EIR（设备身份注册）

概述

HSS 包含一个内置的 EIR（设备身份注册），为移动设备提供设备身份验证。EIR 验证 IMEI（国际移动设备身份）号码，以确定移动设备是否被授权、被盗或正在观察，然后允许网络访问。

关键功能

- **S13** 接口：通过 Diameter 协议进行设备身份检查
- **IMEI** 验证：使用 IMEI/IMEISV 验证设备身份
- 灵活匹配：基于正则表达式的 IMEI、IMEISV 和 IMSI 模式匹配
- 三级分类：支持白名单、黑名单和灰名单
- 可配置策略：可自定义未知设备的行为
- **REST API**：EIR 规则管理的完整 CRUD 操作

架构

Diameter 接口

接口	应用 **ID**	对端	目的
S13	16,777,252	MME/SGSN	设备身份验证

设备规则数据库

EIR 使用灵活的基于规则的匹配系统：

EIR_RULE		
int	id	PK
string	action	
string	regex	
timestamp	inserted_at	
timestamp	updated_at	

规则操作:

- `whitelist` - 允许设备
- `blacklist` - 阻止设备
- `greylist` - 监控设备

正则表达式模式: 与 IMEI、IMEISV 或 IMSI 匹配

设备状态值

状态	代码	意义	网络操作
白名单	0	设备已批准	允许网络访问
黑名单	1	设备被盗/被阻止	拒绝网络访问
灰名单	2	设备正在观察	允许但需监控

S13 接口

支持的操作

设备身份检查请求（**ECR**）/设备身份检查响应（**ECA**）

方向: MME/SGSN → HSS (EIR)

触发：MME 在附加或跟踪区域更新期间验证设备身份

请求 AVP：

Session-Id
Origin-Host, Origin-Realm Destination-Realm
Auth-Session-State
Terminal-Information
IMEI（15 位数字）
Software-Version（2 位数字，可选）
User-Name（IMSI，可选）
Vendor-Specific-Application-Id

EIR 操作：

1. 提取 IMEI、Software-Version（如果存在）和 IMSI（如果存在）
2. 如果提供了 IMSI：
验证订阅者是否存在并已启用
使用最后看到的信息更新订阅者状态
3. 按优先顺序尝试设备查找：
IMEISV 匹配（IMEI + Software-Version 连接）
IMEI 匹配（仅 IMEI）
IMSI 匹配（如果在请求中提供）
未知设备策略（配置的默认行为）
4. 返回设备状态

响应 AVP：

Session-Id（从请求中回显）
Result-Code: 2001（成功）
Equipment-Status: 0（白名单）/ 1（黑名单）/ 2（灰名单）

错误响应：

Experimental-Result: 5422（设备/订阅者未找到） Experimental-Result: 5012（一般错误）

设备匹配逻辑

优先顺序

EIR 使用级联查找策略以最大化匹配灵活性：

```
1. IMEISV（IMEI + Software-Version）
   ↓（如果没有匹配）
2. 仅 IMEI
   ↓（如果没有匹配）
3. IMSI（如果在请求中提供）
   ↓（如果没有匹配）
4. 未知设备政策
```

匹配算法

步骤 **1**: **IMEISV** 匹配

- 连接 IMEI + Software-Version: `"35979139461611" + "08" = "3597913946161108"`
- 测试所有 EIR 规则正则表达式模式
- 返回第一个匹配规则的操作（“whitelist”、“blacklist”、“greylist”）

步骤 **2**: **IMEI** 匹配（后备）

- 仅使用 IMEI: `"35979139461611"`
- 测试所有 EIR 规则正则表达式模式
- 返回第一个匹配规则的操作

步骤 **3**: **IMSI** 匹配（如果提供 IMSI，则后备）

- 使用请求中的 IMSI: `"999999876543210"`
- 测试所有 EIR 规则正则表达式模式
- 返回第一个匹配规则的操作
- 用例：阻止特定订阅者的所有设备

步骤 **4**: 未知设备政策（最终后备）

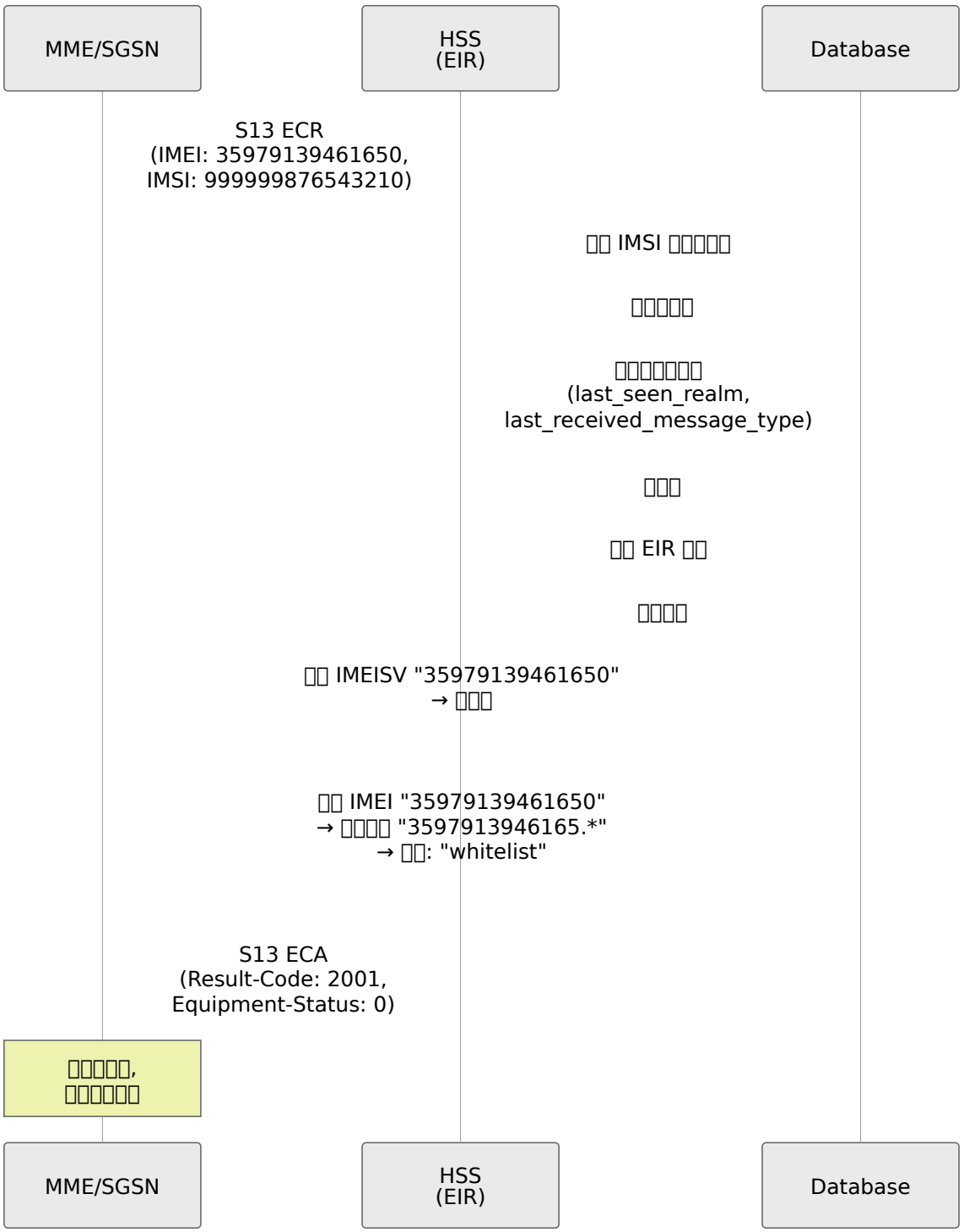
- 配置设置：`eir_unknown_equipment_behaviour`
- 选项：
  - `:whitelist` - 允许未知设备（宽松）
  - `:blacklist` - 阻止未知设备（严格）
  - `:greylist` - 观察未知设备（适中）
  - `:reject_unknown_equipment` - 返回错误 5422（严格）

正则表达式模式示例

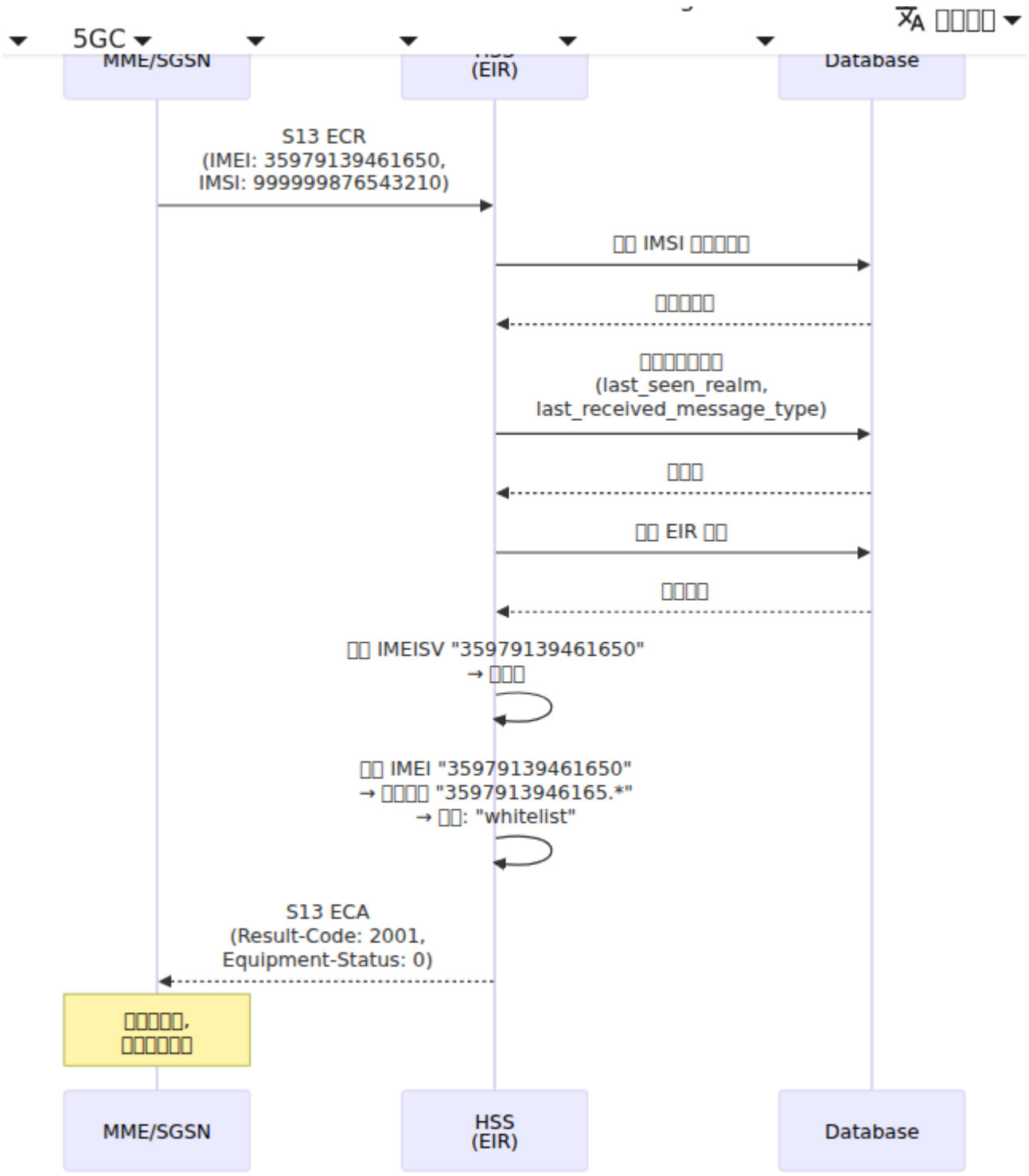
模式	匹配	用例
`"35979139461650"`	精确 IMEI	单个设备白名单/黑名单
`"3597913946165.*"`	IMEI 前缀通配符	制造商/型号范围
`"35979139461611 08"`	精确 IMEISV	特定设备与软件版本
`"999999876543210"`	IMSI	阻止该订阅者的所有设备
`"359791.*"`	TAC 通配符	整个设备类型分配

□□□□□

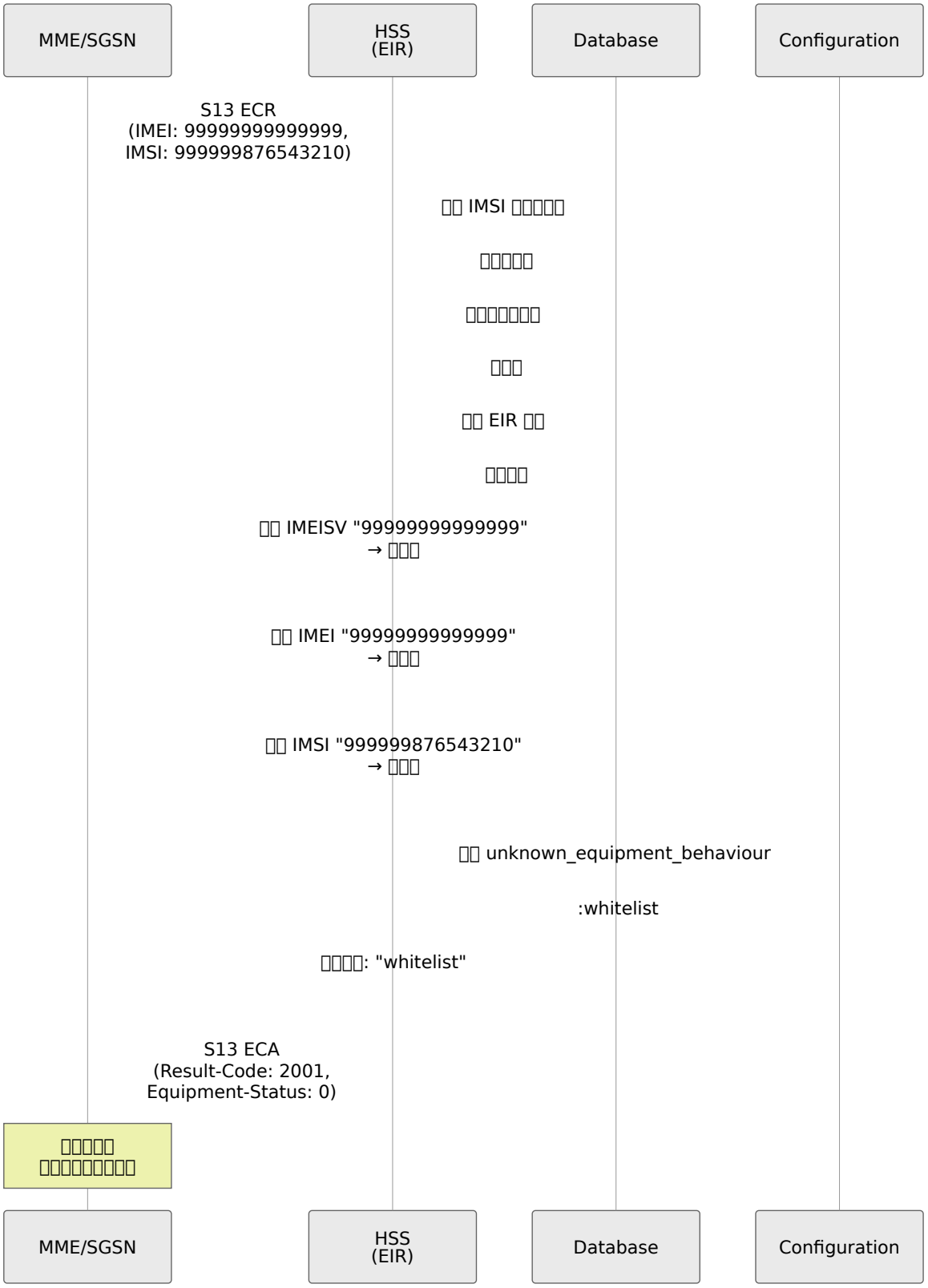
□□ 1□□□□□ - □□□□□ IMEI


MME/SGSN
HSS
(EIR)
Database
S13 ECR
(IMEI: 35979139461650,
IMSI: 999999876543210)
□□ IMSI □□□□□
□□□□□
□□□□□□□□
(last_seen_realm,
last_received_message_type)
□□□
□□ EIR □□
□□□□
□□ IMEISV "35979139461650"
→ □□□
□□ IMEI "35979139461650"
→ □□□□ "3597913946165.*"
→ □□: "whitelist"
S13 ECA
(Result-Code: 2001,
Equipment-Status: 0)
□□□□□,
□□□□□□□
MME/SGSN
HSS
(EIR)
Database

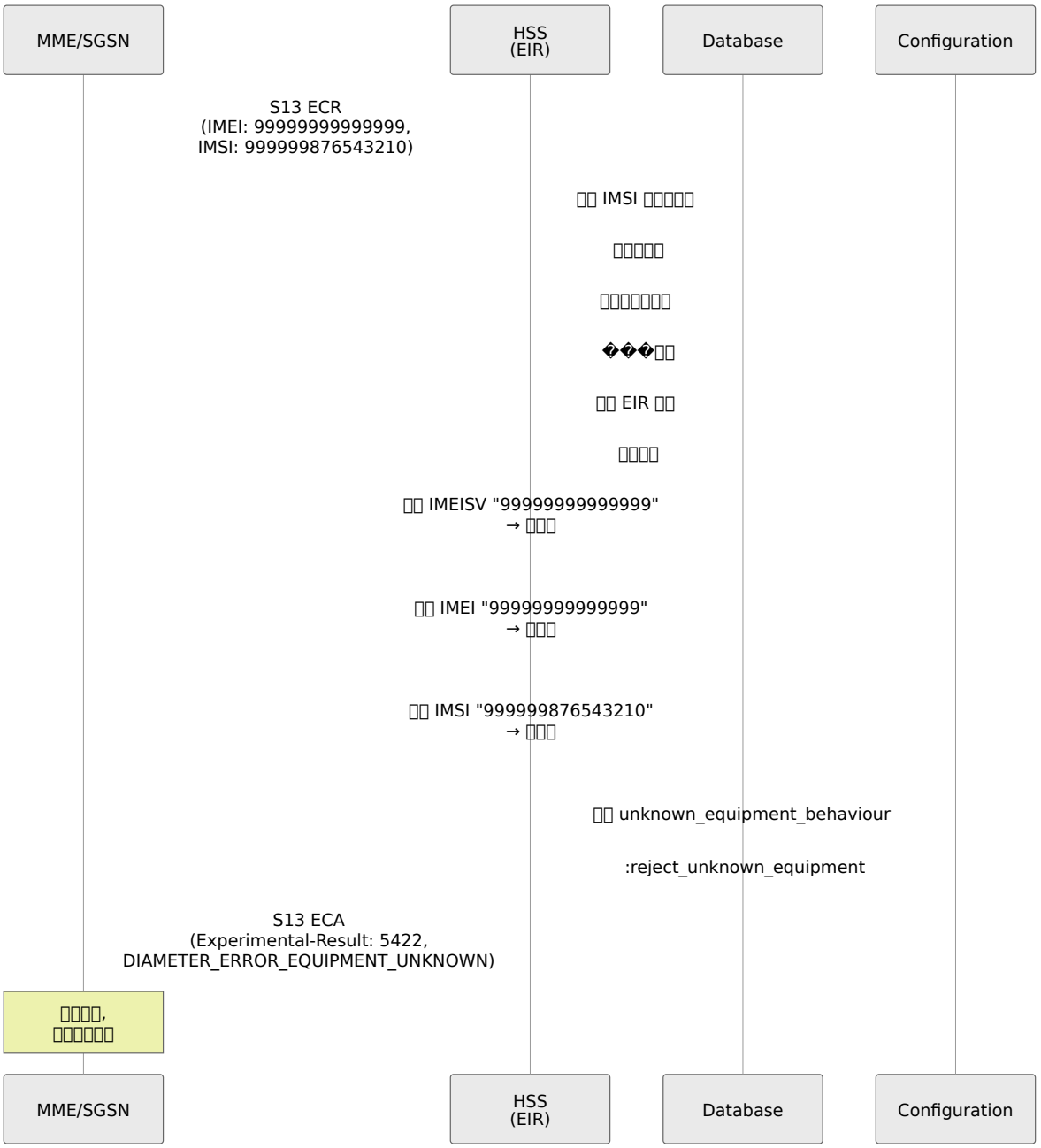
□□ 2□□□□□ - □□□ IMEI□□□□□□


5GC
MME/SGSN
(EIR)
Database
S13 ECR
(IMEI: 35979139461650,
IMSI: 999999876543210)
□□ IMSI □□□□□
□□□□□
□□□□□□□□
(last_seen_realm,
last_received_message_type)
□□□
□□ EIR □□
□□□□
□□ IMEISV "35979139461650"
→ □□□
□□ IMEI "35979139461650"
→ □□□□ "3597913946165.*"
→ □□: "whitelist"
S13 ECA
(Result-Code: 2001,
Equipment-Status: 0)
□□□□□,
□□□□□□
MME/SGSN
HSS
(EIR)
Database

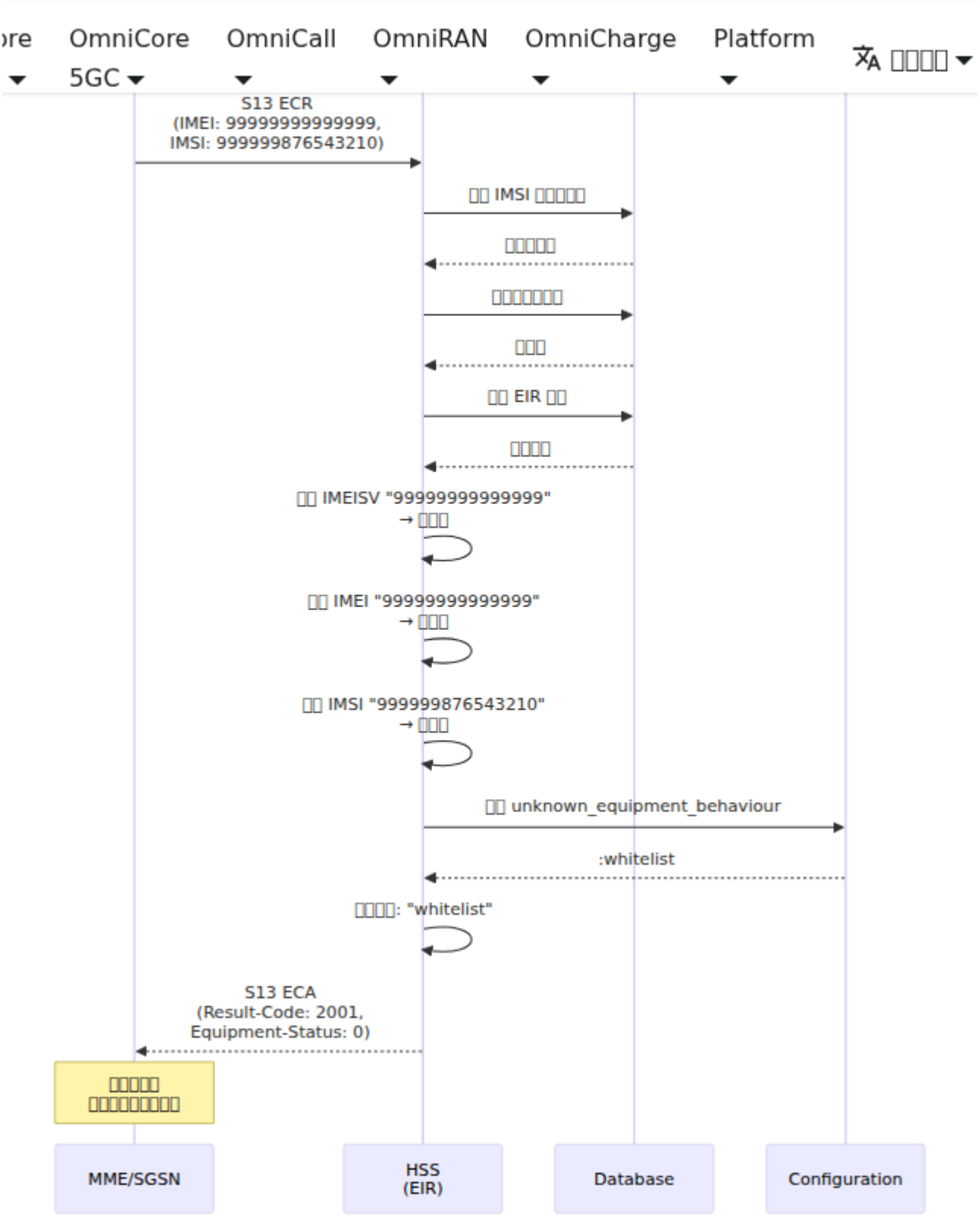
□□ 3□□□□□ - □□□□□□□□□□□□


MME/SGSN
HSS
(EIR)
Database
Configuration
S13 ECR
(IMEI: 99999999999999,
IMSI: 999999876543210)
□□ IMSI □□□□□
□□□□□
□□□□□□□□
□□□
□□ EIR □□
□□□□
□□ IMEISV "99999999999999"
→ □□□
□□ IMEI "99999999999999"
→ □□□
□□ IMSI "999999876543210"
→ □□□
□□ unknown_equipment_behaviour
:whitelist
□□□□: "whitelist"
S13 ECA
(Result-Code: 2001,
Equipment-Status: 0)
□□□□□
□□□□□□□□□□
MME/SGSN
HSS
(EIR)
Database
Configuration

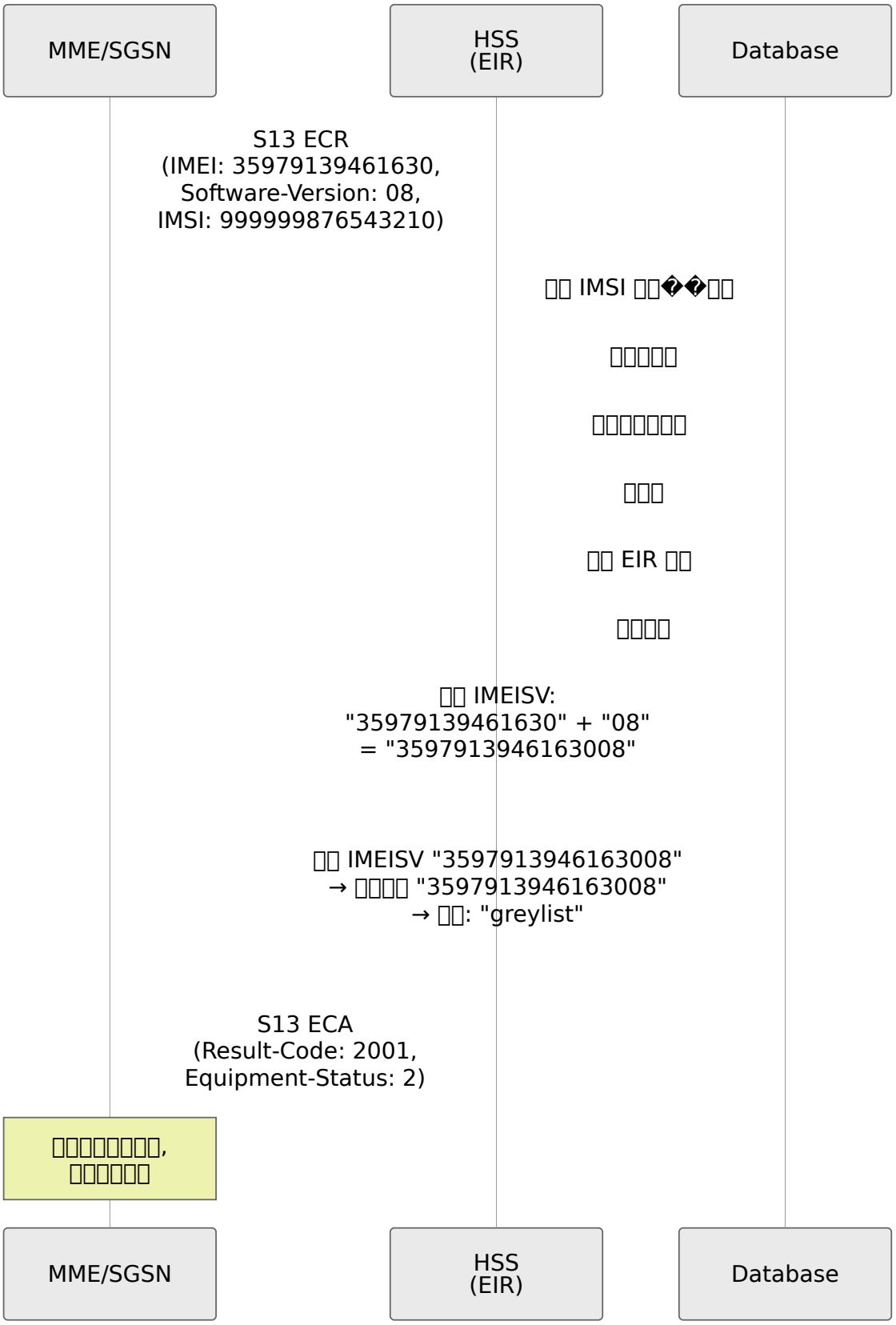
□□ 4□□□□□ - □□□□□□□□□□

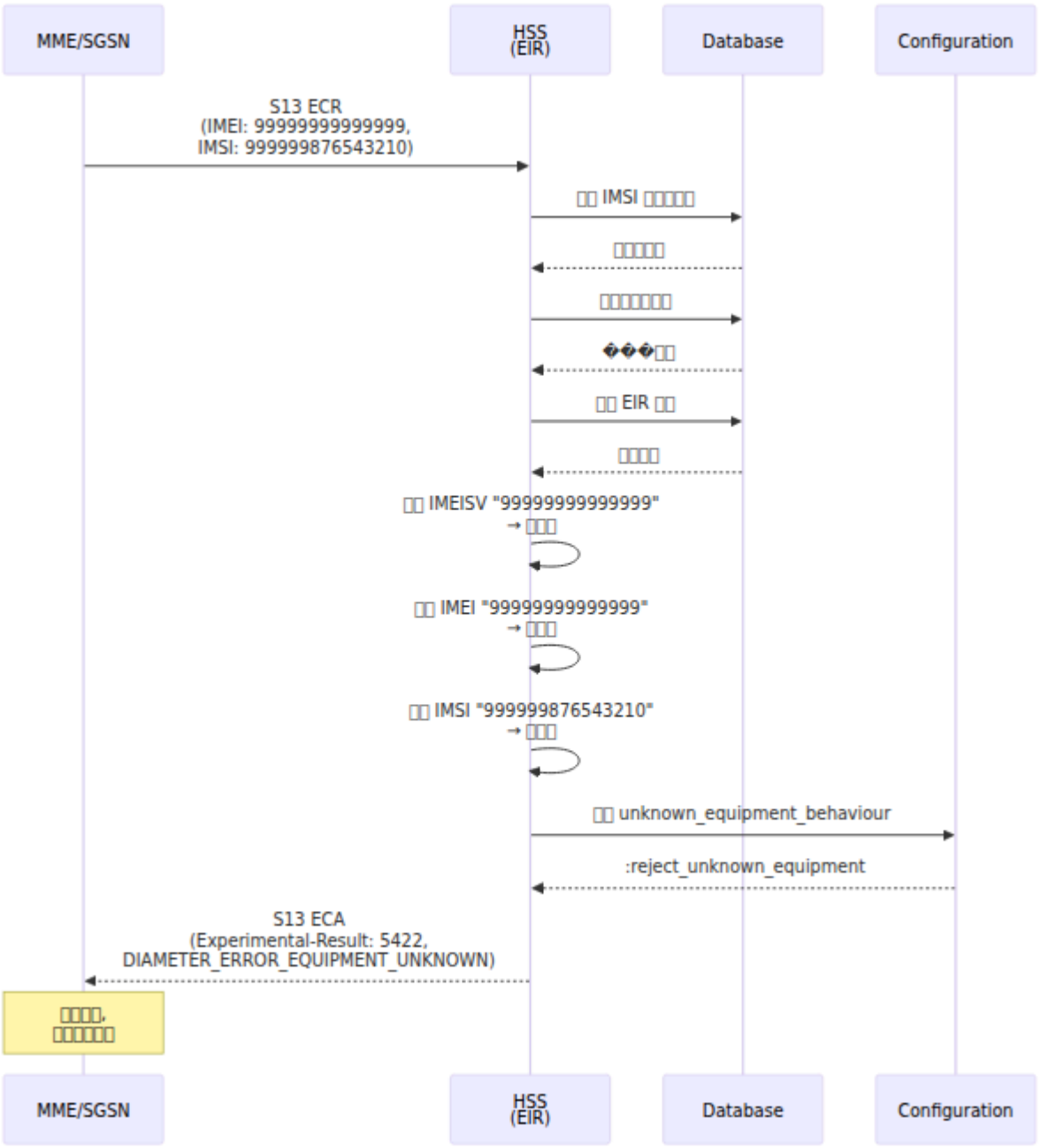

MME/SGSN
HSS
(EIR)
Database
Configuration
S13 ECR
(IMEI: 99999999999999,
IMSI: 999999876543210)
□□ IMSI □□□□□□
□□□□□□
□□□□□□□□
□□□□
□□ EIR □□
□□□□
□□ IMEISV "99999999999999"
→ □□□
□□ IMEI "99999999999999"
→ □□□
□□ IMSI "999999876543210"
→ □□□
□□ unknown_equipment_behaviour
:reject_unknown_equipment
S13 ECA
(Experimental-Result: 5422,
DIAMETER_ERROR_EQUIPMENT_UNKNOWN)
□□□□,
□□□□□□□
MME/SGSN
HSS
(EIR)
Database
Configuration

□□ 5□□□□□ - □□□□□


ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
□□□□
S13 ECR
(IMEI: 99999999999999,
IMSI: 999999876543210)
□□ IMSI □□□□□
□□□□□
□□□□□□□□
□□□
□□ EIR □□
□□□□
□□ IMEISV "99999999999999"
→ □□□
□□ IMEI "99999999999999"
→ □□□
□□ IMSI "999999876543210"
→ □□□
□□ unknown_equipment_behaviour
:whitelist
□□□□: "whitelist"
S13 ECA
(Result-Code: 2001,
Equipment-Status: 0)
□□□□□
□□□□□□□□□
MME/SGSN
HSS
(EIR)
Database
Configuration

□□ 6□□□□□ - IMEISV □□


MME/SGSN
HSS
(EIR)
Database
S13 ECR
(IMEI: 35979139461630,
Software-Version: 08,
IMSI: 999999876543210)
□□ IMSI □□□□□
□□□□□
□□□□□□□□
□□□
□□ EIR □□
□□□□
□□ IMEISV:
"35979139461630" + "08"
= "3597913946163008"
□□ IMEISV "3597913946163008"
→ □□□□ "3597913946163008"
→ □□: "greylist"
S13 ECA
(Result-Code: 2001,
Equipment-Status: 2)
□□□□□□□□□,
□□□□□□□
MME/SGSN
HSS
(EIR)
Database



流程 **7**：设备检查 **- IMSI** 阻止

REST API

EIR 规则管理

基本路径: `/api/eir/rule`

列出所有 **EIR** 规则

请求:

```
GET /api/eir/rule
```

响应（HTTP 200）：

```
{
  "data": [
    {
      "id": "e7f6a5b4-c3d2-4e1f-8a9b-0c1d2e3f4a5b",
      "action": "whitelist",
      "regex": "3597913946165.*",
      "inserted_at": "2025-01-15T10:30:00Z",
      "updated_at": "2025-01-15T10:30:00Z"
    },
    {
      "id": "f8a7b6c5-d4e3-4f2a-9b0c-1d2e3f4a5b6c",
      "action": "blacklist",
      "regex": "35979139461640",
      "inserted_at": "2025-01-16T14:20:00Z",
      "updated_at": "2025-01-16T14:20:00Z"
    }
  ]
}
```

获取特定 **EIR** 规则

请求:

```
GET /api/eir/rule/{id}
```

响应（HTTP 200）：

```
{
  "data": {
    "id": "e7f6a5b4-c3d2-4e1f-8a9b-0c1d2e3f4a5b",
    "action": "whitelist",
    "regex": "3597913946165.*"
  }
}
```

创建 **EIR** 规则

请求:

```
POST /api/eir/rule
Content-Type: application/json

{
  "action": "blacklist",
  "regex": "35979139461640"
}
```

验证:

- `action`: 必填，必须为 "whitelist"、"blacklist" 或 "greylist"
- `regex`: 必填，必须是有效的正则表达式模式，在��有规则中唯一

响应（HTTP 201）:

```
{
  "data": {
    "id": "0362b7ab-9bd6-4d8a-8b1a-2f4c6e8d0a1c",
    "action": "blacklist",
    "regex": "35979139461640"
  }
}
```

错误响应（HTTP 400）:

```
{
  "errors": {
    "regex": ["已被占用"]
  }
}
```

更新 **EIR** 规则（部分）

请求：

```
PATCH /api/eir/rule/{id}
Content-Type: application/json

{
  "action": "greylist"
}
```

响应（HTTP 200）：

```
{
  "data": {
    "id": "0362b7ab-9bd6-4d8a-8b1a-2f4c6e8d0a1c",
    "action": "greylist",
    "regex": "35979139461640"
  }
}
```

替换 **EIR** 规则

请求：

```
PUT /api/eir/rule/{id}
Content-Type: application/json

{
  "action": "whitelist",
  "regex": "359791394616.*"
}
```

响应（HTTP 200）：

```
{
  "data": {
    "id": "0362b7ab-9bd6-4d8a-8b1a-2f4c6e8d0a1c",
    "action": "whitelist",
    "regex": "359791394616.*"
  }
}
```

删除 **EIR** 规则

请求：

```
DELETE /api/eir/rule/{id}
```

响应（HTTP 204 无内容）

配置

Diameter 服务设置

S13 应用（`config/runtime.exs`）：

```
%{
  application_name: :s13,
  application_dictionary: :diameter_gen_3gpp_s13,
  vendor_specific_application_ids: [
    %{vendor_id: 10415, auth_application_id: 16_777_252}
  ]
}
```

未知设备行为

在 `config/runtime.exs` 中配置不匹配任何规则的设备的默认行为：

示例：

```
config :hss, :eir,
  unknown_equipment_behaviour: :whitelist
```

有效值：

- `:whitelist` - 允许未知设备（默认，宽松）
- `:blacklist` - 阻止未知设备（严格）
- `:greylist` - 监控未知设备（适中）
- `:reject_unknown_equipment` - 返回 Diameter 错误 5422（严格）

用例：

- 开发/测试：`:whitelist` - 允许所有设备
- 生产（宽松）：`:whitelist` - 仅阻止已知的坏设备
- 生产（适中）：`:greylist` - 记录未知设备以供审核
- 生产（严格）：`:reject_unknown_equipment` - 仅允许注册设备

错误处理

结果代码	类型	意义	原因
2001	成功	DIAMETER_SUCCESS	设备检查完成
5422	实验性	DIAMETER_ERROR_EQUIPMENT_UNKNOWN	未找到订阅者或未知设备被拒绝
5012	实验性	DIAMETER_ERROR_UNKNOWN	处理错误

用例

1. 被盗设备管理

场景：设备报告被盗

操作：

```
POST /api/eir/rule
{
  "action": "blacklist",
  "regex": "35979139461640"  # 精确 IMEI
}
```

结果：设备在下次附加时被拒绝网络访问

2. 制造商白名单

场景：预先批准整个设备型号范围

操作：

```
POST /api/eir/rule
{
  "action": "whitelist",
  "regex": "359791394.*"  # 制造商/型号的 TAC
}
```

结果：所有 TAC 范围内的设备被批准

3. 订阅者设备锁定

场景：阻止特定订阅者的所有设备（SIM 锁）

操作：

```
POST /api/eir/rule
{
  "action": "blacklist",
  "regex": "999999876543210"  # IMSI
}
```

结果：任何与此 SIM 一起使用的设备都被阻止

4. 测试设备灰名单

场景：在生产中监控测试设备

操作：

```
POST /api/eir/rule
{
  "action": "greylist",
  "regex": "35979139.*"  # 测试设备 TAC 范围
}
```

结果：设备被允许但标记为监控

5. 软件版本控制

场景：阻止特定易受攻击的固件版本

操作：

```
POST /api/eir/rule
{
  "action": "blacklist",
  "regex": "359791394616.*05"  # IMEI 范围 + 软件版本 05
}
```

结果：仅阻止 IMEI 范围内的软件版本为 "05" 的设备

实施细节

内部组件

EIR 功能通过多个内部模块实现：

- **S13** 协议处理器 - ECR/ECA 消息处理
- 设备匹配引擎 - 基于正则表达式的 IMEI/IMEISV/IMSI 匹配
- **EIR** 规则数据库 - 模式存储和查找
- **REST API** 控制器 - 规则管理端点

设备状态查找函数

设备状态查找遵循以下级联逻辑：

1. **IMEISV** 匹配：检查连接的 IMEI + 软件版本
2. **IMEI** 匹配：仅检查 IMEI
3. **IMSI** 匹配：检查 IMSI（如果提供）
4. 未知设备：应用配置的默认策略

可能的结果：

- `whitelist` - 设备被允许
- `blacklist` - 设备被阻止
- `greylist` - 设备正在观察
- `reject_unknown_equipment` - 严格拒绝

安全考虑

IMEI 隐私

IMEI 号码是敏感标识符。EIR：

- 默认情况下不以明文记录 IMEI 值
- 使用哈希数据库查找（如果配置）

- 限制 API 访问仅限经过身份验证的管理员

规则排序

EIR 规则按数据库顺序（按 ID）进行评估。对于冲突模式：

```
规则 1: 正则 "359791.*" 操作 "whitelist"  □广泛)
规则 2: 正则 "35979139461640" 操作 "blacklist"  □具体)
```

建议：在广泛通配符之前创建具体规则，以确保黑名单优先。

速率限制

考虑对以下内容实施速率限制：

- 来自不受信任对等方的 S13 ECR 请求
- REST API EIR 规则修改
- IMEI 查找查��以防止枚举攻击

相关文档

- Diameter 协议 - S13 协议规范
- API 参考 - 完整 API 文档
- 架构 - 整体 HSS 架构
- 操作指南 - 操作程序

附录：**IMEI** 结构

IMEI 格式（**15** 位数字）

```
35 9791 394616 1
│  │    │      └─ 校验位（Luhn 算法）
│  │    └─ 序列号（6 位数字）
│  └─ FAC（最终组装代码，4 位数字）
└─ TAC（类型分配代码，总共 8 位数字，包括 RBI）
   │  └─ RBI（报告主体标识符，2 位数字）
   └─ 制造商/型号（6 位数字）
```

IMEISV 格式（**16** 位数字）

```
35 9791 394616 1 08
│  │    │      │ └─ 软件版本（2 位数字）
└─ IMEI（15 位数字）
```

示例模式

IMEI/IMEISV	模式	匹配
359791394616108	`3597913946161.*`	所有 TAC+FAC+序列号为 359791394616* 的设备
359791394616140	`35979139461614.`	所有序列号为 359791394616141-9 的校验位
35979139461640	`35979139461640`	精确 IMEI 匹配
3597913946163008	`3597913946163008`	精确 IMEISV（IMEI + SV）匹配

PCRF（策略与计费规则功能）

概述

HSS 内含内置的 PCRF（策略与计费规则功能），为移动数据会话提供策略控制和计费规则。PCRF 控制 LTE 网络中默认和专用承载的服务质量（QoS）策略、带宽分配和计费规则。

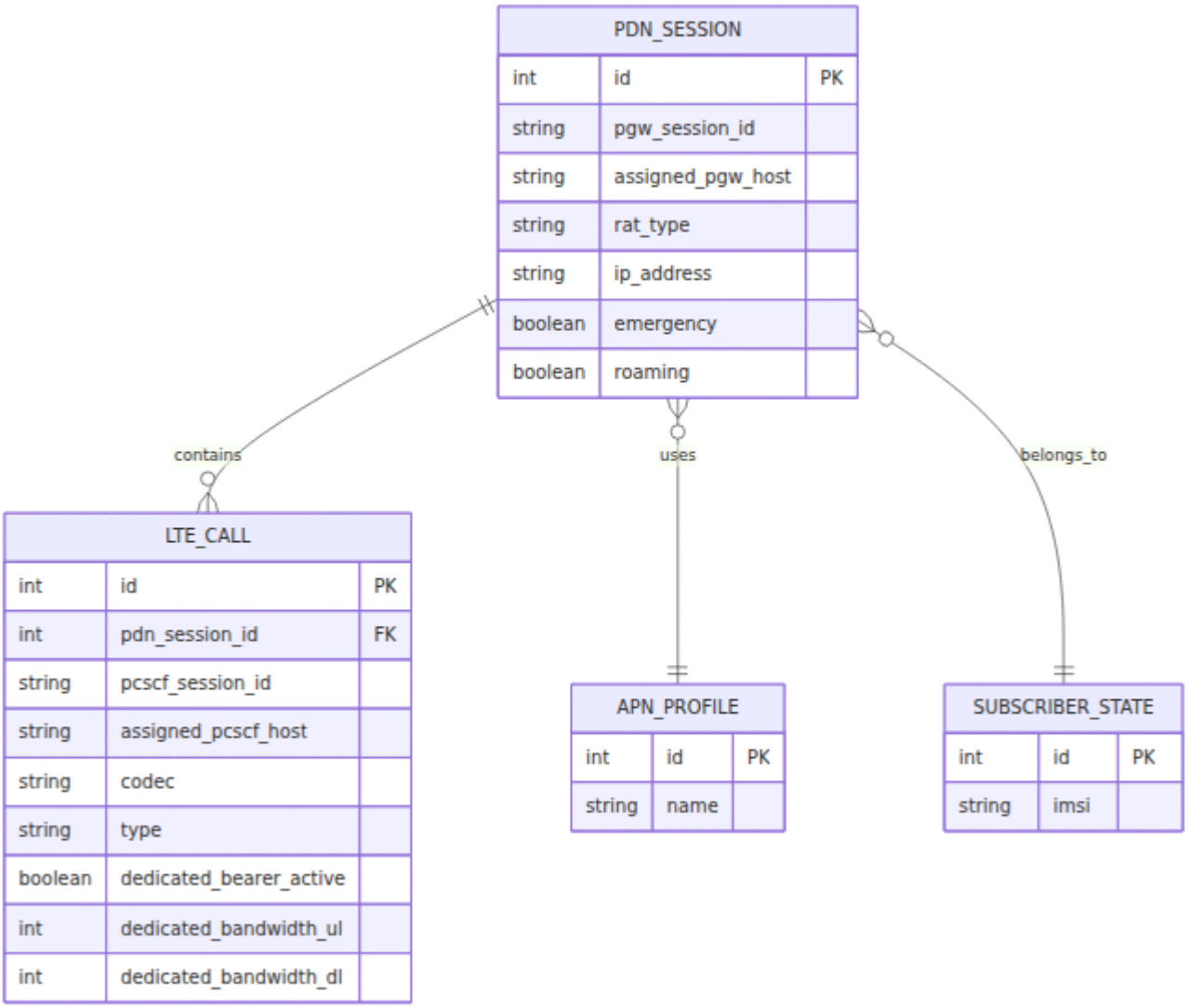
关键能力

- **Gx** 接口：PGW/PCEF（数据网络网关/策略与计费执行功能）的策略控制
- **Rx** 接口：IMS（IP 多媒体子系统）媒体流的授权和 QoS
- **Sy** 接口：针对外部 OCS 的用户支出限额控制 - 参见 Sy 接口
- 动态策略管理：通过重新认证请求（RAR）进行实时策略更新
- **VoLTE** 支持：为语音通话创建保证 QoS 的专用承载
- 计费规则：使用流量流模板（TFT）定义计费行为和速度配置
- **REST API**：程序化控制策略执行和规则管理

架构

Diameter 接口

接口	应用 **ID**	对等方	目的
Gx	16,777,238	PGW（PCEF）	PDN 会话管理、QoS 执行、计费规则
Rx	16,777,236	P-CSCF（AF）	IMS 媒体授权、带宽保留
Sy	16,777,302	OCS	用户支出限额状态、支出驱动的策略 - 参见 Sy 接口



会话状态管理

PCRF 维护活动 PDN 连接和 VoLTE 通话的会话状态:

Gx 接口

支持的操作

1. 信用控制请求 **-** 初始**CCR-I**)

触发: PGW 为用户创建新的 PDN 连接

请求 **AVPs**:

- Session-Id

- Origin-Host, Origin-Realm
- Subscription-Id (含 IMSI)
- Called-Station-Id (APN 名称)
- IP-CAN-Type (IP 连接接入网络类型)
- RAT-Type (无线接入技术)
- Framed-IP-Address (用户设备 IP 地址)

PCRF 操作:

1. 根据 IMSI 查找用户
2. 检索 APN 配置文件和 QoS 配置
3. 创建会话跟踪条目
4. 从 APN 配置文件构建 QoS 策略

响应 **AVPs**:

- Result-Code: 2001 (DIAMETER_SUCCESS)
- QoS-Information (APN 聚合带宽限制)
- Default-EPS-Bearer-QoS (QCI, ARP, 优先级)
- Bearer-Control-Mode
- 在线/离线 (会话级默认计费方法 - 参见 会话级在线/离线计费)

2. 信用控制请求 **-** 更新 **CCR-U**)

触发: PGW 报告会话更改 (位置更新、RAT 更改等)

PCRF 操作:

1. 根据会话 ID 查找现有会话
2. 更新会话参数 (RAT 类型、位置等)
3. 如有需要，返回更新的策略

响应: Result-Code 2001，附带可选的策略更新

3. 信用控制请求 **-** 终止 **CCR-T**)

触发: PGW 终止 PDN 连接

PCRF 操作:

1. 根据会话 ID 查找会话
2. 删除会话及相关通话记录
3. 确认终止

响应: Result-Code 2001

会话级在线/离线计费

在 Gx CCA-I 顶层发送的 `Online`（AVP 1009）和 `Offline`（AVP 1008） AVPs 设置整个 **IP-CAN** 会话的默认计费方法（默认承载），如 3GPP TS 29.212 所定义。这与每条规则的 `Online`/`Offline` 不同，后者仅管理由该特定规则匹配的流量。

默认情况下，PCRF 回显 PGW/PCEF 在 CCR-I 中宣传的任何值。APN QoS 配置文件可以 锁定 会话级计费方法，覆盖回显。这由 APN QoS 配置文件上的两个可选字段控制:

字段	类型	**AVP**	影响
`online_charging_enabled`	布尔	`Online`（1009）	`true` → `ENABLE_ONLINE (1)`，`false` → `DISABLE_ONLINE (0)`
`offline_charging_enabled`	布尔	`Offline`（1008）	`true` → `ENABLE_OFFLINE (1)`，`false` → `DISABLE_OFFLINE (0)`

每个字段独立解析:

计费方法 **AVP** 值（TS 29.212）：

AVP	代码	值	名称	含义
`Online`	1009	0	DISABLE_ONLINE	在线计费（OCS / Gy）未应用于会话
`Online`	1009	1	ENABLE_ONLINE	应用在线计费 - 必须在流量流动之前授予信用
`Offline`	1008	0	DISABLE_OFFLINE	离线计费（CDR / Gz）未应用于会话
`Offline`	1008	1	ENABLE_OFFLINE	应用离线计费 - 用于计费的使用记录

工作原理：当 `online_charging_enabled`/`offline_charging_enabled` 保持未设置（每个现有配置文件的默认值）时，行为不变 - PCRF 反映 PGW 的请求，因此网络的现有计费姿态得以保留。设置任一字段强制在使用该配置文件的任何 APN 的每个 CCR-I 上应用该计费模式，无论 PGW 宣传的内容如何。

用例：在预付费 APN 上锁定在线计费（因此 PGW 必须咨询 OCS，即使它请求离线），或在后付费/物联网 APN 上强制离线计费以确保 CDR 生成。由于设置位于 APN QoS 配置文件上，因此适用于引用该配置文件的每个 APN。

> 注意：这仅控制 □□□□□。每流量计费仍由单个 计费规则 上的 `online_charging_enabled`/`offline_charging_enabled` 字段管理，这些字段出现在 Charging-Rule-Definition AVPs 内。

4. 重新认证请求□**RAR**）

方向：PCRF → PGW（HSS 发起）

触发：

- IMS 通话设置（Rx AAR 触发 Gx RAR）
- IMS 通话拆除（Rx STR 触发 Gx RAR）
- 通过 REST API 手动重新认证

RAR AVPs：

- Session-Id（PGW 会话 ID）
- Auth-Application-Id: 16,777,238
- Re-Auth-Request-Type（0 = 仅授权）
- Charging-Rule-Install/Remove
- QoS-Information（用于专用承载）

PGW 操作：根据计费规则创建/修改/删除专用承载

计费规则和流量流模板

PCRF 支持使用流量流模板（TFT）定义计费规则，以控制：

- 服务特定计费 - 视频、游戏、社交媒体等不同费率
- 速度配置 - 限制或优先处理匹配特定模式的流量
- 基于使用的策略 - 根据流量类型或目的地应用不同的 QoS

计费规则可以：

- 根据应用检测通过 Gx RAR 动态安装

- 预定义并由特定条件（时间、地点、配额）触发
- 使用数据□过滤规则（5 元组：协议、源/目标 IP、源/目标端口）与 TFT 关联

常见用例：

- 零计费 - 对特定服务（Spotify、WhatsApp、Facebook）无限制访问，而不消耗数据配额
- 配额后访问 - 即使用户用尽数据配额，也允许自助服务门户和支持网站访问
- 分层速度 - 高速用于优质服务，标准内容限制速度
- 基于时间的策略 - 非高峰期无限制流媒体，高峰期优先处理
- 漫游政策 - 国际与国内数据���用的不同计费
- 企业 **SLA** - 对业务关键应用保证 QoS

QoS 策略结构

默认承载 **QoS**（来自 APN 配置文件）：

```
{
  "QoS-Class-Identifier": 9,          // QCI□9 = 默认承载)
  "APN-Aggregate-Max-Bitrate-UL": 50000,  // kbps
  "APN-Aggregate-Max-Bitrate-DL": 100000, // kbps
  "Allocation-Retention-Priority": {
    "Priority-Level": 8,
    "Pre-emption-Capability": 1,      // 可能抢占
    "Pre-emption-Vulnerability": 1    // 可能被抢占
  }
}
```

专用承载 **QoS**（用于 VoLTE）：

```
{
  "QoS-Class-Identifier": 1,          // QCI 1 = 对话语音
  "Max-Requested-Bandwidth-UL": 128000,   // bps
  "Max-Requested-Bandwidth-DL": 128000,   // bps
  "Guaranteed-Bitrate-UL": 128000,
  "Guaranteed-Bitrate-DL": 128000
}
```

Rx 接口

支持的操作

1. AA 请求（AAR）/ AA 答复（AAA）

触发：P-CSCF 请求 IMS 媒体会话的授权（VoLTE 通话设置）

请求 **AVPs**：

- Session-Id（P-CSCF 会话标识符）
- Subscription-Id（IMSI 或 SIP URI）
- 媒体组件描述
  - 媒体类型（音频、视频）
  - Max-Requested-Bandwidth-UL/DL
  - Codec-Data
  - Flow-Description（5 元组数据过滤器）
- AF-Application-Identifier

PCRF 操作：

1. 根据 IMSI 或 SIP URI 查找用户
2. 查找活动 IMS 会话
3. 提取媒体参数（编解码器、带宽、流规则）
4. 创建通话跟踪条目
5. 触发 **Gx RAR** 到 **PGW** 创建专用承载
6. 等待 Gx RAA 响应
7. 返回 Rx AAA 及授权结果

响应 **AVPs**：

- Result-Code: 2001（成功）或 5063（服务未授权）

紧急（**SOS**）呼叫，包括未认证/无 **SIM** 卡的呼叫者

紧急呼叫（`urn:service:sos`）即使在呼叫者不是已知用户（无 SIM 卡设备或没有凭证的漫游用户）时也会被授权。这是为了确保紧急呼叫始终获得正确的 QoS（TS 23.167）。

PCRF 如何识别和授权紧急 AAR:

1. 紧急检测。如果 任一 `Subscription-Id` 是含 `sos` 的 SIP URI，或 AAR 的 `Framed-IP-Address` 解析为已由 Gx CCR-I 创建的紧急 PDN 会话（Gx 侧通过 APN / `Called-Station-Id` 含 `sos` 来标记紧急）。第二条路径允许 匿名 紧急 INVITE（没有可用身份）仍然被授权。
2. 用户解析。通过 `Subscription-Id` 查找，否则通过 `Framed-IP-Address` 查找。紧急 孤儿 **PDN**（由 Gx 为未知用户创建）没有用户（`subscriber_state_id: nil`）；AAR 然后走孤儿路径而不是正常用户路径。
3. **SOS APN** 配置文件。专用承载 QoS 从 SOS APN 配置文件中获取（见下文“内置默认 SOS APN 配置文件”）。
4. Gx RAR 在（孤儿）PDN 上安装专用语音承载（QCI 1），AAA 返回 `2001`。

> 注意：只有当 `Framed-IP-Address` 映射到 Gx 已经 作为紧急 创建的 PDN 时，才会产生授权。来自未注册 UE 的非紧急请求仍然在 P-CSCF 被拒绝（它永远不会作为紧急 AAR 到达 PCRF）。

内置默认 **SOS APN** 配置文件。紧急授权不再依赖于运营商在数据库中预配置 `sos` APN 配置文件。PCRF 将 SOS APN 配置文件解析为：

- 如果存在，则为运营商定义的 APN 配置文件，其 APN 标识符为 `sos`；否则
- 由 OmniHSS 提供的 内置默认 SOS APN 配置文件。

内置默认配置文件是一个临时配置文件（没有数据库行，孤儿 PDN 上的 `apn_profile_id: nil` - 该列是可空的），具有紧急 QoS:

参数	值
默认 SOS 承载 QCI	5
专用语音承载 QCI	1（ARP 优先级 2）
分配保留优先级	1
抢占能力	启用
抢占脆弱性	禁用
APN-AMBR 上行/下行	10000 kbps

要覆盖这些值（或通过 PCRF 配置文件设置每个运营商的计费规则），请在数据库中创建 `sos` APN 配置文件 - 该配置文件优先于内置默认配置文件。

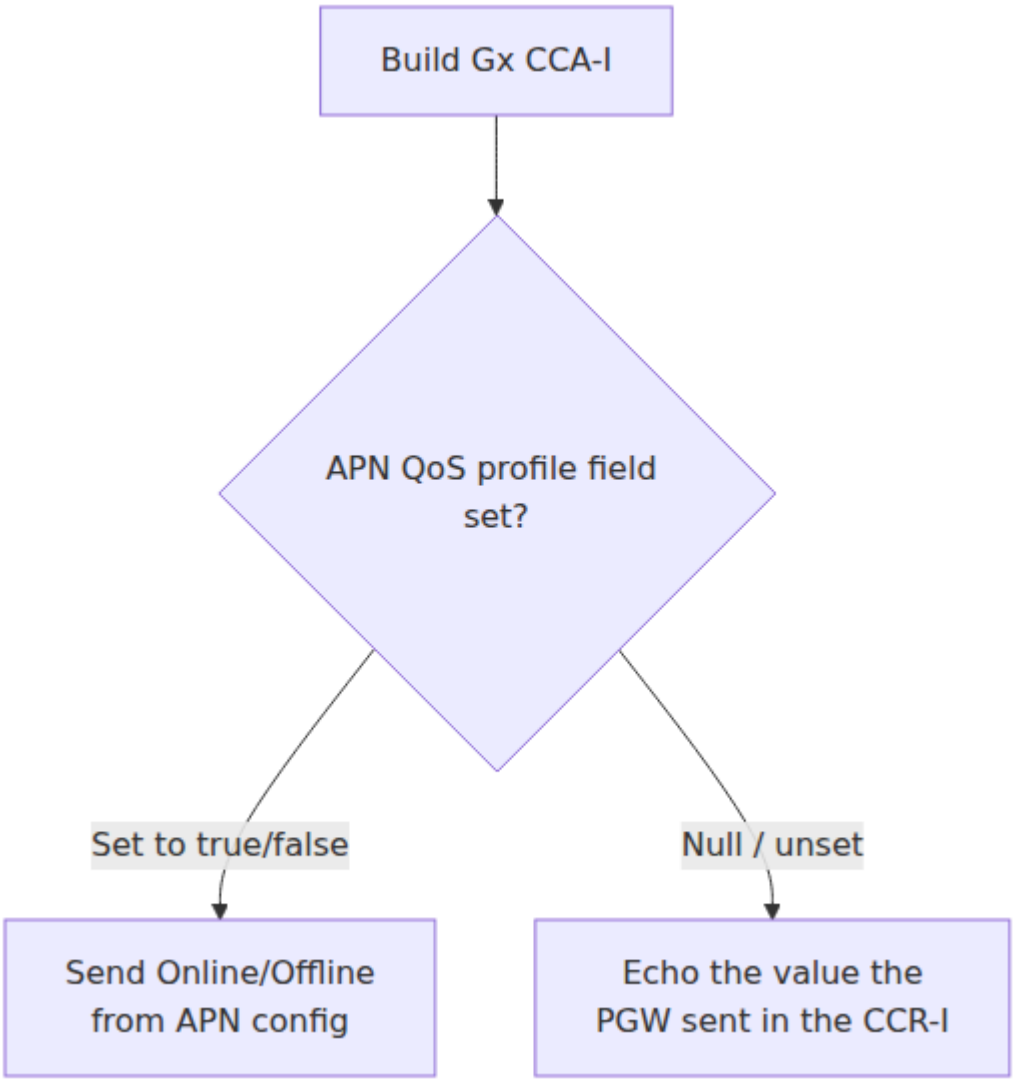
2. 会话终止请求（**STR**） **/** 会话终止答复（**STA**）

触发: P-CSCF 终止 IMS 会话（通话挂断）

PCRF 操作:

1. 根据 P-CSCF 会话 ID 查找通话会话
2. 触发 **Gx RAR** 到 **PGW** 删除专用承载
3. 删除通话跟踪条目
4. 返回 STA 确认

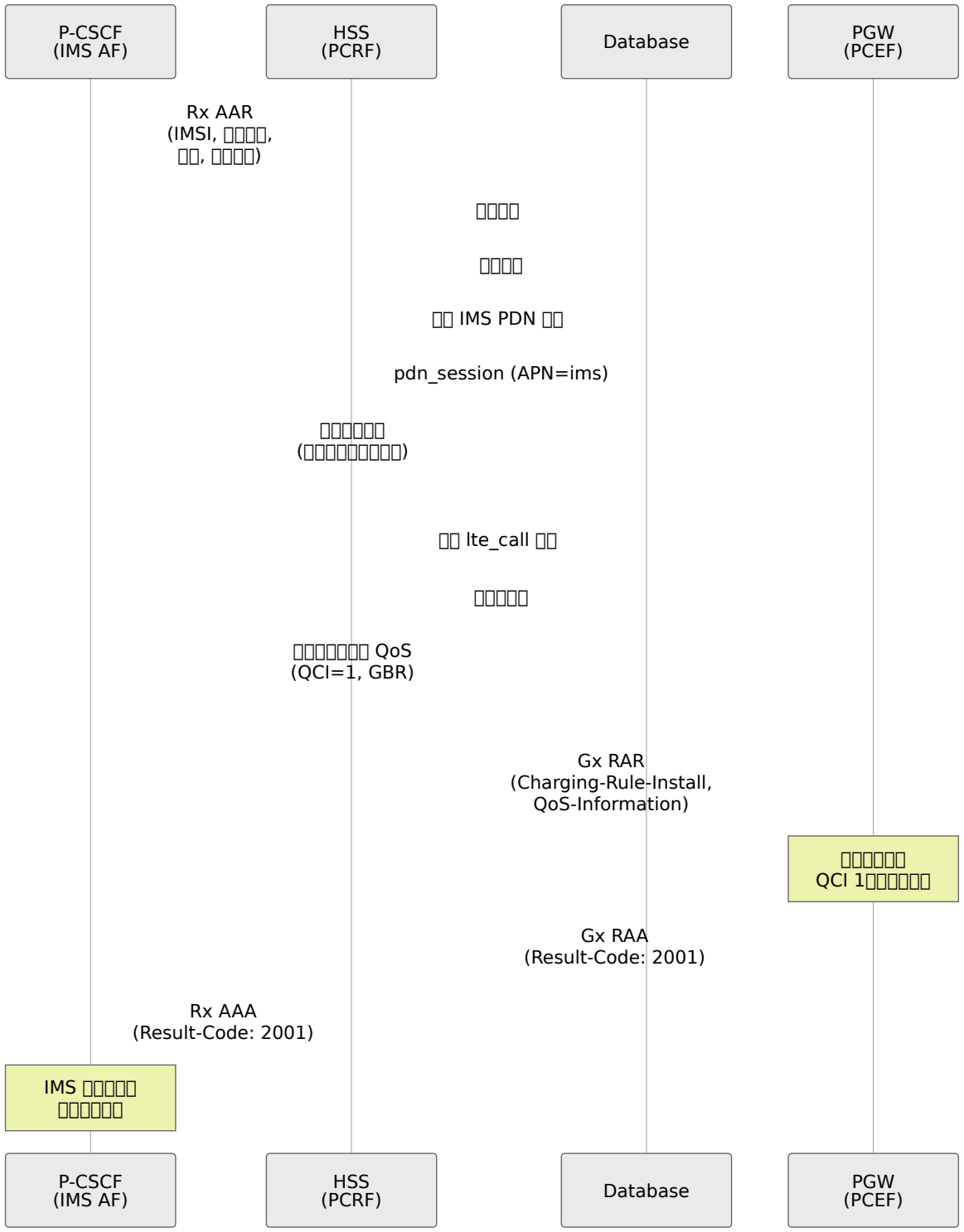
响应: Result-Code 2001



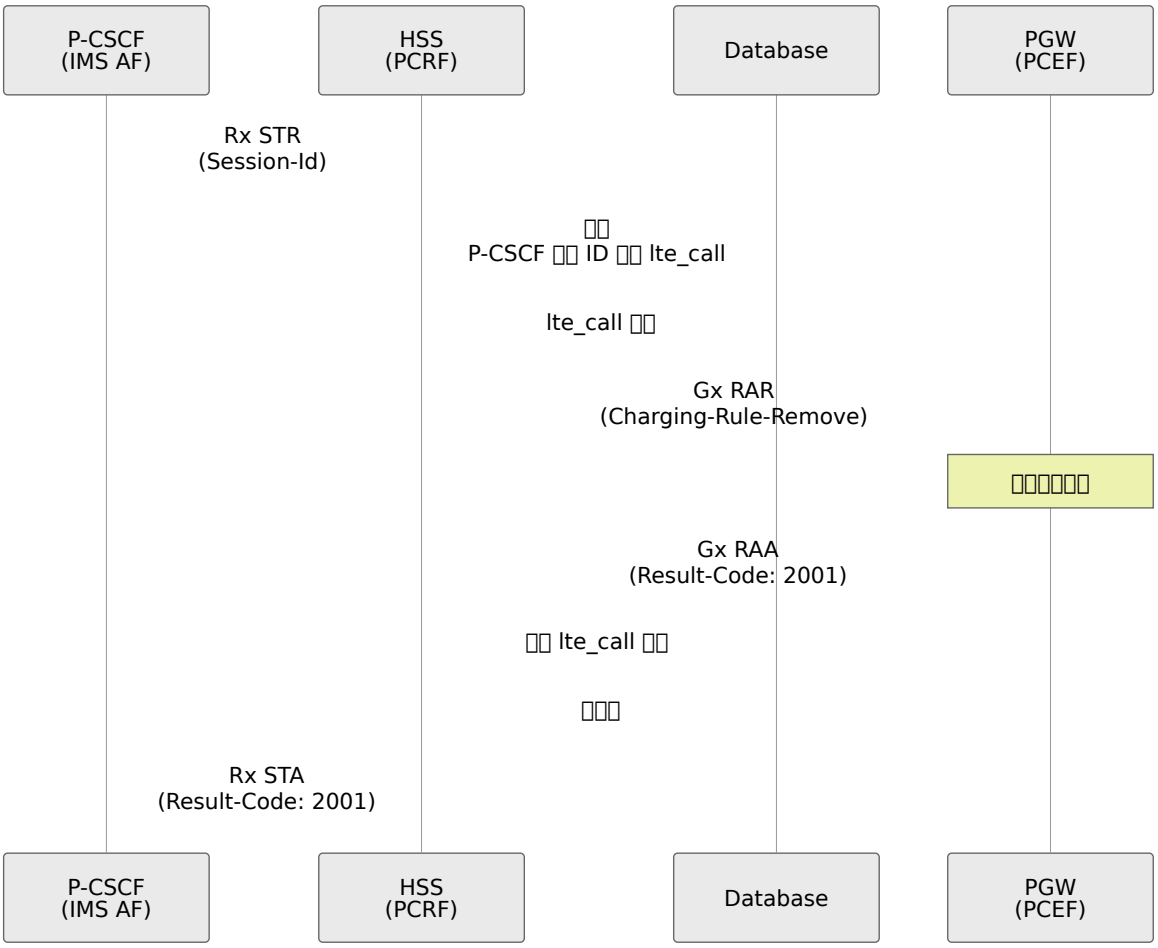
常见消息流

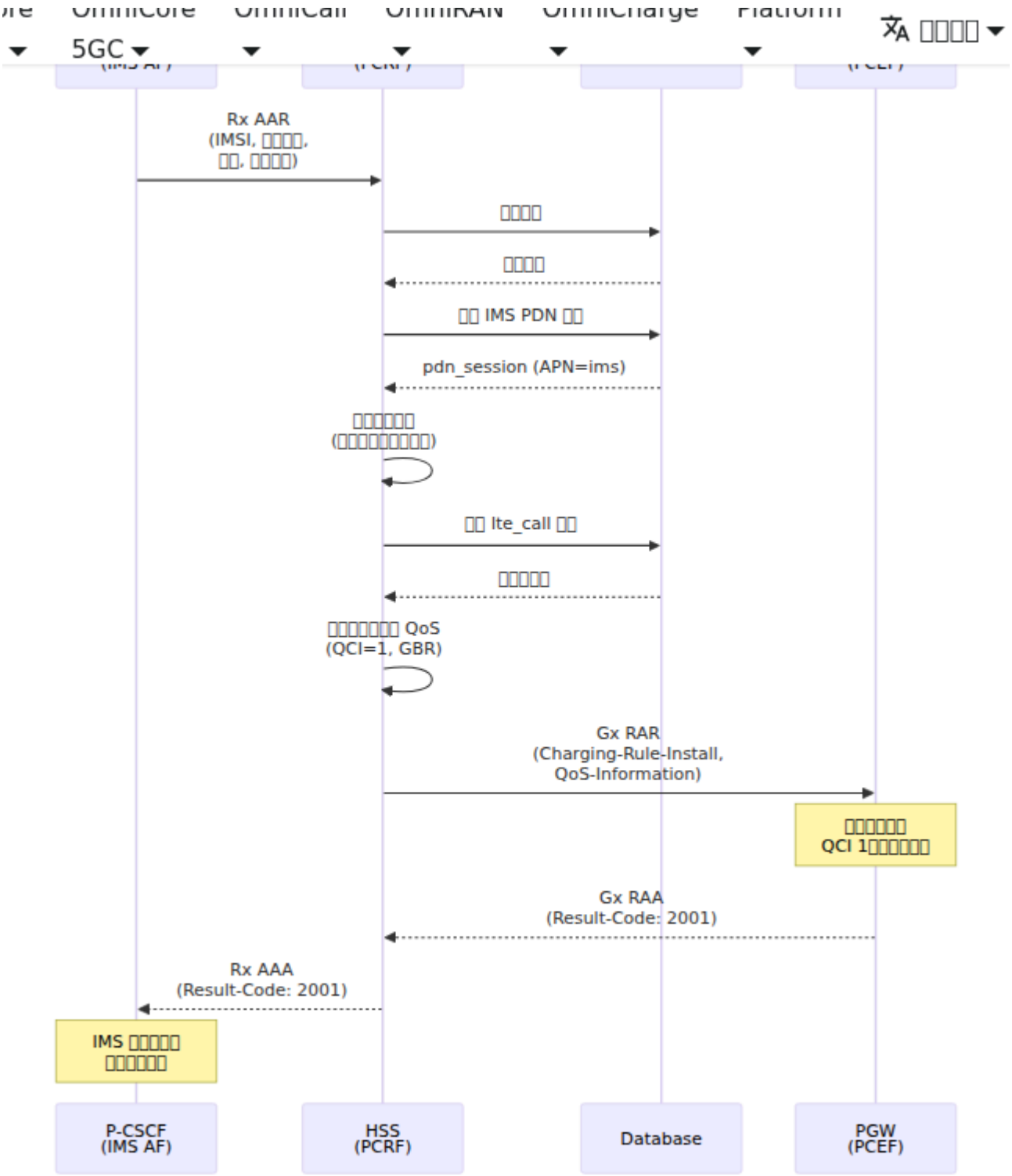
流程 1：PDN 会话建立

□□ 2□VoLTE □□□□□Rx AAR → Gx RAR□



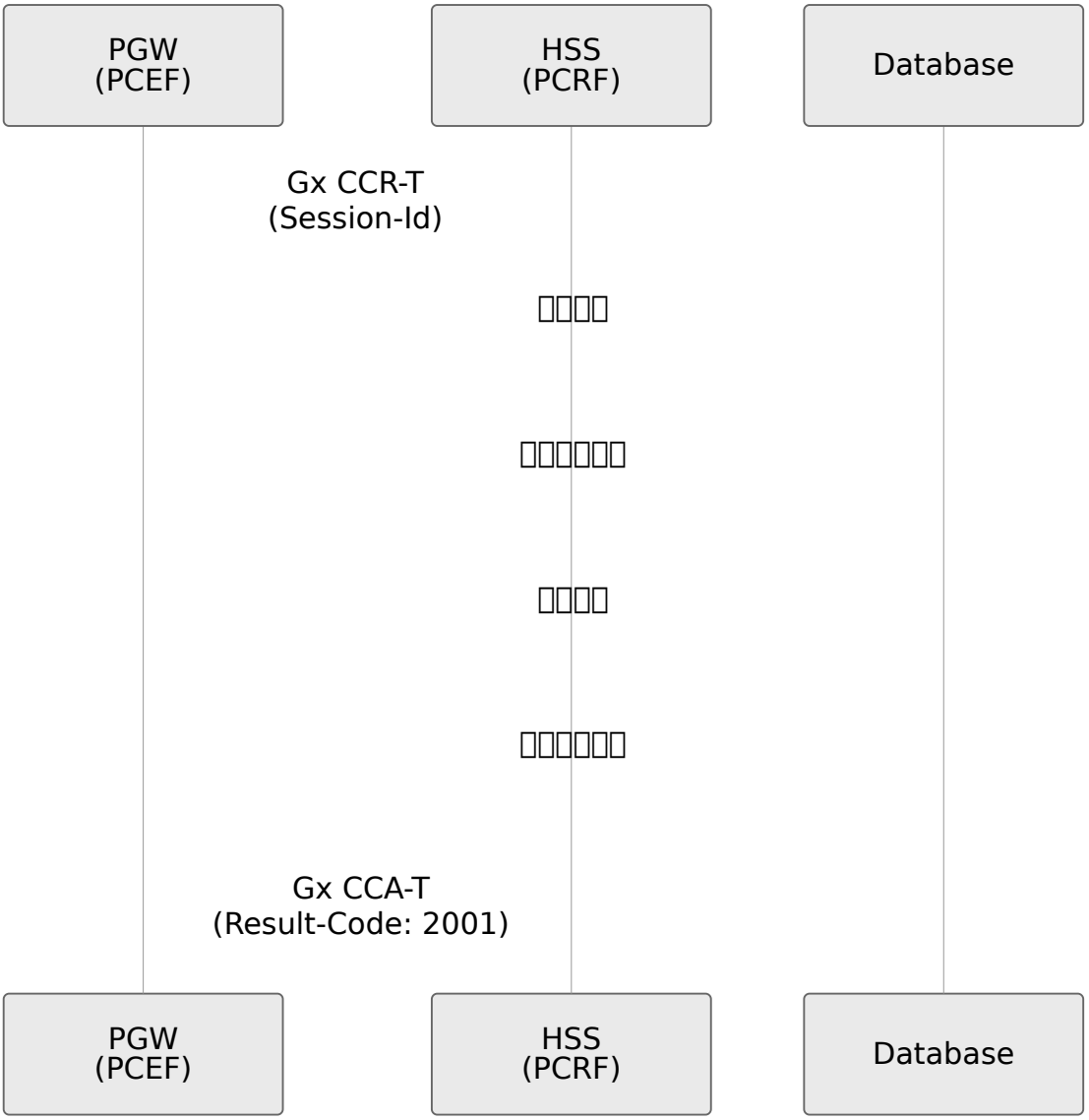
□□ 3□VoLTE □□□□□Rx STR → Gx RAR□





流程 **4**: **PDN** 会话更新

Rx AAR
(IMSI, □□□□□,
□□, □□□□□)
□□□□□
□□□□□
□□ IMS PDN □□
pdn_session (APN=ims)
□□□□□□□
(□□□□□□□□□□)
□□ lte_call □□
□□□□□□
□□□□□□□□ QoS
(QCI=1, GBR)
Gx RAR
(Charging-Rule-Install,
QoS-Information)
□□□□□□□
QCI 1□□□□□□□
Gx RAA
(Result-Code: 2001)
Rx AAA
(Result-Code: 2001)
IMS □□□□□□
□□□□□□□
P-CSCF
(IMS AF)
HSS
(PCRF)
Database
PGW
(PCEF)



流程 **5**: **PDN** 会话终止

PGW
(PCEF)
HSS
(PCRF)
Database
Gx CCR-T
(Session-Id)
查找会话
删除相关通话
删除会话
更新用户状态
Gx CCA-T
(Result-Code: 2001)
PGW
(PCEF)
HSS
(PCRF)
Database

流程 6：通过 **REST API** 手动重新认证

REST API

PCRF 重新认证端点

端点：`POST /api/operation/pcrf_re_auth`

目的：手动触发 Gx 重新认证请求以刷新策略

何时使用：此手动端点通常用于故障排除或强制特定用户的策略刷新。对于常规策略更新（更改 APN QoS 配置文件），系统会自动触发对所有受影响会话的重新认证 - 无需手动操作。

请求体：

```
{
  "imsi": "999999876543210",
  "apn": "ims"
}
```

成功响应（HTTP 200）：

```
{
  "data": "Gx 重新认证请求已发送至 pgw.epc.mnc999.mcc999.3gppnetwork.org，
Result-Code: 2001"
}
```

错误响应（HTTP 400）：

```
{
  "error": "无法发送 999999876543210 的重新认证请求，未找到活动 PDN 会话"
}
```

策略配置 API

PCRF 从存储在数据库中的 APN 配置中检索 QoS 策略。这些策略可以通过 REST API 创建和管理。

自动策略执行：当您更新 APN QoS 配置文件（例如，更改带宽限制或 QCI）时，系统会自动向所有使用该 APN 的活动 PDN 会话的 PGW 发送 Gx 重新认证请求（RAR）。这确保策略更改立即应用于所有连接的用户，而无需手动干预。

策略架构

策略通过三层结构定义：

```
APN 标识符 → APN QoS 配置文件 → APN 配置文件
     ↓                    ↓                  ↓
  "internet"        QCI, AMBR, ARP     将两者链接在一起
```

1. 创建 APN 标识符

定义 APN 名称和 IP 版本支持。

端点：`POST /api/apn/identifier`

请求体：

```
{
  "apn_identifier": {
    "apn": "internet",
    "ip_version": "ipv4v6"
  }
}
```

IP 版本选项：

- `"ipv4"` - 仅 IPv4
- `"ipv6"` - 仅 IPv6
- `"ipv4v6"` - 双栈（IPv4 和 IPv6）
- `"ipv4_or_ipv6"` - 网络决定（IPv4 或 IPv6）

响应（HTTP 201）：

```
{
  "data": {
    "id": "4fa85f64-5717-4562-b3fc-2c963f66afa6",
    "apn": "internet",
    "ip_version": "ipv4v6"
  }
}
```

验证：

- `apn`：必填，1-254 个字符，唯一
- `ip_version`：必填，必须是上述四个选项之一

列出 **APN** 标识符：`GET /api/apn/identifier`

2. 创建 APN QoS 配置文件

定义 QoS 参数（带宽、QCI、优先级）。

端点: `POST /api/apn/qos_profile`

请求体:

```
{
  "apn_qos_profile": {
    "name": "最佳努力互联网",
    "qci": 9,
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 100000,
    "apn_ambr_ul_kbps": 50000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true,
    "online_charging_enabled": false,
    "offline_charging_enabled": true
  }
}
```

QoS 参数:

□□	□□	□□	□□
`name`	□□□	1-254 □□□	□□□□□□□□□□□
`qci`	□□	1-254	QoS □□□□□1-4 = GBR□5-9 = □ GBR□
`allocation_retention_priority`	□□	1-15	ARP □□□1 = □□□□□□
`apn_ambr_dl_kbps`	□□	1-4,294,967,293	APN □□□□□□□□□ □kbps□
`apn_ambr_ul_kbps`	□□	1-4,294,967,293	APN □□□□□□□□□ □kbps□
`pre_emption_capability`	□□	true/false	□□□□□□□□□□□□
`pre_emption_vulnerability`	□□	true/false	□□□□□□□□□□□□□
`online_charging_enabled`	□□	true/false/null	□□□ `Online` AVP □□ Gx CCA-I□ `null`□□□□□□ PGW □ CCR-I □□□ □ □□□□□/□□□□
`offline_charging_enabled`	□□	true/false/null	□□□ `Offline` AVP □□ Gx CCA-I□ `null`□□□□□□ PGW □ CCR-I □□□ □ □□□□□/□□□□

常见 **QCI** 值:

- `1` - 对话语音（VoLTE） - GBR, 100ms 延迟预算
- `2` - 对话视频 - GBR, 150ms 延迟预算
- `5` - IMS 信令 - 非 GBR, 100ms 延迟预算
- `9` - 默认承载（互联网） - 非 GBR, 300ms 延迟预算

响应（HTTP 201）:

```
{
  "data": {
    "id": "8d8ac610-566d-4ef0-9c22-186b2a5ed793",
    "name": "最佳努力互联网",
    "qci": 9,
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 100000,
    "apn_ambr_ul_kbps": 50000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true,
    "online_charging_enabled": false,
    "offline_charging_enabled": true
  }
}
```

列出 **QoS** 配置文件: `GET /api/apn/qos_profile`

3. 创建 **APN** 配置文件

将 APN 标识符与 QoS 配置文件链接。

端点: `POST /api/apn/profile`

请求体:

```
{
  "apn_profile": {
    "name": "互联网 APN",
    "apn_identifier_id": "4fa85f64-5717-4562-b3fc-2c963f66afa6",
    "apn_qos_profile_id": "8d8ac610-566d-4ef0-9c22-186b2a5ed793"
  }
}
```

字段:

- `name`: 配置文件名称（唯一），用于引用
- `apn_identifier_id`: 来自 创建 APN 标识符 的 ID
- `apn_qos_profile_id`: 来自 创建 APN QoS 配置文件 的 ID

响应（HTTP 201）:

```
{
  "data": {
    "id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
    "name": "互联网 APN",
    "apn_identifier_id": "4fa85f64-5717-4562-b3fc-2c963f66afa6",
    "apn_qos_profile_id": "8d8ac610-566d-4ef0-9c22-186b2a5ed793"
  }
}
```

约束:

- `apn_identifier_id` 和 `apn_qos_profile_id` 必须引用现有记录
- 每个 APN 标识符和 QoS 配置文件的组合必须是唯一的

列出 **APN** 配置文件: `GET /api/apn/profile`

完整策略配置示例

步骤 **1**: 创建 **IMS APN** 策略（**VoLTE**）

```
# 1. 创建 APN 标识符
curl -X POST https://hss.example.com:8443/api/apn/identifier \
  -H "Content-Type: application/json" \
  -d '{
    "apn_identifier": {
      "apn": "ims",
      "ip_version": "ipv4v6"
    }
  }'
# 响应: {"data": {"id": "2c3e4f5a-6b7c-4d8e-9f0a-1b2c3d4e5f6a", ...}}

# 2. 创建 QoS 配置文件□IMS 信令)
curl -X POST https://hss.example.com:8443/api/apn/qos_profile \
  -H "Content-Type: application/json" \
  -d '{
    "apn_qos_profile": {
      "name": "IMS 信令 QoS",
      "qci": 5,
      "allocation_retention_priority": 2,
      "apn_ambr_dl_kbps": 5000,
      "apn_ambr_ul_kbps": 5000,
      "pre_emption_capability": true,
      "pre_emption_vulnerability": false
    }
  }'
# 响应: {"data": {"id": "7b44a3e0-27cc-4b21-8d7a-3f19e5c2b8d4", ...}}

# 3. 创建 APN 配置文件
curl -X POST https://hss.example.com:8443/api/apn/profile \
  -H "Content-Type: application/json" \
  -d '{
    "apn_profile": {
      "name": "IMS APN",
      "apn_identifier_id": "2c3e4f5a-6b7c-4d8e-9f0a-1b2c3d4e5f6a",
      "apn_qos_profile_id": "7b44a3e0-27cc-4b21-8d7a-3f19e5c2b8d4"
    }
  }'
# 响应: {"data": {"id": "91c2f3a4-5d6e-4f7a-8b9c-0d1e2f3a4b5c", ...}}
```

步骤 **2**: 分配给用户

创建后，APN 配置文件通过 EPC 配置文件分配给用户。请参见 API 参考 以链接 APN 配置文件到用户。

策略更新和删除

更新 **QoS** 配置文件：

```
PATCH /api/apn/qos_profile/{id}
PUT /api/apn/qos_profile/{id}
```

示例 - 为所有用户增加带宽：

```
# 更新 QoS 配置文件以增加带宽
curl -X PATCH
https://hss.example.com:8443/api/apn/qos_profile/8d8ac610-566d-
4ef0-9c22-186b2a5ed793 \
  -H "Content-Type: application/json" \
  -d '{
    "apn_qos_profile": {
      "apn_ambr_dl_kbps": 150000,
      "apn_ambr_ul_kbps": 75000
    }
  }'
```

自动发生的事情：

1. 数据库中的 QoS 配置文件更新
2. 系统识别所有使用与此 QoS 配置文件链接的 APN 的活动 PDN 会话
3. 对于每个活动会话，向相应的 PGW 发送 Gx RAR
4. PGW 更新承载 QoS 以反映新的带宽限制
5. 所有连接的用户立即接收到更新的策略

示例场景：如果 100 个用户当前连接在使用此 QoS 配置文件的“互联网” APN 上，则在 API 调用完成后的几秒钟内，所有 100 个用户的带宽限制将更新为 150 Mbps 下行/75 Mbps 上行。

注意：当您更新 APN QoS 配置文件时，系统 自动触发重新认证，以便对所有使用该 APN 的活动 PDN 会话应用新策略，无需手动重新认证。

删除资源：

```
DELETE /api/apn/identifier/{id}
DELETE /api/apn/qos_profile/{id}
DELETE /api/apn/profile/{id}
```

删除约束：

- 不能删除被 APN 配置文件引用的 APN 标识符或 QoS 配置文件
- 不能删除分配给活动用户的 APN 配置文件

策略模板

高速互联网（下行 **100 Mbps** / 上行 **50 Mbps**）：

```
{
  "apn_qos_profile": {
    "name": "高速互联网",
    "qci": 9,
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 100000,
    "apn_ambr_ul_kbps": 50000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true
  }
}
```

优质互联网（下行 **500 Mbps** / 上行 **100 Mbps**）：

```
{
  "apn_qos_profile": {
    "name": "优质互联网",
    "qci": 8,
    "allocation_retention_priority": 5,
    "apn_ambr_dl_kbps": 500000,
    "apn_ambr_ul_kbps": 100000,
    "pre_emption_capability": true,
    "pre_emption_vulnerability": false
  }
}
```

物联网/M2M□低带宽）：

```
{
  "apn_qos_profile": {
    "name": "物联网 M2M",
    "qci": 9,
    "allocation_retention_priority": 10,
    "apn_ambr_dl_kbps": 1024,
    "apn_ambr_ul_kbps": 512,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true
  }
}
```

紧急服务□最高优先级）：

```
{
  "apn_qos_profile": {
    "name": "紧急 APN",
    "qci": 5,
    "allocation_retention_priority": 1,
    "apn_ambr_dl_kbps": 10000,
    "apn_ambr_ul_kbps": 10000,
    "pre_emption_capability": true,
    "pre_emption_vulnerability": false
  }
}
```

配置

Diameter 服务设置

Gx 应用（`config/runtime.exs`）：

```
%{
  application_name: :gx,
  application_dictionary: :diameter_gen_3gpp_gx,
  vendor_specific_application_ids: [
    %{vendor_id: 10415, auth_application_id: 16_777_238}
  ]
}
```

Rx 应用（`config/runtime.exs`）：

```
%{
  application_name: :rx,
  application_dictionary: :diameter_gen_3gpp_rx,
  vendor_specific_application_ids: [
    %{vendor_id: 10415, auth_application_id: 16_777_236}
  ]
}
```

QoS 参数

QoS 参数来源于：

- 默认承载：数据库中的 APN 配置文件
  - `apn_qos_profile.qci`（QoS 类标识符）
  - `apn_qos_profile.apn_ambr_ul_kbps`（聚合最大比特率上行）
  - `apn_qos_profile.apn_ambr_dl_kbps`（聚合最大比特率下行）
  - `apn_qos_profile.priority_level`（分配保留优先级）
- 专用承载：从 Rx AAR 媒体组件描述中提取
  - QCI：1（对话语音）
  - 保证比特率：来自 Max-Requested-Bandwidth AVPs
  - 流过滤器：来自 Flow-Description AVPs

错误处理

结果代码	类型	含义	原因
2001	成功	DIAMETER_SUCCESS	请求成功处理
5001	实验性	用户未找到	IMSI 不在用户数据库中
5002	实验性	会话未找到	PDN 会话不存在以进行更新/终止
5063	实验性	服务未授权	IMS 媒体授权被拒绝

实施细节

会话管理

PCRF 跟踪:

- 活动 **PDN** 会话 - 每个 APN 每个用户一个
- **VoLTE** 通话 - 每个 IMS 会话多个通话（支持会议通话）
- **QoS** 策略 - 根据 APN 配置动态应用
- 计费规则 - 流量流模板和服务特定策略

高级策略功能

PCRF 支持高级策略控制，括:

- 通过 Gx 接口的 计费规则安装/删除
- 流量流模板（**TFT**）匹配 以实现服务区分
- 基于应用或流量类型的动态速度配置
- 基于网络条件或用户行为触发的服务感知策略

请联系您的系统管理员以获取有关配置高级计费规则和基于 TFT 的策略的信息。

相关文档

Sy 接口 - 针对外部 OCS 的用户支出限额控制 Diameter 协议 - 详细的协议规范
API 参考 - 完整的 API 文档
架构 - 整体 HSS 架构
数据映射 - 数据库到 Diameter AVP 的映射

API 错误处理

← 返回 API 参考

目录

- 常见错误响应
- 错误处理流程

常见错误响应

400 错误请求

```
{
  "error": "Invalid JSON format"
}
```

原因:

- JSON 格式错误
- 缺少必填字段
- 数据类型无效

404 未找到

```
{
  "error": "Resource not found"
}
```

原因:

- 订阅者/个人资料/实体不存在
- URL 中的 ID 不正确

422 无法处理的实体

```
{
  "errors": {
    "imsi": ["has already been taken"],
    "key_set_id": ["does not exist"]
  }
}
```

原因:

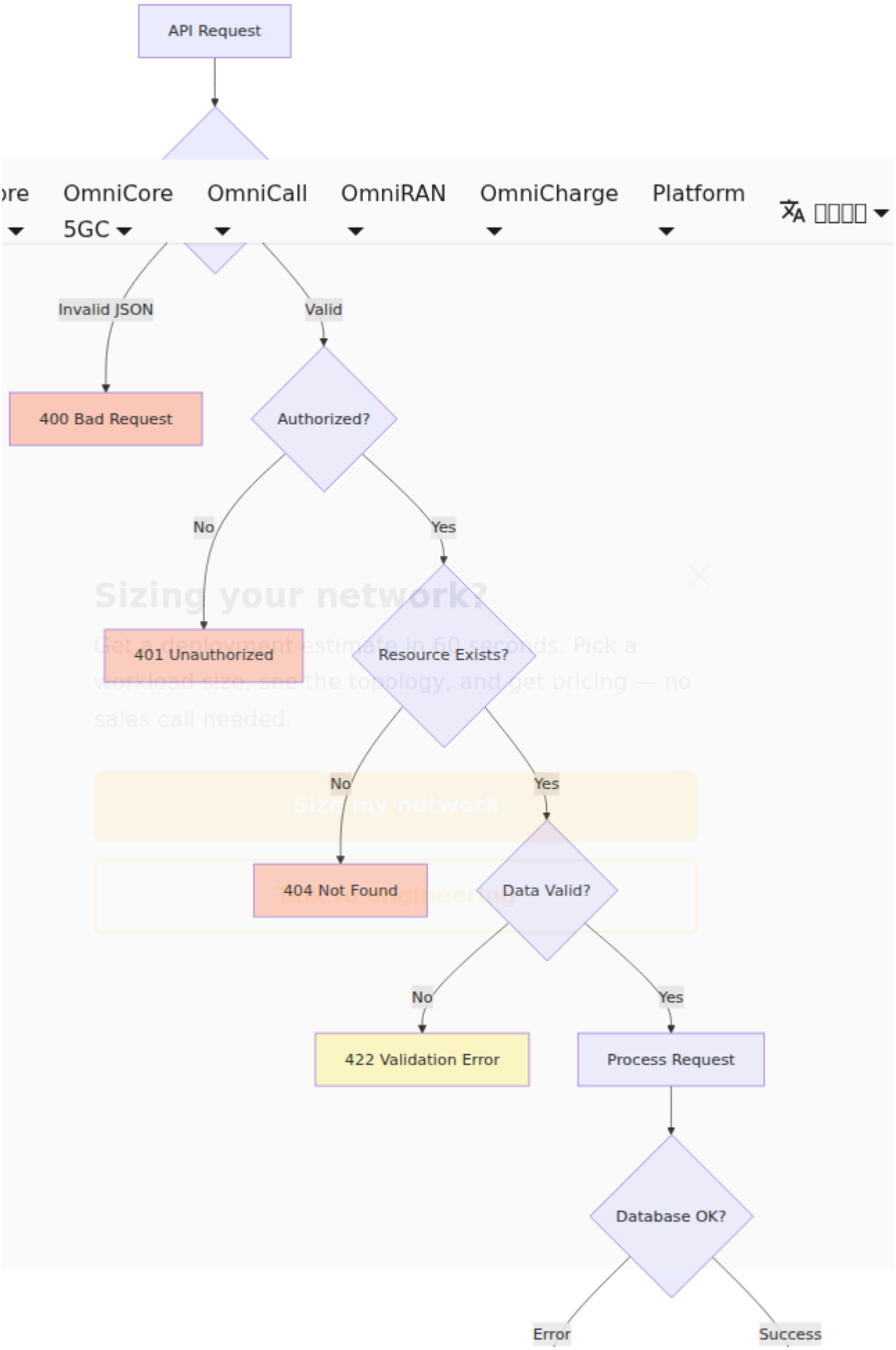
- 验证失败
- 数据库约束被违反
- 外键引用不存在

500 服务器内部错误

```
{
  "error": "Internal server error"
}
```

原因:

- 数据库连接问题
- 意外的应用程序错误


API Request
ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
Invalid JSON
Valid
400 Bad Request
Authorized?
No
Yes
401 Unauthorized
Resource Exists?
No
Yes
404 Not Found
Data Valid?
No
Yes
422 Validation Error
Process Request
Database OK?
Error
Success

← □□ API □□

API 使用示例
← 返回 API 参考

目录

完整的用户配置
休眠用户预配置
完整的静态 IP 配置

完整的用户配置

此示例演示了从头开始配置新用户的完整工作流程。该过程涉及在创建用户之前创建所有必需的配置文件和组件。

先决条件：此示例使用 jq 进行 JSON 解析。使用 apt-get install jq 或 brew install jq 安装。

相关部分：

密钥集管理
APN 配置文件
EPC 配置文件
用户管理

```
# 1. 创建密钥集
KEY_SET_ID=$(curl -k -X POST
https://hss.example.com:8443/api/key_set \
  -H "Content-Type: application/json" \
  -d '{
    "ki": "0123456789ABCDEF0123456789ABCDEF",
    "opc": "FEDCBA9876543210FEDCBA9876543210",
    "authentication_algorithm": "milenage",
    "amf": "8000",
    "sqn": 0
  }' | jq -r '.response.id')

# 2. 创建 APN QoS 配置文件
APN_QOS_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/qos_profile \
  -H "Content-Type: application/json" \
  -d '{
    "name": "默认互联网 QoS",
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 50000,
    "apn_ambr_ul_kbps": 25000,
    "pre_emption_capability": true,
    "pre_emption_vulnerability": true,
    "qci": 9
  }' | jq -r '.response.id')

# 3. 创建 APN 标识符
APN_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/identifier \
  -H "Content-Type: application/json" \
  -d '{
    "apn": "internet",
    "ip_version": "ipv4v6"
  }' | jq -r '.response.id')

# 4. 创建 APN 配置文件
APN_PROFILE_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_identifier_id\": \"$APN_ID\",
    \"apn_qos_profile_id\": \"$APN_QOS_ID\",
    \"name\": \"互联网 APN\"
```

```
  }" | jq -r '.response.id')

# 5. 创建 EPC 配置文件
EPC_PROFILE_ID=$(curl -k -X POST
https://hss.example.com:8443/api/epc/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_profiles\": [$APN_PROFILE_ID],
    \"name\": \"标准数据计划\",
    \"network_access_mode\": \"packet_only\",
    \"tracking_area_update_interval_seconds\": 600,
    \"ue_ambr_dl_kbps\": 100000,
    \"ue_ambr_ul_kbps\": 50000
  }" | jq -r '.response.id')

# 6. 创建用户
SUBSCRIBER_ID=$(curl -k -X POST
https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"imsi\": \"001001123456789\",
    \"key_set_id\": \"$KEY_SET_ID\",
    \"epc_profile_id\": \"$EPC_PROFILE_ID\"
  }" | jq -r '.response.id')

echo "用户成功配置，ID: $SUBSCRIBER_ID"
```

此操作创建了：

此配置工作流程创建了一个完整的用户，□括：

1. 加密密钥（密钥集）- 用于身份验证
2. 数据服务配置文件（EPC 配置文件）- 带宽和网络访问设置
3. **APN** 配置（APN 配置文件）- 带 QoS 的接入点
4. 用户记录（用户）- 实际的用户实体

后续步骤：

- 添加电话号码：请参阅 MSISDN 管理
- 启用语音服务：创建并分配 IMS 配置文件
- 配置漫游：创建并分配 漫游配置文件

- 关联物理 SIM：创建并分配 SIM

另见：

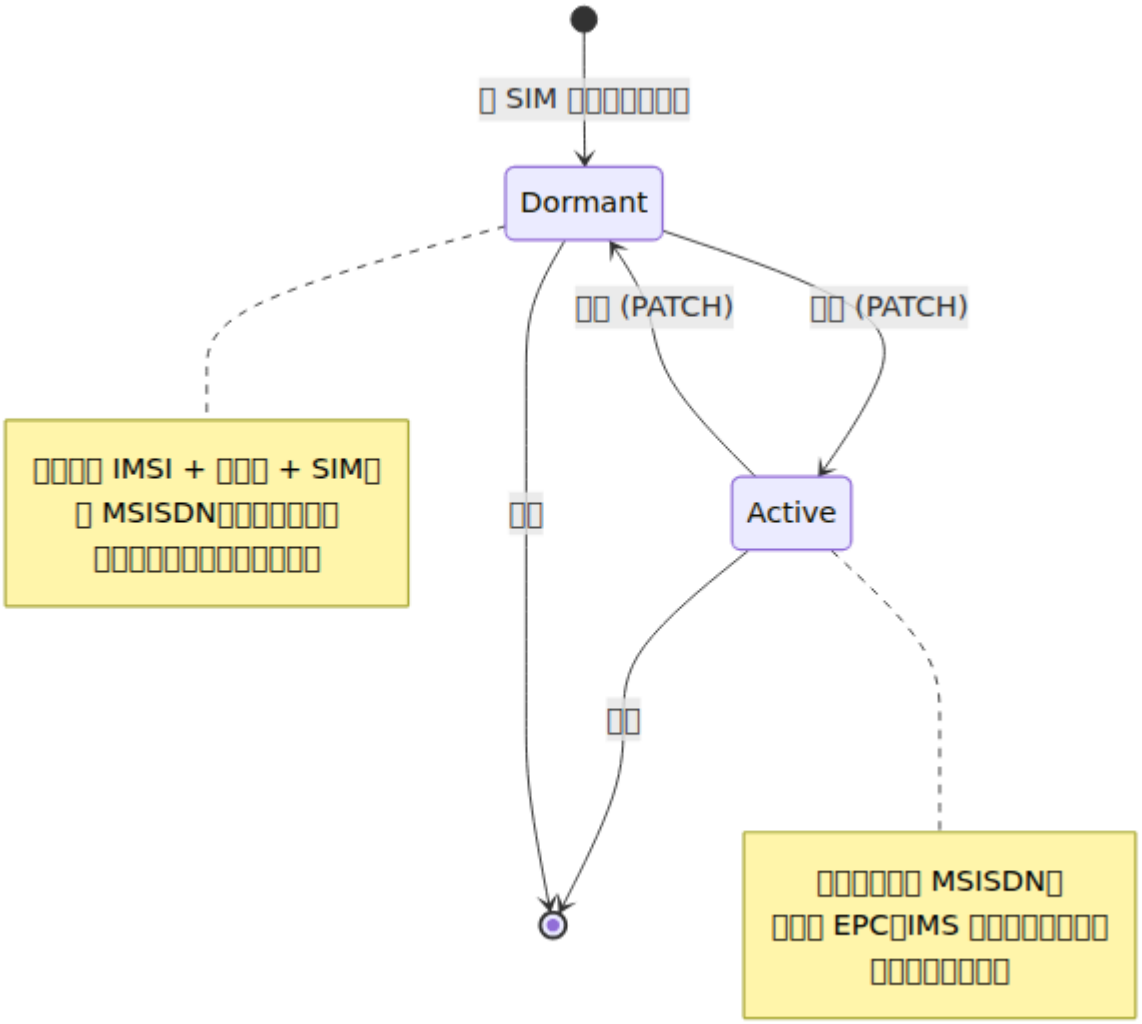
- 多 MSISDN 文档 - 分配多个电话号码
- 配置文件文档 - 高级配置文件配置

