Prometheus □标和监控□南

概述

OmniTAS □ Prometheus 格式导出全面的操作指标，用于监控、警报和可观察性。本指南涵盖所有可用指标、它们的使用、故障排除和监控最佳实践。

□标端点

所有指标在□下地址公开：`http://<tas-ip>:9094/metrics`

完整□标参考

Diameter □标

`diameter_response_duration_milliseconds`

类型： Histogram
标签： `application` (ro, sh), `command` (ccr, cca, 等), `result` (success, error, timeout)
桶： 10, 50, 100, 250, 500, 1000, 2500, 5000, 10000 ms
描述： Diameter 请求的持续时间（□毫秒为单位）

用法：

```
# □均 Diameter 响应时间
rate(diameter_response_duration_milliseconds_sum[5m]) /
rate(diameter_response_duration_milliseconds_count[5m])

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

警报条件:

- P95 > 1000ms - Diameter 响应缓慢

`diameter_requests_total`

类型: Counter
标签: `application` (ro, sh), `command` (ccr, udr, 等)
描述: 发送的 Diameter 请求总数

用法:

```
# 请求速率
rate(diameter_requests_total[5m])
```

`diameter_responses_total`

类型: Counter
标签: `application` (ro, sh), `command` (ccr, udr, 等), `result_code` (2001, 3002, 5xxx, 等)
描述: 接收到的 Diameter 响应总数

用法:

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

`diameter_peer_state`

类型: Gauge
标签: `peer_host`, `peer_realm`, `application` (ro, sh)
描述: Diameter 对等体的状态 (1=up, 0=down)
更新间隔: 每 10 秒

用法:

```
# 检查下线对等体
diameter_peer_state == 0
```

警报条件：

- 任何对等体下线超过 1 分钟

拨号计划生成□标

1. HTTP 请求□标

`http_dialplan_request_duration_milliseconds`

类型： Histogram

标签： `call_type` (mt, mo, emergency, unknown)

描述：端到端 **HTTP** 请求持续时间 从拨号计划 HTTP 请求接收开始到响应发送结束。这包括所有处理：参数解析、授权、Diameter 查找（Sh/Ro）、HLR ���找（SS7 MAP）和 XML 生成。

用法：

```
# □均端到端 HTTP 请求时间
rate(http_dialplan_request_duration_milliseconds_sum[5m]) /
rate(http_dialplan_request_duration_milliseconds_count[5m])

# 按呼叫类型的 P95
histogram_quantile(0.95,
  rate(http_dialplan_request_duration_milliseconds_bucket[5m])
) by (call_type)

# 比较 MT 与 MO 性能
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])
)
```

警报条件:

- P95 > 2000ms - HTTP 响应时间缓慢
- P95 > 3000ms - 关键性能问题
- P99 > 5000ms - 严重性能下降
- 任何请求显示 `call_type="unknown"` - 呼叫类型检测失败

洞察:

- 这是理解用户面临延迟的 最重要□标
- 典型值: P50: 100-500ms, P95: 500-2000ms, P99: 1000-3000ms
- 包括所有组件的时间（Sh + HLR + OCS + 处理）
- 如果这个指标缓慢，请深入分析组件指标（subscriber_data, hlr_data, ocs_authorization）
- 预期范围: 100ms（快速本地呼叫）到 5000ms（慢速带重试/超时）

重要说明:

- 替代了旧的 `dialplan_generation_duration_milliseconds` 指标，该指标仅测量 XML 生成
- 准确反映 FreeSWITCH/SBC 的体验
- 用于 SLA 监控和容量规划

2. 订阅者数据□标

`subscriber_data_duration_milliseconds`

类型： Histogram
标签： `result` (success, error)
描述：从 Sh 接口（HSS）检索订阅者数据所需的时间

用法：

```
# □均 Sh 查找时间
rate(subscriber_data_duration_milliseconds_sum[5m]) /
rate(subscriber_data_duration_milliseconds_count[5m])

# 第 95 百分位 Sh 查找时间
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m])
)
```

警报条件：

- P95 > 100ms - HSS 响应缓慢
- P95 > 500ms - 关键 HSS 性能问题

`subscriber_data_lookups_total`

类型： Counter
标签： `result` (success, error)
描述：订阅者数据查找的总数

用法：

```
# Sh 查找速率
rate(subscriber_data_lookups_total[5m])

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

# Sh 成功率百分比
(rate(subscriber_data_lookups_total{result="success"}[5m]) /
 rate(subscriber_data_lookups_total[5m])) * 100
```

警报条件:

- 错误率 > 5% - HSS 连接问题
- 错误率 > 20% - 关键 HSS 故障

2. HLR 数据□标

`hlr_data_duration_milliseconds`

类型: Histogram
标签: `result` (success, error)
描述: 通过 SS7 MAP 检索 HLR 数据所需的时间

用法:

```
# □均 HLR 查找时间
rate(hlr_data_duration_milliseconds_sum[5m]) /
rate(hlr_data_duration_milliseconds_count[5m])

# 第 95 百分位 HLR 查找时间
histogram_quantile(0.95,
  rate(hlr_data_duration_milliseconds_bucket[5m])
)
```

警报条件:

- P95 > 500ms - SS7 MAP 响应缓慢
- P95 > 2000ms - 关键 SS7 MAP 问题

`hlr_lookups_total`

类型: Counter
标签: `result_type` (msrn, forwarding, error, unknown)
描述: 按结果类型的总 HLR 查找

用法:

```
# 按类型的 HLR 查找速率
rate(hlr_lookups_total[5m])

# MSRN 发现速率（漫游订阅者）
rate(hlr_lookups_total{result_type="msrn"}[5m])

# 呼叫转发发现速率
rate(hlr_lookups_total{result_type="forwarding"}[5m])

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

警报条件:

- 错误率 > 10% - SS7 MAP 问题
- MSRN 速率突然下降 - 可能的漫游问题

洞察:

- 高 MSRN 速率表示许多漫游订阅者
- 高转发率表示许多转发呼叫
- 与呼叫量进行比较□计算漫游百分比

3. OCS 授权□标

`ocs_authorization_duration_milliseconds`

类型: Histogram
标签: `result` (success, error)
描述: OCS 授权所需的时间

用法:

```
# □均 OCS 授权时间
rate(ocs_authorization_duration_milliseconds_sum[5m]) /
rate(ocs_authorization_duration_milliseconds_count[5m])

# 第 95 百分位 OCS 授权时间
histogram_quantile(0.95,
  rate(ocs_authorization_duration_milliseconds_bucket[5m])
)
```

警报条件:

- P95 > 1000ms - OCS 响应缓慢
- P95 > 5000ms - 关键 OCS 性能问题

`ocs_authorization_attempts_total`

类型: Counter
标签: `result` (success, error), `skipped` (yes, no)
描述: OCS 授权尝试的总数

用法:

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

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

# OCS 跳过率（紧急、语音信箱等）
rate(ocs_authorization_attempts_total{skipped="yes"}[5m])

# OCS 成功率百分比
(rate(ocs_authorization_attempts_total{result="success",skipped="no"}
[5m]) /
 rate(ocs_authorization_attempts_total{skipped="no"}[5m])) * 100
```

警报条件:

- 错误率 > 5% - OCS 连接问题
- 成功率 < 95% - OCS 拒绝过多呼叫

洞察：

- 高跳过率表示许多紧急/免费呼叫
- 错误率激增表示 OCS 故障
- 将成功率与业务期望进行比较

4. 呼叫处理□标

`call_param_errors_total`

类型：Counter
标签：`error_type` (parse_failed, missing_required_params)
描述：呼叫参数解析错误

用法：

```
# 参数错误率
rate(call_param_errors_total[5m])

# 按类型的错误
rate(call_param_errors_total[5m]) by (error_type)
```

警报条件：

- 任何错误 > 0 - 表示格式错误的呼叫参数请求
- 错误 > 呼叫量的 1% - 关键问题

`authorization_decisions_total`

类型：Counter
标签：`disposition` (mt, mo, emergency, unauthorized), `result` (success, error)
描述：按呼叫类型的授权决策

用法：

```
# 按处置的授权率
rate(authorization_decisions_total[5m]) by (disposition)

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

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

# 紧急呼叫率
rate(authorization_decisions_total{disposition="emergency"}[5m])

# 未授权呼叫率
rate(authorization_decisions_total{disposition="unauthorized"}
[5m])
```

警报条件：

- 未授权率 > 1% - 可能的攻击或配置错误
- 紧急呼叫的突然激增 - 可能的紧急事件
- MT/MO 比率的意外变化 - 可能的问题

洞察：

- MT/MO 比率指示流量��式
- 紧急呼叫率指示服务使用情况
- 未授权率指示安全态势

`freeswitch_variable_set_duration_milliseconds`

类型： Histogram

标签： `batch_size` (1, 5, 10, 25, 50, 100)

描述：设置拨号计划变量所需的时间

用法：

```
# □均变量设置时间
rate(freeswitch_variable_set_duration_milliseconds_sum[5m]) /
rate(freeswitch_variable_set_duration_milliseconds_count[5m])

# 按批量大小的变量设置时间
histogram_quantile(0.95,
  rate(freeswitch_variable_set_duration_milliseconds_bucket[5m])
) by (batch_size)
```

警报条件：

- P95 > 100ms - 变量设置性能缓慢
- 增长趋势 - 可能的系统性能问题

5. 模块处理□标

`dialplan_module_duration_milliseconds`

类型：Histogram
标签：`module` (MT, MO, Emergency, CallParams, 等), `call_type`
描述：每个拨号计划模块的处理时间

用法：

```
# 按模块的处理时间
histogram_quantile(0.95,
  rate(dialplan_module_duration_milliseconds_bucket[5m])
) by (module)

# MT 模块处理时间
histogram_quantile(0.95,
  rate(dialplan_module_duration_milliseconds_bucket{module="MT"}
[5m])
)
```

警报条件：

- 任何模块 P95 > 500ms - 性能问题
- 任何模块的增长趋势 - 潜在泄漏或问题

洞察：

- 确定哪个模块最慢
- 优化最慢的模块
- 比较不同呼叫类型的模块时间

6. 呼叫量□标

`call_attempts_total`

类型： Counter
标签： `call_type` (mt, mo, emergency, unauthorized), `result` (success, rejected)
描述：总呼叫尝试

用法：

```
# 呼叫尝试速率
rate(call_attempts_total[5m])

# 按呼叫类型的成功率
(rate(call_attempts_total{result="success"}[5m]) /
 rate(call_attempts_total[5m])) * 100 by (call_type)

# 被拒绝的呼叫率
rate(call_attempts_total{result="rejected"}[5m])
```

警报条件：

- 被拒绝率 > 5% - 可能的问题
- 呼叫量突然下降 - 服务中断
- 呼叫量突然激增 - 可能的攻击

`active_calls`

类型： Gauge
标签： `call_type` (mt, mo, emergency)
描述：当前活跃呼叫

用法：

```
# 当前活跃呼叫
active_calls

# 按类型的活跃呼叫
active_calls by (call_type)

# 峰值活跃呼叫（过去一小时）
max_over_time(active_calls[1h])
```

警报条件：

- 活跃呼叫 > 容量 - 超载
- 活跃呼叫 = 0 超过一段时间 - 服务中断

7. 模拟□标

`call_simulations_total`

类型：Counter
标签：`call_type` (mt, mo, emergency, unauthorized), `source` (web, api)
描述：运行的呼叫模拟

用法：

```
# 模拟速率
rate(call_simulations_total[5m])

# 按类型的模拟
rate(call_simulations_total[5m]) by (call_type)
```

洞察：

- 跟踪诊断工具的使用情况
- 确定重度用户
- 与故障排除活动相关联

8. SS7 MAP □标

`ss7_map_http_duration_milliseconds`

类型: Histogram
标签: `operation` (sri, prn), `result` (success, error, timeout)
桶: 10, 50, 100, 250, 500, 1000, 2500, 5000, 10000 ms
描述: SS7 MAP HTTP 请求的持续时间（□毫秒为单位）

用法:

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

警报条件:

- P95 > 500ms - SS7 MAP 响应缓慢
- 错误率 > 50% - 关键 SS7 MAP 问题

`ss7_map_operations_total`

类型: Counter
标签: `operation` (sri, prn), `result` (success, error)
描述: SS7 MAP 操作的总数

9. 在线计费□标

`online_charging_events_total`

类型: Counter
标签: `event_type` (authorize, answer, reauth, hangup), `result` (success, nocredit, error, timeout)
描述: 在线计费事件的总数

用法:

```
# OCS 信用失败
rate(online_charging_events_total{result="nocredit"}[5m])
```

警报条件：

- 信用失败的高比率

10. 系统状态□标

`tracked_registrations`

类型：Gauge
描述：当前活跃的 SIP 注册数量（来自 FreeSWITCH Sofia 注册数据库）
更新间隔：每 10 秒

说明：

- 注册过期时会自动减少（FreeSWITCH 管理过期）

`tracked_call_sessions`

类型：Gauge
描述：当前在 ETS 中跟踪的呼叫会话数量
更新间隔：每 10 秒

11. HTTP 请求□标

`http_requests_total`

类型：Counter
标签：`endpoint` (dialplan, call_event, directory, voicemail, sms_ccr, metrics), `status_code` (200, 400, 500, 等)
描述：按端点统计的 HTTP 请求总数

用法：

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

警报条件：

- HTTP 5xx 错误率 > 10%

12. 呼叫拒绝□标

`call_rejections_total`

类型：Counter
标签：`call_type` (mo, mt, emergency, unknown), `reason` (nocredit, unauthorized, parse_failed, missing_params, hlr_error, 等)
描述：按原因统计的呼叫拒绝总数

用法：

```
# 按原因的呼叫拒绝率
sum by (reason) (rate(call_rejections_total[5m]))
```

警报条件：

- 拒绝率 > 1/秒 - 需要调查

13. 事件套接字连接□标

`event_socket_connected`

类型：Gauge
标签：`connection_type` (main, log_listener)
描述：事件套接字连接状态 (1=connected, 0=disconnected)
更新间隔：连接状态变化时实时更新

用法：

```
# 事件套接字连接状态
event_socket_connected
```

警报条件：

- 连接下线超过 30 秒

`event_socket_reconnections_total`

类型：Counter
标签：`connection_type` (main, log_listener), `result` (attempting, success, failed)
描述：事件套接字重连尝试的总数

Grafana 仪表板集成

可使用 Prometheus 数据源在 Grafana 中可视化这些指标。推荐面板：

仪表板 1：呼叫量

- 活跃呼叫仪表
- 按类型的呼叫尝试速率（MO/MT/紧急）
- 呼叫拒绝率

仪表板 2：Diameter 性能

- 响应时间热图
- 请求/响应速率
- 对等状态表
- 按结果代码的错误率

仪表板 3：在线计费健康

- 信用授权成功率
- "无信用"事件率
- OCS 超时率

仪表板 **4**：系统性能

- 拨号计划生成延迟 (P50/P95/P99)
- SS7 MAP 响应时间
- 整体系统可用性

推荐的 **Grafana** 仪表板布局

** 1**：呼叫量

- 按类型的呼叫尝试速率
- 活跃呼叫仪表
- 成功率百分比

行 **2**：性能

- P95 HTTP 拨号计划请求时间（按呼叫类型） - 主要□标
- P95 Sh 查找时间
- P95 HLR 查找时间
- P95 OCS 授权时间
- P95 拨号计划模块处理时间（按模块）

行 **3**：成功率

- Sh 查找成功率
- HLR 查找成功率
- OCS 授权成功率
- 呼叫尝试成功率

行 **4**：模块性能

- 按模块的 P95 处理时间
- 模块呼叫计数

行 **5**：错误

- 参数错误

- 未授权尝试
- Sh 错误
- HLR 错误
- OCS 错误

关键警报

优先级 **1**（立即分页）：

```
# 拨号计划完全宕机
rate(call_attempts_total[5m]) == 0

# HSS 完全宕机
rate(subscriber_data_lookups_total{result="error"}[5m]) /
rate(subscriber_data_lookups_total[5m]) > 0.9

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

优先级 **2**（警报）：

```
# 拨号计划生成缓慢
histogram_quantile(0.95,
  rate(dialplan_generation_duration_milliseconds_bucket[5m])
) > 1000

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

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

优先级 **3**（警告）：

```
# HSS 延迟升高
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m])
) > 100

# OCS 延迟升高
histogram_quantile(0.95,
  rate(ocs_authorization_duration_milliseconds_bucket[5m])
) > 1000

# 中等错误率
rate(call_attempts_total{result="rejected"}[5m]) /
rate(call_attempts_total[5m]) > 0.05
```

警报示例

Diameter 对等体下线

```
alert: DiameterPeerDown
expr: diameter_peer_state == 0
for: 1m
annotations:
  summary: "Diameter 对等体 {{ $labels.peer_host }} 已下线"
```

高 **Diameter** 延迟

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

OCS 信用失败

```
alert: HighOCSCreditFailures
expr: rate(online_charging_events_total{result="nocredit"}[5m]) >
0.1
for: 2m
annotations:
  summary: "OCS 信用失败率高"
```

SS7 MAP 网关错误

```
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 错误率超过 50%"
```

事件套接字断开连接

```
alert: EventSocketDown
expr: event_socket_connected == 0
for: 30s
annotations:
  summary: "事件套接字 {{ $labels.connection_type }} 已断开连接"
```

高呼叫拒绝率

```
alert: HighCallRejectionRate
expr: rate(call_rejections_total[5m]) > 1
for: 2m
annotations:
  summary: "高呼叫拒绝率: {{ $value }} 拒绝/秒"
```

HTTP 错误率高

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

使用□标进行故障排除

问题：呼叫类型显示为 "unknown"

症状：

- 所有指标显示 `call_type="unknown"` 而不是 `mt`, `mo` 或 `emergency`
- 无法区分呼叫类型之间的性能

根本原因：呼叫类型提取失败或未正确传递通过处理管道。

调查：

1. 检查日志中是否有 "HTTP 拨号计划请求" 消息 - 它们应该显示正确的呼叫类型
2. 审查系统日志□查找呼叫类型处理错误

解决方案：如果呼叫类型检测持续失败，请联系支持。

问题：呼叫缓慢

调查：

1. 检查 `http_dialplan_request_duration_milliseconds` P95 - 从这里开始
2. 如果高，��查组件时间：
   - 检查 `subscriber_data_duration_milliseconds` □查找 Sh 延迟
   - 检查 `hlr_data_duration_milliseconds` □查找 HLR 延迟
   - 检查 `ocs_authorization_duration_milliseconds` □查找 OCS 延迟

    - 检查 `dialplan_module_duration_milliseconds` 查找模块特定的延迟
3. 检查是否 `call_type="unknown"` - 表示呼叫类型检测失败
4. 比较 MT 与 MO 与紧急处理时间
5. 与系统日志进行关联获取详细错误消息

解决方案：优化最慢的组件

问题：呼叫失败

调查：

1. 检查 `call_attempts_total{result="rejected"}` 速率
2. 检查 `subscriber_data_lookups_total{result="error"}` 查找 Sh 问题
3. 检查 `hlr_lookups_total{result_type="error"}` 查找 HLR 问题
4. 检查 `ocs_authorization_attempts_total{result="error"}` 查找 OCS 问题
5. 检查 `authorization_decisions_total{disposition="unauthorized"}` 查找授权问题

解决方案：修复失败的组件

问题：高负载

调查：

1. 检查当前的 `active_calls` 值
2. 检查 `call_attempts_total` 速率
3. 检查速率是否与预期流量匹配
4. 比较 MT 与 MO 比率
5. 检查是否有异常模式（激增、稳定增长）

解决方案：扩大规模或调查异常流量

问题：漫游问题

调查：

1. 检查 `hlr_lookups_total{result_type="msrn"}` 速率

2. 检查 `hlr_data_duration_milliseconds` 查找延迟
3. 使用 HLR 查找工具查找特定订阅者
4. 检查 MSRN 是否被正确检索

解决方案：修复 HLR 连接或配置

性能基准

典型值（调优良好的系统）

- **HTTP** 拨号计划请求（端到端）： P50: 100-500ms, P95: 500-2000ms, P99: 1000-3000ms
- **Sh** 查找时间： P50: 15ms, P95: 50ms, P99: 100ms
- **HLR** 查找时间： P50: 100ms, P95: 300ms, P99: 800ms
- **OCS** 授权时间： P50: 150ms, P95: 500ms, P99: 1500ms
- 拨号计划模块处理： P50: 1-5ms, P95: 10-25ms, P99: 50ms
- **Sh** 成功率： > 99%
- **HLR** 成功率： > 95%（由于离线订阅者，较低是正常的）
- **OCS** 成功率： > 98%
- 呼叫成功率： > 99%

注意： HTTP 拨号计划请求时间是所有组件时间加上开销的总和。它应该大致等于：Sh 查找 + HLR 查找 + OCS 授权 + 拨号计划模块处理 + 网络/解析开销。最低预期时间约为 ~100ms（仅需要 Sh 查找时），最大典型时间约为 ~2000ms（包含所有查找和重试）。

容量规划

监控这些趋势：

- `call_attempts_total` 速率的增长
- `active_calls` 峰值的增长
- P95 延迟稳定或改善
- 成功率稳定或改善

当下情况发生时，计划扩展：

- 活跃呼叫接近 80% 的容量
- 尽管负载稳定，P95 延迟增长
- 尽管外部系统稳定，成功率下降

与日志的集成

将指标与日志关联：

1. 指标中的高错误率 → 在日志中搜索 ERROR 消息
2. 响应时间缓慢 → 在日志中搜索关于超时的 WARNING 消息
3. 特定呼叫问题 → 按呼叫 ID 或电话号码搜索日志
4. 使用模拟工具重现和调试

最佳实践

1. 在问题发生之前设置仪表板
2. 根据基准定义警报阈值
3. 通过使用呼叫模拟器测试警报
4. 每周审查□标以识别趋势
5. 将□标与业务事件相关联（活动、故障等）
6. 使用□标为基础设施投资辩护
7. 与运营团队共享仪表板
8. 记录您的警报响应程序

配置

指标收集在应用程序启动时自动启用。指标端点在与 API 相同的端口上公开（默认：9094）。

要配置 Prometheus 抓取指标，请将此作业添加到您的 `prometheus.yml` 中：

```yaml
scrape_configs:
  - job_name: 'omnitas'
    static_configs:
      - targets: ['<tas-ip>:9094']
    metrics_path: '/metrics'
    scrape_interval: 10s
```

标基数

这些指标的设计具有受控的基数，避免对 Prometheus 造成过大压力：

- 对等体标签：限制为仅配置的对等体
- 呼叫类型：固定集（mo, mt, emergency, unauthorized）
- 结果代码：限制为实际接收到的 Diameter/OCS 结果代码
- 操作：每个接口的固定集（sri/prn 用于 MAP，ccr/cca 用于 Diameter）

估计的时间序列总数：~200-500，具体取决于配置的对等体数量和活动结果代码。

标保留

推荐的保留期限：

- 原始标：30 天（高分辨率）
- **5** 分钟聚合：90 天
- **1** 小时聚合：1 年
- 每日聚合：5 年

这支持：

- 实时故障排除（原始指标）
- 每周/月分析（5 分钟/1 小时聚合）
- 容量规划（每日聚合）
- 历史比较（年度聚合）

HLR 查询和呼叫模拟器 - 用户□南

概述

用于帮助运营人员排除呼叫路由故障的诊断工具。**HLR** 查询和呼叫模拟器为只读，不影响实时流量。回声测试呼叫则不同。它会发起一个真实的 MT 呼叫（请参见其单独部分）。

HLR 查询工具

目的

HLR 查询工具通过 SS7 MAP 协议查询家庭位置寄存器（HLR），□获取实时的用户路由信息。

访问

导航到 `/hlr` 或点击导航菜单中的 "HLR"。

显示内容

对于任何电话号码，HLR 查询显示：

1. **MSRN**（移动台漫游号码）
    - 在用户漫游到 2G/3G 网络时分配的临时路由号码
    - 仅在用户当前漫游时存在
    - 被拨号计划用于将呼叫路由到漫游用户的当前位置
2. 呼叫转移设置
    - 来自 HLR 的实时呼叫转移配置
    - 类型：无条件、忙碌、无应答、不可达
    - 显示转移目的号码
    - 显示通知是否启用

3. 拨号计划变量

将设置的确切通道变量
变量与实际呼叫处理中的变量匹配
显示 HLR 数据如何覆盖 Sh 数据

使用案例

诊断漫游问题

场景：对漫游用户的来电失败或路由不正确

步骤：

1. 打开 HLR 查询页面
2. 输入用户的电话号码
3. 点击 "查询 HLR 数据"
4. 检查结果中的 MSRN
5. 如果 MSRN 存在：用户正在漫游，验证 MSRN 是否有效
6. 如果没有 MSRN：用户可能在 LTE/VoLTE（不需要 MSRN）

验证呼叫转移

场景：呼叫转移未按预期工作

步骤：

1. 打开 HLR 查询页面
2. 输入用户的电话号码
3. 点击 "查询 HLR 数据"
4. 在结果中查找 "呼叫转移"
5. 验证转移类型（无条件、忙碌等）
6. 验证转移目的号码
7. 注意：HLR 数据覆盖任何 Sh/HSS 数据

测试 HLR 连接性

场景：验证 SS7 MAP 网关是否正常工作

步骤：

1. 打开 HLR 查询页面
2. 输入任何已知的用户号码
3. 点击 "查询 HLR 数据"
4. 检查结果中的 "错误"
5. 如果有错误：检查 SS7 MAP 网关连接性
6. 常见错误：
    - "SS7 MAP 被禁用" - 检查配置
    - "超时" - 网络问题到 HLR
    - "没有 VLR 号码" - 用户离线或不存在

信息框

HLR 查询页面包含教育信息，解释：

- MSRN 是什么□及何时使用
- HLR 中呼叫转移的工作原理
- 这如何与呼叫处理集成
- SS7 MAP 协议基础知识

呼叫模拟器工具

目的

呼叫模拟器允许您模拟完整的呼叫路由，而无需实际拨打电话或影响实时流量。

访问

导航到 `/simulator` 或点击导航菜单中的 "模拟器"。

特性

输入参数

1. 源号码（呼叫者）
    - 呼叫方的电话号码
    - 对于 MT 呼叫：可□是任何号码
    - 对于 MO 呼叫：必须是已配置的用户
2. 目的号码（被叫方）
    - 被叫方的电话号码
    - 对于 MT 呼叫：必须是已配置的用户
    - 对于 MO 呼叫：可□是任何号码
    - 对于紧急情况：使用 "urn:service:sos" 或类似的
3. 源 **IP** 地址
    - SIP 信令源的 IP 地址
    - 必须在 `allowed_sbc_source_ips`（对于 MT）或 `allowed_cscf_ips`（对于 MO）中
    - 决定呼叫处置（MT 与 MO）
4. 强制处置
    - 自动：根据 IP 地址确定（正常行为）
    - MT：强制移动终止（来电）
    - MO：强制移动发起（去电）
    - 紧急：强制紧急呼叫处理
5. 选项
    - 跳过 OCS 授权：绕过在线计费（更快的模拟）
    - 跳过 HLR 查询：绕过 SS7 MAP 查询（更快的模拟）

输���

模拟器显示全面的结果：

1. 呼叫类型横幅
    - MT、MO 或紧急
    - 颜色编码□便快速识别

- 显示源和目的号码

2. 处理步骤（左列）
    - 用户数据：来自 Sh 接口的结果（HSS）
    - **HLR** 数据：来自 SS7 MAP 查询的结果（仅 MT）
    - **OCS** 授权：来自在线计费的结果（仅 MO）
    - 网内状态：目的地是否在您的网络上（仅 MO）
3. 拨号计划变量（右列）
    - 每个将在通道上设置的变量
    - 按字母顺序排序□便于阅读
    - 颜色编码的值（绿色为正常，红色为错误）
4. 处理说明
    - 逐步解释发生了什么
    - 描述数据流和决策点
    - 帮助理解为何设置某些变量

使用案例

起飞前测试

场景：在部署到生产环境之前测试配置更改

步骤：

1. 在开发/测试环境中进行配置更改
2. 打开呼叫模拟器
3. 测试多个场景：
    - 从您的 SBC 发起的 MT 呼叫
    - 从您的 CSCF 发起的 MO 呼叫
    - 紧急呼叫
    - 网内目的地
    - 网外目的地
4. 验证所有变量是否正确

5. 检查处理说明□查找任何问题
6. 自信地部署到生产环境

调试 **MT** 呼叫问题

场景：对用户的来电失败

步骤：

1. 打开呼叫模拟器
2. 输入目的地为有问题的用户
3. 输入源为测试号码
4. 将源 IP 设置为您的 SBC IP
5. 将强制处置留为 "自动"
6. 点击 "模拟呼叫"
7. 检查用户数据部分□确认 Sh 查询成功
8. 检查 HLR 数据部分□获取 MSRN 或转移
9. 检查最终变量□获取 `hangup_case`
10. 如果 `hangup_case` 为 "UNALLOCATED_NUMBER"：用户未配置
11. 如果变量看起来正确：问题可能出在拨号计划模板中

调试 **MO** 呼叫问题

场景：用户的去电失败

步骤：

1. 打开呼叫模拟器
2. 输入源为有问题的用户
3. 输入目的地为测试号码
4. 将源 IP 设置为您的 CSCF IP
5. 如果测试计费，取消选中 "跳过 OCS 授权"
6. 点击 "模拟呼叫"
7. 检查呼叫者数据部分□确认 Sh 查询成功
8. 检查 OCS 授权部分□获取成功/失败
9. 检查网内状态□验证正确路由

10. 检查最终变量□获取 `allocated_time` 或 `hangup_case`
11. 如果 `hangup_case` 为 "OUTGOING_CALL_BARRED": OCS 拒绝了呼叫

测试紧急呼叫处理

场景：验证紧急呼叫是否正常工作

步骤：

1. 打开呼叫模拟器
2. 输入源为测试用户
3. 输入目的地为 "urn:service:sos"
4. 设置任意源 IP（紧急呼叫绕过 IP 认证）
5. 点击 "模拟呼叫"
6. 验证呼叫类型显示为 "紧急 (SOS)"
7. 验证 `hangup_case` 为 "none"（紧急呼叫始终进行）
8. 检查 OCS 和 HLR 是否被绕过
9. 验证呼叫者数据已获取位置信息

培训员工

场景：教授运营人员呼叫路由的工作原理

步骤：

1. 打开呼叫模拟器
2. 运行各种场景并解释每个部分：
    - 显示 MT 呼叫并解释 Sh + HLR 查询
    - 显示 MO 呼叫并解释 OCS 授权
    - 显示紧急呼叫并解释绕过行为
    - 显示未授权 IP 并解释拒绝
3. 让员工尝试不同组合
4. 使用处理说明解释每个决策
5. 比较不同场景之间的变量

比较 **Sh** 与 **HLR** 数据

场景：理解 HLR 如何覆盖 Sh 数据

步骤：

1. 打开 MT 呼叫的呼叫模拟器
2. 取消选中 "跳过 HLR 查询"
3. 点击 "模拟呼叫"
4. 比较用户数据变量与 HLR 数据变量
5. 检查最终变量查看哪些值胜出
6. 注意：HLR 数据始终优先：
    - MSRN
    - `call_forward_all_destination`
    - `call_forward_not_reachable_destination`

提示

- 在测试其他方面时，使用 "跳过 OCS 授权" 和 "跳过 HLR 查询" 加快模拟速度
- 从日志中复制/粘贴电话号码到模拟器快速测试
- 使用 "强制处置" 测试特定呼叫类型，而不考虑 IP
- 如果不确定为何设置某些变量，请检查处理说明
- 多次运行模拟验证一致性
- 将模拟结果与实际呼叫日志进行比较

限制

模拟器：

- 不会实际拨打电话
- 不会影响呼叫路由系统
- 不会消耗 OCS 配额（即使查询了 OCS）
- 不会生成 CDR
- 在生产系统上使用是安全的

模拟器确实：

- 如果未跳过，则查询实际的 Sh 接口（HSS）

- 如果未跳过，则通过 SS7 MAP 查询实际的 HLR
- 如果未跳过，则查询实际的 OCS
- 显示在真实呼叫中会发生的确切情况
- 使用真实的配置值

回声测试呼叫（真实的 **MT** 呼叫）

> 与上述呼叫模拟器不同，这发起一个真实的呼叫。它会响铃目的地 用户，并将他们通过完整的终止路径（Sh 查询、OCS、CDR、媒体）进行处理。

目的

发起一个真实的移动终止呼叫到一个用户，将他们连接到 FreeSWITCH 回声应用程序，便他们听到自己的音频回声。这对于对 MT 路径和媒体/编解码器腿的端到端验证非常有用，而无需真实的上游呼叫者。

API

`POST /simulator/echo_call`

字段	必需	描述
`destination`	✅	要响铃的 MSISDN: 终止用户（必须已配置/注册）。数字，前导 `+` 可选。
`source`	否	显示为呼叫者 ID 的 MSISDN。数字，前导 `+` 可选。

```
curl -k -X POST https://<tas>:8444/simulator/echo_call \
  -H 'content-type: application/json' \
  -d '{"destination":"99988037866","source":"61400000001"}'
```

响应（通过 `bgapi` 发出，立即返回）：

```
{
  "ok": true,
  "command": "bgapi originate
{tas_call_reason=echo_test,ignore_early_media=true,origination_caller
&echo",
  "result": "+OK Job-UUID: <uuid>"
}
```

工作原理

它发出一个 FreeSWITCH `originate`，将呼叫循环回本地内部配置文件（`sofia/internal/<destination>@<tas_ip>`）。然后，邀请从 TAS 自己的 IP 到达，该 IP 在 `allowed_sbc_source_ips` 中。因此，TAS 将其分类为**MT（就像由上游 SBC 提供的那样）**并运行真实的 MT 拨号计划，通过 IMS 核心路由到用户。 `&echo` 的应答使得响铃的用户听到自己的音频。

回环目标是从 Diameter 监听 IP（本地盒子）服务器端获取的；它不是请求参数。

它的功能和不功能

与干运行模拟器不同，回声呼叫确实：

- 发起一个真实的呼叫并响铃目的地用户
- 运行 Sh 查询、OCS 授权，并生成 CDR / Ro CCRs
- 运行真实的 MT 拨号计划和媒体路径

它对 MO / 计费欺诈是安全的：

- 始终**MT**，从不 **MO**：回环来自本地 IP，这永远不能是 CSCF 源 IP（唯一会导致 MO 的东西）。
- 它不能拨打任意外部号码：MT 桥接目标来自 Sh 用户查询（`NokiaCSCF/+<msisdn>`），而不是原始 `destination`；未知目的地 没有 MSISDN，桥接失败。它只能响铃已配置的网内用户。
- 回环 IP 仅为服务器端，因此呼叫永远不能指向外部 网关。`source`/`destination` 在到达 ESL 命令之前被验证为电话号码（没有命令注入）。

呼叫者 ID（`source`）按提供的方式显示给响铃的用户。请保持端点在 您的正常 API 访问控制之后，因为它确实会发起真实的呼叫。

与监控的集成

这两个工具与 Prometheus 指标集成:

- 通过工具进行的 HLR 查询计入 `hlr_lookups_total`
- 呼叫模拟计入 `call_simulations_total{call_type, source}`
- 处理时间在各自的持续时间指标中跟踪

这有助于:

- 跟踪诊断工具的使用
- 监控诊断查询的性能
- 识别诊断工具的重度用户

有关完整□标文档��请参见 metrics.md □获取所有可用指标、查询示例和监控设置。

最佳实践

1. 首先使用呼叫模拟器
    - 在进行配置更改之前
    - 在排除用户特定问题时
    - 理解培训的呼叫流
2. 使用 **HLR** 查询进行
    - 快速检查漫游状态
    - 验证来自 HLR 的呼叫转移
    - 测试 SS7 MAP 连接性
3. 记录发现
    - 截取模拟器结果的屏幕截图
    - 注意任何意外行为
    - 与团队分享结果□进行分析
4. 与日志进行比较

使用与失败呼叫相同的参数运行模拟
将模拟变量与实际呼叫日志进行比较
识别差异

5. 定期测试

每周使用模拟器进行抽查
测试每种呼叫类型（MT/MO/紧急）
验证 OCS 和 HLR 集成

故障排除工具

HLR 查询问题

工具显示 "SS7 MAP 被禁用"

检查 config/runtime.exs 中的 ss7_map.enabled
在配置更改后重启应用程序

工具显示超时错误

检查 SS7 MAP 网关是否可达
检查到 HLR 的网络连接
检查配置中的 ss7_map.timeout_ms

工具显示 "没有 VLR 号码"

用户离线或在 HLR 中不存在
对于关闭电源的用户是正常的
对于不存在的号码是正常的

呼叫模拟器问题

模拟器显示 "没有 Sh 数据"

用户未在 HSS 中配置
HSS 无法访问

- 检查 `diameter.sh_application` 配置

模拟器显示 "源 **IP** 未授权"

- IP 不在 `allowed_sbc_source_ips` 或 `allowed_cscf_ips` 中
- 使用 "强制处置" 来覆盖基于 IP 的认证

模拟器显示 "缺少必需参数"

- 除选项外，所有字段都是必需的
- 输入有效的电话号码
- 输入有效的 IP 地址

模拟器耗时过长

- 如果不测试 OCS，请取消选中 "跳过 OCS 授权"
- 如果不测试 HLR，请取消选中 "跳过 HLR 查询"
- 检查实际系统性能（Sh/HLR/OCS 响应时间）

支持

有关这些工具的问题：

1. 检查应用程序日志□获取错误
2. 验证配置（Sh、HLR、OCS）
3. 测试与外部系统的连接性
4. 联系支持团队并提供屏幕截图和错误消息

IMS 会议服务器 - 用户□南

概述

IMS 会议服务器提供符合 3GPP IMS 会议框架（RFC 4579、RFC 4575、TS 24.147）的多方会议能力。它使用户能够通过 IMS 应用服务器创建和管理音频/视频会议。

架构

IMS 会议服务器是 OmniTAS 的一个集成组件，提供：

- 会议工厂 **URI**：用于创建新会议的 SIP URI
- 会议焦点：管理会议状态和参与者
- 会议策略控制：强制执行参与者角色和权限
- 媒体混合：处理会议参与者的音频/视频混合

IMS 会议工厂架构

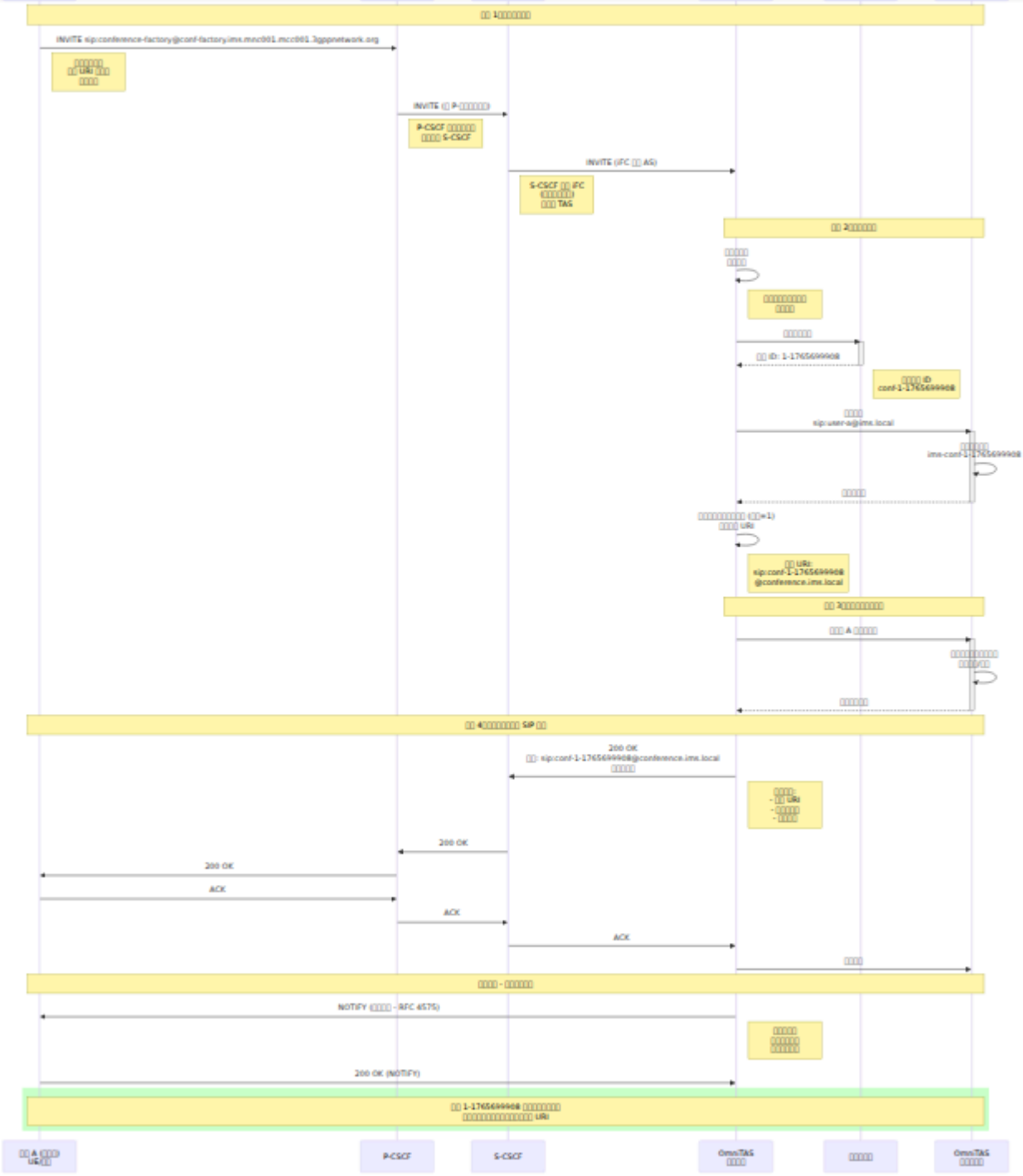
TAS 实现了 TS 24.147 和 RFC 4579 中定义的 3GPP 会议工厂模式：

Parse error on line 17: ... IMS 应用服务器] Factory[会议工厂
sip... ----------------------
^ Expecting 'SEMI', 'NEWLINE', 'SPACE', 'EOF', 'subgraph', 'end', 'acc_title', 'acc_descr', 'acc_descr_multiline_value', 'AMP', 'COLON', 'STYLE', 'LINKSTYLE', 'CLASSDEF', 'CLASS', 'CLICK', 'DOWN', 'DEFAULT', 'NUM', 'COMMA', 'NODE_STRING', 'BRKT', 'MINUS', 'MULT', 'UNICODE_TEXT', 'direction_tb', 'direction_bt', 'direction_rl', 'direction_lr', got 'LINK_ID'

□试

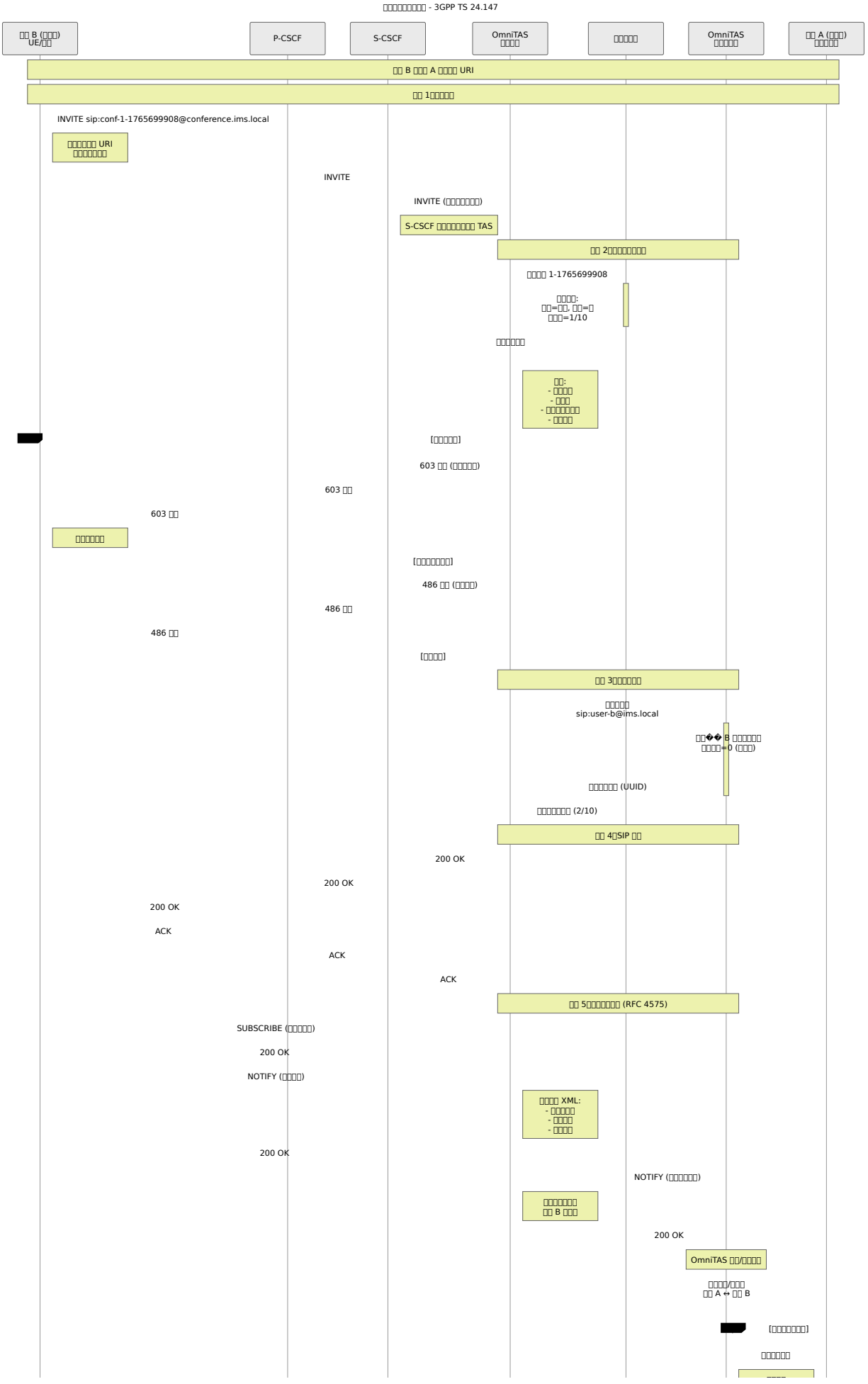
会议创建流程（RFC 4579 工厂模式）

该图显示用户如何通过会议工厂 URI 创建新会议：

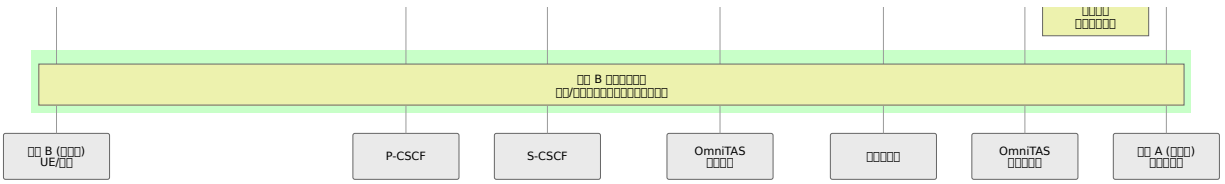


[redacted]

[redacted]

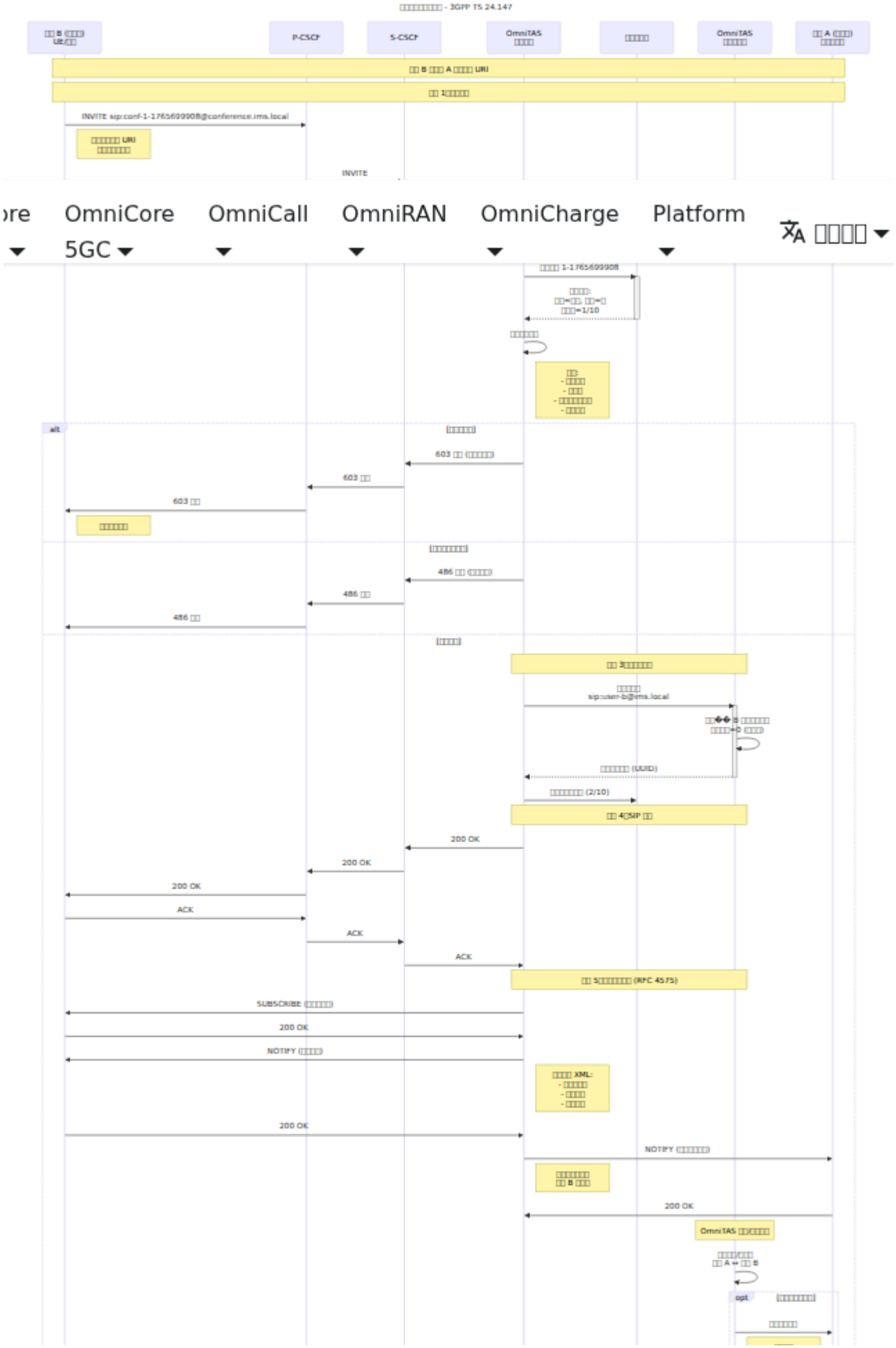


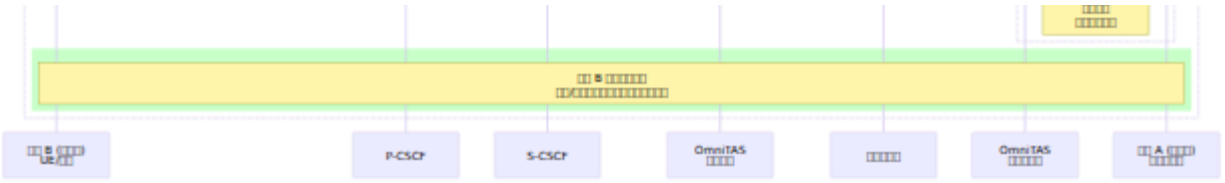
参与者加入会议流程 - 3GPP TS 24.147
用户 B (参与者)
UE/电话
P-CSCF
S-CSCF
OmniTAS
会议焦点
会议数据库
OmniTAS
媒体服务器
用户 A (创建者)
已在会议中
用户 B 从用户 A 收到会议 URI
阶段 1: 加入请求
INVITE sip:conf-1-1765699908@conference.ims.local
用户拨打会议 URI
或点击加入链接
INVITE
INVITE (路由到会议焦点)
S-CSCF 根据会议域路由到 TAS
阶段 2: 授权与策略检查
查询会议 1-1765699908
会议信息:
状态=活动, 锁定=否
参与者=1/10
检查会议策略
验证:
- 会议存在
- 未锁定
- 未达到最大容量
- 用户授权
[会议已锁定]
603 拒绝 (会议已锁定)
603 拒绝
603 拒绝
用户无法加入
[达到最大参与者]
486 繁忙 (会议已满)
486 繁忙
486 繁忙
[会议可用]
阶段 3: 添加参与者
添加参与者
sip:user-b@ims.local
将用户 B 添加到会议室
分配角色=0 (参与者)
参与者已添加 (UUID)
更新参与者计数 (2/10)
阶段 4: SIP 响应
200 OK
200 OK
200 OK
ACK
ACK
ACK
阶段 5: 会议状态通知 (RFC 4575)
SUBSCRIBE (会议事件包)
200 OK
NOTIFY (会议状态)
会议状态 XML:
- 参与者列表
- 会议属性
- 媒体状态
200 OK
NOTIFY (会议状态更新)
通知现有参与者
用户 B 已加入
200 OK
OmniTAS 音频/视频混合
混合音频/视频流
用户 A ⇔ 用户 B
[加入公告已启用]
播放加入音效



□□□□□ (RFC 4575)

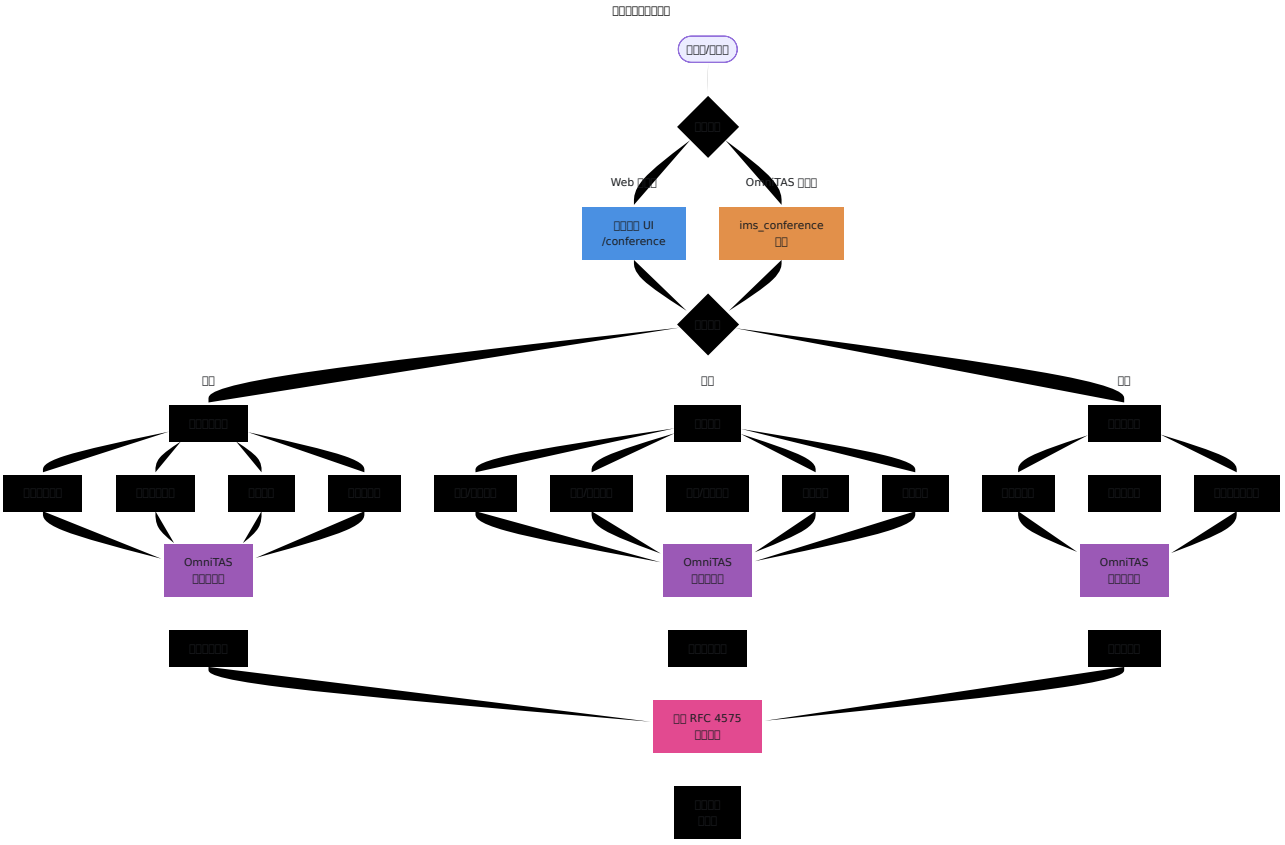
□□□□□□□□□□□□□□□□□□□□□□□□□□□□


□□□□□□□□□ - 3GPP TS 24.147
P-CSCF
S-CSCF
OmniTAS □□□□
□□□□□
OmniTAS □□□□□
□□ A (□□□)
□□ B □□□ A □□□□ URI
□□ 1□□□□□
INVITE sip:conf-1-1765699908@conference.ims.local
□□□□□□ URI □□□□□□□
INVITE
ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
□□□□
□□□□ 1-1765699908
□□□□:
□□=□□, □□=□
□□□=1/10
□□□□□□
□□:
- □□□□
- □□□
- □□□□□□□
- □□□□
alt
[□□□□□]
603 □□ (□□□□□)
603 □□
603 □□
□□□□□□
[□□□□□□□]
486 □□ (□□□□)
486 □□
486 □□
[□□□□]
□□ 3□□□□□□
□□□□
sip:user-b@ims.local
□□ B □□□□□□
□□□□ID (UUID)
□□□□□□□ (2/10)
□□ 4□SIP □□
200 OK
200 OK
200 OK
ACK
ACK
ACK
□□ 5□□□□□□ (RFC 4575)
SUBSCRIBE (□□□□□)
200 OK
NOTIFY (□□□□)
□□□□ XML:
- □□□□□
- □□□□
- □□□□
200 OK
NOTIFY (□□□□□)
□□□□□□
□□ B □□□
200 OK
OmniTAS □□/□□□□
□□□/□□□
□□ A ↔ □□ B
opt
[□□□□□□□]
□□□□□



□□□□□□

□□ Web UI □ OmniTAS □□□□□□□□□□



□□

Web □□

□□□ `/conference` □□□□□□□□□□“□□”□□□□□□□□□□

OmniTAS □□□

□□ `ims_conference` □□□ OmniTAS □□□□□□□□□□□

功能

会议管理界面

Web 界面提供对活动 IMS 会议的实时监控和管理：

统计仪表板

显示高层次的会议服务器统计信息：

- 活动会议：正在进行的会议总数
- 总参与者：所有会议的参与者总数
- 视频会议：启用视频的会议数量
- 锁定会议：锁定新参与者的会议数量

仪表板还显示服务器配置：

- 域：会议服务器域（例如，conference.ims.local）
- 工厂 **URI**：用于会议创建请求的 SIP URI
- **MNC/MCC**：移动网络代码和国家代码
- 接入网络：网络类型（例如，3GPP-E-UTRAN-FDD）
- 默认最大参与者：每个会议的最大参与者
- 默认启用视频：是否默认启用视频
- 启用录音：会议录音是否可用

会议列表

显示所有活动会议，包括：

- 会议 **ID**：会议的唯一标识符
- **URI**：会议的 SIP URI
- 参与者：当前参与者数量
- 创建者：会议创建者的电话号码/URI

点击任何会议□展开并查看详细信息。

会议详情

展开会议显示：

会议信息：

- ID 和 URI
- 房间名称
- 创建者身份
- 会议状态
- 参与者计数（当前/最大和最小）
- 视频状态（启用/禁用）
- 锁定状态（锁定/解锁）
- 录音状态（活动/非活动）

参与者列表：

- 每个参与者的 SIP URI
- 会话 UUID
- 参与者状态
- 角色（0 = 参与者，1 = 主持人）
- 视频状态

会议操作：

- 锁定/解锁会议
- 启用/禁用视频
- （通过 CLI 可用的其他操作）

自动刷新

界面每 5 秒自动刷新一次，显示实时会议状态。您可切换自动刷新开/关或使用“刷新”按钮手动刷新。

OmniTAS 控制台命令

所有会议管理操作均可通过 OmniTAS 控制台中的 `ims_conference` 命令访问。

命令语法

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

可用命令

list

列出所有活动 IMS 会议。

```
omnitas@server> ims_conference list
IMS 会议:
会议 ID         会议 URI                                    参与者 创建者
=================================================================
1-1765699908         sip:conf-1-1765699908@conference.ims.local 3
12025550151

总计: 1 个会议
```

info

显示特定会议的详细信息。

语法：`ims_conference info <conf_id>`

重要：使用会议 ID（例如，`1-1765699908`），而不是带前缀的会议名称。

```
omnitas@server> ims_conference info 1-1765699908
会议信息:
  ID: 1-1765699908
  URI: sip:conf-1-1765699908@conference.ims.local
  房间: ims-conf-1-1765699908
  创建者: 12025550151
  状态: 1
  参与者: 3/10 (最小: 2)
  视频: 启用
  锁定: 否
  录音: 非活动

参与者:
  - sip:1235;phone-
context=ims.mnc001.mcc001.3gppnetwork.org@ims.mnc001.mcc001.3gppnetwo
(342d50e0-9f67-4cc5-9179-4acae6f65f34)
      状态: 3, 角色: 0, 视频: 开
  - sip:1235;phone-
context=ims.mnc001.mcc001.3gppnetwork.org@ims.mnc001.mcc001.3gppnetwo
(bd98ca37-64fd-4618-b2db-aaba108c73e2)
      状态: 3, 角色: 0, 视频: 开
  - 12025550151 (6270da85-9b94-4285-8130-8769b11d0aa2)
      状态: 3, 角色: 1, 视频: 开
```

stats

显示整体会议服务器统计信息和配置。

```
omnitas@server> ims_conference stats
IMS 会议服务器统计信息:
================================
活动会议: 1
总参与者: 3
视频会议: 1
锁定会议: 0

配置:
  域: conference.ims.local
  工厂 URI: sip:conference-factory@conf-
factory.ims.mnc001.mcc001.3gppnetwork.org
  MNC/MCC: 001/001
  接入网络: 3GPP-E-UTRAN-FDD
  默认最大参与者: 10
  允许匿名: 是
  默认启用视频: 是
  启用录音: 是
  公告: 加入=开，离开=开，计数=开
```

create

创建一个新会议。

语法: `ims_conference create <creator_uri>`

```
omnitas@server> ims_conference create sip:12025550151@ims.local
会议已创建: 1-1765699909
会议 URI: sip:conf-1-1765699909@conference.ims.local
```

destroy

终止一个会议并断开所有参与者。

语法: `ims_conference destroy <conf_id>`

```
omnitas@server> ims_conference destroy 1-1765699908
会议 1-1765699908 已销毁
```

add

将参与者添加到现有会议。

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

```
omnitas@server> ims_conference add 1-1765699908
sip:12025550152@ims.local
将参与者 sip:12025550152@ims.local 添加到会议 1-1765699908
```

remove

从会议中移除参与者。

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

注意: 使用 `info` 命令输出中的参与者会话 UUID。

```
omnitas@server> ims_conference remove 1-1765699908 342d50e0-9f67-
4cc5-9179-4acae6f65f34
已从会议 1-1765699908 中移除参与者
```

lock

锁定会议以防止新参与者加入。

语法: `ims_conference lock <conf_id>`

```
omnitas@server> ims_conference lock 1-1765699908
会议 1-1765699908 已锁定
```

unlock

解锁会议以允许新参与者。

语法: `ims_conference unlock <conf_id>`

```
omnitas@server> ims_conference unlock 1-1765699908
会议 1-1765699908 已解锁
```

video

控制会议的视频。

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

```
omnitas@server> ims_conference video 1-1765699908 on
会议 1-1765699908 的视频已启用

omnitas@server> ims_conference video 1-1765699908 off
会议 1-1765699908 的视频已禁用
```

record

控制会议录音。

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

```
omnitas@server> ims_conference record 1-1765699908 start
会议 1-1765699908 的录音已开始

omnitas@server> ims_conference record 1-1765699908 stop
会议 1-1765699908 的录音已停止
```

announce

向所有会议参与者播放公告。

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

```
omnitas@server> ims_conference announce 1-1765699908 "本次会议将在 5 分钟后结束"
公告已发送到会议 1-1765699908
```

subscribers

列出当前在会议中的所有订阅者（`info` 的替代视图）。

语法：`ims_conference subscribers <conf_id>`

```
omnitas@server> ims_conference subscribers 1-1765699908
会议 1-1765699908 中的订阅者:
  - sip:1235;phone-
context=ims.mnc001.mcc001.3gppnetwork.org@ims.mnc001.mcc001.3gppnetwo
  - 12025550151
```

会议状态

会议和参与者具有数字状态值：

会议状态

- **0**：初始化
- **1**：活动
- **2**：终止中
- **3**：已终止

参与者状态

- **0**：已邀请
- **1**：拨号中
- **2**：警报中
- **3**：已连接
- **4**：断开中
- **5**：已断开

参与者角色

- **0**：普通参与者
- **1**：主持人/创建者

用例

监控活动会议

场景：操作团队需要查看有多少会议处于活动状态

步骤：

1. 打开会议管理界面（`/conference`）
2. 查看统计仪表板□获取高层次指标
3. 审查会议列表□获取特定会议
4. 使用自动刷新实时监控

CLI 替代方案：

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

排查会议问题

场景：用户报告他们无法加入会议

步骤：

1. 从用户那里获取会议 ID
2. 运行 `ims_conference info <conf_id>` 检查会议状态
3. 检查会议是否已锁定（锁定：是）
4. 检查当前参与者计数与最大值
5. 审查参与者列表□查找任何连接问题
6. 检查 OmniTAS 日志□获取 SIP 邀请失败的信息

常见问题：

- 会议已锁定：`ims_conference unlock <conf_id>`
- 达到最大参与者：检查 default_max_participants 配置
- 网络问题：检查 SIP 连接性和防火墙规则

管理会议带宽

场景：在网络拥塞期间需要减少带宽使用

步骤：

1. 确定启用视频的会议
2. 对于非关键会议，禁用视频：

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

3. 监控带宽使用情况
4. 拥塞解除后重新启用视频

处理干扰参与者

场景：会议中有参与者干扰

步骤：

1. 获取会议 ID 和参与者的会话 UUID
2. 移除参与者：

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

3. 锁定会议□防止他们重新加入：

```
ims_conference lock <conf_id>
```

4. 如有需要，手���添加合法参与者：

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

录音重要会议

场景：需要录音□满足合规或文档要求

步骤：

1. 确定会议 ID
2. 开始录音：

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

3. 监控录音是否处于活动状态（录音：在 `info` 输出中为活动）
4. 完成后停止录音：

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

5. 录音文件存储在 OmniTAS 录音目录中

紧急终止会议

场景：需要立即终止会议

步骤：

1. 可选地向参与者公告：

