OmniHSS □作指南

介绍

OmniHSS □一个为 4G LTE (EPC) 和 IMS (IP 多媒体子系统) 网络设计的家庭用户服务器 (HSS) 实现。作为移动网络的中央数据库和认证中心，OmniHSS 管理用户凭证、配置数据，并为数据和语音服务提供认证和授权服务。

OmniHSS 基于 Elixir 和 Erlang VM 构建，提供现代通信基础设施所需的高可用性、容错性和可扩展性。

什么是家庭用户服务器?

HSS □ LTE 和 IMS 网络中的关键组件，具有以下功能:

- 存储用户数据 - 凭证、配置资料和服务订阅
- 执行认证 - 验证尝试访问网络的用户
- 管理授权 - 控制用户可以访问的服务
- 跟踪位置 - 维护当前位置信息以进行路由
- 控制漫游 - 根据访问网络强制执行漫游政策
- 管理设备 - 作为设备身份注册 (EIR) 进行设备控制

关键特性

□作特性

- **S6a** 接口 - LTE/EPC 网络的认证和位置管理
- **Cx** 接口 - IMS 注册和认证
- **Sh** 接口 - IMS 配置��据访问和订阅通知
- **S13** 接口 - 设备身份检查 (OmniHSS 作为 EIR)
- **Gx** 接口 - 策略和计费控制 (OmniHSS 作为 PCRF)
- **Rx** 接口 - IMS 媒体策略控制 (OmniHSS 作为 PCRF)

- 漫游控制 - 按 PLMN 对数据和 IMS 漫游进行细粒度控制
- 多个 **MSISDN** - 支持每个用户多个电话号码
- **RESTful API** - 完整的集成配置 API (也被 OmniHLR 使用)
- **Web** 控制面板 - 实时监控和系统状态

网络元素集成

OmniHSS 与以下网络元素接口:

- **MME** (移动管理实体) - LTE 移动性和会话管理
- **P-GW** (PDN 网关) - 从 OmniHSS 接收策略 (PCRF 功能)
- **P-CSCF** (代理呼叫会话控制功能) - IMS 媒体授权
- **I-CSCF** (查询 CSCF) - IMS 路由查询
- **S-CSCF** (服务 CSCF) - IMS 注册和认证
- **AS** (应用服务器) - IMS 用户数据访问
- **OmniHLR** - 通过 API 与 OmniHSS 通信的传统 HLR

文档结构

本操作指南分为以下文档:

核心文档

- 架构概述 - 系统架构、组件和 Diameter 堆栈
- 配置指南 - 完整的配置参考及示例
- 实体关系 - 数据模型和实体关系

□作指南

- 控制面板 - 使用基于网络的监���界面
- 指标与监控 - 系统监控和健康检查
- 故障排除指南 - 诊断和解决常见问题
- **API** 参考 - 完整的 API 端点文档
- **Webhooks** - 实时事件通知和集成

特性文档

- 配置管理 - EPC、IMS、APN 和漫游配置
- 漫游控制 - 配置漫游政策
- 协议流程 - Diameter 协议程序和消息流程
- **PCRF** - 策略和计费规则功能 (Gx/Rx 接口、QoS、VoLTE)
- **EIR** - 设备身份注册 (S13 接口、IMEI 验证)
- 多 **MSISDN** 和多 **IMSI** 特性 - 支持多个电话号码和多个 IMSI

高可用性

- **Galera** 数据库复制 - 用于 HA 部署的多节点集群

□作快速入门

访问系统

控制面板 **(Web** 界面**)**

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

控制面板提供对用户和 Diameter 对等体的实时监控。

API 端点

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

RESTful API 允许配置和用户管理。

关键配置文件

- `config/runtime.exs` - 运行时配置 (数据库、Diameter、网络设置)
- `priv/cert/` - HTTPS 和 Diameter 的 TLS 证书

基本□作

1. 检查系统状态 - 访问控制面板概览页面
2. 监控 **Diameter** 对等体 - 访问控制面板 Diameter 页面
3. 查询用户 - 使用 API 端点 `/api/subscriber/imsi/:imsi`
4. 查看数据库 - 连接到配置的主机名上的 SQL 数据库

支持与故障排除

日志文件

系统日志输出到 stdout/stderr，可以被您的进程管理器 (systemd、supervisord 等) 捕获。

常见检查

- **Diameter** 连接性 - 检查 Diameter 页面以获取对等体状态
- 数据库连接性 - 验证 runtime.exs 中的数据库配置
- 用户认证失败 - 检查用户状态以获取失败计数

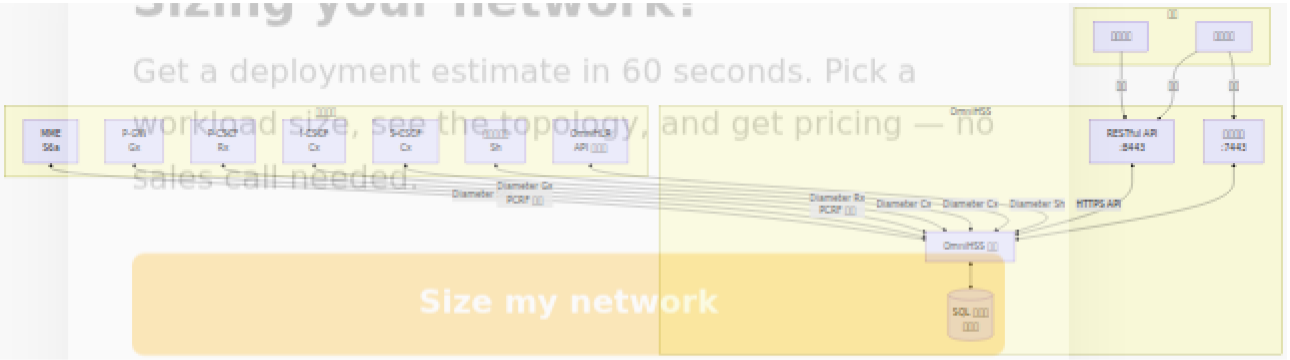
健康监控

- **API** 健康检查 - `GET /api/status`
- 控制面板 - 访问任何控制面板页面
- 数据库 - 连接到 SQL 数据库并验证表访问

安全考虑

- 需要 **TLS** - API 和控制面板均使用 HTTPS
- 证书管理 - `priv/cert/` 中的证书必须有效
- 数据库安全 - 在 runtime.exs 中保护数据库凭证
- 网络隔离 - Diameter 接口应在管理网络上
- **API** 认证 - 考虑为生产使用实施认证

一览架构



下一步

有关详细的操作程序，请参考特定文档部分：

- 从 架构概述 开始，了解系统组件
- 查看 配置指南 以自定义您的部署
- 探索 控制面板 以进行日常监控
- 查阅 **API** 参考 以进行配置自动化

文档版本： 1.0
维护者： Omnitouch 运营团队

EIR（设备身份注册）

概述

HSS 包含一个内置的 EIR（设备身份注册），提供移动设备的设备身份验证。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，然后才允许网络访问。

主要功能

- **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**：完整的 CRUD 操作用于 EIR 规则管理

架构

Diameter 接口

接口	应用 **ID**	对等方	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□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□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	允许并监控

S13 接口

支持的□作

设备身份检查请求（**ECR**）/设备身份检查应答（**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 **AVPs**：

- 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 □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□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（��� 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 **AVPs**：

- 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. 未知设备策略
```

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���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**: 未知设备策略（最终后备）

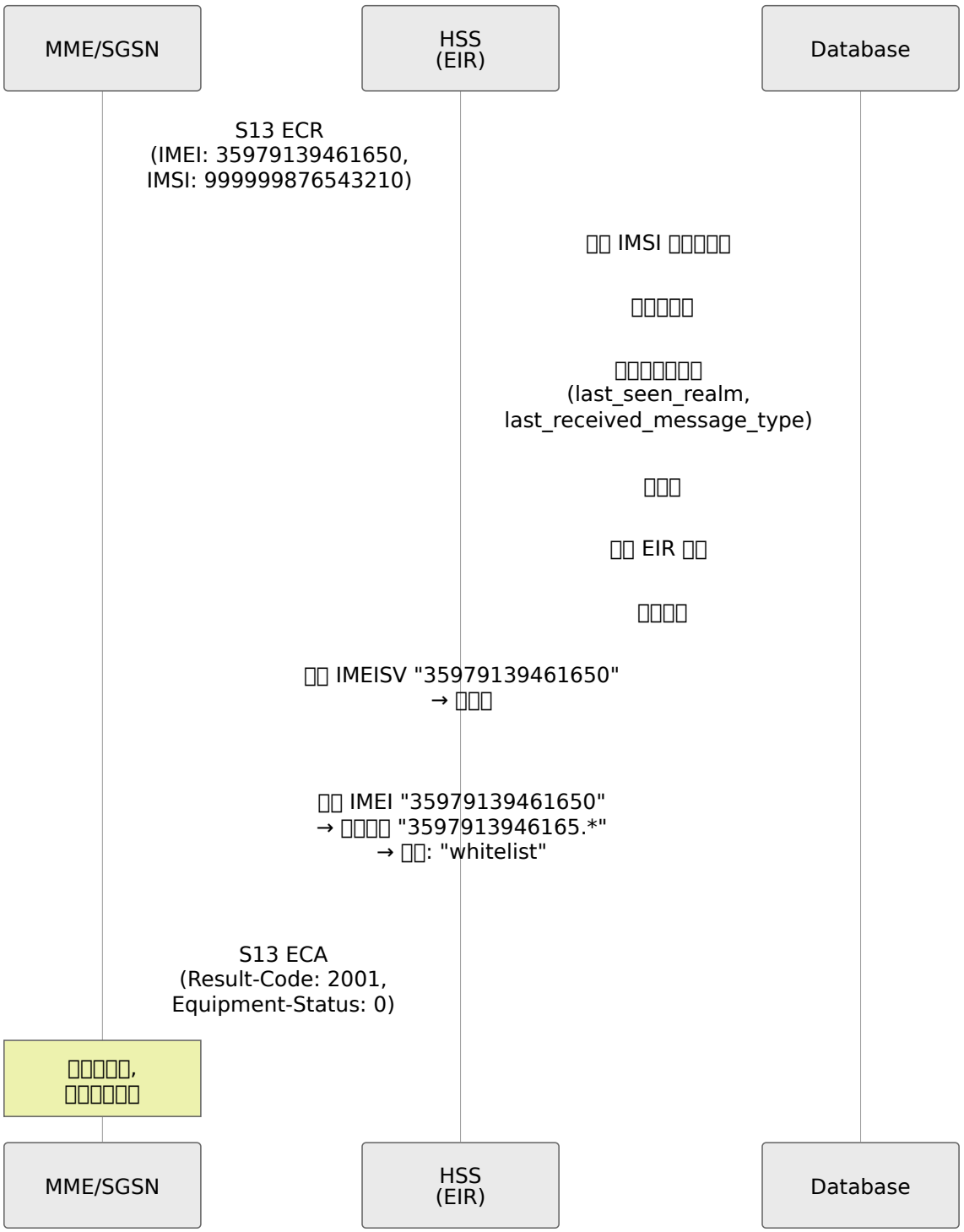
- 配置设置：`eir_unknown_equipment_behaviour`
- 选项：
  - `:whitelist` - 允许未知设备（宽松）
  - `:blacklist` - 阻止未知设备（严格）
  - `:greylist` - 观察未知设备（适中）
  - `:reject_unknown_equipment` - 返回错误 5422（严格）

正则表达式模式示例

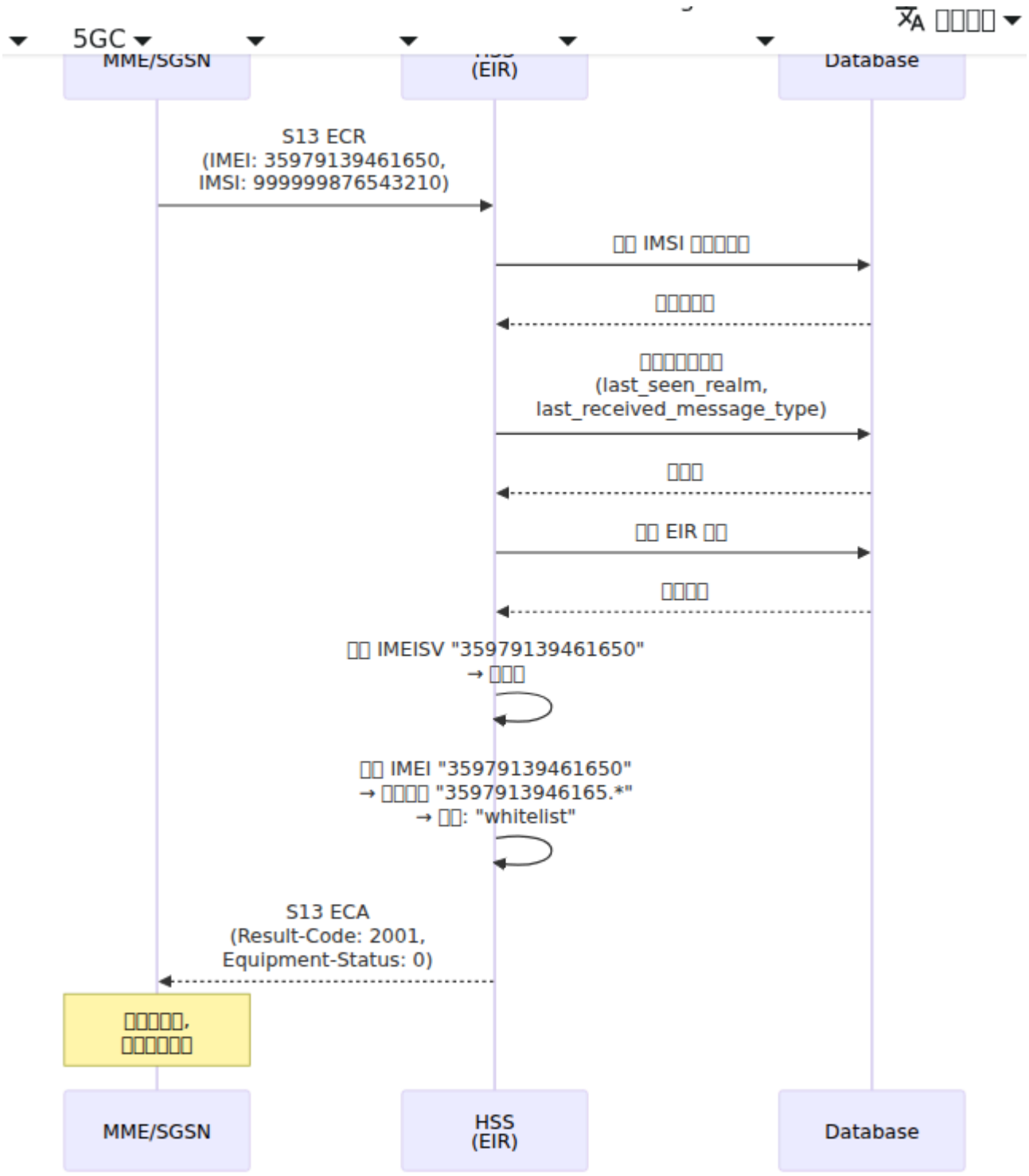
模式	匹配	用例
`"35979139461650"`	精确 IMEI	单个设备白名单/黑名单
`"3597913946165.*"`	IMEI 前缀通配符	制造商/型号范围
`"35979139461611 08"`	精确 IMEISV	特定设备与软件版本
`"999999876543210"`	IMSI	阻止该订阅者的所有设备
`"359791.*"`	TAC 通配符	整个设备类型分配

□□□□□□

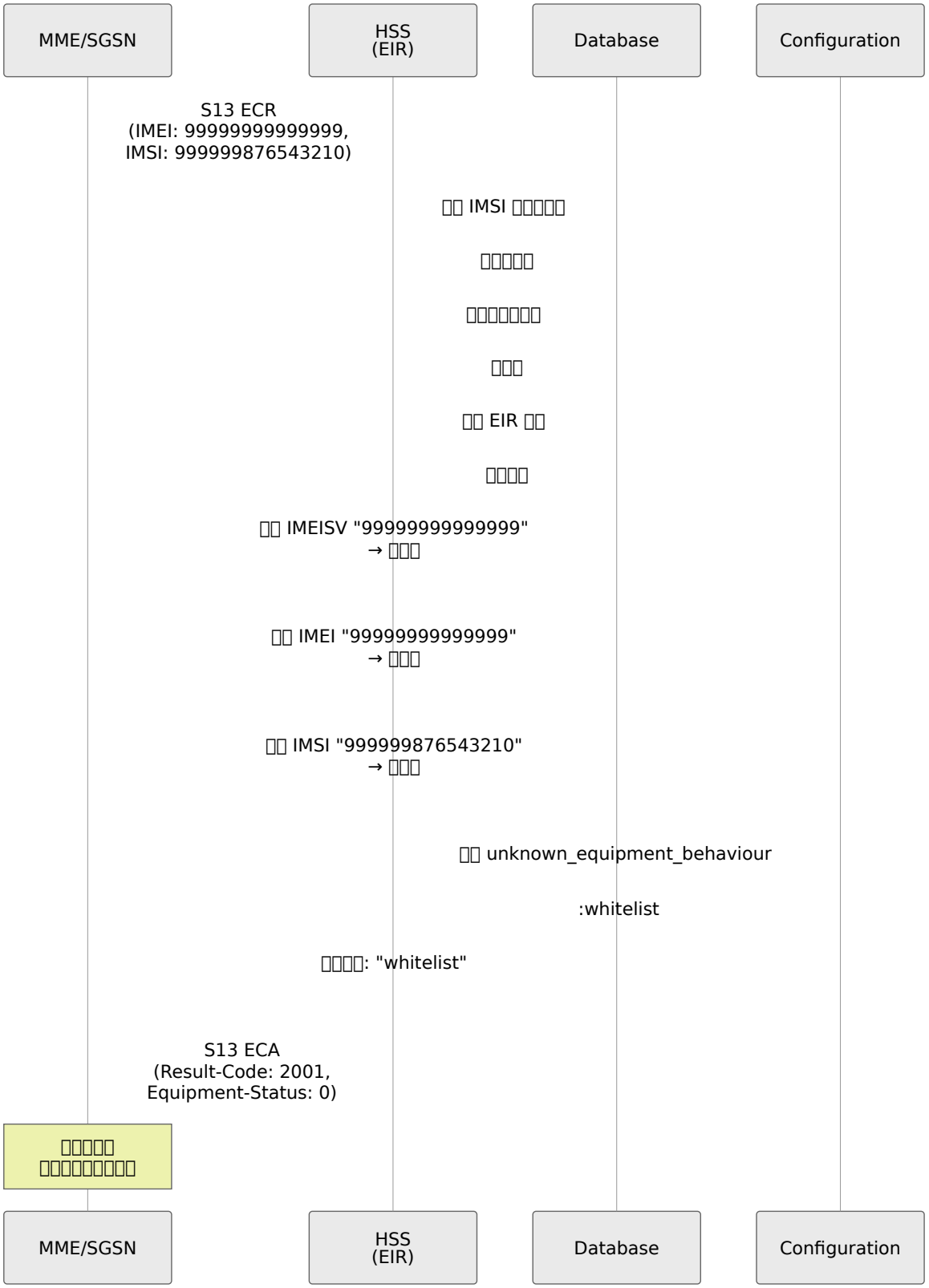
□□ 1□□□□□ - □□□□□ IMEI


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

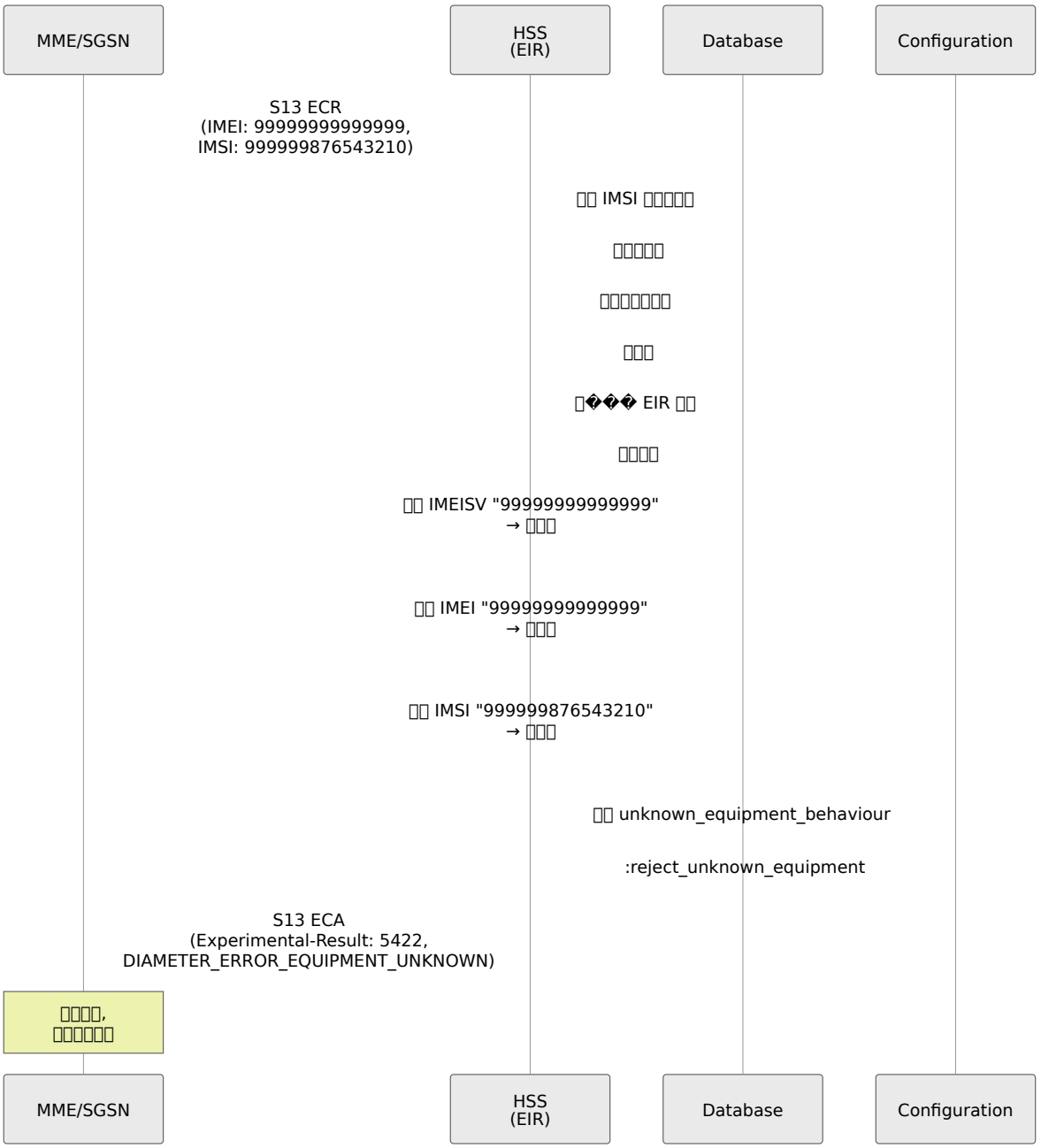
□□ 2□□□□□ - □□□ IMEI□□□□□□


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

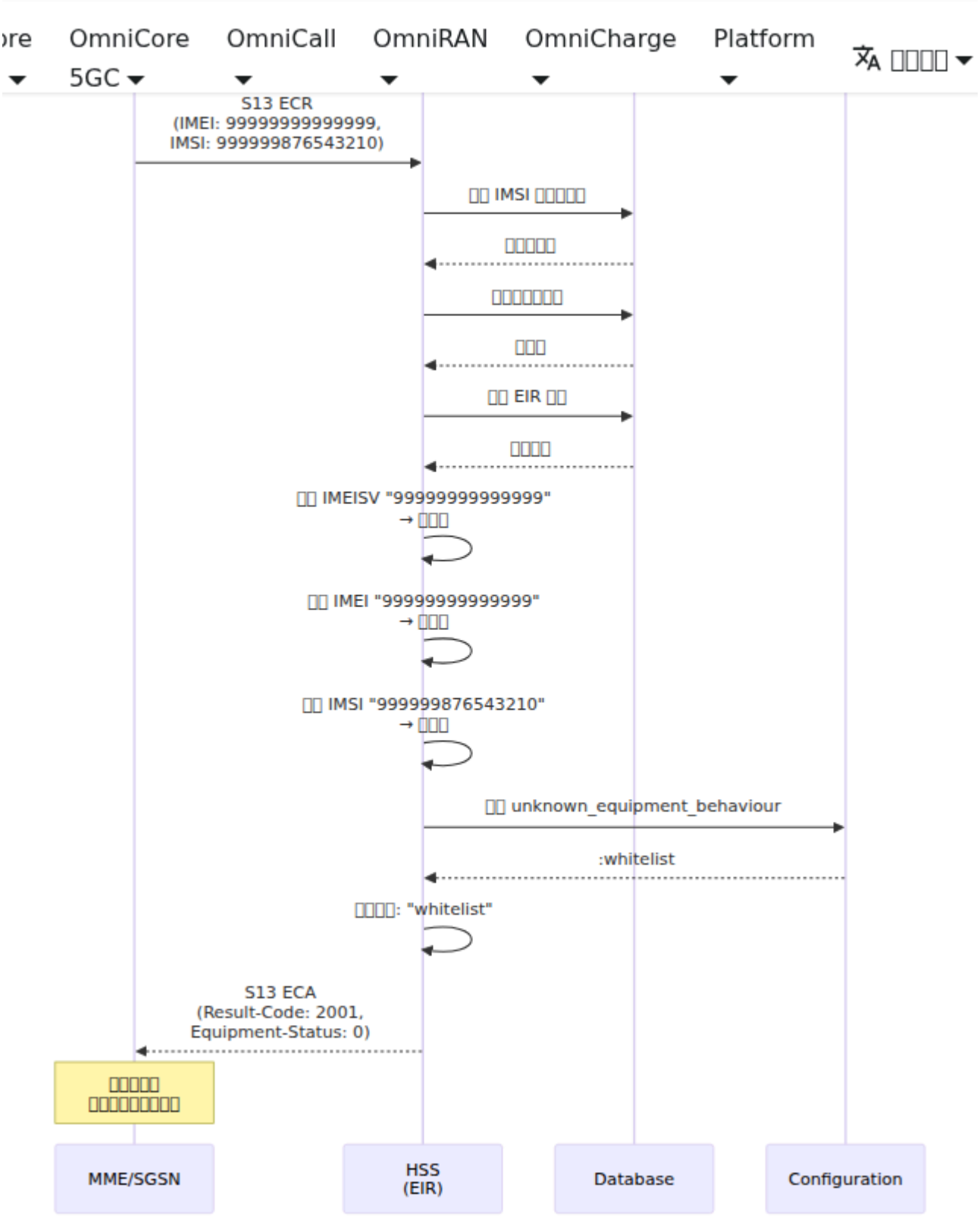
□□ 3□□□□□ - □□□□□□□□□□□□


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

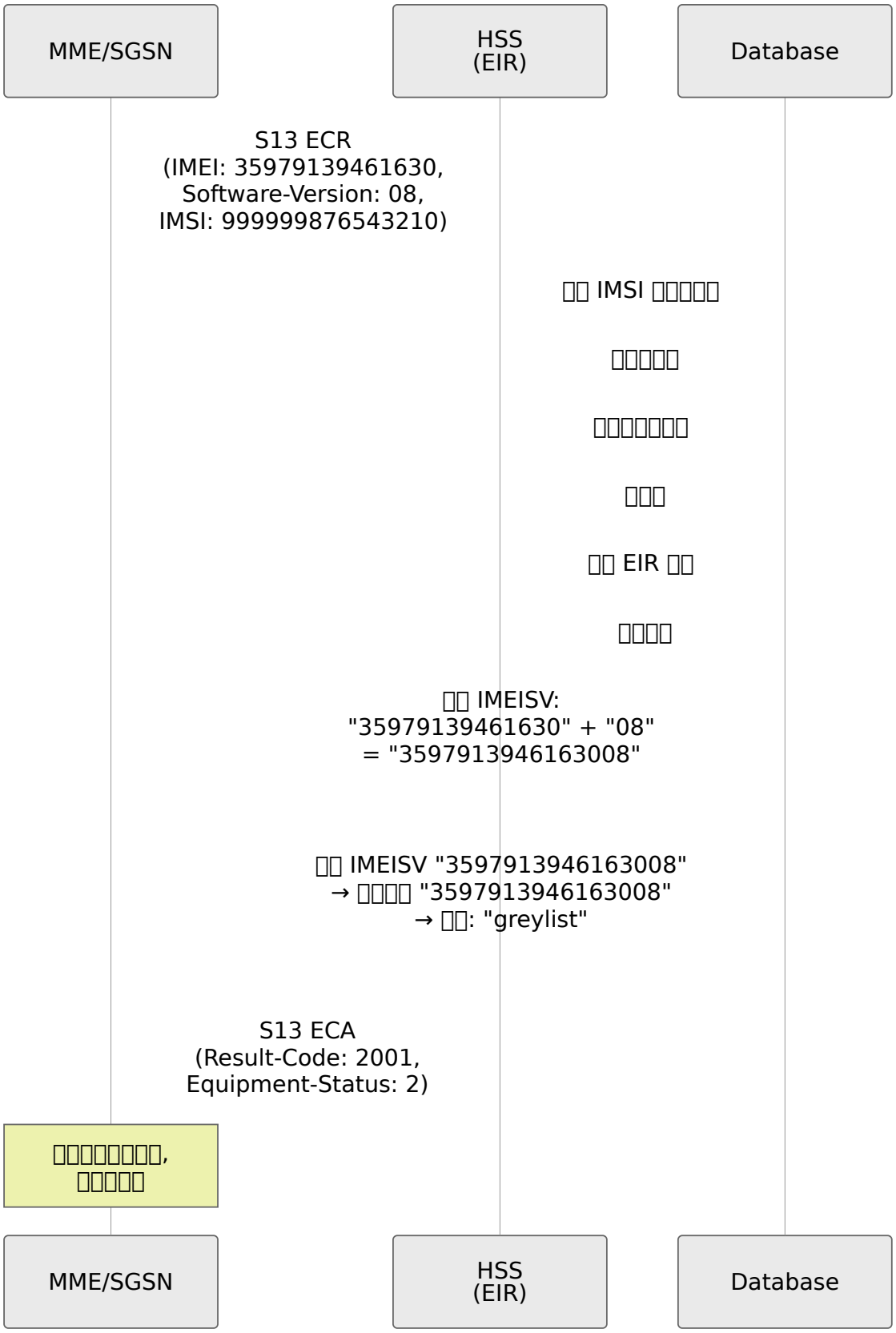
□□ 4□□□□□ - □□□□□□□□□□


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

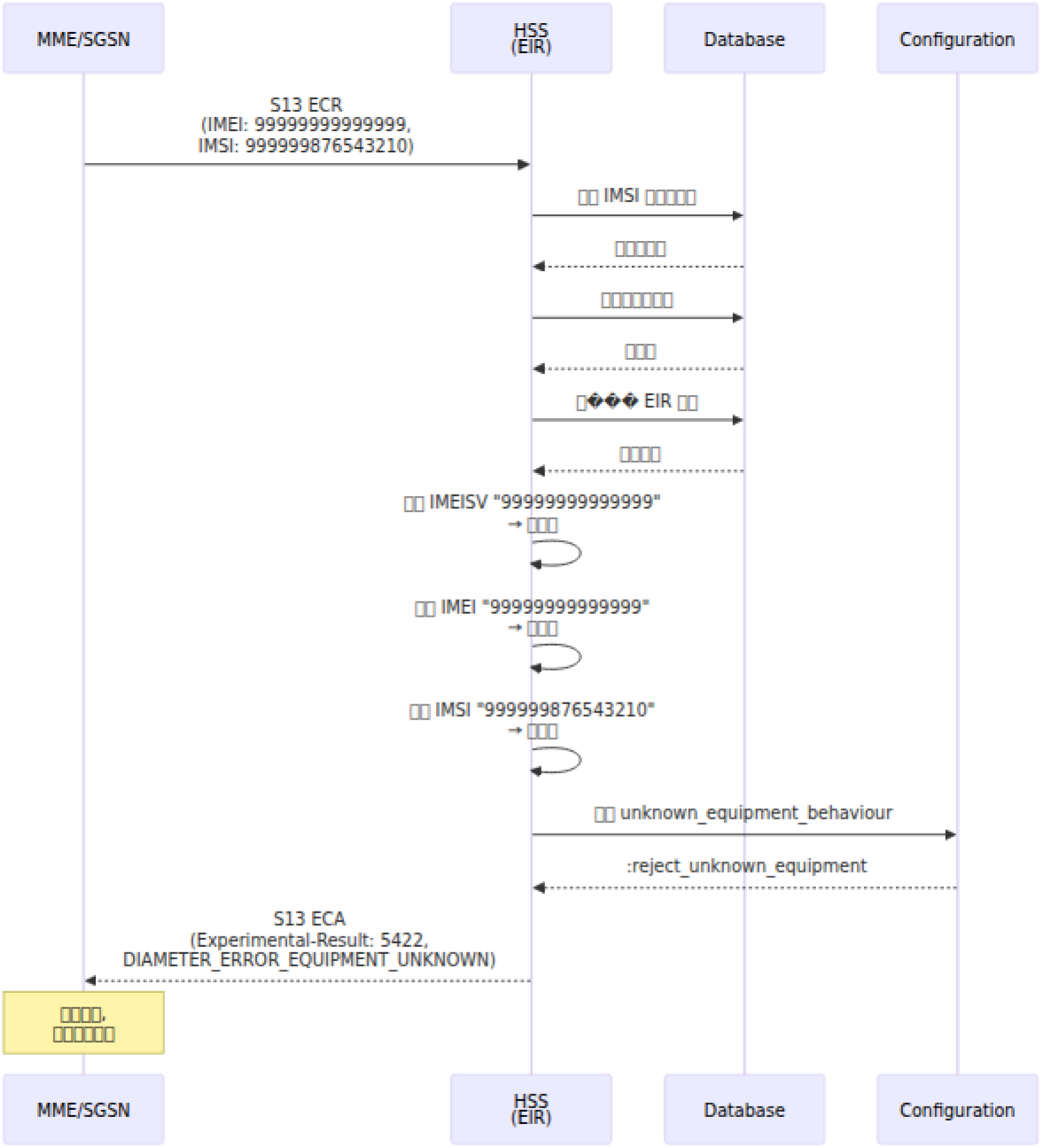
□□ 5□□□□□ - □□□□□


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

基础路径: `/api/eir/rule`

列出所有 **EIR** 规则

请求:

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

响应（HTTP 200）：