休眠用户预配置

使用此工作流程从 SIM 供应商输出文件批量加载用户，并在稍后激活每个用户。这是大多数运营商推荐的模型。

休眠用户仅保留其身份和身份验证密钥。您在激活时添加电话号码和服务配置文件。这与 SIM 供应商发送给您的数据相匹配。SIM 供应商文件□含 ICCID、PIN、PUK 和 ADM 代码，以及身份验证密钥（Ki、OPc）。该文件不□含电话号码或服务计划。

数据库只需要 IMSI 和密钥集即可插入用户。这使得休眠记录成为可能。这并不意味着其他对象在服务中是可选的。每个对象都需要其自己的服务：EPC 配置文件用于数据，IMS 配置文件和 MSISDN 用于语音，以及用于身份验证的 SIM。休眠用户在您分配匹配对象之前无法在网络上工作。您在客户激活服务时分配这些对象。由于 SIM、密钥集和用户是独立对象，请参阅 密钥集、SIM 和用户。



配置生命周期

第 1 步：导入□创建 SIM、密钥集和禁用的用户）

为 SIM 供应商文件中的每一行创建一个密钥集、一个 SIM 和一个禁用的用户。此示例使用 `jq` 从每个响应中读取返回的标识符。

```
# 1. 从 SIM 供应商文件创建密钥集 (Ki, OPc)
KEY_SET_ID=$(curl -k -X POST
https://hss.example.com:8443/api/key_set \
  -H "Content-Type: application/json" \
  -d '{
    "ki": "0123456789ABCDEF0123456789ABCDEF",
    "opc": "FEDCBA9876543210FEDCBA9876543210",
    "authentication_algorithm": "milenage",
    "amf": "8000",
    "sqn": 0
  }' | jq -r '.response.id')

# 2. 从 SIM 供应商文件创建 SIM (ICCID, PIN/PUK/ADM)
SIM_ID=$(curl -k -X POST https://hss.example.com:8443/api/sim \
  -H "Content-Type: application/json" \
  -d '{
    "iccid": "89440000000000000001",
    "is_esim": false,
    "ind_length": 5
  }' | jq -r '.response.id')

# 3. 创建禁用状态的用户。
#    imsi 和 key_set_id 是必需的。链接 SIM。添加无 MSISDN 和无配置文件。
SUBSCRIBER_ID=$(curl -k -X POST
https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"imsi\": \"001010000000001\",
    \"key_set_id\": \"$KEY_SET_ID\",
    \"sim_id\": \"$SIM_ID\",
    \"enabled\": false,
    \"ims_enabled\": false
  }" | jq -r '.response.id')
```

休眠用户可以安全地保留在数据库中。用户在禁用状态下不会连接、身份验证或注册。如果网络查询一个休眠用户，HSS 将拒绝请求。请参阅 休眠用户在网络上的行为。

第 2 步：激活（分配号码和配置文件，然后启用）

要激活用户，请发送一个 `PATCH` 请求。添加 MSISDN，分配配置文件，并将 `enabled` 设置为 `true`。

```
curl -k -X PATCH
https://hss.example.com:8443/api/subscriber/$SUBSCRIBER_ID \
  -H "Content-Type: application/json" \
  -d "{
    \"enabled\": true,
    \"ims_enabled\": true,
    \"msisdns\": [\"$MSISDN_ID\"],
    \"epc_profile_id\": \"$EPC_PROFILE_ID\",
    \"ims_profile_id\": \"$IMS_PROFILE_ID\",
    \"roaming_profile_id\": \"$ROAMING_PROFILE_ID\"
  }"
```

`PATCH` 请求仅更改您发送的字段。IMSI、密钥集和 SIM 保持不变。首先创建 MSISDN 和配置文件，或重用现有标识符。请参阅 完整的用户配置 以了解如何创建配置文件。

第 3 步：停用禁用用户）

要停用用户，将 `enabled` 设置为 `false`。用户记录及其数据保留在数据库中。

```
curl -k -X PATCH
https://hss.example.com:8443/api/subscriber/$SUBSCRIBER_ID \
  -H "Content-Type: application/json" \
  -d '{
    "enabled": false,
    "ims_enabled": false
  }'
```

要仅删除 IMS 语音并保留数据服务，将 `ims_enabled` 设置为 `false`，并将 `enabled` 保持为 `true`。

休眠用户在网络上的行为

用户在休眠或禁用状态下会失败关闭。HSS 在网络接口上拒绝请求。HSS 还会记录警告日志，说明确切原因。原因如下：

- 用户被禁用（`enabled` 为 `false`）。
- 用户的 IMS 被禁用（`ims_enabled` 为 `false`）。
- 仍未分配所需对象。例如，没有用于 LTE 附加的 EPC 配置文件，或没有用于 IMS 注册的 IMS 配置文件。

拒绝使用实验结果代码 DIAMETER_ERROR_USER_UNKNOWN (5001) 在 S6a 和 Cx 接口上，依据 3GPP TS 29.272 和 3GPP TS 29.229。警告日志命名 IMSI 和确切原因。该日志使您能够区分休眠用户和未知 IMSI。

完整的静态 IP 配置

此示例演示了从头开始配置具有静态 IP 地址的用户。

场景：配置一个需要在“互联网” APN 上具有静态 IPv4 地址的 IoT 设备用户。

```
# 先决条件: 必须安装 jq (apt-get install jq 或 brew install jq)

# 1. 创建密钥集
KEY_SET_ID=$(curl -k -X POST
https://hss.example.com:8443/api/key_set \
  -H "Content-Type: application/json" \
  -d '{
    "ki": "0123456789ABCDEF0123456789ABCDEF",
    "opc": "FEDCBA9876543210FEDCBA9876543210",
    "authentication_algorithm": "milenage",
    "amf": "8000",
    "sqn": 0
  }' | jq -r '.response.id')

# 2. 创建 APN QoS 配置文件
APN_QOS_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/qos_profile \
  -H "Content-Type: application/json" \
  -d '{
    "name": "IoT 最佳努力",
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 10000,
    "apn_ambr_ul_kbps": 5000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": false,
    "qci": 9
  }' | jq -r '.response.id')

# 3. 创建 APN 标识符
APN_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/identifier \
  -H "Content-Type: application/json" \
  -d '{
    "apn": "internet",
    "ip_version": "ipv4"
  }' | jq -r '.response.id')

# 4. 创建 APN 配置文件
APN_PROFILE_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_identifier_id\": \"$APN_ID\",
```

```
    \"apn_qos_profile_id\": \"$APN_QOS_ID\",
    \"name\": \"IoT 互联网 APN\"
  }" | jq -r '.response.id')

# 5. 为 APN 创建静态 IP
STATIC_IP_ID=$(curl -k -X POST
https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_profile_id\": \"$APN_PROFILE_ID\",
    \"ipv4_static_ip\": \"100.64.1.100\"
  }" | jq -r '.response.id')

# 6. 创建 EPC 配置文件
EPC_PROFILE_ID=$(curl -k -X POST
https://hss.example.com:8443/api/epc/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_profiles\": [$APN_PROFILE_ID],
    \"name\": \"IoT 数据计划\",
    \"network_access_mode\": \"packet_only\",
    \"tracking_area_update_interval_seconds\": 600,
    \"ue_ambr_dl_kbps\": 10000,
    \"ue_ambr_ul_kbps\": 5000
  }" | jq -r '.response.id')

# 7. 创建 MSISDN（电话号码）
MSISDN_ID=$(curl -k -X POST
https://hss.example.com:8443/api/msisdn \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": "14155551000"
  }' | jq -r '.response.id')

# 8. 创建具有静态 IP 的用户
SUBSCRIBER_ID=$(curl -k -X POST
https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"imsi\": \"001001999999999\",
    \"key_set_id\": \"$KEY_SET_ID\",
    \"epc_profile_id\": \"$EPC_PROFILE_ID\",
    \"msisdns\": [$MSISDN_ID],
    \"static_ips\": [$STATIC_IP_ID]
```

```
  }" | jq -r '.response.id')

echo "IoT 用户成功配置！"
echo "  用户 ID: $SUBSCRIBER_ID"
echo "  IMSI: 001001999999999"
echo "  MSISDN: 14155551000"
echo "  静态 IPv4: 100.64.1.100（在 'internet' APN 上）"
```

此操作创建了：

此配置工作流程创建了一个完整的 IoT 用户，包括：

1. 加密密钥（密钥集）- 用于身份验证
2. **APN** 配置（APN 配置文件）- “互联网”接入点
3. 静态 **IP** 分配（静态 IP）- 固定的 IPv4 地址 100.64.1.100
4. 数据服务配置文件（EPC 配置文件）- IoT 优化的带宽限制
5. 电话号码（MSISDN）- 用于设备识别
6. 用户记录（用户）- 完整的用户实体

结果：

当此用户连接到网络并连接到“互联网” APN 时，他们将收到静态 IP 地址 `100.64.1.100`，而不是动态 DHCP 地址。

后续步骤：

- 添加具有静态 IP 的其他 APN：对每个 APN 重复步骤 2-5
- 启用语音服务：创建并分配 IMS 配置文件
- 配置漫游：创建并分配 漫游配置文件
- 关联物理 SIM：创建并分配 SIM

另见：

- 静态 IP 管理 - 详细的静态 IP 文档
- 完整的用户配置 - 不带静态 IP 的基本配置
- 多 MSISDN 文档 - 分配多个电话号码

← 返回 API 参考

OmniHSS API 参考

← 返回操作指南

目录

- API 概述
- 密钥集管理
- 用户管理
- Sh RepositoryData (透明数据)
- MSISDN 管理
- IMS 订阅管理
- SIM 管理
- 密钥集管理
- 配置文件管理
- 静态 IP 管理
- 漫游管理
- EIR 管理
- 状态和健康
- 错误处理
- API 使用示例

API 概述

基本 URL

```
https://[hostname]:8443/api
```

请求格式

- **Content-Type:** `application/json`
- 协议：仅支持 HTTPS
- 端口：8443（可配置）

重要：所有 API 端点期望接收没有包装对象的“扁平” JSON 有效负载。

正确格式：

```
{
  "name": "value",
  "field": "value"
}
```

错误格式（请勿使用）：

```
{
  "subscriber": {
    "name": "value",
    "field": "value"
  }
}
```

示例：

```
# ✓ 正确
curl -X POST https://hss.example.com:8443/api/ims/profile \
  -H "Content-Type: application/json" \
  -d '{"name": "default", "ifc_template": "..."}'

# ✗ 错误
curl -X POST https://hss.example.com:8443/api/ims/profile \
  -H "Content-Type: application/json" \
  -d '{"ims_profile": {"name": "default", "ifc_template": "..."}}'
```

响应格式

所有响应均为 JSON，具有以下结构：

成功响应：

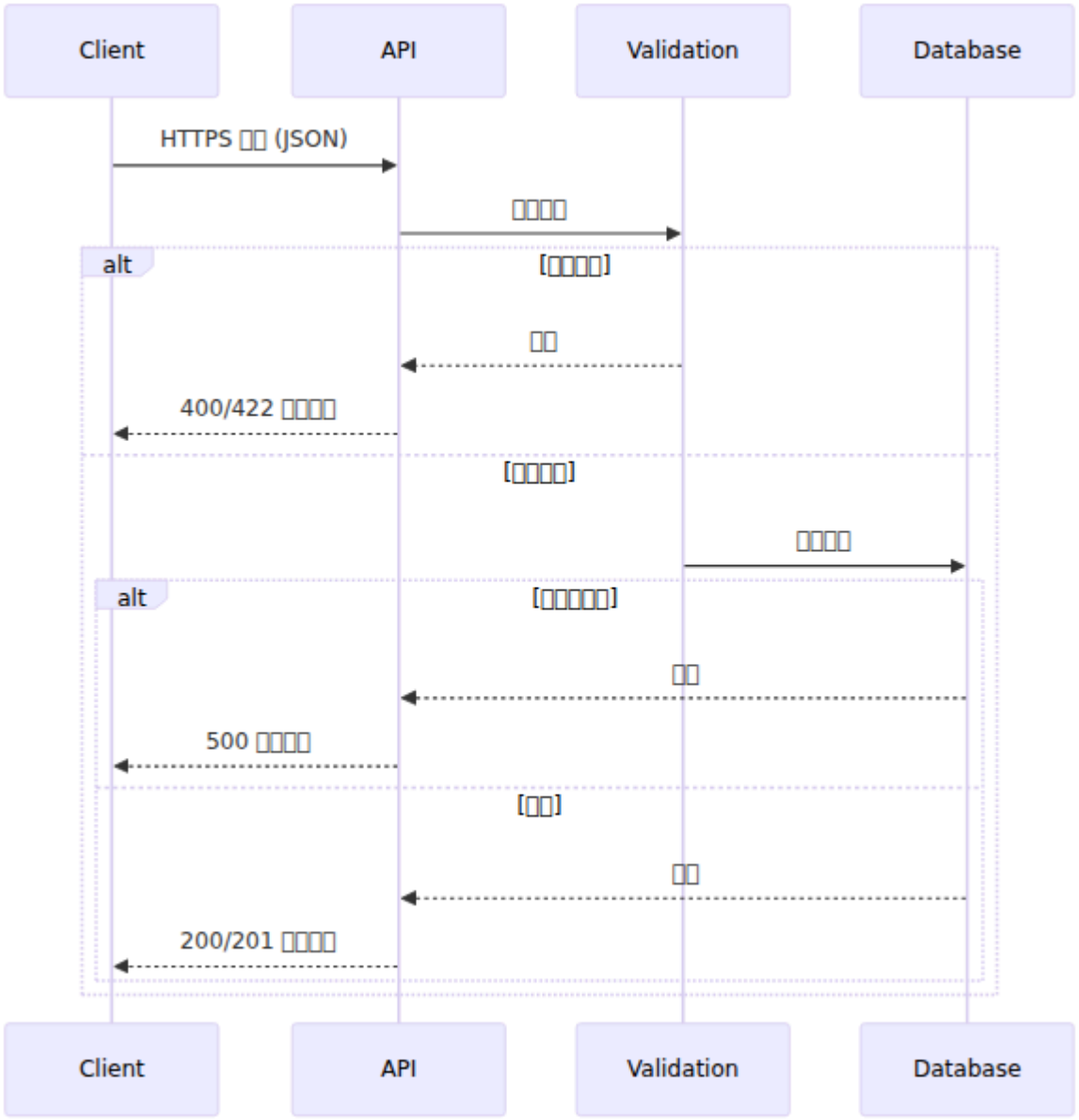
```
{
  "status": "success",
  "response": { ... }
}
```

错误响应：

```
{
  "status": "error",
  "response": {
    "invalid_fields": {
      "field_name": "error message"
    }
  }
}
```

HTTP 状态码

代码	含义	用例
200	OK	成功的 GET、PUT、DELETE
201	创建	成功的 POST
400	错误请求	输入数据无效
404	未找到	资源不存在
422	无法处理的实体	验证错误
500	服务器内部错误	服务器端错误



API 请求流程

用户管理

列出用户

检索所有用户或按条件过滤。

端点：`GET /api/subscriber`

查询参数：

参数	类型	描述
`enabled`	boolean	按启用状态过滤
`ims_enabled`	boolean	按 IMS 启用状态过滤

示例请求：

```
curl -k https://hss.example.com:8443/api/subscriber
```

示例响应：

```
{
  "data": [
    {
      "id": "d290f1ee-6c54-4b01-90e6-d701748f0851",
      "imsi": "001001123456789",
      "enabled": true,
      "ims_enabled": true,
      "sim_id": "3f2504e0-4f89-41d3-9a0c-0305e82c3301",
      "key_set_id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
      "epc_profile_id": "e58ed763-928c-4155-bee9-fdbaaadc15f3",
      "ims_profile_id": "7c9e6679-7425-40de-944b-e07fc1f90ae7",
      "roaming_profile_id": "9c858901-8a57-4791-81fe-
4c455b099bc9",
      "custom_attributes": {},
      "inserted_at": "2025-10-15T10:30:00Z",
      "updated_at": "2025-10-15T10:30:00Z"
    }
  ]
}
```

按 ID 获取用户

通过数据库 ID 检索特定用户。

端点：`GET /api/subscriber/:id`

路径参数：

参数	类型	描述
`id`	string (UUID)	用户数据库 ID

示例请求：

```
curl -k https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-4b01-90e6-d701748f0851
```

按 IMSI 获取用户

通过其 IMSI 检索用户。

端点：`GET /api/subscriber/imsi/:imsi`

路径参数：

参数	类型	描述	格式
`imsi`	string	国际移动用户身份	14-15 位数字

示例请求：

```
curl -k https://hss.example.com:8443/api/subscriber/imsi/001001123456789
```

用例：通过其 IMSI 进行特定用户的故障排除。

按 MSISDN 获取用户

通过其电话号码检索用户。

端点：`GET /api/subscriber/msisdn/:msisdn`

路径参数：

参数	类型	描述	格式
`msisdn`	string	移动台 ISDN 号��?	1-15 位数字 (E.164)

示例请求：

```
curl -k
https://hss.example.com:8443/api/subscriber/msisdn/14155551234
```

用例：当您只有他们的电话号码时查找用户信息。

创建用户

配置一个新用户。

端点：`POST /api/subscriber`

请求体：

```
{
  "subscriber": {
    "imsi": "001001123456789",
    "enabled": true,
    "ims_enabled": true,
    "sim_id": "3f2504e0-4f89-41d3-9a0c-0305e82c3301",
    "key_set_id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
    "epc_profile_id": "e58ed763-928c-4155-bee9-fdbaaadc15f3",
    "ims_profile_id": "7c9e6679-7425-40de-944b-e07fc1f90ae7",
    "roaming_profile_id": "9c858901-8a57-4791-81fe-4c455b099bc9",
    "custom_attributes": {
      "note": "测试用户"
    }
  }
}
```

必填字段：

- `imsi` - 必须为 14-15 位数字，唯一
- `key_set_id` - 必须引用现有的 密钥集
- `epc_profile_id` - 必须引用现有的 EPC 配置文件

可选字段：

- `enabled` - 默认值: true
- `ims_enabled` - 默认值: true
- `sim_id` - 引用 SIM 卡
- `ims_profile_id` - 引用 IMS 配置文件（IMS 服务所需）
- `roaming_profile_id` - 引用 漫游配置文件（漫游控制所需
- `msisdns` - MSISDN ID 数组（电话号码）
- `static_ips` - 静态 IP ID 数组，用于 APN 分配
- `ims_subscription_id` - 引用 IMS 订阅，以将多个用户分组到一个共享的公共身份和 S-CSCF 下
- `custom_attributes` - 自定义键值对

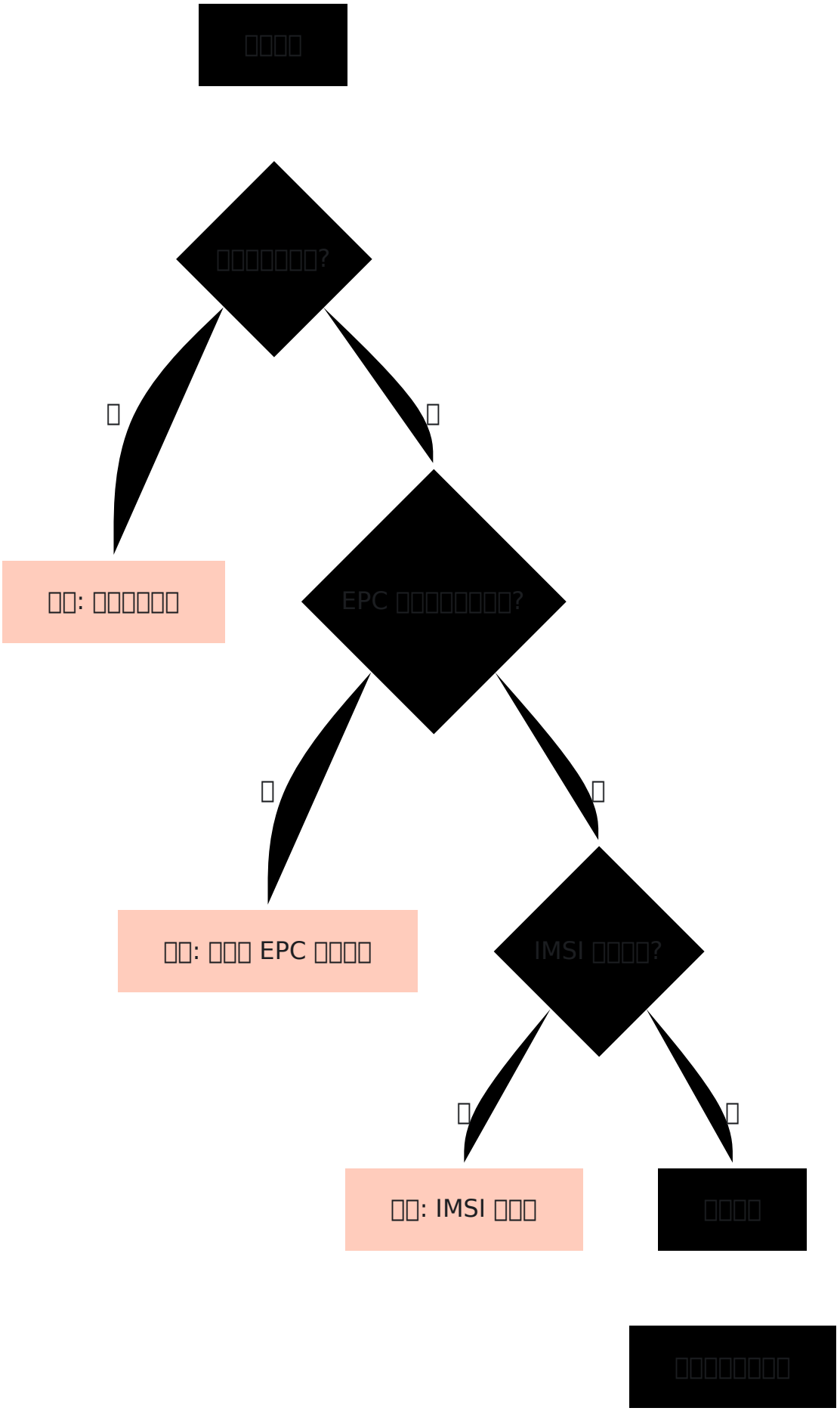
另见：

- 完整的用户配置示例 - 端到端工作流
- 多 MSISDN 文档 - 将电话号码分配给用户
- 静态 IP 管理 - 将静态 IP 分配给 APN

示例请求：

```
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "imsi": "001001123456789",
      "key_set_id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
      "epc_profile_id": "e58ed763-928c-4155-bee9-fdbaaadc15f3"
    }
  }'
```

配置流程：



□□□□
□□□□□□□?
否
是
错误：未找到密钥集
EPC □□□□□□□□?
否
是
错误：未找到 EPC 配置文件
IMSI □□□□?
否
是
错误：IMSI 已存在
□□□□
□□□□□□□□□

201 创建成功

更新用户

修改现有用户。

端点：`PUT /api/subscriber/:id`

路径参数：

参数	类型	描述
`id`	string (UUID)	用户数据库 ID

请求体：

```
{
  "subscriber": {
    "enabled": false,
    "ims_enabled": false,
    "epc_profile_id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "custom_attributes": {
      "note": "暂时禁用"
    }
  }
}
```

可更新字段：

- `enabled` - 启用/禁用所有服务
- `ims_enabled` - 启用/禁用 IMS 服务
- `sim_id` - 更改 SIM 卡分配
- `key_set_id` - 更改加密密钥（小心！）
- `epc_profile_id` - 更改数据服务配置文件
- `ims_profile_id` - 更改语音服务配置文件
- `roaming_profile_id` - 更改漫游策略

- `msisdns` - 更新分配给用户的 电话号码
- `static_ips` - 更新分配给 APN 的 静态 IP
- `custom_attributes` - ��新自定义数据

不可更新：

- `imsi` - 不能更改 IMSI（请删除并重新创建）

另见：

- 配置文件管理 - 管理服务配置文件

示例请求：

```
curl -k -X PUT
https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-4b01-
90e6-d701748f0851 \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "enabled": false
    }
  }'
```

用例：

- 暂时禁用用户：`{"enabled": false}`
- 仅禁用语音服务：`{"ims_enabled": false}`
- 更改服务配置文件：`{"epc_profile_id": "f47ac10b-58cc-4372-a567-0e02b2c3d479"}`（见 EPC 配置文件）
- 更新漫游策略：`{"roaming_profile_id": "1b671a64-40d5-491e-99b0-da01ff1f3341"}`（见 漫游管理）

删除用户

从系统中移除用户。

端点：`DELETE /api/subscriber/:id`

路径参数：

参数	类型	描述
`id`	string (UUID)	用户数据库 ID

示例请求：

```
curl -k -X DELETE
https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-4b01-
90e6-d701748f0851
```

警告：这将永久删除用户及所有相关状态数据（PDN 会话、通话等）。删除后可以重用 IMSI。

注意：删除用户不会删除相关的：

- ��钥集 - 可以重用于其他用户
- SIM - 可以重新分配给新用户
- 配置文件 - 多个用户共享的资源
- MSISDNs - 如果需要，必须单独删除

取消位置请求（强制分离）

发送取消位置请求（CLR）以强制将用户从当前注册的 MME 中分离。

端点：`POST /api/subscriber/cancel_location`

请求体：

```
{
  "imsi": "001001123456789"
}
```

参数：

参数	类型	必需	描述
`imsi`	string	是	要分离的用户的 IMSI（14-15 位数字）

示例请求：

```
curl -k -X POST
https://hss.example.com:8443/api/subscriber/cancel_location \
  -H "Content-Type: application/json" \
  -d '{"imsi": "001001123456789"}'
```

成功响应 **(200 OK)**：

```
{
  "data": {
    "message": "取消位置请求成功发送",
    "imsi": "001001123456789",
    "destination_host": "mme01.operator.com",
    "destination_realm": "epc.operator.com"
  }
}
```

错误响应 **(404** 未找到**)**：

```
{
  "error": "用户未找到或当前未在任何 MME 注册"
}
```

行为：

- 向当前注册的 MME 发送 S6a CLR（`subscriber_state.last_seen_mme`）
- 使`Cancellation-Type: subscription_withdrawal`（强制完全分离）
- 设置 `CLR-Flags: {s6a_indicator: 1, reattach_required: 1}`（UE 必须重新认证）
- 如果用户从未注册或 `last_seen_mme` 为 null，则返回 404
- 影响所有与 **IMSI** 关联的 **MSISDNs**（同一物理设备/SIM）

用例：

- 防止欺诈：立即分离可疑用户
- 订阅终止：当账户被禁用时强制注销
- 故障排除：清除过时的 MME 注册以进行调试
- 迁移：强制重新认证以应用新的配置文件设置
- 安全：立即断开被攻击的用户

多 **IMSI** 考虑：

在使用 CLR 的多 MSISDN 场景中：

1. 多个 **MSISDN**，单个 **IMSI**：

```
// 用户的 IMSI 为 001001123456789，MSISDNs 为 ["+1234567890",
"+9876543210"]
POST /api/subscriber/cancel_location
{"imsi": "001001123456789"}

// 结果： 发送一个 CLR，两个 MSISDN 受影响（同一设备）
```

2. 不同 **IMSIs**（不同设备）：

```
// 两个用户具有相同的 MSISDN，但不同的 IMSIs（号码移植场景）
// 用户 A: IMSI 001001111111111, MSISDN "+1234567890"
// 用户 B: IMSI 001001222222222, MSISDN "+1234567890"

POST /api/subscriber/cancel_location
{"imsi": "001001111111111"}

// 结果： 仅用户 A 被分离，用户 B 不受影响
```

**要说明：

- 基于 **IMSI**：CLR 始终按 IMSI 发送，而不是按 MSISDN
- 异步：CLR 是异步发送的；成功响应意味着 CLR 已发送，而不是 MME 已处理
- 不验证 **MME** 状态：即使 MME 无法访问，仍会发送 CLR（标准 HSS 行为）
- 幂等性：对同一 IMSI 多次调用是安全的

相关文档：

- 取消位置请求协议流程
- 多 IMSI 场景
- S6a 接口架构

Sh RepositoryData（透明数据）

为用户的 Sh 透明文档（MMTEL-服务、IMS-ODB-信息等）提供配置访问。这些是应用服务器通过 Diameter Sh 接口读取和写入的相同不透明文档。访问方式为 `GET`/`PUT`/`DELETE` `/api/subscriber/repository_data/:imsi`。

请参见 **Sh RepositoryData（透明数据）** 以获取数据模型、消息流、**SequenceNumber** 并发规则和完整的 **REST API** 参考。

MSISDN 管理

MSISDN（电话号码）可以分配给用户以启用语音服务。有关将多个号码分配给单个用户的详细信息，请参见 多 MSISDN 文档。

`shared` 标志

携带 MSISDN 的读取响应包括一个计算的只读 `shared` 布尔值。当号码被 多个用户 持有时（例如，一个手机和一个在不同 SIM 上的智能手表的多 IMSI 共享号码），它为 `true`，否则为 `false`。它出现在 MSISDN 实体（`GET /api/msisdn` 和 `GET /api/msisdn/:id`）、嵌入用户的每个号码（`GET /api/subscriber/:id`）以及通过 `GET /api/subscriber/msisdn/:msisdn` 返回的用户号码中。它从不在写入时接受；HSS 会计算它。请参见 跨接口的 MSISDN 解析。

列出 **MSISDNs**

检索所有电话号码。

端点：`GET /api/msisdn`

示例请求：

```
curl -k https://hss.example.com:8443/api/msisdn
```

获取 MSISDN

检索特定电话号码。

端点：`GET /api/msisdn/:id`

示例请求：

```
curl -k https://hss.example.com:8443/api/msisdn/16fd2706-8baf-433b-82eb-8c7fada847da
```

示例响应：

```
{
  "status": "success",
  "response": {
    "id": "16fd2706-8baf-433b-82eb-8c7fada847da",
    "msisdn": "14155551234",
    "shared": false
  }
}
```

创建 MSISDN

创建一个新的电话号码。

端点：`POST /api/msisdn`

请求体：

```
{
  "msisdn": {
    "msisdn": "14155551234"
  }
}
```

验证：

- 必须为 1-15 位数字
- 必须��–
- 必须遵循 E.164 格式（国际格式，不带 + 符号）

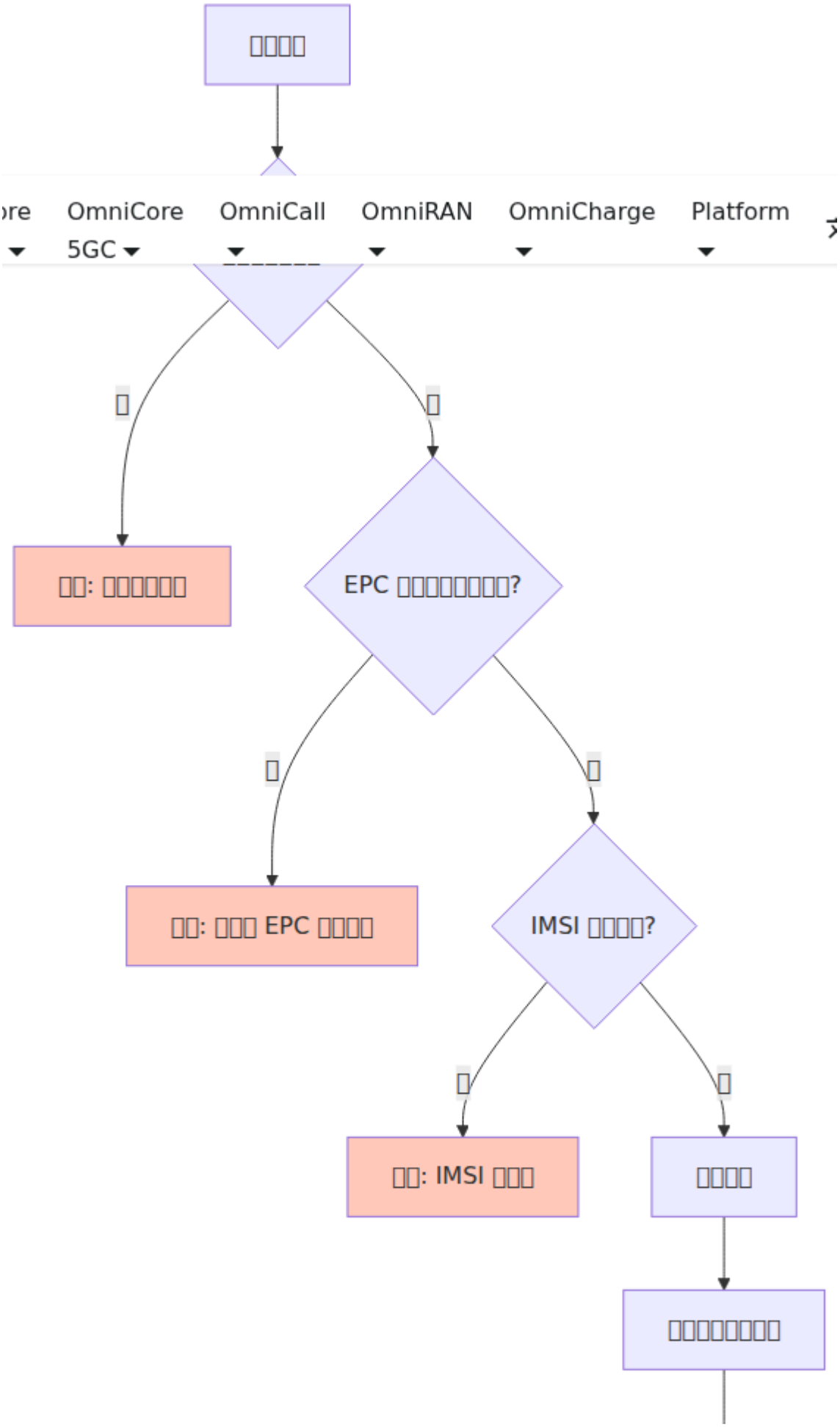
示例请求：

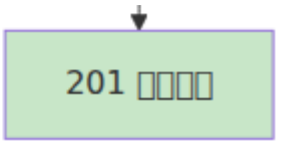
```
curl -k -X POST https://hss.example.com:8443/api/msisdn \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": {
      "msisdn": "14155551234"
    }
  }'
```

将 **MSISDN** 分配给用户

要将电话号码分配给用户，您需要创建一个连接记录。这通常通过用户更新端点或直接数据库操作完成。

多 **MSISDN** 模式：


□□□□
ore
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
□
□
□□: □□□□□□□
EPC □□□□□□□□□?
□
□
□□: □□□ EPC □□□□
IMSI □□□□?
□
□
□□: IMSI □□□
□□□□
□□□□□□□□□



请参见 多 MSISDN 和多 IMSI 功能 以获取详细用法。

删除 MSISDN

移除一个电话号码。

端点：`DELETE /api/msisdn/:id`

示例请求：

```
curl -k -X DELETE
https://hss.example.com:8443/api/msisdn/16fd2706-8baf-433b-82eb-
8c7fada847da
```

IMS 订阅管理

IMS 订阅将多个用户（IMSI / IMPIs）分组到一个 IMS 订阅中，该订阅共享一个公共身份和一个单一的服务 S-CSCF。请参见 共享公共用户身份。

列出 IMS 订阅

端点：`GET /api/ims_subscription`

```
curl -k https://hss.example.com:8443/api/ims_subscription
```

获取 IMS 订阅

端点：`GET /api/ims_subscription/:id`

```
curl -k
https://hss.example.com:8443/api/ims_subscription/16fd2706-8baf-
433b-82eb-8c7fada847da
```

创建 IMS 订阅

端点：`POST /api/ims_subscription`

```
curl -k -X POST https://hss.example.com:8443/api/ims_subscription
\
  -H "Content-Type: application/json" \
  -d '{"name": "Smith family - shared number"}'
```

字段：

- `name` - 可读名称（必填，唯一）
- `ims_profile_id` - 可选的共享 IMS 配置文件，用于该组的共享身份

`assigned_scscf` 和 `last_seen_scscf_timestamp` 是由 HSS 写入的运行时状态，不由客户端设置。

更新 IMS 订阅

端点：`PATCH /api/ims_subscription/:id`

删除 IMS 订阅

端点：`DELETE /api/ims_subscription/:id`

删除 IMS 订阅不会删除其用户；它们的 `ims_subscription_id` 会被清除。要将用户添加到组中，请在用户上设置 `ims_subscription_id`（请参见 更新用户）。

SIM 管理

SIM 卡记录存储物理 SIM 卡信息，括 ICCID、供应商详细信息、PIN/PUK 代码和 OTA 密钥。SIM 记录可以选择性地链接到用户。

另见：

- 多 IMSI 文档 - 在一张物理 SIM 上的多个用户

列出 SIMs

检索所有 SIM 卡。

端点：`GET /api/sim`

示例请求：

```
curl -k https://hss.example.com:8443/api/sim
```

获取 SIM

检索特定 SIM 卡。

端点：`GET /api/sim/:id`

示例请求：

```
curl -k https://hss.example.com:8443/api/sim/3f2504e0-4f89-41d3-
9a0c-0305e82c3301
```

创建 SIM

创建一个新的 SIM 卡记录。

端点：`POST /api/sim`

请求体：

```
{
  "sim": {
    "iccid": "89911012000003204510",
    "sim_vendor": "Gemalto",
    "batch_name": "2025-Q1-Batch-01",
    "is_esim": false,
    "pin1": "1234",
    "pin2": "5678",
    "puk1": "12345678",
    "puk2": "87654321",
    "adm1": "admin-code-1",
    "kic": "0123456789ABCDEF0123456789ABCDEF",
    "kid": "FEDCBA9876543210FEDCBA9876543210"
  }
}
```

必填字段：

- `iccid` - 19-20 位数字，唯一

可选但重要的字段：

- `sim_vendor` - 制造商名称
- `batch_name` - 用于跟踪
- `is_esim` - eSIM 的布尔标志
- `pin1`, `pin2` - 最终用户 PIN 码
- `puk1`, `puk2` - PIN 解锁码
- `adm1-adm10` - 管理代码
- `kic`, `kid` - OTA 安全密钥（十六进制字符串）

示例请求：

```
curl -k -X POST https://hss.example.com:8443/api/sim \
  -H "Content-Type: application/json" \
  -d '{
    "sim": {
      "iccid": "89911012000003204510",
      "sim_vendor": "Gemalto"
    }
  }'
```

更新 SIM

修改 SIM 卡数据。

端点: `PUT /api/sim/:id`

示例请求:

```
curl -k -X PUT https://hss.example.com:8443/api/sim/3f2504e0-4f89-
41d3-9a0c-0305e82c3301 \
  -H "Content-Type: application/json" \
  -d '{
    "sim": {
      "batch_name": "Updated-Batch-Name"
    }
  }'
```

删除 SIM

移除 SIM 卡记录。

端点: `DELETE /api/sim/:id`

警告: 确保在删除之前没有用户引用此 SIM。

密钥集管理

密钥集□含用于通过 Milenage 算法进行用户身份验证的加密材料（Ki、OPC/OP、AMF、SQN）。每个 用户 必须引用一个密钥集。

另见：

- 协议流程 - 使用密钥集的身份验证程序

列出密钥集

检索所有加密密钥集。

端点：`GET /api/key_set`

示例请求：

```
curl -k https://hss.example.com:8443/api/key_set
```

获取密钥集

检索特定的密钥集。

端点：`GET /api/key_set/:id`

示例请求：

```
curl -k https://hss.example.com:8443/api/key_set/6ba7b810-9dad-
41d1-80b4-00c04fd430c8
```

响应示例：

```
{
  "data": {
    "id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
    "ki": "0123456789ABCDEF0123456789ABCDEF",
    "opc": "FEDCBA9876543210FEDCBA9876543210",
    "op": null,
    "amf": "8000",
    "sqn": 0,
    "authentication_algorithm": "milenage",
    "ota_counter": 0
  }
}
```


创建密钥集

创建一个新的加密密钥集。

端点: `POST /api/key_set`

请求体:


```
{
  "key_set": {
    "ki": "0123456789ABCDEF0123456789ABCDEF",
    "opc": "FEDCBA9876543210FEDCBA9876543210",
    "amf": "8000",
    "sqn": 0,
    "authentication_algorithm": "milenage"
  }
}
```


必填字段:

- `ki` - 128 位密钥（32 个十六进制字符）
- `opc` 或 `op`（OPC 可以从 OP 导出）
- `authentication_algorithm` - 当前仅支持 "milenage"

可选字段:

- `amf` - 默认值："8000"
- `sqn` - 默认值：0
- `ota_counter` - 默认值：0

密钥格式：

- 所有密钥均为十六进制字符串
- Ki、OPC、OP：32 个十六进制字符（128 位）
- AMF：4 个十六进制字符（16 位）

示例请求：

```
curl -k -X POST https://hss.example.com:8443/api/key_set \
  -H "Content-Type: application/json" \
  -d '{
    "key_set": {
      "ki": "0123456789ABCDEF0123456789ABCDEF",
      "opc": "FEDCBA9876543210FEDCBA9876543210",
      "authentication_algorithm": "milenage"
    }
  }'
```

安全警告：密钥集□含高度敏感的加密材料。相应地保护 API 访问。

更新密钥集

修改现有的密钥集。

端点：`PUT /api/key_set/:id`

警告：更改活动 用户 的密钥将导致身份验证失败。仅在维护窗口或为新用户更新密钥。

影响：更新会立即影响所有使用此密钥集的用户。活动用户将在下次连接尝试时失败身份验证。

删除密钥集

移除一个密钥集。

端点：`DELETE /api/key_set/:id`

警告：确保在删除之前没有 用户 引用此密钥集。首先查询用户以检查引用。

配置文件管理

EPC 配置文件

EPC（演进分组核心）配置文件定义用户的数据服务参数。这些配置文件在创建 用户 时被引用。

列出 **EPC** 配置文件

端点：`GET /api/epc/profile`

获取 **EPC** 配置文件

端点：`GET /api/epc/profile/:id`

创建 **EPC** 配置文件

端点：`POST /api/epc/profile`

请求体：

```
{
  "apn_profiles": [],
  "name": "标准数据计划",
  "network_access_mode": "仅数据包",
  "tracking_area_update_interval_seconds": 600,
  "ue_ambr_dl_kbps": 100000,
  "ue_ambr_ul_kbps": 50000
}
```

字段：

字段	描述	单位	典型值
`name`	配置文件名称	文本	唯一标识符
`ue_ambr_dl_kbps`	下载带宽限制	Kbps	10000-1000000
`ue_ambr_ul_kbps`	上传带宽限制	Kbps	5000-500000
`network_access_mode`	访问类型	字符串	"仅数据□" 或 "数据□和电路"
`tracking_area_update_interval_seconds`	TAU 定时器	秒	600（典型）
`apn_profiles`	APN ���置文件 ID 列表	数组	[] 或 [1, 2, 3]

示例请求：

```
curl -k -X POST https://hss.example.com:8443/api/epc/profile \
  -H "Content-Type: application/json" \
  -d '{
    "apn_profiles": [],
    "name": "高级 100Mbps",
    "network_access_mode": "仅数据包",
    "tracking_area_update_interval_seconds": 600,
    "ue_ambr_dl_kbps": 100000,
    "ue_ambr_ul_kbps": 50000
  }'
```

另见：

- 配置文件文档 - 详细的配置文件配置指南
- 完整用户配置 - 在配置中使用 EPC 配置文件

更新 **EPC** 配置文件

端点：`PUT /api/epc/profile/:id`

注意：对 EPC 配置文件的更改会影响所有使用此配置文件的 用户。活动会话可能需要重新建立。

删除 **EPC** 配置文件

端点：`DELETE /api/epc/profile/:id`

警告：确保在删除之前没有 用户 引用此配置文件。

IMS 配置文件

IMS（IP 多媒体子系统）配置文件定义语音服务参数和初始过滤标准（IFC）供用户使用。这些配置文件在创建启用 IMS 服务的 用户 时被引用。

列出 **IMS** 配置文件

端点：`GET /api/ims/profile`

创建 **IMS** 配置文件

端点：`POST /api/ims/profile`

请求体：

```
{
  "name": "标准 VoLTE",
  "ifc_template": "<IMS-XML-Template-Here>"
}
```

必填字段：

- `name` - 配置文件名称（必须唯一）
- `ifc_template` - 带有 Liquid 模板变量的 IFC（初始过滤标准）XML 模板

IFC 模板变量：

IFC 模板支持以下 Liquid 模板变量，这些变量会动态替换：

变量	描述	示例值
`{{ imsi }}`	用户 IMSI	`001001123456789`
`{{ msisdns }}`	MSISDN 数组（用于循环）	`["14155551234", "14155555678"]`
`{{ mcc }}`	移动国家代码	`001`
`{{ mnc }}`	移动网络代码	`001`

模板渲染工作原理：

IFC 模板作为 **Liquid** 模板（类似于 Jinja2）存储，并在 IMS 操作期间动态渲染：

1. 存储：创建 IMS 配置文件时，模板按原样存储，变量如 `{{ imsi }}` 和 `{% for msisdn in msisdns %}`
2. 验证：API 通过使用测试数据渲染模板来验证其有效性，以确保有效的 XML 语法
3. 运行时渲染：当用户执行 IMS 注册（MAA/SAA）时，HSS：
    - 检索用户的 IMS 配置文件
    - 使用用户的实际数据渲染模板：
        - `{{ imsi }}` → 用户的 IMSI
        - `{{ msisdns }}` → 用户的电话号码
        - `{{ mcc }}` → 配置的移动国家代码
        - `{{ mnc }}` → 配置的移动网络代码
    - 通过 Cx/Diameter 将渲染的 XML 返回给 S-CSCF

模板语法：

```
<!-- 简单的变量替换 -->
{{ imsi }}

<!-- 对数组的循环 -->
{% for msisdn in msisdns %}
  <MSISDN>{{ msisdn }}</MSISDN>
{% endfor %}

<!-- 组合变量 -->
{{ imsi }}@ims.mnc{{ mnc }}.mcc{{ mcc }}.3gppnetwork.org
```

IFC 模板示例:

```
<?xml version="1.0" encoding="UTF-8"?>
<IMSSubscription>
<PrivateID>{{ imsi }}@ims.mnc{{ mnc }}.mcc{{ mcc
}}.3gppnetwork.org</PrivateID>
<ServiceProfile>
{% for msisdn in msisdns %}
<PublicIdentity>
<Identity>sip:{{ msisdn }}@ims.mnc{{ mnc }}.mcc{{ mcc
}}.3gppnetwork.org</Identity>
<Extension>
<IdentityType>0</IdentityType>
</Extension>
</PublicIdentity>
<PublicIdentity>
<Identity>tel:{{ msisdn }}</Identity>
<Extension>
<IdentityType>0</IdentityType>
</Extension>
</PublicIdentity>
{% endfor %}
<InitialFilterCriteria>
<Priority>10</Priority>
<TriggerPoint>
<ConditionTypeCNF>0</ConditionTypeCNF>
<SPT>
<ConditionNegated>0</ConditionNegated>
<Group>0</Group>
<Method>REGISTER</Method>
</SPT>
</TriggerPoint>
<ApplicationServer>
<ServerName>sip:as.ims.mnc{{ mnc }}.mcc{{ mcc
}}.3gppnetwork.org</ServerName>
<DefaultHandling>0</DefaultHandling>
</ApplicationServer>
</InitialFilterCriteria>
</ServiceProfile>
</IMSSubscription>
```

□□□□□curl□□

```
curl -k -X POST https://hss.example.com:8443/api/ims/profile \
  -H "Content-Type: application/json" \
  -d '{
    "name": "default",
    "ifc_template": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<IMSSubscription><ServiceProfile>...</ServiceProfile>
</IMSSubscription>"
  }'
```

示例请求（**Python**）：

```
import requests

response = requests.post(
    "https://hss.example.com:8443/api/ims/profile",
    json={
        "name": "default",
        "ifc_template": ifc_template_string
    },
    verify=False  # 对于自签名证书
)
```

成功响应（**201** 创建）：

```
{
  "status": "success",
  "response": {
    "id": "0f8fad5b-d9cb-469f-a165-70867728950e",
    "name": "default",
    "ifc_template": "<?xml version=\"1.0\" encoding=\"UTF-8\"?
>..."
  }
}
```

验证：

- API 验证 IFC 模板是否为有效的 XML
- 模板变量使用测试数据进行渲染以验证语法
- `name` 字段必须唯一且非空

另见：

- 配置文件文档 - IFC 模板详细信息和示例
- 协议流程 - IMS 注册和呼叫流程
- 默认 IFC 模板 - 参考实现

APN 配置文件

APN（接入点名称）配置文件由三个组件组成，它们共同工作：

1. **APN** 标识符 - 定义 APN 名称和 IP 版本
2. **APN QoS** 配置文件 - 定义服务质量参数
3. **APN** 配置文件 - 结合标识符和 QoS，链接到 EPC 配置文件

另见 **PCRF** 文档 以获取详细的策略配置、**QoS** 管理和自动重新认证。另见 配置文件文档 以获取 APN 配置示例。

列出 **APN** 标识符

端点：`GET /api/apn/identifier`

创建 **APN** 标识符

端点：`POST /api/apn/identifier`

请求体：

```
{
  "apn": "internet",
  "ip_version": "ipv4v6"
}
```

IP 版本值：

- `"ipv4"` - 仅 IPv4
- `"ipv6"` - 仅 IPv6
- `"ipv4v6"` - IPv4v6（双栈）
- `"ipv4_or_ipv6"` - IPv4 或 IPv6（网络选择）

列出 **APN QoS** 配置文件

端点: `GET /api/apn/qos_profile`

创建 **APN QoS** 配置文件

端点: `POST /api/apn/qos_profile`

请求体:

```
{
  "name": "最佳努力互联网",
  "allocation_retention_priority": 8,
  "apn_ambr_dl_kbps": 50000,
  "apn_ambr_ul_kbps": 25000,
  "pre_emption_capability": false,
  "pre_emption_vulnerability": true,
  "qci": 9
}
```

列出 **APN** 配置文件

端点: `GET /api/apn/profile`

创建 **APN** 配置文件

端点: `POST /api/apn/profile`

请求体:

```
{
  "apn_identifier_id": "4fa85f64-5717-4562-b3fc-2c963f66afa6",
  "apn_qos_profile_id": "8d8ac610-566d-4ef0-9c22-186b2a5ed793",
  "name": "互联网 APN"
}
```

必填字段:

- `apn_identifier_id` - 必须引用现有的 APN 标识符
- `apn_qos_profile_id` - 必须引用现有的 APN QoS 配置文件

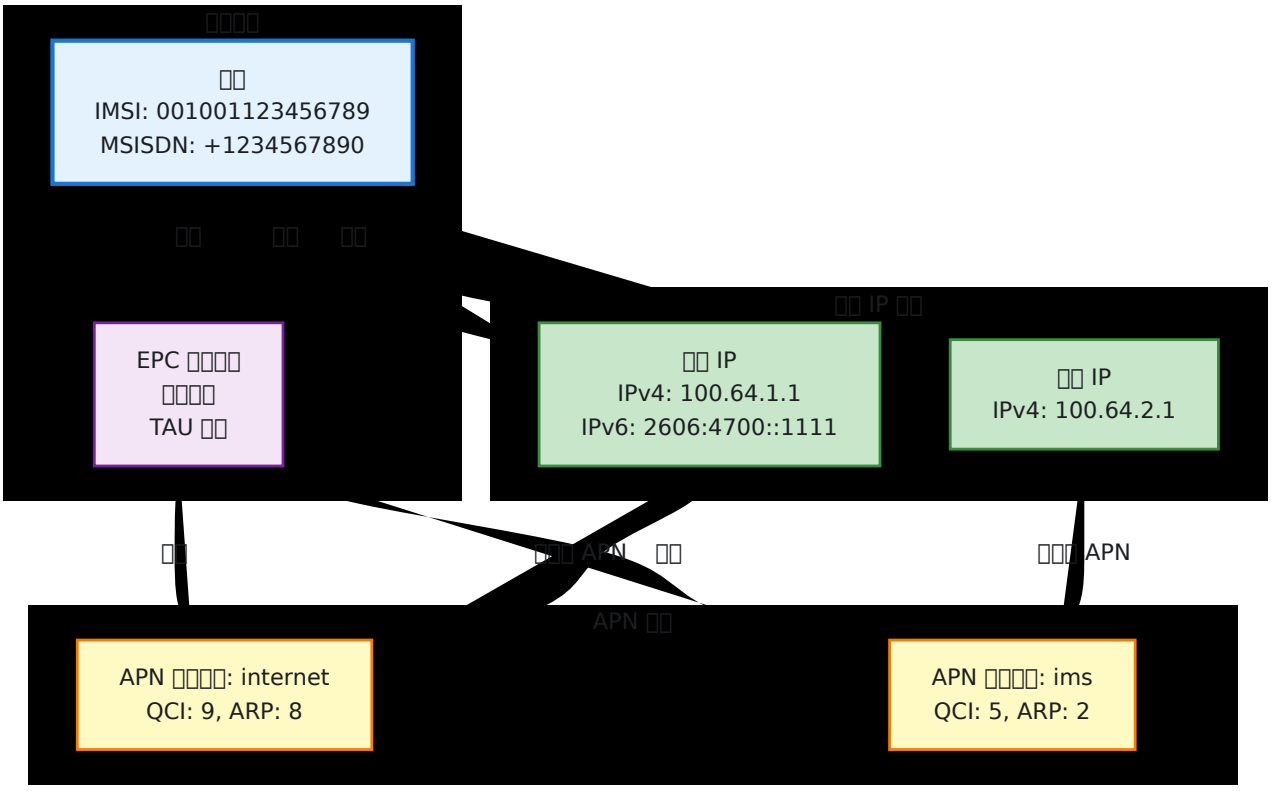
□□□

- □□□□□□□ - □□ APN □□□□□□□□
- EPC □□□□ - APN □□□□□□□□ EPC □□□□

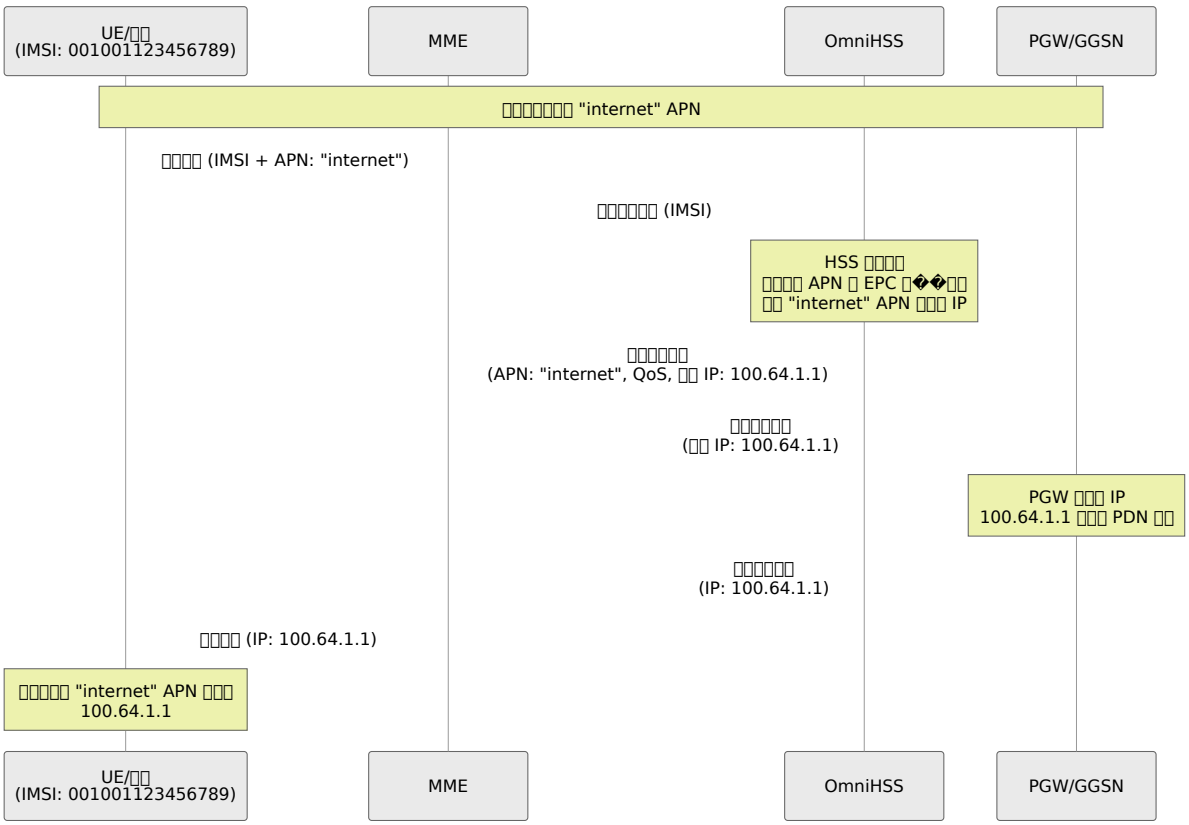
□□ IP □□

□□ IP □□□□□□□□□ APN□□□□□□□□□□□□□□□□□□□□□□□□ APN □□□□□□ IPv4 □/□ IPv6 □□□□□□□□ DHCP □□□□□□□□□□

□□□

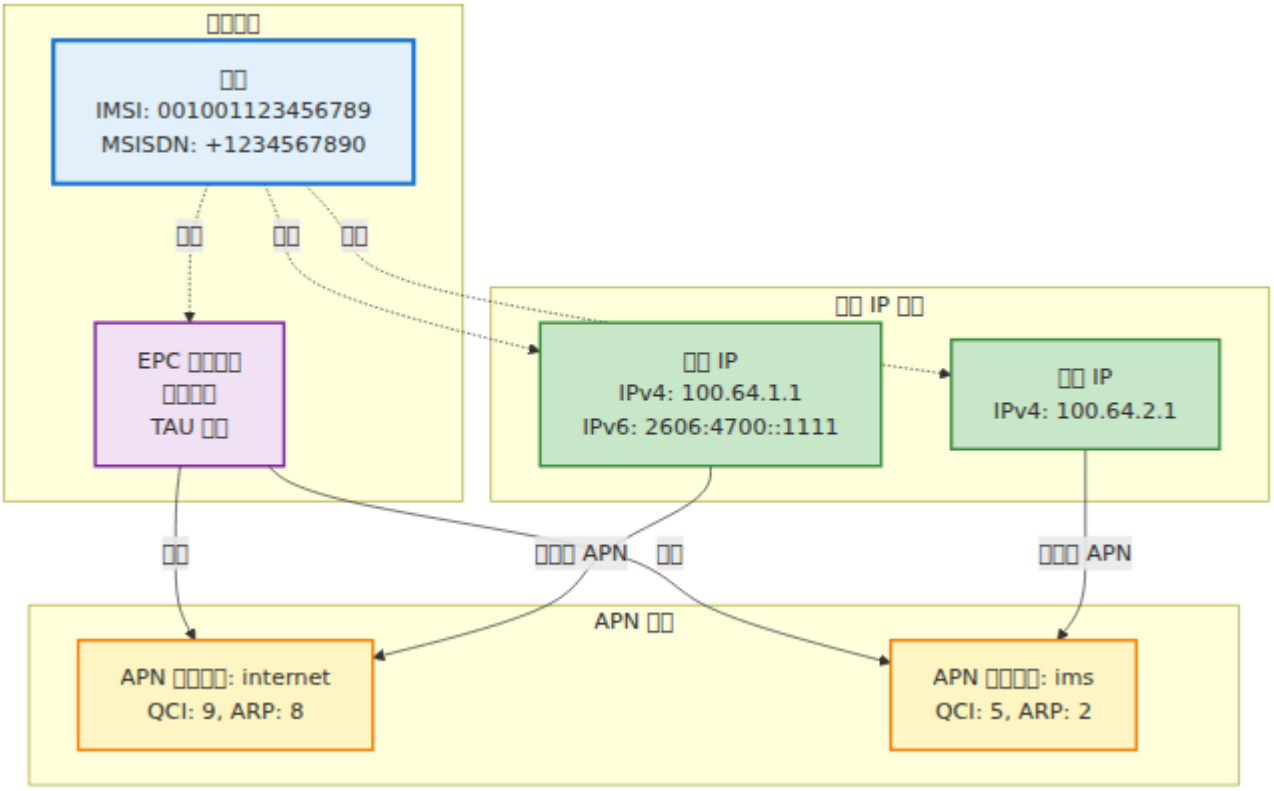


□□□□□□□□□□□



□□□□□□ - **APN** □□□□□□□

□□□□□ S6a □□□□□□ APN-□□ AVP □□□□□□□□□□□□□□□□□



□□□□□

- 每个**APN**分配：每个静态IP与特定的APN配置文件相关联
- 每个订阅者每个**APN**一个**IP**：每个订阅者在每个APN下只能有一个静态IP分配
- **IPv4**和**IPv6**支持：静态IP可以是仅IPv4、仅IPv6或双栈
- 全球**IP**唯一性：每个IP地址在系统中的所有静态IP记录中必须是全球唯一的
  - 相同的IPv4或IPv6地址不能分配给多个订阅者（即使在不同的APN上）
  - 这可以防止路由冲突和IP地址歧义
  - 通过`ipv4_static_ip`和`ipv6_static_ip`字段上的数据库唯一索引强制执行
- 多对多关系：订阅者和静态IP通过连接表关联

用例：

- IoT设备的固定IP地址
- 移动设备上的服务器托管（需要静态IP以进行入站连接）
- 需要特定IP地址的遗留应用程序
- 基于源IP的网络策略路由
- 需要IP地址跟踪的合规性

列出静态**IP**

检索所有静态IP分配。

端点：`GET /api/epc/static_ip`

示例请求：

```
curl -k https://hss.example.com:8443/api/epc/static_ip
```

示例响应：

```
{
  "data": [
    {
      "id": "c56a4180-65aa-42ec-a945-5fd21dec0538",
      "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111",
      "apn_profile": {
        "id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
        "name": "Internet APN",
        "apn_identifier": {
          "apn": "internet",
          "ip_version": "ipv4v6"
        }
      },
      "inserted_at": "2025-11-15T10:30:00Z",
      "updated_at": "2025-11-15T10:30:00Z"
    }
  ]
}
```


获取静态IP

检索特定的静态IP分配。

端点: `GET /api/epc/static_ip/:id`

路径参数:

参数	类型	描述
`id`	string (UUID)	静态IP数据库ID

示例请求:

```
curl -k https://hss.example.com:8443/api/epc/static_ip/c56a4180-65aa-42ec-a945-5fd21dec0538
```

创建静态IP

为APN创建新的静态IP分配。

端点: `POST /api/epc/static_ip`

请求体:

```
{
  "static_ip": {
    "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
    "ipv4_static_ip": "100.64.1.1",
    "ipv6_static_ip": "2606:4700:4700::1111"
  }
}
```

必填字段:

- `apn_profile_id` - 必须引用现有的APN配置文件
- 必须指定至少一个`ipv4_static_ip`或`ipv6_static_ip`

可选字段:

- `ipv4_static_ip` - IPv4地址（点分十进制表示法）
- `ipv6_static_ip` - IPv6地址（标准表示法）

IP格式验证:

- IPv4: 标准点分十进制格式（例如，`100.64.1.1`）
- IPv6: 标准冒号分隔的十六进制格式（例如，`2606:4700:4700::1111`）
- IPv4和IPv6地址必须在所有静态**IP**记录中全球唯一
  - 这可以防止网络中的IP地址冲突
  - 相同的IP不能分配给多个订阅者，即使在不同的APN上
  - 这是通过唯一索引在数据库级别强制执行的约束

配置选项:

配置	IPv4	IPv6	示例
仅IPv4	✓	-	`{"ipv4_static_ip": "100.64.1.1"}`
仅IPv6	-	✓	`{"ipv6_static_ip": "2606:4700:4700::1111"}`
双栈	✓	✓	同时指定两个字段

示例请求:

仅IPv4的静态IP:

```
curl -k -X POST https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
      "ipv4_static_ip": "100.64.1.1"
    }
  }'
```

仅IPv6的静态IP:

```
curl -k -X POST https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "b2c3d4e5-f6a7-4b1c-9d2e-3f4a5b6c7d8e",
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }'
```

双栈的静态IP:

```
curl -k -X POST https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }'
```

成功响应（**201**已创建）：

```
{
  "data": {
    "id": "c56a4180-65aa-42ec-a945-5fd21dec0538",
    "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
    "ipv4_static_ip": "100.64.1.1",
    "ipv6_static_ip": "2606:4700:4700::1111",
    "inserted_at": "2025-11-15T10:30:00Z",
    "updated_at": "2025-11-15T10:30:00Z"
  }
}
```

另请参见：

- 将静态IP分配给订阅者 - 如何将其链接到订阅者
- APN配置文件 - 管理APN配置

更新静态IP

修改现有的静态IP分配。

端点：`PUT /api/epc/static_ip/:id`

路径参数：

参数	类型	描述
`id`	string (UUID)	静态IP数据库ID

请求体:

```
{
  "static_ip": {
    "ipv4_static_ip": "100.64.1.2",
    "ipv6_static_ip": "2606:4700:4700::1112"
  }
}
```

可更新字段:

- `ipv4_static_ip` - 更改IPv4地址
- `ipv6_static_ip` - 更改IPv6地址
- `apn_profile_id` - 更改APN分配

不可更新:

- `id` - 主键（只读）

警告：更改活动订阅者的IP地址将影响其下一个PDN连接。活动PDN会话将继续使用旧IP，直到它们断开并重新连接。

示例请求:

```
curl -k -X PUT
https://hss.example.com:8443/api/epc/static_ip/c56a4180-65aa-42ec-
a945-5fd21dec0538 \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "ipv4_static_ip": "100.64.1.2"
    }
  }'
```

删除静态IP

移除静态IP分配。

端点：`DELETE /api/epc/static_ip/:id`

路径参数：

参数	类型	描述
`id`	string (UUID)	静态IP数据库ID

示例请求：

```
curl -k -X DELETE
https://hss.example.com:8443/api/epc/static_ip/c56a4180-65aa-42ec-
a945-5fd21dec0538
```

行为：

- 移除静态IP分配
- 不影响APN配置文件（APN仍可供其他订阅者使用）
- 使用此静态IP的订阅者将在下次连接时接收动态IP
- IP地址在删除后变为可重用

警告：如果订阅者正在使用此静态IP，删除它将导致他们在下次PDN连接时接收动态IP。确保订阅者处于离线状态或在删除之前发送取消位置请求。

将静态IP分配给订阅者

要将静态IP分配给订阅者，您需要在创建或更新时将静态IP记录与订阅者关联。

分配模式：

1. 创建静态IP（见创建静态IP）
2. 使用`static_ips`字段分配给订阅者

创建带有静态IP的订阅者：

```
# 步骤1: 为“internet”APN创建静态IP
STATIC_IP_ID=$(curl -k -X POST
https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }' | jq -r '.data.id')

# 步骤2: 创建分配了静态IP的订阅者
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriber\": {
      \"imsi\": \"001001123456789\",
      \"key_set_id\": \"6ba7b810-9dad-41d1-80b4-00c04fd430c8\",
      \"epc_profile_id\": \"e58ed763-928c-4155-bee9-
fdbaaadc15f3\",
      \"static_ips\": [$STATIC_IP_ID]
    }
  }"
```

更新现有的带有静态**IP**的订阅者:

```
curl -k -X PUT
https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-4b01-
90e6-d701748f0851 \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "static_ips": ["c56a4180-65aa-42ec-a945-5fd21dec0538",
"5a8f9c3e-2d4b-4e6f-8a1b-3c5d7e9f0a2b"]
    }
  }'
```

多个静态**IP**（不同**APN**）：

一个订阅者可以拥有多个静态IP，只要每个IP对应不同的APN:

```
# 为“internet”APN创建静态IP
INTERNET_IP=$(curl -k -X POST
https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "a1b2c3d4-e5f6-4a3b-8c2d-1e0f9a8b7c6d",
      "ipv4_static_ip": "100.64.1.1"
    }
  }' | jq -r '.data.id')

# 为“ims”APN创建静态IP
IMS_IP=$(curl -k -X POST
https://hss.example.com:8443/api/epc/static_ip \
  -H "Content-Type: application/json" \
  -d '{
    "static_ip": {
      "apn_profile_id": "b2c3d4e5-f6a7-4b1c-9d2e-3f4a5b6c7d8e",
      "ipv4_static_ip": "100.64.2.1"
    }
  }' | jq -r '.data.id')

# 将两个IP都分配给订阅者
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriber\": {
      \"imsi\": \"001001123456789\",
      \"key_set_id\": \"6ba7b810-9dad-41d1-80b4-00c04fd430c8\",
      \"epc_profile_id\": \"e58ed763-928c-4155-bee9-
fdbaaadc15f3\",
      \"static_ips\": [$INTERNET_IP, $IMS_IP]
    }
  }"
```

验证规则：

- ✓ 允许：不同APN的多个静态IP
- ✗ 拒绝：同一APN的多个静态IP

错误示例 - 重复**APN**：

```
# 如果两个静态IP引用相同的APN，这将失败
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "imsi": "001001123456789",
      "static_ips": ["c56a4180-65aa-42ec-a945-5fd21dec0538",
"5a8f9c3e-2d4b-4e6f-8a1b-3c5d7e9f0a2b"]
    }
  }'

# 错误响应：
{
  "errors": {
    "static_ips": [
      "每个订阅者每个APN的静态IP必须是唯一的。例如，订阅者不能为internet分配静态IP 100.64.1.1，同时
也为internet分配静态IP 100.64.1.2"
    ]
  }
}
```

另请参见：

- 创建订阅者 - 订阅者配置
- 更新订阅者 - 修改订阅者配置
- 完整的静态IP配置示例 - 端到端工作流

漫游管理

漫游配置文件控制订阅者是否可以在访问网络上访问数据和IMS服务。配置文件分配给订阅者，并由MCC/MNC匹配的规则组成。

列出漫游配置文件

端点：`GET /api/roaming/profile`

创建漫游配置文件

端点: `POST /api/roaming/profile`

请求体:

```
{
  "roaming_profile": {
    "name": "仅限美国运营商",
    "data_action_if_no_rules_match": "deny",
    "ims_action_if_no_rules_match": "deny",
    "roaming_rules": []
  }
}
```

操作值:

- `"allow"` - 允许
- `"deny"` - 拒绝

默认操作:

- `data_action_if_no_rules_match` - 当没有漫游规则匹配时的操作
- `ims_action_if_no_rules_match` - IMS特定的默认操作

列出漫游规则

端点: `GET /api/roaming/rule`

创建漫游规则

端点: `POST /api/roaming/rule`

请求体:

```
{
  "roaming_rule": {
    "name": "允许AT&T",
    "mcc": "310",
    "mnc": "410",
    "data_action": "allow",
    "ims_action": "allow"
  }
}
```

字段：

- `mcc` - 移动国家代码（3位数字）
- `mnc` - 移动网络代码（2-3位数字）
- `data_action` - `"allow"`或`"deny"`数据服务
- `ims_action` - `"allow"`或`"deny"`IMS/语音服务

另请参见：

- 漫游文档 - 详细配置和示例
- 协议流程 - 漫游控制在Diameter流程中的工作原理

EIR管理

OmniHSS通过S13 Diameter接口作为设备身份注册器（EIR）运行。EIR规则根据IMEI模式控制设备访问。

有关设备身份检查、**S13**接口流程和**IMEI**验证的详细信息，请参见**EIR**文档。

列出**EIR**规则

端点：`GET /api/eir/rule`

创建**EIR**规则

端点：`POST /api/eir/rule`

请求体：

```
{
  "eir_rule": {
    "name": "阻止iPhone 6",
    "imei_regex": "^35[0-9]{6}0[0-9]{7}$",
    "action": 1
  }
}
```

字段：

- `name` - 规则的描述性名称
- `imei_regex` - 匹配IMEI号码的正则表达式
- `action` - 白名单（0）、黑名单（1）或灰名单（2）

操作值：

- `0` - 白名单（允许）
- `1` - 黑名单（拒绝）
- `2` - 灰名单（允许但跟踪）

用例：

- 阻止被盗设备（黑名单特定IMEI）
- 限制设备类型（按TAC模式黑名单）
- 仅允许批准的设备（使用拒绝所有的默认白名单模式）

另请参见：

- 协议流程 - S13接口和EIR检查流程
- 架构概述 - OmniHSS EIR功能

其他文档

有关更多信息，请参见以下文档：

状态和健康 - API健康检查端点
错误处理 - 常见错误和故障排除
API使用示例 - 完整的配置工作流

← 返回操作指南 | 下一步：控制面板 →

API 状态和健康

← 返回 API 参考

系统状态

检查 API 是否响应。

端点：`GET /api/status`

示例请求：

```
curl -k https://hss.example.com:8443/api/status
```

示例响应：

```
{
  "status": "ok"
}
```

用例：负载均衡器和监控系统的健康检查。

← 返回 API 参考

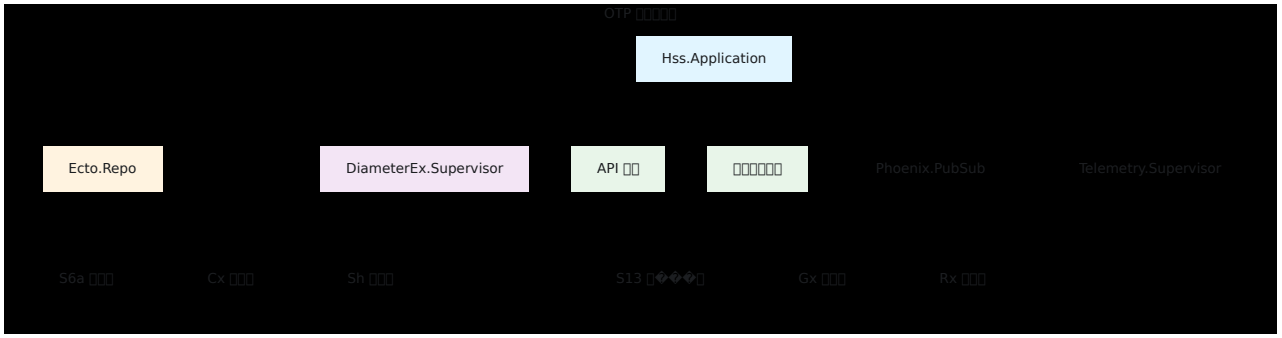
OmniHSS 架构概述
← 返回操作指南

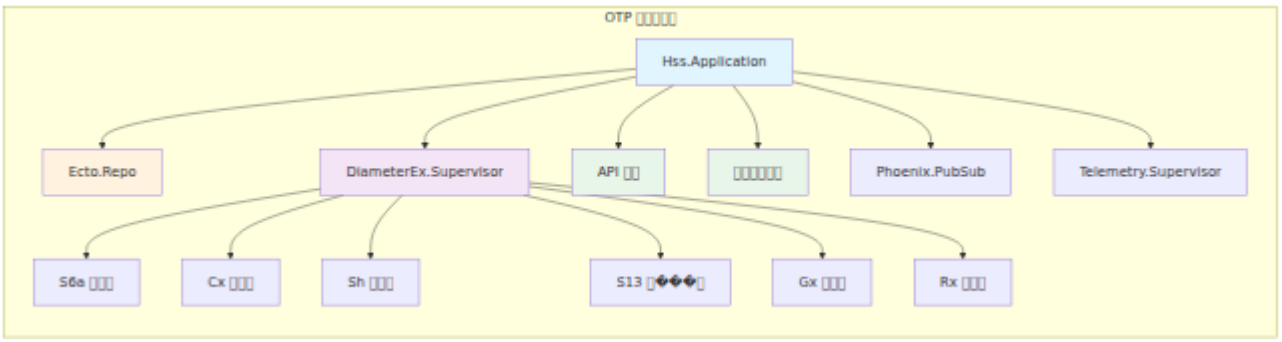
目录

系统概述
组件架构
Diameter 栈
应用层
数据层
外部接口
部署架构

系统概述

OmniHSS 基于 Elixir 和 Erlang/OTP 平台构建，提供一个高度并发、容错的系统，旨在处理电信工作负载。该架构遵循分层方法，具有明确的关注点分离。





组件架构

核心组件

Diameter 应用处理器

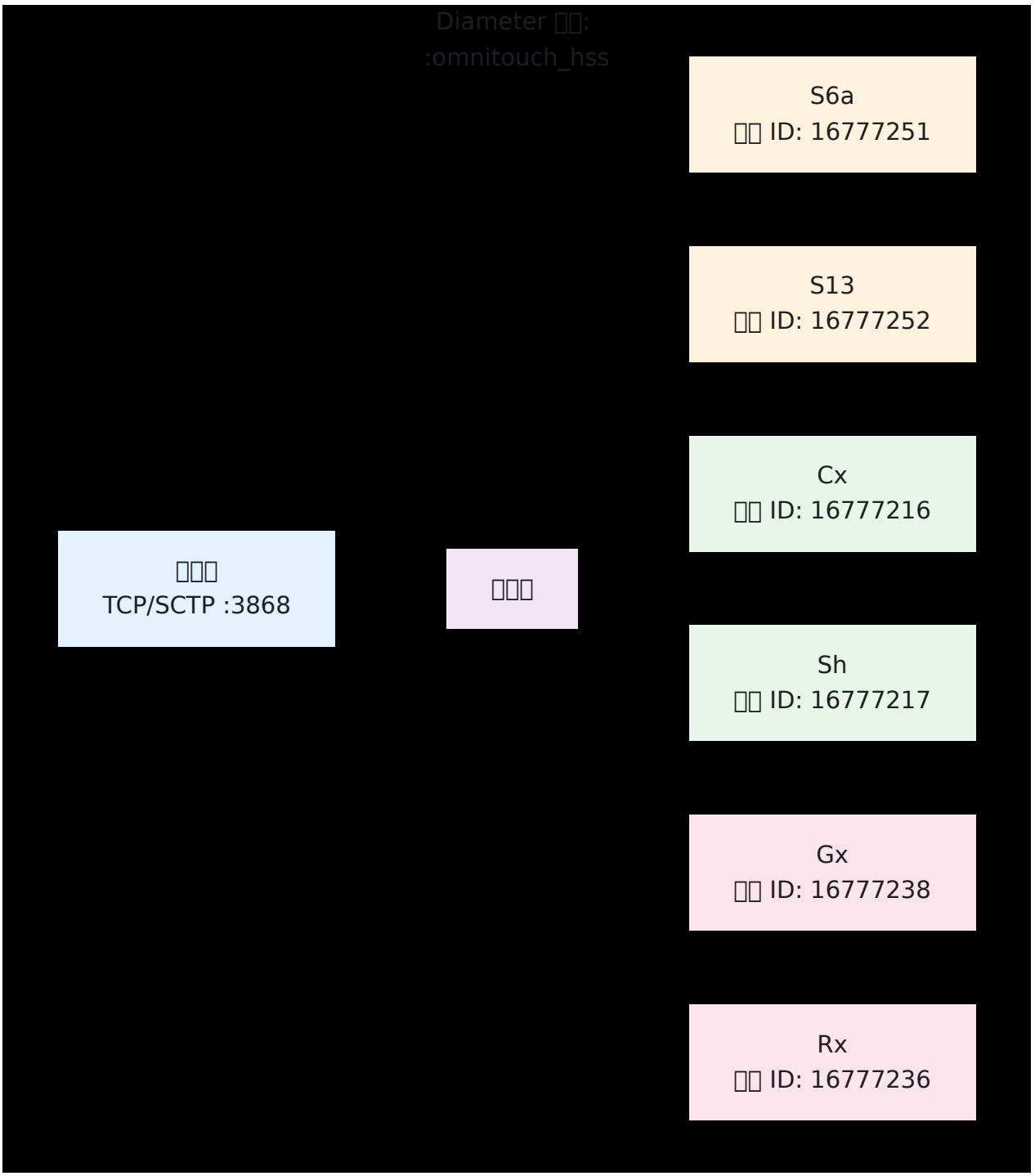
每个 Diameter 应用（S6a, Cx, Sh, S13, Gx, Rx）都实现为一个 DiameterEx 处理器模块，具有以下功能：

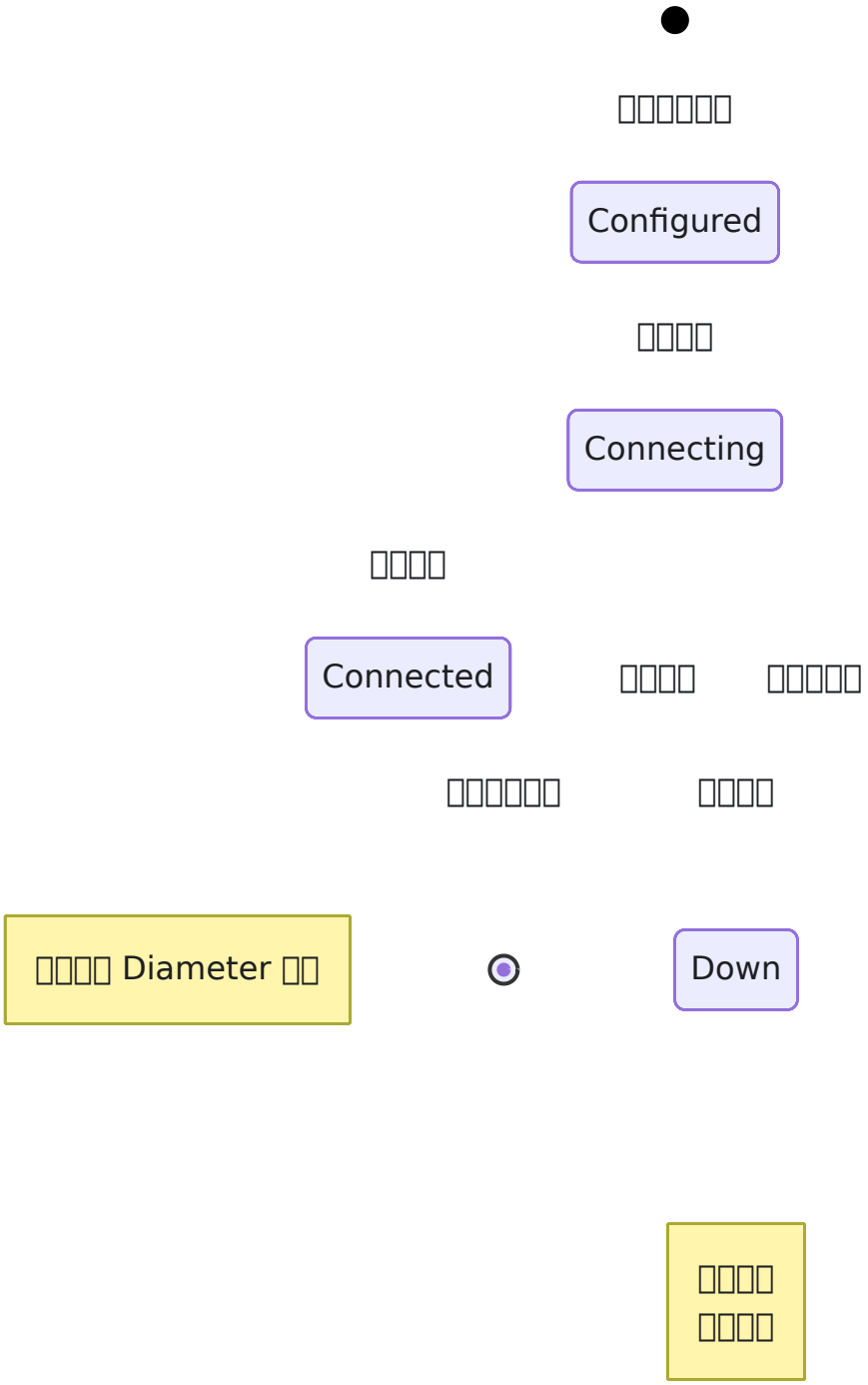
1. 向 **DiameterEx** 注册 - 订阅特定的 Diameter 应用 ID
2. 验证请求 - 提取 AVP，验证订阅者状态
3. 处理业务逻辑 - 调用适当的业务逻辑模块
4. 构建响应 - 使用 AVP 构建 Diameter 答复消息
5. 处理错误 - 返回适当的 Diameter 结果代码

Diameter □

Diameter □□□□

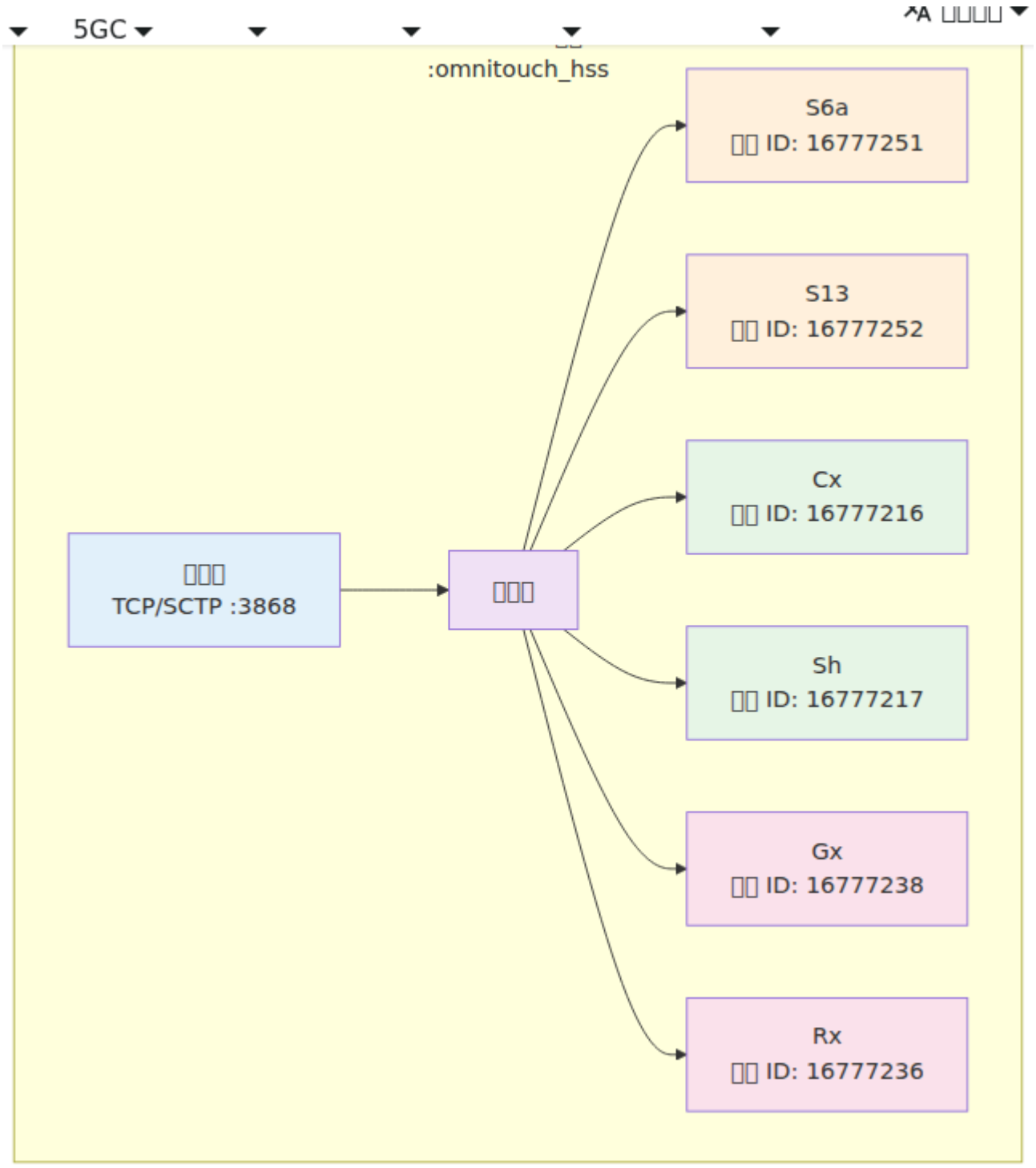
OmniHSS □□□□□□□ Diameter □���□□□□□□□□



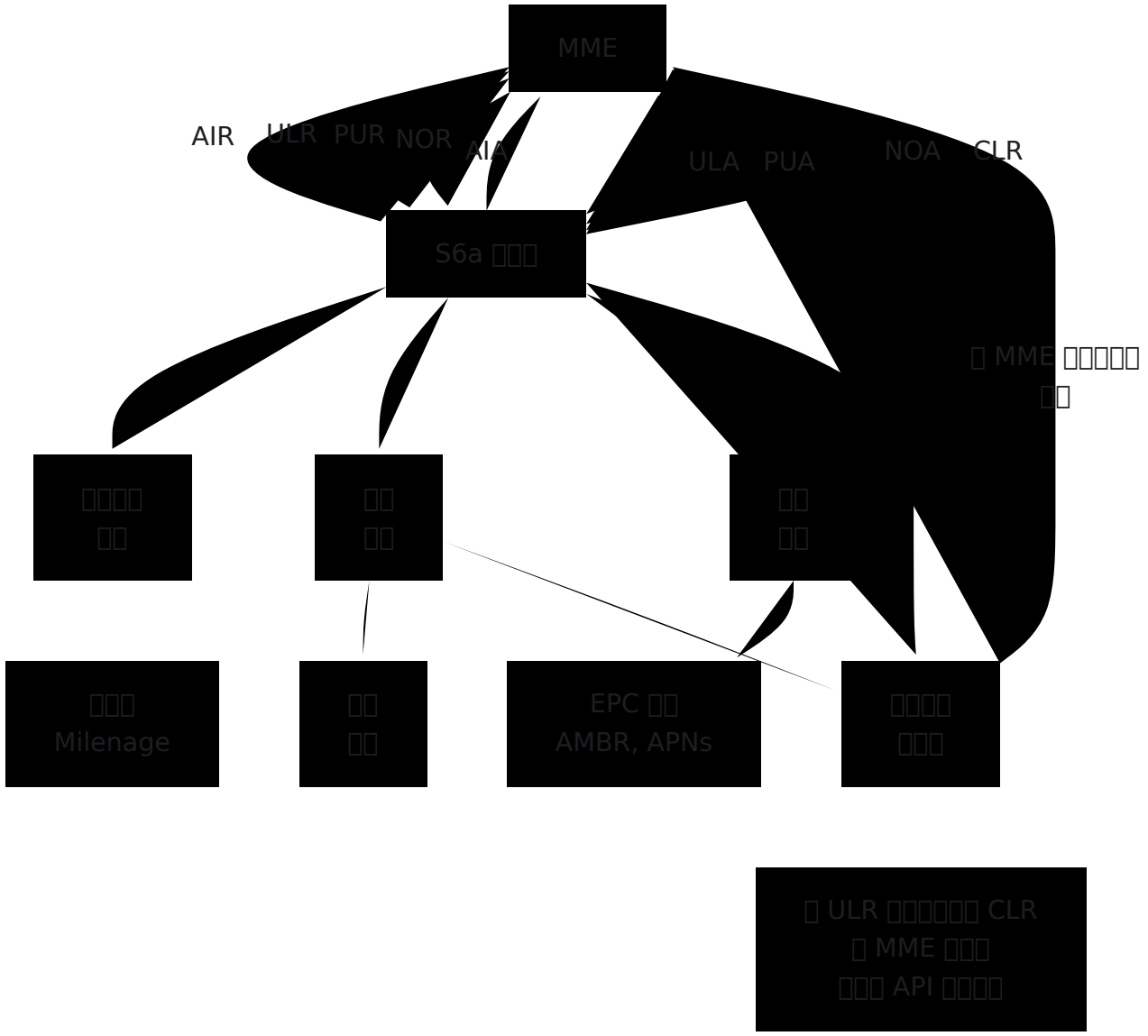


对等连接管理

对等体已添加
Configured
启动连接
Connecting
握手成功
Connected
握手失败
重连计时器
对等体已移除
连接丢失
可以交换 Diameter 消息
Down
自动重连
尝试继续



Diameter 消息流



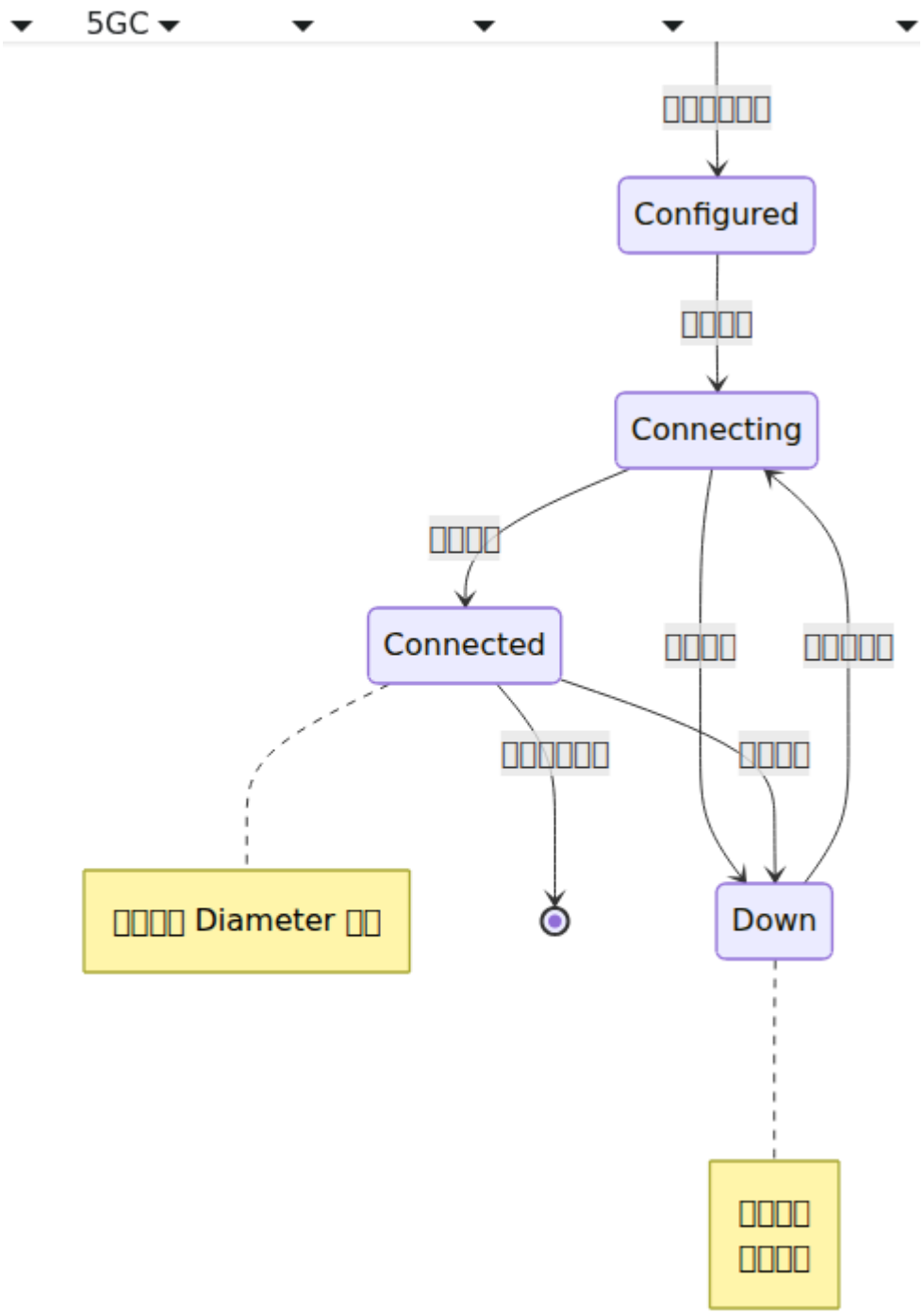
应用层

S6a 接口 (LTE/EPC)

处理 LTE 网络的身份验证和移动管理。

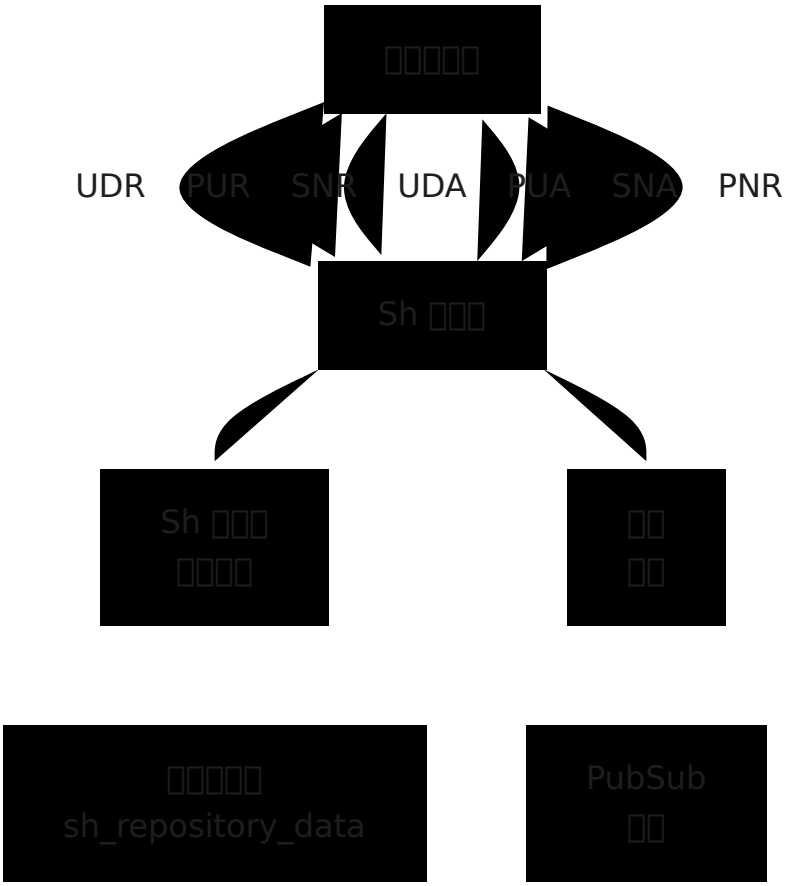
Cx 接口 (IMS)