```
ims_conference announce <conf_id> "本次会议正在终止"
```

2. 等待几秒钟播放公告
3. 销毁会议：

```
ims_conference destroy <conf_id>
```

4. 所有参与者将立即断开连接

与 IMS 网络的集成

会议创建流程

1. 用户向会议工厂 URI 发送 SIP INVITE
2. IMS 应用服务器接收请求
3. 会议服务器创建新的会议实例

4. 生成会议 ID 和 URI
5. 根据创建者初始化会议策略
6. 将创建者添加为第一个参与者，角色为主持人
7. 会议 URI 返回给创建者
8. 其他参与者现在可通过会议 URI 加入

参与者角色

主持人（角色：**1**）

- 可锁定/解锁会议
- 可移除其他参与者
- 可控制视频设置
- 接收会议通知

参与者（角色：**0**）

- 可加入/离开会议
- 可发言和倾听
- 可启用/禁用自己的视频
- 受会议政策约束

3GPP 合规性

IMS 会议服务器实现了关键的 3GPP 规范：

- **TS 24.147**：使用 IP 多媒体 (IM) 核心网络 (CN) 子系统的会议
- **RFC 4579**：会话发起协议 (SIP) 呼叫控制 - 用户代理的会议
- **RFC 4575**：用于会议状态的会话发起协议 (SIP) 事件包
- **RFC 5239**：集中会议的框架

网络元素集成

- **P-CSCF**：处理来自 UE 的初始 SIP 信令
- **S-CSCF**：将会议请求路由到应用服务器
- **OmniTAS**：托管会议服务器功能并提供媒体混合

HSS：提供用户身份验证和授权

配置

会议服务器配置通过 OmniTAS 配置文件管理：

关键参数：

domain：会议服务器域
factory_uri：用于会议创建的 SIP URI
mnc_mcc：移动网络标识符
access_network：网络接入类型
default_max_participants：每个会议的默认最大参与者 allow_anonymous：是否允许匿名参与者 video_by_default：新会议的默认视频设置 recording_enabled：录音功能是否可用
announce_join：参与者加入时播放音效
announce_leave：参与者离开时播放音效
announce_count：公告参与者计数

最佳实践

容量规划

监控活动会议数量和参与者数量
计划高峰使用（例如，工作时间）
为媒体混合分配足够的 CPU/内存
考虑视频与仅音频的带宽管理

安全性

确保会议 URI 不易猜测
对于私密会议使用会议锁定

- 监控未授权访问尝试
- 实施最大参与者限制
- 审查会议录音访问控制

操作监控

- 设置会议服务器错误的警报
- 监控会议创建/销毁率
- 跟踪平均会议持续时间
- 审查参与者连接失败
- 监控媒体质量指标

有关详细□标文档：请参见 metrics.md □获取：

- RTP/RTCP 媒体质量指标（端口 9093）
- 活动呼叫和会话指标（端口 9090）
- 系统和 Erlang VM 指标（端口 9094）
- Prometheus 查询示例

故障排除

- 检查 OmniTAS 日志□获取与会议相关的错误
- 验证会议服务器配置
- 审查网络连接性和防火墙规则
- 联系 Omnitouch 支持，提供会议 ID 和时间戳

ANSSI R226 拦截合规文档
文档目的：本文档提供了根据法国刑法第 R226-3 和 R226-7 条款，针对 OmniTAS IMS 应用服务器所需的 ANSSI R226 授权的技术规格。

分类：监管合规文档

目标机构：法国国家信息系统安全局（ANSSI）

法规： R226 - 通信隐私保护和合法拦截

1. 详细技术规格

1.1 商业技术数据表

产品名称： OmniTAS IMS 应用服务器
产品类型：电信应用服务器 (TAS)
主要功能： IMS (IP 多媒体子系统) 呼叫处理和会话管理
网络协议： SIP, Diameter, HTTP/HTTPS, SS7/MAP 部署模型：本地服务器应用

核心能力

呼叫处理：

会话发起协议 (SIP) 代理和 B2BUA 功能
IMS 初始过滤标准 (iFC) 处理
会话路由和呼叫控制
紧急呼叫处理 (E.164 PSAP 路由)
呼叫详细记录 (CDR) 生成

网络接口：

北向： IMS S-CSCF 接口 (SIP over TCP/UDP)
南向： SBC/网关接口 (SIP 中继)

- **Diameter**：Sh (用户数据), Ro (在线计费)
- **SS7**：MAP 网关接口用于 HLR/MSC 互通
- **HTTP/HTTPS**：外部服务集成 (SMS, TTS, MAP 网关)

存储和处理：

- 实时会话状态管理
- CDR 存储和检索
- 用户注册数据库 (Sofia SIP)
- 配置数据库 (SQLite)

1.2 拦截能力

TAS 实现了 ETSI TS 103 221 拦截接口 (X1, X2, X3)。TAS 作为网络元素和拦截点。执法监控设施通过 X1 配置目标。TAS 通过 X2 □ 3GPP TS 33.128 xIRI 形式提供 SIP 信令。TAS 通过 X3 □ RTP/IP 数据包形式提供语音媒体。有关接口角色和完整参数参考，请参见 合法拦截 (X1/X2/X3)。

1.2.1 信号获取

SIP 信令捕获：

- OmniTAS 处理 IMS 用户和外部网络之间的所有 SIP 信令消息
- 完全访问 SIP 头，包括：
  - 主叫方标识 (From, P-Asserted-Identity)
  - 被叫方标识 (To, Request-URI)
  - 联系 URI 和网络位置
  - 呼叫路由信息
  - 会话描述 (SDP)，包括媒体编解码器和端点

呼叫元数据获取：

- 完整的呼叫详细记录 (CDR) 存储在数据库中，包含：
  - 时间戳 (开始、应答、结束时间)
  - 主叫和被叫标识符 (MSISDN, IMSI, SIP URI)
  - 呼叫方向 (移动发起/终止)
  - 呼叫结果 (应答、忙、失败等)
  - 持续时间和计费信息

- 网络位置数据（可用时的基站信息）

会话录音接口 **(SIPREC)**:

- 支持 SIPREC 协议□进行合法拦截
- 能够将 SIP 信令复制到外部录音服务器
- 可配置的会话录音策略
- 许可控制： SIPREC 功能需要明确的许可授权
- 访问控制： SIPREC 配置仅限于授权管理员

1.2.2 媒体处理能力

媒体平面:

- 具有 RTP 媒体中继能力的 B2BUA
- RTP 流通过服务器传递
- 访问媒体流□进行拦截
- SDP 解析□获取媒体端点和编解码器信息

信令平面:

- SIP 消息解析和分析
- Diameter 消息编码/解码 (Sh, Ro 接口)
- HTTP/HTTPS 请求/响应处理

1.2.3 分析能力

实时呼叫监控:

- Web UI 仪表板显示活动呼叫，包括:
  - 呼叫状态（尝试、响铃、活动、终止）
  - 主叫/被叫信息
  - 呼叫持续时间
  - 媒体编解码器信息
  - 网络端点

历史分析:

- 可按□下条件查询 CDR 数据库：
  - 时间范围
  - 主叫/被叫号码
  - 呼叫类型（语音、紧急等）
  - 呼叫结果/处置
  - 持续时间阈值

用户跟踪：

- 活跃注册监控
- 通过□下方式跟踪用户位置：
  - IMS 注册联系 URI
  - P-Access-Network-Info 头（基站标识）
  - IP 地址和端口信息
- 历史注册记录

网络分析：

- 呼叫量指标（Prometheus 集成）
- 网关状态和连接性
- Diameter 对等连接
- 系统性能指标

有关全面□标文档：请参见 metrics.md □获取详细的监控、警报和可观察性配置。

位置智能：

- 基站数据库集成
- E.164 号码到地理位置映射（北美编号计划）
- 紧急服务路由（PSAP 映射）

1.3 反制能力

1.3.1 隐私保护机制

通信机密性：

- Diameter TLS 传输安全
- Web 界面和 API 的 HTTPS
- 静态数据加密（可配置）

访问控制：

- Web UI 的基于角色的访问控制（RBAC）
- 使用 SHA-512 和盐进行密码哈希（65,532 次迭代）

审计日志：

- 完整的管理操作审计跟踪
- 配置更改日志
- 身份验证事件日志
- 防篡改日志存储

1.3.2 反拦截功能

安全通信：

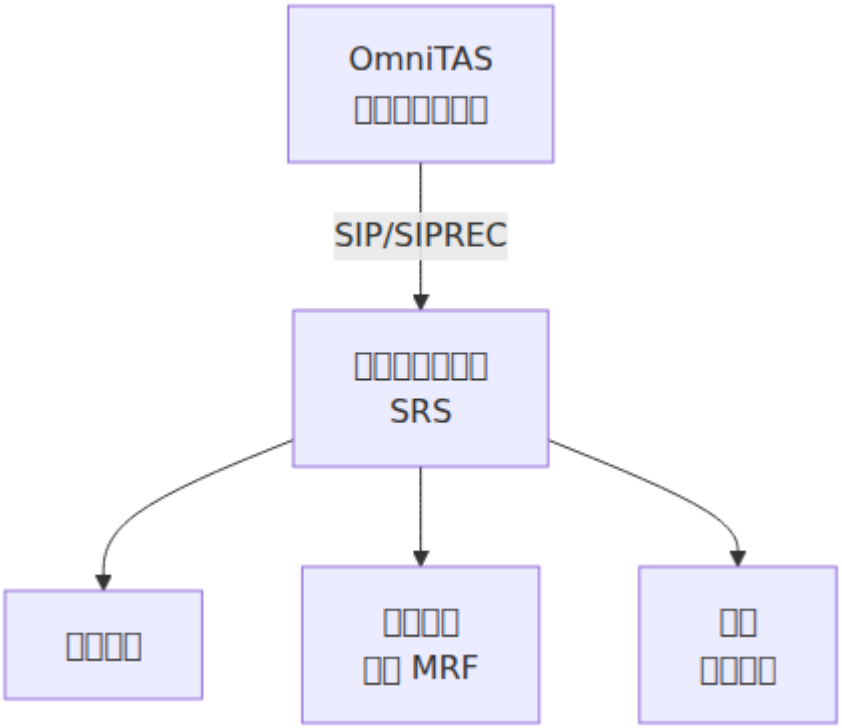
- 外部接口的强制 TLS（可配置）
- 基于证书的身份验证
- 完美前向保密（PFS）密码套件

数据保护：

- 自动 CDR 保留政策
- 安全数据删除能力
- 数据库访问控制
- 网络分段支持（单独的管理/信令/媒体网络）

系统加固：

- 启动参数保护
- 完整性验证机制
- 最小攻击面（仅启用所需��务）



1.4 合法拦截的技术架构

合法拦截集成点

1. SIPREC 接口 (会话录音协议 - RFC 7866):

2. CDR 导出接口:

- CDR 导出到外部系统
- 标准格式 (CSV, JSON)
- 安全传输 (HTTPS)

3. 直接数据库访问:

- 授权系统的只读数据库凭据
- 对 CDR 表的 SQL 查询访问
- 用户注册数据访问
- 审计日志访问

4. API 集成:

- 用于呼叫监控的 RESTful API
- 实时活动呼叫查询

- 历史 CDR 检索
- 用户注册状态

拦截触发机制

基于目标的拦截：

- 用户标识符匹配 (MSISDN, IMSI, SIP URI)
- 应用逻辑中的可配置拦截规则
- 基于主叫/被叫身份的 SIPREC 会话分叉

基于事件的拦截：

- 紧急呼叫检测和录音
- 特定目的地号码监控
- 基于地理区域的触发（基站位置）

基于时间的拦截：

- 计划录音窗口
- 保留期限强制执行
- 拦截令的自动到期

---

2. 加密和密码分析能力

2.1 密码学能力概述

OmniTAS IMS 应用服务器实现了加密机制，□保护通信和敏感数据。此部分记录了所有符合 ANSSI 要求的加密能力。

2.2 传输层加密

2.2.1 TLS/SSL 实现

支持的协议：

- TLS 1.2 (RFC 5246)

TLS 1.3 (RFC 8446)
SSL 2.0/3.0：禁用 (已知漏洞)
TLS 1.0/1.1：弃用 (可配置，默认禁用)

密码套件 (可配置优先级列表)：

首选 - TLS 1.3：

TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256

支持 - 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

安全特性：

强制完美前向保密 (PFS)
强 Diffie-Hellman 组 (2048 位最小)
椭圆曲线密码学：NIST P-256, P-384, P-521
服务器名称指示 (SNI) 支持
OCSP stapling 用于证书验证

证书管理：

支持 X.509 证书
RSA 密钥大小：2048 位最小，4096 位推荐
支持 ECDSA (P-256, P-384)
证书链验证
CRL 和 OCSP 撤销检查
自签名证书 (仅限开发)
外部 CA 集成

应用：

Web UI 和 API 访问的 HTTPS
通过 TLS 的 Diameter

2.3 静态数据加密

2.3.1 数据库加密

SQLite 加密：

SQLCipher 集成支持
AES-256 加密
敏感数据的加密存储 (CDR, 用户数据)

2.3.2 文件系统加密

敏感数据存储：

CDR 文件：AES-256 加密 (可选)
配置文件：凭据的加密存储
私钥：加密密钥库 (PKCS#12, 带密码的 PEM)
日志文件：归档日志的加密支持

密钥存储：

带密码保护的文件密钥库
安全密钥轮换机制

2.4 身份验证和密码加密

2.4.1 密码哈希

算法： SHA-512 带盐
配置：

随机生成的盐 (128 位最小)
65,532 次迭代轮次 (可配置)

- 盐与哈希一起存储
- 抵抗彩虹表攻击

存储格式:

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

应用:

- Web UI 用户身份验证
- API 令牌生成
- 管理员密码存储
- 数据库用户凭据

2.4.2 SSH 密钥身份验证

支持的密钥类型:

- RSA: 1024-4096 位 (推荐 2048 位最小)
- DSA: 1024-4096 位 (弃用, 推荐 RSA)
- ECDSA: P-256, P-384, P-521 曲线
- Ed25519: 256 位 (新部署的首选)

密钥管理:

- 外部密钥生成支持
- 用于客户端身份验证的公钥导入
- 服务器主机密钥管理
- 单个密钥撤销
- 密钥轮换程序

SSH 协议:

- 仅支持 SSH-2 协议 (禁用 SSH-1)
- 强大的 MAC 算法 (HMAC-SHA2-256, HMAC-SHA2-512)
- 密钥交换: curve25519-sha256, ecdh-sha2-nistp256, diffie-hellman-group14-sha256

2.5 Diameter 协议安全

2.5.1 Diameter 安全机制

传输安全：

Diameter 对等连接的 TCP 上的 TLS
互证书身份验证

应用层安全：

通过 Origin-Host/Origin-Realm 验证进行对等身份验证
共享密钥配置 (遗留，弃用)
敏感数据的 AVP (属性-值对) 加密
使用 CMS (加密消息语法) 的端到端安全

2.6 SIP 身份机制

P-Asserted-Identity：

受信网络声明
身份验证和转换
隐私头支持

注意：用户身份验证由 IMS 核心 (P-CSCF/S-CSCF) 执行，而不是 TAS。

2.7 密码分析和安全评估能力

2.7.1 协议分析工具

内置调试能力：

SIP 消息跟踪，完整头/体捕获
Diameter 消息日志 (AVP 解码)
TLS 握手调试
证书链验证日志

外部集成：

- Wireshark/tcpdump 数据包捕获支持
- SSLKEYLOGFILE 导出□进行 TLS 解密（仅限开发）
- PCAP 导出□进行离线分析

2.7.2 漏洞评估考虑

已知的密码学弱点：

- SIP 摘要中的遗留 MD5 支持（为向后兼容而维护）
- 可配置的弱密码套件（默认禁用）
- 自签名证书支持（仅限开发/测试）

安全测试：

- 建议定期进行安全审计
- 支持渗透测试
- 密码套件强度验证
- 证书到期监控

2.8 密钥管理基础设施

2.8.1 密钥生成

内部密钥生成：

- RSA 密钥生成：OpenSSL 库（符合 FIPS 140-2 标准的算法）
- 随机数生成：/dev/urandom (Linux 内核 CSPRNG)
- 熵源：硬件 RNG，系统熵池

2.8.2 密钥存储和保护

私钥存储：

- 权限受限的文件系统 (0600)
- 带密码的加密 PEM 格式
- 密钥轮换时的安全删除

密钥备份：

加密备份程序
分割密钥恢复机制
安全密钥托管 (如法规要求)

2.8.3 密钥分发

证书分发：

通过 Web UI 手动导入
通过 API 自动配置
ACME 协议支持 (Let's Encrypt，未来增强)

对称密钥分发：

Diameter 对等方的带外密钥交换
TLS 中的 Diffie-Hellman 密钥协商
不明文传输密钥

2.9 合规性和标准

密码学标准合规性：

NIST SP 800-52：TLS 指南
NIST SP 800-131A：密码算法过渡 RFC 7525：TLS 建议
ETSI TS 133 310：IMS 网络安全 3GPP TS 33.203：IMS 访问安全

法国密码学法规：

密码手段声明 (如适用)
ANSSI 密码产品认证 (如要求)
无出口限制的密码学 (所有标准算法)

2.10 密码分析抵抗

2.10.1 设计原则

抵御密码分析：

- 不使用自定义/专有的密码算法
- 仅使用行业标准、经过同行评审的算法
- 定期更新密码库确保安全
- 弃用弱算法

2.10.2 操作安全

密钥轮换：

- TLS 证书续订（建议每年）
- 会话密钥轮换（每会话 TLS）
- 密码到期政策（可配置）

监控和检测：

- 失败的身份验证尝试日志
- 证书到期警报
- 密码套件协商日志
- 加密失败的异常检测

---

3. 拦截控制和授权

3.1 合法拦截的访问控制

管理授权：

- 合法拦截功能需要管理员级别的权限
- SIPREC 配置访问：仅限超级管理员角色
- CDR 访问：可配置的基于角色的权限
- 所有与拦截相关的操作的审计日志

法律框架集成：

- 拦截令跟踪（外部系统集成）

目标标识符授权列表
有时间限制的拦截激活
在令到期时自动停用

3.2 数据保留和隐私

保留政策：

CDR 保留：可配置 (默认 90 天，法规要求 1 年)
注册日志：可配置保留
审计日志：至少 1 年保留
自动清除过期数据

隐私保护：

最小数据收集原则
目的限制 (电信服务提供)
访问日志记录和监控

3.3 执法交接接口

标准合法拦截接口：

ETSI LI (合法拦截) 接口支持 (通过外部调解设备) SIPREC 到 LI 网关集成
X1, X2, X3 接口支持 (外部系统)

交付格式：

IRI (拦截相关信息)：CDR 元数据
CC (通信内容)：SIP 信令 + 媒体 (通过 MRF) 结构化报告：XML, JSON 格式

4. 系统安全和完整性

4.1 启动安全

安全启动机制：

启动参数保护 (ANSSI R226 要求)
配置完整性验证
启动时的篡改检测
安全配置加载

4.2 网络安全

网络安全：

最小暴露端口 (仅 SIP, Diameter, HTTPS)
基于端口的访问控制
IP 白名单/黑名单

4.3 入侵检测

监控能力：

失败的身份验证监控
异常呼叫模式检测
异常 Diameter 流量检测
安全事件警报 (SIEM 集成)

5. 文档参考

5.1 技术手册

项目库中提供的文档：

- **README.md**：系统概述、架构和操作特性
- **docs/deployment_guide.md**：部署说明（如可用）
- **docs/configuration.md**：配置参考（如可用）

5.2 安全认证

- 渗透测试报告：［根据请求提供］
- 安全审计报告：［根据请求提供］
- 密码模块验证： OpenSSL FIPS 140-2 合规性

5.3 合规文档

- **ANSSI R226** 授权请求：本文档
- 合法拦截合规性：根据法国电信法规的���求

---

6. 联系信息

供应商/运营商信息：

- 公司名称: Omnitouch Network Services Pty Ltd
- 地址: PO BOX 296, QUINNS ROCKS WA 6030, AUSTRALIA
- 联系人：合规团队
- 邮箱: compliance@omnitouch.com.au

技术安全联系人：

- 姓名：合规团队
- 邮箱: compliance@omnitouch.com.au

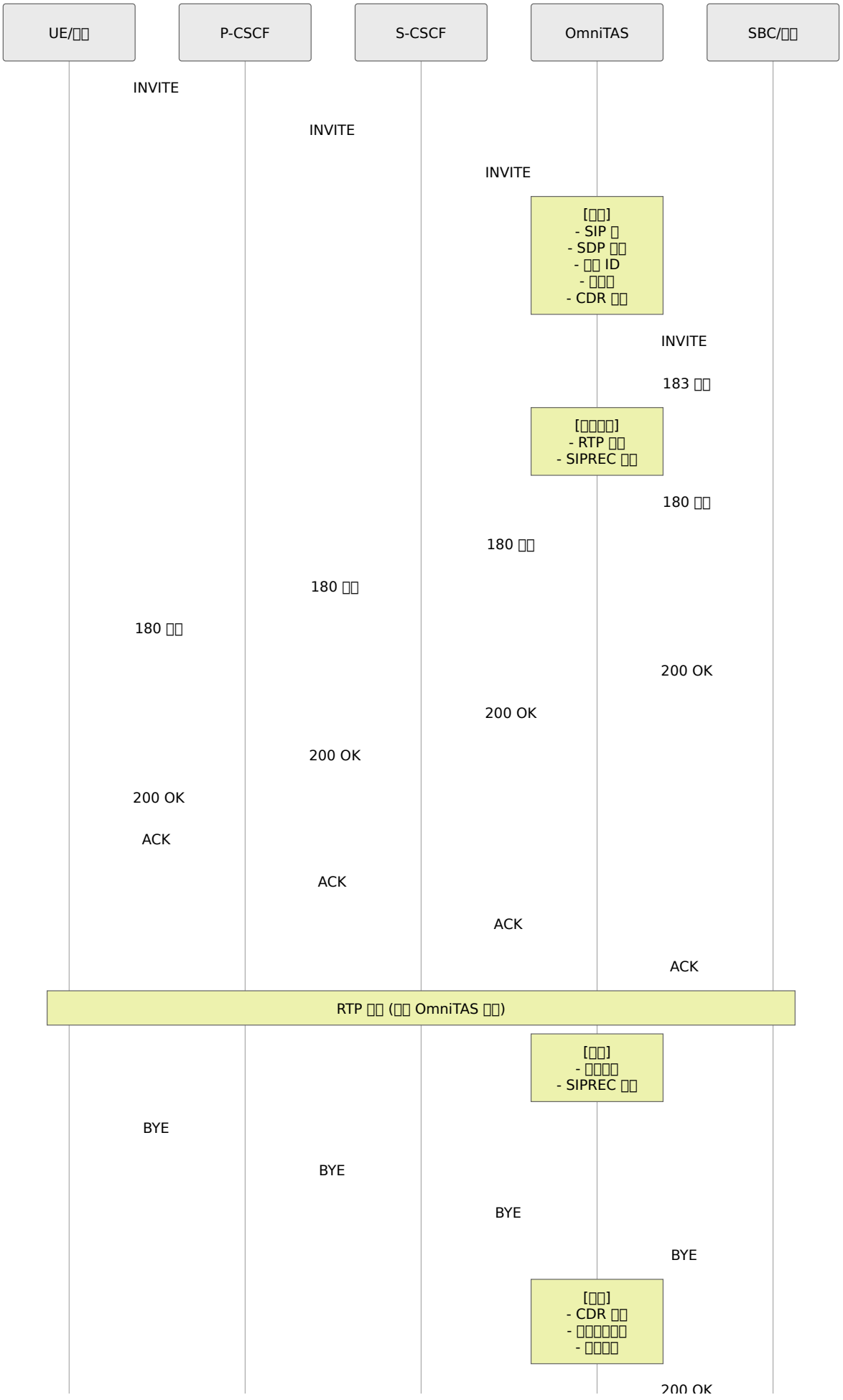
法律/合规联系人：

- 姓名：合规团队
- 邮箱: compliance@omnitouch.com.au

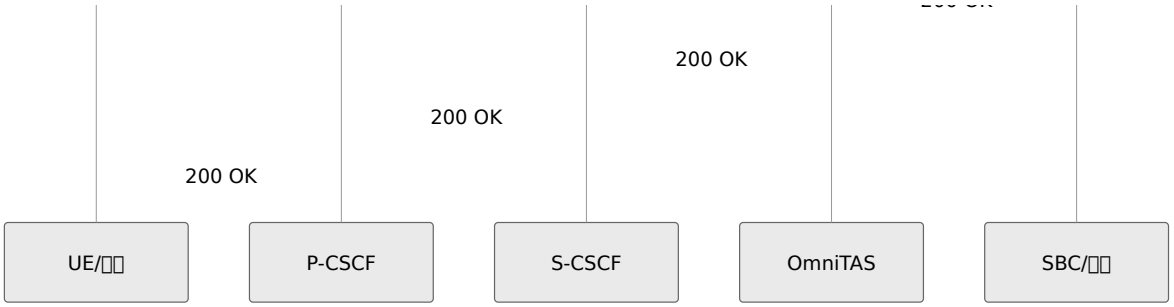
附录

附录 A: SIP 消息流示例

A.1 移动发起呼叫流与拦截点

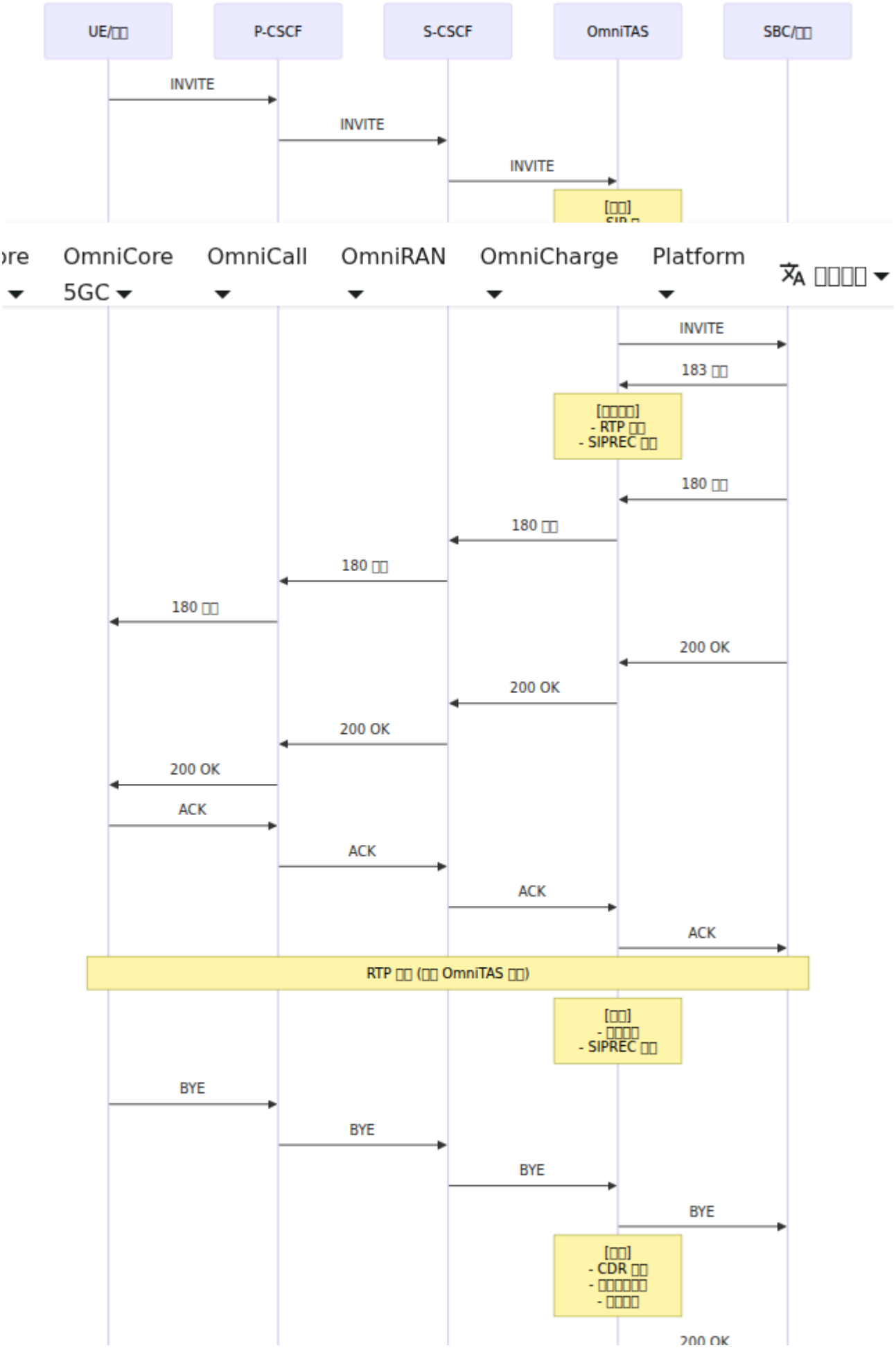


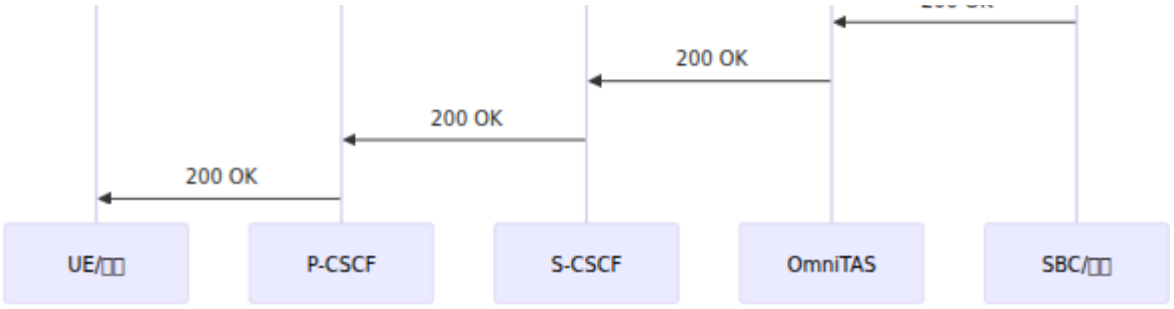
UE/手机
P-CSCF
S-CSCF
OmniTAS
SBC/网关
INVITE
INVITE
INVITE
【拦截】
- SIP 头
- SDP 信息
- 主叫 ID
- 被叫方
- CDR 创建
INVITE
183 进展
【媒体中继】
- RTP 锚定
- SIPREC 分叉
180 响铃
180 响铃
180 响铃
180 响铃
200 OK
200 OK
200 OK
200 OK
ACK
ACK
ACK
ACK
RTP 媒体 (通过 OmniTAS 中继)
【拦截】
- 媒体副本
- SIPREC 发送
BYE
BYE
BYE
BYE
【拦截】
- CDR 更新
- 呼叫结束时间
- 持续时间
200 OK



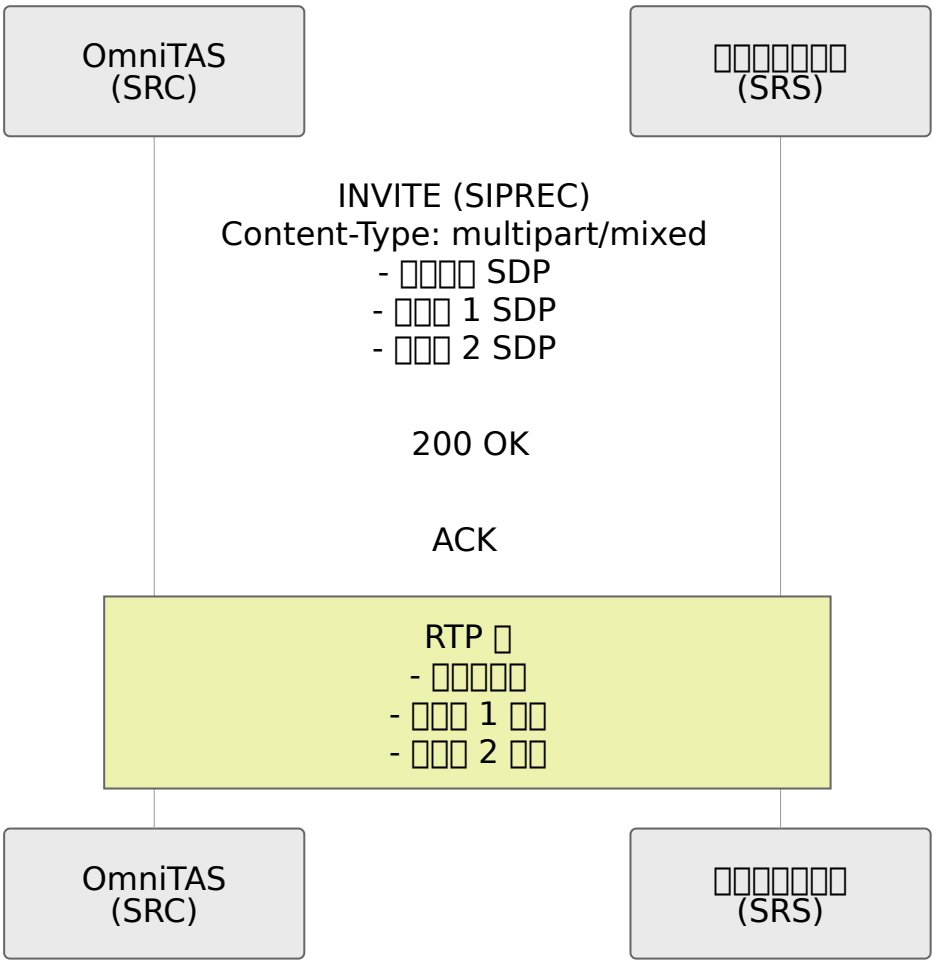
图例：［拦截］= 捕获合法拦截数据的点

A.2 紧急呼叫与位置跟踪


UE/□□
P-CSCF
S-CSCF
OmniTAS
SBC/□□
INVITE
INVITE
INVITE
[□□]
SIP □
ore
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
□□□□□
INVITE
183 □□
[□□□□□]
- RTP □□
- SIPREC □□
180 □□
180 □□
180 □□
180 □□
200 OK
200 OK
200 OK
200 OK
ACK
ACK
ACK
ACK
RTP □□ (□□ OmniTAS □□)
[□□]
- □□□□□
- SIPREC □□
BYE
BYE
BYE
BYE
[□□]
- CDR □□
- □□□□□□□
- □□□□□
200 OK



A.3 SIPREC □□□□□□



□□ B□CDR □□

OmniTAS □□□□□ `/etc/freeswitch/db/cdr.db` ◆◆ SQLite □□□□□□□□□□□□□□ (FreeSWITCH CDR □□)□

B.1 □□□□□□□CDR□□

字段名称	类型	描述	拦截相关性
`uuid`	TEXT	唯一通话标识符	会话关联
`caller_id_number`	TEXT	主叫方号码（MSISDN）	目标追踪的主要标识符
`caller_id_name`	TEXT	主叫方显示名称	身份验证
`destination_number`	TEXT	被叫方号码	目标目的地追踪
`start_stamp`	DATETIME	通话开始时间戳	事件时间线
`answer_stamp`	DATETIME	通话接听时间戳	通话建立时间
`end_stamp`	DATETIME	通话结束时间戳	会话持续时间计算
`duration`	INTEGER	通话总时长（秒）	会话长度
`billsec`	INTEGER	可计费秒数（接听时间）	实际通话时长
`hangup_cause`	TEXT	通话终止原因	通话结果分析
`sip_hangup_disposition`	TEXT	SIP 终止详情	协议级别终止
`network_addr`	TEXT	网络IP地址	源位置追踪
`sip_from_user`	TEXT	SIP From头部用户部分	原始SIP身份
`sip_to_user`	TEXT	SIP To头部用户部分	SIP目的地
`sip_call_id`	TEXT	SIP Call-ID头部	**SIP**会话关联

B.2 合法拦截的CDR查询示例

按目标号码查询通话：

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

在时间窗口内查询通话：

```
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;
```

导出给执法部门的**CSV**：

```
.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保留

- 默认保留：可配置（通常为90天到1年）
- 自动清除：支持
- 手动导出：通过Web UI在`/cdr`或API
- 格式：SQLite数据库，可导出为CSV/JSON

附录C：SIPREC配置示例

SIPREC（会话发起协议录音）使OmniTAS能够将通话信令和媒体发送到外部会话录音服务器□进行合法拦截。

C.1 SIPREC架构

C.2 触发SIPREC录音

录音可□基于□下条件触发:

基于目标:

- 主叫电话号码 (caller_id_number)
- 被叫电话号码 (destination_number)
- SIP URI匹配

基于事件:

- 所有紧急呼叫 (911、112等)
- 到/从特定目的地的通话
- 基于时间窗口的录音

地理位置:

- 基站位置 (通过P-Access-Network-Info头部)
- IP地址范围

C.3 SIPREC会话内容

SIPREC会话发送到SRS:

信令元数据:

- 完整的SIP头部 (From、To、P-Asserted-Identity)
- Call-ID和会话标识符
- 时间戳 (开始、接听、结束)
- 主叫/被叫信息

媒体流:

- 参与者1 RTP流（主叫音频）
- 参与者2 RTP流（被叫音频）
- 编解码器信息
- DTMF音调

C.4 与执法部门的集成

会话录音服务器提供：

- **X1**接口：管理功能（授权管理）
- **X2**接口：拦截相关信息（IRI） - 通话元数据
- **X3**接口：通信内容（CC） - 实际媒体

OmniTAS作为会话录音客户端（SRC），并通过标准化接口将IRI和CC交付给SRS，便转交给执法部门。

附录D：加配置南

D.1 证书生成

生成**TLS**证书：

```
# 生成私钥
openssl genrsa -out server.key 4096

# 生成证书签名请求
openssl req -new -key server.key -out server.csr

# 自签名证书（用于测试）
openssl x509 -req -days 365 -in server.csr -signkey server.key -
out server.crt

# 生产环境：从受信任的CA获取证书
```

注意： SIP信令到/从IMS不使用TLS。SIP通信为未加密的TCP/UDP。

D.2 Web UI的HTTPS配置

API/Web服务器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配置

SIP接口使用未加密的TCP/UDP传输。无需TLS配置。

FreeSWITCH SIP配置文件:

```
<!-- SIP配置文件仅使用TCP/UDP -->
<profile name="external">
  <settings>
    <param name="sip-port" value="5060"/>
    <param name="context" value="public"/>
  </settings>
</profile>
```

D.4 Diameter TLS配置

Diameter对等体TLS:

```
# 为Diameter连接启用TLS
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 数据库加密

使用**SQLCipher**的**SQLite**加密:

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

注意：数据库加密是可选的。出于合法拦截目的，物理访问控制和数据库访问日志记录可能已足够。

D.6 密码安全配置

密码哈希使用SHA-512和盐自动配置:

```
# 默认密码哈希配置
config :pbkdf2_elixir,
  rounds: 65_532,
  salt_len: 16
```

无需额外配置 - 默认安全。

附录E：术语表

监管和标准机构

ANSSI：法国国家信息系统安全局
ETSI：欧洲电信标准协会
3GPP：第三代合作伙伴计划 - 移动通信标准组织
IETF：互联网工程任务组 - 互联网标准机构

IMS网络组件

IMS： IP多媒体子系统 - 用于多媒体服务的全IP网络架构
CSCF：呼叫会话控制功能 - IMS核心中的SIP服务器 P-CSCF：代理-CSCF - UE的第一个接触点，SIP代理 I-CSCF：查询-CSCF - 运营商网络的入口点 S-CSCF：服务-CSCF - 会话控制和服务触发
HSS：家庭用户服务器 - 用户数据库
TAS：电话/电信应用服务器 - 服务逻辑执行

协议和信令

SIP：会话发起协议（RFC 3261） - 语音/视频通话的信令协议 SDP：会话描述协议（RFC 4566） - 媒体会话参数 RTP：实时传输协议（RFC 3550） - 媒体流传输
RTCP： RTP控制协议 - RTP的质量监控 SRTP：安全RTP（RFC 3711） - 加密媒体流 Diameter：在IMS中使用的AAA协议（认证、授权、计费） Sh：用于用户数据访问的Diameter接口 Ro：用于在线计费的Diameter接口
SIPREC：会话发起协议录音（RFC 7866） - 通话录音协议

电信设备

SBC：会话边界控制器 - 网络边缘安全和媒体网关
MRF：媒体资源功能 - 媒体处理（转码、混合、录音）
UE：用户设备 - 移动手持设备或设备

- **PSAP**：公共安全应答点 - 紧急服务呼叫中心
- **DRA**：Diameter路由代理 - Diameter消息路由

合法拦截

- **LI**：合法拦截 - 电信的法律监控
- **IRI**：拦截相关信息 - 执法部门的通话元数据
- **CC**：通信内容 - 实际语音/媒体内容
- **SRC**：会话录音客户端 - SIPREC客户端（OmniTAS角色）
- **SRS**：会话录音服务器 - 用于录音存储的SIPREC服务器
- **X1**接口：LI管理接口（授权提供）
- **X2**接口：LI接口用于IRI交付
- **X3**接口：LI接口用于CC交付
- **R226**：法国刑法第R226-3和R226-7条款，规定拦截设备

呼叫处理

- **CDR**：通话详细记录 - 每个通话的计费和日志记录
- **B2BUA**：双向用户代理 - 作为客户端和服务器的SIP元素
- **DTMF**：双音多频 - 按键音信号
- **MSISDN**：移动站国际用户目录号码 - 电话号码
- **IMSI**：国际移动用户身份 - 唯一用户标识符
- **E.164**：电话号码的国际编号计划

安全和加密

- **TLS**：传输层安全（RFC 5246, RFC 8446） - 加密协议
- **PFS**：完美前向保密 - 确保会话密钥安全的密码学属性
- **SHA-512**：输出为512位的安全哈希算法
- **AES**：高级加密标准
- **RSA**：Rivest-Shamir-Adleman - 公钥密码算法
- **ECDSA**：椭圆曲线数字签名算法
- **PKI**：公钥基础设施 - 证书管理系统
- **CA**：证书颁发机构 - 发布数字证书
- **CRL**：证书撤销列表

OCSP：在线证书状态协议

网络和位置

MAP：移动应用部分 - SS7协议用于移动网络 HLR：家庭位置注册 - 用户位置数据库（遗留） SS7：信令系统第7号 - 遗留电话信令
NANP：北美编号计划
基站/小区ID：用于位置追踪的移动网络基站标识符

数据格式和存储

SQLite：嵌入式关系数据库
SQLCipher：带有加密支持的SQLite扩展 CSV：逗号分隔值 - 导出格式
JSON： JavaScript对象表示法 - 数据交换格式 XML：可扩展标记语言 - 结构化数据格式

应用组件

API：应用程序编程接口 - 程序访问
UI：用户界面 - 基于Web的控制面板
RBAC：基于角色的访问控制 - 权限系统
UUID：通用唯一标识符 - 会话追踪

文档版本： 1.0
日期： 2025-11-29
准备对象： ANSSI R226授权申请
文档分类：监管合规 - 机密

配置□南

□ 返回主文档

本文档提供了 TAS 应用服务器的全面配置参考。

相关文档

核心配置

- □ 主 **README** - 概述和快速入门
- □ 操作□南 - 监控和操作任务
- □ □标参考 - Prometheus 指标和监控

集成接口

- □ **Sh** 接口 - 从 HSS/存储库检索订阅者数据
- □ 在线计费 **(Ro)** - OCS 集成和信用控制
- □ **SS7 MAP** - 用于漫游和呼叫转移的 HLR 查询
- □ 合法拦截 **(X1/X2/X3)** - ETSI TS 103 221 拦截 (X1, X2, X3)

呼叫处理

- □ 拨号计划配置 - XML 拨号计划和呼叫路由逻辑
- □ 号码翻译 - E.164 规范化规则
- ⚙ 补充服务 - 呼叫转移、CLI 阻止、紧急服务

增值服务

- □ 语音邮件 - 带 SMS 通知的语音邮件服务
- □ **TTS** 提示 - 文本转语音提示配置
- □ **IMS** 会议服务器 - 多方会议

测试与合规

- **HLR & 呼叫模拟器** - 测试工具
- **ANSSI R226 合规性** - 法国市场合规性

配置

应用服务器需要：

- 连接到 SIP 中继 / SBC 进行外部呼叫
- 连接到 DRA 或 HSS 获取 `Sh`
- 可选地连接到 DRA 或 OCS 进行 `Ro` 在线计费
- 拨号计划配置
- 拨号规则 / 号码翻译的配置
- 语音邮件配置
- 提示
- 测试
- 指标 (Prometheus)

事件套接字配置

事件套接字用于呼叫控制、监控活动呼叫及与电信引擎交互。此连接允许 TAS 控制呼叫路由、检索通道变量并管理活动会话。

配置位置：`config/runtime.exs`

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

配置参数：

- **`host`** (字符串，必需)：事件套接字服务器的主机名或 IP 地址

    - 默认值：`"127.0.0.1"`（本地主机）
    - 如果电信引擎与 TAS 运行在同一服务器上，请使用 localhost
    - 对于分布式部署，请使用远程 IP
    - 示例：`"10.8.82.60"` 用于远程连接
- **`port`**（整数，必需）：事件套接字连接的 TCP 端口
    - 默认值：`8021`
    - 标准事件套接字端口为 8021
    - 必须与电信引擎中的事件套接字配置匹配
    - 示例：`8021`
- **`secret`**（字符串，必需）：事件套接字的身份验证密码
    - 必须与电信引擎中配置的密码匹配
    - 用于身份验证 ESL 连接
    - 安全提示：使用强随机密码并保持安全
    - 示例：`"cd463RZ8qMk9AHMMDGT3V"`

使用案例：

- 实时呼叫控制和路由
- 检索活动呼叫信息用于控制面板中的 `/calls` 视图
- 程序化执行拨号计划应用
- 监控呼叫状态变化和事件
- 管理会议电话

连接行为：

- TAS 建立与事件套接字的持久连接
- 在连接失败时自动重新连接
- 用于入站（接收事件）和出站（控制呼叫）模式
- 连接超时和重试逻辑是内置的

安全考虑：

- 始终为 `secret` 参数使用强大且唯一的密码

- 如果使用远程连接，请确保防火墙规则仅允许受信任的 TAS 服务器
- 考虑在 TAS 和电信引擎共存时使用仅本地主机连接
- 不要将事件套接字端口暴露给公共网络

故障排除：

- 连接被拒绝：验证电信引擎是否正在运行且事件套接字已启用
- 身份验证失败：检查 `secret` 是否与电信引擎配置匹配
- 超时错误：验证网络连接和防火墙规则
- 无法控制呼叫：确保 TAS 已成功连接（检查日志）

多 **IMSI MSISDN** 分叉

多 IMSI MSISDN 是一个号码，由多个设备在不同的 SIM 卡上共享（例如手机和智能手表）。在移动终止呼叫中，TAS 允许呼叫分叉到每个注册的设备。每个其他号码保持单一联系交付。请参见 移动终止呼叫分叉。

此功能无需 TAS 配置。TAS 根据 HSS 数据决定每个呼叫。在终止环节，TAS 通过 Sh 接口 读取被叫方。HSS 返回每个共享号码的成员 IMSI 的一个 `<IMSPrivateUserIdentity>`。TAS 计算这些身份：

- 计数大于一 - 多 IMSI 共享号码。TAS 允许分叉（它不发送 `Request-Disposition: no-fork`），因此 S-CSCF 振铃每个设备。
- 计数为一 - 普通号码。TAS 发送 `Request-Disposition: no-fork` 实现单一联系交付。

这将共享号码集保持在一个地方（HSS），因此新的共享号码无需更改 TAS 配置。

要求：

- S-CSCF 必须配置为每个公共身份存储多个联系。如果它每个号码仅保留一个联系，则分叉无效。请参见 OmniCSCF S-CSCF 文档。
- 在 HSS 中将号码配置为共享公共身份：将成员分组为一个 IMS 订阅并为每个分配相同的 MSISDN。然后 HSS 在 Sh 上返回每个成员的一个 `<IMSPrivateUserIdentity>`。

控制面板配置

控制面板提供基于 Web 的接口，用于监控和管理 TAS 系统。这包括查看订阅者、CDR、活动呼叫、Diameter 对等体、网关和系统配置。

配置位置：`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"
  ]
```

配置参数：

页面顺序配置

- **`page_order`** (字符串列表)：控制控制面板中配置页面的显示顺序
  - 指定哪些页面出现在导航中及其顺序
  - 示例：`["/application", "/configuration"]`
  - 默认值：如果未设置，页面按默认字母顺序出现

Web 端点配置

- **`url`** (映射)：控制面板的公共 URL 配置
  - **`host`**：生成 URL 的主机名（例如，`"tas.example.com"` 或 `"0.0.0.0"`）
  - **`path`**：所有控制面板路由的基本路径（默认：`"/"`）
  - 用于生成重定向和链接中的绝对 URL
- **`https`** (映射)：安全访问的 HTTPS/TLS 配置
  - **`port`** (整数)：HTTPS 端口号（标准为 `443`）
  - **`keyfile`** (字符串)：TLS 私钥文件的路径（PEM 格式）
  - **`certfile`** (字符串)：TLS 证书文件的路径（PEM 格式）
  - 两个文件必须可由 TAS 应用程序读取

证书管理：

控制面板需要有效的 TLS 证书□进行 HTTPS 访问:

1. 自签名证书（开发/测试）:

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

2. 生产证书:
   - 使用受信任的证书颁发机构 (CA) 的证书
   - 常见提供商: Let's Encrypt（免费）、商业 CA
   - 确保证书包含完整链□获得浏览器信任
   - 使用适当的文件权限（`chmod 600`）保持私钥安全

访问控制:

控制面板提供对敏感操作数据的访问:

- 订阅者信息：注册详细信息、呼叫历史、位置信息
- 呼叫详细记录：包含 MSISDN 数据的完整呼叫记录
- 系统配置： Diameter 对等体、网关、路由
- 活动呼叫：实时监控正在进行的会话

推荐的安全措���:

- 在生产环境中部署在防火墙或 VPN 后面
- 使用来自受信任 CA 的强 TLS 证书
- 实施网络级访问控制（IP 白名单）
- 如果公开访问，考虑额外的身份验证层
- 定期审核访问日志
- 仅使用 HTTPS - 切勿通过普通 HTTP 提供服务

常见部署模式:

1. 仅内部访问:

```
url: [host: "10.8.82.60", path: "/"]  # 仅内部网络
```

2. 具有域的外部访问：

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

3. 在反向代理后面：

```
url: [host: "tas.internal", path: "/panel"]  # Nginx/Apache 转发到此
```

故障排除：

- 证书错误：验证 `keyfile` 和 `certfile` 的路径是否正确且文件可读
- 端口已被使用：检查是否有其他服务正在使用端口 443，或更改为其他端口
- 无法访问 **UI**：验证防火墙规则允许访问配置的 HTTPS 端口
- **SSL** 握手失败：确保证书和密钥匹配并且为 PEM 格式

API 配置

TAS 包含一个 REST API，用于对系统功能、订阅者管理和操作数据的编程访问。API 支持 OpenAPI/Swagger 文档，并通过 TLS 进行安全保护。

配置位置：`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"
  }
```

配置参数:

- **`port`** (整数, 必需): API 服务器的 TCP 端口
  - 默认值: `8444`
  - 选择一个不与其他服务冲突的端口
  - 标准 HTTPS 端口为 443, 但自定义端口在 API 中很常见
  - 示例: `8444`、`8443`、`9443`
- **`listen_ip`** (字符串, 必需): 绑定 API 服务器的 IP 地址
  - `"0.0.0.0"`: 在所有网络接口上监听（外部访问）
  - `"127.0.0.1"`: 仅在本地主机上监听（仅内部访问）
  - 特定 IP: 绑定到特定接口（例如, `"10.8.82.60"`）
  - 安全性: 如果 API 仅在内部需要, 请使用 `"127.0.0.1"`
- **`product_name`** (字符串): API 元数据的产品标识符
  - 用于 API 响应和文档中
  - 示例: `"OmniTAS"`、`"MyOperator-IMS"`
- **`title`** (字符串): API 文档的人类可读标题
  - 显示在 OpenAPI/Swagger UI 标头中
  - 示例: `"API - OmniTAS"`、`"IMS 应用服务器 API"`
- **`hostname`** (字符串): 文档中 API 服务器的主机名

    - 用于 OpenAPI 规范生成示例 URL
    - 应与客户端访问 API 的方式匹配
    - 示例：`"localhost"`、`"api.operator.com"`、`"10.8.82.60"`
- **`enable_tls`** (布尔值)：启用或禁用 API 的 TLS/HTTPS
    - `true`：通过 HTTPS 提供 API（推荐用于生产）
    - `false`：通过 HTTP 提供 API（仅用于测试/开发）
    - 安全性：在生产环境中始终使用 `true`
- **`tls_cert_path`** (字符串)：TLS 证书文件的路径（PEM 格式）
    - 当 `enable_tls: true` 时必需
    - 必须可由 TAS 应用程序读取
    - 示例：`"priv/cert/server.crt"`
- **`tls_key_path`** (字符串)：TLS 私钥文件的路径（PEM 格式）
    - 当 `enable_tls: true` 时必需
    - 必须可由 TAS 应用程序读取
    - 安全性：使用文件权限保护（`chmod 600`）
    - 示例：`"priv/cert/server.key"`

API 功能：

REST API 提供对下内容的编程访问：

- 订阅者管理和配置
- 呼叫详细记录 (CDR) 查询
- 系统状态和健康检查
- Diameter 对等体状态
- 网关状态和统计信息
- 活动呼叫监控
- 配置管理

OpenAPI/Swagger 文档：

API 包含内置的 OpenAPI (Swagger) 文档：

- 访问 Swagger UI: `https://hostname:port/api/swaggerui`
- OpenAPI JSON 规范: `https://hostname:port/api/openapi`
- 直接从浏览器进行交互式 API 测试
- 完整的端点文档，包括请求/响应模式

安全考虑:

- 身份验证：根据您的安全要求实施 API 身份验证
- 网络访问：使用防火墙规则限制 API 访问仅限授权客户端
- **TLS** 要求：在生产中始终启用 TLS (`enable_tls: true`)
- 证书验证：对于生产 API 使用受信任的证书
- 速率限制：考虑为公共 API 实施速率限制
- 访问日志：监控 API 访问日志□发现可疑活动

示例用法:

```
# 使用 curl 查询 API（替换为实际端点）
curl -k https://localhost:8444/api/health

# 访问 Swagger 文档
https://localhost:8444/api/swaggerui
```

常见部署场景:

1. 仅内部 **API**:

   ```
   listen_ip: "127.0.0.1"   # 仅可从本地主机访问
   enable_tls: false        # 内部测试使用 HTTP
   ```

2. 生产 **API** 与 **TLS**:

   ```
   listen_ip: "0.0.0.0"     # 可从网络访问
   enable_tls: true         # 需要 HTTPS
   hostname: "api.operator.com"
   ```

3. 开发/测试:

```
listen_ip: "0.0.0.0"
enable_tls: false        # 便于测试的 HTTP
port: 9094               # 非特权端口
```

故障排除：

- 端口绑定失败：验证端口未被其他服务使用，或 root 身份运行获取 < 1024 的端口
- **TLS** 错误：检查证书和密钥路径是否正确且文件可读
- 无法连接：验证防火墙允许访问配置的端口
- 证书不匹配：确保 `hostname` 与证书的公共名称 (CN) 或 SAN 匹配
- **API** 返回 **404**：检查 API 应用程序是否在日志中成功启动

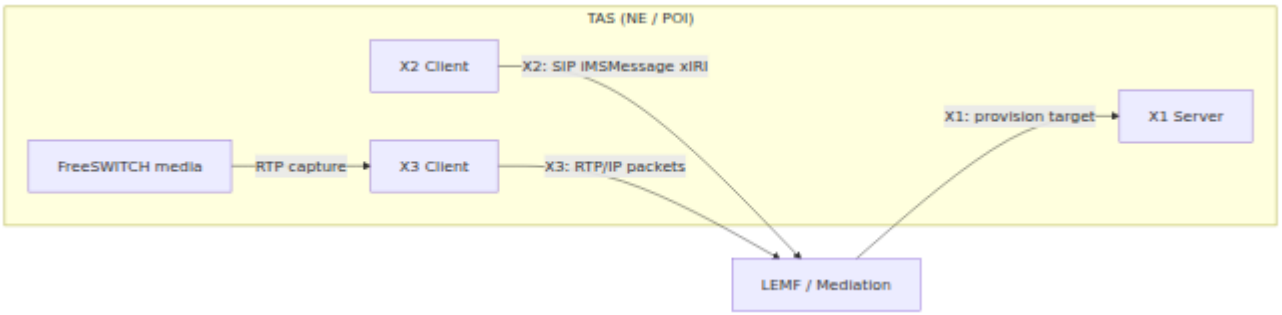
合法拦截 (X1/X2/X3)

TAS 作为语音和 IMS 拦截点 (POI)。TAS 实现 ETSI TS 103 221 拦截接口。TAS 作为网络元素 (NE) 运行。执法监控设施 (LEMF) 通过 X1 配置拦截目标。TAS 通过 X2 传递 SIP 信令。TAS 通过 X3 传递语音媒体。

接口概述

接口	目的	**TAS** 角色	传输
X1	配置和移除拦截任务和目标	服务器	通过 HTTPS 进行双向 TLS
X2	传递拦截相关信息 (SIP 信令作为 TS 33.128 iMSMessage xIRI)	客户端	TCP，可选 TLS
X3	传递通信内容（语音媒体作为 RTP/IP 数据包）	客户端	TCP，可选 TLS

标准参考：



参考	标题
ETSI TS 103 221-1	内部拦截接口 X1 (配置)
ETSI TS 103 221-2	内部拦截接口 X2/X3 (交付)
3GPP TS 33.128	合法拦截有效载荷格式

架构

配置

在 `config :tas, :li` 下配置拦截。为每个部署在 `config/runtime.exs` 中设置值。默认情况下，拦截是禁用的。在您设置 `enabled: true` 之前，不会启动拦截进程，也不会发生拦截。

```
config :tas, :li,
  enabled: true,
  ne_identifier: "tas-poi-01",
  ne_ip: "198.51.100.60",
  media_iface: "eth0",
  tcpdump_path: "/usr/bin/tcpdump",
  x1: [
    enabled: true,
    port: 8443,
    tls: [
      certfile: "/etc/omnitas/li/x1-cert.pem",
      keyfile: "/etc/omnitas/li/x1-key.pem",
      cacertfile: "/etc/omnitas/li/lemf-ca.pem"
    ]
  ],
  x2: [enabled: true, address: "192.0.2.10", port: 12345, tls:
false],
  x3: [enabled: true, address: "192.0.2.10", port: 12346, tls:
false]
```

□□□□

□□	□□	□□	□□	□□
`enabled`	□□□	□	`false`	□□□□□ `false` □□□□□□□□□□□□
`ne_identifier`	□□□	□	`"tas-01"`	□□□□□□□□ (TS 103 221-2 □□ 18)□TAS □□□ X2 □ X3 □□□□□□□□□LEMF □□□□□□□□□□□ IP □□□□□□□□
`ne_ip`	□□□	□	`"0.0.0.0"`	□□□□□□ IP □□□TAS □ X2 □ X3 □□□□□□□□□
`media_iface`	□□□	□	`"eth0"`	□□ RTP □□□□□□□□□X3 □□□□□□□□□□□□
`tcpdump_path`	□□□	□	`"/usr/bin/tcpdump"`	`tcpdump` □□□□□□□X3 □□□□□□□ RTP □□□□□□ TAS □ □□□□□□□□□
`x1`	□□□□□	□	`[]`	X1 □□□□□□□□□□ X1 □□□
`x2`	□□□□□	□	`[]`	X2 □□□□□□□□□□ X2 □ X3 □□□

参数	类型	必需	默认	描述
`x3`	关键字列表	否	`[]`	X3 交付客户端。请参见 X2 和 X3 参数。

X1 参数

TAS 运行 X1 服务器。LEMF 连接到服务器配置目标。TLS 是强制性的 (TS 103 221-1 条款 7.2.2.1)。除非您设置 `allow_insecure: true`，否则服务器不会明文模式启动。

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	启动 X1 服务器。
`port`	整数	否	`8443`	X1 服务器的 TCP 端口。
`tls`	关键字列表	是	-	X1 服务器的 TLS 设置。请参见 X1 TLS 参数。
`allow_insecure`	布尔值	否	`false`	通过明文 HTTP 运行 X1 服务器。仅在实验室中使用此选项。不要在生产中使用此选项。

X1 TLS 参数

参数	类型	必需	默认	描述
`certfile`	字符串	是	-	服务器证书的路径，采用 PEM 格式。
`keyfile`	字符串	是	-	服务器私钥的路径，采用 PEM 格式。
`cacertfile`	字符串	否	-	验证客户端证书的证书颁发机构的路径。设置此值□启用双向 TLS。生产部署需要双向 TLS。

X2 和 X3 参数

TAS 运行 X2 客户端和 X3 客户端。X2 客户端将 SIP 信令作为 TS 33.128 `iMSMessage` xIRI 记录发送。X3 客户端将语音媒体作为 RTP/IP 数据包发送（TS 33.128 条款 7.12.6.3，IPv4 的数据包格式 5 和 IPv6 的格式 6）。两个客户端使用相同的参数。

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	启动客户端。
`address`	字符串	是	-	LEMF 端点的 IP 地址或主机名。
`port`	整数	是	-	LEMF 端点的 TCP 端口。
`tls`	布尔值或关键字列表	否	`false`	设置为 `true` 或 TLS 选项的关键字列表□通过 TLS 传递。设置为 `false` □进行明文 TCP。

持久性

默认情况下，TAS 将配置的拦截任务和目标保存在内存中。要在重启后保留配置的目标，请在部署配置中启用持久拦截存储。TAS 将任务和目标写入磁盘。TAS 会在下次启动时重新加载它们。

合规性

有关拦截合规性的概述，请参见 ANSSI R226 拦截合规性。

SIP 中继配置

Ansible 负责为每个外发网关创建 XML 配置，这些配置在 `Gateways` 选项卡中可见，用于外发呼叫。

CSCF 地址和网关地址必须包含在运行时配置中，□便我们知道允许哪些 IP 进行呼叫，我们在网关/SBC（将 MT 流量发送到网络的源）和 CSCF（MO 流量的来源）中使用 `allowed_sbc_source_ips` 和 `allowed_cscf_ips`。

注意 - 如果您将呼叫从 TAS 路由到自身（即 MO 呼叫路由回 MT 拨号计划），则您的 TAS IP 也必须在允许的源 IP 列表中。

```
config :tas,
  allowed_sbc_source_ips: ["10.5.198.200", "203.0.113.36"],
  allowed_cscf_ips: ["10.8.3.34"],
```

通过 Web UI，我们可□查看每个网关的状态，□及：

- SIP 注册状态（如果启用了注册）
- SIP 领域
- SIP 代理地址（如果使用）
- 用户名
- Ping 时间（平均 SIP OPTIONs 响应时间（如果启用 SIP OPTIONs））
- 正常运行时间（自配置文件重启或启动□来的秒数）
- 呼叫进 / 呼叫出 / 失败的呼叫进 / 失败的呼叫出
- 最后 SIP OPTIONs ping 时间（纪元）
- SIP OPTIONs ping 频率
- 在 **detail** 按钮中查看更多信息

网关配置参考

网关 XML 格式配置。每个网关表示与外部 SBC、运营商或 PSTN 网关的 SIP 中继连接。

基本网关示例：

```
<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>
```

无注册的网关：

```
<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>
```

网关参数

必需参数

`name`（网关属性）

- 此网关的唯一名称标识符
- 在拨号计划中用于引用网关：`sofia/gateway/name/destination`
- 示例：`<gateway name="my_trunk">`

`proxy`

- SIP 代理/网关 IP 地址或主机名
- 可包含端口和传输协议
- 示例：
  - `value="203.0.113.50"`（默认端口 5060，UDP）
  - `value="203.0.113.50:5061"`（自定义端口）
  - `value="203.0.113.50;transport=tcp"`（TCP 传输）
  - `value="203.0.113.50:5061;transport=tls"`（TLS 在端口 5061）

`register`

- 是否向网关发送 SIP REGISTER
- 值：`true` | `false`
- 如果中继需要注册，则设置为 `true`
- 对于基于静态 IP 的中继，设置为 `false`

认证参数

`username`

- SIP 认证用户名
- 在 REGISTER 和摘要认证中使用
- 如果 `register="true"`，则为必需
- 示例：`value="trunk_account_123"`

`password`

- SIP 认证密码
- 用于摘要认证挑战
- 如果 `register="true"`，则为必需
- 示例：`value="MySecureP@ssw0rd"`

`realm`

- SIP 认证领域
- 可选 - 通常从挑战中自动检测
- 示例：`value="sip.carrier.com"`

`auth-username`

- 用于认证的备用用户名（如果与 `username` 不同）
- 很少需要 - 仅在运营商要求在 auth 与 From 头中使用不同用户时
- 示例: `value="auth_user_456"`

注册参数

`register-transport`

- 注册消息的传输协议
- 值: `udp` | `tcp` | `tls`
- 必须与 `proxy` 参数中指定的传输匹配
- 示例: `value="tcp"`

`register-proxy`

- 注册的替代代理地址（如果与呼叫路由不同）
- 当注册服务器与呼叫路由服务器不同时很有用
- 示例: `value="register.carrier.com:5060"`

`retry-seconds`

- 失败注册后等待重试的秒数
- 默认: `30`
- 范围: `5` 到 `3600`
- 示例: `value="30"`

`expire-seconds`

- 注册过期时间（□秒为单位）
- 默认: `3600`（1 小时）
- 网关将在过期之前重新注册
- 示例: `value="1800"`（30 分钟）

`caller-id-in-from`

- 在 SIP From 头中包含来电号码

值：true | false
true：From 头包含实际来电号码（大多数运营商要求） false：From 头使用网关用户名
建议：对于大多数部署，设置为 true
示例：value="true"

监控参数

ping

每 N 秒发送 SIP OPTIONS ping
监控网关可用性并测量延迟
如果未指定或设置为 0，则禁用
典型值：15 到 60 秒
在网关状态 UI 中显示为 "Ping 时间"
示例：value="25"

ping-max

在标记网关为故障之前重试 ping 的最大时间（秒）
默认：从 ping 间隔计算
示例：value="3"

呼叫路由参数

extension

固定目的号码，始终在此网关上拨打
很少使用 - 通常目的地来自拨号计划
示例：value="+12125551234"

extension-in-contact

在 Contact 头中包含扩展
值：true | false
默认：false
示例：value="false"

contact-params

附加参数，附加到 Contact 头
对于运营商特定要求很有用
示例：value="line=1;isup=true"

高级参数

from-user

覆盖 From 头中的用户名
默认：使用呼叫号码或网关用户名
示例：value="trunk_pilot"

from-domain

覆盖 From 头中的域
默认：使用代理域
示例：value="my-domain.com"

outbound-proxy

所有 SIP 消息的出站代理
与 proxy不同 - 用作 Route 头目标示例：value="edge-proxy.carrier.com:5060"

context

从此网关接收的来电的拨号计划上下文
默认：public
允许每个网关的不同来电路由
示例：value="from-carrier"

channels

此网关的最大并发呼叫数
默认：无限制
用于容量管理

- 示例：`value="100"`

`dtmf-type`

- DTMF 传输方法
- 值：`rfc2833` | `info` | `inband` | `auto`
- 默认：`rfc2833`（推荐）
- `rfc2833`：RTP 电话事件（最常见）
- `info`：SIP INFO 消息
- `inband`：音频音调
- 示例：`value="rfc2833"`

`codec-prefs`

- 此网关的首选编解码器列表
- 逗号分隔的首选顺序列表
- 示例：`value="PCMU,PCMA,G729"`
- 常见编解码器：`PCMU`、`PCMA`、`G729`、`AMR`、`AMR-WB`、`G722`、`OPUS`

`rtp-timeout-sec`

- 如果 N 秒内未收到 RTP，则挂断呼叫
- 默认：`0`（禁用）
- 有助于检测死呼叫
- 示例：`value="120"`

`rtp-hold-timeout-sec`

- 挂起状态下无 RTP 的呼叫超时
- 默认：`0`（禁用）
- 示例：`value="1800"`（30 分钟）

SIP 信令选项

`sip-port`

- 此网关使用的本地 SIP 端口
- 默认：配置文件的端口

很少需要
示例：value="5060"

rtp-ip

RTP 媒体的本地 IP 地址
默认：配置文件的 RTP IP
示例：value="10.0.0.5"

register-proxy-port

注册代理的端口
仅在与代理端口不同的情况下需要
示例：value="5061"

contact-host

覆盖 Contact 头的主机部分
对于 NAT 场景很有用
示例：value="public-ip.example.com"

distinct-to

使用不同于请求 URI 的不同 To 头
值：true | false
运营商特定要求
示例：value="false"

cid-type

在 Remote-Party-ID 或 P-Asserted-Identity 头中的来电号码类型值：rpid | pid | none
rpid：Remote-Party-ID 头
pid：P-Asserted-Identity 头
示例：value="pid"

extension-in-contact

将扩展参数添加到 Contact URI
值：true | false
示例：value="true"

传输安全

transport（在代理参数中）

传输协议
值：udp | tcp | tls | ws | wss
作为代理值的一部分指定
示例：proxy="203.0.113.50;transport=tcp"

对于 TLS 连接，可能需要在 SIP 配置文件中进行额外的证书配置。

包含常见选项的完整示例

```
<include>
  <gateway name="primary_carrier">
    <!-- 必需：基本连接 -->
    <param name="proxy"
value="sbc.carrier.com:5060;transport=tcp"/>
    <param name="register" value="true"/>

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

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

    <!-- 来电号码 -->
    <param name="caller-id-in-from" value="true"/>

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

    <!-- 媒体 -->
    <param name="codec-prefs" value="PCMU,PCMA,G729"/>
    <param name="dtmf-type" value="rfc2833"/>

    <!-- 呼叫限制 -->
    <param name="channels" value="100"/>

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

拨号计划中的网关使用

在拨号计划中使用 `sofia/gateway/name/destination` 格式引用网关：

```
<!-- 路由到特定网关 -->
<action application="bridge" data="sofia/gateway/primary_carrier/+121

<!-- 使用变量路由 -->
<action application="bridge" data="sofia/gateway/primary_carrier/${ta

<!-- 带有自定义 SIP 头的路由 -->
<action application="bridge" data="{sip_h_X-Custom=Value}sofia/gatewa

<!-- 网关之间的故障转移 -->
<action application="bridge"
data="sofia/gateway/primary_carrier/${tas_destination_number}|sofia/g
```

