- **Example:** Business calls forwarded to personal phone

2. Call Forward Busy (CFB)

- **When Active:** Call forwarded when subscriber is already on a call
- **SIP Response:** 486 Busy triggers forwarding
- **Common Use:** Forward to voicemail when on another call

3. Call Forward No Reply (CFNRy)

- **Variable:** no_reply_timer
- **When Active:** Call forwarded after ringing for specified seconds with no answer
- **Timeout:** Typically 15-30 seconds
- **Common Use:** Forward to voicemail if not answered

4. Call Forward Not Reachable (CFNRc)

- **Variable:** call_forward_not_reachable_destination
- **When Active:** Subscriber is offline, unregistered, or unreachable
- **SIP Response:** 480 Temporarily Unavailable
- **Common Use:** Forward to voicemail when phone is off
- **Default:** Configuration parameter used if no MMTel-Config

Data Source Priority

Call forwarding data is retrieved from multiple sources with this priority:

1. HLR Data (SS7 MAP) [Highest Priority - overrides all]
↓ (if no HLR forwarding active)
2. MMTel-Config (Sh Interface) [Subscriber-specific settings from HSS]
↓ (if no MMTel-Config returned)
3. Configuration Defaults [Lowest Priority - fallback values]

Why This Priority?

- **HLR Data:** Real-time forwarding status for roaming/network scenarios
- **MMTel-Config:** Subscriber-configured preferences in IMS
- **Config Defaults:** Network-wide fallback (typically voicemail)

Dialplan Variables for Call Forwarding

Variable	Type	Source	Example Value	Description
call_forward_all_destination	string	Sh/MMTel or "none"	"61403555123"	CFA destination if active
call_forward_not_reachable_destination	string	Sh/MMTel or config	"2222"	CFNRc destination (voicemail)
no_reply_timer	integer	Sh/MMTel or config	30	Seconds to ring before CFNRy
msrn	string	HLR (MT only)	"61400123456"	MSRN or forwarded number

Variable	Type	Source	Example Value	Description
tas_destination_number	string	Calculated	"2222"	from HLR Actual routing destination (may be forwarding number)

Implementing Call Forwarding in Dialplan

Example MT Dialplan with Call Forwarding:

```

<!-- Check for Call Forward All (highest priority after HLR) -->
<extension name="Check-CFA" continue="true">
  <condition field="${call_forward_all_destination}"
expression="^(?!none$).+$">
    <action application="log" data="INFO Call Forward All active to
${call_forward_all_destination}"/>
    <action application="set"
data="tas_destination_number=${call_forward_all_destination}"/>
  </condition>
</extension>

<!-- Attempt to bridge to subscriber -->
<extension name="Bridge-To-Subscriber">
  <condition field="${msrn}" expression="^none$">
    <!-- No MSRN, route to local subscriber -->
    <action application="set" data="call_timeout=${no_reply_timer}"/>
    <action application="bridge" data="sofia/
internal/${tas_destination_number}@${scscf_address}"/>

    <!-- If bridge fails, check forwarding -->
    <action application="log" data="INFO Bridge failed, checking call
forwarding"/>

    <!-- Call Forward Not Reachable -->
    <action application="set"
data="forward_destination=${call_forward_not_reachable_destination}"/>
    <action application="log" data="INFO Forwarding to
${forward_destination}"/>
    <action application="answer"/>
    <action application="voicemail" data="default default ${msisdn}"/>
  </condition>
</extension>

```

Configuring Default Call Forwarding

Set network-wide defaults in config/runtime.exs:

```

config :tas,
  # Default CFNRC destination (used when no MMTel-Config)

```

```
call_forward_not_reachable_destination: "2222", # Voicemail access number

# Default timeout before CFNRy activates (used when no MMTel-Config)
default_no_reply_timer: 30 # Ring for 30 seconds
```

When Defaults Are Used:

- Subscriber exists in HSS but has no MMTel-Config provisioned
- Sh lookup succeeds but returns no call forwarding settings
- New subscribers before call forwarding is configured

Troubleshooting Call Forwarding

Problem: Calls not forwarding as expected

1. Check Sh Data:

- Use Web UI /sh_test to query subscriber
- Verify MMTel-Config contains CDIV rules
- Check call_forward_all_destination value

2. Check Dialplan Variables:

- Review call logs for variable values
- Confirm call_forward_all_destination != "none"
- Verify tas_destination_number is set to forwarding destination

3. Check HLR Data (if SS7 MAP enabled):

- Use Web UI /hlr to query subscriber
- HLR forwarding overrides Sh data
- Verify msrn variable doesn't contain unexpected forwarding number

4. Check Configuration Defaults:

- Verify call_forward_not_reachable_destination in config
- Confirm default_no_reply_timer is appropriate
- These only apply when no MMTel-Config exists