处理 IMS 注册和身份��证。



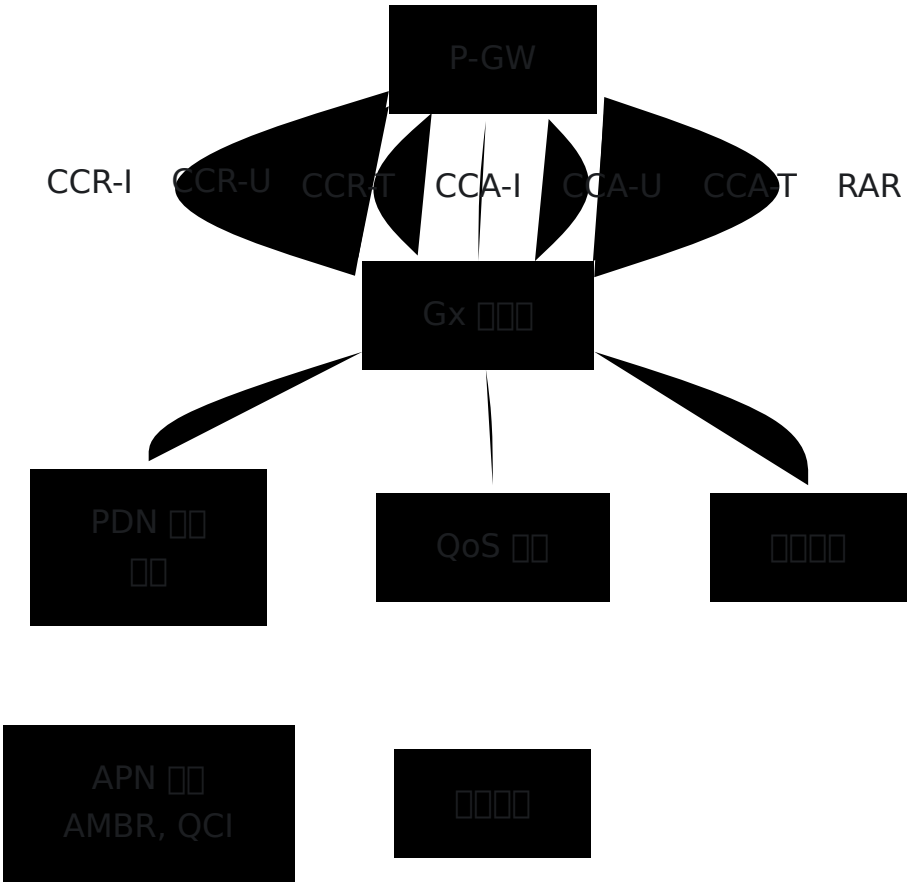
Sh 接口 (IMS 配置数据)

为 IMS 应用服务器提供访问订阅者配置数据的能力。



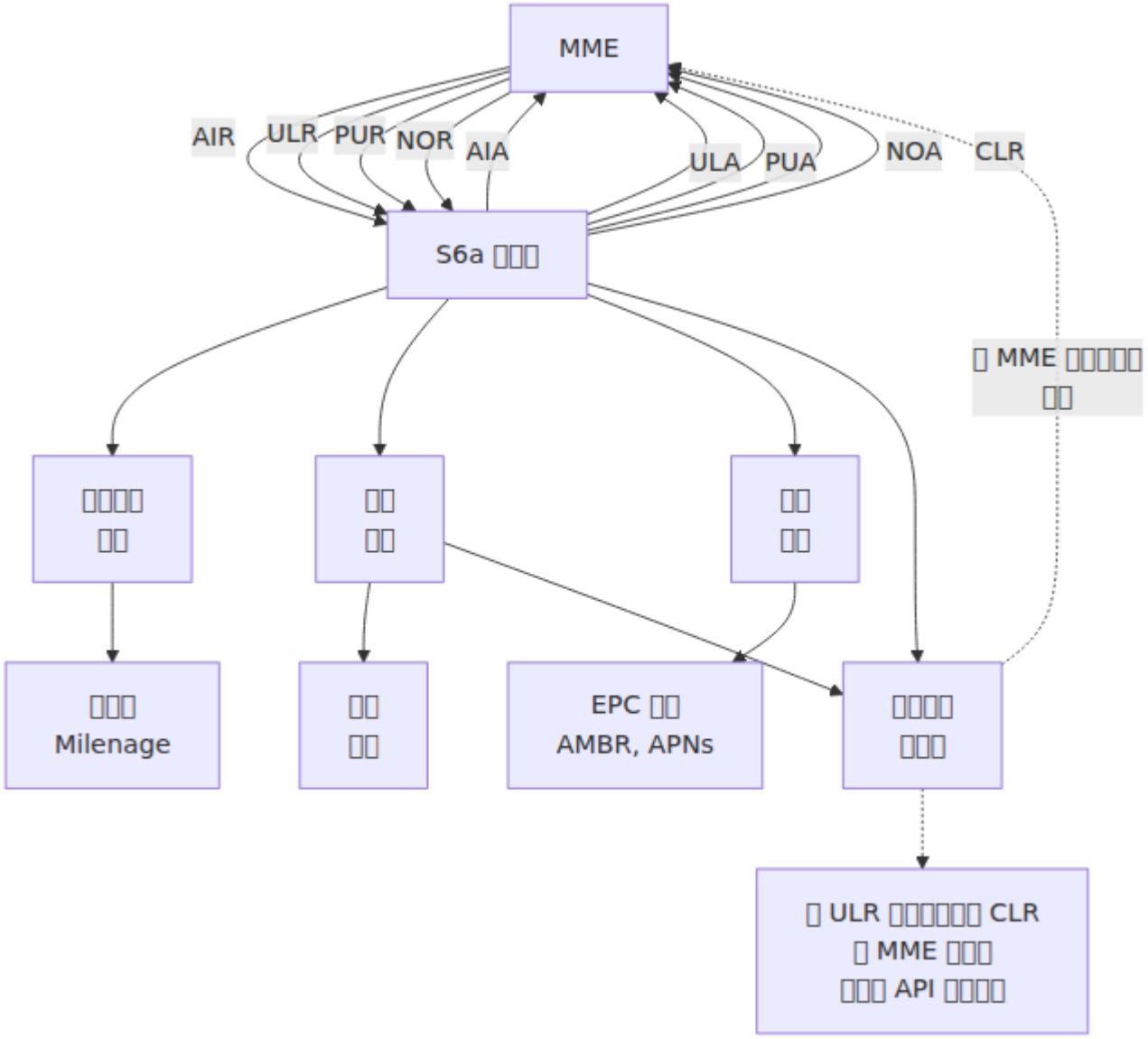
Gx

PCRF



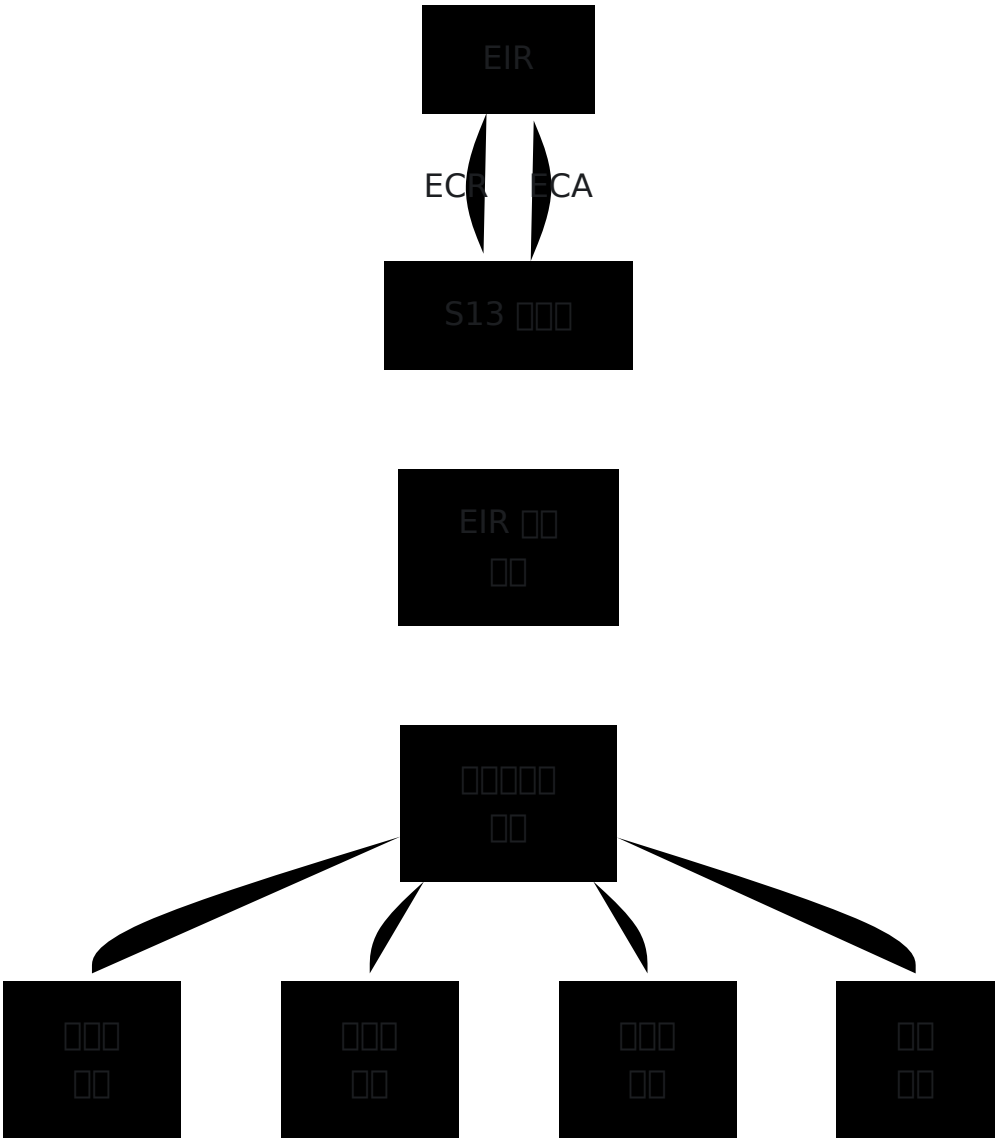
Rx 接口 (IMS 媒体)

控制 IMS 媒体策略和 VoLTE 的专用承载。有关详细信息，请参见 **PCRF** 文档。



S13 接口 (EIR)

根据设备身份规则验证设备 IMEI。有关详细信息，请参见 EIR 文档。



数据层

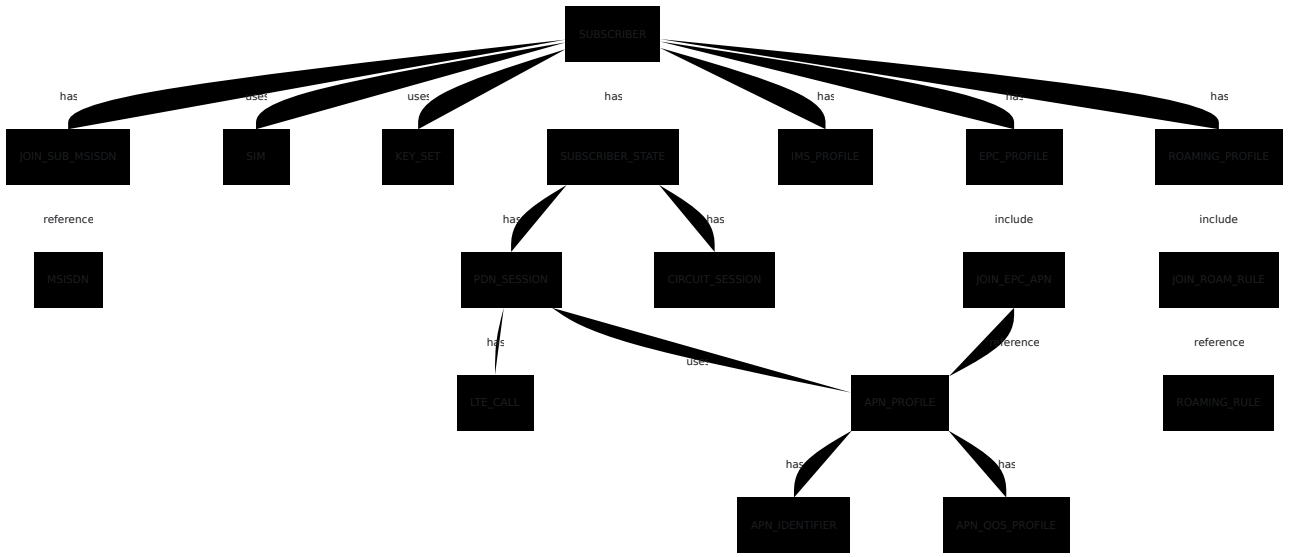
数据库后端

OmniHSS 使用 **PostgreSQL** 作为其数据库后端，并使用 **Ecto** 作为其数据库抽象层。所有表使用 UUID 主键，以便在各个站点之间进行全球唯一标识。

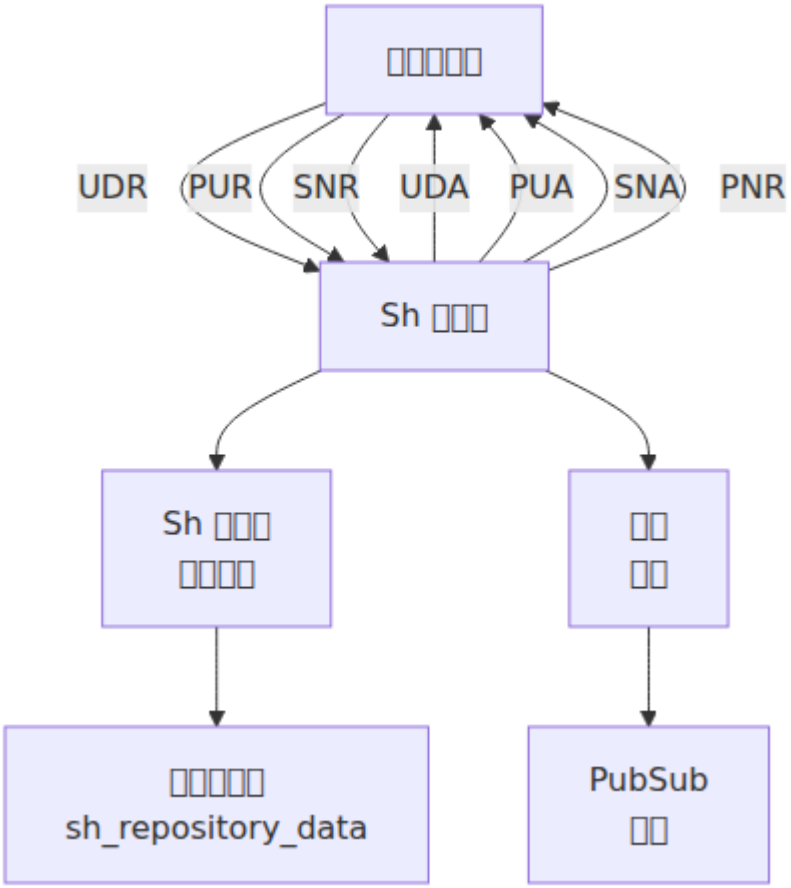
对于多站点部署，OmniHSS 使用 PostgreSQL 的双向逻辑复制。每个站点作为独立的读/写主数据库运行，站点之间使用异步 WAL 基于的复制。

有关多站点复制配置和操作，请参见 数据库复制。

□□□□□□□

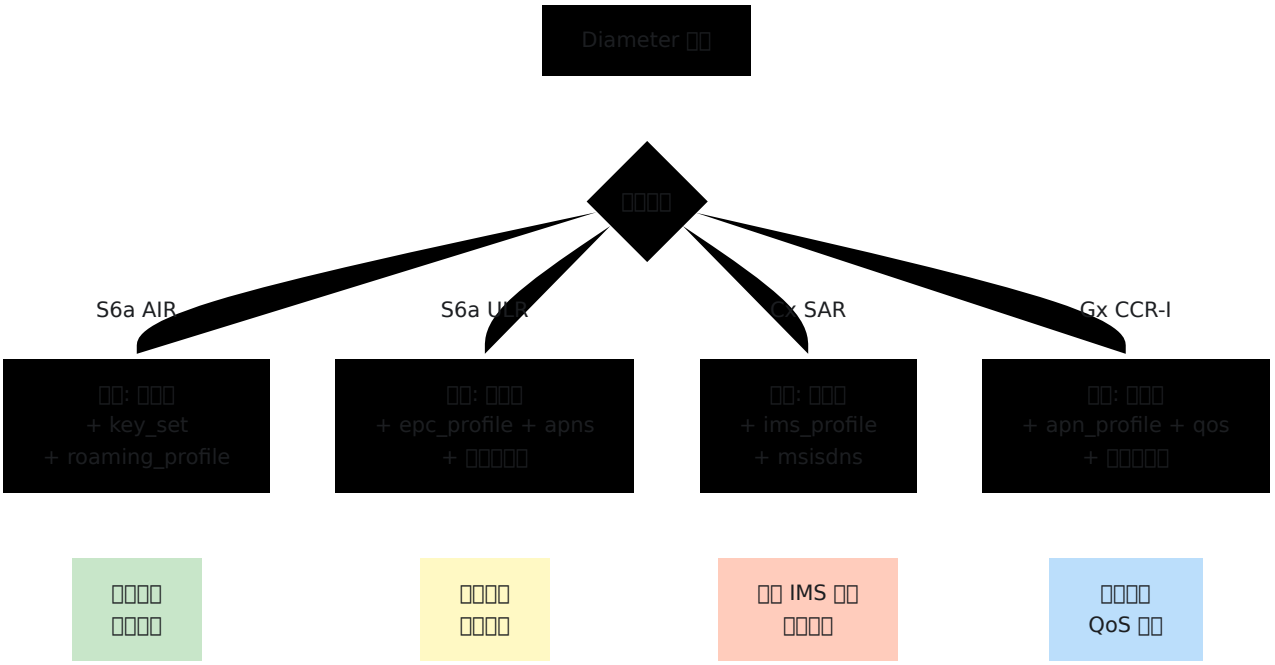

SUBSCRIBER
has
has
has
has
has
JOIN_SUB_MSISDN
SIM
KEY_SET
SUBSCRIBER_STATE
IMS_PROFILE
EPC_PROFILE
ROAMING_PROFILE
reference
include
include
MSISDN
PDN_SESSION
CIRCUIT_SESSION
JOIN_EPC_APN
JOIN_ROAM_RULE
has
reference
reference
LTE_CALL
APN_PROFILE
ROAMING_RULE
has
has
APN_IDENTIFIER
APN_QOS_PROFILE

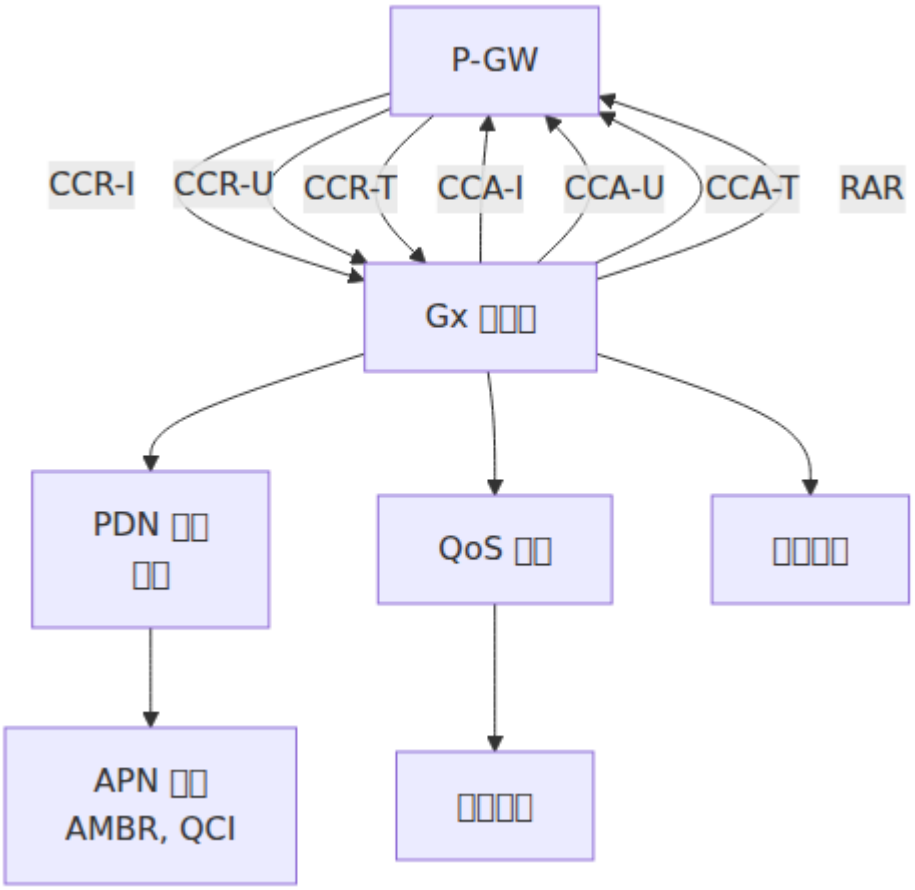
Ecto □□□□


□□□□□
UDR
PUR
SNR
UDA
PUA
SNA
PNR
Sh □□□
Sh □□□
□□□□
□□
□□
□□□□□
sh_repository_data
PubSub
□□

□□□□□□□

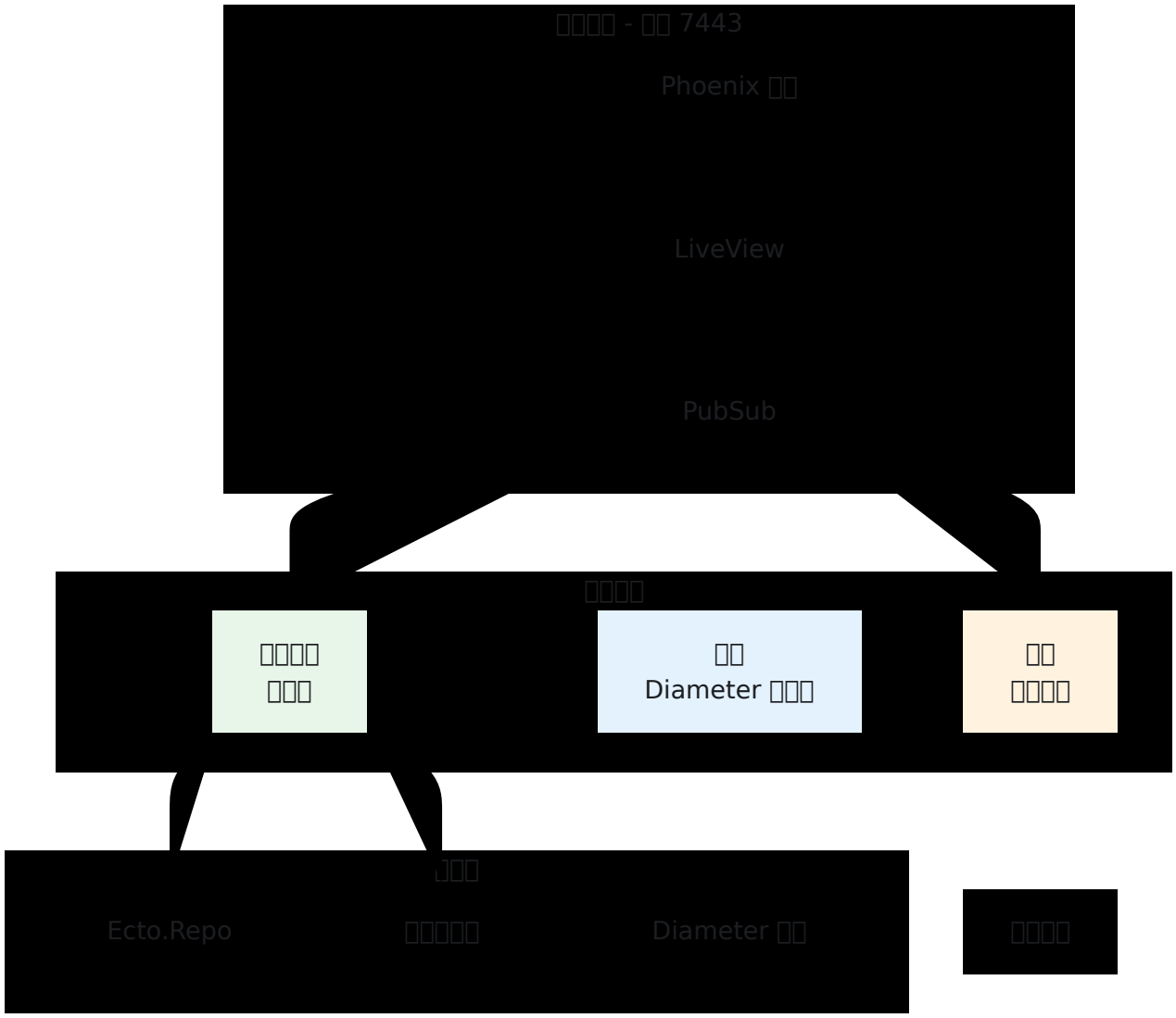
□□ Diameter □□□□□□□□□□□□□□□□□□□□□□□□


Diameter □□
□□□□□
S6a AIR
S6a ULR
Gx SAR
Gx CCR-I
□□: □□□□
+ key_set
+ roaming_profile
□□: □□□□
+ epc_profile + apns
+ □□□□□□
□□: □□□□
+ ims_profile
+ msisdns
□□: □□□□
+ apn_profile + qos
+ □□□□□□
□□□□
□□□□
□□□□
□□□□
□□ IMS □□
□□□□
□□□□
QoS □□



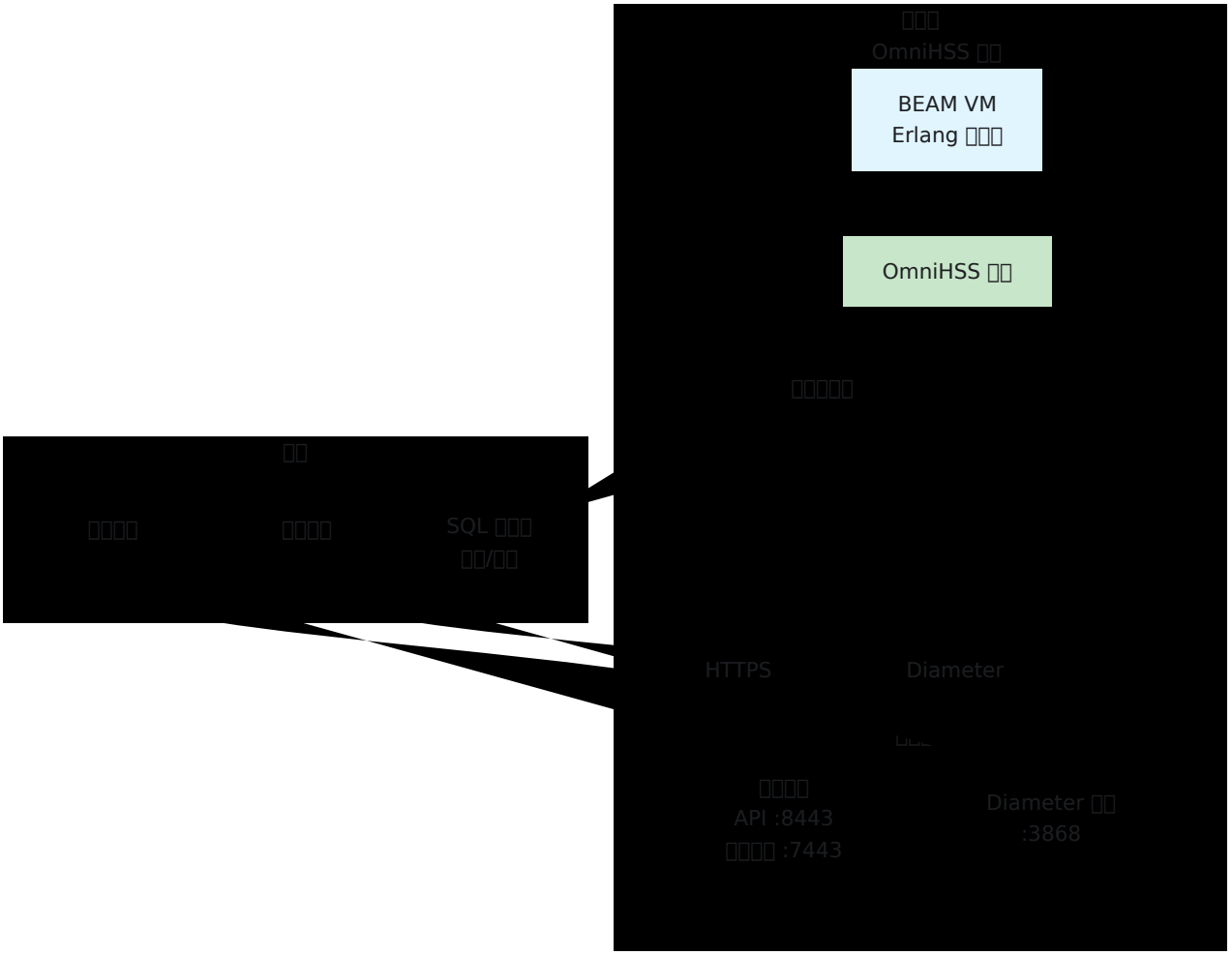
外部接口

API 架构



控制面板架构

Phoenix
LiveView
PubSub
实时概述
订阅者
实时
Diameter 对等体
实时
应用资源
Ecto.Repo
Diameter

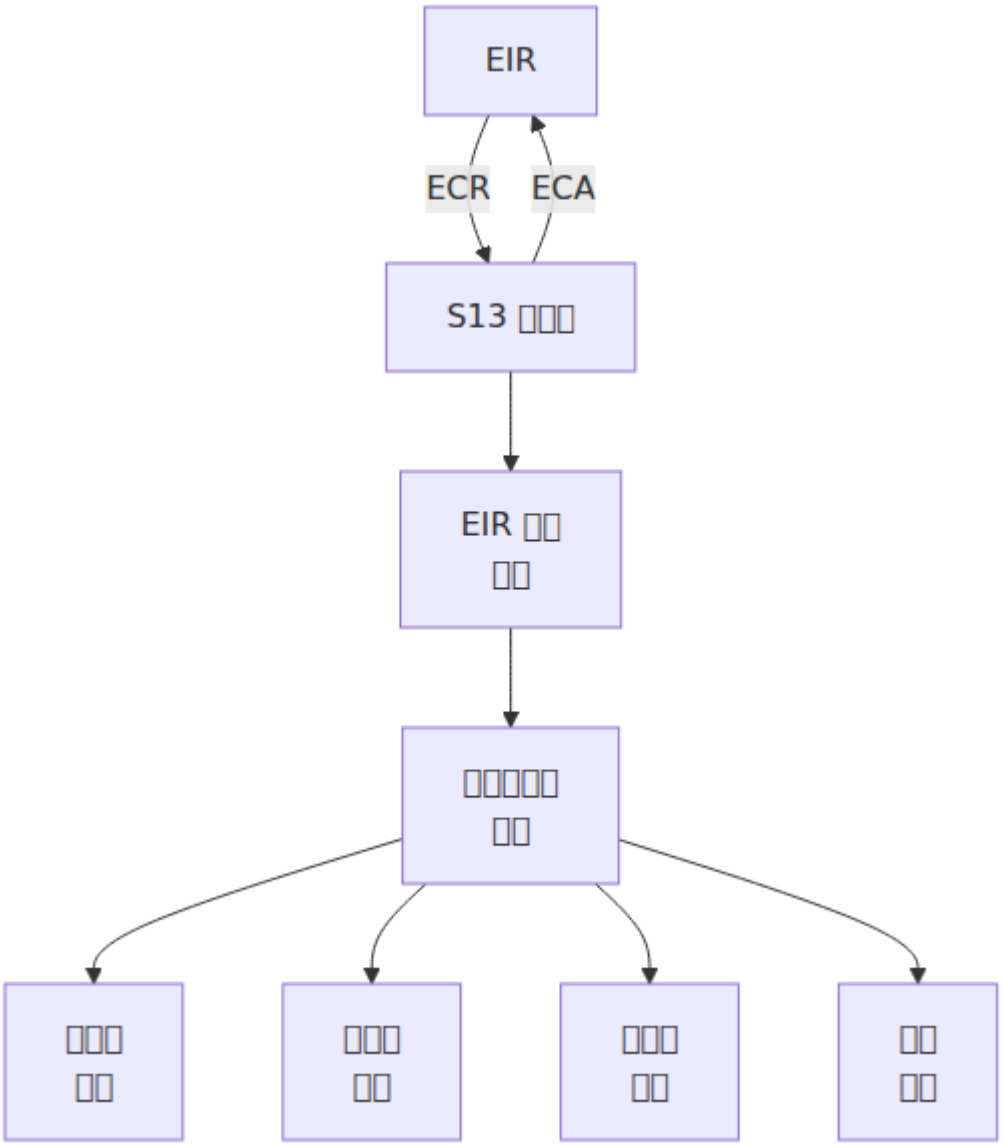


部署架构

单节点部署

多站点高可用部署（逻辑复制）

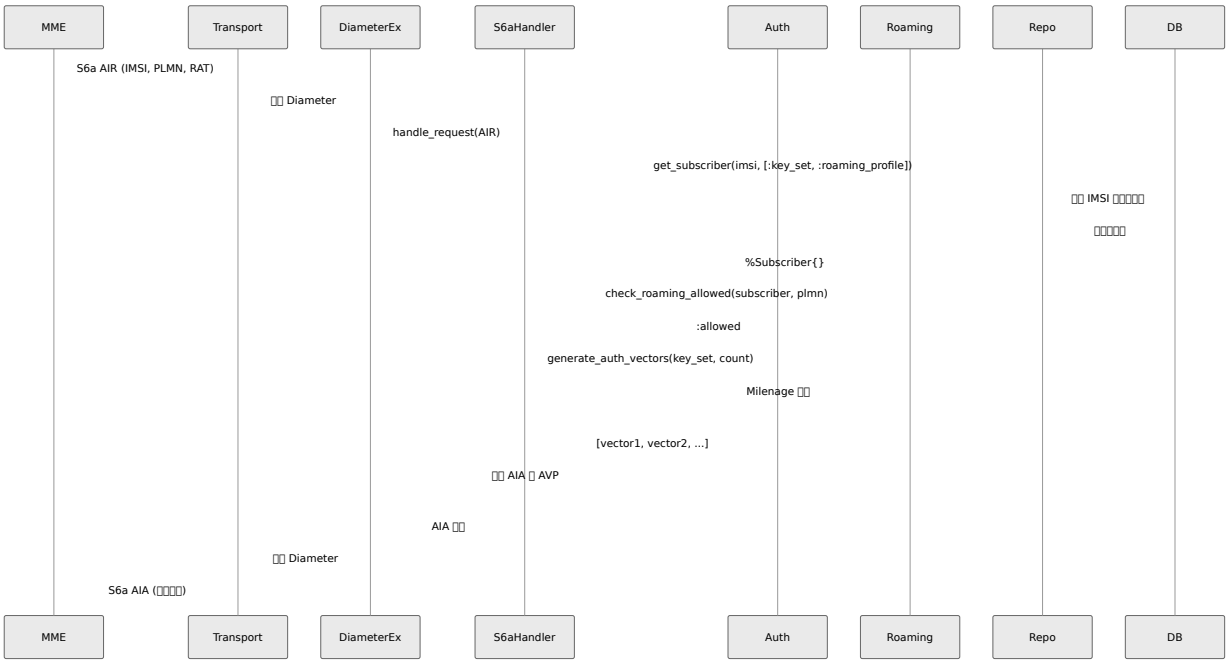
对于跨地理站点的高可用部署，OmniHSS 使用 PostgreSQL 双向逻辑复制。每个站点都有一个独立的读/写数据库。



关键特性：

- 两个站点均可读/写：本地写入即时，永不被远程站点阻塞
- 异步复制：更改通过 WAL 流式传输。复制不会增加写延迟惩罚
- 分区容忍：如果站点间链接断开，两个站点继续运行。WAL 本地排队并在恢复时赶上
- 无法 **quorum**：不需要最小节点计数。一个站点或多个站点均可正常工作
- **UUID** 主键：全球唯一 ID 消除站点之间的冲突

有关详细配置和操作，请参见 数据库复制。



流程示例：身份验证

此示例显示身份验证请求的完整流程：

关键架构原则

1. 容错性

- Erlang/OTP 监控树自动重启失败的进程
- 隔离的 Diameter 处理器防止级联故障
- 数据库连接池具有自动重连功能

2. 并发性

- 每个 Diameter 请求在其自己的进程中处理
- 请求处理器之间没有共享状态
- 数据库连接池用于并行查询

3. 模块化

- 每个 Diameter 应用在单独的模块中
- 接口、业务逻辑和数据层之间有明确的分离
- 可插拔的认证算法

4. 性能

- 优化的数据库查询，选择性预加载
- 每种过程类型的最小数据传输
- 连接池和保持活动

5. 可观察性

- 通过控制面���进行实时监控
- 应用程序中结构化日志记录
- Diameter 对等体状态跟踪
- 带时间戳的订阅者状态跟踪

← 返回操作指南 | 下一步：配置 →

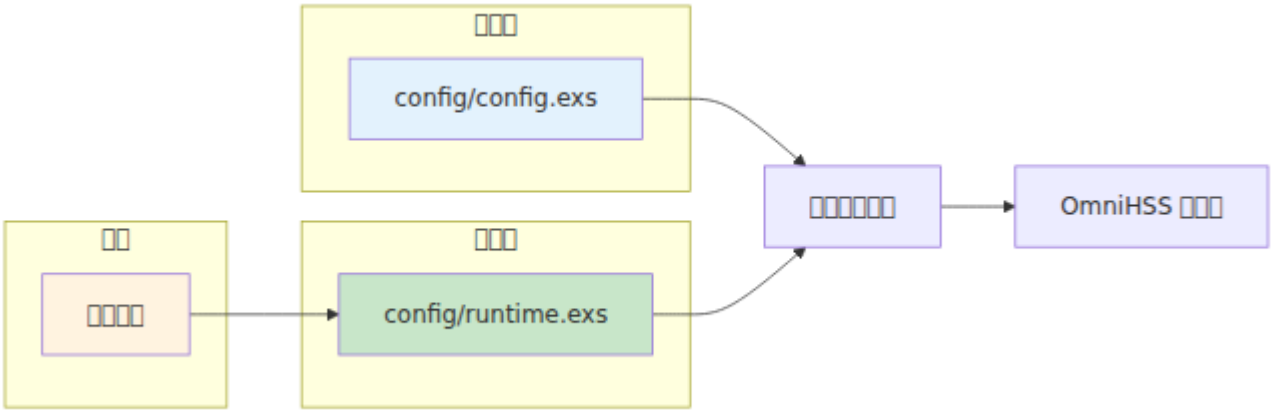
OmniHSS 配置指南
← 返回操作指南

目录

配置文件概述
许可证客户端配置
运行时配置
数据库配置
Diameter 配置
网络配置
家庭 PLMN 配置
HSS 核心配置
IMS 配置
EIR 配置
API 和控制面板配置
配置工作流程

配置文件概述

OmniHSS 使用两个主要配置文件：



config/config.exs（编译时）

□含在不同环境之间不会改变的静态配置：

- 控制面板页面配置
- API 端点配置
- 监控设置

config/runtime.exs（运行时）

□含每次部署时特定于环境的配置：

- 数据库连接参数
- Diameter 对等体配置
- 家庭 PLMN 设置
- IMS S-CSCF 选择
- 网络接口绑定

许可证客户端配置

许可证客户端通过远程许可证服务器验证 HSS 许可证：

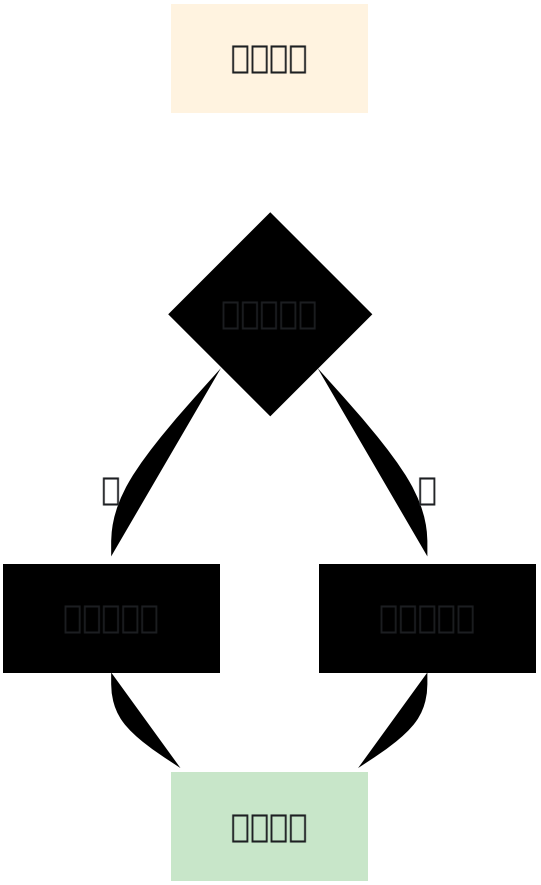
```
# config/runtime.exs

config :license_client,
  # 许可证服务器 API 端点□用于故障转移的列表）
  license_server_api_urls:
["https://license.example.com:8443/api"],

  # 许可证组织名称
  licensee: "您的组织名称",

  # 产品标识符
  product_name: "omnihss"
```

许可证配置参数：

参数	描述	必需	示例
`license_server_api_urls`	许可证服务器 URL 列表	是	`["https://10.0.0.1:8443/api"]`
`licensee`	许可证上的组织名称	是	`"ACME Telecom"`
`product_name`	许可证的产品标识符	是	`"omnihss"`

重要说明:

- 许可证服务器必须可以从 HSS 访问
- 使用 HTTPS 进行安全的许可证验证
- 多个 URL 提供故障转移能力
- 许可证验证在启动时和定期进行



运行时配置

配置优先级

环境变量模式

OmniHSS 遵循以下配置模式:

- 环境变量名称为大写并使用下划线
- 默认值在 runtime.exs 中提供
- 在生产环境中，数据库凭据应使用环境变量

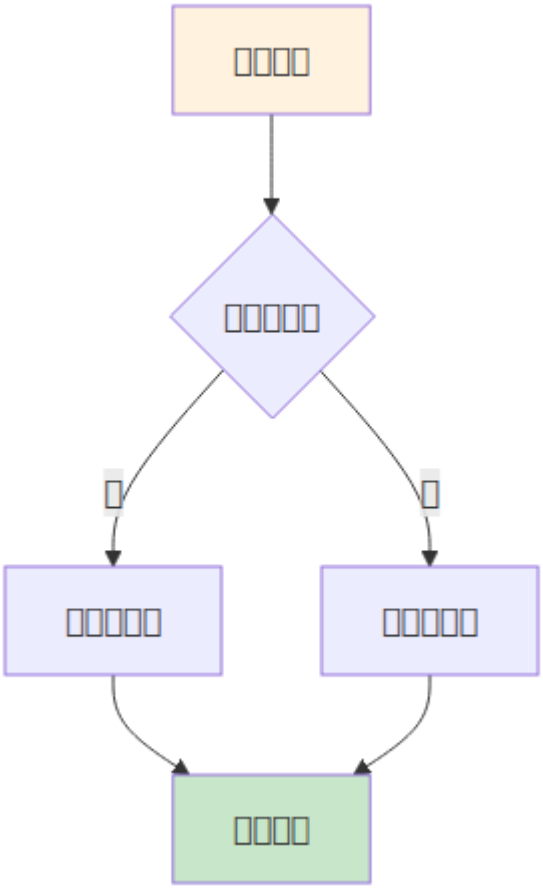
数据库配置

基本数据库配置

```
# config/runtime.exs

config :hss, Hss.Repo,
  # 数据库连接参数
  username: System.get_env("DATABASE_USERNAME", "root"),
  password: System.get_env("DATABASE_PASSWORD", "password"),
  hostname: System.get_env("DATABASE_HOSTNAME", "localhost"),
  database: System.get_env("DATABASE_NAME", "omnihss"),

  # 连接池设置
  pool_size:
String.to_integer(System.get_env("DATABASE_POOL_SIZE", "20")),

  # 超时□以毫秒为单位）
  timeout: 15_000,
  connect_timeout: 15_000,

  # 其他选项
  show_sensitive_data_on_connection_error: false
```

数据库配置参数

参数	描述	默认	推荐
`username`	SQL 数据库用户名	`"root"`	在生产中使用专用用户
`password`	SQL 数据库密码	`"password"`	使用强密码，存储在环境变量中
`hostname`	SQL 数据库服务器主机名	`"localhost"`	在生产中使用 FQDN 或 IP
`database`	数据库名称	`"omnihss"`	除非有多个实例，否则保持默认
`pool_size`	连接池大小	`20`	根据负载调整（10-50 为典型）



池大小调整

指导原则：

- 从 20 个连接开始
- 监控“连接池超时”错误
- 如果在正常负载下发生超时，则增加 10
- 每个连接使用约 4MB 的内存
- 连接过多可能会降低 SQL 数据库性能

示例：生产数据库配置

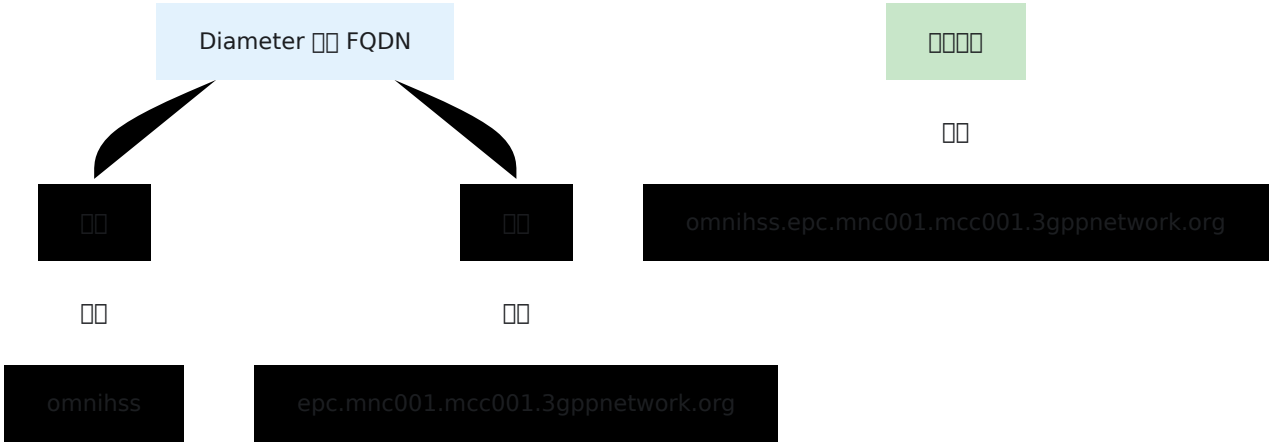
```
# config/runtime.exs - 生产示例

config :hss, Hss.Repo,
  username: System.fetch_env!("DATABASE_USERNAME"),       # 生产中必需
  password: System.fetch_env!("DATABASE_PASSWORD"),       # 生产中必需
  hostname: System.get_env("DATABASE_HOSTNAME",
"db.internal.example.com"),
  database: System.get_env("DATABASE_NAME", "omnihss"),
  port: String.to_integer(System.get_env("DATABASE_PORT",
"5432")),
  pool_size:
String.to_integer(System.get_env("DATABASE_POOL_SIZE", "30")),
  ssl: true,
  ssl_opts: [
    cacertfile: "/etc/ssl/certs/postgres-ca.pem",
    verify: :verify_peer
  ]
```

Diameter 配置

Diameter 服务配置

```
# config/runtime.exs

diameter_config = %{
  service_name: :omnitouch_hss,

  # 网络绑定
  listen_ip: System.get_env("DIAMETER_LISTEN_IP", "10.7.25.186"),
  listen_port:
String.to_integer(System.get_env("DIAMETER_LISTEN_PORT", "3868")),

  # Diameter 身份
  host: System.get_env("DIAMETER_HOST", "omnihss"),
  realm: System.get_env("DIAMETER_REALM",
"epc.mnc001.mcc001.3gppnetwork.org"),

  # 产品标识
  product_name: "OmniHSS",
  vendor_id: 10415,  # 3GPP
  supported_vendor_ids: [5535, 10415],

  # 协议设置
  request_timeout: 5000,

  # 对等体配置
  peers: [
    # 在此添加对等体配置
  ]
}

config :hss, :diameter, diameter_config
```

Diameter □□□□



□□□□□

- □□□ HSS □□□□□□□□□□“omnihss”□“hss01”□
- □□□ □□□ PLMN □□□ Diameter □□□□ □□“epc.mnc001.mcc001.3gppnetwork.org”□
- □□□□□ □□□ `{host}.{realm}`

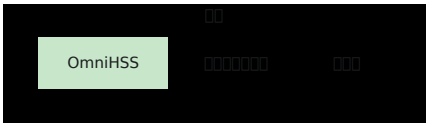
□□ Diameter □□□

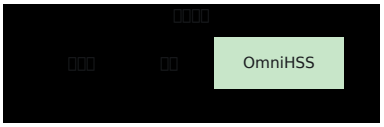
□□□□□□□□□□□□□□

```
# config/runtime.exs

peers: [
  # MME 对等体示例
  %{
    host: "mme01.epc.mnc001.mcc001.3gppnetwork.org",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.7.25.100",
    port: 3868,
    transport: :sctp,  # 或 :tcp
    applications: [:s6a]
  },

  # P-GW 对等体示例
  %{
    host: "pgw01.epc.mnc001.mcc001.3gppnetwork.org",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    ip: "10.7.25.101",
    port: 3868,
    transport: :sctp,
    applications: [:gx]
  },

  # I-CSCF 对等体示例
  %{
    host: "icscf01.ims.mnc001.mcc001.3gppnetwork.org",
    realm: "ims.mnc001.mcc001.3gppnetwork.org",
    ip: "10.7.25.102",
    port: 3868,
    transport: :tcp,
    applications: [:cx]
  }
]
```

仅监听模式

对于对等体主动连接 HSS 的环境：





```
# config/runtime.exs

diameter_config = %{
  # ... 其他配置 ...
  peers: []  # 空 - 仅接受传入连接
}
```

Diameter 对等体连接模式

传输协议选择

传输	优点	缺点	推荐
SCTP	多流，更好的故障检测	需要内核支持，防火墙配置	推荐用于 Diameter
TCP	通用支持，更简单的防火墙规则	单流，故障检测较慢	如果 SCTP 不可用则使用

网络配置

家庭 PLMN 配置

家庭 PLMN 标识您的网络运营商：

```
# config/runtime.exs

config :hss, :home_plmn, %{
  mcc: System.get_env("HOME_PLMN_MCC", "001"),  # 移动国家代码
  mnc: System.get_env("HOME_PLMN_MNC", "001")   # 移动网络代码
}
```

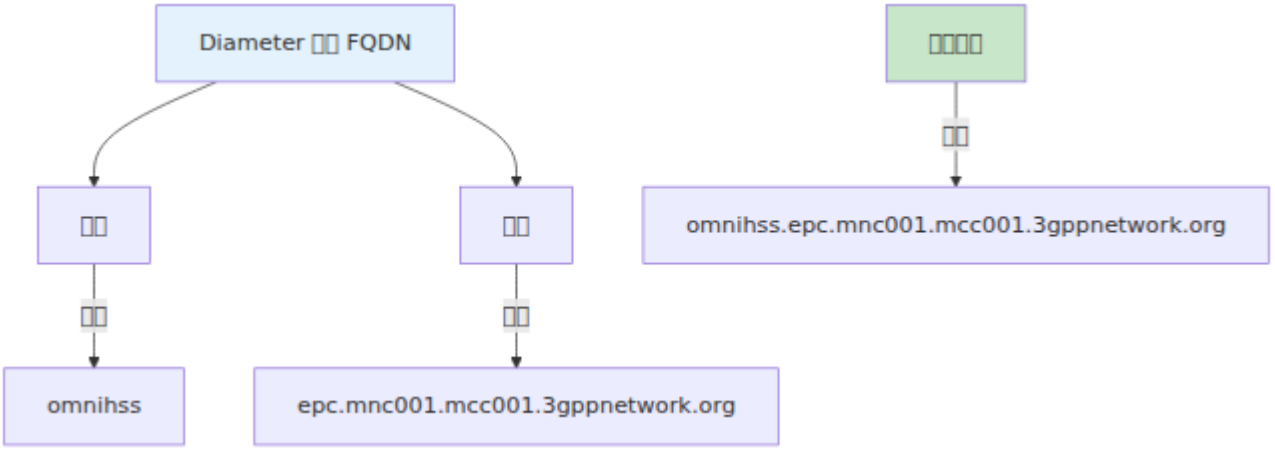
HSS 核心配置

这些设置控制 HSS 的行为和功能：

```
# config/runtime.exs

config :hss,
  # 用于数据库操作的 Ecto 存储库
  ecto_repos: [Hss.Repo],

  # MME 变更时的 CLR（取消位置请求）
  send_clr_on_mme_change: true,

  # 数据库故障期间停止 Diameter 服务
  stop_diameter_on_database_failure: true,

  # 许可证强制配置
  license_enforced: true,
  license_module: LicenseClient
```

HSS 核心参数：

参数	描述	默认	推荐
`ecto_repos`	应用程序使用的 Ecto 存储库列表	`[Hss.Repo]`	数据库操作所需
`send_clr_on_mme_change`	当订阅者更改 MME 时发送取消位置请求	`true`	保持启用以确保正确的移动性
`stop_diameter_on_database_failure`	如果数据库变得不可用，则禁用 Diameter 服务	`true`	启用以确保数据一致性
`license_enforced`	启用许可证强制	`true`	生产中所需
`license_module`	处理许可证检查的模块	`LicenseClient`	不要更改



PLMN 代码格式

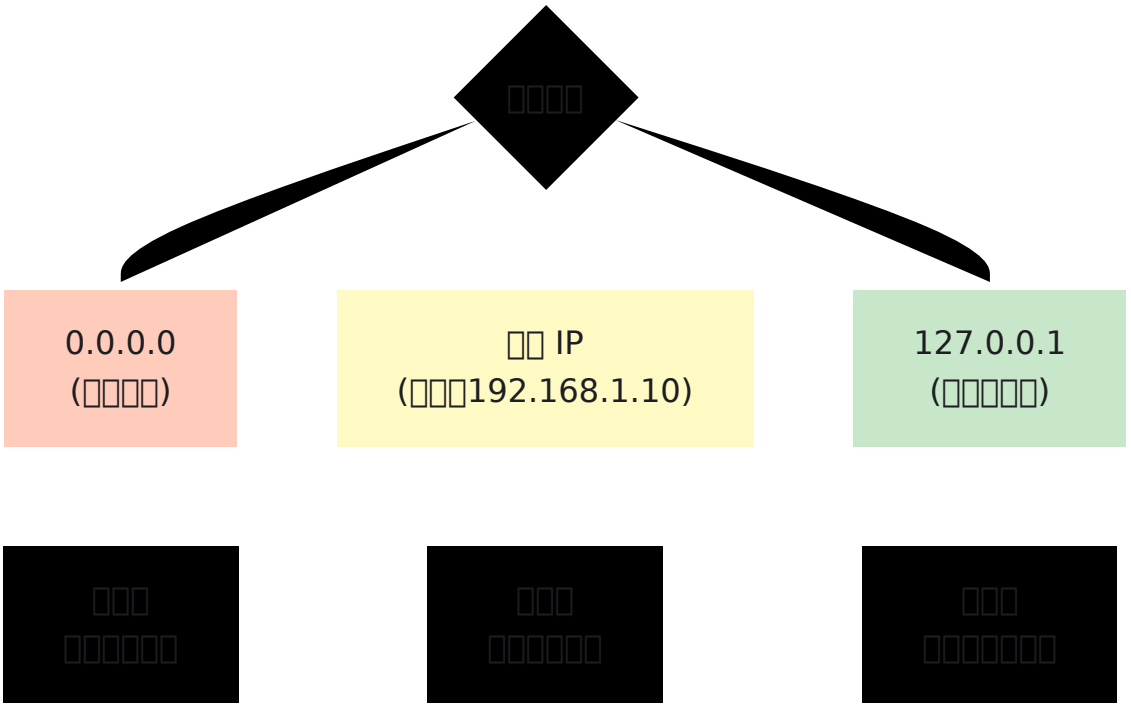
示例:

- AT&T (美国): MCC=310, MNC=410
- Verizon (美国): MCC=311, MNC=480
- Vodafone (英国): MCC=234, MNC=15
- 测试网络: MCC=001, MNC=01

网络接口绑定

```
# config/runtime.exs

# Diameter 接口
listen_ip: System.get_env("DIAMETER_LISTEN_IP", "0.0.0.0"),  # 所有接
口
# 或特定接口:
# listen_ip: "10.7.25.186",

# API 接口
config :hss, HssWeb.Api.Endpoint,
  http: [
    ip: {0, 0, 0, 0},  # 所有接口
    port: 8443
  ]

# 控制面板接口
config :hss, HssWeb.ControlPanel.Endpoint,
  http: [
    ip: {0, 0, 0, 0},  # 所有接口
    port: 7443
  ]
```

接口绑定选项:



IMS 配置

S-CSCF 选择配置

HSS 仅在第一次 REGISTER 时选择 S-CSCF，当没有 S-CSCF 记录在订阅者中时。存储名称后（见 S-CSCF 分配状态），HSS 返回存储的名称并不重新选择。这使得每个 REGISTER 在同一个 IMS-AKA 挑战周期内保持在同一个 S-CSCF 上，从而使周期收敛。

存储的名称始终是一个真实的 S-CSCF 节点。HSS 从 SAR 学习，因为 SAR 是第一个告诉 HSS 哪个节点进行了注册的消息。每次重新注册 UAR 和每个终止 LIR 都会返回该节点。一个共享的 IMS 订阅为其所有成员存储一个这样的节点。请参见 每个订阅一个服务 S-CSCF。

```
# config/runtime.exs

config :hss, :ims, %{
  scscf: %{
    # 第一次 REGISTER 选择方法: :pool_uri 或 :random_peer
    selection_method: :pool_uri,

    # :pool_uri 的逻辑 S-CSCF 池名称。没有端口，因此 I-CSCF 在池中进行 SRV 查找。HSS 仅在第一次 REGISTER 时返回此名称。它从不将其存储为服务 S-CSCF。
    pool_uri: "sip:scscf.ims.mnc001.mcc001.3gppnetwork.org",

    # IMS-AKA 挑战在 HSS 停止返回挑战 S-CSCF 之前可以保持多长时间
    auth_pending_ttl_seconds: 30,

    # :random_peer 的 S-CSCF 节点也是部署的 S-CSCF 清单）
    peers: [
      %{
        host: "sip:scscf01.ims.mnc001.mcc001.3gppnetwork.org:5060"
      },
      %{
        host: "sip:scscf02.ims.mnc001.mcc001.3gppnetwork.org:5060"
      }
    ]
  }
}
```

参数:

参数	类型	必需	默认	描述
`selection_method`	原子	是	`:pool_uri`	第一次 REGIST S-CSCF 选择方 用 `:pool_ur` `:random_pe` 请参见 S-CSCF 方法。
`pool_uri`	字符串	对于 `:pool_uri` 是必需的	-	第一次 REGIST 返回的逻辑 S-CS 池名称。设置无端口 I-CSCF 进行 S 查找并通过 DNS 分配负载。HSS 从 名称存储为订阅者的 S-CSCF。如果 `pool_uri`，H 会记录警告并回退到 `peers` 中的节点
`auth_pending_ttl_seconds`	整数	否	30	HSS 尊重认证待定 CSCF 的时间（以 位）。在此时间后，H 忽略待定名称并使用 定。请参见 S-CS 分配状态。
`peers`	列表	对于 `:random_peer` 是必需的	-	S-CSCF 节点的 请参见下面的对等体 表。

对等体参数：

参数	类型	必需	默认	描述
`host`	字符串	是	-	S-CSCF SIP URI，带 FQDN 和端口。I-CSCF 必须能够解析它。

S-CSCF 选择方法

选择方法：

方法	描述	用例
`:pool_uri`	返回单个 `pool_uri` 。I-CSCF 通过 DNS SRV 解析并在池中分配。DNS 在第一次 REGISTER 时分配负载并跳过死节点。	负载分配加上第一次 REGISTER 故障转移。推荐默认。
`:random_peer`	从 `peers` 中返回一个节点，按订阅者确定性选择（对排序的对等体列表进行 IMSI 哈希）。同一订阅者始终映射到同一节点。	固定的每个订阅者 S-CSCF，不依赖于 SRV 记录。

名称 `:random_peer` 是历史遗留的。选择是针对每个订阅者的确定性，而不是随机的。需要稳定的映射，以便[illegible]一个挑战周期内每个 REGISTER 返回同一 S-CSCF。

如果 `selection_method` [illegible]含未知值，HSS 会记录警告并回退到 `:random_peer` 行为。HSS 始终回答 UAR。

选择仅适用于第一次 **REGISTER**。下表显示每个状态下 UAA 返回的内容。

订阅者状态	**UAA** 服务器名称	**UAA** 结果代码
没有 S-CSCF 记录	新选择：`pool_uri` 或 `peers` 中的节点	2001（第一次注册）
IMS-AKA 挑战正在进行	挑战的 S-CSCF 节点	2002（后续注册）
已注册，或以保留名称未注册	服务 S-CSCF 节点	2002（后续注册）
共享 IMS 订阅的成员，另一个成员已经服务	订阅的锚定 S-CSCF 节点	2001 或 2002

IMS 领域配置

通常，IMS 使用与 EPC 不同的领域：

```
# EPC 领域
"epc.mnc001.mcc001.3gppnetwork.org"

# IMS 领域
"ims.mnc001.mcc001.3gppnetwork.org"
```

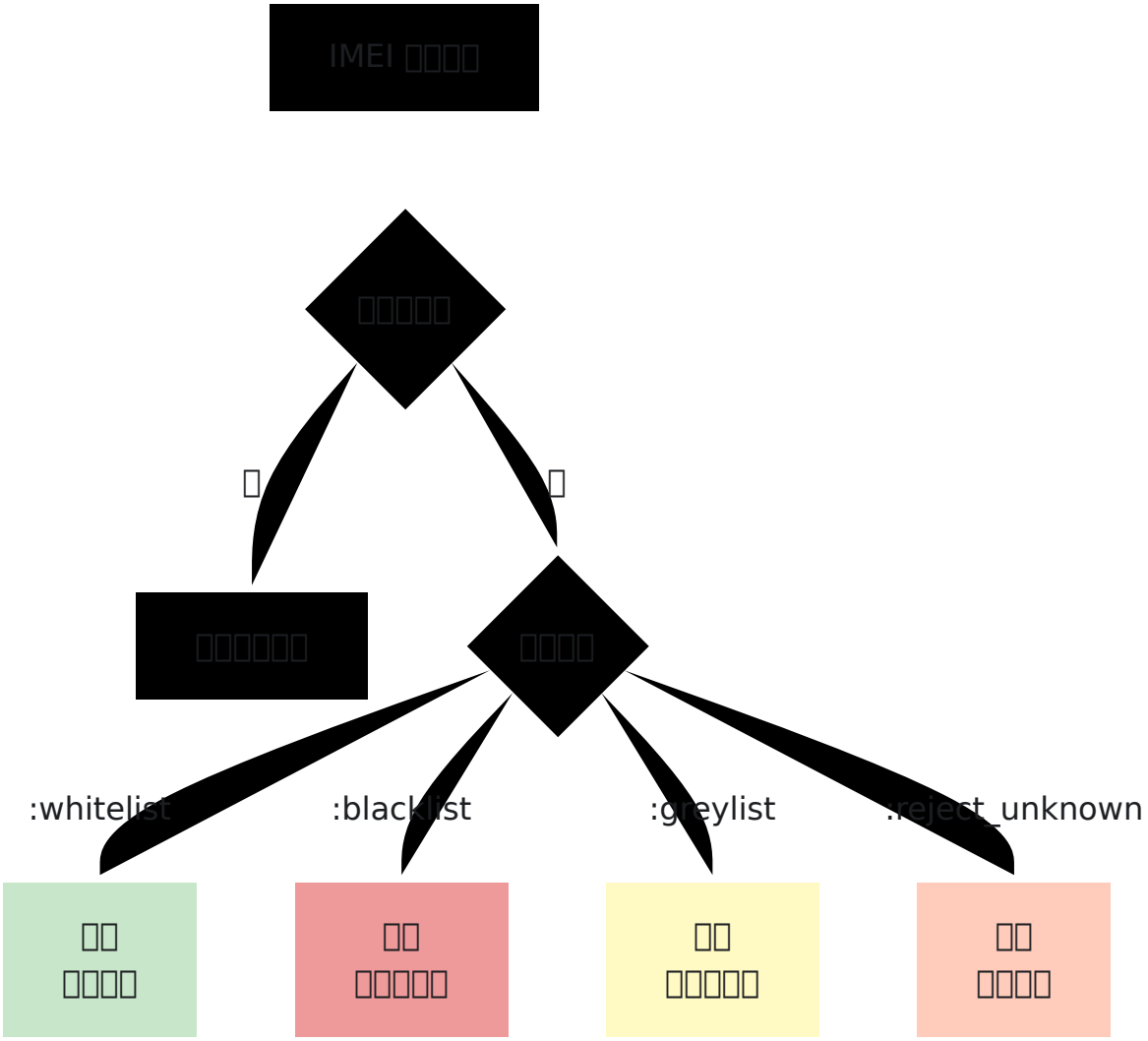
EIR 配置

请参见 **EIR** 文档 以获取完整的设备身份检查详细信息。

设备身份注册设置

```
# config/runtime.exs

config :hss, :eir, %{
  # 对于未知设备的行为（没有匹配规则）
  unknown_equipment_behaviour: :whitelist
  # 选项：
  #   :whitelist - 允许未知设备
  #   :blacklist - 阻止未知设备
  #   :greylist - 跟踪但允许未知设备
  #   :reject_unknown_equipment - 以特定结果代码拒绝
}
```

□□□□□□□



□□□□□□

□□	□□	□□
`:whitelist`	□□□□□□□ IMEI	□□□□□□□□
`:blacklist`	□□□□□□□ IMEI	□□□□□□
`:greylist`	□□□□□□□□ IMEI	□□□□□
`:reject_unknown_equipment`	□□□□□□□□	□□□□□

□□□ □□□□□□□ `:whitelist` □□□□□□ `:greylist` □□□□□□□□□□ `:blacklist` □□□□□□□□□

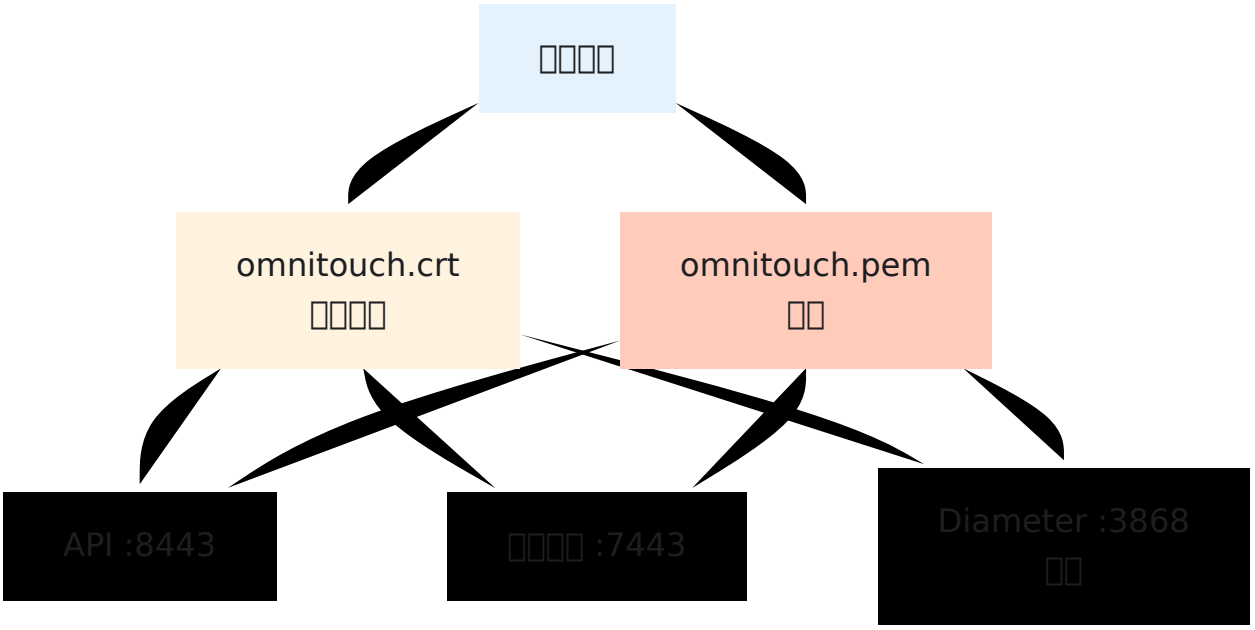
API 和控制面板配置

API 端点配置

```
# config/config.exs

config :hss, HssWeb.Api.Endpoint,
  url: [host: "localhost"],
  render_errors: [view: HssWeb.ErrorView, accepts: ~w(json)],
  pubsub_server: Hss.PubSub,

  # HTTPS 配置
  https: [
    port: 8443,
    cipher_suite: :strong,
    certfile: "priv/cert/omnitouch.crt",
    keyfile: "priv/cert/omnitouch.pem"
  ]
```

控制面板配置

```
# config/config.exs

config :hss, HssWeb.ControlPanel.Endpoint,
  url: [host: "localhost"],
  render_errors: [view: HssWeb.ErrorView, accepts: ~w(html json)],
  pubsub_server: Hss.PubSub,
  live_view: [signing_salt: "some-secret"],

  # HTTPS 配置
  https: [
    port: 7443,
    cipher_suite: :strong,
    certfile: "priv/cert/omnitouch.crt",
    keyfile: "priv/cert/omnitouch.pem"
  ]
```



TLS 证书配置

证书要求：

- 有效的 X.509 证书
- 匹配的私钥
- 如有需要，□含中间证书
- CN 或 SAN 必须与主机名匹配

生产环境：

```
https: [
  port: 8443,
  cipher_suite: :strong,
  certfile: System.get_env("TLS_CERT_FILE",
"/etc/ssl/certs/omnihss.crt"),
  keyfile: System.get_env("TLS_KEY_FILE",
"/etc/ssl/private/omnihss.key"),
  cacertfile: System.get_env("TLS_CA_FILE", "/etc/ssl/certs/ca-
bundle.crt")
]
```

配置工作流程

初始部署配置

配置检查清单

基本配置

- [ ] 数据库连接（���机名，凭据）
- [ ] 家庭 PLMN（MCC，MNC）
- [ ] 直径主机和领域
- [ ] 直径监听 IP 和端口
- [ ] API 和控制面板的 TLS 证书
- [ ] 许可证客户端配置（服务器 URL，许可证持有者，产品名称）
- [ ] HSS 核心设置（send_clr_on_mme_change，stop_diameter_on_database_failure）

网络元素集成

- [ ] 配置直径对等体（如果使用连接模式）
- [ ] 防火墙规则允许直径流量（端口 3868）
- [ ] 防火墙规则允许 HTTPS 流量（端口 7443，8443）

- [ ] 直径身份的 DNS 解析

IMS 配置（如果使用 IMS 功能）

- [ ] 配置 S-CSCF 对等体列表
- [ ] 选择 S-CSCF 选择方法
- [ ] 配置 IMS 领域

可选配置

- [ ] 配置 EIR 行为
- [ ] 调整数据库池大小
- [ ] 限制网络接口绑定

验证配置

修改配置后:

1. 语法检查:

   检查日志以获取配置加载错误

2. 控制面板访问:

   访问 `https://[hostname]:7443`
   验证概览页面是否加载

3. **API** 访问:

   ```
   curl -k https://[hostname]:8443/api/status
   ```

4. 直径状态:

   检查控制面板直径页面
   验证对等连接

5. 数据库连接性:

检查控制面板的订阅者数据
或直接连接到 SQL 数据库

完整的运行时配置示例

```
# config/runtime.exs - 完整的生产示例

import Config

#
==========================================================================
# 数据库配置
#
==========================================================================
config :hss, Hss.Repo,
  username: System.fetch_env!("DATABASE_USERNAME"),
  password: System.fetch_env!("DATABASE_PASSWORD"),
  hostname: System.get_env("DATABASE_HOSTNAME", "db.omnihss.internal'
  database: System.get_env("DATABASE_NAME", "omnihss"),
  port: String.to_integer(System.get_env("DATABASE_PORT", "5432")),
  pool_size: String.to_integer(System.get_env("DATABASE_POOL_SIZE", '
  timeout: 15_000,
  connect_timeout: 15_000,
  ssl: true,
  ssl_opts: [
    cacertfile: "/etc/ssl/certs/postgres-ca.pem",
    verify: :verify_peer
  ]

#
==========================================================================
# 许可证客户端配置
#
==========================================================================
config :license_client,
  license_server_api_urls: [System.get_env("LICENSE_SERVER_URL",
"https://license.example.com:8443/api")],
  licensee: System.get_env("LICENSE_ORGANIZATION", "您的组织"),
  product_name: "omnihss"

#
==========================================================================
# 家庭 PLMN 和 HSS 核心配置
#
==========================================================================
```

```
config :hss,
  ecto_repos: [Hss.Repo],
  home_plmn: %{
    mcc: System.get_env("HOME_PLMN_MCC", "001"),
    mnc: System.get_env("HOME_PLMN_MNC", "001")
  },
  send_clr_on_mme_change: true,
  stop_diameter_on_database_failure: true,
  license_enforced: true,
  license_module: LicenseClient

#
============================================================================
# □□□□
#
============================================================================
diameter_config = %{
  service_name: :omnitouch_hss,
  listen_ip: System.get_env("DIAMETER_LISTEN_IP", "10.7.25.186"),
  listen_port: String.to_integer(System.get_env("DIAMETER_LISTEN_PORT
"3868")),
  host: System.get_env("DIAMETER_HOST", "omnihss01"),
  realm: System.get_env("DIAMETER_REALM",
"epc.mnc001.mcc001.3gppnetwork.org"),
  product_name: "OmniHSS",
  vendor_id: 10415,
  supported_vendor_ids: [5535, 10415],
  request_timeout: 5000,
  peers: [
    %{
      host: "mme01.epc.mnc001.mcc001.3gppnetwork.org",
      realm: "epc.mnc001.mcc001.3gppnetwork.org",
      ip: "10.7.25.100",
      port: 3868,
      transport: :sctp,
      applications: [:s6a]
    }
  ]
}

config :hss, :diameter, diameter_config

#
============================================================================
```

```
# IMS □□
#
============================================================================
config :hss, :ims, %{
  scscf: %{
    selection_method: :random_peer,
    peers: [
      %{host: "sip:scscf01.ims.mnc001.mcc001.3gppnetwork.org:5060"},
      %{host: "sip:scscf02.ims.mnc001.mcc001.3gppnetwork.org:5060"}
    ]
  }
}

#
============================================================================
# EIR □□
#
============================================================================
config :hss, :eir, %{
  unknown_equipment_behaviour: :whitelist
}

#
============================================================================
# API □□□□
#
============================================================================
config :hss, HssWeb.Api.Endpoint,
  http: [ip: {0, 0, 0, 0}, port: 8443],
  https: [
    port: 8443,
    cipher_suite: :strong,
    certfile: System.get_env("TLS_CERT_FILE", "/etc/ssl/certs/omnihss
    keyfile: System.get_env("TLS_KEY_FILE", "/etc/ssl/private/omnihss
  ],
  url: [host: System.get_env("API_HOST", "api.omnihss.internal"), por
8443]

#
============================================================================
# □���□□□□□□
#
============================================================================
config :hss, HssWeb.ControlPanel.Endpoint,
```

```
  http: [ip: {0, 0, 0, 0}, port: 7443],
  https: [
    port: 7443,
    cipher_suite: :strong,
    certfile: System.get_env("TLS_CERT_FILE", "/etc/ssl/certs/omnihss
    keyfile: System.get_env("TLS_KEY_FILE", "/etc/ssl/private/omnihss
  ],
  url: [host: System.get_env("CP_HOST", "hss.omnihss.internal"), port
```

← 返回操作指南 | 下一步：实体关系 →

OmniHSS 控制面板指南
← 返回操作指南

目录

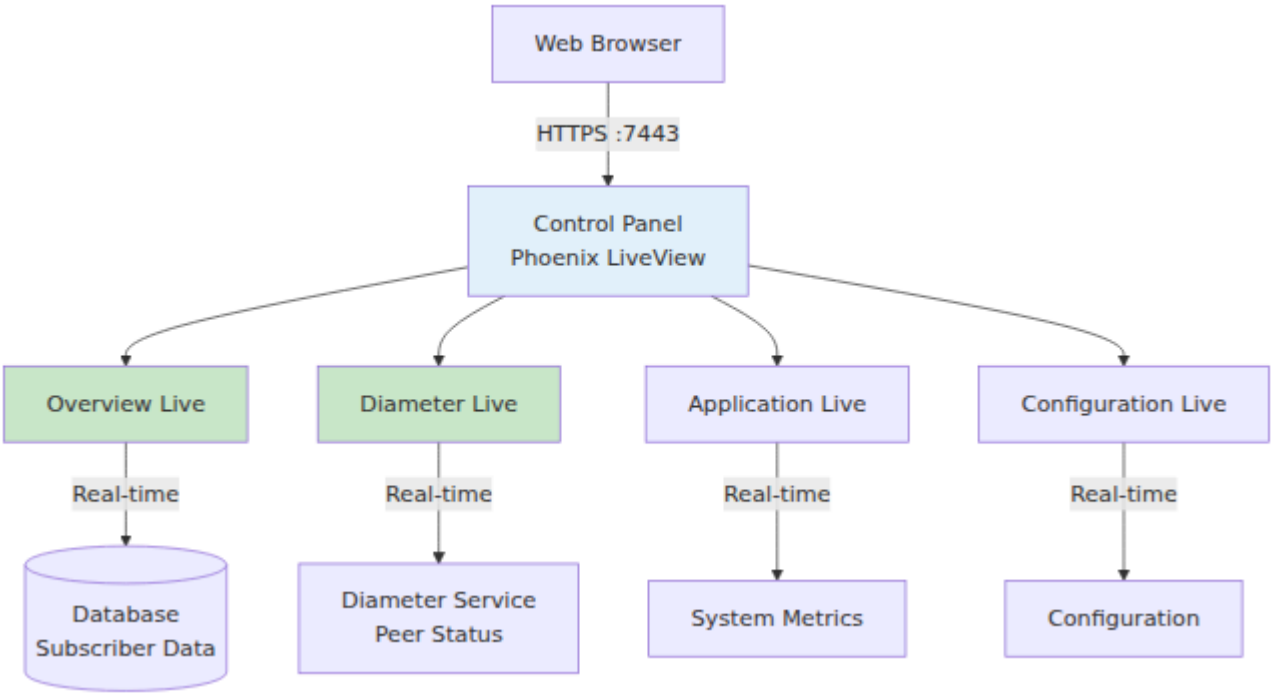
控制面板概述
访问控制面板
概览页面
直径页面
应用页面
配置页面
导航和界面

控制面板概述

OmniHSS 控制面板是一个基于网络的监控界面，提供系统状态、用户活动和直径连接的实时可视化。它使用 Phoenix LiveView 构建，能够自动更新而无需刷新页面。

主要特点

实时更新 - 每秒自动刷新
用户监控 - 查看活跃用户及其当前状态
直径状态 - 实时监控对等连接
系统资源 - 跟踪应用性能
配置查看器 - 检查运行时配置



访问信息

```
URL: https://[hostname]:7443
Protocol: HTTPS Only
Port: 7443 (configurable)
Certificate: Configured in config/config.exs
```

控制面板架构

访问控制面板

初始访问

1. 打开一个网络浏览器
2. 导航到 `https://[hostname]:7443`
3. 接受 TLS 证书（如果是自签名的）
4. 默认情况下，您将看到概览页面

TLS 证书警告

如果使用自签名证书，浏览器将显示安全警告。这在内部部署中是预期的。

对于生产环境：使用受信任的证书颁发机构签发的证书。

网络要求

端口 7443 必须可以从您的管理网络访问
HTTPS 是强制性的 - 不支持 HTTP
防火墙规则 必须允许流量通过端口 7443

浏览器兼容性

控制面板使用现代网络技术（LiveView, WebSockets）：

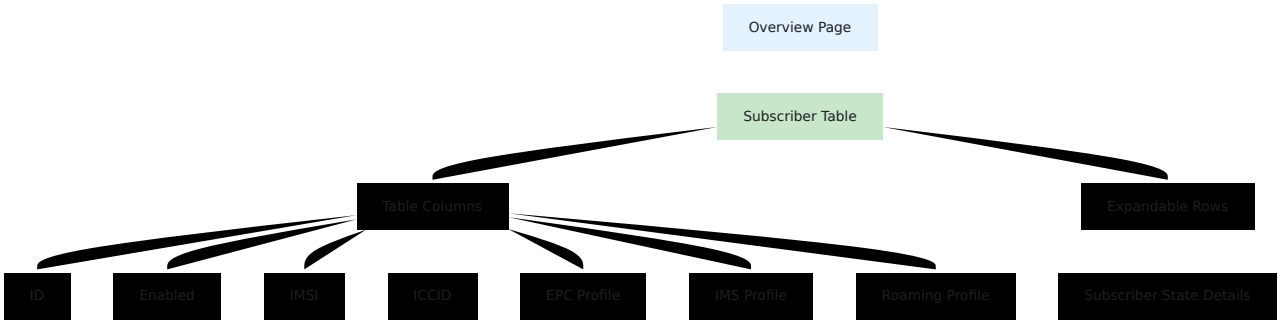
Chrome/Chromium（推荐）
Firefox
Safari
Edge

注意：不支持 Internet Explorer。

概览页面

URL: https://[hostname]:7443/overview

概览页面显示所有用户及其实时状态信息。



页面布局

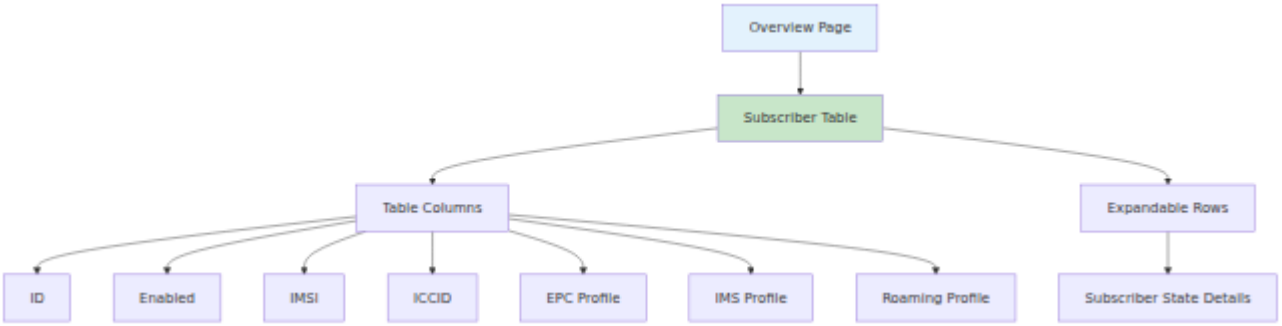
表格列

列	描述	值
ID	用户数据库 ID	字符串 (UUID)
Enabled	服务状态	✓ (启用) / ✗ (禁用)
IMSI	国际移动用户身份	14-15 位数字
ICCID	SIM 卡 ID	19-20 位数字或 "N/A"
EPC Profile	数据服务配置文件名称	配置文件名称或 ID
IMS Profile	语音服务配置文件名称	配置文件名称、ID 或 "N/A"
Roaming Profile	漫游策略名称	配置文件名称、ID 或 "N/A"

可扩展行详细信息

单击任何行以展开并查看详细的用户状态：

位置信息



字段：

- **MCC** - 移动国家代码（3 位数字）
- **MNC** - 移动网络代码（2-3 位数字）
- **TAC** - 跟踪区域代码
- **Cell ID** - 服务小区标识符
- **eNodeB ID** - 基站标识符
- **ECI** - E-UTRAN 小区标识符

网络信息

字段：

- 最后看到的 **MME** - 当前服务的 MME 主机名
- 最后看到的领域 - MME 的直径领域
- **RAT** 类型 - 无线接入技术（例如，“E-UTRAN”用于 LTE）
- 最后看到的时间 - 最后直径消息的时间戳

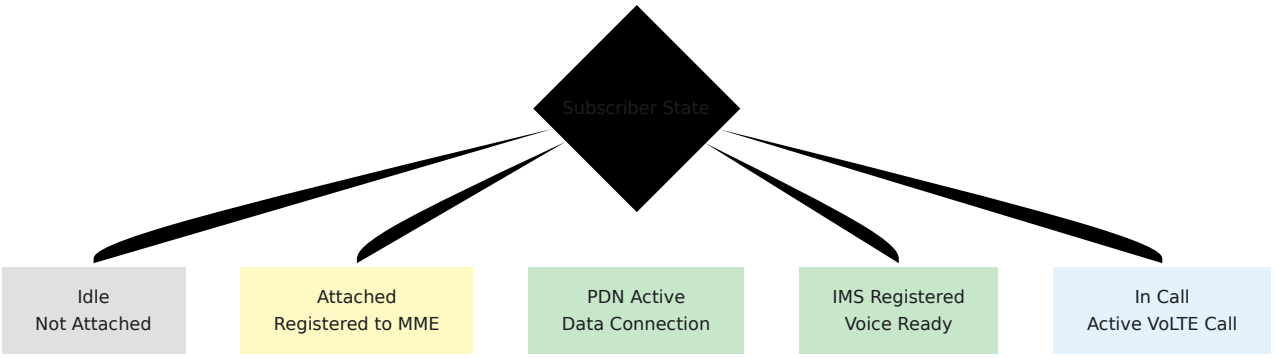
IMS 信息

字段：

- 分配的 **S-CSCF** - 当前分配的 S-CSCF SIP URI
- **IMS** 公共身份 - SIP URI（例如，sip:+14155551234@ims.example.com）
- 最后看到的 **P-CSCF** - 最后一个联系 HSS 的 P-CSCF
- 最后看到的 **I-CSCF** - 最后一个联系 HSS 的 I-CSCF

会话信息

字段：



- **PDN** 会话 - 活跃数据连接的数量
- 活跃通话 - 活跃 VoLTE 通话的数量

状态指示器

如何识别状态：

- **Idle:** 没有位置信息，没有 MME
- **Attached:** 最后看到的 MME 存在，位置信息可用
- **PDN Active:** PDN 会话计数 > 0
- **IMS Registered:** 分配的 S-CSCF 存在
- **In Call:** 活跃通话计数 > 0

自动刷新

概览页面每 **1** 秒自动刷新，以显示实时更新。

视觉指示：

- 新数据在不重新加载页面的情况下出现
- 时间戳实时更新
- 无需手动刷新

用例

1. 监控活跃用户
    - 查看当前连接的用户
    - 检查当前服务网络（对于漫游）

- 验证 IMS 注册状态

2. 故障排除

- 验证用户是否启用
- 检查最后看到的时间戳（用户是否响应？）
- 确认配置文件分配
- 查看当前位置信息

3. 容量监控

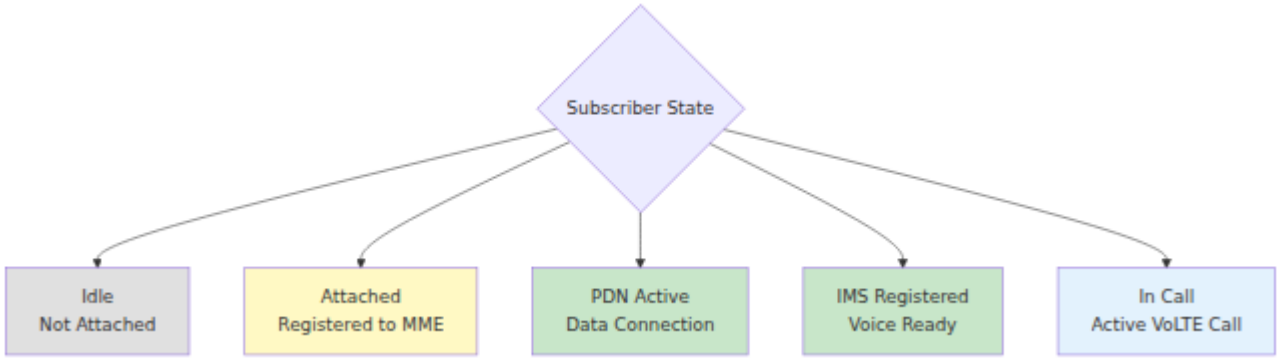
- 计算总连接用户数
- 监控 PDN 会话计数
- 跟踪活跃 VoLTE 通话

直径页面

URL: `https://[hostname]:7443/diameter`

直径页面显示所有直径对等连接的实时状��。

页面布局



表格列

列	描述	值
Hostname	直径对等主机名	FQDN
Realm	直径领域	域名
IP:Port	网络地址	IP 地址和端口
Transport	传输协议	TCP 或 SCTP
Status	连接状态	Connected / Disconnected

连接状态

可扩展行详细信息

单击任何对等以查看附加信息：

连接信息：

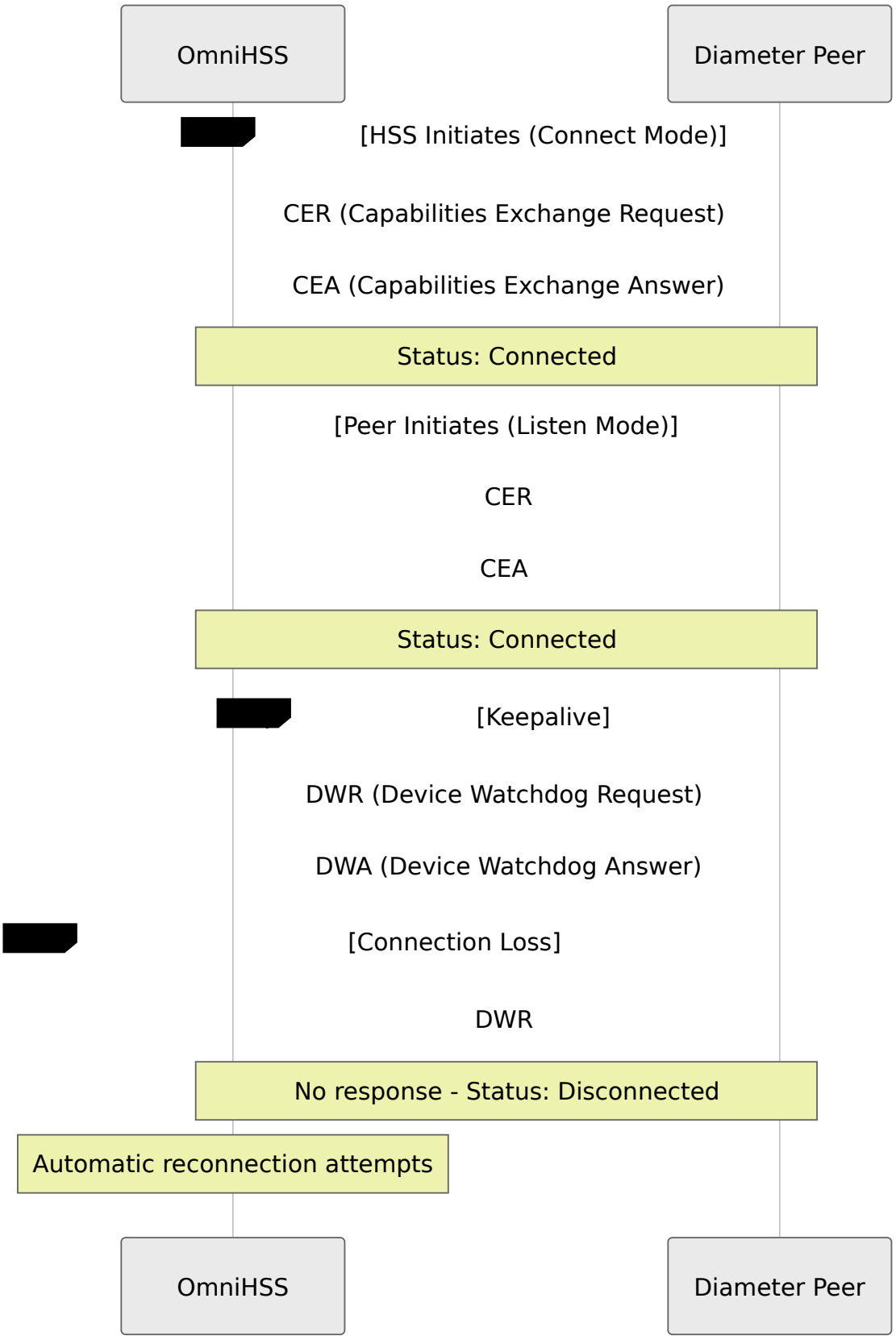
- 连接类型 - 由 HSS 或对等方发起
- 产品名称 - 对等方的产品标识
- 应用程序 **ID** - 支持的直径应用程序

应用程序 **ID** 示例：

- 16777251 - S6a (MME)

- 16777238 - Gx (P-GW)
- 16777216 - Cx (I-CSCF, S-CSCF)
- 16777217 - Sh (应用服务器)
- 16777236 - Rx (P-CSCF)
- 16777252 - S13 (EIR 客户端，如果是外部)

[redacted]


OmniHSS
Diameter Peer
[redacted]
[HSS Initiates (Connect Mode)]
CER (Capabilities Exchange Request)
CEA (Capabilities Exchange Answer)
Status: Connected
[Peer Initiates (Listen Mode)]
CER
CEA
Status: Connected
[redacted]
[Keepalive]
DWR (Device Watchdog Request)
DWA (Device Watchdog Answer)
[redacted]
[Connection Loss]
DWR
No response - Status: Disconnected
Automatic reconnection attempts
OmniHSS
Diameter Peer

自动刷新

直径页面每 1 秒 自动刷新。

用例

1. 验证连接

确保所有预期的对等方已连接
立即识别断开连接的对等方
监控连接波动

2. 故障排除

检查对等方是否可达
验证传输协议（TCP 与 SCTP）
确认应用程序 ID 是否符合预期
确定是哪个方发起了连接

3. 容量规划

计算总连接的对等方
监控连接稳定性
规划额外的对等容量

常见问题

对等方显示断开连接

可能原因：

1. 网络连接问题
2. 对等方关闭或重启
3. 防火墙阻止流量
4. 直径配置不匹配
5. 证书问题（如果使用 TLS）

故障排除步骤：

1. 检查网络连接：ping [peer-ip]
2. 验证端口是否可达：telnet [peer-ip] 3868
3. 检查防火墙规则
4. 查看 HSS 日志中的错误信息
5. 验证对等方的直径配置与 HSS 匹配

对等方反复连接和断开

可能原因：

1. 网络不稳定
2. 保持活动超时不匹配
3. 对等方资源问题
4. 直径应用程序不匹配

故障排除步骤：

1. 检查网络稳定性
2. 查看双方的保持活动计时器
3. 检查对等方系统资源
4. 验证双方的应用程序 ID 是否匹配

应用页面

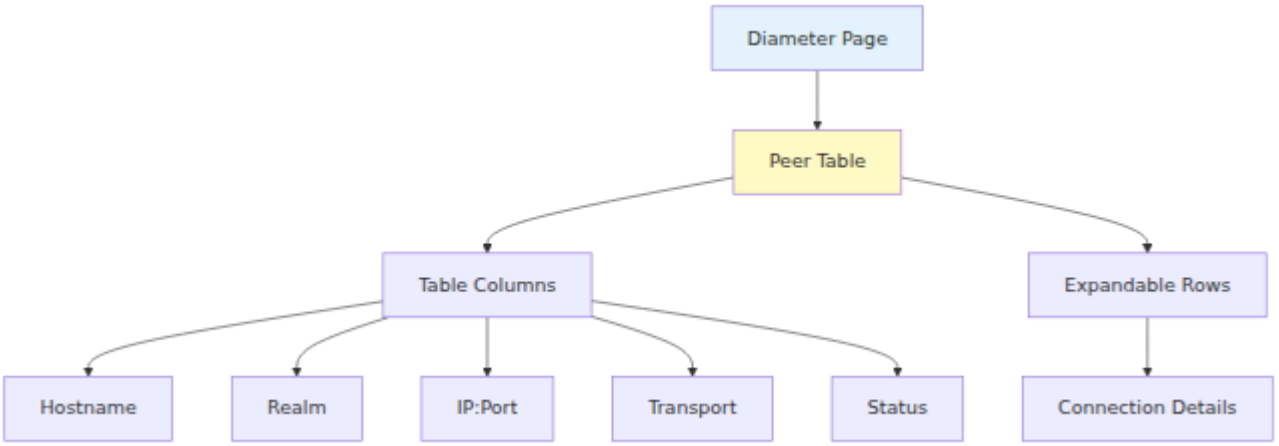
URL: https://[hostname]:7443/application

应用页面提供系统级监控和资源使用信息。

特点

进程信息 - Erlang VM 进程计数和内存
系统内存 - 总内存和已用内存
应用运行时间 - OmniHSS 运行的时间
Erlang VM 版本 - 运行时版本信息

关键指标



用例

1. 健康监控

验证应用是否在运行
检查内存泄漏（内存随时间增加）
监控进程计数增长

2. 容量规划

跟踪内存使用趋势
根据进程计数规划扩展
验证系统资源是否充足

3. 故障排除

识别资源耗尽
检查是否需要重启
验证 Erlang VM 版本

配置页面

URL: https://[hostname]:7443/configuration

配置页面显示 OmniHSS 当前的运行时配置。

特点

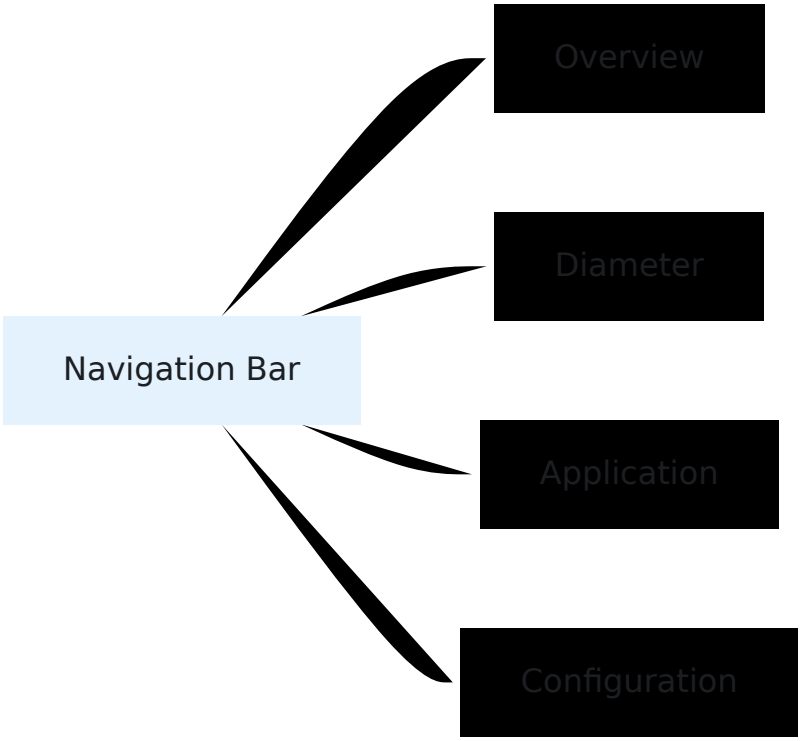
- 查看配置 - 检查所有配置参数
- 搜索配置 - 查找特定设置
- 环境变量 - 查看解析后的值

配置类别

用例

1. 配置验证
    - 验证 runtime.exs 设置是否已应用
    - 确认数据库连接参数
    - 检查直径对等配置
2. 故障排除
    - 识别配置错误
    - 验证环境变量是否正确设置
    - 比较预期与实际配置
3. 文档
    - 导出当前配置以供文档使用
    - 与支持团队共享配置

安全提示：配置页面可能显示敏感信息（数据库密码、密钥）。请适当限制访问。



导航和界面

顶部导航栏

导航始终可见，位于页面顶部以便快速访问。

键盘快捷键

控制面板不实现自定义键盘快捷键。标准浏览器快捷键有效：

- **Ctrl+R / F5** - 手动刷新页面（尽管自动刷新使其不必要）
- **Ctrl+F** - 在页面上搜索
- **Ctrl+T** - 打开新标签页（用于多个页面）