故障排除网关问题

网关无法注册：

- 验证 `username` 和 `password` 是否正确
- 检查 `proxy` 地址是否可达
- 确认 `register-transport` 是否符合运营商要求
- 查看日志获取认证失败的信息

呼叫失败：

- 检查 Web UI 中的网关状态 (`/gw`)
- 验证 `caller-id-in-from` 设置是否符合运营商要求
- 确认与 `codec-prefs` 的编解码器兼容性
- 检查防火墙是否允许 SIP 和 RTP 流量

通话质量差：

- 查看网关状态中的 `ping` 时间
- 检查 `rtp-timeout-sec` 是否过于激进
- 确认编解码器偏好与网络能力匹配
- 监控网络延迟和丢包

SBC 模式（跳过直径查找）

有时此节点仅作为 会话边界控制器（**SBC**）部署，直接传递呼叫，而不作为完整的电话应用服务器。在这种角色中，您不希望它与直径通信 - 不进行 HSS 订阅者查找，不进行 OCS 信用控制。

`skip_diameter_lookups` 标志将节点置于此通行模式：

```
config :tas,
  skip_diameter_lookups: true
```

当 **`true`** 时的行为：

- **Sh** 接口（**HSS**）：不发送用户数据请求（UDR）。MO 呼叫者和 MT 被叫方解析为默认的订阅者数据形状（所有字段为 `"none"`，`no_reply_timer` 来自 `default_no_reply_timer`），因此呼叫将 `hangup_case = "none"` 进行，而不是被拒绝为 `UNALLOCATED_NUMBER`。
- 内网检测（**MO**）：被跳过 - 每个目的地都被视为外网，并通过正常的网关/SBC 路径路由。
- **Ro** 接口（**OCS**）：不发送信用控制请求（CCR-I / CCR-U / CCR-T）。授权、应答和挂断计费都被绕过；呼叫被授权为无限时间。这会覆盖 `online_charging.enabled` - 当 `skip_diameter_lookups` 为 `true` 时，无论在线计费配置如何，Ro 都保持关闭状态。

默认值和兼容性：

- 该标志是 可选的，当省略时默认为 `false`（完整 TAS 行为），因此现有配置不受影响，并且在配置验证中不是必需的键。
- 您可保留 `diameter_ex` 对等配置和 `online_charging` 块；设置标志后，两者都不会被使用。您不需要为仅 SBC 节点定义直径对等体。

```
# 完整 TAS 节点（默认）
config :tas,
  skip_diameter_lookups: false

# 仅 SBC 节点 - 无直径流量
config :tas,
  skip_diameter_lookups: true
```

Sh 数据引用选择（`sh_data_references`）

可选。控制 TAS 在查找订阅者数据时请求的 Sh `Data-Reference` 值，每个呼叫的侧面（`caller` = MO 源，`called` = MT 目的地）。未设置时，TAS 会请求两侧的完整默认集，因此此键仅在 HSS 拒绝该集的一部分时才重要（例如，回答 `5012 UNABLE_TO_COMPLY` 或 `5101 OPERATION_NOT_ALLOWED` 对于某些引用，这将导致整个 UDR 失败）。

```
config :tas,
  # 每侧覆盖；省略的侧面回退到完整的默认集。
  sh_data_references: [
    caller: [10, 16, 17, 32],
    called: [10, 16, 17, 32]
  ]

# ...或应用于两侧的单一裸列表：
config :tas, sh_data_references: [10, 16, 17, 32]
```

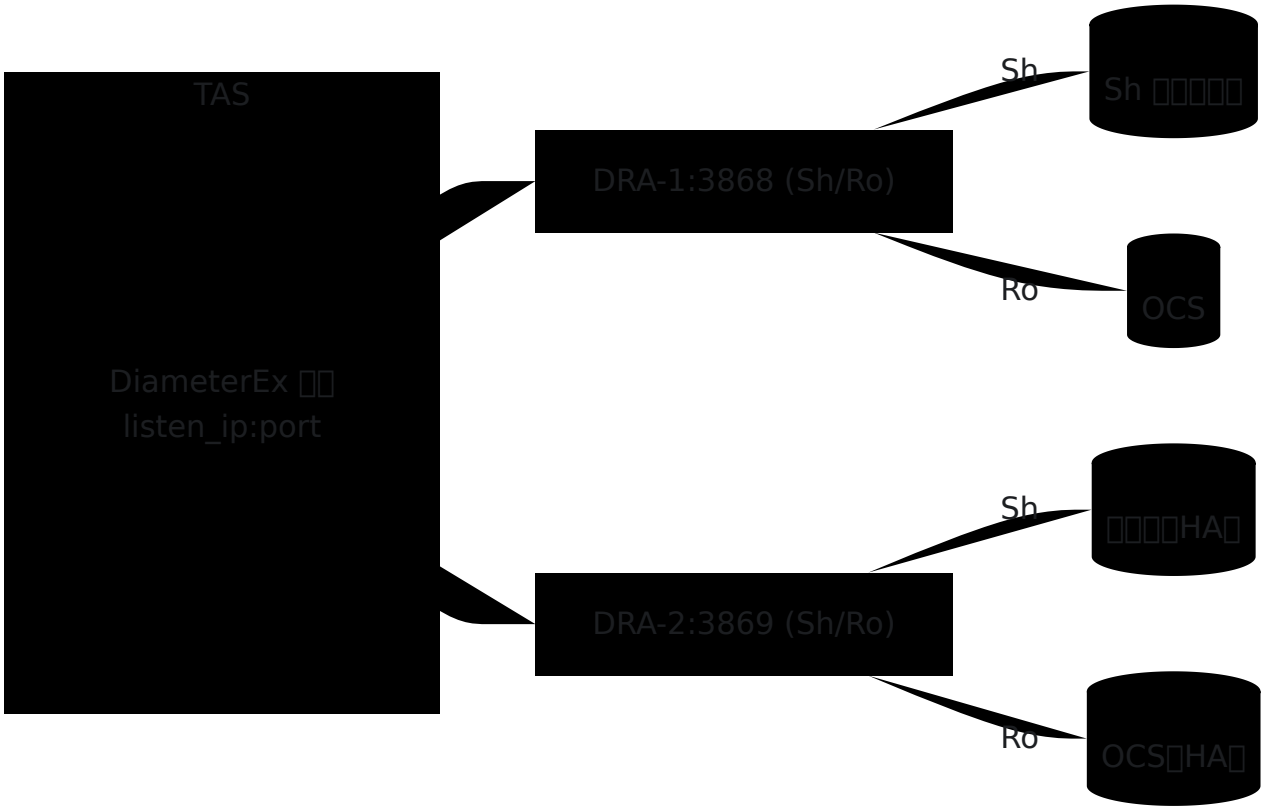
有关数据引用表、接受的形状□及每个 UDR 的会话 ID 呼叫处置（`;mo` / `;mt`）标签的详细信息，请参见 Sh 接口 指南。

直径对等配置

直径对等体必须在运行时配置中定义。

此配置主要是样板。

如果在您的部署中不使用 Ro 接口，则不需要将其包含在应用程序中。



直径对等连接

TAS
DiameterEx □□
listen_ip:port
DRA-1:3868 (Sh/Ro)
DRA-2:3869 (Sh/Ro)
Sh
Ro
Sh □□□□□
OCS
□□□□HA□
OCS□HA□

```
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],
    # 可选: 所有应用程序的全局 destination_realm
    # destination_realm: "global.destination.realm",
    applications: [
      %{
        application_name: :sh,
        application_dictionary: :diameter_gen_3gpp_sh,
        # 可选: Sh 请求的特定应用程序 destination_realm
        # destination_realm: "sh.destination.realm",
        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,
        # 可选: Ro 请求的特定应用程序 destination_realm
        # destination_realm: "ocs.destination.realm",
        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
      }
    ]
  }
```

直径配置参数

服务配置:

- **`service_name`**（原子）：此直径服务实例的唯一标识符
  - 示例: `:omnitouch_tas`
  - 在内部用于服务管理
- **`listen_ip`**（字符串）：用于直径连接的绑定 IP 地址

  - 示例: `"10.8.82.60"`
  - 使用 `"0.0.0.0"` 在所有接口上监听
  - 对等体将连接到此 IP
- **`listen_port`**（整数）：直径连接的 TCP 端口
  - 标准直径端口: `3868`
  - 不得与其他服务冲突
- **`host`**（字符串）：直径主机身份（不带领域）
  - 示例: `"example-dc01-as01"`
  - 与 `realm` 组合形成 Origin-Host AVP
  - 在直径网络中必须唯一
- **`realm`**（字符串）：直径领域身份
  - 示例: `"epc.mnc001.mcc001.3gppnetwork.org"`
  - 用于 Origin-Realm AVP
  - 必须符合 3GPP 网络标识符约定
- **`product_name`**（字符串）：CER/CEA 消息中的产品标识符
  - 示例: `"OmniTAS"`
  - 用于能力交换消息
- **`request_timeout`**（整数）：直径请求的超时时间（毫秒为单位）
  - 示例: `5000`（5 秒）
  - 在此时间内未响应的请求将超时
- **`peer_selection_algorithm`**（原子）：在多个可用对等体中选择对等体的算法
  - 值: `:random` | `:round_robin` | `:priority`
  - `:random`: 随机选择对等体
  - `:round_robin`: 均匀分配请求到对等体
- **`vendor_id`**（整数）：3GPP 供应商 ID
  - 标准 3GPP 供应商 ID: `10415`

- 用于 Vendor-Specific-Application-Id AVP

目的地领域配置

`destination_realm` 参数控制包含在直径请求中的 `Destination-Realm` AVP。此 AVP 告诉直径路由代理（DRA）将请求路由到何处。

三个级别的配置：

1. 特定于应用程序（优先级最高）：在每个应用程序配置中设置 `destination_realm`
2. 全局：在直径配置的顶层设置 `destination_realm`
3. 回退（优先级最低）：果上述都未配置，则使用 `realm` 值

配置示例：

```
# 示例 1: 特定于应用程序的目的地领域
config :diameter_ex,
  diameter: %{
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    applications: [
      %{
        application_name: :sh,
        destination_realm:
"hss.epc.mnc001.mcc001.3gppnetwork.org",
        # ... 其他配置
      },
      %{
        application_name: :ro,
        destination_realm:
"ocs.epc.mnc001.mcc001.3gppnetwork.org",
        # ... 其他配置
      }
    ]
  }

# 示例 2: 具有应用程序特定覆盖的全局目的地领域
config :diameter_ex,
  diameter: %{
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    destination_realm: "dra.epc.mnc001.mcc001.3gppnetwork.org",  #
所有应用程序的默认值
    applications: [
      %{
        application_name: :sh,
        # 将使用全局值: "dra.epc.mnc001.mcc001.3gppnetwork.org"
      },
      %{
        application_name: :ro,
        destination_realm:
"ocs.epc.mnc001.mcc001.3gppnetwork.org",  # 覆盖
      }
    ]
  }

# 示例 3: 未配置目的地领域（使用领域）
config :diameter_ex,
  diameter: %{
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
```

```
    # 未在任何地方指定目的地领域
    applications: [
      %{
        application_name: :sh,
        # 将使用领域回退："epc.mnc001.mcc001.3gppnetwork.org"
      }
    ]
  }
```

何时使用目标领域：

- 不同的后端系统：当 Sh 访问 HSS 而 Ro 访问不同领域的 OCS 时
- **DRA** 路由：当 DRA 使用目标领域将请求路由到不同的后端集群时
- 多租户部署：将不同的应用程序路由到不同的租户领域
- 测试场景：为每个应用程序覆盖目标领域进行测试，而无需更改对等体

回退层级：

```
应用程序特定的 destination_realm
   ↓ （如果未设置）
全局 destination_realm
   ↓ （如果未设置）
领域
```

这确保了在外发请求中始终存在强制性的 `Destination-Realm` AVP。

您可从 Web UI 的 **Diameter** 选项卡检查 Diameter 对等体的状态。

您还可在 Web UI 的 **Sh** 选项卡上测试检索 `Sh` 数据，尝试从 Sh 获取任何数据。

FreeSWITCH 日志处理程序

FreeSWITCH 日志事件（通过事件套接字日志监听器接收）被写入一个专用的 JSON 日志文件，与 TAS 应用程序日志分开。这保持了 TAS 日志的整洁，并使原始 FreeSWITCH 输出在 `/logs` 视图中可用。

配置位置: `config/runtime.exs`

```
config :tas,
  freeswitch_log: %{
    path: "/var/log/omnitas-freeswitch.log",
    level: :notice
  }
```

配置参数:

- **`path`** (字符串): FreeSWITCH 日志处理程序写入的文件
  - 默认值: `"/var/log/omnitas-freeswitch.log"`
  - 必须可由 TAS 进程写入
  - 文件会自动按大小轮换（每个文件 50 MB，保留 5 个文件）
- **`level`** (原子): 从 FreeSWITCH 捕获的最低严重性
  - 默认值: `:notice`
  - 适用标准 Erlang `:logger` 级别（`:debug`, `:info`, `:notice`, `:warning`, `:error`, ...）

- 降低级别（例如 `:info`）□进行详细故障排除；提高级别□减少日志量

整个块是可选的。省略时，将使用上述默认值。

拨号计划配置与呼叫路由

 返回主文档

XML 拨号计划配置、呼叫路由逻辑和拨号计划变量的综合指南。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - SIP 中继和网关配置
- 操作南 - 拨号计划测试和模板查看器

呼叫处理流程

- 号码翻译 - E.164 标准化（发生在拨号计划之前）
- **Sh** 接口 - 为拨号计划变量检索的用户数据
- **SS7 MAP** - 拨号计划变量中的 MSRN/HLR 数据
- 在线计费 - 呼叫流程中的 OCS 授权

服务实现

- ⚙ 补充服务 - 在拨号计划中实现呼叫转移、CLI 阻止
- 语音信箱 - 拨号计划中的语音信箱路由和存储/检索
- **TTS** 提示 - 在��号计划中使用播放提示

监控

- 拨号计划标 - 拨号计划特定的指标和监控
- 标参考 - 一般系统指标

拨号计划配置 / 呼叫路由

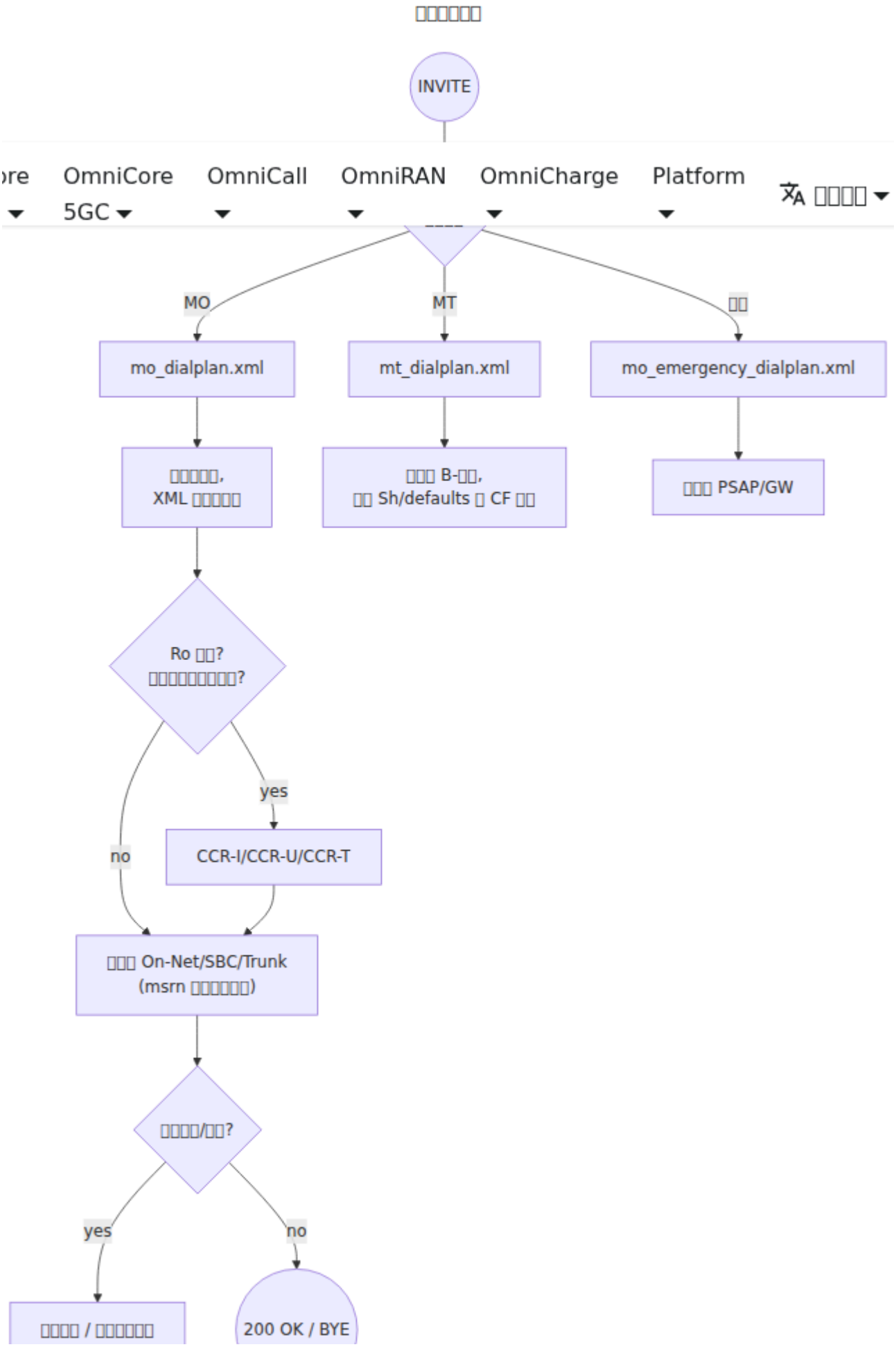
TAS 使用与标准电信 XML 拨号计划格式兼容的 XML 拨号计划，变量由 TAS 填充。这意味着您可根据需要定义自己的拨号计划，包含运营商的业务逻辑，但拥有所有所需的数据，如存储库数据、SS7 路由信息、IMPI / IMPU 身份、拨号计划标准化等。

拨号计划写入 `priv/templates`，格式如下：

- `mo_dialplan.xml` - 移动发起呼叫拨号计划
- `mo_emergency_dialplan.xml` - 移动发起紧急呼叫拨号计划
- `mt_dialplan.xml` - 移动终止呼叫拨号计划

您可通过 Web UI 查看拨号计划。

在 XML 被解析之前，TAS 设置了各种变量，这些变量在呼叫开始时其当前值打印到日志中，在定义自己的呼叫逻辑时非常有帮助。


□□□□□□□
INVITE
ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
□□□□
MO
MT
□□
mo_dialplan.xml
mt_dialplan.xml
mo_emergency_dialplan.xml
□□□□□□,
XML □□□□□□
□□□ B-□□,
□□ Sh/defaults □ CF □□
□□□ PSAP/GW
Ro □□?
□□□□□□□□□□?
yes
no
CCR-I/CCR-U/CCR-T
□□□ On-Net/SBC/Trunk
(msrn □□□□□□)
□□□□/□□?
yes
no
□□□□ / □□□□□□
200 OK / BYE

FreeSWITCH XML 拨号计划基础

OmniTAS 使用与 FreeSWITCH 项目相同的 XML 呼叫路由系统，这允许灵活的呼叫路由□满足您的需求。

本节解释核心概念并提供实际示例。

基本结构

拨号计划由 扩展 组成，包含 条件 和 动作：

```
<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>
```

扩展 按从上到下的顺序进行评估。当条件匹配时，其动作执行。

条件和正则匹配

条件测试变量与正则表达式。如果正则表达式匹配，则执行动作；如果不匹配，���执行反向动作。

基本精确匹配：

```
<condition field="${tas_destination_number}" expression="2222">
  <action application="log" data="INFO 正在拨打语音信箱接入号码"/>
</condition>
```

多个号码匹配：

```
<condition field="${tas_destination_number}"
expression="^(2222|3444|3445)$">
  <action application="log" data="INFO 正在拨打特殊服务"/>
</condition>
```

带捕获组的模式匹配：

```
<condition field="${tas_destination_number}" expression="^1(8[0-9]
{9})$">
  <!-- 匹配 1 后跟 8 和 9 个数字 -->
  <action application="log" data="INFO 匹配免付费号码: $1"/>
  <action application="bridge"
data="sofia/gateway/trunk/${tas_destination_number}"/>
</condition>
```

前缀匹配:

```
<condition field="${tas_destination_number}" expression="^00">
  <!-- 匹配以 00 开头的任何号码（国际） -->
  <action application="log" data="INFO 检测到国际呼叫"/>
</condition>
```

范围匹配:

```
<condition field="${msisdn}" expression="^5551241[0-9]{4}$">
  <!-- 匹配 55512410000 到 55512419999 -->
  <action application="log" data="INFO 订阅者在范围内"/>
</condition>
```

动作与反向动作

动作 在条件匹配时执行。反向动作 在条件不匹配时执行。

```
<condition field="${cli_withheld}" expression="true">
  <!-- 如果 CLI 被隐藏则执行 -->
  <action application="set"
data="effective_caller_id_number=anonymous"/>
  <action application="set"
data="origination_privacy=hide_number"/>

  <!-- 如果 CLI 未被隐藏则执行 -->
  <anti-action application="log" data="DEBUG CLI 是正常的"/>
  <anti-action application="set"
data="effective_caller_id_number=${msisdn}"/>
</condition>
```

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="${hangup_case}"
expression="OUTGOING_CALL_BARRED">
    <action application="log" data="ERROR 余额不足"/>
    <action application="hangup" data="${hangup_case}"/>
  </condition>
</extension>

<extension name="Route-Call">
  <!-- 这个扩展仍然会被评估 -->
  <condition field="${tas_destination_number}"
expression="^(.*)$">
    <action application="bridge"
data="sofia/gateway/trunk/${tas_destination_number}"/>
  </condition>
</extension>
```

使用 `continue="true"` 用于:

- 日志记录/调试扩展
- 设置适用于多种场景的变量
- 不路由呼叫的验证检查

常见应用

呼叫控制

answer - 接听呼叫（发送 200 OK）

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

hangup - 特定原因终止呼叫

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

bridge - 将呼叫连接到另一个目的地

```
<!-- 桥接到外部网关 -->
<action application="bridge"
data="sofia/gateway/trunk/+12125551234"/>

<!-- 桥接到具有编解码器首选项的内部分机 -->
<action application="bridge" data="{absolute_codec_string=AMR-
WB,AMR,PCMA}sofia/internal/sip:user@domain.com"/>

<!-- 带超时的桥接 -->
<action application="bridge" data="
{originate_timeout=30}sofia/gateway/trunk/${tas_destination_number}"/
```

变量和通道数据

set - 设置通道变量

```
<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 - 移除通道变量

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

export - 设置变量并导出到 B-leg（桥接呼叫）

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

媒体和提示

playback - 播放音频文件

```
<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 - 暂停指定的毫秒数

```
<action application="sleep" data="1000"/>  <!-- 暂停 1 秒 -->
```

echo - 将音频回显给呼叫者（测试）

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

conference - 将呼叫放入会议

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

语音信箱（**TAS ESL** 语音信箱）

语音信箱由 TAS 通过 ESL 出站套接字提供，而不是 FreeSWITCH 的 `voicemail` 应用程序。设置模式/邮箱/呼叫者变量并将呼叫交给套接字。请参见 语音信箱 获取完整流程。

```
<!-- 留下语音信箱给邮箱（存储） -->
<action application="set" data="tas_vm_mode=deposit"/>
<action application="set" data="tas_vm_mailbox=${msisdn}"/>
<action application="set"
data="tas_vm_caller=${effective_caller_id_number}"/>
<action application="socket" data="127.0.0.1:8084 async full"/>

<!-- 检查语音信箱（检索） -->
<action application="set" data="tas_vm_mode=retrieve"/>
<action application="set" data="tas_vm_mailbox=${msisdn}"/>
<action application="socket" data="127.0.0.1:8084 async full"/>
```

日志记录和调试

log - 写入日志文件

```
<action application="log" data="INFO 正在处理来自 ${msisdn} 的呼叫"/>
<action application="log" data="DEBUG 目的地:
${tas_destination_number}"/>
<action application="log" data="ERROR 呼叫失败，原因: ${hangup_cause}"/>
```

info - 将所有通道变量转储到日志

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

其他应用

say - 文本转语音号码读取

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

send_dtmf - 发送 DTMF 音调

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

实际示例

紧急服务路由:

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

基于余额的条件路由：

```
<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 与 **Off-Net** 路由（使用 **HOMER X-CID** 关联）：

```
<extension name="Route-Decision">
  <condition field="${on_net_status}" expression="true">
    <!-- On-net: 通过 TAS 进行 MT 处理 -->
    <action application="log" data="INFO 路由到 on-net 订阅者"/>
    <action application="set" data="sip_copy_custom_headers=false"/>
    <action application="bridge" data="{sip_h_X-
CID=${original_call_id},absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU'}sofia/internal/${tas_destination_number}@${local_ip

    <!-- Off-net: 路由到 SIP 中继 -->
    <anti-action application="log" data="INFO 路由到 off-net"/>
    <anti-action application="set" data="sip_copy_custom_headers=fals
    <anti-action application="bridge" data="{sip_h_X-
CID=${original_call_id},absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU'}sofia/gateway/trunk/+${tas_destination_number}"/>
  </condition>
</extension>
```

> 注意：`sip_h_X-CID` 必须作为内联桥接变量 `{sip_h_X-CID=...}` 设置，而不是通过 `set`，因为 `sip_copy_custom_headers=false` 会删除通过 `set` 设置的 `sip_h_` 变量。有关跨腿关联的完整详细信息，请参见 HOMER 集成。

匿名呼叫者 **ID** 处理：

```
<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>
```

无应答时的语音信箱：

```
<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"/>

    <!-- 如果桥接失败，则转到语音信箱（TAS ESL 语音信箱；请参见 voicemail.md） -->
    <action application="log" data="INFO 桥接失败，路由到语音信箱"/>
    <action application="answer" data=""/>
    <action application="set" data="tas_vm_mode=deposit"/>
    <action application="set"
data="tas_vm_mailbox=${tas_destination_number}"/>
    <action application="set"
data="tas_vm_caller=${effective_caller_id_number}"/>
    <action application="socket" data="127.0.0.1:8084 async
full"/>
  </condition>
</extension>
```

号码范围路由：

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

<extension name="National-Numbers">
  <condition field="${tas_destination_number}"
expression="^555\d{7}$">
    <!-- 国家移动号码 -->
    <action application="log" data="INFO 国家移动呼叫"/>
    <action application="bridge"
data="sofia/gateway/national_trunk/${tas_destination_number}"/>
  </condition>
</extension>

<extension name="International">
  <condition field="${tas_destination_number}"
expression="^00\d+$">
    <!-- 以 00 开头的国际呼叫 -->
    <action application="log" data="INFO 国际呼叫"/>
    <action application="bridge"
data="sofia/gateway/intl_trunk/${tas_destination_number}"/>
  </condition>
</extension>
```

进一步文档

有关每个应用程序的完整详细信息：

- **FreeSWITCH** 拨号计划文档：
  https://freeswitch.org/confluence/display/FREESWITCH/Dialplan
- **FreeSWITCH mod_dptools**：
  https://freeswitch.org/confluence/display/FREESWITCH/mod_dptools（完整应用程序参考）
- 正则表达式参考：
  https://freeswitch.org/confluence/display/FREESWITCH/Regular+Expression

通道变量：
https://freeswitch.org/confluence/display/FREESWITCH/Channel+Variables

FreeSWITCH wiki 包含每个拨号计划应用程序的详细文档，包括所有参数和用例。

拨号计划变量

TAS 在 XML 拨号计划逻辑中设置的变量：

常见变量（所有呼叫类型）

初始设置：

destination_number - 翻译后的目的地号码
tas_destination_number - 翻译后的目的地号码
effective_caller_id_number - 翻译后的源号码

呼叫方向（CDR 标记）：

tas_direction - CDR 的粗略呼叫方向：MO、MT 或 UNKNOWN
tas_disposition - 完整的路由处置：mo、mt、emergency 或 conference

这些在拨号计划处理期间通过 ESL (uuid_setvar) 设置在 A-leg 上，在处置计算后立即设置。没有 这个，每个呼叫都作为来自 S-CSCF 的入站 INVITE 到达 FreeSWITCH，因此内置的 direction 通道变量始终为 inbound，CDR 无法区分 MO 和 MT。

方向是从处置中派生的：

处置 如何确定（Authorization） tas_direction

:mo 源 IP 是允许的 CSCF MO

:mt 源 IP 是允许的 SBC（或本地 on-net B-leg） MT

:emergency 紧急 URN / 代码（用户发起） MO

:conference IMS 会议工厂 INVITE（用户发起） MO

> 内部重新进入（例如 `:credit_exhausted` 转移）不被标记，因此它们不能覆盖原始呼叫腿上设置的方向。

要在 CDR 中显示这些，必须在 CDR 后端中存在相应的列（`cdr_sqlite` / `cdr_csv`）并在模板中列出。请参见 OmniCore ansible `applicationserver` 角色（`cdr_sqlite.conf.xml` 模板 + `重新创建 cdr 表架构` 任务）。标记仅限于 A-leg（与现有翻译号码变量匹配）；B-leg 行通过 `bleg_uuid` 关联。

紧急呼叫

- `hangup_case` - "none"
- `ims_private_identity` - 私有用户身份
- `ims_public_identity` - 公共用户身份
- `msisdn` - 订阅者号码（去掉 +）
- `imsi` - 来自私有身份的 IMSI
- `ims_domain` - 来自私有身份的域

MT 呼叫（移动终止）

- `ims_private_identity` - 私有用户身份
- `ims_public_identity` - 公共用户身份
- `msisdn` - 订阅者号码（去掉 +）
- `msisdns` - 订阅者的所有号码，逗号分隔（去掉 +），未知时为空
- `imsi` - 来自私有身份的 IMSI
- `ims_domain` - 来自私有身份的域
- `call_forward_all_destination` - CFA 目的地或 "none"
- `call_forward_not_reachable_destination` - CFNRc 目的地
- `scscf_address` - S-CSCF 地址或 "none"
- `scscf_domain` - S-CSCF 域或 "none"
- `no_reply_timer` - 无应答超时
- `hangup_case` - "none" 或 "UNALLOCATED_NUMBER"
- `msrn` - 来自 PRN 的 MSRN（如果漫游）或来自 SRI 的转发号码（如果呼叫转移处于活动状态）
- `tas_destination_number` - 路由目的地覆盖（设置为 MSRN 或转发号码）

MO 呼叫（移动发起）

- `hangup_case` - "none"、"OUTGOING_CALL_BARRED" 或 "UNALLOCATED_NUMBER"

- `ims_private_identity` - 私有用户身份
- `ims_public_identity` - 公共用户身份
- `msisdn` - 订阅者号码（去掉 +）
- `msisdns` - 订阅者的所有号码，逗号分隔（去掉 +），未知时为空
- `imsi` - 来自私有身份的 IMSI
- `ims_domain` - 来自私有身份的域
- `allocated_time` - OCS 分配的时间（如果启用在线计费）
- `cli_withheld` - "true" 或 "false" 字符串
- `on_net_status` - "true" 或 "false" 字符串（目的地是否在网内，由对拨打号码的 Sh 查找确定）
- `original_call_id` - 来自 UE 的原始 A-leg SIP Call-ID（用于跨 B2BUA 腿的 HOMER X-CID 关联）
- `msrn` - 漫游订阅者的 MSRN（如适用）
- `tas_destination_number` - MSRN 覆盖（如果漫游）

拨号计划为 OmniTAS 设置的变量：

- `tas_zero_rate` - 设置为 `"true"` 授权呼叫但不计费（请参见 零计费路由）

零计费路由

任何扩展都可通过在通道上设置 `tas_zero_rate=true` 来使路由免费：

```
<extension name="Zero-Rated-Routes" continue="true">
  <condition field="${on_net_status}" expression="^true$"
break="never">
    <action application="set" data="tas_zero_rate=true"/>
  </condition>
  <condition field="${tas_destination_number}"
expression="^(1800|0800)" break="never">
    <action application="set" data="tas_zero_rate=true"/>
  </condition>
</extension>
```

在拨号计划运行时，CCR-Initial 已经发送，因此呼叫者仍然会进行信用检查，如果余额不足则仍然被禁止。该标志仅抑制使用报告 - 在接听时没有 CCR-Update，没有周期性 CCR-Update（因此没有通话中断），并且 CCR-Terminate 报告使用的单位为零。用户的余额没有受到影响。

这是一个普通的通道变量，故意与路线免费 □□□ 无关 - 在网仅仅是一个例子。要完全取消对某个目的地的收费（根本不进行授权），请使用 `online_charging.skipped_regex`。有关完整比较，请参见 Online Charging。

紧急呼叫

紧急呼叫通过 `emergency_call_codes` 配置参数进行控制，并在呼叫授权期间自动检测。

配置

在您的 TAS 配置文件中配置紧急呼叫代码：

配置参数：

- `emergency_call_codes`：要检测的紧急服务号码列表
- 常见代码：“911”（美国）、“112”（欧盟）、“000”（澳大利亚）、“999”（英国）、“sos”
- 这些代码除了 SIP 紧急 URN（例如，`<urn:service:sos>`）外还会被检查
- 系统对目的地号码执行 精确匹配 比较

示例配置值：

- 美国部署：`["911", "933"]` - 911 用于紧急情况，933 用于测试
- 欧洲部署：`["112", "999"]`
- 澳大利亚部署：`["000", "106"]` - 000 用于紧急情况，106 用于文本中继
- 多区域：`["911", "112", "000", "sos"]`

紧急检测工作原理

系统检查两个条件：

1. **SIP URI** 紧急服务 **URN**：检测 `<urn:service:sos>` 或任何包含 "service:sos" 的 URI
2. 目的地号码匹配：将 `Caller-Destination-Number` 与配置的 `emergency_call_codes` 进行比较

如果 任一条件 为真，则该呼叫被分类为紧急呼叫。

处理流程

呼叫流程详细信息：

1. 呼叫到达 TAS
2. 授权模块检查目的地与紧急模式的匹配
3. 如果检测到紧急情况：
   - 呼叫类型设置为 `:emergency`
   - 使用 `mo_emergency_dialplan.xml` 模板
   - 通常绕过 OCS 授权
   - 呼叫被路由到 PSAP 网关
4. 记录指标，标签为 `call_type: emergency`

拨号计划路由

在 `priv/templates/mo_emergency_dialplan.xml` 中定义紧急呼叫的路由。该模板确定如何根据市场需求将呼叫路由到您的 PSAP（公共安全应答点）网关或 SIP URI。

示例紧急拨号计划：

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

最佳实践

- 始终在您的紧急代码列表中包含 **"sos"** 确保与 SIP URN 的兼容性

- 包括您管辖区内的所有本地紧急号码（例如，911、112、000、999）
- 定期使用呼叫模拟器测试紧急路由
- 绕过 **OCS** 确保紧急呼叫始终连接（通过 `skipped_regex` 配置）
- 配置 **PSAP** 网关 实现高可用性和冗余
- 监控紧急呼叫标 确保系统可靠性

IVR 菜单

IVR 菜单允许呼叫者在路由之前通过 DTMF 数字选择选项。菜单被定义为 XML 文件，并通过拨号计划使用 `ivr` 应用程序调用。

添加 **IVR** 菜单

1. 在 `hosts/<environment>/group_vars/ivr_menus/` 中创建一个 XML 文件：

```
<include>
  <menu name="my_menu"
        greet-
long="$${base_dir}/sounds/en/us/callie/misc/8000/my_greeting.wav"
        greet-
short="$${base_dir}/sounds/en/us/callie/misc/8000/my_greeting.wav"
        invalid-sound="silence_stream://250"
        exit-sound=""
        timeout="5000"
        max-failures="1"
        max-timeouts="1"
        digit-len="1">

    <!-- 每个数字可以顺序执行多个操作 -->
    <entry action="menu-exec-app" digits="1" param="playback
$${base_dir}/sounds/option1.wav"/>
    <entry action="menu-exec-app" digits="1" param="bridge
sofia/internal/100@$${domain}"/>

    <entry action="menu-exec-app" digits="2" param="playback
$${base_dir}/sounds/option2.wav"/>
    <entry action="menu-exec-app" digits="2" param="bridge
sofia/internal/200@$${domain}"/>

    <!-- 处理超时和无效输入 -->
    <entry action="menu-exec-app" digits="timeout" param="bridge
sofia/internal/default@$${domain}"/>
    <entry action="menu-exec-app" digits="invalid" param="bridge
sofia/internal/default@$${domain}"/>
  </menu>
</include>
```

2. 从您的拨号计划中调用菜单：

```
<extension name="my_ivr_extension">
  <condition field="destination_number" expression="^1234$">
    <action application="answer"/>
    <action application="ivr" data="my_menu"/>
  </condition>
</extension>
```

3. 通过 Ansible 部署

菜单属性

属性	描述
`name`	在拨号计划 `ivr` 应用程序中使用的菜单标识符
`greet-long`	首次进入时播放的音频文件
`greet-short`	无效输入后播放的音频文件
`timeout`	等待输入的毫秒数
`max-failures`	在触发 `invalid` 操作之前的无效输入次数
`max-timeouts`	在触发 `timeout` 操作之前的超时次数
`digit-len`	要收集的数字数量

条目操作

条目使用 `action="menu-exec-app"` 来执行拨号计划应用程序。具有相同数字的多���条目按顺序执行：

条目类型	描述
`digits="1"`	当呼叫者按下 `1` 时执行
`digits="timeout"`	当呼叫者没有按下任何内容时执行
`digits="invalid"`	当呼叫者按下未识别的数字时执行

常见应用程序：`playback`、`bridge`、`transfer`、`hangup`

从网内移动发起呼叫到网内移动终止用户

当一个用户拨打您网络中的另一个用户（网内呼叫）时，正确的方法是将 MO 呼叫路由回 TAS 进行 MT 处理。这确保被叫方获得完整的 MT 呼叫处理，包括呼叫转移、语音信箱、漫游的 MSRN 路由□及所有其他用户服务。

为什么将 **MO** 路由到 **MT**?

没有 **MT** 处理（直接路由）：

- 被叫方的呼叫转移设置被忽略
- 没有无应答时的语音信箱
- 没有漫游用户的 MSRN 路由
- 缺少用户服务逻辑

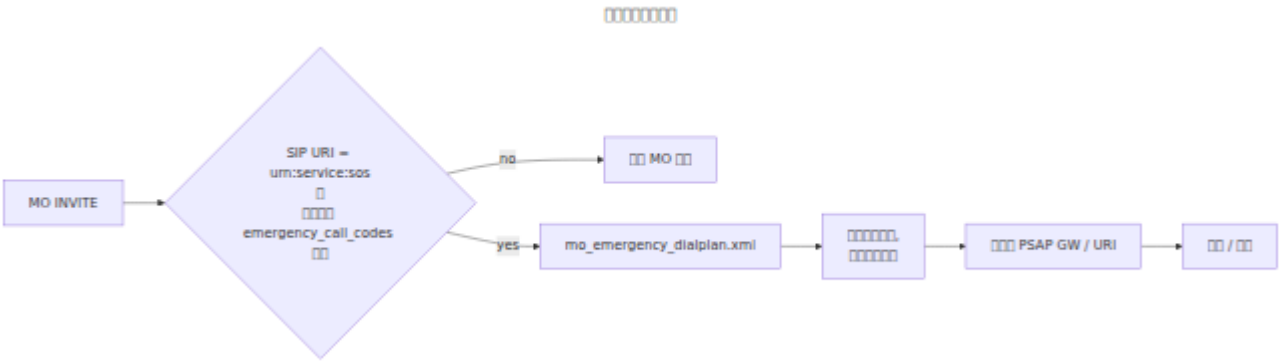
有 **MT** 处理（路由回 TAS）：

- 完整的呼叫转移支持（CFU、CFB、CFNRy、CFNRc）
- 忙碌/无应答时的语音信箱
- CS 漫游用户的 MSRN 路由
- 完整的用户服务体验
- 正确的呼叫状态跟踪

实施

MO 拨号计划检查目的地是否在网内（由您的 TAS 提供），如果是，则将呼叫路由回 TAS 本身。TAS 将其视为新的 MT 呼叫，并通过 `mt_dialplan.xml` 模板进行处理。

示例拨号计划片段：



```
<extension name="On-Net-Route">
  <condition field="${on_net_status}" expression="true">
    <action application="log" data="DEBUG 网内 MO 呼叫 - 路由回 TAS" />

    <!-- 清理内部路由的头部 -->
    <action application="set" data="sip_copy_multipart=false"/>
    <action application="set" data="sip_h_Request-Disposition=no-fork

    <!-- 路由回 TAS（变为 MT 呼叫） -->
    <action application="bridge"
      data="{absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU',originate_retries=1,originate_timeout=60,sip_invite
/>
    <action application="hangup" data="" />
  </condition>
</extension>
```

关键参数：

- `${sip_local_network_addr}` - TAS IP 地址（例如，`10.179.3.60`）
- `${tas_destination_number}` - 被叫方的 MSISDN
- `sip_invite_call_id=${sip_call_id}` - 保留呼叫 ID 便跟踪
- `sip_copy_multipart=false` - 防止多部分消息复制
- `sip_h_Request-Disposition=no-fork` - 确保顺序处理

呼叫流程：

重要配置：

- TAS IP（例如，`10.179.3.60`）必须在您的 `allowed_sbc_source_ips` 配置列表中
- 这允许 TAS 从自身接收呼叫进行 MT 处理

- 如果没有此项，TAS 将拒绝来自未授权源的呼叫

MSRN 用于 2G/3G 漫游用户的使用

当用户在 2G/3G 电路交换（CS）网络中漫游时，TAS 必须获取 MSRN（移动台漫游号码）□将来电路由到用户的当前位置。本节解释 MSRN 检索和路由的工作原理。

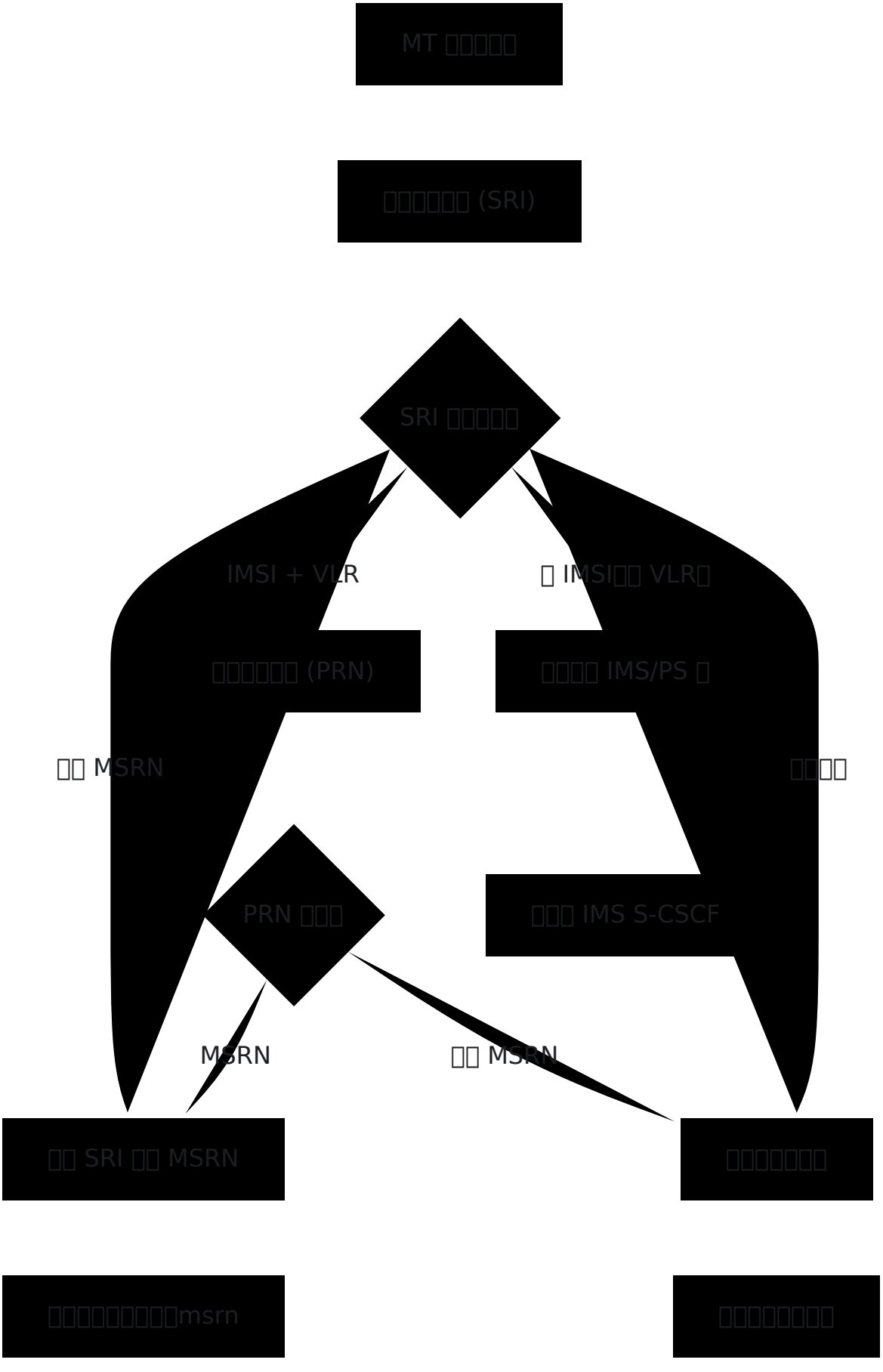
什么是 MSRN?

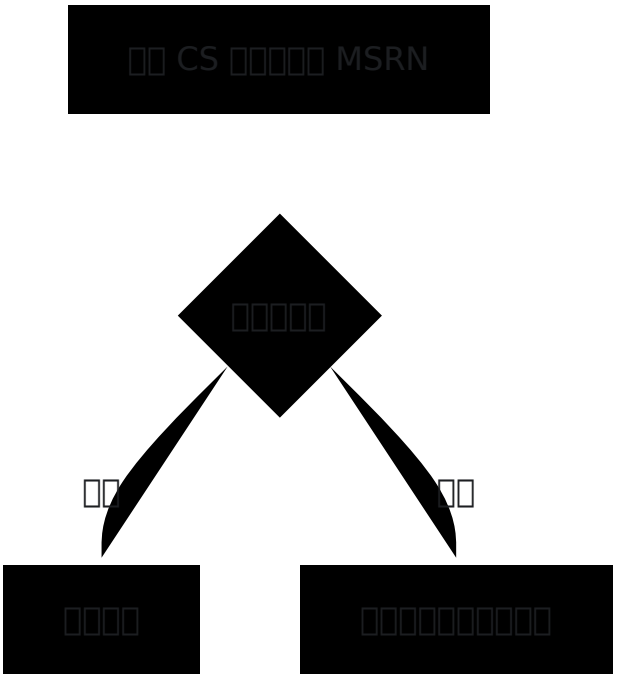
MSRN（移动台漫游号码）是由访问网络的 VLR（访客位置寄存器）分配的临时路由号码，用于将呼叫路由到漫游用户。它充当一个临时目 ��地号码，指向用户在 CS 网络中的当前位置。

MSRN 检索流程

TAS 通过 SS7 MAP（移动应用部分）协议使用两步流程检索 MSRN 数据：

□□□□□□ MT □□□ MSRN □□


MT □□□□□□
□□□□□□□□ (SRI)
SRI □□□□□□
IMSI + VLR
□ IMSI□□ VLR□
□□□□□□□□ (PRN)
□□□□□ IMS/PS □
□□ MSRN
□□□□
PRN □□□
□□□□ IMS S-CSCF
MSRN
□□ MSRN
□□ SRI □□ MSRN
□□□□□□□□□
□□□□□□□□□□□msrn
□□□□□□□□□□



实施细节

第 **1** 步：发送路由信息 **(SRI)**

TAS 通过 SS7 MAP 查询 HLR □获取被叫用户的路由信息。

SRI 响应场景：

1. **SRI** 中直接包含 **MSRN** - 漫游用户的 MSRN 已可用
    - 响应包括：MSISDN、GMSC、IMSI 和 MSRN
    - 示例 MSRN：`61412345678`（澳大利亚手机号码格式）
2. **IMSI + VLR** 号码 - 用户在 CS 网络中注册（需要 PRN）
    - 响应包括：MSISDN、GMSC、IMSI 和 MSC/VLR 号码
    - 表示用户在 CS 网络中，但必须请求 MSRN
3. 仅 **IMSI**（无 **VLR**） - 用户不在 CS 网络中（仅 IMS/PS）
    - 响应包括：MSISDN、GMSC、IMSI
    - 表示用户仅在 IMS/4G 中注册，不在 CS 网络中
4. 呼叫转移激活 - SRI 返回转移信息

- 响应包括转移原因（无条件、忙碌、无应答、不可达）
- 响应包括转移到的号码

第 **2** 步：提供漫游号码 **(PRN) -** 如果需要

如果 SRI 返回 IMSI + VLR 但没有 MSRN，TAS 会向 VLR 发送 PRN 请求获取 MSRN。

VLR 从其池中分配一个临时 MSRN，并将其返回给 TAS。该 MSRN 仅在此特定呼叫设置期间有效。

示例 **PRN** 响应： MSRN `61412345678`

拨号计划变量：`msrn`

一旦 TAS 通过 SS7 MAP 检索到 MSRN，它将 MSRN 设置为拨号计划变量。MT 拨号计划可使用此变量。

变量： `${msrn}`

- 类型：字符串（E.164 格式的号码，不带前导 +）
- 示例： `"61412345678"`（澳大利亚机格式）
- 用法：路由呼叫到 CS 漫游用户
- 由以下设置：在 MT 呼叫处理期间的 HLR 数据检索过程

在 mt_dialplan.xml 中路由到 MSRN

MSRN 变量用于 MT 拨号计划模板，将呼叫路由到漫游用户。

拨号计划逻辑：

1. 检查 **MSRN**：扩展检查 `msrn` 变量是否已设置（包含数字）
2. 设置超时参数：
   - 进度超时：10 秒接收早期媒体
   - 桥接应答超时：使用用户配置的无应答计时器
3. 桥接到 **MSRN**：通过 CS 网关将呼叫路由到 MSRN
   - 使用 `ignore_early_media=ring_ready` 确保一致的振铃回声
   - 编解码器优先级：AMR（移动）、PCMA/PCMU（固定电话）
   - 网关：`sofia/gateway/CS_Gateway/+${msrn}`
4. 失败时的回退：如果桥接失败，则路由到呼叫转移目的地

示例拨号计划片段：

```
<extension name="Route-to-CS-MSRN" continue="false">
  <condition field="msrn" expression="^(\d+)$">
    <!-- □置超时 -->
    <action application="set" data="progress_timeout=10" />
    <action application="set" data="bridge_answer_timeout=${no_reply_

    <!-- 通过 CS 网关桥接到 MSRN -->
    <action application="bridge"
      data="
{ignore_early_media=ring_ready,absolute_codec_string='AMR,PCMA,PCMU',
/>

    <!-- 回退到语音信箱/呼叫转移 -->
    <action application="bridge"
      data="sofia/internal/${call_forward_not_reachable_destination}@
  </condition>
</extension>
```

关键点

1. **MSRN** 是临时的 - 仅在呼叫设置期间有效
2. 仅限 **CS** 网络 - MSRN 用于 2G/3G 漫游，不适用于 VoLTE/IMS 漫游
3. **MT** 流中的优先级 - MSRN 检查在标准 IMS 路由之前进行
4. 回退到转移 - 如果 MSRN 桥接失败，则路由到呼叫转移目的地
5. **HLR** 优先于 **Sh** - HLR 中的 MSRN 优先于 Sh 用户数据

配置

必须在 TAS 配置中启用 SS7 MAP 集成：

所需设置：

- **enabled**：设置为 `true` □启用 SS7 MAP 查询
- **http_map_server_url_base**：您的 SS7 MAP 网关的 URL（例如，`"http://10.1.1.100:5001"`）
- **gmsc**：用于 SRI/PRN 请求的网关 MSC 号码（例如，`"61400000000"`）

- **timeout_ms**：查询超时（默认：5000 毫秒）

有关完整配置详细信息，请参见 **SS7 MAP Documentation**。

呼叫转移数据使用

呼叫转移设置决定了当主要目的地不可用时，呼叫如何路由。TAS 从两个来源检索呼叫转移数据：Sh 接口（HSS）和 SS7 MAP（HLR），其中 HLR 数据优先。

呼叫转移类型

系统支持四种类型的呼叫转移：

转移类型	变量	激活时机
无条件呼叫转移 **(CFU)**	`call_forward_all_destination`	始终立即转移所有呼叫
忙碌呼叫转移 **(CFB)**	`call_forward_not_reachable_destination`	用户线路忙碌
无应答呼叫转移 **(CFNRy)**	`call_forward_not_reachable_destination`	用户在超时内未应答
不可达呼叫转移 **(CFNRc)**	`call_forward_not_reachable_destination`	用户不可达/离线

数据来源

1. Sh 接口（HSS）

静态配置 存储在 HSS 用户配置文件中。

TAS 在呼叫处理期间通过 Sh 接口从 HSS 检索呼叫转移设置。这些是为用户配置的/默认设置。

示例检索数据：

- `call_forward_all_destination`: CFU 目的地（例如，`"61412345678"`）
- `call_forward_not_reachable_destination`: CFB/CFNRy/CFNRc 目的地（例如，`"61487654321"`）
- `no_reply_timer`: CFNRy 触发前的秒数（例如，`"20"`）

2. SS7 MAP（HLR）

实时数据 来自 HLR，可能与 HSS 不同，如果用户通过 USSD/MMI 代码更改了设置（例如，拨打 `*21*` 代码）。

TAS 在呼叫设置期间通过 SS7 MAP 查询 HLR 获取当前/活动的转移设置。

HLR 转移响应包括：

- **forwarded_to_number**：转移的目的地号码（例如，`"61412345678"`）
- **reason**：转移类型（无条件、忙碌、无应答、不可达）
- **notification flags**：是否通知呼叫方、转移方等。

映射到拨号计划变量：

- 如果原因是 无条件 → 设置 `call_forward_all_destination`
- 如果原因是 忙碌、无应答或不可达 → 设置 `call_forward_not_reachable_destination`

变量合并优先级

HLR 数据优先于 **Sh** 数据 当两者都存在时。

TAS 在 MT 呼叫处理期间从两个来源检索用户数据：

1. 首先，通过 Sh 接口从 HSS 检索静态配置
2. 然后，通过 SS7 MAP 查询 HLR 获取实时设置
3. 合并数据，HLR 值优先于 Sh 值

这确保了最近的用户更改（通过 USSD 代码）得到尊重，即使 HSS 尚未更新。

拨号计划变量

在 **MT** 呼叫中可用：

变量	类型	示例	[
`call_forward_all_destination`	字符串	`"61412345678"`	CFU 目的地[
`call_forward_not_reachable_destination`	字符串	`"61487654321"`	CFB/CFN 目的地
`no_reply_timer`	字符串	`"20"`	CFNRy 的[

默认值：

- 如果未配置：`"none"`（字符串）
- 检查存在性：使用正则表达式 `^(?!none$).*` 匹配任何值，除了 "none"

在 **mt_dialplan.xml** 中的呼叫转移

示例 **1**：无条件呼叫转移 **(CFU)**

立即将所有来电路由到转移目的地。转移目的地通常是一个网外号码，因此使用外部网关。

使用的网关：`sofia/gateway/ExternalSIPGateway`（您的 PSTN/互联网关）

模板示例：

```
<extension name="Check-Call-Forward-All">
  <condition field="${call_forward_all_destination}" expression="^(?!
    <action application="log" data="INFO 呼叫转移全部设置为重定向到 ${call_forwarc

    <!-- 设置历史信息头以进行呼叫转移 -->
    <action application="set" data="sip_h_History-Info=<sip:${destina

    <!-- 标记呼叫 ID 以指示呼叫转移类型 -->
    <action application="set" data="sip_call_id=${sip_call_id};CALL_F

    <!-- 桥接到网外转移目的地 -->
    <action application="bridge"
      data="{absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU',originate_retries=1,originate_timeout=60}sofia/gate
/>
  </condition>
</extension>
```

关键点：

- 使用外部网关，因为转发通常是到非网内号码
- 使用 `;CALL_FORWARD_UNCONDITIONAL` 标记呼叫 ID □进行跟踪
- 设置 `History-Info` 头□识别原始被叫号码
- 示例：用户 `61412345678` 将 CFU 转发至 `61487654321` - 所有呼叫立即转发

示例 **2**：呼叫转发无应答/不可达

当桥接到主要目的地失败时（用户未应答、忙或不可达）用作后备。

示例拨号计划片段：

```
<!-- 在桥接到 MSRN 或 IMS 失败后... -->
<action application="log" data="INFO Failed to bridge Call - Routing

<!-- 设置 History-Info 以指示转发 -->
<action application="set" data="sip_h_History-Info=<sip:${destinatior

<!-- 路由到转发目的地 -->
<action application="bridge"
  data="
{absolute_codec_string='AMR,PCMU,PCMA',originate_timeout=65}sofia/gat
/>
```

示例场景：

- 用户 `61412345678` 将 CFNRy 转发至语音信箱号码 `61487654321`
- 来电尝试联系用户
- 20 秒后无应答（no_reply_timer）
- 呼叫转发至 `61487654321`，并保留 History-Info 头□保持原始目的地

History-Info 头

`History-Info` SIP 头跟踪呼叫转发：

```
<action application="set" data="sip_h_History-Info=
<sip:${destination_number}@${ims_domain}>;index=1.1" />
```

目的：

- 指示呼叫最初是针对 `${destination_number}`
- 允许下游系统识别转发的呼叫
- 被语音信箱系统用于存储到正确的邮箱

语音信箱路由中的示例：

```
<extension name="Voicemail Route" continue="false">
  <condition field="${tas_destination_number}"
expression="^(555121|555122)$">
    <!-- 从 History Info 中提取电话号码 -->
    <action application="set"
data="history_info_value=${sip_i_history_info}"/>
    <action application="log" data="DEBUG Called Voicemail Deposit
Number for ${history_info_value}" />

    <!-- 将语音信箱存储到原始被叫方，而不是语音信箱号码
         (TAS ESL 语音信箱; 见 voicemail.md) -->
    <action application="set" data="tas_vm_mode=deposit"/>
    <action application="set"
data="tas_vm_mailbox=${history_info_value}"/>
    <action application="set"
data="tas_vm_caller=${effective_caller_id_number}"/>
    <action application="socket" data="127.0.0.1:8084 async
full"/>
  </condition>
</extension>
```

工作原理：

- 语音信箱服务号码：`555121`，`555122`（通用短码）
- 当呼叫被转发至语音信箱时，History-Info 包含原始目的地
- 语音信箱系统从 History-Info 头中提取原始号码
- 语音信箱存储到原始被叫方的邮箱，而不是语音信箱服务号码

最佳实践

1. 始终检查 **"none"** - 使用正则表达式 `^(?!none$).*` 避免路由到字面字符串 "none"
2. 设置 **History-Info** - 转发时始终设置□确保正确的呼叫跟踪
3. 使用 **continue_on_fail** - 如果主要路由失败，允许后备转发
4. 调整 **CLI** 格式 - 国家与国际前缀格式（见来电显示部分）
5. 测试转发循环 - 确保转发目的地不会产生路由循环

移动终止呼叫分叉（多 **IMSI** 号码）

默认情况下，TAS 将移动终止（MT）呼叫传递给单个设备。它向 S-CSCF 发送 `Request-Disposition: no-fork`，因此 S-CSCF 仅将呼叫传递给一个注册的联系人。

一个 多 **IMSI MSISDN** 是一个号码，由多个设备在不同 SIM 上共享（例如，一部手机和一块智能手表）。对于这些号码，TAS 允许呼叫分叉到每个注册的设备：它不发送 `no-fork`，因此 S-CSCF 可能将呼叫传递□□□所有注册的联系人。

TAS 如何决定

TAS 根据 HSS 数据为每个呼叫做出决定。在终止腿上，TAS 通过 Sh 接口 读取被叫方。HSS 为共享该号码的每个成员 IMSI 返回一个 `<IMSPrivateUserIdentity>`。TAS 计算这些身份。多个成员标记为多 IMSI 共享号码，因此 TAS 允许分叉。普通号码返回一个成员，因此 TAS 保持单联系人传递。此功能无需 TAS 配置。请参见 多 IMSI MSISDN 分叉。

对于被叫号码，MT 流设置 FreeSWITCH 通道变量 `tas_mt_allow_forking`：

- `tas_mt_allow_forking = "true"` - HSS 报告该号码有多个成员 IMSI（多 IMSI 共享号码）。拨号计划不发送 `no-fork`。呼叫可□分叉。
- `tas_mt_allow_forking = "false"`（默认） - HSS 报告一个成员。拨号计划发送 `Request-Disposition: no-fork`。呼叫传递给一个联系人。

拨号计划网关

`MT-Fork-Control` 扩展在 `mt_dialplan.xml` 中仅在不允许分叉时应用 `no-fork`：

```
<extension name="MT-Fork-Control" continue="true">
  <condition field="${tas_mt_allow_forking}"
expression="^(?!true$).*">
    <action application="set" data="sip_h_Request-Disposition=no-
fork"/>
  </condition>
</extension>
```

要求

- **S-CSCF** 必须为每个公共身份存储多个联系人。如果 S-CSCF 仅保留一个联系人（在每次注册时覆盖），则只有一个设备可□联系，分叉没有效果。配置 S-CSCF 注册器□允许每个记录地址多个联系人。

该号码必须在 HSS 中配置为共享号码（相同的 MSISDN 分配给每个设备的用户）。请参见 HSS "共享公共用户身份" 文档。

来电显示（CLI）管理

TAS 管理呼叫线路识别（CLI）的呈现和格式化，处理隐私请求、前缀标准化和网络特定格式要求。

CLI 变量

拨号计划中的核心 CLI 变量：

变量	用法	示例
`msisdn`	用户的号码（无 +）	`"61412345678"`
`msisdns`	用户的所有号码，逗号分隔（无 +）。屏蔽呈现的 CLI 或按号码路由。	`"61412345678,61412345679`
`effective_caller_id_number`	显示的来电号码	`"+61412345678"` 或 `"anonymous"`
`effective_caller_id_name`	显示的来电名称	`"+61412345678"` 或 `"anonymous"`
`origination_caller_id_number`	外发腿的 CLI	`"+61412345678"`
`caller_id_number`	标准 FreeSWITCH CLI 变量	`"+61412345678"`
`sip_from_user`	SIP From 头的用户部分	`"0412345678"` 或 `"+61412345678"`
`cli_withheld`	隐私标志	`"true"` 或 `"false"`（字符串）
`origination_privacy`	隐私设置	`"hide_number"`

CLI 隐私（保留/匿名）

检测方法

TAS 通过三种方法检测 CLI 隐私请求:

1. 拨打号码中的阻止前缀

用户在目标号码前拨打前缀阻止其来电 ID。

常见前缀：

- `*67` - 北美标准
- `#31#` - 欧洲/GSM 标准
- `1831` - 替代格式

TAS 检查拨打的号码是否□任何配置的阻止 CLI 前缀开头。如果检测到，则将 `cli_withheld` 变量设置为 `"true"`。

示例：用户拨打 `*67555 1234` - 检测到并移除 `*67` 前缀，呼叫继续到 `5551234`，CLI 被保留。

2. From 头中的匿名

用户设备（UE）在 SIP From 头中将来电名称设置为 "anonymous"。

TAS 检查 `Caller-Orig-Caller-ID-Name` 字段（不区分大小写）是否包含字符串 "anonymous"。如果找到，则将 `cli_withheld` 设置为 `"true"`。

3. SIP 隐私头

S-CSCF 可能在 SIP INVITE 中设置 `Privacy: id` 头，拨号计划会遵循这些头。

拨号计划实现

拨号计划检查 `cli_withheld` 变量，并相应地设置所有与 CLI 相关的变量。

示例拨号计划片段：

```
<extension name="Manage-Caller-ID" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <!-- CLI 被保留 - 设置为匿名 -->
    <action application="log" data="DEBUG CLI withheld detected"
/>
    <action application="set"
data="effective_caller_id_name=anonymous" />
    <action application="set"
data="effective_caller_id_number=anonymous" />
    <action application="set"
data="origination_caller_id_number=anonymous" />
    <action application="set"
data="origination_privacy=hide_number" />

    <!-- CLI 未被保留 - 使用正常 MSISDN -->
    <anti-action application="log" data="DEBUG CLI is normal (not
withheld)" />
    <anti-action application="set"
data="effective_caller_id_number=${msisdn}" />
  </condition>
</extension>
```

注意：此扩展使用 `continue="true"`，因此即使设置了 CLI，呼叫处理仍会继续到路由扩展。

CLI 格式：国家与国际

不同的目的地可能根据您的网络要求需要不同的 CLI 格式。

示例：国家格式

对于您国家内的国内呼叫，您可能需要呈现不带国家代码的 CLI。

示例拨号计划片段（澳大利亚移动网络）：

```
<extension name="Outgoing-Call-CLI-National" continue="true">
  <condition field="${msisdn}" expression="^61(.*)$">
    <action application="log" data="Setting source CLI to $1 for
national" />
    <action application="set"
data="effective_caller_id_number=$1"/>  <!-- 0412345678 -->
    <action application="set" data="effective_caller_id_name=$1"/>
    <action application="set" data="sip_from_user=$1"/>
    <action application="set" data="sip_cid_type=pid"/>
  </condition>
</extension>
```

工作原理：

- 正则表达式 `^61(.*)$` 捕获国家代码 `61` 后的所有内容
- 输入：`msisdn="61412345678"` → 输出：`$1="412345678"` 或 `"0412345678"`
- 为国内呼叫□国家格式呈现 CLI

示例：国际格式

对于国际呼叫，□完整的 E.164 格式呈现 CLI，并带有 `+` 前缀。

示例拨号计划片段：

```
<extension name="Outgoing-Call-CLI-International" continue="true">
  <condition field="${tas_destination_number}"
expression="^61(.*)$">
    <action application="log" data="Call is to national" />

    <!-- 当目的地不是国家时执行反向操作 -->
    <anti-action application="log" data="Setting source CLI for
international" />
    <anti-action application="set"
data="effective_caller_id_number=+${msisdn}"/>  <!-- +61412345678
-->
    <anti-action application="set"
data="effective_caller_id_name=+${msisdn}"/>
    <anti-action application="set"
data="sip_from_user=+${msisdn}"/>
    <anti-action application="set" data="sip_cid_type=pid"/>
  </condition>
</extension>
```

工作原理：

- 条件检查目的地是否□国家前缀开头（例如，澳大利亚的 `61`）
- `<anti-action>` 在条件不匹配时执行（国际呼叫）
- 为国际呼叫添加 `+` 前缀□获得完整的 E.164 格式

呼叫转发的 CLI 格式

在路由到呼叫转发目的地时，您可能需要根据转发到网内或非网内号码来调整 CLI 格式。

示例：调整呼叫转发的 **CLI** 前缀

```
<!-- 如果需要，调整转发目的地的 CLI 格式 -->
<action application="set"
data="effective_caller_id_number=${effective_caller_id_number:3}"/>
<action application="set"
data="effective_caller_id_name=${effective_caller_id_name:3}"/>
```

字符串切片：`${variable:N}` 移除前 N 个字符

- 输入：`effective_caller_id_number="+61412345678"`，使用 `:3` → 输出：`"412345678"`
- 输入：`effective_caller_id_number="+61412345678"`，使用 `:1` → 输出：`"61412345678"`

用例：

- 对于国家转发，移除 `+`：使用 `:1`
- 对于本地格式，移除国家代码：使用适当的偏移量（`+61` 使用 `:3`，`+1` 使用 `:2` 等）

SIP P-Asserted-Identity (PAI)

`sip_cid_type=pid` 设置控制来电 ID 的呈现：

```
<action application="set" data="sip_cid_type=pid"/>
```

效果：

- 设置 SIP `P-Asserted-Identity` 头，包含来电信息
- 用于可信网络来电 ID 断言
- IMS 网络的标准

移除专有头

为了防止泄露内部网络信息，拨号计划应在将呼叫路由到非网内之前移除专有或内部头。

示例：在外部路由之前清理头