```
{
  "data": [
    {
      "id": 1,
      "action": "whitelist",
      "regex": "3597913946165.*",
      "inserted_at": "2025-01-15T10:30:00Z",
      "updated_at": "2025-01-15T10:30:00Z"
    },
    {
      "id": 2,
      "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": 1,
    "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`: 必需，必须□ "whitelist"、"blacklist" 或 "greylist"
- `regex`: 必需，必须□有效的正则表达式模式，在所有规则中唯一

响应（HTTP 201）：

```
{
  "data": {
    "id": 3,
    "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": 3,
    "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": 3,
    "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查
- 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	DIAMETER_ERROR_EQUIPMENT_UNKNOWN	未找到订阅者或未知设备被拒绝
5012	实验	DIAMETER_ERROR_UNKNOWN	处理错误

用例

1. 被盗设备管理

场景：设备报告被盗

□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

□作:

```
POST /api/eir/rule
{
  "action": "whitelist",
  "regex": "359791394.*"  # 制��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）

□作:

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

结果：任何使用此 SIM 的设备均被阻止

4. 测试设备灰名单

场景：在生产中监控测试设备

□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

□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** 协议处理程序 - 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□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。对于冲突的模式：

```
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 查找查询以防止枚举攻击

相关文档

- Diameter 协议 - S13 协议规范
- API 参考 - 完整的 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+序列为 359791394616* 的设备
359791394616140	`35979139461614.`	序列 359791394616141-9 的所有校验位
35979139461640	`35979139461640`	精确 IMEI 匹配
3597913946163008	`3597913946163008`	精确 IMEISV（IMEI + SV）匹配

PCRF（策略和计费规则功能）

概述

HSS 包含内置的 PCRF（策略和计费规则功能），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（数据包数据网络网关/策略和计费执行功能）的策略控制
- **Rx** 接口：IMS（IP 多媒体子系统）媒体流的授权和 QoS
- 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**：编程控制策略执行和规则管理

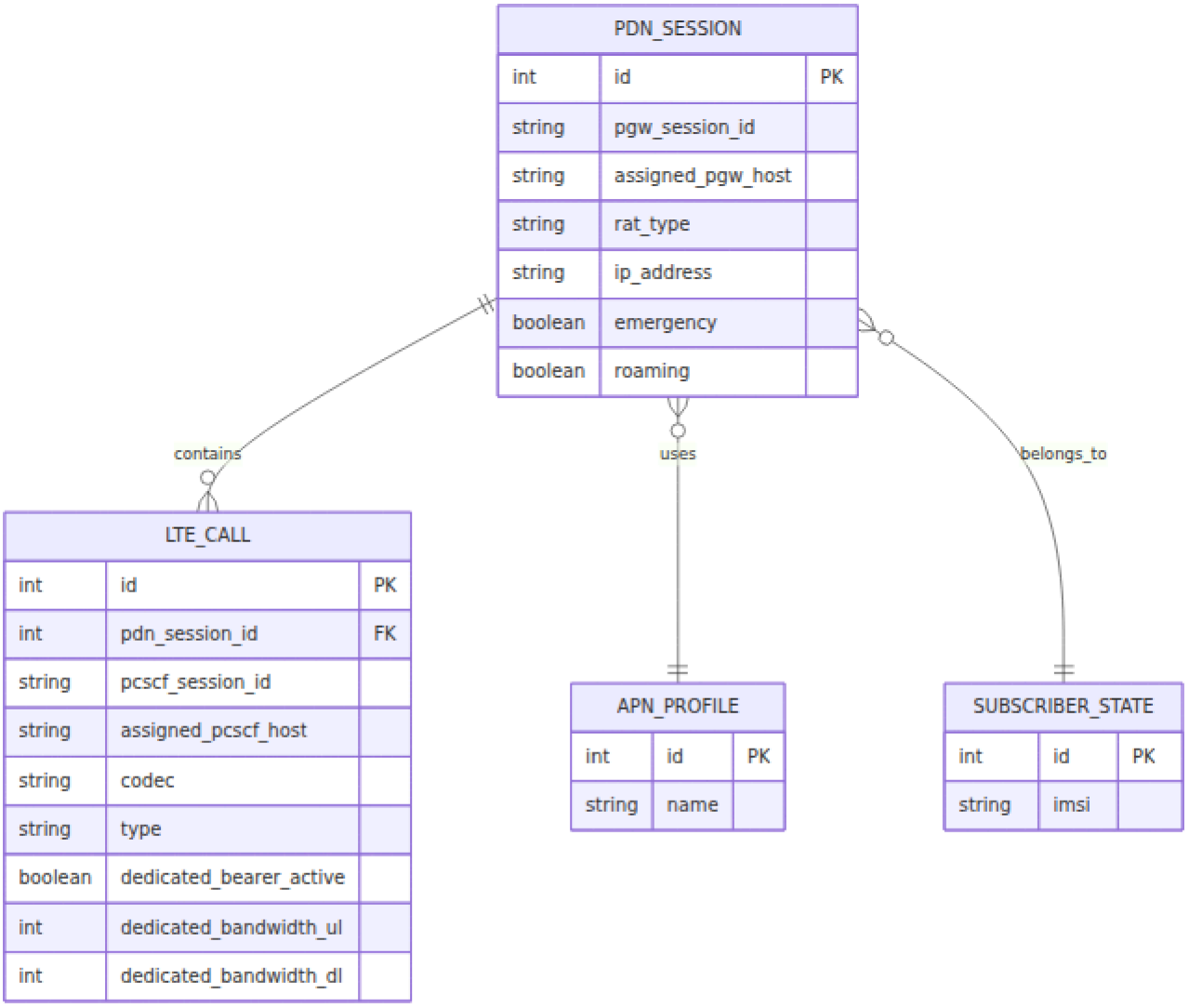
架构

Diameter 接口

接口	应用 **ID**	对端	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、带宽预留

会话状态管理

PCRF 维护活动 PDN 连��和 VoLTE 通话的会话状态：



Gx 接口

支持的□作

1. 计费控制请求 **-** 初始（**CCR-I**）

触发：PGW 为订阅者创建新的 PDN 连接

请求 **AVP**：

- Session-Id
- Origin-Host, Origin-Realm
- Subscription-Id（包含 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（UE IP 地址）

PCRF 操作：

1. 通过 IMSI 查找订阅者
2. 检索 APN 配置和 QoS 配置
3. 创建会话跟踪条目
4. 从 APN 配置构建 QoS 策略

响应 **AVP**：

- 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

2. 计费控制请求 **-** 更新（**CCR-U**）

触发：PGW 报告会话变化（位置更新、RAT 变化等）

PCRF 操作：

1. 通过会话 ID 定位现有会话
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. 计费控制请求 **-** 终止（**CCR-T**）

触发：PGW 终止 PDN 连接

PCRF 操作：

1. 通过会话 ID 定位会话
2. 删除会话和相关通话记录
3. 确认终止

响应: Result-Code 2001

4. 重新认证请求（**RAR**）

方向: PCRF → PGW（HSS 发起）

触发:

- IMS 通话建立（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 AVP:

- 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（针对专用承载）

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、位置、配额）触发
- 使用数据包过滤规则（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）的无限制访问而不消耗数据配额

- 配额后访问 - 即使在订阅者耗尽数据配额后，仍允许自助服务门户和支持网站访问
- 分级速度 - 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 - 国际与国内数据使用的不同计费
- 企业 **SLA** - 对业务关键应用的保证 QoS

QoS 策略结构

默认承载 **QoS**（来自 APN 配置）：

```
{
  "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□作

1. AA 请求（AAR）/ AA 响应（AAA）

触发: P-CSCF 请求 IMS 媒体会话的授权（VoLTE 通话建立）

请求 **AVP**:

- Session-Id（P-CSCF 会话标识符）
- Subscription-Id（IMSI 或 SIP URI）
- Media-Component-Description
  - Media-Type（音频、视频）
  - Max-Requested-Bandwidth-UL/DL
  - Codec-Data
  - Flow-Description（5 元组数据包过滤器）
- AF-Application-Identifier

PCRF 操作:

1. 通过 IMSI 或 SIP URI 查找订阅者
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 **AVP**:

- Result-Code: 2001（成功）或 5063（服务未授权）

2. 会话终止请求（**STR**）**/** 会话终止响应**STA**）

触发: P-CSCF 终止 IMS 会话（通话挂断）

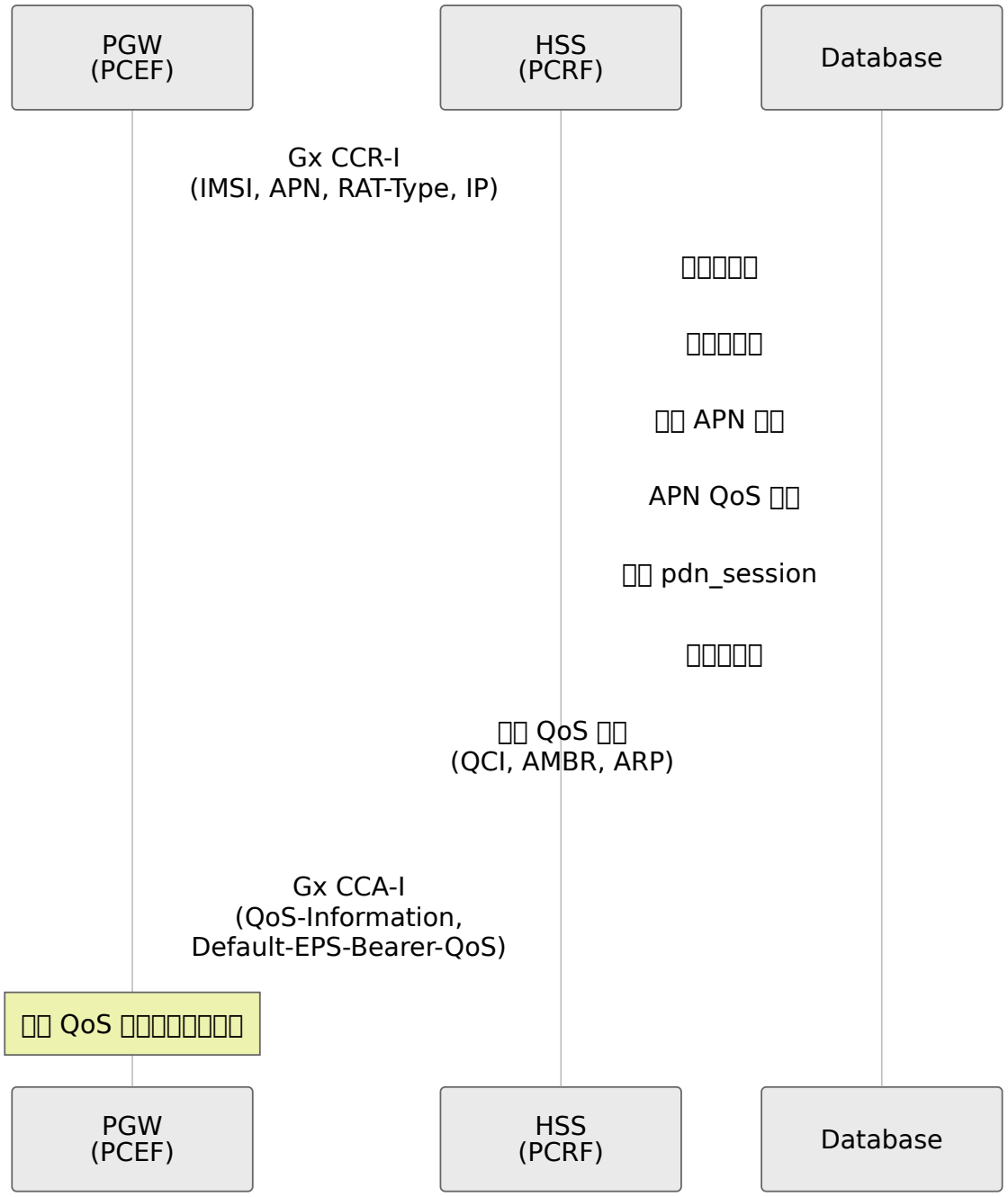
PCRF 操作:

1. 通过 P-CSCF 会话 ID 定位通话会话
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

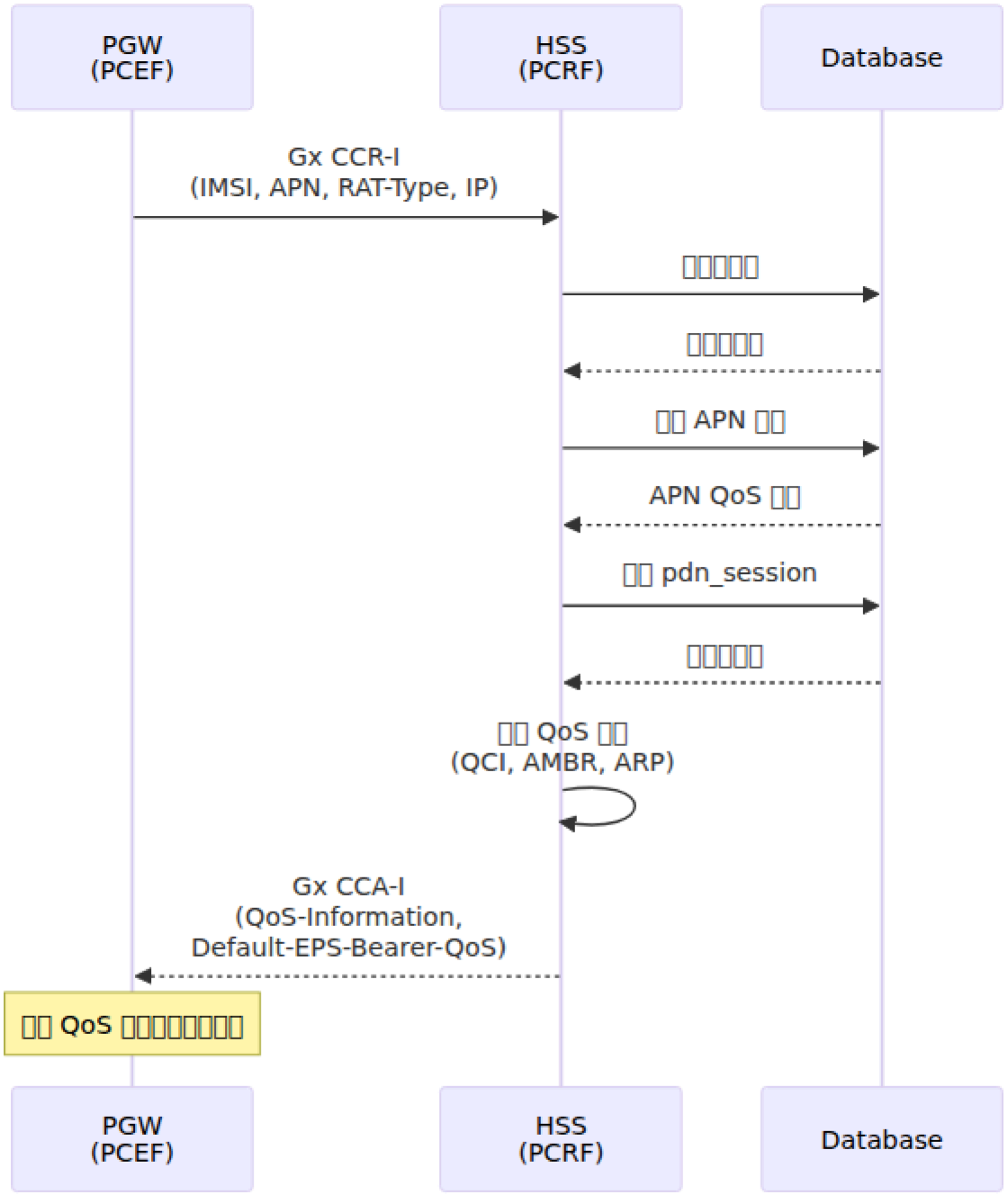
□□□Result-Code 2001

□□□□□

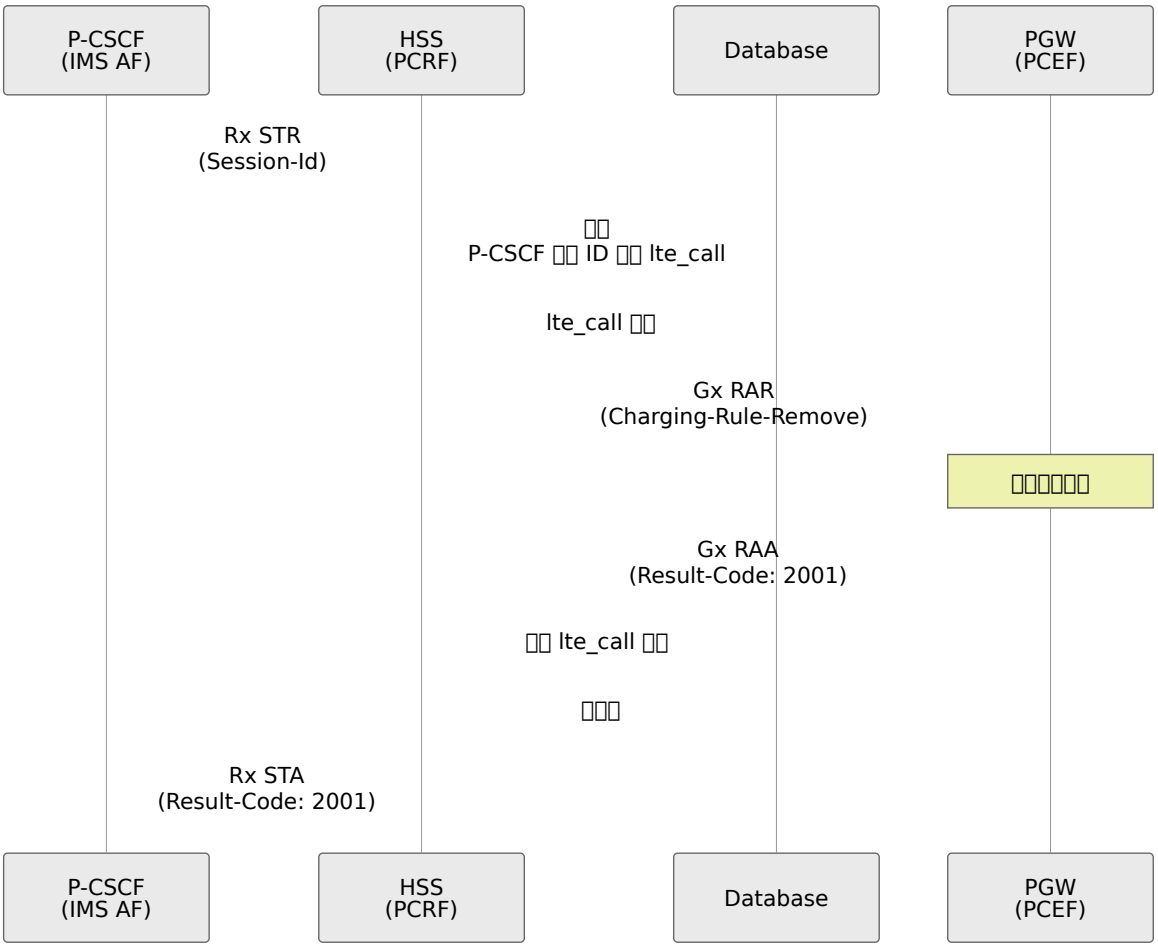
□□ 1□PDN □□□□



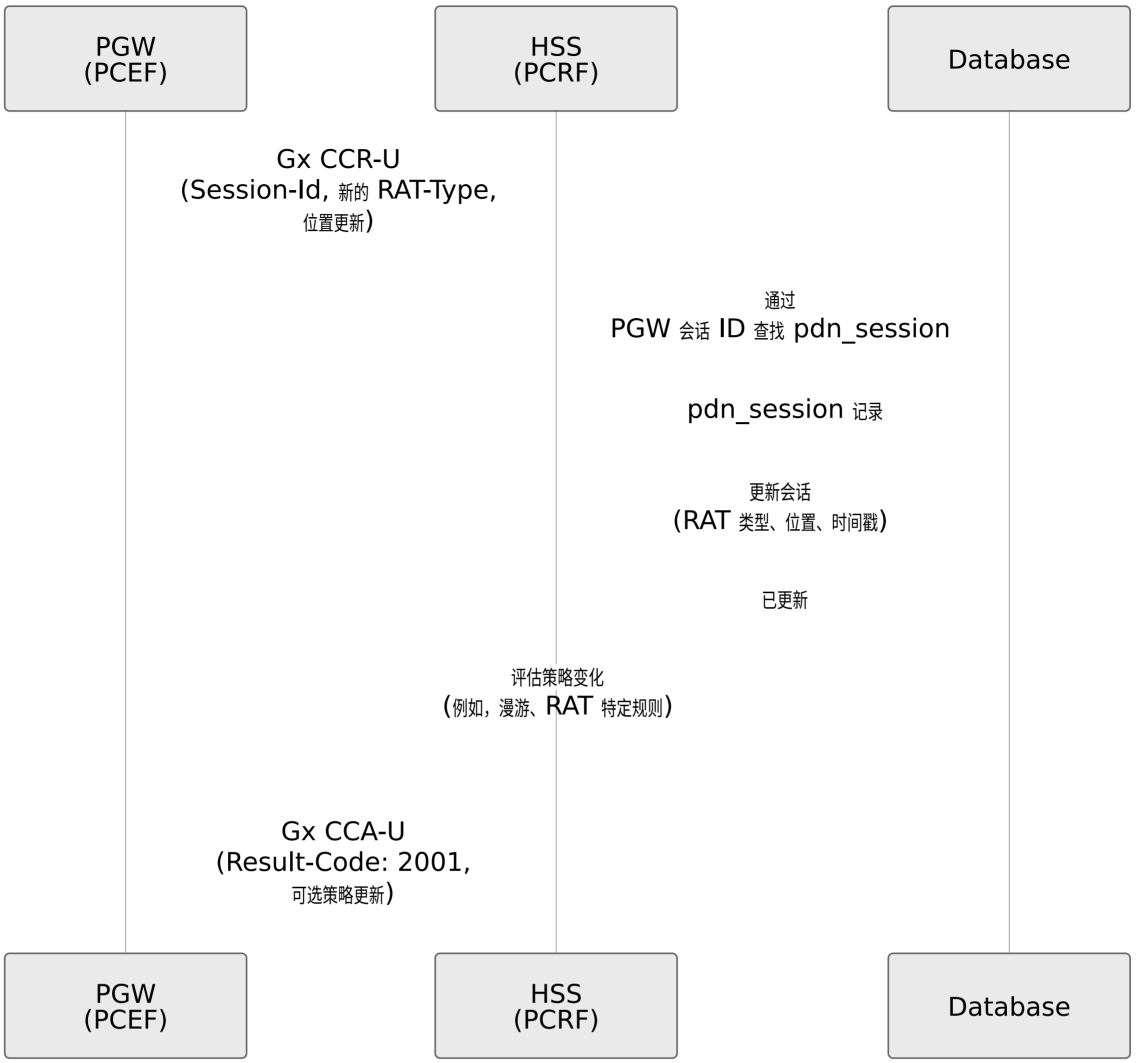
流程 2：VoLTE 通话建立（Rx AAR → Gx RAR）



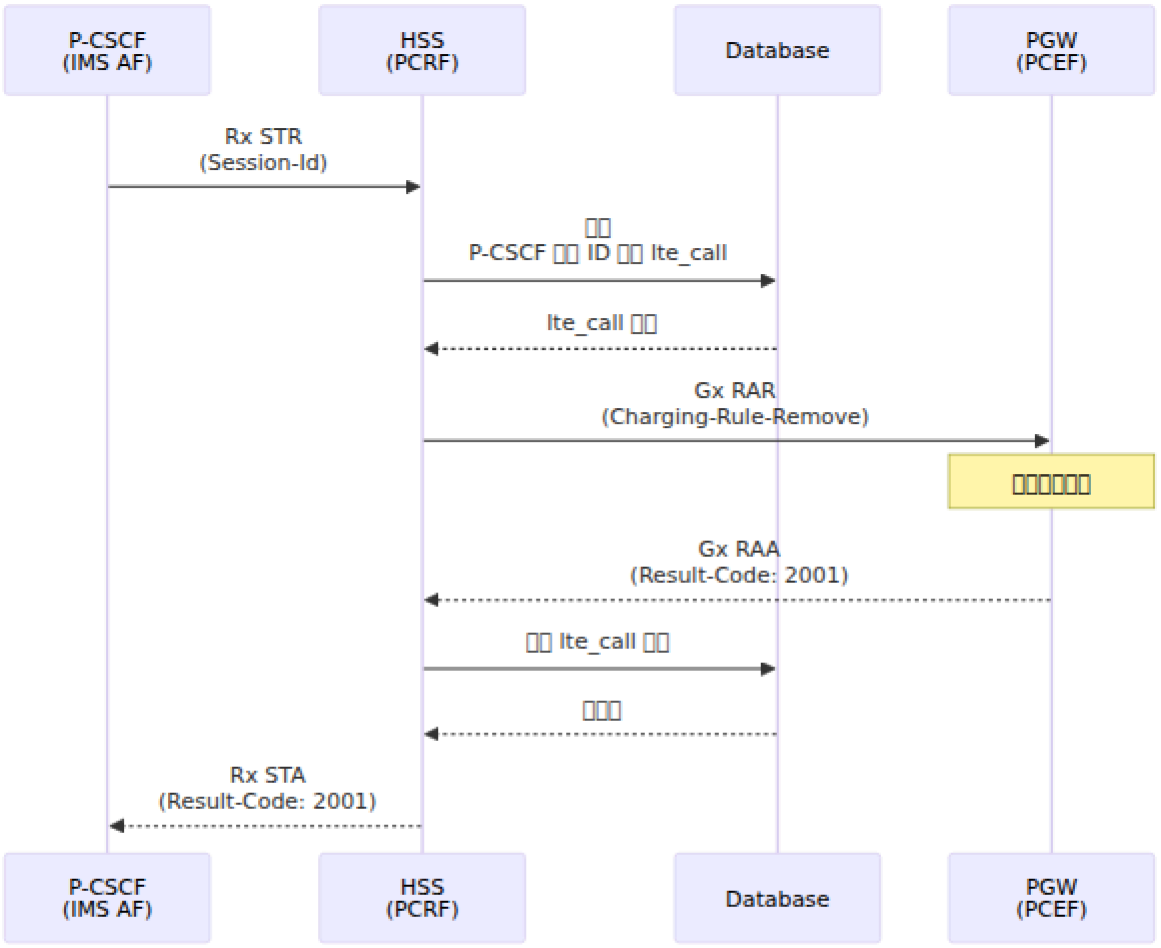
□□ 3□VoLTE □□□□□Rx STR → Gx RAR□



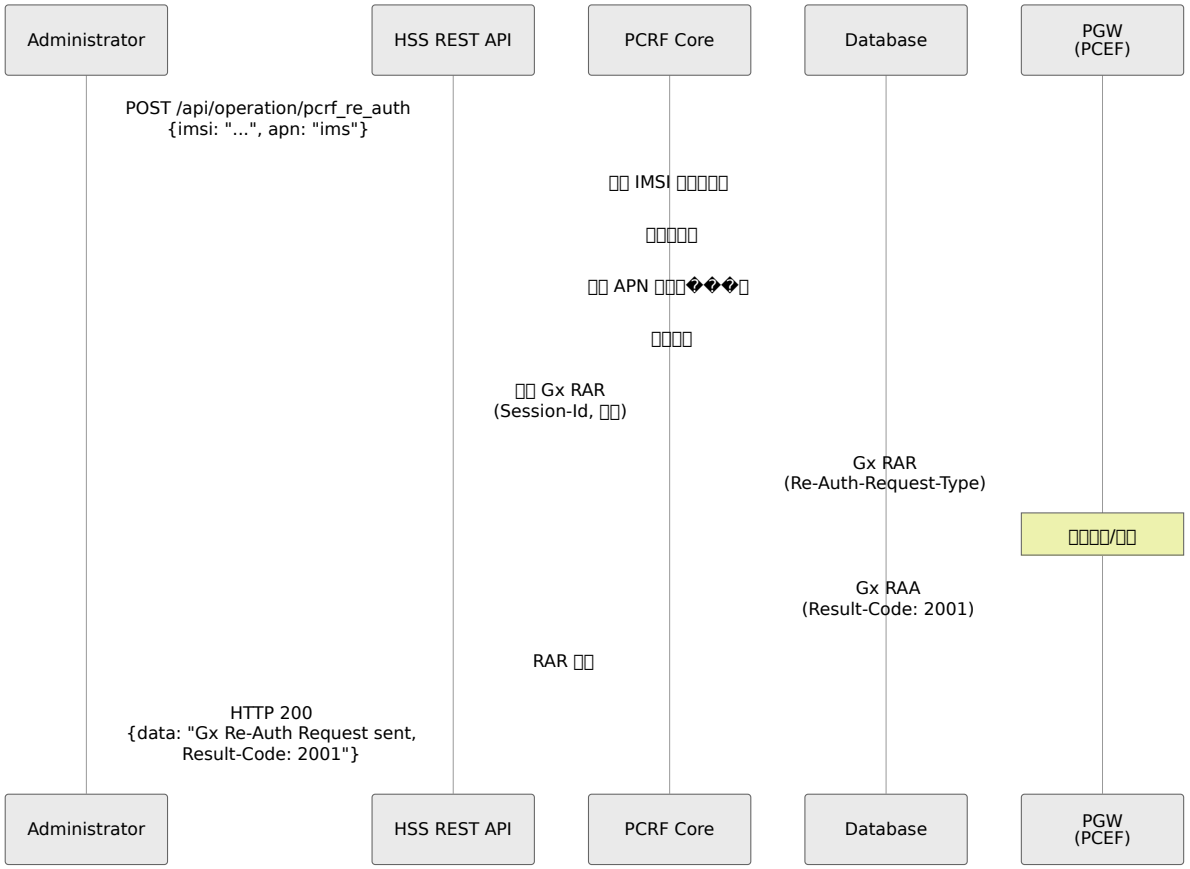
流程 4: PDN 会话更新


PGW
(PCEF)
HSS
(PCRF)
Database
Gx CCR-U
(Session-Id, 新的 RAT-Type,
位置更新)
通过
PGW 会话 ID 查找 pdn_session
pdn_session 记录
更新会话
(RAT 类型、位置、时间戳)
已更新
评估策略变化
(例如，漫游、RAT 特定规则)
Gx CCA-U
(Result-Code: 2001,
可选策略更新)
PGW
(PCEF)
HSS
(PCRF)
Database

流程 5: PDN 会话终止


P-CSCF
(IMS AF)
HSS
(PCRF)
Database
PGW
(PCEF)
Rx STR
(Session-Id)
□□
P-CSCF □□ ID □□ lte_call
lte_call □□
Gx RAR
(Charging-Rule-Remove)
□□□□□□□
Gx RAA
(Result-Code: 2001)
□□ lte_call □□
□□□
Rx STA
(Result-Code: 2001)
P-CSCF
(IMS AF)
HSS
(PCRF)
Database
PGW
(PCEF)

□□ 6□□□ REST API □□□□□□



REST API

PCRF □□□□□□

□□□`POST /api/operation/pcrf_re_auth`

□□□□□□□□ Gx □□□□□□□□□□□□

□□□□□□□□□□□

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

成功响应（HTTP 200）：

```
{
  "data": "Gx Re-Auth Request for 999999876543210 sent to
pgw.epc.mnc999.mcc999.3gppnetwork.org, Result-Code: 2001"
}
```

错误响应（HTTP 400）：

```
{
  "error": "Unable to send Re-Auth Request for 999999876543210 on
APN ims, no active PDN Session found"
}
```

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���略执行：当您更新 APN QoS 配置（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订阅者，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 → APN 配置
     ↓                    ↓                  ↓
  "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": 1,
    "apn": "internet",
    "ip_version": "ipv4v6"
  }
}
```

验证：

- `apn`：必填，1-254 个字符，唯一
- `ip_version`：必填，必须□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

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": "Best Effort Internet",
    "qci": 9,
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 100000,
    "apn_ambr_ul_kbps": 50000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true
  }
}
```

QoS 参数：

字段	类型	范围	描述
`name`	string	1-254 字符	配置名称（唯一）
`qci`	integer	1-254	QoS 类标识符（1-4 = GBR，5-9 = 非 GBR）
`allocation_retention_priority`	integer	1-15	ARP 级别（1 = 最高优先级）
`apn_ambr_dl_kbps`	integer	1-4,294,967,293	APN 聚合最大比特率下行（kbps）
`apn_ambr_ul_kbps`	integer	1-4,294,967,293	APN 聚合最大比特率上行（kbps）
`pre_emption_capability`	boolean	true/false	可以抢占低优先级承载
`pre_emption_vulnerability`	boolean	true/false	可能被高优先级承载抢占