多标签监控

您可以在不同的浏览器标签中打开多个控制面板页面以进行同时监控：

示��设置：

- 标签 1：概览页面（监控用户）
- 标签 2：直径页面（监控连接）

标签 3：应用页面（监控资源）

所有标签将独立自动更新。

响应式设计

控制面板针对桌面浏览器进行了优化。移动浏览器也受支持，但可能需要水平滚动以查看表格。

推荐分辨率： 1920x1080 或更高，以便舒适查看。

监控最佳实践

每日操作

1. 班次开始

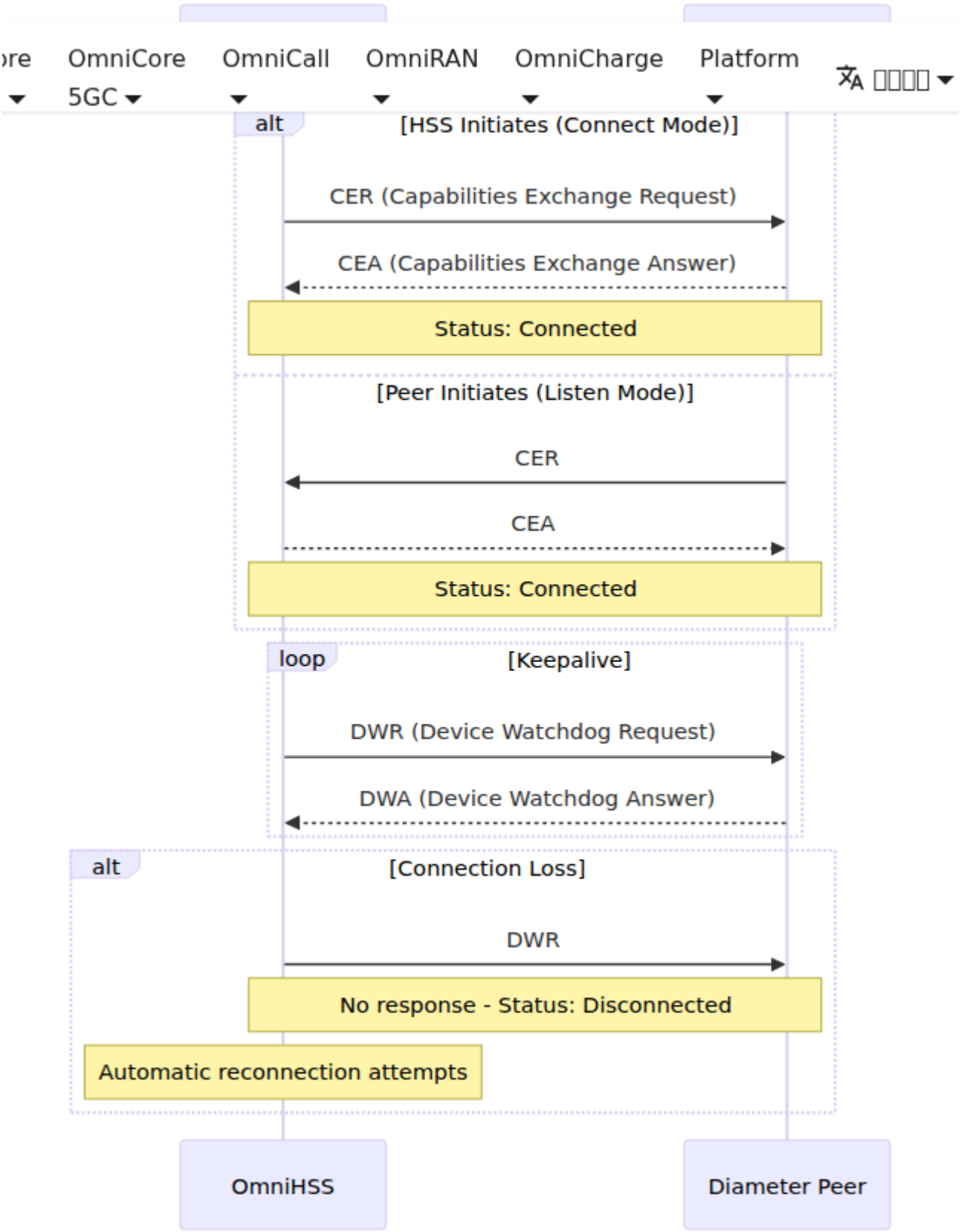
打开控制面板概览页面
验证预期的用户数量是否已连接
检查直径页面 - 所有对等方已连接

2. 班次期间

保持概览页面打开以进行实时监控
注意异常状态变化
监控直径页面上的断开连接的对等方

3. 班次结束

验证系统是否稳定
检查应用页面的资源使用趋势
记录任何异常



故障排除工作流程

ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
alt
[HSS Initiates (Connect Mode)]
CER (Capabilities Exchange Request)
CEA (Capabilities Exchange Answer)
Status: Connected
[Peer Initiates (Listen Mode)]
CER
CEA
Status: Connected
loop
[Keepalive]
DWR (Device Watchdog Request)
DWA (Device Watchdog Answer)
alt
[Connection Loss]
DWR
No response - Status: Disconnected
Automatic reconnection attempts
OmniHSS
Diameter Peer

警报阈值

建立监控阈值以进行主动警报：

指标	警告	严重
断开连接的直径对等方	1 对等方	2+ 对等方或关键对等方
内存使用	> 80%	> 90%
用户认证失败	> 5%	> 10%
进程计数	> 80% 的限制	> 95% 的限制

← 返回操作指南 | 下一步：指标与监控 →

数据库复制

← 返回操作指南

目录

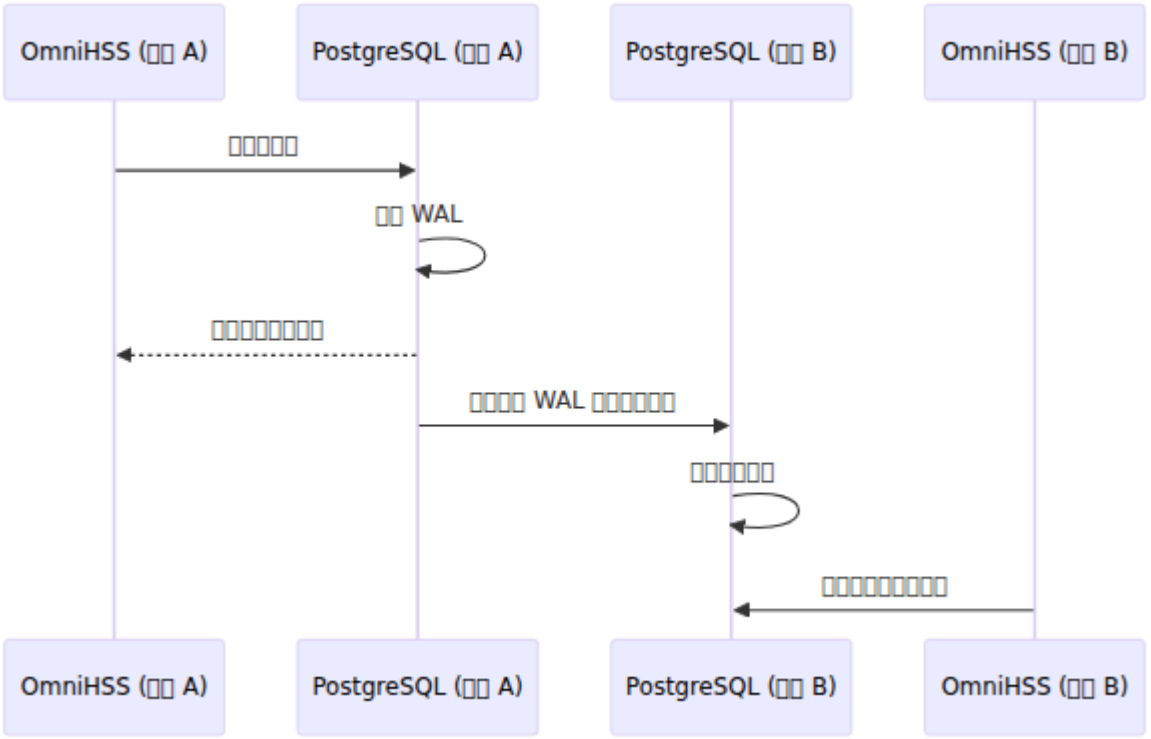
- 概述
- 工作原理
- 部署架构
- 配置参考
- 对等管理
- 监控
- 故障排除

概述

OmniHSS 使用 **PostgreSQL** 和 双向逻辑复制 进行多站点高可用性部署。每个站点作为独立的读/写主节点运行。没有单一主节点或只读副本。

主要特征

- 两个站点均可读/写：每个站点都有一个完整的主数据库。无论站点之间的链接质量如何，本地写入都是即时的
- 异步复制：更改通过 PostgreSQL 的预写日志（WAL）进行复制。没有写入会因等待远程站点确认而被阻塞
- 分区容忍：如果站点之间的链接断开，两个站点将继续独立运行。WAL 段在本地排队，并在链接恢复时自动重放
- 不需要法定人数：与同步集群方法不同，没有法定人数的概���。单站点部署与多站点部署的操作方式相同
- **UUID** 主键：所有记录使用 UUIDv4 主键，消除了站点之间没有协调的 ID 冲突

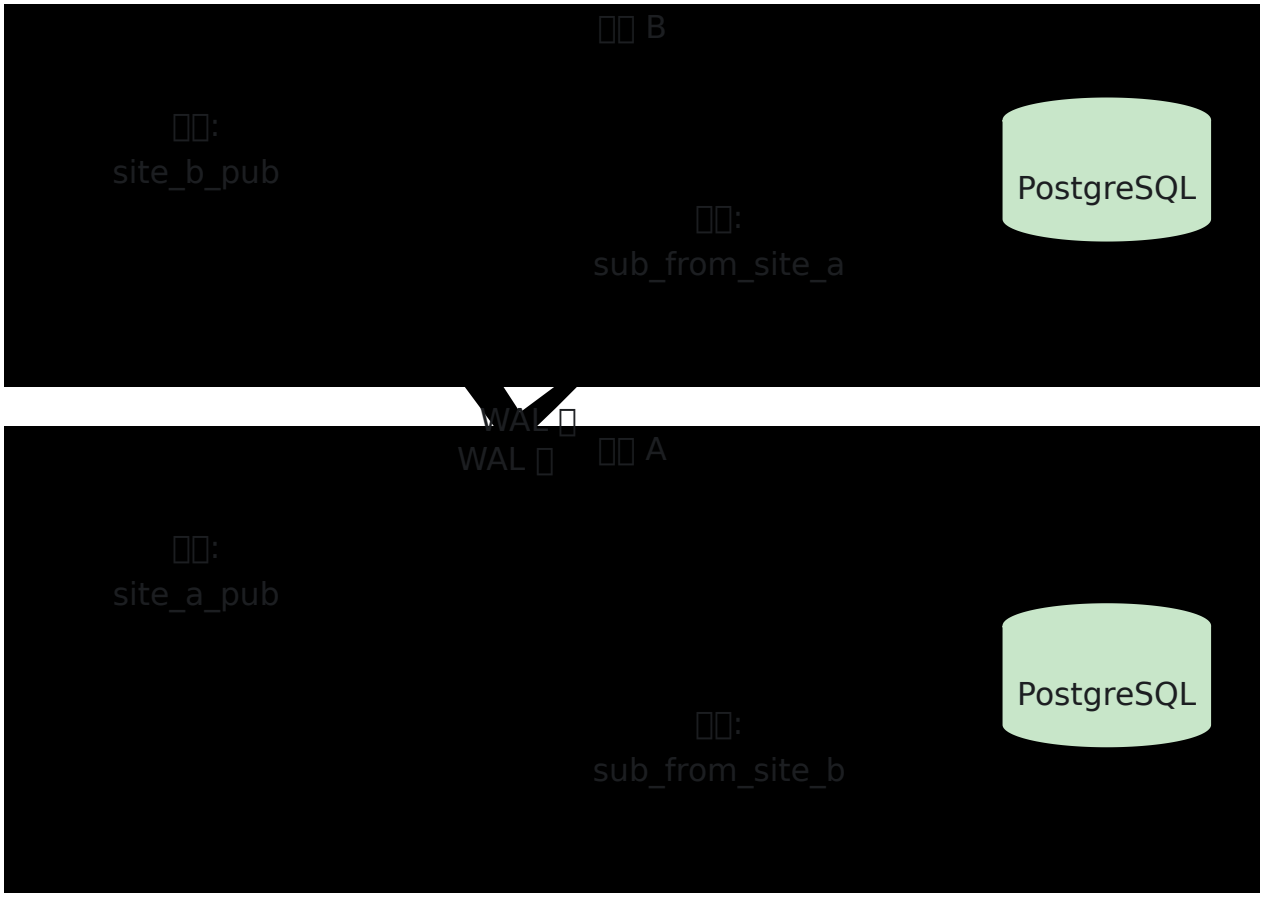


与同步复制的区别

	同步	异步逻辑复制
写入延迟	被最慢节点阻塞	仅本地：亚毫秒
链接故障	少数站点变为只读	两个站点继续读/写
需要法定人数	是（大多数节点）	否
数据一致性	强（所有节点相同）	最终（短暂的复制延迟）
适合卫星链接	否（延迟影响写入性能）	是（WAL 排队并赶上）

工作原理

逻辑复制流程



发布/订阅模型

每个站点都有：

- 一个 发布，广播所有表的更改（除了 `schema_migrations`）
- 一个 订阅，订阅每个对等站点的发布

链接中断期间的 **WAL** 排队

当站点之间的链接不可用时：

1. 本地写入继续进行而不受干扰
2. WAL 段在发布方累积
3. 复制槽跟踪每个订阅者的最后位置
4. 当链接恢复时，排队的 WAL 从最后确认的位置重放
5. 两个站点趋向于相同的状态

WAL 保留受 `max_slot_wal_keep_size` 限制，以防止在长时间中断期间磁盘无限增长。

冲突预防

UUID 主键消除了双向复制中的主要冲突源。由于每个站点独立生成全局唯一标识符，INSERT 操作从不冲突。

origin = none 订阅选项防止复制循环。通过复制到达的更改不会重新发布给其他订阅者。

schema_migrations 表被排除在发布之外，因为两个站点独立运行相同的迁移。

动态状态的最后写入胜出

动态状态表（subscriber_state、pdn_session、lte_call）通过 Diameter 信令频繁更新。每个 ULR 更新服务 MME，每个 CCR 更新 PGW 会话，每个 AAR 更新活动呼叫。

在多站点部署中，每次只由一个站点为订阅者提供服务。当订阅者在站点之间移动时，两个站点可能对同一 subscriber_state 行有来自不同时间的更新。如果没有冲突解决，复制顺序将决定哪个值存活。这可能会用过时的值覆盖当前服务的 MME。

OmniHSS 在动态状态表上使用 最后写入胜出 (LWW) 触发器：

CREATE TRIGGER lww_subscriber_state_trigger
BEFORE UPDATE ON subscriber_state
FOR EACH ROW

EXECUTE FUNCTION lww_subscriber_state();

触发器比较传入复制更新的 updated_at 时间戳与现有行。如果现有行更新更晚，则更新将被静默抑制。这确保：

当订阅者从站点 A 移动到站点 B 时，站点 B 的更新总是胜出
来自站点 A 的过时更新（在订阅者移动之前）被丢弃
两个站点趋向于相同的值：最近的写入

具有 LWW 触发器的表：

表	更新者	频率
`subscriber_state`	ULR, SAR, AIR, PUR	每次附加/位置更新
`pdn_session`	CCR-I, CCR-T	每次 PDN 会话创建/删除
`lte_call`	AAR, STR	每次 VoLTE 呼叫设置/拆除
`ims_subscription`	UAR, SAR（共享 S-CSCF）	每次分组成员注册

静态配置数据（订阅者、配置文件、密钥、APN）不需要 LWW。配置 API 仅更新这些记录，通常来自单一管理平面。

命名约定□新表所需）

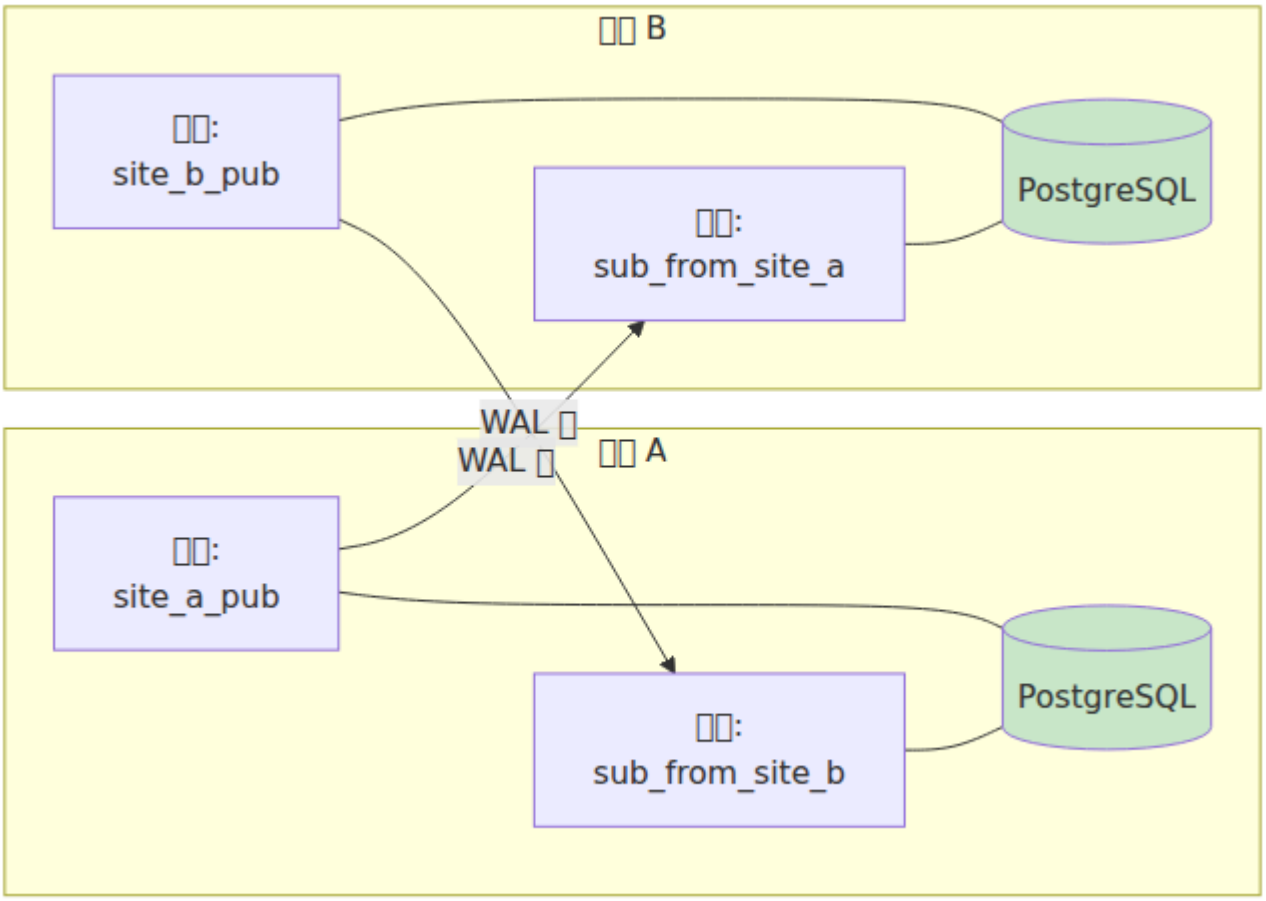
LWW 函数和触发器名称是一个 契约，而不是自由选择。对于表 `foo`，函数必须命名为 `lww_foo()`，触发器必须命名为 `lww_foo_trigger`。

OmniCore 的 `omnihss` Ansible 复制角色 不 保留这些表的自有列表。它会发现每个名称匹配 `lww_<table>_trigger` 的触发器，并确保每个触发器在整个网络中处于 `ENABLE REPLICA` 模式。任何其他名称的触发器对该协调不可见，因此跨站点冲突解决将不会在该表上默默执行。

当您添加一个新的动态状态表（一个在多个站点上同时写入的表）时，请遵循以下步骤：

1. 给表添加一个 `updated_at` 列（Ecto 的 `timestamps()` 宏会添加它）。
2. 在同一迁移中，创建 `lww_<table>()` 和 `lww_<table>_trigger`，名称必须完全相同。
3. 在同一迁移中，运行 `ALTER TABLE <table> ENABLE REPLICA TRIGGER lww_<table>_trigger`，以便触发器在复制应用期间触发，而不是在本地写入时触发。

`ims_subscription` 表是参考示例：��迁移创建 `lww_ims_subscription()` 和 `lww_ims_subscription_trigger`，然后在副本模式下启用触发器。遵循相同的格式，不要重命名函数或触发器。



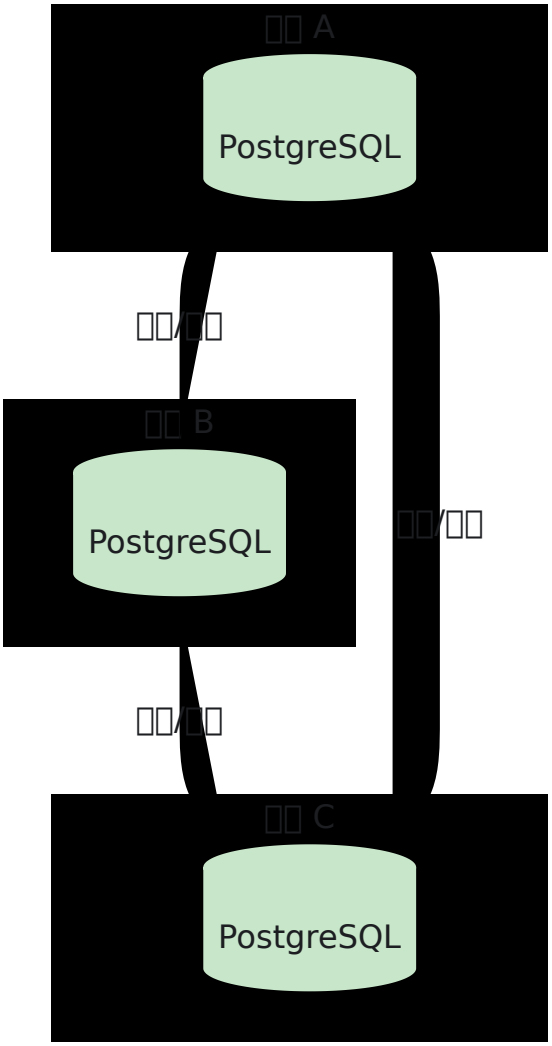
部署架构

单站点（无复制）

多个 HSS 应用实例共享一个 PostgreSQL 数据库。如果需要，可以使用标准 PostgreSQL 流复制进行本地数据库故障转移。

两站点部署

两个站点都是完整的主节点。任何站点上的订阅者配置都会复制到另一个站点。每个 MME 连接到其本地 HSS。没有跨站点的 Diameter 依赖。



多站点部署

每个站点都订阅其他每个站点的发布。对等方从 Ansible 清单中自动发现。要添加一个站点，请将其添加到 `hss` 清单组中并运行剧本。

网络要求

端口	协议	目的
5432	TCP	PostgreSQL 客户端和复制连接

配置参考

Ansible 变量

复制在您的清单中按 HSS 组配置:

```yaml
hss:
  hosts:
    site-a-hss01:
      ansible_host: 10.80.12.140
    site-a-hss02:
      ansible_host: 10.80.12.141
  vars:
    omnihss:
      database_type: postgres
      database_host: localhost
      database_port: 5432
      database_name: omnihss
      database_username: omnihss
      database_password: "secure_password"
      replication:
        enabled: true
```

Ansible □□

□□	□□	□□	□□	□□□
`database_type`	□□□□	□	`postgres`	□□□□□□□□□□□□□ `postgres`□
`database_host`	□□□□	□	`localhost`	PostgreSQL □□□□ Postgres □□ OmniHSS □□□□□□□□□□□□ `localhost`□
`database_port`	□□□	□	`5432`	PostgreSQL □□□
`database_name`	□□□□□	□	`omnihss`	□□□□□□
`database_username`	□□□□	□	-	PostgreSQL □□□□□□□ `CREATEDB` □ `REPLICATION` □□
`database_password`	□□□□	□	-	PostgreSQL □□□
`replication.enabled`	□□□□□	□	`false`	□□□□□□□□□□□ `true` □□ Ansible □□□□□□□□□□□□□□ `pg_hba.conf` □□□

□□□□

□□□□□□□□□□□□□□□□

1. □□□□□□□ `hss` Ansible □□□□□□□□□

2. 远程站点：来自 `prod_master_peers.yml` 的 `connected_sites.hss` 中的条目

不需要每个主机的对等配置。将主机添加到 `hss` 组或 `connected_sites.hss` 中并运行剧本即可。

远程站点对等方 (prod_master_peers.yml)

用于跨地理位置的站点间复制：

```
# group_vars/prod_master_peers.yml
connected_sites:
  hss:
    - remote-site-hss01: 10.80.20.140
    - remote-site-hss02: 10.80.20.141
```

PostgreSQL 配置

当启用复制时，Ansible 角色会自动配置以下 PostgreSQL 参数：

参数	值	目的
`wal_level`	`logical`	逻辑复制所需。在 WAL 中□含完整的行数据。
`max_wal_senders`	`10`	最大并发 WAL 发送进程（每个订阅者一个）。
`max_replication_slots`	`10`	最大复制槽。每个对等方使用一个槽。
`listen_addresses`	`*`	接受来自对等站点的连接。

该角色还会添加 `pg_hba.conf` 条目，以允许来自每个发现的对等 IP 的复制连接。

对等管理

添加新站点

1. 将新主机添加到 `hss` 清单组，或在 `prod_master_peers.yml` 中添加到 `connected_sites.hss`

2. 运行 OmniHSS 剧本

该角色将：

- 在新主机上安装 PostgreSQL 和 OmniHSS
- 运行数据库迁移
- 为新站点创建发布
- 创建对所有现有对等方的订阅
- 更新所有现有对等方的 `pg_hba.conf` 以允许新站点
- 创建从现有对等方到新站点的订阅

移除站点

```
ansible-playbook -i hosts site.yml --tags hss-remove-peer \
  -e "peer_name=site_b confirm_remove=yes"
```

`confirm_remove=yes` 参数是出于安全考虑而必需的。没有它剧本会拒绝继续并显示将要发生的情况。

移除对等方：

- 删除订阅（停止来自该对等方的传入复制）
- 从 `pg_hba.conf` 中移除对等方的 IP
- 已经从对等方复制的数据仍保留在本地数据库中
- 对等方对该站点的订阅必须在对等方上单独移除

临时添加对等方

对于需要在标准剧本流程之外添加对等方的情况：

```
ansible-playbook -i hosts site.yml --tags hss-add-peer \
  -e "peer_name=site_b peer_host=10.80.20.140"
```

监控

Prometheus 指标

postgres_exporter 服务在每个 HSS 主机上运行，端口为 `9187`，向 Prometheus 暴露 PostgreSQL 指标。

复制指标

指标	类型	描述
`pg_replication_slots_active`	Gauge	如果复制槽处于活动状态（对等连接），则为 `1`，如果不活动，则为 `0`
`pg_replication_slots_pg_wal_lsn_diff`	Gauge	以字节为单位的复制延迟。当前 WAL 位置与对等方最后确认位置之间的距离
`pg_stat_replication_pg_current_wal_lsn_bytes`	Gauge	此节点上的当前 WAL 位置

示例 **Prometheus** 查询

```
# 复制槽活动（1 = 连接，0 = 断开连接）
pg_replication_slots_active{slot_name=~"sub_from_.*"}

# 以兆字节为单位的复制延迟
pg_replication_slots_pg_wal_lsn_diff / 1048576

# 警报：复制槽在 60 秒内不活动
pg_replication_slots_active == 0
```

推荐警报

警报	条件	严重性	描述
复制槽不活动	`pg_replication_slots_active == 0` 持续 60 秒	严重	对等方已断开连接。WAL 正在累积。
WAL 延迟过大	`pg_replication_slots_pg_wal_lsn_diff > 104857600` 持续 5 分钟	警告	复制延迟超过 100MB。对等方可能较慢或链接降级。
WAL 槽丢失	WAL 状态为 `lost`	严重	槽已失效。对等方需要完全重新播种。

Ansible 验证

OmniHSS 剧本在每次运行结束时验证复制健康。它断言：

1. 所有订阅工作者都处于活动状态
2. 所有复制槽都处于活动状态
3. WAL 延迟在阈值内（默认 100MB）
4. WAL 槽状态为 `reserved` 或 `extended`（而不是 `lost`）

如果任何断言失败，剧本将失败，并清楚地显示识别问题的错误消息。

手动验证

```
# 检查订阅工作者是否处于活动状态
sudo -u postgres psql -d omnihss -c \
  "SELECT subname, pid IS NOT NULL as worker_alive FROM
pg_stat_subscription;"

# 检查复制槽和延迟
sudo -u postgres psql -d omnihss -c \
  "SELECT slot_name, active,
    pg_wal_lsn_diff(pg_current_wal_lsn(), confirmed_flush_lsn) as
lag_bytes,
    wal_status
  FROM pg_replication_slots WHERE slot_type = 'logical';"

# 比较站点之间的订阅者计数
sudo -u postgres psql -d omnihss -c "SELECT count(*) FROM
subscriber;"
```

故障排除

订阅工作者死亡

症状：`pg_stat_subscription` 显示某个订阅的 `pid IS NULL`。Ansible 验证失败，提示“工作者死亡”。

常见原因：

- 发布者上存在表但订阅者上不存在（DDL 不匹配）
- 手动插入记录导致重复键冲突
- 对等方长时间不可达导致 WAL 槽失效

解决方案：

1. 检查 PostgreSQL 日志以获取特定错误：

   ```
   journalctl -u postgresql | grep "logical replication" | tail
   -20
   ```

2. 如果 DDL 不匹配：在此站点应用缺失的迁移，然后工作者将自动重启
3. 如果重复键：手动解决冲突，然后重新启用订阅：

```
ALTER SUBSCRIPTION sub_from_<peer> ENABLE;
```

复制槽不活动

症状：`pg_replication_slots` 显示 `active = false`。Prometheus 警报触发。

常见原因：

- 站点之间的网络连接问题
- 对等站点上的 PostgreSQL 已停止
- 防火墙阻止来自对等方 IP 的 5432 端口

解决方案：

1. 验证与对等方的网络连接：

```
pg_isready -h <peer_ip> -U <db_user> -d omnihss
```

2. 检查对等站点的 PostgreSQL 是否正在运行
3. 验证防火墙规则允许来自该站点 IP 的 5432 端口

当对等方变得可达时，订阅工作者将自动重新连接。

WAL 槽丢失

症状：`pg_replication_slots` 显示 `wal_status = 'lost'`。Ansible 验证失败，提示“WAL 槽严重”。

原因：对等方断开连接的时间过长，以至于保留的 WAL 超过 `max_slot_wal_keep_size` 并被清理。

解决方案：对等方需要完全重新播种：

1. 在此站点上删除损坏的订阅：

```
DROP SUBSCRIPTION sub_from_<peer>;
```

2. 在对等站点上删除损坏的槽：

```
SELECT pg_drop_replication_slot('sub_from_<this_site>');
```

3. 重新运行 OmniHSS 剧本。它会使用 `copy_data = true` 重新创建订阅以执行初始同步。

WAL 延迟过大

症状：`pg_replication_slots_pg_wal_lsn_diff` 较大且持续增长。Prometheus 警报触发。

常见原因：

- 站点之间的链接延迟高或拥堵
- 对等站点负载过重（应用工作者无法跟上）
- 大批量操作（例如大规模配置）生成大量 WAL

解决方案：

1. 检查延迟是增长还是稳定：

```
SELECT slot_name,
  pg_wal_lsn_diff(pg_current_wal_lsn(), confirmed_flush_lsn) as
lag_bytes
FROM pg_replication_slots WHERE slot_type = 'logical';
```

2. 如果稳定：对等方正在应用更改，但落后。它将赶上。
3. 如果增长：调查网络链接或对等方的负载。
4. 对于批量操作：在操作期间的延迟是预期的，操作完成后将解决。

架构迁移协调

重要：数据库架构更改（迁移）不会通过逻辑复制进行复制。当部署□含架构更改的新 OmniHSS 版本时：

1. 在 所有站点 上应用迁移，然后再部署依赖于新列/表的应用代码
2. Ansible 剧本会自动处理此操作。它会在每个主机上运行迁移
3. 如果手动部署，请首先在所有站点上运行迁移：

```
/opt/omnihss/bin/hss rpc "Hss.Command.Database.migrate()"
```

OmniHSS 实体关系
← 返回操作指南

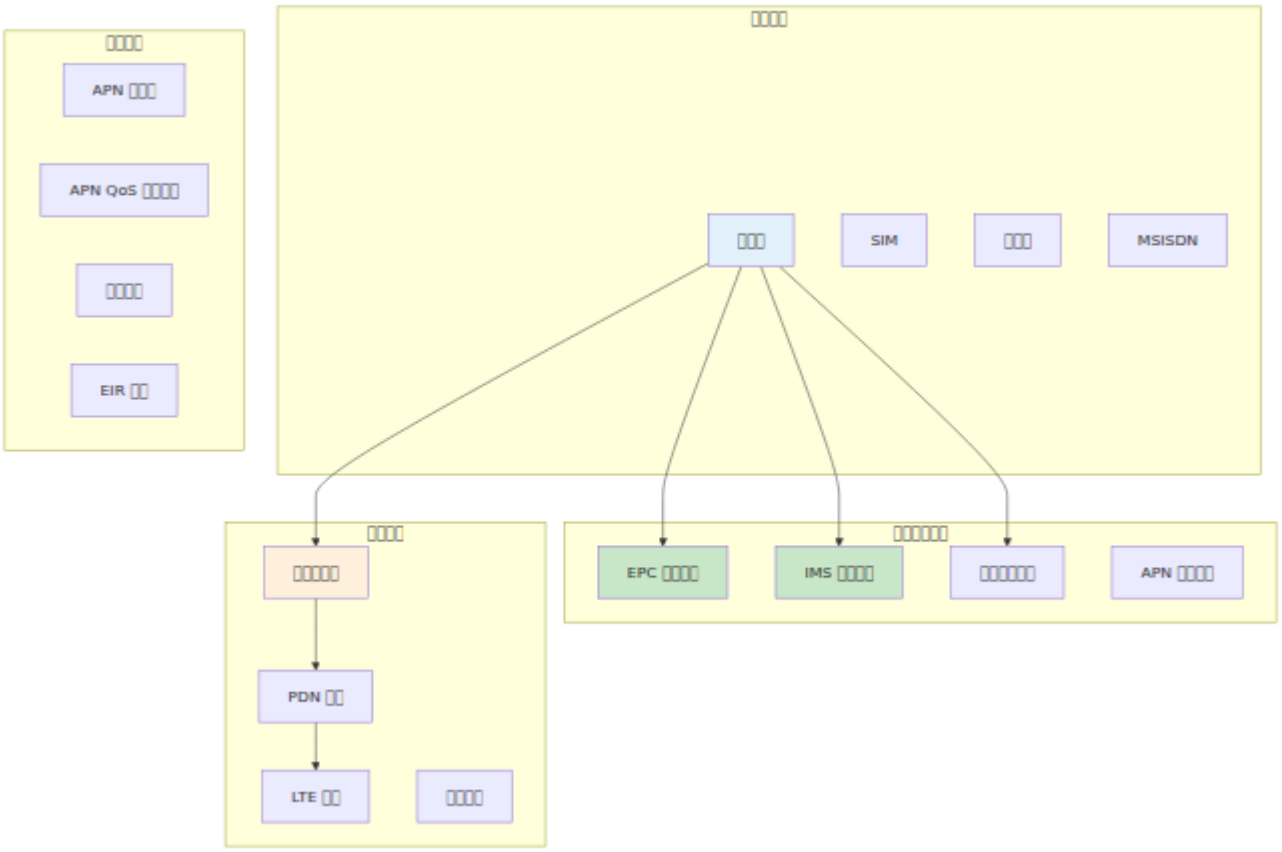
目录

实体概述
核心实体
配置文件实体
状态实体
实体关系图
实体生命周期
数据流模式

实体概述

OmniHSS 将订阅者数据组织成具有清晰关系的逻辑实体。理解这些实体对于诸如配置、故障排除和容量规划等操作任务至关重要。

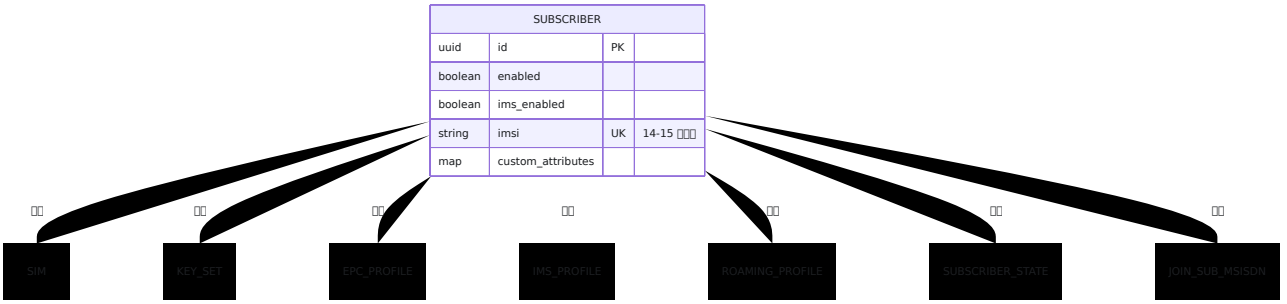
□□□□


□□□□
APN □□□
APN QoS □□□□
□□□□
EIR □□
□□□□
□□□
SIM
□□□
MSISDN
□□□□
□□□□□□
PDN □□
LTE □□
□□□□
□□□□□□□
EPC □□□□
IMS □□□□
□□□□□□□
APN □□□□

□□□□

□□□

□□□□□□□□□□□□□□□□□□□□


SUBSCRIBER
uuid
id
PK
boolean
enabled
boolean
ims_enabled
string
imsi
UK
14-15 □□□
map
custom_attributes
□□
□□
□□
□□
□□
□□
□□
SIM
KEY_SET
EPC_PROFILE
IMS_PROFILE
ROAMING_PROFILE
SUBSCRIBER_STATE
JOIN_SUB_MSISDN

□□□

字段	类型	描述	约束
`id`	uuid	主键	UUID，生成（`gen_random_uuid()`）
`enabled`	boolean	服务启用标志	默认：true
`ims_enabled`	boolean	IMS 服务启用	默认：true
`imsi`	string	国际移动用户身份	14-15 位数字，唯一
`custom_attributes`	map	自定义键值数据	可选
`sim_id`	uuid	SIM 的外键	身份验证所需（在休眠状态时可以为 null）
`key_set_id`	uuid	密钥集的外键	必需
`epc_profile_id`	uuid	EPC 配置文件的外键	数据服务所需（在休眠状态时可以为 null）
`ims_profile_id`	uuid	IMS 配置文件的外键	IMS/语音所需（在休眠状态时可以为 null）
`roaming_profile_id`	uuid	漫游配置文件的外键	可选；如果为 null，则订阅者无法漫游（仅限本地网络）
`subscriber_state_id`	uuid	订阅者状态的外键	自动创建
`ims_subscription_id`	uuid	IMS 订阅的外键	可选

上面的“约束”列说明每个对象在服务中所需的内容，而不是数据库插入行所需的内容。数据库接受仅具有 IMSI 和密钥集的订阅者。这样的订阅者是□□□：它存在，但在您分配匹配的对象之前没有服务可用。使用此方法预配置订阅者，然后稍后激活它们。请参阅 休眠订阅者预配置。

每个对象启用的内容：

要使其工作	订阅者需要
身份验证（附加，IMS 身份验证）	`key_set` 和 `sim`
LTE 数据服务（S6a 附加）	`epc_profile`
IMS / 语音（Cx 注册）	`ims_profile`，以及至少一个 MSISDN
漫游到访问网络	一个具有匹配规则的 `roaming_profile`

如果在订阅者出现在网络上时缺少所需对象，HSS 将拒绝请求并记录确切原因。请参阅 休眠订阅者在网络上的行为。

关键点：

- 每个订阅者必须具有一个 IMSI。
- IMSI 必须为 14-15 位数字（无字母或特殊字符）。
- 一个订阅者可以拥有多个 MSISDN（电话号码）。
- 订阅者状态在创建订阅者时自动创建。
- `enabled` 标志控制所有服务（数据和 IMS）。
- `ims_enabled` 标志仅控制 IMS 服务。
- 数据库仅需要 IMSI 和密钥集来插入订阅者。上面列出的每个其他对象都需要其自己的服务。没有 IMS 配置文件的订阅者无法使用 IMS，即使记录在没有配置文件的情况下也能保存。
- 没有 `ims_subscription_id` 的订阅者是独立的。将 `ims_subscription_id` 设置为将多个订阅者（IMSI）分组到一个 IMS 订阅中，该订阅共享公共身份和单一 S-CSCF。请参阅 IMS 订阅 和 共享公共用户身份。

SIM

SIM 实体表示物理或嵌入式 SIM 卡。

字段：

字段	类型	描述	安全级别
`iccid`	string	集成电路卡 ID	公开
`sim_vendor`	string	SIM 制造商	公开
`batch_name`	string	生产批次	公开
`is_esim`	boolean	嵌入式 SIM 标志	公开
`pin1`, `pin2`	string	PIN 代码	敏感
`puk1`, `puk2`	string	PUK 代码	敏感
`adm1` - `adm10`	string	管理代码	高度敏感
`kic`, `kid`	binary	OTA 安全密钥	高度敏感

关键点：

- ICCID 唯一标识 SIM 卡
- 一张 SIM 卡一次只能分配给一个订阅者
- PIN/PUK 代码用于最终用户 SIM 锁定
- ADM 代码用于管理 SIM 操作
- KIC/KID 用于 SIM OTA（空中）更新

密钥集

密钥集□含用于身份验证的加密密钥。

KEY_SET			
uuid	id	PK	
binary	ki		128 位
binary	opc		128 位
binary	op		128 位
binary	amf		16 位
bigint	sqn		48 位序列
string	authentication_algorithm		

被使[

字段:

字段	类型	描述	大小
`ki`	binary	秘密密钥	128 位（16 字节）
`opc`	binary	运营商变体密钥（派生）	128 位
`op`	binary	运营商密钥（用于派生 OPC）	128 位
`amf`	binary	身份验证管理字段	16 位（2 字节）
`sqn`	bigint	序列号（防重放）	48 位
`authentication_algorithm`	string	算法名称	当前为 "milenage"
`ota_counter`	bigint	OTA 操作计数器	整数

关键点：

- 多个订阅者可以共享相同的密钥集
- Ki 是与 SIM 共享的主密钥
- 必须提供 OPC 或 OP（OPC 可以从 OP 派生）
- SQN 在每次身份验证时递增
- Milenage 目前是唯一支持的算法

身份验证算法：

密钥集、**SIM** 和订阅者

SIM、密钥集和订阅者是独立的对象。每个对象都有自己的生命周期。订阅者是将它们链接的记录。

- **SIM** - 物理卡或 eSIM。它保存 ICCID 以及 PIN、PUK 和 ADM 代码。它是库存。
- 密钥集 - 身份验证密钥（Ki、OPc、AMF、SQN）。网络根据这些密钥进行身份验证。
- 订阅者 - 网络身份和服务（IMSI、MSISDN、配置文件、启用状态）。

订阅者引用一个 SIM (`sim_id`) 和一个密钥集 (`key_set_id`)。密钥集不引用 SIM。要查找密钥集属于哪个 SIM，请读取引用两者的订阅者。

KEY_SET			
uuid	id	PK	
binary	ki		128 □
binary	opc		128 □
binary	op		128 □
binary	amf		16 □
bigint	sqn		48 □□□
string	authentication_algorithm		

为什么对象是分开的

对象是分开的，因为它们在每种情况下并不一一对应：

- **SIM** 更换。订阅者掉落手机并需要新的 SIM。您保留订阅者及其 MSISDN、配置文���和 IMS 数据。您将其指向新的 SIM 和新的密钥集。服务继续。
- 多 **IMSI SIM**。一张 ICCID 承载多个 IMSI。每个 IMSI 是一个独立的订阅者。有些 IMSI 共享一个密钥集。有些 IMSI 有自己的密钥集。密钥属于 IMSI，而不是卡。请参阅 多 IMSI SIM 支持。
- 托管或赞助漫游。 SIM 小程序使用访问 IMSI 进行身份验证。主网络在请求到达 HSS 之前将访问 IMSI 转换为您的本地 IMSI。这两个 IMSI 共享一个密钥集。

因此，密钥集可以服务于多个订阅者。这就是为什么密钥集是一个独立的对象，而不是 SIM 上的一个字段。

导入时的关联

SIM 供应商文件为您提供 ICCID 和密钥在一行中。在导入时，您创建密钥集，创建 SIM，并创建引用两者的订阅者。订阅者是连接记录。有关导入工作流，请参阅 休眠订阅者预配置。

MSISDN

MSISDN 表示一个电话号码。

MSISDN			
uuid	id	PK	
string	msisdn	UK	1-15 位数字

分配[

JOIN_SUB_MSISDN		
uuid	subscriber_id	FK
uuid	msisdn_id	FK

属[

字段:

字段	类型	描述	格式
`msisdn`	string	移动站 ISDN 号码	1-15 位数字, E.164 格式

关键点:

- MSISDN 是国际格式的电话号码。
- 一个订阅者可以有多个 MSISDN（一个 IMSI，多个号码）。

- 一个 MSISDN 可以分配给多个订阅者（多个 IMSI，一个号码）。例如，用于共享号码，例如手机和智能手表。该号码存储为一个 MSISDN 记录，每个订阅者持有一个连接行。
- 通过 MSISDN 进行反向查找（例如 API `GET /subscriber/msisdn/:msisdn`，或以号码为键的 Diameter 查找）返回一个订阅者。当号码被共享时，使用 IMSI 或 ICCID 选择特定的订阅者。
- 格式：国家代码 + 国内号码，仅数字 - 不要 `+` 前缀（例如，`14155551234` 而不是 `+14155551234`）
- 在 Diameter Cx/Rx 查找期间，当 S-CSCF 或 P-CSCF 发送带或不带 `+` 的 SIP URI 时，`+` 前缀会自动去除。

多 **MSISDN** 模式：

配置文件实体

EPC 配置文件

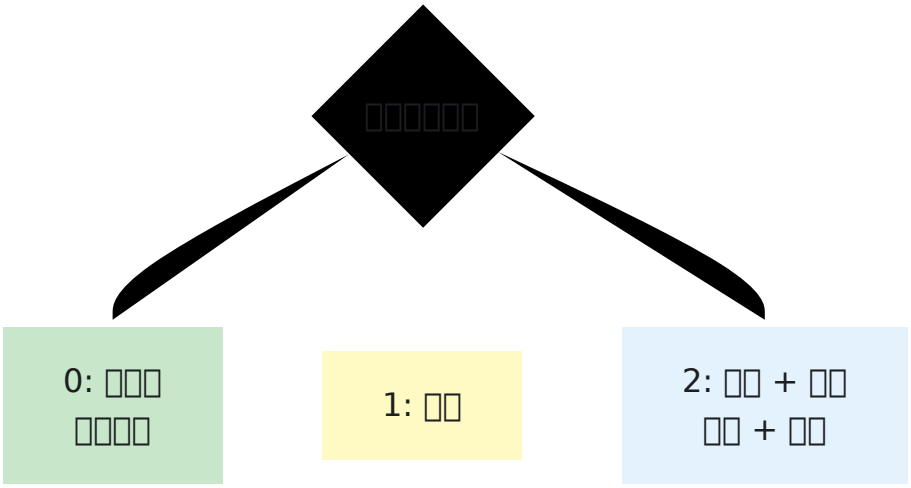
EPC 配置文件定义 LTE 的数据服务特性。

EPC_PROFILE			
uuid	id	PK	
string	name	UK	
integer	ue_ambr_dl_kbps		
integer	ue_ambr_ul_kbps		
string	network_access_mode		仅数据或数据和电路
integer	tracking_area_update_interval_seconds		

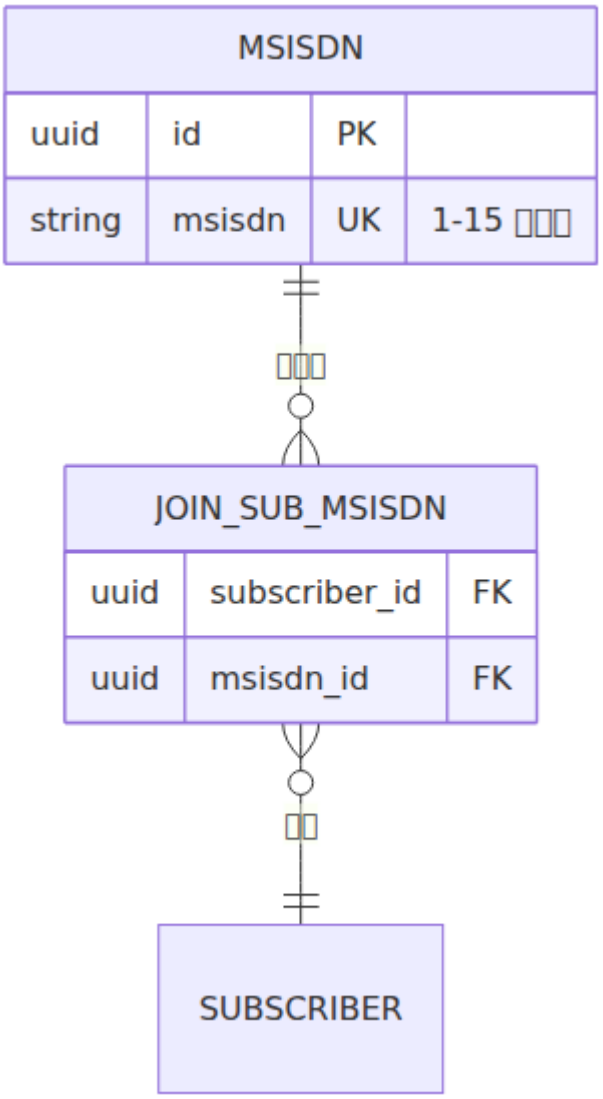
字段:

□□	□□	□□	□□
`name`	string	□□□□□□	□□
`ue_ambr_dl_kbps`	integer	□□□□□□	Kbps
`ue_ambr_ul_kbps`	integer	□□□□□□	Kbps
`network_access_mode`	string	□□□□	"□□□" □ "□□□□□"
`tracking_area_update_interval_seconds`	integer	TAU □□	□

□□□□□□□□

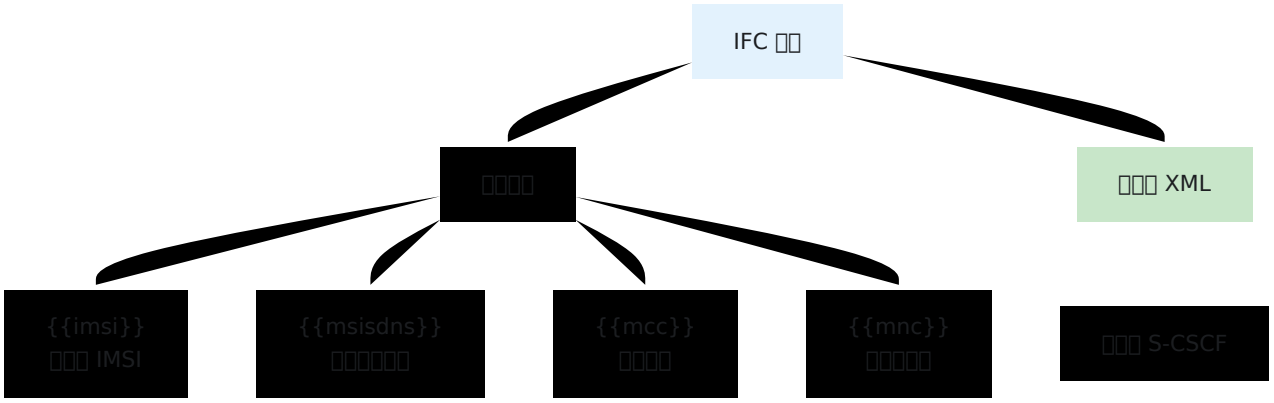


AMBR□□□□□□□□□□□□



IMS 配置文件

IMS 配置文件定义语音/视频服务特性。



IMS_PROFILE		
uuid	id	PK
string	name	UK
text	ifc_template	

分配[

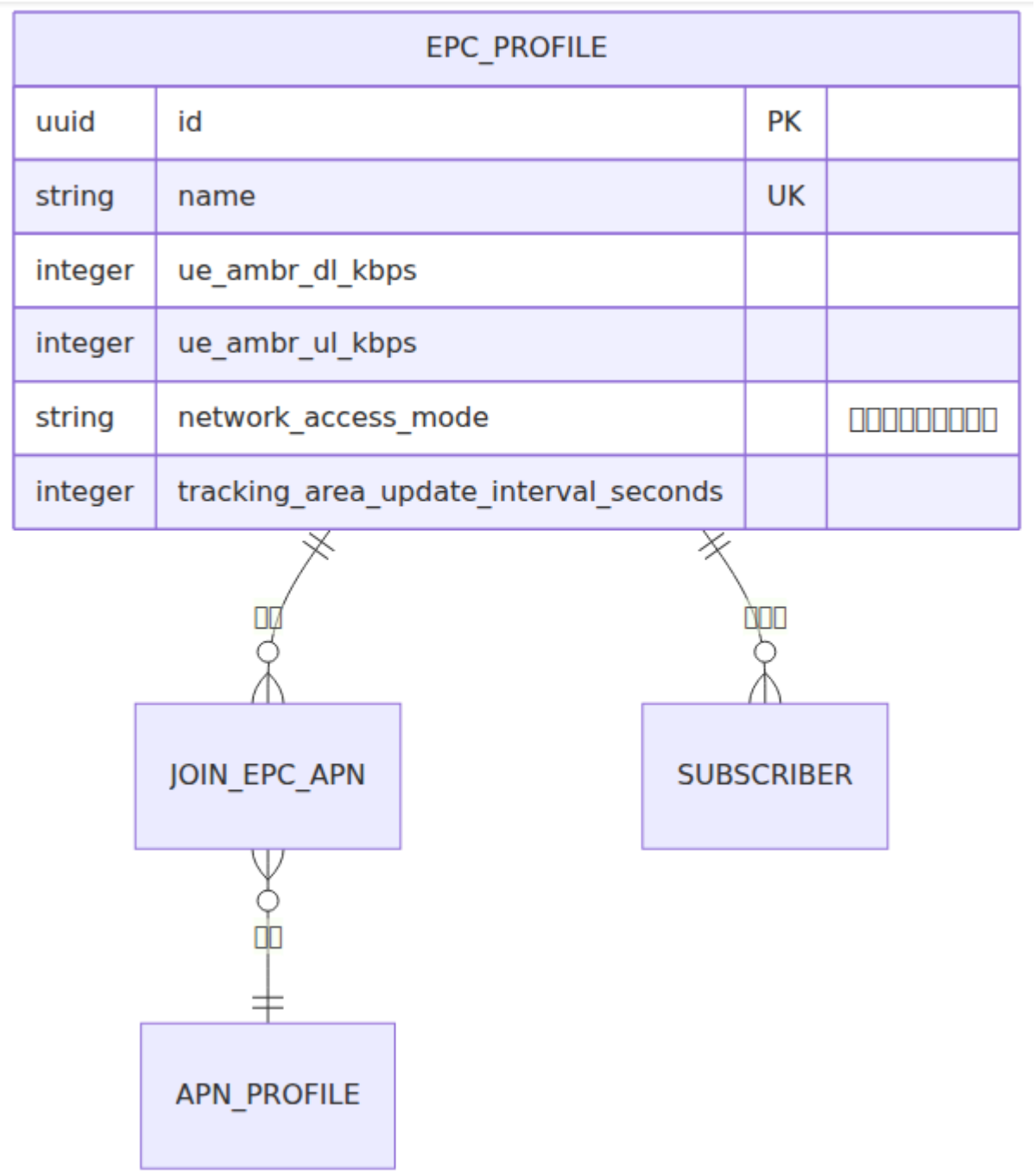
字段:

字段	类型	描述	格式
`name`	string	配置文件名称	文本
`ifc_template`	text	初始过滤标准 XML 模板	带变量的 XML

IFC 模板变量:

关键点:

- IFC（初始过滤标准）控制 IMS 中的呼叫路由
- 模板在订阅者注册时渲染



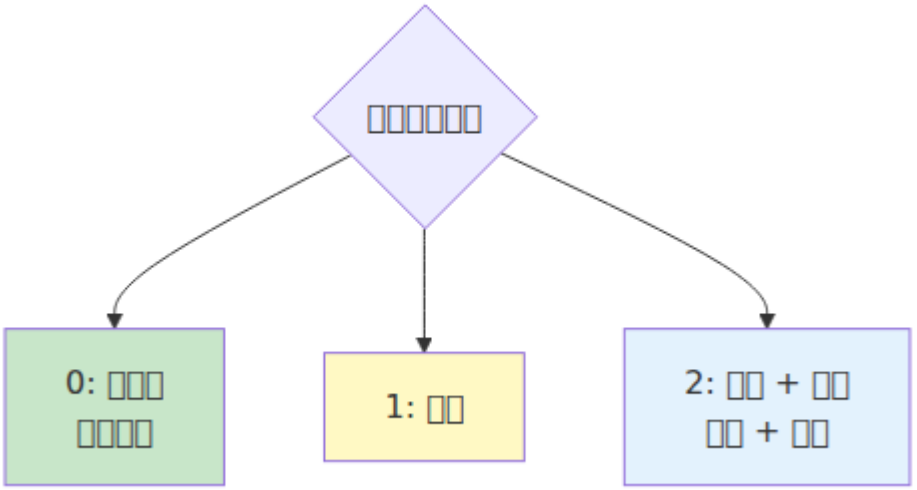
- 变量用实际的订阅者数据替换
- 在 IMS 注册期间发送到 S-CSCF

APN 配置文件

APN 配置文件定义特定数据接入点的特性。

相关实体:

APN 标识符



APN_IDENTIFIER			
uuid	id	PK	
string	apn	UK	
string	ip_version		ipv4, ipv6, ipv4v6, 或 ipv4_or_ipv6

字段	类型	描述	示例
`apn`	string	APN 名称	"internet", "ims", "mms"
`ip_version`	string	IP 协议支持	见下文

IP 版本选项:

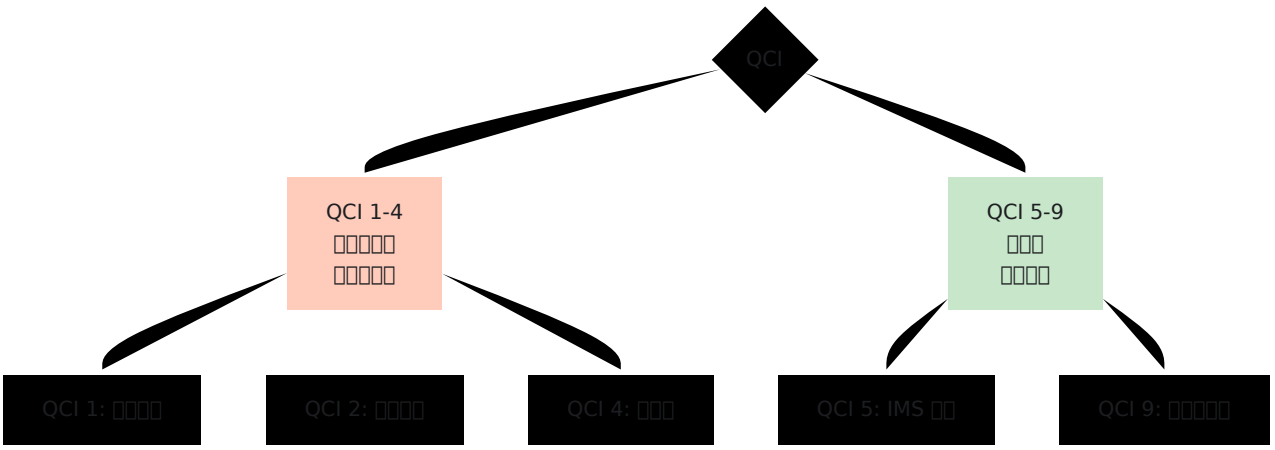
APN QoS 配置文件

APN_QOS_PROFILE		
uuid	id	PK
string	name	UK
integer	qci	
integer	allocation_retention_priority	
integer	apn_ambr_dl_kbps	
integer	apn_ambr_ul_kbps	
boolean	pre_emption_capability	
boolean	pre_emption_vulnerability	

QoS 参数:

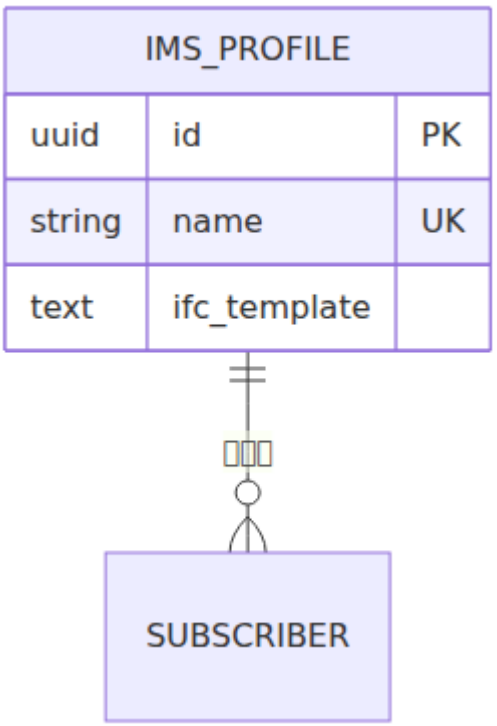
参数	描述	范围	默认承载
`qci`	QoS 类别标识符	1-9	QCI 9（互联网）
`allocation_retention_priority`	ARP 优先级	1-15	8（较低优先级）
`apn_ambr_dl_kbps`	APN 下载限制	0+	变化
`apn_ambr_ul_kbps`	APN 上传限制	0+	变化
`pre_emption_capability`	可以抢占其他	true/false	false
`pre_emption_vulnerability`	可以被抢占	true/false	true

QCI 值:

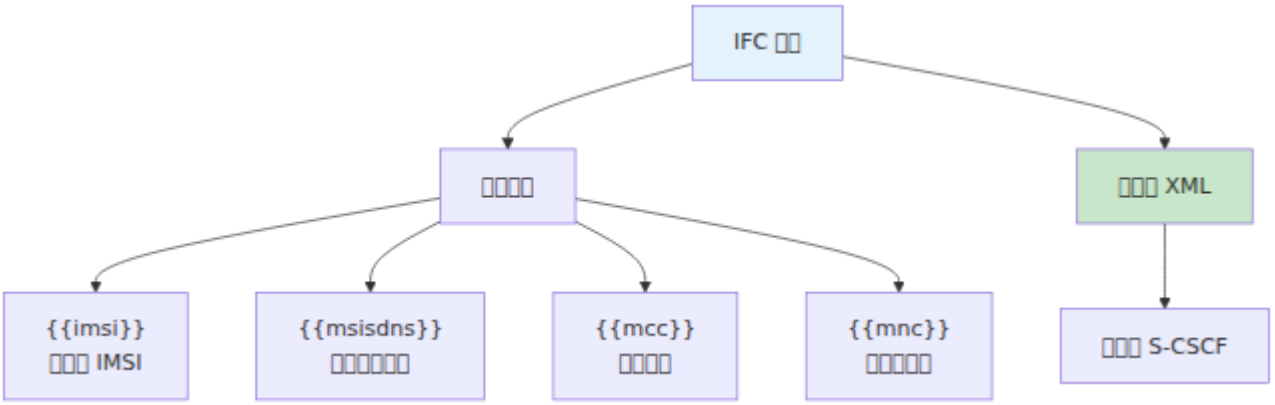

QCI
QCI 1-4
□□□□□□
□□□□□□
QCI 5-9
□□□
□□□□
QCI 1: □□□□
QCI 2: □□□□
QCI 4: □□□
QCI 5: IMS □□
QCI 9: □□□□□

□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□


IMS_PROFILE
uuid
id
PK
string
name
UK
text
ifc_template
□□□
SUBSCRIBER

□□□□□



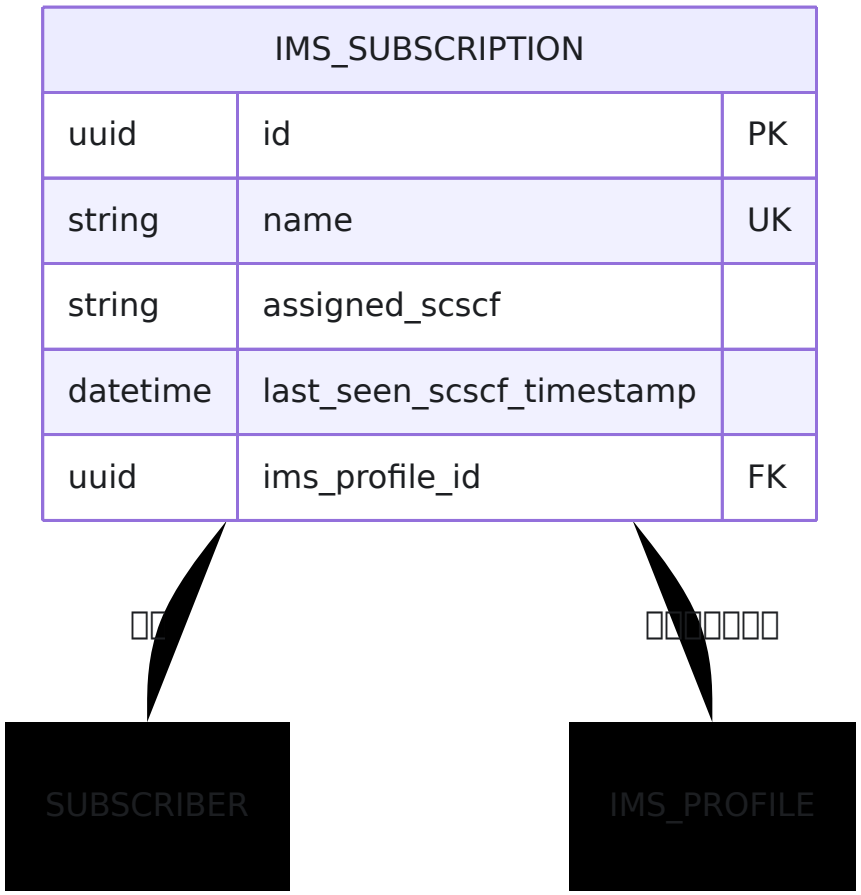
ROAMING_RULE			
uuid	id	PK	
string	name	UK	
string	mcc		3 位数字
string	mnc		2-3 位数字
string	data_action		允许或拒绝
string	ims_action		允许或拒绝

规则评估:

状态实体

IMS 订��

IMS 订阅将多个订阅者（IMPIs / IMSIs）分组到一个 IMS 订阅中，该订阅共享公共用户身份和单一服务 S-CSCF（3GPP TS 23.228 共享 IMPU）。用于在多个设备上提供一个号码，例如在不同 SIM 上的手机和智能手表。

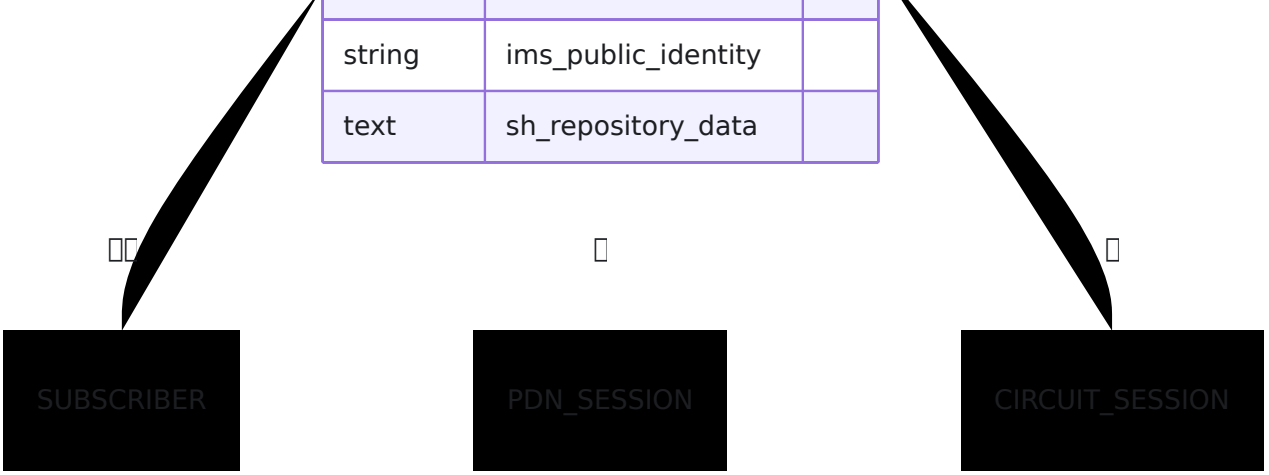


字段：

字段	类型	描述	约束
`name`	string	可读名称	必需，唯一
`assigned_scscf`	string	组的共享服务 S-CSCF	运行时状态，由 HSS 设置
`last_seen_scscf_timestamp`	datetime	上次写入共享 S-CSCF 的时间	运行时状态
`ims_profile_id`	uuid	可选的共享服务配置文件，用于组的共享身份	可选

关键点：

- 成员通过 `subscriber.ims_subscription_id` 引用组。
- HSS 在第一个成员注册时分配一个 S-CSCF，并在最后一个成员注销时释放它。请勿自行设置 `assigned_scscf`。



- 当设置 `ims_profile_id` 时，组的 IMS 配置文件呈现共享身份；���则每个成员呈现自己的 IMS 配置文件。
- 删除 IMS 订阅不会删除其订阅者；它们的 `ims_subscription_id` 会被清除。

查看 共享公共用户身份 以获取完整模型和配置。

订阅者状态

订阅者状态 跟踪实时订阅者状态。

SUBSCRIBER_STATE		
uuid	id	PK
string	last_seen_mme	
string	last_seen_realm	
integer	last_seen_rat_type	
datetime	last_seen_at	
string	last_seen_mcc	
string	last_seen_mnc	
string	last_seen_tac	
string	last_seen_cell_id	
string	last_seen_enodeb_id	
string	last_seen_eci	
string	assigned_scscf	
string	ims_public_identity	
text	sh_repository_data	

关键字段：

位置信息：

- `last_seen_mcc`, `last_seen_mnc` - 访问的网络
- `last_seen_tac` - 跟踪区域代码
- `last_seen_cell_id` - 小区 ID
- `last_seen_enodeb_id` - eNodeB ID
- `last_seen_eci` - E-UTRAN 小区标识符

网络元素：

- `last_seen_mme` - 当前服务订阅者的 MME
- `last_seen_realm` - MME 的 Diameter 领域
- `last_seen_rat_type` - 无线接入技术 (LTE, 5G 等)

IMS 信息：

- `assigned_scscf` - 当前服务订阅者的 S-CSCF
- `ims_public_identity` - SIP URI (例如, sip:+14155551234@ims.example.com)
- `sh_repository_data` - 自定义 IMS 配置文件数据

时间戳：

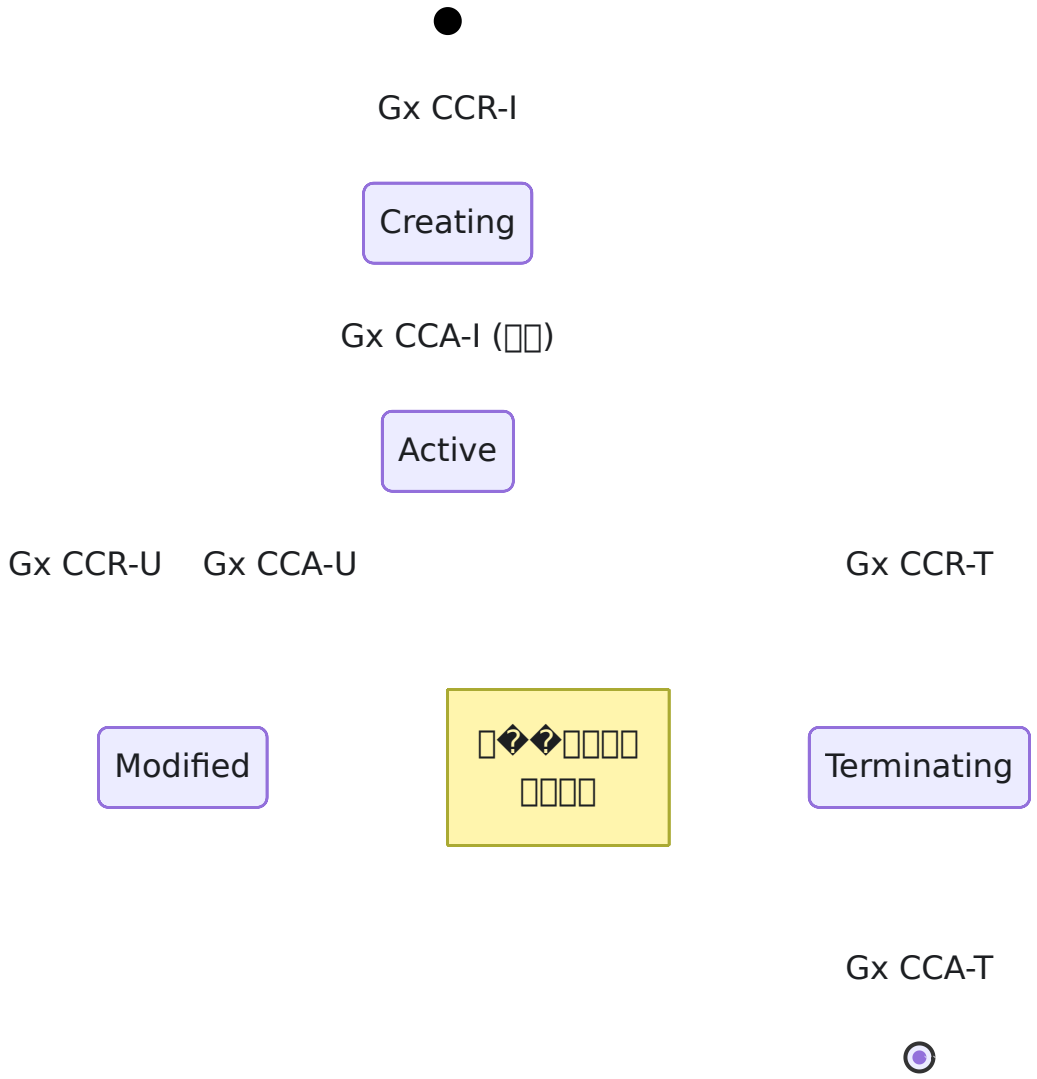
- `last_seen_at` - 最近接收到的 Diameter 消息
- 各种 `last_*_at` 时间戳用于不同的过程

PDN 会话

PDN 会话 表示一个活跃的数据连接。

APN_IDENTIFIER			
uuid	id	PK	
string	apn	UK	
string	ip_version		ipv4, ipv6, ipv4v6, □ ipv4_or_ipv6

PDN 会话生命周期：



LTE □□

LTE □□ □□□□□□□□ VoLTE □□/□□□□□□

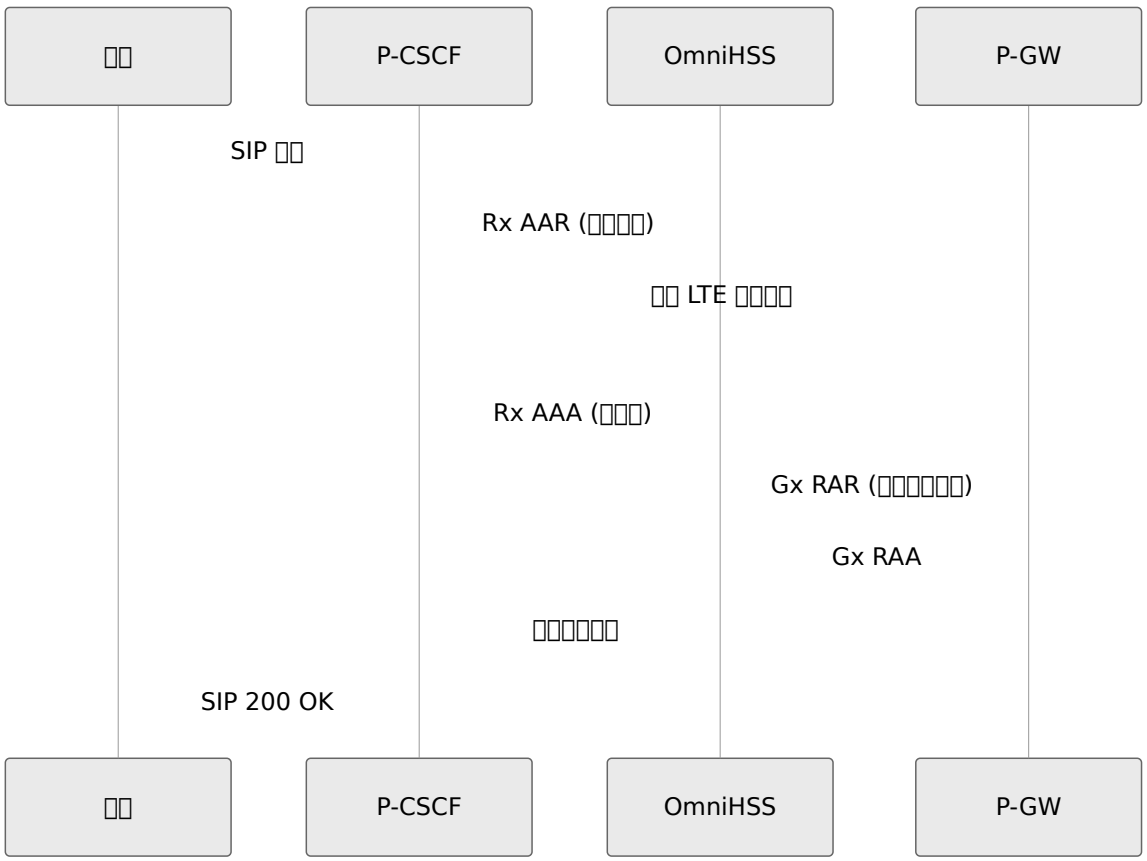
LTE_CALL		
uuid	id	PK
string	pcscf_session_id	
string	assigned_pcscf_host	
string	codec	
integer	type	
boolean	dedicated_bearer_active	
integer	dedicated_bandwidth_ul	
integer	dedicated_bandwidth_dl	
text	flow_rules	
datetime	created_at	

□□

□□□□□

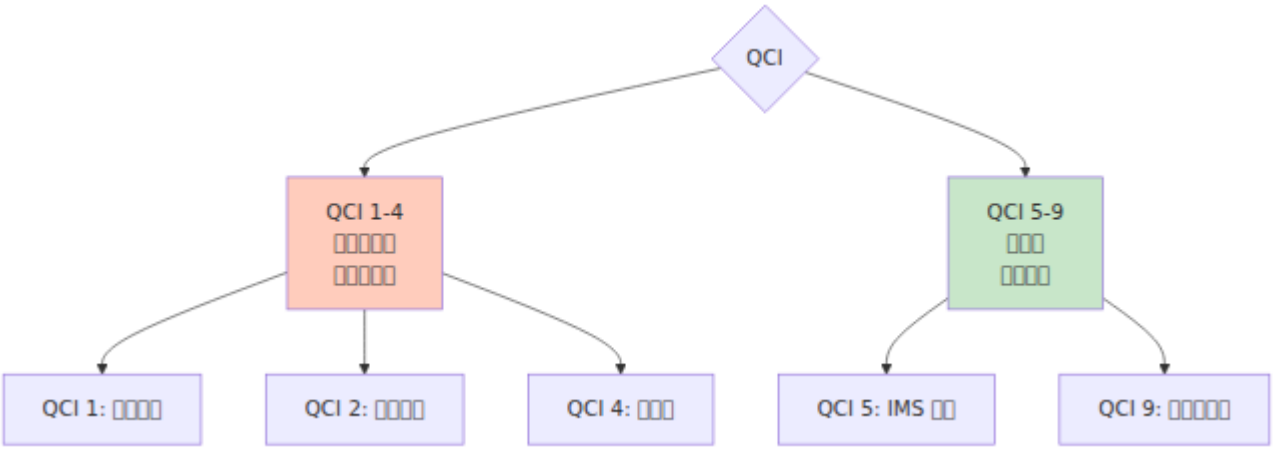
APN_QOS_PROFILE		
uuid	id	PK
string	name	UK
integer	qci	
integer	allocation_retention_priority	
integer	apn_ambr_dl_kbps	
integer	apn_ambr_ul_kbps	
boolean	pre_emption_capability	
boolean	pre_emption_vulnerability	

VoLTE □□□□□



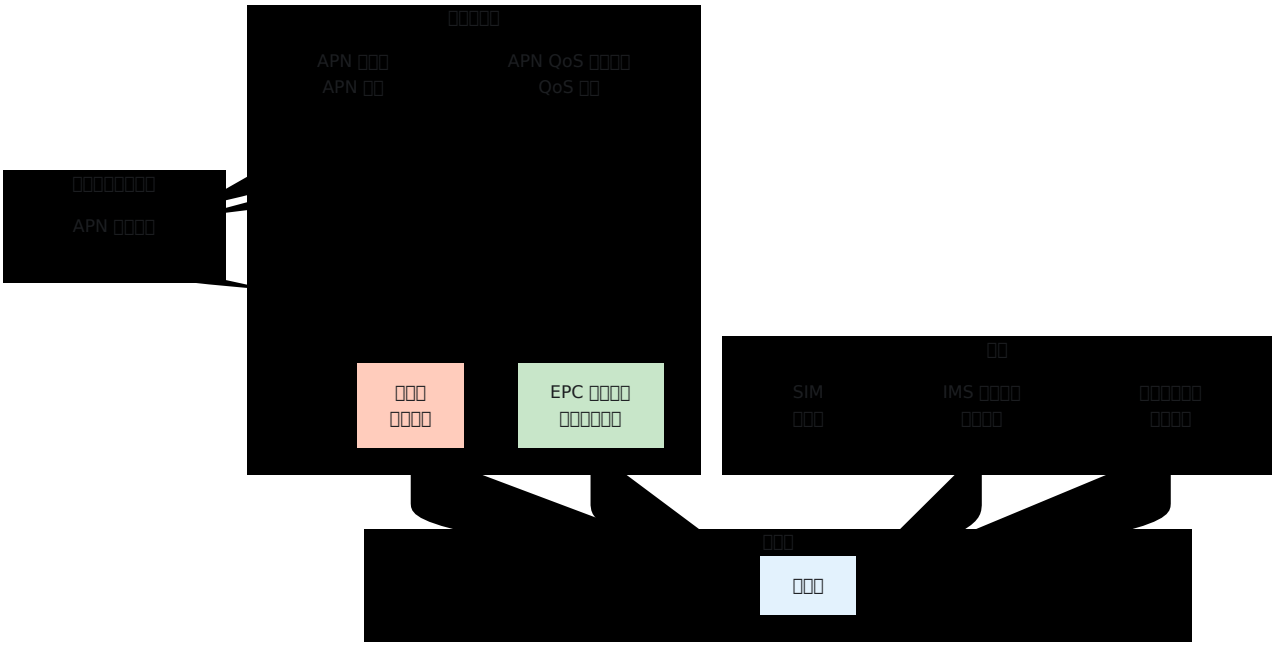
□□□□□

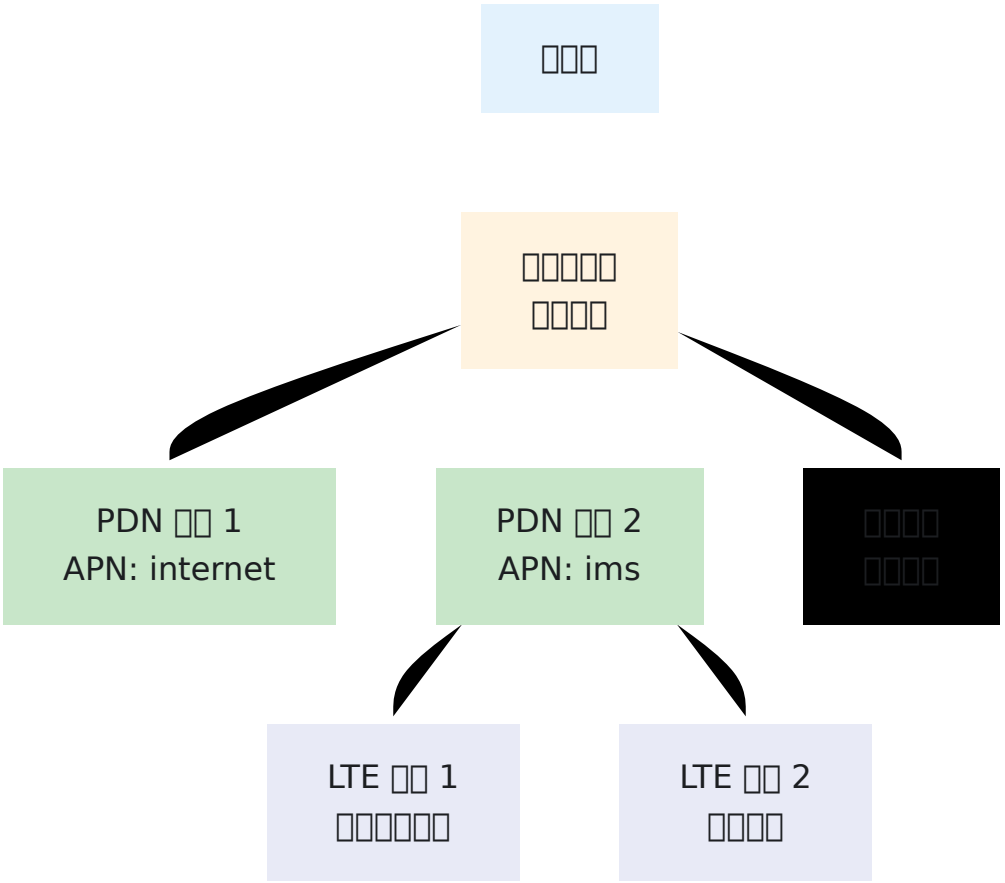
□□□□□□


QCI
QCI 1-4
□□□□□
□□□□□
QCI 5-9
□□□
□□□□
QCI 1: □□□□
QCI 2: □□□□
QCI 4: □□□
QCI 5: IMS □□
QCI 9: □□□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□□


□□□□□
APN □□□
APN □□
APN QoS □□□□
QoS □□
□□□□□□□□□
APN □□□□
□□□
□□□□
EPC □□□□
□□□□□□
□□
SIM
□□□
IMS □□□□
□□□□
□□□□□□
□□□□
□□□
□□□



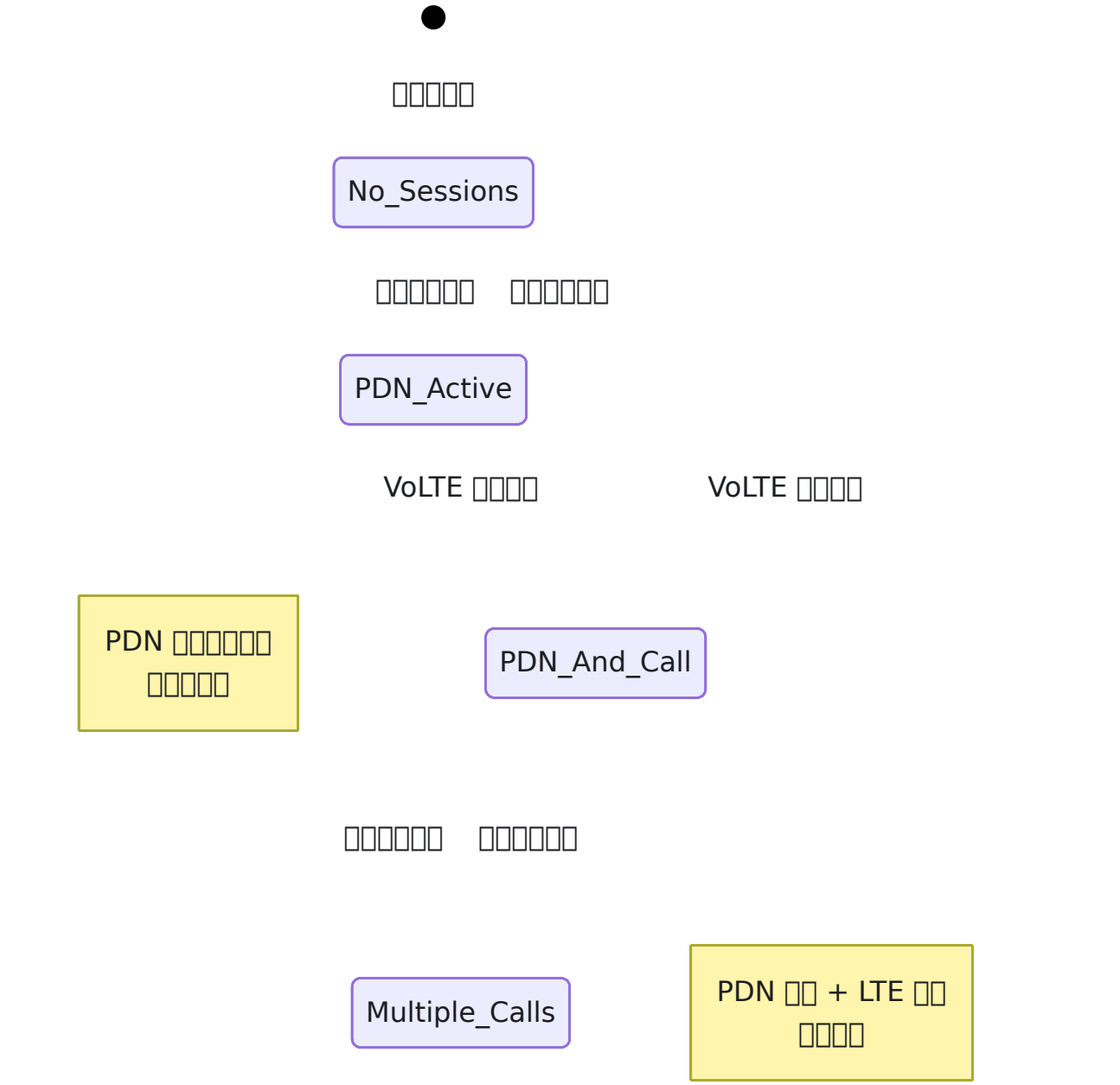
会话状态关系

订阅者
订阅者状态
始终存在
PDN 会话 1
APN: internet
PDN 会话 2
APN: ims
LTE 通话 1
语音通话活跃
LTE 通话 2
会议通话

实体生命周期

订阅者配置生命周期

ROAMING_RULE			
uuid	id	PK	
string	name	UK	
string	mcc		3 □□□
string	mnc		2-3 □□□
string	data_action		□□□□□
string	ims_action		□□□□□

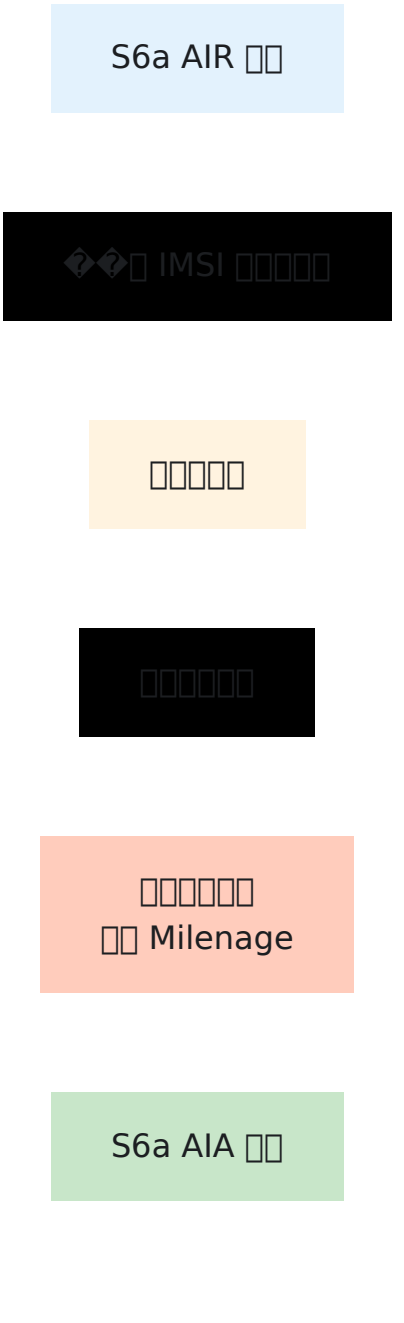


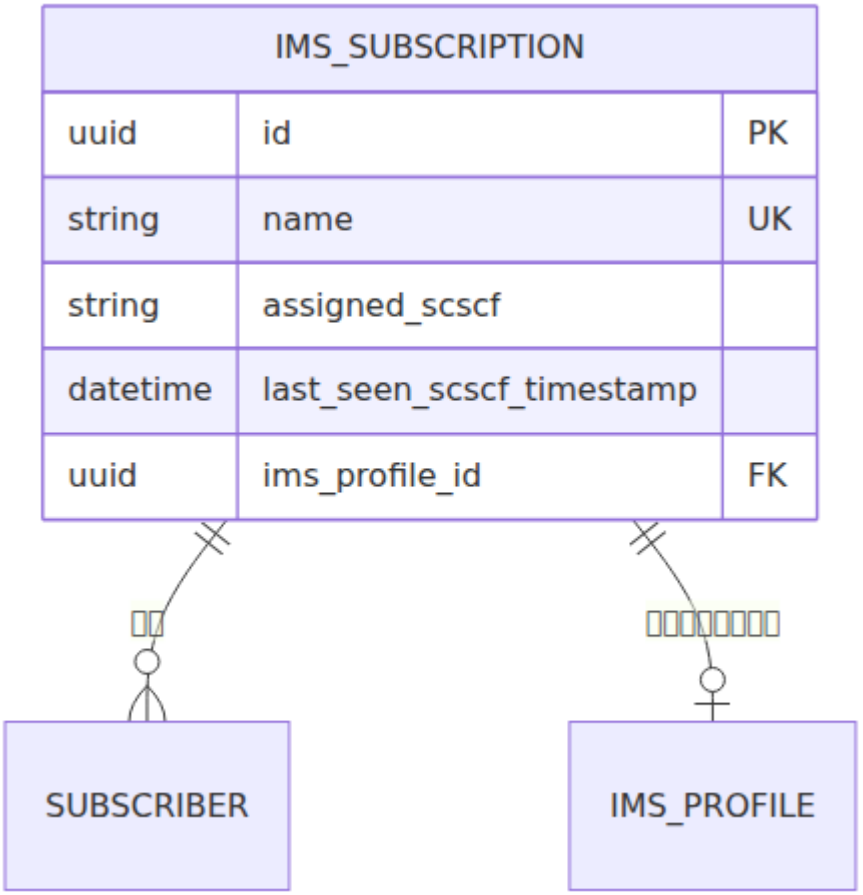
会话生命周期

订阅者空闲
No_Sessions
数据连接开始
数据连接结束
PDN_Active
VoLTE 通话开始
VoLTE 通话结束
PDN 会话记录存在
在数据库中
PDN_And_Call
第二通话开始
第二通话结束
Multiple_Calls
PDN 会话 + LTE 通话
记录存在

□□□□□

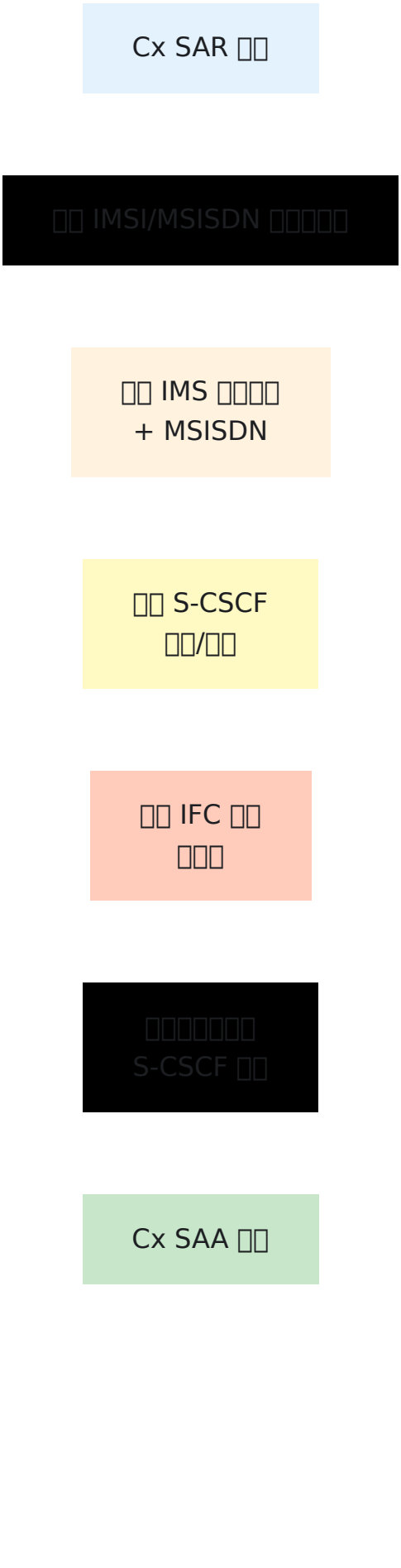
□□□□


S6a AIR □□
□□□□□□ IMSI □□□□□□
□□□□□□
□□□□□□□
□□□□□□□
□□ Milenage
S6a AIA □□



位置更新流程

IMS_SUBSCRIPTION
uuid
id
PK
string
name
UK
string
assigned_scscf
datetime
last_seen_scscf_timestamp
uuid
ims_profile_id
FK
SUBSCRIBER
IMS_PROFILE