```
<action application="set" data="sip_copy_multipart=false"/>
<action application="set" data="sip_copy_custom_headers=false"/>
<action application="unset" data="sip_h_P-Internal-Correlation-
ID"/>
<action application="unset" data="sip_h_P-Access-Network-Info"/>
<!-- 根据需要添加更多供应商特定或内部头 -->
```

目的：

防止内部路由数据到达外部网络
移除供应商特定的专有头
隐私和安全最佳实践
减少 SIP 消息大小

常见需要移除的头：

内部关联/跟踪 ID
访问网络信息（可能泄露网络拓扑）
供应商特定的 P 头
仅供内部使用的自定义应用头

最佳实践

1. 对 CLI 扩展使用 continue="true" - 允许多个 CLI 格式化规则
2. 设置 sip_cid_type=pid - 符合 IMS 网络要求
3. 测试 CLI 保留 - 验证 *67 和#31# 前缀是否有效
4. 根据目的地格式化 - 国家与国际 CLI 格式
5. 移除专有头 - 防止内部数据泄露
6. 优雅处理匿名 - 显示和路由应与匿名 CLI 一起工作

桥接到网关

TAS 使用 FreeSWITCH 的 bridge 应用程序将呼叫桥接到外部网关（IMS 核心、PSTN 等），并仔细配置编解码器协商、超时处理和重试逻辑的参数。

网关配置

网关被配置为 SIP 中继到外部系统。TAS 使用单个 SIP 接口处理所有流量，为不同目的地定义不同的网关。

示例网关配置：

```
<gateway name="CS_Gateway">
  <param name="proxy" value="10.1.1.100:5060"/>
  <param name="register" value="false"/>
  <param name="caller-id-in-from" value="true"/>
  <param name="extension-in-contact" value="true"/>
</gateway>
```

请参见 配置□南 □获取完整的网关设置。

桥接语法

呼叫通过□下语法桥接到网关：

基本语法：

```
<action application="bridge"
data="sofia/gateway/GATEWAY_NAME/DESTINATION_NUMBER" />
```

带参数：

```
<action application="bridge" data="
{param1=value1,param2=value2}sofia/gateway/GATEWAY_NAME/DESTINATION_N
/>
```

其中 `GATEWAY_NAME` 是您配置中定义的网关名称（例如，`IMS_Core`，`PSTN_Primary`，`International_Gateway`）。

桥接参数

编解码器选择

`absolute_codec_string` - 优先的编解码器列表用于协商：

```
<action application="bridge" data="
{absolute_codec_string='AMR,PCMA,PCMU'}sofia/gateway/IMS_Gateway/+${m
/>
```

编解码器优先级顺序：

1. **AMR**（自适应多速率） - 针对移动优化，优先用于蜂窝网络
2. **PCMA**（G.711 a-law） - 欧洲/国际的固定线路标准
3. **PCMU**（G.711 μ-law） - 北美的固定线路标准

模板使用：`priv/templates/mt_dialplan.xml:80`，`mo_dialplan.xml:124`，`mo_dialplan.xml:202`

超时配置

`originate_timeout` - 等待应答的最大秒数（包括响铃）：

```
<action application="set" data="originate_timeout=60"/>
<action application="bridge" data="
{originate_timeout=60}sofia/gateway/CS_Gateway/+${msisdn}" />
```

`progress_timeout` - 等待 180/183（早期媒体/响铃）的秒数：

```
<action application="set" data="progress_timeout=10" />
```

`bridge_answer_timeout` - 在响铃开始后等待 200 OK 的秒数：

```
<action application="set"
data="bridge_answer_timeout=${no_reply_timer}" />
```

`leg_progress_timeout` - 每条腿的进度超时：

```
<action application="set"
data="leg_progress_timeout=${no_reply_timer}" />
```

模板示例：`priv/templates/mt_dialplan.xml:73-76`

```
<action application="set" data="progress_timeout=10" />
<!-- 我们在 INVITE 和 200 OK 之间等待多长时间（包括响铃） -->
<action application="set"
data="bridge_answer_timeout=${no_reply_timer}" />
<action application="set"
data="leg_progress_timeout=${no_reply_timer}" />
```

变量：`${no_reply_timer}` 来自用户数据（通常为 20-30 秒）

重试和失败处理

`originate_retries` - 重试次数：

```
<action application="bridge" data="
{originate_retries=1}sofia/gateway/CS_Gateway/+${msisdn}" />
```

`continue_on_fail` - 在桥接失败后继续拨号计划执行：

```
<action application="set" data="continue_on_fail=true" />
<action application="bridge" data="
{continue_on_fail=true}sofia/gateway/CS_Gateway/+${msisdn}" />
<!-- 如果桥接失败，后续操作执行 -->
<action application="log" data="INFO Bridge failed - routing to
voicemail" />
```

`hangup_after_bridge` - 当 B-leg 挂断时挂断 A-leg：

```
<action application="set" data="hangup_after_bridge=true"/>
```

早期媒体处理

`ignore_early_media` - 控制早期媒体行为：

```
<action application="set" data="ignore_early_media=ring_ready" />
<action application="bridge" data="
{ignore_early_media=ring_ready}sofia/gateway/CS_Gateway/+${msisdn}"
/>
```

选项：

- `ring_ready` - 生成本地响铃，忽略远程早期媒体
- `true` - 完全忽略早期媒体
- `false`（默认） - 传递早期媒体（公告、音调）

为什么使用 **`ring_ready`**？ - 防止呼叫者听到来自远程网络的网络公告或音调

模板示例： `priv/templates/mt_dialplan.xml:78-79`

```
<action application="set" data="ignore_early_media=ring_ready" />
<action application="bridge" data="
{ignore_early_media=ring_ready,...}sofia/gateway/CS_Gateway/+${msrn}"
/>
```

网内与非网内呼叫者处理：

```
<extension name="Route-to-IMS-Sub-Early-Media" continue="true">
  <condition field="${on_net_caller}" expression="true">
    <!-- 网内呼叫者 - 使用 ring_ready -->
    <action application="log" data="INFO On-net caller
${effective_caller_id_number} - using
ignore_early_media=ring_ready"/>
    <action application="set"
data="ignore_early_media=ring_ready"/>

    <!-- 非网内呼叫者 - 提供即时响铃 -->
    <anti-action application="log" data="INFO Off-net caller
${effective_caller_id_number} - setting instant ringback"/>
    <anti-action application="set" data="instant_ringback=true"/>
    <anti-action application="set" data="ringback=${fr-ring}"/>
    <anti-action application="set" data="transfer_ringback=${fr-
ring}"/>
  </condition>
</extension>
```

注意：`${on_net_caller}` 变量根据您网络的用户编号计划设置。您还可使用正则表达式模式来匹配您的特定号码范围。

来电 **ID** 参数

`sip_cid_type=pid` - 使用 P-Asserted-Identity 作为来电 ID:

```
<action application="set" data="sip_cid_type=pid" />
<action application="bridge" data="
{sip_cid_type=pid}sofia/gateway/CS_Gateway/+${msisdn}" />
```

常见桥接模式

模式 **1**: 通过 **IMS** 域路由到 **IMS** 用户

通过将 MT 呼叫发送到 IMS 域（S-CSCF 将解析并路由）来路由到 IMS 用户。

模板示例:

```
<extension name="Route-to-IMS-Sub" continue="false">
  <condition field="destination_number" expression="^(.*)$">
    <action application="set" data="continue_on_fail=true" />
    <action application="set" data="hangup_after_bridge=true"/>
    <action application="set" data="progress_timeout=10" />

    <!-- 我们在 INVITE 和 200 OK 之间等待多长时间（包括响铃） -->
    <action application="set" data="bridge_answer_timeout=${no_reply_
    <action application="set" data="leg_progress_timeout=${no_reply_t

    <!-- 将呼叫发送到 IMS ??? (S-CSCF 解析) -->
    <action application="set" data="ignore_early_media=ring_ready" />
    <action application="set" data="sip_cid_type=pid" />

    <action application="bridge"
      data="{absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU',ignore_early_media=ring_ready,continue_on_fail=true
/>

    <!-- 如果桥接失败，则后备到呼叫转发 -->
    <action application="log" data="INFO Failed to bridge Call - Rout
    <action application="set" data="sip_h_History-Info=<sip:${destina
    <action application="set" data="sip_h_Diversion=<sip:${destinatio

    <!-- 路由到非网关的呼叫转发 -->
    <action application="bridge"
      data="{absolute_codec_string='AMR-WB,AMR,PCMU,PCMA',originate_t
  </condition>
</extension>
```

关键点：

- 路由到 `${msisdn}@${ims_domain}`（例如，`5551234567@ims.mnc001.mcc001.3gppnetwork.org`）
- IMS 核心（S-CSCF/I-CSCF）处理最终路由到订阅者
- `ignore_early_media=ring_ready` 提供一致的回铃音
- 在失败时，使用外部网关进行非网呼叫转接
- 设置 `History-Info` 和 `Diversion` 头□跟踪呼叫转接

模式 **2**：路由到 **MSRN**（**CS** 漫游）

通过 CS 网络路由到漫游订阅者：

模板：`priv/templates/mt_dialplan.xml:67-80`

```
<extension name="Route-to-CS-MSRN" continue="false">
  <condition field="msrn" expression="^(\d+)$">
    <action application="set" data="continue_on_fail=true" />
    <action application="set" data="hangup_after_bridge=true"/>
    <action application="set" data="progress_timeout=10" />
    <action application="set" data="bridge_answer_timeout=${no_reply_
    <action application="set" data="leg_progress_timeout=${no_reply_t

    <!-- 通过网关将呼叫发送到 MSRN -->
    <action application="set" data="ignore_early_media=ring_ready" />
    <action application="set" data="sip_cid_type=pid" />
    <action application="bridge"
      data="
{ignore_early_media=ring_ready,absolute_codec_string='AMR,PCMA,PCMU',
/>
  </condition>
</extension>
```

模式 **3**：网内路由（**MO** 到 **MT** 通过 **TAS**）

当订阅者呼叫另一位网内订阅者时，呼叫必须路由回 TAS 进行完整的 MT 处理。此模式对于确保网内呼叫获得与外部 MT 呼叫相同的服务待遇至关重要。

为什么需要此模式：

如果不路由回 TAS，网内呼叫将完全绕过 MT 处理，这意味着：

- 呼叫转接设置将不被遵守
- 忙音/无人接听的语音信箱将无法工作
- 漫游订阅者的 MSRN 路由将失败
- 订阅者服务逻辑将被跳过
- 呼叫跟踪和 CDR 将不完整

通过将 MO 呼叫作为新的 MT 呼叫路由回 TAS，目标订阅者获得完整的服务待遇。

模板示例：

```
<extension name="On-Net-Route">
  <condition field="${on_net_status}" expression="true">
    <action application="log" data="DEBUG On-Net MO call - Routing ba

    <!-- 清理内部路由的头部 -->
    <action application="set" data="sip_copy_multipart=false"/>
    <action application="set" data="sip_h_Request-Disposition=no-fork

    <!-- 路由回 TAS（成为 MT 呼叫） -->
    <action application="bridge"
      data="{absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU',originate_retries=1,originate_timeout=60,sip_invite
/>
    <action application="hangup" data="" />
  </condition>
</extension>
```

工作原理：

1. **MO** 呼叫到达：订阅者 A 呼叫订阅者 B（均为网内）
2. 检查网内状态：TAS 通过 `${on_net_status}` 变量确定目标为网内
3. 路由到 **TAS**：桥接到 `sofia/internal/${tas_destination_number}@${sip_local_network_addr}`
   - 使用 TAS 自己的 IP 地址作为目标
   - 保留原始呼叫 ID 便跟踪
4. **MT** 处理：TAS 将呼叫作为新的 MT 呼叫接收并处理 `mt_dialplan.xml`
   - 检查呼叫转接设置（CFU、CFB、CFNRy、CFNRc）
   - 查询漫游订阅者的 MSRN
   - 路由到 IMS 域或适当转接
5. 完整服务：目标订阅者获得完整的 MT 处理

关键点：

- 路由到 `${sip_local_network_addr}`（TAS IP 地址，例如，`10.179.3.60`）
- 呼叫作为 MT 呼叫重新处理到目标订阅者
- 使用 `sip_invite_call_id` 参数保留呼叫 ID 便端到端跟踪
- 启用所有 MT 功能：呼叫转接、语音信箱、MSRN 路由、订阅者服务
- 对双方进行适当的呼叫状态跟踪和 CDR 生成

网内呼叫获得与外部 MT 呼叫相同的服务待遇
TAS IP 必须在 allowed_sbc_source_ips 配置列表中

变量： ${on_net_status} 在目标号码由您的 TAS 提供服务时设置为 "true"。这在 MO 呼叫授权期间通过检查目标 MSISDN 是否存在于您的订阅者数据库中来确定。

模式 4：非网路由（MO 到 PSTN/外部）

通过网关将 MO 呼叫路由到外部 PSTN、互联或其他外部网络。

使用的网关： sofia/gateway/ExternalSIPGateway 或 sofia/gateway/PSTN_Gateway

模板示例：

```
<extension name="Outgoing-Call-Off-Net">
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="log" data="Sending call off-net" />

    <!-- 在外部路由之前清理头部 -->
    <action application="set" data="sip_copy_multipart=false"/>

    <!-- 设置呼叫事件钩子以进行 CDR/计费 -->
    <action application="set" data='api_body=caller=${msisdn}&called=
    <action application="set" data='api_on_answer=curl http://localhc
${api_body}'/>
    <action application="set" data='api_body=caller=${msisdn}&called=
    <action application="set" data='api_hangup_hook=curl http://local
${api_body}'/>

    <!-- 为受信网络设置 P-Asserted-Identity -->
    <action application="set" data="sip_h_Request-Disposition=no-fork
    <action application="set" data="sip_h_P-Asserted-Identity=<sip:${

    <action application="set" data="hangup_after_bridge=true"/>
    <action application="set" data="continue_on_fail=true"/>

    <!-- 桥接到外部 PSTN/互联网关 -->
    <action application="set" data="used_gateway=ExternalSIPGateway"/
    <action application="bridge"
      data="{absolute_codec_string='AMR-
WB,AMR,PCMA,PCMU',originate_retries=1,originate_timeout=60,sip_invite
/>

    <!-- 如果桥接失败，提供错误处理 -->
    <action application="answer" data="" />
    <action application="log" data="INFO Bridge failed with SIP code
    <action application="sleep" data="500"/>
    <action application="transfer" data="${last_bridge_proto_specific
  </condition>
</extension>
```

关键点：

- 使用 `sofia/gateway/ExternalSIPGateway` 进行外部路由
- 为受信互联设置 `P-Asserted-Identity` 进行来电显示

- 呼叫事件钩子用于 CDR/计费跟踪
- `continue_on_fail=true` 允许错误处理
- 在失败时，根据 SIP 代码转接到错误公告
- `used_gateway` 变量用于报告/故障排除

常见的非网关：

- `ExternalSIPGateway` - 主要 PSTN 中继/互联
- `BackupSIPGateway` - 备用 PSTN 中继（用于故障转移）
- `International_GW` - 国际呼叫网关
- `Emergency_GW` - 紧急服务网关

网关路由策略

所有呼叫都通过网关路由。了解在不同呼叫场景中使用哪个网关对于正确的呼叫路由至关重要：

网内与非网决策树

网关类型和组织

您需要根据网络的互联要求定义自己的网关。网关通常按流量类型、目的地或提供商组织，便实现灵活的路由策略和成本优化。

常见的网关使用模式：

网关类型	用途	示例
PSTN/互联	非网呼叫终止	• PSTN 终止 • 国际运营商 • 其他国内提供商
CS 网络	电路交换网络	• 2G/3G 漫游 (MSRN) • CS 网络集成
紧急服务	紧急呼叫路由	• 911/112/000 呼叫 • PSAP 路由
语音信箱平台	语音信箱服务	• 语音信箱存储 • 消息检索

常见的网关组织模式：

1. 按目的地类型：
    - `sofia/gateway/International_Gateway` - 国际呼叫（最低成本路由）
    - `sofia/gateway/National_Gateway` - 国内/国家呼叫
    - `sofia/gateway/Mobile_Gateway` - 移动到移动互联
    - `sofia/gateway/Emergency_Gateway` - 紧急服务（911/112/000）
2. 按提供商：
    - `sofia/gateway/Provider_A_Primary` - 提供商 A 流量的主要运营商
    - `sofia/gateway/Provider_A_Backup` - 提供商 A 的备用路由
    - `sofia/gateway/Provider_B` - 次要运营商互联
    - `sofia/gateway/Transit_Provider` - 过境/集线提供商
3. 按地理区域：
    - `sofia/gateway/APAC_Gateway` - 亚太地区
    - `sofia/gateway/EMEA_Gateway` - 欧洲/中东/非洲
    - `sofia/gateway/Americas_Gateway` - 北美/南美
4. 按功能：

- `sofia/gateway/Voice_Gateway` - 标准语音流量
- `sofia/gateway/SMS_Gateway` - 基于 SIP 的 SMS（如果支持）
- `sofia/gateway/Wholesale_Gateway` - 批发/运营商流量
- `sofia/gateway/CS_Network_Gateway` - 电路交换（2G/3G）集成

示例：多网关配置

```
<profile name="external">
  <gateways>
    <!-- 国际流量 -->
    <gateway name="International_Primary">
      <param name="proxy" value="10.1.1.100:5060"/>
      <param name="register" value="false"/>
    </gateway>

    <!-- 国内/国家提供商 -->
    <gateway name="Domestic_Provider_A">
      <param name="proxy" value="10.1.2.100:5060"/>
      <param name="register" value="false"/>
    </gateway>

    <gateway name="Domestic_Provider_B">
      <param name="proxy" value="10.1.3.100:5060"/>
      <param name="register" value="false"/>
    </gateway>

    <!-- 紧急服务 -->
    <gateway name="Emergency_PSAP">
      <param name="proxy" value="10.1.4.100:5060"/>
      <param name="register" value="false"/>
    </gateway>

    <!-- CS 网络集成（用于 MSRN 路由） -->
    <gateway name="CS_Network">
      <param name="proxy" value="10.1.5.100:5060"/>
      <param name="register" value="false"/>
    </gateway>
  </gateways>
</profile>
```

路由逻辑示例：

您可根据业务逻辑将呼叫路由到不同的网关：

```
<!-- 国际呼叫（国家代码不是国内的） -->
<extension name="Route-International">
  <condition field="${tas_destination_number}"
expression="^(?!61).*$">
    <action application="bridge"
data="sofia/gateway/International_Primary/${tas_destination_number}"
/>
  </condition>
</extension>

<!-- 根据目的地前缀路由到特定提供商 -->
<extension name="Route-Provider-A">
  <condition field="${tas_destination_number}"
expression="^614\d{8}$">
    <action application="bridge"
data="sofia/gateway/Domestic_Provider_A/${tas_destination_number}"
/>
  </condition>
</extension>

<!-- 带有多个网关的后备路由 -->
<extension name="Route-With-Failover">
  <condition field="${tas_destination_number}" expression="^(.*)$">
    <action application="set" data="continue_on_fail=true"/>
    <action application="bridge"
data="sofia/gateway/Primary_Gateway/$1"/>
    <!-- 如果主要失败，尝试备用 -->
    <action application="bridge"
data="sofia/gateway/Backup_Gateway/$1"/>
  </condition>
</extension>
```

网关组织的最佳实践：

- 使用描述性名称，反映网关的目的
- 规划冗余，使用主要/备用网关
- 按成本组织，启用最低成本路由策略
- 分离关键流量（紧急呼叫使用专用网关）
- 记录每个网关的互联细节（SIP 中继规范、编解码器支持）

- 监控网关健康，并在拨号计划中实现故障转移逻辑

常见路由模式

模式：网内 **MO** 呼叫

```
订阅者 A (MO) → TAS → sofia/internal/B@TAS_IP → TAS (MT 处理) → IMS → 订阅者 B
```

模式：非网内 **MO** 呼叫

```
订阅者 A (MO) → TAS → sofia/gateway/ExternalSIPGateway/+123456789 → PSTN
```

模式：**MT** 呼叫到 **IMS** 订阅者

```
外部 → TAS (MT) → sofia/internal/msisdn@ims.domain → I-CSCF → S-CSCF → 订阅者
```

模式：**MT** 呼叫与 **CFU** 转接到非网内

```
外部 → TAS (MT) → 检测到 CFU → sofia/gateway/ExternalSIPGateway/+forwarding_number → PSTN
```

模式：**MT** 呼叫到 **CS** 漫游订阅者

```
外部 → TAS (MT) → 检索到 MSRN → sofia/gateway/CS_Gateway/+msrn → CS 网络 → 订阅者
```

配置

请参阅 配置□南 获取完整的网关配置详细信息。

网关配置示例：

```
<!-- 外部 PSTN 网关 -->
<gateway name="ExternalSIPGateway">
  <param name="proxy" value="10.1.1.100:5060"/>
  <param name="register" value="false"/>
  <param name="caller-id-in-from" value="true"/>
</gateway>

<!-- CS 网络网关（用于 MSRN 路由） -->
<gateway name="CS_Gateway">
  <param name="proxy" value="10.1.1.200:5060"/>
  <param name="register" value="false"/>
  <param name="caller-id-in-from" value="true"/>
</gateway>

<!-- 紧急服务网关 -->
<gateway name="Emergency_GW">
  <param name="proxy" value="10.1.1.250:5060"/>
  <param name="register" value="false"/>
  <param name="caller-id-in-from" value="true"/>
</gateway>
```

最佳实践

1. 始终设置编解码器 - 使用 `absolute_codec_string` 确保正确的编解码器协商
2. 配置超时 - 适当设置 `progress_timeout` 和 `bridge_answer_timeout`
3. 处理失败 - 使用 `continue_on_fail=true` 和后备操作
4. 清理头部 - 在外部路由之前删除专有头部
5. 明智地使用早期媒体 - `ring_ready` 防止意外公告
6. 防止分叉 - 设置 `Request-Disposition: no-fork` 实现顺序路由
7. 记录桥接结果 - 在桥接前后添加日志便故障排除
8. 测试重试逻辑 - 验证 `originate_retries` 是否按预期工作

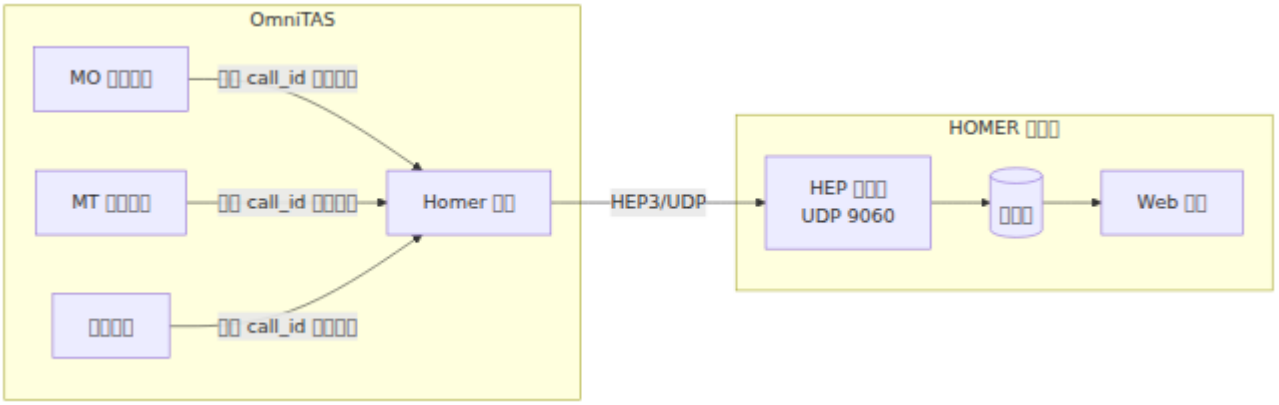
故障排除

启用 **FreeSWITCH** 调试日志：

```
<action application="set" data="sip_trace=on"/>
<action application="info" data=""/>  <!-- 将所有变量转储到日志 -->
```

常见问题：

问题	原因	解决方案
没有音频	编解码器不匹配	检查 `absolute_codec_string`
呼叫立即掉线	网关不可达	验证网关配置
超时太短	`originate_timeout` 太低	增加超时值
不必要的公告	早期媒体通过	使用 `ignore_early_media=ring_ready`
错误的来电显示	`sip_cid_type` 未设置	设置 `sip_cid_type=pid`



HOMER 集成

本文档涵盖了可选的 HOMER 集成，用于将应用程序日志与 SIP 跟踪相关联。

概述

OmniTAS 可□通过 UDP 使用 HEP3（Homer 封装协议）将应用程序日志转发到 HOMER 服务器。使用 SIP Call-ID 将日志与 SIP 跟踪相关联，从而实现呼叫处理与信令的端到端可见性。

工作原理

1. 日志元数据：在呼叫处理期间，OmniTAS 在记录器元数据中设置 SIP Call-ID
2. 日志捕获：Homer 模块通过 PubSub 订阅日��消息
3. 过滤：仅转发包含 `call_id` 元数据的日志
4. **HEP3** 编码：日志被编码为 HEP3 数据包，协议类型为 100（日志）
5. 关联：HOMER 使用 Call-ID 将日志与 SIP 跟踪相关联

配置

将 `homer_server` 配置添加到您的 `config/runtime.exs`：

```
config :tas,
  homer_server: %{
    host: "10.179.2.136",
    port: 9060,
    auth_key: nil,
    capture_id: 2001
  }
```

参数

参数	类型	必需	默认	描述
`host`	字符串	是	-	HOMER 服务器 IP 地址或主机名
`port`	整数	否	9060	HEP 接收器 UDP 端口
`auth_key`	字符串	否	nil	可选的 HEP 认证密钥。如果启用，请设置为与 heplify-server 配置中的 `authkey` 匹配
`capture_id`	整数	否	2001	捕获代理标识符。在 HOMER UI 中显示□识别日志源

禁用 **HOMER** 集成

要禁用 HOMER 集成，可□:

- 完全删除 `homer_server` 配置，或
- 注释掉配置块

禁用后，Homer 模块不会启动，也不会转发日志。

HOMER 服务器要求

HOMER 服务器必须运行 heplify-server，并在 UDP 上启用 HEP 接收。

heplify-server 配置

验证您的 heplify-server.toml 包含:

```
HEPAddr = "0.0.0.0:9060"
```

支持的协议类型

OmniTAS 发送协议类型为 100 的日志，heplify-server 将其路由到 `logs_capture`。确保您的 HOMER 数据库模式支持日志存储。

B2BUA 呼叫腿关联 (X-CID)

OmniTAS 作为 B2BUA，在桥接呼叫时创建新的 SIP 对话（带有新的 Call-ID）。HOMER 需要 `X-CID` 头将所有腿链接到单个呼叫流。

工作原理

1. TAS 在 MO 处理期间将 `original_call_id` 设置为通道变量，包含来自 UE 的原始 A-leg SIP Call-ID
2. MO 拨号计划桥接包括 `sip_h_X-CID=${original_call_id}` 作为内联变量，在 B-leg INVITE 中注入该头
3. MT 拨号计划从传入的 INVITE 中读取 `${sip_h_X-CID}` 并将其传递到其出站桥接
4. 每个 B-leg INVITE 都携带指向原始 UE Call-ID 的 `X-CID`

```
UE INVITE (Call-ID: abc@ue)
  -> S-CSCF -> TAS (MO 处理)
    -> B-leg INVITE (Call-ID: fs-123, X-CID: abc@ue)  <- 内部桥接
      -> TAS (MT 处理)
        -> B-leg INVITE (Call-ID: fs-456, X-CID: abc@ue)  <- MT 桥接
          -> S-CSCF -> 被叫方
```

所有三个 Call-ID（`abc@ue`、`fs-123`、`fs-456`）在 HOMER 中显示为一个呼叫。

拨号计划配置

X-CID 必须设置为 内联桥接变量，而不是通过 `set`，因为 `sip_copy_custom_headers=false` 会删除通过 `set` 设置的 `sip_h_` 变量。

MO 拨号计划（内部桥接）：

```
<action application="bridge" data="{sip_h_X-
CID=${original_call_id},...}sofia/internal/${tas_destination_number}@
/>
```

MO 拨号计划（外部桥接）：

```
<action application="bridge" data="{sip_h_X-
CID=${original_call_id},...}sofia/gateway/trunk/${tas_destination_num
/>
```

MT 拨号计划（所有桥接）**--** 传递从 **MO** 桥接接收到的 **X-CID**:

```
<action application="bridge" data="{sip_h_X-CID=${sip_h_X-
CID},...}sofia/internal/${tas_destination_number}@${domain}" />
```

heplify-server 配置

HOMER heplify-server 必须配置提取和关联 X-CID:

```
AlegIDs                 = ["X-CID"]
CustomHeader            = ["X-CID"]
SIPHeader               =
["callid","callid_aleg","method","ruri_user","from_user","to_user","v
```

- `AlegIDs` 告诉 heplify-server 从 X-CID 头中提取 A-leg Call-ID，并将其存储在 `callid_aleg` 中
- `CustomHeader` 将原始 X-CID 值存储在 data_header JSON 中供搜索
- `SIPHeader` 必须包含 `callid_aleg`，便 HOMER 在呼叫流关联中使用它

更改 heplify-server.toml 后，重启：`systemctl restart heplify-server`

日志关联

自动 **Call-ID** 注入

OmniTAS 在□下情况下自动将 Call-ID 注入到记录器元数据中：

- **MO** 呼叫处理：处理移动发起的呼叫时
- **MT** 呼叫处理：处理移动终止的呼叫时
- 呼叫事件：处理接听和挂断事件时

日志格式

发送到 HOMER 的日志包括：

字段	描述
关联 ID	用于跟踪关联的 SIP Call-ID
时间戳	微秒精度的时间戳
级别	日志级别（调试、信息、警告、错误）
消息	日志消息内容
源 IP	OmniTAS 服务器 IP 地址
捕获 ID	配置的捕获代理标识符

HOMER 中的示例日志

呼叫处理后，HOMER 显示关联数据：

```
[info] 处理来自: 61400123456 到: 61400654321 的 MO 呼叫
[debug] 查找呼叫者: 61400123456
[info] OCS 授权: 授予 (120 秒)
[debug] 设置变量: hangup_case = none
```

这些日志与 SIP INVITE、200 OK 和 BYE 消息一起出现在 HOMER 呼叫详细视图中。

HEP3 协议细节

OmniTAS 实现了 HOMER 项目定义的 HEP3，兼容标准 HEP3 SIP 捕获代理。

使用的块类型

块 ID	名称	描述
0x0001	IP 家族	始终为 2（IPv4）
0x0002	IP 协议	始终为 17（UDP）
0x0003	源 IPv4	OmniTAS 服务器地址
0x0004	目标 IPv4	HOMER 服务器地址
0x0007	源端口	0（不适用于日志）
0x0008	目标端口	配置的 HEP 端口
0x0009	时间戳秒	Unix 时间戳秒
0x000A	时间戳微秒	微秒组件
0x000B	协议类型	100（日志）
0x000C	捕获 ID	配置的 capture_id
0x0011	关联 ID	SIP Call-ID
0x000F	有效载荷	日志消息
0x000E	认证密钥	可选的认证

故障排除

日志未出现在 HOMER 中

症状：呼叫成功完成，但 HOMER 中没有日志出现

可能的原因:

- HOMER 服务器在配置的端口上无法访问
- 防火墙阻止 UDP 流量到端口 9060
- heplify-server 未配置为接收 HEP
- 日志缺少 call_id 元数据

解决方案:

1. 验证网络连接: `nc -zvu <homer_host> 9060`
2. 检查 heplify-server 是否在监听: `ss -ulnp | grep 9060`
3. 查看 heplify-server 日志□获取传入数据包
4. 验证 call_id 是否出现在 OmniTAS 日志中（检查日志输出中的 `call_id=`）

认证失败

症状: heplify-server 日志显示认证错误

可能的原因:

- OmniTAS 和 heplify-server 之间的 auth_key 不匹配
- 一方配置了认证密钥，但另一方没有

解决方案:

1. 如果 heplify-server 没有配置认证密钥，请在 OmniTAS 中��置 `auth_key: nil`
2. 如果 heplify-server 需要认证，请在两个系统中配置匹配的密钥

缺少关联

症状: 日志出现在 HOMER 中，但未与 SIP 跟踪关联

可能的原因:

- SIP 和日志之间的 Call-ID 格式不匹配
- HOMER 关联配置

解决方案:

1. 验证日志中的 Call-ID 是否与 SIP Call-ID 头完全匹配
2. 检查 HOMER 的关联设置□获取 logs_capture 表

□标

启用 HOMER 集成时，请监控□下指标：

OmniTAS 日志

注意指示 HEP 传输失败的警告：

```
[warning] 无法将 HEP 数据包发送到 Homer: <reason>
```

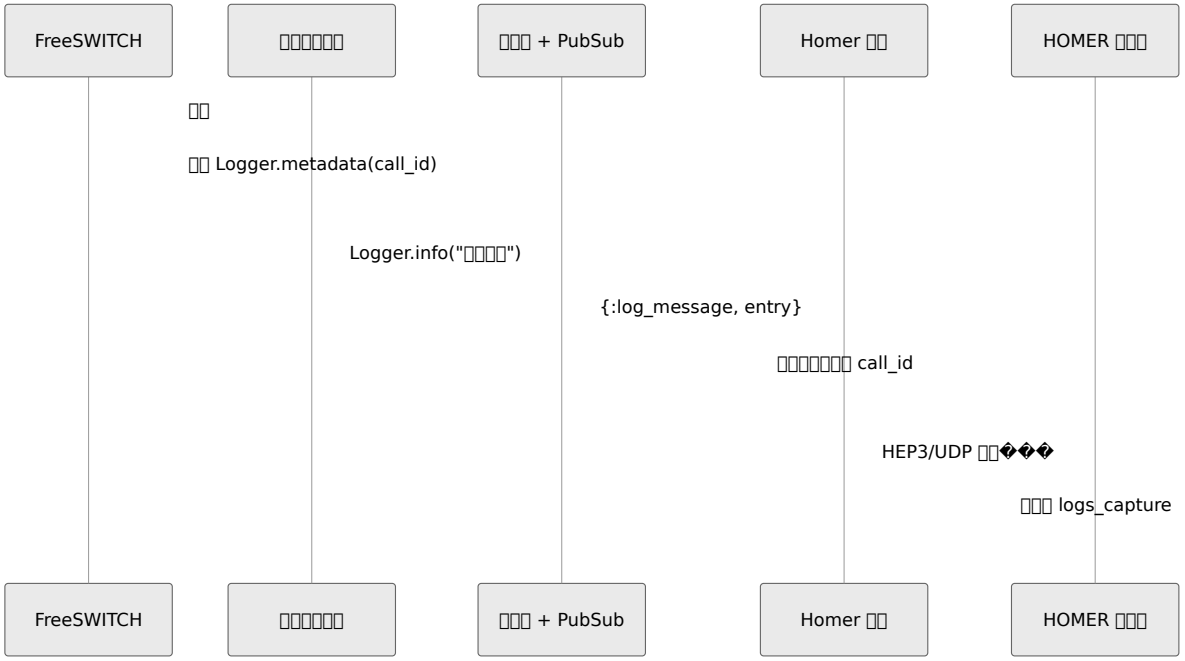
heplify-server □标

如果在 heplify-server 上启用了 Prometheus 指标，请监控：

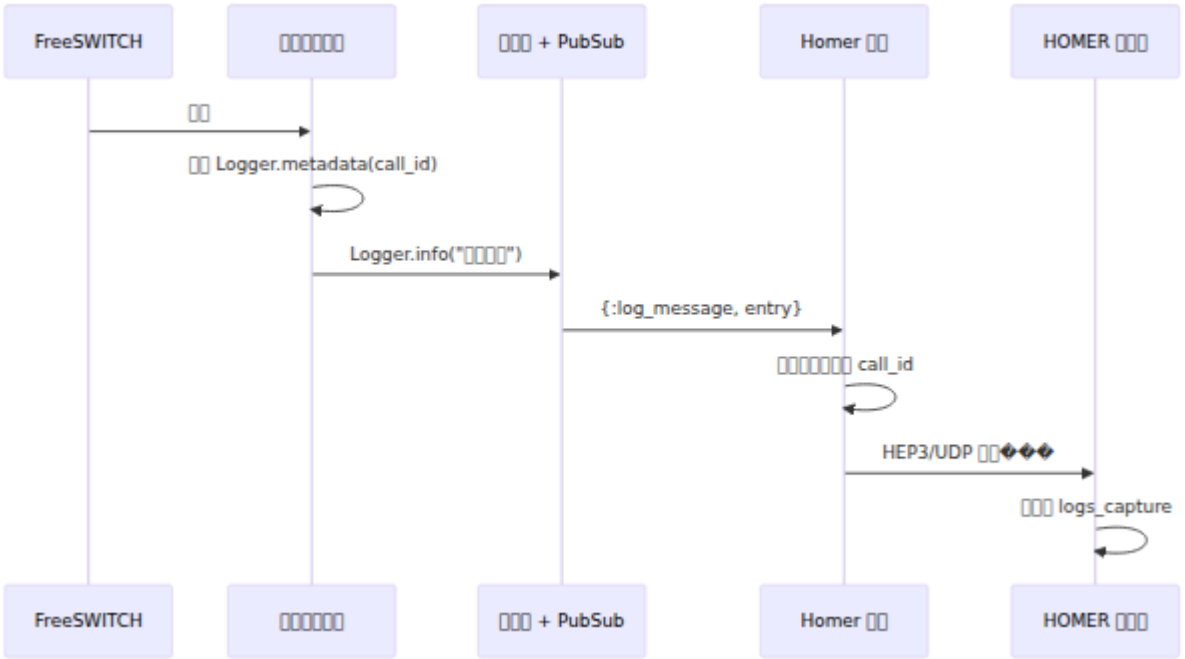
- `heplify_packets_total{type="log"}` - 接收到的日志数据包总数
- `heplify_packets_dropped_total` - 丢弃的数据包（缓冲/解析错误）

□□□□

□□□□


FreeSWITCH
□□□□□□□
□□□ + PubSub
Homer □□
HOMER □□□
□□
□□ Logger.metadata(call_id)
Logger.info("□□□□")
{:log_message, entry}
□□□□□□□ call_id
HEP3/UDP □□□
□□□ logs_capture
FreeSWITCH
□□□□□□□
□□□ + PubSub
Homer □□
HOMER □□□

□□□


FreeSWITCH
□□□□□□□
□□□ + PubSub
Homer □□
HOMER □□□
□□
□□ Logger.metadata(call_id)
Logger.info("□□□□")
{:log_message, entry}
□□□□□□□ call_id
HEP3/UDP □□□
□□□ logs_capture
FreeSWITCH
□□□□□□□
□□□ + PubSub
Homer □□
HOMER □□□

另见

- HOMER 项目 - 开源 VoIP/RTC 捕获和监控
- HEP 协议规范 - HEP/EEP 协议文档
- heplify-server - HEP 捕获服务器

标文档

本文档描述了IMS应用服务器组件暴露的Prometheus指标。

目录

- 指标端点
- 端口9090：系统指标
  - 呼叫和会话指标
  - 系统资源指标
  - 内存指标
  - 编解码器状态指标
  - 端点状态指标
  - 模块状态指标
  - 注册指标
  - Sofia网关指标
  - 出口商健康指标
- 端口9094：TAS引擎指标
  - 应用呼叫指标
  - Diameter协议指标
  - 电话操作指标
  - 在线计费系统（OCS）指标
  - 拨号计划和处理指标
  - 事件套接字指标
  - 功能使用指标
  - 短信触发指标
  - 语音邮件集群指标
  - Erlang Mnesia数��库指标
  - Erlang VM内存指标
  - Erlang VM统计信息
  - Erlang VM系统信息
  - Erlang VM微状态会计（MSACC）

    - Erlang VM分配器
- 端口9093：媒体和呼叫质量指标
    - RTP音频：字节计数器
    - RTP音频：数据包计数器
    - RTP音频：特殊数据包类型
    - RTP音频：抖动和质量指标
    - RTCP指标
- Go运行时指标
- 进程指标
- Prometheus HTTP指标
- 指标类型
- 用法
- 示例查询
- 指标类型
- Grafana仪表板集成
- 警报示例
- 使用指标进行故障排除
- 性能基线
- 最佳实践

□标端点

端口	端点	目的	跳转到部分
9090	`/metrics`	系统、网关和核心电话指标	���□ 9090 →
9094	`/metrics`	TAS引擎、Diameter、HLR、OCS和Erlang VM指标	端口9094 →
9093	`/esl?module=default`	RTP/RTCP媒体质量和呼叫统计	端口9093 →

端口**9090**：系统□标

呼叫和会话□标

□标名称	端口	描述
`freeswitch_bridged_calls`	9090	当前活动的桥接呼叫数量
`freeswitch_detailed_bridged_calls`	9090	当前活动的详细桥接呼叫数量
`freeswitch_current_calls`	9090	当前活动的呼叫数量
`freeswitch_detailed_calls`	9090	当前活动的详细呼叫数量
`freeswitch_current_channels`	9090	当前活动的通道数量
`freeswitch_current_sessions`	9090	当前活动的会话数量
`freeswitch_current_sessions_peak`	9090	自启动□来的会话峰值数量
`freeswitch_current_sessions_peak_last_5min`	9090	最近5分钟的会话峰值数量
`freeswitch_sessions_total`	9090	自启动□来的会话总数（计数器）
`freeswitch_current_sps`	9090	每秒当前会话数量
`freeswitch_current_sps_peak`	9090	自启动□来的每秒峰值会话数量
`freeswitch_current_sps_peak_last_5min`	9090	最近5分钟的每秒峰值会话数量
`freeswitch_max_sessions`	9090	允许的最大会话数量
`freeswitch_max_sps`	9090	允许的最大每秒会话数量

系统资源□标

□标名称	端口	描述
`freeswitch_current_idle_cpu`	9090	当前CPU空闲百分比
`freeswitch_min_idle_cpu`	9090	记录的最低CPU空闲百分比
`freeswitch_uptime_seconds`	9090	运行时间（秒）
`freeswitch_time_synced`	9090	系统时间是否与出口商主机时间同步（1=同步，0=不同步）

内存□标

□标名称	端口	描述
`freeswitch_memory_arena`	9090	总非mmapped字节（malloc arena）
`freeswitch_memory_ordblks`	9090	空闲块数量
`freeswitch_memory_smblks`	9090	空闲快速块数量
`freeswitch_memory_hblks`	9090	映射区域数量
`freeswitch_memory_hblkhd`	9090	映射区域中的字节数
`freeswitch_memory_usmblks`	9090	最大总分配空间
`freeswitch_memory_fsmblks`	9090	在快速块中持有的空闲字节
`freeswitch_memory_uordblks`	9090	总分配空间
`freeswitch_memory_fordblks`	9090	总空闲空间
`freeswitch_memory_keepcost`	9090	最上面的可释放块

编解码器状态□标

□标名称	端口	描述
`freeswitch_codec_status`	9090	编解码器状态，标签: ikey（模块），name（编解码器名称），type（编解码器）。值=1表示编解码器可用

可用编解码器包括:

- 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

端点状态□标

□标名称	端口	描述
`freeswitch_endpoint_status`	9090	端点状态，标签: ikey（模块），name（端点名称），type（端点）。值=1表示端点可用

可用端点包括:

- error, group, pickup, user (mod_dptools)
- loopback, null (mod_loopback)
- rtc (mod_rtc)

- rtp, sofia (mod_sofia)
- modem (mod_spandsp)

模块状态□标

□标名称	端口	描述
`freeswitch_load_module`	9090	模块加载状态（1=已加载，0=未加载），标签: module

监控的关键模块:

- mod_sofia (SIP)
- mod_conference, mod_conference_ims
- mod_opus, mod_g729, mod_amr, 等等
- mod_event_socket
- mod_dptools
- mod_python3
- mod_rtc
- □及更多...

注册□标

□标名称	端口	描述
`freeswitch_registrations`	9090	活动注册的总数
`freeswitch_registration_defails`	9090	详细注册信息，标签: expires, hostname, network_ip, network_port, network_proto, realm, reg_user, token, url

Sofia网关□标

□标名称	端口	描述
`freeswitch_sofia_gateway_status`	9090	网关状态，标签: context, name, profile, proxy, scheme, status (UP/DOWN)
`freeswitch_sofia_gateway_call_in`	9090	通过网关的入站呼叫数量
`freeswitch_sofia_gateway_call_out`	9090	通过网关的出站呼叫数量
`freeswitch_sofia_gateway_failed_call_in`	9090	失败的入站呼叫数量
`freeswitch_sofia_gateway_failed_call_out`	9090	失败的出站呼叫数量
`freeswitch_sofia_gateway_ping`	9090	最后一次ping时间戳（Unix纪元）
`freeswitch_sofia_gateway_pingtime`	9090	最后一次ping时间（毫秒）
`freeswitch_sofia_gateway_pingfreq`	9090	ping频率（秒）
`freeswitch_sofia_gateway_pingcount`	9090	发送的ping数量
`freeswitch_sofia_gateway_pingmin`	9090	记录的最小ping时间
`freeswitch_sofia_gateway_pingmax`	9090	记录的最大ping时间

出口商健康□标

□标名称	端口	描述
`freeswitch_up`	9090	最后一次抓取是否成功（1=成功，0=失败）
`freeswitch_exporter_total_scrapes`	9090	执行的抓取总数（计数器）
`freeswitch_exporter_failed_scrapes`	9090	失败的抓取总数（计数器）

↑ 返回顶部

端口**9094**：**TAS**引擎□标

这些指标由电话应用服务器引擎暴露，提供有关呼叫处理、数据库操作和Erlang VM性能的见解。

> 端口：`9094` 默认。这是OmniTAS HTTP API监听器，同时也提供REST API和FreeSWITCH调用的`fsapi_dialplan`/`fsapi_directory`端点。通过在`runtime.exs`中使用`config :tas, api_port: <port>`进行覆盖（如果更改，则FreeSWITCH的`xml_curl`网关URL和Prometheus抓取目标必须更新□匹配）。

应用呼叫□标

□标名称	端口	描述
`call_simulations_total`	9094	呼叫模拟的总数（计数器）
`call_attempts_total`	9094	呼叫尝试的总数（计数器）
`call_rejections_total`	9094	按原因拒绝的呼叫总数（计数器�□
`call_param_errors_total`	9094	呼叫参数解析错误的总数（计数器）
`active_calls`	9094	当前活动呼叫的数量，标签: call_type (mo/mt/emergency)
`tracked_call_sessions`	9094	当前在ETS中跟踪的呼叫会话数量

Diameter协议□标

□标名称	端口	描述
`diameter_peer_state`	9094	Diameter对等体的状态（1=正常，0=故障），标签: peer_host, peer_realm, application。每10秒轮询每个配置的Ro和Sh对等体。
`diameter_requests_total`	9094	Diameter请求的总数（计数器），标签: application, command
`diameter_responses_total`	9094	Diameter响应的总数（计数器），标签: application, command, result_code
`diameter_response_duration_milliseconds`	9094	Diameter请求的持续时间（毫秒）（直方图），标签: application, command, result

`application` / **`command`**标签值:

application	command	接口
`ro`	`ccr_i`	Credit-Control-Request, INITIAL (呼叫设置)
`ro`	`ccr_u`	Credit-Control-Request, UPDATE (临时重新授权)
`ro`	`ccr_t`	Credit-Control-Request, TERMINATION (呼叫拆除)
`sh`	`udr`	User-Data-Request

> 注意： Ro计费控制指标按请求类型（`ccr_i`/`ccr_u`/`ccr_t`）拆分，因此例如，失败的临时更新的峰值可□独立可见。Sh `udr`路径记录请求计数和延迟（`diameter_requests_total` / `diameter_response_duration_milliseconds`），□及响应代码，与Ro路径匹配。`result_code`携带Diameter结果代码（例如`2001`）；`0`表示超时或无法解析的回复。持续时间直方图上的`result`标签是`success`、`nocredit`、`error`，加上`unknown_user` / `no_reply`用于Sh。

电话操作□标

□标名称	端口	描述
`hlr_lookups_total`	9094	HLR查找的总数（计数器）
`hlr_data_duration_milliseconds`	9094	HLR数据检索的持续时间（毫秒）（直方图）
`subscriber_data_lookups_total`	9094	订阅者数据查找的总数（计数器）
`subscriber_data_duration_milliseconds`	9094	Sh订阅者数据检索的持续时间（毫秒）（直方图）
`ss7_map_operations_total`	9094	SS7 MAP操作的总数（计数器）
`ss7_map_http_duration_milliseconds`	9094	SS7 MAP HTTP请求的持续时间（毫秒）（直方图）
`tracked_registrations`	9094	当前跟踪的SIP注册数量

在线计费系统（**OCS**）□标

□标名称	端口	描述
`ocs_authorization_attempts_total`	9094	OCS授权尝试的总数（计数器）
`ocs_authorization_duration_milliseconds`	9094	OCS授权的持续时间（毫秒）（直方图
`online_charging_events_total`	9094	在线计费事件的总数（计数器），标签：event_type, result
`authorization_decisions_total`	9094	授权决策的总数（计数器）
`ro_charging_quota_seconds`	9094	Ro计费控制配额（秒）（直方图），标[request_type (`ccr_i`/`ccr_u`/`ccr_t`) (`requested`/`granted`/

`online_charging_events_total`标签:

- **event_type:** `authorize`, `answer`, `reauth`, `hangup`, `credit_exhaustion_hangup`, `hangup_rescheduled`
- **result:** `success`, `nocredit`, `timeout`, `error`, `triggered`

`ro_charging_quota_seconds`: 每个**CCR**观察到的配额:

- `kind="requested"`: 在CCR-I/CCR-U上请求的保留（`Requested-Service-Unit` CC-Time）
- `kind="granted"`: OCS在CCA中批准的配额（`Granted-Service-Unit` CC-Time; `0` = 无信用）
- `kind="used"`: 在CCR-T上报告为消耗的单位（`Used-Service-Unit` CC-Time）

比较`granted`与`used`暴露了过度或不足的保留，并且是Ro接口的主要计费正确性信号。

拨号计划和处理□标

□标名称	端口	描述
`http_requests_total`	9094	HTTP请求的总数，标签: endpoint, status_code（计数器）
`http_dialplan_request_duration_milliseconds`	9094	HTTP拨号计划请求的持续时间（毫秒）（直方图）
`dialplan_module_duration_milliseconds`	9094	单个拨号计划模块处理的持续时间（毫秒）（直方图）
`freeswitch_variable_set_duration_milliseconds`	9094	变量设置操作的持续时间（毫秒）（直方图）

事件套接字□标

□标名称	端口	描述
`event_socket_connected`	9094	事件套接字连接状态（1=已连接，0=未连接），标签: connection_type
`event_socket_reconnections_total`	9094	事件套接字重新连接尝试的总数（计数器），标签: connection_type, result
`event_socket_commands_total`	9094	执行的事件套接字命令总数（计数器），标签: command_type, result
`event_socket_command_timeouts_total`	9094	事件套接字命令超时的总数（计数器），标签: command_type

跟踪的命令类型:

- uuid_setvar, uuid_dump, uuid_kill, uuid_transfer
- uuid_set_media_stats
- sched_hangup, sched_transfer
- vm_boxcount
- status, echo, show, sofia

结果值:

- success: 命令成功完成
- timeout: 命令��过超时阈值
- error: 命令返回意外响应

功能使用□标

□标名称	端口	描述
`feature_invocations_total`	9094	TAS功能调用的总数（计数器），标签: feature, call_type, result
`feature_data_source_total`	9094	功能数据源使用的总数（计数器），标签: feature, source

功能:

- `call_forward_all` - 无条件呼叫转移
- `call_forward_not_reachable` - 当订阅者不可达时呼叫转移
- `call_forward_no_reply` - 无回复时呼叫转移
- `call_barring` - 基于OCS的呼叫限制（信用不足）
- `cli_withheld` - CLI隐私/筛选

呼叫类型: `mo`, `mt`

数据源: `sh_interface`, `hlr`, `config_fallback`

结果值: `success`, `error`, `skipped`

短信触发□标

□标名称	端口	描述
`sms_trigger_attempts_total`	9094	短信触发尝试的总数（计数器），标签: trigger_type, result
`sms_trigger_errors_total`	9094	短信触发错误的总数（计数器），标签: trigger_type, error_stage
`smsc_requests_total`	9094	SMSC HTTP请求的总数（计数器），标签: message_type, result

触发类型: `voicemail_deposit`, `voicemail_clear`

错误阶段: `vm_boxcount`, `template_render`, `smsc_request`

消息类型: `notification`, `mwi`

结果值: `success`, `error`

语音邮件集群□标

仅在配置了多TAS语音邮件集群时发出（`config :tas, voicemail: %{cluster: ...}`）。在集群中，邮箱请求会分发到每个对等TAS；该计数器记录超时或错误的对等体，□便可□发出警报，而不是默默容忍缺失条件。

□标名称	端口	描述
`voicemail_cluster_fanout_errors_total`	9094	语音邮件集群对等体分发失败（计数器），标签: operation

操作: `count`, `messages`, `media`, `greeting`, `mark_read`, `delete`, `clear_greeting`, `clear_mailbox`

用法:

```
# 按操作的集群分发错误率
sum by (operation)
(rate(voicemail_cluster_fanout_errors_total[5m]))
```

警报时:

- 持续错误 > 0: 对等TAS不可达，因此跨节点邮箱视图可能不完整

Erlang Mnesia数据库□标

□标名称	端口	描述
erlang_mnesia_held_locks	9094	持有的锁数量
erlang_mnesia_lock_queue	9094	等待锁的事务数量
erlang_mnesia_transaction_participants	9094	参与事务的数量
erlang_mnesia_transaction_coordinators	9094	协调事��的数量
erlang_mnesia_failed_transactions	9094	失败（中止）事务的数量（计数器）
erlang_mnesia_committed_transactions	9094	提交的事务数量（计数器）
erlang_mnesia_logged_transactions	9094	记录的事务数量（计数器）
erlang_mnesia_restarted_transactions	9094	事务重启的总数（计数器）
erlang_mnesia_memory_usage_bytes	9094	所有mnesia表分配的总字节数
erlang_mnesia_tablewise_memory_usage_bytes	9094	每个mnesia表分配的字节数，标签: table
erlang_mnesia_tablewise_size	9094	每个表的行数，标签: table

Erlang VM内存□标

□标名称	端口	描述
`erlang_vm_memory_atom_bytes_total`	9094	分配给原子的内存，标签: usage (used/free)
`erlang_vm_memory_bytes_total`	9094	分配的总内存，标签: kind (system/processes)
`erlang_vm_memory_dets_tables`	9094	DETS表的数量
`erlang_vm_memory_ets_tables`	9094	ETS表的数量
`erlang_vm_memory_processes_bytes_total`	9094	分配给进程的内存，标签: usage (used/free)
`erlang_vm_memory_system_bytes_total`	9094	模拟器的内存（与进程无关），标签: usa (atom/binary/code/ets/c

Erlang VM统计信息

��标名称	端口	描述
`erlang_vm_statistics_bytes_output_total`	9094	输出到端口的总字节数（计数器）
`erlang_vm_statistics_bytes_received_total`	9094	通过端口接收的总字节数（计数器）
`erlang_vm_statistics_context_switches`	9094	自启动□来的上下文切换总数（计数器）
`erlang_vm_statistics_dirty_cpu_run_queue_length`	9094	脏CPU运行队列的长度
`erlang_vm_statistics_dirty_io_run_queue_length`	9094	脏IO运行队列的长度
`erlang_vm_statistics_garbage_collection_number_of_gcs`	9094	垃圾收集的数

×××××	××	××
		××× ×××
`erlang_vm_statistics_garbage_collection_bytes_reclaimed`	9094	××× ××× ××× ××× ×××
`erlang_vm_statistics_garbage_collection_words_reclaimed`	9094	××× ××× ××× ××× ××
`erlang_vm_statistics_reductions_total`	9094	××× ××× ×××
`erlang_vm_statistics_run_queues_length`	9094	××× ××× ×××
`erlang_vm_statistics_runtime_milliseconds`	9094	××× ××× ××× ×× ××× ××
`erlang_vm_statistics_wallclock_time_milliseconds`	9094	××× ××× ××× ×××

Erlang VM系统信息

□标名称	端口	描述
erlang_vm_dirty_cpu_schedulers	9094	脏CPU调度线程的数量
erlang_vm_dirty_cpu_schedulers_online	9094	在线的脏CPU调度器数量
erlang_vm_dirty_io_schedulers	9094	脏I/O调度线程的数量
erlang_vm_ets_limit	9094	允许的最大ETS表数量
erlang_vm_logical_processors	9094	配置的逻辑处理器数量
erlang_vm_logical_processors_available	9094	可用的逻辑处理器数量
erlang_vm_logical_processors_online	9094	在线的逻辑处理器数量
erlang_vm_port_count	9094	当前存在的端口数量
erlang_vm_port_limit	9094	允许的最大端口数量
erlang_vm_process_count	9094	当前存在的进程数量
erlang_vm_process_limit	9094	允许的最大进程数量
erlang_vm_schedulers	9094	调度线程的数量
erlang_vm_schedulers_online	9094	在线的调度器数量
erlang_vm_smp_support	9094	如果编译时支持SMP则为1，否则为0
erlang_vm_threads	9094	如果编译时支持线程则为1，否则为0
erlang_vm_thread_pool_size	9094	池中异步线程的数量

□标名称	端口	描述
`erlang_vm_time_correction`	9094	如果启用时间校正则为1，否则为0
`erlang_vm_wordsize_bytes`	9094	Erlang术语字的字节大小
`erlang_vm_atom_count`	9094	当前存在的原子数量
`erlang_vm_atom_limit`	9094	允许的最大原子数量

Erlang VM微状态会计（MSACC）

详细的调度器活动时间跟踪，标签: type, id

□标名称	端口	描述
`erlang_vm_msacc_aux_seconds_total`	9094	处理辅助作业的时间（计数器）
`erlang_vm_msacc_check_io_seconds_total`	9094	检查新I/O事件的时间（计数器）
`erlang_vm_msacc_emulator_seconds_total`	9094	执行Erlang进程的时间（计数器）
`erlang_vm_msacc_gc_seconds_total`	9094	垃圾收集的时间（计数器）
`erlang_vm_msacc_other_seconds_total`	9094	在未记录活动上花费的时间（计数器）
`erlang_vm_msacc_port_seconds_total`	9094	执行端口的时间（计数器）
`erlang_vm_msacc_sleep_seconds_total`	9094	睡眠时间（计数器）
`erlang_vm_msacc_alloc_seconds_total`	9094	管理内存的时间（计数器）
`erlang_vm_msacc_bif_seconds_total`	9094	在BIFs中花费的时间（计数器）
`erlang_vm_msacc_busy_wait_seconds_total`	9094	忙等待的时间（计数器）
`erlang_vm_msacc_ets_seconds_total`	9094	在ETS BIFs中花费的时间（计数器）
`erlang_vm_msacc_gc_full_seconds_total`	9094	在全扫GC中花费的时间（计数器）
`erlang_vm_msacc_nif_seconds_total`	9094	在NIFs中花费的时间（计数器）
`erlang_vm_msacc_send_seconds_total`	9094	发送消息的时间（计数器）
`erlang_vm_msacc_timers_seconds_total`	9094	管理定时器的时间（计数器）

Erlang VM分配器

详细的内存分配器指标，标签: alloc, instance_no, kind, usage

□标名称	端口	描述
`erlang_vm_allocators`	9094	不同分配器的分配（carriers_size）和使用（blocks_size）内存。请参见erts_alloc(3)。

分配器类型包括： temp_alloc, sl_alloc, std_alloc, ll_alloc, eheap_alloc, ets_alloc, fix_alloc, literal_alloc, binary_alloc, driver_alloc

---

↑ 返回顶部

端口**9093**：媒体和呼叫质量□标

这些指标提供每个通道的实时RTP/RTCP统计和呼叫质量信息。

□标名称	端口	描述
`freeswitch_info`	9093	系统信息，标签：version
`freeswitch_up`	9093	就绪状态（1=就绪，0=未就绪）
`freeswitch_stack_bytes`	9093	堆栈大小（字节）
`freeswitch_session_total`	9093	会话总数
`freeswitch_session_active`	9093	活动会话数量
`freeswitch_session_limit`	9093	会话限制
`rtp_channel_info`	9093	RTP通道信息，带有通道详细信息的标签

RTP音频：字节计数器

□标名称	端口	描述
`rtp_audio_in_raw_bytes_total`	9093	接收的总字节数（包括头部）
`rtp_audio_out_raw_bytes_total`	9093	发送的总字节数（包括头部）
`rtp_audio_in_media_bytes_total`	9093	接收的总媒体字节数（仅有效载荷）
`rtp_audio_out_media_bytes_total`	9093	发送的总媒体字节数（仅有效载荷）

RTP音频：数据包计数器

□标名称	端口	描述
`rtp_audio_in_packets_total`	9093	接收的总数据包数
`rtp_audio_out_packets_total`	9093	发送的总数据包数
`rtp_audio_in_media_packets_total`	9093	接收的总媒体数据包数
`rtp_audio_out_media_packets_total`	9093	发送的总媒体数据包数
`rtp_audio_in_skip_packets_total`	9093	丢弃的入站数据包
`rtp_audio_out_skip_packets_total`	9093	丢弃的出站数据包

RTP音频：特殊数据包类型

□标名称	端口	描述
`rtp_audio_in_jitter_packets_total`	9093	接收的抖动缓冲区数据包
`rtp_audio_in_dtmf_packets_total`	9093	接收的DTMF数据包
`rtp_audio_out_dtmf_packets_total`	9093	发送的DTMF数据包
`rtp_audio_in_cng_packets_total`	9093	接收的舒适噪声生成数据包
`rtp_audio_out_cng_packets_total`	9093	发送的舒适噪声生成数据包
`rtp_audio_in_flush_packets_total`	9093	刷新的数据包（缓冲区重置）

RTP音频：抖动和质量□标

□标名称	端口	描述
`rtp_audio_in_jitter_buffer_bytes_max`	9093	最大抖动缓冲区大小（字节）
`rtp_audio_in_jitter_seconds_min`	9093	最小抖动（秒）
`rtp_audio_in_jitter_seconds_max`	9093	最大抖动（秒）
`rtp_audio_in_jitter_loss_rate`	9093	由于抖动造成的数据包丢失率（比率）
`rtp_audio_in_jitter_burst_rate`	9093	由于抖动造成的数据包突发率（比率）
`rtp_audio_in_mean_interval_seconds`	9093	入站数据包之间的平均间隔
`rtp_audio_in_flaw_total`	9093	检测到的音频缺陷总数（故障、伪影）
`rtp_audio_in_quality_percent`	9093	音频质量的百分比（0-100）
`rtp_audio_in_quality_mos`	9093	平均意见分数（1-5，5为最佳）

RTCP□标

□标名称	端口	描述
`rtcp_audio_bytes_total`	9093	总RTCP字节数
`rtcp_audio_packets_total`	9093	总RTCP数据包数

Go运行时□标

□标名称	端口	描述
`go_goroutines`	9090	当前运行的goroutine数量
`go_threads`	9090	创建的操作系统线程数量
`go_info`	9090	关于Go环境的信息（带版本标签）
`go_gc_duration_seconds`	9090	垃圾收集周期的暂停持续时间（摘要）
`go_memstats_alloc_bytes`	9090	分配并仍在使用的字节数
`go_memstats_alloc_bytes_total`	9090	分配的总字节数（计数器）
`go_memstats_heap_alloc_bytes`	9090	分配并仍在使用的堆字节数
`go_memstats_heap_idle_bytes`	9090	等待使用的堆字节数
`go_memstats_heap_inuse_bytes`	9090	当前使用的堆字节数
`go_memstats_heap_objects`	9090	分配的堆对象数量
`go_memstats_heap_released_bytes`	9090	释放给操作系统的堆字节数
`go_memstats_heap_sys_bytes`	9090	从系统获取的堆字节数
`go_memstats_sys_bytes`	9090	从系统获取的总字节数

进程□标

□标名称	端口	描述
`process_cpu_seconds_total`	9090	花费的总用户和系统CPU时间（计数器）
`process_max_fds`	9090	最大打开文件描述符数量
`process_open_fds`	9090	当前打开的文件描述符数量
`process_resident_memory_bytes`	9090	常驻内存大小（字节）
`process_virtual_memory_bytes`	9090	虚拟内存大小（字节）
`process_virtual_memory_max_bytes`	9090	可用的最大虚拟内存量
`process_start_time_seconds`	9090	自Unix纪元□来的进程启动时间

Prometheus HTTP□标

□标名称	端口	描述
`promhttp_metric_handler_requests_in_flight`	9090	当前正在服务的抓取数量
`promhttp_metric_handler_requests_total`	9090	按HTTP状态代码抓取的总数（计数器）

↑ 返回顶部

□标类型

- **gauge**: 可□上升或下降的指标（例如，current_calls, cpu_idle）
- **counter**: 仅增加的指标（例如，sessions_total, failed_scrapes）

- **summary**：在滑动时间窗口中跟踪分位数的指标（例如，gc_duration_seconds）

↑ 返回顶部

用法

要抓取这些指标，请配置您的Prometheus服务器□抓取所有三个端点：

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

  - job_name: 'ims_as_engine'
    static_configs:
      - targets: ['localhost:9094']
    metrics_path: '/metrics'

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

↑ 返回顶部

示例查询

快速链接：

- 常规指标（端口9090）
- 媒体质量指标（端口9093）
- TAS引擎指标（端口9094）

常规□标

当前呼叫量：

```
freeswitch_current_calls
```

网关健康：

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

到网关的平均**ping**时间：

```
avg(freeswitch_sofia_gateway_pingtime)
```

每秒会话率：

```
freeswitch_current_sps
```

内存使用情况：

```
freeswitch_memory_uordblks
```

媒体质量□标

呼叫质量（**MOS**分数）：

```
rtp_audio_in_quality_mos
```

音频质量百分比：

```
rtp_audio_in_quality_percent
```

抖动率：

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

数据包丢失率：

```
rtp_audio_in_jitter_loss_rate
```

平均抖动：

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

RTP带宽（入站）：

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

检测到的音频缺陷：

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

TAS引擎指标

按类型的活动呼叫：

```
active_calls
```

Diameter对等体健康：

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

呼叫尝试率：

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

HLR查找延迟（第**95**百分位）：

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

OCS授权延迟：

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

订阅者数据查找率：

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

Diameter请求成功率：

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

按请求类型的**CCR**率（发现临时更新的峰值）：

```
sum by (command) (rate(diameter_requests_total{application="ro"}
[5m]))
```

每种请求类型的**CCR**错误率：

```
sum by (command) (rate(diameter_responses_total{application="ro",
result_code!="2001"}[5m]))
```

Sh (UDR) P95延迟：

```
histogram_quantile(0.95,
rate(diameter_response_duration_milliseconds_bucket{application="sh"}
[5m]))
```

呼��设置时授予的计费控制配额的中位数：

```
histogram_quantile(0.5,
rate(ro_charging_quota_seconds_bucket{request_type="ccr_i",
kind="granted"}[5m]))
```

每秒的无信用授予（零秒分配）：

```
rate(ro_charging_quota_seconds_bucket{kind="granted", le="0"}[5m])
```

事件套接字连接状态:

```
event_socket_connected
```

Mnesia事务性能:

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

Mnesia失败事务率:

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

Erlang VM进程数量:

```
erlang_vm_process_count
```

Erlang VM内存使用情况:

```
erlang_vm_memory_bytes_total
```

垃圾收集率:

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

调度器运行队列长度：

```
erlang_vm_statistics_run_queues_length
```

ETS表数量：

```
erlang_vm_memory_ets_tables
```

HTTP拨号计划请求持续时间（中位数）：

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

↑ 返回顶部

Grafana仪表板集成

这些指标可通过Prometheus数据源在Grafana中可视化。

推荐的仪表板布局

第**1**行：呼叫量与健康

- 活动呼叫仪表（`active_calls`）
- 按类型的呼叫尝试率（`rate(call_attempts_total[5m])`）
- 呼叫拒绝率（`rate(call_rejections_total[5m])`）
- 网关健康（`freeswitch_sofia_gateway_status`）

第**2**行：性能（延迟百分位）

- 按呼叫类型的P95 HTTP拨号计划请求时间
- P95 Sh订阅者数据查找时间
- P95 HLR查找时间

P95 OCS授权时间
按应用程序的P95 Diameter响应时间

第3行：成功率

订阅者数据查找成功率
HLR查找成功率
OCS授权成功率
Diameter对等体状态

第4行：媒体质量

呼叫质量MOS分数（rtp_audio_in_quality_mos）音频质量百分比（rtp_audio_in_quality_percent）
抖动统计
数据包丢失率

第5行：系统资源

Erlang VM进程数量
Erlang VM内存使用情况
ETS表数量
调度器运行队列长度
垃圾收集率

第6行：错误跟踪

呼叫参数错误
授权失败
事件套接字连接状态
Mnesia事务失败

示例面板查询

按类型的活动呼叫：

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

P95拨号计划生成延迟:

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

Diameter成功率:

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

媒体质量 - 平均**MOS**:

```
avg(rtp_audio_in_quality_mos)
```

↑ 返回顶部

警报示例

关键警报（立即分页）

系统故障 - 无呼叫尝试:

```
alert: SystemDown
expr: rate(call_attempts_total[5m]) == 0
for: 2m
labels:
  severity: critical
annotations:
  summary: "TAS系统似乎已关闭 - 无呼叫尝试"
  description: "2分钟内未检测到呼叫尝试"
```

Diameter对等体故障:

```
alert: DiameterPeerDown
expr: diameter_peer_state == 0
for: 1m
labels:
  severity: critical
annotations:
  summary: "Diameter对等体{{ $labels.peer_host }}已关闭"
  description: "{{ $labels.application }}应用程序的对等体不可用"
```

事件套接字断开连接:

```
alert: EventSocketDisconnected
expr: event_socket_connected == 0
for: 30s
labels:
  severity: critical
annotations:
  summary: "事件套接字{{ $labels.connection_type }}断开连接"
  description: "关键通信通道故障"
```

高严重性警报

高**Diameter**延迟:

```
alert: HighDiameterLatency
expr: |
  histogram_quantile(0.95,
    rate(diameter_response_duration_milliseconds_bucket[5m])
  ) > 1000
for: 5m
labels:
  severity: high
annotations:
  summary: "检测到高Diameter延迟"
  description: "P95延迟为{{ $value }}ms"
```

OCS授权失败:

```
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: "OCS无信用响应的高比例"
  description: "{{ $value | humanizePercentage }}的请求被拒绝信用"
```

高呼叫拒绝率:

```
alert: HighCallRejectionRate
expr: |
  rate(call_rejections_total[5m]) /
  rate(call_attempts_total[5m]) > 0.05
for: 5m
labels:
  severity: high
annotations:
  summary: "呼叫拒绝率超过5%"
  description: "{{ $value | humanizePercentage }}的呼叫被拒绝"
```

差的媒体质量:

```
alert: PoorMediaQuality
expr: avg(rtp_audio_in_quality_mos) < 3.5
for: 3m
labels:
  severity: high
annotations:
  summary: "检测到差的呼叫质量"
  description: "□均MOS分数为{{ $value }}"
```

警告警报

高内存使用:

```
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内存使用高"
  description: "进程内存为{{ $value | humanizePercentage }}"
```

高调度器运行队列:

```
alert: HighSchedulerRunQueue
expr: erlang_vm_statistics_run_queues_length > 10
for: 5m
labels:
  severity: warning
annotations:
  summary: "高调度器运行队列长度"
  description: "运行队列长度为{{ $value }}"
```

Mnesia事务失败:

```
alert: MnesiaTransactionFailures
expr: rate(erlang_mnesia_failed_transactions[5m]) > 1
for: 5m
labels:
  severity: warning
annotations:
  summary: "检测到Mnesia事务失败"
  description: "{{ $value }}每秒的失败"
```

↑ 返回顶部

使用□标进行故障排除

问题：呼叫缓慢

调查步骤：

1. 检查整体拨号计划生成时间：

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

2. 按组件细分：

```
# 订阅者数据查找
histogram_quantile(0.95,
rate(subscriber_data_duration_milliseconds_bucket[5m]))

# HLR查找
histogram_quantile(0.95,
rate(hlr_data_duration_milliseconds_bucket[5m]))

# OCS授权
histogram_quantile(0.95,
rate(ocs_authorization_duration_milliseconds_bucket[5m]))
```

3. 检查模块特定延迟：

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

常见原因：

- 外部系统延迟（HSS、HLR、OCS）
- 网络问题
- 数据库争用
- 高系统负载

问题：呼叫失败

调查步骤：

1. 检查呼叫拒绝原因：

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

2. 检查授权决策：

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

3. 检查**Diameter**对等体健康：

```
diameter_peer_state
```

4. 检查事件套接字连接：

```
event_socket_connected
```

问题：高负载

调查步骤：

1. 检查呼叫量：

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

2. 检查**Erlang VM**资源：

```
erlang_vm_process_count
erlang_vm_statistics_run_queues_length
erlang_vm_memory_bytes_total
```

3. 检查垃圾收集:

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

问题：媒体质量差

调查步骤:

1. 检查**MOS**分数:

```
rtp_audio_in_quality_mos
rtp_audio_in_quality_percent
```

2. 检查抖动:

```
rtp_audio_in_jitter_seconds_max
rtp_audio_in_jitter_loss_rate
```

3. 检查数据包丢失:

```
rtp_audio_in_skip_packets_total
rtp_audio_in_flaw_total
```

4. 检查带宽使用:

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

↑ 返回顶部

性能基线

典型值（调优良好的系统）

延迟（P95）：

HTTP拨号计划请求：200-500ms
订阅者数据（Sh）查找：50-150ms
HLR数据查找：100-300ms
OCS授权：100-250ms
Diameter请求：50-200ms
拨号计划模块处理：每个模块10-50ms

成功率：

呼叫完成：>95%
订阅者数据查找：>99%
HLR查找：>98%
OCS授权：>99%（不包括合法的无信用） Diameter对等体正常运行时间：>99.9%

媒体质量：

MOS分数：>4.0
音频质量百分比：>80%
抖动：<30ms
数据包丢失率：<1%

系统资源：

Erlang进程数量：<50% 的限制
Erlang内存使用：<70% 的可用
调度器运行队列：<5
ETS表：<1000

容量规划

每服务器容量（推荐最大值）：

- 并发呼叫：500-1000（取决于硬件）
- 每秒呼叫：20-50 CPS
- 注册订阅者：10,000-50,000

扩展□标（增加容量时）：

- 活动呼叫持续>70% 的容量
- Erlang进程数量>70% 的限制
- P95延迟恶化
- 调度器运行队列持续>10

↑ 返回顶部

最佳实践

监控策略

1. 为不同受众设置仪表板：
    - 操作仪表板：呼叫量、成功率、系统健康
    - 工程仪表板：延迟百分位、错误率、资源使用
    - 高管仪表板：高层KPI、正常运行时间、成本指标
2. 在多个级别配置警报：
    - 关键：分页值班（系统故障、大规模故障）
    - 高：在工作时间内警报（性能下降）
    - 警告：在工单系统中跟踪（潜在问题）
3. 使用适当的时间范围：
    - 实时监控：5分钟窗口

- 故障排除：15分钟到1小时窗口
- 容量规划：每日/每周汇总

4. 关注用户影响：

- 优先考虑端到端延迟指标
- 跟踪成功率而不是单个错误计数器
- 监控媒体质量□提高用户体验

查询性能

1. 为频繁使用的查询使用记录规则：

```
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. 在查询中避免高基数标签（例如，不按电话号码分组）

3. 使用适当的速率间隔：

- 短期趋势：`[5m]`
- 中期趋势：`[1h]`
- 长期趋势：`[1d]`

□标基数

监控基数□防止Prometheus性能问题：

```
# 检查指标基数
count by (__name__) ({__name__=~".+"})
```

高基数风险：

每个呼叫都有唯一值的标签（电话号码、呼叫ID）
无界标签值
超过1000个唯一值的标签

解决方案：

使用标签进行分类，而不是唯一标识符
在外部系统中聚合高基数数据
使用记录规则进行预聚合

↑ 返回顶部

数字翻译

 返回主文档

数字翻译在不同格式之间转换电话号码，确保系统中一致的 E.164 格式。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - 数字翻译配置 (`number_translate`)
- 操作南 - 控制面板中的数字翻译测试

呼叫处理流程

- 拨号计划配置 - 在拨号计划中使用翻译后的号码（翻译首先发生）
- **Sh** 接口 - Sh 查找使用翻译后的号码
- 在线计费 - OCS 接收翻译后的号码
- **SS7 MAP** - HLR 查询使用翻译后的号码

相关服务

- ⚙ 补充服务 - 在翻译过程中去除 CLI 阻止前缀
- 语音邮件 - 翻译中的语音邮件号码

监控

- 标参考 - 数字翻译指标

□□□□

□□□

配置

```
config :tas,
  number_translate: %{
    country_code: :PF,         # ISO 3166-1 alpha-2 国家代码
    localAreaCode: "617"       # 短号码的默认区号
  }
```

参数:

- **`country_code`:** 作为原子（atom）的 ISO 国家代码（例如，`:AU`, `:US`, `:PF`）
- **`localAreaCode`:** 添加到短本地号码的区号

支持的国家代码

TAS 包括 **248** 个国家和地区 的翻译逻辑，涵盖所有具有电话服务的 ISO 3166-1 alpha-2 代码。

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:AC`	阿森松岛	`247`	5	无	00	
`:AD`	安道尔	`376`	6-9	无	00	
`:AE`	阿联酋	`971`	9	0	00	
`:AF`	阿富汗	`93`	9	0	00	
`:AG`	安提瓜和巴布达	`1`	10	无	011	NANP，委托给美国
`:AI`	安圭拉	`1`	10	无	011	NANP，委托给美国
`:AL`	阿尔巴尼亚	`355`	9	0	00	
`:AM`	亚美尼亚	`374`	8	0	00	
`:AO`	安哥拉	`244`	9	无	00	
`:AR`	阿根廷	`54`	10	0	00	
`:AS`	美属萨摩亚	`1`	10	无	011	NANP，委托给美国
`:AT`	奥地利	`43`	10-13	0	00	
`:AU`	澳大利亚	`61`	9	0	0011	13 xx xx（6+ 位数字）和 1300/1800 xxx xxx（10+ 位数字）前面加 61，且截断为 6/10 位数字；190x 优质号码不加前缀
`:AW`	阿鲁巴	`297`	7	无	00	

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:AX`	奥兰群岛	`358`	9-10	0	00	委托给芬兰
`:AZ`	阿塞拜疆	`994`	9	0	00	
`:BA`	波斯尼亚和黑塞哥维那	`387`	8	0	00	
`:BB`	巴巴多斯	`1`	10	无	011	NANP，委托给美国
`:BD`	孟加拉国	`880`	10	0	00	
`:BE`	比利时	`32`	8-9	0	00	
`:BF`	布基纳法索	`226`	8	无	00	
`:BG`	保加利亚	`359`	9	0	00	
`:BH`	巴林	`973`	8	无	00	
`:BI`	布隆迪	`257`	8	无	00	
`:BJ`	贝宁	`229`	8	无	00	
`:BL`	圣巴泰勒米	`590`	9	0	00	
`:BM`	百慕大	`1`	10	无	011	NANP，委托给美国
`:BN`	文莱	`673`	7	无	00	
`:BO`	玻利维亚	`591`	8	无	00	
`:BQ`	博内尔/圣尤斯特歇斯/萨巴	`599`	7	无	00	

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:BR`	巴西	`55`	10-11	0	00	
`:BS`	巴哈马	`1`	10	无	011	NANP，委托给美国
`:BT`	不丹	`975`	8	无	00	
`:BV`	布维岛	`47`	8	无	00	委托给挪威
`:BW`	博茨瓦纳	`267`	8	无	00	
`:BY`	白俄罗斯	`375`	9	8	810	
`:BZ`	伯利兹	`501`	7	无	00	
`:CA`	加拿大	`1`	10	无	011	NANP，委托给美国
`:CC`	科科斯群岛	`61`	9	0	0011	委托给澳大利亚
`:CD`	刚果民主共和国	`243`	9	0	00	
`:CF`	中非共和国	`236`	8	无	00	
`:CG`	刚果共和国	`242`	9	无	00	
`:CH`	瑞士	`41`	9	0	00	
`:CI`	科特迪瓦	`225`	10	无	00	
`:CK`	库克群岛	`682`	5	无	00	
`:CL`	智利	`56`	9	0	00	

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:CM`	喀麦隆	`237`	9	无	00	
`:CN`	中国	`86`	11	无	00	
`:CO`	哥伦比亚	`57`	10	0	00	
`:CR`	哥斯达黎加	`506`	8	无	00	
`:CU`	古巴	`53`	8	0	119	
`:CV`	佛得角	`238`	7	无	00	
`:CW`	库拉索	`599`	7-8	无	00	
`:CX`	圣诞岛	`61`	9	0	0011	委托给澳大利亚
`:CY`	塞浦路斯	`357`	8	无	00	
`:CZ`	捷克共和国	`420`	9	无	00	
`:DE`	德国	`49`	10-12	0	00	
`:DJ`	吉布提	`253`	8	无	00	
`:DK`	丹麦	`45`	8	无	00	
`:DM`	多米尼加	`1`	10	无	011	NANP，委托给美国
`:DO`	多米尼加共和国	`1`	10	无	011	NANP，委托给美国
`:DZ`	阿尔及利亚	`213`	9	0	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:EC	厄瓜多尔	593	9	0	00	
:EE	爱沙尼亚	372	7-8	无	00	
:EG	埃及	20	10	0	00	
:EH	西撒哈拉	212	9	0	00	委托给摩洛哥
:ER	厄立特里亚	291	7	0	00	
:ES	西班牙	34	9	无	00	
:ET	埃塞俄比亚	251	9	0	00	
:FI	芬兰	358	9-10	0	00	
:FJ	斐济	679	7	无	00	
:FK	福克兰群岛	500	5	无	00	
:FM	密克罗尼西亚	691	7	无	011	
:FO	法罗群岛	298	6	无	00	
:FR	法国	33	9	0	00	
:GA	加蓬	241	7	无	00	
:GB	英国	44	10	0	00	
:GD	格林纳达	1	10	无	011	NANP，委托给美国
:GE	格鲁吉亚	995	9	0	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:GF	法属圭亚那	594	9	0	00	
:GG	根西岛	44	10	0	00	委托给英国
:GH	加纳	233	9	0	00	
:GI	直布罗陀	350	8	无	00	
:GL	格林兰	299	6	无	00	
:GM	冈比亚	220	7	无	00	
:GN	几内亚	224	9	无	00	
:GP	瓜德罗普	590	9	0	00	
:GQ	赤道几内亚	240	9	无	00	
:GR	希腊	30	10	无	00	
:GS	南乔治亚	500	5	无	00	委托给福克兰
:GT	危地马拉	502	8	无	00	
:GU	关岛	1	10	无	011	NANP，委托给美国
:GW	几内亚比绍	245	7	无	00	
:GY	圭亚那	592	7	无	001	
:HK	香港	852	8	无	001	
:HM	赫德岛和麦克唐纳岛	61	9	0	0011	委托给澳大利亚

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:HN	洪都拉斯	504	8	无	00	
:HR	克罗地亚	385	9	0	00	
:HT	海地	509	8	无	00	
:HU	匈牙利	36	9	06	00	
:ID	印度尼西亚	62	10-12	0	001	
:IE	爱尔兰	353	9	0	00	
:IL	□色列	972	9	0	00	
:IM	马恩岛	44	10	0	00	委托给英国
:IN	印度	91	10	0	00	
:IO	英属印度洋领地	246	7	无	00	
:IQ	伊拉克	964	10	0	00	
:IR	伊朗	98	10	0	00	
:IS	冰岛	354	7	无	00	
:IT	意大利	39	9-10	无	00	保留固定电话的前导 0
:JE	泽西岛	44	10	0	00	委托给英国
:JM	牙买加	1	10	无	011	NANP，委托给美国

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:JO	约旦	962	9	0	00	
:JP	日本	81	10	0	010	
:KE	肯尼亚	254	9	0	000	
:KG	吉尔吉斯斯坦	996	9	0	00	
:KH	柬埔寨	855	8-9	0	001	
:KI	基里巴斯	686	5	无	00	
:KM	科摩罗	269	7	无	00	
:KN	圣基茨和尼维斯	1	10	无	011	NANP，委托给美国
:KP	朝鲜	850	10	0	00	
:KR	韩国	82	9-10	0	001	
:KW	科威特	965	8	无	00	
:KY	开曼群岛	1	10	无	011	NANP，委托给美国
:KZ	哈萨克斯坦	7	10	8	810	
:LA	老挝	856	10	0	00	
:LB	黎巴嫩	961	8	0	00	
:LC	圣卢西亚	1	10	无	011	NANP，委托给美国

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:LI	列支敦士登	423	7	无	00	
:LK	斯里兰卡	94	9	0	00	
:LR	利比里亚	231	7-9	无	00	
:LS	莱索托	266	8	无	00	
:LT	立陶宛	370	8	8	00	
:LU	卢森堡	352	9	无	00	
:LV	拉脱维亚	371	8	无	00	
:LY	利比亚	218	9	0	00	
:MA	摩洛哥	212	9	0	00	
:MC	摩纳哥	377	8-9	无	00	
:MD	摩尔多瓦	373	8	0	00	
:ME	黑山	382	8	0	00	
:MF	圣马丁	590	9	0	00	
:MG	马达加斯加	261	9	0	00	
:MH	马绍尔群岛	692	7	无	011	
:MK	北马其顿	389	8	0	00	
:ML	马里	223	8	无	00	

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:MM`	缅甸	`95`	8-10	0	00	
`:MN`	蒙古	`976`	8	无	001	
`:MO`	澳门	`853`	8	无	00	
`:MP`	北马里亚纳群岛	`1`	10	无	011	NANP，委托给美国
`:MQ`	马提尼克	`596`	9	0	00	
`:MR`	毛里塔尼亚	`222`	8	无	00	
`:MS`	蒙特塞拉特	`1`	10	无	011	NANP，委托给美国
`:MT`	马耳他	`356`	8	无	00	
`:MU`	毛里求斯	`230`	8	无	00	
`:MV`	马尔代夫	`960`	7	无	00	
`:MW`	马拉维	`265`	9	0	00	
`:MX`	墨西哥	`52`	10	无	00	
`:MY`	马来西亚	`60`	9-10	0	00	
`:MZ`	莫桑比克	`258`	9	无	00	
`:NA`	纳米比亚	`264`	9	0	00	
`:NC`	新喀里多尼亚	`687`	6	无	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:NE	尼日尔	227	8	无	00	
:NF	诺福克岛	672	5-6	无	00	
:NG	尼日利亚	234	10	0	009	
:NI	尼加拉瓜	505	8	无	00	
:NL	荷兰	31	9	0	00	
:NO	挪威	47	8	无	00	
:NP	尼泊尔	977	10	0	00	
:NR	瑙鲁	674	7	无	00	
:NU	纽埃	683	4	无	00	
:NZ	新西兰	64	8-9	0	00	
:OM	阿曼	968	8	无	00	
:PA	巴拿马	507	8	无	00	
:PE	秘鲁	51	9	0	00	
:PF	法属波利尼西亚	689	8	无	00	
:PG	巴布亚新几内亚	675	8	无	05	
:PH	菲律宾	63	10	0	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:PK	巴基斯坦	92	10	0	00	
:PL	波兰	48	9	无	00	
:PM	圣皮埃尔和密克隆	508	6	无	00	
:PN	皮特凯恩群岛	64	8-9	0	00	委托给新西兰
:PR	波多黎各	1	10	无	011	NANP，委托给美国
:PS	巴勒斯坦	970	9	0	00	
:PT	葡萄牙	351	9	无	00	
:PW	帕劳	680	7	无	011	
:PY	巴拉圭	595	9	0	00	
:QA	卡���尔	974	8	无	00	
:RE	留尼旺	262	9	0	00	
:RO	罗马尼亚	40	9	0	00	
:RS	塞尔维亚	381	9	0	00	
:RU	俄罗斯	7	10	8	810	
:RW	卢旺达	250	9	0	00	
:SA	沙特阿拉伯	966	9	0	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:SB	所罗门群岛	677	5-7	无	00	
:SC	塞舌尔	248	7	无	00	
:SD	苏丹	249	9	0	00	
:SE	瑞典	46	9	0	00	
:SG	新加坡	65	8	无	001	
:SH	圣赫勒拿	290	4-5	无	00	
:SI	斯洛文尼亚	386	8	0	00	
:SJ	斯瓦尔巴特	47	8	无	00	委托给挪威
:SK	斯洛伐克	421	9	0	00	
:SL	塞拉利昂	232	8	0	00	
:SM	圣马力诺	378	10	无	00	
:SN	塞内加尔	221	9	无	00	
:SO	索马里	252	8	无	00	
:SR	苏里南	597	7	无	00	
:SS	南苏丹	211	9	0	00	
:ST	圣多美和普林西比	239	7	无	00	
:SV	萨尔瓦多	503	8	无	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:SX	荷属圣马丁	1721	7	无	00	
:SY	叙利亚	963	9	0	00	
:SZ	斯威士兰	268	8	无	00	
:TC	特克斯和凯科斯	1	10	无	011	NANP，委托给美国
:TD	乍得	235	8	无	00	
:TG	多哥	228	8	无	00	
:TH	泰国	66	9	0	001	
:TJ	塔吉克斯坦	992	9	无	810	
:TK	托克劳	690	4	无	00	
:TL	东帝汶	670	7-8	无	00	
:TM	土库曼斯坦	993	8	8	810	
:TN	突尼斯	216	8	无	00	
:TO	汤加	676	5-7	无	00	
:TR	土耳其	90	10	0	00	
:TT	特立尼达和多巴哥	1	10	无	011	NANP，委托给美国
:TV	图瓦卢	688	5	无	00	

代码	国家/地区	**E.164** 前缀	国家长度	中继	**IDD**	备注
`:TW`	台湾	`886`	9	0	002	
`:TZ`	坦桑尼亚	`255`	9	0	00	
`:UA`	乌克兰	`380`	9	0	00	
`:UG`	乌干达	`256`	9	0	00	
`:US`	美国	`1`	10	无	011	NANP 基本规则
`:UY`	乌拉圭	`598`	8	0	00	
`:UZ`	乌兹别克斯坦	`998`	9	无	810	
`:VA`	梵蒂冈	`39`	9-10	无	00	委托给意大利
`:VC`	圣文森特和格林纳丁斯	`1`	10	无	011	NANP，委托给美国
`:VE`	委内瑞拉	`58`	10	0	00	
`:VG`	英属维尔京群岛	`1`	10	无	011	NANP，委托给美国
`:VI`	美属维尔京群岛	`1`	10	无	011	NANP，委托给美国
`:VN`	越南	`84`	9-10	0	00	
`:VU`	瓦努阿图	`678`	5-7	无	00	

代码	国家/地区	E.164 前缀	国家长度	中继	IDD	备注
:WF	瓦利斯和富图纳	681	6	无	00	
:WS	萨摩亚	685	5-7	无	0	
:XK	科索沃	383	8	0	00	
:YE	也门	967	9	0	00	
:YT	马约特	262	9	0	00	委托给留尼旺
:ZA	南非	27	9	0	00	
:ZM	赞比亚	260	9	0	00	
:ZW	津巴布韦	263	9	0	00	

特殊翻译行为

1. CLI 阻止前缀去除

在格式翻译之前，去除 CLI 阻止前缀:

```
输入:    *67555123456
步骤 1: 去除 *67 → 555123456
步骤 2: 翻译 → 1555123456 (如果是美国)
```

2. SIP 参数去除

分号后的参数被去除:

```
输入:   61403123456;npdi;rn=+61400000000
步骤 1: 去除 ;npdi;rn=... → 61403123456
步骤 2: 翻译 → 61403123456
```

3. 非数字字符去除

去除所有非数字字符（除了 `+`）：

```
输入:   +61 (403) 123-456
步骤 1: 去除格式 → +61403123456
步骤 2: 翻译 → 61403123456
```

4. 超拨截断（澳大利亚非地理号码）

澳大利亚的 `13` 和 `1300/1800` 号码通常□比号码更多的数字拨打；网络会忽略额外的数字，翻译也会如此：

```
输��:   18007383773（1800 反向）
步骤 1: 识别 1800 号码 → 保留前 10 位数字
步骤 2: 翻译 → 611800738377
```

翻译后设置的变量

变量	值	描述
`destination_number`	E.164 格式	标准化的目的地号码
`tas_destination_number`	E.164 格式	与 `destination_number` 相同（两个都设置□兼容）
`effective_caller_id_number`	E.164 格式	标准化的源号码（MO 呼叫）

翻译失败时发生的情况

场景：未定义的国家代码

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

结果：`{:error, "未定义的国家代码"}` - 呼叫被拒绝

场景：无效的号码格式

输入：`"abc123"`（包含字母）
步骤 1：去除非数字 → `"123"`
步骤 2：太短，无法匹配任何模式
结果：可能按原样通过或根据拨号计划逻辑拒绝

最佳实践：始终使用正确的 E.164 号码在 HSS 中验证用户订阅。

测试数字翻译

Web UI 翻译测试器（`/translate`）：

1. 在控制面板中导航到 `/translate`
2. 从下拉菜单中选择国家代码
3. 输入任何格式的测试号码
4. 查看翻译后的 E.164 输出
5. 测试多种格式□进行验证

常见测试场景：

- 本地短代码 → E.164
- 国家格式 (0NSN) → E.164
- 国际格式 (+CC) → E.164
- 带有 CLI 前缀的号码 → 去除并翻译
- 带有格式（空格、破折号）的号码 → 清洁的 E.164
- 国家非地理号码（澳大利亚 13/1300/1800） → E.164

故障排除数字翻译

问题：呼叫失败，显示 **"UNALLOCATED_NUMBER"**

1. 检查翻译后的号码格式：
   - 使用 `/translate` 工具测试号码
   - 验证输出是否符合预期的 E.164 格式
   - 确认国家代码和区号是否正确
2. 检查 **Sh** 查找：
   - 翻译后的号码用于 Sh 查询
   - 使用 `/sh_test` 测试翻译后的号码
   - 验证订阅者是否存在该 MSISDN
3. 检查拨号计划变量：
   - 查看日志中的 `destination_number` 值
   - 确认翻译发生在拨号计划之前

问题：应用了错误的区号

```
# 配置
config :tas, number_translate: %{
  country_code: :AU,
  localAreaCode: "617"  # 对于您的地区错误
}

# 输入：12345678（8 位本地号码）
# 输出：6161712345678（不正确 - 双区号）
# 修复：为您的部署设置正确的 localAreaCode
```

问题：国际号码未被识别

检查号码是否包含国家代码：

- ✅ `+61403123456` 或 `61403123456` → 被识别
- ❌ `0403123456` 在错误的 country_code 配置中 → 错误路由

MO 和 MT 的区别

MT（被叫）的处理流程：

- 接收来自运营商的呼入请求
- 根据被叫号码查找对应的 SIP 账户
- 通过预设的 Sh 接口查询用户状态

MO (主叫呼出) 流程:

- 接收来自用户的呼出请求
- 对主叫号码进行鉴权
- 通过预设的接口向 Sh
- 呼叫结束后，生成话单记录（CDR）

号码处理

1. 国际号码处理流程：
    - 检查 `country_code` 是否在允许列表中
    - 根据国家代码添加前缀
2. 国内号码处理流程：
    - `localAreaCode` 是否在允许的区号列表中
    - 根据区号添加前缀
3. 本地号码处理流程：
    - 检查号码长度
    - 如果0NSN 开头
    - 如果+CC 开头
    - 根据号码长度添加相应的前缀
4. 特殊号码处理：
    - 检查“特殊号码列表”中
    - 根据特殊规则处理

验证 E.164 输出是否符合预期

5. 记录您的编号计划:

定义用户将使用的格式
在 /translate 工具中测试每种格式
培训运营人员了解预期格式

在线计费系统（OCS）集成

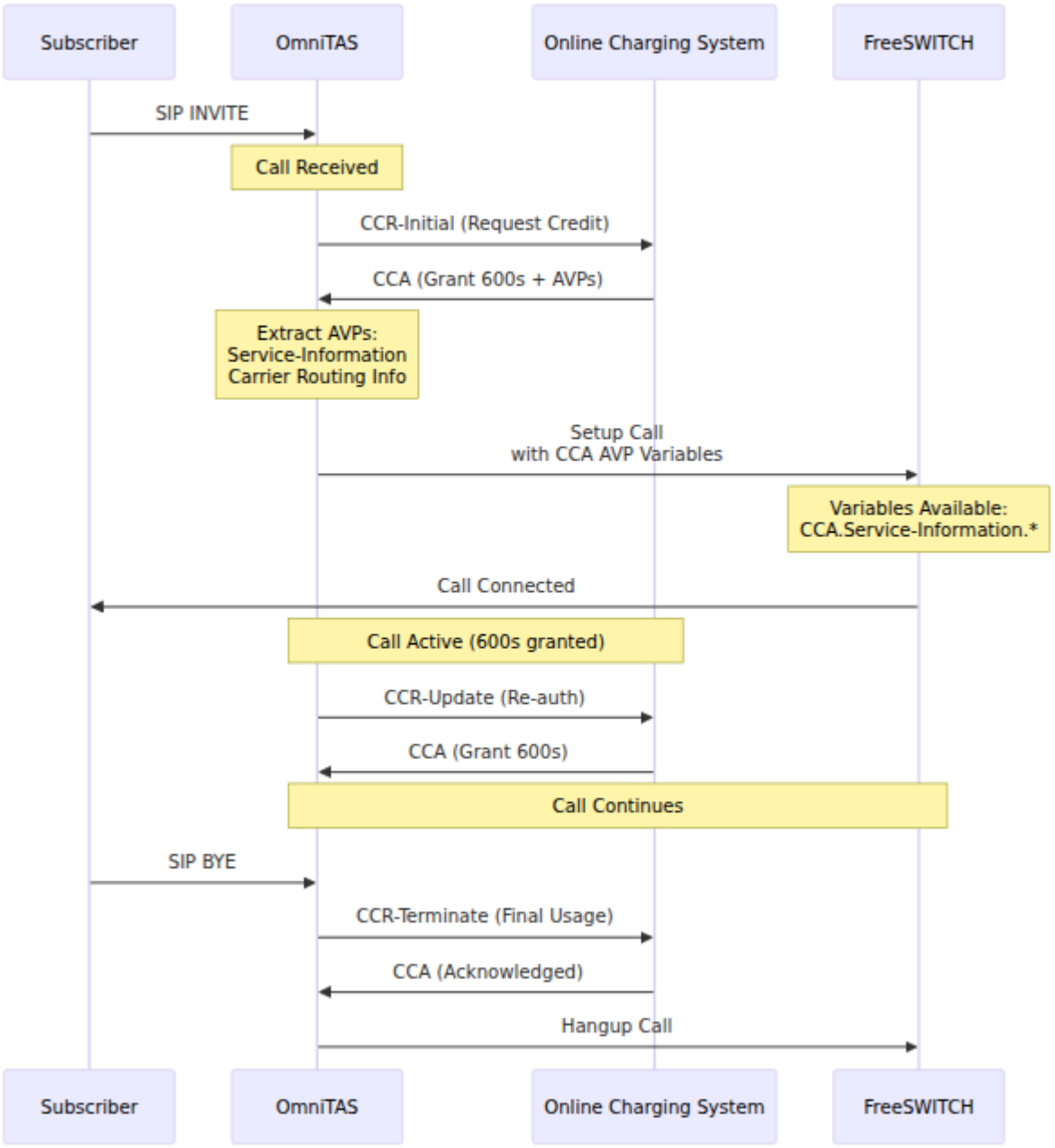
关于 OmniTAS 与在线计费系统通过 Diameter Ro 接口集成的综合指南，包括实时信用控制、AVP 提取和 FreeSWITCH 变量映射。

目录

- 架构概述
- 控制通话是否收费的因素
- 信用控制流程
- AVP 解析和变量映射
- 配置
- FreeSWITCH 集成
- 应答和挂断通知
- Diameter 消息
- 指标
- 故障排除
- 参考
  - FreeSWITCH 通道变量
  - AVP 代码参考

架构概述

OmniTAS 根据 3GPP TS 32.299 实现了 Diameter Ro 接口□进行实时在线计费。该系统通过在呼叫设置之前向 OCS 请求信用来授权呼叫，在呼叫期间监控信用，并在终止时报告最终使用情况。



关键组件

信用控制请求 **(CCR):**

- **CCR-Initial (**类型 **1)**: 在呼叫设置之前发送□请求初始信用授权
- **CCR-Update (**类型 **2)**: 在活动呼叫期间发送□进行重新授权或临时更新
- **CCR-Terminate (**类型 **3)**: 在呼叫终止时发送，报告最终使用情况

信用控制应答 **(CCA):**

- 包含授予的服务单位（□秒为单位的时间配额）

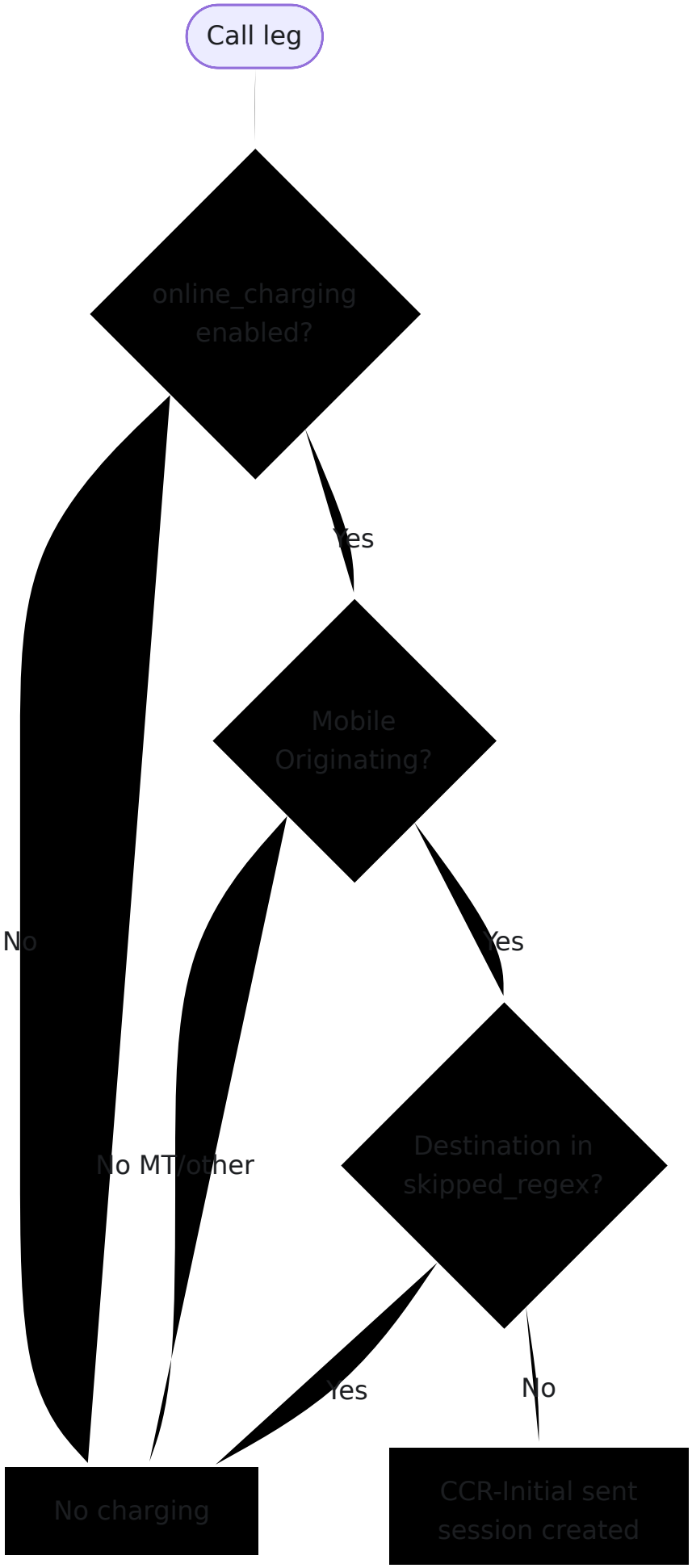
- 包括带有附加计费数据的供应商特定 AVP
- 提供路由信息、收费方详细信息和服务标识符

控制通话是否收费的因素

通话仅在下三个条件 全部满足 时产生 Diameter CCR。这些条件在 OmniTAS 为呼叫构建移动发起 (MO) 拨号计划时进行评估：

1. 在线计费已启用：`online_charging.enabled` 为 `true`。当为 `false` 时，OmniTAS 会在本地授权每个呼叫，并且从不联系 OCS。
2. 呼叫为移动发起 **(MO)**：仅对 MO 分支收费。移动终止 **(MT)** 呼叫从不收费：MT 路径不执行 OCS 授权，因此不会为其创建信用控制会话。
3. 目的地不在豁免名单中：拨打的号码与 `skipped_regex` 中的任何模式不匹配（例如紧急号码、服务代码）。

如果全部满足，OmniTAS 会发送 **CCR-Initial** 并打开一个 SIP Call-ID 为键的信用控制会话。从那时起，**CCR-Update** 和 **CCR-Terminate** 由应答/挂断事件驱动（请参见 应答和挂断通知），并且仅为具有开放会话的呼叫分支生成。MT 分支、桥接 B 分支和豁免呼叫因此不会产生 CCR。


Call leg
online_charging enabled?
Yes
No
Mobile Originating?
Yes
No MT/other
Destination in skipped_regex?
Yes
No
No charging
CCR-Initial sent session created

> 注意：到达 OCS 并不保证呼叫连接。成功授权 零 授予单位会阻止呼叫。请参见 信用耗尽和呼叫阻止。阻止发生在发送 CCR 之后；它与上述决定是否发送 CCR 的条件不同。

从拨号计划中零收费一个路线

`skipped_regex` 完全移除目的地的收费 - 没有会话，没有 CCR-I，因此没有信用检查。这对于紧急号码是正确的行为，但对于希望在不计费的情况下继续授权的路线（免费内部通话、免费电话范围、促销目的地、测试号码）则不是。

对于这些，请在拨号计划中设置通道变量 **`tas_zero_rate=true`**：

```
<action application="set" data="tas_zero_rate=true"/>
```

该变量在应答和挂断事件中从通道读取，因此可从任何分机设置，针对任何条件 - 它不携带 路线是免费的知识。

在 FreeSWITCH 执行拨号计划时，CCR-Initial 已经发送（OmniTAS 在构建拨号计划时授权呼叫），因此订阅者仍然会进行信用检查，且信用不足的呼叫者仍然会被阻止。该标志抑制的是 使用报告：

	正常路线	**`tas_zero_rate=true`**
CCR-Initial (授权)	发送	发送
CCR-Update 在应答时	发送	不发送
定期 CCR-Update	发送	不发送
信用耗尽时的中途切断	适用	不适用
CCR-Terminate	发送，报告消耗的 `CC-Time`	发送，报告 `CC-Time` **0**
对余额的净影响	为通话扣费	未受影响

CCR-T 中的其他所有内容保持真实 - 时间戳、Cause-Code、Service-Information - 因此 OCS 仍然会记录该通话的准确 CDR，费用为零，并完全释放 CCR-Initial 预留。

注意: 由于未发送 CCR-Update，OCS 会在整个通话期间保留 CCR-Initial 预留，并仅在 CCR-Terminate
时释放它。在长时间的零收费通话中，这可能会使订阅者在并发呼叫中看起来信用不足，直到清除。

priv/templates/mo_dialplan.xml 提供了一个 Zero-Rated-Routes 扩展，其中包含注释示例（内部通话和目的地前缀）。零收费默认 关闭 - 取消注释或添加您想要的条件。例如，要使内部通话免费，使用 OmniTAS 从拨打号码的 Sh 查找中设置的 on_net_status 变量：

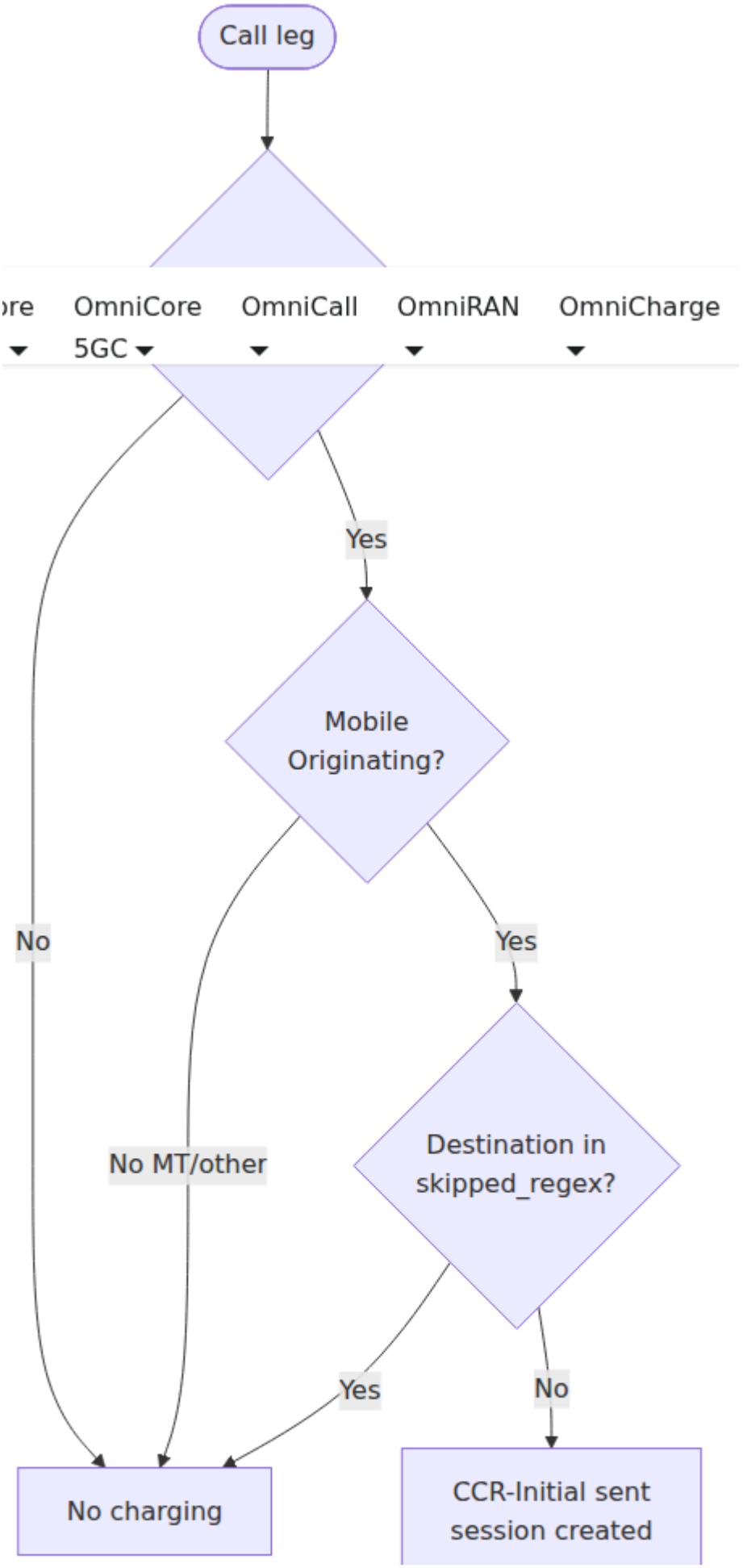
<extension name="Zero-Rated-Routes" continue="true">

<condition field="${on_net_status}" expression="^true$" break="never">

<action application="set" data="tas_zero_rate=true"/>
</condition>
</extension>

信用控制流程

呼叫授权序列


Call leg
ore
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Yes
Mobile Originating?
No
Yes
No MT/other
Destination in skipped_regex?
Yes
No
No charging
CCR-Initial sent session created

信用耗尽和呼叫阻止

呼叫设置时的阻止。如果 OCS 返回成功的 CCA（结果代码 `2001`，但授予的单位为 零（或显式拒绝，如 `4012` (DIAMETER_CREDIT_LIMIT_REACHED) 或 `4010` (DIAMETER_END_USER_SERVICE_DENIED)），OmniTAS 将该呼叫视为没有信用，并挂断原因 `OUTGOING_CALL_BARRED` 阻止它。订阅者永远不会连接。这是未配置或信用不足的订阅者的预期结果。

通话中途耗尽。如果在设置时授予信用，但后来耗尽（定期 CCR-Update 返回零单位或信用限制错误），OmniTAS 将终止进行中的呼叫。配置了 `credit_exhaustion_announcement` 时，它会在挂断之前将呼叫者转移到公告；否则，它会直接挂断。

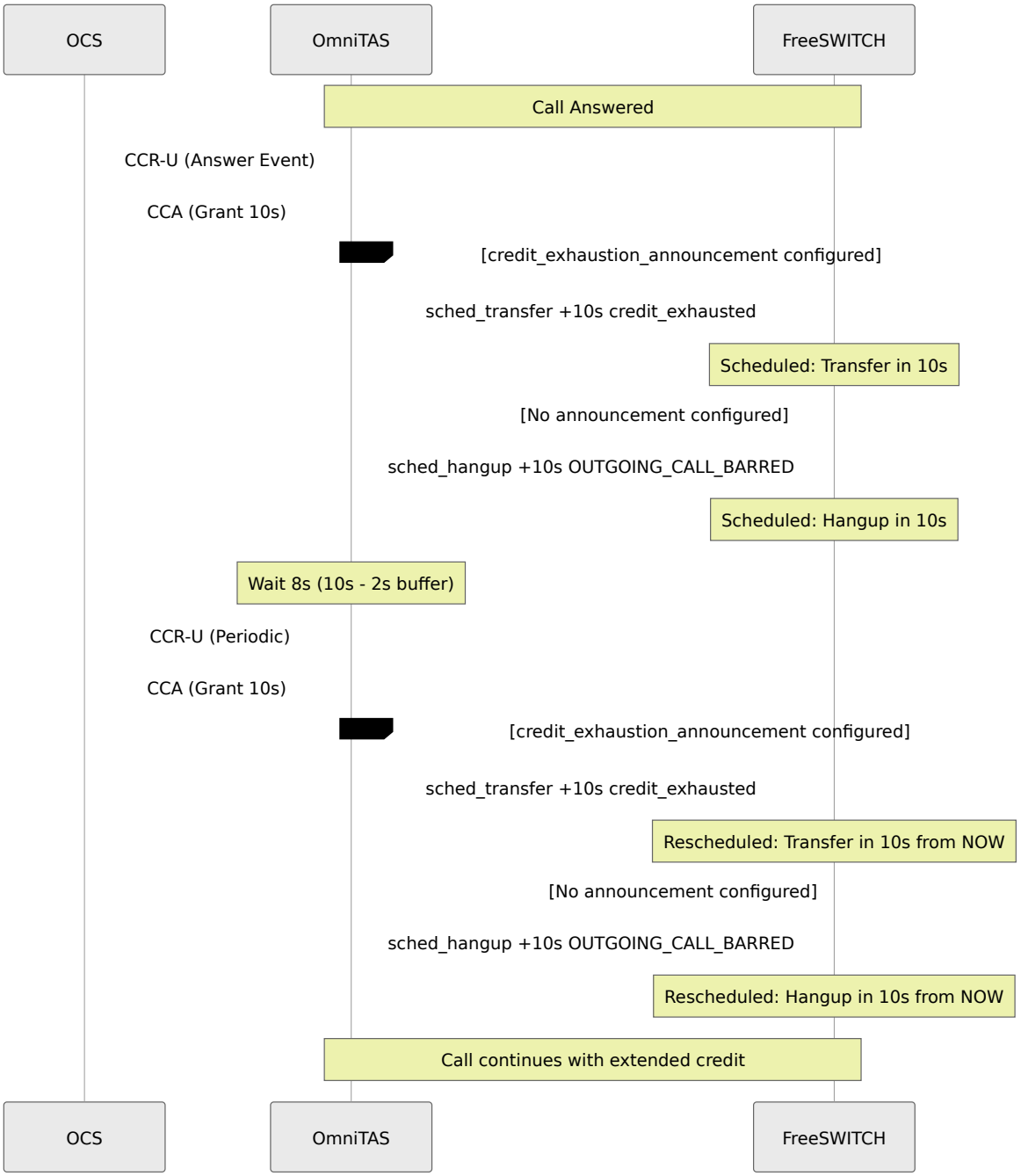
CCR-Terminate 在耗尽时触发，在任何清理之前。当余额耗尽时，OmniTAS 会 立即 发送 CCR-Terminate（收费应答→耗尽），播放信用耗尽公告或挂断。这是故意的：订阅者花费在“您已余额不足”提示上花费的时间是 不收费的，因此 CDR 必须在信用耗尽时结束，而不是在用户从录音中挂断时结束。会话在那时终止，因此最终的 FreeSWITCH 挂断不会找到会话并且不会发送任何内容（幂等，不会有第二个 CCR-T）。这适用于无论清理是转移到公告还是直接挂断，因此不依赖于 FreeSWITCH 挂断原因。

OmniTAS 支持多种处理信用耗尽的机制，自动集成计划挂断和信用耗尽公告。

动态重新调度的计划挂断

当 `schedule_hangup_auth` 启用时，OmniTAS 会调度一个 FreeSWITCH 计时器，在授予的信用到期时自动终止呼叫。每次通过 CCR-Update 响应授予新信用时，该计时器会 动态重新调度。

工作原理：



缓冲逻辑：

OmniTAS 在授予的信用到期之前发送 CCR-Update 消息，□确保持续服务。缓冲时间可□通过 `ccr_update_buffer_seconds` 配置（默认：2 秒）。

示例时间线：

- **T+0s**：呼叫应答，OCS 授予 10s，计时器调度到 T+10s
- **T+8s**：发送 CCR-U（10s - 2s 缓冲）
- **T+8.1s**：OCS 授予 10s，计时器重新调度到 T+18.1s（从现在起 10s）
- **T+16.1s**：发送 CCR-U

- **T+16.2s**: OCS 授予 10s，计时器重新调度到 T+26.2s
- 呼叫持续，只要 OCS 不断授予信用

需要关注的日志：

```
[OCS HANGUP RESCHEDULE] Found UUID <uuid> for call <id> -
rescheduling timer to 10s from now
[SCHED TRANSFER] Scheduling transfer to credit_exhausted dialplan
for <uuid> in 10s
[OCS HANGUP RESCHEDULE] Successfully rescheduled timer for call
<id> (UUID: <uuid>)
```

集成: schedule_hangup_auth + credit_exhaustion_announcement

当 两个 功能都启用时，OmniTAS 会自动使用计划 转移 而不是直接挂断，允许呼叫者在呼叫终止之前听到公告。

未配置公告的情况：

```
config :tas, :online_charging,
  schedule_hangup_auth: true,
  credit_exhaustion_announcement: nil
```

→ 使用 `sched_hangup` - 当信用到期时直接挂断

配置公告的情况：

```
config :tas, :online_charging,
  schedule_hangup_auth: true,
  credit_exhaustion_announcement:
"${base_dir}/sounds/en/us/callie/misc/8000/credit_exhausted.wav"
```

→ 使用 `sched_transfer` - 转移到 `credit_exhausted` 拨号计划，播放公告然后挂断

转移的工作原理：

1. OmniTAS 设置 `tas_call_reason=credit_exhausted` 通道变量
2. 调度转移到 `ims_as` 拨号计划上下文中的 `credit_exhausted` 扩展
3. 当计时器触发时：

FreeSWITCH 将 A-leg 转移到 credit_exhausted 拨号计划桥接自动断开，B-leg 收到 BYE
拨号计划向 A-leg 播放公告
呼叫在公告后终止

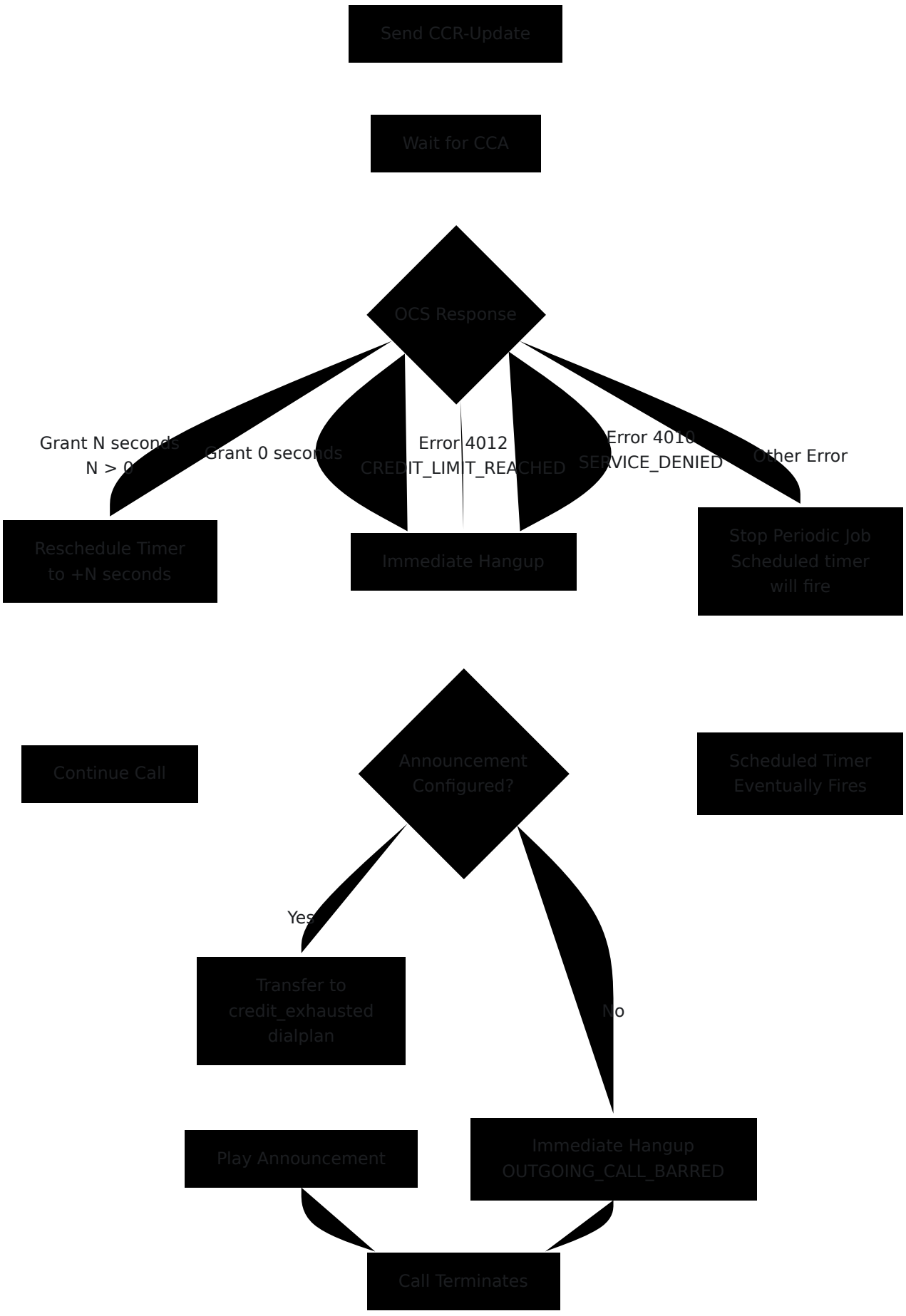
好处：

呼叫者听到专业的公告，而不是突然断开
B-leg（被叫方）不会听到公告
CCR-T 仍然发送实际使用情况
公告路径：必须相对于 FreeSWITCH 基目录（使用 ${base_dir} 变量）

CCR-Update 期间的即时信用耗尽

如果 OCS 在 CCR-Update 期间 拒绝信用 或返回 零秒，OmniTAS 会立即触发信用耗尽处理，覆盖任何计划计时器。

OCS 响应场景：


Send CCR-Update
Wait for CCA
OCS Response
Grant N seconds
N > 0
Grant 0 seconds
Error 4012
CREDIT_LIMIT_REACHED
Error 4010
SERVICE_DENIED
Other Error
Reschedule Timer
to +N seconds
Immediate Hangup
Stop Periodic Job
Scheduled timer
will fire
Continue Call
Announcement
Configured?
Scheduled Timer
Eventually Fires
Yes
No
Transfer to
credit_exhausted
dialplan
Play Announcement
Immediate Hangup
OUTGOING_CALL_BARRED
Call Terminates

□□□□□□□□

OCS 响应	动作	日志
`{:ok, 0}` (零秒)	立即信用耗尽挂断	`Credit exhausted (zero seconds allocated) - triggering immediate hangup`
`{:error, 4012}` (CREDIT_LIMIT_REACHED)	立即信用耗尽挂断	`Credit exhausted (4012 CREDIT_LIMIT_REACHED) - triggering immediate hangup`
`{:error, 4010}` (END_USER_SERVICE_DENIED)	立即信用耗尽挂断	`Service denied (4010 END_USER_SERVICE_DENIED) - triggering immediate hangup`
`{:error, reason}` (其他错误)	停止定期 CCR 作业，计划计时器触发	`Periodic CCR failed with error <reason> - Stopping job`
`{:ok, N}` 其中 N > 0	重新调度计时器到 +N 秒	`Periodic CCA allocated Ns, will send next CCR-U in (N-buffer)s`

优先级：立即信用耗尽处理 优先 于计划计时器。如果 OCS 在 T+8s 拒绝信用，但计时器计划在 T+10s，则在 T+8s 立即挂断发生，计划计时器变得无关紧要。

中途信用拒绝的示例时间线：

```
T+0s:   呼叫应答
T+0.1s: OCS 授予 10s → 计时器计划到 T+10.1s
T+8s:   发送 CCR-U（缓冲 = 2s）
T+8.1s: OCS 返回 0 秒 → 立即转移到 credit_exhausted 拨号计划
T+8.2s: 向呼叫者播放公告
T+10s:  呼叫终止（计划计时器无关紧要）
```

即时信用耗尽的日志：

```
[warning] Credit exhausted (zero seconds allocated) - triggering
immediate hangup
[warning] Hanging up call <id> (UUID: <uuid>) due to credit
exhaustion
[info] Credit exhaustion announcement config:
"${base_dir}/sounds/..."
[info] Playing announcement before hangup: ...
[info] Setting tas_call_reason=credit_exhausted for <uuid>
[info] Transferring to credit exhausted dialplan: uuid_transfer
<uuid> credit_exhausted XML ims_as
```

总结：信用耗尽机制

OmniTAS 提供两种互补机制：

1. 计划计时器（`schedule_hangup_auth`）：
   - 授予信用到期时自动���断/转移
   - 在每个 CCR-U 响应时动态重新调度
   - 使用缓冲逻辑在到期前发送 CCR-U
   - 与公告功能集成
2. 即时耗尽处理：
   - 在 OCS 拒绝信用时触发 CCR-U
   - 覆盖计划计时器
   - 支持公告播放
   - 处理特定的 Diameter 错误代码

这两种机制都尊重 `credit_exhaustion_announcement` 配置，并将在配置时播放配置的音频，然后终止呼叫。

AVP 解析和变量映射

概述

OmniTAS 自动从信用控制应答消息中提取属性值对 (AVP)，并将其作为通道变量提供给 FreeSWITCH。这使得拨号计划逻辑可使用 OCS 提供的数据进行路由决策、计费目的或呼叫处理。

支持的 **AVP** 类型：

- 简单值 (UTF8String, Unsigned32, Integer32)
- 带有嵌套结构的分组 AVP
- 供应商特定 AVP（例如，3GPP 服务信息）

变量命名约定：AVP 被展平为带有前缀 `CCA` 的点表示法通道变量：

```
CCA.<AVP-Name>.<Nested-AVP-Name>.<Value-AVP-Name> = "value"
```

常见 **AVP** 映射

服务信息 **AVP (3GPP)**

服务信息分组 AVP (AVP 代码 873, 供应商 ID 10415) 包含 IMS 特定的计费详细信息：

示例 **OCS** 响应：

```
Service-Information
  ├── IMS-Information
  │     ├── Carrier-Select-Routing-Information: "1408"
  │     └── Node-Functionality: 6
  └── Alternate-Charged-Party-Address: "NickTest"
```

结果 **FreeSWITCH** 变量：

```
CCA.Service-Information.Carrier-Select-Routing-Information =
"1408"
CCA.Service-Information.Alternate-Charged-Party-Address =
"NickTest"
```

在拨号计划中访问：变量使用点表示法和连字符，如上所示：

```
<action application="log" data="INFO Carrier: ${CCA.Service-
Information.Carrier-Select-Routing-Information}"/>
```

使用 **uuid_dump** 查看**:** 在 FreeSWITCH 控制台或 ESL 中，变量 `variable_` 前缀出现:

```
variable_CCA.Service-Information.Carrier-Select-Routing-
Information: 1408
variable_CCA.Service-Information.Alternate-Charged-Party-Address:
NickTest
```

注意**:** FreeSWITCH 在变量名称中保留点和连字符。这些变量在所有拨号计划上下文和应用程序中均有效。

授予服务单位 **AVP**

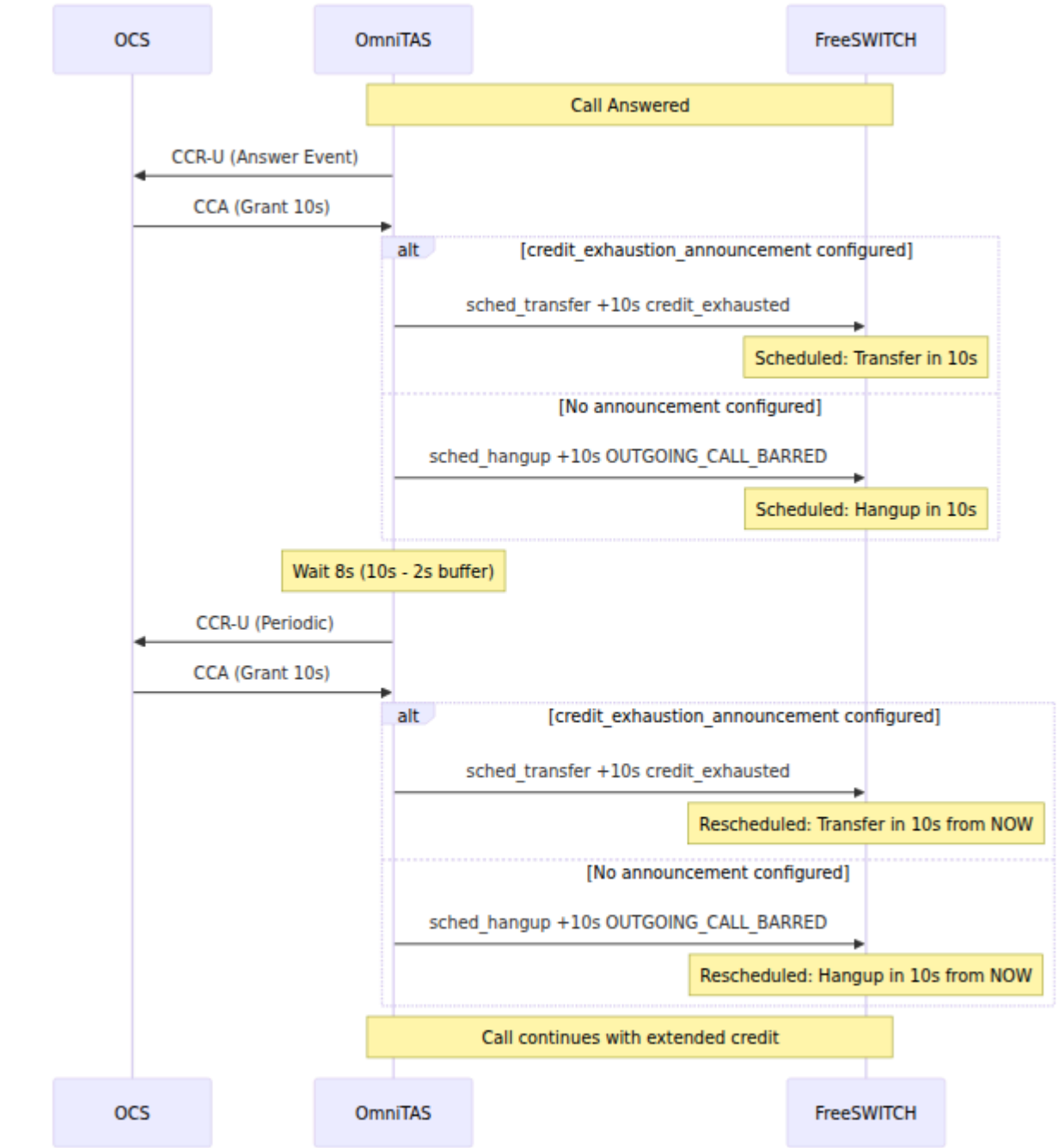
时间配额被提取并可用:

OCS 响应**:**

```
Granted-Service-Unit
  └── CC-Time: 600
```

变量**:**

```
allocated_time = 600
```



AVP 处理逻辑

处理规则：

1. 分组 **AVP** 为变量名称层次结构添加一个级别，但本身没有值
2. 简单 **AVP** 映射到具有其完整点路径的变量
3. 供应商特定 **AVP** 的处理与标准 AVP 相同
4. 未知 **AVP** 安全跳过而不产生错误

示例：多级嵌套

OCS CCA 结构：

```
服务信息（分组）
  ├── IMS 信息（分组）
  │     ├── 节点功能: 6
  │     ├── 节点角色: 1
  │     ├── 主叫方地址: "tel:+313380000000670"
  │     └── 时间戳（分组）
  │           ├── SIP 请求时间戳: "2026-01-24T22:40:18Z"
  │           └── SIP 响应时间戳: "2026-01-24T22:40:18Z"
  └── IN 信息（分组）
        └── 实际被叫号码: "15551234567"
```

创建的 **FreeSWITCH** 变量：

```
CCA.Service-Information.IMS-Information.Node-Functionality = "6"
CCA.Service-Information.IMS-Information.Role-Of-Node = "1"
CCA.Service-Information.IMS-Information.Calling-Party-Address =
"tel:+313380000000670"
CCA.Service-Information.IMS-Information.Time-Stamps.SIP-Request-
Timestamp = "2026-01-24T22:40:18Z"
CCA.Service-Information.IMS-Information.Time-Stamps.SIP-Response-
Timestamp = "2026-01-24T22:40:18Z"
CCA.Service-Information.IN-Information.Real-Called-Number =
"15551234567"
```

配置

在线计费参数

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	在线计费的主开关。当 `false` 时[授权（不发送 CCR）。请参见 控制[
`requested_units_seconds`	整数	否	`0`	每次预留请求的通话时间（秒），在 CC Initial/Update 的 `Reque` `Service-Unit` 中。`0`（默[**RSU**（AVP 存在但不携带 `CC-` OCS 选择授予。这与诺基亚 TAS CGRateS OCS 的匹配，后者[己的授予。设置一个正值根据 RFC OCS 预留特定的 `CC-Time`。
`report_and_reserve`	布尔值	否	`false`	3GPP SCUR 使用报告，与诺基 `true` 时，每个 CCR-Updat Terminate 都携带一个 `Used` `Unit`，报告自上次报告来消耗的 量），加上 `3GPP-Reportin` 案更新时为 `RATING_CONDIT` `(6)`，在定期更新时为 `QUOTA_E` `(3)`，在终止时为 `FINAL (2)` 量，并在终止时结算上次授予的未使用部 时，仅 CCR-T 报告使用情况（从答 因）。通话期间的增量 `CC-Time` 间。
`service_identifier`	整数	否	`1`	每个 CCR 中发送的稳定、预配置的 `Identifier`（AVP 439[`{Service-Context-Id`

参数	类型	必需	默认	描述
				`Identifier}` 键入计费，因此 而不是随机值。
`periodic_ccr_time_seconds`	整数	否	`10`	当 OCS 不返回可用授予时，CCR 间的回退间隔（秒）。在正常操作中，下 间是根据授予的信用动态计算的（`all` `- ccr_update_buffer`
`ccr_update_buffer_seconds`	整数	否	`2`	授予的信用到期前的安全缓冲区（秒）。 `(allocated_time - b` 一个 CCR-U，便在信用用尽之前 秒。
`schedule_hangup_auth`	布尔值	否	`false`	当授予的信用到期时，启用自动呼叫挂断 时，OmniTAS 根据每个 CCA `allocated_time` 调度 Fr 时器，并在每个 CCR-U 响应时重新 `credit_exhaustion_a` 一起使用。
`credit_exhaustion_announcement`	字符串	否	`nil`	信用耗尽公告的音频文件路径。当与 `schedule_hangup_aut` 的 转移 在挂断之前播放公告。当单独配 时播放公告。路径必须使用 FreeSW 式：`"${base_dir}/sour` 为 `nil` 直接挂断而不公告。
`skipped_regex`	列表[字符串]	否	`[]`	与 目标号码 匹配的正则表达式模式列表 OCS（没有会话，没有 CCR，没 号码和服务代码（例如，`"^911$"` 常有用。要继续授权目标但不收费，请使 `tas_zero_rate` 变量 - 请 率路由。

Diameter □□□□

□□	□□	□□	□□	□□
`origin_host`	□□□□	□	-	OmniTAS Diameter □□□FQDN□□□□□ Diam □□□□□□□ □□`"tas01.epc.mnc123.mcc456.3gppnetw`
`origin_realm`	□□□□	□	-	OmniTAS Diameter □□□□□□□□□□□ □□`"epc.mnc123.mcc456.3gppnetwork.or`
`destination_realm`	□□□□	□	-	OCS Diameter □□□□□□□□□□□□□□□□□
`destination_host`	□□□□	□	`nil`	□□ OCS Diameter □□□□ `nil` □□□□□ `destination_realm` □□□□□□□□□□□□□□□□□ □□