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": 1,
    "name": "Best Effort Internet",
    "qci": 9,
    "allocation_retention_priority": 8,
    "apn_ambr_dl_kbps": 100000,
    "apn_ambr_ul_kbps": 50000,
    "pre_emption_capability": false,
    "pre_emption_vulnerability": true
  }
}
```

列出 **QoS ���置**: `GET /api/apn/qos_profile`

3. 创建 APN 配置

将 APN 标识符与 QoS 配置链接。

端点: `POST /api/apn/profile`

请求体:

```
{
  "apn_profile": {
    "name": "Internet APN",
    "apn_identifier_id": 1,
    "apn_qos_profile_id": 1
  }
}
```

字段:

- `name`: 配置名称（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 配置的 ID

响应（HTTP 201）:

```
{
  "data": {
    "id": 1,
    "name": "Internet APN",
    "apn_identifier_id": 1,
    "apn_qos_profile_id": 1
  }
}
```

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 配置的组合必须唯一的

列出 **APN** 配置: `GET /api/apn/profile`

完整的策略配置示例

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": 2, ...}}

# 2. 创建 QoS 配置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 Signaling 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": 2, ...}}

# 3. 创建 APN 配置
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": 2,
      "apn_qos_profile_id": 2
    }
  }'
# 响应: {"data": {"id": 2, ...}}
```

步骤 **2**: 分配给订阅者

创建后，APN 配置通过 EPC 配置分配给订阅者。请参见 API 参考 以链接 APN 配置到订阅者。

策略更新和删除

更新 **QoS** 配置:

```
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 ID 1 以增加带宽
curl -X PATCH https://hss.example.com:8443/api/apn/qos_profile/1 \
  -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. QoS 配置在数据库中更新
2. 系统识别所有使用与此 QoS 配置关联的 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. 所有连接的订阅者在 API 调用完成后几秒钟内立即收到更新的策略

示例场景: 如果 100 个订阅者当前连接在使用 QoS 配置 ID 1 的 "internet" APN 上，所有 100 个订阅者的带宽限制将在 API 调用完成后的几秒钟内更新为 150 Mbps 下行 / 75 Mbps 上行。

注意: 当您更新 APN QoS 配置时，系统会自动触发重新认证以应用新的策略，立即应用于附加的订阅者。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：

- 无法删除被 APN 配置引用的 APN 标识符或 QoS 配置
- 无法删除分配给活动订阅者的 APN 配置

策略模板

高速互联网（下行 **100 Mbps / 上行 50 Mbps**）：

```
{
  "apn_qos_profile": {
    "name": "High Speed Internet",
    "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": "Premium Internet",
    "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": "IoT 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": "Emergency 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
  - `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 AVP
  - 流过滤器：来自 Flow-Description AVP

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 不在订阅者数据库中
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、每个订阅者一个
- **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，包括:

- 通过 Gx 接口的计费规则安装/删除
- 流量流模板（**TFT**）匹配以实现服务差异化
- 基于应用或流量类型的动态速度配置
- 由网络条件或订阅者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

- 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

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:

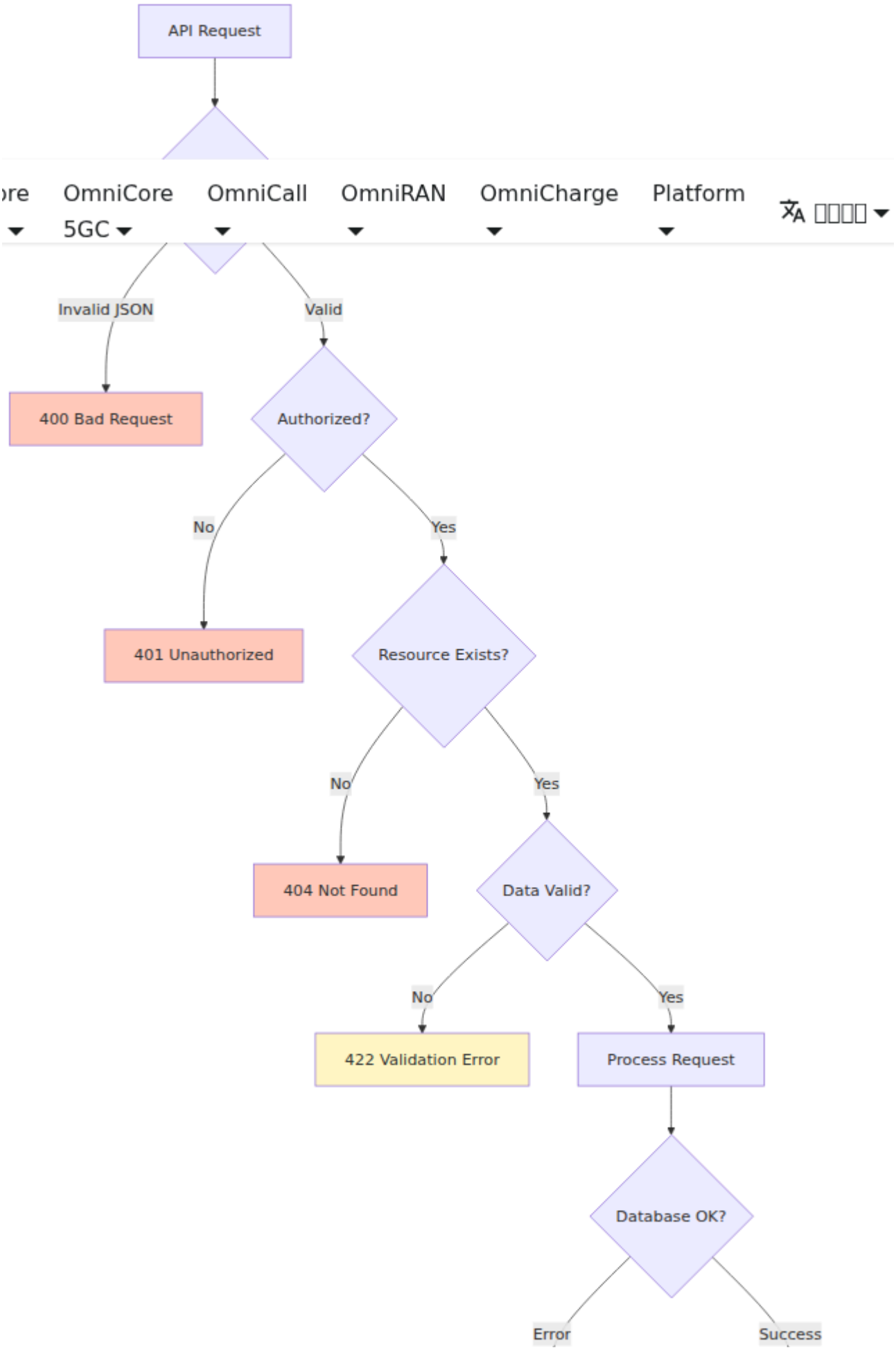
- 验证失败
- 数据库约束被违反
- 外键引用不存在

500 内部服务器错误

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

原因:

- 数据库连接问题
- 意外的应用程序错误


API Request
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

此示例演示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"
```

创建内容：

此配置工作流程创建一个完整的用户，包括：

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）- 带有 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

完整的静态 IP 配置

此示例演示从头开始配置带有静态 IP 地址的用户。

场景：配置一个需要在“互联网” APN 上使用静态 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. 创建带有静态 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）"
```

创建内容：

此配置工作流程创建一个完整的 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）- 针对 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：

当该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：

- 添加其他 APN 和静态 IP：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 - 没有静态 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

基础 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 "扁平" JSON 有效负载，没有包装对象。

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": { ... }
}
```

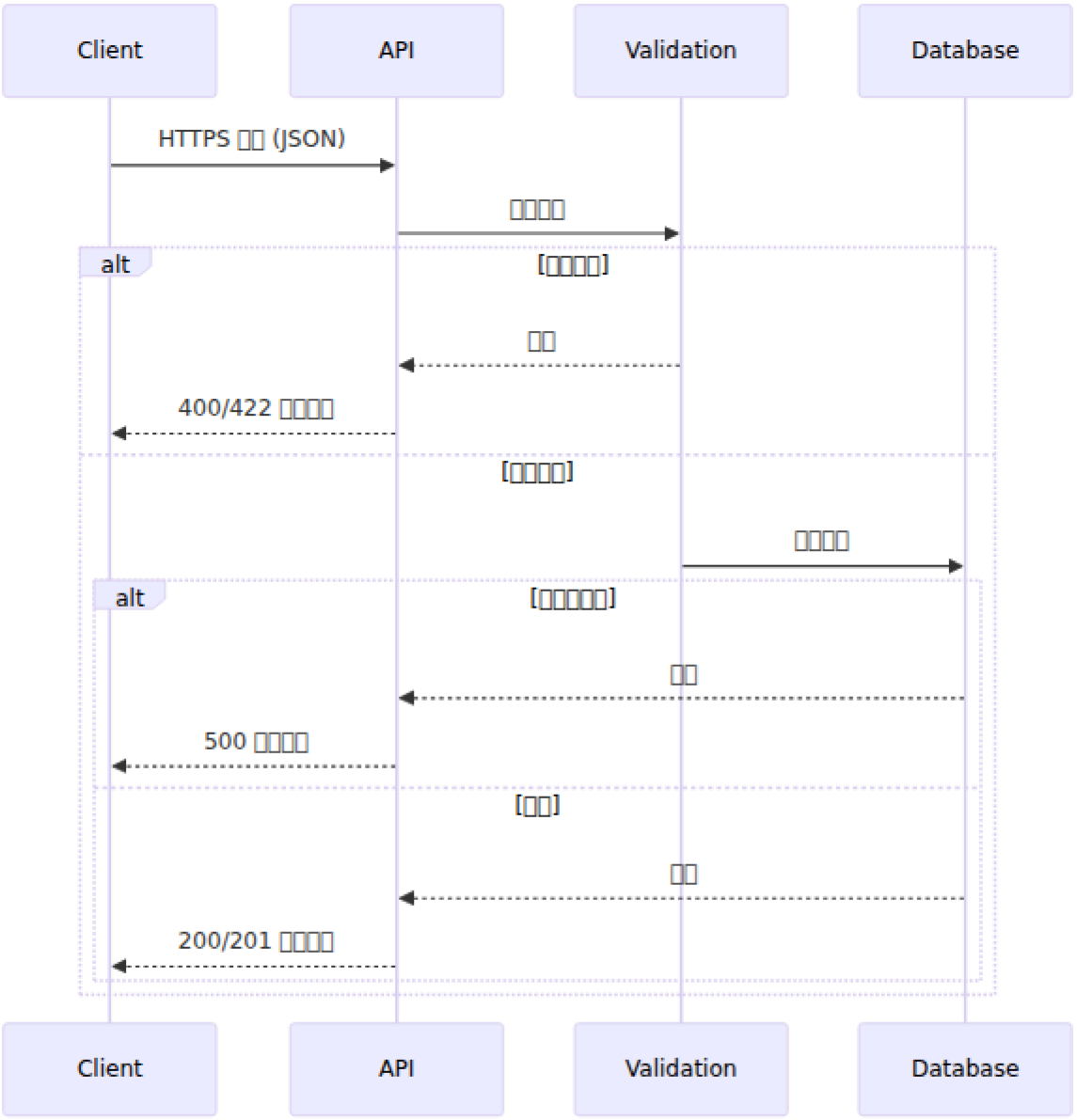
错误响应：

```
{
  "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	无效的输入数据
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": 1,
      "imsi": "001001123456789",
      "enabled": true,
      "ims_enabled": true,
      "sim_id": 1,
      "key_set_id": 1,
      "epc_profile_id": 1,
      "ims_profile_id": 1,
      "roaming_profile_id": 1,
      "custom_attributes": {},
      "inserted_at": "2025-10-15T10:30:00Z",
      "updated_at": "2025-10-15T10:30:00Z"
    }
  ]
}
```

根据 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`	integer	用户数据库 ID

示例请求：

```
curl -k https://hss.example.com:8443/api/subscriber/1
```

根据 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：根据其 IMSI 进行特定用户的故障排除。

根据 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	移动站 ISDN 号码	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": 1,
    "key_set_id": 1,
    "epc_profile_id": 1,
    "ims_profile_id": 1,
    "roaming_profile_id": 1,
    "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: true
- `ims_enabled` - 默认: 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
- `custom_attributes` - 自定义键值对

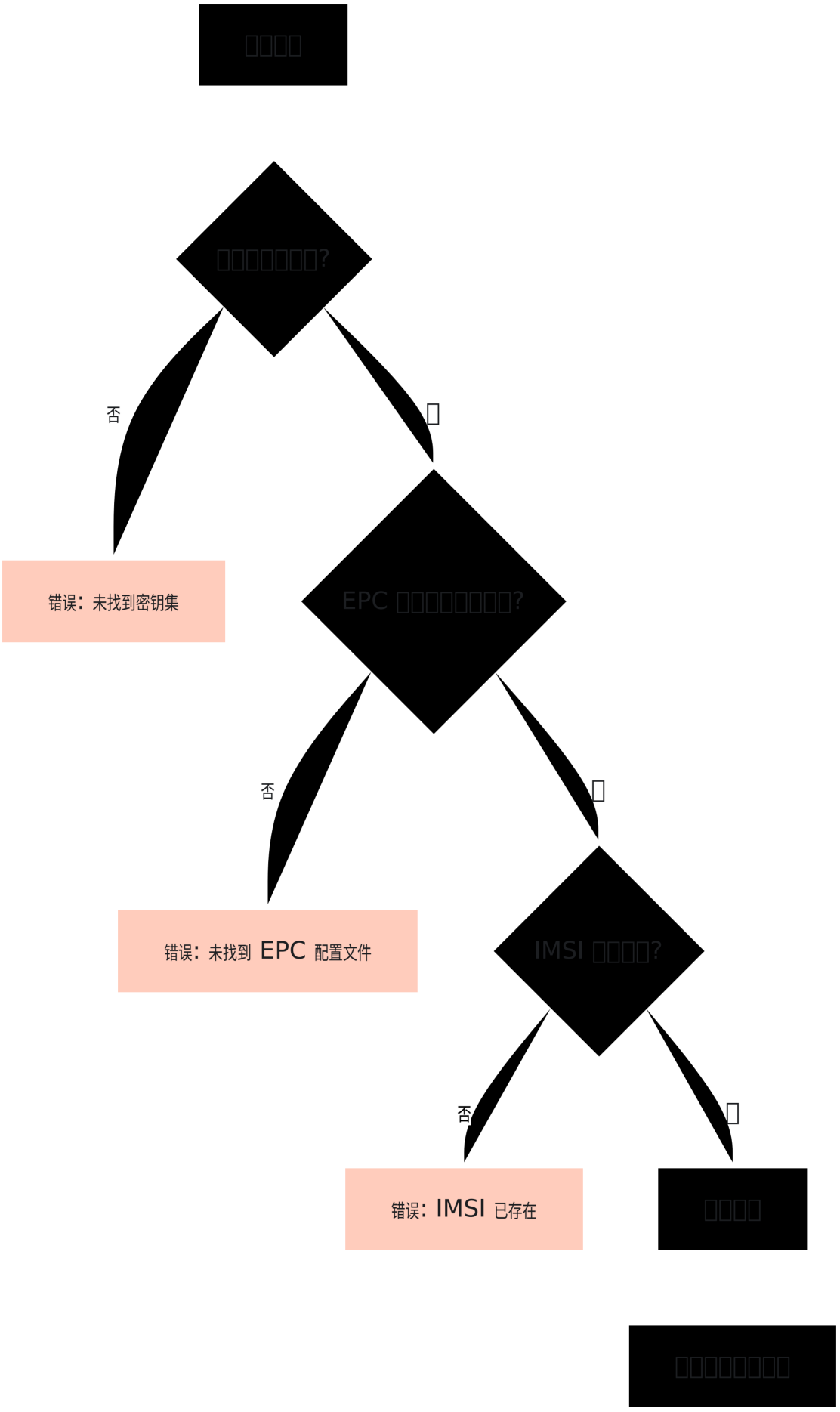
另见：

- 完整用户配置示例 - 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": 1,
      "epc_profile_id": 1
    }
  }'
```

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`	integer	用户数据库 ID

请求体：

```
{
  "subscriber": {
    "enabled": false,
    "ims_enabled": false,
    "epc_profile_id": 2,
    "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` - 更新自定义数据

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/1 \
  -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": 2}`（见 EPC 配置文件）
- 更新漫游策略：`{"roaming_profile_id": 3}`（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`	integer	用户数据库 ID

示例请求：

```
curl -k -X DELETE https://hss.example.com:8443/api/subscriber/1
```

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：

- 密钥集 - 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）以强制将用户从其当前注册的 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	□	要分离的用户 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`)
- 使用 `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
- 终止订阅**:** 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：立即断开被破坏的用户

多 **IMSI** 考虑：

在使用 CLR 进行多 MSISDN 场景时：

1. 多个 **MSISDN**，单个 **IMSI**：

```
// 用户具有 IMSI 001001123456789 和 MSISDNs ["+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（号码迁移场景）
// 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
```

重要说明：

- 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，CLR 也会发送（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

MSISDN □□

MSISDN□□ □ MSISDN □□□

□□ MSISDNs

□□□□□□□□□

□□: `GET /api/msisdn`

□□□□:

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

□□ MSISDN

□□□□□□□□□

□□: `GET /api/msisdn/:id`

□□□□:

```
curl -k https://hss.example.com:8443/api/msisdn/1
```

□□ MSISDN

□□□□□□□□□□

□□: `POST /api/msisdn`

□□□:

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

验证：

- 必须为 1-15 位数字
- 必须唯一
- 必须遵循 E.164 格式（国际格式，无 + 符号）

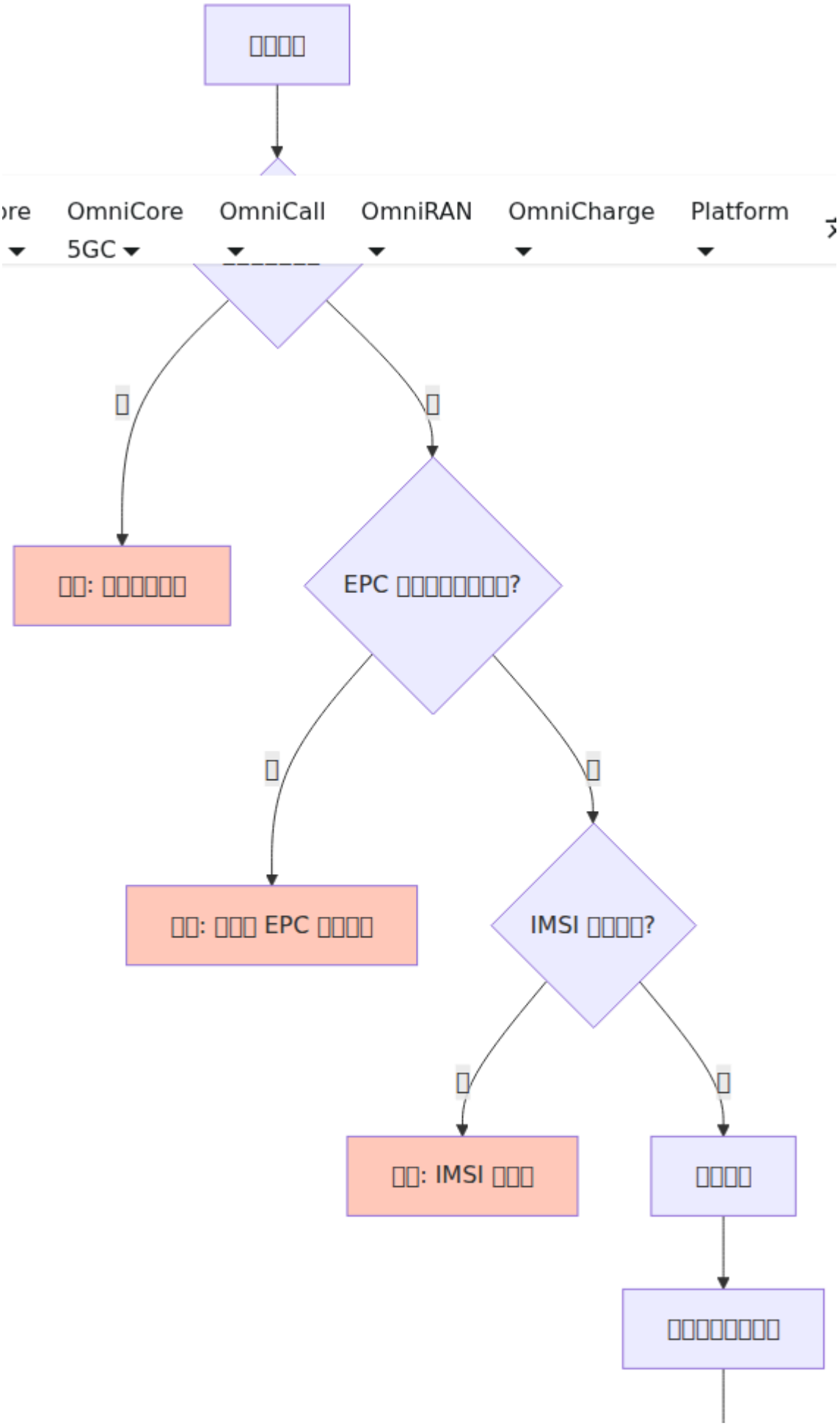
示例请求：

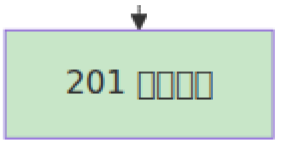
```
curl -k -X POST https://hss.example.com:8443/api/msisdn \
  -H "Content-Type: application/json" \
  -d '{
    "msisdn": {
      "msisdn": "14155551234"
    }
  }'
```

将 **MSISDN** 分配给用户

要将电话号码分配给用户，您需要创建一个关联记录。这通常通过用户更新端点或通过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/1
```

SIM 管理

SIM 卡记录存储物理 SIM 卡信息，包括 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 - 在一个物理 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/1
```

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 代码
- `puk1`, `puk2` - PIN 解锁代码
- `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/1 \
  -H "Content-Type: application/json" \
  -d '{
    "sim": {
      "batch_name": "更新的批次名称"
    }
  }'
```

删除 SIM

移除一个 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

密钥集包含用于通过 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

检索特定密钥集。

端点：`GET /api/key_set/:id`

示例请求：

```
curl -k https://hss.example.com:8443/api/key_set/1
```

响应示例：

```
{
  "data": {
    "id": 1,
    "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 派生）
- `authentication_algorithm` - 目前仅支持 "milenage"

可选字段：

- `amf` - 默认："8000"
- `sqn` - 默认：0
- `ota_counter` - 默认：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：密钥集包含高度敏感的加密材料。请相应地保护 API 访问。

更新密钥集

修改现有密钥集。

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。仅在维护窗口期间或对新用户进行更新。

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身份验证失败。

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	"仅数据包" 或 "数据包和电路"
`tracking_area_update_interval_seconds`	TAU 计时器	秒	600 (典型)
`apn_profiles`	APN 配置文件 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用户的初始过滤标准（IFC）。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-模板-在这里>"
}
```

必填字段：

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

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 模板存储为 **Liquid** 模板（类似于 Jinja2），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模板，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** 创建成功**):**

```
{
  "status": "success",
  "response": {
    "id": 1,
    "name": "default",
    "ifc_template": "<?xml version=\"1.0\" encoding=\"UTF-8\"?
>..."
  }
}
```

验证**:**

- API 验证 IFC 模板□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：

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 - 组合标识符和 QoS，链接到 EPC 配置文件

有关详细的策略配置、**QoS** 管理和自动重新认证，请参见 **PCRF** 文档。另请参见 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": 1,
  "apn_qos_profile_id": 1,
  "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

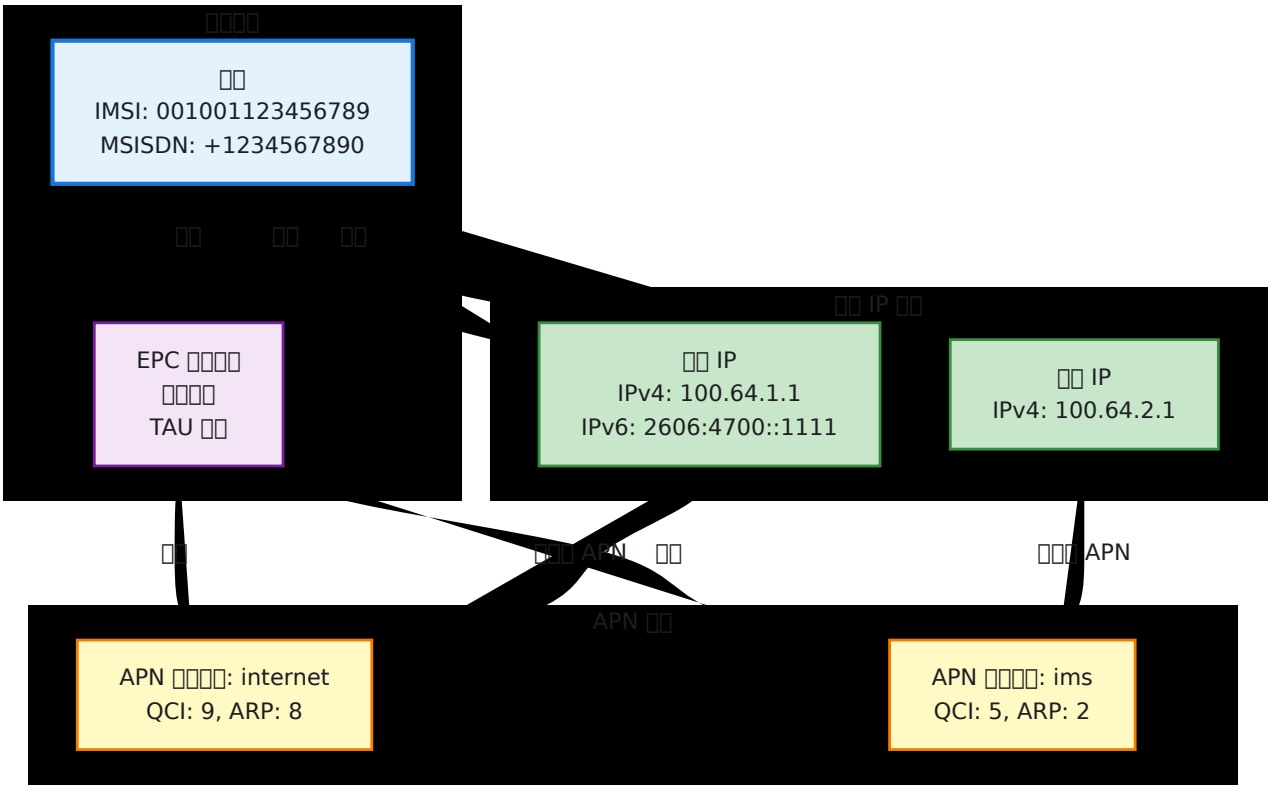
□□:

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

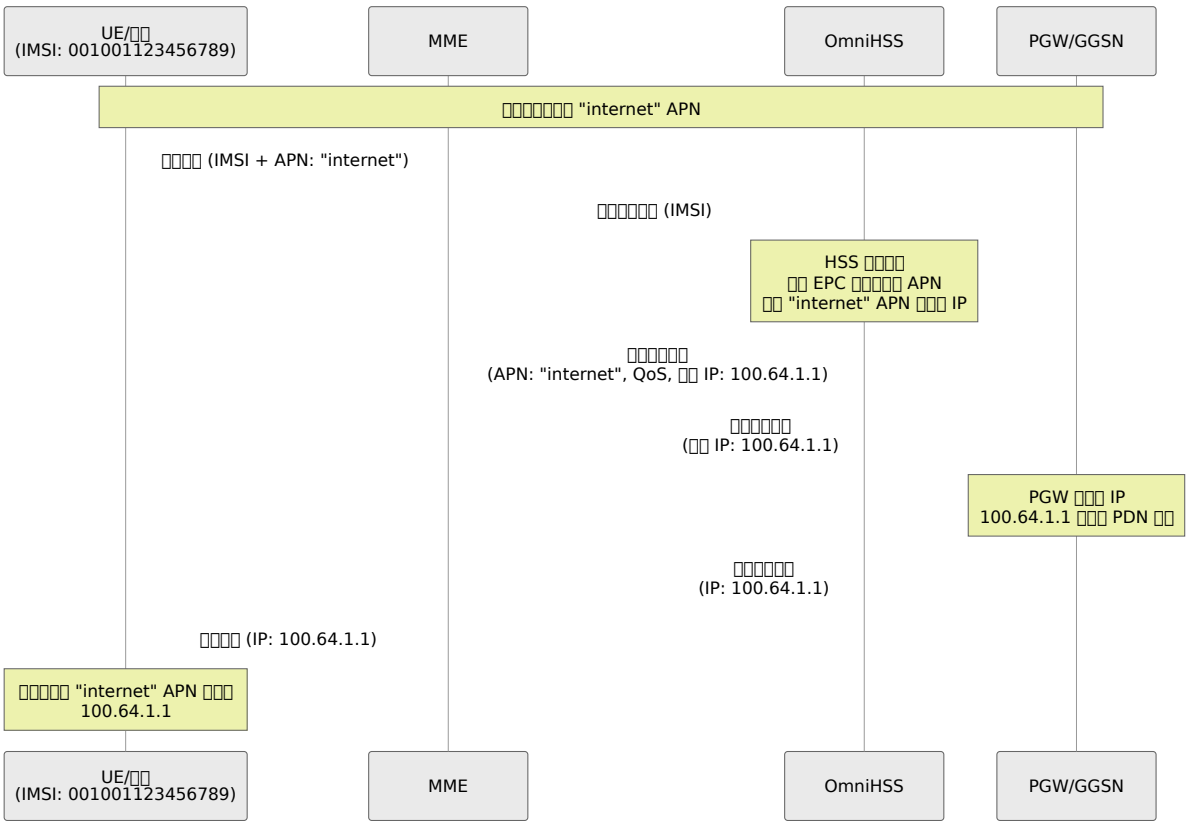
□□ IP □□

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

□□:

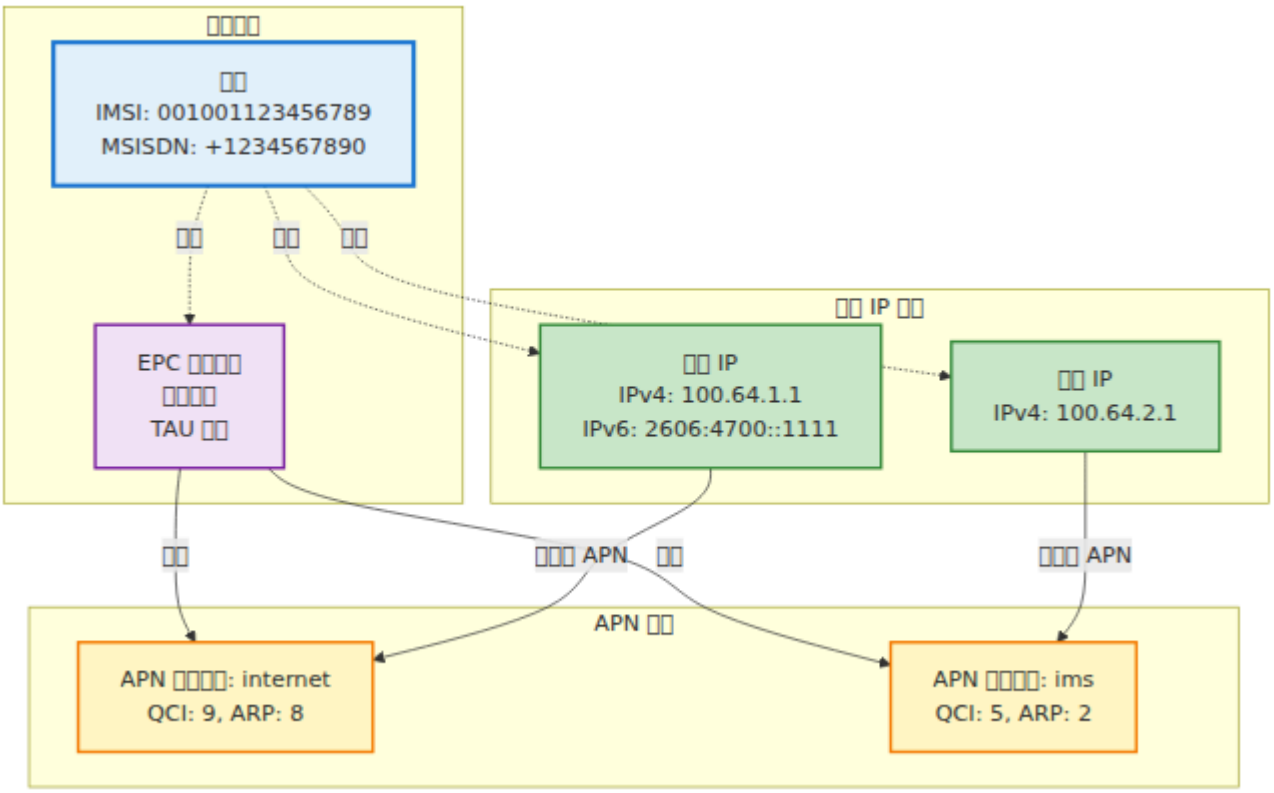


□□□□□□□□□□□:



□□□□□□□ - **APN** □□□□□□□:

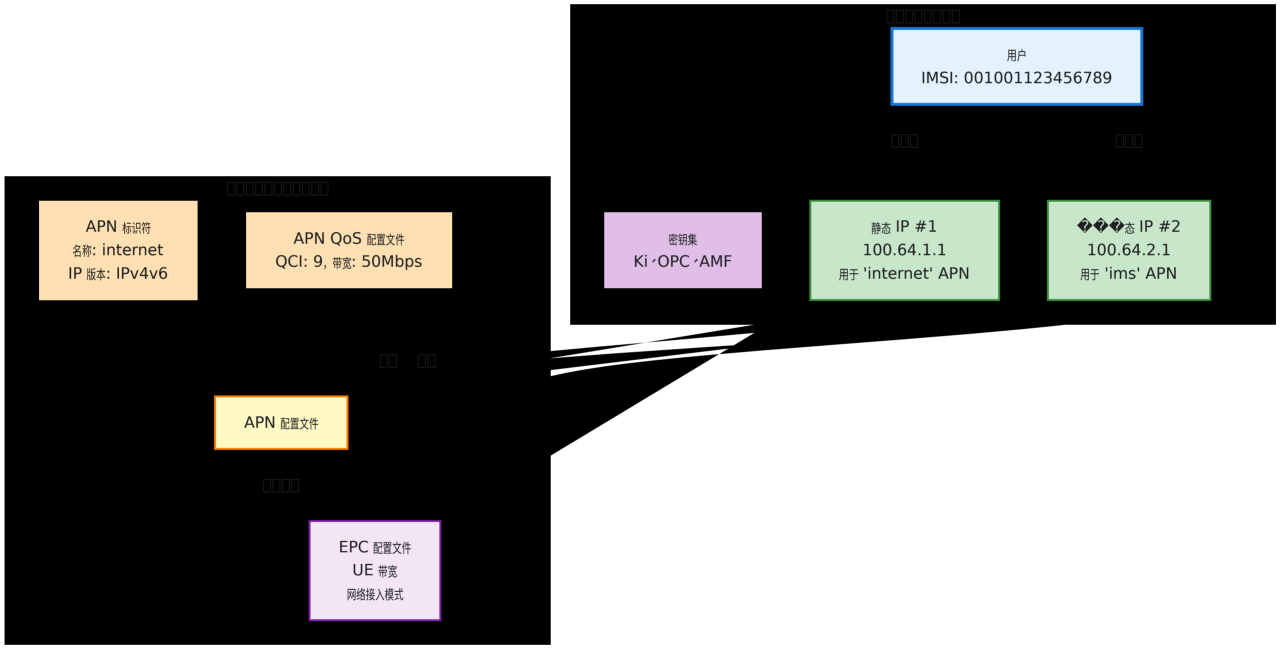
□□□□□ S6a □□□□□□□ APN-□□ AVP □□□□□□□□□□□□



□□□□:

1. 上下文标识符：每个 APN 在配置文件中的顺序索引（0、1、2...）
2. 服务选择：直接来自 `apn_identifier.apn`（例如，"internet"、"ims"）
3. **PDN** 类型：从 `apn_identifier.ip_version` 编码（ipv4=0, ipv6=1, ipv4v6=2, ipv4_or_ipv6=3）
4. **QoS** 参数：所有来自 `apn_qos_profile` 表
5. **AMBR** 带宽：值乘以 1000（kbps → bps 转换）
6. 服务方 **IP** 地址：仅在此用户+APN 组合存在静态 IP 时包含
   - 查找过程：`subscriber.static_ips` → 按 `apn_profile_id` 过滤 → 提取 IP
   - IP 版本兼容性检查与 `apn_identifier.ip_version`
7. **VPLMN-动态地址-允许**：硬编码为 0（不允许） - 强制使用静态 IP（如果提供）

关系层次结构：



关键概念：

- 每 **APN** 分配：每个静态 IP 链接到特定的 APN 配置文件
- 每用户每 **APN** 仅一个 **IP**：用户每个 APN 只能有一个静态 IP 分配
- **IPv4** 和 **IPv6** 支持：静态 IP 可以□仅 IPv4、仅 IPv6 或双栈
- 全局 **IP** 唯一性：每个 IP 地址必须在系统中的所有静态 IP 记录中全局唯一
  - 相同的 IPv4 或 IPv6 地址不能分配给多个用户（即使在不同的 APN 上）
  - 这防止了路由冲突和 IP 地址模糊性
  - 通过数据库唯一索引在 `ipv4_static_ip` 和 `ipv6_static_ip` 字段上强制执行
- 多对多关系：用户和静态 IP 通过连接表链接

用例：

- IoT 设备的固定 IP 地址
- 在移动设备上托管服务器（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 IPs

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": 1,
      "apn_profile_id": 5,
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111",
      "apn_profile": {
        "id": 5,
        "name": "互联网 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

检索特定静态 IP 分配。

端点：`GET /api/epc/static_ip/:id`

路径参数：

参数	类型	描述
`id`	integer	静态 IP 数据库 ID

示例请求：

```
curl -k https://hss.example.com:8443/api/epc/static_ip/1
```

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": 5,
    "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
- 至少必须指定 `ipv4_static_ip` 或 `ipv6_static_ip` 之一

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 记录中全局唯一
  - 这防止网络中的 IP 地址冲突
  - 相同的 IP 不能分配给多个用户，即使在不同的 APN 上
  - 这通过唯一索引在数据库级别强制执行的

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 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": 5,
      "ipv4_static_ip": "100.64.1.1"
    }
  }'
```

仅 IPv6 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": 6,
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }'
```

双栈静态 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": 5,
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }'
```

成功响应 **(201** 创建成功**):**

```
{
  "data": {
    "id": 1,
    "apn_profile_id": 5,
    "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"
  }
}
```

另见**:**

- 将静态 IP 分配给用户 - 如何将其链接到用户
- 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**:**

���数	类型	描述
`id`	integer	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：更改活动用户的 IP 地址将影响他们的下一个 PDN 连接。活动 PDN 会话将继续使用旧 IP，直到他们断开并重新连接。

示例请求：

```
curl -k -X PUT https://hss.example.com:8443/api/epc/static_ip/1 \
  -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`	integer	静态 IP 数据库 ID

示例请求：

```
curl -k -X DELETE https://hss.example.com:8443/api/epc/static_ip/1
```

行为：

- 移除静态 IP 分配
- 不影响 APN 配置文件（APN 仍可用于其他用户）
- 使用此静态 IP 的用户将在下次连接时��收动态 IP
- 删除后，IP 地址变为可重用

警告：如果用户正在使用此静态 IP，删除它将导致他们在下次 PDN 连接时接收动态 IP。确保用户离线或在删除之前发送 取消位置请求。

将静态 IP 分配给用户

要将静态 IP 分配给用户，您需要在创建或更新时将静态 IP 记录与 用户 关联。

分配模式：

1. 创建静态 IP（请参见 创建静态 IP）
2. 使用 `static_ips` 字段分配给用户

创建带有静态 IP 的用户：

```
# 第 1 步：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": 5,
      "ipv4_static_ip": "100.64.1.1",
      "ipv6_static_ip": "2606:4700:4700::1111"
    }
  }' | jq -r '.data.id')

# 第 2 步：创建分配了静态 IP 的用户
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriber\": {
      \"imsi\": \"001001123456789\",
      \"key_set_id\": 1,
      \"epc_profile_id\": 1,
      \"static_ips\": [$STATIC_IP_ID]
    }
  }"
```

更新现有用户以添加静态 **IP:**

```
curl -k -X PUT https://hss.example.com:8443/api/subscriber/1 \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "static_ips": [1, 2]
    }
  }'
```

多个静态 **IP**（不同 **APNs**）**:**

用户可以拥有多个静态 IP，只要每个 IP 针对不同的 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": 5,
      "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": 6,
      "ipv4_static_ip": "100.64.2.1"
    }
  }' | jq -r '.data.id')

# 将两个 IP 分配给用户
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d "{
    \"subscriber\": {
      \"imsi\": \"001001123456789\",
      \"key_set_id\": 1,
      \"epc_profile_id\": 1,
      \"static_ips\": [$INTERNET_IP, $IMS_IP]
    }
  }"
```

验证规则：

- ✓ 允许：针对不同 APNs 的多个静态 IP
- ✗ 拒绝：针对同一 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": [1, 2]
    }
  }'

# 错误响应：
{
  "errors": {
    "static_ips": [
      "每个用户每个 APN 的静态 IP 必须唯一。例如，用户不能同时被分配静态 IP 100.64.1.1 用于
internet 和 100.64.1.2 用于 internet"
    ]
  }
}
```