IMS 注册流程

Cx SAR 请求

□□ IMSI/MSISDN □□□□□

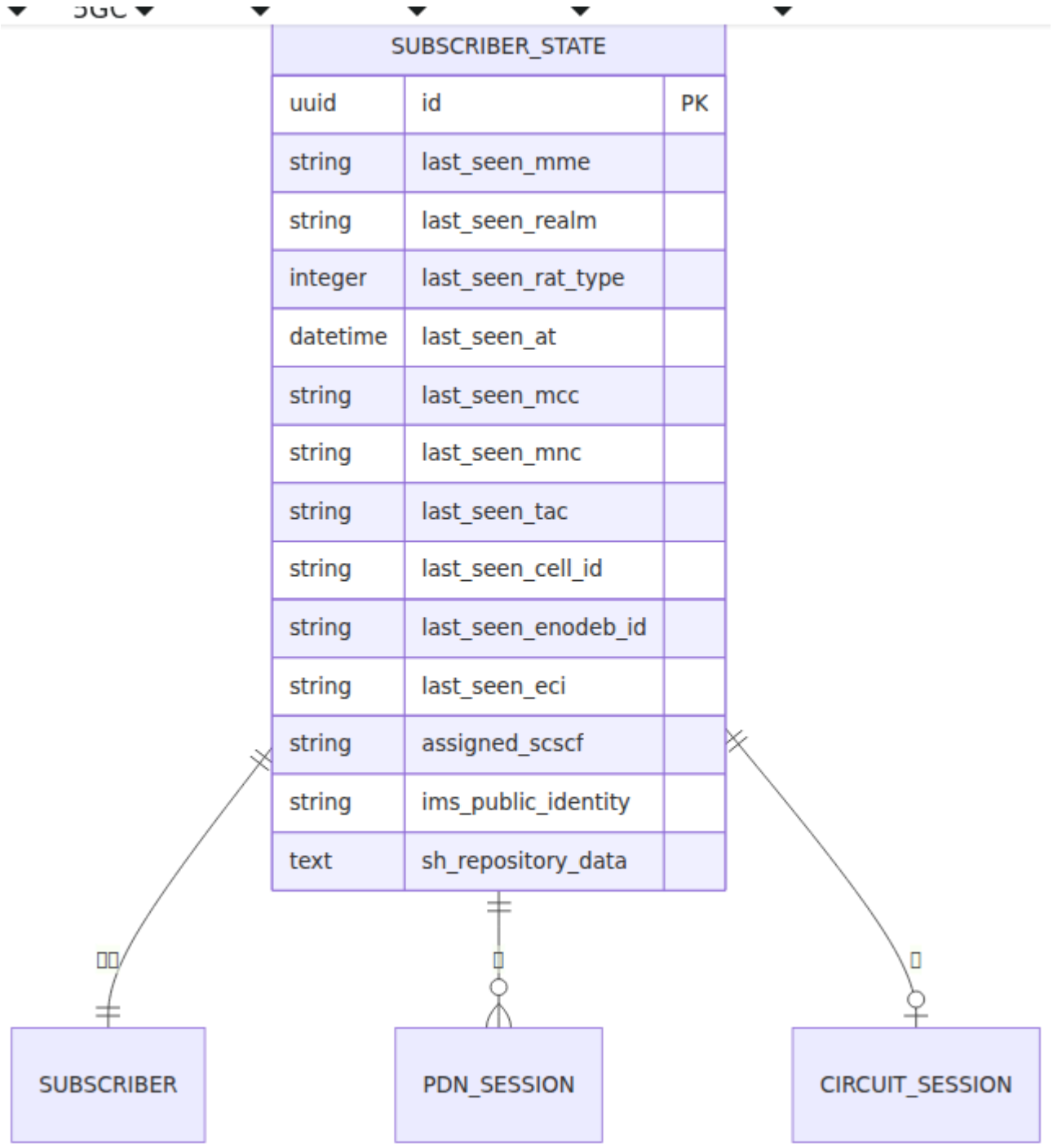
加载 IMS 配置文件
+ MSISDN

选择 S-CSCF
随机/轮询

渲染 IFC 模板
带变量

□□□□□□□
S-CSCF □□

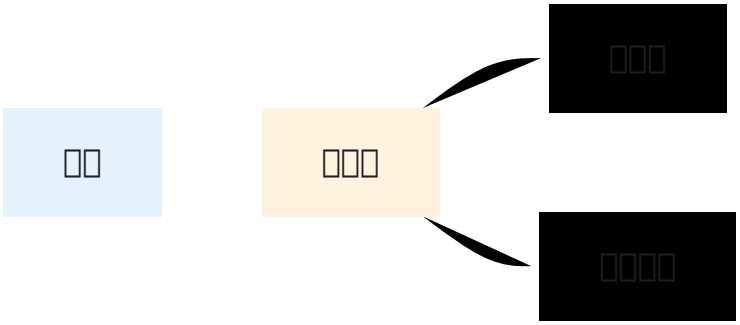
Cx SAA 响应



会话建立流程

查询优化模式

OmniHSS 通过选择性预加载每个操作所需的关联来优化数据库查询：

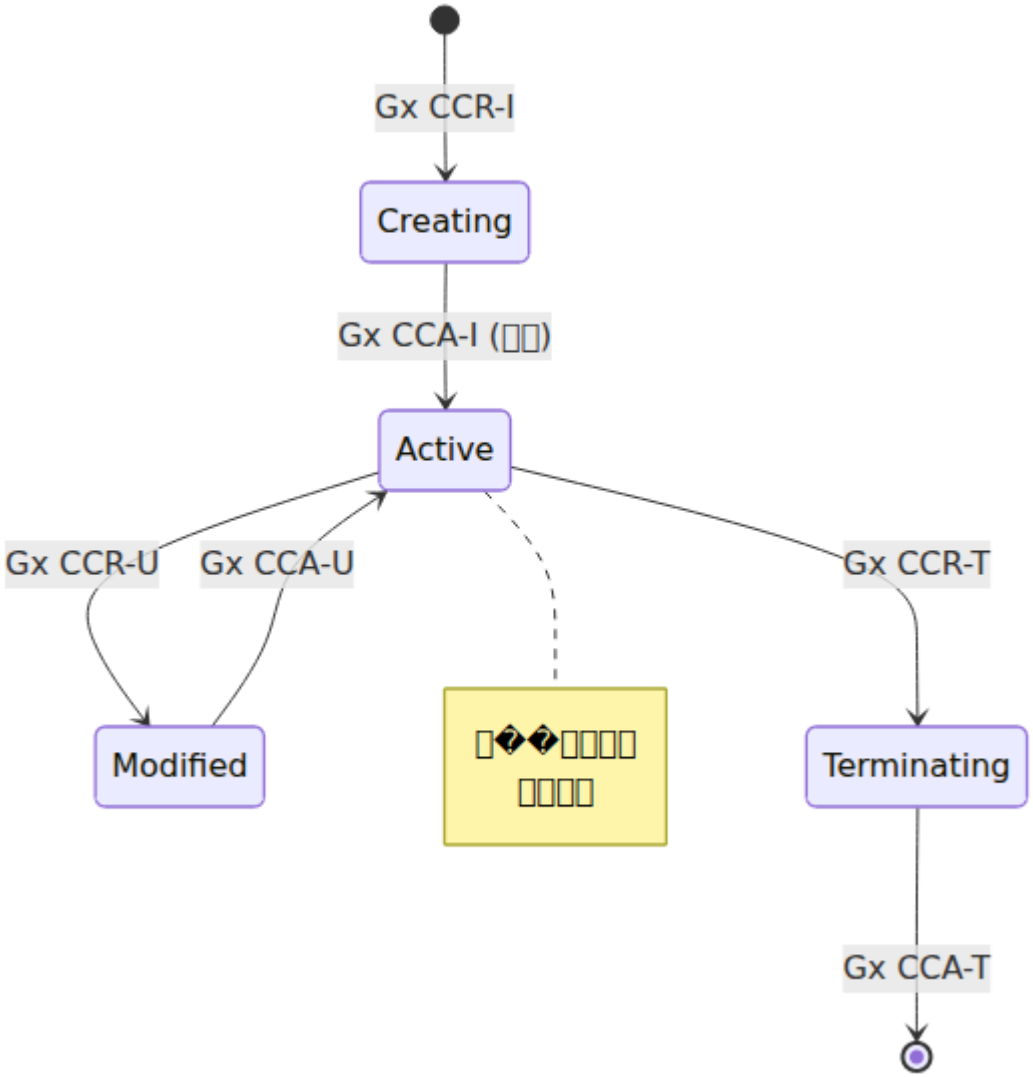


最小查询【认证】

用例： S6a AIR - 仅需要加密密钥和漫游规则

中等查询【位置更新】

用例： S6a ULR - 需要完整的 EPC 配置文件数据



完整查询（IMS 注册）

用例： Cx SAR - 需要 IMS 配置文件和所有电话号码

← 返回操作指南 | 下一步：API 参考 →

Diameter 响应数据映射

← 返回操作指南

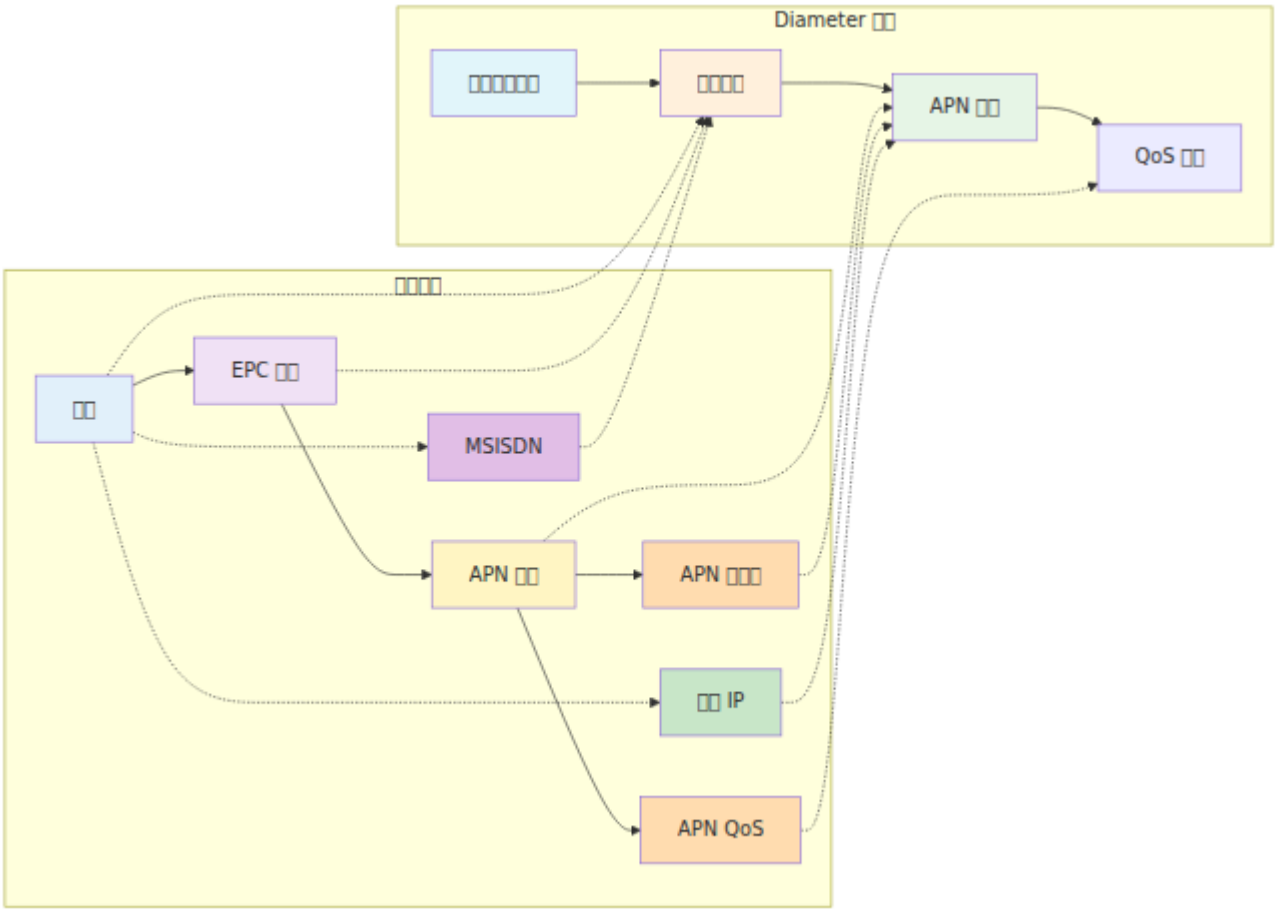
本文档提供详细的 mermaid 图表，展示了 Diameter 协议响应中每个字段在 OmniHSS 系统中的来源。

目录

- 更新位置应答 (S6a ULA)
- 认证信息应答 (S6a AIA)
- 服务器分配应答 (Cx SAA)
- 计费控制应答 (Gx CCA)
- 用户数据应答 (Sh UDA)
- ME 身份检查应答 (S13 ECA)

更新位置应答 (S6a ULA)

更新位置应答由 HSS 在 LTE 附着过程中发送给 MME。该图展示了从数据库表到 Diameter AVP 的完整数据流。



数据源映射

详细字段映射

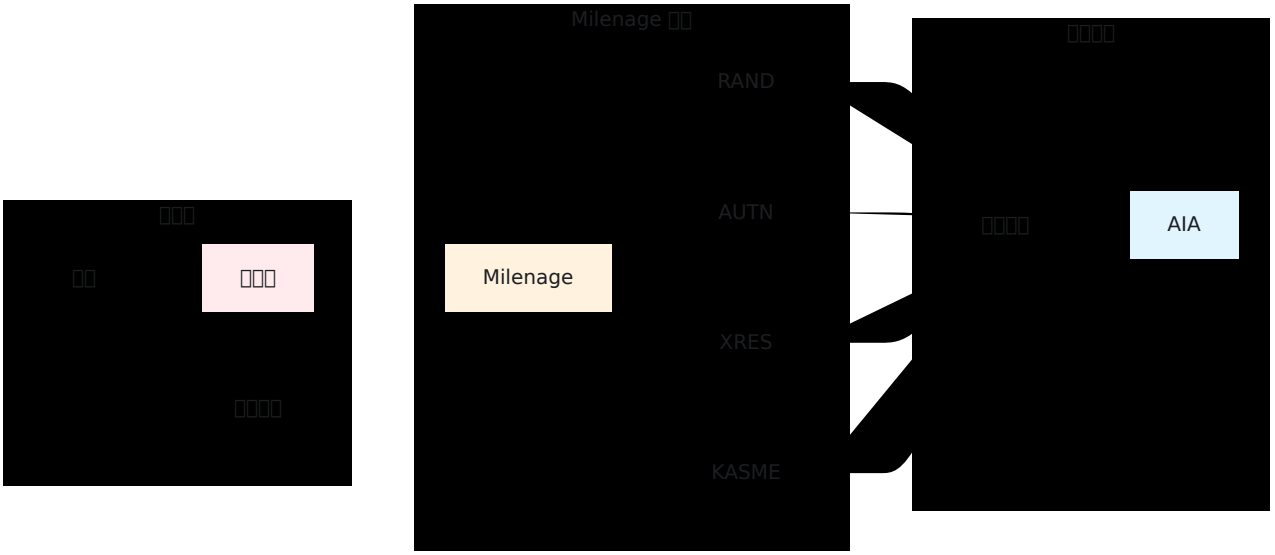
数据库源	字段	D
subscriber.enabled	`true/false`	Su St
msisdn.msisdn	`'14155551234'`	MS
epc_profile.ue_ambr_ul_kbps	`50000`	Ma Re Ba UL
epc_profile.ue_ambr_dl_kbps	`100000`	Ma Re Ba DL
epc_profile.network_access_mode	`'packet_only'`	Ne Ac Mo
apn_identifier.apn	`'internet'`	Se Se
apn_identifier.ip_version	`'ipv4v6'`	PD
apn_qos_profile.qci	`9`	Qo Id
apn_qos_profile.allocation_retention_priority	`8`	Pr Le

数据库源	字段	D
apn_qos_profile.pre_emption_capability	`false`	Pr en Ca
apn_qos_profile.pre_emption_vulnerability	`true`	Pr en Vu
apn_qos_profile.apn_ambr_ul_kbps	`25000`	AP UL
apn_qos_profile.apn_ambr_dl_kbps	`50000`	AP DL
static_ip.ipv4_static_ip	`'100.64.1.1'`	Se Pa Ac (IP
static_ip.ipv6_static_ip	`'2606:4700::1111'`	Se Pa Ac (IP

关键转换：

1. **AMBR** 带宽：数据库以 kbps 存储，Diameter 期望 bps（乘以 1000）
2. **IP** 版本编码：`0=IPv4, 1=IPv6, 2=IPv4v6, 3=IPv4_or_IPv6`
3. 用户状态：`enabled: true → 0 (SERVICE_GRANTED)`，`enabled: false → 1 (OPERATOR_DETERMINED_BARRING)`
4. 上下文标识符：每个 APN 在配置文件中的顺序编号（0, 1, 2...）
5. 静态 **IP**：仅在通过 `static_ips` 多对多关系分配时含

业务逻辑验证：



- 漫游检查：匹配访问的 PLMN 与 `roaming_profile.roaming_rules`
- 用户启用检查：`subscriber.enabled == true`
- 过滤 APN：如果漫游政策拒绝 IMS，可能会排除 IMS APN

认证信息应答 **(S6a AIA)**

认证信息应答为 LTE/EPC 用户提供认证向量。

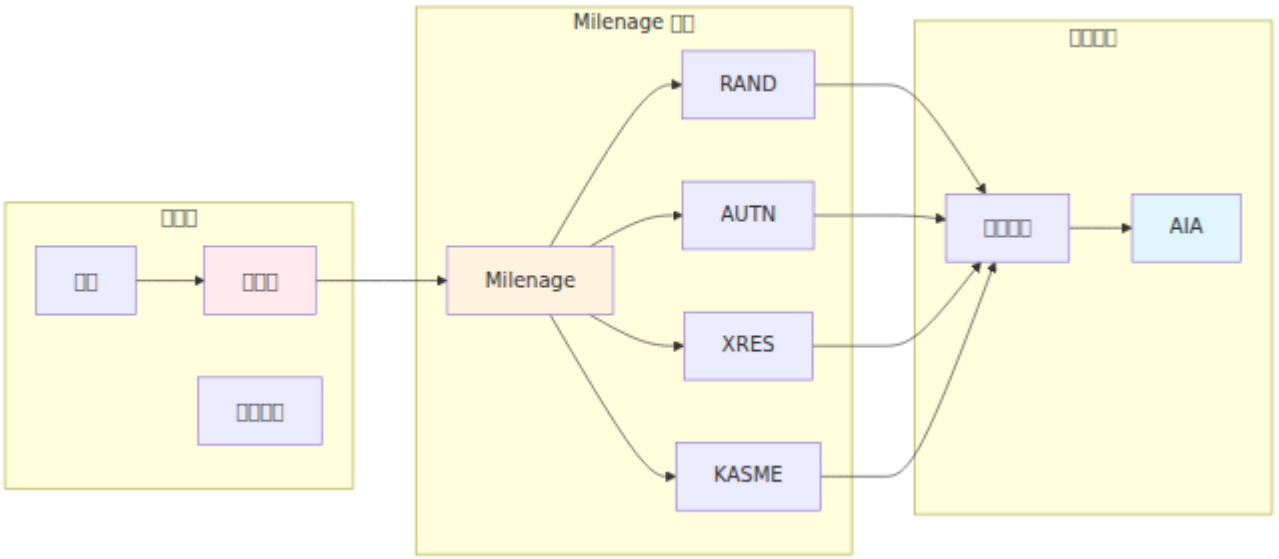
数据源映射

关键组件：

1. 加密密钥：所有密钥以十六进制字符串存储在 `key_set` 表中
2. **SQN** 管理：每次生成认证向量后递增序列号（防止重放攻击）
3. **Milenage** 算法：3GPP TS 35.206 - 生成认证向量
4. **KASME** 派生：根据 CK||IK 使用 KDF 按 TS 33.401 派生密钥

安全特性：

- 每个用户存储 SQN（非全局）
- Ki/OPc 永远不离开 HSS（仅传输派生值）
- AUTN 含序列号（SQN）和 AMF 进行网络认证
- Milenage 算法提供 UE 和网络之间的双向认证



服务器分配应答（Cx SAA）

服务器分配应答由 HSS 在 IMS 注册过程中发送给 S-CSCF。

数据源映射

关键特性：

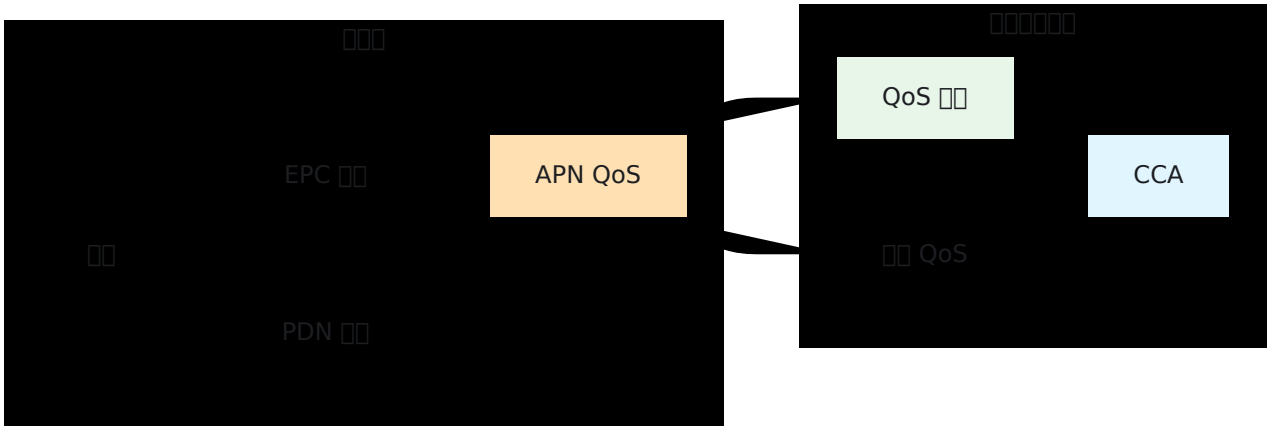
1. **IFC** 模板：XML ���板存储在 `ims_profile.ifc_template`
2. 动态替换：在运行时替换 `{{msisdn}}`、`{{imsi}}`、`{{impu}}`
3. **S-CSCF** 分配：在 `subscriber_state.assigned_scscf` 中存储分配的 S-CSCF
4. **IMS** 公共身份：格式：`sip:+{msisdn}@{ims_domain}` 或 `tel:+{msisdn}`

IFC 模板参数：

- `{{msisdn}}` - 用户的第一个 MSISDN
- `{{imsi}}` - 用户 IMSI
- `{{impu}}` - IMS 公共用户身份（来自 subscriber_state）
- `{{impi}}` - IMS 私有用户身份（通常为 IMSI@realm）

计费控制应答（Gx CCA）

计费控制应答由 PCRF 功能在承载建立期间发送给 PGW。



数据源映射

关键特性：

1. 会话跟踪：为每个承载创建/更新 `pdn_session` 记录
2. **QoS** 强制：提供来自 APN QoS 配置的 QCI 和带宽限制
3. 计费规则：返回用于计费集成的默认计费规则
4. **CC-Request-Type**：处理 INITIAL (1)、UPDATE (2)、TERMINATION (3)

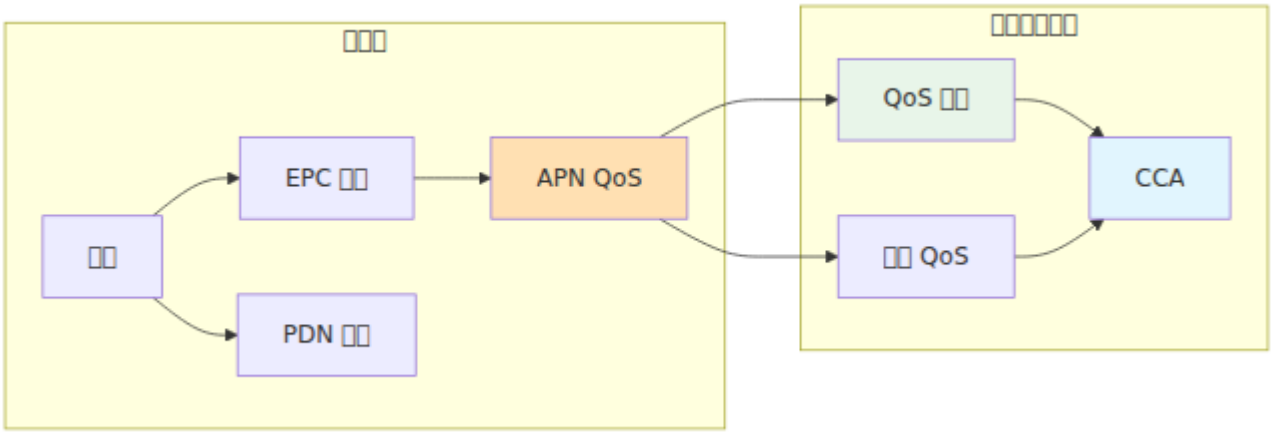
会话状态管理：

- `INITIAL_REQUEST`：创建新的 PDN 会话记录
- `UPDATE_REQUEST`：更新现有 PDN 会话
- `TERMINATION_REQUEST`：删除 PDN 会话记录

用户数据应答 (Sh UDA)

用户数据应答由 HSS 通过 Sh 接口发送给 AS（应用服务器）。

数据源映射



关键特性：

1. 存储库数据：可以在 `subscriber_state.sh_repository_data` 中存储自定义 XML
2. 服务指示：按请求的服务过滤数据（例如，存在、消息）
3. 公共身份：返回用户的所有 IMS 公共身份
4. 引用与透明：支持引���和透明数据模式

ME 身份检查应答 (S13 ECA)

ME 身份检查应答由 EIR 功能发送给 MME 进行 IMEI 验证。

数据源映射

关键特性：

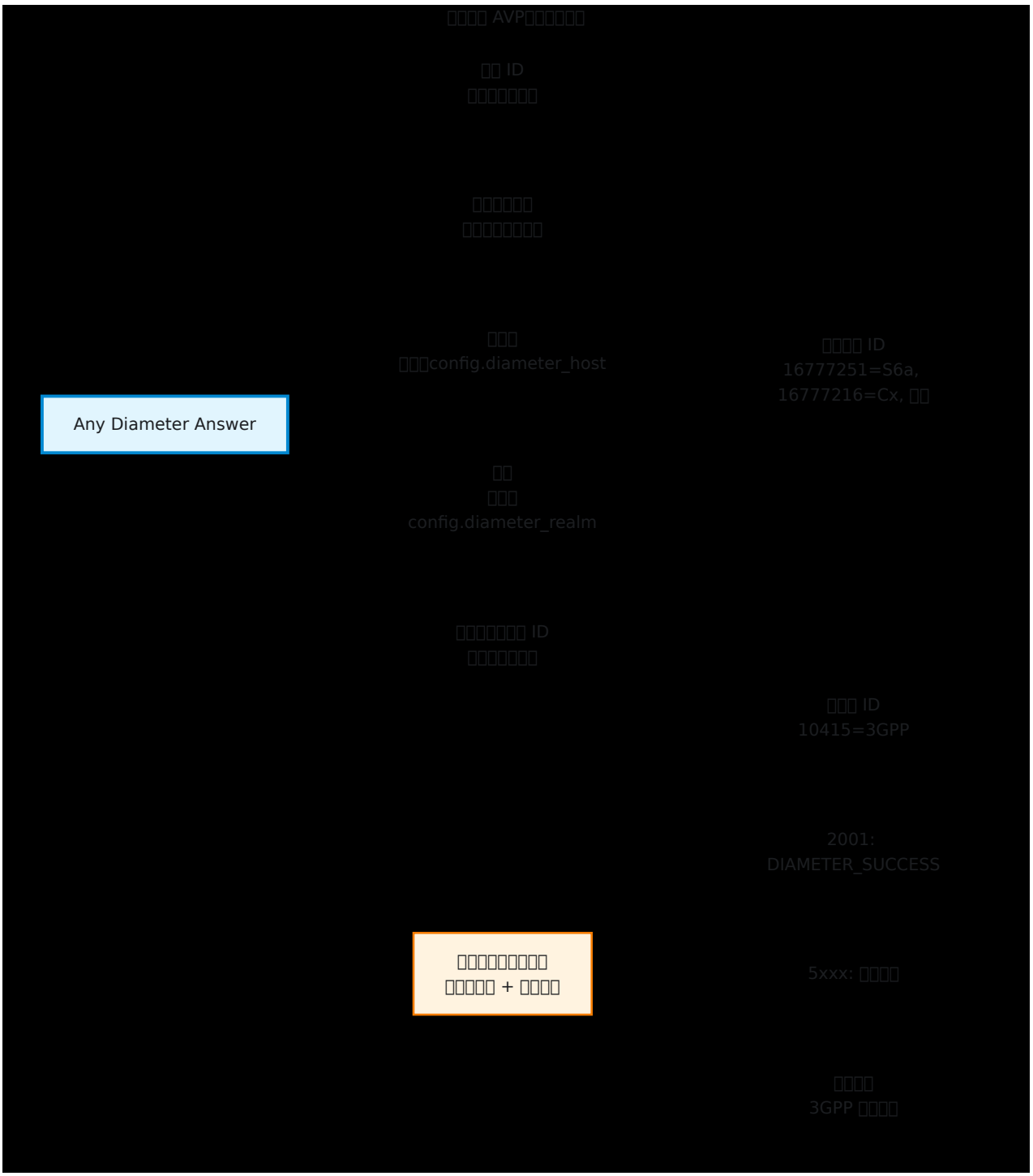
1. **IMEI** 正则匹配：规则使用正则表达式进行灵活匹配
2. 基于 **TAC** 的规则：可以匹配类型分配代码（前 8 位数字）
3. 默认行为：可配置未知 IMEI（接受或拒绝）
4. 设备状态值：
   - `0` = 白名单（明确允许）
   - `1` = 黑名单（被盗/阻止）
   - `2` = 灰名单（允许但监控）
   - `5` = 未知（没有匹配规则）

用例：

通过精确 IMEI 阻止被盗设备
通过 TAC 模式阻止设备型号
仅允许白名单中的设备
跟踪灰色市场设备

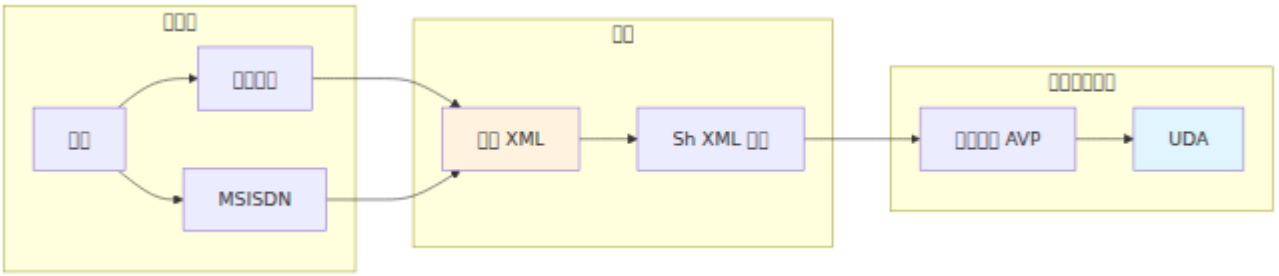
共同响应元素

所有 Diameter 响应共享这些共同的 AVP：



□□□□□

```
config :diameter_ex,
  diameter_host: "hss",
  diameter_realm: "example.com",
  diameter_service_name: "OmniHSS"
```



数据流总结

请求处理管道

实施说明

协议处理程序

系统实现了以下 Diameter 协议的处理程序:

- **S6a** - LTE/MME 接口用于认证和位置更新
- **Cx** - IMS/CSCF 接口用于 IMS 注册和服务器分配
- **Sh** - IMS/AS 接口用于用户数据检索
- **Gx** - PCRF 接口用于策略和计费控制
- **Rx** - IMS/AF 接口用于媒体授权
- **S13** - EIR 接口用于 IMEI 验证
- **SWx** - WiFi/IMS 接口用于 3GPP 访问认证

数据模型

数据库架构括以下核心实体:

- 用户 - 核心用户记录，含 IMSI
- 密钥集 - 用于认证的加密密钥
- **EPC** 配置 - LTE 服务配置
- **APN** 配置 - 接入点配置
- **IMS** 配置 - 含 IFC 模板的 IMS 服务配置

漫游配置 - 漫游规则和限制
用户状态 - 动态会话和状态跟踪
PDN 会话 - 活动承载会话跟踪
静态 IP - 静态 IP 地址分配
EIR 规则 - IMEI 验证规则

← 返回操作指南 | API 参考 → | 协议流程 →

OmniHSS 监控与监测指南
← 返回操作指南

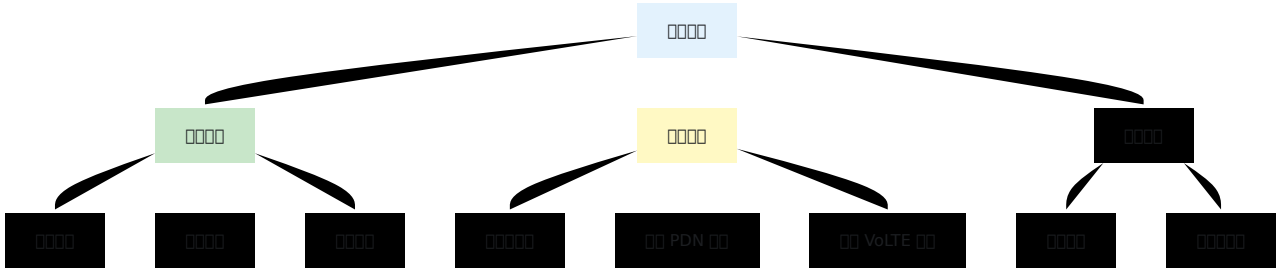
目录

监控概述
控制面板监控
数据库监控
日志监控
外部监控集成
关键绩效指标
警报策略

监控概述

OmniHSS 提供多种机制来监控系统健康、性能和用户活动。操作人员应结合使用这些工具以获得全面的可见性。

监控层



控制面板监控

控制面板提供主要的实时监控界面。

概览页面监控

URL: `https://[hostname]:7443/overview`

可用的关键指标

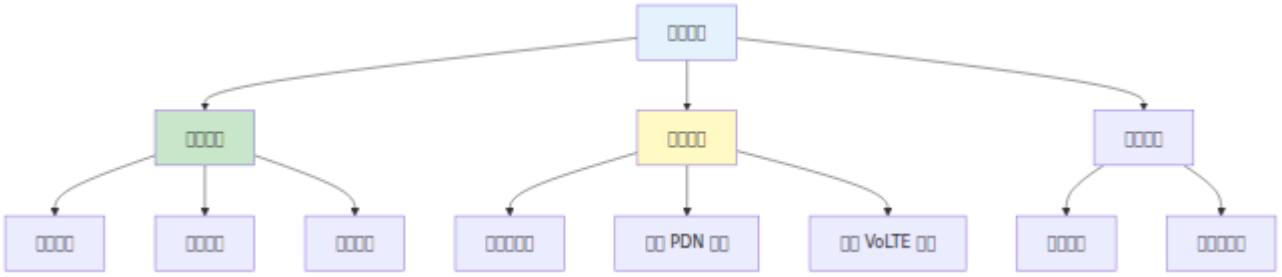
监控的用户状态

状态	指标	含义
空闲	无位置信息	用户已关闭或超出覆盖范围
已连接	MME 存在	用户已注册到网络
PDN 活动	PDN 会话计数 > 0	活动数据连接
IMS 注册	S-CSCF 分配	语音服务已准备好
通话中	活动通话计数 > 0	VoLTE 通话正在进行

从概览中提取指标

控制面板不直接导出指标。您可以：

1. 计算可见行以获取总用户数
2. 扫描绿色勾号以计算启用的用户
3. 查看扩展详细信息以获取状态信息



4. 记录最后看到的时间戳以评估响应性

Diameter 页面监控

URL: `https://[hostname]:7443/diameter`

关键指标

关键对等体监控

识别关键对等体并监控其状态：

对等体类型	关键性	断开时的影响
MME	高	无法进行新的 LTE 附加
P-GW	高	无法建立数据会话
S-CSCF	高	无法进行 IMS 注册
P-CSCF	高	无法进行 VoLTE 通
I-CSCF	中	IMS 路由问题
AS	低-中	特定服务不可用

应用页面监控

URL: `https://[hostname]:7443/application`

关键指标

指标	描述	正常范围	行动阈值
进程计数	活动 Erlang 进程	根据负载变化	> 90% 的限制
内存使用	总内存消耗	< 80%	> 90%
运行时间	自上次重启以来的时间	N/A	跟踪稳定性

数据库监控

直接数据库查询

连接到 SQL 数据库以提取详细指标：

用户计数

查询数据库以检索：

- 所有用户的总计数
- 启用用户的计数
- IMS 启用用户的计数

会话统计

查询数据库以检索：

- 活动 PDN 会话的计数
- 活动 VoLTE 通话的计数
- 按 APN 配置文件划分的 PDN 会话

位置统计

查询数据库以检索：

- 按访问网络（MCC-MNC 组合）分组的用户计数
- 当前漫游的用户计数（不在本 PLMN 001-001 上）

- 用户在不同访问网络中的分布

最近活动

查询数据库以检索：

- 在过去一小时内看到的用户计数
- 按服务 MME 划分的用户分布
- 最近用户活动的时间戳分析

数据库健康监控

通过查询监控��据库健康：

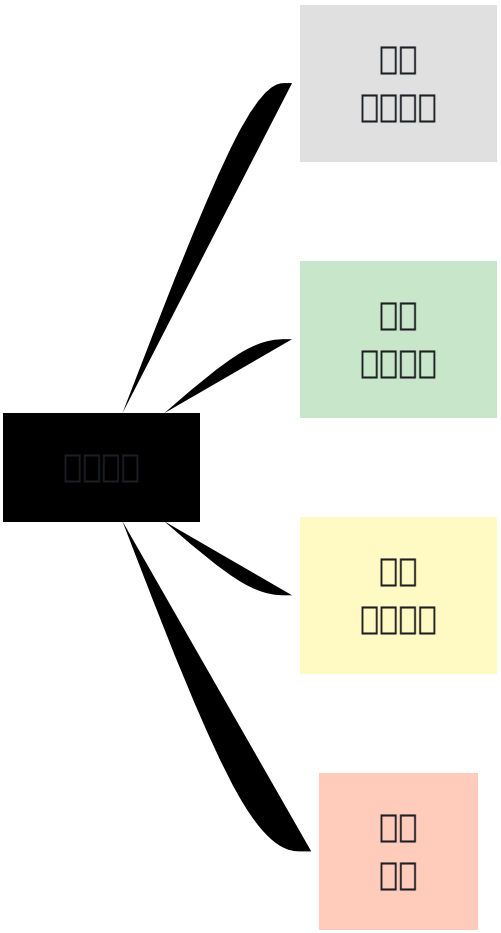
- 数据库的总大小和增长趋势
- 各个表的大小和行计数
- 当前数据库连接计数
- 查询性能和资源使用

日志监控

日志输出

OmniHSS 将日志输出到 **stdout/stderr**，应由您的进程管理器捕获。

日志级别



□□□□□□□□□

Diameter □□□□□:

```
[info] Diameter peer connected: mme01.epc.example.com
[warn] Diameter peer disconnected: pgw01.epc.example.com
[error] Diameter peer connection failed: timeout
```

□□□□□:

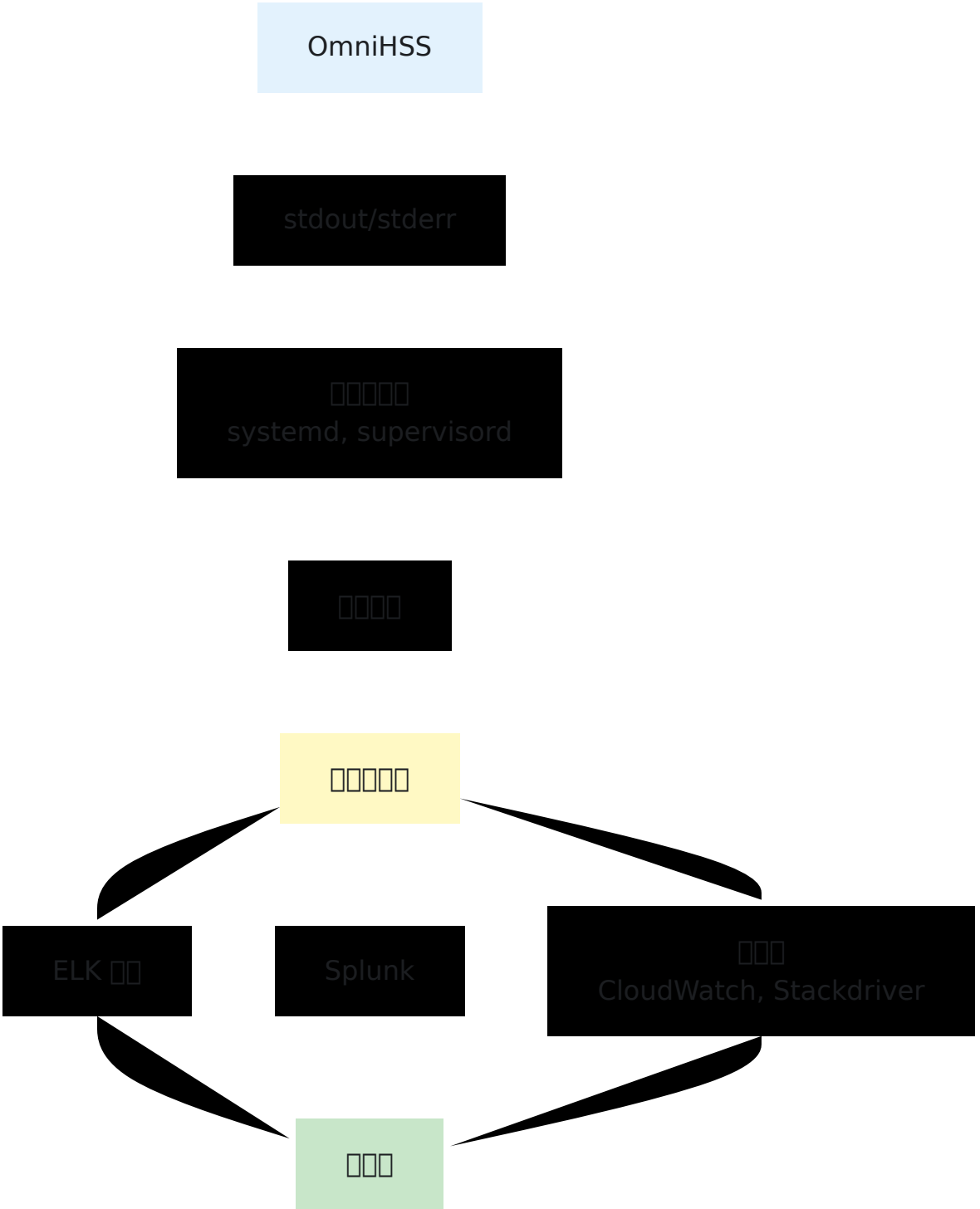
```
[info] Database connection established
[error] Database connection lost: timeout
[error] Database query failed: deadlock detected
```

□□□□:

```
[info] Authentication successful: IMSI 001001123456789
[warn] Authentication failed: IMSI 001001123456789, invalid vector
[error] Roaming denied: IMSI 001001123456789, MCC 310 MNC 410
```

□□□□

□□□□□□□□□□□□□□□□



外部监控集成

健康检查端点

API 健康检查**:** `GET /api/status`

```
curl -k https://hss.example.com:8443/api/status
```

预期响应**:**

```
{"status": "ok"}
```

HTTP 状态**:** 200 OK

监控工具集成

Nagios/Icinga 示例

```
#!/bin/bash
# check_omnihss.sh

API_URL="https://hss.example.com:8443/api/status"

response=$(curl -k -s -o /dev/null -w "%{http_code}" "$API_URL" --
max-time 5)

if [ "$response" = "200" ]; then
    echo "OK - OmniHSS API responding"
    exit 0
else
    echo "CRITICAL - OmniHSS API not responding (HTTP $response)"
    exit 2
fi
```

Prometheus 集成

OmniHSS 暴露了原生的 Prometheus 指标端点。抓取 `http://<host>:9568/metrics`。端口由 `prometheus_metrics_port` 设置，默认值为 `9568`。

该端点包括 BEAM 运行时指标、数据库池指标、Diameter 对等体状态、每个应用的 Diameter 消息计数器，以及下面的 Cx 计数器。

S-CSCF 认证放弃计数器

指标：`hss_cx_auth_abandon_total` 类型：计数器 描述：计算结束 IMS-AKA 挑战但未成功注册的 Cx SAR。这是一个被放弃或失败的挑战。当其 `auth_vector_timeout` 在被放弃的挑战上过期时，S-CSCF 发送 `AUTHENTICATION_TIMEOUT` SAR，或者在响应错误时发送 `AUTHENTICATION_FAILURE` SAR。请参见 S-CSCF 分配状态。标签：

- `sar_type` - 服务器分配类型：`authentication_timeout` 或 `authentication_failure`。
- `outcome` - 当 S-CSCF 名称被释放时为 `name_cleared`，或者当保留了已注册绑定时为 `kept_registered`。当用户未注册时，HSS 会释放名称，并且当 SAR 来自与记录中不同的 S-CSCF 时也会释放。第二种情况意味着 I-CSCF 重新分叉了 REGISTER，因为它无法到达记录中的节点，因此该名称是过时的。
- `origin_host` - 发送 SAR 的 S-CSCF。

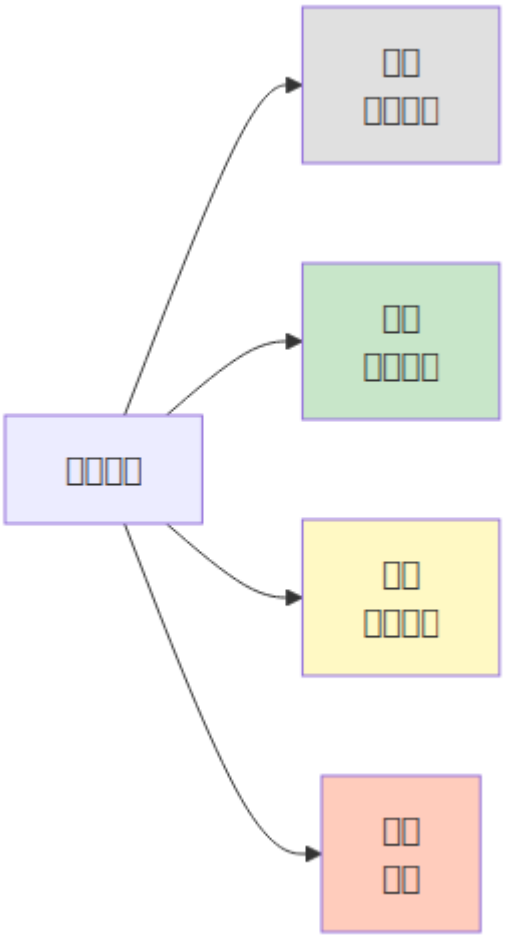
示例查询：

```
# 每秒被放弃/失败的挑战率
rate(hss_cx_auth_abandon_total[5m])

# 按原因拆分
sum by (sar_type) (rate(hss_cx_auth_abandon_total[5m]))

# 按被放弃挑战的 S-CSCF 排名
topk(5, sum by (origin_host)
(rate(hss_cx_auth_abandon_total[5m])))
```

一个 `origin_host` 上的高比率可能表明 S-CSCF 正在失败挑战，或者用户在该节点上放弃注册。

SNMP 集成

对于基于 SNMP 的监控，可以使用自定义 SNMP 扩展脚本查询数据库或 API 获取指标，并通过 SNMP OID 返回值。



关键绩效指标

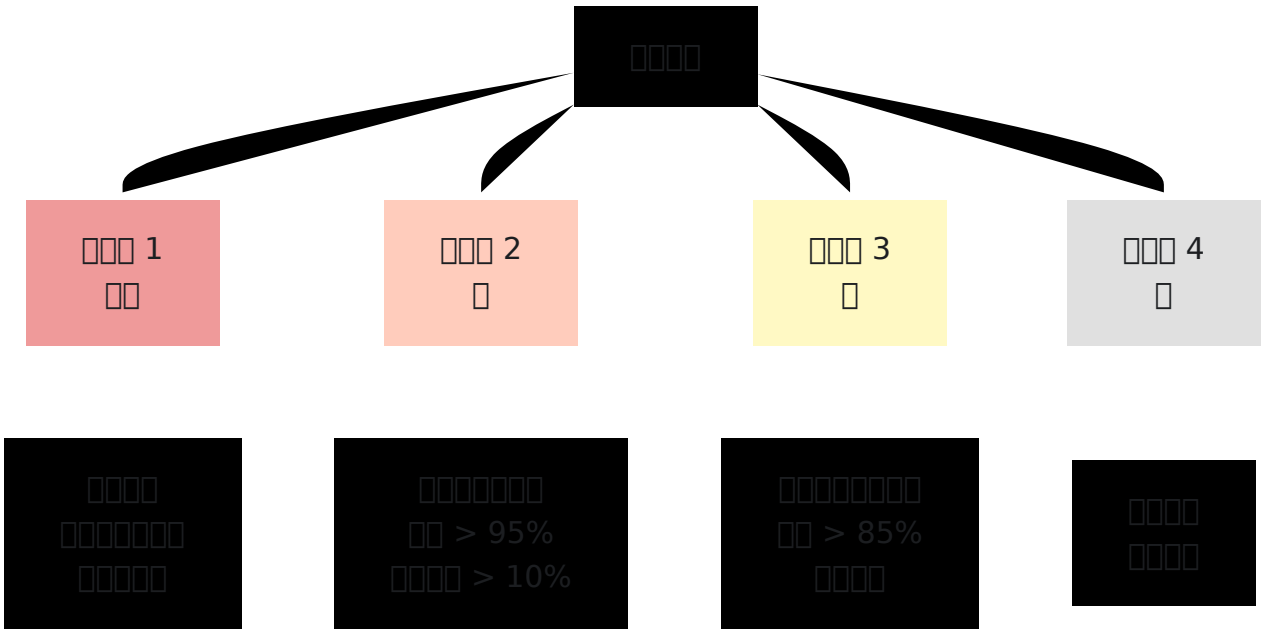
操作 **KPI**

推荐的 **KPI** 阈值

KPI	目标	警告	危急
系统正常运行时间	99.99%	< 99.95%	< 99.9%
Diameter 对等体正常运行时间	99.9%	< 99.5%	< 99%
认证成功率	> 99%	< 99%	< 95%
Diameter 响应时间	< 100ms	> 200ms	> 500ms
数据库查询时间	< 50ms	> 100ms	> 500ms
错误率	< 0.1%	> 0.5%	> 1%

容量 **KPI**

指标	监控	计划行动
总用户数	当前计数	预期容量的 80%
并发 **PDN** 会话	活动会话	预期最大值的 70%
数据库大小	使用的 MB	分配存储的 80%
数据库连接	活动连接	池大小的 80%



警报策略

警报优先级

警报定义

关键警报 **(P1)**

系统不可用：

- API 健康检查失败
- 控制面板无法访问
- 数据库连接失败
- 行动：立即调查和升级

所有 **Diameter** 对等体断开：

- 无连接对等体
- 行动：检查网络，如有必要重启

数据库宕机：

- 无法连接到 SQL 数��库

行动: 调查数据库服务器，如有必要重启

高优先级警报 (P2)

关键 Diameter 对等体宕机:

主 MME 断开
主 P-GW 断开
主 S-CSCF 断开
行动: 在 15 分钟内调查对等体连接

高内存使用:

内存 > 95%
行动: 调查内存泄漏，计划重启

高认证失败率:

10% 的认证请求失败

行动: 检查用户配置，调查原因

中优先级警报 (P3)

非关键对等体宕机:

次要对等体断开
应用服务器断开
行动: 在 1 小时内调查

内存使用升高:

内存 > 85%
行动: 监控趋势，计划容量升级

错误率升高:

错误率 > 1%
行动: 审查日志，确定根本原因

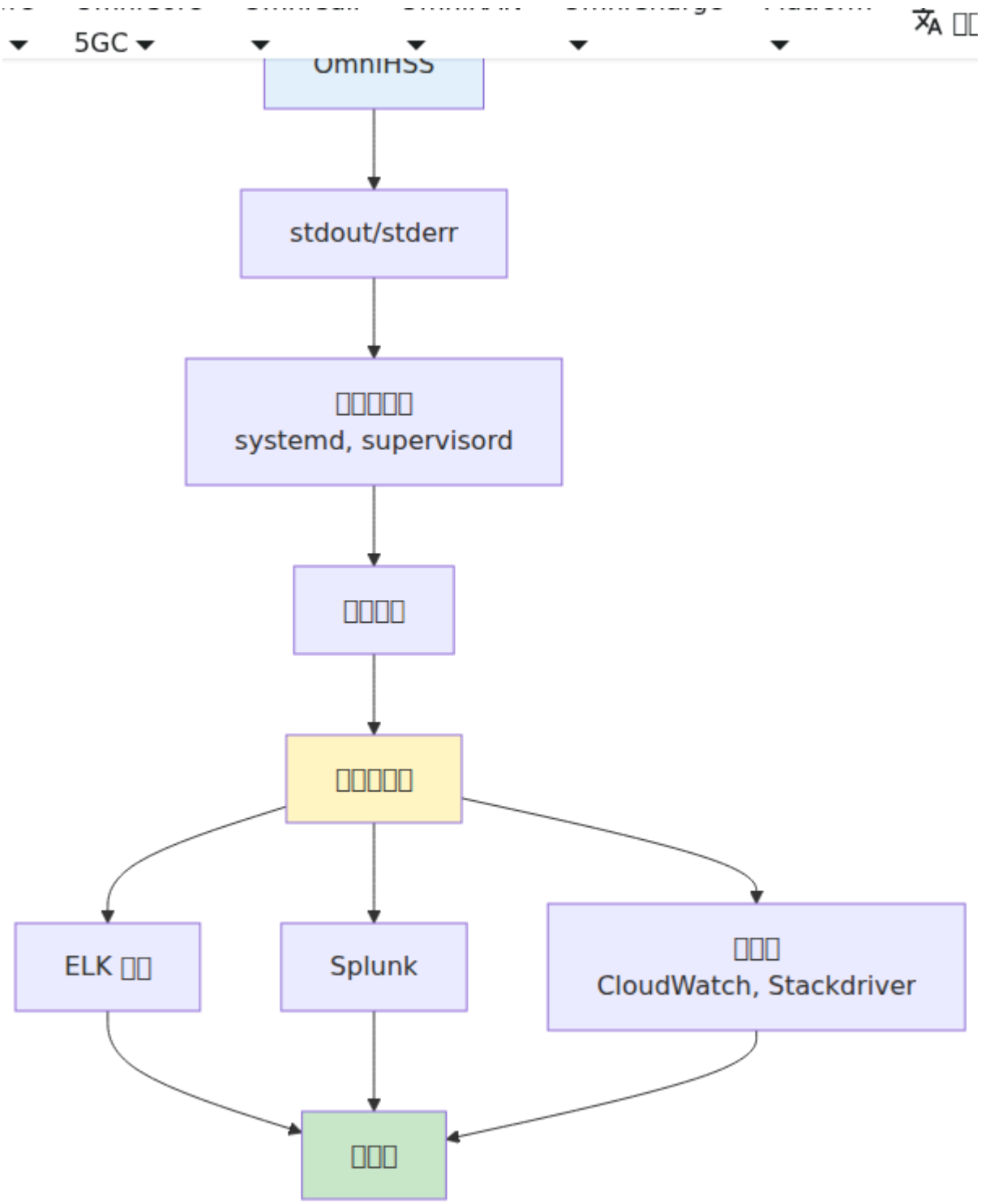
低优先级警报 (P4)

容量警告:

用户 > 80% 的容量
数据库 > 80% 的分配存储
行动: 计划容量扩展

性能下降:

响应时间升高但可接受
行动: 监控并优化查询



警报通知渠道

5GC
OmniHSS
stdout/stderr
systemd, supervisord
ELK
Splunk
CloudWatch, Stackdriver

监控检查清单

每日检查

审查控制面板概览 - 用户计数正常
审查 Diameter 页面 - 所有关键对等体已连接审查应用页面 - 内存和进程在限制内
检查错误日志 - 在过去 24 小时内没有关键错误
验证备份成功完成

每周检查

审查容量趋势 - 用户增长
审查性能趋势 - 响应时间
审查数据库大小 - 增长率可接受
审查错误率 - 确定模式
测试警报通知 - 确保正常工作

每月检查

容量规划审查 - 项目提前 6 个月
性能优化审查 - 确定慢查询
安全审查 - 证书过期、漏洞
文档审查 - 更新运行手册
灾难恢复测试 - 验证备份能否正确恢复

← 返回操作指南 | 下一步: 多功能 →

OmniHSS 多 MSISDN 和多 IMSI 功能

← 返回操作指南

目录

- 概述
- 多 MSISDN：多个电话号码
- 多 IMSI SIM：多个网络身份
- 组合场景
- 跨接口 MSISDN 解析
- 共享公共用户身份
- 配置示例
- 操作程序

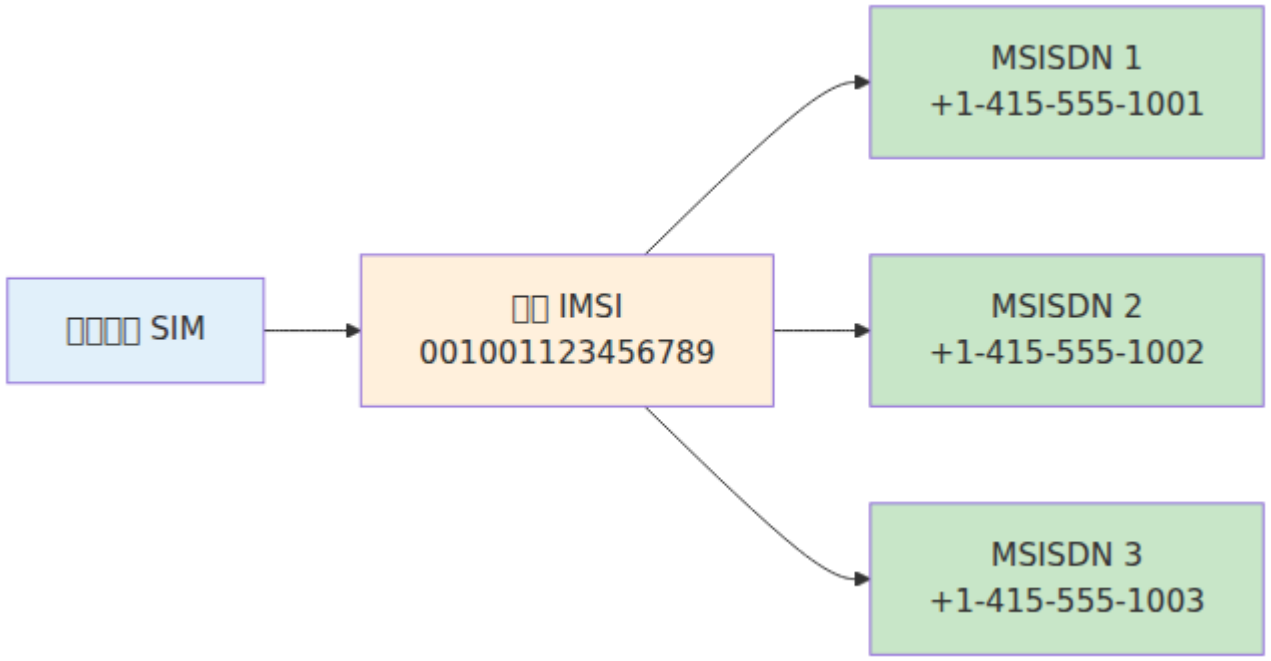
概述

OmniHSS 支持先进的配置能力，能够实现灵活的服务配置：

多 **MSISDN** 支持

一个 **IMSI** → 多个电话号码

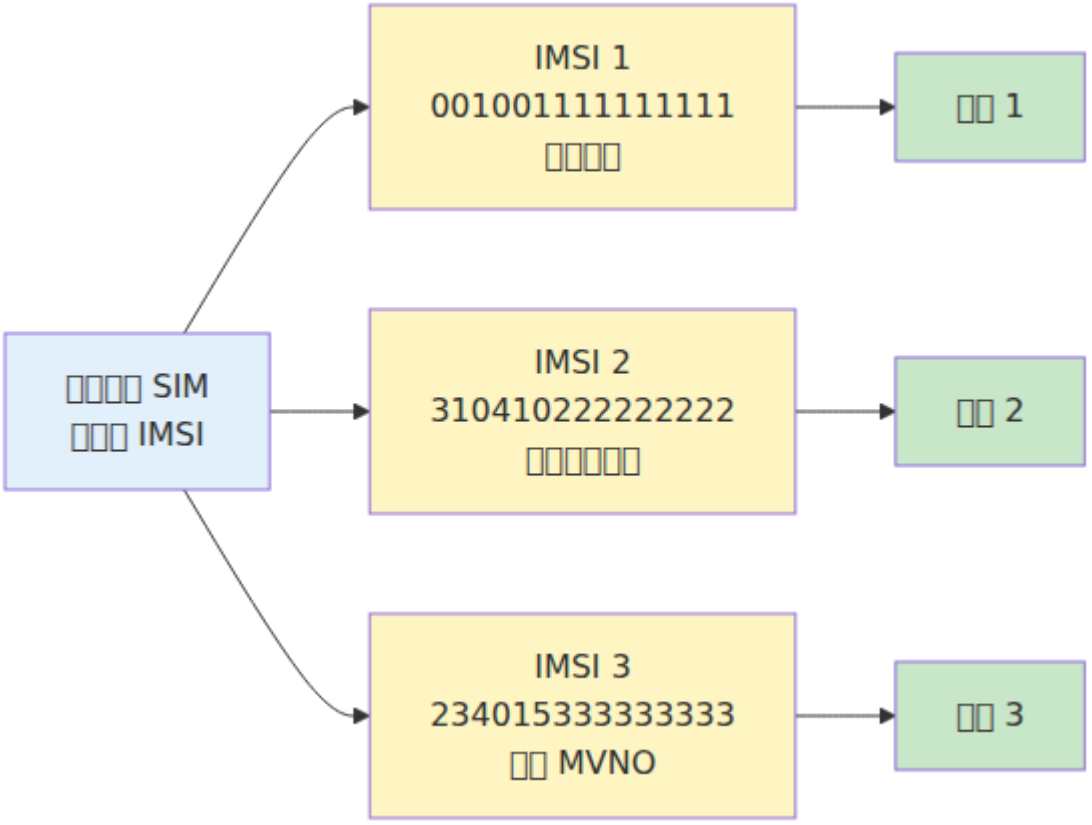
单个用户（通过一个 IMSI 识别）可以分配多个 MSISDN（电话号码）。所有号码在同一设备上响铃，并共享相同的服务配置文件。



□ IMSI SIM □□

□□ SIM → □□ IMSI

□□□□ SIM □□□□□□□□ IMSI□□□□□□□



多 **MSISDN**：多个电话号码

工作原理

HSS 中的一个用户记录通过连接表链接多个 MSISDN。当用户注册到 IMS 时，所有 MSISDN 都□含在 IMS 配置文件中，允许任何号码的入站呼叫到达设备。

主要特征

- 一个 **IMSI** - 用户有一个与其 SIM 卡关联的 IMSI
- 多个 **MSISDN** - 用户可以拥有多��电话号码
- **IMS** 集成 - 所有 MSISDN 都在 IMS 中注册
- 共享服务 - 所有号码共享相同的服务配置文件（EPC、IMS、漫游）

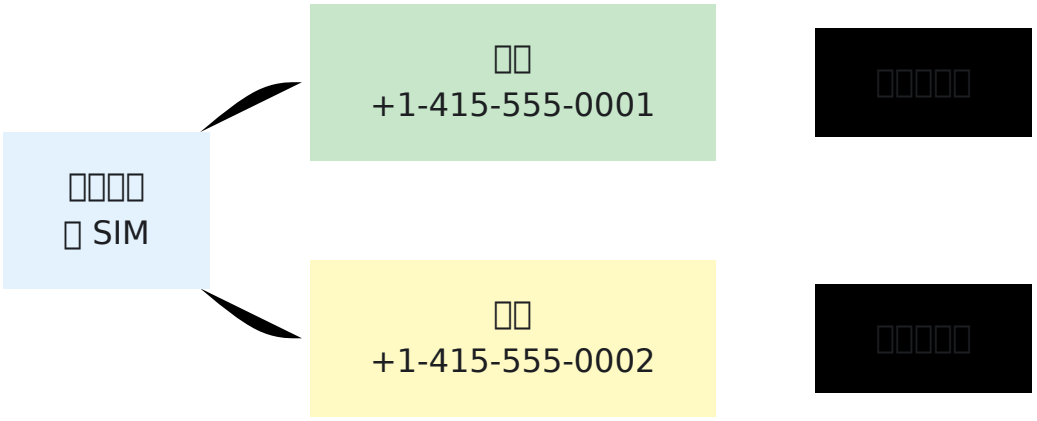
数据模型

重要：一个 MSISDN 只能同时分配给一个用户。然而，一个用户可以拥有多个 MSISDN。

□□

1. □□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



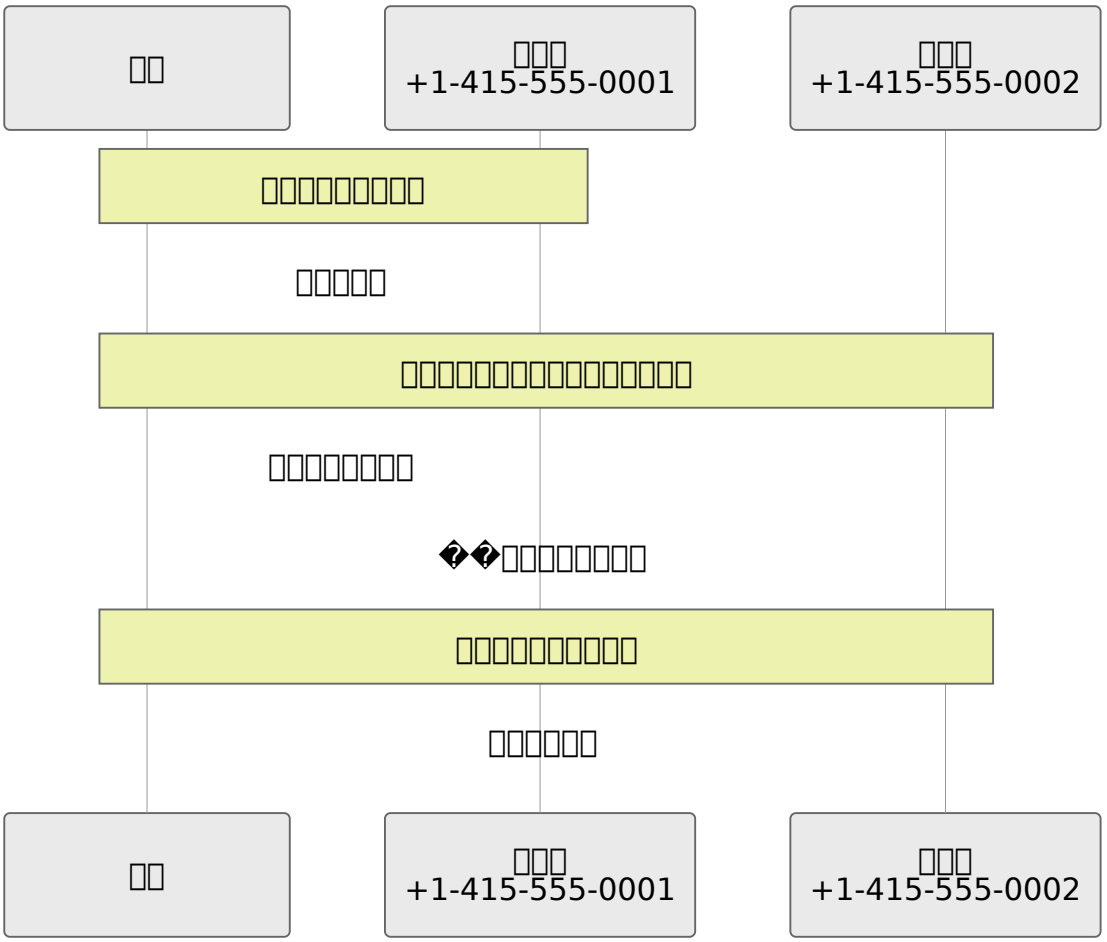
2. □□□□

□□□□□□□□□□□□□□□□□□□

□□□ □ OmniHSS □□□□□□□□□□□□□□□ SIM/IMSI □□□□□□□□□□□□□□□□ MSISDN□

4. □□□□□□□

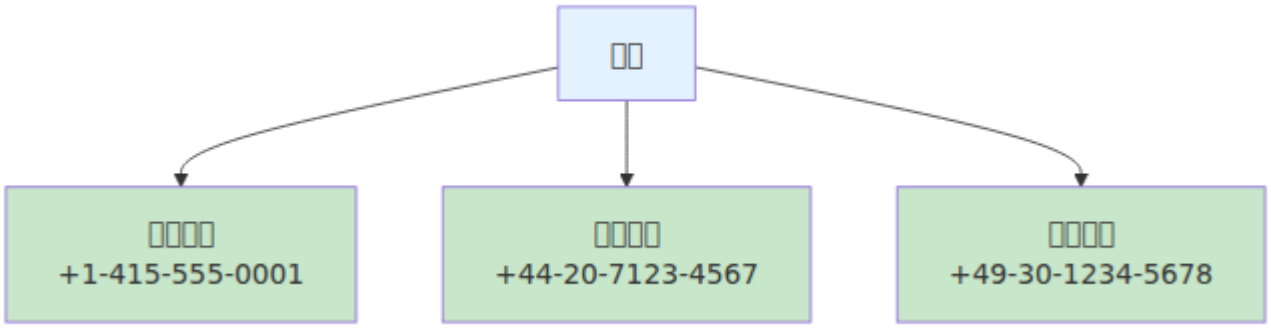
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



□□

□□ MSISDN

□□□□□□□□□□□□□□□ MSISDN□



```
# 创建第一个 MSISDN
curl -k -X POST https://hss.example.com:8443/api/msisdn \
  -H "Content-Type: application/json" \
  -d '{"msisdn": {"msisdn": "14155551001"}}'

# 创建第二个 MSISDN
curl -k -X POST https://hss.example.com:8443/api/msisdn \
  -H "Content-Type: application/json" \
  -d '{"msisdn": {"msisdn": "14155551002"}}'
```

将 **MSISDN** 分配给用户

分配通过数据库中的连接表完成。

数据库方法：

1. 查询数据库以获取目标 IMSI 的用户 ID
2. 查询数据库以获取电话号码的 MSISDN ID
3. 将记录插入连接表，将 subscriber_id 链接到每个 msisdn_id

这创建了用户与其电话号码之间的多对多关系。

配置工作流

验证分配

查询数据库以检索用户及其所有链接的 MSISDN：

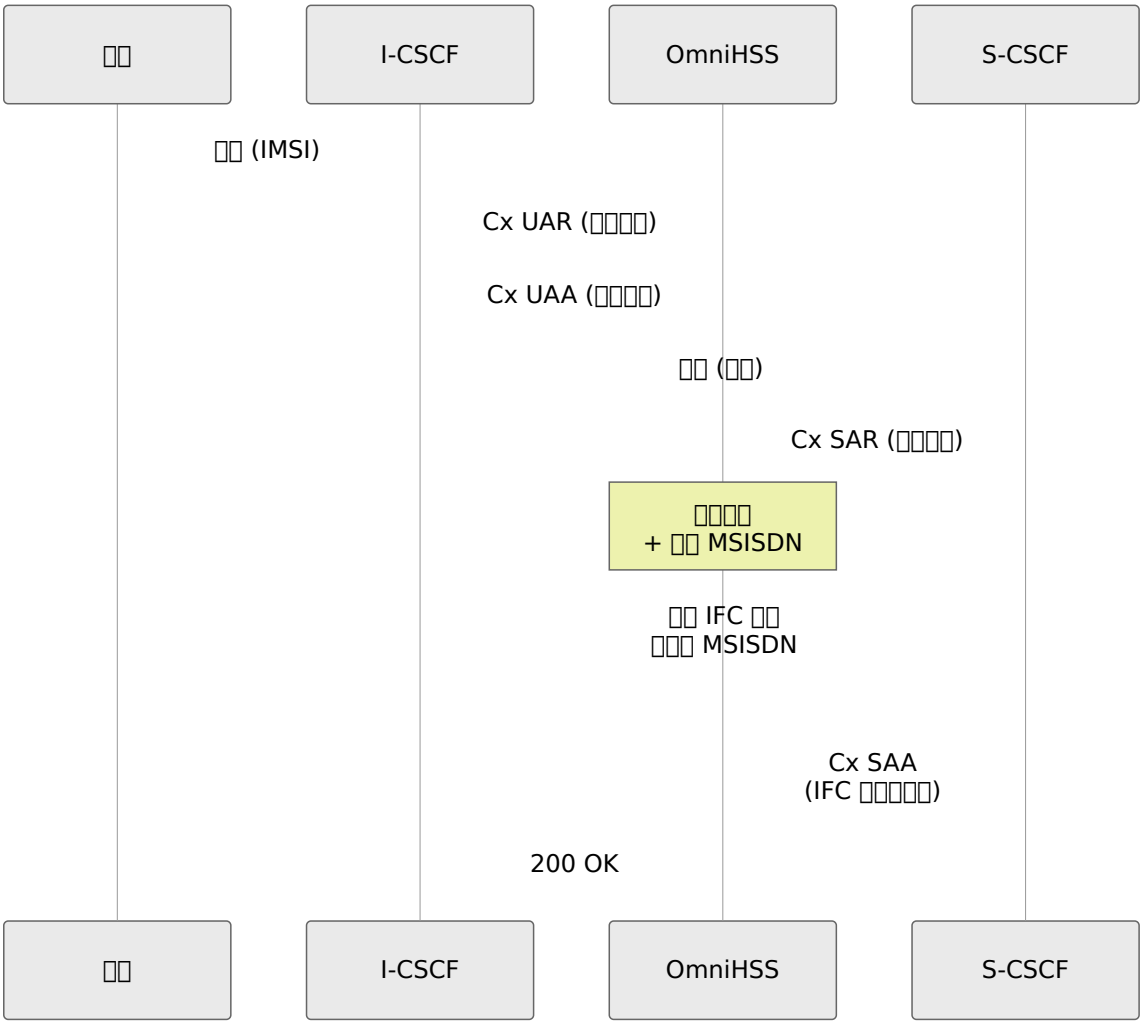
- 将用户表与连接表连接
- 将连接表与 msisdn 表连接
- 按用户分组结果以查看所有电话号码

□□□□□□□ ID□IMSI □□□□□□□ MSISDN □□□

IMS □□

IMS □□

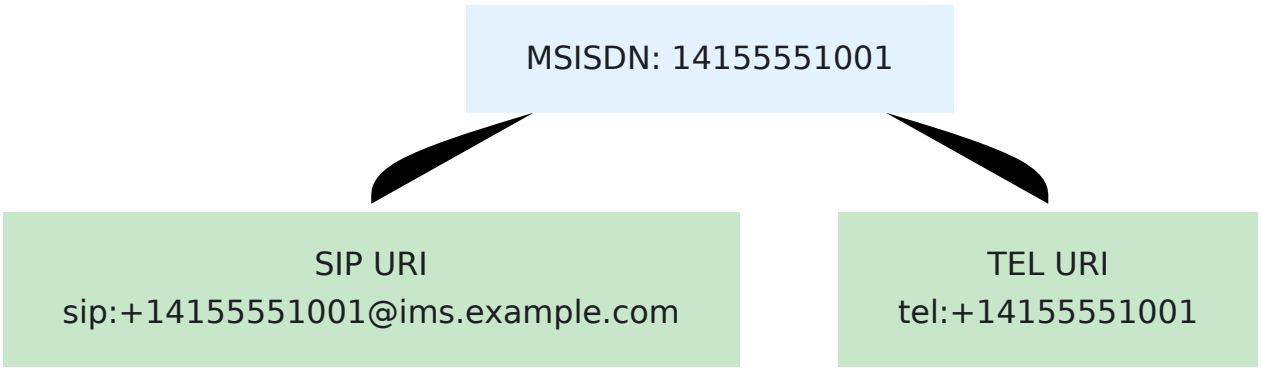
□□□□□□□ IMS □□□□□□□□ **MSISDN** □□□□□□□□ S-CSCF □ IMS □□□□□□□



IFC □□□□

IMS IFC □□□□□□□ `{{msisdns}}` □□□□□□□ MSISDN□

□□ **IFC □□□**



```
<ServiceProfile>
  <PublicIdentity>
    <Identity>sip:
{{imsi}}@ims.mnc{{mnc}}.mcc{{mcc}}.3gppnetwork.org</Identity>
  </PublicIdentity>
  <!-- 为每个 MSISDN 重复 -->
  <PublicIdentity>
    <Identity>sip:+14155551001@ims.example.com</Identity>
  </PublicIdentity>
  <PublicIdentity>
    <Identity>tel:+14155551001</Identity>
  </PublicIdentity>
  <PublicIdentity>
    <Identity>sip:+14155551002@ims.example.com</Identity>
  </PublicIdentity>
  <PublicIdentity>
    <Identity>tel:+14155551002</Identity>
  </PublicIdentity>
  <!-- ... -->
</ServiceProfile>
```

模板变量：

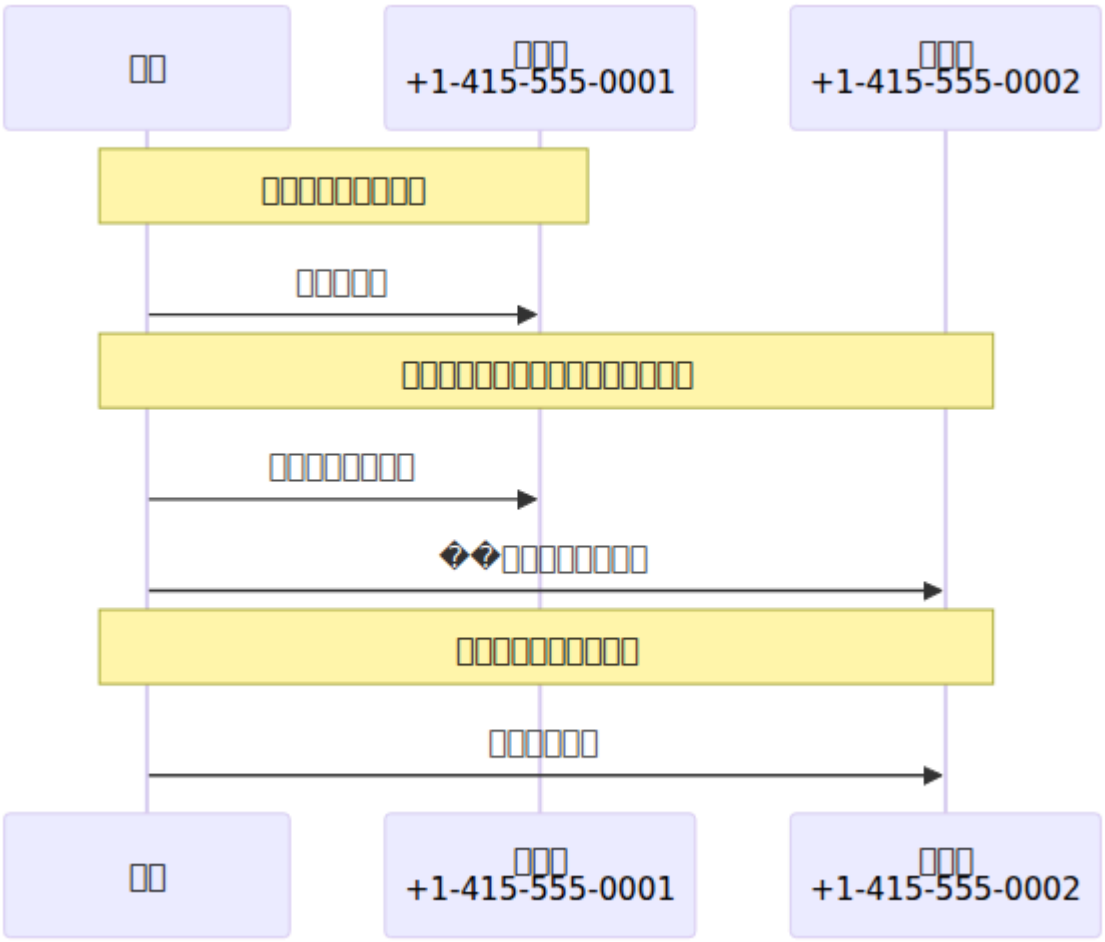
- `{{msisdns}}` - 分配给用户的所有 MSISDN 列表

公共身份

每个 MSISDN 通常会产生两个 IMS 公共身份：

入站呼叫路由

当有人拨打用户的号码时，IMS 网络将路由到正确的 SIP URI：



出站呼叫呈现

手机可以选择在出站呼叫中呈现哪个号码作为来电显示。

SIP INVITE 示例:

```
INVITE sip:+15105551234@ims.example.com SIP/2.0
From: "+14155551002" <sip:+14155551002@ims.example.com>;tag=123
To: <sip:+15105551234@ims.example.com>
P-Asserted-Identity: <sip:+14155551002@ims.example.com>
```

`From` 和 `P-Asserted-Identity` 头部指示正在使用用户的哪个号码。

故障排除多 **MSISDN**

问题: **MSISDN** 在 **IMS** 注册中未出现

症状:

- S-CSCF 仅显示一个公共身份
- 拨打第二个号码失败

故障排除步骤：

1. 验证数据库中的 **MSISDN** 分配：
    - 查询数据库以检索与用户的 IMSI 关联的所有 MSISDN
    - 检查连接表以确保关系存在
2. 检查 **IMS** 配置文件模板：
    - 验证模板是否包含 `{{msisdns}}` 变量
    - 确认模板语法是有效的 XML
3. 查看 **HSS** 日志：
    - 查找 IMS 注册（Cx SAR）消息
    - 验证所有 MSISDN 是否包含在响应中
4. 测试 **IMS** 注册：
    - 在手机上触发重新注册
    - 检查 S-CSCF 日志以查看注册的公共身份

问题：无法将 **MSISDN** 分配给用户

症状：

- 数据库插入失败
- 错误：“重复条目”或“外键约束”

可能原因：

1. **MSISDN** 已分配：
    - 查询数据库以检查 MSISDN 是否已链接到另一个用户
    - 解决方案：首先移除现有分配，然后创建新分配
2. **MSISDN** 不存在：

- 查询数据库以验证 MSISDN 记录是否存在
- 解决方案：首先通过 API 或数据库插入创建 MSISDN 记录

问题：一个号码可以拨打，另一个号码无法拨打

症状：

- 拨打主号码有效
- 拨打辅助号码失败或路由不正确

故障排除步骤：

1. 验证 **IMS** 注册中的两个号码：
    - 检查 S-CSCF 注册的公共身份
    - 确认两个 SIP URI 存在
2. 检查 **IMS** 路由规则：
    - 验证 IFC 模板路由规则适用于所有身份
    - 检查是否特定号码需要特殊路由
3. 测试两个号码：

```
# 从 SIP 客户端测试
sip:+14155551001@ims.example.com  # 应该有效
sip:+14155551002@ims.example.com  # 也应该有效
```

问题：**API** 按 **MSISDN** 查询返回错误的用户

症状：

- API 查询 `/api/subscriber/msisdn/:msisdn` 返回意外的用户

验证：

查询数据库以查找 MSISDN 分配给哪个用户。这应该返回确切的一个用户。如果返���多个或错误的用户，则连接表具有需要更正的不正确数据。

最佳实践

配置顺序

1. 首先创建所有 MSISDN
2. 创建用户
3. 将 MSISDN 分配给用户
4. 在激活之前验证分配

MSISDN 管理

在用户的 custom_attributes 中记录主要与次要号码
在迁移时按顺序迁移号码以避免服务中断
在交付给客户之前测试所有号码

IMS 配置

确保 IFC 模板正确处理多个公共身份
测试所有号码的入站路由
验证出站呼叫的来电显示

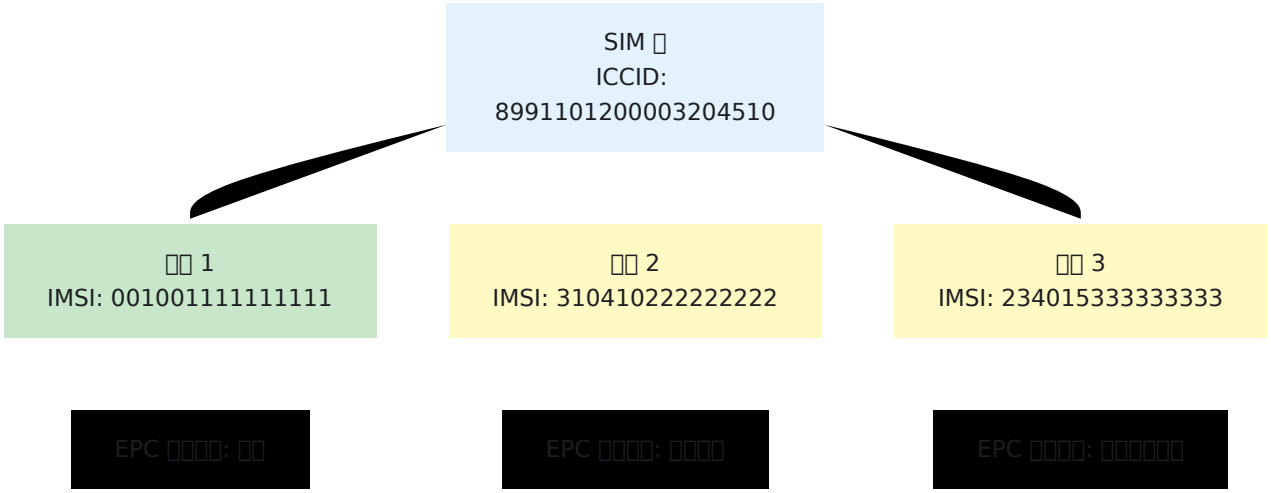
迁移

从单一到多 MSISDN 迁移时：

多 **IMSI SIM**：多个网络身份

工作原理

多 IMSI SIM 包含多个完整的用户配置文件，每个配置文件都有自己的 IMSI、密钥和凭证。设备可以在 IMSI 之间切换，以连接到不同的网络，通常基于位置或网络可用性自动进行切换。



重要：任何时候只能激活一个 **IMSI**。当设备在同一 SIM 卡上切换到不同的 IMSI 时，HSS 会自动注销之前激活的 IMSI。

OmniHSS 实现

在 OmniHSS 中，多 IMSI SIM 上的每个 IMSI 被配置为单独的用户记录，但都引用同一张 **SIM** 卡：

用例

1. 国际漫游优化

- 家庭 IMSI: 001-001（家庭网络费率）
- 美国漫游 IMSI: 310-410（当地美国费率）
- 欧盟漫游 IMSI: 234-015（当地欧盟费率）
- 设备根据位置切换 IMSI

2. MVNO 服务

- 主 IMSI: MVNO 网络（转售商）
- 后备 IMSI: 主网络（母运营商）
- 如果 MVNO 覆盖不可用，自动故障转移

3. IoT/M2M 多网络

- IMSI 1: 主要运营商
- IMSI 2: 备份运营商以实现冗余

IMSI 3: 紧急/低成本后备
关键设备保持连接

4. 合规性

不同的 IMSI 用于不同的监管区域
遵守当地数据驻留要求
根据管辖区使用当地网络身份

多 IMSI 特性

独立认证

每个 IMSI 具有自己的 Ki、OPC 和密钥集
每个 IMSI 的独立认证向量
每个网络的不同安全凭证

单独的服务配置文件

不同的 EPC 配置文件（带宽、APN）
不同的 IMS 配置文件（语音服务）
每个 IMSI 的不同漫游规则

共享物理身份

所有 IMSI 引用同一 SIM（通过 sim_id）
所有用户记录共享相同的 ICCID
通过 SIM 卡进行逻辑分组

网络选择

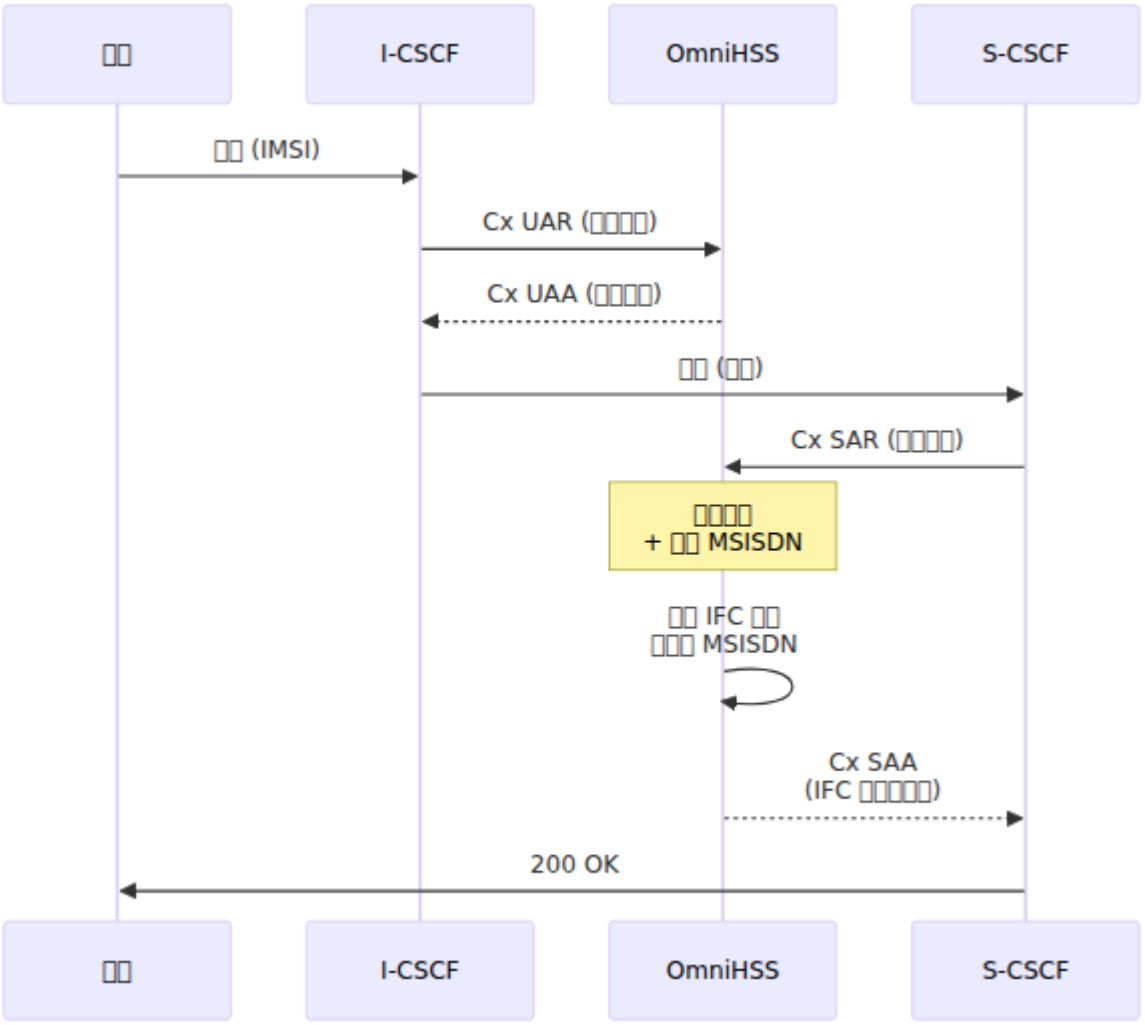
设备或 SIM 卡决定使用哪个 IMSI
基于可用网络、位置、策略
HSS 认证设备呈现的任何 IMSI

配置

```
# 1. 创建 SIM 卡（支持多 IMSI）
SIM_ID=$(curl -k -X POST https://hss.example.com:8443/api/sim \
  -d '{"sim": {"iccid": "89911012000003204510", "is_esim": false}}'
\
  | jq -r '.data.id')

# 2. 为 IMSI 1 创建密钥集（家庭网络）
KEYSET1=$(curl -k -X POST https://hss.example.com:8443/api/key_set
\
  -d '{"key_set": {"ki": "0123456789ABCDEF...", "opc":
"FEDCBA9876..."}}' \
  | jq -r '.data.id')

# 3. 创建用户 1（家庭 IMSI）
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -d "{\"subscriber\": {
    \"imsi\": \"001001111111111\",
    \"sim_id\": \"$SIM_ID\",
    \"key_set_id\": \"$KEYSET1\",
    \"epc_profile_id\": \"e58ed763-928c-4155-bee9-fdbaaadc15f3\"
  }}"

# 4. 为 IMSI 2 创建密钥集（漫游合作伙伴）
KEYSET2=$(curl -k -X POST https://hss.example.com:8443/api/key_set
\
  -d '{"key_set": {"ki": "1111111111111111...", "opc":
"2222222222..."}}' \
  | jq -r '.data.id')

# 5. 创建用户 2（漫游 IMSI）
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -d "{\"subscriber\": {
    \"imsi\": \"310410222222222\",
    \"sim_id\": \"$SIM_ID\",
    \"key_set_id\": \"$KEYSET2\",
    \"epc_profile_id\": \"f47ac10b-58cc-4372-a567-0e02b2c3d479\"
  }}"

# 6. 对 SIM 上的其他 IMSI 重复此过程...
```

认证流程

当多 IMSI 设备附加时：



HSS 不需要知道这是一个多 IMSI SIM。它认证设备呈现的任何 IMSI。

IMSI 切换和自动注销

当多 IMSI SIM 从一个 IMSI 切换到另一个 IMSI 时，网络上只能注册一个 IMSI。OmniHSS 通过在新的 IMSI 从同一 SIM 卡注册时自动发送 取消位置请求 (CLR) 来处理此事，以注销之前激活的 IMSI。

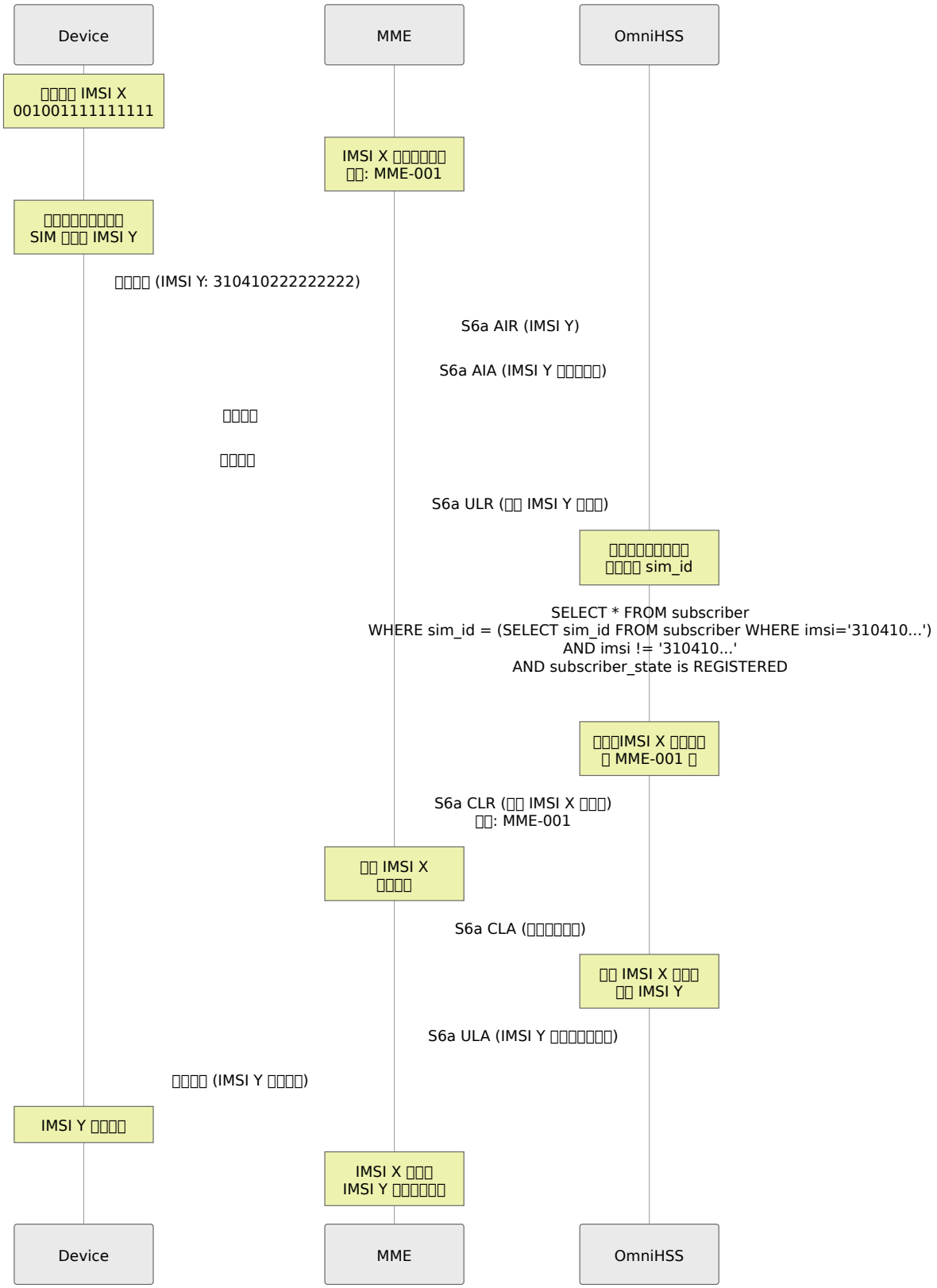
单一活动 IMSI 规则
关键概念：每张 SIM 卡只能有一个用户（IMSI）在任何时候处于活动状态。

如果用户在 MME 上注册了 IMSI X
并且 HSS 收到针对 IMSI Y（在与 IMSI X 相同的 SIM 上）的更新位置请求

- HSS □□□□□□ □□□□□□□ □□□ **IMSI X**

□□□□ IMSI □□□□□□□□□□□□□□□□□□□□□□□□

IMSI □□□□



为什么这很重要

网络完整性：

防止来自同一物理SIM的重复注册
确保网络资源得到正确释放
维护准确的用户位置数据

计费准确性：

每次网络访问仅对一个IMSI收费
IMSI切换之间的清晰会话边界
准确的CDR（通话详细记录）生成

资源管理：

旧IMSI的MME资源被释放
PDP上下文和承载被清理
位置跟踪保持准确

IMSI切换触发器

设备/SIM根据以下情况决定何时切换IMSI：

1. 网络可用性

家庭IMSI网络不可用
切换到漫游合作伙伴IMSI

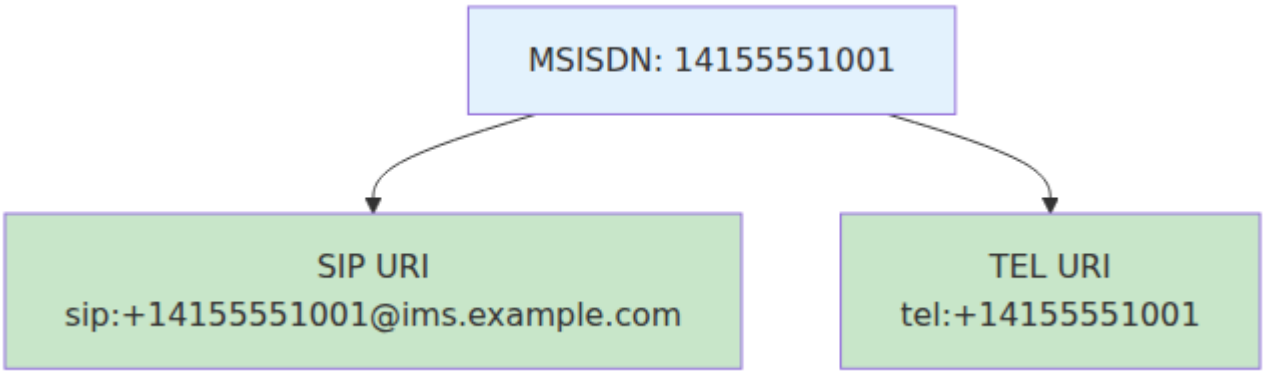
2. 手动选择

用户手动选择网络
SIM切换到相应的IMSI

3. 基于策略

SIM卡有内部规则（例如，在某些国家优先使用本地IMSI）基于MCC/MNC的自动切换

4. 成本优化



- 切换到漫游费率较低的IMSI
- 使用本地IMSI以避免漫游费用

IMS考虑事项

相同的取消位置请求行为适用于IMS注册：

操作影响

对于运营人员：

1. 用户显示离线：当IMSI切换时，旧IMSI将在HSS中显示为“注销”。这是正常行为。
2. 一个**SIM**的两个用户记录：多IMSI SIM将有多个共享相同`sim_id`的用户记录。一次只有一个处于“注册”状态。
3. 位置跟踪：`subscriber_state`表跟踪每个IMSI注册的MME/SGSN。当IMSI切换时，旧位置被清除。
4. 故障排除：如果设备无法访问：
   - 检查当前注册的是哪个IMSI
   - 验证当前网络使用的是正确的IMSI
   - 确认每个SIM只有一个IMSI处于注册状态

组合场景

多**IMSI + 多MSISDN**

您可以结合这两个功能：一个SIM上的多个IMSI，每个IMSI有多个MSISDN。

示例用例：

- 家庭网络（**IMSI 1**）：
  - 个人号码：+1-415-555-1001
  - 商务号码：+1-415-555-1002
- 美国漫游网络（**IMSI 2**）：
  - 个人号码：+1-212-555-2001
  - 商务号码：+1-212-555-2002

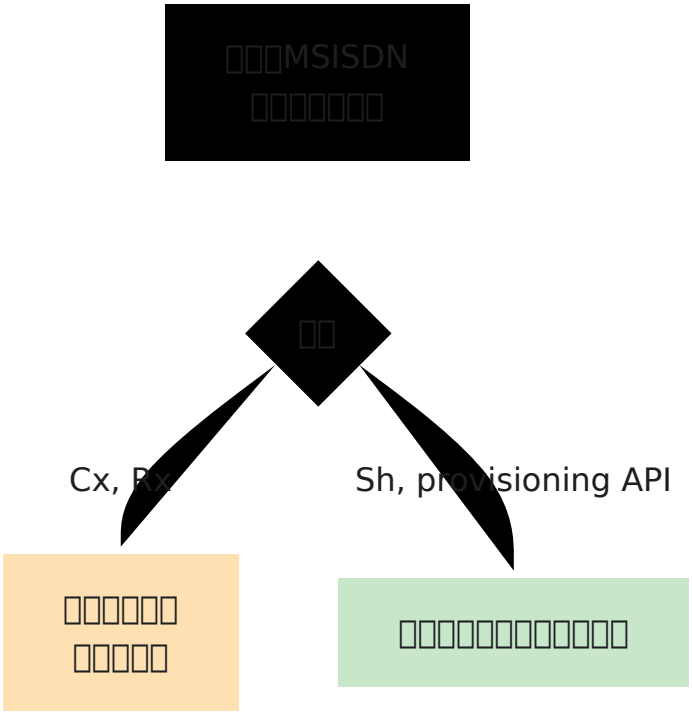
当设备在家庭区域时，使用IMSI 1及其MSISDN。当在美国漫游时，切换到IMSI 2及不同的MSISDN，以优化美国网络。

跨接口的**MSISDN**解析

MSISDN是数据模型中的共享实体。一个用户可以持有多个MSISDN（一个IMSI，多个号码）。一个MSISDN可以被多个用户持有（多个IMSI，一个号码，例如手机和智能手表）。请参见用户和MSISDN实体以获取数据模型。

由于一个号码并不总是唯一属于一个用户，因此每个接口必须决定如何解析以MSISDN或含MSISDN的公共身份为键的请求。OmniHSS使用两种不同的解析规则，您必须知道接口使用哪种规则。

□□□□□□



- □□□□□ □□□□□□□□□□□□□□□□□□□□**Cx**□**Rx**□□□□□□□□
- □□□□□ □□□□□□□□□□□□□□□□□□**Sh**□□□□□API□□□□□□

按接口的行为

接口	以...为键	每个**IMSI**的多**MSISDN**	每个**MSISDN**的多**IMSI**（共享号码）
Cx UAR, MAR	私有身份（IMSI）	正确。IMSI是精确的。	正确。每个IMSI是独立的。
Cx SAR	私有身份或公共身份	正确。IFC列出了用户的每个MSISDN。	如果以共享号码为键，则解析为一个用户。用私有身份（IMSI）选择特定用户。
Cx LIR	公共身份	正确。	当号码共享时，解析为注册用户（具有分配的S-CSCF），因此终止请求返回可达的服务节点。
Sh UDR, PUR, SNR	用户身份（MSISDN或公共身份）	正确。数据按用户分配。	返回所有持有该号码的用户。UDR聚合他们的数据。PUR更新所有用户。
Rx AAR	数据会话的用户	用户可以有多个号码，但答案携带一个MSISDN作为Subscription-Id（协议允许一个）。	正确。号码是从数据会话中选择的，而不是通过反向查找。
API `GET /subscriber/msisdn/:msisdn`	MSISDN	返回用户。	返回所有持有该号码的用户。