配置示例

```
config :tas, :online_charging,
  # 主开关
  enabled: true,

  # 0 = 空 RSU，让 OCS 决定授予（与诺基亚 TAS / CGRateS 匹配）。
  # 仅为根据请求单位计费的标准 OCS 设置正值。
  requested_units_seconds: 0,

  # 稳定的预配置服务标识符
  service_identifier: 1,

  # 回退重新授权间隔（正常时间动态来自授予）
  periodic_ccr_time_seconds: 10,

  # 在授予的信用到期前 2 秒重新授权
  ccr_update_buffer_seconds: 2,

  # 根据授予的信用调度挂断
  schedule_hangup_auth: true,

  # 在信用耗尽挂断前播放公告
  credit_exhaustion_announcement: "ivr/ivr-
account_balance_low.wav",

  # 跳过 OCS 的紧急呼叫和语音信箱
  skipped_regex: [
    "^911$",       # 紧急（美国）
    "^000$",       # 紧急（澳大利亚）
    "^\*86$"      # 语音信箱访问
  ]

config :tas, :diameter,
  # 服务身份
  origin_host: "tas01.epc.mnc001.mcc001.3gppnetwork.org",
  origin_realm: "epc.mnc001.mcc001.3gppnetwork.org",

  # OCS 路由
  destination_realm: "epc.mnc001.mcc001.3gppnetwork.org",
  destination_host: nil  # 基于领域的路由
```

工作原理：

当接收到呼叫时：

1. 检查目标号码是否与 `skipped_regex` 模式匹配
2. 如果匹配，呼叫绕过 OCS（对紧急服务有用）
3. 如果不匹配，CCR-Initial 发送到 OCS 的 `destination_realm`
4. 解析 CCA 响应□获取授予的单位和 AVP
5. 将 AVP 映射到 FreeSWITCH 变量（请参见 AVP 映射）
6. 呼叫继续进行，`allocated_time` 和 AVP 数据可用
7. 在呼叫期间每 `periodic_ccr_time_seconds` 发送 CCR-Update
8. 如果启用 `schedule_hangup_auth`，则在信用到期时自动挂断
9. 呼叫完成时发送 CCR-Terminate

用例：

- 基本 **OCS**：使用默认值启用标准信用控制
- 高价值呼叫：将 `periodic_ccr_time_seconds` 减少到 30 秒□进行频繁的重新授权
- 预付费服务：启用 `schedule_hangup_auth` 并设置 `credit_exhaustion_announcement`
- 紧急合规：将紧急号码添加到 `skipped_regex` □确保始终连接

FreeSWITCH 集成

在拨号计划中访问 **AVP** 变量

从 CCA 消息中提取的 AVP 数据作为通道变量在 FreeSWITCH 拨号计划中可用：

```
<extension name="Route_with_OCS_Data">
  <condition field="destination_number" expression="^(.+)$">

    <!-- 从 OCS 访问运营商路由信息 -->
    <action application="log"
            data="INFO Carrier Code: ${CCA.Service-
Information.Carrier-Select-Routing-Information}"/>

    <!-- 从 OCS 访问收费方 -->
    <action application="log"
            data="INFO Charged Party: ${CCA.Service-
Information.Alternate-Charged-Party-Address}"/>

    <!-- 访问授予的时间 -->
    <action application="log"
            data="INFO Allocated Time: ${allocated_time}
seconds"/>

    <!-- 根据运营商代码路由 -->
    <action application="set"
            data="carrier_code=${CCA.Service-Information.Carrier-
Select-Routing-Information}"/>
    <action application="bridge"

data="sofia/external/$1@carrier-${carrier_code}.sip.example.com"/>

  </condition>
</extension>
```

变量可用性

时机:

- 变量在 FreeSWITCH 呼叫设置 之前 设置
- 在整个呼叫持续期间可用
- 在呼叫转移和更新中保持不变

范围:

- 通道范围（特定于单个呼叫腿）

- 不会被桥接/转移的腿继承
- 在所有拨号计划应用中安全使用

示例用例

1. 基于 OCS 数据的运营商选择

使用 OCS 提供的运营商代码来路由呼叫：

```
<extension name="Carrier_Selection">
  <condition field="${CCA.Service-Information.Carrier-Select-
Routing-Information}" expression="^(.+)$">
    <action application="bridge"

data="sofia/external/${destination_number}@carrier-$1.example.com"/>
  </condition>

  <!-- 如果未指定运营商则回退 -->
  <condition field="${CCA.Service-Information.Carrier-Select-
Routing-Information}" expression="^$">
    <action application="bridge"
            data="sofia/external/${destination_number}@default-
carrier.example.com"/>
  </condition>
</extension>
```

工作原理：OCS 在 Service-Information AVP 中返回运营商代码 "1408"。FreeSWITCH 根据此数据将呼叫路由到 `carrier-1408.example.com` 网关。

2. 替代计费方

根据 OCS 响应将计费路由到不同的方：

```
<extension name="Alternate_Billing">
  <condition field="${CCA.Service-Information.Alternate-Charged-
Party-Address}" expression="^(.+)$">

    <!-- 记录计费方以用于 CDR -->
    <action application="set"
            data="billed_party=$1"/>
    <action application="export"
            data="billed_party=$1"/>

    <!-- 包含在 SIP 头中 -->
    <action application="set"
            data="sip_h_X-Billed-Party=$1"/>

    <action application="bridge"

data="sofia/external/${destination_number}@trunk.example.com"/>
  </condition>
</extension>
```

工作原理： OCS 指定替代收费方（例如，企业账户）。OmniTAS 从 AVP 中提取 "NickTest"，并使其在拨号计划中可用于 CDR 记录和 SIP 头插入。

3. 有时间限制的呼叫与警告

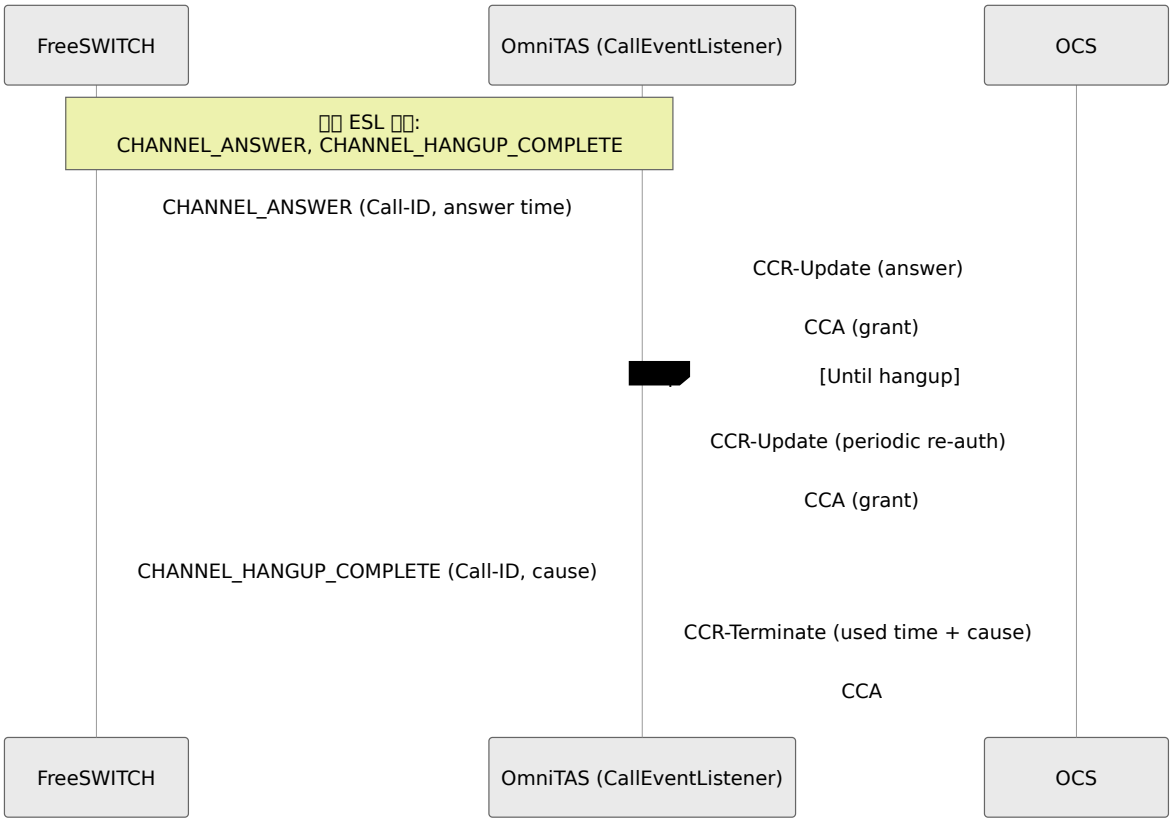
在信用到期之前提供警告：

```
<extension name="Credit_Warnings">
  <condition field="destination_number" expression="^(.+)$">

    <!-- 在挂断前 30 秒调度警告 -->
    <action application="set"
            data="warning_time=${expr(${allocated_time} - 30)}"/>

    <action application="sched_hangup"
            data="+${allocated_time} ALLOTTED_TIMEOUT"/>

    <action application="sched_broadcast"
            data="+${warning_time} playback::ivr/ivr-
account_balance_low.wav"/>

    <action application="bridge"
            data="sofia/external/$1@trunk.example.com"/>
  </condition>
</extension>
```

工作原理：使用 OCS 的 `allocated_time` 来调度自动挂断，并在断开连接前 30 秒播放警告公告。

答复与挂断通知

在 CCR-Initial 授权呼叫后，OmniTAS 仍然需要知道 何时接听呼叫（□开始计量并发送 CCR-Update）和 何时结束（□发送 CCR-Terminate）。这些事件通过 FreeSWITCH 的事件套接字层（ESL）获取。



工作原理:

- OmniTAS 保持与本地 FreeSWITCH 的持久入站 ESL 连接，并订阅 `CHANNEL_ANSWER` 和 `CHANNEL_HANGUP_COMPLETE`。
- 在 `CHANNEL_ANSWER` 上，它读取 SIP Call-ID 和实际应答时间戳，并发送开始计量的 CCR-Update。
- 在 `CHANNEL_HANGUP_COMPLETE` 上，它读取 SIP Call-ID 和 FreeSWITCH 挂断原因，并发送带有最终使用时间的 CCR-Terminate。
- 事件通过 SIP Call-ID 匹配到信用控制会话。没有会话的呼叫腿（MT 腿、桥接 B 腿、豁免呼叫）的事件将被忽略。这就是防止未收费呼叫产生 CCR 的原因。
- 重复应答是幂等的。桥接呼叫的两个腿都会触发 `CHANNEL_ANSWER`，并可能解析到相同的 Call-ID；只有第一个发送 CCR-Update 并启动周期发送器，其余的被忽略（否则会产生重复的 CCR-U ���双重周期发送器）。
- 转移结束收费会话。 `BLIND_TRANSFER`/`ATTENDED_TRANSFER` 挂断被视为正常挂断 → CCR-Terminate（收费方已离开呼叫，或呼叫已转移到无信用公告）。一旦会话终止，任何后续的同一 Call-ID 的挂断将找不到会话并不发送任何内容。

`CallEventListener` 始终运行（真实部署总是有共置的 FreeSWITCH；仅在 `test_mode` 下跳过）。它不受单独开关的限制。是否发送 CCR 是由每个事件的 `online_charging.enabled` 决定的。关闭计费时，监听器仍然接收并记录 `CHANNEL_ANSWER`/`CHANNEL_HANGUP_COMPLETE` 的调试信息，但不发送 CCR。（早期版本有一个单独的 `esl_notifications` 切换；它被移除，因为它必须与 `online_charging.enabled` 一致，并且静默破坏了计费。它仅在计费开启但切换缺失时发送 CCR-Initial，而不发送 CCR-U/CCR-T。）

已弃用：`/call_event` HTTP 端点

早期版本通过 FreeSWITCH 拨号计划 `curl` 钩子检测应答/挂断，这些钩子 POST 到 OmniTAS 的 `/call_event` HTTP 端点。此机制已弃用。拨号计划钩子已被移除，ESL 现在是通知源。

`/call_event` 端点仍然存在并返回 `200 OK`，□便任何残留的拨号计划引用不会出错，但它执行 没有计费操作。升级���旧版本的运营商应确保其拨号计划不再依赖于此。

Diameter 消息

CCR-Initial（请求类型 1）

在呼叫设置之前发送□请求授权和初始信用分配。

发送的关键 **AVP**：

AVP	代码	类型	描述
Session-Id	263	UTF8String	会话标识符，在整个信用控制会话中保持不变（CCR-I/U/T 共享它）。根据 SIP Call-ID 确定性派生。
Auth-Application-Id	258	Unsigned32	根据 RFC 4006 的 Diameter 信用控制应用程序的值 `4`
Service-Context-Id	461	UTF8String	根据 TS 32.299 的 IMS 计费 `"000.000.12.32260@3gpp.org"`
CC-Request-Type	416	枚举	值 `1`（INITIAL_REQUEST）
CC-Request-Number	415	Unsigned32	序列号：`0` 为 **INITIAL_REQUEST**，然后为会话中的后续请求 `1`，`2`，… RFC 4006 §8.2
Service-Identifier	439	Unsigned32	稳定的预配置服务标识符（`service_identifier` 配置）
Subscription-Id	443	分组	订阅者 MSISDN 或 IMSI
Requested-Service-Unit	437	分组	预留请求：`CC-Time` = `requested_units_seconds`
Service-Information	873	分组	IMS 特定的呼叫细节（主叫/被叫方、节点角色、时间戳）

示例 **CCR-I**：

```
Session-Id: "tas01.example.org;1463927445;1744753804"
Auth-Application-Id: 4
CC-Request-Type: 1 (INITIAL_REQUEST)
CC-Request-Number: 0
Subscription-Id:
  - Subscription-ID-Type: 0 (END_USER_E164)
    Subscription-ID-Data: "313380000000670"
Multiple-Services-Credit-Control:
  - Service-Identifier: 1
    Requested-Service-Unit:
      - CC-Time: 300 (请求的预留)
Service-Information:
  - IMS-Information:
      - Role-Of-Node: 0 (ORIGINATING_ROLE)
      - Node-Functionality: 6 (AS)
      - Calling-Party-Address: "tel:+313380000000670"
      - Called-Party-Address: "tel:+15551234567"
```

> `Requested-Service-Unit` 携带真实的预留（而不是零）。OCS 决定实际的授予，并在 CCA 的 `Granted-Service-Unit` 中返回。

CCA（信用控制应答）

来自 OCS 的响应，包含授权决策和授予的信用。

接收的关键 **AVP**：

AVP	代码	类型	描述
Result-Code	268	Unsigned32	`2001` 表示成功。有关错误值，请参见 结果代码。
Granted-Service-Unit	431	分组	分配的信用（□秒为单位的时间）
Service-Information	873	分组	额外的计费数据（运营商信息、收���方等）

带 **AVP** 的示例 **CCA**：

```
Session-Id: "tas01.example.org;1769294418268;8a078232"
Result-Code: 2001 (DIAMETER_SUCCESS)
CC-Request-Type: 1
CC-Request-Number: 1
Granted-Service-Unit:
  - CC-Time: 600 (授予 10 分钟)
Service-Information:
  - IMS-Information:
      - Carrier-Select-Routing-Information: "1408"
  - Alternate-Charged-Party-Address: "NickTest"
```

结果变量:

```
allocated_time = 600
CCA.Service-Information.Carrier-Select-Routing-Information =
"1408"
CCA.Service-Information.Alternate-Charged-Party-Address =
"NickTest"
```

CCR-Update（请求类型 2）

在活动呼叫期间发送，用于定期重新授权或临时使用报告。

发送时机:

- 在呼叫应答时（来自 ESL 的 `CHANNEL_ANSWER`）：`CC-Request-Number` = `1`，在 `SIP-Response-Timestamp` 中携带实际应答时间
- 在活动呼叫期间定期，动态定时为 `allocated_time - ccr_update_buffer_seconds` 从每个授予（回退到 `periodic_ccr_time_seconds`）

与 **CCR-I** 的关键区别:

- `CC-Request-Type`: `2`（UPDATE_REQUEST）
- `CC-Request-Number`: 应答时为 `1`，然后在每次定期重新授权时递增
- `Requested-Service-Unit`: 下一个预留（`requested_units_seconds`）
- 与 CCR-Initial 相同的常量 `Session-Id`

> 消耗的时间在 CCR-Terminate 中报告（`Used-Service-Unit`），而不是在临时 CCR-Update 中。临时更新延长预留；最终使用在终止时结算。

CCR-Terminate（请求类型 3）

在呼叫挂断时（来自 ESL 的 `CHANNEL_HANGUP_COMPLETE`）发送，并进行最终使用报告。

关键 **AVP**：

- `CC-Request-Type`：`3`（TERMINATION_REQUEST）
- `CC-Request-Number`：序列中的下一个（例如 `2`）
- `Used-Service-Unit`：总消耗通话时间（`CC-Time`，自应答□来的秒数）
- `Termination-Cause`：`1`（DIAMETER_LOGOUT）根据 RFC 6733 §8.15
- `Cause-Code`（AVP 861，IMS-信息）：`0` 表示正常应答并释放的呼叫，`2` 表示会话设置不成功，依据 TS 32.299

结果代码

代码	名称	描述	**OmniTAS** 操作
2001	DIAMETER_SUCCESS	请求批准	解析 AVP，设置呼叫
4010	DIAMETER_END_USER_SERVICE_DENIED	订阅者服务被拒绝	拒绝呼叫并返回 `CALL_REJECTED`
4012	DIAMETER_CREDIT_LIMIT_REACHED	信用不足	拒绝呼叫并返回 `OUTGOING_CALL_BAR`
5003	DIAMETER_AUTHORIZATION_REJECTED	OCS 策略拒绝	拒绝呼叫并返回 `CALL_REJECTED`
5xxx	永久性故障	OCS 配置或系统错误	拒绝呼叫，记录错误

参考：RFC 6733 §7.1 和 3GPP TS 32.299

□标

> 请参阅 docs/metrics.md 获取完整的指标目录。□下指标与在线计费相关。

Diameter 请求 / 响应□标

CCRs 通过 `command` 标签按请求类型拆分，因此 CCR-Initial（设置），CCR-Update（临时重新认证）和 CCR-Terminate（拆除）可□独立跟踪。然后，失败的临时更新的峰值将单独可见。

□标：`diameter_requests_total`（计数器）：发送的 Diameter 请求
□标：`diameter_responses_total`（计数器）：接收到的 Diameter 响应

□标：`diameter_response_duration_milliseconds`（直方图）：请求往返时间

标签：

- `application` - `ro`（在线计费）或 `sh`（用户数据）
- `command` - `ccr_i`，`ccr_u`，`ccr_t`（Ro）或 `udr`（Sh）
- `result_code` -（仅响应）Diameter 结果代码：`2001`，`4012`，等；`0` = 超时 / 无法解析的回复
- `result` -（仅持续时间）`success`，`nocredit`，`error`

示例查询：

```
# 按请求类型的 CCR 速率
sum by (command) (rate(diameter_requests_total{application="ro"}[5m]))

# 每种请求类型的 CCR 错误率（任何不是 2001 的）
sum by (command) (rate(diameter_responses_total{application="ro",
result_code!="2001"}[5m]))

# 信贷限制拒绝（4012）
rate(diameter_responses_total{application="ro", result_code="4012"}
[5m])

# 按请求类型的第 95 百分位 CCR 延迟
histogram_quantile(0.95,
  sum by (le, command)
(rate(diameter_response_duration_milliseconds_bucket{application="ro'
[5m]))
)
```

信用控制配额□标

□标：`ro_charging_quota_seconds`

类型：直方图

描述：每个 CCR 观察到的配额（□秒为单位），用于验证计费的正确性

标签：

- `request_type` - `ccr_i`，`ccr_u`，`ccr_t`
- `kind` - `requested`（请求服务单元中的 CC-Time），`granted`（授予的服务单元，`0` = 无信贷），`used`（终止时的已使用服务单元）

示例查询：

```
# 呼叫设置时授予的配额中位数
histogram_quantile(0.5,
rate(ro_charging_quota_seconds_bucket{request_type="ccr_i",
kind="granted"}[5m]))

# 每秒的无信贷授予（零秒分配）
rate(ro_charging_quota_seconds_bucket{kind="granted", le="0"}[5m])

# 每秒报告的终止时的总使用秒数（与计费分钟的合理性检查）
rate(ro_charging_quota_seconds_sum{kind="used"}[5m])
```

OCS 授权与事件□标

□标：`ocs_authorization_attempts_total`（计数器）
标签：`result`（`success`，`nocredit`，`timeout`，`error`），`skipped`（`true` 如果通过正则表达式跳过，否则为 `false`）

□标：`online_charging_events_total`（计数器）：生命周期事件
标签：`event_type`（`authorize`，`answer`，`reauth`，`hangup`，`credit_exhaustion_hangup`，`hangup_rescheduled`），`result`（`success`，`nocredit`，`timeout`，`error`，`triggered`）

示例查询：

```
# 授权成功率（不包括跳过的）
sum(rate(ocs_authorization_attempts_total{result="success",
skipped="false"}[5m]))
  / sum(rate(ocs_authorization_attempts_total{skipped="false"}[5m]))

# 由于信贷耗尽而在通话中释放的呼叫
rate(online_charging_events_total{event_type="credit_exhaustion_hangu
[5m])
```

故障排除

FreeSWITCH 中未提供 AVP 变量

症状：

- FreeSWITCH 拨号计划无法访问 `${CCA.Service-Information.*}` 变量
- 变量显示为空或未定义

可能的原因：

1. OCS 未在 CCA 中返回服务信息 AVP
2. AVP 解析因意外结构而失败
3. 变量未导出到 FreeSWITCH 通道

解决方案：

1. 验证 **OCS** 响应包含 **AVP**

   检查 OmniTAS 日志中的 CCA 消息：

   ```
   [debug] Credit Control Answer: {:diameter_packet, ...}
   [debug] Parsed AVP variables: %{
     "CCA.Service-Information.Carrier-Select-Routing-Information"
   => "1408",
     "CCA.Service-Information.Alternate-Charged-Party-Address" =>
   "NickTest"
   }
   ```

   如果 "Parsed AVP variables" 为空 `%{}`，则 OCS 未返回预期的 AVP。

2. 检查 **AVP** 解析错误

   在日志中查找警告：

   ```
   [warning] got back another type of reply: {...}
   ```

   这表明 AVP 结构与预期格式不匹配。检查 Diameter 数据包结构。

3. 验证 FreeSWITCH 变量导出

在 FreeSWITCH 控制台或 ESL 中:

freeswitch> uuid_dump <call-uuid>

查找带有 variable_前缀和 CCA. 的变量名称:

variable_CCA.Service-Information.Carrier-Select-Routing-Information: 1408

variable_CCA.Service-Information.Alternate-Charged-Party-Address: NickTest

variable_CCA.Auth-Application-Id: 4
variable_CCA.Result-Code: 2001

注意: FreeSWITCH 保留变量名称中的点和连字符。它们在拨号计划中正常工作:

<action application="log" data="Carrier: ${CCA.Service-Information.Carrier-Select-Routing-Information}"/>

呼叫被拒绝，错误为 "unhandled"

症状:

日志显示: [warning] Could not authorize call::unhandled
有效的 CCA 响应 (结果代码 2001) 被拒绝
尽管 OCS 批准，呼叫仍然失败

可能的原因:

CCA 消息结构与预期模式不匹配
供应商特定的 AVP 在意外位置
AVP 位置索引不匹配

解决方案:

这是一个已知问题，在最近的版本中已修复。确保您正在运行当前版本。

以前的行为：模式匹配要求：

- 授予的服务单元 AVP 必须准确位于第 7 位
- 空的供应商特定 AVP 列表 `[]`

当前行为：模式匹配接受：

- 授予的服务单元 AVP 可□位于任何位置
- 非空的供应商特定 AVP 列表

如果问题仍然存在：

1. 从日志中捕获 CCA 数据包结构
2. 检查 AVP 是否符合预期的 Diameter 格式
3. 验证结果代码是否为 2001

OCS 在所有请求上超时

症状：

- 所有 CCR 请求超时
- 日志显示：`[debug] Got back response for authorize: {:error, :timeout}`
- 5 秒内未收到 CCA

可能的原因：

- 与 OCS/DRA 的网络连接问题
- 防火墙阻止 Diameter 端口 (3868)
- 不正确的 `destination_realm` 或 `destination_host`
- OCS 未对请求做出响应

解决方案：

1. 验证网络连接

   测试与 OCS 的 TCP 连接：

   ```
   telnet ocs.example.com 3868
   ```

应成功连接。如果连接被拒绝或超时，请检查防火墙规则。

2. 检查 Diameter 配置

验证 destination_realm 是否与 OCS 配置匹配:

config:tas,:diameter,
destination_realm: "epc.mnc001.mcc001.3gppnetwork.org" # 必须与
OCS 领域匹配

3. 查看 OCS 日志

检查 OCS 是否接收到 CCR 消息。如果 OCS 收到请求但未响应:

验证 OmniTAS 的 origin_host 是否被 OCS 识别
检查 OCS 对等配置是否允许来自 OmniTAS 的连接
验证服务上下文 ID 和应用程序 ID 是否与 OCS 预期匹配

信贷耗尽未挂断呼叫

症状:

呼叫超出授予的信贷时间继续进行
当 allocated_time 到期时没有自动挂断 schedule_hangup_auth 已启用但未生效

可能的原因:

FreeSWITCH 的计划挂断未配置
schedule_hangup_auth 为 false
呼叫状态未正确跟踪

解决方案:

1. 验证配置

确保 schedule_hangup_auth 已启用:

```
config :tas, :online_charging,
  schedule_hangup_auth: true
```

2. 检查 **FreeSWITCH ESL** 连接

   验证 OmniTAS 是否可□向 FreeSWITCH 发送命令：

   ```
   [debug] Schedule Hangup Response: {:ok, "+OK"}
   ```

   如果出现错误或没有响应，请检查 FreeSWITCH 事件套接字配置。

3. 监控呼叫状态

   检查��叫 UUID 是否在呼叫状态中被跟踪：

   ```
   [debug] Setting Scheduled Hangup for call in 600 seconds
   ```

   如果未找到 UUID，则呼叫状态跟踪可能存在问题。

跳过的正则表达式未绕过 **OCS**

症状：

- 紧急呼叫 (911, 000) 仍然经过 OCS 授权
- 匹配 `skipped_regex` 模式的号码未被绕过
- 紧急呼叫延迟

可能的原因：

- 正则表达式模式语法错误
- 目标号码格式不匹配
- 正则表达式未正确转义

解决方案：

1. 验证正则表达式模式

测试正则表达式编译：

```
Regex.compile("^911$")  # 应返回 {:ok, ~r/^911$/}
```

常见错误：

- 缺少锚点：使用 `^911$` 而不是 `911`
- 转义：使用 `\*` 表示字面星号，而不是 `\*`

2. 检查号码格式

验证目标号码格式是否与模式匹配：

```
[debug] Checking if dialled number "911" matches skipped
regex...
```

如果号码格式为 "+1911"，但模式为 "^911$"，则不会匹配。

3. 示例模式

```
config :tas, :online_charging,
  skipped_regex: [
    "^911$",            # 美国紧急
    "^000$",            # 澳大利亚紧急
    "^112$",            # 国际紧急
    "^\*86$",          # 语音信箱（转义星号）
    "^1?800\d{7}$"     # 免费电话
  ]
```

参考

3GPP 规范

规范	标题	相关章节
TS 32.299	Diameter 计费应用	§6.3 (Ro 接口), §7.2 (AVP 定义)
TS 32.240	计费架构和原则	§5 (在线计费)
TS 29.229	Cx 和 Dx 接口	IMS 中的服务信息 AVP 使用

IETF RFCs

RFC	标题	相关章节
RFC 6733	Diameter 基础协议	§3 (协议概述), §7 (错误处理)
RFC 4006	Diameter 信贷控制应用	§8 (信贷控制消息)

AVP 代码参考

供应商 ID `0` = IETF 基础 (RFC 6733 / RFC 4006); 供应商 ID `10415` = 3GPP (TS 32.299).

完整的 **CCR** 结构（语音呼叫）

每个 AVP OmniTAS 在呼叫 Credit-Control-Request 中包含，分组如下:

```
Credit-Control-Request (Command 272, App 4)
├─ Session-Id (263)
├─ Origin-Host (264)
├─ Origin-Realm (296)
├─ Destination-Realm (283)
├─ Destination-Host (293)                          [仅在配置时]
├─ Auth-Application-Id (258)            = 4
├─ Service-Context-Id (461)             =
000.000.12.32260@3gpp.org
├─ CC-Request-Type (416)
├─ CC-Request-Number (415)
├─ Event-Timestamp (55)
├─ User-Name (1)                                   [仅在设置用户名时]
├─ Termination-Cause (295)                         [CCR-T 仅]
├─ Subscription-Id (443)
│   ├─ Subscription-Id-Type (450)       = 0 (END_USER_E164)
│   └─ Subscription-Id-Data (444)       = 用户的 MSISDN
├─ Multiple-Services-Credit-Control (456)
│   ├─ Service-Identifier (439)
│   ├─ Requested-Service-Unit (437)             [CCR-I, CCR-U]
│   │   └─ CC-Time (420)
│   └─ Used-Service-Unit (446)                  [CCR-T]
│       └─ CC-Time (420)
└─ Service-Information (873, v10415)
    ├─ IN-Information                              [运营商扩展]
    │   └─ Real-Called-Number
    └─ IMS-Information (876, v10415)
        ├─ Role-Of-Node (829)           = 0 (ORIGINATING_ROLE)
        ├─ Node-Functionality (862)     = 6 (AS)
        ├─ User-Session-Id (830)        = SIP Call-ID
        ├─ Calling-Party-Address (831)
        ├─ Called-Party-Address (832)
        ├─ Requested-Party-Address (1251)
        ├─ Time-Stamps (833)
        │   ├─ SIP-Request-Timestamp (834)
        │   ├─ SIP-Request-Timestamp-Fraction (2301)
        │   ├─ SIP-Response-Timestamp (835)              [一旦回答]
        │   └─ SIP-Response-Timestamp-Fraction (2302)    [一旦回答]
        └─ Cause-Code (861)                       [CCR-T]
```

OmniTAS 发送的 AVP (CCR)

In 列显示哪些请求类型携带 AVP: **I** = CCR-Initial, **U** = CCR-Update, **T** = CCR-Terminate.

AVP	代码	供应商	类型	In	值 / 来源
Session-Id	263	0	UTF8String	I U T	每个呼叫的常量（源自 SIP Cal I/U/T 中相同
Origin-Host	264	0	DiameterIdentity	I U T	OmniTAS Diameter (`origin_host`.`orig`
Origin-Realm	296	0	DiameterIdentity	I U T	`origin_realm`
Destination-Realm	283	0	DiameterIdentity	I U T	OCS 领域
Destination-Host	293	0	DiameterIdentity	I U T	仅在配置特定 OCS 主机时存在
Auth-Application-Id	258	0	Unsigned32	I U T	`4` (Diameter 信贷控制应
Service-Context-Id	461	0	UTF8String	I U T	`000.000.12.32260@`
CC-Request-Type	416	0	Enumerated	I U T	`1`=初始，`2`=更新，`3`=终[
CC-Request-Number	415	0	Unsigned32	I U T	初始为 `0`，然后为 `1`，`2`，...

AVP	代码	供应商	类型	In	值 / 来源
Event-Timestamp	55	0	Time	I U T	请求生成的时间
User-Name	1	0	UTF8String	I U T	可选；仅在提供用户名/IMSI 时
Termination-Cause	295	0	Enumerated	T	`1` (DIAMETER_LOG
Subscription-Id	443	0	Grouped	I U T	订阅者身份（见子项）
→ Subscription-Id-Type	450	0	Enumerated	I U T	`0` (END_USER_E164
→ Subscription-Id-Data	444	0	UTF8String	I U T	订阅者 MSISDN
Multiple-Services-Credit-Control	456	0	Grouped	I U T	信贷控制容器（见子项）
→ Service-Identifier	439	0	Unsigned32	I U T	`service_identifie` 置)
→ Requested-Service-Unit	437	0	Grouped	I U	预留请求

AVP	代码	供应商	类型	In	值 / 来源
→ → CC-Time	420	0	Unsigned32	I U	`requested_units_s`
→ Used-Service-Unit	446	0	Grouped	T	最终消耗
→ → CC-Time	420	0	Unsigned32	T	自回答□来消耗的秒数
Service-Information	873	10415	Grouped	I U T	3GPP 服务容器
→ IN-Information	-	10415	Grouped	I U T	运营商扩展，携带真实拨打的号码
→ → Real-Called-Number	-	10415	UTF8String	I U T	拨打（被叫）号码
→ IMS-Information	876	10415	Grouped	I U T	IMS 计费详情
→ → Role-Of-Node	829	10415	Enumerated	I U T	`0` (ORIGINATING_R 是被收费的腿
→ → Node-Functionality	862	10415	Enumerated	I U T	`6` (AS)
→ → User-Session-Id	830	10415	UTF8String	I U T	SIP Call-ID (OCS 关联[

AVP	代码	供应商	类型	In	值 / 来源
→ → Calling-Party-Address	831	10415	UTF8String	I U T	`tel:+<calling MSI`
→ → Called-Party-Address	832	10415	UTF8String	I U T	`tel:+<called numb`
→ → Requested-Party-Address	1251	10415	UTF8String	I U T	`tel:+<called numb`
→ → Time-Stamps	833	10415	Grouped	I U T	SIP 请求/响应时间
→ → → SIP-Request-Timestamp	834	10415	Time	I U T	INVITE 时间（整秒）
→ → → SIP-Request-Timestamp-Fraction	2301	10415	Unsigned32	I U T	INVITE 时间（毫秒）
→ → → SIP-Response-Timestamp	835	10415	Time	U T	回答（200 OK）时间：仅在回
→ → → SIP-Response-Timestamp-Fraction	2302	10415	Unsigned32	U T	回答时间（毫秒）：仅在回答后

AVP	代码	供应商	类型	In	值 / 来源
→ → Cause-Code	861	10415	Integer32	T	`0` 正常释放 / `2` 设置失败

> 代码显示为 `-` 的是运营商/供应商扩展，携带在 `Service-Information` 下；它们 没有 3GPP 分配的代码，并在 OmniTAS Diameter 字典中定义。

从 **OCS** 接收的 **AVP (CCA)**

AVP	代码	供应商	类型	描述
Result-Code	268	0	Unsigned32	`2001` = 成功；见 结果代码
Granted-Service-Unit	431	0	Grouped	分配的信贷
→ CC-Time	420	0	Unsigned32	授予的秒数 (`allocated_time`)。`0` ⇒ 无信贷 ⇒ 拦截/挂断
Service-Information	873	10415	Grouped	OCS 返回的可选计费数据
→ Carrier-Select-Routing-Information	2023	10415	UTF8String	运营商路由代码（映射到 FreeSWITCH 变量）
→ Alternate-Charged-Party-Address	1280	10415	UTF8String	计费方标识符（映射到 FreeSWITCH 变量）

所有在 CCA 中返回的 `Service-Information` 子 AVP 都被扁平化为 FreeSWITCH 通道 变量。请参阅 AVP 解析和变量映射 和 FreeSWITCH 通道变量。

FreeSWITCH 通道变量

所有提取的 AVP 数据作为 FreeSWITCH 通道变量可用:

变量名称	来源	示例值
`${allocated_time}`	授予的服务单元 / CC-Time	`600`
`${CCA.Session-Id}`	Session-Id AVP	`omni-as01.epc...;1769299669873;`
`${CCA.Result-Code}`	Result-Code AVP	`2001`
`${CCA.Auth-Application-Id}`	Auth-Application-Id AVP	`4`
`${CCA.CC-Request-Type}`	CC-Request-Type AVP	`1`
`${CCA.CC-Request-Number}`	CC-Request-Number AVP	`1`
`${CCA.CC-Time}`	CC-Time AVP (如果存在)	`600`
`${CCA.Origin-Host}`	Origin-Host AVP	`ocs01.epc.mnc001.mcc001.3gppnet`
`${CCA.Origin-Realm}`	Origin-Realm AVP	`epc.mnc001.mcc001.3gppnetwork.o`
`${CCA.Service-Information.Carrier-`	Service-Information → Carrier-	`1408`

变量名称	来源	示例值
`Select-Routing-Information}`	Select-Routing-Information	
`${CCA.Service-Information.Alternate-Charged-Party-Address}`	Service-Information → Alternate-Charged-Party-Address	`NickTest`

变量格式：

- 所有 CCA AVP 使用前缀 `CCA.`
- 嵌套 AVP 使用点表示法：`CCA.Parent.Child`
- 变量名称中的点和连字符被保留
- 在 uuid_dump 中，变量□ `variable_` 前缀出现

示例 **uuid_dump** 输出：

```
variable_allocated_time: 600
variable_CCA.Service-Information.Carrier-Select-Routing-
Information: 1408
variable_CCA.Service-Information.Alternate-Charged-Party-Address:
NickTest
variable_CCA.Result-Code: 2001
```

操作□南

□ 返回主文档

本文档涵盖了控制面板中可用的操作监控和管理功能。

相关文档

核心文档

- □ 主**README** - 概述和快速入门
- □ 配置□南 - 系统配置参考
- □ □标参考 - Prometheus指标和监控

监控与测试工具

- □ **HLR**和呼叫模拟器 - HLR和呼叫模拟的测试工具
- □ **IMS**会议服务器 - 会议管理和监控
- □ 拨号计划□标 - 拨号计划特定指标

呼叫处理与服务

- □ 拨号计划配置 - 呼叫路由和拨号计划参考
- □ **Sh**接口 - 用户数据测试
- □ 在线计费 - OCS测试
- □ 号码翻译 - 号码翻译测试
- □ 语音邮件 - 语音邮件管理

集成接口

- □ **SS7 MAP** - HLR/MAP测试
- ⚙ 补充服务 - 紧急呼���、呼叫转移
- □ **HOMER**集成 - SIP跟踪和日志关联

操作

本节涵盖了在OmniTAS控制面板中可用的操作监控和管理功能。

目录

- 用户视图
- 呼叫详细记录（CDR）
- 活动呼叫监控
- IMS会议服务器
- 网关状态
- Diameter对等状态
- 日志查看器
- 呼叫模拟器
- HLR/MAP测试
- 其他视图

用户视图

用户视图提供了对存储在Sofia SIP注册数据库中的IMS用户注册的实时监控。

访问：在控制面板中导航到 `/subscribers`

功能

- 注册列表：查看所有活动用户注册
- 注册详情：点击任何注册□查看完整详情，包括：
  - SIP用户和领域
  - 联系URI
  - 注册状态和到期
  - 网络信息（IP、端口、主机名）
  - 认证详情
  - 从P-Access-Network-Info解析的网络位置信息（MCC/MNC、无线类型、小区ID），如果可用

□□□

□□□

功能

- 分页视图：浏览呼叫记录（每页100条，带有上一页/下一页控制）
- 高级搜索：强大的搜索功能，支持精确匹配、反向/排除和多个术语
- 列选择：自定义显示哪些字段
  - 点击 "列" 按钮打开列选择器模态
  - 选择/取消选择单个列
  - 全选 / 取消全选 快速操作
  - 选择在会话之间保持（保存到浏览器localStorage）
  - 显示 "X / Y 列" 计数器
- 可排序列：点击任何列标题进行排序（升序/降序）
  - 视觉指示器（▲升序，▼降序）
  - 排序的列□蓝色高亮显示
  - 排序更改时重置为第1页
- 多个过滤选项：
  - 文本搜索：在所有字段中使用高级运算符进行搜索
  - 日期范围过滤：按开始/结束日期和时间过滤（日期时间选择器）
  - 字段特定过滤：按确切字段值过滤（挂断原因、主叫ID、目的地、上下文）
  - 活动过滤显示：视觉芯片显示当前活动的过滤器
  - 清除所有：一键删除所有活动过滤器
- 详细信息：点击任何CDR行□展开并查看所有字段：
  - 呼叫方（主叫ID名称/号码，目的地号码）
  - 时间戳（开始、应答、结束）
  - 持续时间和计费秒数
  - 挂断原因（颜色编码：绿色＝正常，黄色＝取消，红色＝错误）
  - 呼叫UUID（A-leg和B-leg）

    - 上下文和账户代码
    - 所有可用数据库字段按字母顺序排列
- 颜色编码的挂断原因：
    - □ 绿色：`NORMAL_CLEARING`
    - □ 黄色：取消的呼叫
    - □ 红色：错误条件
- 总计数：实时显示总匹配记录
- 响应式布局：在较小屏幕上适当换行过滤器

如何使用

1. 基本查看：
    - 页面加载最新的100条CDR记录（按 `start_stamp` 降序排序）
    - 右上角显示总记录计数
    - 使用 上一页 / 下一页 按钮导航页面
    - 点击任何行□展开并查看所有字段
2. 自定义列：
    - 点击右上角的 "列" 按钮
    - 模态显示所有可用字段
    - 勾选/取消勾选字段□显示/隐藏列
    - 使用 "全选" 或 "取消全选" 进行快速选择
    - 设置会自动保存到浏览器
    - 关闭模态□应用更改
3. 排序数据：
    - 点击任何列标题按该字段排序
    - 第一次点击：降序（▼）
    - 第二次点击：升序（▲）
    - 第三次点击：返回降序
    - 排序的列□蓝色高亮显示
4. 搜索记录：

- 在 "搜索" 框中输入搜索查询
- 支持高级运算符（见下文搜索语法）
- 在多个字段中搜索：`caller_id_number`、`destination_number`、`uuid`、`caller_id_name`、`hangup_cause`
- 点击 "应用" 执行搜索

5. 按日期范围过滤：

- 使用 "开始日期" 和 "结束日期" 日期时间选择器
- 日期过滤需要两个日期
- 支持日期和时间选择
- 点击 "应用" 进行过滤

6. 按特定字段过滤：

- 从 "选择过滤字段" 下拉菜单中选择字段：
  - 挂断原因
  - 主叫ID
  - 目的地
  - 上下文
- 在 "输入过滤值" 中输入确切值
- 点击 "应用" 进行过滤

7. 组合过滤器：

- 所有过滤器可同时使用：
  - 文本搜索 + 日期范围 + 字段过滤器共同工作
- 活动过滤器显示为过滤表单下方的芯片
- ***击 "清除所有" 一次性删除所有过滤器

8. 查看详细信息：

- 点击任何CDR行展开
- 网格布局显示所有数据库字段
- 字段按字母顺序显示
- 挂**原因颜色编码便快速识别
- 再次点击行折叠

高级搜索语法

搜索框支持强大的查询语法，□便在多个字段中同时进行精确记录过滤。

搜索工作原理：

搜索引擎检查每个CDR记录中的 所有可搜索字段。当记录在 任何 这些字段中匹配您的搜索条件时，该记录将包含在结果中：

- `caller_id_number`
- `destination_number`
- `uuid`
- `caller_id_name`
- `hangup_cause`

搜索运算符（可以组合）：

1. 包含搜索（默认）：
    - 语法：`term`（无引号）
    - 匹配：记录的 任何字段包含 该术语
    - SQL：使用 `LIKE '%term%'` 在所有可搜索字段中连接 `OR`
    - 示例：`61480` 匹配 "61480123456"、"55561480999" 等。
2. 精确匹配：
    - 语法：`"term"`（带双引号）
    - 匹配：记录的 任何字段完全等于 该术语
    - SQL：使用 `= 'term'` 在所有可搜索字段中连接 `OR`
    - 示例：`"911"` 仅匹配完全 "911"，而不是 "9115" 或 "1911"
3. 反向/排除：
    - 语法：`!term`（感叹号前缀，无引号）
    - 匹配：记录的 任何字段不包含 该术语
    - SQL：使用 `NOT LIKE '%term%'` 在所有可搜索字段中连接 `AND`
    - 示例：`!NORMAL` ��除任何字段中包含 "NORMAL" 的记录
4. 精确反向/排除：

- 语法：`!"term"`（感叹号 + 双引号）
- 匹配：记录的 任何字段完全不等于 该术语
- SQL：使用 `!= 'term'` 在所有可搜索字段中连接 `AND`
- 示例：`!"NORMAL_CLEARING"` 排除任何字段完全等于 "NORMAL_CLEARING" 的记录

5. 多个术语与**AND**：

- 语法：`term1 AND term2`（不区分大小写的AND）
- 匹配：匹配 所有术语 的记录（每个术语可□匹配不同字段）
- 每个术语使用其自己的运算符进行处理（引号、!等）
- 术语在SQL中用 `AND` 组合
- 示例：`"911" AND "12345"` 查找在一个字段中有 "911" 和在另一个字段中有 "12345" 的记录

搜索执行逻辑：

```
对于每个CDR记录：
   对于正常搜索（无 !）：
      - 检查是否 **任何字段包含/等于** 该术语 → 如果为真则包含
      - SQL: field1 LIKE '%term%' OR field2 LIKE '%term%' OR ...

   对于反向搜索（!）：
      - 检查是否 **所有字段都不包含/等于** 该术语 → 如果为真则包含
      - SQL: field1 NOT LIKE '%term%' AND field2 NOT LIKE '%term%'
AND ...

   对于AND搜索：
      - 每个术语单独评估
      - 所有术语条件必须为真 → 如果为真则包含
      - SQL: (term1_conditions) AND (term2_conditions) AND ...
```

复杂搜索示例：

查询	工作原理	结果
`61480`	在所有字段中进行包含搜索	所有包含 "61480" 的记录（主叫、目的地、UUID等）
`"911"`	在所有字段中进行精确匹配	任何字段完全等于 "911" 的记录
`!NORMAL_CLEARING`	反向包含搜索	排除任何字段中包含 "NORMAL_CLEARING" 的记录（失败的呼叫）
`!"NORMAL_CLEARING"`	精确反向	排除任何字段完全等于 "NORMAL_CLEARING" 的记录
`"911" AND "12345"`	精确 "911" 和精确 "12345"	同时包含两个值的记录（例如，主叫="12345"，目的地="911"）
`!NORMAL AND 61480`	反向包含 "NORMAL" 和包含 "61480"	涉及 "61480" 的非正常呼叫
`!"ANSWER" AND !NORMAL`	精确反向 "ANSWER" 和反向包含 "NORMAL"	排除已接听的呼叫和任何包含 "NORMAL" 的记录
`61480 AND !NORMAL_CLEARING`	包含 "61480" 和反向包含 "NORMAL_CLEARING"	涉及 "61480" 的失败呼叫

实际用例：

- 查找特定号码：`61480123456` - 包含搜索找到部分匹配
- 查找精确紧急呼叫：`"911"` - 仅呼叫到完全 "911"
- 所有失败的呼叫：`!NORMAL_CLEARING` - 排除成功的呼叫
- 特定主叫的失败呼叫：`"61480123456" AND !NORMAL` - 组合精确主叫与反向
- 排除测试号码：`!test AND !demo` - 多个反向搜索
- 复杂调试：`61480 AND !"ANSWER" AND !CANCEL` - 包含一个术语，排除精确和部分其他

数据源

CDR数据直接从TAS CDR SQLite数据库查询。

架构可能因特定要求而在不同部署之间有所不同。

CDR导出选项

重要： CDR记录可□□多种格式导出，□支持与计费系统、分析平台和报告工具的集成。

CDR数据库架构和导出格式是特定于部署的。在设置系统时，请向您的集成工程师请求您所需的特定**CDR**输出格式。常见的导出格式包括：

- CSV（逗号分隔值）
- JSON（用于API集成）
- XML
- 直接数据库访问
- 自定义格式导出

您的集成工程师可□配置针对您的操作和计费要求量身定制的CDR导出机制。

用例

- 呼叫故障排除：按号码或UUID搜索特定呼叫□调试问题
- 计费对账：按日期范围过滤□匹配计费周期
- 质量分析：按挂断原因过滤□识别问题模式
- 紧急呼叫审计：搜索 "911" □验证紧急呼叫处理
- 客户支持：按主叫ID或目的地查找特定客户呼叫
- 模式分析：按持续时间或时间戳排序□识别异常
- 合规与记录保存：日期范围过滤用于法规报告
- 失败呼叫分析：使用 `!NORMAL_CLEARING` 查找所有失败的呼叫
- 基于上下文的报告：按上下文过滤□分析特定呼叫流

配置

默认可见列

您可□通过在 `config/runtime.exs` 中设置 `cdrs_field_list` 来配置在LiveView中 默认显示 的CDR字段：

```
config :tas,
  cdrs_field_list: [
    "caller_id_number",
    "destination_number",
    "start_stamp",
    "duration",
    "hangup_cause"
  ]
```

行为：

- 如果未设置 `cdrs_field_list`：默认显示所有可用CDR字段
- 如果设置了 `cdrs_field_list`：仅默认显示指定字段，但 所有其他字段仍可在列选择器中使用
- 如果列表中的字段在CDR数据中不存在，系统会自动跳过
- 字段名称可□指定为字符串或原子
- 用户可□随时从列选择器中手动选择其他列

用例：

- 设置一个干净的默认视图，仅显示基本字段
- 减少新用户的信息过载
- 在所有用户中标准化初始列布局
- 默认隐藏高级字段，但仍可访问

示例配置：

```
# 默认仅显示基本呼叫信息
cdrs_field_list: [
  "start_stamp",
  "caller_id_number",
  "destination_number",
  "duration",
  "billsec",
  "hangup_cause"
]
```

注意：此配置设置 □□□□ 列。所有CDR字段仍可在 "列" 选择器中使用 - 用户可□手动显示/隐藏任何需要的字段。

故障排除

未找到结果

1. 检查搜索词中的拼写错误
2. 尝试去掉引号□进行更广泛的搜索
3. 验证该术语是否存在于可搜索字段中
4. 检查日期范围是否过于严格

结果过多

1. 添加更多AND术语□缩小范围
2. 使用引号进行精确匹配
3. 应用日期范围过滤
4. 使用字段特定过滤

结果意外

1. 请记住，搜索适用于 所有可搜索字段
2. 检查该术语是否出现在意外字段中（如UUID）
3. 使用精确匹配□避免部分匹配
4. 验证反向逻辑（AND与OR）

提示

- 列选择：隐藏未使用的列□专注于相关数据并提高性能
- 组合过滤器：同时使用搜索 + 日期范围 + 字段过滤□进行精确查询
- 日期范围性能：缩小日期范围可更快返回结果，适用于大型数据库
- 排序以进行分析：按持续时间排序□查找长/短呼叫，或按时间戳查看呼叫模式
- 活动过滤芯片：使用视觉芯片验证当前活动的过滤器
- 持久设置：列选择在每个浏览器中保存，适用于不同的分析任务
- 颜色编码：快速扫描挂断原因 - 绿色表示良好，红色需要调查
- 可展开的详细信息：点击行□查看所有字段，而不干扰主视图
- 搜索运算符：掌握搜索语法□进行强大的过滤：
  - 使用引号进行精确匹配：`"911"`
  - 使用 `!` 排除：`!NORMAL_CLEARING`
  - 与AND组合：`"61480" AND !NORMAL`

分页：请记住，过滤器在页面之间保持 - 使用分页查看大型结果集

活动呼叫监控

活动呼叫视图显示有关系统中正在进行的呼叫的实时信息。

访问：在控制面板中导航到 /calls

功能

实时状态：活动呼叫会话的实时视图
呼叫详情：查看通道变量和呼叫状态信息
UUID跟踪：监控A-leg和B-leg呼叫标识符

IMS会议服务器

IMS会议服务器提供符合3GPP IMS标准（RFC 4579、RFC 4575、TS 24.147）的多方会议能力。

访问：在控制面板中导航到 /conference

文档：详细文档请参见 IMS会议服务器用户指南

功能

实时监控：活动会议和参与者的实时视图
会议统计仪表板：
活动会议数量
所有会议的总参与者
视频会议数量
锁定会议数量
服务器配置详情（域、MNC/MCC、最大参与者）
会议列表：查看所有活动会议，包括：
会议ID和SIP URI
当前参与者数量
会议创建者身份

- 会议详情：点击任何会议□展开并查看：
  - 完整的会议信息（状态、视频状态、锁定状态、录音状态）
  - 完整的参与者列表及其角色和状态
  - 参与者视频状态
- 会议控制操作：
  - 锁定/解锁会议□控制访问
  - 启用/禁用会议视频
  - 实时状态更新与操作反馈
- 自动刷新：可配置的自动刷新（默认：5秒）□进行实时监控

OmniTAS控制台管理

所有会议操作也可□通过OmniTAS控制台使用 `ims_conference` 命令进行：

```
ims_conference list                      # 列出所有活动会议
ims_conference info <conf_id>            # 显示会议详情
ims_conference stats                     # 显示服务器统计信息
ims_conference lock <conf_id>            # 锁定会议
ims_conference unlock <conf_id>          # 解锁会议
ims_conference video <conf_id> on|off    # 控制视频
ims_conference record <conf_id> start|stop  # 控制录音
ims_conference add <conf_id> <sip_uri>      # 添加参与者
ims_conference remove <conf_id> <uuid>      # 移除参与者
ims_conference destroy <conf_id>         # 终止会议
```

用例

- 操作监控：实时可见活动会议和资源使用情况
- 容量管理：监控参与者数量和视频使用情况□管理带宽
- 故障排除：诊断会议访问问题、参与者连接问题
- 会议控制：锁定会议□保护隐私，管理视频□控制带宽
- 合规性：监控和记录会议□确保合规性

3GPP合规性

会议服务器实现了关键的3GPP IMS会议规范：

- **TS 24.147**：使用IM核心网络子系统的会议

- **RFC 4579**: SIP呼叫控制 - 用户代理的会议
- **RFC 4575**: SIP事件包用于会议状态
- **RFC 5239**: 集中会议的框架

网关状态

监控连接到TAS的SIP网关/中继的状态和健康状况。

访问：在控制面板中导航到 `/gw`

功能

- 注册状态：查看网关注册状态
- 呼叫统计：跟踪进出呼叫和失败
- **Ping**监控： SIP OPTIONS ping时间和可达性
- 网关详情：完整的配置和状态信息

监控的□标

- SIP注册状态
- Ping时间（平均SIP OPTIONS响应时间）
- 正常运行时间（自配置文件重启□来的秒数）
- 进呼叫/出呼叫
- 失败的进呼叫/失���的出呼叫
- 最后ping时间和频率

Diameter对等状态

监控Diameter对等连接的Sh和Ro接口。

访问：在控制面板中导航到 `/diameter`

功能

- 对等状态：每个配置对等体的连接状态

- 应用支持：查看支持的Diameter应用程序（Sh、Ro）
- 看门狗状态： Diameter看门狗监控

系统日志查看器

实时统一日志查看器，用于TAS后端（Elixir）和TAS呼叫处理（FreeSWITCH）日志。

访问：在控制面板中导航到 `/logs`

功能

- 统一日志流：在一个界面中查看来自TAS后端和呼叫处理的日志
- 实时更新：日志消息实时流式传输（每1秒自动刷新）
- 颜色编码的日志级别：
  - □ 控制台 - 控制台消息（紫色/洋红色）
  - □ 警报/严重 - 需要立即关注的紧急问题（红色）
  - □ 错误 - 错误条件（浅红色）
  - □ 警告 - 警告消息（黄色）
  - □ 通知 - 显著的信息消息（青色）
  - □ 信息 - 一般信息消息（蓝色）
  - ○ 调试 - 调试/详细日志（灰色）
- 源徽章：
  - □ **TAS**后端 - Elixir应用程序日志（蓝色徽章）
  - □ **TAS**呼叫处理 - FreeSWITCH日志（紫色徽章）
- 左边框□示器：颜色编码的左边框与日志级别匹配，便于快速视觉扫描
- 多个过滤器：

    - 源过滤器：所有源 / TAS后端 / TAS呼叫处理
    - 级别过滤器：所有 / 控制台 / 警报 / 严重 / 错误 / 警告 / 通知 / 信息 / 调试
    - 文本搜索：在日志消息中实时关键字搜索
- 暂停/恢复：冻结日志流□分析特定条目而不失去上下文
- 清除日志：从显示中删除所有当前日志条目
- 日志计数器：显示过滤日志与总日志（例如，“显示150条，共500条日志”）
- 尾部行为：为了性能，保持最后500条日志条目
- 元数据展示：文件名和行号用于源代码引用（如果可用）
- 可滚动视图：固定高度容器，自动滚动□显示最新日志

如何使用

1. 基本查看：
    - 页面加载最新的500条来自两个源的日志条目
    - 日志实时生成时出现
    - 最近的日志出现在顶部
    - 每1秒自动刷新
2. 按源过滤：
    - 从 "源" 下拉菜单中选择：
        - 所有源 - 显示TAS后端和呼叫处理日志
        - **TAS**后端 - 仅Elixir应用程序日志
        - **TAS**呼叫处理 - 仅FreeSWITCH/拨号计划日��
    - 过滤立即应用
3. 按日志级别过滤：
    - 从 "级别" 下拉菜单中选择：
        - 所有 - 显示所有日志级别
        - 控制台 到 调试 - 仅显示该特定级别
    - 有助于专注于错误或调试特定问题
4. 搜索关键字：
    - 在 "搜索日志..." 框中输入

- 在日志消息中进行不区分大小写的搜索
- 输入时实时过滤
- 与源和级别过滤器结合

5. 暂停/恢复流：

- 点击 "暂停" 按钮（橙色）□冻结日志更新
- "已暂停" 指示器出现在标题中
- 在没有新日志干扰的情况下查看特定日志条目
- 点击 "恢复" 按钮（绿色）□重新启动实时流

6. 清除日志：

- 点击 "清除" 按钮（红色）□删除所有显示的日志
- 清除TAS后端和呼叫处理日志
- 新日志将在生成时出现

7. 阅读日志条目：

- 时间戳：□HH:MM:SS.毫秒格式显示时间
- 源徽章：指示TAS后端（蓝色）或呼叫处理（紫色）
- 日志级别：颜色编码的级别在方括号中 [ERROR]、[INFO] 等
- 文件/行：源代码位置（如果可用）
- 消息：实际日志消息内容

日志级别解释

级别	颜色	使用时机	示例
控制台	紫色	控制台特定消息	高优先级FreeSWITCH控制台输出
警报	红色	需要立即采取行动	系统组件故障
严重	红色	严重条件	数据库连接丢失
错误	浅红色	错误条件	无法处理呼叫，配置无效
警告	黄色	警告条件	使用了已弃用的功能，重试尝试
通知	青色	显著的正常事件	配置重新加载，服务启动
信息	蓝色	信息消息	呼叫连接，Diameter请求发送
调试	灰色	调试级别消息	函数进入/退出，变量值

用例

- 实时故障排除：在活动呼叫期间监控日志□调试问题
- 错误调查：过滤错误/严重级别□查找问题
- 呼叫流分析：搜索呼叫ID或电话号码□追踪呼叫路径
- 性能监控：在负载测试期间观察警告和错误
- 集成调试：过滤TAS后端□查看Diameter/Sh/Ro消息
- 拨号计划调试：过滤TAS呼叫处理□查看FreeSWITCH呼叫路由
- 系统健康监控：保持日志打开□观察异常
- 开发与测试：使用调试级别查看详细的应用程序行为

提示

- 组合过滤器：同时使用源 + 级别 + 搜索进行精确过滤
  - 示例：源="TAS后端" + 级别="错误" + 搜索="Diameter" → 查找Diameter错误
- 搜索前暂停：在输入搜索查询之前暂停流□避免日志滚动
- 明智使用调试：调试级别是详细的 - 过滤到特定源□减少噪音
- 颜色扫描：快速扫描左边框 - 红色边框表示问题

- 源徽章：蓝色徽章（后端）用于应用逻辑，紫色徽章（呼叫处理）用于呼叫
- 时间戳精度：毫秒时间戳有助于跨系统关联事件
- 文件引用：点击/记录文件:行引用□跳转到源代码
- 定期清除：在切换调查上下文时清除日志□保持清晰
- 搜索**UUID**：搜索呼叫ID/UUID□跟踪特定呼叫通过整个系统
- 紧急搜索：搜索 "911" 或 "紧急" □快速查找紧急呼叫处理

技术细节

- 日志限制：显示最大500条日志（达到限制时丢弃最旧的）
- 刷新率：每1000毫秒（1秒）自动刷新
- 搜索：仅在消息字段上进行不区分大小写的子字符串匹配
- 空过滤：自动过滤掉空/占位符日志消息
- 源检测：日志标记为 `:elixir` 或 `:freeswitch` 源
- 排序：日志按时间戳降序排序（最新的在前）
- **PubSub**：Elixir日志通过Phoenix PubSub实时更新
- **FreeSWITCH**日��：通过事件套接字接口（ESI）日志监听器收集

呼叫模拟器

交互式呼叫模拟工具，用于测试拨号计划逻辑而无需进行真实呼叫。

访问：在控制面板中导航到 `/simulator`

详细文档：请参见 HLR和呼叫模拟器指南

功能

- 模拟呼叫类型：测试MO、MT和紧急呼叫
- 可配置参数：
  - 源和目的地号码
  - 源IP地址（模拟SBC/CSCF）
  - 强制特定呼叫处置
  - 跳过OCS授权□加快测试
- 全面结果：

    - 完整的拨号计划变量输出
    - Sh/HLR查找结果
    - OCS授权结果
    - SS7 MAP查询结果（如适用）
    - 生成的拨号计划XML
- 逐步处理：查看呼叫处理的每个阶段

用例

- 在部署之前测试拨号计划更改
- 验证用户订阅
- 调试呼叫路由问题
- 培训员工了解呼叫流
- 验证OCS/HLR集成
- 测试紧急呼叫处理

HLR/MAP测试

测试SS7 MAP操作，包括发送路由信息（SRI）和提供漫游号码（PRN）查询。

访问：在控制面板中导航到 `/hlr`

详细文档：请参见 HLR和呼叫模拟器指南

功能

- **SRI**查询：测试发送路由信息□进行呼叫路由
- **PRN**查询：测试提供漫游号码□供漫游用户使用
- 实时结果：实际查询配置的MAP网关
- 响应显示：查看MSRN、MSC地址和转发状态
- 错误处理：清晰显示MAP错误和超时

用例

- 验证HLR连接
- 测试漫游号码分配

- 调试对漫游用户的呼叫路由
- 验证MAP网关配置
- 排查呼叫转移问题

OCS测试

直接针对您的OCS测试Diameter Ro（在线计费）信用控制请求（CCR）操作。

访问：在控制面板中导航到 `/ocs_test`

功能

- 灵活的**CCR**类型：发送INITIAL、UPDATE、TERMINATION或EVENT请求
- 会话模拟：重用相同的呼叫ID□模拟完整的会话生命周期
- 事件类型选择：测试SMS（基于事件）和呼叫（基于会话）计费
- 方向控制：测试出站（MO）和入站（MT）场景
- 可选参数：指定目标主机和用户名□进行高级测试
- 实时结果：查看完整的CCA（信用控制应答）响应

如何使用

1. 输入测试参数：
    - 被叫**MSISDN**：目的地号码（例如，`61400123456`）
    - 主叫**MSISDN**：源号码（例如，`61400987654`）
    - 事件类型：选择 `sms` 或 `call`
        - SMS默认使用EVENT_REQUEST（类型4）

        - 呼叫默认使用INITIAL_REQUEST（类型1）
    - 方向：`out` 表示MO或 `in` 表示MT

2. 配置**CCR**类型：

    - 请求类型：选择CCR类型：
        - `1`: INITIAL_REQUEST - 启动新会话
        - `2`: UPDATE_REQUEST - 会话中重新授权
        - `3`: TERMINATION_REQUEST - 结束会话并报告使用情况
        - `4`: EVENT_REQUEST - 一次性事件（SMS，立即事件）
    - 请求编号：从1开始，在同一会话中递增

3. 会话测试：

    - 呼叫**ID**：自动生成的唯一标识符□进行关联
    - 点击 **"新ID"** □生成新的呼叫ID进行新的测试
    - 保持相同的呼叫**ID** □模拟完整的会话：
        - 第一个请求：INITIAL_REQUEST（类型1，编号1）
        - 会话中：UPDATE_REQUEST（类型2，编号2、3、4...）
        - 最后请求：TERMINATION_REQUEST（类型3，编号N+1）

4. 高级选项：

    - 目标主机：针对特定OCS节点（可选）
    - 用户名：重写订阅者标识符（可选）

5. 运行并查看：

    - 点击 **"运行CCR"** 发送请求
    - 查看完整的CCA响应及所有AVP
    - 检查结果代码、授予的单位和有效时间
    - 最后运行时间戳显示在右上角

用例

- **OCS**连接测试：验证Diameter Ro连接和身份验证
- 信用控制逻辑：测试信用分配、消费和耗尽场景
- 会话流测试：模拟完整的呼叫生命周期（INITIAL → UPDATE → TERMINATION）

- 计费验证：验证不同号码范围的正确计费率
- 故障转移测试：通过针对特定目标主机测试OCS冗余
- 集成调试：通过详细的AVP检查排查OCS集成问题
- 负载测试准备：在负载测试之前验证OCS行为
- 紧急号码绕过：验证紧急号码是否正确绕过计费

提示

- 使用相同的呼叫ID和递增的请求编号测试会话连续性
- 同时监控OCS日志□关联测试请求
- 测试UPDATE请求□验证会话中重新授权逻辑
- 验证TERMINATION请求是否正确关闭会话并防止泄漏
- 通过在消耗授予的单位后发送UPDATE请求来测试信用耗尽

Sh接口测试

测试Diameter Sh用户数据请求（UDR）操作，□从HSS检索用户配置文件数据。

访问：在控制面板中导航到 `/sh_test`

功能

- 多个数据引用：查询超过20种不同的用户数据类型
- 实时**HSS**查询：对配置的HSS进行实时Diameter Sh请求
- 完整响应显示：查看完整的XML用户数据和AVP
- 会话跟踪：显示HSS主机名、领域和会话ID
- 错误处理：清晰显示Diameter结果代码和错误条件

如何使用

1. 输入公共身份：
    - 公共身份：订阅者的IMS公共身份
    - 格式：`sip:61400123456@ims.mncXXX.mccXXX.3gppnetwork.org`
    - 也可使用 `tel:+61400123456` 格式
2. 选择数据引用：选择要检索的用户数据类型：
    - **RepositoryData (0)**：完整的用户配置文件
    - **IMSPublicIdentity (10)**：公共身份列表
    - **IMSUserState (11)**：注册状态
    - **S-CSCFName (12)**：分配的S-CSCF
    - **InitialFilterCriteria (13)**：应用服务器的iFC触发器
    - **LocationInformation (14)**：当前位置信息
    - **ChargingInformation (16)**：P-计费地址
    - **MSISDN (17)**：电话号码
    - **IMSI (32)**：国际移动用户身份
    - **IMSPrivateUserIdentity (33)**：私有用户身份
    - ���有更多...
3. 运行并查看：
    - 点击 "获取**SH**数据" 发送UDR请求
    - 查看完整的用户数据应答（UDA）响应
    - 检查用户配置文件XML、服务数据和iFC规则
    - 会话元数据显示哪个HSS响应

用例

- 用户验证：确认用户在HSS中已配置
- **iFC**调试：查看初始过滤标准和触发点
- 注册故障排除：检查用户状态和S-CSCF分配
- 计费配置：验证P-计费功能地址
- **HSS**连接测试：验证Diameter Sh连接
- 配置文件验证：确保分配了正确的服务配置文件

- 集成测试：在配置更改后测试HSS集成
- 漫游分析：检查位置信息和服务网络

提示

- 使用 **IMSPublicIdentity (10)** 查看用户的所有别名
- 使用 **RepositoryData (0)** 一次查询获取完整的用户配置文件
- 检查 **IMSUserState (11)** 验证用户是否已注册
- **InitialFilterCriteria (13)** 显示将触发的应用服务器
- 会话ID可用于关联HSS日志中的查询
- 错误响应包括Diameter结果代码（例如，5001 = 用户未知）

号码翻译测试

测试号码翻译规则和格式，而无需进行实际呼叫。

访问：在控制面板中导航到 `/translate`

功能

- 实时翻译：输入时自动翻译
- 国家代码支持：测试不同国家代码上下文
- 处置感知：根据呼叫处置应用不同规则
- 实时结果：立即反馈翻译后的号码
- 调试信息：查看原始返回值进行故障排除

如何使用

1. 配置参数：

    - 国家代码：拨号上下文（例如，`AU`、`US`、`NZ`）
        - 默认使用 `config/runtime.exs` 中配置的国家代码
        - 接受格式：`AU`、`:AU`、`au`
    - 电话号码：要翻译的号码
        - 示例：`+61400111222`、`0400111222`、`61400111222`
    - 处置：（可选）呼叫上下文□进行条件规则
        - 示例：`originate`、`route`、`emergency`