另见：

- 创建用户 - 用户配置
- 更新用户 - 修改用户配置
- 完整静态 IP 配置示例 - 端到端工作流

漫游管理

漫游配置文件控制用户□否可以在访问网络上访问数据和 IMS 服务。配置文件分配给 用户，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": "仅美国运营商",
    "data_action_if_no_rules_match": "deny",
    "ims_action_if_no_rules_match": "deny",
    "roaming_rules": []
  }
}
```

□作值：

- `"allow"` - 允许
- `"deny"` - 拒绝

默认□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

另见：

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

EIR 管理

OmniHSS 通过 S13 Diameter 接口作为设备身份注册（EIR）功能。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）

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 IMEIs）
- 限制设备类型（按 TAC 模式黑名单）
- 仅允许批准的设备（白名单模式，默认拒绝所有）

另见：

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

附加文档

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

← 返回到 API 参考

系统状态

检查 API 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。

← 返回到 API 参考

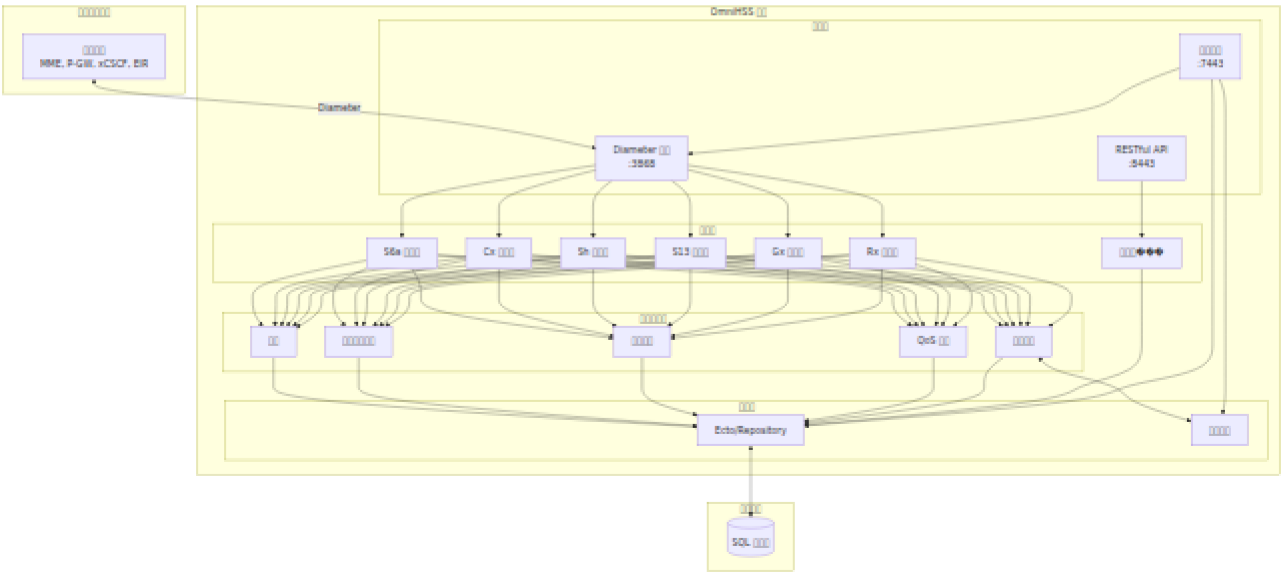
OmniHSS 架构概述

← 返回操作指南

目录

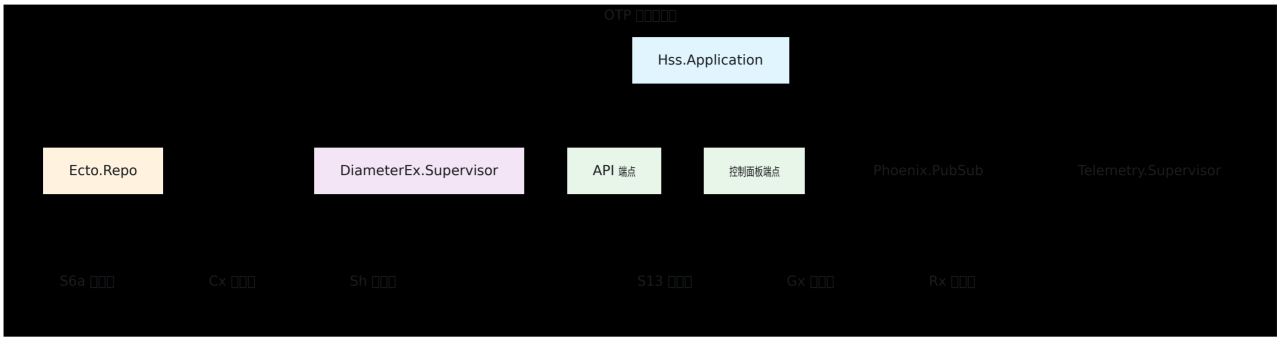
系统概述

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

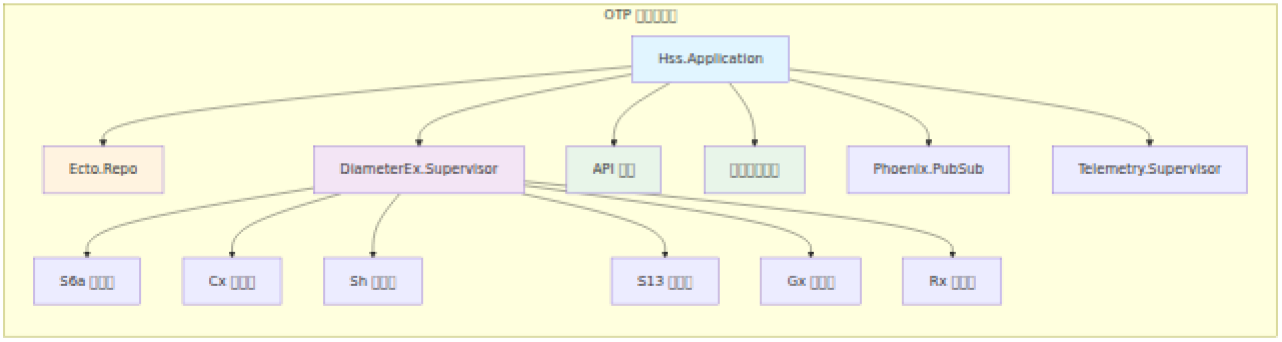
核心组件



Diameter 应用处理器

每个 Diameter 应用（S6a、Cx、Sh、S13、Gx、Rx）实现为一个 DiameterEx 处理器模块，该模块：

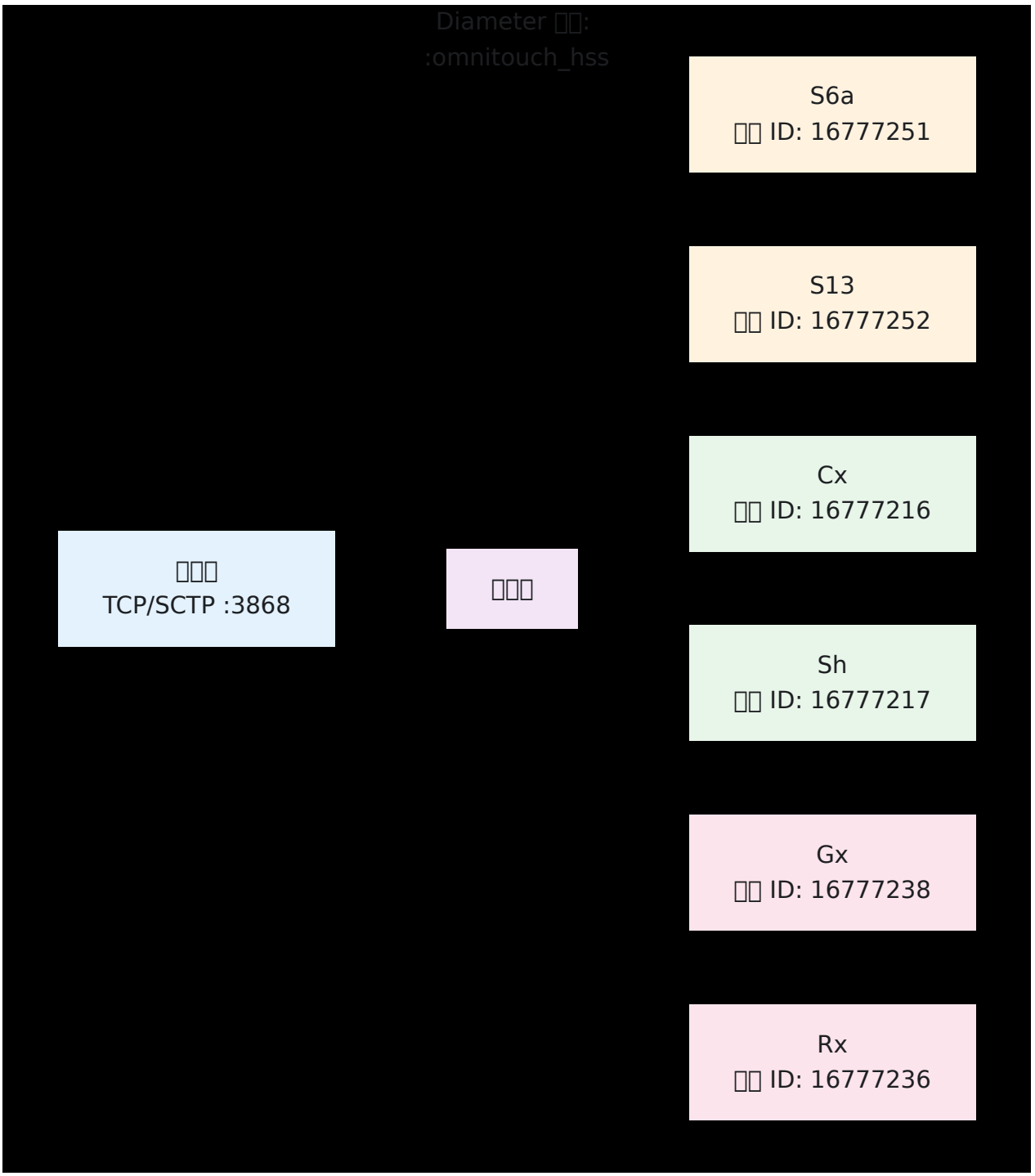
1. 向 **DiameterEx** 注册 - 订阅特定的 Diameter 应用 ID
2. 验证请求 - 提取 AVP，验证订阅者状态
3. 处理业务逻辑 - 调用适当的业务逻辑模块
4. 构建响应 - 使用 AVP 构建 Diameter 答复消息
5. 处理错误 - 返回适当的 Diameter 结果代码



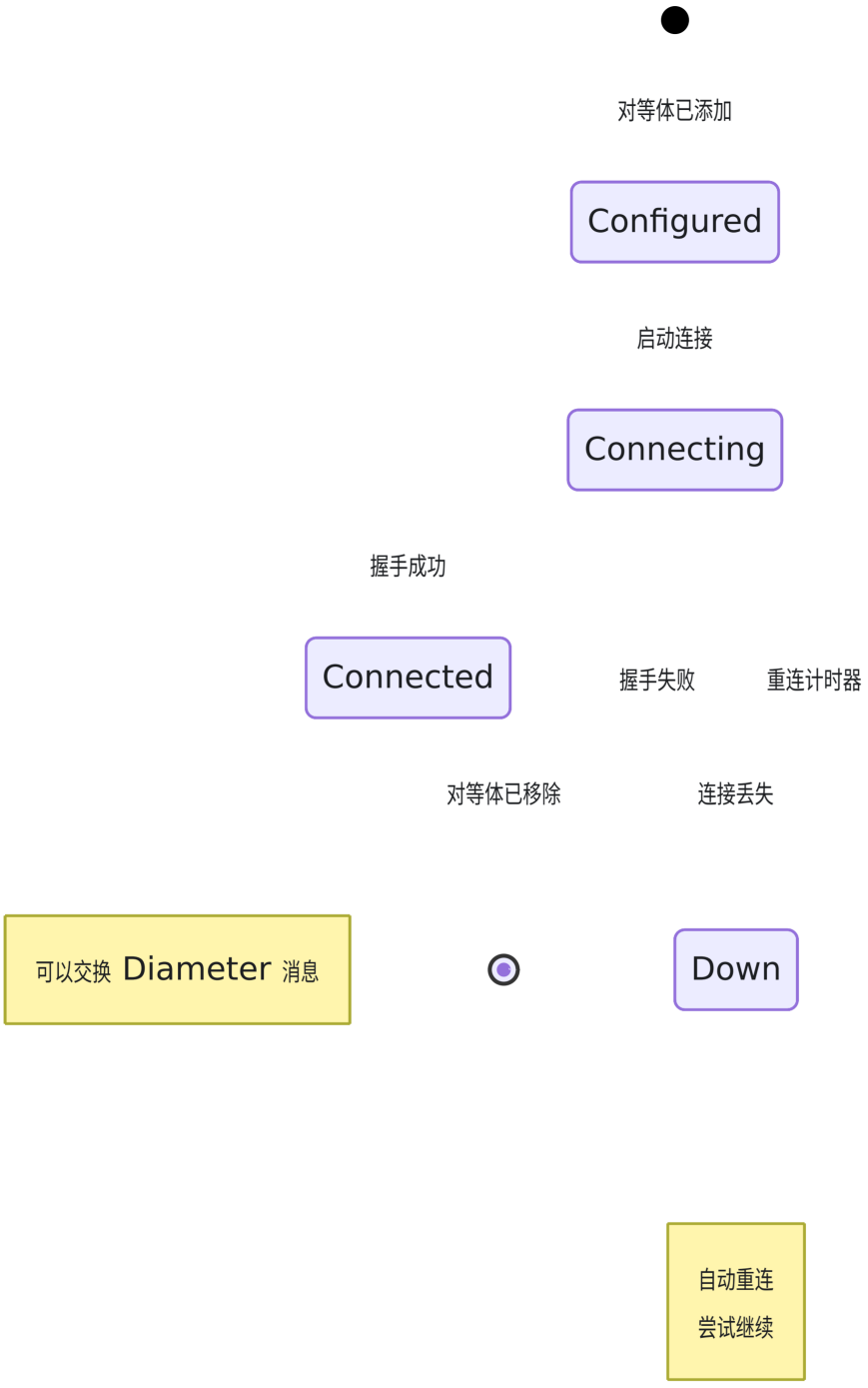
Diameter □□

Diameter □□□□

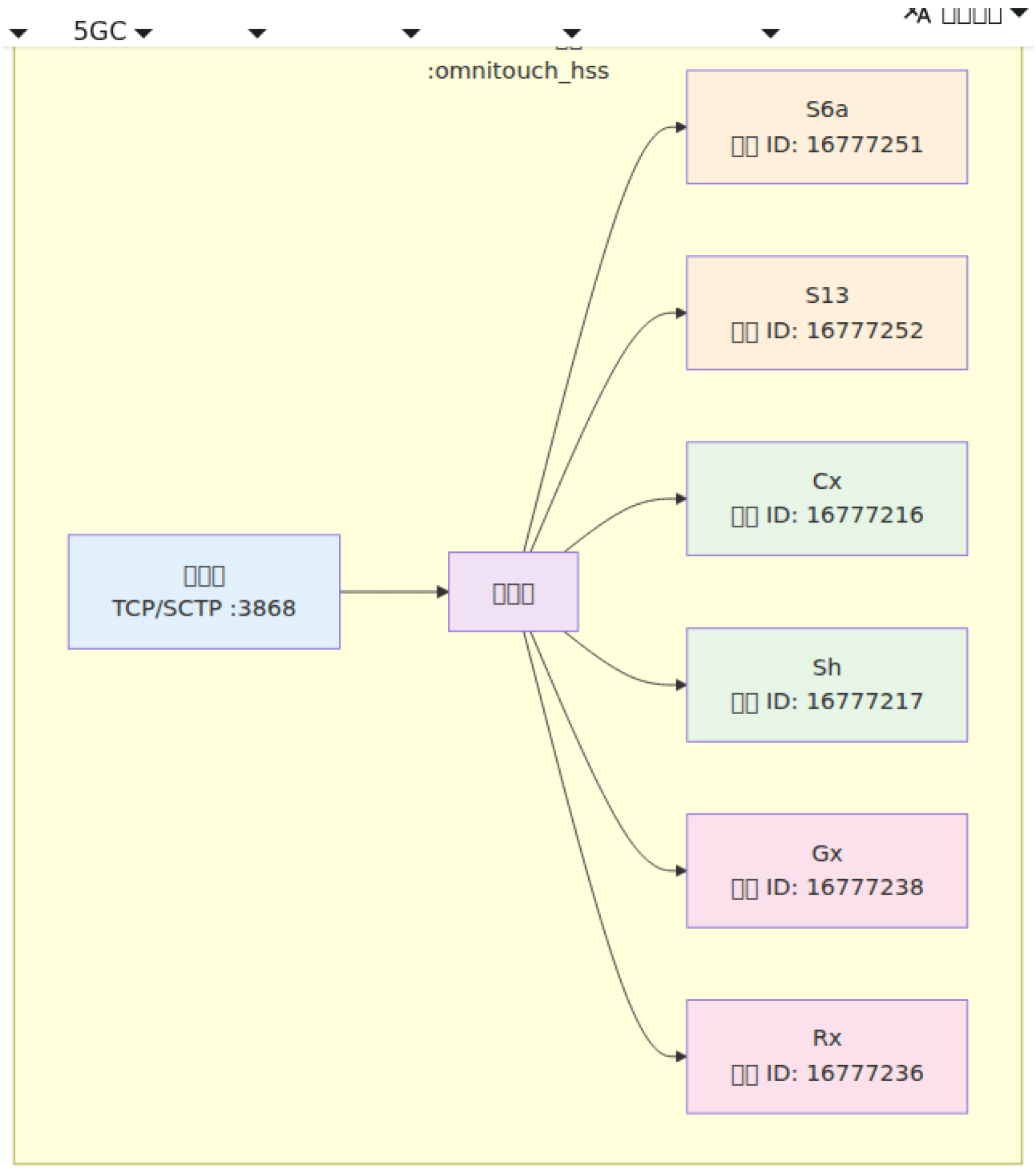
OmniHSS □□□□□□□□□□□□□□□□ Diameter □□□



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

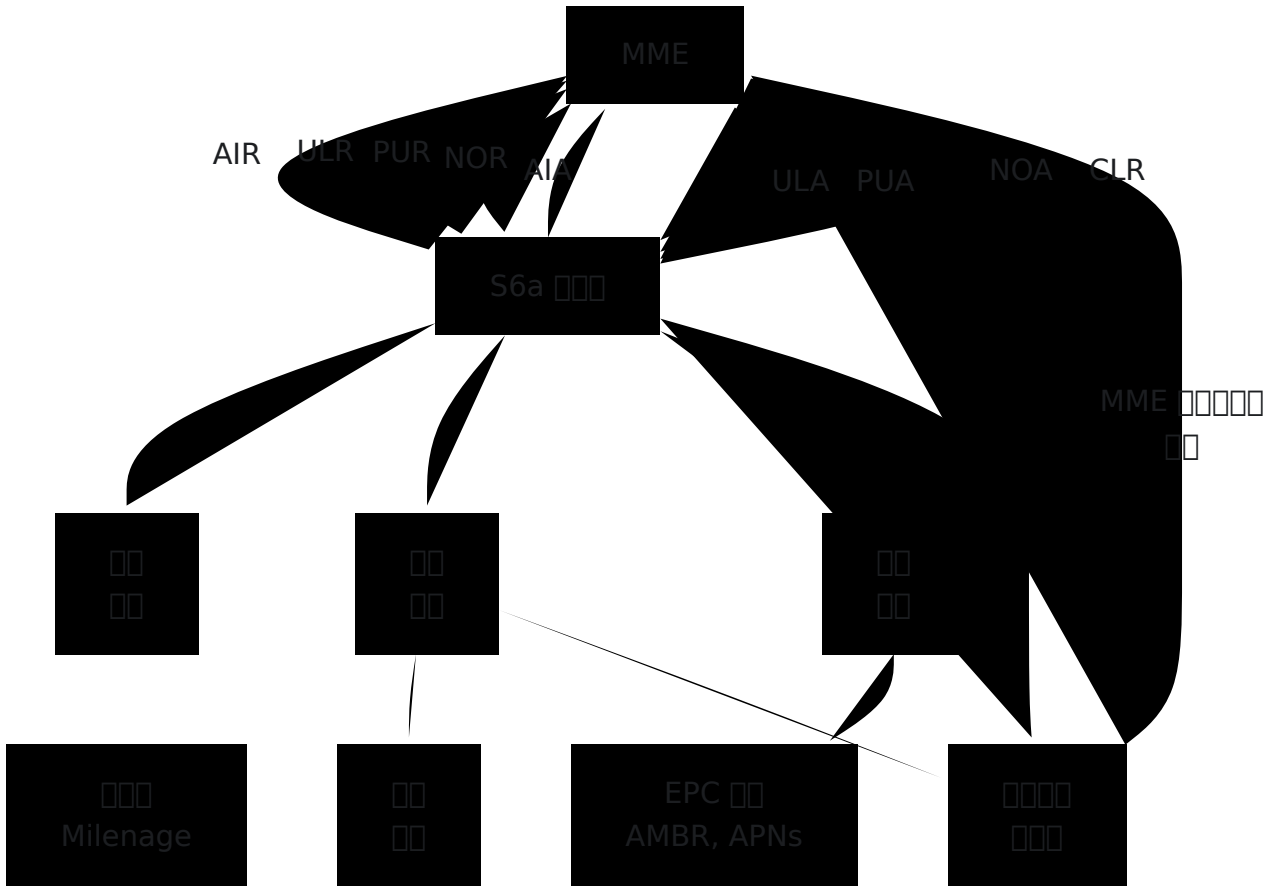
Diameter 消息流



□□□

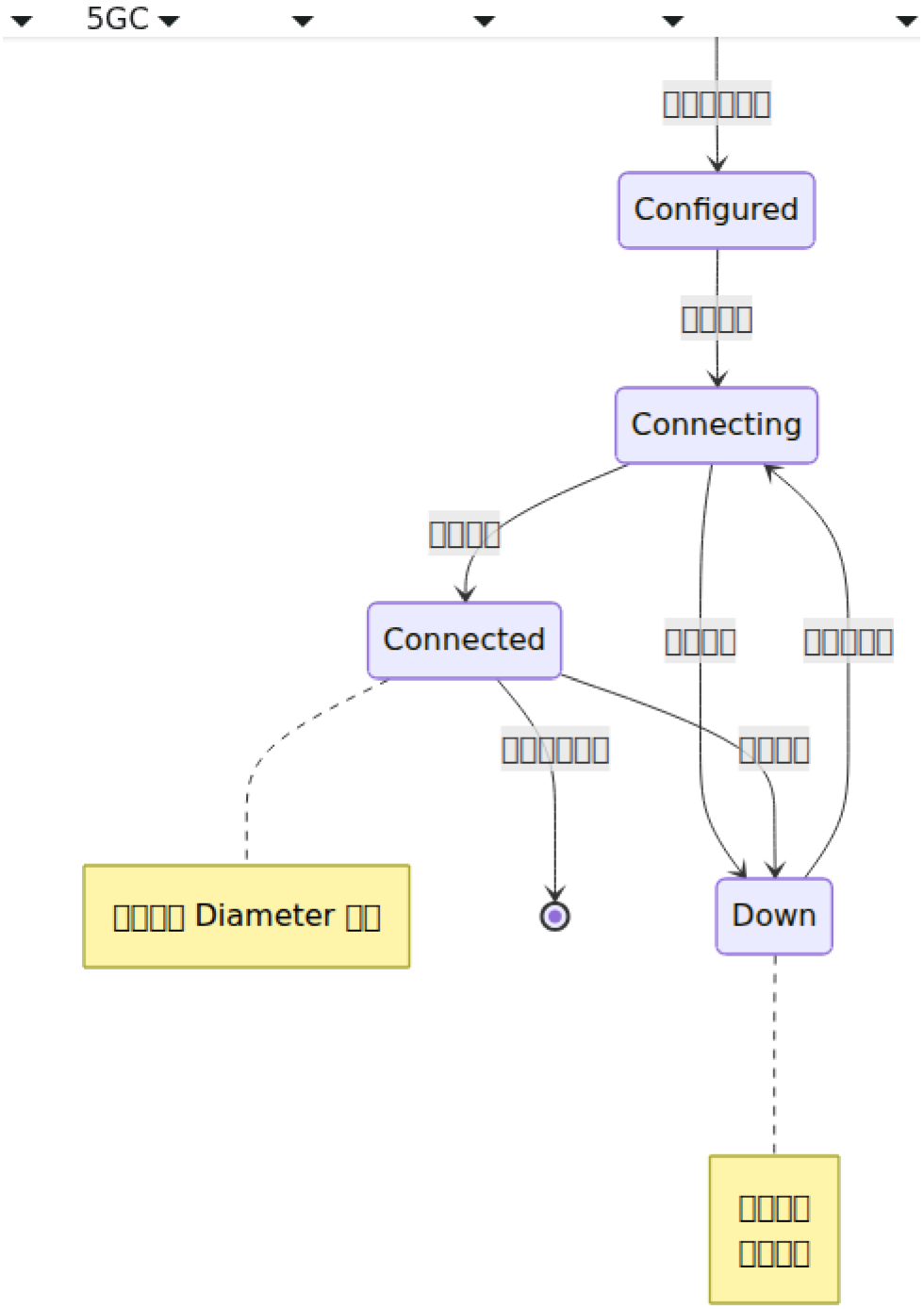
S6a □□ (LTE/EPC)

□□ LTE □□□□□□□□□□□□□



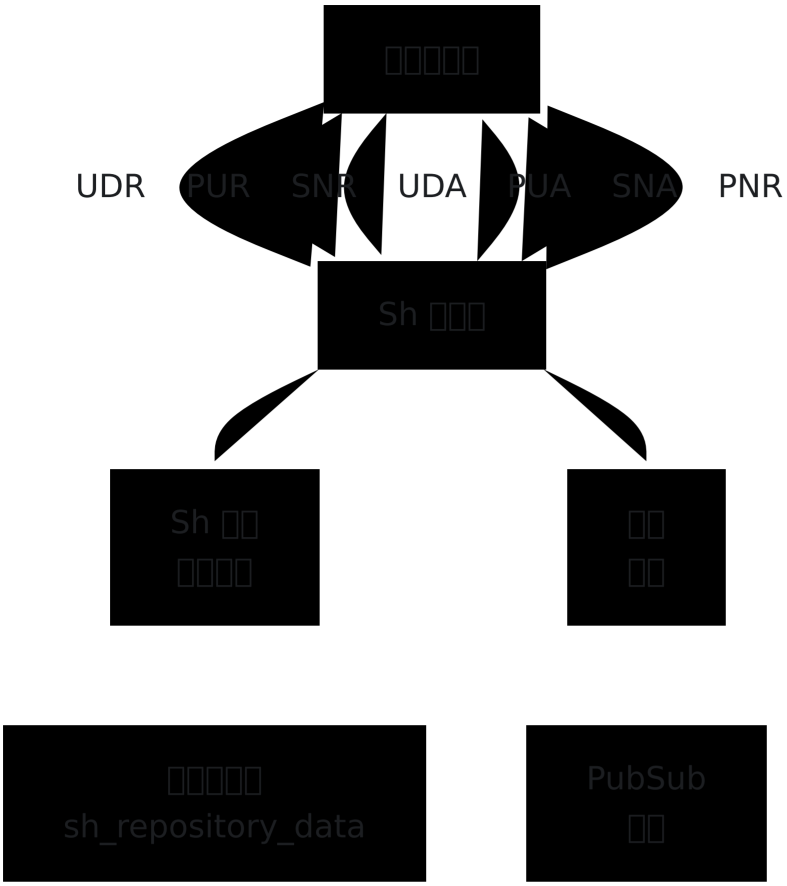
Cx □□ (IMS)

□□ IMS □□□□□□



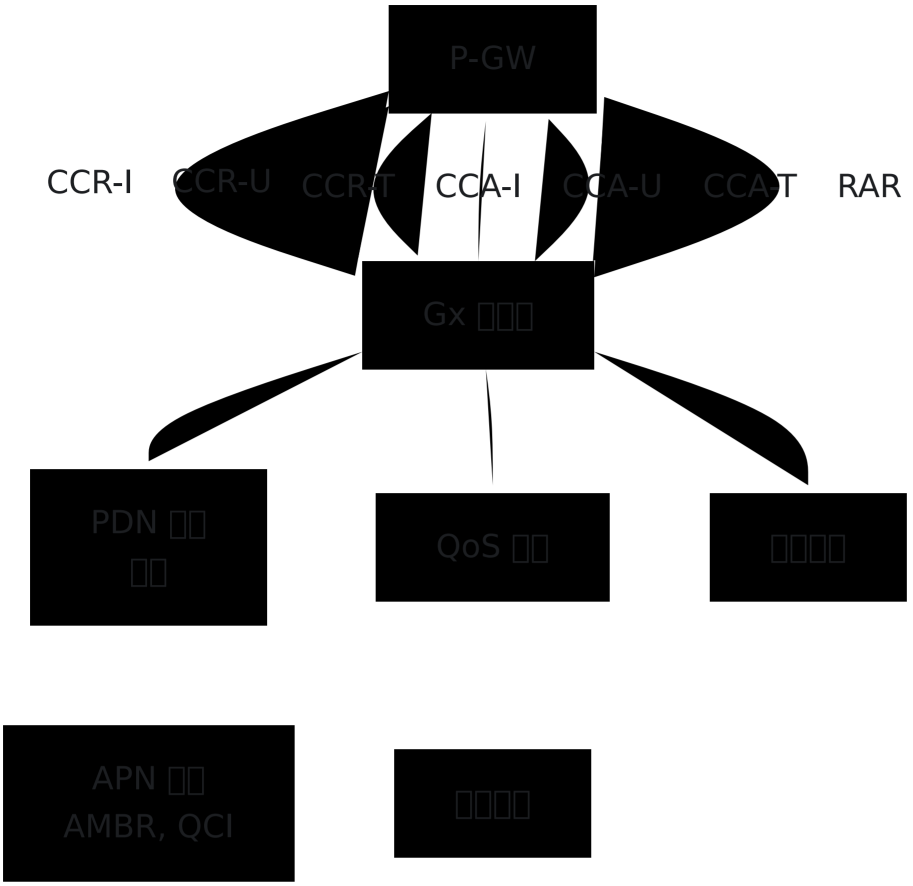
Sh 接口 (IMS 个人资料数据)

为 IMS 应用服务器提供访问订阅者个人资料数据的权限。



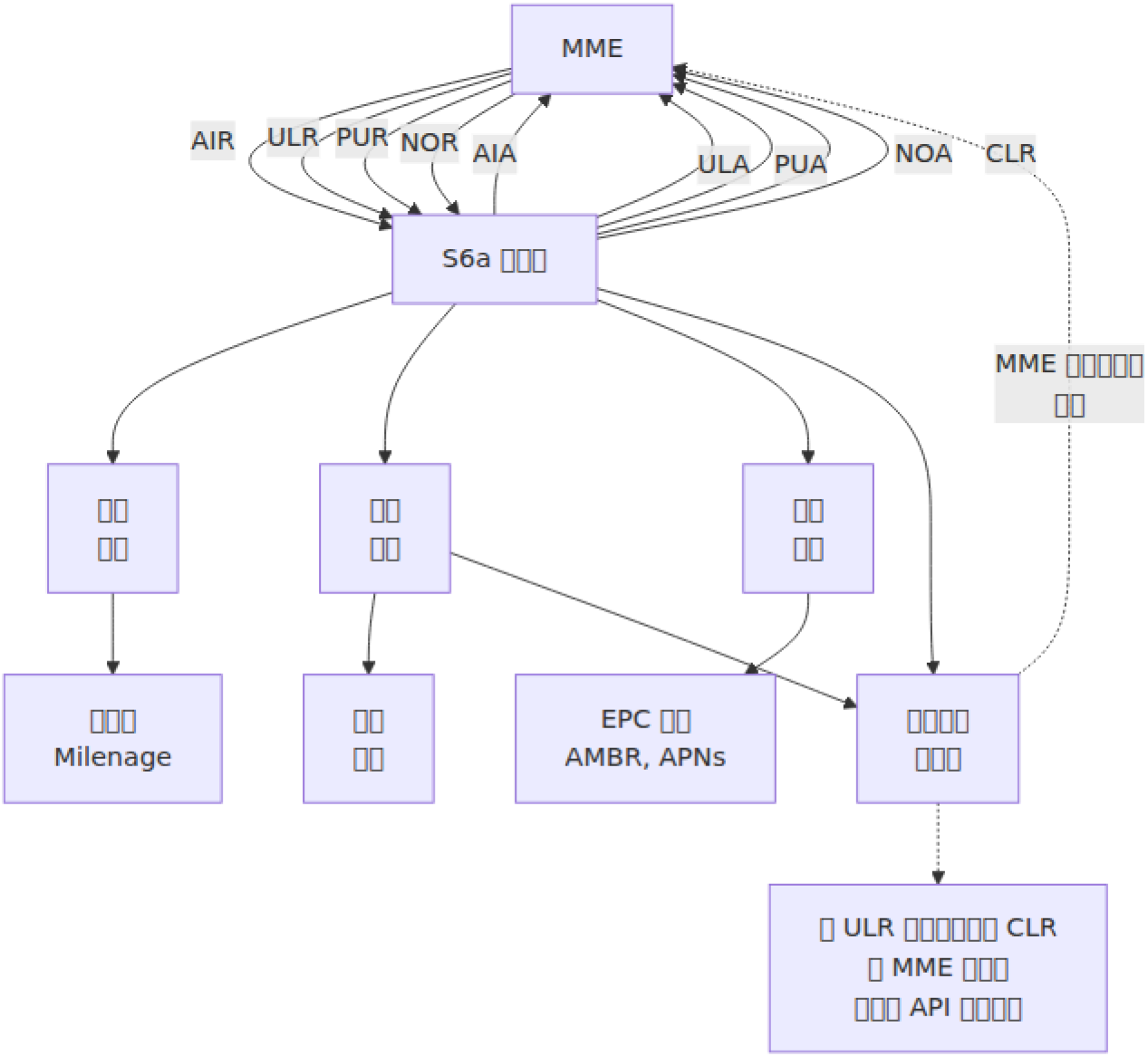
Gx 接口（策略控制）

管理数据会话的策略和计费控制。有关详细信息，请参见 **PCRF** 文档。



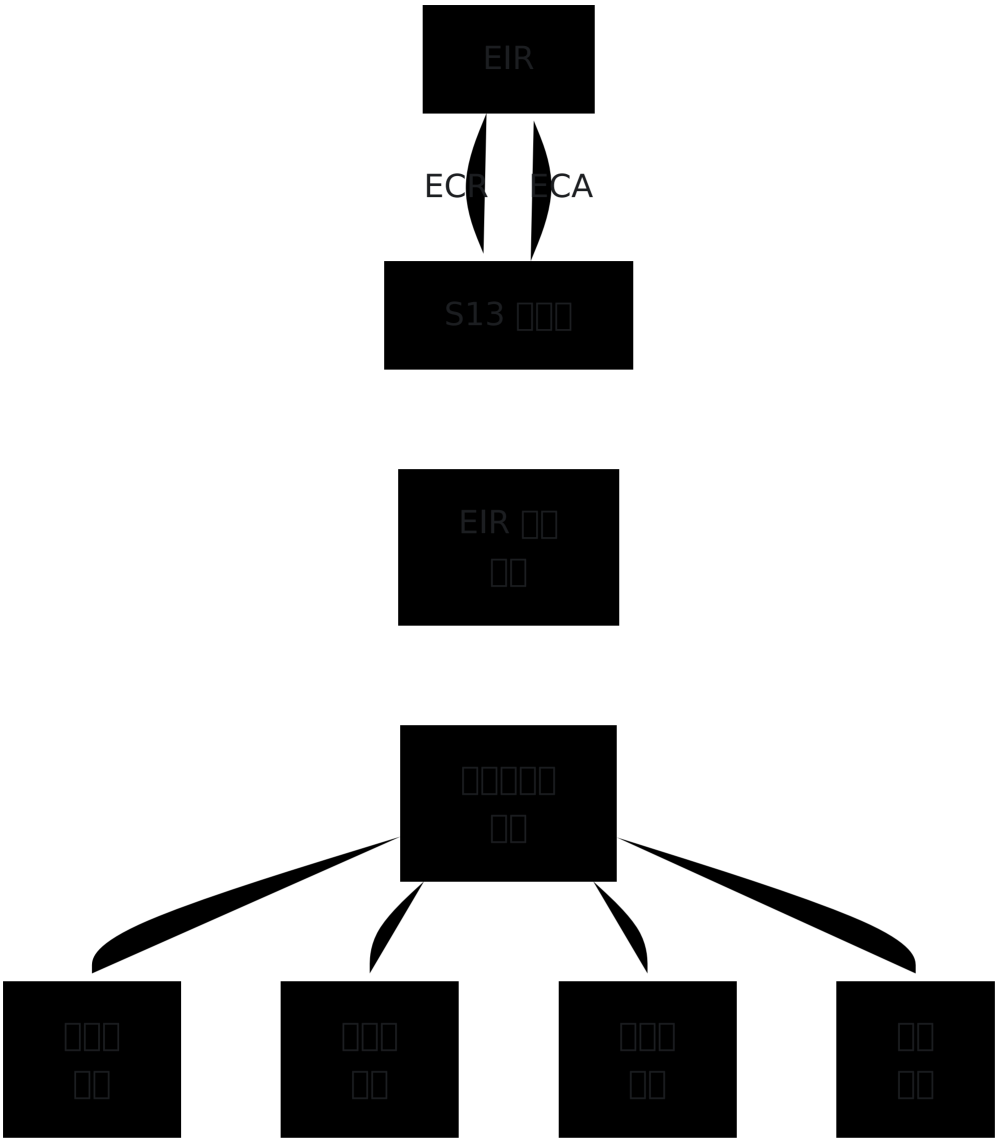
Rx 接口 (IMS 媒体)

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

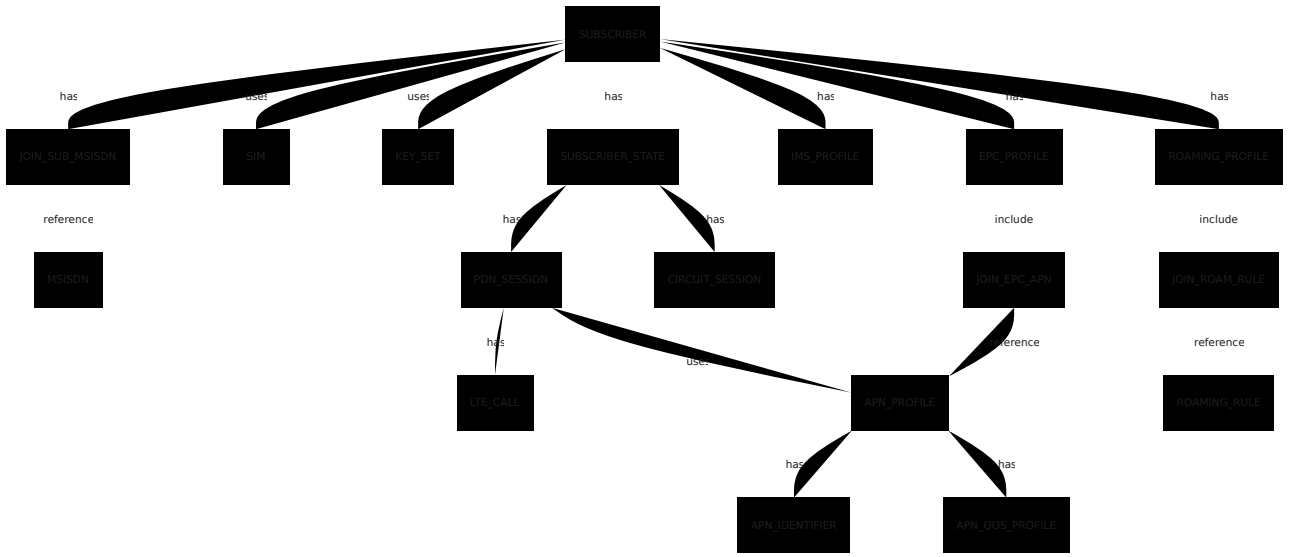
数据库后端

OmniHSS 使用 **Ecto** 作为其数据库抽象层。Ecto 支持多种关系数据库后端，允许在数据库选择上具有灵活性。**MariaDB** 与 Galera 集群□支持的配置之一。

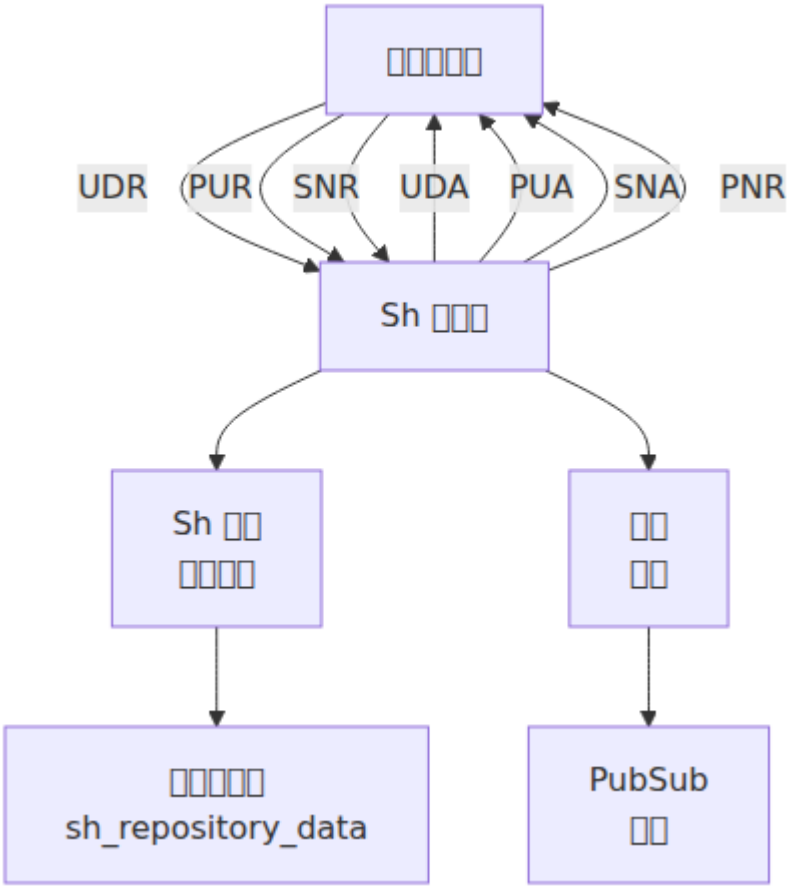
根据您的基础设施需求，可以使用其他数据库后端。与 **ONS** 的集成团队合作，确定最合适的数据库后端和复制策略。

有关 Galera 集群配置的信息，请参见 Galera 数据库复制。

□□□□□□□

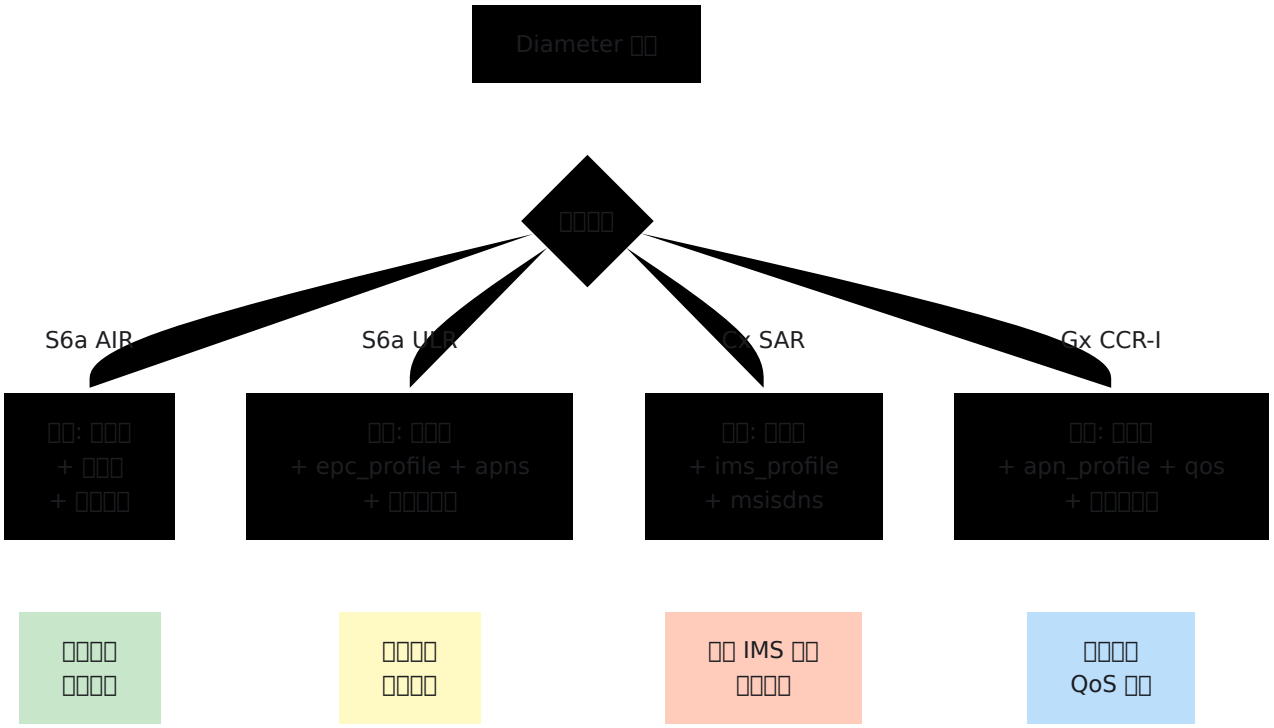

SUBSCRIBER
has
has
has
has
has
JOIN_SUB_MSISDN
SIM
KEY_SET
SUBSCRIBER_STATE
IMS_PROFILE
EPC_PROFILE
ROAMING_PROFILE
reference
has
has
include
include
MSISDN
PDN_SESSION
CIRCUIT_SESSION
JOIN_EPC_APN
JOIN_ROAM_RULE
has
reference
reference
LTE_CALL
APN_PROFILE
ROAMING_RULE
has
has
APN_IDENTIFIER
APN_QOS_PROFILE

Ecto □□□□


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

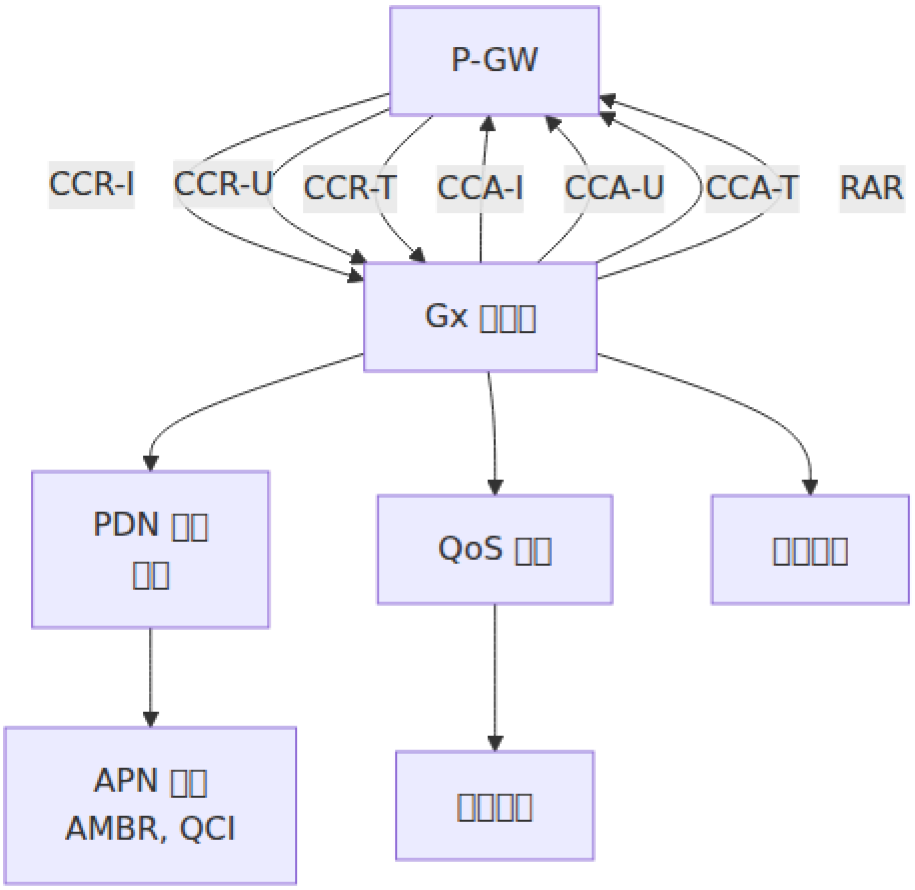
□□□□□□□

□□ Diameter □□□□□□□□□□□□□□□□□□□□□□□□

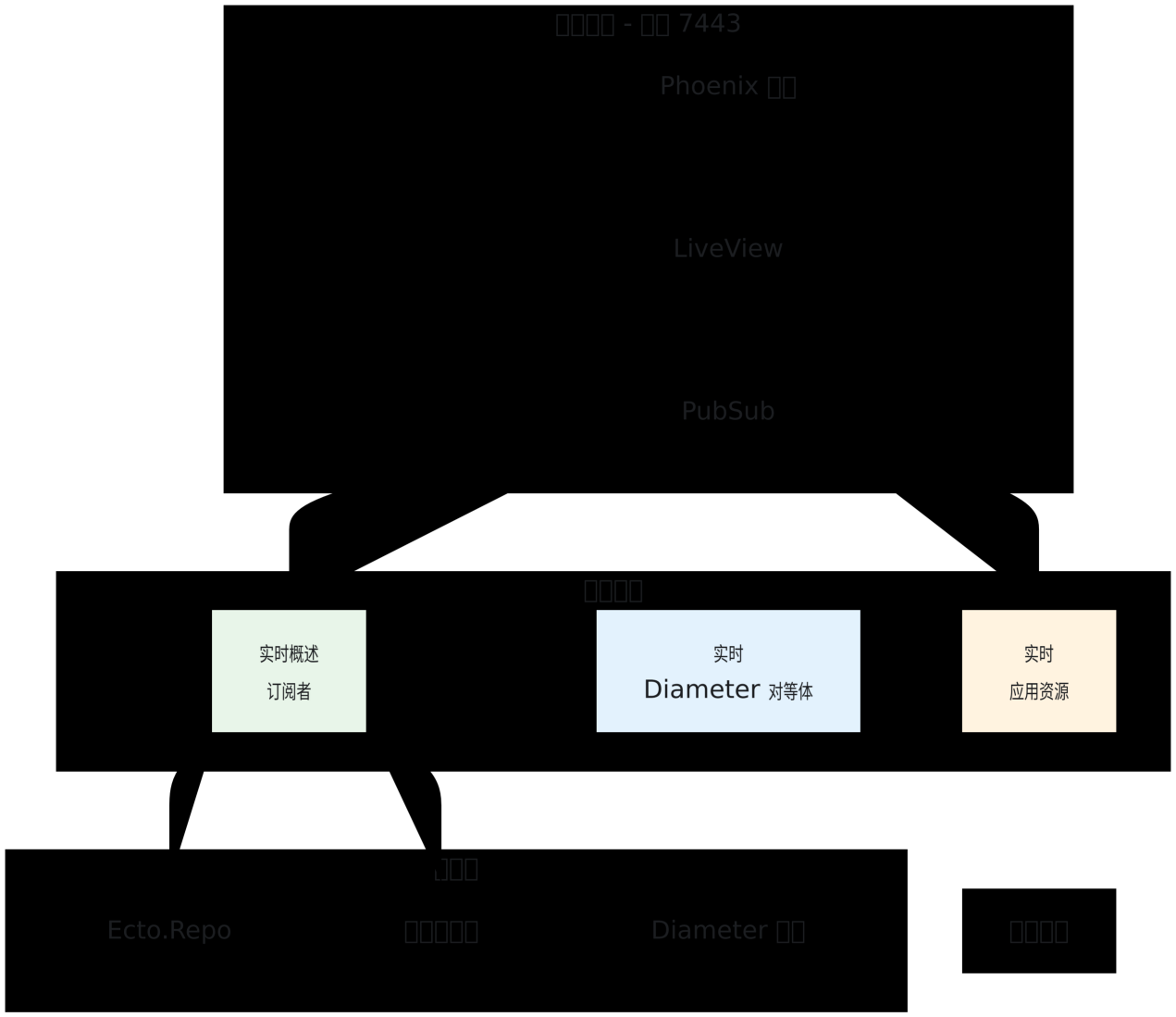

Diameter □□
□□□□□
S6a AIR
S6a ULR
Gx SAR
Gx CCR-I
□□: □□□
+ □□□
+ □□□□
□□: □□□
+ epc_profile + apns
+ □□□□□□
□□: □□□
+ ims_profile
+ msisdns
□□: □□□
+ apn_profile + qos
+ □□□□□□
□□□□
□□□□
□□□□
□□□□
□□ IMS □□
□□□□
□□□□
QoS □□

外部接口

API 架构

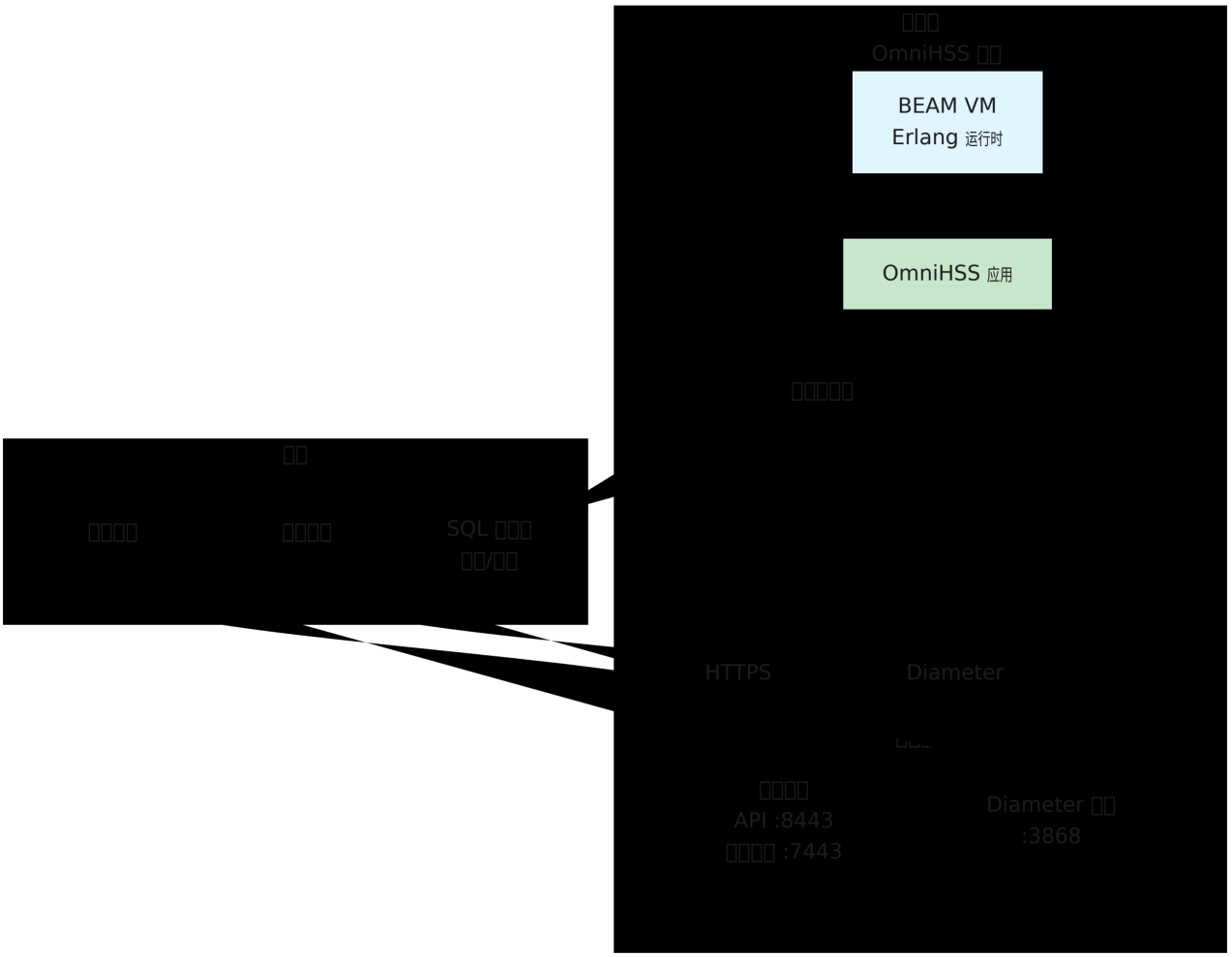


控制面板架构


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

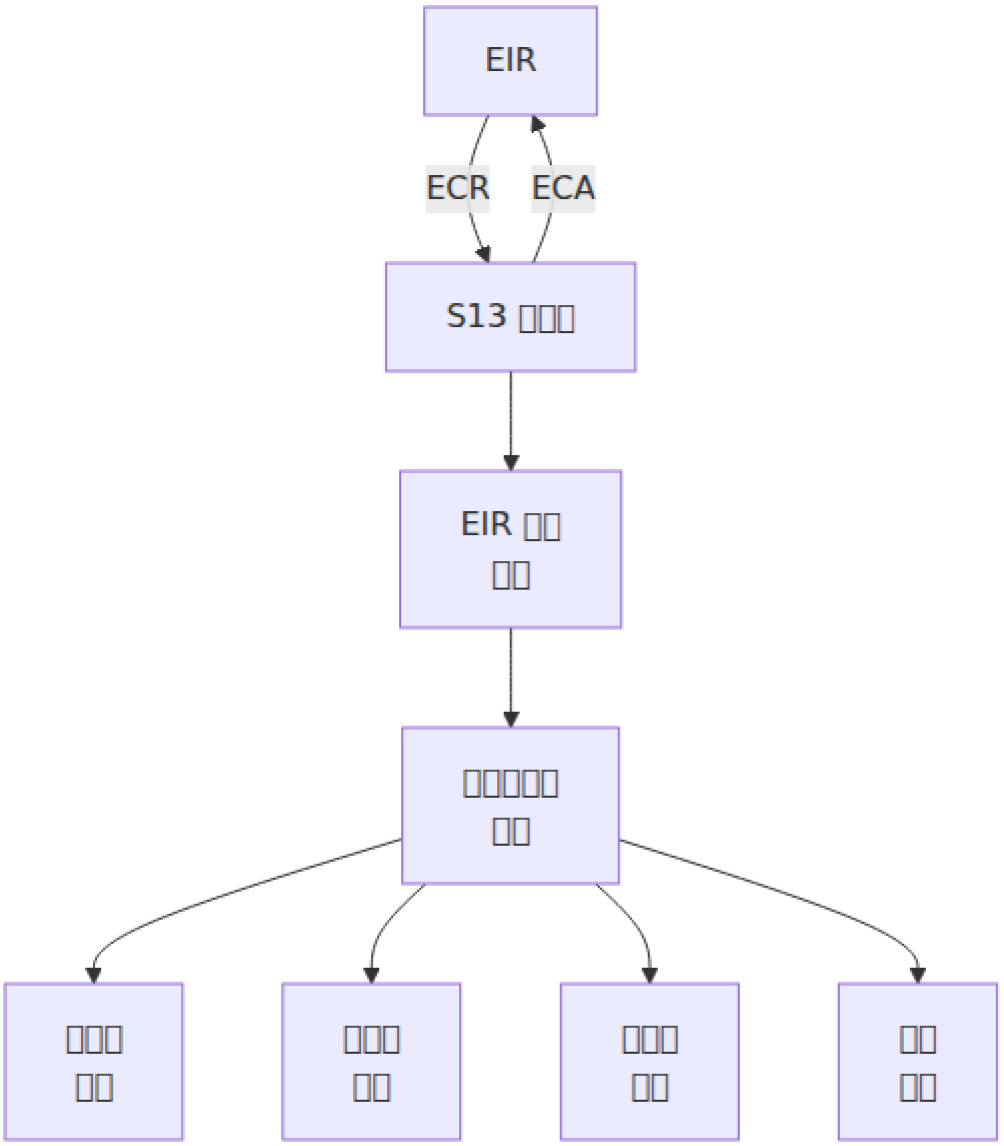
部署架构

单节点部署



多节点高可用部署 **(Galera** 集群**)**

对于高可用性部署，OmniHSS 支持 MariaDB Galera 集群进行同步的多主数据库复制。



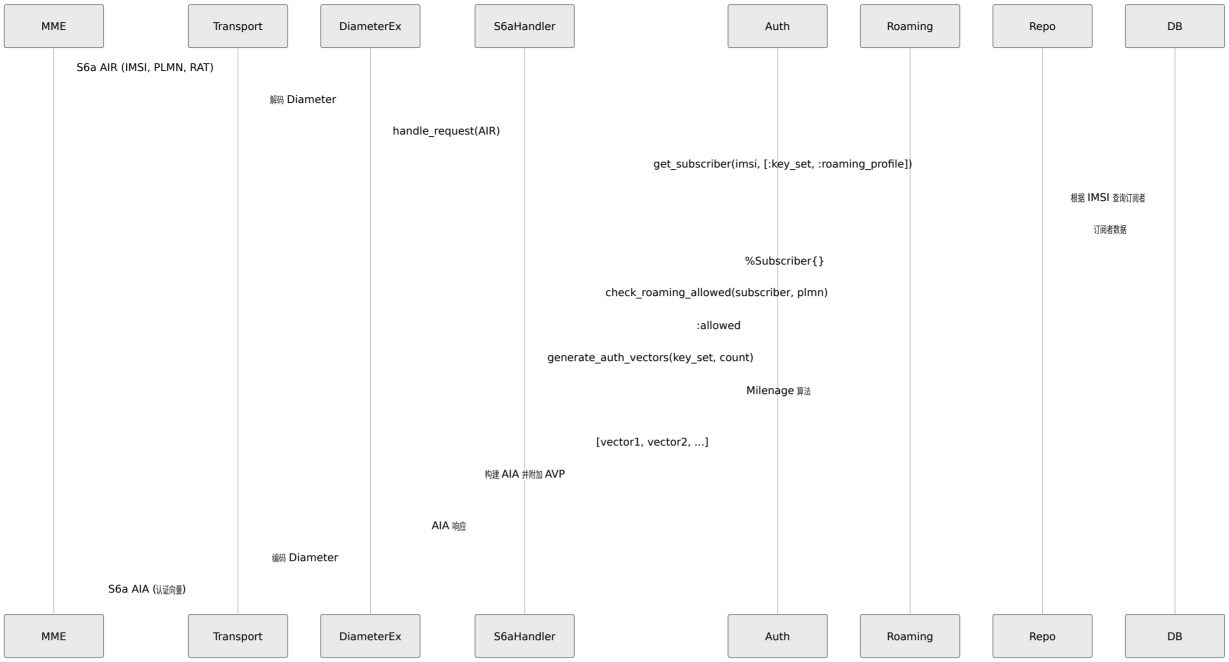
主要特征：

- 同步复制：所有写入在返回成功之前都在所有节点上提交
- 多主：任何节点都可以接受读写操作
- 自动故障转移：如果一个节点失败，其他节点继续运行而不会丢失数据
- 基于认证：跨所有节点验证事务以防止冲突

有关详细配置和□作的信息，请参见 **Galera** 数据库复制。

流程示例：认证

此示例显示了认证请求的完整流程：



关键架构原则

1. 容错性

- Erlang/OTP 监督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

- 优化的数据库查询具有选择性预加载
- 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

- 通过控制面板��行实时监控
- 应用程序中的结构化日志记录
- Diameter 对等体状态跟踪
- 订阅者状态跟踪及时间戳

← 返回操作指南 | 下一步：配置 →

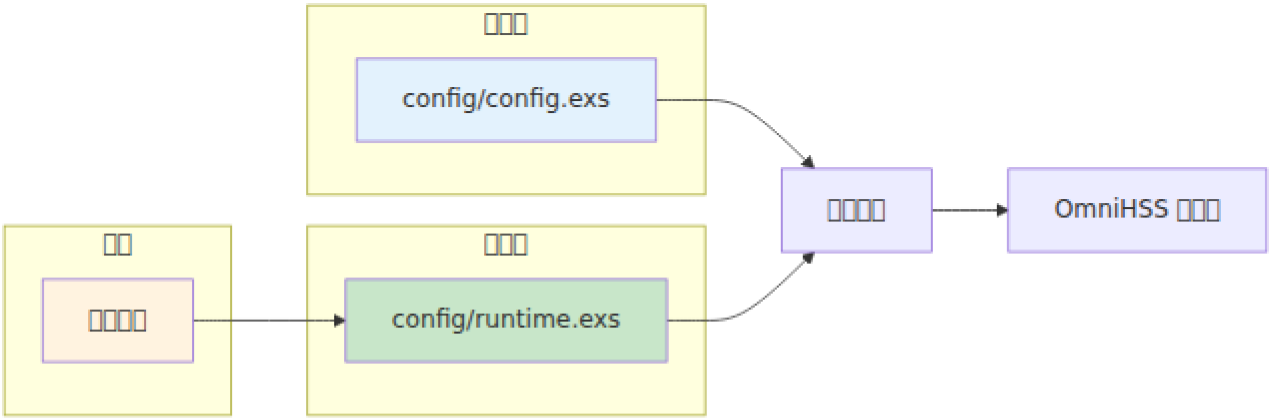
OmniHSS 配置指南

← 返回操作指南

目录

配置文件概述

OmniHSS 使用两个主要配置文件：



config/config.exs（编译时）

包含在不同环境之间不变的静态配置：

- 控制面板页面配置
- API 端点配置
- 遥测设置

config/runtime.exs（运行时）

包含每次部署时变化的特定于环境的配置：

- 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：

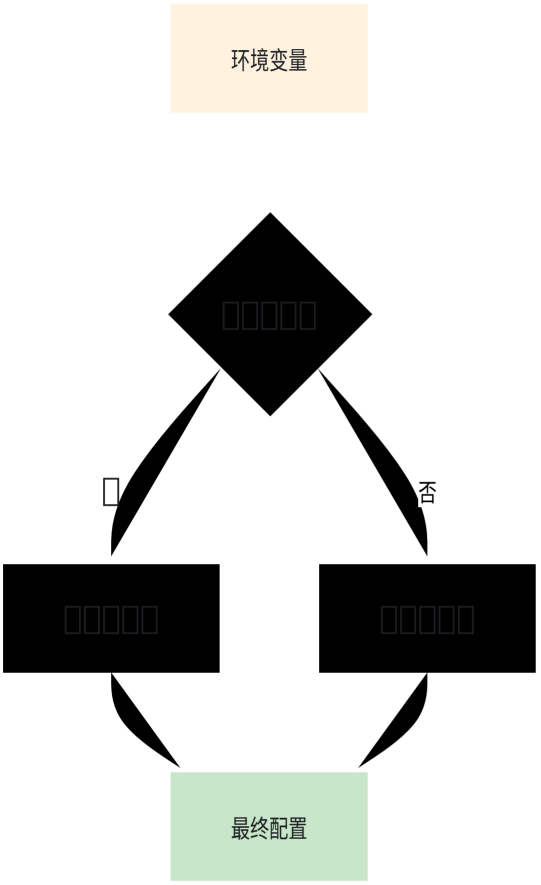
参数	描述	必需	示例
`license_server_api_urls`	许可证服务器 URL 列表	✅	`["https://10.0.0.1:8443/api"]`
`licensee`	许可证上的组织名称	✅	`"ACME Telecom"`
`product_name`	许可证的产品标识符	✅	`"omnihss"`

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发生

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带有下划线
- 默认值在 runtime.exs 中提供
- 数据库凭据在生产环境中应使用环境变量

数据库配置

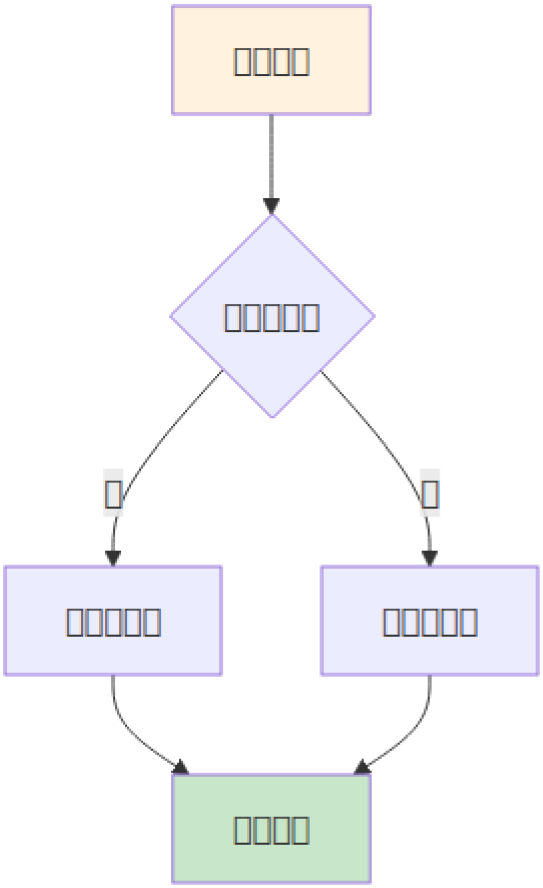
基本数据库配置