实际意义

- 注册和认证始终正确。 IMS注册路径（Cx MAR, SAR, UAR）以私有身份为键，其中□含IMSI。每个设备使用其自己的IMSI进行注册，因此共享号码的手机和智能手表独立注册和认证。
- 用户保留其所有号码。对于一个IMSI有多个MSISDN的情况，Cx SAR用户数据列出每个号码作为公共身份（`sip:`和`tel:`）。
- 共享号码在**Cx**上解析为一个用户，在**Sh**上解析为所有共享用户。这是上表中的解析差异。在终止路径（Cx LIR）上，HSS更喜欢注册共享用户，因此返回��达设备的服务S-CSCF。然后如何将呼叫传递给注册设备是IMS核心的问题，而不是HSS的问题。
- 要针对特定用户操作，请使用**IMSI**或**ICCID**。通过号码的反向查找在号码共享时并不唯一。

规范

这两种模型具有不同的标准状态。请了解您使用的是哪一种。

每个**IMSI**的多**MSISDN**是标准化的。 3GPP定义了一个订阅的多个号码：

- 多号码选项（3GPP TS 23.008，第2.1.3条）。HLR/HSS存储一个基本MSISDN以及额外的MSISDN。基本MSISDN指示符标记VLR和SGSN使用的号码。
- 附加**MSISDN□A-MSISDN**）（3GPP TS 23.003，第18.9条）。A-MSISDN是“分配给具有PS订阅的用户，除了已经分配的MSISDN（s）”并且“应唯一分配每个IMSI”。

每个**MSISDN**的多**IMSI**是一个实现模式，而不是单一网络标准。它通过一个多IMSI小程序在SIM上实现，该小程序存储多个IMSI并在它们之间切换（行业术语：多IMSI-多MSISDN，MIMM）。相关的GSMA指导是GSMA IR.73 漫游引导，对于eSIM则是SGP.2x/SGP.32配置规范。在网络方面，3GPP并未将一个MSISDN标准化为多个订阅的服务；相反，它��其进行了约束：

- 关联**MSISDN□C-MSISDN**）（3GPP TS 23.003，第18.7条）。C-MSISDN“在IMSI和IMS私有用户身份下唯一分配”。只有“未被多个IMS私有身份共享的MSISDN”可以作为C-MSISDN。因此，无法使用共享号码进行SRVCC会话关联。请使用每个IMSI号码规划SRVCC和CS回退，而不是共享号码。

接口规范：

- Cx（MAR, SAR, UAR, LIR）：3GPP TS 29.228和3GPP TS 29.229。
- Sh（UDR, PUR, SNR）：3GPP TS 29.328和3GPP TS 29.329。

模型	状态	参考
每个IMSI的多MSISDN	标准化	TS 23.008 §2.1.3（多号码）；TS 23.003 §18.9（A-MSISDN，唯一每个IMSI）
每个MSISDN的多IMSI	SIM小程序模式（MIMM）；在网络上受到约束	GSMA IR.73; TS 23.003 §18.7（共享MSISDN不能是C-MSISDN）

共享公共用户身份（一个号码，多个设备）

为了使一个号码响铃多个设备（例如，一个手机和一个在不同SIM上的伴侣智能手表），将设备分组为一个**IMS**订阅。这是3GPP共享公共用户身份的模型（3GPP TS 23.228）：多个私有用户身份（IMPI，每个IMSI一个）共享一个公共身份，并且关键是一个单一的服务S-CSCF。然后，S-CSCF将终止呼叫分发到每个注册的设备。

没有组的情况下，仅共享一个号码的两个用户是两个独立的IMS订阅。它们可以分配不同的S-CSCF，因此终止呼叫仅到达其中一个的设备。`ims_subscription`组通过为成员提供一个共同的S-CSCF来消除该差距。

工作原理

- 一个`ims_subscription`持有组的共享S-CSCF（`assigned_scscf`）。
- 每个成员用户引用该组（`ims_subscription_id`）。
- 当第一个成员注册时，HSS选择一个S-CSCF。当节点的Cx SAR确认注册时，HSS将该组锚定到该节点。每个后续成员都被指向相同的S-CSCF，无论其注册声称哪个S-CSCF。
- 锚定始终持有一个真实的S-CSCF节点。使用`selection_method: :pool_uri`时，HSS在第一次REGISTER时返回池名称，但它锚定DNS选择的节点，而不是池名称。请参见每个订阅一个服务S-CSCF。
- 重新注册返回锚定的节点，而不是池名称。
- 共享S-CSCF仅在最后一个成员注销时释放。
- 在终止请求（Cx LIR）上，HSS在任何成员注册时返回共享S-CSCF。
- Cx SAR用户数据列出每个成员IMPI（隐式注册集），因此S-CSCF注册整个集合。组上的可选共享服务配置文件为共享身份呈现一个iFC。
- 在对共享公共身份的Sh请求（数据引用`IMSPublicIdentity`）中，HSS在用户数据中包含每个成员IMSI的一个`<IMSPrivateUserIdentity>`。应用服务器（例如MMTel TAS）计算这些以检测共享号码：计数大于1标记为多

IMSI共享号码，因此AS允许终止呼叫分发到每个设备。普通号码返回一个`<IMSPrivateUserIdentity>`并传递给一个联系人。

配置

创建组，将每个设备的用户附加到组，并将共享号码分配给每个成员。

```
# 1. 创建IMS订阅□组）。可选的ims_profile_id设置组共享身份的共享服务配置文件。
GROUP_ID=$(curl -k -X POST
https://hss.example.com:8443/api/ims_subscription \
  -H "Content-Type: application/json" \
  -d '{"name": "Smith family - shared number"}' | jq -r
'.response.id')

# 2. 将每个成员用户附加到组并给予共享MSISDN。
curl -k -X PATCH
https://hss.example.com:8443/api/subscriber/$PHONE_SUBSCRIBER_ID \
  -H "Content-Type: application/json" \
  -d "{\"ims_subscription_id\": \"$GROUP_ID\", \"msisdns\":
[\"$SHARED_MSISDN_ID\"]}"

curl -k -X PATCH
https://hss.example.com:8443/api/subscriber/$WATCH_SUBSCRIBER_ID \
  -H "Content-Type: application/json" \
  -d "{\"ims_subscription_id\": \"$GROUP_ID\", \"msisdns\":
[\"$SHARED_MSISDN_ID\"]}"
```

参数：

参数	类型	必需	描述
`name`	字符串	是	IMS订阅的人类可读名称（唯一）。
`ims_profile_id`	UUID	否	组共享身份的共享服务配置文件（iFC）。缺失时，每个成员呈现自己的IMS配置文件。
`ims_subscription_id`	UUID	否	在用户上：它所属的组。当缺失时，用户是独立的（其自己的每用户S-CSCF，默认）。

组上的`assigned_scscf`和`last_seen_scscf_timestamp`是HSS写入的运行时状态；您不需要设置它们。

注意事项

- 没有`ims_subscription_id`的用户是独立的，行为与之前完全相同。

- 给予成员相同的MSISDN，以��共享号码解析为该组。成员可以保留自己的其他号码。
- 删除组不会删除其用户；他们的`ims_subscription_id`被清除，变为独立。

操作程序

管理多**MSISDN**用户

查看用户的所有MSISDN:

通过API查询: `GET /api/subscriber/imsi/:imsi`

响应□括所有链接的MSISDN。

故障排除多**IMSI**

设备未使用第二个**IMSI**进行附加:

1. 验证该IMSI是否存在第二个用户记录
2. 检查key_set是否为该IMSI正确配置
3. 验证EPC配置文件是否已分配
4. 确认漫游规则允许附加

设备意外切换**IMSI**:

- 这是由设备/SIM逻辑控制，而不是HSS
- HSS认证任何呈现的IMSI
- 检查设备IMSI选择设置

故障排除多**MSISDN**

第二个号码未响铃:

1. 验证MSISDN是否在连接表中链接
2. 检查IMS配置文件模板是否□含`{{msisdns}}`变量

3. 确认IMS注册□括所有公共身份
4. 查看S-CSCF日志以获取注册身份

外呼仅显示一个号码：

- 设备选择哪个号码作为来电显示
- 这是设备配置，而不是HSS
- HSS提供所有身份；设备选择

优势总结

多**MSISDN**优势

✓ 一张SIM，多部手机号码
✓ 分开商务和个人线路
✓ 国际本地存在
✓ 简化设备管理
✓ 所有号码���享相同的数据服务
✓ 按IMSI集中计费

多**IMSI SIM**优势

✓ 优化漫游成本
✓ 自动网络选择
✓ 冗余和故障转移
✓ 本地网络身份
✓ 符合监管要求
✓ 跨网络服务连续性

组合优势

✓ 最大灵活性
✓ 每个网络不同的号码集
✓ 针对每个用例优化

✓ 复杂的商业场景

✓ 国际和本地优化

← 返回操作指南

OmniHSS 配置管理

← 返回操作指南

概述

OmniHSS 使用 配置 来定义订阅者的服务特性。配置允许您创建可重用的服务模板，这些模板可以分配给多个订阅者，从而简化配置过程并确保一致性。

配置类型

EPC 配置

EPC（演进分组核心）配置定义 LTE 订阅者的数据服务特性。

关键参数

参数	描述	典型值
`ue_ambr_dl_kbps`	下载速度限制	10,000 - 1,000,000 Kbps
`ue_ambr_ul_kbps`	上传速度限制	5,000 - 500,000 Kbps
`network_access_mode`	服务类型	"packet_only" 或 "packet_and_circuit"
`tracking_area_update_interval_seconds`	TAU 计时器	54 秒（典型）

创建 EPC 配置

```
curl -k -X POST https://hss.example.com:8443/api/epc/profile \
  -H "Content-Type: application/json" \
  -d '{
    "apn_profiles": [],
    "name": "Premium 100Mbps",
    "network_access_mode": "packet_only",
    "tracking_area_update_interval_seconds": 600,
    "ue_ambr_dl_kbps": 100000,
    "ue_ambr_ul_kbps": 50000
  }'
```

常见 EPC 配置模板

基础互联网：

下载: 10 Mbps (10,000 Kbps)
上传: 5 Mbps (5,000 Kbps)

标准:

下载: 50 Mbps (50,000 Kbps)
上传: 25 Mbps (25,000 Kbps)

高级:

下载: 100 Mbps (100,000 Kbps)
上传: 50 Mbps (50,000 Kbps)

无限制:

下载: 1 Gbps (1,000,000 Kbps)
上传: 500 Mbps (500,000 Kbps)

IMS 配置

IMS 配置定义语音服务特性，主要通过 IFC（初始过滤标准）模板实现。

IFC 模板

IFC 模板是定义 S-CSCF 呼叫路由规则的 XML 文档。

模板变量:

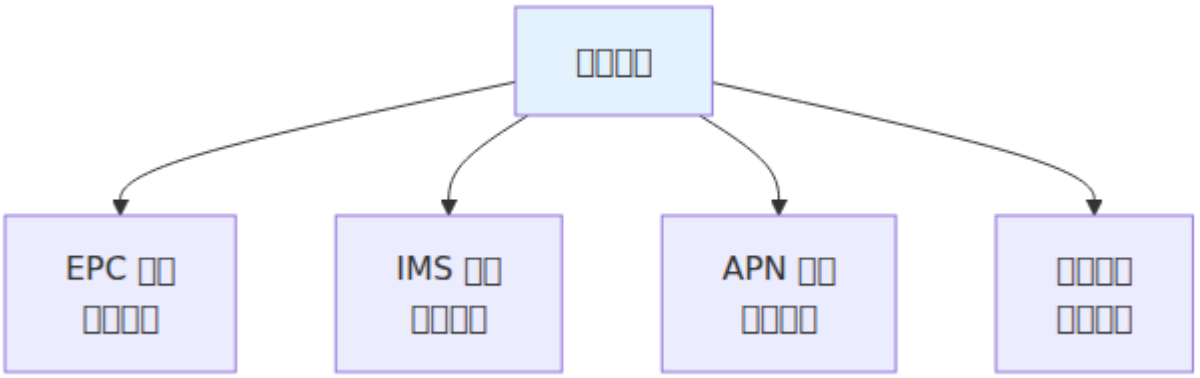
{{imsi}} - 订阅者 IMSI
{{msisdns}} - 电话号码列表
{{mcc}} - 本国代码
{{mnc}} - 本网络代码

□□ IMS □□

```
curl -k -X POST https://hss.example.com:8443/api/ims/profile \
  -H "Content-Type: application/json" \
  -d '{
    "ims_profile": {
      "name": "Standard VoLTE",
      "ifc_template": "<InitialFilterCriteria>...
</InitialFilterCriteria>"
    }
  }'
```

IFC □□□□

```
<ServiceProfile>
  <PublicIdentity>
    <Identity>sip:
{{imsi}}@ims.mnc{{mnc}}.mcc{{mcc}}.3gppnetwork.org</Identity>
  </PublicIdentity>
  <InitialFilterCriteria>
    <Priority>0</Priority>
    <TriggerPoint>
      <ConditionTypeCNF>0</ConditionTypeCNF>
      <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
      </SPT>
    </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:as.ims.example.com</ServerName>
      <DefaultHandling>0</DefaultHandling>
    </ApplicationServer>
  </InitialFilterCriteria>
</ServiceProfile>
```



APN 配置

APN（接入点名称）配置定义数据连接的网络接入点。

APN 组件

APN 标识符

定义 APN 名称和 IP 协议支持。

常见 **APN:**

- `internet` - 一般互联网接入
- `ims` - IMS/VoLTE 信令
- `mms` - 多媒体消息
- `vzwadmin` - 运营商特定

IP 版本选项**:**

- `"ipv4"`: 仅 IPv4
- `"ipv6"`: 仅 IPv6
- `"ipv4v6"`: IPv4v6（双栈）
- `"ipv4_or_ipv6"`: IPv4 或 IPv6（网络选择）

APN QoS 配置

定义服务质量参数。

QCI（服务质量类别标识符）值：

QCI	类型	用例	优先级
1	GBR	对话语音	最高
2	GBR	对话视频	高
4	GBR	视频流	高
5	非 GBR	IMS 信令	中
9	非 GBR	互联网（默认）	最低

会话级计费方法（可选）：

APN QoS 配置还可以固定 IP-CAN 会话的默认计费方法，作为 Gx CCA-I 中的顶级 `Online`/`Offline` AVP 发送。

字段	类型	默认	描述
`online_charging_enabled`	boolean	`null`	设置后，强制会话级 `Online` AVP（`true` = 启用，`false` = 禁用）。当为 `null` 时，PCRF 回显 PGW 在 CCR-I 中发送的值。
`offline_charging_enabled`	boolean	`null`	设置后，强制会话级 `Offline` AVP（`true` = 启用，`false` = 禁用）。当为 `null` 时，PCRF 回显 PGW 在 CCR-I 中发送的值。

有关完整行为和 AVP 值参考，请参见 PCRF: 会话级在线/离线计费。

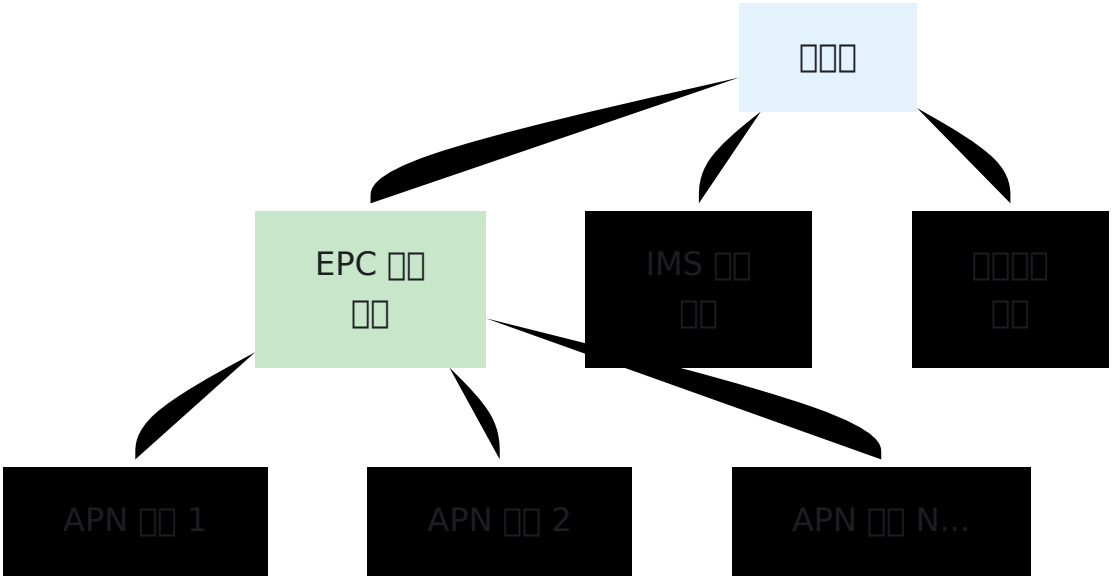
创建完整的 APN 配置

```
# 1. 创建 APN 标识符
APN_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/identifier \
  -H "Content-Type: application/json" \
  -d '{"apn": "internet", "ip_version": "ipv4v6"}' \
  | jq -r '.response.id')

# 2. 创建 APN QoS 配置
QOS_ID=$(curl -k -X POST
https://hss.example.com:8443/api/apn/qos_profile \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Best Effort",
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 50000,
    "apn_ambr_ul_kbps": 25000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true,
    "qci": 9
  }' | jq -r '.response.id')

# 3. 创建 APN 配置
curl -k -X POST https://hss.example.com:8443/api/apn/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"apn_identifier_id\": \"$APN_ID\",
    \"apn_qos_profile_id\": \"$QOS_ID\",
    \"name\": \"Internet APN\"
  }"
```

将 APN 分配给 EPC 配置

APN 通过 `join_epc_profile_to_apn_profile` 表与 EPC 配置链接。

插入记录到连接表中，将 APN 配置 ID 链接到 EPC 配置 ID。多个 APN 配置可以分配给一个 EPC 配置。



漫游配置

请参见 漫游控制指南 中的详细文档。

配置分配

订阅者配置关系

将配置分配给订阅者

```
# 在创建订阅者时分配 EPC 和 IMS 配置
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "imsi": "001001123456789",
      "key_set_id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
      "epc_profile_id": "e58ed763-928c-4155-bee9-fdbaaadc15f3",
      "ims_profile_id": "7c9e6679-7425-40de-944b-e07fc1f90ae7",
      "roaming_profile_id": "9c858901-8a57-4791-81fe-4c455b099bc9"
    }
  }'

# 更新订阅者配置
curl -k -X PUT
https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-4b01-
90e6-d701748f0851 \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "epc_profile_id": "f47ac10b-58cc-4372-a567-0e02b2c3d479"
    }
  }'
```

配置管理最佳实践

设计原则

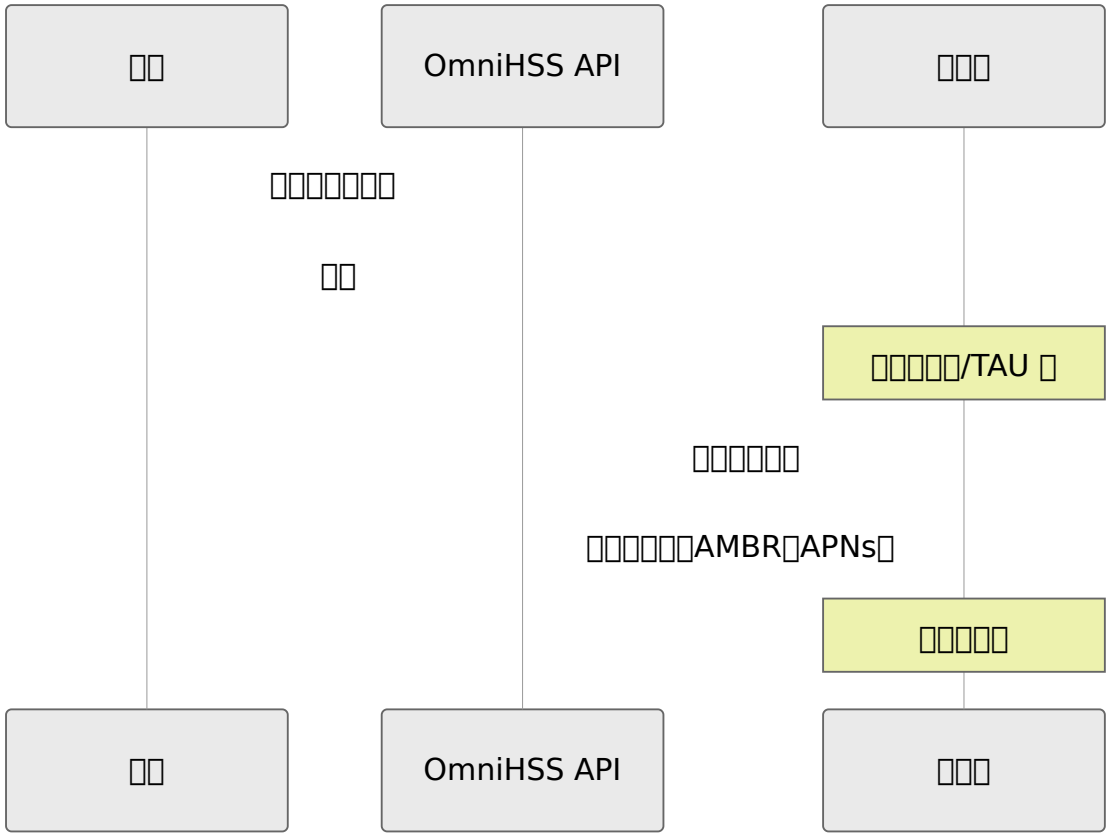
1. 创建标准配置 - 定义常见服务层级（基础、标准、高级）
2. 重用配置 - 将相同配置分配给多个订阅者
3. 记录更改 - 跟踪配置修改
4. 生产前测试 - 首先验证配置在测试订阅者上的有效性

□□□□□□

```
[□□□□□]-[□□]-[□□]

□□:
- "Basic-10Mbps-Internet"
- "Premium-100Mbps-VoLTE"
- "Enterprise-1Gbps-MultiAPN"
```

□□□□

□□□□□□□□□□□□□



□□: □□□□□□□□□□□□

- □□□□□□□□TAU□
- □□
- IMS □□□□□□ IMS □□□□□□

故障排除配置问题

订阅者未获得预期速度:

1. 检查分配的 EPC 配置 AMBR 值
2. 检查 APN QoS 配置 AMBR 值
3. 验证 MME/P-GW 是否正确执行 QoS
4. 检查网络拥塞

IMS 注册失败:

1. 验证已分配 IMS 配置
2. 检查 IFC 模板 XML 的有效性
3. 审查 S-CSCF 日志以获取 IFC 处理错误
4. 确认 S-CSCF 选择配置

APN 不可用:

1. 验证 APN 配置是否链接到 EPC 配置
2. 检查 APN 标识符是否与网络请求匹配
3. 审查 UE 的 PDN 连接请求

← 返回操作指南 | 下一步: 漫游控制 →

OmniHSS 协议流程
← 返回操作指南

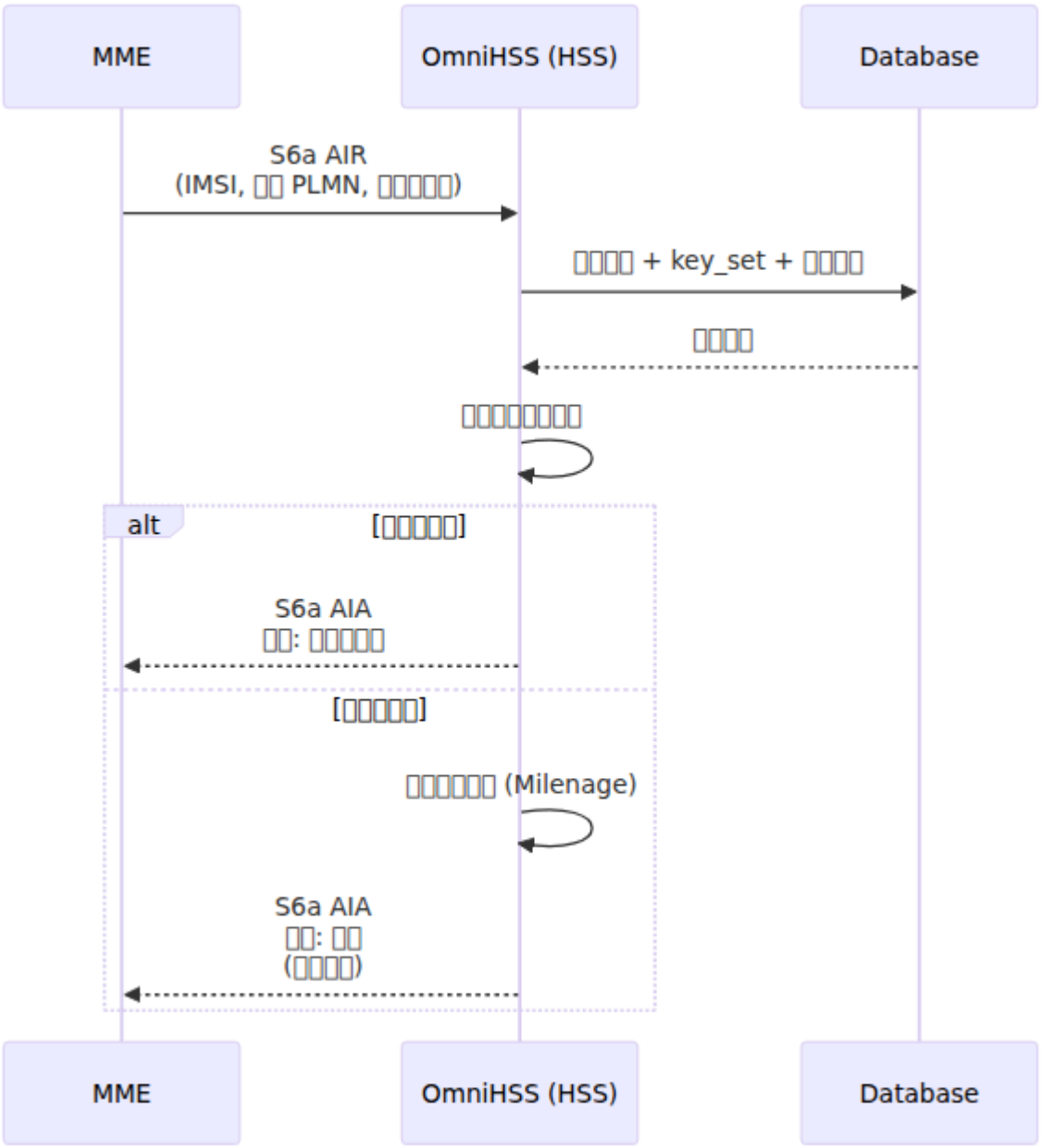
概述

本文档详细介绍了 OmniHSS 支持的 Diameter 协议消息流程。理解这些流程对于故障排除和操作至关重要。

S6a 接口 (LTE/EPC)

认证信息请求 (AIR/AIA)

MME 请求用户的认证向量。

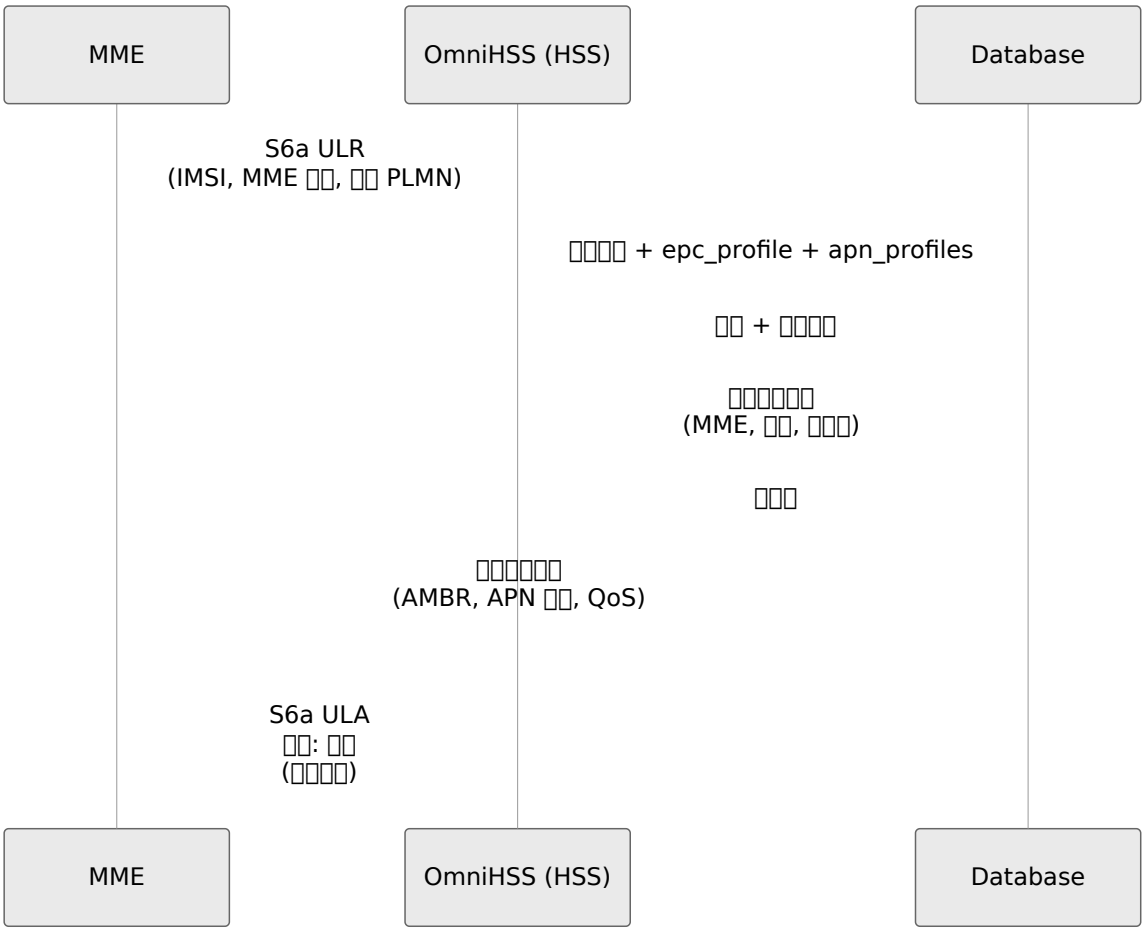


关键 **AVP:**

- 请求: 用户名 (IMSI), 访问-PLMN-Id, 请求的向量数量
- 响应: 认证信息 (RAND, AUTN, XRES, KASME)

更新位置请求 **(ULR/ULA)**

MME 通知 HSS 用户位置并检索订阅数据。

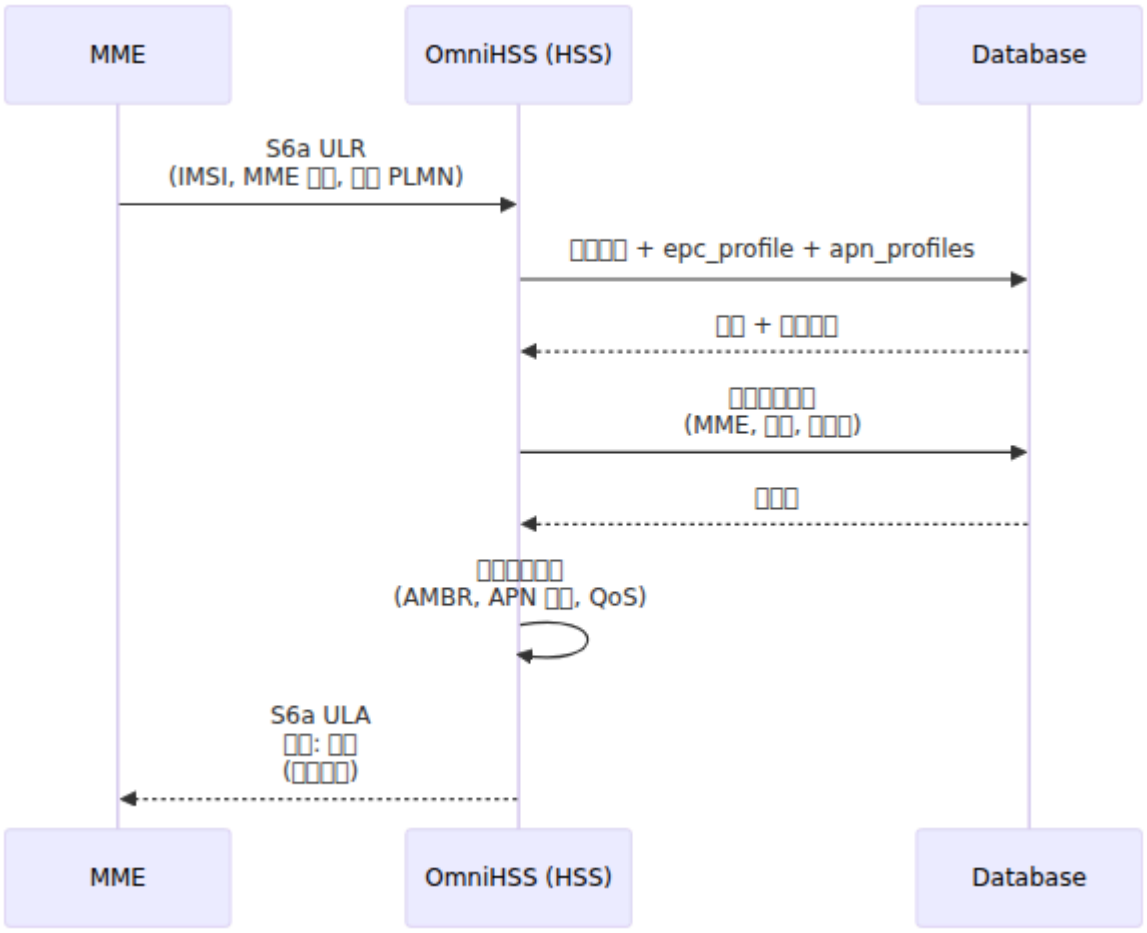


□□ **AVP:**

- □□: □□□ (IMSI), RAT □□, ULR □□, □□-PLMN-Id, UE-SRVCC □□
- □□: □□□□ (AMBR, APN □□, □□□□□□)

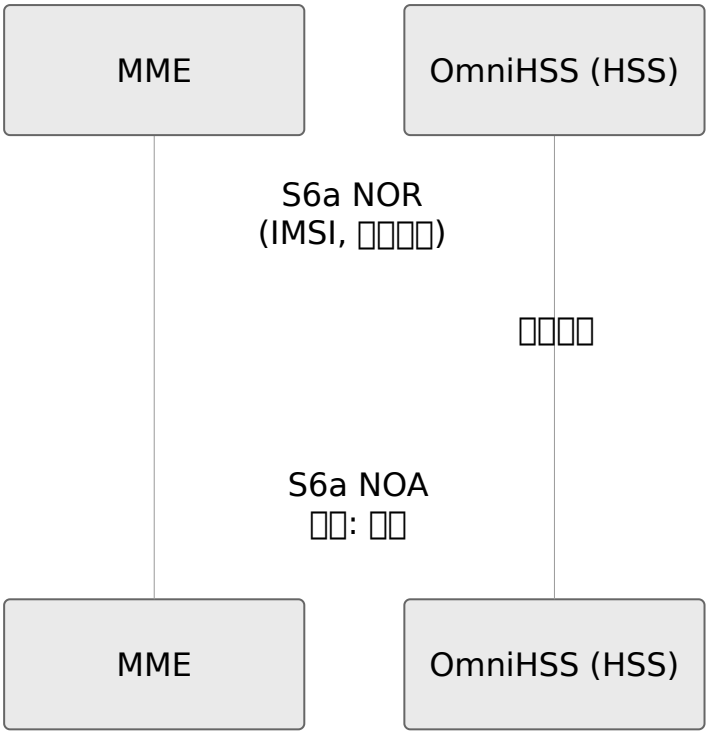
□□ UE □□ (PUR/PUA)

MME □□ HSS □□□□□□□□□□□□



通知请求 **(NOR/NOA)**

MME 通知 HSS 各种事件。

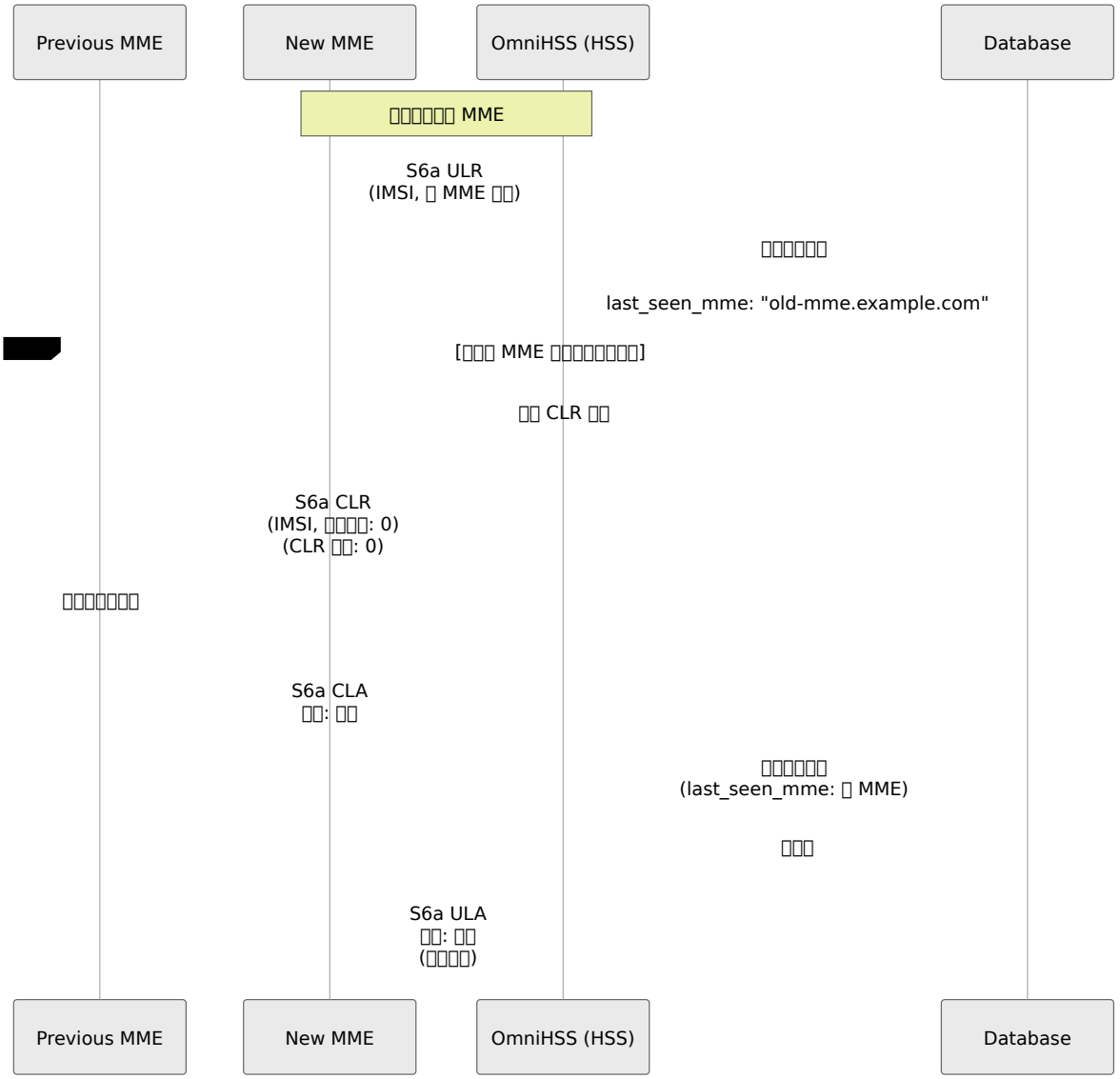


取消位置请求 **(CLR/CLA)**

HSS 发起位置取消以通知 MME 用户应被分离。OmniHSS 支持自动和程序化 CLR 发送。

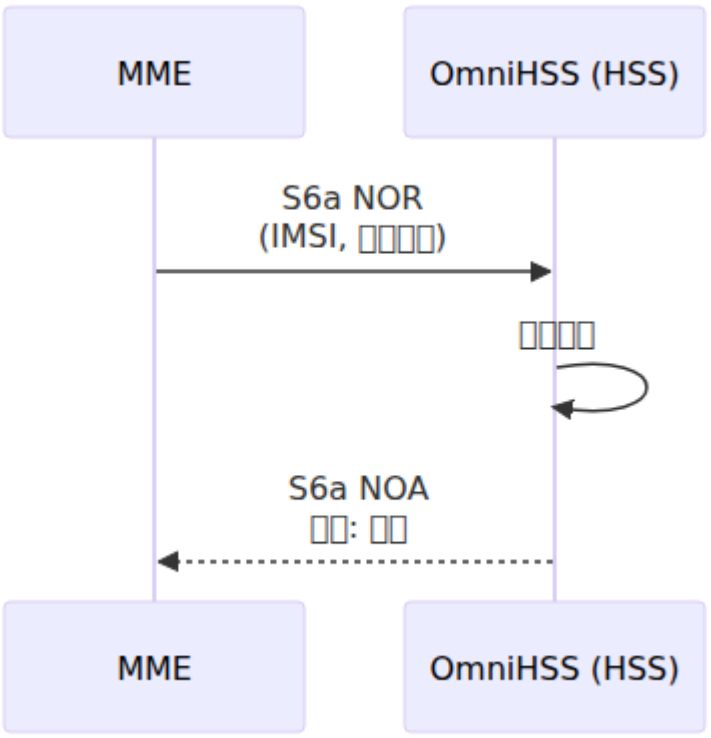
自动 **CLR (MME** 切换**)**

当用户从新 MME 执行更新位置请求时，OmniHSS 自动向先前的 MME 发送 CLR，以清理过时的注册。



□□ AVP (□□ CLR):

- □□□: □□□ IMSI
- □□□□: □□ MME □□□
- □□□: □□ MME □
- □□□□: 0 (MME □□□□)
- CLR □□: 0
- □□□□: □□□□□□□□□



关键 **AVP (**程序化 **CLR):**

- 用户名: 用户的 IMSI
- 目标主机: 最后看到的 MME 主机名
- 目标域: 最后看到的 MME 域
- 取消类型: `:subscription_withdrawal` (按 3GPP TS 29.272 编码为整数)
- CLR 标志:
  - s6a_indicator: 1 (表示使用 S6a 接口)
  - reattach_required: 1 (UE 必须重新认证以重新附着)

取消类型

OmniHSS 支持多种取消类型，符合 3GPP TS 29.272:

类型	值	描述	使用案例
MME 更新过程	0	正常 MME 更改	从新 MME 自动进行 ULR
SGSN 更新过程	1	SGSN 切换	3G/2G 切换场景
订阅撤回	2	管理终止	通过 API 手动分离
更新过程 IWF	3	互通功能更新	传统网络互操作
初始附着过程	4	新注册	强制重新认证

CLR 标志

CLR 标志 AVP 是一个位掩码，具有以下字段：

标志	位	描述
S6a/S6d 指示器	0	1 = 使用 S6a 接口
重新附着所需	1	1 = UE 必须执行新附着

示例 **CLR** 标志配置**：**

```
clr_flags: %{
  s6a_indicator: 1,         # 使用 S6a 接口
  reattach_required: 1      # 强制重新认证
}
```

多 **IMSI** 场景

OmniHSS 跟踪每个用户 (IMSI) 的 MME 注册，而不是每个 MSISDN。这对于理解多 IMSI 场景中的 CLR 行为至关重要：

场景 **1:** 多个 **MSISDN**，单个 **IMSI**

```
用户 A:
  - IMSI: 999000123456789
  - MSISDNs: ["+1234567890", "+9876543210"]
  - last_seen_mme: "mme01.operator.com"
```

当此用户移动到新 MME 时:

- 发送一个 **CLR** 到 "mme01.operator.com", IMSI 为 999000123456789
- 两个 MSISDN 都受到影响（同一用户，同一 SIM）
- 用户名 AVP 包含 IMSI, 而不是 MSISDNs

场景 **2:** 多个用户（不同 **IMSI**），相同 **MSISDN**

OmniHSS 强制 唯一 **MSISDN** 约束（一个 MSISDN 不能同时属于多个用户）。但是，在迁移/移植期间:

```
用户 A:
  - IMSI: 999000111111111
  - MSISDN: "+1234567890"
  - last_seen_mme: "mme01.operator.com"

用户 B（迁移后）:
  - IMSI: 999000222222222
  - MSISDN: "+1234567890"  # 相同 MSISDN, 不同 SIM/IMSI
  - last_seen_mme: "mme02.operator.com"
```

当用户 B 注册时:

- 不发送 **CLR**（不同 IMSI = 不同用户）
- 用户 A 保持在 mme01 注册
- 用户 B 在 mme02 注册
- 两者可以同时处于活动状态（不同的物理设备）

场景 **3:** 程序化 **CLR** 对于多 **MSISDN** 用户

结果:

- 发送一个 **CLR** 到用户的 last_seen_mme
- 所有 **MSISDN** 与该 IMSI 关联的都有效地被分离

- IMSI 是跟踪 MME 注册的主键

重要说明

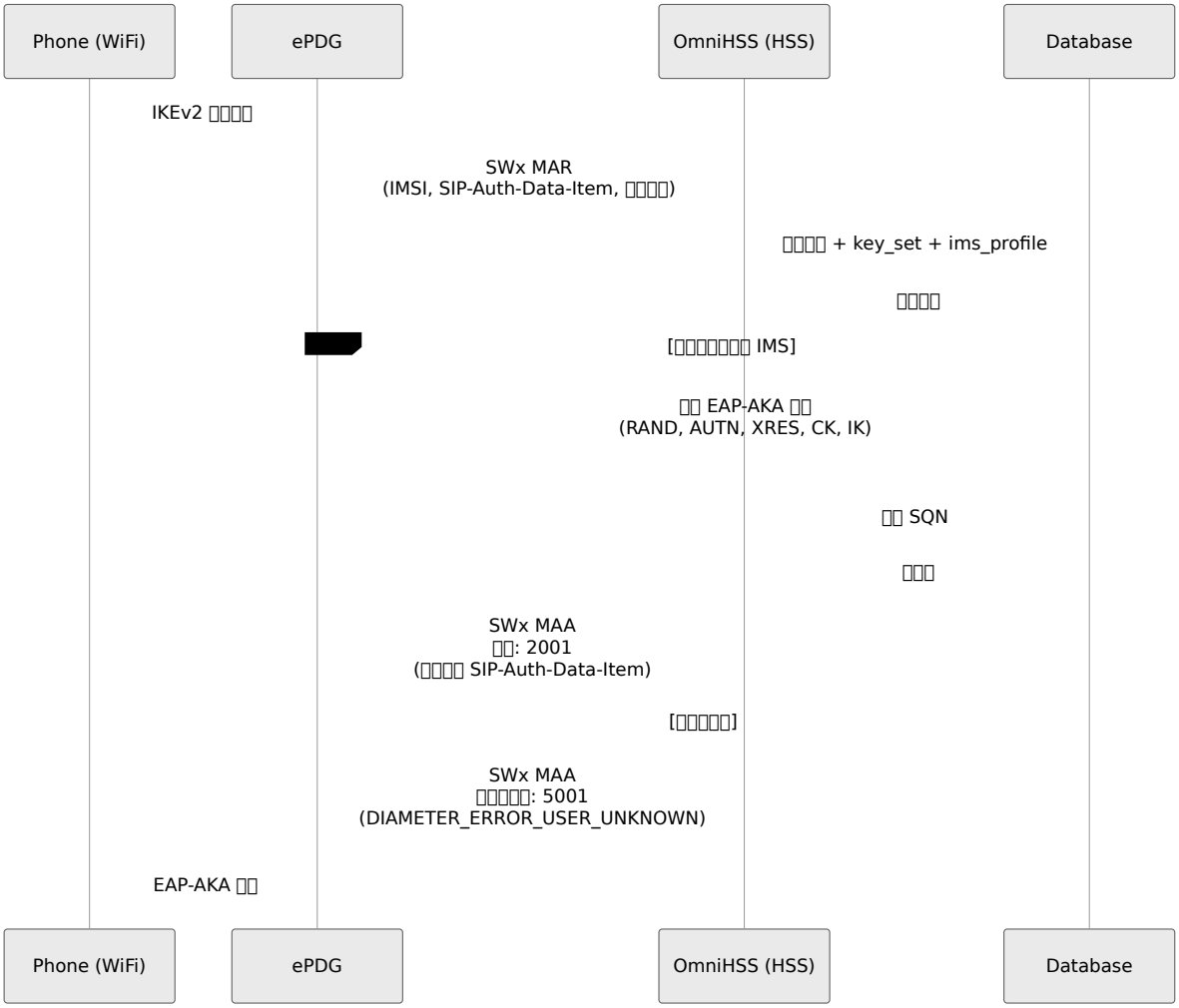
1. **IMSI** 是关键**:** CLR 操作始终是按 **IMSI**，而不是按 MSISDN。`subscriber_state` 表按用户 (IMSI) 跟踪 `last_seen_mme`。
2. 原子操作**:** 每个用户一次只能在一个 MME 注册���自动 CLR 通过清理旧注册来确保这一点。
3. 如果没有先前的 **MME** 则不发送 **CLR:** 如果 `last_seen_mme` 为 `nil`（用户从未注册），在 ULR 期间不发送 CLR。
4. 包含订阅数据**:** 自动 CLR（在 ULR 期间）括完整的 `Subscription-Data` AVP，以帮助旧 MME 正确清理上下文。
5. 异步**:** CLR 是异步发送的（火并忘记）。对新 MME 的 ULA 响应不等待来自旧 MME 的 CLA。
6. **CLA** 处理**:** OmniHSS 接收 CLA 响应，但当前会丢弃它们。这防止了消息循环，并且是标准 HSS 行为。

SWx 接口 (非 3GPP 接入 / WiFi 通话)

SWx 接口通过将 ePDG（演进数据网关）连接到 HSS，以便在非 3GPP 接入网络上进行用户认证和授权，从而启用 WiFi 通话。

多媒体认证请求 (MAR/MAA)

ePDG 请求 WiFi 通话注册的 EAP-AKA 认证向量。



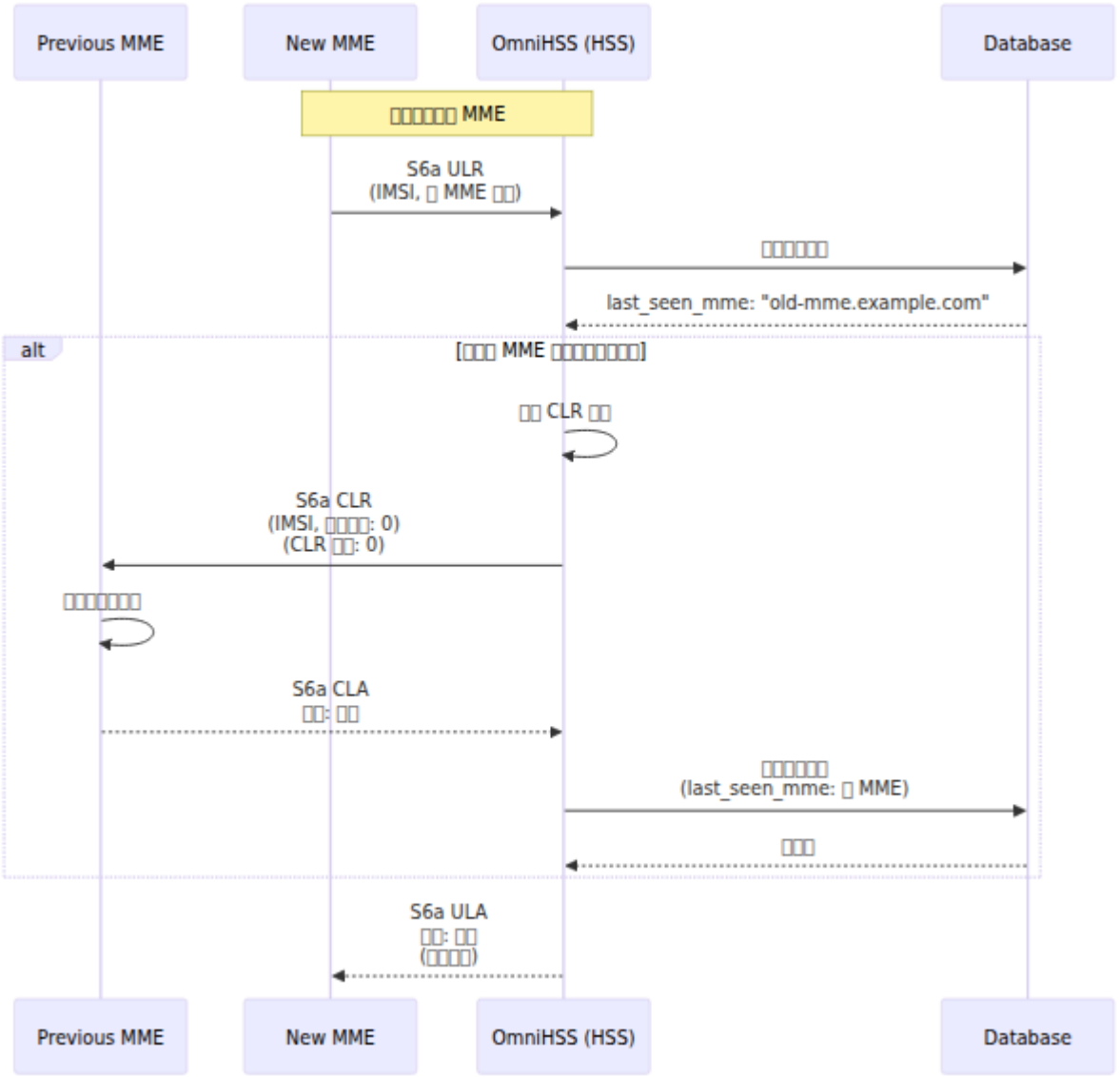
□□ AVP:

- □□: □□□ (IMSI □ IMSI@realm), SIP-Auth-Data-Item (□□□□), SIP-Number-Auth-Items
- □□: SIP-Auth-Data-Item (SIP-Authenticate, SIP-Authorization, □□□□, □□□□□)

SQN □□□□: □ UE □□□□□ SQN □□□□□□□□□□□□□ePDG □ SIP-Authorization AVP □□□□□□ □□□□OmniHSS □□□□□□□ AUTS□□□□ 3GPP TS 33.102 □□ IND □□□□□□□□□□ SQN□

□□□□□□□ (SAR/SAA)

ePDG □□□□□□□□□□ WiFi □□□□□



Previous MME
New MME
OmniHSS (HSS)
Database
□□□□□□□ MME
S6a ULR
(IMSI, □ MME □□)
□□□□□□□
last_seen_mme: "old-mme.example.com"
alt
[□□□ MME □□□□□□□□]
□□ CLR □□
S6a CLR
(IMSI, □□□□: 0)
(CLR □□: 0)
□□□□□□□□
S6a CLA
□□: □□
□□□□□□□
(last_seen_mme: □ MME)
□□□
S6a ULA
□□: □□
(□□□□)
Previous MME
New MME
OmniHSS (HSS)
Database

服务器分配类型：

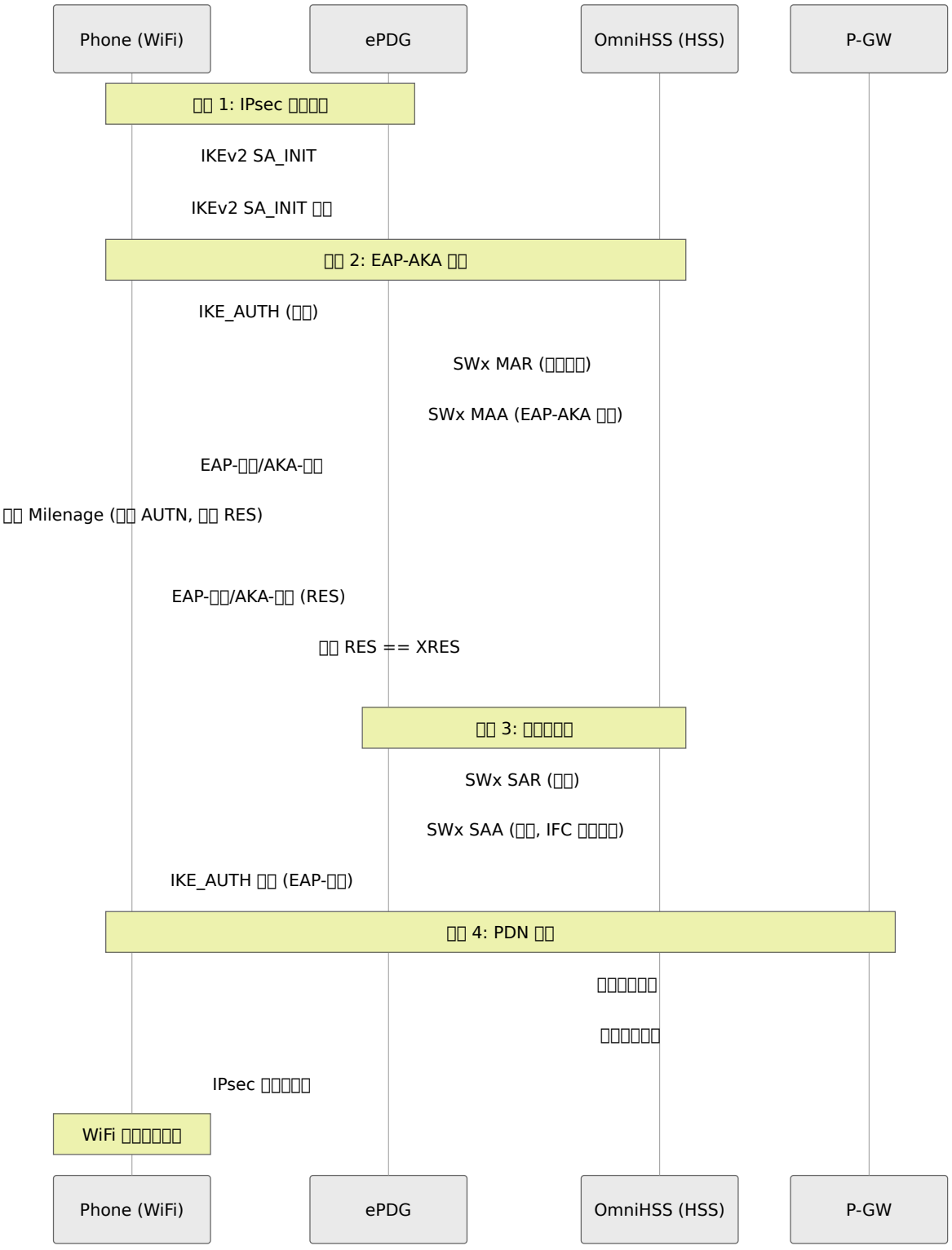
类型	值	描述
NO_ASSIGNMENT	0	仅查询，无状态变化
REGISTRATION	1	初始 WiFi 通话注册
RE_REGISTRATION	2	注册刷新
UNREGISTERED_USER	3	未注册用户的服务
TIMEOUT_DEREGISTRATION	4	注册超时
USER_DEREGISTRATION	5	用户发起的注销
AUTHENTICATION_FAILURE	7	认证失败，清除状态
ADMINISTRATIVE_DEREGISTRATION	8	管理员发起的注销

关键 **AVP:**

- 请求：用户名 (IMSI)，服务器分配类型，服务器名称 (ePDG 身份)，服务选择 (APN)
- 响应：用户数据 (IFC 配置文件 XML)，非 3GPP 用户数据

WiFi 通话注册流程

完整的端到端 WiFi 通话注册。

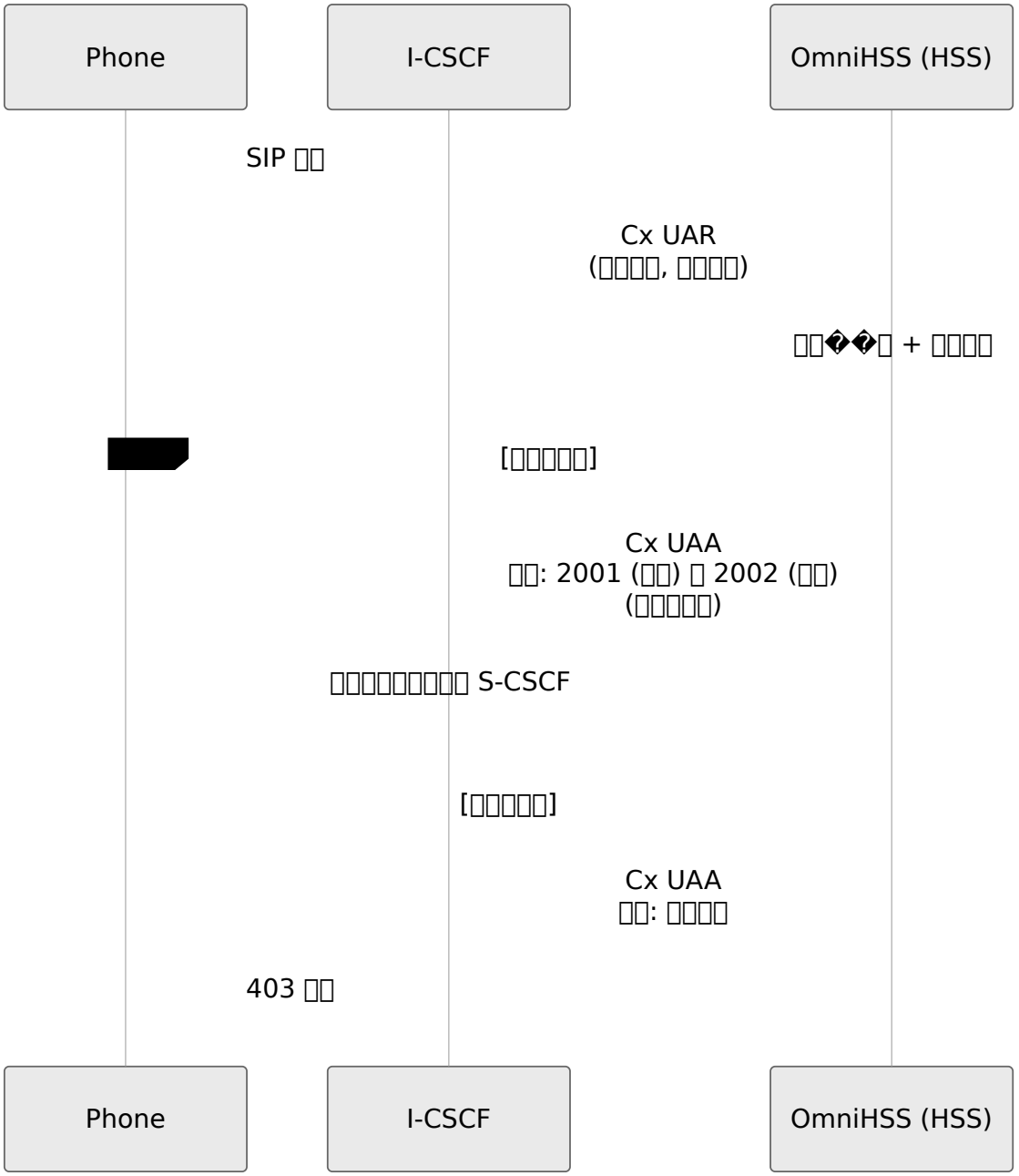


Phone (WiFi)
ePDG
OmniHSS (HSS)
P-GW
阶段 1: IPsec 隧道设置
IKEv2 SA_INIT
IKEv2 SA_INIT 响应
阶段 2: EAP-AKA 认证
IKE_AUTH (身份)
SWx MAR (请求向量)
SWx MAA (EAP-AKA 向量)
EAP-请求/AKA-挑战
运行 Milenage (验证 AUTN, 计算 RES)
EAP-响应/AKA-挑战 (RES)
验证 RES == XRES
阶段 3: 服务器分配
SWx SAR (注册)
SWx SAA (成功, IFC 配置文件)
IKE_AUTH 完成 (EAP-成功)
阶段 4: PDN 连接
创建会话请求
创建会话响应
IPsec 隧道已建立
WiFi 通话准备就绪
Phone (WiFi)
ePDG
OmniHSS (HSS)
P-GW

Cx □□ (IMS)

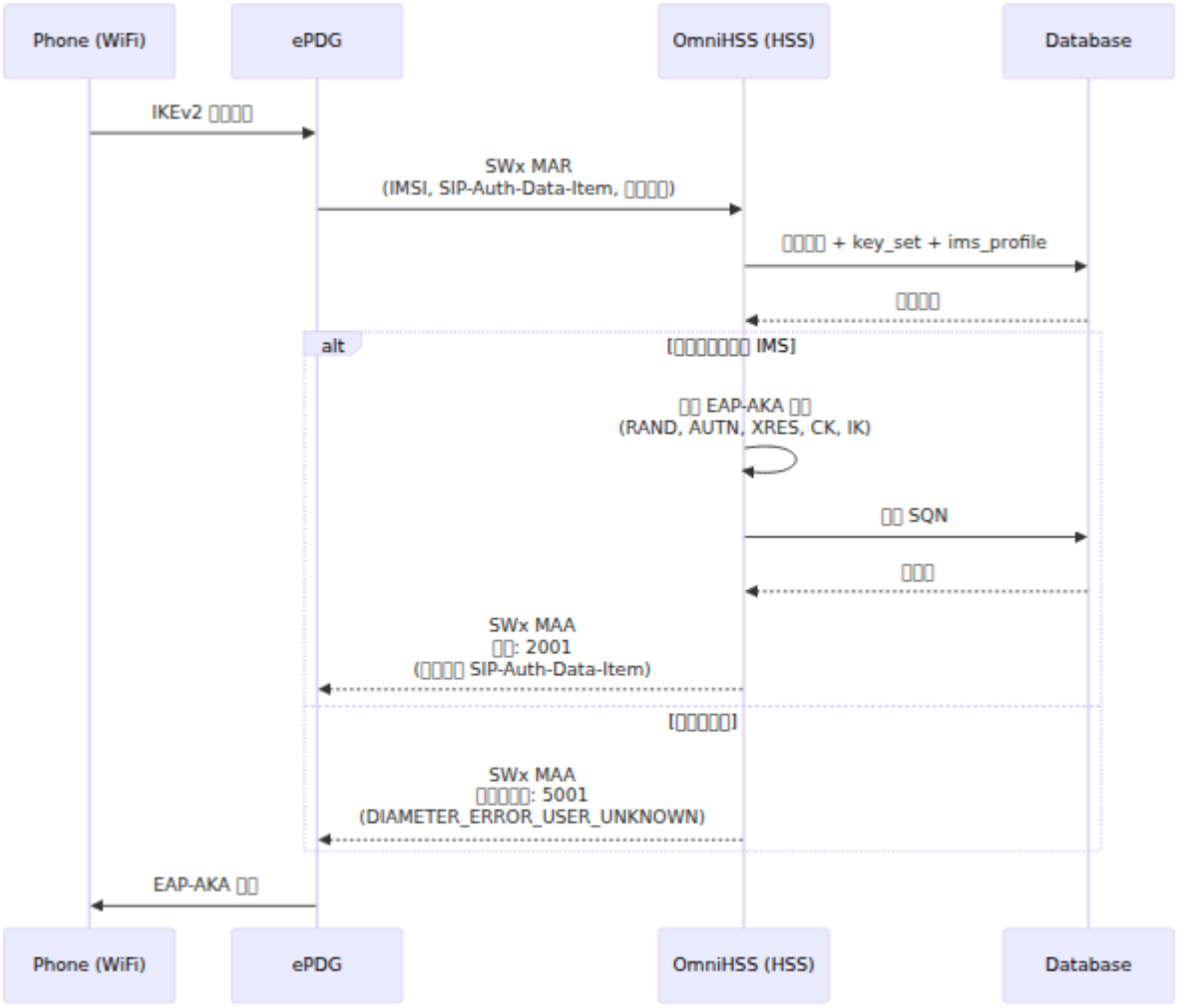
□□□□□□□ (UAR/UAA)

I-CSCF □□□□□□□□□□□□□□□□□□□ S-CSCF□HSS □□□□ S-CSCF □□□□□□ `Server-Name`□□□□ □□□□□□□□□□□□□□□□ S-CSCF □□□□ □□□



□□□□□□□ (SAR/SAA)

S-CSCF □□/□□□□□□□ IMS □□□□□



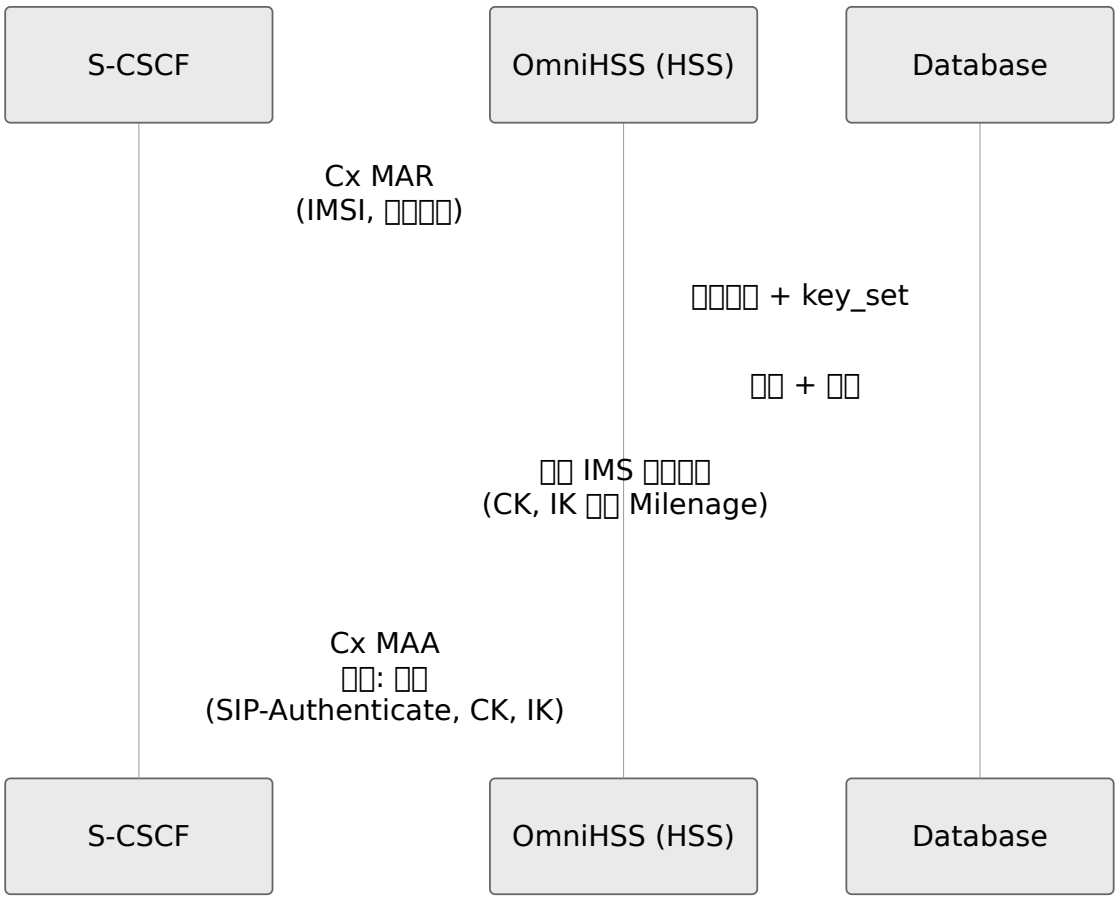
IFC 模板渲染**:**

- `{{imsi}}` → 实际 IMSI
- `{{msisdns}}` → 电话号码列表
- `{{mcc}}, {{mnc}}` → 家庭 PLMN 代码

有关 SAR 如何更新注册状态和认证待处理状态，以及如何处理 `AUTHENTICATION_FAILURE` 和 `AUTHENTICATION_TIMEOUT`，请参见 S-CSCF 分配状态。

多媒体认证请求 **(MAR/MAA)**

S-CSCF 请求 IMS 注册的认证向量。



S-CSCF 分配状态

HSS 为每个用户持有两个独立的 IMS 状态。注册状态和认证待处理状态是独立的，符合 3GPP TS 29.228 §3.1。

状态字段	设置于	清除于	含义
`assigned_scscf`	SAR 成功（注册）	SAR 注销，认证失败/超时（未注册），或认证失败/超时使过时名称失效	用户在此 S-CSCF 上注册。
`auth_pending_scscf`	MAR	SAR（成功或认证失败/超时），UAR 注销，或 TTL 到期	此 S-CSCF 上正在进行 IMS-AKA 挑战。
`auth_pending_since`	MAR	与 `auth_pending_scscf` 一起	挑战开始的时间。HSS 使用它来使未完成的挑战过期。

MAR 存储 S-CSCF 名称并设置认证待处理状态，符合 TS 29.228 §6.3。SAR 确认或清除它，符合 TS 29.228 §6.1.2。

UAA S-CSCF 选择顺序。对于每个 UAR，HSS 返回一个 S-CSCF 名称和一个结果代码，优先级如下：

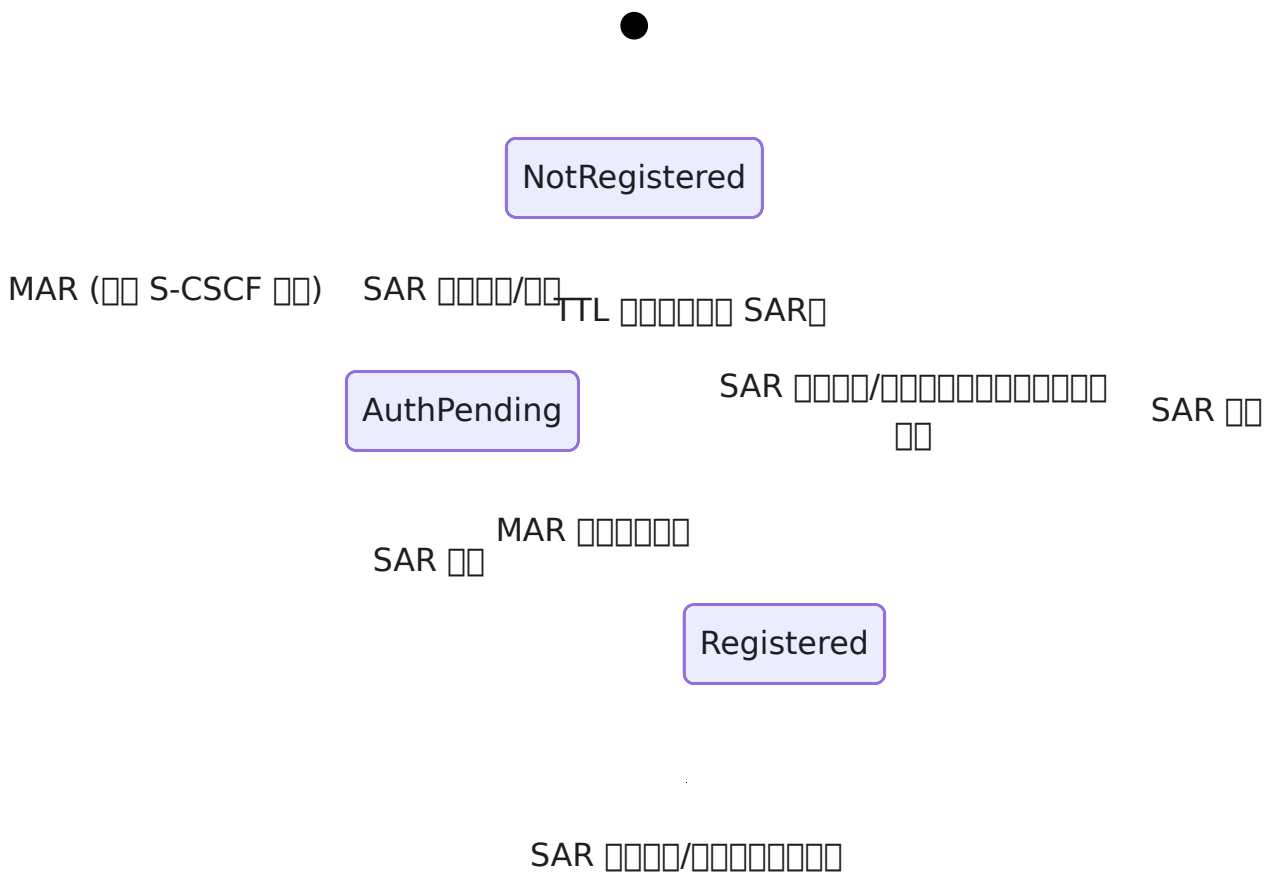
1. 记录中有一个活动的认证待处理 S-CSCF：返回它，带有 `DIAMETER_SUBSEQUENT_REGISTRATION` (2002)。当 `auth_pending_scscf` 被设置且 `auth_pending_since` 在 `auth_pending_ttl_seconds` 内时，挑战是活动的。
2. `assigned_scscf` 持有一个真实的 S-CSCF 节点：返回它，带有 `DIAMETER_SUBSEQUENT_REGISTRATION` (2002)。
3. 两者都不适用：选择一个新的 S-CSCF（见 S-CSCF 选择配置），带有 `DIAMETER_FIRST_REGISTRATION` (2001)。

规则 2 有意测试真实节点。池 URI 是整个池的名称，因此它从不识别为用户提供服务的节点。如果早期版本记录了池 URI，HSS 将用户视为未服务，并回答 2001。下一个 SAR 将存储真实节点。

重新注册始终返回服务 **S-CSCF**。规则 2 使分配保持不变。在节点接收注册后，每个重新注册 UAR 和每个终止 LIR 都返回该节点。HSS 仅对尚未服务的用户返回空 URI。

HSS 优先选择认证待处理 S-CSCF 而不是 `assigned_scscf`。这使得正在进行的挑战周期收敛到发起挑战的节点。对于健康注册的用户，发起挑战的节点是分配的节点，因此优先级没有影响。它仅在 S-CSCF 故障恢复 中重要。

认证待处理状态会过期。挑战周期在几秒钟内完成，每个结果都会发送一个 SAR 来清除状态。如果发起挑战的 S-CSCF 从未发送该 SAR，则待处理名称将保留在记录中。HSS 会忽略超过 `auth_pending_ttl_seconds`（默认 30 秒）的待处理名称。如果没有 TTL，孤立或被遗弃的 MAR 将永远超越已注册的绑定。发起的流量将发送到持有无内容的节点，而终止流量仍将发送到 `assigned_scscf`。



Cx □□□□:

□□	□□	□□	□□
2001	DIAMETER_FIRST_REGISTRATION	□□□□□ S-CSCF□□□□□□□□□□	TS 29.229
2002	DIAMETER_SUBSEQUENT_REGISTRATION	□□□□□□ S-CSCF□HSS □□□□□□□	TS 29.229

□□□□□□□□□□□ (SAR):

值	名称	**HSS 行为**
0	NO_ASSIGNMENT	无需 S-CSCF 分配，仅下载用户数据
1	REGISTRATION	用户已注册，通常在重新注册时使用
2	RE_REGISTRATION	用户已重新注册
4	TIMEOUT_DEREGISTRATION	由于 S-CSCF 超时，用户已注销
5	USER_DEREGISTRATION	用户已请求注销
6	TIMEOUT_DEREGISTRATION_STORE_SERVER_NAME	由于超时，用户已注销，但保留 S-CSCF 名称
7	USER_DEREGISTRATION_STORE_SERVER_NAME	用户已请求注销，但保留 S-CSCF 名称
8	ADMINISTRATIVE_DEREGISTRATION	由于 S-CSCF 强制，用户已注销
9	AUTHENTICATION_FAILURE	由于未能成功完成身份验证，用户已注销
10	AUTHENTICATION_TIMEOUT	由于身份验证未在合理时间内完成，用户已注销
11	DEREGISTRATION_TOO_MUCH_DATA	由于 S-CSCF 存储过多数据，用户已注销

类型 6 和 7 结束注册，但保留 S-CSCF 名称，依据 TS 29.228 §6.1.2。该节点继续为订阅者的未注册状态提供服务，因此稍后的 UAR 仍然解析到它。

过时名称检测在认证失败时。 AUTHENTICATION_FAILURE 或 AUTHENTICATION_TIMEOUT SAR 通常保持已注册订阅者的绑定。只有一个例外。如果 SAR 来自于与记录上不同的 S-CSCF，则 I-CSCF 必须将 REGISTER 重新分叉到池中，因为它无法到达记录上的节点。HSS 然后将该名称清除为过时。如果没有此规则，S-CSCF 故障恢复 不会收敛：HSS 会在每个后续的 UAR 中返回死节点，只有 UE 在任何超时 SAR 之前完成的挑战才能恢复订阅者。

请参见 hss_cx_auth_abandon_total 指标以获取被放弃或失败的挑战的可见性。

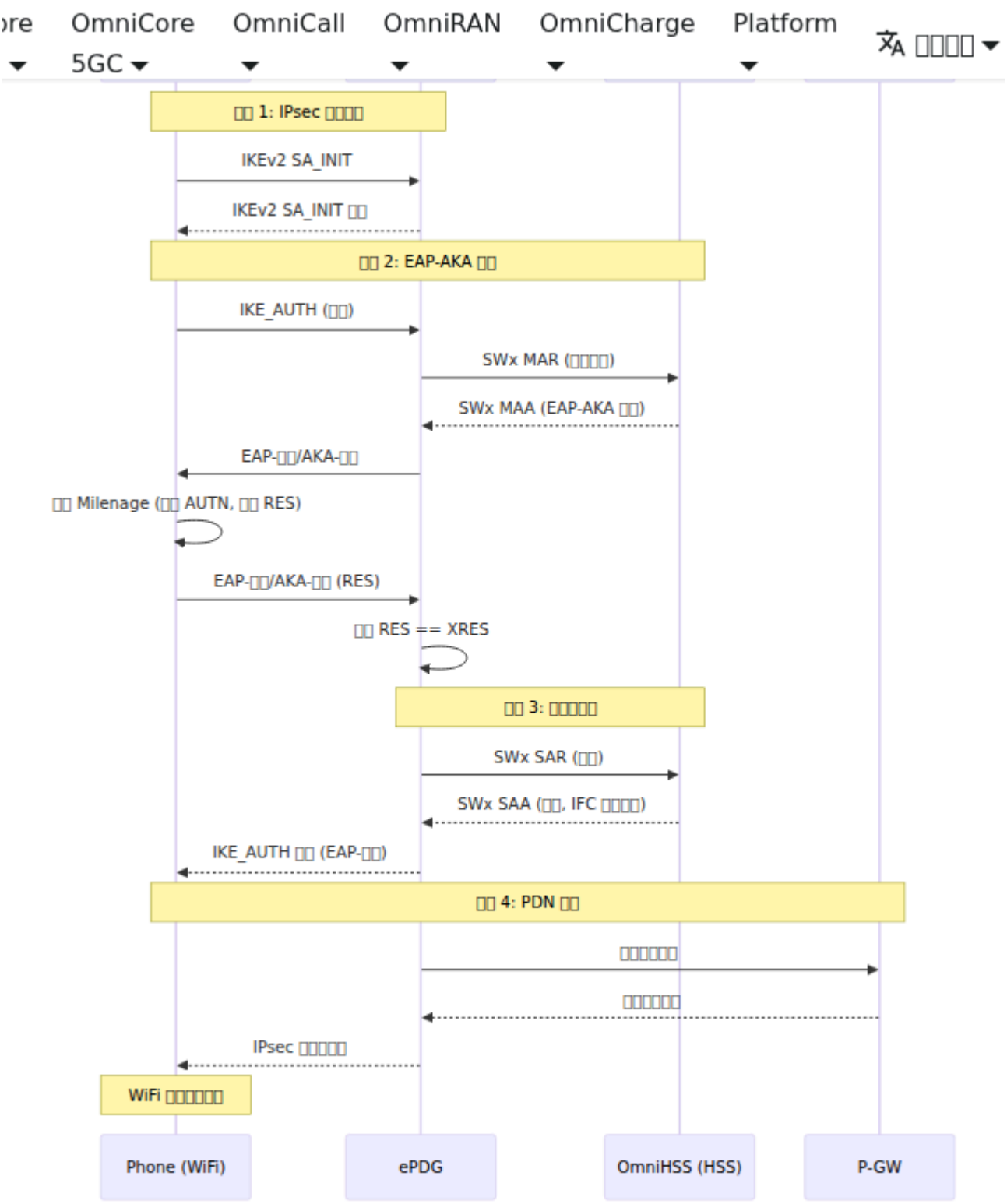
每个订阅一个 S-CSCF

共享 IMS 订阅将多个 IMSI 或多个 MSISDN 组合在一个身份下（见 多 MSISDN 和多 IMSI 特性）。必须有且仅有一个 S-CSCF 为整个订阅提供服务。如果两个成员在不同节点上注册，终止呼叫将到达一个没有被叫方绑定的节点。

HSS 在 IMS 订阅上持有一个共享的 S-CSCF 名称，称为锚：

1. 第一个成员注册。没有设置锚，因此 UAA 返回一个选择（一个池 URI，或:random_peer 下的节点）。
2. 一个真实节点接受注册并发送其 SAR。HSS 将订阅锚定到该节点。
3. 每个后续成员读取锚。其 UAA 返回锚定的节点，因此每个成员都落在同一个 S-CSCF 上。
4. 当最后一个成员注销时，锚被释放。

HSS 从不锚定池 URI。池 URI 命名池，而不是节点，因此 HSS 仅在 I-CSCF 解析它的时间内持有它。如果锚持有来自早期版本的池 URI，则下一个 SAR 会用真实的服务节点替换它。



S-CSCF 故障恢复

如果记录上的 S-CSCF 无法访问，I-CSCF 会将 REGISTER 重新发送到 S-CSCF 池。一个活跃的 S-CSCF 然后为订阅者提供服务。这需要 I-CSCF 池回退路由和 HSS `auth_pending_scscf` 偏好一起使用。

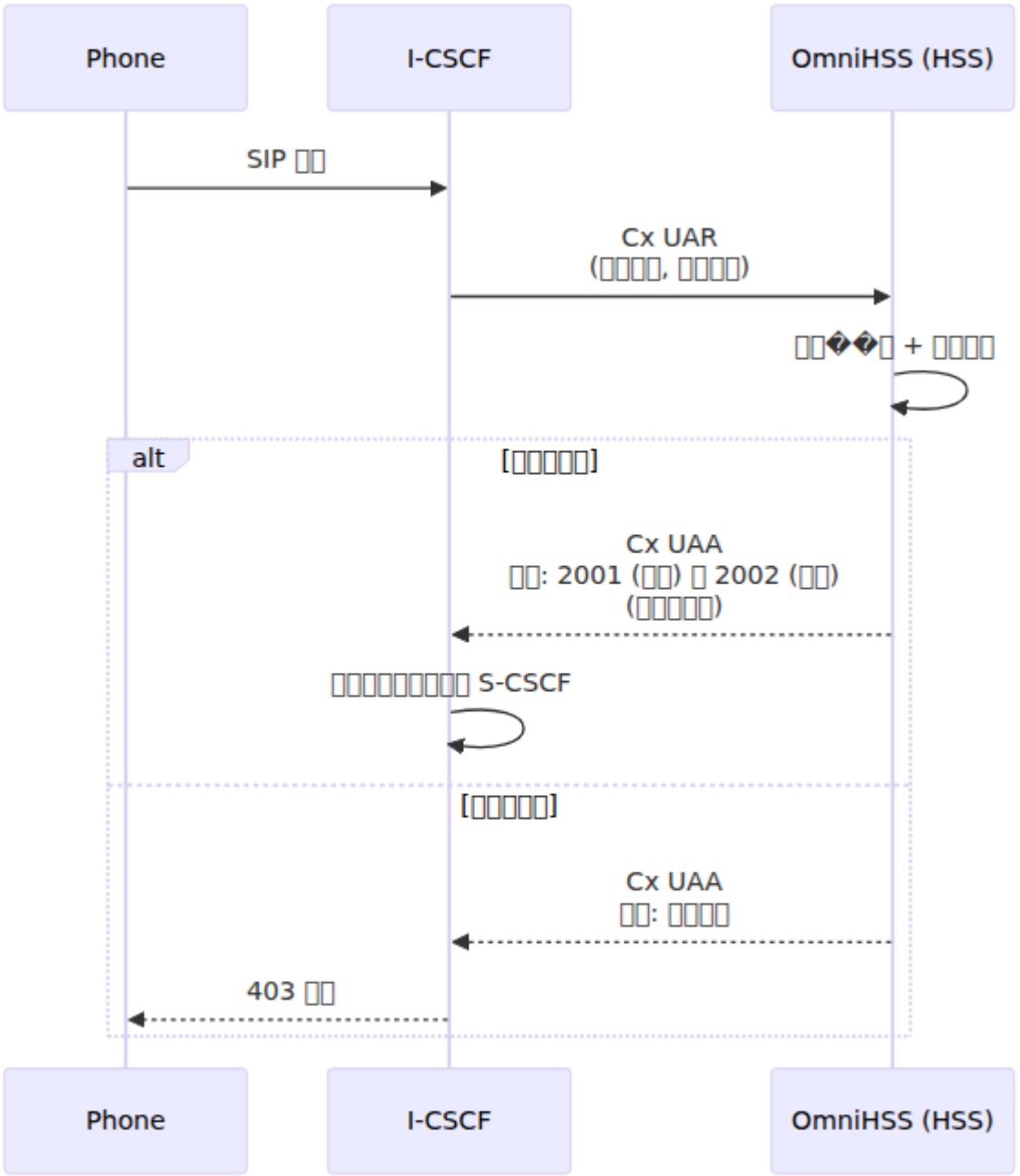
有两种故障模式：

故障模式	表现方式	恢复
无法解析的 S-CSCF 名称（HSS 数据中的退役或过时名称）	DNS 返回 NXDOMAIN。I-CSCF 转发失败，在创建分支之前。	I-CSCF 重新发送到池 URI。DNS SRV 选择一个活跃的 S-CSCF。
可解析但死 S-CSCF（S-CSCF 服务已关闭）	TCP 连接拒绝。分支异步失败。	I-CSCF 从其故障路由重新发送到池 URI。`use_dst_blocklist` 和 `use_dns_failover` 使 DNS 跳过死节点。

经过认证的 REGISTER 必须到达发出挑战的同一个活跃 S-CSCF。UAA 中的 `auth_pending_scscf` 偏好使这一点得以实现：在活跃节点发送其 MAR 之后，HSS 在下一个 UAR 中返回该节点，而不是记录上的死节点。如果没有该偏好，这两个 REGISTER 将发送到不同的池成员，循环将无法收敛。

限制。 如果 S-CSCF 在 IMS-AKA 挑战周期中被杀死，UE 将被新的 S-CSCF 以新的随机数重新挑战。UE 在下一个注册时恢复。这是由于在挑战中间更换 S-CSCF 的固有特性，因为正在进行的认证向量无法转移到不同的 S-CSCF。

要启用此恢复，请设置 `selection_method: :pool_uri`（见 S-CSCF 选择配置）并部署 I-CSCF 池回退路由。



位置信息请求 **(LIR/LIA)**

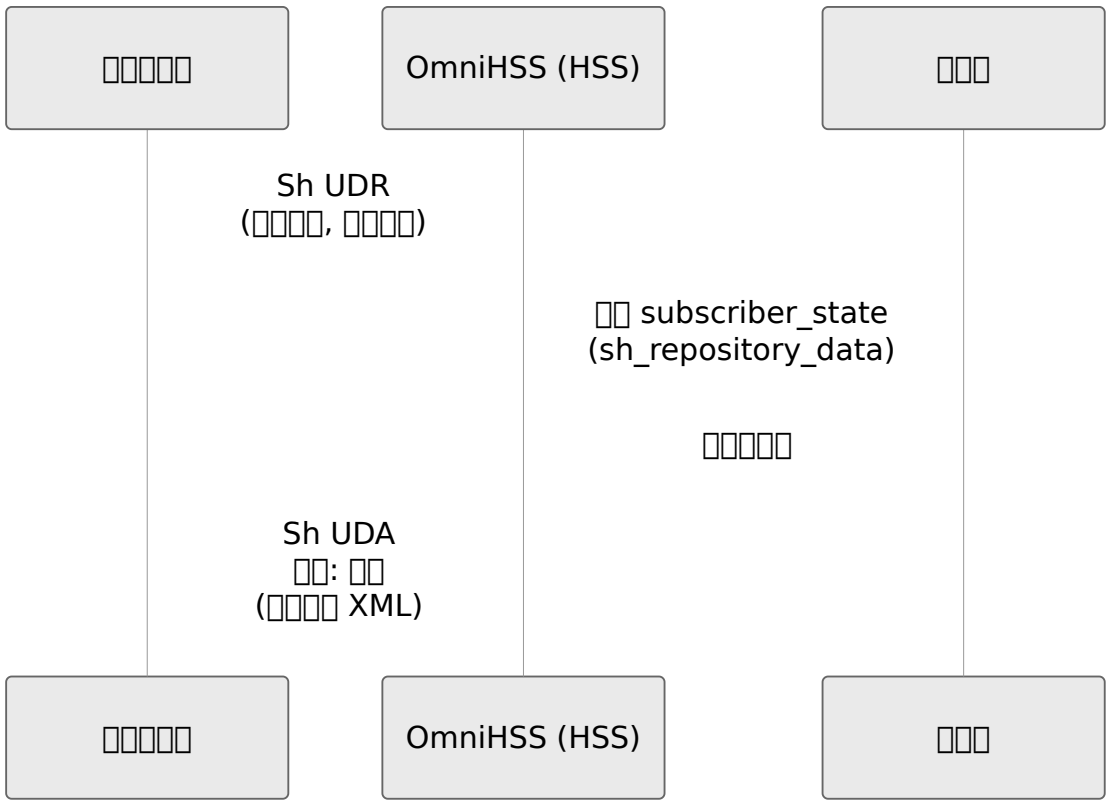
I-CSCF 查询哪个 S-CSCF 正在为用户提供服务。

Sh 接口 (IMS 配置数据)

请参见 **Sh RepositoryData (透明数据)** 以获取数据模型、序列号/并发规则，以及 **MMTEL** 服务 / **IMS-ODB** 信息文档的 **REST** 供应 **API**。

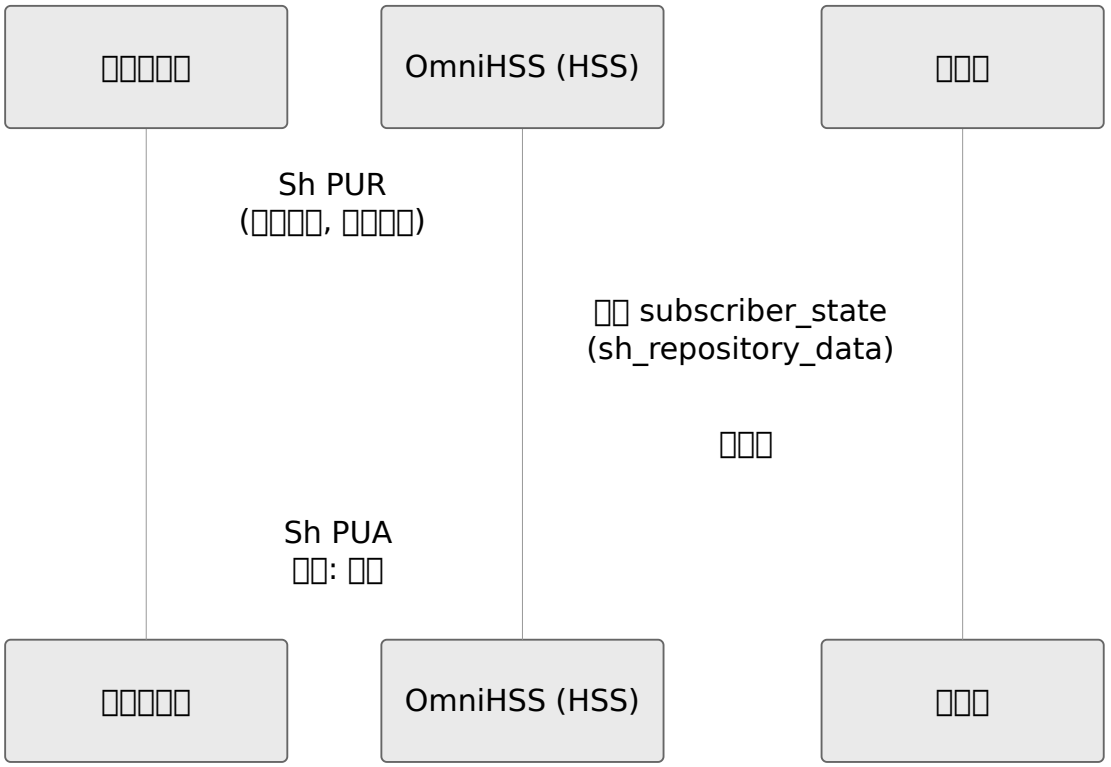
□□□□□□□ (UDR/UDA)

□□□□□□□□□□□□□□□□□□□□



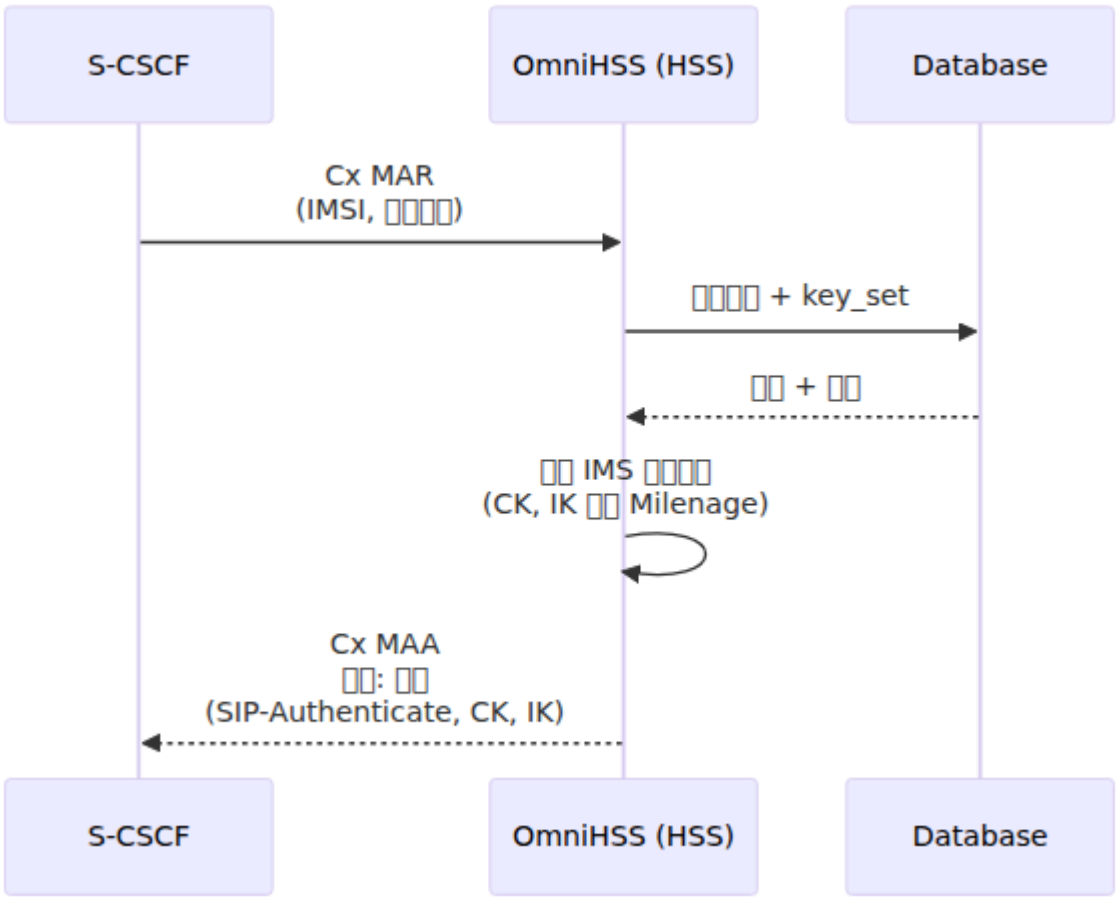
□□□□□□□ (PUR/PUA)

□□□□□□□□□□□□□□□□□□□□


□□□□□□
OmniHSS (HSS)
□□□
Sh PUR
(□□□□□, □□□□□)
□□ subscriber_state
(sh_repository_data)
□□□
Sh PUA
□□: □□
□□□□□□
OmniHSS (HSS)
□□□