2. 测试翻译：

    - 在表单中输入值
    - 输入时自动运行翻译
    - 或点击 "翻译" 手动触发
    - 立即查看翻译结果

3. 查看结果：

    - 翻译：显示格式化后的输出号码
    - 错误：显示验证错误或翻译失败
    - 原始返回（调试）：显示完整的Elixir元组□进行调试

用例

- 拨号计划开发：在部署之前测试号码格式规则
- 格式验证：验证E.164转换是否正常工作
- 国家代码测试：确保正确处理国��前缀
- 紧急号码检测：验证紧急号码是否被正确识别
- 短码处理：测试特殊服务代码（语音邮件等）
- 中继准备：正确格式化号码□满足SIP中继要求
- 处置逻辑：测试MO与MT场景的不同规则
- 调试翻译问题：排查特定号码为何无法路由

提示

- 测试本地格式（`0400111222`）和国际格式（`+61400111222`）

- 验证紧急号码（`000`、`112`）是否被正确检测
- 使用处置字段测试不同呼叫场景（MO、MT、紧急）
- 检查短码和内部号码是否被适当处理
- 调试输出显示原始返回值 - 有助于调查问题
- 测试边缘情况，如前导零、国际前缀和特殊字符

语音邮件管理

管理和收听存储在系统中的语音邮件消息。

访问：在控制面板中导航到 `/voicemail`

功能

- 完整语音邮件列表：查看所有邮箱中的语音邮件消息
- 浏览器内播放：直接在网页界面中收听语音邮件录音
- 消息详情：查看用户名、UUID、时间戳、文件路径和元数据
- 删除功能：删除单个语音邮件消息
- 自动刷新：刷新按钮重新加载最新的语音邮件数据
- 动态列：自动显示所有可用的数据库字段

如何使用

1. 查看语音邮件列表：
    - 页面自动加载所有语音邮件记录
    - 表格显示来自语音邮件数据库的所有字段
    - 时间戳自动从纪元值格式化
    - 文件路径缩短便于阅读

2. 收听消息：

    - 点击任何语音邮件旁的 "▶ 播放" 按钮
    - 音频播放器出现，带有控制（播放、暂停、快进、音量）
    - 支持WAV、MP3和OGG格式
    - 点击 "停止" 关闭音频播放器

3. 删除消息：

    - 点击 "删除" 按钮□移除语音邮件
    - 确认提示防止意外删除
    - 成功删除后页面自动刷新

4. 刷新数据：

    - 点击右上角的 "刷新" 按钮□重新加载语音邮件列表
    - 在留下新语音邮件后非常有用

显示的消息详情

表格动态显示所有可用字段，通常包括：

- 用户名：邮箱所有者
- **UUID**：唯一消息标识符
- 创建纪元：留下消息的时间（自动格式化为可读日期/时间）
- 读取纪元：访问消息的时间（如果适用）
- 文件路径：音频文件的位置
- 来自语音邮件数据库的其他元数据

用例

- 用户支持：收听语音邮件消息□进行故障排除
- 测试语音邮件��递：验证语音邮件是否被正确存储
- 消息管理：清理旧的或测试的语音邮件消息
- 故障排除录音问题：检查文件路径并验证音频文件是否存在
- 邮箱维护：监控语音邮件存储和使用情况
- 质量保证：审查录制的消息□确保音频质量

提示

- 文件路径自动缩短□仅显示相关部分
- 纪元时间戳自动转换为人类可读格式
- 空的语音邮件数据库显示 "未找到语音邮件记录"
- 音频播放使用HTML5音频元素 - 在所有现代浏览器中支持
- 删除确认防止意外删除重要消息

TTS提示管理

管理系统中使用的文本到语音（TTS）生成的音频提示。

访问：在控制面板中导航到 `/prompts`

功能

- 提示设置显示：查看当前TTS语音、响应格式和指令
- 录音状态：查看哪些提示存在，哪些缺失
- 文件详情：查看每个提示的文件大小、修改时间和路径
- 浏览器内播放：直接在网页界面中收听提示
- 生成缺失：自动创建所有缺失的提示文件
- 单独重新录制：使用更新的设置重新生成特定提示
- 全部重新录制：重新生成所有提示（在更改���音或设置后非常有用）

如何使用

1. 查看提示设置：

- 语音：正在使用的TTS语音（例如，`alloy`、`nova`、`shimmer`）
- 响应格式：音频格式（例如，`wav`、`mp3`、`opus`）
- □令：传递给TTS引擎的特殊指令

2. 检查录音状态：

   - 文本：要说出的提示文本
   - 相对路径：音频文件存储的位置
   - 存在：如果文件存在则为绿色 "是"，如果缺失则为黄色 "否"
   - 大小：文件大小（□字节/KiB/MiB为单位）
   - 修改：最后修改时间戳

3. 生成提示：

   - 生成缺失：仅创建尚不存在的提示
     - 对于初始设置或添加新提示后非常有用
   - 全部重新录制：无论存在与否，重新生成所有提示
     - 在更改语音、格式或指令后非常有用
     - 使用时请谨慎，因为它会重新生成所有内容

4. 管理单个提示：

   - ▶ 播放：收听提示（仅在文件存在时启用）
   - □ 重新录制：仅重新生成此一个提示
     - 如果某个提示听起来不正确非常有用
     - 使用当前语音和设置

5. 收听提示：

   - 点击 "▶ 播放" 收听提示
   - 音频播放器在底部出现，带有完整控制
   - 点击 "停止" 关闭播放器

提示配置

提示在您的应用程序配置中进行配置：

```
config :tas, :prompts,
  voice: "nova",
  response_format: "wav",
  instructions: "清晰且专业地说话。",
  recordings: [
    %{path: "/sounds/en/us/callie/voicemail/vm-enter_id.wav",
      text: "请输入您的邮箱ID，后面跟上井号"},
    # ... 更多提示
  ]
```

用例

- 初始设置：在系统安装后生成所有提示
- 语音更改：使用不同的TTS语音重新录制所有提示
- 质量改进：修复听起来不正确的单个提示
- 格式更新：□不同的音频格式重新生成提示（wav → mp3）
- 文本更新：在配置中更改提示文本后重新录制
- 测试**TTS**：在部署之前预览提示的声音
- 故障排除播放：验证提示文件是否存在且可访问
- 存储管理：检查文件大小并管理磁盘使用情况

提示

- 使用 "生成缺失" 进行初始设置 - 它不会覆盖现有提示
- 在配置中更改语音或格式后使用 "全部重新录制"
- 单独的 "重新录制" 对于特定提示的迭代非常有用
- 在部署之前收听提示□确保质量
- 较大的响应格式（wav）质量更好，但占用更多磁盘空间
- 指��字段可□指导TTS引擎的语调和节奏
- 重新录制可能需要时间，如果您有很多提示，请耐心等待
- 提示存储在FreeSWITCH声音目录中，便于访问

拨号计划**XML**模板

查看和检查用于呼叫路由的FreeSWITCH拨号计划XML模板。

访问：在控制面板中导航到 `/routing`

功能

- 模板列表：查看来自 `priv/templates/` 目录的所有XML拨号计划模板
- 文件详情：查看每个模板的文件名和最后修改时间戳
- 语法高亮：颜色编码的XML显示，便于阅读
  - 标签为青色
  - 属性为浅蓝色
  - 值为橙色/棕色
  - 注释为绿色
- 可展开视图：点击任何模板查看其完整的XML内容
- 只读视图：安全检查而不冒险意外修改
- 可滚动内容：大模板在固定高度容器内滚动（最大600px）

如何使用

1. 查看模板列表：
   - 页面加载来自模板目录的所有 `.xml` 文件
   - 按文件名字母顺序排序
   - 显示每个文件的修改时间戳
2. 检查模板：
   - 点击任何行展开并查看XML内容
   - 模板语法高亮显示
   - 再次点击折叠

3. 阅读**XML**内容：

    - 标签（青色）：XML元素名称，如 `<extension>`、`<condition>`
    - 属性（浅蓝色）：属性名称，如 `name=`、`field=`
    - 值（橙色）：属性值，如 `"public"`、`"destination_number"`
    - 注释（绿色）：XML注释 `<!-- ... -->`

用例

- 审查拨号计划逻辑：检查路由规则和呼叫流模板
- 故障排除呼叫路由：理解不同呼叫类型使用的模板
- 验证模板语法：在部署之前检查XML结构
- 培训与文档：与团队成员共享模板内容
- 更改审计：比较修改时间戳跟踪更新
- 模板开发：在创建新模板时参考现有模板

提示

- 模板从TAS应用程序中的 `priv/templates/` 加载
- 仅显示 `.xml` 文件
- 通过Web界面只读方式查看模板
- 修改时间戳有助于识别最近的更改
- 使用此视图验证模板是否符合您的拨号计划预期
- 语法高亮使复杂的XML更易于视觉解析
- 与 `/logs` 视图结合使用，将路由行为与模板关联

技术细节

- 位置：模板存储在 `priv/templates/` 目录中
- 格式：FreeSWITCH XML拨号计划格式
- 文件扩展名：仅列出 `.xml` 文件
- 排序：按文件名字母顺序排序
- 语法高亮：使用正则表达式模式进行客户端侧着色
- 最大显示高度：600px，超大文件滚动

ESL命令运行器

直接从Web界面执行FreeSWITCH事件套接字层（ESL）命令。

访问：在控制面板中导航到 `/command`

功能

- 命令执行：运行任何ESL/FreeSWITCH API命令
- 实时输出：实时查看命令结果
- 命令历史：最近命令下拉菜单（最后10个命令）
- 自动完成准备：单宽输入□精确输入命令
- 错误处理：清晰显示命令错误和异常
- 无自动执行：选择历史填充输入，但需要明确点击 "运行"

如何使用

1. 输入命令：
    - 在输入框中输入ESL命令
    - 示例：
        - `status` - 显示FreeSWITCH状态
        - `show channels` - 列出活动呼叫
        - `uuid_dump <uuid>` - 转储呼叫的所有变量
        - `sofia status` - 显示SIP配置文件状态
        - `reloadxml` - 重新加载XML拨号计划
        - `version` - 显示FreeSWITCH版本

2. 运行命令:

   - 点击 "运行" 按钮□执行
   - 按钮在执行时显示 "正在运行..."
   - 不能同时运行多个命令

3. 查看输出:

   - 结果出现在下面的 "输出" 部分
   - 成功的命令显示原始响应
   - 错误□ "错误: " 前缀显示
   - 输出可滚动，最大高度为600px
   - 单宽字体□对齐数据

4. 使用命令历史:

   - 最近的命令在第一次执行后出现在下拉菜单中
   - 从 "最近: " 下拉菜单中选择□填充输入框
   - 历史保持最后10个唯一命令
   - 最近的命令在顶部
   - 选择历史不会自动执行（安全功能）

常用命令

命令	描述	示例输出
`status`	系统状态和正常运行时间	FreeSWITCH运行信息
`show channels`	列出所有活动呼叫	呼叫列表或 "0 总计"
`show calls`	活动呼叫的摘要	呼叫计数摘要
`uuid_dump <uuid>`	呼叫的所有变量	完整变量转储
`uuid_kill <uuid>`	挂断特定呼叫	"+OK" 或错误
`sofia status`	SIP配置文件状态	配置文件列表和状态
`sofia status profile <name>`	特定配置文件详情	注册计数等
`reloadxml`	重新加载拨号计划XML	"+OK" 确认
`version`	FreeSWITCH版本信息	版本字符串
`global_getvar <var>`	获取全局变量	变量值
`api help`	列出可用命令	命令参考

用例

- 呼叫调试：使用 `uuid_dump` 获取活动呼叫的详细信息
- 系统状态：使用 `status` 和 `show calls` 检查FreeSWITCH健康状况
- **SIP**故障排除：使用 `sofia status` 检查SIP配置文件
- 拨号计划重新加载：使用 `reloadxml` 应用配置更改
- 紧急操作：使用 `uuid_kill` 挂断卡住的呼叫
- 变量检查：检查全局或通道变量

故障排除

用户未显示

- 验证OmniTAS是否正在运行
- 检查Sofia配置文件是否处于活动状态：`sofia status profile internal`
- 验证配置中的数据库路径是否与实际数据库位置匹配

CDR记录未出现

- 确认OmniTAS CDR模块已加载
- 检查CDR数据库是否存在于配置路径
- 验证OmniTAS中的CDR模块配置

性能考虑

- 大型CDR数据库（>1M记录）可能需要额外索引□获得最佳性能
- 考虑定期归档旧的CDR记录
- 用户注册查询通常很快，因为注册数据库很小

配置

访问控制

控制面板应在适当的访问控制（防火墙、VPN、身份验证）后部署，因为它提供了对用户活动和呼叫记录的可见性。

文本转语音（TTS）提示

 返回主文档

OmniTAS 播放预生成的提示音频（IVR 提示和语音邮件的日期/时间）。OmniTAS 不生成此音频。提示生成是在 Omnicore Ansible 剧本中的部署时步骤。

生成发生的地方

音频由 Omnicore Ansible 仓库生成，而不是由 OmniTAS 生成：

- 每个客户的配置（`group_vars/tts.yml`）声明了引擎、声音和提示文本。
- 生成脚本（`scripts/generate_tts.py`）合成每个提示和 `mod_say` 日期/时间词集，使用配置的引擎（Azure、OpenAI 或 ElevenLabs），规范化日期/时间词的静音，并将 WAV 文件写入客户的 `group_vars/recordings/` 目录。
- `applicationserver` 角色将 `recordings/` 部署到 FreeSWITCH 声音目录（`/usr/local/freeswitch/sounds/`）。

请参阅 Omnicore 文档（`docs/tts_recordings.md`）获取完整的工作流程和配置参考。

为什么生成是外部的

- 盒子上没有 **API** 密钥。 TTS 引擎密钥仅在生成时使用，来自 Ansible ，它从不存储在 OmniTAS 配置中。
- 可审查的工件。生成的 WAV 文件按客户提交并按原样部署，因此每个盒子播放相同的已知音频。
- 音频后处理属于构建时。日期/时间词被规范化（固定的前导和尾随间隙），便连接的日期流畅播放。这是在生成器中完成的。

OmniTAS 仍然控制的内容

OmniTAS 使用 FreeSWITCH `say` 引擎播放存储的语音邮件日期和时间。它使用 相对 格式（`SHORT_DATE_TIME`），因此消息会说“今天 14:30”，“昨天 …”或最近消息的星期几。只有当消息来自不同的日历年份时，才会说出完整的日期和年份。这使得公告简短。

要更改提示声音或措辞，请在 Omnicore Ansible 仓库中重新生成录音并重新部署。无需更改 OmniTAS。

MMTel / ODB Sh 存储库数据架构（参考）

权威的 XML 架构用于 OmniTAS 通过 Sh 接口读取的透明（RepositoryData）文档。真相来源：**3GPP TS 29.364 "IMS 应用服务器（AS）服务数据描述用于 AS 互操作性"**，V18.0.0（2024-05），发布为 ETSI TS 129 364。完整规范 PDF 在此处包含□证明来源。

这些仅供参考 — OmniTAS 将 RepositoryData 视为不透明，并不对其进行验证（根据 TS 29.328，HSS 按字节存储 `ServiceData`）。

文件

文件	**TS 29.364** 条款	根元素	服[
`MMTelServices-root.xsd`	§4 / 附录	`<MMTelServices>`	`MMTEL Servi`
`IMS-ODB-Information.xsd`	§10.2	`<OdbForImsOrientedServices>`	`IMS-O Infor`
`TS_129364_v18.0.0.pdf`	—	—	（完整规□

`MMTelServices-root.xsd` 是顶级架构；在规范中，它 `xs:include`s 每个服务的子架构（通信转移、通信限制、运营商-*、身份呈现等） — 请参阅 PDF 附录□获取这些文件。

两种合法形式 — 不要混淆它们

MMTel 附加服务文档存在于 两个规范定义的形状，用于两个不同的接口：

- **`<MMTelServices>`** — **TS 29.364**，作为 RepositoryData 存储在 HSS 中并由 AS 读取的 **Sh** 透明数据 形式。使用 `complete-*` 和 `operator-*` 包装元素。这是 **OmniTAS** 在 **Sh** 上看到的形式，也是

HSS 存储的形式。

- `<simservs>` — ETSI TS 183 023 / 3GPP TS 24.623，**XCAP-over-Ut** 形式，订阅者/UE 直接操作（这是 OmniSEP EntitlementServer 在 Ut 上提供的内容）。相同的 `simservs` 命名空间和通用策略规则集，根不同且没有 `complete-*`/`operator-*` 包装。

两者都符合标准；正确的形式取决于接口。对于 Sh / HSS RepositoryData，请使用 `<MMTelServices>`。

标准化服务示值（TS 29.364）

- `MMTEL-Services` — MMTel 附加服务（`<MMTelServices>`）
- `IMS-ODB-Information` — 运营商确定的限制（§10）
- `IMS-CAMEL-Services` — IMS CAMEL 订阅数据（§11）

专有文档（例如运营商特定的 `E911` / `PBX` / `Customer-Address` 块）必须使用非标准化的服务指示，且其主体没有限制 — 同样根据 TS 29.364 §10.1。

命名空间说明（ODB）

TS 29.364 ODB 架构声明 没有目标命空间；其根是一个裸 `<OdbForImsOrientedServices>`。供应商部署（OpenCloud Sentinel VoLTE / Omnitouch）将其包装在一个供应商命名空间中，例如 `xmlns="odb.mmtel.omnitou.ch"` — 该命名空间是供应商添加的，不是 3GPP 架构的一部分。

仓库数据块

 返回主文档

本指南解释了 OmniTAS 如何将 Sh **RepositoryData** 文档作为每个文档的通道变量（“块”）暴露给 FreeSWITCH 拨号计划。它涵盖了如何请求文档，OmniTAS 如何将每个文档转换为变量，及拨号计划如何安全地读取它们。

相关文档

- **Sh** 接口 - 如何从 HSS 获取订阅者数据
- ☎ 拨号计划配置 - 拨号计划如何使用通道变量
- ⚙ 补充服务 - 呼叫转移和阻止行为
- **SS7 MAP** - `call_forward_*` 变量的 HLR 来源

目录

- 概念
- 架构
- 获取文档
- 从文档到变量
  - 消息流
  - 扁平化规则
  - 变量家族
  - 名称作用域
  - 活动和存在标志
- 在拨号计划中读取数据
- 示例块
- 限制
- 参考表

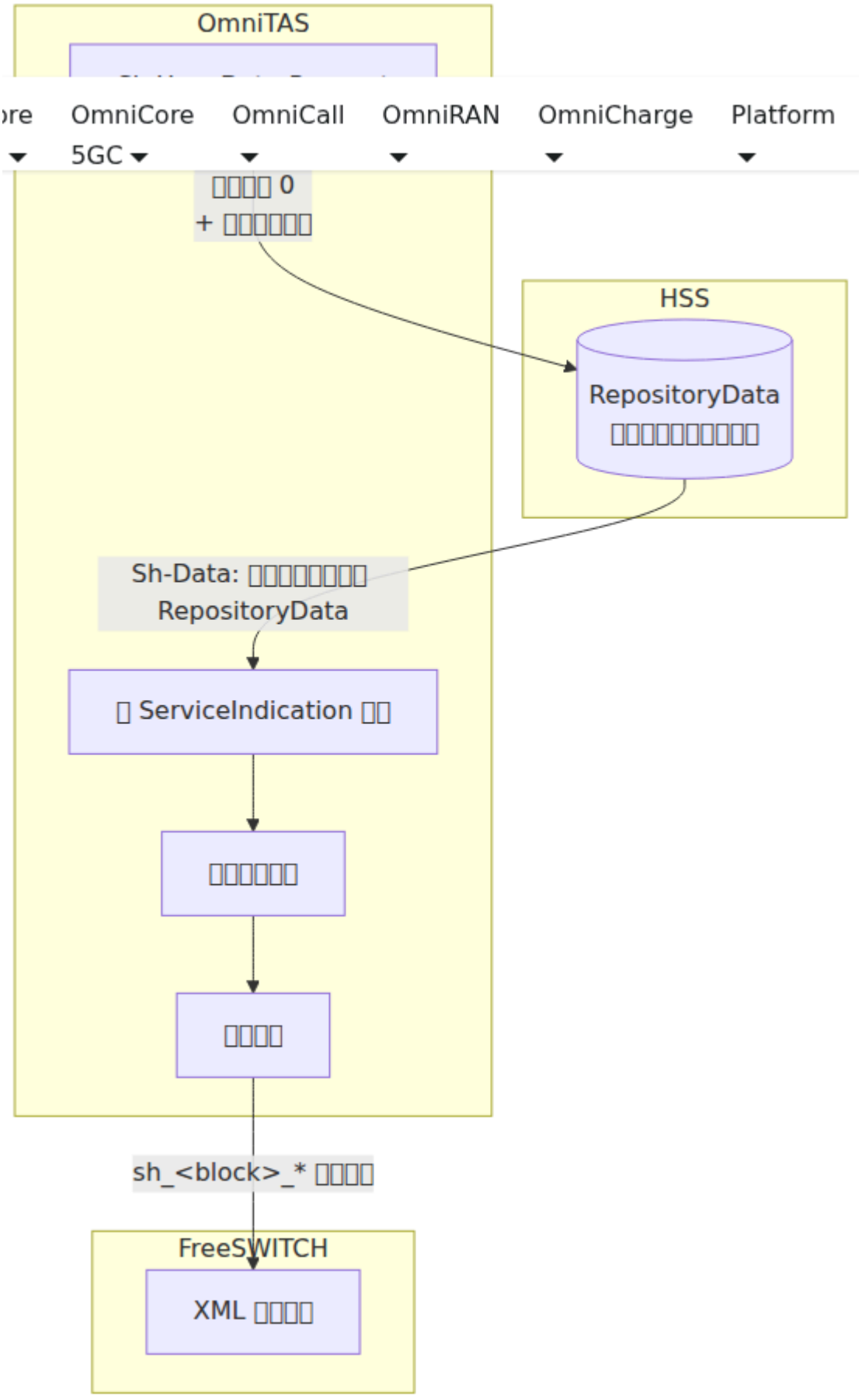
概念

HSS 中的订阅者配置文件可包含任意数量的 透明文档。每个文档都是 Sh RepositoryData，并且每个文档由 服务示 命名。`MMTEL-Services` 和 `IMS-ODB-Information` 是两个标准示例。运营商还可配置私有文档，例如 `E911` 或 `PBX`。

“透明”在 3GPP TS 29.328 中有明确的含义。HSS 存储并返回文档主体 逐字节。HSS 不解释主体。

OmniTAS 遵循相同的规则。OmniTAS 不 将任何 RepositoryData 主体解析为固定字段。OmniTAS 将 Sh 响应拆分为每个文档一个 块，并将每个块作为通道变量暴露给拨号计划。拨号计划读取变量并决定该做什么。这使得 OmniTAS 保持通用：运营商可添加新的文档类型而无需更改代码。

> 共享的呼叫转移变量是分开的。`call_forward_all_destination`、`call_forward_not_reachable_destination` 和 `no_reply_timer` 不是 从 MMTel 生成的。它们来自 HLR/SS7（见 SS7 MAP）和配置默认值。



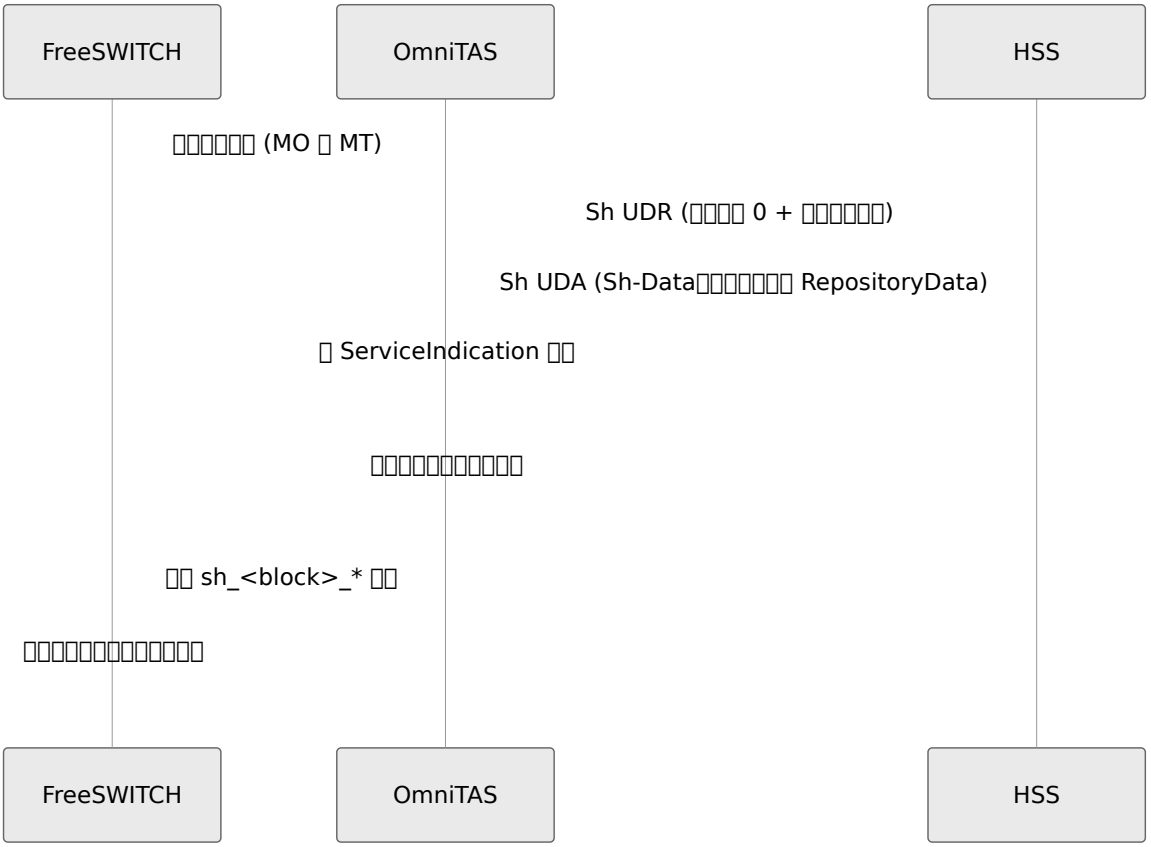
OmniTAS 每个呼叫方发送一个 Sh User-Data-Request。请求携带一个 `Data-Reference` 值 `0` 和每个文档的一个 `Service-Indication` 。HSS 返回一个 `Sh-Data` 文档，该文档包含每个请求文档一个

`RepositoryData` □□□OmniTAS □ Service-Indication □□□□

□□□ RepositoryData □□□□□□□□□S-CSCF□□□□□□□□□ Sh □□□

□□□□□□

□□□



□□□□

OmniTAS □□□□□□□□□□□□□□□□

两个规则使输出稳定且完整：

- 按身份作用域，而不是位置。当容器携带 `id` 或 `name` 时，OmniTAS 将该值折叠到每个后代的键中。请参见 名称作用域。
- 捕获状态，而不仅仅是数据。 OmniTAS 捕获容器属性和作用域内的空元素，��便拨号计划可□判断服务或规则是否处于活动状态。请参见 活动和存在标志。

变量家族

OmniTAS 从响应中设置三类变量。

每块变量（每个文档一组，前缀为块名称）：

变量	内容
`sh_<block>_xml_b64`	块的 `ServiceData` XML，Base64 编码。解码此内容□读取扁平化变[何内容。
`sh_<block>_<key>`	叶子元素的文本，或作用域存在标志的 `true`。键包括祖先作用域。例如： `sh_mmtel_services_busy_target`。
`sh_<block>_<key>_<attr>`	叶子或容器上属性的值。例如： `sh_mmtel_services_communication_diversion_a`

块作用域防止冲突。`E911` 文档中的 `Number` 元素变为 `sh_e911_Number`。`PBX` 文档中的 `Number` 元素变为 `sh_pbx_Number`。这两个永远不会相互覆盖。

全局扁平变量（所有文档中每个叶子，一个命名空间）：

变量	内容
`sh_data_<key>`	叶子元素的文本（或存在标志）。
`sh_data_<key>_<attr>`	属性的值。
`sh_raw_xml_b64`	完整合并的 `Sh-Data` 主体，Base64 编码。

全局家族方便但在订阅者持有多个文档时会产生歧义。优先使用每块变量。

名称作用域

OmniTAS 从其祖先的身份构建变量名称，而不是从位置。当容器携带 `id` 或 `name` 属性时，OmniTAS 将该值折叠到每个后代的键中。OmniTAS 优先使用 `id` 而不是 `name`。

MMTel 的 `<cp:rule id="busy">` 和 `<target>` 元素生成 `sh_<block>_busy_target`。一个兄弟 `<cp:rule id="noanswer">` 生成 `sh_<block>_noanswer_target`。拨号计划通过名称读取每个条件，而不依赖于文档顺序。

名称遵循配置系统分配的规则 id。OmniHSS 和 Metaswitch 使用 `busy`、`noanswer`、`notreachable` 和 `unregistered`。OmniSEP 使用 `cfb`、`cfna`、`cfnrc`、`cfnl` 和 `cfu`。在编写拨号计划逻辑之前，请确认您

的配置系统的 id 约定。

OmniTAS 还将这些规则应用于每个名称和值：

- 名称仅保留 `A-Z`、`a-z`、`0-9` 和 `_`。命名空间前缀或连字符变为下划线。
- OmniTAS 使用本地名称，因此键在 HSS 命名空间选择中是稳定的。
- OmniTAS 将值中的每个空格运行压缩为一个空格。
- 如果两个叶子仍然产生相同的键，OmniTAS 会在文档顺序中添加位置后缀：`sh_<block>_busy_target`、`sh_<block>_busy_target_2`。

活动和存在标志

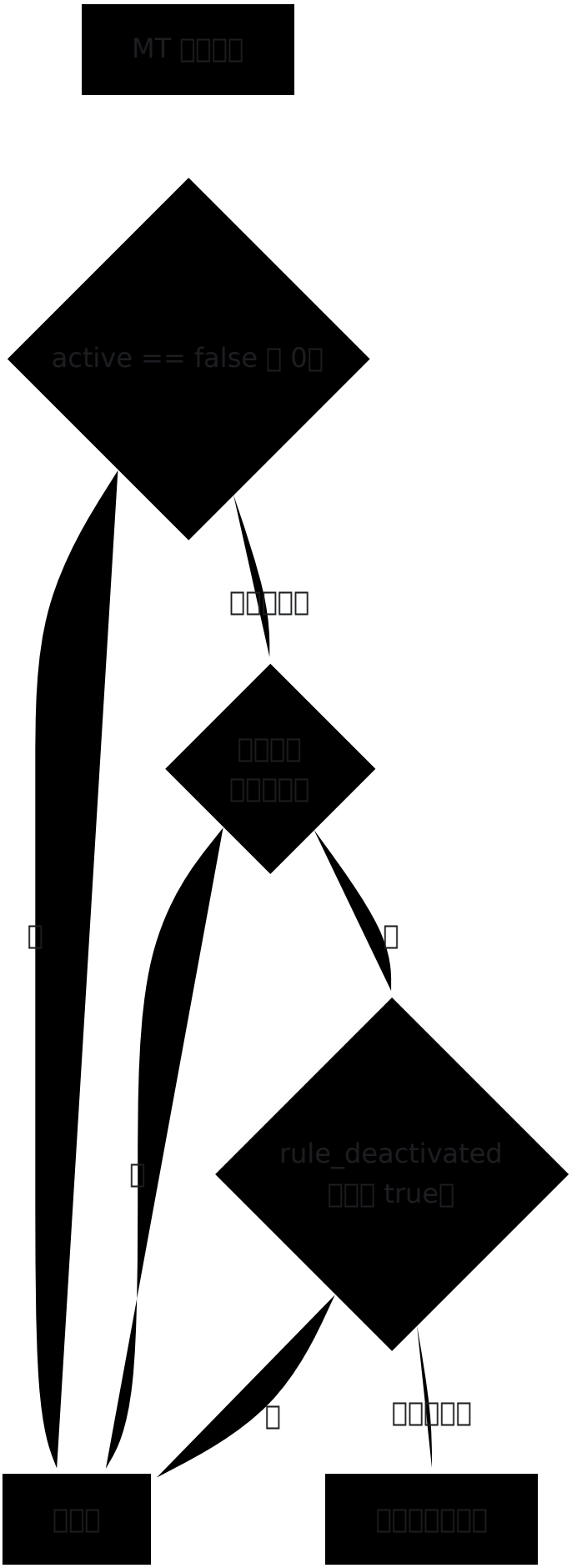
服务并不是仅仅因为规则存在就“开启”。OmniTAS 提供两个信号，拨号计划可先检查状态。

- 服务启用标志。 OmniTAS 捕获容器的 `active`（或 `authorized`）属性。`<communication-diversion active="true">` 变为 `sh_<block>_communication_diversion_active=true`。
- 存在标志。在 id/name 作用域内的空元素变为 `true` 标志。`<rule-deactivated/>` 标记在 `<cp:rule id="noanswer">` 下变为 `sh_<block>_noanswer_rule_deactivated=true`。像 `<no-answer/>` 这样的条件变为 `sh_<block>_noanswer_no_answer=true`。

> 缺失意味着活动。在 MMTel 中，当缺失时，`active` 属性默认为 `true`（3GPP TS 24.623 §6.1）。OmniTAS 仅在 **HSS** 发送属性时 设置 `..._active` 变量。因此有三种状态：设置为 true、设置为 false 和未设置（缺失 = 活动）。除非值为布尔 false，否则将服务视为活动。不要 测试 `^true$`：默认活动的服务省略该属性，不设置变量，并会失败该测试。该属性是 `xs:boolean`，因此 false 值 `false` 或 `0` 形式到达。`rule-deactivated` 标志仅在规则关闭时存在，因此缺失 = 活动。

在拨号计划中读取数据

在对目标采取行动之前应用状态检查。下图显示了终止（MT）呼叫转移决策的顺序。



MT □□□□
active == false □ 0□
否或未设置
□□□□
□□□□□
□
是
rule_deactivated
□□□ true□
□
是
否或□设置
□□□
□□□□□□□□

等效的拨号计划扩展 - 除非转移关闭或规则被停用，否则尊重忙碌目标：

```
<extension name="MMTel-CFB" continue="true">
  <!-- 除非明确为 false 或 0，否则处于活动状态。缺失属性不设置变量，这意味着活动。 -->
  <condition
field="${sh_mmtel_services_communication_diversion_active}"
expression="^(?!(false|0)$).*"/>
  <!-- 规则停用标志仅在规则关闭时设置；缺失为活动。 -->
  <condition field="${sh_mmtel_services_busy_rule_deactivated}"
expression="^(?!true$).*"/>
  <condition field="${sh_mmtel_services_busy_target}"
expression="^(?!$).+">
    <action application="log" data="INFO CFB to
${sh_mmtel_services_busy_target}"/>
  </condition>
</extension>
```

示例块

每个示例显示 HSS 存储的 `ServiceData` 主体，□及 OmniTAS 设置的变量。权威模式在 `docs/reference/mmtel-schemas/` 中。

MMTEL-Services

MMTel 补充服务文档。通过 Sh 使用 3GPP TS 29.364 定义的 `MMTelServices` 根。

```
<MMTelServices
xmlns="http://uri.etsi.org/ngn/params/xml/simservs/xcap"
               xmlns:cp="urn:ietf:params:xml:ns:common-policy">
  <complete-communication-diversion>
    <communication-diversion active="true">
      <NoReplyTimer>20</NoReplyTimer>
      <cp:ruleset>
        <cp:rule id="busy">
          <cp:conditions><busy/></cp:conditions>
          <cp:actions><forward-to>
<target>sip:+61400000001@ims.example.org</target></forward-to>
</cp:actions>
        </cp:rule>
      </cp:ruleset>
    </communication-diversion>
  </complete-communication-diversion>
  <complete-communication-barring>
    <operator-outgoing-communication-barring authorized="true"/>
  </complete-communication-barring>
</MMTelServices>
```

```
sh_mmtel_services_communication_diversion_active=true
sh_mmtel_services_NoReplyTimer=20
sh_mmtel_services_busy_target=sip:+61400000001@ims.example.org
sh_mmtel_services_busy_busy=true
sh_mmtel_services_operator_outgoing_communication_barring_authorized=
sh_mmtel_services_xml_b64=PE1NVGVsU2VydmljZXM+...
```

服务标志（`communication_diversion_active`）、每条规则目标（`busy_target`）和条件存在标志（`busy_busy`）都由其作用域键入。请参见 活动和存在标志。

IMS-ODB-Information

运营商确定的阻止文档。它使用 3GPP TS 29.364 第 10 条 定义的 `OdbForImsOrientedServices` 根。每个阻止元素携带一个枚举代码。

```
<OdbForImsOrientedServices>
  <OdbForImsMultimediaTelephonyServices>
    <OutgoinBarring>1</OutgoinBarring>
    <IncomingBarring>1</IncomingBarring>
    <BarringOfRoaming>0</BarringOfRoaming>
  </OdbForImsMultimediaTelephonyServices>
</OdbForImsOrientedServices>
```

```
sh_ims_odb_information_OutgoinBarring=1
sh_ims_odb_information_IncomingBarring=1
sh_ims_odb_information_BarringOfRoaming=0
sh_ims_odb_information_xml_b64=PE9kYkZvckltc09yaWVudGVkU2VydmljZXM+..
```

有关枚举代码，请参见 ODB 枚举。

自定义运营商文档

运营商可在私有服务指示下配置任何文档。此示例使用 `E911` 来保存地址。

```
<Address><Number>221B</Number><Street>Baker Street</Street>
<City>London</City></Address>
```

```
sh_e911_Number=221B
sh_e911_Street=Baker Street
sh_e911_City=London
sh_e911_xml_b64=PEFkZHJlc3M+...
```

限制

- 重复元素需要 **id** 或 **name**。OmniTAS 通过其 `id` 或 `name` 属性为重复元素键入。MMTel 规则集工作良好，因为每个 `<cp:rule>` 都携带一个 `id`。如果兄弟元素共享标签且没有 `id` 或 `name`，OmniTAS 会在文档顺序中添加位置后缀。给这些元素一个 `id`，或将键放在标签名称中。
- 键遵循配置 **id**。使用 `id="cfb"` 配置的文档生成 `sh_<block>_cfb_target`，而不是 `sh_<block>_busy_target`。确认您的配置系统的约定。

- 缺失的活动属性意味着活动。测试布尔 false，而不是 `true`。请参见 活动和存在标志。
- 空元素仅在作用域内是标志。空元素仅在 id/name 作用域内变为 `true` 标志。顶级空元素不生成变量。使用 `sh_<block>_xml_b64` 来查看它。
- 混合内容未被捕获。位于子元素之间的文本不会被扁平化。使用 `sh_<block>_xml_b64` 来获取该内容。

参考表

标准服务示

服务示	文档根	目的	参考
`MMTEL-Services`	`MMTelServices`	MMTel 补充服务（转移、阻止、身份、等待、保持）	3GPP TS 29.364
`IMS-ODB-Information`	`OdbForImsOrientedServices`	运营商确定的阻止	3GPP TS 29.364 §10
`IMS-CAMEL-Services`	`im-csi-information`	IMS CAMEL 订阅数据	3GPP TS 29.364 §11

运营商特定文档必须使用非标准服务指示。主体没有固定模式。

ODB 枚举

元素	代码	意义
`OutgoinBarring`	0	所有外发通信被阻止
`OutgoinBarring`	1	外发国际通信被阻止
`OutgoinBarring`	2	外发国际通信被阻止，除非是回国 PLMN 国家
`OutgoinBarring`	3	在回国 PLMN 国家外漫游时外发通信被阻止
`IncomingBarring`	0	所有入站通信被阻止
`IncomingBarring`	1	在回国 PLMN 国家外漫游时入站通信被阻止

相关规范

规范	标题
TS 29.328	Sh 接口信令流和消息内容
TS 29.329	基于 Diameter 协议的 Sh 接口
TS 29.364	IMS 应用服务器服务数据描述□实现 AS 互操作性
TS 24.623	通过 Ut 接口操纵补充服务的 XCAP
TS 24.604	通信转移（CDIV）
TS 22.041	运营商确定的阻止（ODB）

Sh 接口（用户数据检索）

 返回主文档

Sh 接口通过 Diameter 提供对 HSS/Repository 中用户档案数据的访问。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - Diameter 对等体配置
- 操作南 - 控制面板中的 Sh 接口测试

呼叫处理集成

- 拨号计划配置 - 在拨号计划变量中使用 Sh 数据
- ⚙ 补充服务 - 呼叫转移的 MMTel-Config
- **SS7 MAP** - HLR 数据与 Sh 数据优先级

相关接口

- 在线计费 - Ro 接口（也使用 Diameter）
- 号码翻译 - Sh 查找前的号码标准化

监控

- 标参考 - Sh 接口指标和监控

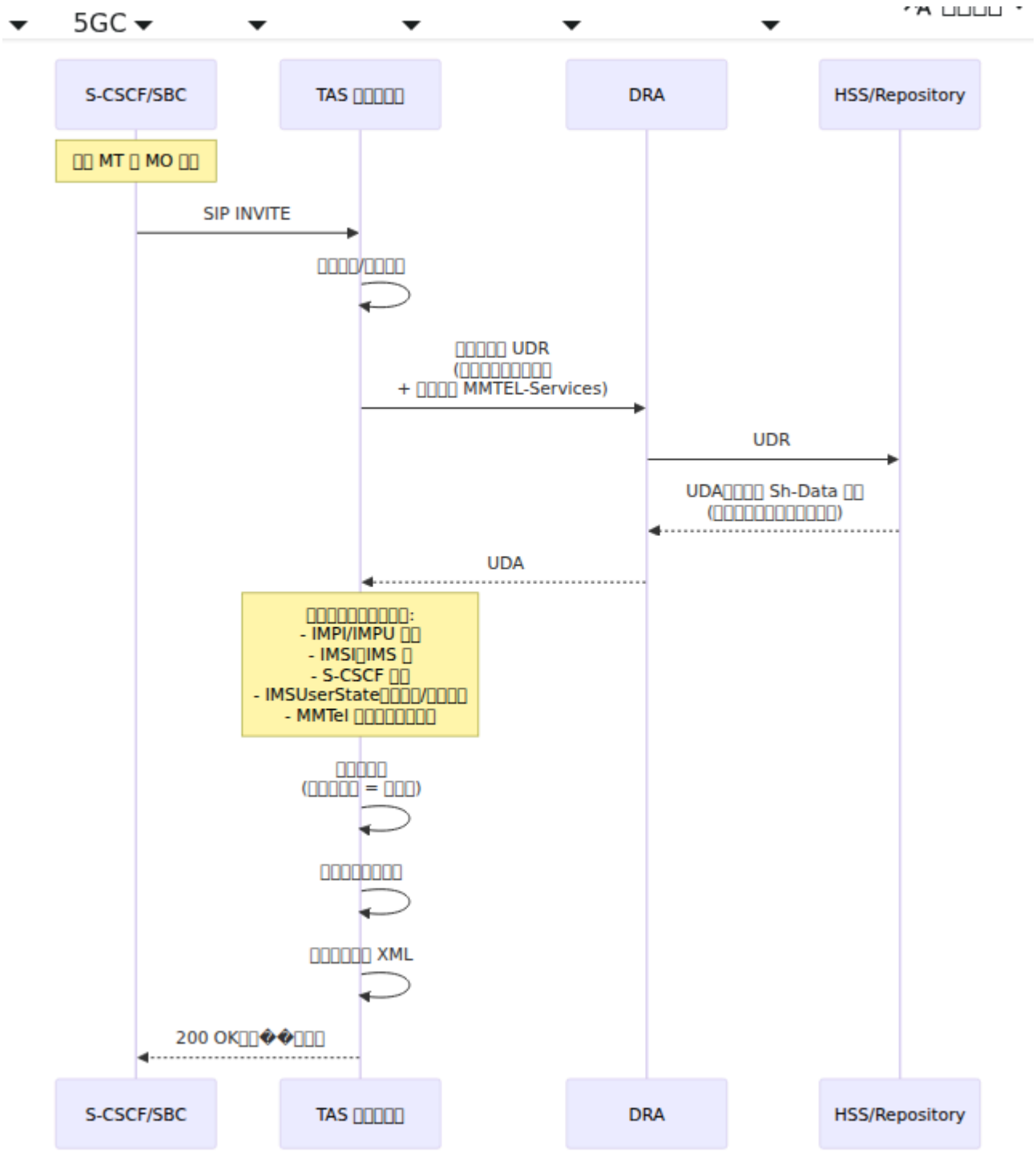
Sh 接口（用户数据检索）

Sh 接口用于在处理呼���之前从 HSS/Repository 检索用户档案数据。这些数据包括用户身份、服务和 MMTel 配置。

什么是 Sh 接口？

Sh 接口是 3GPP 标准化的 Diameter 接口，位于 TAS 和 HSS/Repository（Repo）之间。它提供实时访问：

IMS 用户身份（IMPI/IMPU）
呼叫转移设置（MMTel-Config）
用户服务授权
S-CSCF 分配



Sh 查找何时发生

Sh 查找发生在：

- **MT** 呼叫：查找被叫方（目的用户）
- **MO** 呼叫：查找主叫方（源用户）
- 紧急呼叫：查找主叫方（用于位置/身份）

在每种情况下，TAS 每个呼叫分支发出一个 UDR。MO 和 MT 使用相同的多数据引用 UDR 形状。只有指标标签不同。

多数据引用 **UDR**（**Notif-Eff**）

根据 3GPP TS 29.328 §6.1.1.1，当 AS 和 HSS 都支持 **Notif-Eff** 功能（通过 `Supported-Features` AVP 协商），一个单一的 UDR 可□携带多个 `Data-Reference` AVP，HSS 响应一个 UDA 其 `User-Data-Sh` 是一个单一的 `<Sh-Data>` 文档，将每个引用的子树作为兄弟节点连接。TAS 依赖于此。每个 Sh 查找 在一次往返中获取系统知道如何处理的所有引用的完整集合。

请求的数据引用（**TS 29.328** 表 **7.6.1**）

Ref	元素	TAS 用于填充
0	RepositoryData（每个 `Service-Indication` 一个，例如 `MMTEL-Services`，`IMS-ODB-Information`）	通用 `sh_<block>_*` 变量（未解析为专用字段）
10	IMSPublicIdentity	`ims_public_identity`，`msisdn`
11	IMSUserState	`ims_user_state`（来自 TS 29.328 ��7.6.3 的原始枚举值）
12	SCSCFName	`scscf_address`，`scscf_domain`
13	InitialFilterCriteria	（返回给 TAS 的主体，但目前未作为拨号计划变量显现）
14	LocationInformation	`location_rat_type`，`location_mme_name`，`location_vplmn_id`，`location_age_seconds`
15	UserState	`user_state`（来自 TS 29.328 §7.6.7 的原始枚举值）
17	MSISDN	（与 IMPU 交叉检查）
32	IMSI	`imsi`
33	IMSPrivateUserIdentity	`ims_private_identity`，`ims_domain`（从后缀解析）

所有单字符串字段都作为原始字符串变量显现到拨号计划中。TAS 不会解释它们的值。请参见 从 **Sh** 数据设置的拨号计划变量 表□获取完整列表。

示例 **UDA** 主体（已清理）

在实时跟踪中成功合并的响应如下。每个引用的 子树在请求的顺序下出现在一个 `<Sh-Data>` 包装器下：

```
[debug] 为 +614xxxxxxxx 获取了呼叫者数据
        (Data-Ref [0, 10, 11, 12, 13, 14, 15, 17, 32, 33],
         SI="MMTEL-Services"): 4453 字节

<?xml version="1.0" encoding="UTF-8"?>
<Sh-Data>
  <RepositoryData></RepositoryData>
  <PublicIdentifiers>
    <IMSPublicIdentity>sip:+614xxxxxxxx@ims.mnc001.mcc999.3gppnetwork
    <IMSPublicIdentity>tel:+614xxxxxxxx</IMSPublicIdentity>
  </PublicIdentifiers>
  <ShIMSData>
    <IMSUserState>1</IMSUserState>
  </ShIMSData>
  <ShIMSData>
    <SCSCFName>sip:scscf01.ims.mnc001.mcc999.3gppnetwork.org:5060</SC
  </ShIMSData>
  <IMSSubscription>
    <PrivateID>9999990000xxxxx@ims.mnc001.mcc999.3gppnetwork.org</Pri
    <ServiceProfile>
      ... InitialFilterCriteria 条目 ...
    </ServiceProfile>
  </IMSSubscription>
  <ShIMSData>
    <LocationInformation>
      <RAT-Type>eutran</RAT-Type>
      <MMEName>mme01.epc.mnc001.mcc999.3gppnetwork.org</MMEName>
      <VPLMNId>999001</VPLMNId>
      <AgeOfLocationInformation>NNNN</AgeOfLocationInformation>
    </LocationInformation>
  </ShIMSData>

<IMSPrivateUserIdentity>9999990000xxxxx@ims.mnc001.mcc999.3gppnetwork
</Sh-Data>
```

解析器如何处理合并的主体

对于身份/S-CSCF/状态数据引用，每个引用解析器是 独立的并且是 基于标签的：它在合并的主体中���索特定 元素名称（例如 `<SCSCFName>`, `<IMSPublicIdentity>`, `<IMSUserState>`）并 仅提取它关心的值。 RepositoryData 文档（数据引用 0 - MMTel, ODB 等）不会解析为字段；它们被通用地扁平化为 `sh_<block>_*` 变量（请参见 Repository Data Blocks）。没有解析器识别的子树会被静默忽略。

每个解析器的结果是一个部分用户数据映射；部分结果 按顺序与默认映射合并。这使得查找对 异构 HSS 实现和部分响应具有鲁棒性。请参见 下面的 优雅降级 部分。

选择请求哪些数据引用（`sh_data_references`）

默认情况下，TAS 请求上述 完整 集合（`[0, 10, 11, 12, 13, 14, 15, 17, 32, 33]` + `Service-Indication = "MMTEL-Services"`）用于每次查找，在 MO（主叫）和 MT（被叫）两侧。这是支持完整 Sh 数据模型的 HSS 的正确默认值。

一些 HSS 部署 不 接受完整集。因为 UDR 返回 [illegible] 的单引用结果，一个被拒绝的引用会导致整个请求失败。在诺基亚 HSS 上观察到的情况，例如：

- `5012 DIAMETER_UNABLE_TO_COMPLY` 对于 `LocationInformation`（14）/ `UserState`（15），当用户已断开连接时（这些需要对服务节点的 实时查询）。
- `5101 DIAMETER_ERROR_OPERATION_NOT_ALLOWED` 对于 AS 未配置读取的引用（例如 `RepositoryData`（0），`IMSUserState`（11），`SCSCFName`（12），`InitialFilterCriteria`（13），`IMSPrivateUserIdentity`（33））。

可选的 `:tas` 配置键 **`sh_data_references`** 允许操作员按 每侧 削减请求的集合。它是 可选的。如果未设置，则两侧使用完整的默认集合，因此现有部署不受影响。

```
# config/runtime.exs

# 每侧覆盖（主叫 = MO 源，被叫 = MT 目的地）。
# 省略的一侧将回退到完整的默认集合。
config :tas,
  sh_data_references: [
    caller: [10, 16, 17, 32],
    called: [10, 16, 17, 32]
  ]

# 或者应用于两侧的单一裸列表：
config :tas, sh_data_references: [10, 16, 17, 32]
```

接受的形状：

值	效果
(□□□)	两侧的完整默认集合（`@sh_data_references`）。
关键字列表 `[caller: [...], called: [...]]`	每侧覆盖；省略的一侧回退到默认。
裸列表 `[10, 17, 32]`	相同的列表应用于两侧。

`Service-Indication = "MMTEL-Services"` 始终发送；当 `RepositoryData`（0）不在列表中时，它是无害的。削减引用在数据广度与兼容性之间进行权衡：如果你删除 `SCSCFName`（12）或 MMTel `RepositoryData`（0），相应的拨号计划变量将回退到其默认值（请参见下面的优雅降级）。

Session-Id 呼叫处置标签

每个 Sh UDR 的 `Session-Id` 作为 RFC 6733 §8.8 实现特定的“可选值”（第 4 个 `;` 分隔字段）携带呼叫处置，因此可□在 HSS 日志和数据包捕获中区分 MO 与 MT 查找：

```
example-dc01-
as01.epc.mnc001.mcc001.3gppnetwork.org;1781225439574;7468896e;mo
                                       └ 时间戳 ┘ └ 随机 ┘ └
处置
```

`mo` 被附加到主叫/发起方查找，`mt` 被附加到被叫方/终止方查找。唯一性仍然由 时间戳 + 随机提供；该标签仅仅是一个关联提示，并被清理为 `[a-z0-9_]`，因此它永远不会破坏 Session-Id 语法。没有处置的查找（例如 Web UI Sh 测试工具）发出普通的 3 字段 Session-Id。

从 Sh 接口检索的数据

TAS 每个呼叫分支发出一个单一的多数据引用 UDR（请参见 多数据引用 UDR（Notif-Eff）上面的请求形状和合并响应）。TAS 从合并的 `<Sh-Data>` 主体中提取的字段分为三组：

1. IMS 身份：

- **IMPI**（私有身份）：从 `<IMSPrivateUserIdentity>` 元素解析。格式：`{IMSI}@{IMS-domain}`。TAS 在 `@` 上拆分□独立��复 IMSI 和 IMS 域。

- **IMPU**（公共身份）：从 `<IMSPublicIdentity>` 元素解析。格式：`sip:+{MSISDN}@{IMS-domain}`。MSISDN 去掉前面的 `+` 并作为 `msisdn` 拨号计划变量显现。

2. S-CSCF 分配：

- 当前用户注册的 S-CSCF 服务器名称和域名，从 `<SCSCFName>` 元素（数据引用 12）解析。MT 拨号计划用于将 INVITE 直接路由到注册的 S-CSCF，而不是扩展到 IMS 领域。
- 注意：TS 29.328 附录 D 中的规范 XML 元素名称是 `SCSCFName`（没有连字符）。带连字符的形式 "S-CSCF" 仅出现在规范文本中。

3. MMTel 服务（多媒体电话配置）：

- 在 `<RepositoryData>` 中返回，`Service-Indication = "MMTEL-Services"` 为键。
- 包含用户的补充服务（CDIV 呼叫转移、OIP/OIR、ICB/OCB 阻止、通信等待/保持）。TAS 不会将这些解析为字段。它们通用地作为 `sh_mmtel_services_*` 变量显现，供拨号计划解释（请参见 Repository Data Blocks）。

什么是 **MMTel-Config**?

MMTel-Config 是用户的多媒体电话服务配置 作为透明（存储库）数据存储在 HSS 中，`Service-Indication = "MMTEL-Services"` 为键。它作为与身份查找相同的 多数据引用 UDR 的一部分获（数据引用 0 加上服务指示 AVP）。通过 Sh，该文档是 **`<MMTelServices>`** 形式，由 **3GPP TS 29.364** 定义（AS/HSS 透明数据架构，与 UE 操作的 `<simservs>` XCAP-over-Ut 形式不同）。它通常包含一个 `complete-communication-diversion` 块，带有一个或多个 `cp:rule` 条目（`busy`，`noanswer`，`unregistered`，`notreachable`），一个可选的 `<NoReplyTimer>` 值，及其他 MMTel 子服务，如通信阻止和身份展示。权威架构（MMTelServices，IMS-ODB-Information）在 `docs/reference/mmtel-schemas/` 下提供。

TAS 不会将 MMTel 解析为专用字段。与每个 RepositoryData 文档一样，它通用地作为 `sh_mmtel_services_*` 通道变量（加上 `sh_mmtel_services_xml_b64`）显现到拨号计划中。拨号计划从这些变量中解释 CDIV / OIP / TIP / 阻止子服务。共享的 `call_forward_*` / `no_reply_timer` 变量不来自 MMTel；它们来自 HLR/SS7 和配置默认值。

从 **Sh** 数据设置的拨号计划变量

在成功的 Sh 查找后，这些变量被填充：

变量	来源
`ims_private_identity`	IMPI
`ims_public_identity`	IMPU
`msisdn`	IMPU（解析）
`imsi`	IMPI（解析）
`ims_domain`	IMPI/IMPU
`scscf_address`	SCSCFName
`scscf_domain`	SCSCFName（解析）
`call_forward_all_destination`	HLR/SS7（`Tas.SS7Map`）或 `"none`
`call_forward_not_reachable_destination`	HLR/SS7 或配置默认值
`no_reply_timer`	配置（`default_no_reply_timer`）
`ims_user_state`	IMSUserState（数据引用 11）
`user_state`	UserState（数据引用 15）

变量	来源
`location_rat_type`	LocationInformation/RAT-Type
`location_mme_name`	LocationInformation/MMEName
`location_vplmn_id`	LocationInformation/VPLMNId
`location_age_seconds`	LocationInformation/AgeOfLoca

Repository 数据块

每个 Sh RepositoryData 文档（例如 `MMTEL-Services`，`IMS-ODB-Information`，或特定于运营商的文档）是透明数据。OmniTAS 不会解析主体。OmniTAS 按服务指示拆分 Sh 响应，并将每个文档作为 `sh_<block>_*` 通道变量显现给拨号计划，外加文档的 Base64 副本作为 `sh_<block>_xml_b64`。OmniTAS 还设置全局的 `sh_data_*` 扁平变量和 `sh_raw_xml_b64`。

共享的 `call_forward_all_destination`，`call_forward_not_reachable_destination` 和 `no_reply_timer` 变量不来自 MMTel。它们来自 HLR/SS7 和配置默认值。

有关完整指南（配置、变量命名规则、示例块主体和限制），请参见 **Repository** 数据块。

优先级：Sh 数据与配置默认值

共享的 `call_forward_*` / `no_reply_timer` 变量使用下优先级顺序：

1. 来自 **SS7 MAP** 的 **HLR** 数据。对于 MT 呼叫，当漫游/转移在访问网络中处于活动状态时，设置 `call_forward_all_destination`（无条件）或 `call_forward_not_reachable_destination`（任何其他 转移原因）。请参见 SS7 MAP。
2. 配置默认值。当 HLR 不提供值时使用。配置在 `runtime.exs` 中的 `config :tas` 下：`call_forward_not_reachable_destination` 和 `default_no_reply_timer`。

MMTel 补充服务数据 不 在此路径中。它通用地显现为 `sh_mmtel_services_*`，并由拨号计划直接解释。

当 **Sh** 查找失败时会发生什么

整个请求失败场景：

1. 用户未在 **HSS** 中配置：
    - HSS 返回 `Experimental-Result-Code 5001`（DIAMETER_ERROR_USER_UNKNOWN）
    - TAS 将呼叫分支视为不可解析
    - `hangup_case` 变量设置为 `"UNALLOCATED_NUMBER"`
    - 呼叫被拒绝，返回适当的 SIP 响应
2. **HSS** 无法访问/超时：
    - Sh 请求超时（默认：5000ms，参见 Diameter `request_timeout` 在 `runtime.exs` 中）
    - 记录错误并记录指标
    - 呼叫分支与情况（1）相同的方式失败
3. **HSS** 不支持多数据引用 **UDR**：
    - HSS 要么返回错误，要么静默丢弃请求（取决于 HSS）
    - 从 TAS 方面看，这看起来像情况（1）或（2）：查找整体失败 并且呼叫分支被拒绝
    - HSS 必须实现 Notif-Eff 功能，使 TAS 正常工作。请参见 TS 29.328 §6.1.1.1 获取功能定义。

优雅的每子树降级

当 UDR 本身成功（`Result-Code: 2001`）但合并的 `<Sh-Data>` 主体的 个别子树缺失时，TAS 不会 使呼叫失败。每个引用解析器是独立的，并在其标签缺失时回退到定义的默认值。操作员只需担心整体请求 失败（如上所述）；部分数据降级是自动的，并在调试日志中可观察到。

子树缺失	结果
`<SCSCFName>`（数据引用 12）	`scscf_address` 和 `scscf_domain` 设置为 `"none"`
`<IMSPrivateUserIdentity>`（数据引用 33）	`ims_private_identity`，`imsi`，`ims_domain` 设置为 `"none"`
MMTel RepositoryData 存在	通用地显现为 `sh_mmtel_services_*`；不填充 `call_forward_*`（这些是 HLR/配置拥有的）
MMTel RepositoryData 缺失	`call_forward_not_reachable_destination` / `no_reply_timer` 回退到其配置默认值；`call_forward_all_destination` 为 `"none"`，除非由 HLR 设置
空的 `<RepositoryData> </RepositoryData>`	所有 MMTel 派生字段回退到配置默认值
空/缺失的 `<IMSUserState>`，`<LocationInformation>`，`<InitialFilterCriteria>`	当前没有拨号计划副作用（已解析但尚未连接）

唯一的硬要求是响应必须包含一个 `<IMSPublicIdentity>` 元素。如果该标签缺失，查找返回 `{:error, :sh_parse_failed}`，呼叫分支被视为不可解析（与上述情况 1 的下游行为相同）。每个其他字段都是“自由请求，尽可能获取”。

这使得 TAS 对异构 HSS 部署具有弹性：一个实现 Notif-Eff 的 HSS 但仅填充 IMSPublicIdentity、MSISDN 和 SCSCFName（例如）仍将产生有效的呼叫；拨号计划只是回退到配置默认值□获取 MMTel 派生变量。

监控 **Sh** 接口

关键□标：

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

# Sh 查找延迟（P95）
histogram_quantile(0.95,
  rate(subscriber_data_duration_milliseconds_bucket[5m]))

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

警报阈值：

- P95 延迟 > 100ms：HSS 响应缓慢
- 错误率 > 5%：HSS 连接问题
- 错误率 > 20%：HSS 关键故障

故障排除：

1. 检查 Web UI 中的 Diameter 对等体状态（`/diameter`）
2. 在 Web UI 中测试 Sh 查找（`/sh_test`），使用已知用户
3. 查看日志中的“用户数据”错误
4. 验证 HSS/Repository 是否可从 TAS 访问
5. 检查 `subscriber_data_lookups_total` 指标□查找模式

测试 **Sh** 接口

使用 Web UI Sh 测试工具（`/sh_test`）：

1. 在控制面板中导航到 `/sh_test`
2. 输入用户 MSISDN（例如，`+614xxxxxxxx`）
3. 点击“查询 Sh”
4. 查看返回的数据：
   - IMPI/IMPU 身份
   - S-CSCF 分配
   - MMTel 服务
   - 呼叫转移配置

常见测试场景：

验证新配置的用户是否在 HSS 中
检查特定用户的呼叫转移设置
验证 IMS 注册后的 S-CSCF 分配
测试 HSS 连接性和响应时间

SRVCC（单无线电语音通话连续性）

 返回主文档

TAS 如何锚定一个活跃的 VoLTE 通话，并在无线电将通话从 LTE（PS）切换到 2G/3G 电路交换（CS）承载时执行 SIP 会话转移。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - 完整的配置参考
- 拨号计划配置 - 通话路由和拨号计划模板的工作原理

通话处理

- 补充服务 - 紧急呼叫、呼叫转移、CLI 阻止
- **Sh** 接口 - 来自 HSS 的用户数据

概述

SRVCC（3GPP TS 23.216）在用户设备（UE）离开 LTE 覆盖并必须转移到 2G/3G CS 承载时保持语音通话。TAS 是每个 VoLTE 通话的锚点：它同时持有通向 UE 的接入腿和通向远端方的远程腿。当切换发生时，TAS 将远程腿移动到新的 CS 路径，并释放旧的 LTE 接入腿。远程方保持连接。只有 TAS 和 UE 之间的路径发生变化。

本页描述 PS 到 CS 的方向（LTE 到 CS）。反向 CS 到 PS 的方向计划作为后续内容。

通话流程（**PS** 到 **CS**）

1. eNB 检测到 LTE 信号下降，并向 MME 发送 S1AP 切换请求。

2. MME 向 MSC-Server 发送 Sv PS-to-CS 请求。该请求携带 STN-SR，一个路由到 TAS 的 E.164 号码。
3. MSC-Server 向 STN-SR 发起一个正常的 CS 通话。MSC-Server 对 SIP 或 IMS 一无所知。
4. CS 通话 SIP INVITE 的形式到达 TAS，地址为 STN-SR 号码。
5. TAS 识别 STN-SR，推导出用户 MSISDN，找到活跃的 VoLTE 会话，并执行 SIP 会话转移。
6. TAS 将远程腿转移到 CS 路径，并拆除 LTE 接入腿。

```
MSC-S ──INVITE sip:+9<MSISDN>──► TAS
        (C-MSISDN: <MSISDN>)
                                  │ strip prefix -> find active
session
        ◄──100 Trying──────────────┤
              TAS ──REFER──► Remote S-CSCF (Refer-To: CS MGW
endpoint)
                  ◄──202 Accepted──
                  ◄──NOTIFY (200 OK sipfrag)──  transfer succeeded
        ◄──200 OK──────────────────┤
              TAS ──BYE──► UE VoLTE access leg (teardown)
```

STN-SR 前缀方案

STN-SR（SRVCC 会话转移号码）是 TAS 所有的 E.164 号码前缀范围内的号码。TAS 将 STN-SR 形成如下：国际逃逸码，然后是配置的前缀，然后是用户 MSISDN 数字：

```
STN-SR = <escape> + <prefix> + <MSISDN digits>
```

对于前缀为 `9` 和用户 `+61412345678`，STN-SR 为 `+961412345678`。TAS 通过去掉前缀并保留逃逸码来恢复 MSISDN：

```
+961412345678  --(strip "9")-->  +61412345678
```

前缀方案有三个好处：

- 无需每个用户的配置。每个用户都有一个隐式的 STN-SR。
- MSISDN 通过简单的前缀剥离恢复，因此用户查找是即时的。

- 范围是一个单一的内部 DID 块，容易保持不可路由。

> STN-SR 范围必须在核心外部不可路由。请在 PSTN 网关处阻止它，并仅将其作为内部 DID 范围保留。

配置

SRVCC 是可选的，默认关闭。当 `srvcc` 键缺失或 `enabled` 不是 `true` 时，通话将永远不会被视为 STN-SR 触发，正常的通话处理保持不变。

```
config :tas,
  ...
  srvcc: %{
    enabled: true,
    stn_sr_prefix: "9"
  },
  ...
```

配置参数：

参数	类型	必需	描述
`enabled`	boolean	是	开启 SRVCC 转移处理。当为 `false` 或未设置时，STN-SR 号码被视为正常通话。
`stn_sr_prefix`	string	是	标识 TAS STN-SR 范围的前缀。紧跟在 `+` 或 `00` 国际逃逸码后。示例 `"9"`。

两个键都是 SRVCC 激活所必需的。如果 `enabled` 为 `true` 但 `stn_sr_prefix` 为空，则 SRVCC 将保持关闭，防部分配置意外转移通话。

TAS 如何处理 STN-SR INVITE

当 INVITE 到达时，授权阶段会在应用正常终止通话逻辑之前检查请求 URI 号码是否与 STN-SR 前缀匹配。STN-SR INVITE 被路由到转移处理程序，而不是标准 MT 处理。

转移处理程序然后:

1. 去掉前缀□推导出用户 MSISDN。
2. 确认节点已为 SRVCC 配置。
3. 在实时通话状态中查找用户的活跃 VoLTE 会话。
4. 使用 C-MSISDN 头在多个通话进行时选择正确的会话（根据 TS 24.237 的通话保持或通话等待）。
5. 回应 MSC-Server 的 INVITE 并执行会话转移。

无法转移时的响应

情况	**SIP** 响应	行为
MSISDN 没有活跃的 VoLTE 会话	404 Not Found	通知 MSC-Server 转移无法进行。
用户未配置 SRVCC	480 Temporarily Unavailable	通话被干净地拒绝。
多个会话且 C-MSISDN 无法解析一个	480 Temporarily Unavailable	通话被拒绝，而不是猜测。
���求 URI 不是有效的 STN-SR	484 Address Incomplete	号码未携带前缀。

在转移过程中失败的转移不会中断原始的 VoLTE 通话。接入腿仅在转移确认后释放。

拨号计划

SRVCC 使用与紧急呼叫相同的模式: TAS 检测 通话并 呈现专用拨号计划模板，变量已填充。STN-SR 前缀匹配存在于 Elixir 路由层（`Tas.Dialplan.Authorization` 使用配置的 `stn_sr_prefix`），而不在一般的 MT 拨号计划中。当识别到 STN-SR INVITE 时，TAS 呈现专用的 SRVCC 模板，而不是正常的 MT 模板。

- `priv/templates/srvcc_transfer_dialplan.xml` 是专用的 SRVCC 模板。它记录转移，作为通道变量（`srvcc_msisdn`，`srvcc_c_msisdn`，`srvcc_transferred_call_id`，`srvcc_cs_target`）暴露决策，回应 MSC-Server 的 INVITE，并推动远程腿转移到 CS 承载。

相关接口

- **MME Sv** 接口触发 PS-to-CS 并将 STN-SR 发送到 MSC-Server。
- **HSS** 在 S6a 更新位置应答中提供 STN-SR 和 C-MSISDN。

3GPP 参考

规范	主题
TS 23.216 §6.2	PS-to-CS SRVCC 程序
TS 24.237	IMS 服务连续性（SIP 级别细节）
TS 29.280	Sv 接口（MME 到 MSC-Server）