Problem: Forwarding loops

Symptoms: Call forwards to a number that forwards back, creating a loop

Prevention in Dialplan:

```
<!-- Track forwarding hop count -->
<extension name="Prevent-Forward-Loop" continue="true">
  <condition field="${sip_h_X-Forward-Hop-Count}" expression="^$">
    <action application="set" data="sip_h_X-Forward-Hop-Count=1"/>
    <anti-action application="set" data="sip_h_X-Forward-Hop-Count=${expr(${sip_h_X-Forward-Hop-Count}+1)}/>
  </condition>
</extension>

<extension name="Check-Forward-Hop-Limit">
```

```
<condition field="${sip_h_X-Forward-Hop-Count}"
expression="^([3-9]|[1-9][0-9]+)$">
  <action application="log" data="ERROR Forwarding loop detected, hop count:
${sip_h_X-Forward-Hop-Count}"/>
  <action application="hangup" data="LOOP_DETECTED"/>
</condition>
</extension>
```

Monitoring Call Forwarding

Key Indicators:

- High rate of calls to voicemail numbers
- Pattern of calls timing out at `no_reply_timer` value
- Calls consistently routed to same forwarding destinations

Useful Logs:

```
INFO Call Forward All active to 61403555123
INFO Forwarding to 2222
INFO Bridge failed, checking call forwarding
```

Business Intelligence:

- Track forwarding activation rates by subscriber
- Monitor voicemail usage patterns
- Identify subscribers with unconditional forwarding



Voicemail & Missed Call Service

◊ [Back to Main Documentation](#)

Configuration and implementation of voicemail service with SMS notifications.

Related Documentation

Core Documentation

- ◊ [Main README](#) - Overview and quick start
- ◊ [Configuration Guide](#) - Voicemail configuration (timezone, SMS, notification templates)
- ◊ [Operations Guide](#) - Voicemail management in Control Panel

Call Processing Integration

- ◊ [Dialplan Configuration](#) - Voicemail deposit/retrieval in dialplan
- ⚙ [Supplementary Services](#) - Call forward on busy/no-answer to voicemail
- ◊ [TTS Prompts](#) - Voicemail greeting prompts

Related Services

- ◊ [Number Translation](#) - Voicemail access number translation

Monitoring

- ◊ [Metrics Reference](#) - Voicemail usage metrics

Voicemail / Missed Call Service

Voicemail is added in the XML dialplan as needed and is not turned on unless you call it in your dialplan.

You can view the voicemail box usage and message status from the Control Pannels' voicemail tab, for example putting this after your bridge command, to be called if the bridge fails:

```
<action application="log"
  data="INFO Failed to bridge Call - Routing to Call Forward No-Answer Destination" />
<action application="set"
  data="sip_h_History-Info=<sip:${destination_number}@${ims_domain}>;index=1.1" />
<action application="set" data="sip_call_id=${sip_call_id};CALL_FORWARD_NO_ANSWER" />
<action application="log" data="DEBUG Called Voicemail Deposit Number for ${msisdn}" />
<action application="set" data="default_language=fr"/>
<action application="answer" />
<action application="sleep" data="500"/>
<!--This notifies the TAS of missed calls or deposited voicemails so we can send
notifications after the call hangs up-->
<action application="set"
data='vm_post_body=mailbox=${msisdn}&caller=${effective_caller_id_number}&action="deposit"' />
<action application="set" data='api_hangup_hook=curl http://localhost:8080/vm_end content-
type application/x-www-form-urlencoded post ${vm_post_body}' />
<action application="voicemail" data="default default ${msisdn}" />
```

You can also access voicemails with a block like this:

```
<extension name="Static-Route-Voicemail-Check">
  <condition field="${tas_destination_number}" expression="^(2222|55512411520)$">
    <action application="log" data="DEBUG Called Voicemail Check Number" />
    <action application="set" data="default_language=fr"/>
```



```

    <action application="answer" />
    <action application="set" data="voicemail_authorized=true"/>
    <action application="set" data='vm_post_body=mailbox=${msisdn}&action="clear"'/>
    <action application="set" data='api_hangup_hook=curl http://localhost:8080/vm_end
content-type application/x-www-form-urlencoded post ${vm_post_body}'/>
    <action application="voicemail" data="check auth default default ${msisdn}"/>
  </condition>
</extension>

```

You can also enable missed call (but no voicemail left) SMS notifications and voicemail MWI notification SMS from the configuration.

Variables available in the missed call notification include:

```

bindings = [
  caller: caller,
  day: day,
  month: month,
  hour: hour,
  minute: minute,
  message_count: message_count
]

```

NB: message_count is only set when message count is *greater* than 1.

```

config :tas,
  ...
  voicemail: %{
    timezone: "Pacific/Tahiti",           #Timezone used in Timestamps
    smsc: %{
      smsc_url: "http://10.8.81.215",      #SMS Sc API Base URL
      smsc_api_key: "nicktestkey123",      #API key on SMS Sc
      source_msisdn: "2222"               #Source (Sender) for the notification
    }
  },
  messages
  },
  #For usage of variables in this section see docs.
  voicemail_notification_text: %{
    not_left:
      "Vous avez 1 appel manqué du <%= caller %> le <%= day %>/<%= month %> à <%= hour
%>:<%= minute %>",
    single_voicemail:
      "Vous avez un nouveau message vocal du <%= caller %> le <%= day %>/<%= month %> à <%=
hour %>:<%= minute %>. Pour le consulter, composez le 2222.",
    multiple_voicemails:
      "Vous avez <%= message_count %> nouveaux messages vocaux. Pour les consulter,
composez le 2222."
  }
}

```