□□□□□□□ (SNR/SNA)

□□□□□□□□□□□□□□□



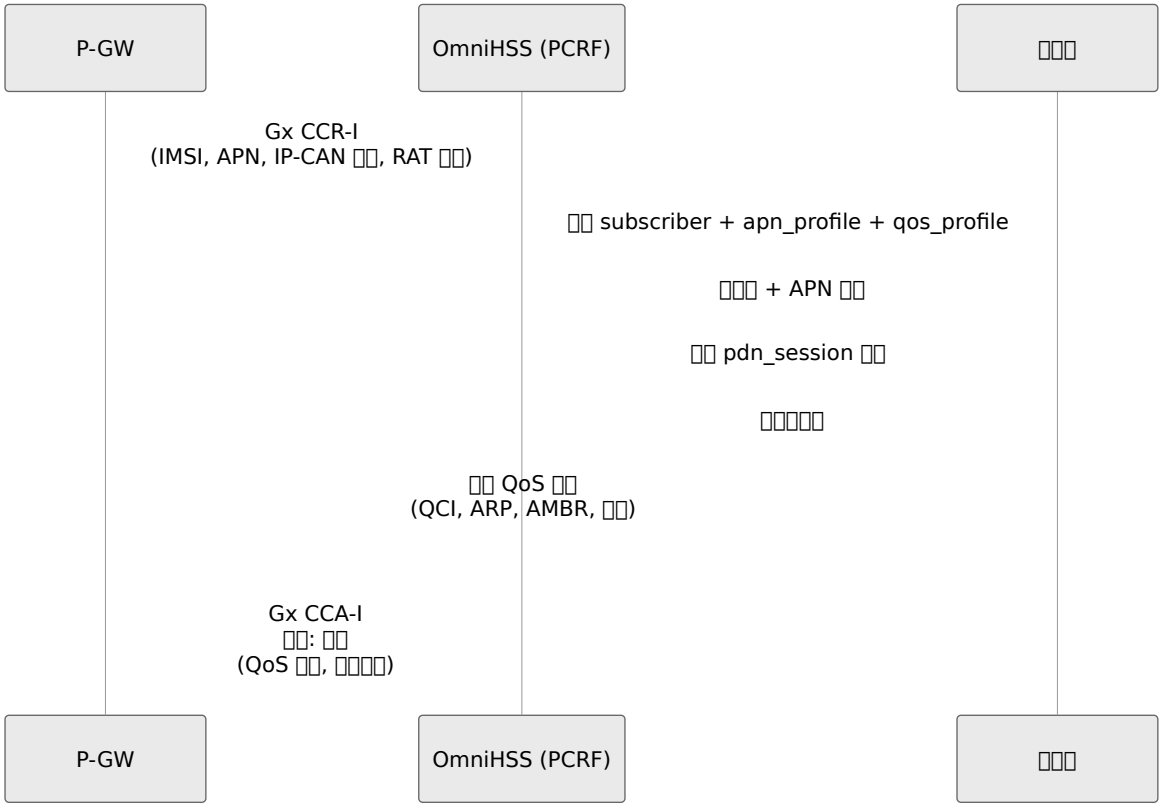
Gx 接口（策略控制）

OmniHSS 通过 Gx 接口作为 PCRF（策略和计费规则功能）运行。

请参见 **PCRF** 文档以获取详细的架构、策略配置和 **QoS** 管理。

信用控制请求 - 初始 (CCR-I/CCA-I)

P-GW 在 PDN 会话建立时请求策略规则。

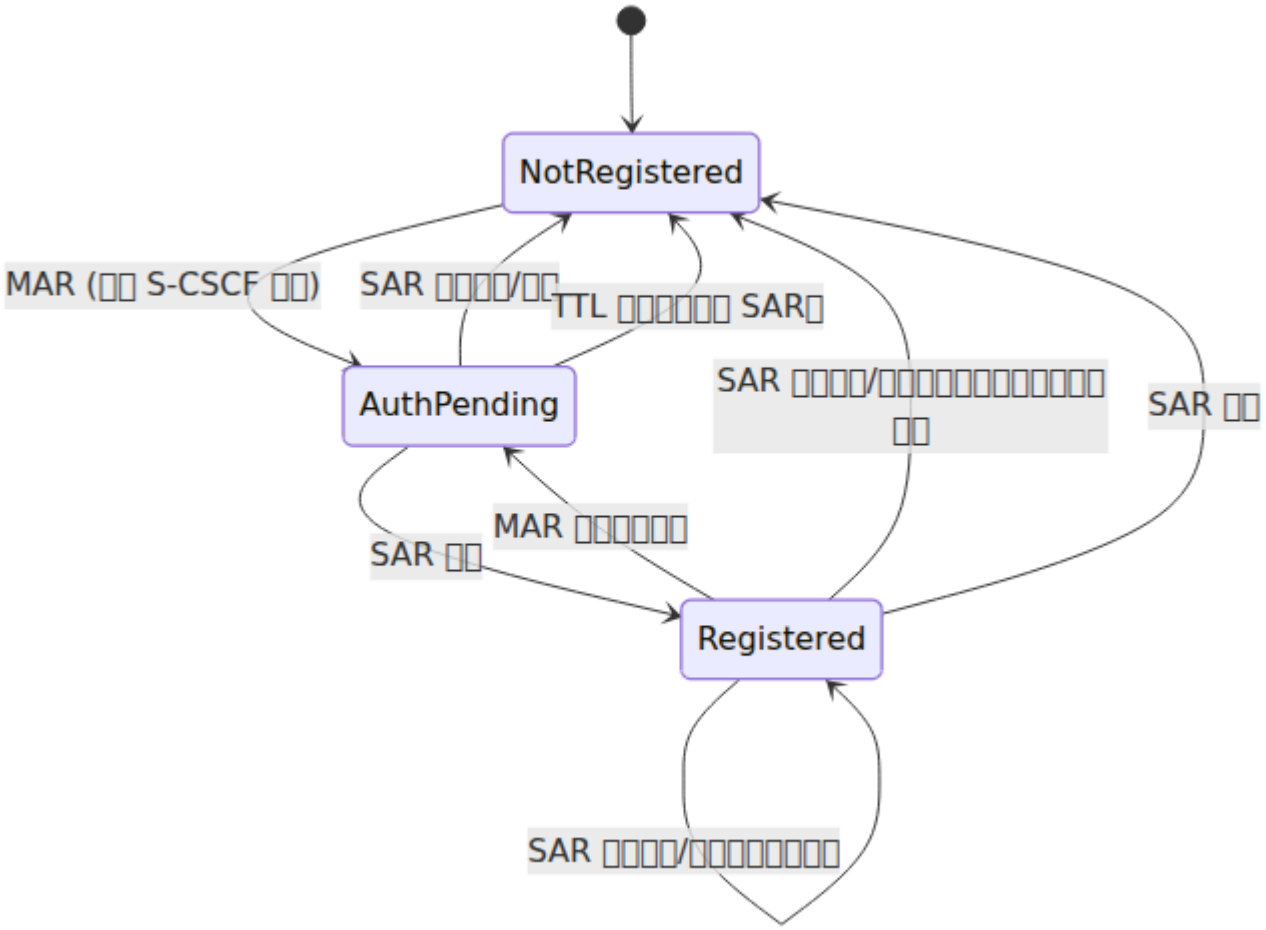


□□ AVP:

- □□: Subscription-Id (IMSI), Called-Station-Id (APN), RAT-Type, IP-CAN-Type
- □□: QoS-Information (QCI, ARP, AMBR), Charging-Rule-Install

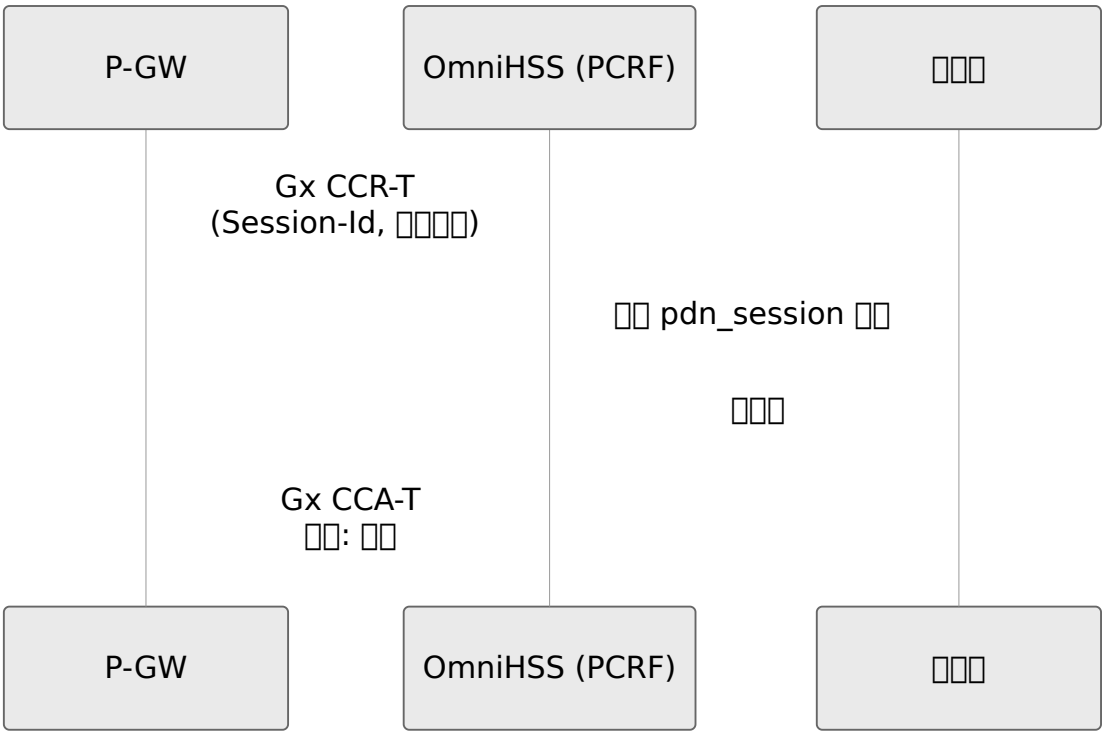
□□□□□□□ - □□ (CCR-U/CCA-U)

P-GW □□□□□□□□



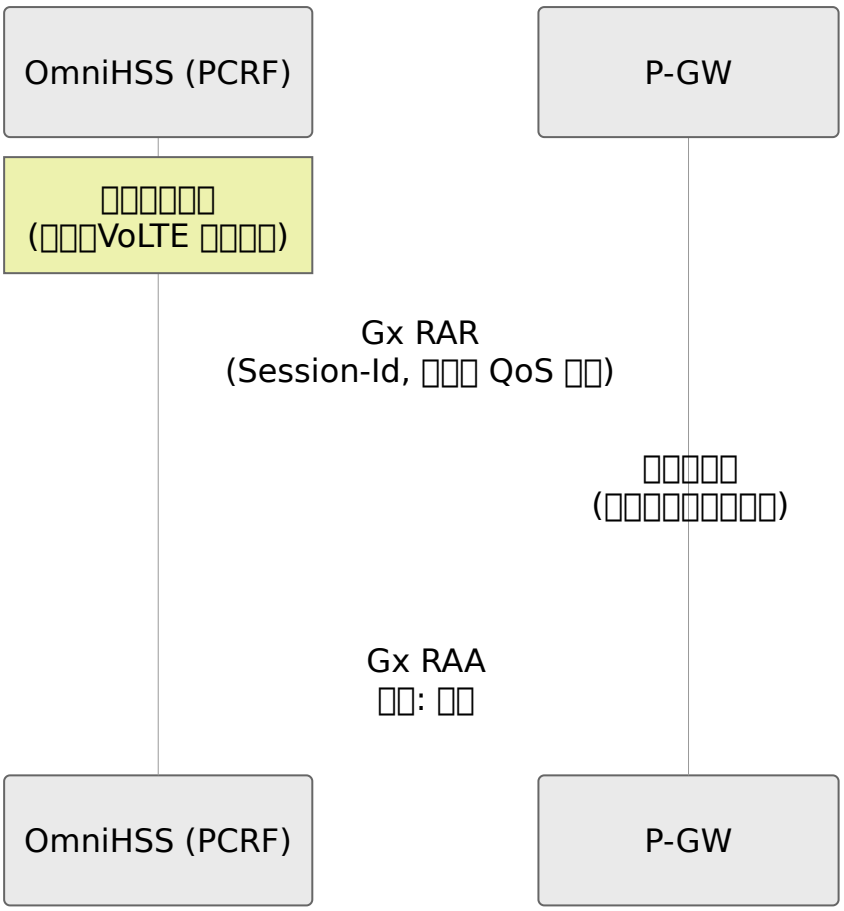
信用控制请求 - 终止 (CCR-T/CCA-T)

P-GW 通知 PDN 会话结束。



□□□□□□□ (RAR/RAA)

OmniHSS (PCRF) □ P-GW □□□□□□□□



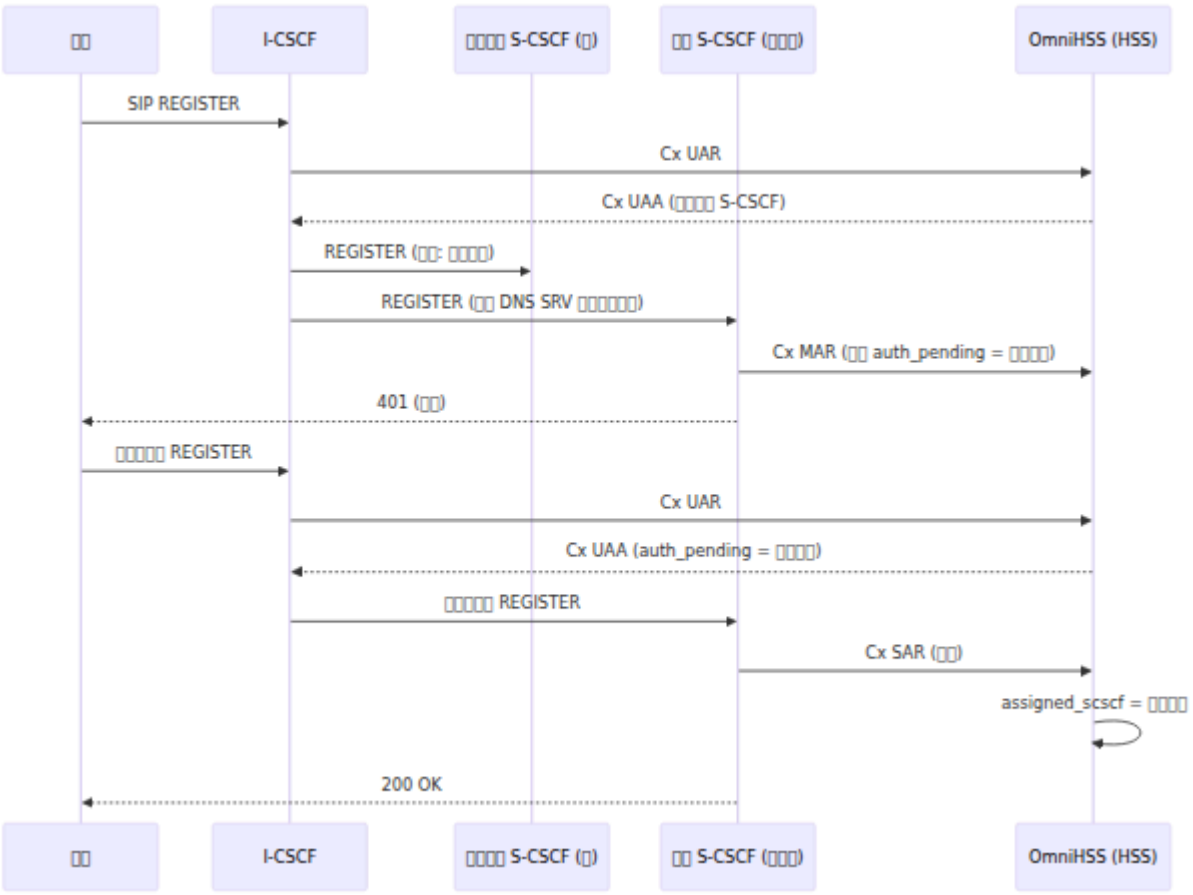
Rx □□ (IMS □□□□)

OmniHSS □□ Rx □□□□ PCRF □□□□□□ IMS □□□□□

□□□ **PCRF □□** □□□□□□□ **VoLTE** □□□□□□□□□□□

AA □□ (AAR/AAA)

P-CSCF □□ IMS □□□□□□□□

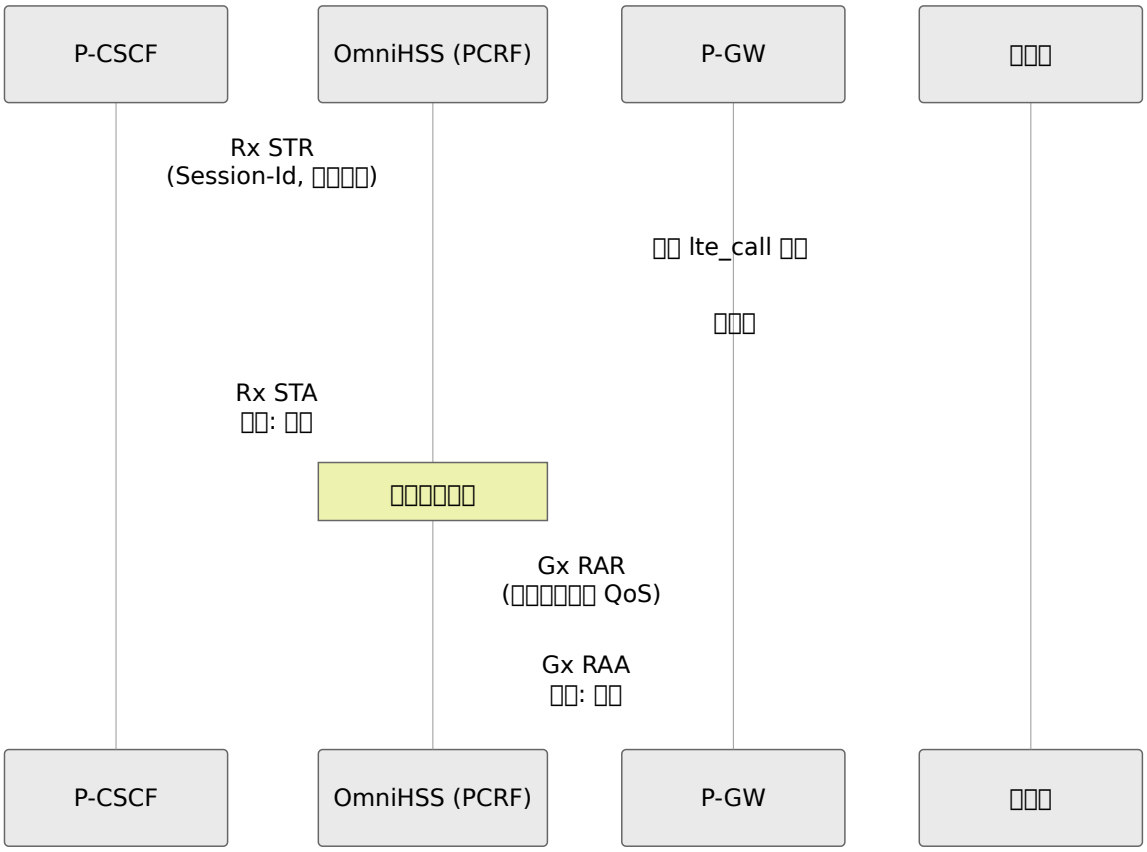


关键信息：

- 解析 SDP 以确定编解码器和带宽
- 计算所需带宽（上行/下行）
- 为媒体流创建 SDF 过滤器
- 通过 Gx RAR 触发专用承载

会话终止请求 **(STR/STA)**

P-CSCF 通知 IMS 会话结束。



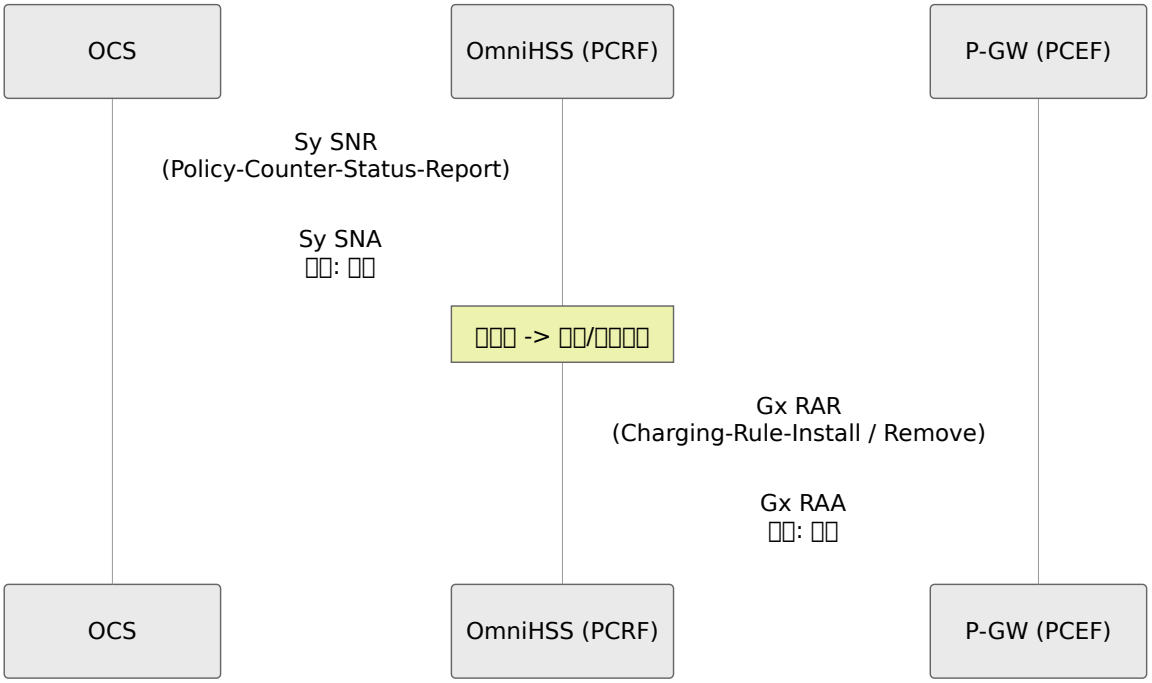
Sy 接口（支出限制控制）

OmniHSS 作为 PCRF，通过 Sy 接口从外部 OCS（在线计费系统）获取订阅者支出限制状态，并在 Gx 上应用基于支出的策略。Sy 使用直径应用 ID `16777302`（3GPP TS 29.219）。HSS 是 Sy 客户端；OCS 是服务器。请参见 Sy 接口 以获取完整参考（策略计数器映射、AVP、配置）。

支出限制请求（SLR/SLA）

PCRF 打开每个订阅者的 Sy 会话以获取 OCS 的策略计数器的当前状态，然后将结果规则折叠到它已经返回的 Gx CCA 中。

Parse error on line 9: ...HSS: 通过计数器标识符解析计费规则;
根据状态安装/移除 H ----------^ Expecting 'SPACE', 'NEWLINE', 'create', 'box', 'end', 'autonumber', 'activate', 'deactivate', 'title', 'legacy_title', 'acc_title', 'acc_descr', 'acc_descr_multiline_value', 'loop', 'rect', 'opt', 'alt', 'par', 'par_over', 'critical', 'break', 'option', 'and', 'else', 'participant', 'participant_actor', 'destroy', 'note', 'links', 'link', 'properties', 'details', 'ACTOR', got 'INVALID'



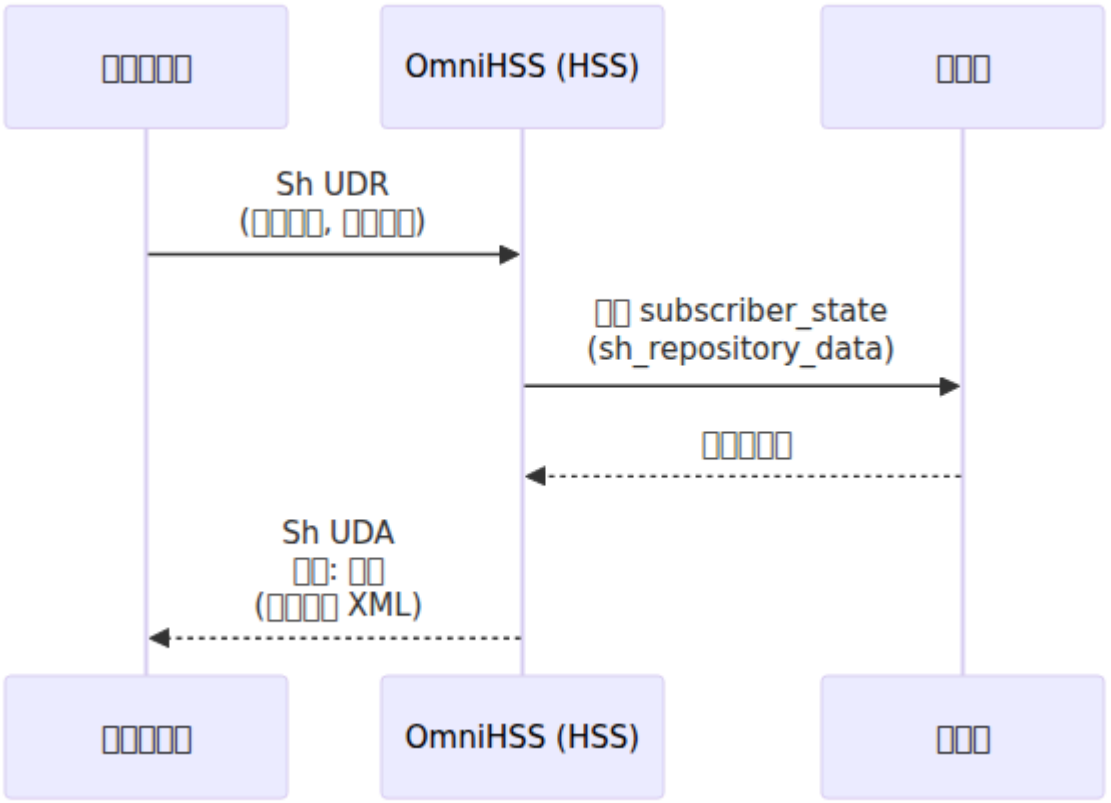
□试

支出状态通知 **(SNR/SNA)**

当计数器状态变化时（例如数据配额耗尽），OCS 通知 PCRF，PCRF 通过 Gx RAR 将策略更改推送到 PGW。

会话终止 **(STR/STA)**

当订阅者的最后一个数据会话结束时，PCRF 关闭 Sy 会话。



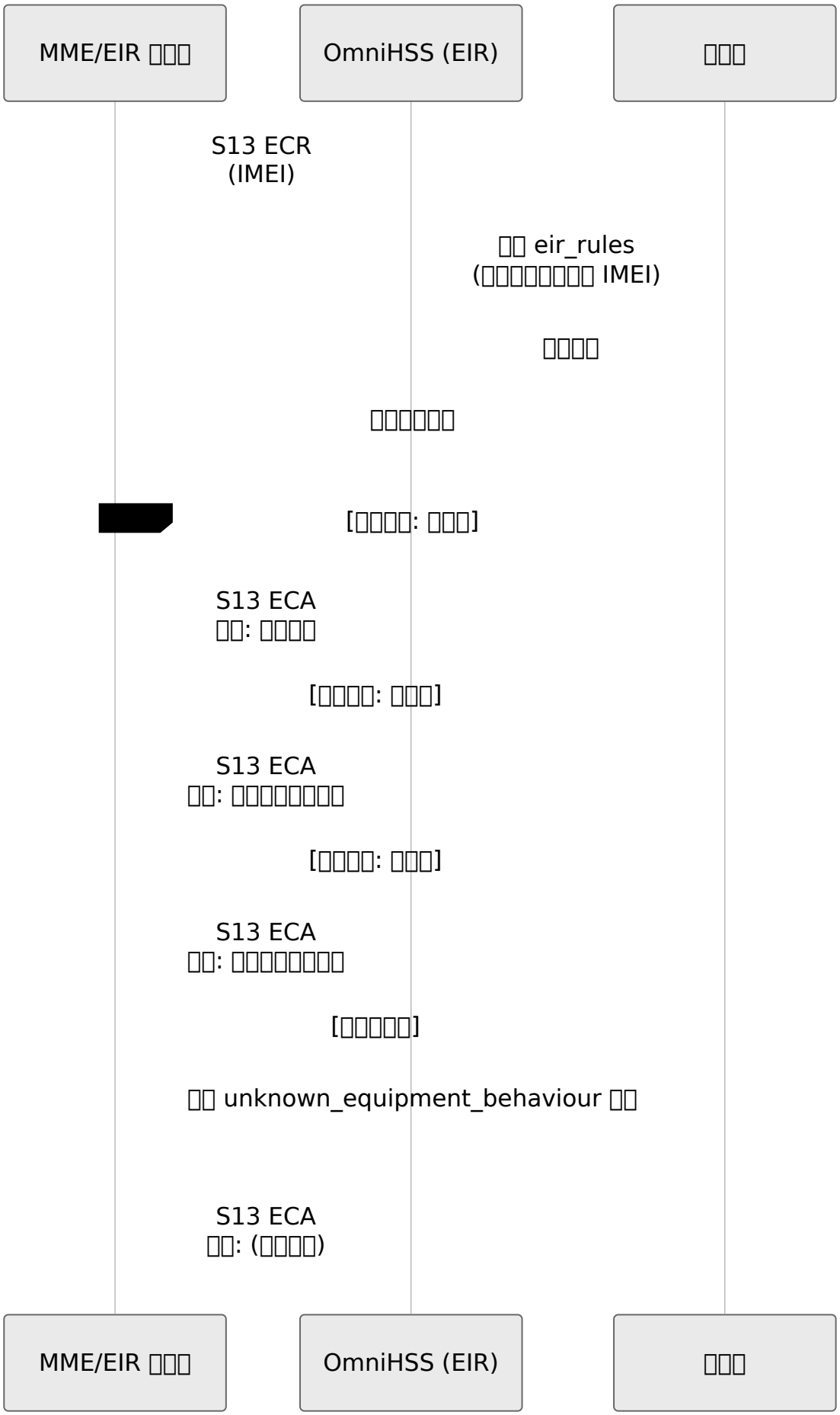
S13 接口 (EIR)

OmniHSS 通过 S13 接口作为 EIR（设备身份注册）运行。

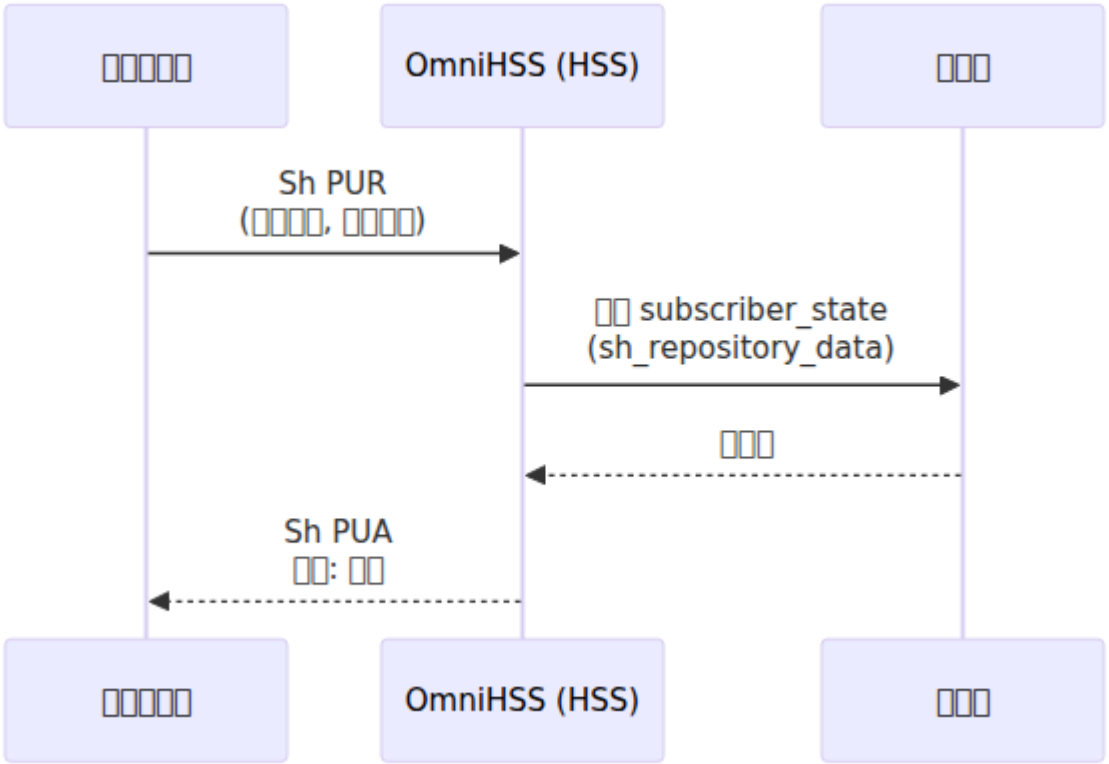
请参见 **EIR** 文档 以获取详细的设备身份检查、**IMEI** 验证和黑名单管理。

ME 身份检查请求 (ECR/ECA)

外部 EIR 客户端（或 MME）请求设备验证。



MME/EIR 客户端
OmniHSS (EIR)
数据库
S13 ECR
(IMEI)
查询 eir_rules
(根据正则规则匹配 IMEI)
匹配规则
评估规则操作
[规则匹配：白名单]
S13 ECA
结果：设备未知
[规则匹配：黑名单]
S13 ECA
结果：设备被列入黑名单
[规则匹配：灰名单]
S13 ECA
结果：设备被列入灰名单
[无规则匹配]
应用 unknown_equipment_behaviour 配置
S13 ECA
结果：(根据配置)
MME/EIR 客户端
OmniHSS (EIR)
数据库



设备状态值：

- 设备未知 **(0)** - 设备允许（白名单）
- 设备被列入黑名单 **(1)** - 设备被阻止
- 设备被列入灰名单 **(2)** - 设备允许但被跟踪

SLh 接口 (LCS / 位置服务)

OmniHSS 回应 SLh `LCS-Routing-Info-Request` (RIR)，以便 GMLC 在通过 SLg 请求其位置之前发现当前为订阅者提供服务的 MME。提供服务的 MME 是在 S6a 注册（更新位置）时记录的。SLh 使用直径应用 ID `16777291` (3GPP TS 29.173)。

LCS 路由信息请求 (RIR/RIA)

GMLC 请求订阅者的服务节点。

注意：

- IMSI 取自 `用户名称`；MSISDN 查找用作后备。

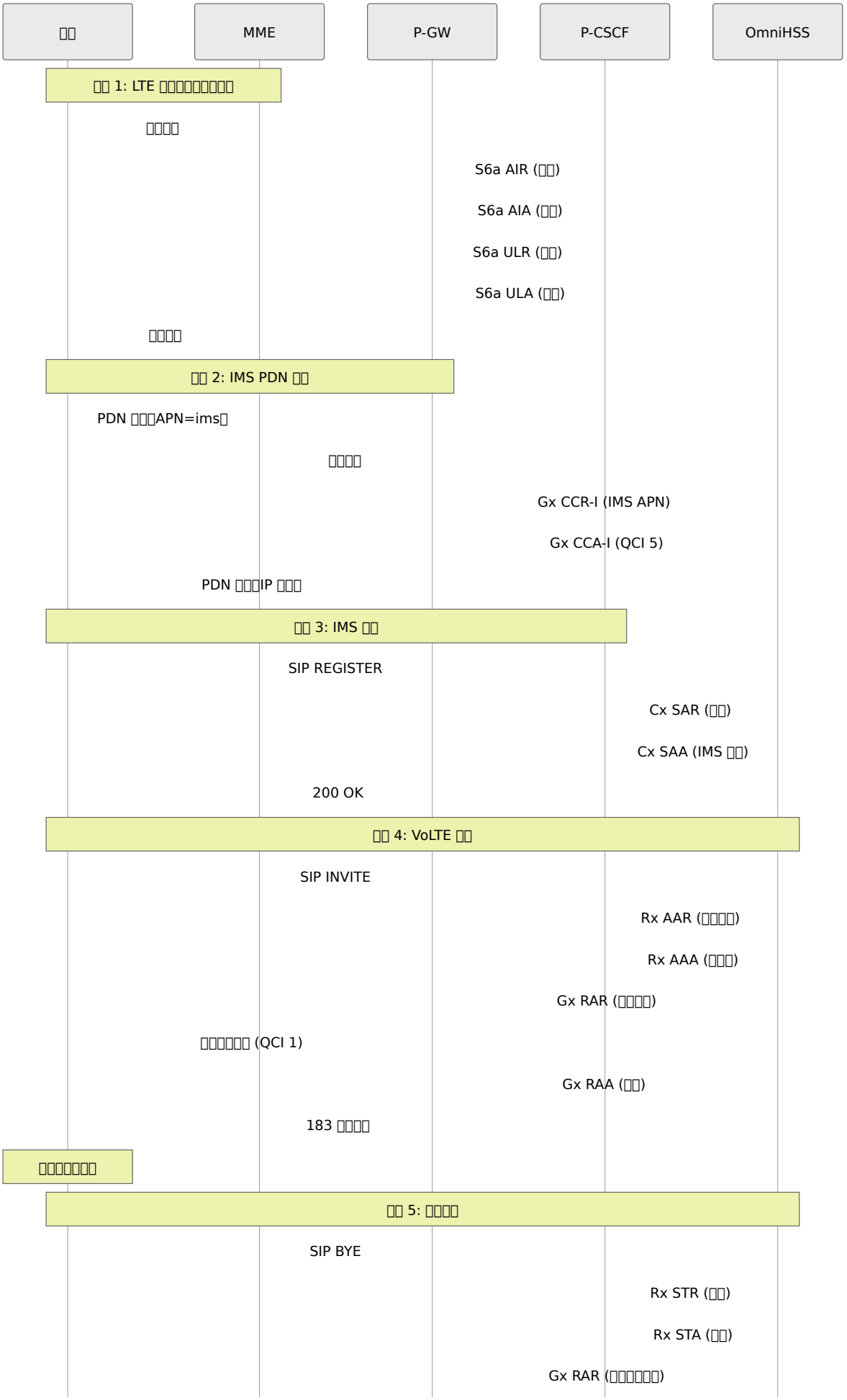
返回的 服务节点 携带 S6a AIR/ULR 处理程序最后记录的 MME 名称/ MME 域（subscriber_state.last_seen_mme / last_seen_realm） - 无需额外配置。
GMLC 通常在成功的 RIA 后跟随 SLg Provide-Location-Request 到返回的 MME。

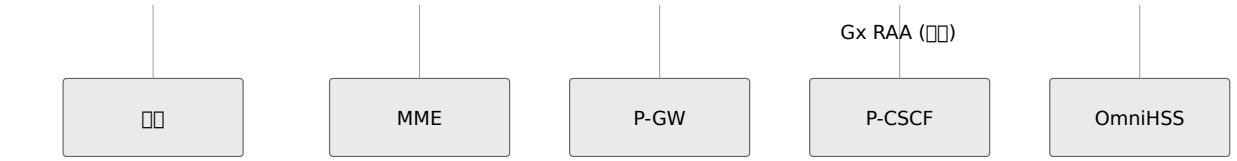
结果代码:

2001 - 成功；返回 服务节点。
5001 - DIAMETER_ERROR_USER_UNKNOWN；IMSI/MSISDN 未配置。
4201 - DIAMETER_ERROR_ABSENT_USER；已知订阅者但没有记录的服务 MME（从未附加，或位置已取
消）。

完整呼叫流程: VoLTE 呼叫

端到端 VoLTE 呼叫设置，显示多个接口。


UE
MME
P-GW
P-CSCF
OmniHSS
阶段 1: LTE 附着和默认承载建立
附着请求
S6a AIR (请求)
S6a AIA (响应)
S6a ULR (请求)
S6a ULA (响应)
附着接受
阶段 2: IMS PDN 连接
PDN 连接请求(APN=ims)
创建会话
Gx CCR-I (IMS APN)
Gx CCA-I (QCI 5)
PDN 连接接受(IP 地址)
阶段 3: IMS 注册
SIP REGISTER
Cx SAR (请求)
Cx SAA (IMS 用户)
200 OK
阶段 4: VoLTE 呼叫
SIP INVITE
Rx AAR (媒体协商)
Rx AAA (成功)
Gx RAR (专用承载)
专用承载激活 (QCI 1)
Gx RAA (成功)
183 会话进度
被叫振铃并接听
阶段 5: 呼叫释放
SIP BYE
Rx STR (终止)
Rx STA (成功)
Gx RAR (释放专用承载)

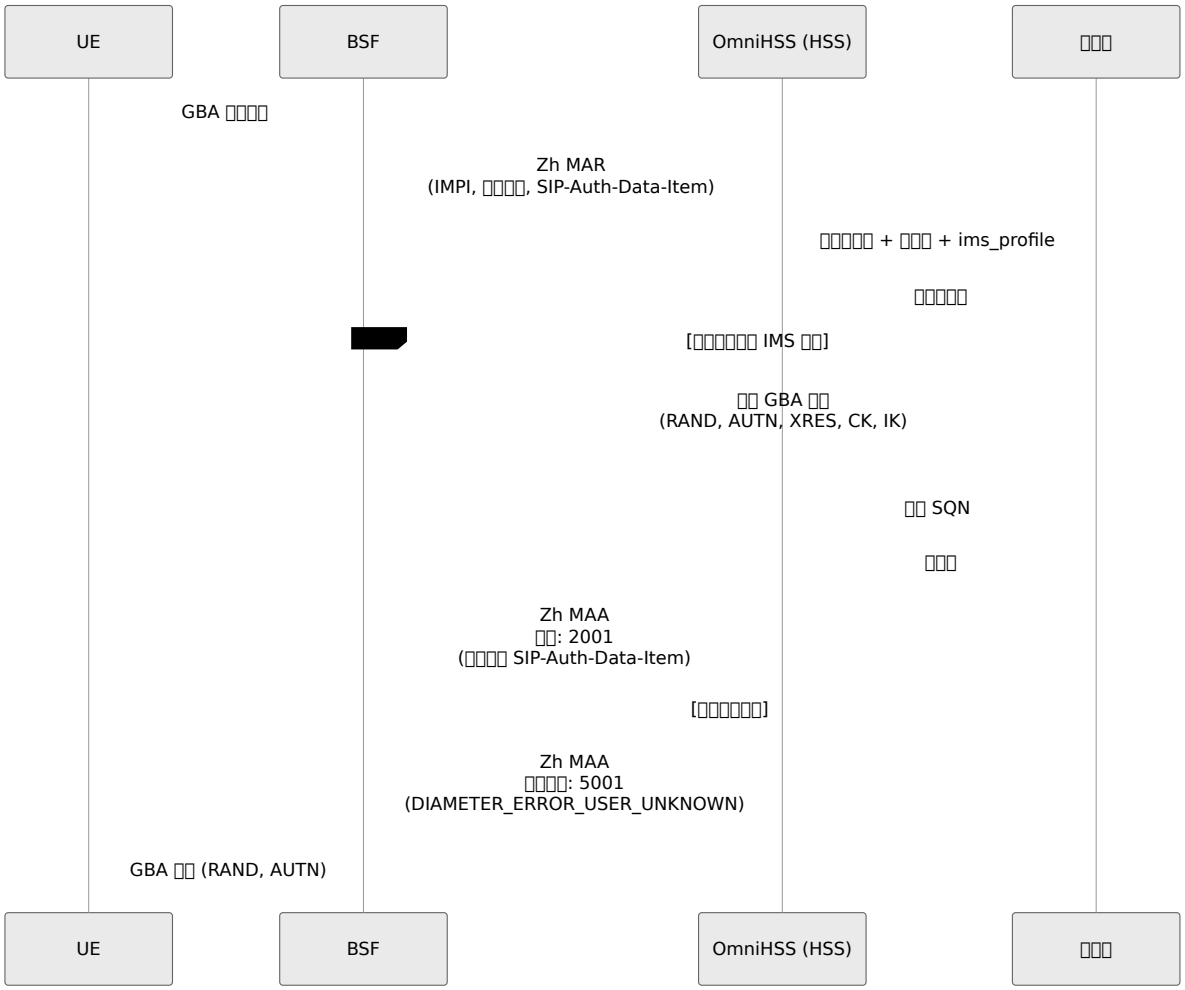


Zh □□ (□□□□□□□)

Zh □□□□□□□□□□□□□□□BSF□□□□□ HSS □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (GBA)□

□□□□□□□□ (MAR/MAA)

BSF □□ GBA □□□□□□□□□□□□



□□ AVP:

- □□: User-Name (IMPI), Public-Identity, SIP-Auth-Data-Item (□□□□), Destination-Realm

- 响应: SIP-Auth-Data-Item (SIP-Authenticate, SIP-Authorization, Confidentiality-Key, Integrity-Key), GBA-UserSecSettings

GBA 使用案例**:**

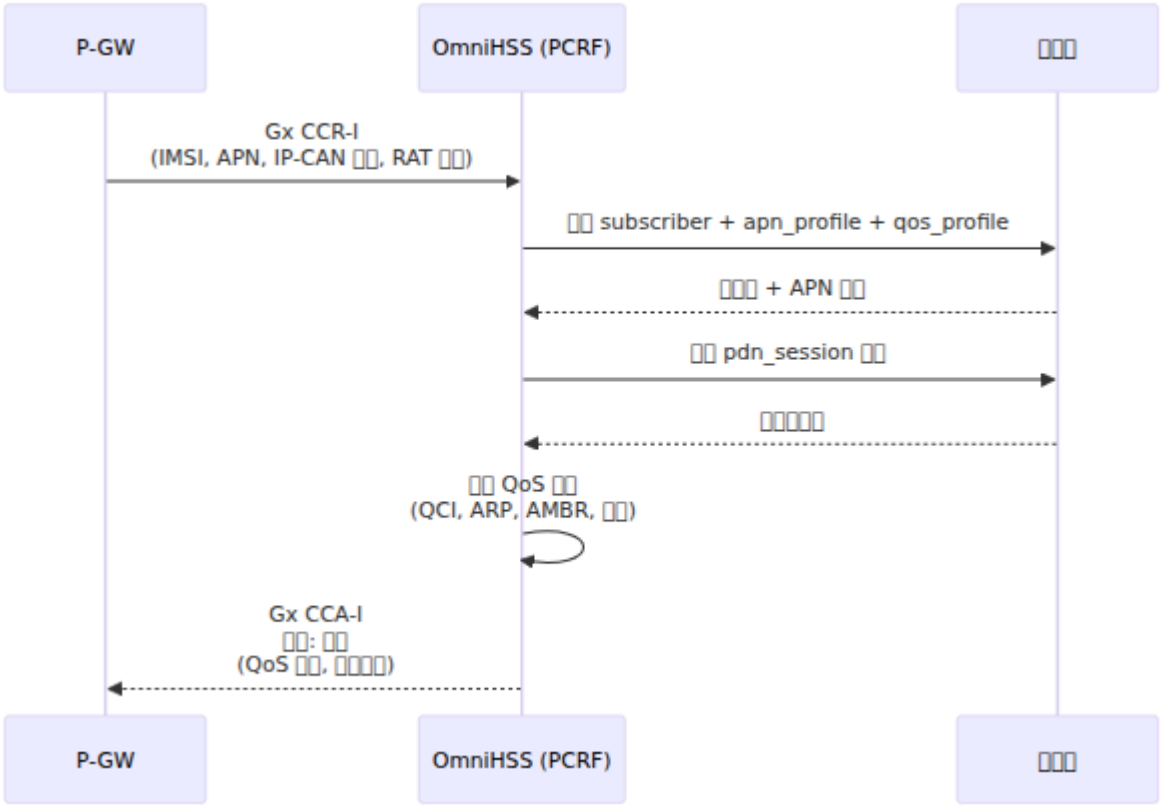
应用程序	描述
MBMS	移动广播多播服务密钥管理
HTTPS	基于 GBA 的证书配置
NAF	通用网络应用功能认证
Presence	安全的在线状态服务认证

SWm 接口（不受信的非 **3GPP** 访问）

SWm 接口通过将 3GPP AAA 服务器连接到 HSS，为不受信的非 3GPP 访问网络（例如，通过 ePDG 的 WiFi）启用 EAP-AKA 认证。

Diameter-EAP-Request (DER/DEA)

WiFi 访问的 EAP 认证交换。

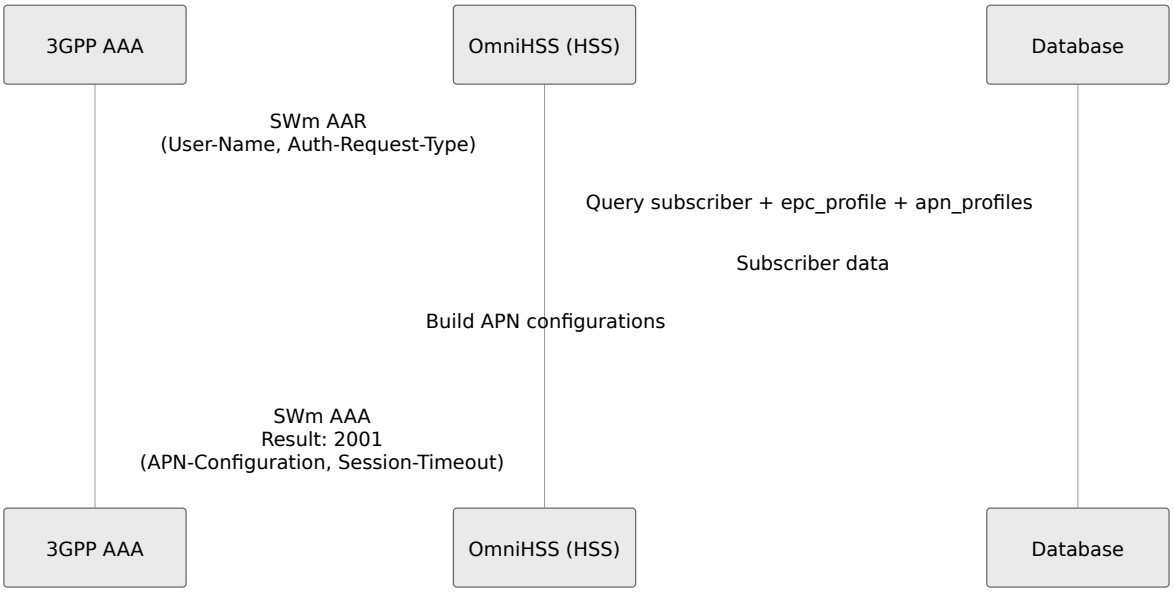


关键 **AVP:**

- 请求: Session-Id, Auth-Application-Id, EAP-Payload, Auth-Request-Type, User-Name (可选)
- 响应: EAP-Payload, EAP-Master-Session-Key (MSK), APN-Configuration, Session-Timeout

AA-Request (AAR/AAA)

成功 EAP 认证后的授权。

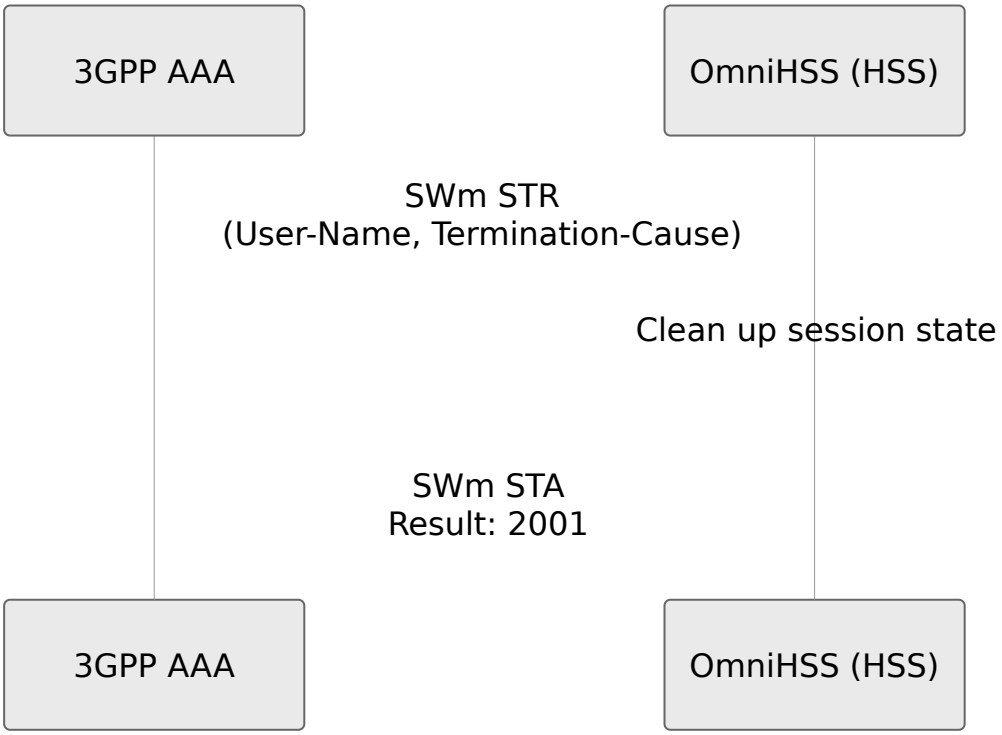


□□ AVP:

- □□: Session-Id, Auth-Application-Id, User-Name, Auth-Request-Type, AAR-Flags
- □□: APN-Configuration, 3GPP-Charging-Characteristics, Session-Timeout

Session-Termination-Request (STR/STA)

WiFi □□□□□□□



终止原因：

原因	值	描述
DIAMETER_LOGOUT	1	用户注销
DIAMETER_SERVICE_NOT_PROVIDED	2	服务不可用
DIAMETER_BAD_ANSWER	3	协议错误
DIAMETER_ADMINISTRATIVE	4	管理员终止
DIAMETER_LINK_BROKEN	5	连接丢失
DIAMETER_AUTH_EXPIRED	6	认证超时
DIAMETER_USER_MOVED	7	用户移动到其他接入
DIAMETER_SESSION_TIMEOUT	8	会话超时

故障排除协议问题

认证失败 **(S6a AIR)**

检查：

1. 密钥集配置正确 (Ki, OPC, AMF)
2. SQN 同步（如果重复失败）
3. 漫游规则允许访问的网络

位置更新失败 **(S6a ULR)**

检查：

1. EPC 配置文件存在并已配置 APN

2. 数据服务���许漫游
3. MME 身份格式正确

IMS 注册失败（Cx SAR）

检查：

1. IMS 配置文件已分配给用户
2. IFC 模板有效 XML
3. S-CSCF 选择已配置
4. 如果在模板中使用，则分配 MSISDN

PDN 连接失败（Gx CCR-I）

检查：

1. APN 在 EPC 配置文件的 APN 列表中存在
2. APN QoS 配置文件已配置
3. PDN 会话表未满（如果存在限制）

← 返回操作指南

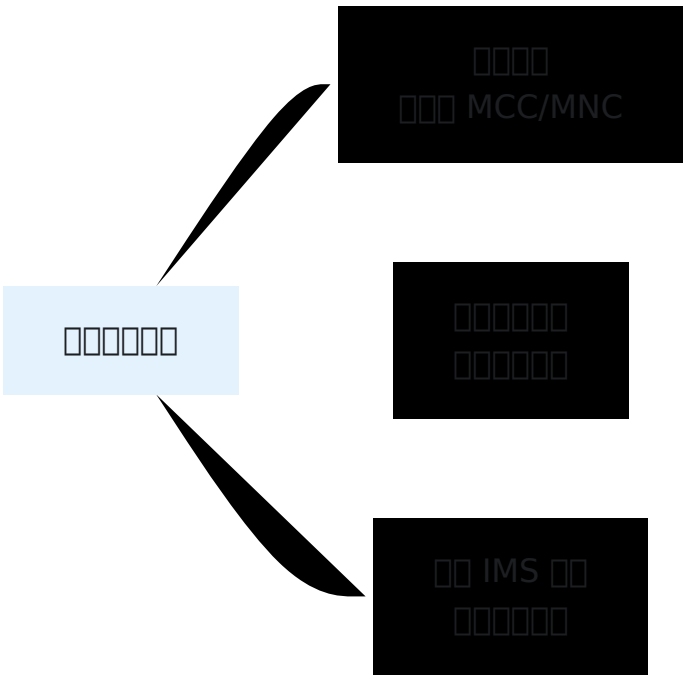
OmniHSS 漫游控制

← 返回操作指南

概述

OmniHSS 提供细粒度的漫游控制，允许您定义用户在漫游时可以访问哪些网络的数据信息和 IMS 服务。

漫游控制流程



漫游配置文件结构

组件

漫游规则

每条规则为特定网络（MCC/MNC 组合）指定操作。

字段：

- `name` - 描述性名称
- `mcc` - 移动国家代码（3 位数字）
- `mnc` - 移动网络代码（2-3 位数字）
- `data_action` - "allow" 或 "deny"
- `ims_action` - "allow" 或 "deny"

漫游配置文件

定义默认行为并链接到规则。

字段：

- `name` - 配置文件名称
- `data_action_if_no_rules_match` - "allow" 或 "deny"
- `ims_action_if_no_rules_match` - "allow" 或 "deny"

配置示例

允许所有漫游

```
# 创建允许所有内容的配置文件
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_profile": {
      "name": "允许所有",
      "data_action_if_no_rules_match": "allow",
      "ims_action_if_no_rules_match": "allow",
      "roaming_rules": []
    }
  }'
```

拒绝所有漫游

```
# 创建阻止所有内容的配置文件
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_profile": {
      "name": "不允许漫游",
      "data_action_if_no_rules_match": "deny",
      "ims_action_if_no_rules_match": "deny",
      "roaming_rules": []
    }
  }'
```

允许特定网络□白名单）

```
# 创建 AT&T 规则
RULE1=$(curl -k -X POST
https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "允许 AT&T",
      "mcc": "310",
      "mnc": "410",
      "data_action": "allow",
      "ims_action": "allow"
    }
  }' | jq -r '.response.id')

# 创建 Verizon 规则
RULE2=$(curl -k -X POST
https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "允许 Verizon",
      "mcc": "311",
      "mnc": "480",
      "data_action": "allow",
      "ims_action": "allow"
    }
  }' | jq -r '.response.id')

# 创建默认拒绝并链接规则的配置文件
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"roaming_profile\": {
      \"name\": \"仅限美国运营商\",
      \"data_action_if_no_rules_match\": \"deny\",
      \"ims_action_if_no_rules_match\": \"deny\",
      \"roaming_rules\": [$RULE1, $RULE2]
    }
  }"
```

允许数据，阻止语音

```
# 创建允许数据但阻止 IMS 的规则
curl -k -X POST https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "仅限数据 - T-Mobile",
      "mcc": "310",
      "mnc": "260",
      "data_action": "allow",
      "ims_action": "deny"
    }
  }'
```

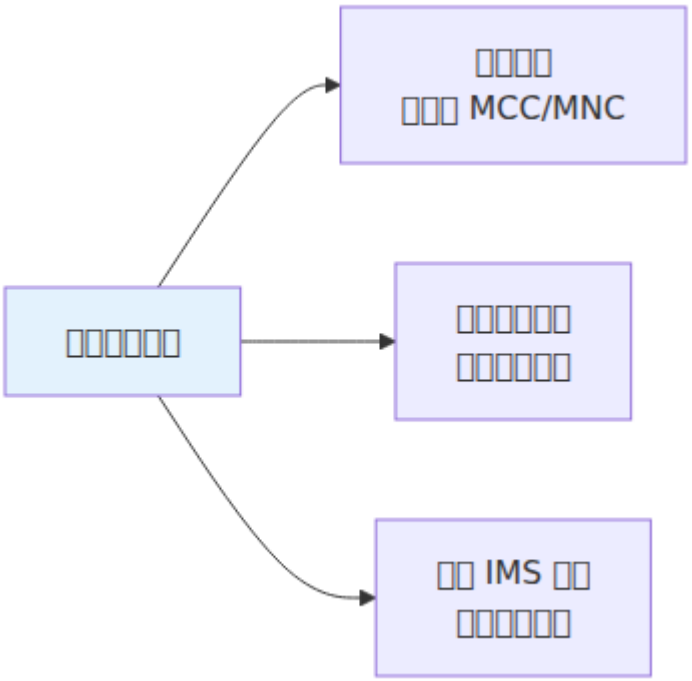
阻止特定网络（黑名单）

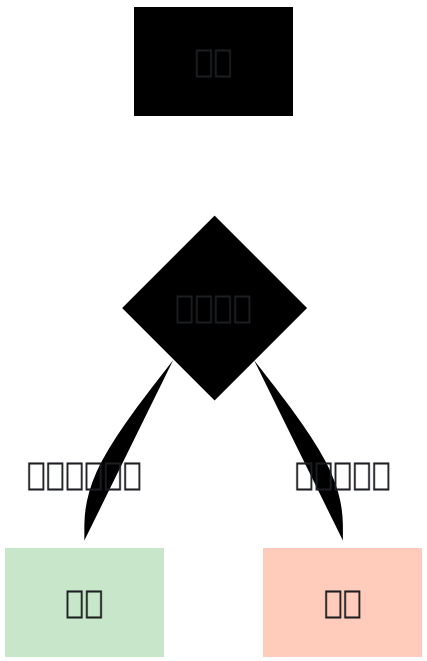
```
# 创建阻止昂贵网络的规则
RULE=$(curl -k -X POST
https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "阻止昂贵网络",
      "mcc": "206",
      "mnc": "01",
      "data_action": "deny",
      "ims_action": "deny"
    }
  }' | jq -r '.response.id')

# 创建默认允许的配置文件
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"roaming_profile\": {
      \"name\": \"阻止昂贵网络\",
      \"data_action_if_no_rules_match\": \"allow\",
      \"ims_action_if_no_rules_match\": \"allow\",
      \"roaming_rules\": [$RULE]
    }
  }"
```

常见漫游场景

场景 1：仅限国内漫游

用户可以在本国漫游，但不能国际漫游。





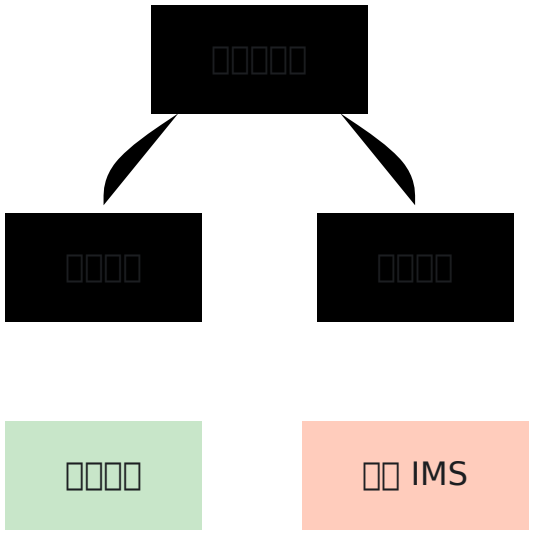
配置：

- 默认：拒绝所有
- 规则：允许所有美国 MCC 代码（310, 311, 312, 313, 314, 315, 316）

场景 2：仅限漫游合作伙伴

用户只能在与商业协议的网络上漫游。

配置：



- 默认：拒绝所有
- 规则：明确允许每个合作伙伴网络（按 MCC/MNC）

场景 3：数据漫游，无语音漫游

用户可以在国外使用数据，但必须使用 WiFi 进行语音通话。

配置：

- 规则：`data_action: "allow"`，`ims_action: "deny"`

场景 4：紧急服务访问

始终允许紧急服务，即使漫游被阻止。

注意：紧急服务处理通常在 MME/网络级别进行。OmniHSS 漫游规则适用于正常服务。

MCC/MNC 参考

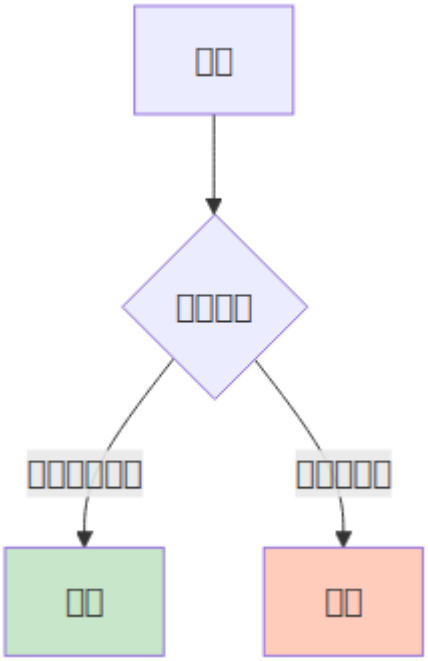
常见国家代码（MCC）

MCC	国家	网络
310-316	美国	AT&T, Verizon, T-Mobile 等
302	加拿大	Rogers, Bell, Telus
234-235	英国	Vodafone, O2, EE
262	德国	Deutsche Telekom, Vodafone
208	法国	Orange, SFR, Bouygues
222	意大利	TIM, Vodafone, Wind
214	西班牙	Movistar, Vodafone

常见美国运营商（MCC 310-316）

MCC	MNC	运营商
310	410	AT&T
311	480	Verizon
310	260	T-Mobile
310	120	Sprint
313	380	（示例测试网络）

完整列表：请参见 ITU-T E.212 或 MCC/MNC 数据库



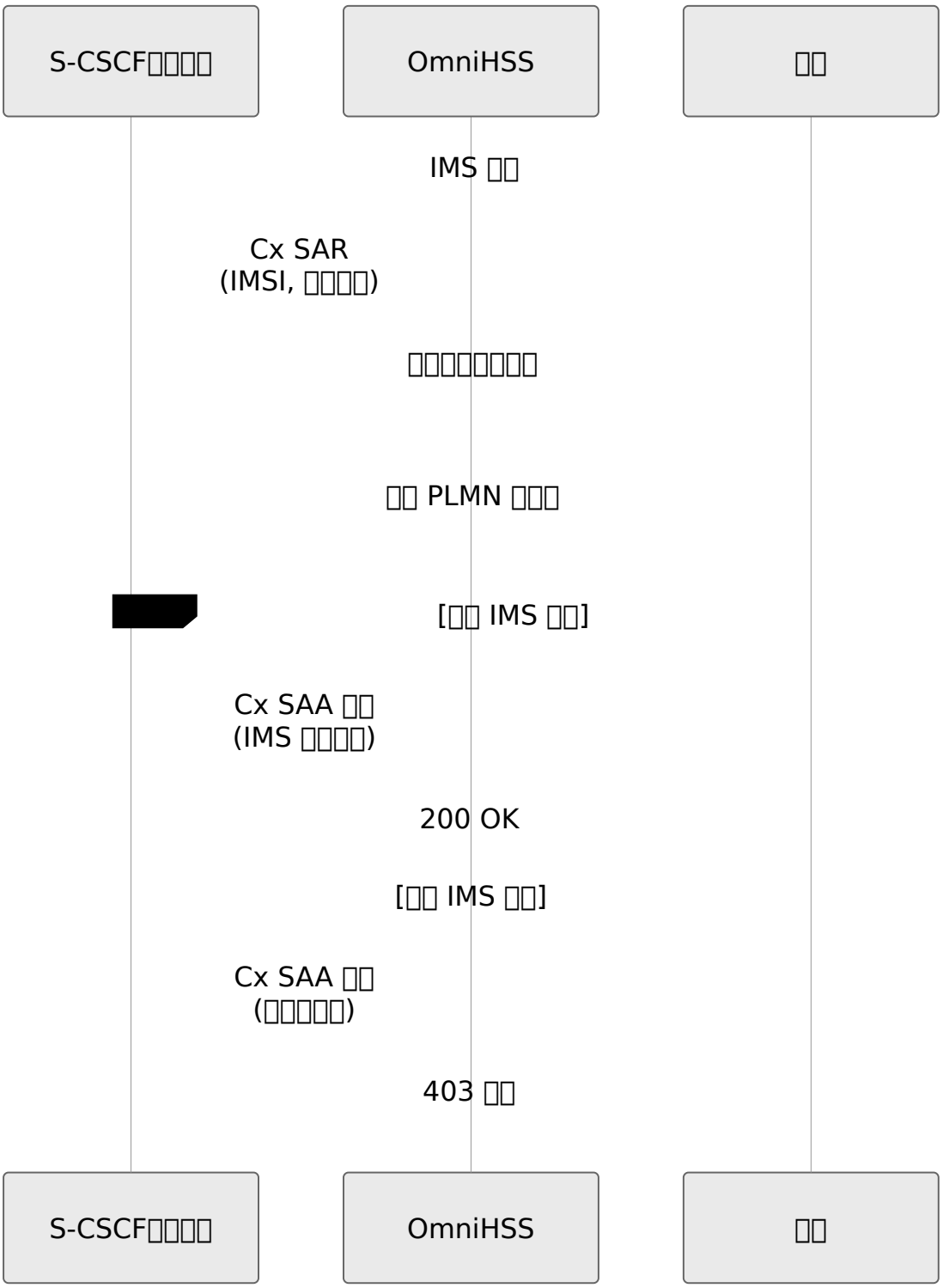
漫游执行点

S6a 接口（数据）

当用户附加到访问网络时：

Cx 接口（IMS）

当用户在访问网络中注册 IMS 时：



S-CSCF (访问)
OmniHSS
用户
IMS 注册
Cx SAR
(IMSI, 访问网络)
检查漫游配置文件
查找 PLMN 的规则
[允许 IMS 漫游]
Cx SAA 成功
(IMS 配置文件)
200 OK
[拒绝 IMS 漫游]
Cx SAA 错误
(不允许漫游)
403 禁止
S-CSCF (访问)
OmniHSS
用户

故障排除漫游问题

用户无法在访问网络中附加

检查漫游配置文件分配：

- 查询数据库以查看用户分配的漫游配置文件
- 验证配置文件名称和默认操作设置

检查访问网络是否存在规则：

- 查询数据库以查找与访问网络的 MCC/MNC 匹配的漫游规则
- 验证用户的漫游配置文件中是否存在规则
- 检查该特定网络的 data_action 值

用户可以附加但无法注册 IMS

单独检查 **IMS** 操作：

- 查询访问网络的漫游规则
- 验证 data_action 和 ims_action 值
- 查找数据被允许但 IMS 被拒绝的情况

意外的漫游行为

查看漫游检查的日志：

```
[info] 漫游检查: IMSI 001001123456789, 访问 PLMN 310-410
[info] 匹配的漫游规则: "允许 AT&T"
[info] 数据操作: 允许, IMS 操作: 允许
```

最佳实践

配置文件设计

1. 从限制开始 - 默认拒绝，明确允许合作伙伴
2. 彻底测试 - 在生产前验证实验室中的规则
3. 记录规则 - 保持允许网络的列表及原因
4. 定期审查 - 随着漫游协议的变化进行更新

规则管理

1. 使用描述性名称 - "Allow-ATT-Data-Only" 而不是 "Rule1"
2. 验证 MCC/MNC - 对照官方数据库仔细检查代码
3. 考虑两种服务 - 分别考虑数据和 IMS
4. 监控使用情况 - 跟踪用户实际访问的网络

操作程序

1. 紧急变更 - 制定快速启用/禁用漫游的程序
2. 批量更新 - 计划更新多个用户的漫游配置文件
3. 报告 - 跟踪漫游使用情况和拒绝尝试
4. 客户沟通 - 通知客户漫游政策的变更

← 返回操作指南 | 下一步：协议流程 →

Sh RepositoryData (透明数据)
OmniHSS 作为服务数据的 透明存储库，用于应用服务器 (AS) 在 Sh 接口 上针对用户存储的数据。诸如 MMTEL-Services（补充服务/呼叫转移、呼叫等待、OIP/OIR、限制）和IMS-ODB-Information（运营商确定的限制）等文档由 AS 编写，由 HSS 保存，并根据需要读取。

“透明”在 3GPP TS 29.328 中有明确的含义：HSS 仅 语法上理解 这些数据，而 语义上不理解。它是由 AS 拥有的不透明 XML。OmniHSS 以 逐字节 的方式存储和返回 ServiceData 有效负载。它从不重新解析或重新序列化有效负载。AS 生成的文档的 etags 或校验和（例如 XCAP/MMTel）将按原样返回。

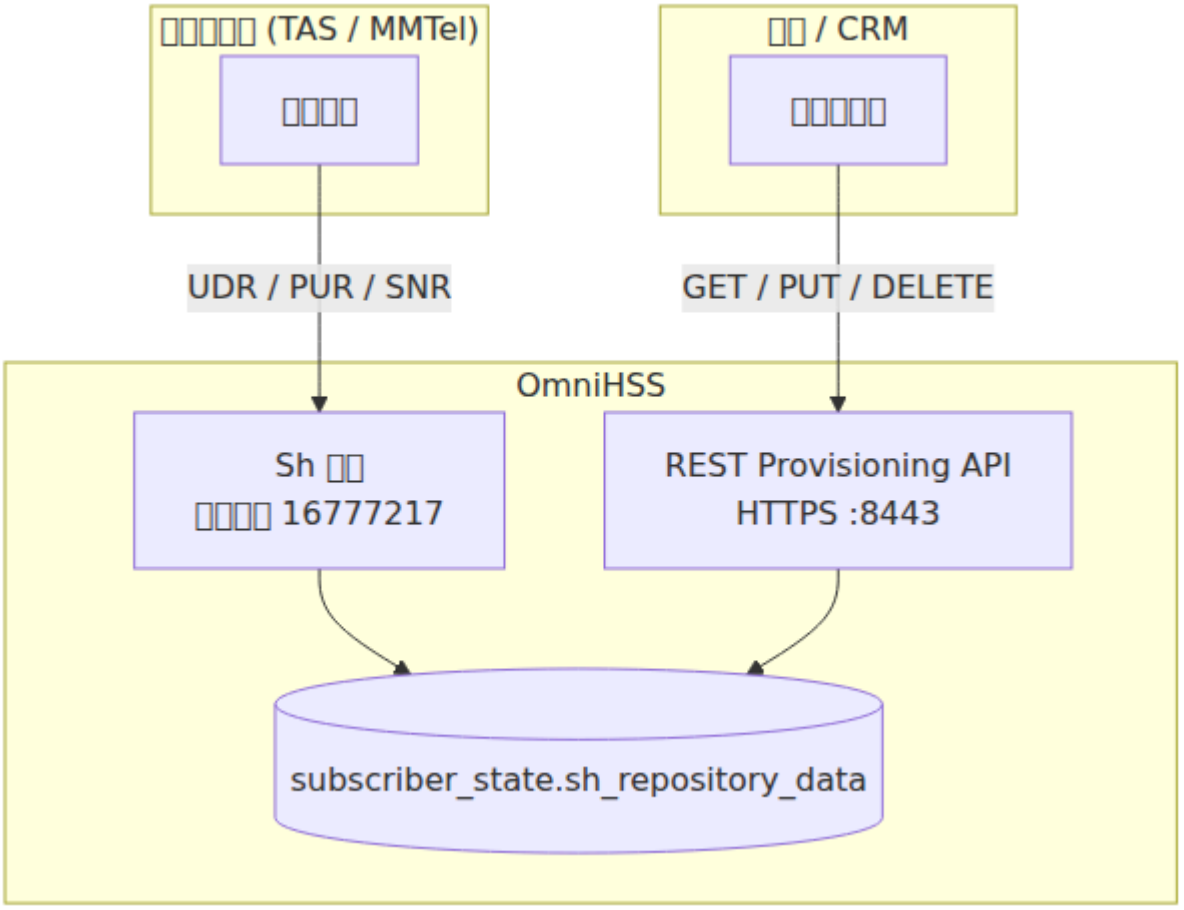
相同的文档也可以通过 REST Provisioning API 由配置/CRM 系统进行带外管理。

目录

概念
数据模型
Sh 消息流
SequenceNumber 和并发
REST Provisioning API
参考表
故障排除

概念

一个用户可以持有任意数量的独立 透明文档，每个文档由 Service-Indication（“文件名”）标识。文档有三个部分：



元素	拥有者	意义
Service-Indication	HSS 管理	文档名称（例如 `MMTEL-Services`，`IMS-ODB-Information`）。
SequenceNumber	HSS 管理	用于乐观并发的版本计数器。请参见 SequenceNumber 和并发。
ServiceData	AS 拥有（不透明）	XML 有效负载本身。逐字存储和返回。

HSS 仅解释 Service-Indication 和 SequenceNumber。`ServiceData` 主体被视为不透明字节。

数据模型

用户的所有透明文档都保存在单个字段 `subscriber_state.sh_repository_data` 中，作为一个以 Service-Indication 为键的 JSON 对象。每个条目存储版本和逐字有效负载：

```
{
  "MMTEL-Services":      { "sequence_number": "0", "service_data":
"<MMTelServices …>…</MMTelServices>" },
  "IMS-ODB-Information": { "sequence_number": "1", "service_data":
"<OdbForImsOrientedServices xmlns=\"odb.mmtel.omnitou.ch\"/>" }
}
```

字段	描述
`sequence_number`	此文档的当前版本，最后被 HSS 接受。在每个 `UDA` 中返回，并用于验证更新。
`service_data`	不透明的 XML 主体，按 AS 发送的内容逐字存储（不进行命名空间、属性顺序或空白规范化）。没有 `ServiceData` 元素的文档（规范的 □□□□□□）省略此字段。

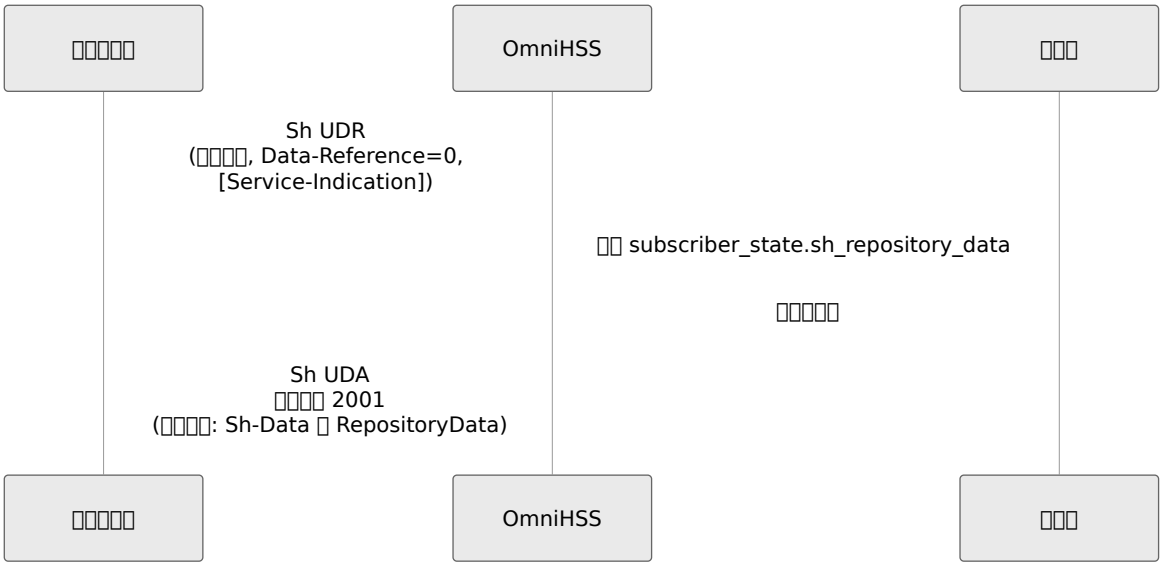
此模型意味着单个用户可以持有许多不同的文档，每个文档独立版本化，HSS 不施加固定的模式。

Sh 消息流

RepositoryData 作为 `Sh-Data` XML 文档□含在 `User-Data` AVP 中。通过 **Data-Reference** 值 **0** 选择 RepositoryData。可选的 **Service-Indication** AVP 将请求缩小到单个文档。

用户数据请求 (UDR/UDA)：读取

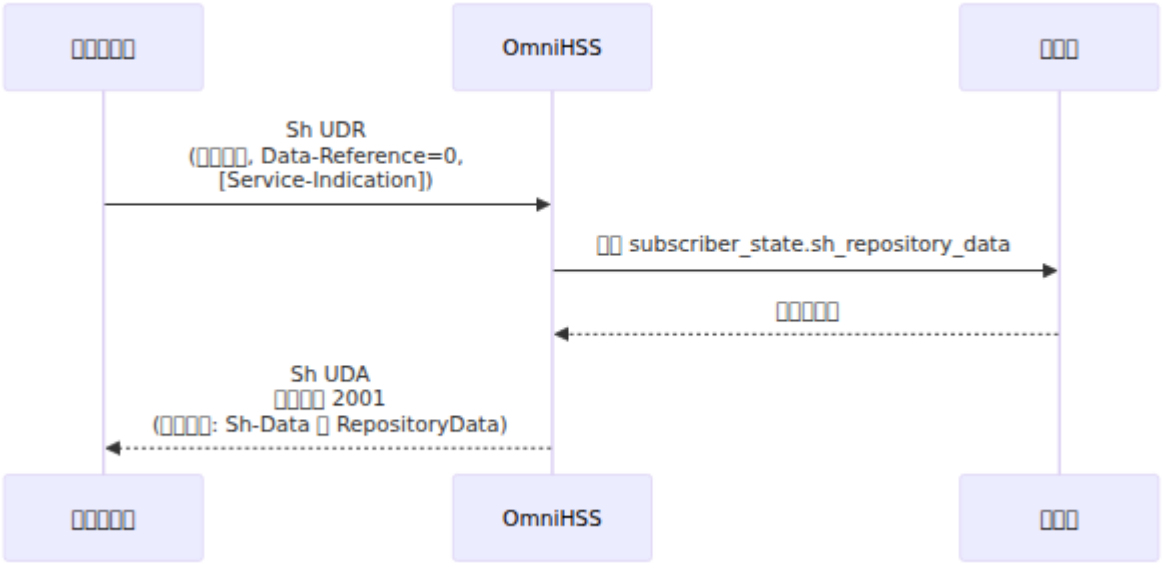
AS 读取□□□个或所有透明□□□档。带有 Service-Indication 时，仅返回匹配的文档；没有时，返回每个存储的文档，每个文档作为其自己的 `<RepositoryData>` 元素。



□□□□ IMS □□□□□□□ **Experimental-Result 5001 (DIAMETER_ERROR_USER_UNKNOWN)**□

□□□□□□□ (PUR/PUA)□□□

AS □□□□□□□□□□□`ServiceData` □□□□□□□□□□□□□ SequenceNumber □□□□□□□□□□□□□□□□□□□□□

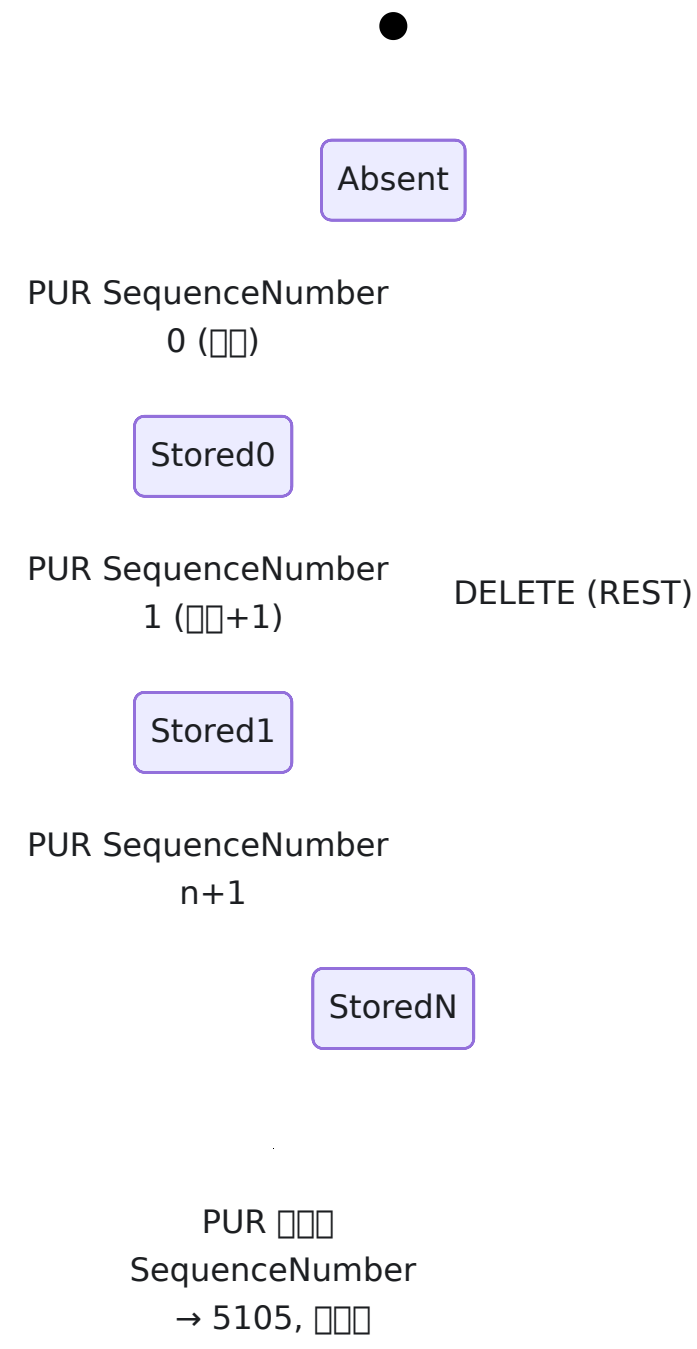


□ PUR □□□□□□□□□□□□□□□□□□□□

SequenceNumber 和并发

SequenceNumber 提供乐观并发控制，以便两个应用服务器不能静默覆盖彼此，遵循 3GPP TS 29.328。

- 创建 新文档使用 **SequenceNumber 0**（值 `0` 专门保留用于新添加的数据）。
- 更新 现有文档必须携带存储 SequenceNumber 的 直接后继（存储 + 1）。
- 计数器 从 **65535** 回绕到 **1**（而不是 0，因为 0 是保留的）。
- SequenceNumber 不是预期后继的更新将被拒绝，返回 **Experimental-Result 5105 (DIAMETER_ERROR_TRANSPARENT_DATA_OUT_OF_SYNC)**，并且存储的文档保持不变。



示例：一个文档存储在 SequenceNumber 0。携带 SequenceNumber 5 的 PUR 被拒绝，返回 5105，且没有任何变化。携带 SequenceNumber 1 的 PUR 被接受，文档版本提升至 1。

REST Provisioning API

相同的文档可以直接由配置/CRM 系统读取和写入。用户通过 IMSI 在路径中寻址；文档通过 service_indication 标识（在更新的请求体中，或在读取/删除的查询字符串中）。不透明的 service_data 逐字往返，正如通过 Sh 一样。

所有响应使用标准 API 信封：

```
{ "status": "success", "response": { } }
```

与 Sh 路径不同，REST API 是权威的配置访问，不会应用 SequenceNumber 不同步检查。写入直接设置文档。

列出所有文档

返回用户的每个存储文档。

端点: `GET /api/subscriber/repository_data/:imsi`

```
curl -k
https://hss.example.com:8443/api/subscriber/repository_data/001001123
```

```
{
  "status": "success",
  "response": {
    "MMTEL-Services":      { "sequence_number": "0",
"service_data": "<MMTelServices …/>" },
    "IMS-ODB-Information": { "sequence_number": "1",
"service_data": "<OdbForImsOrientedServices
xmlns=\"odb.mmtel.omnitou.ch\"/>" }
  }
}
```

获取单个文档

通过 Service-Indication 返回一个文档。

端点: `GET /api/subscriber/repository_data/:imsi?service_indication=:service_indication`

```
curl -k
"https://hss.example.com:8443/api/subscriber/repository_data/00100112
service_indication=MMTEL-Services"
```

```
{
  "status": "success",
  "response": { "sequence_number": "0", "service_data": "
<MMTelServices …/>" }
}
```

□□□□□□□

□□□□□□□□□□□□□□□□□□

□□□ `PUT /api/subscriber/repository_data/:imsi`

```
curl -k -X PUT
https://hss.example.com:8443/api/subscriber/repository_data/001001123
\
  -H "Content-Type: application/json" \
  -d '{
        "service_indication": "MMTEL-Services",
        "service_data": "<MMTelServices version=\"1\"/>",
        "sequence_number": "0"
      }'
```

□□	□□	□□	□□	□□
`service_indication`	□□□	□	-	□□□/□□□□□□□□□□ `MMTEL-Services` □□
`service_data`	□□□	□	-	□□□□ XML □□□□□□□□□□□□□□□□□□□□□□□□ Service-Indication □ SequenceNumber□□
`sequence_number`	□□□	□	`0`	□□□□□□□□□□

响应回显存储的文档：

```
{
  "status": "success",
  "response": { "sequence_number": "0", "service_data": "
<MMTelServices version=\"1\"/>" }
}
```

删除文档

通过 Service-Indication 移除单个文档。

端点：`DELETE /api/subscriber/repository_data/:imsi`

```
curl -k -X DELETE
https://hss.example.com:8443/api/subscriber/repository_data/001001123
\
  -H "Content-Type: application/json" \
  -d '{ "service_indication": "MMTEL-Services" }'
```

字段	类型	必需	默认	描述
`service_indication`	字符串	是	-	要移除的文档。

响应返回用户的剩余文档。

参考表

Data-Reference 值 (RepositoryData)

值	名称	描述	参考
0	RepositoryData	由 Service-Indication 标识的透明服务数据	3GPP TS 29.328 §7.6

���果和实验结果代码

代码	名称	意义
2001	DIAMETER_SUCCESS	UDR/PUR/SN 成功
5001	DIAMETER_ERROR_USER_UNKNOWN	用户未找到或未启用 I
5105	DIAMETER_ERROR_TRANSPARENT_DATA_OUT_OF_SYNC	更新的 SequenceNu 不是存储值的后继

应用 ID

ID	接口	描述	参考
16777217	Sh	AS ↔ HSS 用户数据	3GPP TS 29.329

常见 Service-Indications

Service-Indication	典型内容
`MMTEL-Services`	MMTel 补充服务（呼叫转移、通信等待/保持/限制、OIP/OIR、会议）：simservs XML
`IMS-ODB-Information`	针对 IMS 定向服务的运营商确定的限制信息

> Service-Indication 值由应用服务器及其相应的 XML 架构定义；HSS 不施加固定列表。

故障排除

应用服务器在更新时收到 **5105**

症状： PUR 的响应为 Experimental-Result `5105`，文档未改变。

可能原因：

- AS 发送了一个 SequenceNumber`???`该值不是存储在 HSS 的值的直接后继（其缓存副本已过时）。
- 另一个 AS 的并发更新提升了 SequenceNumber。

解决方案：

1. 使用 UDR 读取当前文档，以获取最新的 SequenceNumber。
2. 重新发出 PUR，SequenceNumber = 存储 + 1。
3. 要强制设置值，请通过 REST API 设置，该 API 不应用并发检查。

返回的 **ServiceData** 与写入的不同

症状： AS 认为有效负载被 HSS 修改。

可能原因：

- 这不应该发生。OmniHSS 逐字存储和返回 `ServiceData`。明显的差异通常来自 AS 自己的 XML 库在比较之前重新序列化文档（更改命名空间前缀、属性顺序或空白）。

解决方案：

1. 将 `UDA`/REST 响应中返回的原始字节与发送的原始字节进行比较，而不是重新解析的表示。
2. 确认 AS 在比较之前没有在其侧进行 XML 规范化。

通过 **REST** 未找到文档

症状： `GET …/repository_data/:imsi?service_indication=…` 返回错误。

可能原因：

- 用户没有存储该 Service-Indication 的文档。

IMSI 不对应于现有用户。

解决方案：

1. 使用 GET /api/subscriber/repository_data/:imsi 列出所有文档，以查看存储的内容。
2. 使用 通过 IMSI 获取用户 验证 IMSI。

SRVCC - 单无线电语音通话连续性

← 返回操作指南

概述

SRVCC（单无线电语音通话连续性）允许用户设备在LTE覆盖失败时，将活跃的VoLTE通话切换到2G/3G电路交换（CS）承载。MME通过**Sv**接口（TS 29.280）协调转移。为了构建Sv PS到CS请求，MME需要从HSS获取两个值：

AVP	代码	目的
`STN-SR`	1433	SRVCC的会话转移号码。MSC-Server拨打此E.164号码以联系TAS/S-CSCF，后者执行SIP REFER将活跃的VoLTE会话转移到CS承载。
`C-MSISDN`	1452	关联MSISDN。TAS使用此信息将传入的STN-SR呼叫与正确的活跃会话匹配，当订阅者有多个呼叫进行时。

OmniHSS在S6a 位置更新应答（ULA）`Subscription-Data` AVP中提供`STN-SR`。MME从ULA中读取它，并在Sv PS到CS请求中转发。

参考规范：

规范	标题
TS 23.216	SRVCC PS到CS过程
TS 29.272	S6a接口（STN-SR §7.3.89，Subscription-Data §7.3.2）
TS 23.003	STN-SR号码格式（§13.4）

□□□□

OmniHSS□□□

参数

参数	类型	必需	默认	描述
`stn_sr_prefix`	字符串	否	（无）	运营商分配的STN-SR前缀，附加到每个订阅者的MSISDN以形成完整的E.164 STN-SR号码。当未设置时，OmniHSS将从所有ULA响应中省略`STN-SR` AVP。

STN-SR号码格式

完整的STN-SR号码为：

```
STN-SR = stn_sr_prefix + 订阅者 MSISDN
```

该号码以**TBCD**（电话二进制编码十进制，按照TS 24.008 §10.5.4.7）编码，并放置在`STN-SR` AVP（代码1433，供应商10415）中。

示例：

输入	值
配置的前缀	`9`
订阅者 MSISDN	`61412345678`
计算的 STN-SR	`961412345678`

MSC-Server拨打`961412345678`。呼叫到达TAS，TAS使用MSISDN部分定位活跃的IMS会话，并发出SIP REFER将其转移到CS承载。

配置示例

启用**SRVCC**

```
# config/runtime.exs

config :hss, :srvcc,
  # MSC-Server拨打的前缀以通过PSTN/CS联系TAS。
  # STN-SR = stn_sr_prefix + 订阅者 MSISDN。
  stn_sr_prefix: "9"
```

工作原理：每个ULA响应在`Subscription-Data`中□含`STN-SR`。该值是前缀与订阅者的MSISDN连接的TBCD编码。MME存储该���，并在触发SRVCC时在Sv PS到CS请求中转发它。

禁用**SRVCC**□默认）

当`srvcc`键缺失或`stn_sr_prefix`未设置时，OmniHSS将省略`STN-SR` AVP。ULA在其他方面是相同的。对于不需要SRVCC的部署使用此配置。

与**MME**的交互

MME（OmniMME）从ULA `Subscription-Data`中读取`STN-SR`并将其存储在订阅者上下文中。当触发SRVCC时，它在发送到MSC-Server的Sv **PS**到**CS**请求中□含`STN-SR`。有关MME实现的详细信息，请参见 Omnitouch/omnimme#426。

故障排除

ULA中未出现**STN-SR**

症状： MME无法填充Sv PS到CS请求。SRVCC切换失败。

可能原因：

- `config/runtime.exs`中未设置`stn_sr_prefix`。

- HSS服务在配置更改生效之前已启动。

解决方案：

1. 验证`config/runtime.exs`中是否存在`srvcc`块，并且`stn_sr_prefix`非空。
2. 重新启动OmniHSS服务以应用配置更改。
3. 从MME触发新的ULR（例如通过强制TAU）。确认ULA `Subscription-Data`现在含`STN-SR`。

STN-SR值不正确

症状： MSC-Server接收到格式错误或无法路由的号码。

可能原因：

- 前缀不正确或缺少前导数字。
- 存储在HSS中的订阅者MSISDN不符合国际E.164格式（仅数字，无`+`）。

解决方案：

1. 通过API检查订阅者的MSISDN（`GET /api/subscriber/imsi/:imsi`）。
2. 验证预期的STN-SR：将`stn_sr_prefix`与存储的MSISDN连接。
3. 确认MSC-Server的路由表处理该号码范围。

← 返回操作指南

Sy 接口（消费限额控制）

概述

Sy 接口将 HSS（作为 **PCRF**，策略和计费规则功能）连接到外部 **OCS**（在线计费系统），用于 用户消费限额控制。它允许 PCRF 基于 OCS 为每个用户维护的 策略计数器 的状态（已消耗的数据配额、每月上限的消费等）来做出政策决策，例如应用哪个计费规则，或是否限制用户。

OmniHSS PCRF 通过 Gx 安装计费规则和 QoS，并通过 Rx 授权 IMS 媒体。通过 Sy，它订阅 OCS 中用户的策略计数器的状态，并在状态变化时收到通知，以便这些 Gx 决策可以考虑用户当前的消费和配额。

Sy 的定义来自 3GPP TS 29.219，使用直径应用 ID **16777302**。

属性	值
应用 ID	16777302
规范	3GPP TS 29.219
HSS 角色	PCRF（SLR/STR 的直径客户端，SNR 的服务器）
对等方	OCS（在线计费系统）
传输	TCP 或 SCTP，端口 3868

> **Sy** 与 **Gy** 的区别。 Sy 不是在线计费。Gy (TS 32.299) 是 PGW 请求 OCS 授予实时流量的信用的地方。Sy 是一个独立的轻量级接口，**PCRF** 订阅 OCS 的策略计数器的 状态，并在状态变化时收到通知，以便它可以调整策略。OCS 在两者中都是服务器。

架构

HSS 是 **Sy** 客户端：当它首次需要用户的消费状态时，它打开一个 Sy 会话，并在用户的最后一个数据会话结束时关闭它。OCS 是 **Sy** 服务器：它回答状态请求，并在任何计数器的状态变化时通知 PCRF，以便它可以在 Gx 上重新应用政策。

策略计数器和计费规则

OCS 为每个用户���护一个或多个 策略计数器。每个计数器具有：

- 策略计数器标识符（AVP 2901） - 计数器的稳定名称，以及
- 策略计数器状态（AVP 2902） - 描述计数器当前状态的标签（例如 `active`，`over_quota`，`throttled`）。状态值的集合并未由 3GPP 定义；由运营商在 OCS 和 PCRF 上共同配置（TS 29.219 §3.1）。

OmniHSS 将这些绑定到其数据库中已经存在的 计费规则：

Sy AVP	性质	映射到	角色
`Policy-Counter-Identifier` (2901)	静态 - 命名计数器	HSS 数据库中的 计费规则名称	选择计数器管理的 □□ 计费规则
`Policy-Counter-Status` (2902)	动态 - 在阈值处变化	执行操作	决定现在对该规则 □□□：安装、移除或选择一个级别

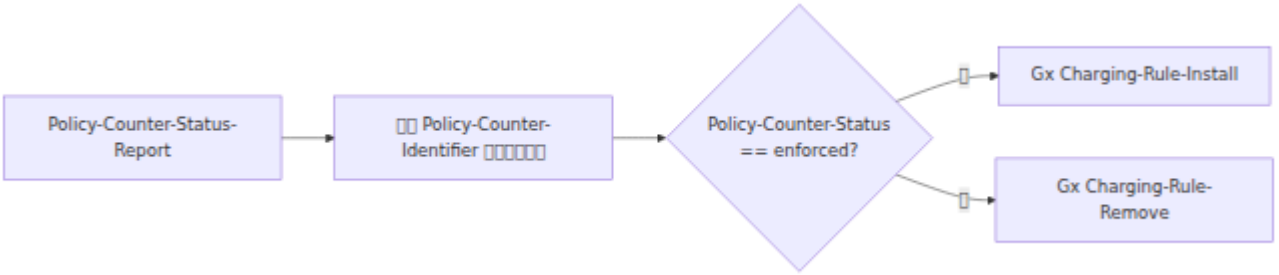
标识符在计数器的生命周期内是稳定的，因此单独它无法表达“低于配额”与“超过配额”。状态是当用户跨越阈值时变化的值，因此它驱动操作：

- 固定状态 **`enforced`** 安装命名的计费规则（在 Gx 上使用 `Charging-Rule-Install`）；
- 任何其他状态都会移除它（`Charging-Rule-Remove`）。

> **Sy** 本身不携带计��规则。 Sy 没有计费规则、QoS 或流 AVP - 这些仅存在于 Gx 上。因此，OCS 不能向 PCRF 提供规则定义；它只能 □□ 规则（通过计数器标识符）并信号 □□ 应用它（通过状态）。规则定义本身是在 HSS 中配置的。这就是为什么 `Policy-Counter-Identifier` 必须与计费规则名称匹配，并且 OCS 必须在规则应用时以状态 `enforced` 发出这些确切名称的原因。