```
# 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	建议
`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 为典型值）

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

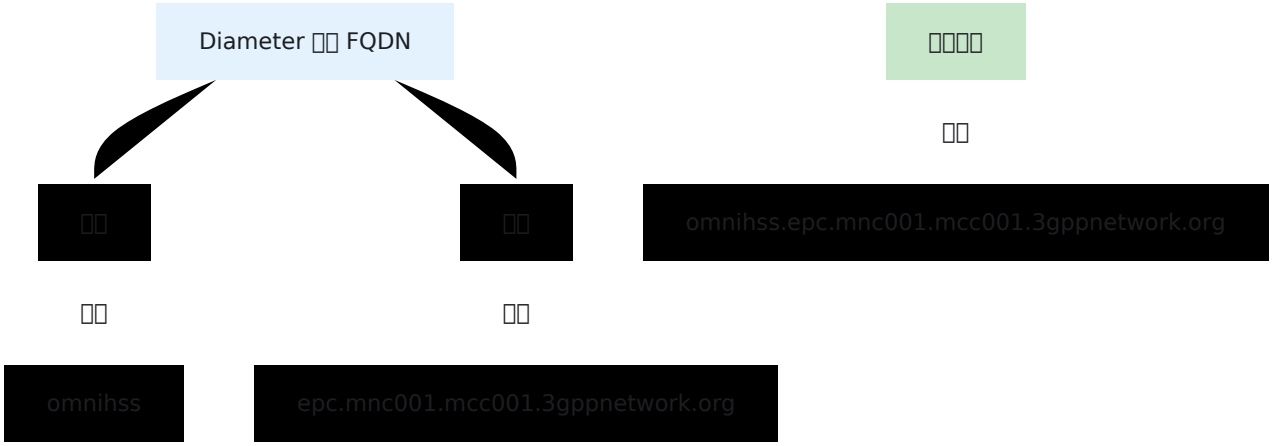
```
# config/runtime.exs - 生产示例

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",
"3306")),
  pool_size:
String.to_integer(System.get_env("DATABASE_POOL_SIZE", "30")),
  ssl: true,
  ssl_opts: [
    cacertfile: "/etc/ssl/certs/mysql-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 □□□

□□□□□□□□□□□□□□□

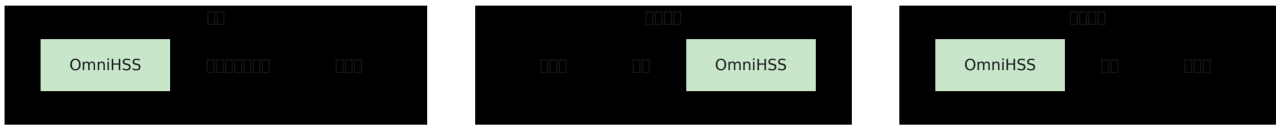
```
# config/runtime.exs

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到 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	优势	劣势	建议
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 识别您的网络运营商：

```
# 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

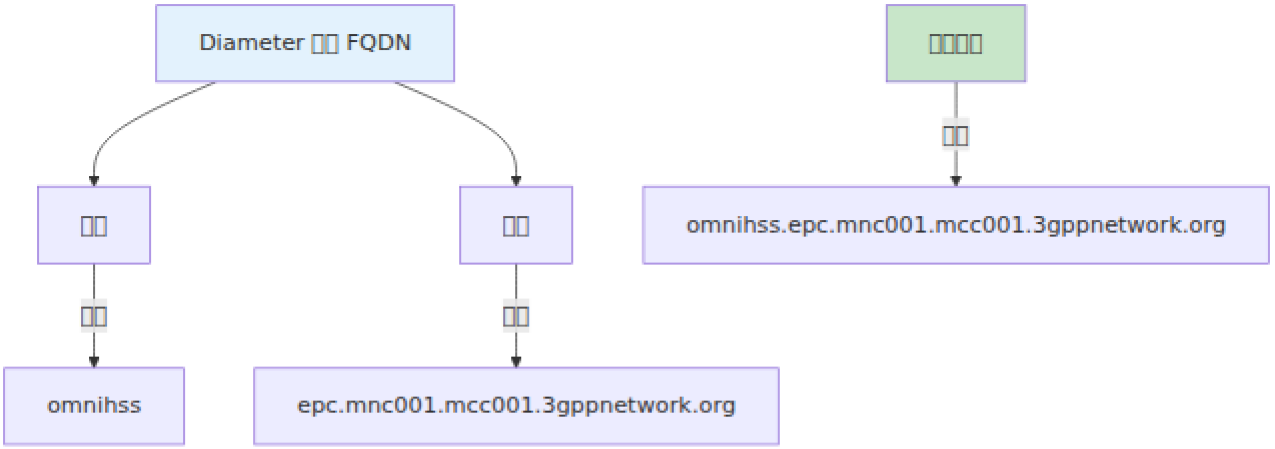
这些设置控制 HSS 的行为和功能：

```
# config/runtime.exs

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

  # MME 更改时的 CLR（取消位置请求）
  send_clr_on_mme_change: true,

  # 在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	建议
`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`	生产中必需
`license_module`	处理许可证检查的模块	`LicenseClient`	不要更改

PLMN 代码格式



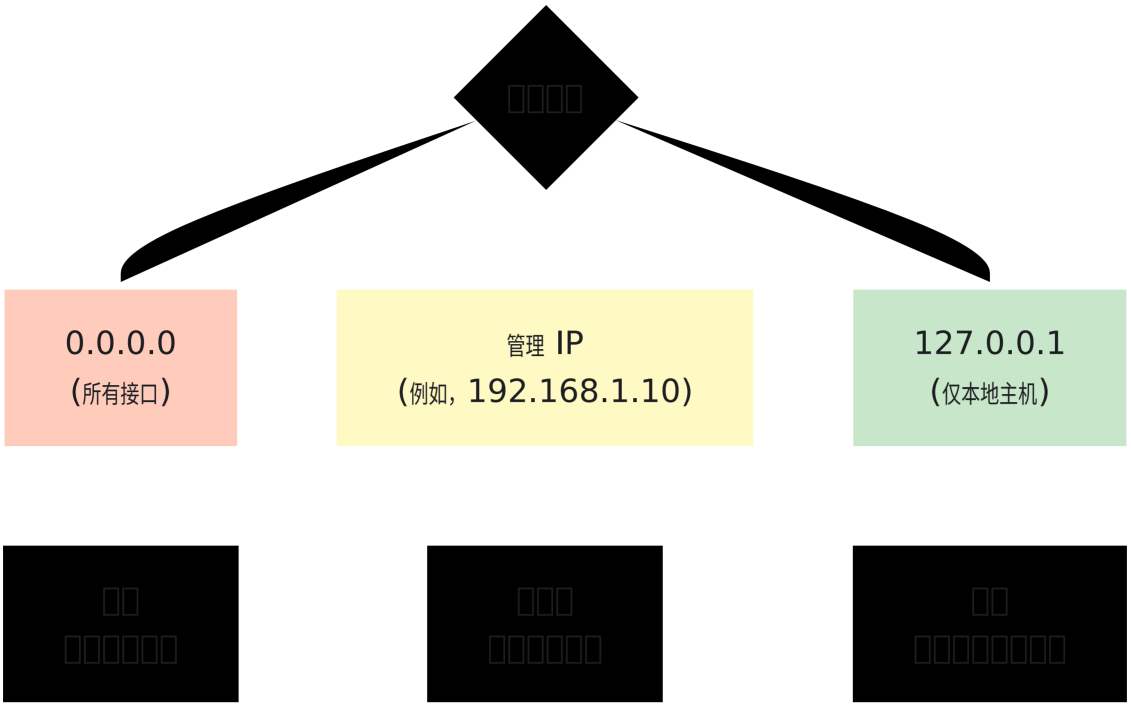
示例:

- 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:

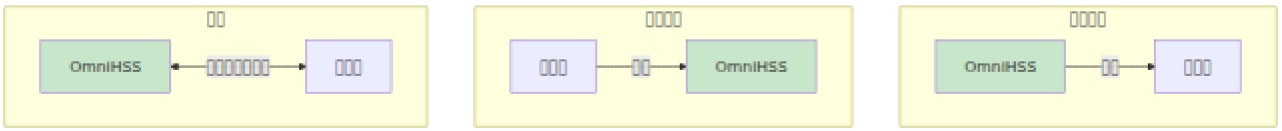

0.0.0.0
(所有接口)
管理 IP
(例如，192.168.1.10)
127.0.0.1
(仅本地主机)

IMS 配置

S-CSCF 选择配置

```
# config/runtime.exs

config :hss, :ims, %{
  scscf: %{
    # 选择法: :random_peer 或 :round_robin
    selection_method: :random_peer,

    # 可用 S-CSCF 对等体列表
    peers: [
      %{
        host:
"sip:scscf01.ims.mnc001.mcc001.3gppnetwork.org:5060",
        capabilities: []  # 可选: 能力匹配
      },
      %{
        host:
"sip:scscf02.ims.mnc001.mcc001.3gppnetwork.org:5060",
        capabilities: []
      }
    ]
  }
}
```

S-CSCF 选择方法



选择方法:

方法	描述	用例
`:random_peer`	随机选择一个 S-CSCF	均匀负载分配
`:round_robin`	顺序分配 S-CSCF	可预测的分配

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"
```

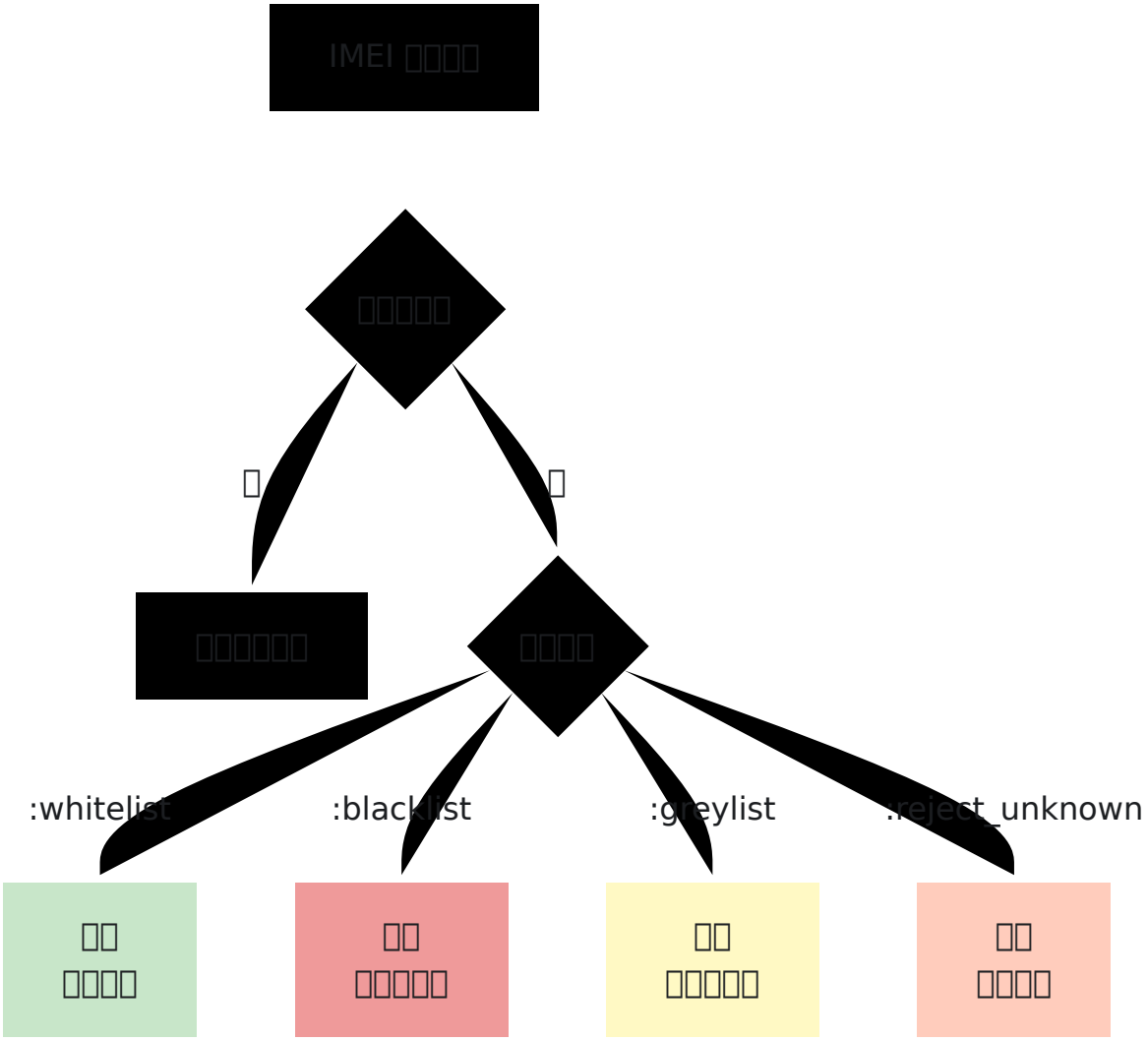
EIR 配置

请参阅 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 - 使用特定结果代码拒绝
}
```

□□□□□□



□□□□□

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

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

API 和控制面板配置

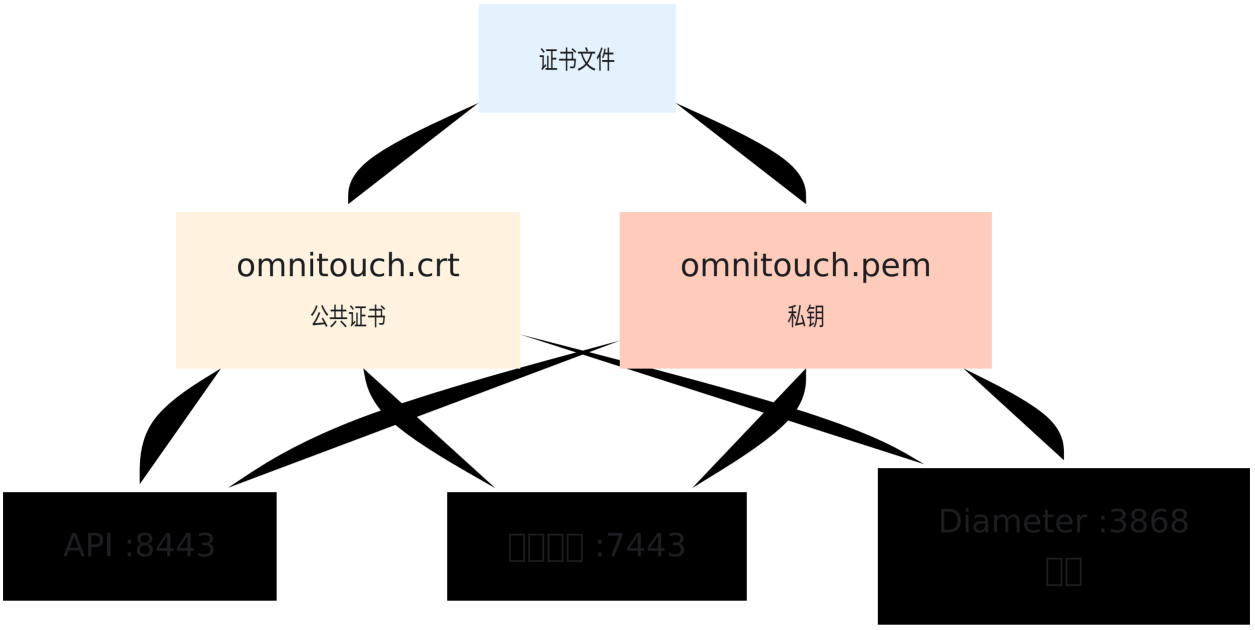
API 端点配置

```
# 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：

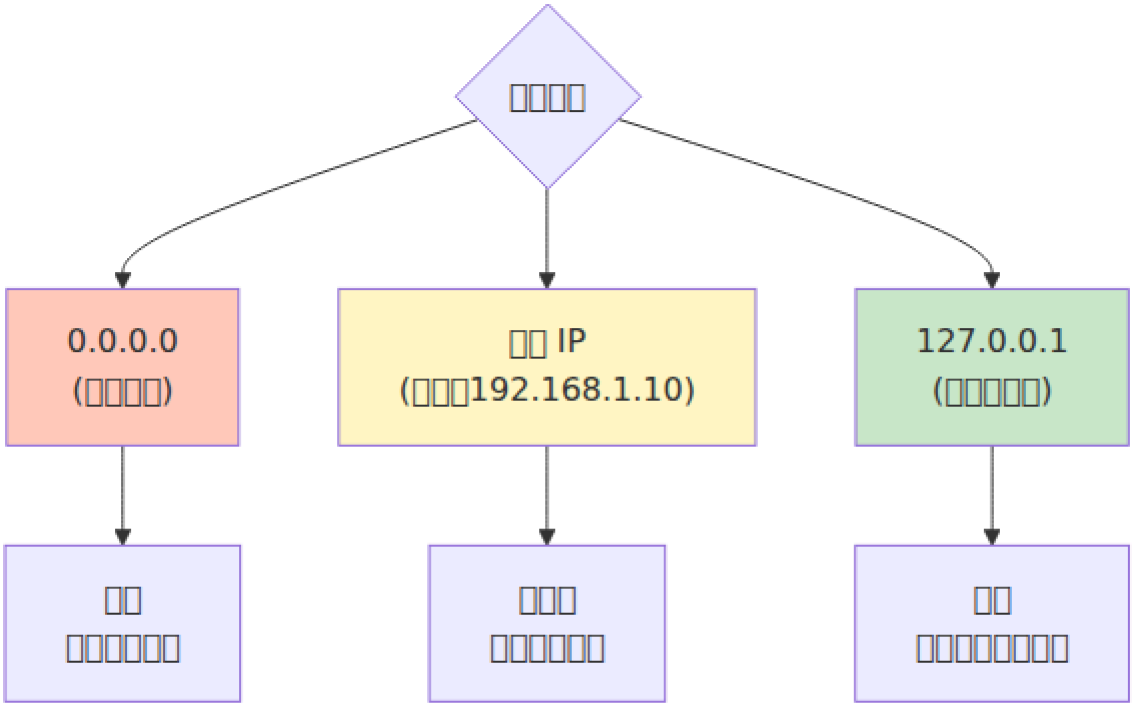
- 有效的 X.509 证书
- 匹配的私钥
- 如有需要，包含中间证书
- CN 或 SAN 必须与主机名匹配

对于生产：

```
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



配置检查表

基本配置

- [ ] 数据库连接（主机名，凭据）
- [ ] 家庭 PLMN（MCC，MNC）
- [ ] Diameter 主机和领域
- [ ] Diameter 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��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 Diameter 对等体（如果使用连接模式）
- [ ] 防火墙规则允许 Diameter 流量（端口 3868）
- [ ] 防火墙规则允许 HTTPS 流量（端口 7443，8443）

- [ ] Diameter 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`
   验证概览页面加载

3. **API** 访问:

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

4. **Diameter** 状态:

   检查控制面板 `Diameter` 页面
   验证对等连接

5. 数据库连接:

检查控制面板的订阅者数据
或直接连接到 SQL 数据库

完整运行时配置示例

```
# config/runtime.exs - 完整生产示例

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", "3306")),
  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/mysql-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 □□
#
============================================================================
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

OmniHSS 控制面板□一个基于网页的监控界面，提供系统状态、用户活动和 Diameter 连接的实时可视化。它使用 Phoenix LiveView 构建，能够自动更新，无需页面刷新。

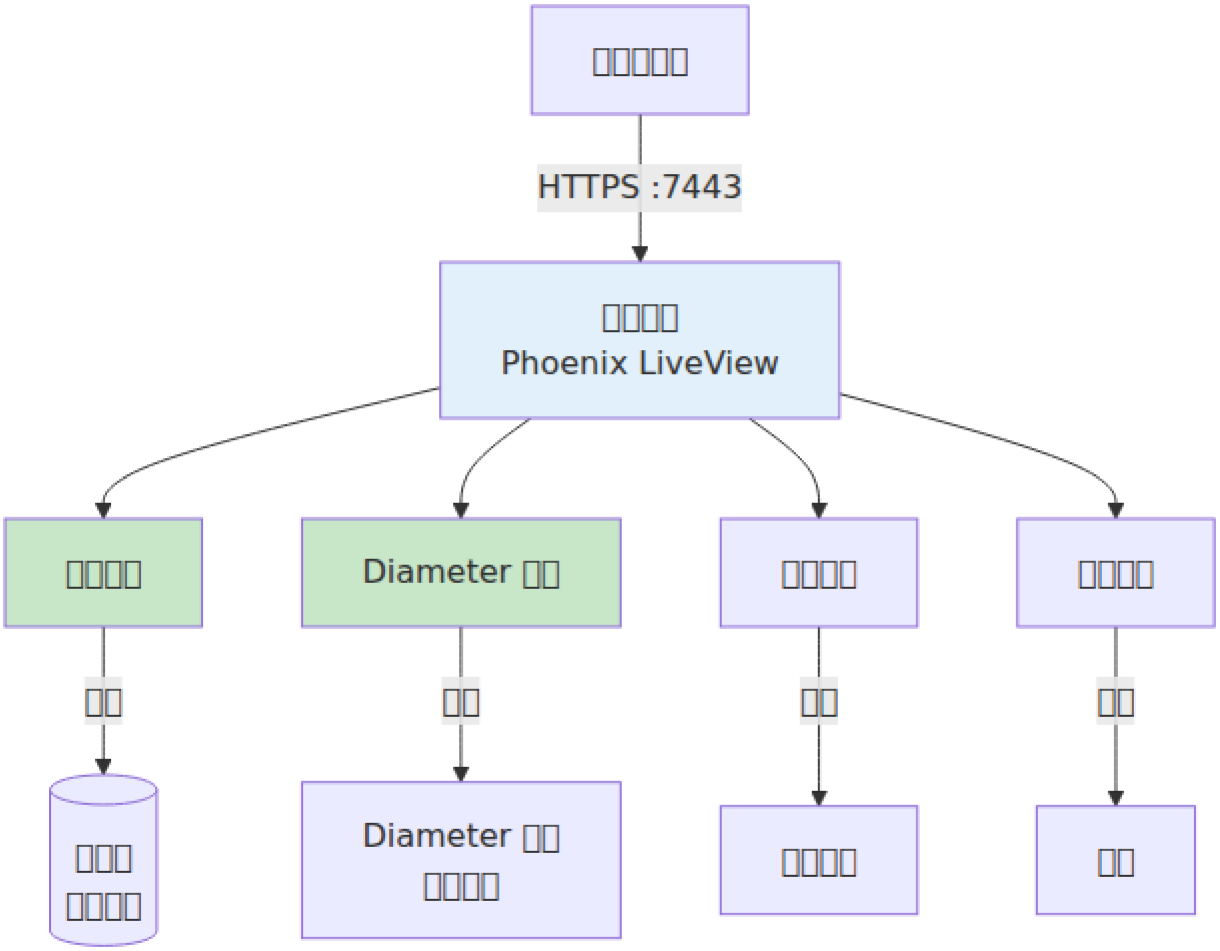
主要特性

- 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
- **Diameter** 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: 仅限 HTTPS
Port: 7443□可配置)
Certificate: 在 config/config.exs 中配置
```