SS7 MAP / Gateway-MSC 配置

 返回主文档

配置 HLR 查询通过 SS7 MAP 检索 MSRN（漫游号码）和呼叫转移信息。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - SS7 MAP 配置 (`ss7_map` 参数)
- 操作南 - 控制面板中的 HLR/MAP 测试

呼叫处理集成

- 拨号计划配置 - 在拨号计划路由中使用 MSRN 和 forwarded_to_number
- ⚙ 补充服务 - 基于 HLR 的呼叫转移（替代 Sh/MMTel）
- **Sh** 接口 - Sh 与 MAP 数据优先级
- 号码转换 - HLR 查询的号码格式

测试与监控

- **HLR** 和呼叫模拟器 - 测试 HLR/MAP 集成
- 标参考 - HLR/MAP 查询指标

Gateway-MSC 配置

TAS 可询 HLR 检索漫游号码（MSRN）或 MSC，当用户在 2G/3G 网络上漫游时，也可检索呼叫转移信息。

这将设置 `msrn` 或 `forwarded_to_number` 拨号计划变量，然后可用于适当地路由呼叫。

配置参数：

- `enabled` - 启用/禁用 SS7 MAP 功能
- `http_map_server_url_base` - MAP 网关 HTTP API 的基本 URL
- `gmsc` - 用于 SRI/PRN 查询的网关 MSC 地址
- `timeout_ms` - MAP 操作的 HTTP 超时（毫秒为单位，默认值：5000）

```
config :tas,
  ...
  ss7_map: %{
    enabled: true,
    http_map_server_url_base: "http://10.5.1.216:8080",
    gmsc: "55512411506",
    timeout_ms: 5000  # 可选，默认为 5000ms
  },
```

功能：TAS 执行 SRI（发送路由信息）并根据下优先级处理路由：

1. 呼叫转移激活 - 如果 SRI 响应包含转移号码，则将其视为 MSRN（不执行 PRN）。转移号设置在 `msrn` 变量中并用于路由。
2. 漫游（**2G/3G**） - 如果用户正在漫游（VLR 存在）且没有激活呼叫转移，则执行 PRN（提供漫游号码）获取用于路由到 V-MSC 的 MSRN
3. 正常 - 如果既不适用转移也不适用漫游，则呼叫按标准路由进行

msrn 和 tas_destination_number 拨号计划变量被适当地设置用于路由（无论是来自 PRN 还是来自转移号码）

补充服务

 返回主文档

呼叫转移、CLI 阻止和紧急呼叫服务的配置和实施。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - 服务配置参数（紧急代码、CLI 阻止、默认呼叫转移）
- 操作南 - 测试补充服务

呼叫处理与数据源

- 拨号计划配置 - 在拨号计划逻辑中实施服务
- **Sh** 接口 - 呼叫转移设置的 MMTel-Config
- **SS7 MAP** - 基于 HLR 的呼叫转移（Sh 的替代方案）
- 号码翻译 - CLI 阻止前缀处理

服务交互

- 在线计费 - 紧急呼叫绕过 OCS
- 语音信箱 - 在忙碌/无人接听的情况下转接到语音信箱

监控

- 标参考 - 呼叫转移和服务指标
- 拨计划标 - 服务使用指标

补充服务（呼叫转移 / 阻止 CLI / 紧急代码）

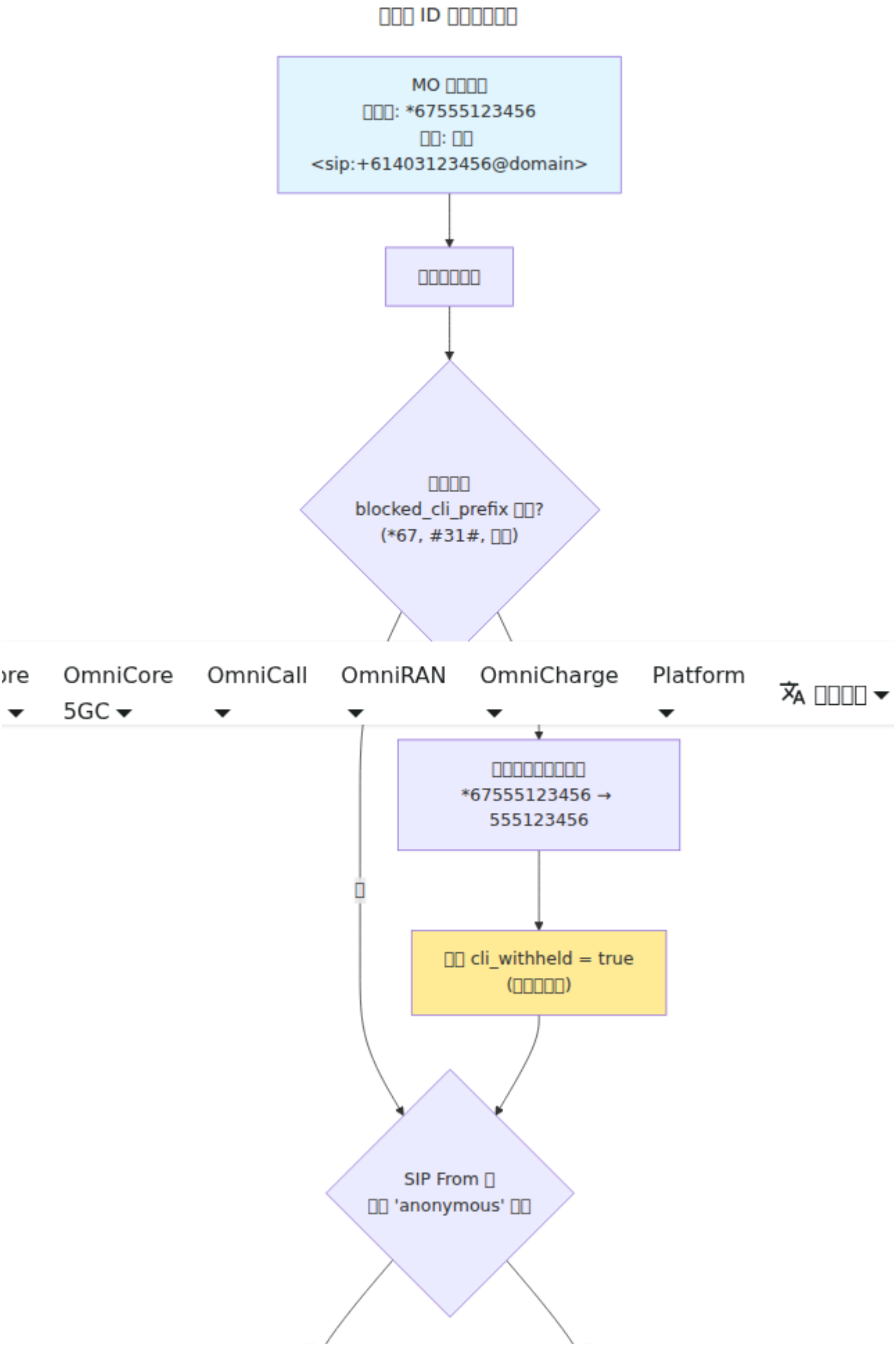
阻止 CLI 前缀、紧急呼叫代码和默认呼叫转移数据的配置（当从 Sh 返回的 MMTel-Config 数据为空时，仅使用呼叫转移 / 无应答数据）。

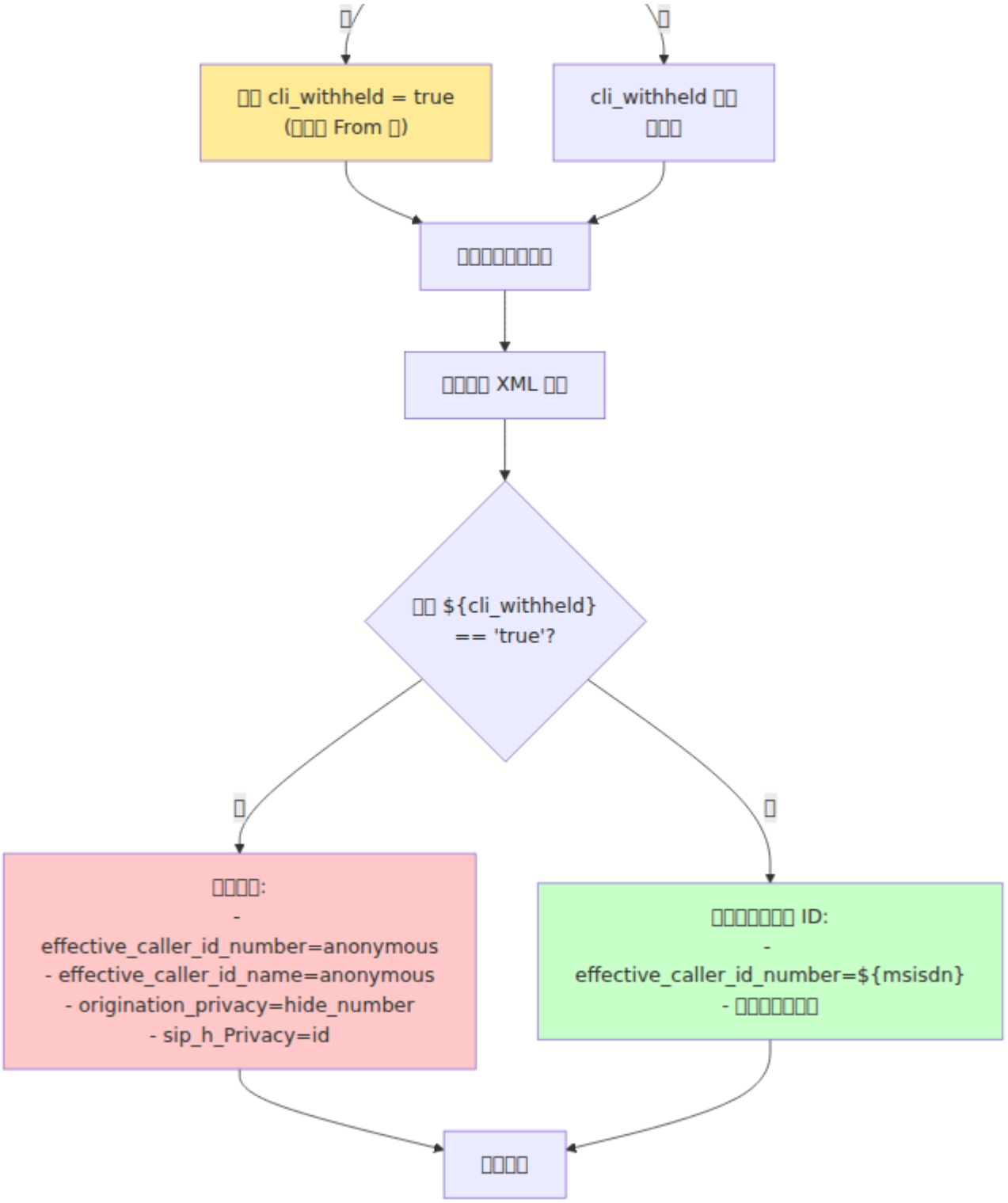
```
config :tas,
  ...
  blocked_cli_prefix: ["*67"],
  call_forward_not_reachable_destination: "2222",
  default_no_reply_timer: 30,
  emergency_call_codes: ["911", "912", "913", "sos"],
  ...
```

配置参数：

- **`blocked_cli_prefix`**（字符串列表）：触发 CLI（呼叫线路 ID）隐藏的前缀
  - 示例：`["*67"]` - 在号码前拨打 *67 隐藏来电号码
  - 在拨号计划中用于设置 `cli_withheld` 变量
- **`call_forward_not_reachable_destination`**（字符串）：默认的呼叫转移未到达目的地（CFNRc）
  - 仅在从 Sh 接口未返回 MMTel-Config 时使用
  - 示例：`"2222"` - 转接到语音信箱
- **`default_no_reply_timer`**（整数）：CFNRc 激活前的默认超时（秒）
  - 仅在从 Sh 接口未返回 MMTel-Config 时使用
  - 示例：`30` - 在转接前响铃 30 秒
- **`emergency_call_codes`**（字符串列表）：您所在辖区的紧急服务号码
  - 在呼叫授权期间检查检测紧急呼叫
  - SIP 紧急 URN（如，`<urn:service:sos>`）始终会与这些代码一起检查
  - 常见示例：`["911", "112", "000", "999", "sos"]`
  - 有关详细用法，请参见 紧急呼叫 部分

呼叫者 ID 阻止的工作原理
TAS 支持两种阻止呼叫者 ID（CLI 隐藏）的方法，这两种方法都将 cli_withheld 拨号计划变量设置为 "true"：


□□□ ID □□□□□□□
MO □□□□
□□□: *67555123456
□□: □□
<sip:+61403123456@domain>
□□□□□□□
□□□□
blocked_cli_prefix □□?
(*67, #31#, □□)
ore
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
□□□□
□□□□□□□□□□
*67555123456 →
555123456
□
□□ cli_withheld = true
(□□□□□)
SIP From □
□□ 'anonymous' □□



方法 **1**：基于前缀的阻止

当用户拨打带有 `blocked_cli_prefix` 中代码的目的号码时：

1. 号码翻译模块检测到前缀（例如，呼叫者拨打 `*67555123456`）
2. 前缀从目的号码中 剥离（变为 `555123456`）
3. `cli_withheld` 变量被设置为 `"true"`
4. 拨号计划可□使用此变量来隐藏呼叫者的身份

示例配置：

```
blocked_cli_prefix: ["*67"]         # 美国风格的阻止
blocked_cli_prefix: ["#31#"]        # 欧洲 GSM 风格的阻止
blocked_cli_prefix: ["*67", "#31#"] # 支持两者
```

方法 **2**：**SIP From** 头检测

当 UE/手机通过 SIP 头请求隐私时：

1. TAS 检查 SIP From 头的显示名称是否包含 "anonymous"（不区分大小写）
2. 如果找到，`cli_withheld` 变量被设置为 `"true"`
3. 这遵循在设备级别设置的用户隐私请求

在拨号计划中实施 **CLI** 阻止

TAS 设置 `cli_withheld` 变量，但您的 拨号计划 **XML** 必须实现实际的阻止行为：

```
<extension name="CLI-Privacy" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <!-- 隐藏呼叫者身份 -->
    <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"/>

    <!-- 可选地设置 P-Asserted-Identity 隐私 -->
    <action application="set" data="sip_h_Privacy=id"/>
  </condition>
</extension>
```

TAS 为 **CLI** 阻止设置的变量：

TAS 在拨号计划执行之前设置这些变量：

变量	类型	值	描述
`cli_withheld`	字符串	`"true"` 或 `"false"`	指示是否通过前缀或 From 头请求了 CLI 阻止
`tas_destination_number`	字符串	规范化号码	去掉阻止前缀后的目的地（例如，`555123456`）
`destination_number`	字符串	规范化号码	与 `tas_destination_number` 相同（两者都被设置）

您的拨号计划应设置的变量（当 **`cli_withheld="true"`** 时）：

这些变量控制呼叫者身份的呈现方式：

变量	推荐值	目的
`effective_caller_id_number`	`"anonymous"`	隐藏呼叫者的电话号码
`effective_caller_id_name`	`"anonymous"`	隐藏呼叫者的显示名称
`origination_privacy`	`"hide_number"`	出站环节的 SIP 隐私标志
`sip_h_Privacy`	`"id"`	SIP 隐私头（RFC 3323）
`sip_h_P-Asserted-Identity`	（未设置或移除）	可选：移除 P-Asserted-Identity 头

完整拨号计划示例：

```
<extension name="CLI-Privacy-Handler" continue="true">
  <condition field="${cli_withheld}" expression="true">
    <!-- 记录以便故障排除 -->
    <action application="log" data="INFO 请求对
${tas_destination_number} 的 CLI 阻止"/>

    <!-- 在出站呼叫中隐藏呼叫者身份 -->
    <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"/>

    <!-- 设置 SIP 隐私头 -->
    <action application="set" data="sip_h_Privacy=id"/>

    <!-- 可选：如果存在则移除 P-Asserted-Identity -->
    <action application="unset" data="sip_h_P-Asserted-Identity"/>

    <!-- 如果 cli_withheld 为 false，则执行反向操作 -->
    <anti-action application="log" data="DEBUG 使用正常呼叫者 ID:
${msisdn}"/>
    <anti-action application="set"
data="effective_caller_id_number=${msisdn}"/>
  </condition>
</extension>

<!-- 此扩展继续到实际的呼叫路由 -->
<extension name="Route-Outbound-Call">
  <condition field="${tas_destination_number}"
expression="^(.+)$">
    <action application="bridge"
data="sofia/gateway/trunk/${tas_destination_number}"/>
  </condition>
</extension>
```

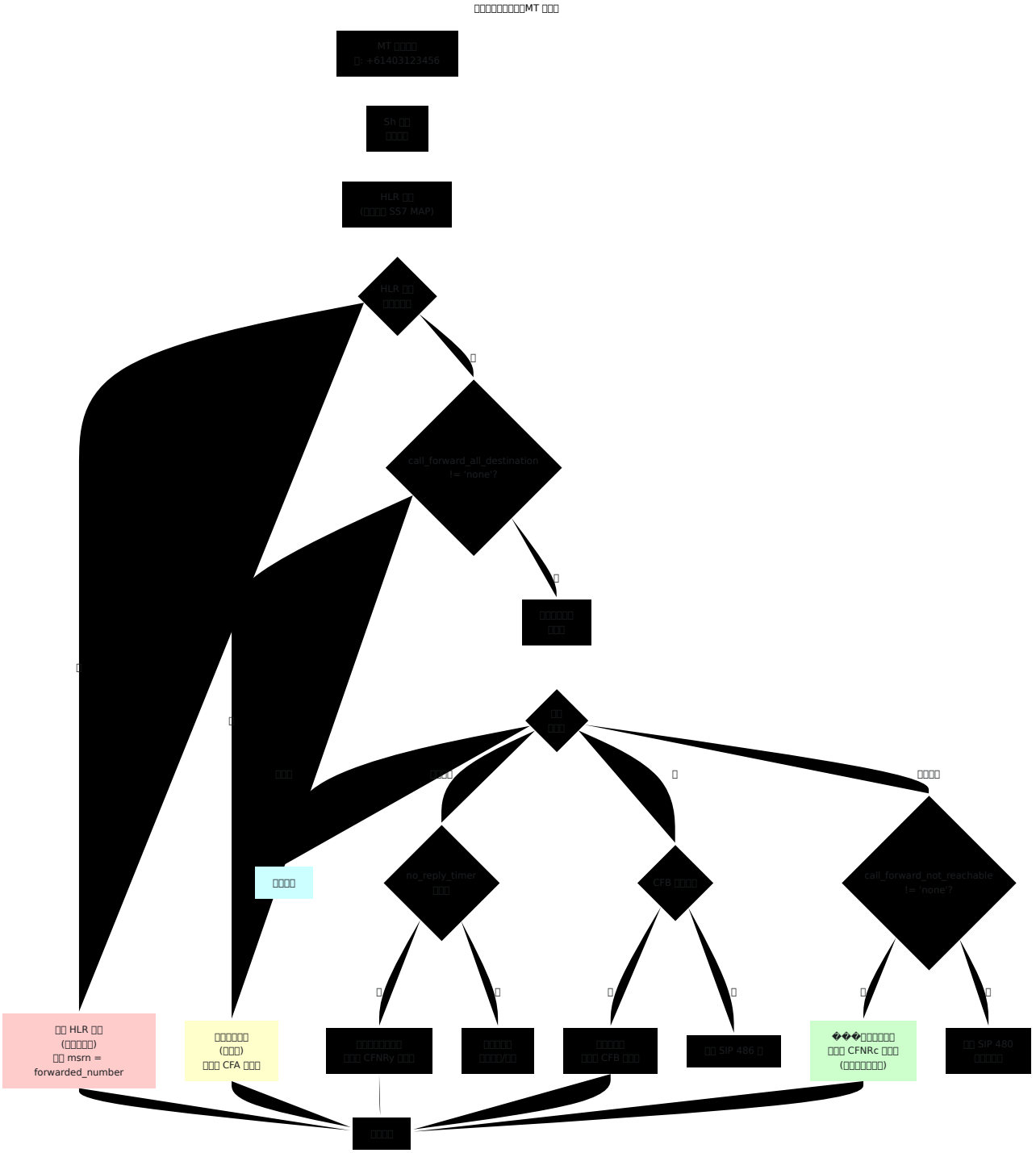
重要说明：

- 两种方法可同时工作（前缀或 SIP 头触发阻止）
- 前缀始终从目的号码中剥离

- `cli_withheld` □□□ □□□□`"true"` □ `"false"`□□□□□□□□
- □□□□ / □□ CLI □□□□□□□□□ XML □□□
- □□□
- □□□ MO□□□□□□□□□□□□□□□

□□□□□□□□□

□□□□□□□□□□□□□□ CDIV□□□□□□□□□□□□□□□□□□□□□□□□TAS □□□□□□□□□□□□□□□□□□□□□□□□□□□



呼叫转移的类型

1. 呼叫转移全部 (CFA) - 无条件转移

- 变量: `call_forward_all_destination`
- 何时激活：所有来电立即转接
- 优先级：首先检查（在 HLR 转移之后）
- 常见用途：用户希望将所有呼叫发送到另一个号码
- 示例：商务电话转接到个人手机

2. 呼叫转移忙 (CFB)

- 何时激活：当用户已经在通话中时转接呼叫
- **SIP** 响应： 486 忙触发转接
- 常见用途：当正在通话时转接到语音信箱

3. 呼叫转移无人接听 (CFNRy)

- 变量: `no_reply_timer`
- 何时激活：在指定秒数内未接听时转接呼叫
- 超时：通常为 15-30 秒
- 常见用途：如果��接听则转接到语音信箱

4. 呼叫转移未到达 (CFNRc)

- 变量: `call_forward_not_reachable_destination`
- 何时激活：用户离线、未注册或无法到达
- **SIP** 响应： 480 暂时不可用
- 常见用途：当手机关闭时转接到语音信箱
- 默认：如果没有 MMTel-Config，则使用配置参数

数据源优先级

呼叫转移数据从多个源检索，优先级如下：

```
1. HLR 数据 (SS7 MAP)              [最高优先级 - 覆盖所有]
   ↓ (如果没有 HLR 转移激活)
2. MMTel-Config (Sh 接口)  [来自 HSS 的用户特定设置]
   ↓ (如果未返回 MMTel-Config)
3. 配置默认值           [最低优先级 - 回退值]
```

为什么使用这个优先级?

- **HLR** 数据：实时转移状态，适用于漫游/网络场景
- **MMTel-Config**：用户在 IMS 中配置的首选项
- 配置默认值：网络范围内的回退（通常是语音信箱）

呼叫转移的拨号计划变量

□□	□□	□□	□□□
`call_forward_all_destination`	□□□	Sh/MMTel □ `"none"`	`"61403555123"`
`call_forward_not_reachable_destination`	□□□	Sh/MMTel □□□	`"2222"`
`no_reply_timer`	□	Sh/MMTel □□□	`30`
`msrn`	□□□	HLR□□ MT□	`"61400123456"`
`tas_destination_number`	□□□	□□□□	`"2222"`

□□□□□□□□□□□□□□

□□ **MT** □□□□□□□□□□

```
<!-- 检查呼叫转移全部（在 HLR 之后的最高优先级） -->
<extension name="Check-CFA" continue="true">
  <condition field="${call_forward_all_destination}"
expression="^(?!none$).+$">
    <action application="log" data="INFO 呼叫转移全部激活到
${call_forward_all_destination}"/>
    <action application="set"
data="tas_destination_number=${call_forward_all_destination}"/>
  </condition>
</extension>

<!-- 尝试桥接到用户 -->
<extension name="Bridge-To-Subscriber">
  <condition field="${msrn}" expression="^none$">
    <!-- 没有 MSRN，路由到本地用户 -->
    <action application="set" data="call_timeout=${no_reply_timer}"/>
    <action application="bridge"
data="sofia/internal/${tas_destination_number}@${scscf_address}"/>

    <!-- 如果桥接失败，检查转移 -->
    <action application="log" data="INFO 桥接失败，检查呼叫转移"/>

    <!-- 呼叫转移未到达 -->
    <action application="set"
data="forward_destination=${call_forward_not_reachable_destination}"/
    <action application="log" data="INFO 转接到
${forward_destination}"/>
    <action application="answer"/>
    <!-- 路由到 TAS ESL 语音信箱（存储）；见 voicemail.md -->
    <action application="set" data="tas_vm_mode=deposit"/>
    <action application="set" data="tas_vm_mailbox=${msisdn}"/>
    <action application="set"
data="tas_vm_caller=${effective_caller_id_number}"/>
    <action application="socket" data="127.0.0.1:8084 async full"/>
  </condition>
</extension>
```

配置默认呼叫转移

在 `config/runtime.exs` 中设置网络范围的默认值：

```
config :tas,
  # 默认 CFNRc 目的地（在没有 MMTel-Config 时使用）
  call_forward_not_reachable_destination: "2222",  # 语音信箱访问号码

  # 默认超时，在没有 MMTel-Config 时激活 CFNRy（秒）
  default_no_reply_timer: 30  # 响铃 30 秒
```

何时使用默认值：

- 用户存在于 HSS 中，但没有配置 MMTel-Config
- Sh 查找成功但未返回呼叫转移设置
- 新用户在配置呼叫转移之前

故障排除呼叫转移

问题：呼叫未按预期转移

1. 检查 **Sh** 数据：
   - 使用 Web UI `/sh_test` 查询用户
   - 验证 MMTel-Config 包含 CDIV 规则
   - 检查 `call_forward_all_destination` 值
2. 检查拨号计划变量：
   - 查看呼叫日志中的变量值
   - 确认 `call_forward_all_destination` != `"none"`
   - 验证 `tas_destination_number` 设置为转接目的地
3. 检查 **HLR** 数据（如果启用 **SS7 MAP**）：
   - 使用 Web UI `/hlr` 查询用户
   - HLR 转移覆盖 Sh 数据
   - 验证 `msrn` 变量不包含意外的转接号码
4. 检查配置默认值：
   - 验证配置中的 `call_forward_not_reachable_destination`
   - 确认 `default_no_reply_timer` 合适

- 这些仅在没有 MMTel-Config 时适用

问题：转移循环

症状：呼叫转接到一个转接回来的号码，形成循环

在拨号计划中的预防：

```
<!-- 跟踪转接跳数 -->
<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 检测到转移循环，跳数: ${sip_h_X-
Forward-Hop-Count}"/>
    <action application="hangup" data="LOOP_DETECTED"/>
  </condition>
</extension>
```

监控呼叫转移

关键□标：

- 高比例的呼叫到语音信箱号码
- 呼叫在 `no_reply_timer` 值时超时的模式
- 呼叫始终路由到相同的转接目的地

有用的日志：

```
INFO 呼叫转移全部激活到 61403555123
INFO 转接到 2222
INFO 桥接失败，检查呼叫转移
```

商业智能：

跟踪每个用户的转移激活率
监控语音信箱使用模式
识别具有无条件转移的用户

语音信箱与未接来电服务

 返回主文档

OmniTAS 提供自己的语音信箱服务用于 留言（留下信息）和 检索（收听信息）。交互流程完全由 TAS 通过 FreeSWITCH 事件套接字库（ESL）驱动，而不是 FreeSWITCH 内置的 `mod_voicemail` IVR。存储与 `mod_voicemail` 兼容，因此语音信箱网页 UI、REST API 和消息等待（MWI）SMS 通知继续正常工作。

相关文档

核心文档

- 主 **README** - 概述和快速入门
- 配置南 - 语音信箱配置（时区、SMSc、通知模板）
- 操作南 - 控制面板中的语音信箱管理

呼叫处理集成

- 拨号计划配置 - 拨号计划中的语音信箱留言/检索
- ⚙ 补充服务 - 忙音/无人接听时转接到语音信箱
- **TTS** 提示 - 语音信箱问候提示

相关服务

- 号码翻译 - 语音信箱访问号码翻译

监控

- 标参考 - 语音信箱使用指标

架构

TAS 已经保持与 FreeSWITCH 的 入站 ESL 连接，用于事件、命令和监控。语音信箱增加了第二条独立路径：一个 出站套接字 监听器，FreeSWITCH 在拨号计划将呼叫交给语音信箱时连接到该监听器。通过该每个呼叫的套接字，TAS 播放提示、录制音频并直接收集 DTMF。

关键点：

- 入站 ESL 路径（呼叫事件、在线计费、监控）未改变。
- 出站套接字仅用于交互式语音���箱 IVR。

- TAS 和 FreeSWITCH 是共置的，因此出站套接字绑定到回环。
- 存储使用 `mod_voicemail` 数据库模式，因此 `vm_boxcount`、网页 UI 和 REST API 继续正常工作，无需修改。`mod_voicemail` 必须在 **FreeSWITCH** 中保持加载，□便数据库和 `vm_boxcount` 命令保持可用。

将呼叫路由到语音信箱

语音信箱在 XML 拨号计划中根据需要添加；除非拨号计划将呼叫路由到它，否则它不会激活。拨号计划设置三个通道变量，然后将呼叫交给出站套接字，TAS 读取这些变量□决定该做什么。

变量	设置时机	描述
`tas_vm_mode`	始终	`deposit` 留下信息，或 `retrieve` 收听信息。录制 个人问候 是 检索 **IVR** 内的选项，而不是单独的模式/号码。
`tas_vm_mailbox`	始终	邮箱所有者的 MSISDN。对于留言，□□ 订阅者；对于检索，□□ 订阅者（邮箱所有者）。
`tas_vm_caller`	留言	主叫方的号码，与消息一起存储并用于通知文本。
`default_language`	可选	检索时用于口语“收到时间”的语言。根据订阅者在拨号计划中设置；未设置时回退到 `say_language` 配置值。

`socket` 应用使 FreeSWITCH 连接到 TAS 语音信箱监听器。IP 和端口 必须 匹配 `outbound_socket` 配置。参数是 FreeSWITCH 标准的：`async` 允许 TAS 在应用程序运行时接收事件，`full` 授予完全事件访问。

留言示例

将此放在 `bridge` 命令后，□便在桥接失败（无人接听/无法到达）时运行：

```
  <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 ${msisdn}" />
  <action application="set" data="default_language=fr"/>
  <action application="answer" />
  <action application="sleep" data="500"/>
  <!-- 将呼叫交给 TAS 语音信箱 IVR（留言）。TAS 录制消息，
       写入 mod_voicemail 行，并自行触发 MWI。
       ip:port 必须与 :tas voicemail.outbound_socket 在 runtime.exs 中匹
。 -->
  <action application="set" data="tas_vm_mode=deposit"/>
  <action application="set" data="tas_vm_mailbox=${msisdn}"/>
  <action application="set"
data="tas_vm_caller=${effective_caller_id_number}"/>
  <action application="socket" data="127.0.0.1:8084 async full"/>
```

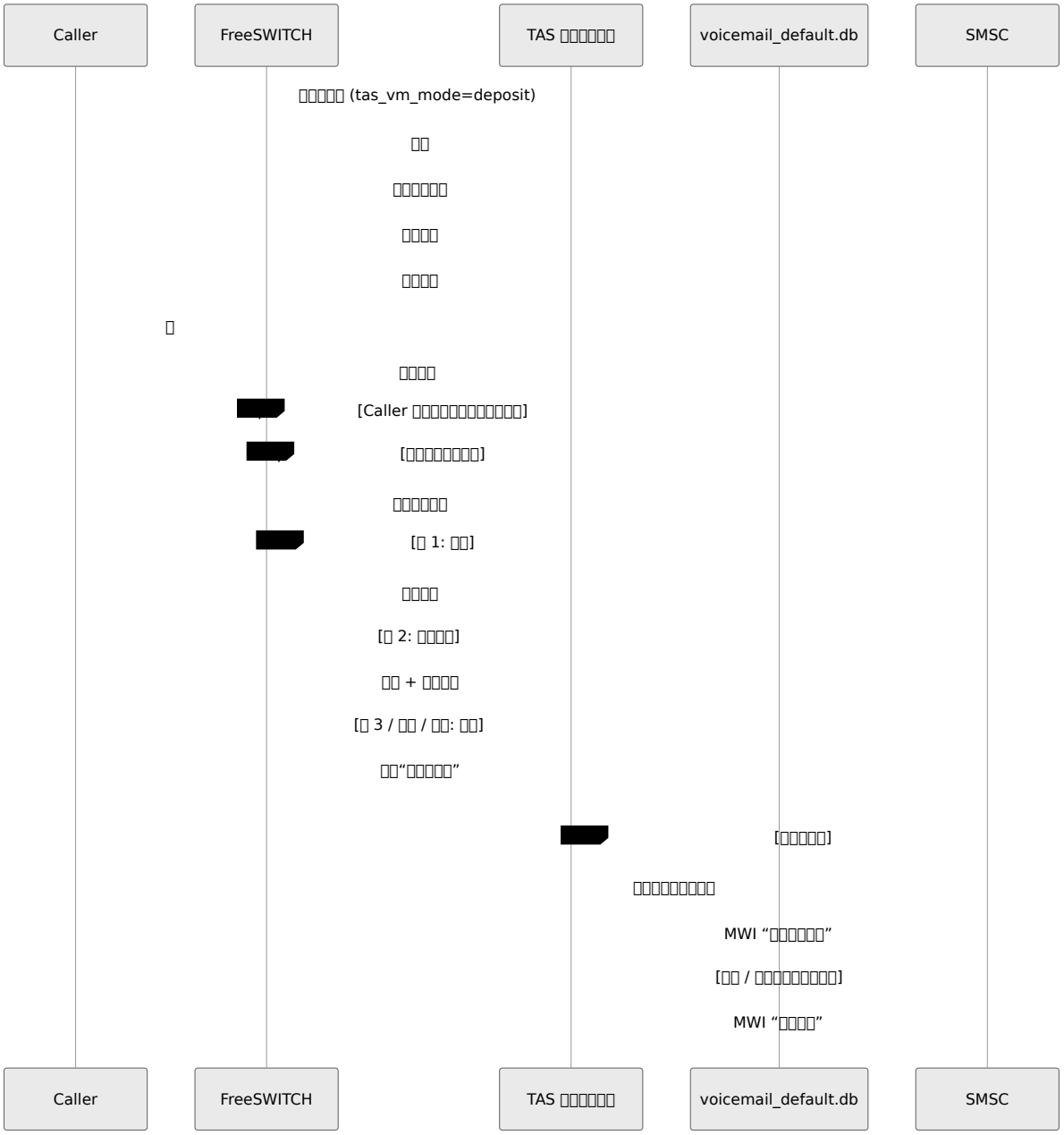
> 当呼叫被转接到语音信箱时，留言到 原始 被叫方，使用 History-Info 值（例如 `tas_vm_mailbox=${history_info_value}`），而不���语音信箱服务号码。有关 History-Info 处理，请参见 拨号计划配置。

检索示例

```
  <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" />
      <!-- 将呼叫交给 TAS 语音信箱 IVR（检索）。TAS 播放消息，
           处理听/删除/保存菜单，并清除 MWI。
           ip:port 必须与 :tas voicemail.outbound_socket 在
runtime.exs 中匹配。 -->
      <action application="set" data="tas_vm_mode=retrieve"/>
      <action application="set" data="tas_vm_mailbox=${msisdn}"/>
      <action application="socket" data="127.0.0.1:8084 async
full"/>
    </condition>
  </extension>
```

> 录制个人问候 不是 单独的访问号码。它是在检索 IVR 内提供的选项（请参见 个人问候）。上面的相同 `2222` 检索路由就是所需的全部。

□□□□



1. □□ □□□

2. □□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□ `prompts.deposit_greeting`□

3. □□□□□□`prompts.beep`□□

4. □□ □ `storage_dir`□□□□□□□#□□□□□□□□□□□□□□□□□□□□□□

5. □□□□□□`prompts.deposit_review_menu`□□□□□□□□□□□□□□□□□□□

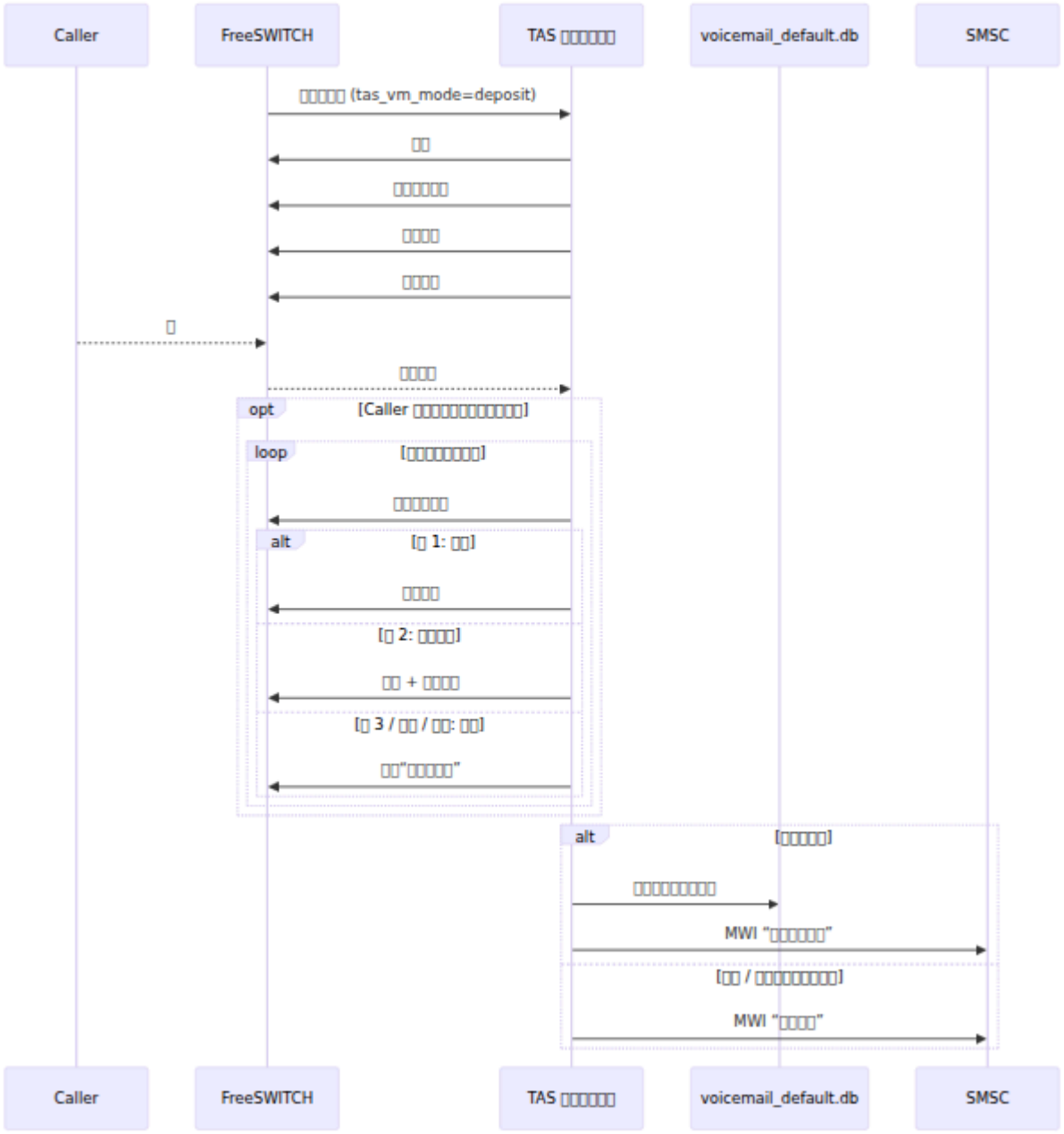
- **1**：收听录音，然后再次听到菜单。
- **2**：重新录制（哔声 + 录制），替换录音，然后再次听到菜单。
- **3**（或超时/挂断）：保存。消息被写入，并播放 `prompts.deposit_saved`（“消息已保存”）。

消息仅在保存时存储，因此重新录制会干净地覆盖早期录音。主叫方在留下可用消息后简单挂断仍然会保存该消息。如果他们不等待菜单，则不会丢失任何内容。过短的录音会被丢弃并视为未接来电。

6. 发送 **MWI** 通知。

通知在消息存储后立即发送。此顺序是故意的：消息必须在生成通知之前写入，便等待消息计数准确。

□□□□


Caller
FreeSWITCH
TAS □□□□□□□
voicemail_default.db
SMSC
□□□□□□ (tas_vm_mode=deposit)
□□
□□□□□□□
□□□□□
□□□□□
□
□□□□
opt
[Caller □□□□□□□□□□□□□]
loop
[□□□□□□□□□]
□□□□□□□
alt
[□ 1: □□]
□□□□□
[□ 2: □□□□]
□□ + □□□□
[□ 3 / □□ / □□: □□]
□□"□□□□□"
alt
[□□□□□□]
□□□□□□□□□□
MWI "□□□□□□□"
[□□ / □□□□□□□□□]
MWI "□□□□"
Caller
FreeSWITCH
TAS □□□□□□□
voicemail_default.db
SMSC

□□□□□□□□

键	操作	效果
1	再次收听	从头播放当前消息。
2	删除	删除消息及其录音，播放“已删除”提示，继续。
3	保存	保留消息（已标记为已读），播放“已保存”提示，继续。
4	转发	将录音转发到另一个邮箱（请参见 转发消息）。启用转发时可用。
（无 / 超时）	（无）	保留消息并继续。

播放消息会标记为 已读，因此不再计为新消息。一旦订阅者查看完他们的消息，邮箱将没有未读消息，MWI 将被清除。只有在整个集群中没有剩余未读消息时，**MWI** 才会被清除，因此在未读消息留下的���况下中途挂断不会错误地清除徽章。

邮箱菜单（问候）：当 个人问候 启用时，在消息之后（或如果邮箱为空则立即）订阅者会被提供一个邮箱选项，通过按配置的 `greetings.menu_key`（默认 `5`）录制他们的问候。这就是录制问候的方式。没有单独的访问号码。

“收到时间”公告结合了固定提示（`prompts.retrieve_received_at`，例如说“语音信箱收到时间”的音频文件）和由 FreeSWITCH 的 `say` 模块 动态口述 的日期/时间，使用消息存储的时间戳。口语语言是通道的 `default_language`（在拨号计划中按订阅者设置），未设置时回退到 `say_language` 配置值。这避免了需要为每个可能的日期/时间准备预录音频文件。

消息等待□示（**MWI**）

通知通过 MWI 通知器发送，该通知器向 SMSc 的 MWI API 发布。它根据邮箱中未读消息的数量选择“未接来电”或“语音信箱等待”消息。计数来自 TAS 自己的 `voicemail_msgs` 行（而不是 FreeSWITCH 的 `vm_boxcount`），在所有 TAS 之间 集群汇总（参见 多 TAS 语音信箱）。

事件	□示	使用的通知文本
留言，未留下消息	不活跃（未接来电）	`voicemail_notification_text.not_left`
留言，一条消息等待	活跃	`voicemail_notification_text.single_voicemail`
留言，多条消息等待	活跃	`voicemail_notification_text.multiple_voicemails`
检索完成	清除	`voicemail_notification_text.cleared`（或默认）

通知模板中可用的变量：

变量	可用性	描述
`caller`	全部	留下消息/未接来电的主叫方号码。
`day`	全部	月中的天数，□配置的 `timezone` 为准。
`month`	全部	月份编号，□配置的 `timezone` 为准。
`hour`	全部	小时（24小时制），□配置的 `timezone` 为准。
`minute`	全部	分钟，零填充，□配置的 `timezone` 为准。
`message_count`	`multiple_voicemails`	未读消息的数量。仅在计数 大于 **1** 时设置。

视觉语音信箱（**VVM**）

OmniTAS 还可□将每条留言呈现为手机上的 视觉语音信箱。手机显示主叫方、时间和播放控制，并通过 IMAP 流式传输音频。OmniTAS 不直接提供 VVM。它将录音推送到 **OmniSEP** 服务端点平台，该平台运行 OMTP/IMAP 服务器并通知手机。

VVM 在 MWI 和 SMS 通知之外 运行。当消息被保存时，OmniTAS 始终发送 MWI ��章和可读的 SMS，并将录音推送到 OmniSEP。推送是尽力而为的：失败不会影响保存的消息或 MWI/SMS 通知。

OmniSEP 仅为已配置 VVM 的订阅者存储录音。订阅者通过 OMTP 流从手机激活 VVM，这会在 OmniSEP 上创建账户。如果订阅者未配置，OmniSEP 将回复 `404` 并不存储任何内容，MWI/SMS 通知仍覆盖该订阅者。

要启用 VVM，请设置 `vvm` 配置并将 `deposit_url` 指向 OmniSEP 存储端点。有关详细信息，请参见 视觉语音信箱参数。

录音作为 OmniTAS 已经存储的 `.wav` 文件发送。OmniSEP 仅在可配置的保留期内保留音频，然后将其删除，因此它不是长期存储。有关此集成的 OmniSEP 方面，请参见 OmniSEP 视觉语音信箱文档。

消息存储

留言□两种方式存储，镜像 `mod_voicemail`：

- 音频文件 写入 `storage_dir`，按邮箱组织。
- 元数据（所有者、主叫方、时间戳、文件路径、长度、已读/未读状态）作为一行写入 FreeSWITCH 的 `voicemail_default.db` SQLite 数据库，存储在标准的 `voicemail_msgs` 表中。

由于模式与 `mod_voicemail` 匹配，`vm_boxcount`、语音信箱网页 UI 和语音信箱 REST API 都可□无更改地读取这些消息。您可□从控制面板的语音信箱选项卡查看语音信箱的使用情况和消息状态。

控制面板（集群感知）

控制面板的 语音信箱 选项卡列出整个集群中的每条消息，而不仅仅是本地节点：

- 同步与刷新 向每个对等 TAS 发散并合并结果；节点 列显示每条消息的所有者（此节点 或对等��点的 URL）。
- 播放 从其拥有的节点流式传输录音（本地文件或对等 HTTP），删除 路由到该所有者，因此远程消息可□完全从任何节点的 UI 中管理。
- 语音信箱设置 面板总结了活动配置：问候（和菜单键/强制首次）、转发、过期，□及联合模式 + 对等列表。

个人问候

默认情况下，拨打语音信箱的来电者会听到系统问候（`prompts.deposit_greeting`）。可选地，订阅者可□录制自己的 问候，然后在拨打他们的邮箱时播放该问候，而不是默认问候。

个人问候是 自愿的（`greetings.enabled`）。当禁用时，或者当订阅者未录制问候时，留言流程将回退到配置的默认问候，因此行为保持不变。

录制问候

问候录制是 检索 **IVR** 内的选项。没有单独的访问号码。当订阅者拨打语音信箱并启用问候时，在他们的消息之后（或如果邮箱为空则立即），他们会被提供一个邮箱菜单：按 `greetings.menu_key`（默认 `5`）播放录制提示、哔声、录制并存储问候。过短的录音会被丢弃，现有问候将保留。

问候□ `greeting.wav` 的形式存储在该邮箱的目录下，位于 `storage_dir` 中（与其消息录音并排）。重新录制会替换先前的问候；要删除它并恢复到默认，请参见 清除问候。

强制首次呼叫录制问候

通过设置 `greetings.force_first_time: true`（默认关闭），尚未录制问候的订阅者 在首次拨打语音信箱呼叫时必须录制一个问候，才能收听任何消息。如果录音过短，将重试（最多 `menu.tries` 次）。一旦存在问候，强制步骤将被跳过。保持关闭□使问候录制成为可选。

> 多 **TAS**：问候可能在一个 TAS 上录制，但留言可能落在另一个 TAS 上（请参见 多 TAS 语音信箱），因此留言流程会在集群范围内查找问候（它首先检查本地，然后向对等节点发散），然后回退到默认。为了保持这一点的明确性，录制问候会首先清除每个 **TAS** 上的任何现有问候，然后写入新的问候，因此在集群范围内仅存在一份副本（在不同节点上重新录制的订阅者不会留下过时的问候）。问候□与远程消息录音相同的方式获取和播放。

转发消息到另一个邮箱

在检索过程中，订阅者可□将他们正在收听的消息转发到另一个订阅者的邮箱。转发是 自愿的（`forwarding.enabled`）；当禁用时，按键 `4` 将被忽略。

1. 订阅者按 `4` 并被提示（`prompts.forward_enter_mailbox`）输入目标邮箱号码，□ `#` 终止。拨打的数字通过号码翻译（节点的 `:number_translate` 国家代码）运行，因此本地号码被规范化为与邮箱存储下的 MSISDN 形式相同。
2. 当前录音被 复制 到目标邮箱，作为一条新的未读消息。原始主叫方的详细信息被保留，`forwarded_by` 列记录转发的订阅者，因此目标可□看到是谁转发的。
3. 一个 MWI 通知 被触发到目标邮箱，就像正常留言一样。
4. 订阅者听到确认（`prompts.forward_done`）；未知/无效的目标播放 `prompts.forward_invalid` 并返回到消息菜单。

原始消息不受影响。转发不会影响订阅者自己的邮箱。

> 多 **TAS**：转发的副本成为由执行��发的节点拥有的正常消息（它在本地复制 WAV 并插入行），并且目标的等待计数在集群范围内计算，因此目标可□从他们检索的任何 TAS 中看到转发的消息。

消息保留与过期

语音信箱不会无限期累积。周期性清理程序会在消息超过配置的年龄后删除消息（行 和 录音），保持邮箱和磁盘使用量的限制。过期是 自愿的（`expiry.enabled`）；当禁用时，消息将保留，直到手动删除。

- 清理程序每 `expiry.sweep_interval_minutes` 运行一次。
- 当消息的年龄（从 `created_epoch`）超过 `expiry.max_age_days` 时，消息会过期。
- 可选地，已 读 消息可□通过 `expiry.read_max_age_days` 更早过期（例如，保留未读消息 30 天，但在 7 天后清除已收听的消息）。
- 过期消息会删除其 `voicemail_msgs` 行及其录音文件，因此网页 UI、REST API 和等待计数都反映删除。

如果过期消息清除了邮箱的最后一条未读消息，则该邮箱的 MWI 将被清除，□保持手机徽章的准确性。

> 多 **TAS**：每个节点在自己的时间表上清理 其自己的 存储。由于每条消息都有一个唯一的拥有节点（见下文），因此不需要跨节点协调。每个所有者过期其持有的消息。

多 **TAS** 语音信箱（集群邮箱）

当多个 OmniTAS 为相同的订阅者提供服务时，订阅者的呼叫可□在 任何 TAS 上落地，任何 时间。没有粘性。每个 TAS 将其记录到自己的本地存储中，因此在 TAS-A 上留下的消息默认对 TAS-B 不可见。集群使订阅者的邮箱在每个 TAS 上表现为 一个逻辑邮箱，因此等待计数反映在 □□ 节点上留下的消息，且从 □□ 节点检索可□列出、播放和删除 □ 条消息。

这通过 不 使用 Erlang 集群、分布式 Mnesia 或共享/中央数据库来实现。每个 TAS 保持自己的本地存储；节点在需要完整视图时只是 通过 **HTTP** 互相询问。集群是 自愿的。当 `cluster` 配置键缺失时，TAS 的行为与单个节点完全相同。

分区所有权，散播-聚合读取

每条语音信箱保留在其录制的地方。接收留言的节点是消息的 拥有节点，并保持元数据行和录音 WAV。没有任何内容被复制，也没有模式更改。所有权只是返回行的节点。当节点需要邮箱的完整视图时，它查询自己的存储并 向每个其他 **TAS** 发散 **HTTP** 请求，然后合并结果。由于每条消息只有一个所有者，因此标记为已读和删除始终路由回该单一所有者，因此 没��分布式写冲突。

节点发现

TAS 通过两种方式之一找到其对等节点（由 `cluster.discovery` 设置）；在这两种情况下，它 排除自身，因此它永远不会通过 HTTP 查询自身。

- 静态（`:static`）：完整的成员列表在 `runtime.exs` 中配置，并□相同方式部署到每个节点。每个节点通过 `self_id` 识别自己并丢弃该条目。可预测且无依赖性；添加/删除节点意味着在每个地方编辑配置。
- **DNS**（`:dns`）：节点解析返回每个 TAS 地址的 DNS 名称（A/AAAA 记录，每个 TAS 一个），并为每个地址构建对等 URL，丢弃自己的地址。较短的 DNS TTL 保持对等集的最新状态，而无需重新部署配置；SRV 记录可□携带每个目标的端口。

无论哪种方式，对等集���是基�� URL 的列表；所有下游内容都是相同的。

获取计数（发散）

等待消息计数通过 向每个其他 **TAS** 发散并求和 来计算。它在每次留言时运行，因为 MWI 文本依赖于此。

发散与每个请求的短时间限制并行运行，因此慢或无法访问的对等节点不会阻塞留言流程。

从任何节点检索（媒体流）

在检索时，处理节点构建 合并邮箱（其自己的消息加上每个对等节点的消息），按最旧的优先排序，每条消息标记其拥有节点。要播放 远程 消息，它通过 `GET /api/voicemail/media` 从所有者流式传输 WAV，将其写入本地 `storage_dir` 下的临时文件（便共置的 FreeSWITCH 可读取），播放后再删除临时文件。录音 惰性获，仅在消息即将播放时获取，因此提前挂断的订阅者不会触发进一步的传输。标记为已读和删除的请求路由到拥有节点（`POST /api/voicemail/mark_read` / `.../delete`）。

TAS 之间的 HTTP API

发散通过现有的 TAS HTTP 监听器运行。读取/单写端点仅在节点的 本地存储 上操作。集群范围的视图由 节点组装。

方法 & 路径	目的
`GET /api/voicemail/count?mailbox=<msisdn>`	此节点存储中的未读计数。
`GET /api/voicemail/all`	此节点存储中的每条消息（所有邮箱）。支持集群范围的控制面板列表。
`GET /api/voicemail/messages?mailbox=<msisdn>`	此节点的邮箱消息行。
`GET /api/voicemail/media?mailbox=<msisdn>&uuid=<uuid>`	流式传输此节点持有的录音 WAV。
`GET /api/voicemail/greeting?mailbox=<msisdn>`	流式传输此节点的邮箱个人问候（如果有）。
`POST /api/voicemail/mark_read` `{mailbox, uuid}`	在此节点的存储中标记一条消息为已读。
`POST /api/voicemail/delete` `{mailbox, uuid}`	从此节点删除一条消息（行 + 录音）。
`DELETE /api/voicemail/greeting?mailbox=<msisdn>`	清除个人问候（恢复到默认）。集群感知。请参见 管理。
`DELETE /api/voicemail/mailbox?mailbox=<msisdn>`	删除 所有 消息（行 + 录音）□供邮箱。集群感知。请参见 管理。

上述 `GET`/`POST` 端点仅在 本地存储 上操作。集群范围的视图由 □□ 节点组装。两个 `DELETE` 管理操作则向对等节点发散；对等请求携带 `?scope=local` 标志，因此发散请求在本地应用 而不是 再次发散，防止递归。

故障处理（尽力而为，绝不静默）

单个无法访问的对等节点不得破坏所有人的语音信箱：

- 计数：超时的对等节点贡献 `0`。这可能会短暂 低估 计数，但每个此类故障都会 记录并作为□标记录；绝不会静默吞没。
- 检索：来自无法访问的对等节点的消息会被省略（记录）；在会话中途失败的媒体获取会被记录并跳过，而不是掉线。

- □□□□

配置

所有语音信箱设置都位于:tas 应用程序的 voicemail 键下，位于 runtime.exs 中。

outbound_socket、record 和 menu 是可选的。它们在代码中默认显示，并在此处注释掉。语音提示由共享的:tas, :prompts 录音块生成（请参见 提示文件）；此处的 prompts 映射仅指向生成的路径。

```
config :tas,
  voicemail: %{
    timezone: "UTC",                                                    # 时间戳使用的时区

    say_language: "fr",                                                 # 口语日期/时间的回退语言
                                                                        # （通道 default_language 在每次呼叫中优先）
    storage_dir: "/usr/local/freeswitch/storage/voicemail", # 录音写入的位置

    # 可选 - 显示的默认值；省略以使用它们：
    # outbound_socket: %{listen_ip: "127.0.0.1", listen_port: 8084},
    # record: %{max_seconds: 120, silence_threshold: 200, silence_seconds: 5, min_seconds: 2},
    # menu: %{tries: 3, timeout_ms: 5000, terminators: "#"},

    # FreeSWITCH 播放的提示文件（绝对路径；$${base_dir} 不会在 ESL 上扩展）。
    # 语音提示由 :tas, :prompts 录音块生成；`beep` 是一个音调。
    prompts: %{
      deposit_greeting: "/usr/local/freeswitch/sounds/tas/vm/deposit_greeting.wav",
      deposit_review_menu: "/usr/local/freeswitch/sounds/tas/vm/deposit_review_menu.wav",
      deposit_saved: "/usr/local/freeswitch/sounds/tas/vm/deposit_saved.wav",
      beep: "tone_stream://%(500,0,800)",
      retrieve_received_at: "/usr/local/freeswitch/sounds/tas/vm/received_at.wav",
      retrieve_menu: "/usr/local/freeswitch/sounds/tas/vm/menu.wav",
      retrieve_no_messages: "/usr/local/freeswitch/sounds/tas/vm/no_messages.wav",
      retrieve_deleted: "/usr/local/freeswitch/sounds/tas/vm/deleted.wav",
      retrieve_saved: "/usr/local/freeswitch/sounds/tas/vm/saved.wav",
      retrieve_goodbye: "/usr/local/freeswitch/sounds/tas/vm/goodbye.wav",

      # 个人问候（仅在 greetings.enabled 时需要）
```

```
      retrieve_greeting_menu:
"/usr/local/freeswitch/sounds/tas/vm/greeting_menu.wav",
      greeting_record:
"/usr/local/freeswitch/sounds/tas/vm/greeting_record.wav",
      greeting_saved:
"/usr/local/freeswitch/sounds/tas/vm/greeting_saved.wav",

      # 转发（仅在 forwarding.enabled 时需要）
      forward_enter_mailbox:
"/usr/local/freeswitch/sounds/tas/vm/forward_enter_mailbox.wav",
      forward_done:
"/usr/local/freeswitch/sounds/tas/vm/forward_done.wav",
      forward_invalid:
"/usr/local/freeswitch/sounds/tas/vm/forward_invalid.wav"
    },

    # 可选 - 每个订阅者的问候，从检索 IVR 录制（按 menu_key）。
    # force_first_time 使没有问候的订阅者在听到消息之前录制一个。
    # greetings: %{enabled: true, menu_key: "5", force_first_time:
false},

    # 可选 - 从检索菜单（按键 4）将消息转发到另一个邮箱。
    # forwarding: %{enabled: true, menu_key: "4"},

    # 可选 - 自动过期旧消息。省略以保留消息，直到手动删除。
    # expiry: %{
    #   enabled: true,
    #   max_age_days: 30,           # 删除超过此天数的消息
    #   read_max_age_days: 7,       # 可选：更早清除已读消息
    #   sweep_interval_minutes: 60
    # },

    # 可选 - 多 TAS 集群。完全省略以运行单节点（无发散）。
    # cluster: %{
    #   discovery: :static,         # :static（节点列表）或 :dns（dns_name）
    #   self_id: "tas-a",
    #   nodes: [
    #     %{id: "tas-a", base_url: "https://10.8.82.60:9094"},
    #     %{id: "tas-b", base_url: "https://10.8.82.61:9094"}
    #   ],
    #   # discovery: :dns, dns_name: "omnitas-
vm.internal.example.com", scheme: "https", port: 9094,
    #   shared_secret: System.get_env("VM_CLUSTER_SECRET"),
    #   request_timeout_ms: 1500
```

```
    # },

    smsc: %{
      smsc_url: "https://10.80.14.219:8443",              # SMSc /
Omnimessage API 基础 URL
      source_msisdn: "2222"                                # 通知消息的发送
者
    },

    # 有关此部分中变量的使用，请参见上面的 MWI 表。
    voicemail_notification_text: %{
      not_left:
        "您有 1 个来自 <%= caller %> 的未接来电，时间为 <%= day %>/<%= month %>,
<%= hour %>:<%= minute %>",
      single_voicemail:
        "您有一条来自 <%= caller %> 的新语音消息，时间为 <%= day %>/<%= month %>,
<%= hour %>:<%= minute %>。要查看，请拨打 2222。",
      multiple_voicemails:
        "您有 <%= message_count %> 条新语音消息。要查看，请拨打 2222。"
    }
  }
```

视觉语音信箱参数

视觉语音信箱是单独的 `:tas` 键 `vvm`。这是可选的。省略它或将 `enabled: false` 设置为仅保留 MWI/SMS 行为。有关流程的详细信息，请参见 视觉语音信箱（VVM）。

```
config :tas, :vvm, %{
  enabled: true,                                          # 打开 VVM 推
送
  deposit_url: "https://10.80.14.219:9014/api/vvm/deposit", #
OmniSEP 存储端点
  timeout_ms: 5000                                        # 请求超时
}
```

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	打开或关闭视觉语音信箱推送。当为 `false` 时，不会将录音发送到 OmniSEP，仅使用 MWI/SMS 通知。
`deposit_url`	字符串	是（启用时）	-	OmniSEP 存储端点的完整 URL，通常为 `https://<omnisep-host>:9014/api/vvm/deposit`。`9014` 是 OmniSEP 的 HTTP 端口。
`timeout_ms`	整数	否	`5000`	等待 OmniSEP 存储响应的最大时间（毫秒为单位）。

□□

□□	□□	□□	
`timezone`	□□□	□	`"E`
`outbound_socket.listen_ip`	□□□	□	`"1`
`outbound_socket.listen_port`	□□	□	`80`
`say_language`	□□□	□	`"e`
`storage_dir`	□□□	□	-
`record.max_seconds`	□□	□	`12`
`record.silence_threshold`	□□	□	`20`
`record.silence_seconds`	□□	□	`5`

□□	□□	□□	
`record.min_seconds`	□□□	□	`2`
`menu.tries`	□□□	□	`3`
`menu.timeout_ms`	□□□	□	`50`
`menu.terminators`	□□□□	□	`"#`
`prompts.*`	□□□□	□	-
`smsc.smsc_url`	□□□□	□	-
`smsc.source_msisdn`	□□□□	□	-
`voicemail_notification_text.not_left`	□□□□	□	-
`voicemail_notification_text.single_voicemail`	□□□□	□	-

□□	□□	□□	
`voicemail_notification_text.multiple_voicemails`	□□□	□	-
`voicemail_notification_text.cleared`	□□□	□	□□[
`greetings.enabled`	□□□	□	`fa`
`greetings.menu_key`	□□□	□	`"5`
`greetings.force_first_time`	□□□	□	`fa`
`forwarding.enabled`	□□□	□	`fa`
`forwarding.menu_key`	□□□	□	`"4`
`expiry.enabled`	□□□	□	`fa`
`expiry.max_age_days`	□□	□	-

□□	□□	□□	
`expiry.read_max_age_days`	□□□	□	□□[
`expiry.sweep_interval_minutes`	□□□	□	`60`
`cluster`	□□□	□	□□[
`cluster.discovery`	□□□	□□□□□□□□ `cluster`□	-
`cluster.self_id`	□□□□	□ □`:static`□	-
`cluster.nodes`	□□□□□□	□ □`:static`□	-
`cluster.dns_name`	□□□□□	□□`:dns`□	-
`cluster.scheme`	□□□□□	□□`:dns`□	`"h`
`cluster.port`	□□□□	□□`:dns`□	`90`

参数	类型	必需	
`cluster.shared_secret`	字符串	是��如果设置了 `cluster`）	-
`cluster.request_timeout_ms`	整数	否	15

提示文件

提示值是 绝对 路径，指向 FreeSWITCH 可读取的音频文件（`$${base_dir}` 快捷方式 不 会扩展用于通过 ESL 播放的文件），或音调的 `tone_stream://` URI。

语音提示由共享的 `:tas, :prompts` 录音块 在启动时生成（与用于信用/公告提示的相同 OpenAI TTS 步骤）。在此处添加录音及其基本相对路径，然后将匹配的 `voicemail.prompts` 条目指向绝对路径（生成器在 `/usr/local/freeswitch` 下写入）。仅生成缺失的文件。

参数	描述
`deposit_greeting`	在留言期间播放给来电者的提示。
`beep`	在录制之前和在检索期间每条消息之前播放。`tone_stream://` URI（例如 `tone_stream://%(500,0,800)`）生成音调而无需文件。
`retrieve_received_at`	固定的“语音信箱收到时间”引导，后跟口语日期/时间。
`retrieve_menu`	选项菜单，例如“要再次收听，请按 1，要删除，请按 2，要保存，请按 3”。
`retrieve_no_messages`	当邮箱为空时播放。
`retrieve_deleted`	删除消息后播放的确认。
`retrieve_saved`	保存消息后播放的确认。
`retrieve_goodbye`	在检索会话结束时播放。
`retrieve_greeting_menu`	邮箱菜单，宣布“录制问候”选项（仅在 `greetings.enabled` 时）。
`greeting_record`	提示录制个人问候（仅在 `greetings.enabled` 时）。
`greeting_saved`	保存个人问候后的确认。
`forward_enter_mailbox`	提示输入目标邮箱 MSISDN（仅在 `forwarding.enabled` 时）。
`forward_done`	消息转发后的确认。
`forward_invalid`	当转发目标邮箱未知/无效时播放。

故障排除

呼叫到语音信箱立即掉线

症状：路由到语音信箱分机的呼叫在问候或菜单之前挂断。

可能原因：

- 拨号计划 `socket` 地址/端口与 `outbound_socket` 不匹配。
- 语音信箱监听器未运行。
- 防火墙阻止回环端口（在共置时不常见）。

解决方案：

1. 确认拨号计划 `socket` 操作使用与 `outbound_socket` 相同的 IP 和端口。
2. 验证 TAS 正在运行且语音信箱监听器已启动。
3. 确认 FreeSWITCH 可访问 `listen_ip`:`listen_port`。

消息未存储或在 UI 中不可见

症状：留言后消息未出现在 UI/REST API 中，或 `vm_boxcount` 返回 0。

可能原因：

- `mod_voicemail` 未加载，因此 `voicemail_msgs` 表 / `vm_boxcount` 不可用。
- 录音短于 `record.min_seconds`，被视为未接来电。
- `storage_dir` 不能被 FreeSWITCH 写入。

解决方案：

1. 确保 `mod_voicemail` 在 FreeSWITCH 中保持加载。
2. 检查录音长度与 `record.min_seconds` 的比较。
3. 验证 `storage_dir` 权限。

没有听到提示音

症状：在应播放问候、哔声或菜单的地方是静音。

可能原因：

- `prompts` 路径错误或 FreeSWITCH 无法读取。
- 使用了 `$${base_dir}` 的 `prompts` 路径，该路径不会在 ESL 上扩展。

解决方案：

1. 确认每个 `prompts` 值是 FreeSWITCH 可读取的文件的绝对路径。
2. 将任何 `$${base_dir}` 风格的路径替换为绝对路径。

消息等待指示未清除

症状：在收听完所有消息后，手机仍显示有消息等待。

可能原因：

- SMSc 拒绝清除，因为消息正文为空。
- 仍有未读消息（会话在听完所有消息之前结束）。

解决方案：

1. 确保配置了 `cleared`（或默认）通知正文。
2. 确认订阅者收听了所有消息；仅播放的消息被标记为已读。

计数过低/集群中缺少消息

症状：订阅者的徽章显示的消息少于留下的消息，或在另一个 TAS 上留下的消息在检索时未听到。

可能原因：

- 对等 TAS 关闭或无法访问，因此未对发散贡献任何内容（已记录 + 指标）。
- `cluster.shared_secret` 在节点之间不匹配，因此对等节点拒绝请求。
- 发现配置错误（错误的 `nodes` 列表，或 `dns_name` 未返回所有地址）。
- 防火墙阻止 TAS 节点之间的 HTTP 监听器端口。

解决方案：

1. 检查集群发散错误日志/指标，查看哪个对等节点失败。
2. 确认 `shared_secret` 在每个节点上相同，并且发现返回每个节点。
3. 验证每个节点可访问其他节点的 HTTP 监听器端口。

个人问候未播放

症状：来电者听到默认问候，即使订阅者录制了自己的问候。

可能原因：

- `greetings.enabled` 未设置。
- 在集群中，问候在不同节点上录制，而在留言时该节点无法访问。

解决方案：

1. 确认 `greetings.enabled` 为 true，且问候已保存（听到确认提示）。
2. 确认录制问候的节点在接收留言的节点可访问。

旧消息未被删除

症状：邮箱无限期保留消息/磁盘使用量增加。

可能原因：

- `expiry.enabled` 未设置，或 `max_age_days` 未设置。
- ��理间隔尚未经过。

解决方案：

1. 确认 `expiry.enabled` 为 true，且配置了 `max_age_days`。
2. 允许最多 `sweep_interval_minutes` 等待下次清理，然后检查日志。