Sy 会话生命周期

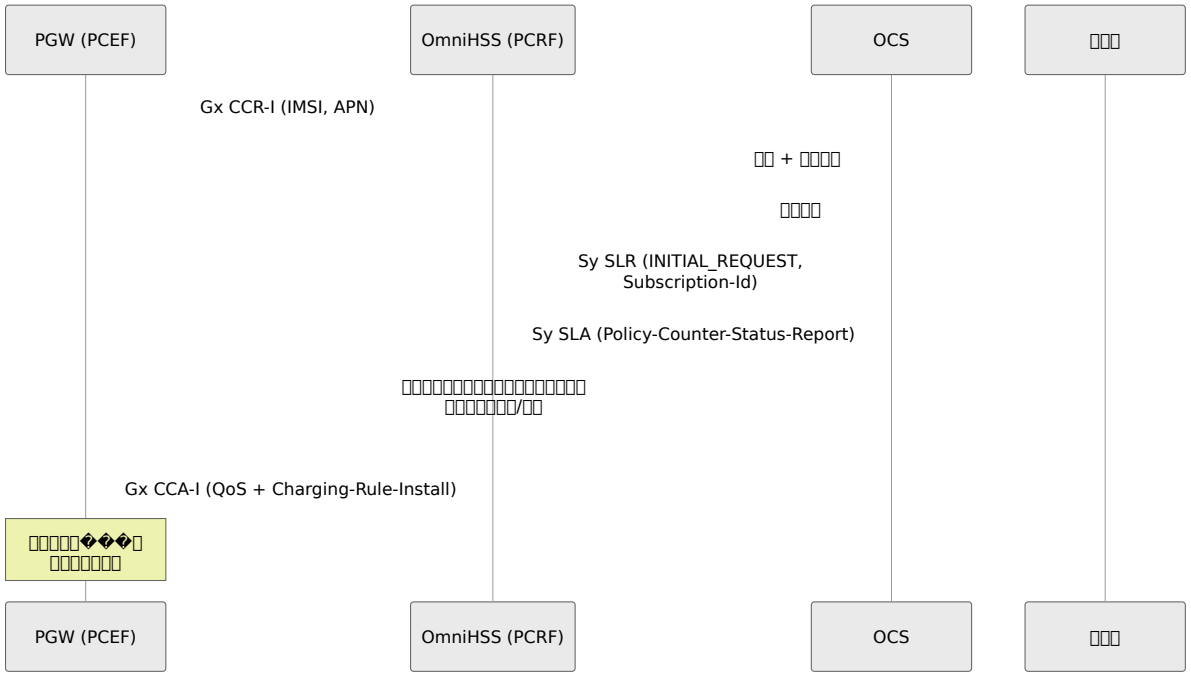
Sy 会话是 每个用户，而不是每个数据承载。PCRF 为用户保持一个 Sy 会话（由单个直径 `Session-Id` 标识），只要该用户的任何数据会话依赖于消费状态（TS 29.219 §4.4.1）。



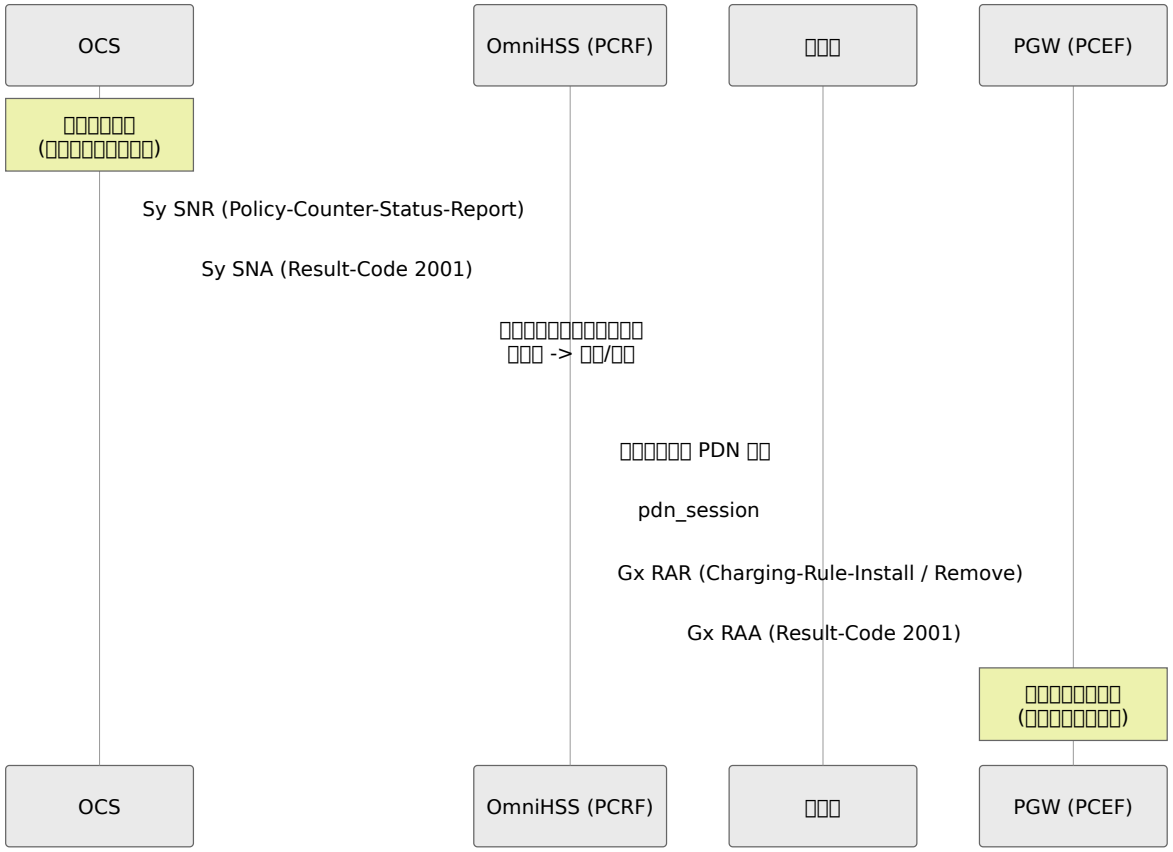
- □□ **(SLR, INITIAL_REQUEST)**□ □□□□ Gx CCR-I □□□□□□□□□□□□□□□□□□□□PCRF □□ Sy □□□□□□□□□□□□□□□□OCS □ SLA □□□□□□□□□□□□□□□
- □□ **(SLR, INTERMEDIATE_REQUEST)**□ □□ PCRF □□□□□□□□□□□□□□□□□□□□ Sy □ □□□□□□□□□□
- □□ **(SNR / SNA)**□ □□□□□□□□□□□□OCS □□□□□□□□□□□□PCRF □ SNA □□□□ Gx □□□□ □□
- □□ **(STR / STA)**□ □□□□□□□□□□□□□□□□□□PCRF □□□□□□□□□ Sy □□□

□□□

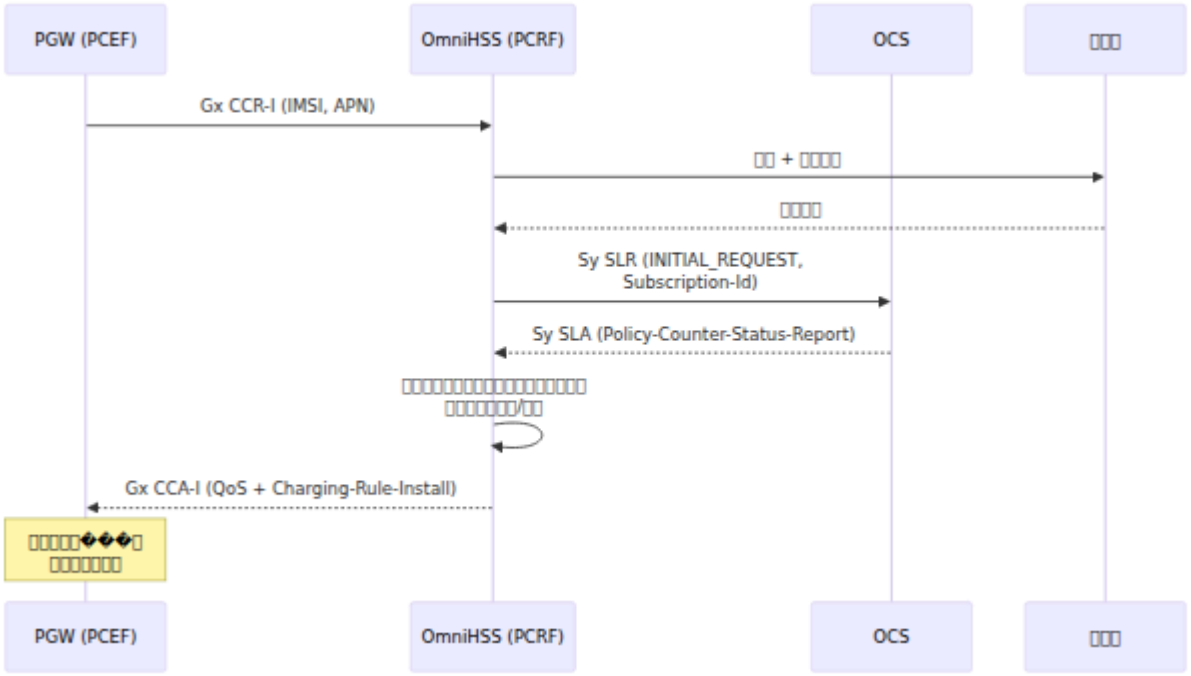
□□ 1□PDN □□□□□□□□□□□□


PGW (PCEF)
OmniHSS (PCRF)
OCS
□□□
Gx CCR-I (IMSI, APN)
□□ + □□□□
□□□□
Sy SLR (INITIAL_REQUEST, Subscription-Id)
Sy SLA (Policy-Counter-Status-Report)
□□□□□□□□□□□□□□□□□□□

□□ 2□□□□□□□□□OCS -> PCRF -> Gx RAR□



□□□□ PCRF □□□ Gx □□□□ □□ - □□ VoLTE □□□□□□□□□□ - □□□□□□□□□□□□□□□ PGW□□□□□□□□ □□ CCR□



流程 **3**：**Sy** 会话终止

配置

Sy 应用

Sy 应用在 HSS 直径服务中注册，与 Gx 和 Rx 一起（`config/runtime.exs`）：

```
%{
  application_name: :sy,
  application_dictionary: :diameter_gen_3gpp_sy,
  vendor_specific_application_ids: [
    %{vendor_id: 10415, auth_application_id: 16_777_302,
acct_application_id: nil}
  ]
}
```

参数	类型	必需	默认	描述
`application_name`	Atom	是	-	接口的内部名称。必须为 `:sy` Sy 应用。
`application_dictionary`	Atom	是	-	直径字典模块。必须为 `:diameter_gen_3` （3GPP TS 29.219[
`vendor_specific_application_ids`	List	是	-	在能力交换中宣传 Sy 应用。[
`vendor_id`	Integer	是	-	供应商 ID。3GPP 的为 `1`
`auth_application_id`	Integer	是	-	认证应用 ID。Sy 的为 `16_777_302`。
`acct_application_id`	Integer | nil	否	`nil`	计费应用 ID。Sy 不使用；设 ��� `nil`。

Sy 设置

Sy 行为在 `:hss, :sy` 键下配置（`config/runtime.exs`）：

```
config :hss,
  sy: %{
    enabled: false
  }
```

参数	类型	必需	默认	描述
`enabled`	Boolean	否	`false`	主开关。当为 `true` 时，PCRF 在 Gx CCR-I 上为每个用户打开一个 Sy 会话。当为 `false` 时，Sy 被完全绕过。

安装命名计费规则的 `Policy-Counter-Status` 值是固定约定 **`enforced`**；任何其他值都会移除该规则。这是与 OCS 达成一致的，并且不可配置。

OCS 路由

HSS 不固定 OCS 主机或领域。发起的 SLR 和 STR 带有 HSS 的自身 `Destination-Realm`，而没有 `Destination-Host`，并通过领域加上 Sy 应用 ID（16777302）进行路由。任何在该领域中宣传 Sy 应用的对等方都会接收请求 - 通常是 **DRA**，它会转发到可用的 OCS。这在通过 DRA 到达 OCS 或直接对等时工作方式相同，并且允许路由层在多个 OCS 之间进行选择。

唯一的要求是，在该领域中存在一个宣传 Sy 应用的对等方。在 DRA 部署中，这就是已经为 HSS 配置的 DRA 对等方；不需要单独的 Sy 特定对等方条目。

策略计数器映射

为了使标识符与规则绑定正常工作，运营商需要在 两个 端一致配置：

- 在 **OCS** 上：每个策略计数器的 `Policy-Counter-Identifier` 设置为 HSS 中定义的 计费规则 的确切 `name`，并且当该规则应应用时，计数器报告状态 `enforced`。
- 在 **HSS** 上：引用的计费规则存在于数据库中。

请参见 策略计数器和计费规则 以获取映射模型。

指标

Sy 在其他 HSS 接口指标旁边公开 Prometheus 指标（见 指标与监控）。

指标：`hss_sy_messages_total` 类型：计数器 描述：处理的 Sy 消息，按命令、方向和结果分类。标签：

- `command` - `slr`，`sla`，`snr`，`sna`，`str`，或 `sta`
- `direction` - `incoming`（从 OCS 接收）或 `outgoing`（发送到 OCS）
- `result` - `sent`，`received`，`success`，或 `error`

指标：`hss_sy_policy_updates_total` 类型：计数器 描述：由 Sy 策略计数器状态驱动的 Gx 计费规则更改，按操作和结果分类。标签：

- `action` - `install` 或 `remove`

- `result` - `success` 或 `error`（Gx RAR 到 PGW 的结果）

示例查询：

```
# 从 OCS 的 Sy 通知率
rate(hss_sy_messages_total{command="snr", direction="incoming"}
[5m])

# 由消费限额驱动的限制/执行规则安装率
rate(hss_sy_policy_updates_total{action="install",
result="success"}[5m])

# Sy Gx RAR 失败比例
sum(rate(hss_sy_policy_updates_total{result="error"}[5m]))
  / sum(rate(hss_sy_policy_updates_total[5m]))
```

参考

命令

命令	缩写	代码	方向	目的
消费限额请求	SLR	8388635	PCRF -> OCS	请求策略计数器状态；订阅更改
消费限额应答	SLA	8388635	OCS -> PCRF	返回当前计数器状态
消费状态通知请求	SNR	8388636	OCS -> PCRF	通知计数器状态变化
消费状态通知应答	SNA	8388636	PCRF -> OCS	确认通知
会话终止请求	STR	275	PCRF -> OCS	关闭 Sy 会话（最终报告）
会话终止应答	STA	275	OCS -> PCRF	确认会话关闭

所有命令都携带应用 ID `16777302`，□括重用的 STR/STA (RFC 6733 会话终止)。

关键 **AVPs**

AVP	代码	类型	描述
`Policy-Counter-Identifier`	2901	UTF8String	命名一个策略计数器。绑定到 OmniHSS 中的计费规则名称。
`Policy-Counter-Status`	2902	UTF8String	��营商定义的计数器状态标签。驱动执行操作。
`Policy-Counter-Status-Report`	2903	Grouped	一个计数器的标识符 + 状态 (+ 任何待处理状态)。
`SL-Request-Type`	2904	Enumerated	第一次请求为 `INITIAL_REQUEST (0)`，之后为 `INTERMEDIATE_REQUEST (1)`。
`Pending-Policy-Counter-Information`	2905	Grouped	一个未来状态及其生效时间。
`Pending-Policy-Counter-Change-Time`	2906	Time	待处理状态生效的时间。
`SN-Request-Type`	2907	Unsigned32	SNR 上的位掩码。位 0 = 正常报告，位 1 = 中止（OCS 请求结束 Sy 会话）。
`Subscription-Id`	443	Grouped	用户身份（IMSI 和/或 MSISDN）在 SLR 中。

结果代码

代码	名称	意义
2001	DIAMETER_SUCCESS	请求成功处理
5030	DIAMETER_USER_UNKNOWN	OCS 不知道该用户；Sy 会话未创建
5570	DIAMETER_ERROR_UNKNOWN_POLICY_COUNTERS	一个或多个请求的策略计数器在 OCS 中未知
4241	DIAMETER_ERROR_NO_AVAILABLE_POLICY_COUNTERS	OCS 没有该用户的策略计数器

代码 5570 和 4241 是 Sy 特定的实验结果代码值（TS 29.219 §5.5）；5030 是由 RFC 4006 定义的。

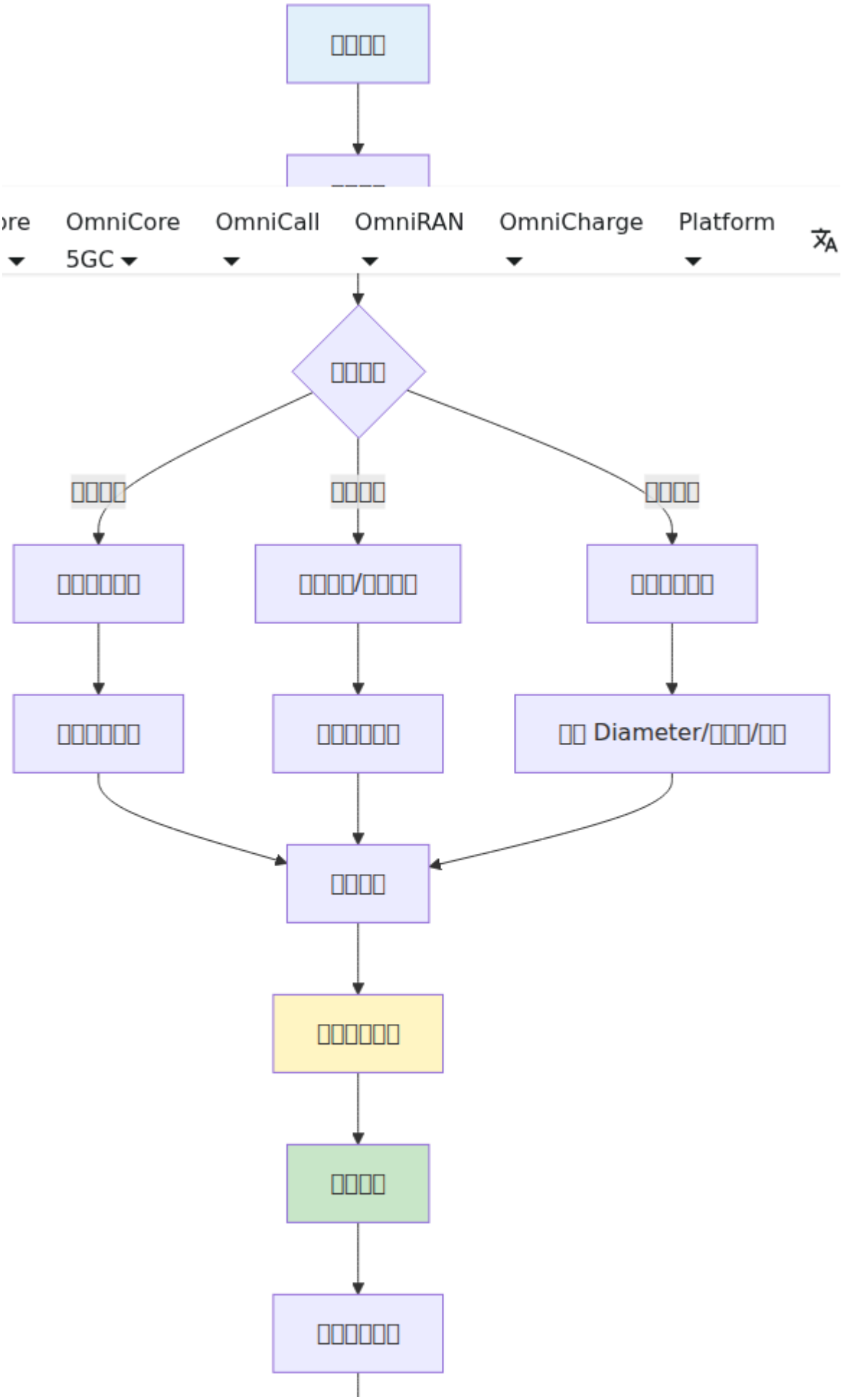
相关文档

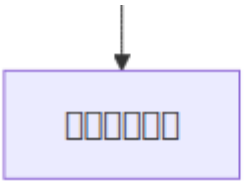
- PCRF - Gx 和 Rx 策略控制，Sy 供其输入
- 直径协议 - 所有 HSS 接口的消息流
- 架构 - 整体 HSS 架构

OmniHSS 故障排除指南
← 返回操作指南

目录

故障排除概述
认证失败
Diameter 连接问题
数据库问题
EPC 注册失败
IMS 注册失败
VoLTE 呼叫失败
漫游问题
EIR 问题
性能问题
用户状态问题
API 问题
诊断工具和命令


OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
Diameter/



收集的信息

在故障排除任何问题之前，请收集：

1. 用户信息（如果是特定用户）
    - IMSI
    - MSISDN（电话号码）
    - 最后已知状态
    - 设备上的错误消息
2. 时间信息
    - 问题何时开始？
    - 是间歇性还是持续性？
    - 上一次成功操作的时间
3. 影响范围
    - 单个用户还是多个用户？
    - 特定网络还是所有网络？
    - 特定服务（数据/语音）还是两者？
4. 系统状态
    - 检查 控制面板 以获取系统状态
    - 查看 Diameter 对等体状态
    - 验证数据库连接

认证失败

症状

- 用户无法连接到网络
- "认证被拒绝" 错误
- 重复的认证尝试

常见原因和解决方案

原因 **1**: 密钥集不正确

症状:

- 特定用户的认证失败持续
- 对于同一配置文件的其他用户��效

诊断步骤:

1. 查询用户以验证 key_set_id:

   ```
   curl -k https://hss.example.com:8443/api/subscriber/imsi/[IMSI]
   ```

2. 验证密钥集存在且具有正确的值:

   ```
   curl -k https://hss.example.com:8443/api/key_set/[KEY_SET_ID]
   ```

3. 将 Ki 和 OPC 值与 SIM 卡文档进行比较

解决方案:

- 使用正确的 密钥集 更新用户
- 如果密钥正确，SIM 卡可能有故障

原因 **2**: **SQN**不同步

症状:

- 认证在之前有效后失败
- 错误：“SQN 同步失败”
- 间歇性有效

诊断步骤：

1. 检查数据库中用户状态的 SQN 值
2. 查看日志中的 SQN 相关错误
3. 验证用户的密钥集 SQN 值

解决方案：

- SQN 在用户发送 AUTS 后会自动重新同步
- 如果持续存在，请在密钥集中将 SQN 重置为 0（需要用户重新连接）

警告：重置 SQN 可能会导致安全问题。仅在维护期间进行。

原因 **3**：用户被禁用

症状：

- 认证立即被拒绝
- 没有生成认证向量

诊断步骤：

1. 检查用户启用状态：

   ```
   curl -k https://hss.example.com:8443/api/subscriber/imsi/[IMSI]
   ```

2. 验证 `enabled` 字段为 `true`

解决方案：

- 启用用户：

```
curl -k -X PUT https://hss.example.com:8443/api/subscriber/[ID]
\
  -H "Content-Type: application/json" \
  -d '{"subscriber": {"enabled": true}}'
```

原因 **4**：缺少 **EPC** 配置文件

症状：

- 用户查找成功但认证失败
- 错误：“未分配 EPC 配置文件”

诊断步骤：

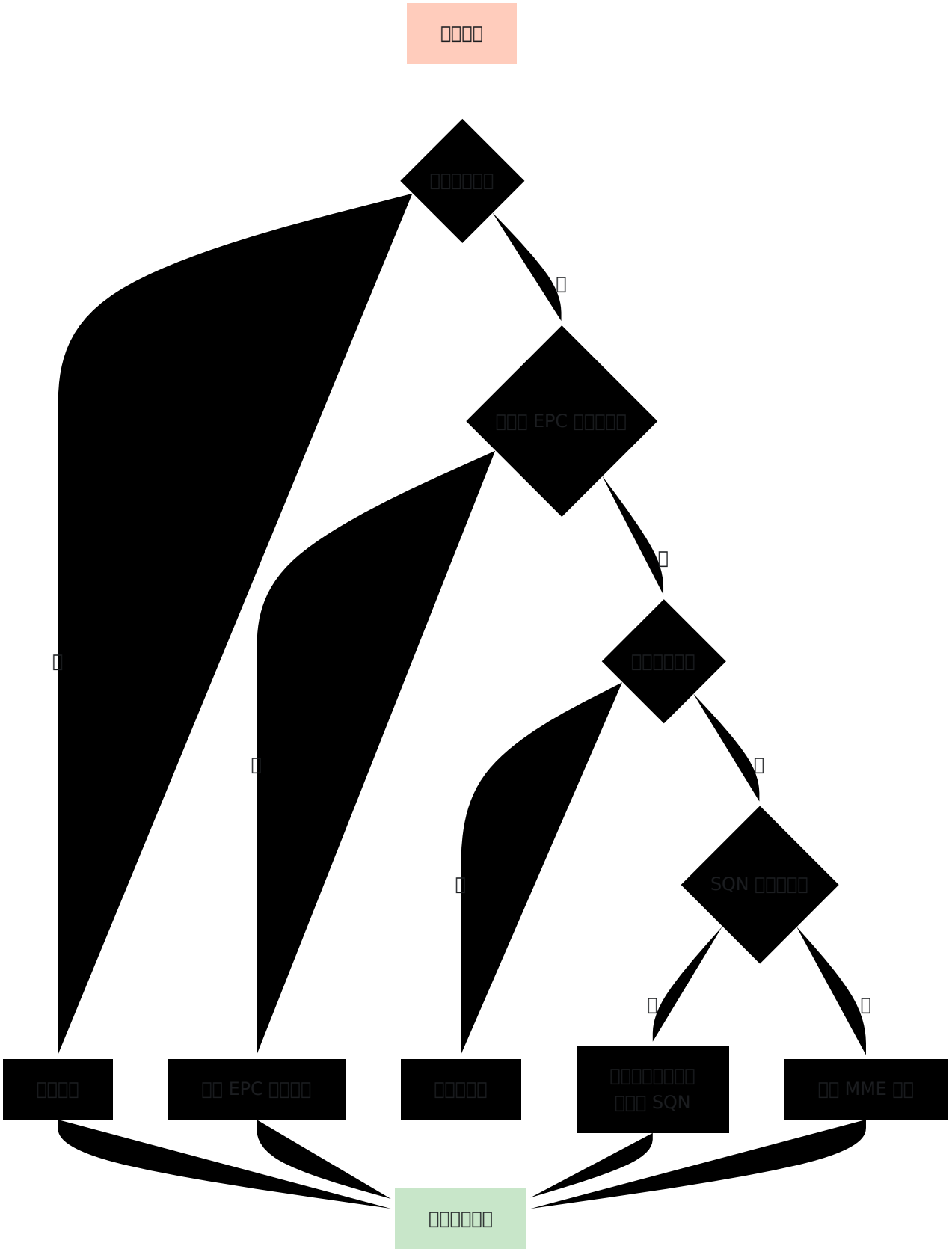
1. 检查用户的 `epc_profile_id` 字段
2. 验证 EPC 配置文件存在：

```
curl -k
https://hss.example.com:8443/api/epc/profile/[PROFILE_ID]
```

解决方案：

- 将有效的 EPC 配置文件 分配给用户

□□□□□□□□□□


□□□□
□□□□□□□
□
□□□ EPC □□□□□□
□
□□□□□□□
□
□
□
□
□
SQN □□□□□□
□
□
□□□□
□□ EPC □□□□
□□□□□
□□□□□□□□□□
□□□ SQN
□□ MME □□
□□□□□□□

Diameter 连接问题

症状

在 控制面板 中显示 Diameter 对等体断开连接 "无法连接到主机" 错误
所有用户的服务失败

常见原因和解决方案

原因 1：网络连接

症状：

对等体从未连接
连接超时错误
Ping 对等体失败

诊断步骤：

1. 验证从 OmniHSS 到对等体的网络连接：

ping [PEER_IP]

2. 检查 Diameter 端口是否可达：

telnet [PEER_IP] 3868

3. 验证防火墙规则允许 Diameter 流量（端口 3868）

解决方案：

修复网络路由
更新防火墙规则
验证对等体正在运行并监听

原因 **2**: **Diameter** 配置不正确

症状:

- 连接尝试失败
- CER/CEA 交换失败
- 对等体拒绝连接

诊断步骤:

1. 查看 runtime.exs 中的 Diameter 配置:
    - 验证对等体 origin_host 与对等体的预期值匹配
    - 检查 origin_realm 配置
    - 验证对等体 IP 地址是否正确
2. 检查日志中的 CER/CEA 错误
3. 验证对等体的配置是否期望 OmniHSS 的 origin_host

解决方��:

- 使用正确的 Diameter 配置 更新 runtime.exs
- 在配置更改后重启 OmniHSS
- 与对等体管理员协调以验证设置

原因 **3**: 证书问题（**TLS Diameter**）

症状:

- 在 TLS 握手期间连接失败
- 证书验证错误
- "证书过期" 或 "证书无效" 错误

诊断步骤:

1. 验证 `priv/cert/` 中存在证书文件
2. 检查证书过期:

```
openssl x509 -in priv/cert/diameter.crt -noout -dates
```

3. 验证证书链是否完整
4. 检查对等体的证书是否为双向 TLS

解决方案：

- 更新过期的证书
- 安装正确的证书链
- 更新证书文件并重启 OmniHSS

原因 **4**：对等应用程序支持不匹配

症状：

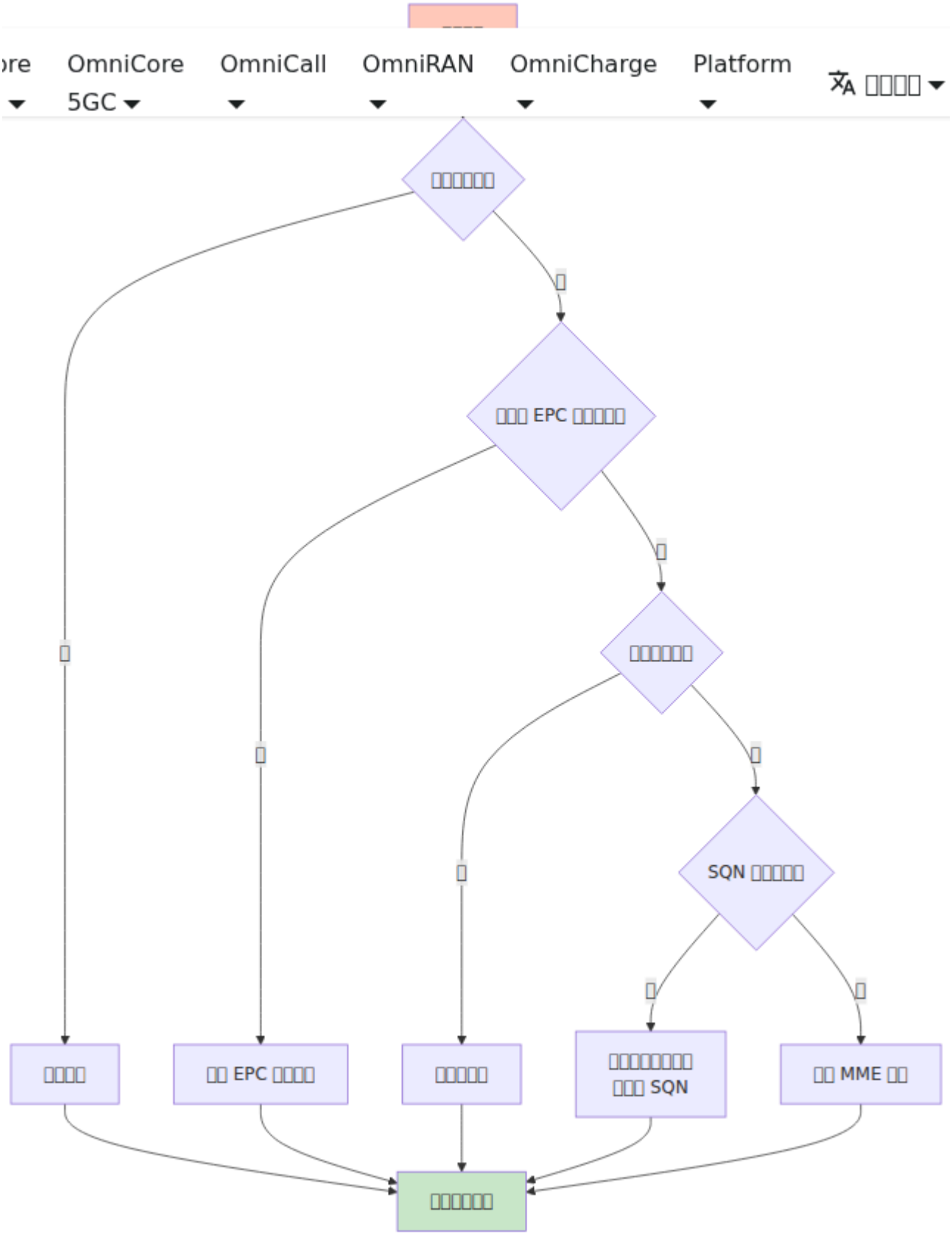
- 对等体连接但不支持所需的应用程序
- 能力交换成功但操作失败
- "不支持的应用程序" 错误

诊断步骤：

1. 检查 控制面板 Diameter 页面 中的对等应用程序
2. 验证对等体支持所需的应用程序（S6a、Cx、Sh 等）
3. 查看日志中的 CER/CEA 交换

解决方案：

- 验证对等体配置是否包括所需的 Diameter 应用程序
- 检查对等体类型是否匹配预期功能：
  - MME 必须支持 S6a（16777251）
  - S-CSCF 必须支持 Cx（16777216）
  - P-GW 必须支持 Gx（16777238）



Diameter 故障排除流程图

数据库问题

症状

API 返回 500 错误
控制面板无法加载
"数据库连接失败" 错误
查询性能缓慢

常见原因和解决方案

原因 1：数据库服务器宕机

症状：

所有 API 调用失败
控制面板显示错误
"连接被拒绝" 错误

诊断步骤：

1. 测试数据库连接：

连接到数据库
psql -h [DB_HOST] -U [DB_USER] -d [DB_NAME]

2. 检查数据库服务器上的数据库服务状态

3. 验证到数据库服务器的网络连接

解决方案：

启动数据库服务
修复数据库服务器问题
验证到数据库服务器的网络路由

原因 2：数据库凭据不正确

症状：

"认证失败" 错误
OmniHSS 在启动时无法连接

诊断步骤：

1. 查看 runtime.exs 中的数据库配置
2. 使用数据库客户端手动测试凭据
3. 检查数据库用户权限

解决方案：

更新 runtime.exs 中的 数据库配置
授予数据库用户正确的权限
在配置更改后重启 OmniHSS

原因 3：连接池耗尽

症状：

间歇性 500 错误
"没有可用连接" 错误
高负载期间触发故障

诊断步骤：

1. 检查数据库中的当前连接数
2. 查看 runtime.exs 中的数据库池大小
3. 在高峰负载期间监控连接使用情况

解决方案：

在 runtime.exs 配置中增加池大小
如果池重复耗尽，请调查连接泄漏
如果负载持续高，考虑数据库扩展

原因 4：查询缓慢

症状：

API 响应非常慢
用户查找超时
数据库 CPU 高

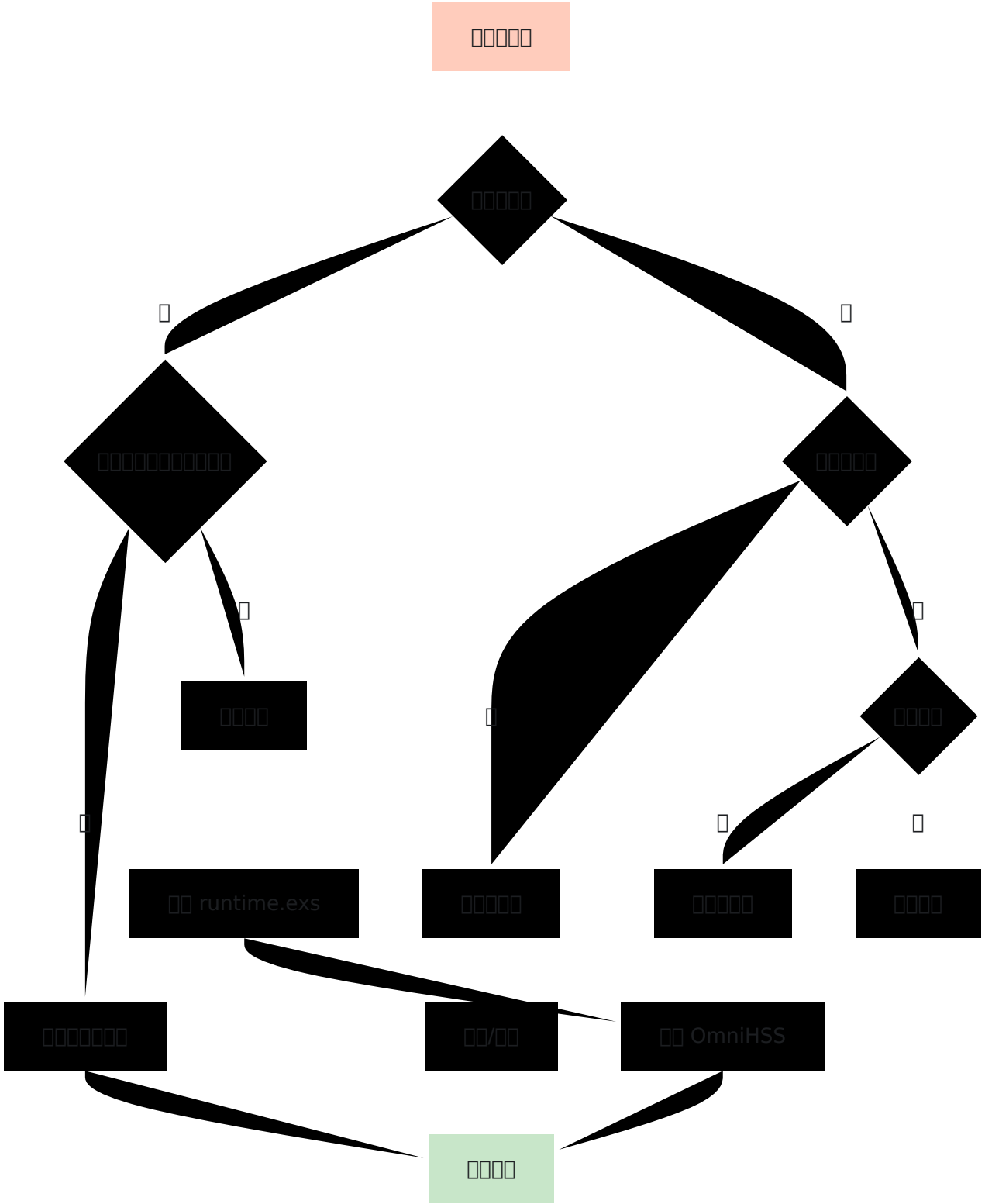
诊断步骤：

1. 查询数据库的慢查询日志
2. 确定具体的慢查询
3. 检查是否缺少索引
4. 验证用户数量和表大小

解决方案：

优化慢查询
添加缺失的索引
考虑数据库性能调优
如有需要，计划数据库扩展

□□□□□□□□□□□


□□□□□
□□□□□
□
□
□□□□□□□□□□□□□
□□□□□
□
□
□□□□
□
□□□□
□
□
□
□□ runtime.exs
□□□□□
□□□□□□
□□□□
□□□□□□□□
□□/□□
□□ OmniHSS
□□□□

EPC 注册失败

症状

- 用户无法连接到 LTE 网络
- MME 拒绝连接
- 没有建立 PDN 会话

常见原因和解决方案

原因 1: 漫游被拒绝

症状:

- 用户在���地网络上有效，但在漫游时失败
- "不允许漫游" 错误
- 对某些网络有效，但对其他网络无效

诊断步骤:

1. 检查用户的 roaming_profile_id
2. 查询漫游配置文件和规则
3. 验证访问网络的 MCC/MNC
4. 检查该网络是否存在漫游规则

解决方案:

- 为访问网络的 MCC/MNC 添加 漫游规则
- 或更新漫游配置文件的默认操作以允许
- 查看 漫游文档 以获取配置

原因 2: 缺少 APN 配置

症状:

- 连接成功但 PDN 会话失败
- MME 返回 "未知 APN" 错误

用户无法获得数据连接

诊断步骤：

1. 检查 EPC 配置文件是否链接了 APN 配置文件
2. 验证 APN 标识符是否与设备请求匹配
3. 查询 APN 配置文件配置

解决方案：

将 APN 配置文件 链接到用户的 EPC 配置文件
确保 APN 名称与设备配置匹配
验证 APN QoS 配置文件是否存在

原因 3：MME 未连接

症状：

所有用户连接失败
与 MME 没有通信
Diameter 对等体宕机

诊断步骤：

1. 检查 控制面板 Diameter 页面
2. 验证 MME 对等体状态为 "已连接"
3. 检查 MME 是否支持 S6a 应用程序

解决方案：

故障排除 Diameter 连接问题
验证 MME 配置
联系 MME 管理员

原因 4：用户状态损坏

症状：

用户显示为已连接，但无法再次连接

状态与现实不符
断开和重新连接失败

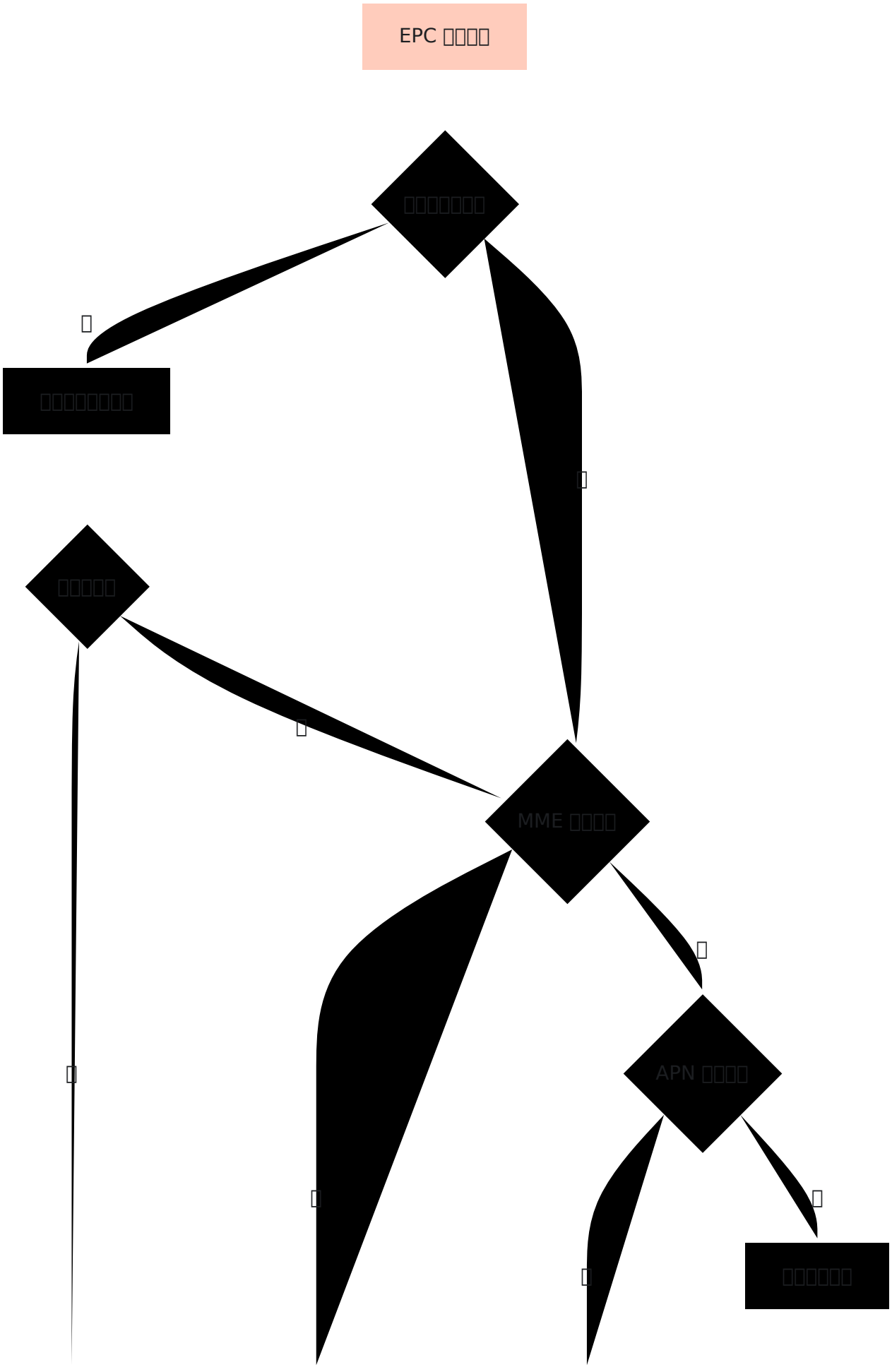
诊断步骤：

1. 从数据库查询用户状态
2. 检查过期的 MME 分配
3. 验证最后更新时间戳

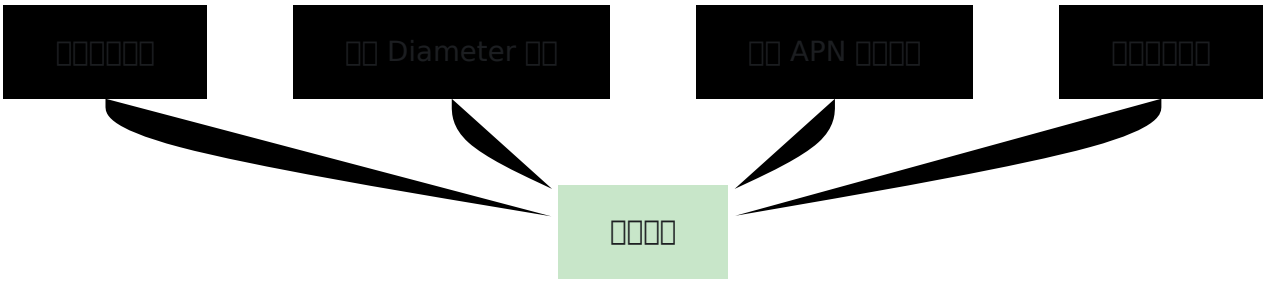
解决方案：

清除用户状态（断开程序）
重置用户状态中的服务 MME
可能需要用户重启

EPC □□□□□□□□□



EPC 注册失败
否
是
是



IMS 注册失败

症状

- 用户无法注册 VoLTE
- 设备上显示 "IMS 注册失败"
- 数据正常但语音不正常

常见原因和解决方案

原因 1: 用户未启用 IMS

症状:

- 用户有数据但没有 IMS
- 注册立即被拒绝

诊断步骤:

1. 查询用户并检查 `ims_enabled` 字段
2. 验证用户是否分配了 `ims_profile_id`

解决方案:

- 为用户启用 IMS
- 分配 IMS 配置文件

原因 2: S-CSCF 未连接

症状:

所有 IMS 注册失败
没有 IMS 相关的 Diameter 流量

诊断步骤：

1. 检查 控制面板 Diameter 页面
2. 验证 S-CSCF 对等体已连接
3. 检查 S-CSCF 是否支持 Cx 应用程序

解决方案：

修复与 S-CSCF 的 Diameter 连接问题
验证 S-CSCF 配置

原因 3：缺少或无效的 IFC 模板

症状：

用户授权应答期间注册失败
日志中出现与 IFC 相关的错误

诊断步骤：

1. 查询用户的 IMS 配置文件
2. 验证 IFC 模板是否存在
3. 检查 IFC XML 语法

解决方案：

使用有效的 IFC 模板更新 IMS 配置文件
查看 配置文件文档 获取 IFC 示例

原因 4：IMS 漫游被拒绝

症状：

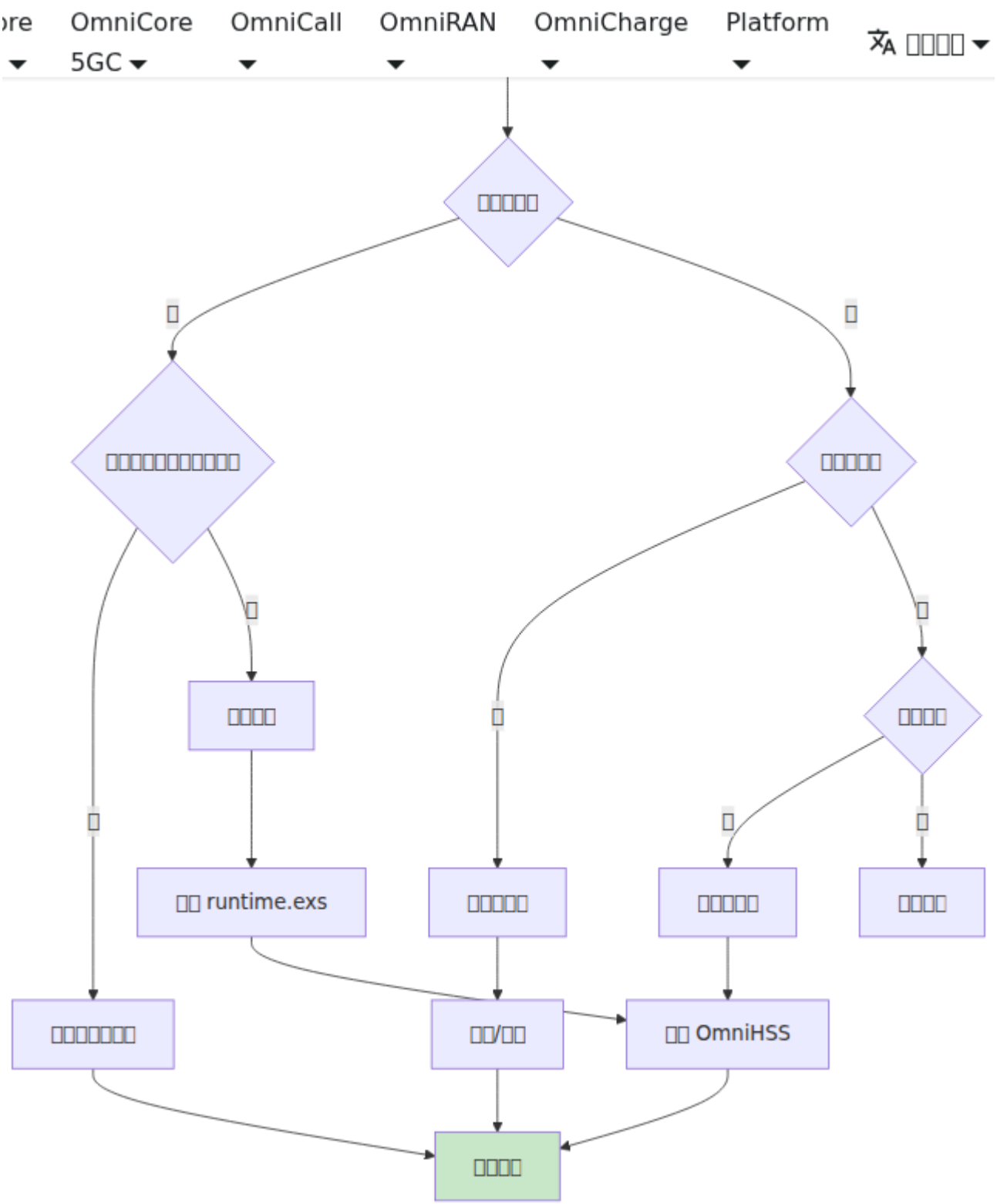
IMS 在本地网络上有效
漫游时失败
数据漫游有效但 IMS 无效

诊断步骤：

1. 检查漫游配置文件 IMS 操作
2. 验证漫游规则是否具有正确的 ims_action

解决方案：

更新 漫游规则 以允许 IMS
或更新漫游配置文件的默认 IMS 操作



IMS 注册故障排除流程图

VoLTE 呼叫失败

症状

IMS 注册成功但呼叫失败
单向音频
呼叫立即中断
设备上显示 "呼叫失败" 错误

常见原因和解决方案

原因 1：P-CSCF 未连接

症状：

注册有效但呼叫失败
媒体授权失败

诊断步骤：

1. 检查 控制面板 Diameter 页面
2. 验证 P-CSCF 对等体已连接
3. 检查 P-CSCF 是否支持 Rx 应用程序（OmniHSS PCRF 功能）

解决方案：

修复与 P-CSCF 的 Diameter 连接问题
验证 P-CSCF 配置指向 OmniHSS 的 Rx

原因 2：缺少媒体授权

症状：

呼叫设置开始但失败
AAR/AAA 交换失败
Rx 接口错误

诊断步骤：

1. 检查日志中的 Rx Diameter 消息
2. 验证是否收到 AAR（AA-请求）
3. 检查 AAA（AA-应答）响应

解决方案：

- 验证 P-CSCF 是否发送 AAR 进行媒体授权
- 检查 OmniHSS Rx 应用程序配置
- 验证用户是否具有有效的 IMS 注册

原因 **3**：**QoS/**承载问题

症状：

- 呼叫连接但没有音频
- 单向音频
- 质量问题

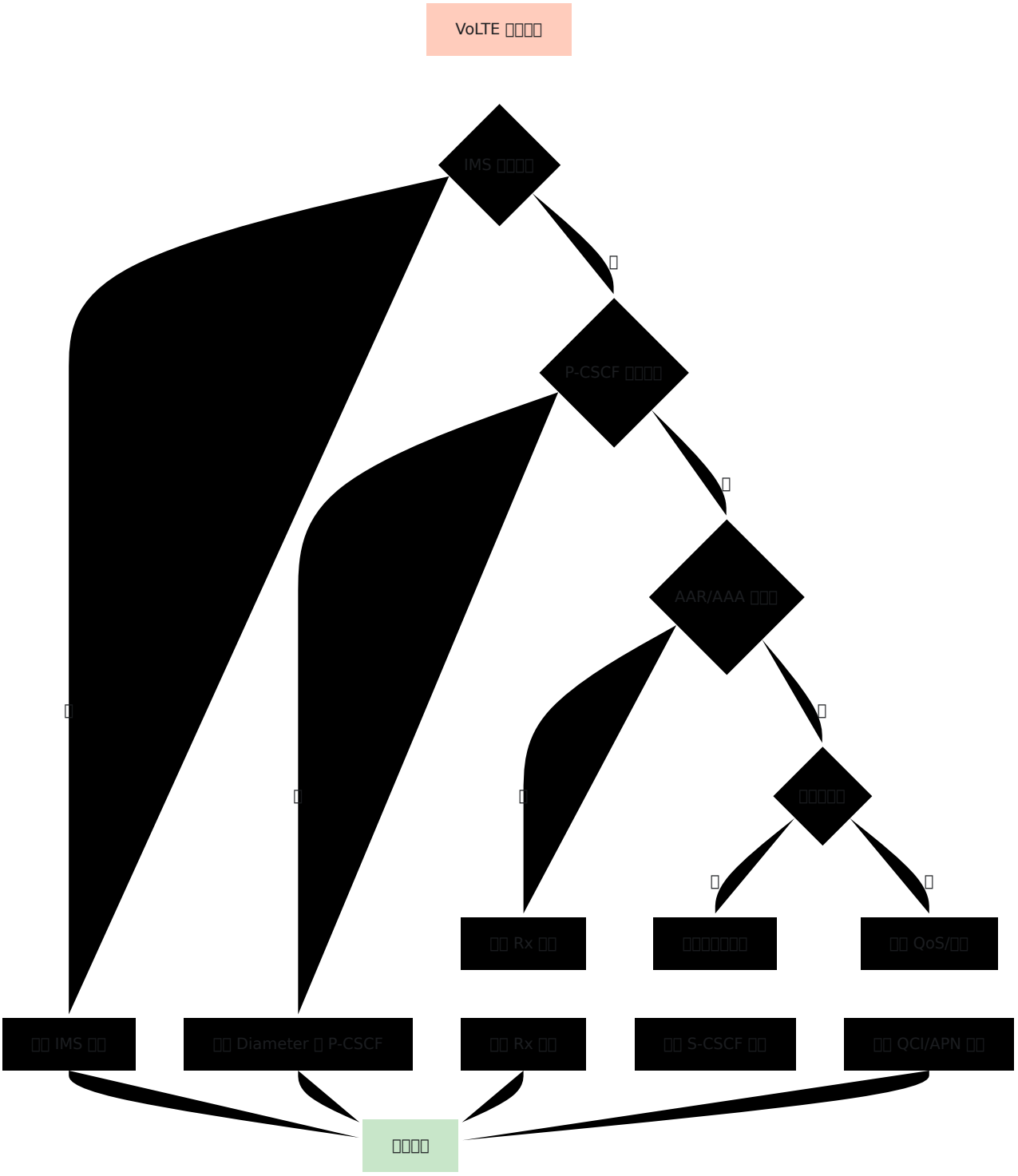
诊断步骤：

1. 检查语音 APN 的 APN QoS 配置文件
2. 验证 QCI 是否正确设置（通常语音为 QCI 1）
3. 检查 P-GW 是否连接 Gx（PCRF 功能）

解决方案：

- 验证 APN QoS 配置文件是否适用于 IMS APN
- 确保语音承载配置为 QCI 1
- 如有需要，修复与 P-GW 的 Diameter 连接问题

VoLTE □□□□□□□□□□


VoLTE □□□□
IMS □□□□
□
P-CSCF □□□□
□
AAR/AAA □□□
□
□
□
□□□□□
□
□
□□ Rx □□
□□□□□□□□
□□ QoS/□□
□□ IMS □□
□□ Diameter □ P-CSCF
□□ Rx □□
□□ S-CSCF □□
□□ QCI/APN □□
□□□□

漫游问题

症状

用户在本地有效但在漫游时无效
某些漫游网络有效，其他无效
漫游数据有效但语音无效（或反之）

常见原因和解决方案

原因 1：未分配漫游配置文件

症状：

用户漫游失败
其他用户成功漫游

诊断步骤：

1. 查询用户的 roaming_profile_id
2. 检查字段是否为 null

解决方案：

将 漫游配置文件 分配给用户

原因 2：政策拒绝漫游

症状：

在特定网络上漫游持续失败
错误指示政策拒绝

诊断步骤：

1. 从用户设备或 MME 确定访问的网络 MCC/MNC
2. 查询用户的漫游配置文件
3. 检查与 MCC/MNC 匹配的漫游规则

4. 检查配置文件的默认操作

解决方案：

- 添加 漫游规则 以允许访问的网络：

```
curl -k -X POST https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "Allow Visited Network",
      "mcc": "310",
      "mnc": "410",
      "data_action": "allow",
      "ims_action": "allow"
    }
  }'
```

原因 **3**：数据允许但 **IMS** 拒绝

症状：

- 数据漫游正常
- 语音/IMS 漫游失败
- 服务可用性分裂

诊断步骤：

1. 查询访问网络的漫游规则
2. 检查 `data_action` 与 `ims_action` 值
3. 验证漫游配置文件的默认操作

解决方案：

- 更新漫游规则以允许 IMS：
  - 设置 `ims_action: "allow"`
- 或更新配置文件的 `ims_action_if_no_rules_match` 为 `"allow"`

请参阅 漫游文档 以获取详细配置。

EIR 问题

症状

设备意外被阻止
被盗设备未被阻止
EIR 检查失败

常见原因和解决方案

原因 1：不正确的 IMEI 正则表达式

症状：

错误的设备被阻止/允许
规则匹配不正确

诊断步骤：

1. 查询 EIR 规则
2. 确定哪个规则正在匹配
3. 测试正则表达式模式与实际 IMEI
4. 检查规则优先级/顺序

解决方案：

使用正确的正则表达式更新 EIR 规则
在应用之前彻底测试正则表达式
考虑规则顺序（先匹配者胜）

原因 2：MME 未发送 S13 请求

症状：

EIR 检查从未发生
所有设备均被允许，无论规则如何

诊断步骤：

1. 检查 MME 是否配置为使用 S13 接口
2. 验证 MME Diameter 对等体是否连接
3. 检查 S13 应用支持
4. 审查 MME 配置

解决方案：

配置 MME 通过 S13 执行 EIR 检查
验证 Diameter 对等体支持 S13 应用 (16777252)
如有需要，请联系 MME 管理员

原因 3：没有默认规则

症状：

不匹配任何规则的设备表现异常

诊断步骤：

1. 查询所有 EIR 规则
2. 检查是否存在捕获所有规则
3. 验证规则排序

解决方案：

添加正则表达式为.* 的默认规则以匹配所有 IMEI
设置适当的操作（白名单或黑名单）
确保在捕获所有规则之前检查特定规则

性能问题

症状

API 响应缓慢
Diameter 请求超时
CPU 或内存使用率高

- 控制面板加载缓慢

常见原因和解决方案

原因 **1**: 数据库负载高

症状:

- 所有操作缓慢
- 数据库 CPU 高
- 查询超时

诊断步骤:

1. 检查数据库服务器资源使用情况
2. 确定慢查询
3. 检查缺失的索引
4. 监控查询模式

解决方案:

- 优化慢查询
- 添加数据库索引
- 增加数据库资源
- 考虑数据库扩展
- 请参见 数据库问题

原因 **2**: 用户数量高

症��:

- 性能随��间下降
- 缓慢与用户增长相关
- 列表操作特别慢

诊断步骤:

1. 查询用户总数

2. 检查表大小
3. 审查查询执行计划
4. 监控资源使用趋势

解决方案：

计划容量升级
针对大数据集优化查询
考虑大结果的分页
如有需要，实施缓存

原因 3：Diameter 对等体问题

症状：

Diameter 操作缓慢
特定对等体超时
某些对等体快，其他对等体慢

诊断步骤：

1. 检查 控制面板 Diameter 页面
2. 确定慢对等体
3. 测试到对等体的网络延迟
4. 检查对等体资源使用情况

解决方案：

调查对等体性能问题
检查网络路径是否拥堵
考虑添加冗余对等体
如有需要，增加 Diameter 超时

原因 4：内存问题

症状：

OmniHSS 内存使用率高

- 内存不足错误
- 性能随时间下降

诊断步骤：

1. 在应用页面检查 OmniHSS 内存使用情况
2. 监控内存趋势
3. 检查内存泄漏
4. 审查 Erlang VM 设置

解决方案：

- 重启 OmniHSS 以清除临时状态
- 如果使用持续增长，请调查内存泄漏
- 在 runtime.exs 中调整 Erlang VM 内存设置
- 如果持续高，请计划硬件升级

用户状态问题

症状

- 用户显示为已连接但实际上没有
- 状态信息过时
- 位置信息不正确
- 无法分离用户

常见原因和解决方案

原因 1: MME 崩溃/重启

症状：

- 用户显示服务 MME 但该 MME 不再提供服务
- 用户在 MME 重启后无法连接
- 状态过时

诊断步骤：

1. 检查用户状态以获取服务 MME
2. 验证 MME 是否已重启
3. 检查 MME 的最后连接时间

解决方案：

等待用户再次连接（状态将更新）
或手动清除用户状态
MME 应在重启时发送 Cancel-Location

原因 2：未收到网络分离

症状：

用户关闭电源但显示为已连接
PDN 会话仍保留在数据库中
位置未清除

诊断步骤：

1. 检查用户的 last_seen 时间戳
2. 验证是否为旧状态（数小时或数天前）
3. 检查用户设备是否可达

解决方案：

当用户再次连接时，状态将被清除
或等待状态超时（如果已实施）
对于非常过时的状态，可能需要手动清理

原因 3：数据库损坏

症状：

表之间状态不一致
外键冲突
状态不合理

诊断步骤:

1. 直接从数据库查询用户状态
2. 检查孤立记录
3. 验证引用完整性

解决方案:

- 确定���修复不一致的数据
- 可能需要手动清理数据库
- 如果损坏广泛，请联系支持

API 问题

症状

- API 返回错误
- API 响应缓慢
- 无法创建/更新实体
- 500 错误

常见原因和解决方案

原因 1: 请求数据无效

症状:

- 400 或 422 错误
- 验证错误消息
- 字段被拒绝

诊断步骤:

1. 查看错误响应以获取特定字段错误
2. 检查 API 请求格式

3. 验证必填字段是否存在
4. 检查数据类型

解决方案：

- 修复请求数据以匹配 API 参考
- 确保□含所有必填字段
- 验证外键引用是否存在（配置文件 ID 等）

原因 2：外键约束

症状：

- 无法创建用户
- 错误：“key_set_id 不存在”
- 引用的实体未找到

诊断步骤：

1. 确定哪个外键失败
2. 验证引用的实体是否存在：
    - key_set_id → 密钥集
    - epc_profile_id → EPC 配置文件
    - ims_profile_id → IMS 配置文件

解决方案：

- 首先创建引用的实体
- 或使用现有实体 ID
- 遵循 完整的配置工作流

原因 3：数据库连接

症状：

- 500 错误
- 所有 API 调用失败
- 数据库连接错误

解决方案：

- 请参见 数据库问题

诊断工具和命令

控制面板快速检查

1. 系统概述
    - URL: `https://[hostname]:7443/overview`
    - 检查：用户计数、活动会话、系统状态
2. **Diameter** 状态
    - URL: `https://[hostname]:7443/diameter`
    - 检查：所有关键对等体是否连接
3. 应用健康
    - URL: `https://[hostname]:7443/application`
    - 检查：内存使用情况、进程计数、运行时间

API 诊断命令

检查系统健康：

```
curl -k https://hss.example.com:8443/api/status
```

查询用户：

```
# 按 IMSI
curl -k
https://hss.example.com:8443/api/subscriber/imsi/001001123456789

# 按 MSISDN
curl -k
https://hss.example.com:8443/api/subscriber/msisdn/14155551234

# 按 ID
curl -k https://hss.example.com:8443/api/subscriber/d290f1ee-6c54-
4b01-90e6-d701748f0851
```

列出所有用户：

```
curl -k https://hss.example.com:8443/api/subscriber
```

检查配置文件配置：

```
# EPC 配置文件
curl -k https://hss.example.com:8443/api/epc/profile/e58ed763-
928c-4155-bee9-fdbaaadc15f3

# IMS 配置文件
curl -k https://hss.example.com:8443/api/ims/profile/7c9e6679-
7425-40de-944b-e07fc1f90ae7

# 漫游配置文件
curl -k https://hss.example.com:8443/api/roaming/profile/9c858901-
8a57-4791-81fe-4c455b099bc9
```

网络诊断命令

测试 **Diameter** 端口连接性：

```
telnet [PEER_IP] 3868
```

检查 **TLS** 证书：

```
openssl s_client -connect [hostname]:8443 -showcerts
```

测试数据库连接性：

```
psql -h [DB_HOST] -U [DB_USER] -d [DB_NAME] -c "SELECT COUNT(*)
FROM subscriber;"
```

日志分析

在日志中搜索特定 **IMSI**：

```
grep "001001123456789" /var/log/omnihss/omnihss.log
```

查找身份验证失败：

```
grep "authentication.*fail" /var/log/omnihss/omnihss.log
```

检查 **Diameter** 对等体事件：

```
grep "Diameter peer" /var/log/omnihss/omnihss.log
```

查找数据库错误：

```
grep -i "database.*error" /var/log/omnihss/omnihss.log
```

升级指南

何时升级

当出现以下情况时，升级到工程/供应商支持：

1. 系统范围的故障，无法通过文档程序解决
2. 数据损坏或数据库状态不一致
3. 疑似软件错误或意外行为
4. 性能问题无法通过调优解决
5. 安全事件或未经授权的访问
6. 关于未记录行为的问题

提供的信息

升级时，请括：

1. 详细症状 - 什么失败，何时，针对谁
2. 采取的步骤 - 您已经完成的故障排除
3. 日志 - 显示问题的相关日志摘录
4. 配置 - runtime.exs 的相关部分（删除敏感数据）
5. 环境 - OmniHSS 版本、数据库版本、操作系统版本
6. 影响 - 受影响的用户数量，业务影响
7. 用户示例 - 显示问题的特定 IMSI

关键与非关键

关键问题立即升级）：

- 系统完全崩溃
- 所有用户无法连接
- 数据库损坏
- 安全漏洞

非关键问题记录并在工作时间内升级）：

- 可以解决的单一用户问题
- 可管理的性能下降
- 增强请求
- 文档问题

常见错误消息参考

身份验证错误

错误消息	原因	解决方案
"身份验证向量生成失败"	缺少或无效的密钥集	检查 密钥集配置
"SQN 同步失败"	SQN 不同步	等待重新同步
"用户未找到"	无效的 IMSI	验证 IMSI，配置用户
"用户已禁用"	enabled=false	启用用户

Diameter 错误

错误消息	原因	解决方案
"Diameter 对等体连接超时"	网络问题	检查网络连接
"CER/CEA 交换失败"	配置不匹配	验证 Diameter 配置
"不支持的应用程序"	对等体不支持所需的应用	检查对等体应用
"TLS 握手失败"	证书问题	检查证书

数据库错误

错误消息	原因	解决方案
"连接被拒绝"	数据库宕机	启动数据库
"身份验证失败"	凭据错误	修复凭据
"没有可用连接"	池耗尽	增加池大小
"查询超时"	慢查询	优化查询

API 错误

错误消息	原因	解决方案
"key_set_id 不存在"	无效的外键	首先创建密钥集
"IMSI 已被占用"	重复的 IMSI	使用不同的 IMSI 或删除现有的
"验证错误"	输入无效	检查字段格式和要求

← 返回操作指南 | 下一步：API 参考 →

OmniHSS Webhook 集成

← 返回操作指南

目录

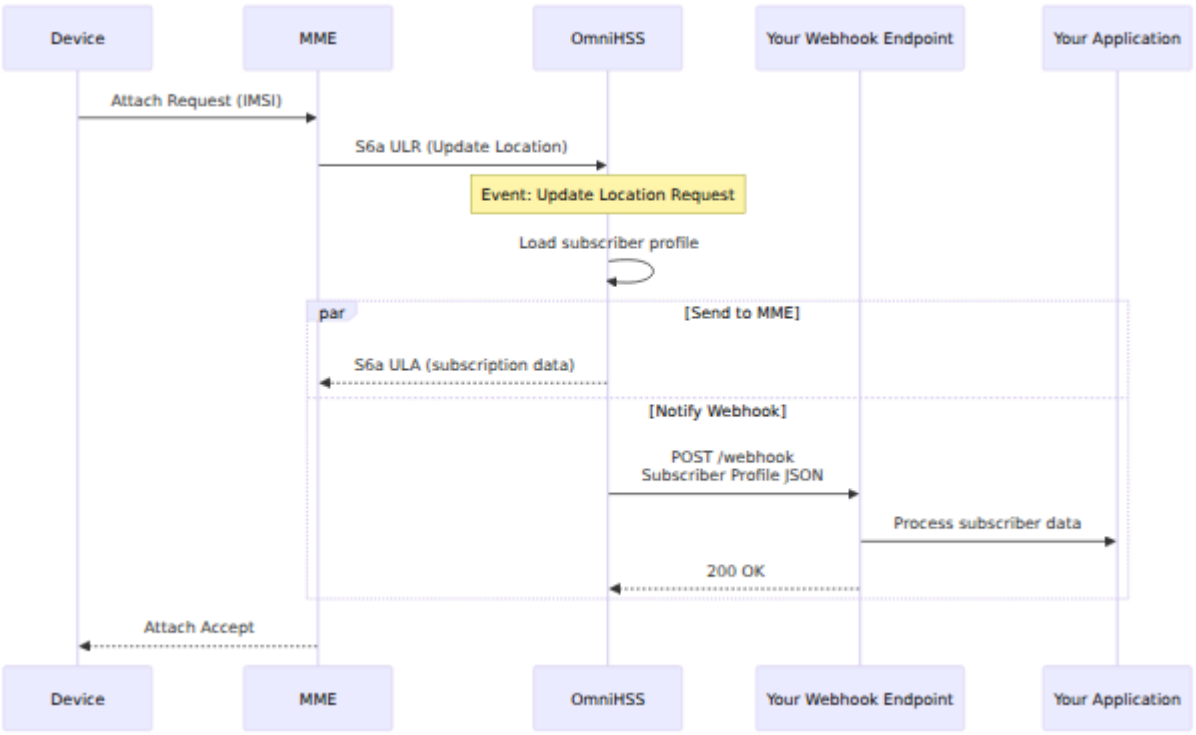
- 概述
- Webhook 的工作原理
- Webhook 事件
- Webhook 负载
- 配置
- 用例
- 安全考虑
- 故障排除

概述

OmniHSS 支持 **webhooks** 以实时通知外部系统有关订阅者事件。当发生特定事件（例如位置更新、身份验证请求或 IMS 注册）时，OmniHSS 可以向您配置的 webhook 端点发送 HTTP POST 请求，□含完整的订阅者配置文件数据。

什么是 **Webhooks**?

Webhooks 是 HTTP 回调，允许 OmniHSS 在事件发生时将通知推送到您的应用程序，而无需您的应用程序轮询 HSS API 以获取更改。



主要好处

- 实时通知 - 当订阅者事件发生时立即获取更新
- 完整的订阅者数据 - 每个 webhook 包含完整的订阅者配置文件（与 `GET /api/subscriber` 相同）
- 事件驱动的自动化 - 根据网络事件触发工作流、分析或配置
- 减少轮询 - 无需持续查询 API 以获取订阅者状态更改
- 集成灵活性 - 将 OmniHSS 连接到计费系统、分析平台或自定义应用程序

Webhook 的工作原理

事件流程

1. 事件发生 - 订阅者执行某个操作（附加、位置更新、IMS 注册等）
2. **HSS** 处理事件 - OmniHSS 正常处理 Diameter 请求/响应
3. **Webhook** 被触发 - 如果为此事件类型注册了 webhook，HSS 将向您的端点发送 HTTP POST
4. 包含订阅者数据 - Webhook 负载包含完整的订阅者配置文件作为 JSON
5. 您的应用程序响应 - 您的端点应返回 HTTP 200-299 以确认接收

交付保证

- 尽力而为的交付 - Webhooks 是异步发送的，不会阻塞网络操作
- 超时 - Webhook 请求在 5 秒后超时
- 不重试 - 如果您的端点不可用或返回错误，则不会重试 webhook
- 顺序不保证 - 在高负载下，事件可能会乱序到达

重要：网络操作（身份验证、位置更新等）不依赖于 webhook 的交付。如果您的 webhook 端点宕机，订阅者服务将正常继续。

Webhook 事件

OmniHSS 可以为以下事件触发 webhooks:

EPC/LTE 事件

事件	触发	描述
`update_location_request`	S6a ULR	订阅者附加或执行跟踪区域更新
`authentication_information_request`	S6a AIR	网络请求订阅者的身份验证向量
`purge_request`	S6a PUR	MME 移除订阅者上下文（设备关机、分离）
`cancel_location_answer`	S6a CLA	MME 确认订阅者注销

IMS 事件

事件	触发	描述
`ims_registration`	Cx SAR	订阅者注册 IMS/VoLTE 服务
`ims_deregistration`	Cx SAR (de-reg)	订阅者注销 IMS
`ims_profile_request`	Sh UDR	应用服务器请求订阅者 IMS ��置文件

策略事件 (PCRF)

事件	触发	描述
`policy_request`	Gx CCR	P-GW 请求订阅者数据会话的策略
`media_authorization`	Rx AAR	P-CSCF 请求 IMS 通话的媒体授权

多 IMSI 事件

事件	触发	描述
`imsi_switch`	ULR for different IMSI on same SIM	设备在多 IMSI SIM 上切换到不同的 IMSI

Webhook 负载

请求格式

当事件发生时，OmniHSS 向您配置的 webhook URL 发送 HTTP POST 请求：

```
POST /your-webhook-endpoint HTTP/1.1
Host: your-server.com
Content-Type: application/json
X-OmniHSS-Event: update_location_request
X-OmniHSS-Event-ID: 550e8400-e29b-41d4-a716-446655440000
X-OmniHSS-Timestamp: 2025-01-15T14:30:00Z

{
  "event": "update_location_request",
  "event_id": "550e8400-e29b-41d4-a716-446655440000",
  "timestamp": "2025-01-15T14:30:00Z",
  "subscriber": {
    "id": "d290f1ee-6c54-4b01-90e6-d701748f0851",
    "imsi": "001001123456789",
    "enabled": true,
    "ims_enabled": true,
    "msisdns": [
      {"id": "16fd2706-8baf-433b-82eb-8c7fada847da", "msisdn":
"14155551001"},
      {"id": "27ae3807-9cbf-4a4c-93fc-9d8afdb958eb", "msisdn":
"14155551002"}
    ],
    "sim": {
      "id": "3f2504e0-4f89-41d3-9a0c-0305e82c3301",
      "iccid": "8991101200003204510",
      "is_esim": false
    },
    "key_set": {
      "id": "6ba7b810-9dad-41d1-80b4-00c04fd430c8",
      "amf": "8000"
    },
    "epc_profile": {
      "id": "e58ed763-928c-4155-bee9-fdbaaadc15f3",
      "name": "Premium 100Mbps",
      "ue_ambr_dl_kbps": 100000,
      "ue_ambr_ul_kbps": 50000
    },
    "ims_profile": {
      "id": "7c9e6679-7425-40de-944b-e07fc1f90ae7",
      "name": "Standard VoLTE"
    },
    "roaming_profile": {
      "id": "9c858901-8a57-4791-81fe-4c455b099bc9",
```

```
      "name": "International Roaming Allowed"
    },
    "subscriber_state": {
      "mme_host": "mme-01.example.com",
      "mme_realm": "epc.mnc001.mcc001.3gppnetwork.org",
      "visited_plmn": "001001",
      "last_update": "2025-01-15T14:30:00Z"
    },
    "custom_attributes": {
      "account_type": "premium",
      "billing_plan": "unlimited"
    }
  },
  "event_context": {
    "visited_plmn": "310410",
    "mme_host": "mme-roaming.example.com",
    "location_update_type": "initial_attach"
  }
}
```

负载结构

字段	类型	描述
`event`	string	事件类型（例如，`update_location_request`）
`event_id`	string	此 webhook 交付的唯一 UUID
`timestamp`	string	事件发生时的 ISO 8601 时间戳
`subscriber`	object	完整的订阅者配置文件（与 `GET /api/subscriber/:id` 相同）
`event_context`	object	附加的事件特定上下文数据

事件上下文字段

`event_context` 对象包含事件特定的信息：

对于 **`update_location_request`**：

```
{
  "visited_plmn": "310410",
  "mme_host": "mme-roaming.example.com",
  "mme_realm": "epc.mnc410.mcc310.3gppnetwork.org",
  "location_update_type": "initial_attach"
}
```

□□ `imsi_switch`:

```
{
  "previous_imsi": "001001111111111",
  "new_imsi": "310410222222222",
  "sim_id": "3f2504e0-4f89-41d3-9a0c-0305e82c3301",
  "previous_mme_host": "mme-home.example.com",
  "new_mme_host": "mme-roaming.example.com"
}
```

□□ `ims_registration`:

```
{
  "scscf_host": "scscf-01.ims.example.com",
  "public_identities": [
    "sip:001001123456789@ims.mnc001.mcc001.3gppnetwork.org",
    "sip:+14155551001@ims.example.com",
    "tel:+14155551001"
  ]
}
```

HTTP 头

头	描述	示例
`Content-Type`	始终为 `application/json`	`application/json`
`X-OmniHSS-Event`	事件类型	`update_location_request`
`X-OmniHSS-Event-ID`	唯一事件标识符	UUID
`X-OmniHSS-Timestamp`	事件时间戳	ISO 8601 格式
`User-Agent`	OmniHSS 版本	`OmniHSS/1.0`

配置

注册 Webhooks

Webhooks 通过 OmniHSS API 配置。

注册一个 Webhook

```
curl -k -X POST https://hss.example.com:8443/api/webhook \
  -H "Content-Type: application/json" \
  -d '{
    "webhook": {
      "url": "https://your-server.com/omnihss-webhook",
      "events": [
        "update_location_request",
        "ims_registration",
        "imsi_switch"
      ],
      "enabled": true,
      "description": "生产计费系统 webhook"
    }
  }'
```

响应：

```
{
  "data": {
    "id": "4e5d6c7b-8a9f-4b0c-8d1e-2f3a4b5c6d7e",
    "url": "https://your-server.com/omnihss-webhook",
    "events": [
      "update_location_request",
      "ims_registration",
      "imsi_switch"
    ],
    "enabled": true,
    "description": "生产计费系统 webhook",
    "created_at": "2025-01-15T14:00:00Z"
  }
}
```

列出 Webhooks

```
curl -k https://hss.example.com:8443/api/webhook
```

更新 Webhook

```
curl -k -X PUT https://hss.example.com:8443/api/webhook/4e5d6c7b-
8a9f-4b0c-8d1e-2f3a4b5c6d7e \
  -H "Content-Type: application/json" \
  -d '{
    "webhook": {
      "enabled": false
    }
  }'
```

删除 **Webhook**

```
curl -k -X DELETE
https://hss.example.com:8443/api/webhook/4e5d6c7b-8a9f-4b0c-8d1e-
2f3a4b5c6d7e
```

Webhook 端点要求

您的 webhook 端点必须:

1. 接受 **POST** 请求，并且 `Content-Type: application/json`
2. 快速响应 - 在 5 秒内返回 HTTP 200-299
3. 是幂等的 - 优雅地处理重复交付
4. 使用 **HTTPS** - 为了安全，使用 TLS/SSL 端点（推荐）
5. 验证负载 - 验证请求来自 OmniHSS（请参见安全部分）

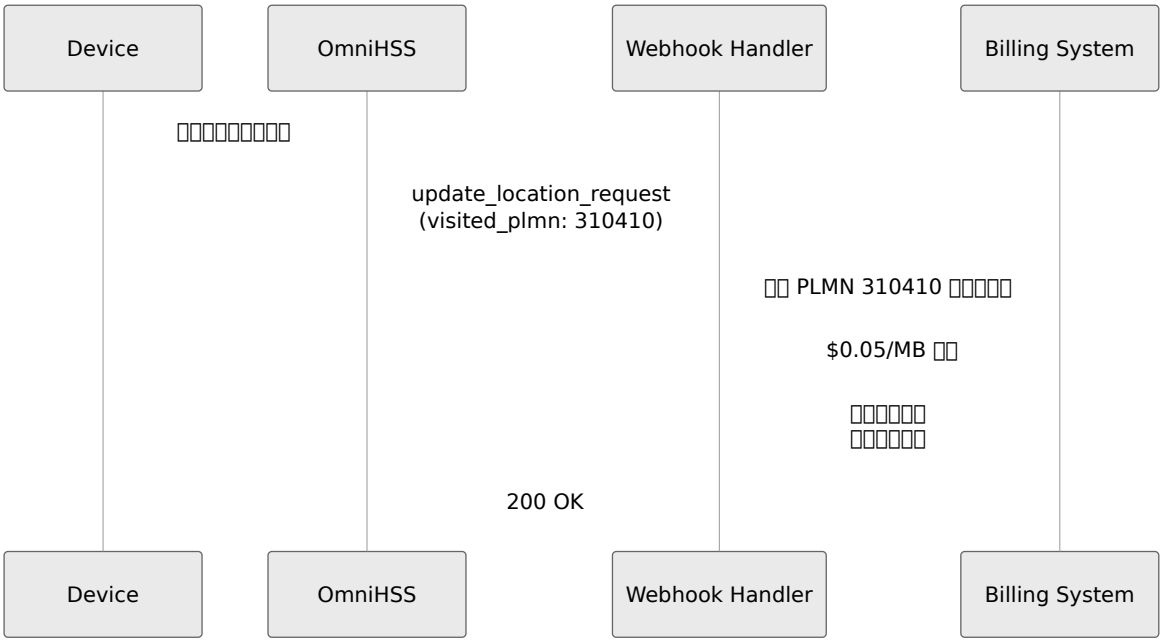
示例 **Webhook** 处理程序 **(Node.js/Express):**

```
const express = require('express');
const app = express();

app.post('/omnihss-webhook', express.json(), (req, res) => {
  const { event, subscriber, event_context } = req.body;

  console.log(`收到事件: ${event}`);
  console.log(`订阅者 IMSI: ${subscriber.imsi}`);

  // 处理订阅者数据
  // ... 您的业务逻辑在这里 ...

  // 立即响应以确认接收
  res.status(200).json({ received: true });

  // 在响应后处理异步处理
  processWebhook(req.body).catch(console.error);
});

async function processWebhook(payload) {
  // ???的异步处理逻辑
  // 例如，更新计费系统，触发分析等
}

app.listen(3000);
```

用例

1. 实时计费和使用跟踪

跟踪订阅者的网络使用情况，并实时触发计费事件。


Device
OmniHSS
Webhook Handler
Billing System
□□□□□□□□□□
update_location_request
(visited_plmn: 310410)
□□ PLMN 310410 □□□□□
$0.05/MB □□
□□□□□□□
□□□□□□□
200 OK
Device
OmniHSS
Webhook Handler
Billing System

□□:

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

2. □□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□: □□□□□□□□□□

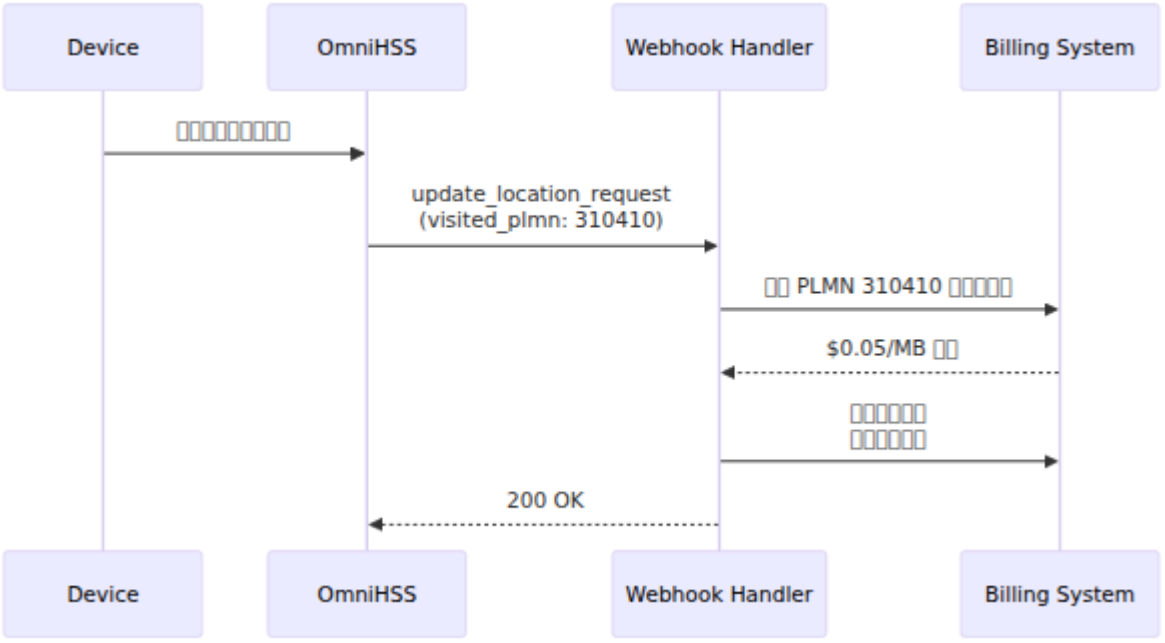
```
// Webhook 处理程序将数据传递给分析平台
app.post('/omnihss-webhook', async (req, res) => {
  const { event, subscriber, event_context } = req.body;

  if (event === 'update_location_request') {
    await analytics.track({
      event: 'subscriber_location_update',
      imsi: subscriber.imsi,
      visited_plmn: event_context.visited_plmn,
      timestamp: req.body.timestamp,
      profile: subscriber.epc_profile.name
    });
  }

  res.status(200).send();
});
```

分析仪表板：

- 每个 MME 的活跃订阅者
- 按国家的漫游订阅者
- 服务等级分布
- IMS 注册成功率

3. 欺诈检测和安全

实时检测可疑活动模式并触发自动响应。



欺诈检测场景：

1. 快速位置变化
   - 订阅者在 A 国附加
   - 30 分钟后，在 B 国附加（身体上不可能）
   - 行动：标记账户，向安全团队发送警报
2. **IMSI** 切换滥用
   - 同一 SIM 上多次快速 IMSI 切换
   - 可能的 SIM 克隆或未经授权的多 IMSI 使用
   - 行动：禁用 SIM 上的所有 IMSI，通知欺诈团队
3. 未经授权的漫游
   - 订阅者漫游到被阻止的国家（制裁、欺诈风险）
   - 行动：自动禁用订阅者，阻止网络访问

示例实现：

```
@app.route('/omnihss-webhook', methods=['POST'])
def webhook_handler():
    data = request.json
    subscriber = data['subscriber']
    event_context = data.get('event_context', {})

    if data['event'] == 'update_location_request':
        visited_plmn = event_context.get('visited_plmn')

        # 检查被阻止的国家
        if visited_plmn in BLOCKED_PLMNS:
            disable_subscriber(subscriber['imsi'])
            alert_security_team(subscriber, '漫游到被阻止的 PLMN')

        # 检查不可能的旅行
        if is_impossible_travel(subscriber['imsi'], visited_plmn):
            flag_for_review(subscriber['imsi'])
            alert_fraud_team(subscriber, '检测到不可能的旅行')

    return jsonify({'status': 'ok'}), 200
```

4. 配置自动化

根据网络事件自动配置或更新订阅者服务。

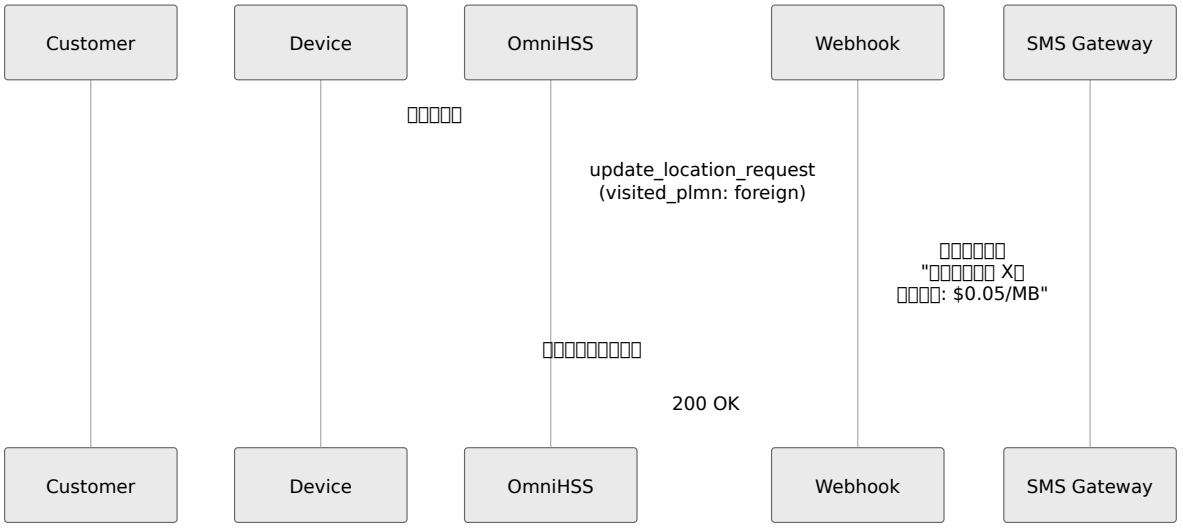
用例：当订阅者首次使用 VoLTE 时自动启用 IMS

```
app.post('/omnihss-webhook', async (req, res) => {
  const { event, subscriber } = req.body;

  if (event === 'ims_registration' && !subscriber.ims_enabled) {
    // □□ IMS □□ - □□□□ IMS
    await omnihss.updateSubscriber(subscriber.id, {
      ims_enabled: true,
      custom_attributes: {
        ...subscriber.custom_attributes,
        volte_activated_at: new Date().toISOString()
      }
    });

    // □□ CRM
    await crm.updateCustomer(subscriber.imsi, {
      features: ['volte']
    });
  }

  res.status(200).send();
});
```

5. □□□□

□□□□□□□□□□□□□□□□□□

□□: □□□□□□□□□□□□□



□□□□:

- "欢迎来到 [国家]！适用漫游费用。"
- "您已使用 80% 的数据配额"
- "您的设备上已激活 VoLTE 服务"
- "您的账户已升级为高级版"

6. 多 IMSI SIM 管理

跟踪和管理具有多 IMSI SIM 的订阅者，在他们切换 IMSI 时接收通知。

```
app.post('/omnihss-webhook', async (req, res) => {
  const { event, subscriber, event_context } = req.body;

  if (event === 'imsi_switch') {
    const { previous_imsi, new_imsi, sim_id } = event_context;

    // 记录 IMSI ⇄换以进行分析
    await db.logImsiSwitch({
      sim_id,
      from_imsi: previous_imsi,
      to_imsi: new_imsi,
      timestamp: req.body.timestamp
    });

    // 更新计费系统
    await billing.endSession(previous_imsi);
    await billing.startSession(new_imsi);

    // 如果切换过于频繁□潜在欺诈），则发出警报
    const switchCount = await db.getSwitchCount(sim_id, '24h');
    if (switchCount > 10) {
      await alertFraudTeam(`过度 IMSI 切换: SIM ${sim_id}`);
    }
  }

  res.status(200).send();
});
```

7. 与外部系统集成

将 OmniHSS 连接到第三方系统，无需轮询。

示例集成：

- **CRM** 系统 - 更新客户记录以反映服务使用情况
- 网络监控 - 将订阅者数据输入网络分析平台
- 计费系统 - 根据网络事件触发费用
- 工单系统 - 自动创建身份验证失败的工单
- 数据仓库 - 流式传输订阅者事件以进行大数据分析

安全考虑

Webhook 密钥/签名

为了验证 webhooks 来自 OmniHSS，请实现签名验证：

```
# 使用密钥配置 webhook
curl -k -X POST https://hss.example.com:8443/api/webhook \
  -H "Content-Type: application/json" \
  -d '{
    "webhook": {
      "url": "https://your-server.com/omnihss-webhook",
      "events": ["update_location_request"],
      "secret": "your-secret-key-here"
    }
  }'
```

OmniHSS 将包含一个 `X-OmniHSS-Signature` 头：

```
X-OmniHSS-Signature:
sha256=5d7a8f9b2c1e3a4d6f7e8b9c0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a
```

验证签名：

```
const crypto = require('crypto');

function verifyWebhook(req) {
  const signature = req.headers['x-omnihss-signature'];
  const secret = process.env.WEBHOOK_SECRET;
  const payload = JSON.stringify(req.body);

  const expectedSignature = 'sha256=' +
    crypto.createHmac('sha256', secret)
      .update(payload)
      .digest('hex');

  return crypto.timingSafeEqual(
    Buffer.from(signature),
    Buffer.from(expectedSignature)
  );
}

app.post('/omnihss-webhook', (req, res) => {
  if (!verifyWebhook(req)) {
    return res.status(401).json({ error: '无效的签名' });
  }

  // 处理 webhook...
  res.status(200).send();
});
```

最佳实践

1. 使用 **HTTPS** - 始终对 webhook 端点使用 TLS
2. 验证签名 - 验证 webhook 签名以防止欺骗
3. 速率限制 - 在 webhook 端点上实施速率限制
4. **IP** 白名单 - 限制 webhook 访问 OmniHSS IP 地址
5. 监控失败 - 跟踪 webhook 交付失败和错误
6. 数据清理 - 在处理之前验证和清理 webhook 负载
7. 安全凭据 - 在安全配置中存储 webhook 密钥（环境变量、秘密管理器）

数据隐私

Webhook 负载□含 敏感的订阅者信息：

- IMSI（订阅者身份）
- MSISDN（电话号码）
- 位置数据（访问的 PLMN、MME）
- 服务配置文件信息

合规要求：

- **GDPR** - 确保 webhook 数据的处理符合 GDPR
- 数据保留 - 实施适当的数据保留政策
- 访问控制 - 限制 webhook 端点访问
- 加密 - 对 webhook 传输使用 TLS
- 审计日志 - 记录所有 webhook 交付以确保合规

故障排除

Webhook 未收到

症状：

- 事件发生但 webhook 未触发
- Webhook 端点从未接收到请求

故障排除步骤：

1. 验证 **webhook** 是否启用：

   ```
   curl -k https://hss.example.com:8443/api/webhook
   # 检查 "enabled": true
   ```

2. 检查 **webhook** 事件配置：

- 确保事件类型□含在 webhook 的 `events` 列表中
- 示例：如果您希望 `ims_registration` 事件，请验证它是否在事件数组中

3. 查看 **HSS** 日志：

   - 检查 webhook 交付错误
   - 查找网络连接问题
   - 验证没有 DNS 解析失败

4. 测试端点可访问性：

```
curl -X POST https://your-server.com/omnihss-webhook \
  -H "Content-Type: application/json" \
  -d '{"test": true}'
```

Webhook 超时

症状：

- HSS 日志显示 webhook 超时错误
- Webhook 端点接收到请求但 HSS 标记为失败

解决方案：

1. 立即响应：

   - 在 5 秒内返回 HTTP 200
   - 在响应后异步处理数据

2. 优化端点性能：

```
// 不好 - 慢的同步处理
app.post('/webhook', (req, res) => {
  processData(req.body); // 阻塞 10 秒
  res.status(200).send();
});

// 好 - 响应后异步处理
app.post('/webhook', (req, res) => {
  res.status(200).send(); // 立即响应
  processData(req.body); // 异步处理
});
```

重复的 Webhooks

症状：

- 相同事件多次交付
- 重复交付的 `event_id` 相同

原因：

- 网络重试（尽管 OmniHSS 不会重试，但网络基础设施可能会）
- 为同一事件注册了多个 webhook

解决方案：

��用 `event_id` 实现幂等性：

```
const processedEvents = new Set();

app.post('/omnihss-webhook', (req, res) => {
  const eventId = req.body.event_id;

  if (processedEvents.has(eventId)) {
    // 已处理，跳过
    return res.status(200).json({ status: 'duplicate' });
  }

  processedEvents.add(eventId);

  // 处理 webhook...
  processWebhook(req.body);

  res.status(200).json({ status: 'processed' });
});
```

Webhook 返回错误

症状:

- 端点返回 HTTP 4xx 或 5xx
- HSS 日志 webhook 交付失败

常见错误:

1. **401 Unauthorized** - 签名验证失败
    - 检查 webhook 密钥是否与配置匹配
    - 验证签名计算算法
2. **400 Bad Request** - 无效负载
    - 检查 webhook 负载解析
    - 确保处理 Content-Type 头
3. **500 Internal Server Error** - 端点崩溃
    - 查看端点错误日志

- 添加错误处理和日志记录

解决方案：

添加全面的错误处理：

```
app.post('/omnihss-webhook', async (req, res) => {
  try {
    // 验证签名
    if (!verifyWebhook(req)) {
      return res.status(401).json({ error: 'Invalid signature' });
    }

    // 验证负载
    if (!req.body.event || !req.body.subscriber) {
      return res.status(400).json({ error: 'Invalid payload' });
    }

    // 处理 webhook
    await processWebhook(req.body);

    res.status(200).json({ status: 'ok' });

  } catch (error) {
    console.error('Webhook processing error:', error);
    // 返回 200 以防止重试，记录错误以供调查
    res.status(200).json({ status: 'error', message: error.message
});
  }
});
```

缺失的订阅者数据

症状：

- 收到 webhook 但订阅者对象不完整
- 预期字段为 null 或缺失

可能原因：

1. 订阅者未完全配置 - 某些配置可能是可选的（IMS、漫游）

2. 数据竞争条件 - 事件触发和 webhook 发送之间订阅者更新

解决方案：

优雅地处理可选字段：

```
const { subscriber } = req.body;

// 检查可选字段
const imsProfile = subscriber.ims_profile || { name: 'No IMS' };
const roamingProfile = subscriber.roaming_profile || { name: 'No
Roaming' };

// 处理缺失的 MSISDN
const msisdns = subscriber.msisdns || [];
```

监控和可观察性

Webhook 指标

跟踪 webhook 性能和可靠性：

需要监控的指标：

- Webhook 交付率（成功与失败）
- Webhook 延迟（从事件到端点响应的时间）
- 端点响应时间
- 按端点的错误率
- 每秒事件数

示例仪表板查询（**Prometheus/Grafana**）：

```
# Webhook 成功率
rate(omnihss_webhook_success_total[5m]) /
rate(omnihss_webhook_attempts_total[5m])

# Webhook 延迟
histogram_quantile(0.95, omnihss_webhook_duration_seconds)
```

Webhook 日志

启用详细的 webhook 日志记录以进行故障排除：

日志格式：

```
{
  "timestamp": "2025-01-15T14:30:00Z",
  "level": "info",
  "component": "webhook",
  "event_id": "550e8400-e29b-41d4-a716-446655440000",
  "webhook_id": "4e5d6c7b-8a9f-4b0c-8d1e-2f3a4b5c6d7e",
  "event_type": "update_location_request",
  "subscriber_imsi": "001001123456789",
  "endpoint": "https://your-server.com/omnihss-webhook",
  "http_status": 200,
  "duration_ms": 145,
  "error": null
}
```

← 返回操作指南 | 下一步：API 参考 →