控制面板架构



访问控制面板

初始访问

1. 打开网页浏览器

2. 导航到 `https://[hostname]:7443`
3. 接受 TLS 证书（如果□自签名）
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□正常的。

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** □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页技术（LiveView, WebSockets）：

- Chrome/Chromium（推荐）
- Firefox
- Safari
- Edge

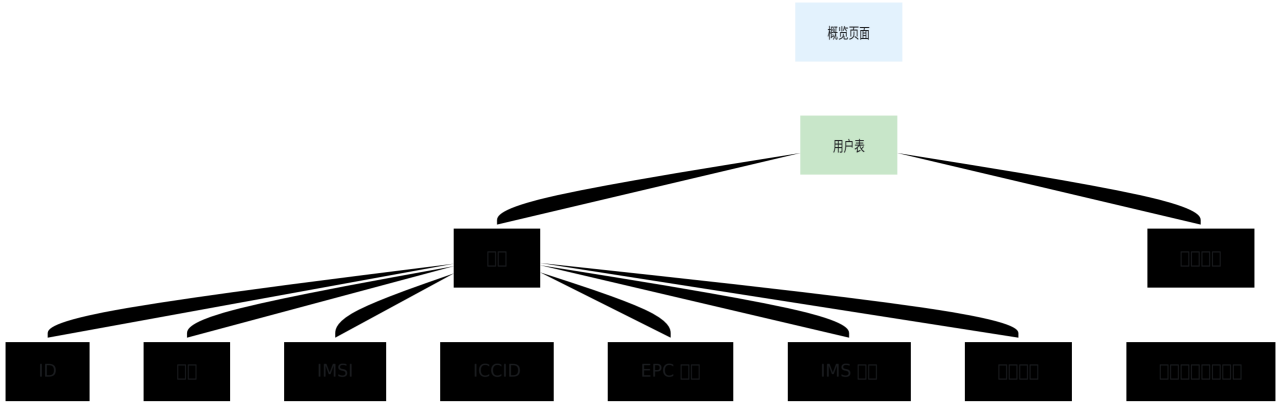
注意：不支持 Internet Explorer。

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



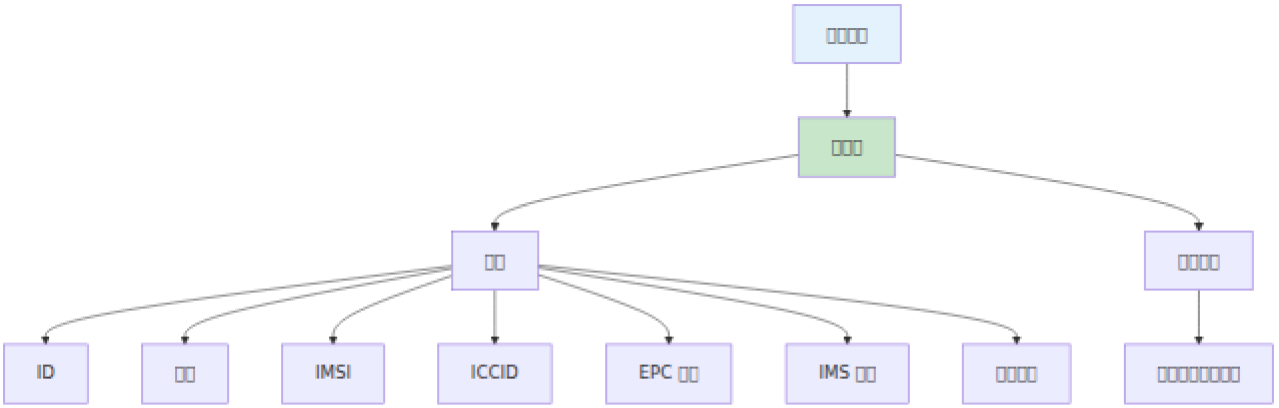
表列

列	描述	值
ID	用户数据库 ID	整数
启用	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 配置	数据服务配置名称	配置名称或 ID
IMS 配置	语音服务配置名称	配置名称、ID 或 "N/A"
漫游配置	漫游策略名称	配置名称、ID 或 "N/A"

可展开行详细信息

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
- 小区 **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 Diameter 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 Diameter 消息的时间戳

IMS 信息

字段：

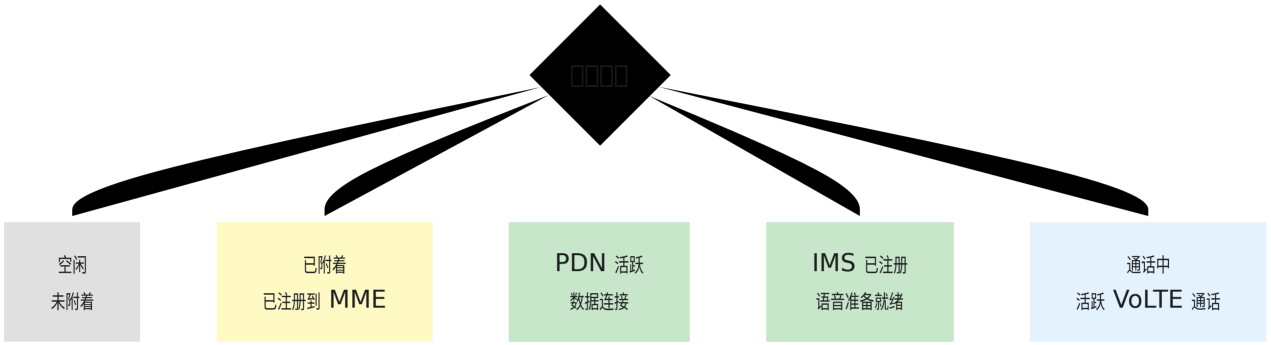
- 分配的 **S-CSCF** - 当前分配的 S-CSCF SIP URI
- **IMS** 公共身份 - SIP URI（例如，sip:+14155551234@ims.example.com）
- 最后看到的 **P-CSCF** - 最后联系 HSS 的 P-CSCF
- 最后看到的 **I-CSCF** - 最后联系 HSS 的 I-CSCF

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：

- 空闲：没有位置信息，没有 MME
- 已附着：最后看到的 MME 存在，位置信息可用
- **PDN** 活跃： PDN 会话计数 > 0
- **IMS** 已注册：分配的 S-CSCF 存在
- 通话中：活跃通话计数 > 0

自动刷新

概览页面每 **1** 秒自动刷新一次，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

使用案例

1. 监控活跃用户
    - 查看当前附着的用户

- 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□否启用
- 检查最后看到的时间戳（用户□否响应？）
- 确认配置分配
- 查看当前位置信息

3. 容量监控

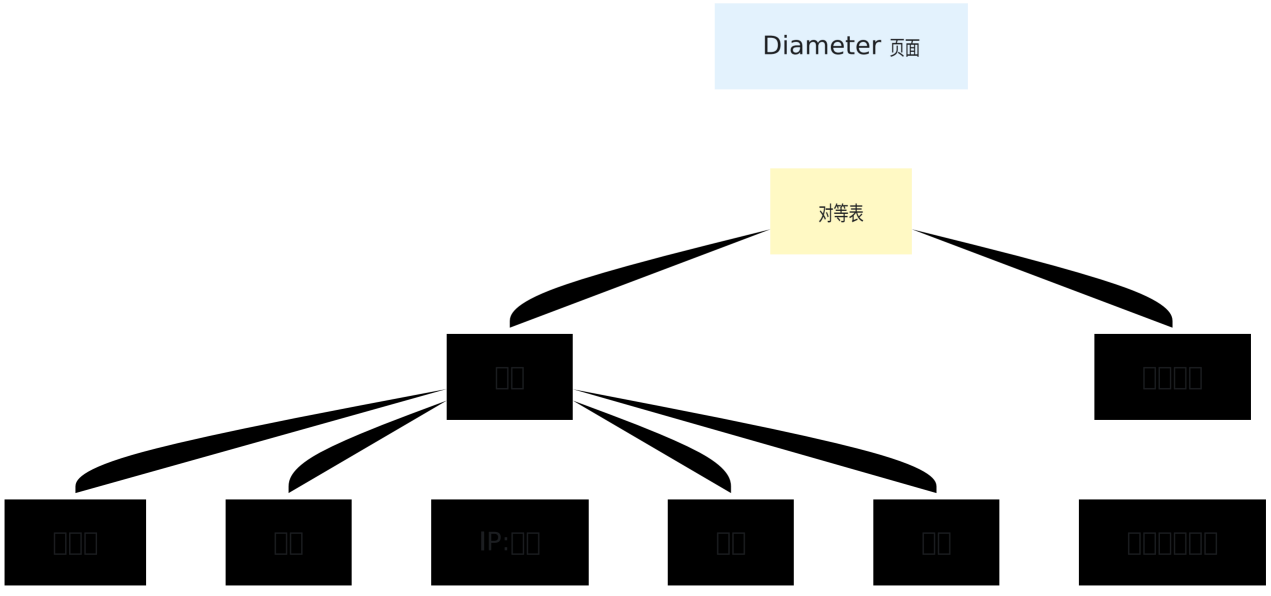
- 计算总附着用户数
- 监控 PDN 会话计数
- 跟踪活跃 VoLTE 通话

Diameter 页面

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

Diameter 页面显示所有 Diameter 对等连接的实时状态。

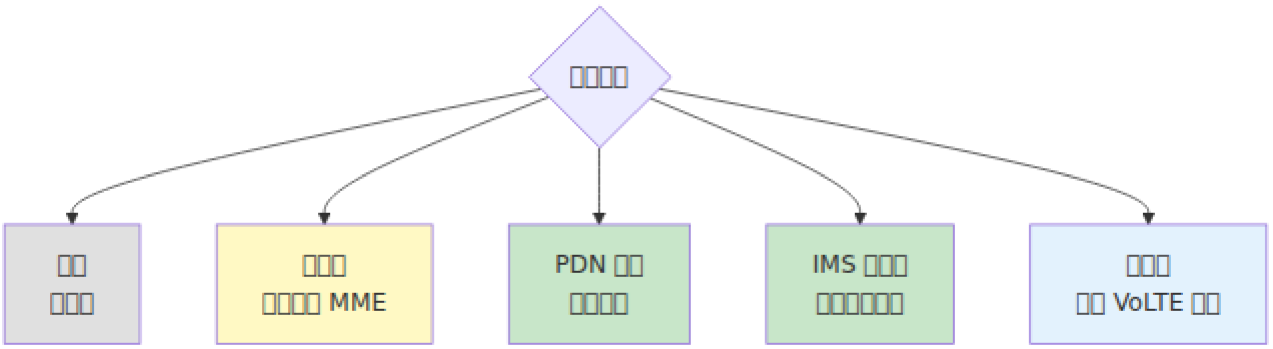
页面布局



表列

列	描述	值
主机名	Diameter 对等主机名	FQDN
领域	Diameter 领域	域名
IP:端口	网络地址	IP 地址和端口
传输	传输协议	TCP 或 SCTP
状态	连接状态	已连接 / 未连接

连接状态



可展开行详细信息

单击任何对等以查看附加信息：

连接信息：

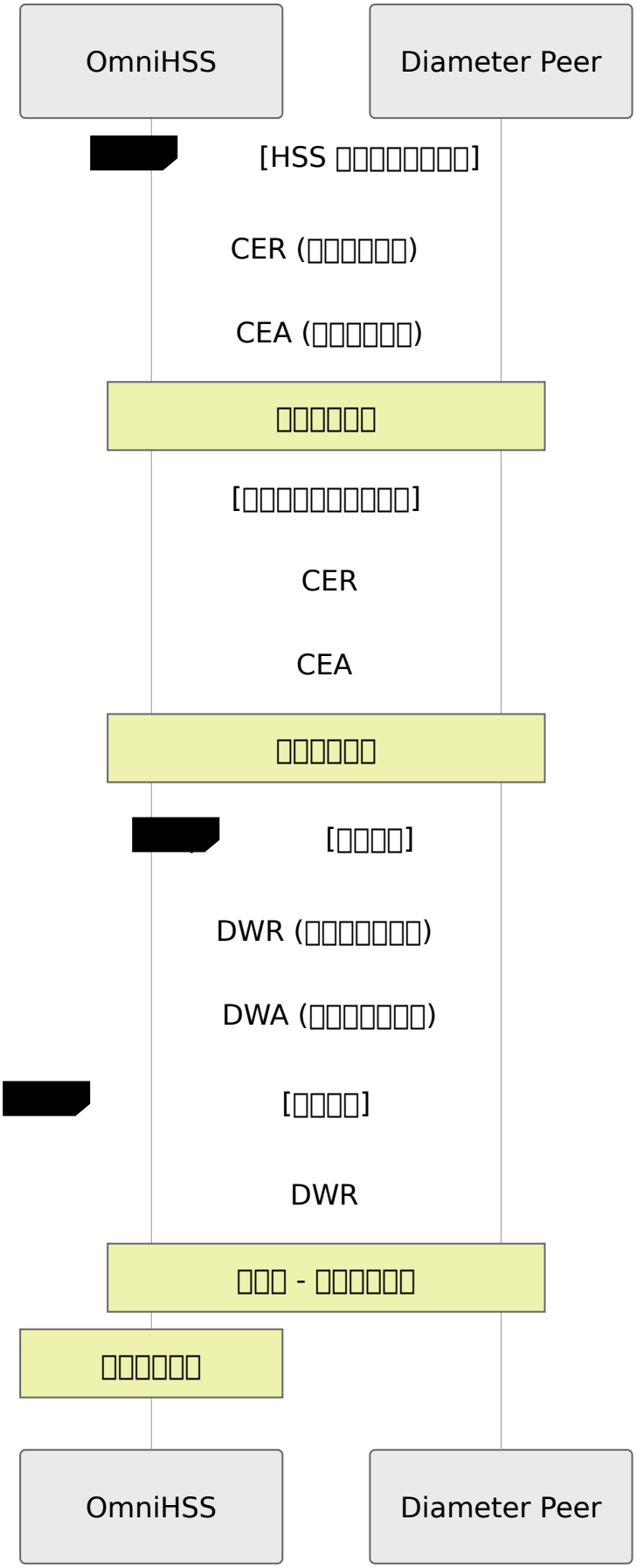
- 连接类型 - 由 HSS 或对等发起
- 产品名称 - 对等的产品标识
- 应用 **ID** - 支持的 Diameter 应用

应用 **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，如果外部)

[redacted]


OmniHSS
Diameter Peer
[redacted] [HSS [redacted]]
CER ([redacted])
CEA ([redacted])
[redacted]
[[redacted]]
CER
CEA
[redacted]
[redacted] [[redacted]]
DWR ([redacted])
DWA ([redacted])
[redacted] [[redacted]]
DWR
[redacted] - [redacted]
[redacted]
OmniHSS
Diameter Peer

自动刷新

Diameter 页面每 **1** 秒自动刷新。

使用案例

1. 验证连接性
    - 确保所有预期的对等已连接
    - 立即识别未连接的对等
    - 监控连接波动
2. 故障排除
    - 检查对等□否可达
    - 验证传输协议（TCP 与 SCTP）
    - 确认应用 ID □否符合预期
    - 确定□哪个端发起了连接
3. 容量规划
    - 计算总连接的对等数
    - 监控连接稳定性
    - 规划额外的对等容量

常见问题

对等显示未连接

可能原因：

1. 网络连接问题
2. 对等处于关闭或重启状态
3. 防火墙阻止流量
4. Diameter 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□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. 验证对等的 Diameter 配置与 HSS 匹配

对等反复连接和断开

可能原因：

1. 网络不稳定
2. 保持连接超时不匹配
3. 对等资源问题
4. Diameter 应用不匹配

故障排除步骤：

1. 检查网络稳定性
2. 查看双方的保持连接计时器
3. 检查对等系统资源
4. 验证双方的应用 ID □否匹配

应用页面

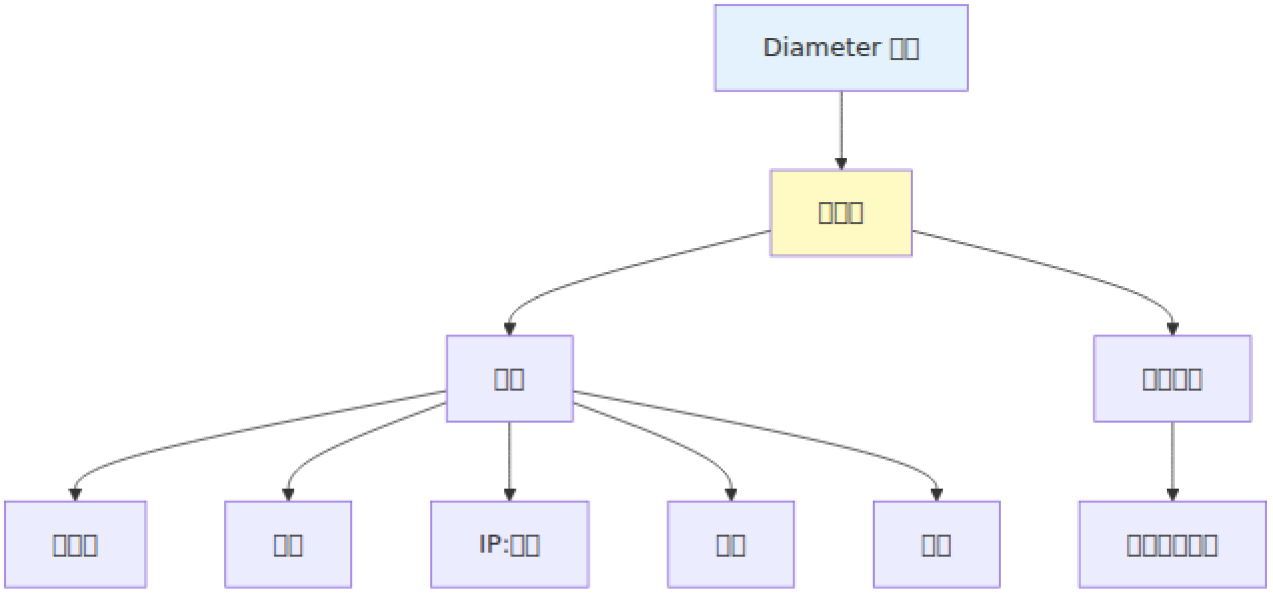
URL: `https://[hostname]:7443/application`

应用页面提供系统级监控和资源使用信息。

特性

- 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



使用案例

1. 健康监控
    - 验证应用□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□否充足
3. 故障排除
    - 识别资源耗尽
    - 检查□否需要重启
    - 验证 Erlang VM 版本

配置页面

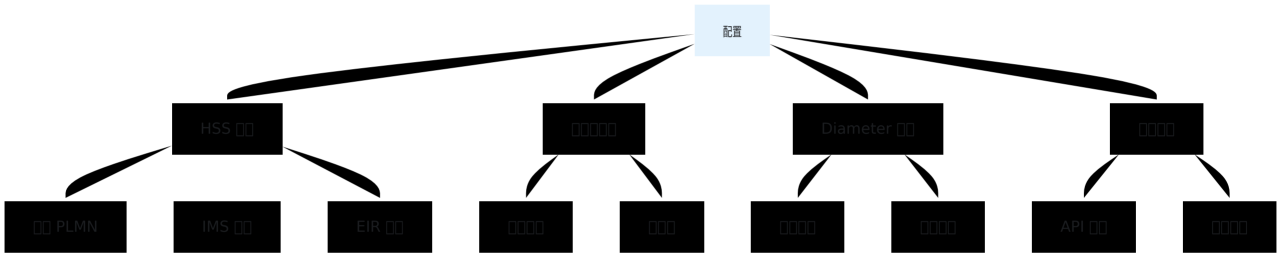
URL: `https://[hostname]:7443/configuration`

配置页面显示 OmniHSS 当前的运行时配置。

特性

- 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



使用案例

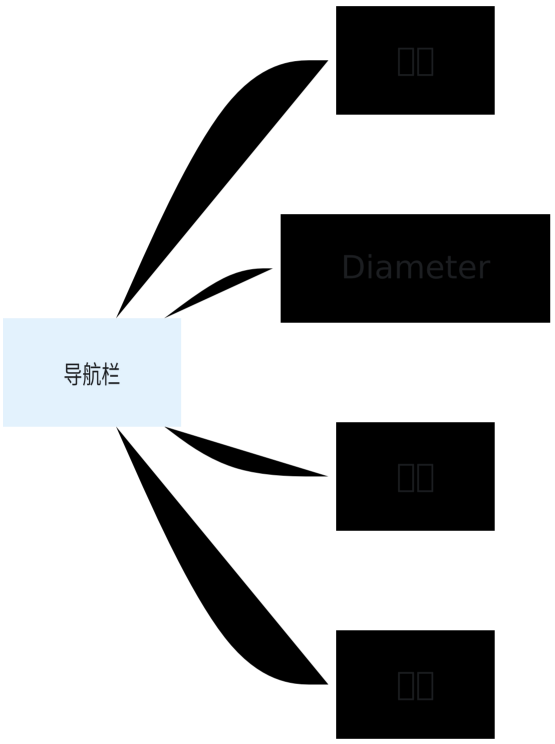
1. 配置验证
    - 验证 runtime.exs 设置□否已应用
    - 确认数据库连接参数
    - 检查 Diameter 对等配置
2. 故障排除
    - 识别配置错误
    - 验证环境变量□否正确设置
    - 比较预期与实际配置
3. 文档
    - 导出当前配置以供文档使用

- 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

虽然控制面板没有实现自定义键盘快捷键，但标准浏览器快捷键有效：

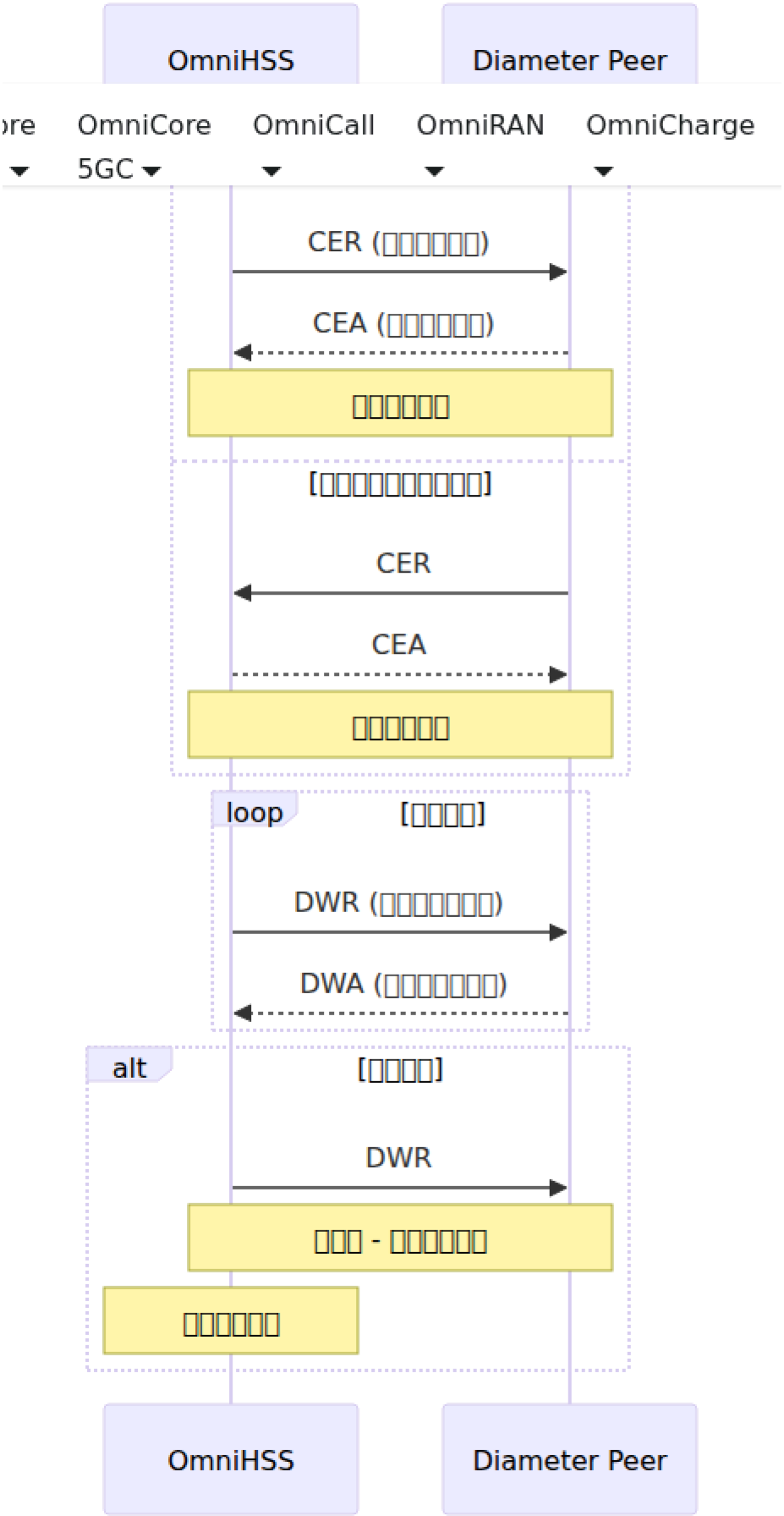
- **Ctrl+R / F5** - 手动页面刷新（尽管自动刷新使其不必要）
- **Ctrl+F** - 在页面上搜索
- **Ctrl+T** - 打开新标签页（用于多个页面）

□□□□□

□□□□

- 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


OmniHSS
Diameter Peer
ore
OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
CER (□□□□□□□)
CEA (□□□□□□□)
□□□□□□□
[□□□□□□□□□□□□]
CER
CEA
□□□□□□□
loop
[□□□□□]
DWR (□□□□□□□□)
DWA (□□□□□□□□)
alt
[□□□□□]
DWR
□□□ - □□□□□□□
□□□□□□□
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
未连接的 Diameter 对等	1 对等	2+ 对等或关键对等
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 →

OmniHSS 实体关系

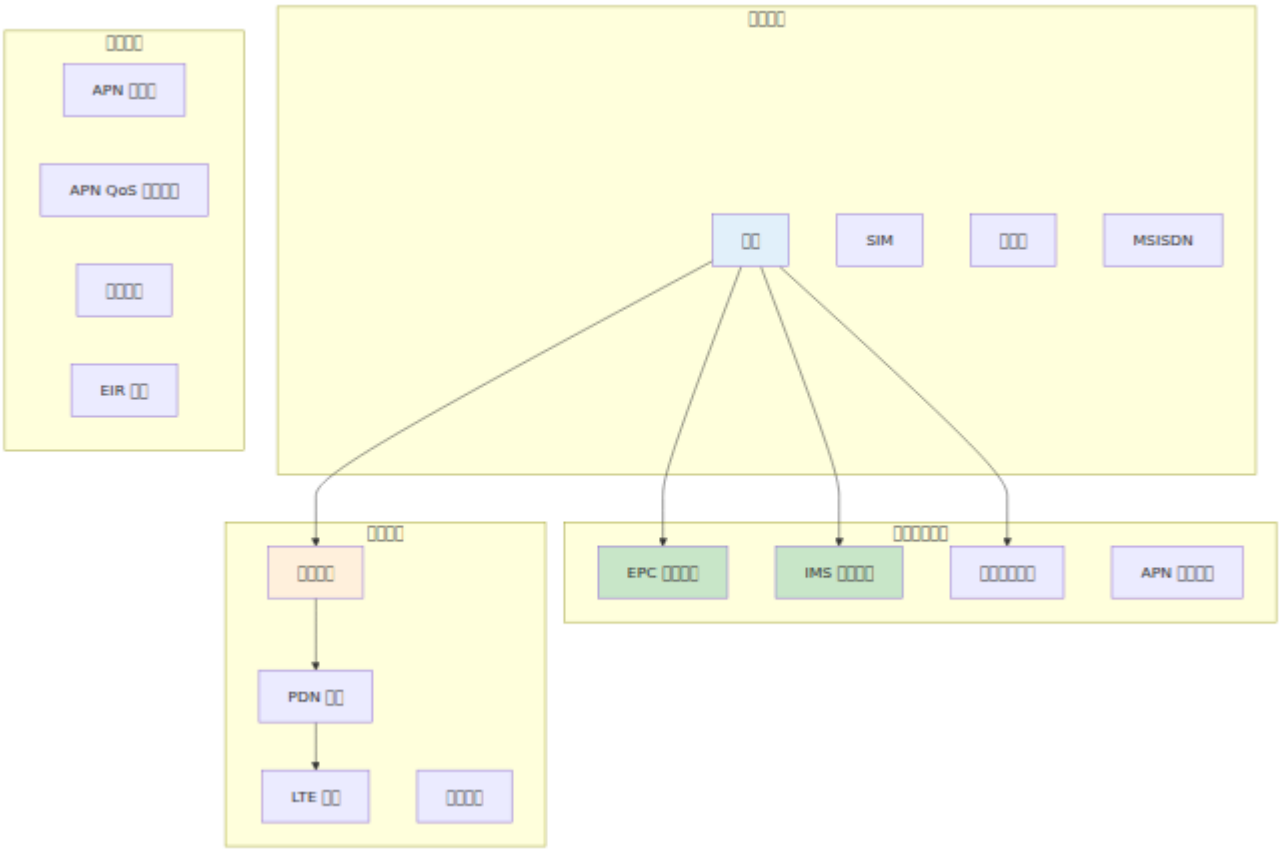
← 返回操作指南

目录

实体概述

OmniHSS 将用户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。

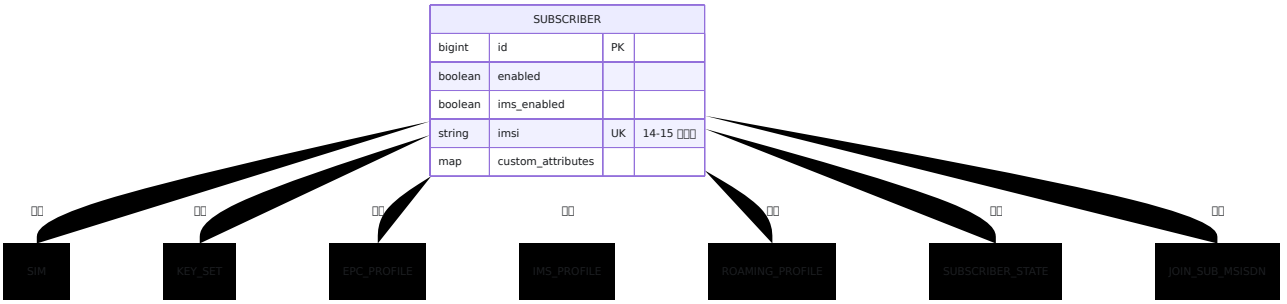
□□□□


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

□□□□

□□

□□□□□□□□□□□□□□□□□□


SUBSCRIBER
bigint 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

□□:

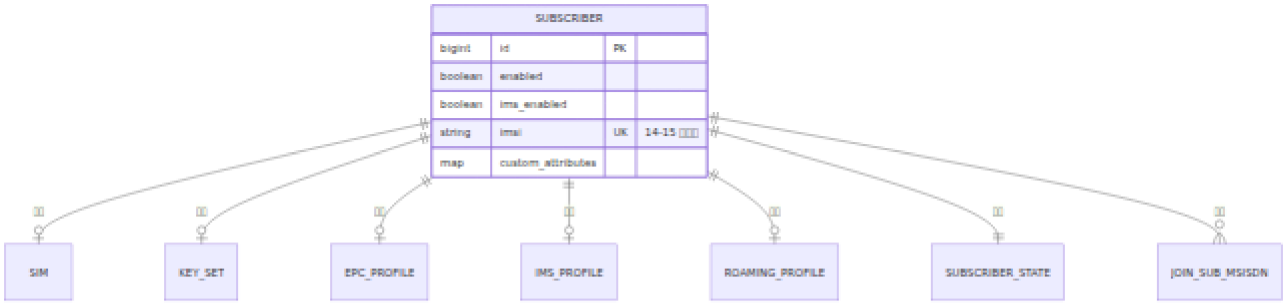
字段	类型	描述	约束
`id`	bigint	主键	自动递增
`enabled`	boolean	服务启用标志	默认: true
`ims_enabled`	boolean	IMS 服务启用	默认: true
`imsi`	string	国际移动用户身份	14-15 位数字，唯一
`custom_attributes`	map	自定义键值数据	可选
`sim_id`	bigint	外键指向 SIM	可选
`key_set_id`	bigint	外键指向密钥集	必需
`epc_profile_id`	bigint	外键指向 EPC 配置文件	必需
`ims_profile_id`	bigint	外键指向 IMS 配置文件	可选
`roaming_profile_id`	bigint	外键指向漫游配置文件	可选
`subscriber_state_id`	bigint	外键指向用户状态	自动创建

关键点：

- 每个用户必须有且仅有一个 IMSI
- IMSI 必须为 14-15 位数字（不能包含字母或特殊字符）
- 一个用户可以拥有多个 MSISDN（电话号码）
- 用户状态在用户创建时自动创建
- `enabled` 标志控制所有服务（数据和 IMS）
- `ims_enabled` 标志仅控制 IMS 服务

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	制造批次	公开
`is_esim`	boolean	嵌入式 SIM 标志	公开
`pin1`, `pin2`	string	PIN 码	敏感
`puk1`, `puk2`	string	PUK 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用户
- PIN/PUK 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（空中下载）更新

密钥集

密钥集包含用于身份验证的加密密钥。

KEY_SET			
bigint	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[

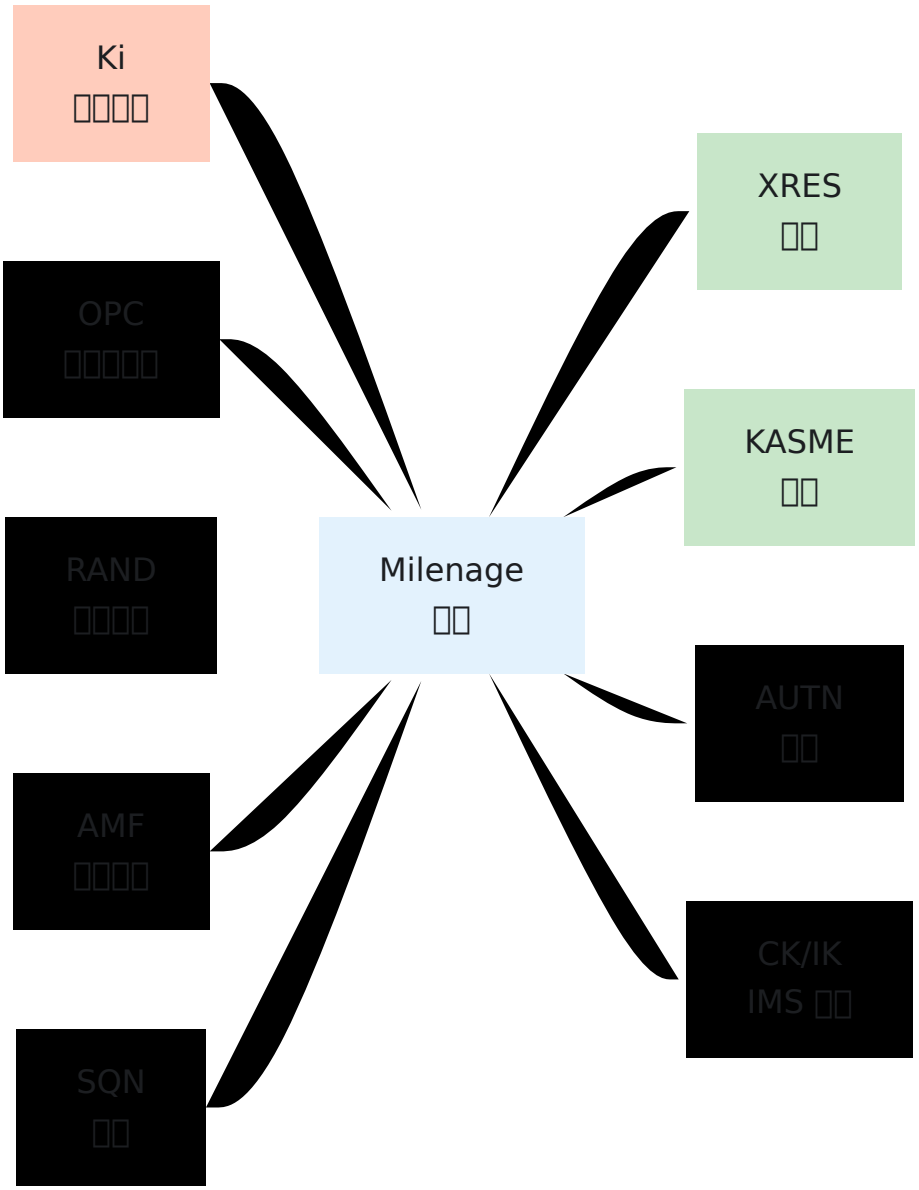
字段：

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：

- 多个用户可以共享同一个密钥集
- Ki 与 SIM 共享的主密钥
- 必须提供 OPC 或 OP 之一（OPC 可以从 OP 派生）
- SQN 在每次身份验证时递增
- Milenage 目前唯一支持的算法

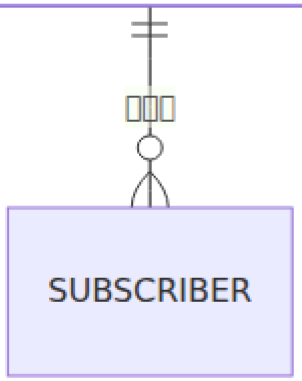
身份验证算法：



MSISDN

MSISDN □□□□□□□□□□

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



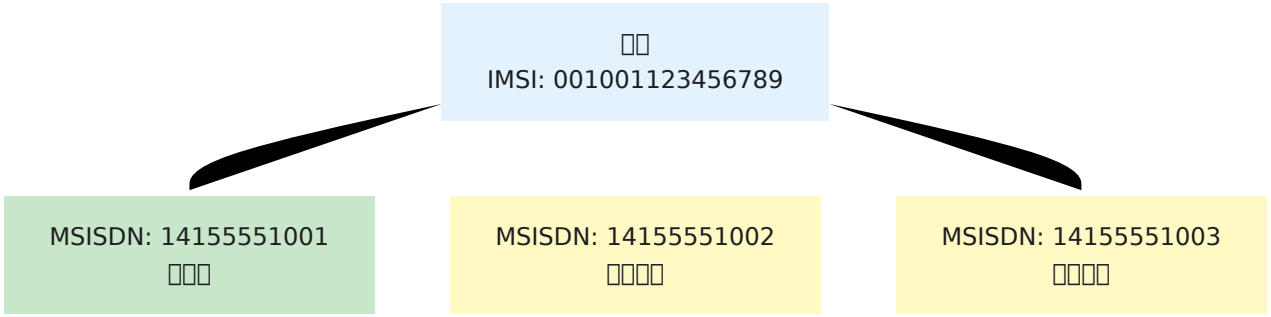
字段：

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

关键点：

- MSISDN □国际格式的电话号码
- 多个 MSISDN 可以分配给一个用户
- 一个 MSISDN 不能在多个用户之间共享
- 格式：国家代码 + 国内号码（例如，"+1 415-555-1234" 的格式为 "14155551234"）

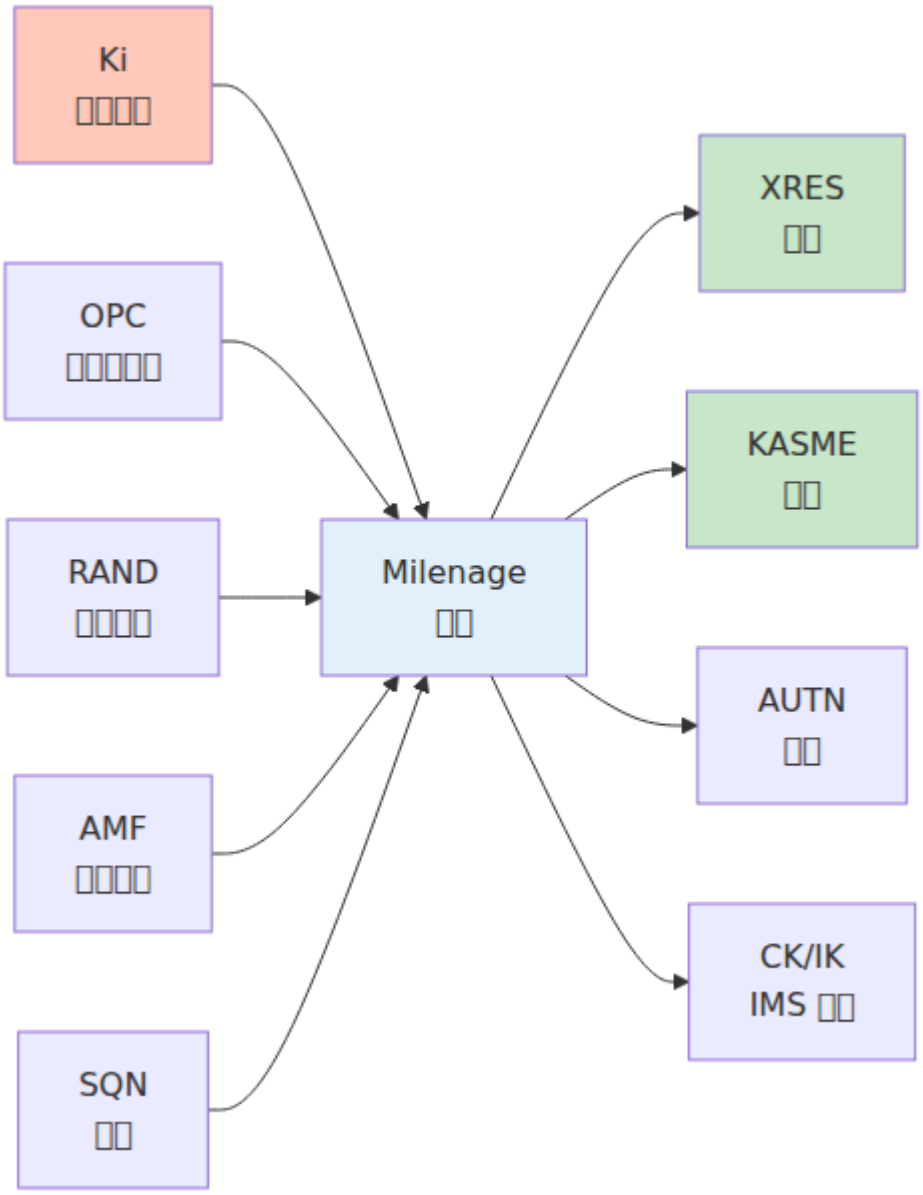
多 **MSISDN** 模式：


□□
IMSI: 001001123456789
MSISDN: 14155551001
□□□
MSISDN: 14155551002
□□□□
MSISDN: 14155551003
□□□□

□□□□□□

EPC □□□□

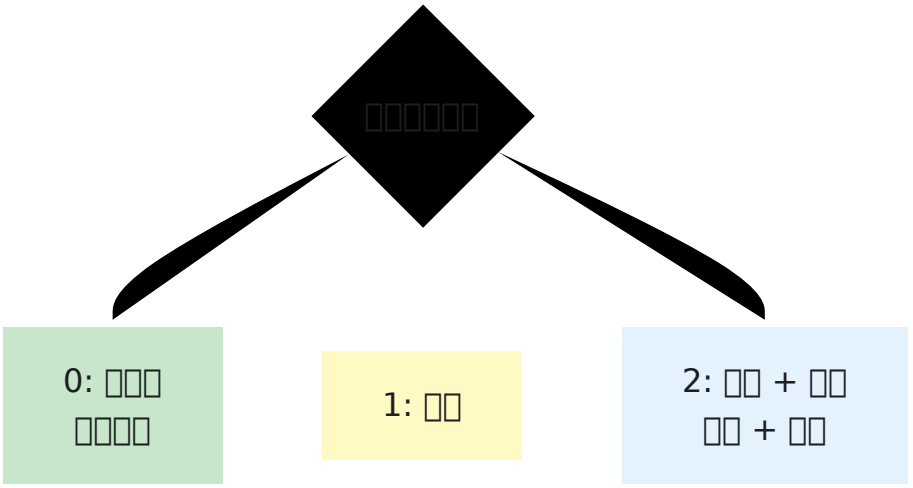
EPC □□□□□□□ LTE □□□□□□□□□


Ki
□□□□
OPC
□□□□□
RAND
□□□□
AMF
□□□□
SQN
□□
Milenage
□□
XRES
□□
KASME
□□
AUTN
□□
CK/IK
IMS □□

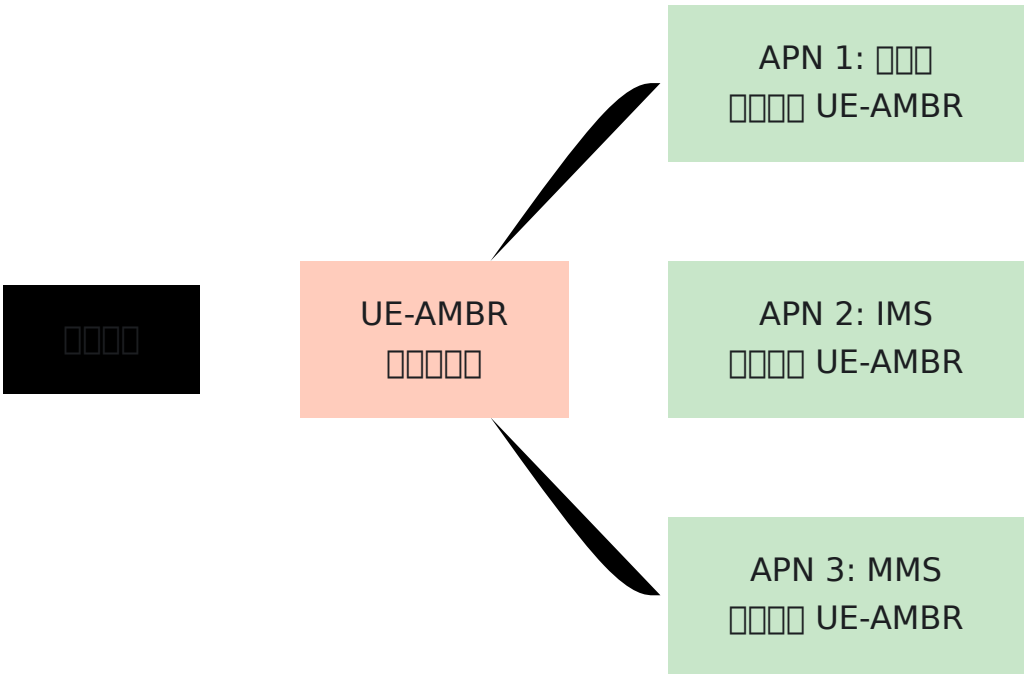
□□:

□□	□□	□□	□□
`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 □□	□

□□□□□□□:

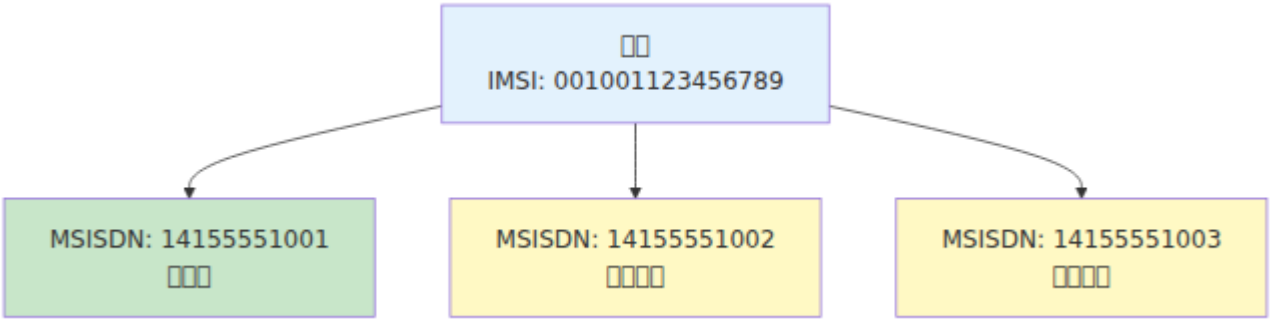


AMBR□□□□□□□□□□:



IMS

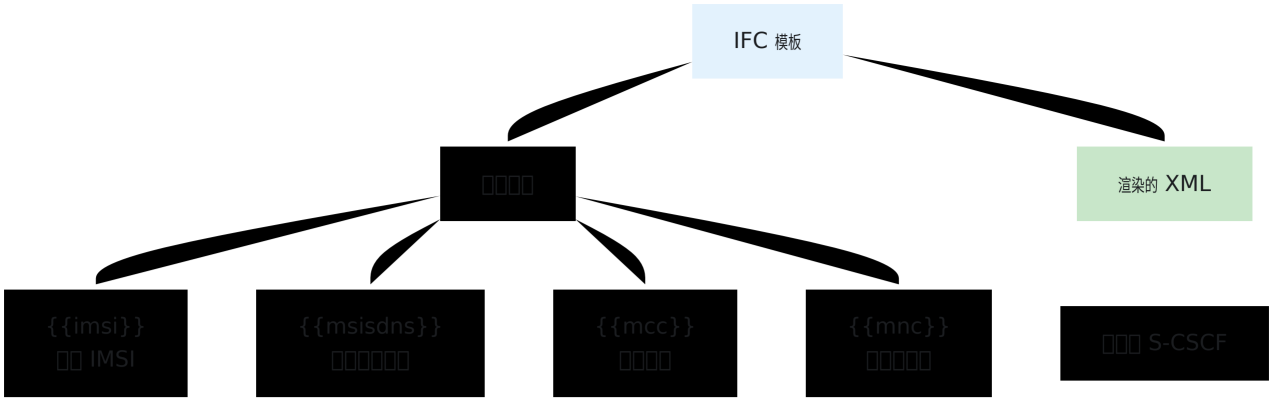
IMS



:

`name`	string		
`ifc_template`	text	XML	XML

IFC :

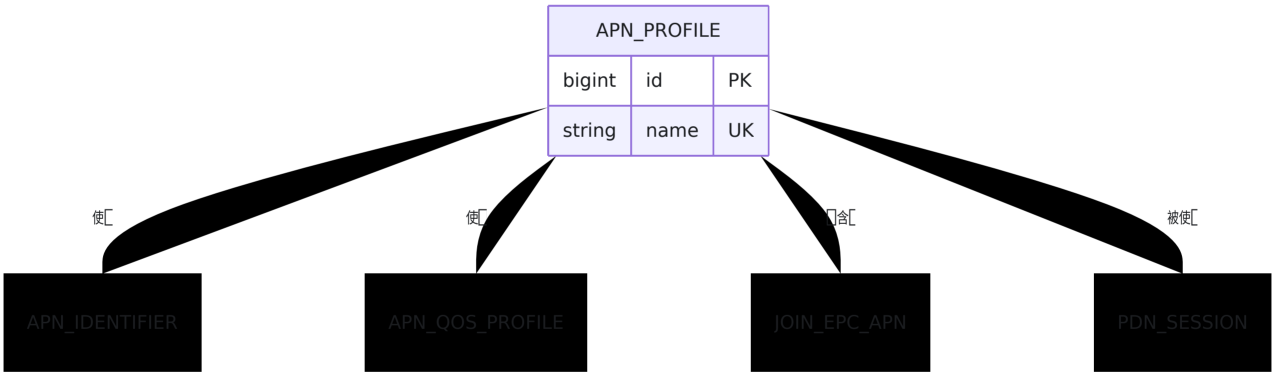


关键点：

- IFC（初始过滤标准）控制 IMS 中的呼叫路由
- 当用户注册时模板被渲染
- 变量用实际用户数据替换
- 在 IMS 注册期间发送到 S-CSCF

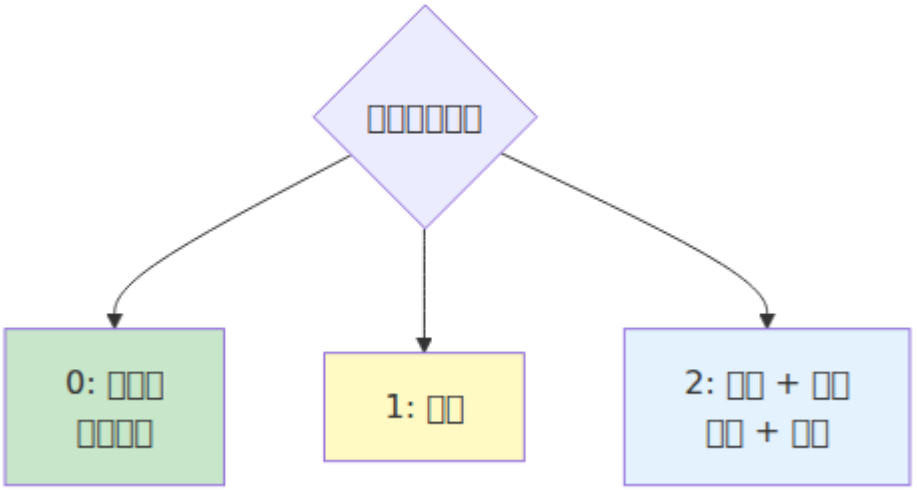
APN 配置文件

APN 配置文件定义特定数据接入点的特性。



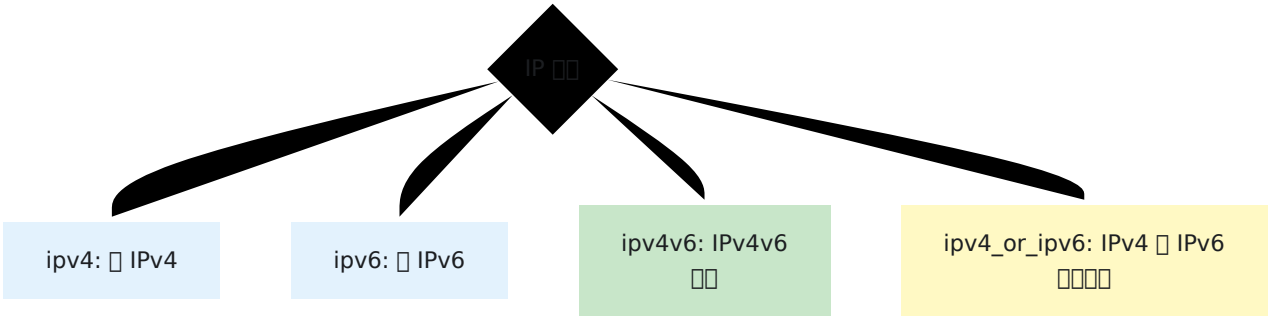
相关实体：

APN 标识符

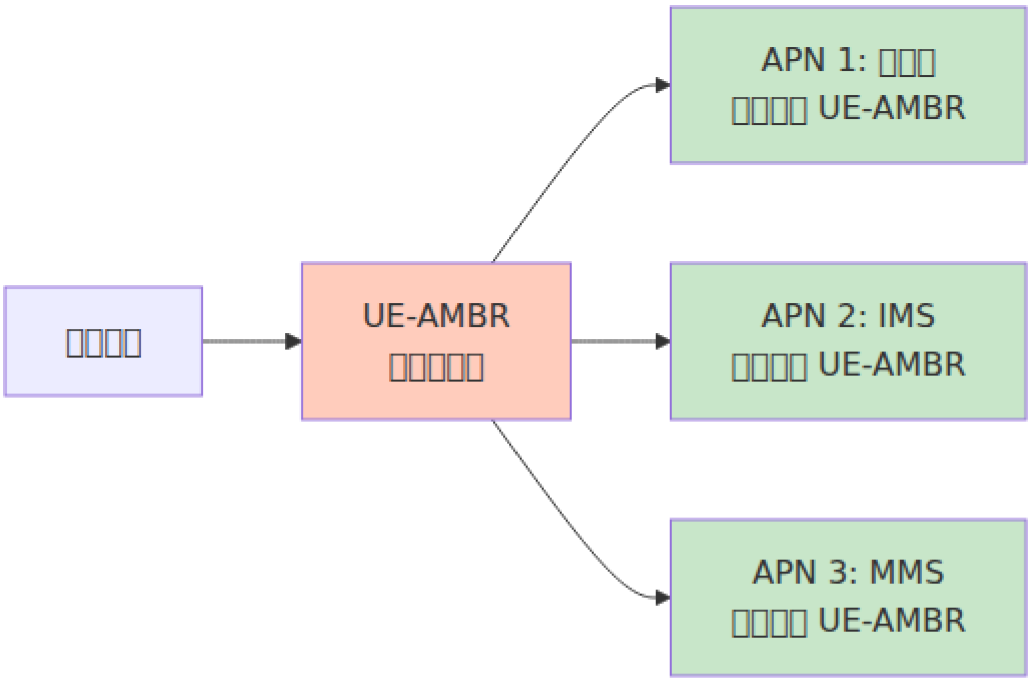


□□	□□	□□	□□
`apn`	string	APN □□	"internet", "ims", "mms"
`ip_version`	string	IP □□□□	□□□

IP □□□□:



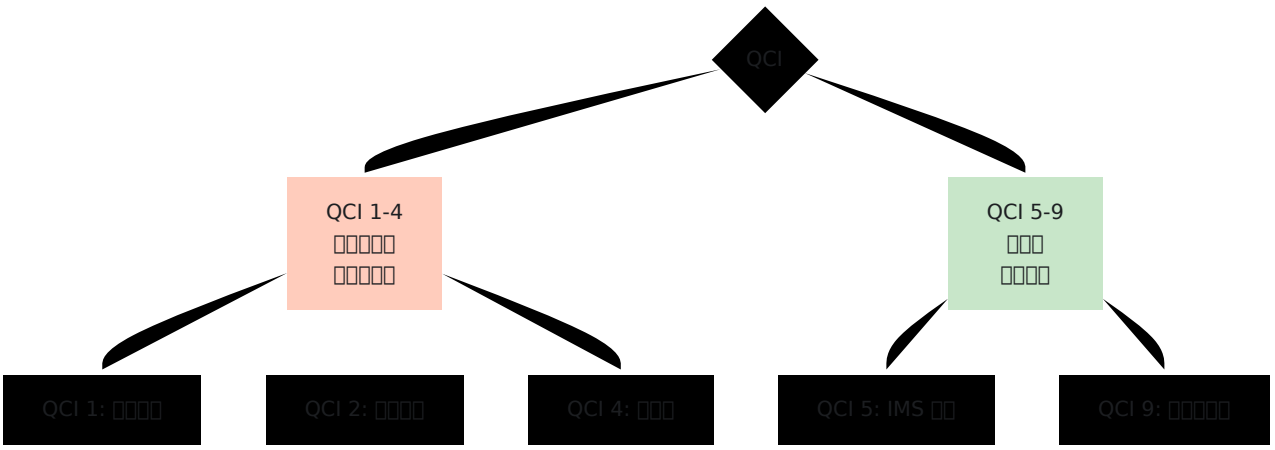
APN QoS □□□□



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

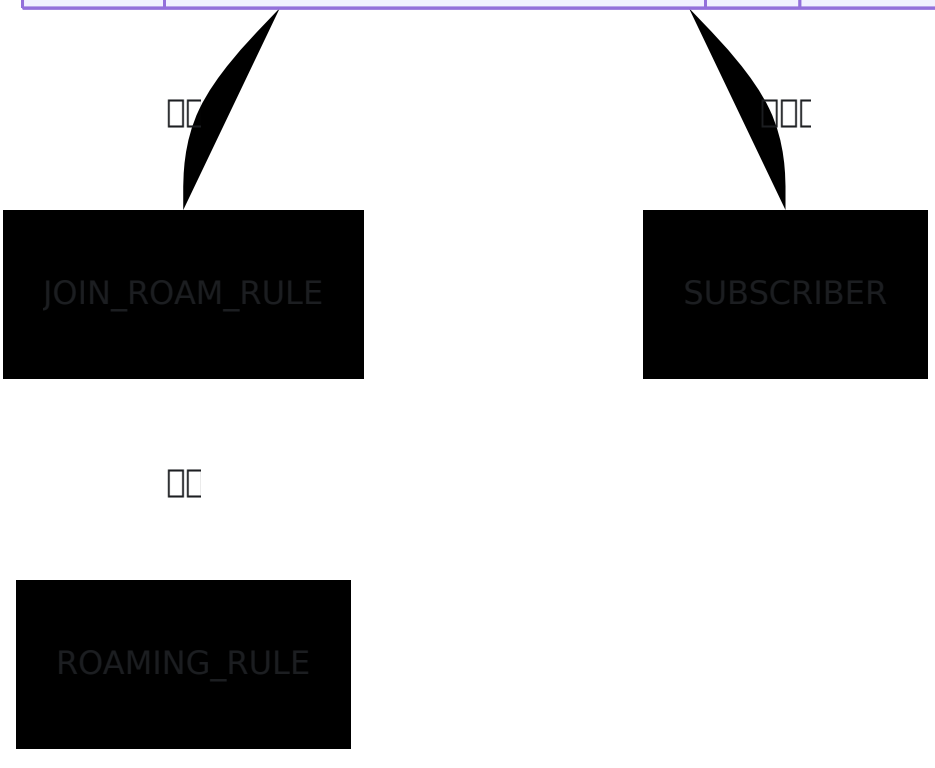
QCI 值**:**



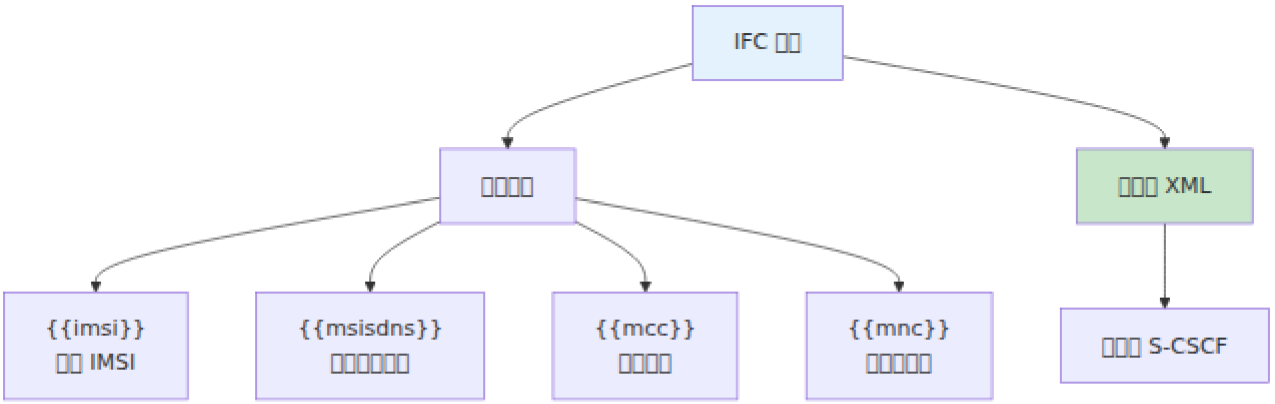
□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□

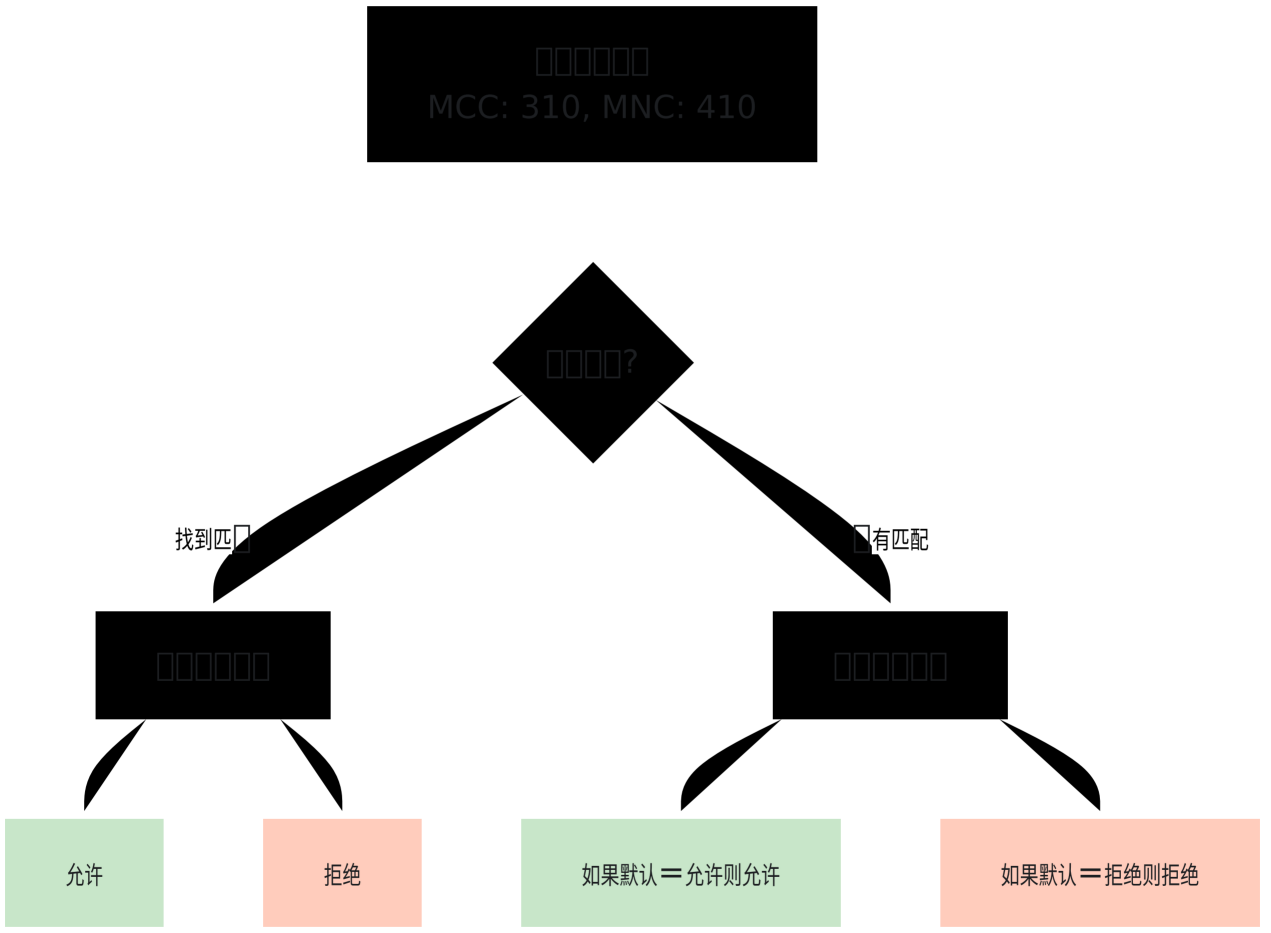
ROAMING_PROFILE			
bigint	id	PK	
string	name	UK	
string	data_action_if_no_rules_match		□□□□□
string	ims_action_if_no_rules_match		□□□□□



漫游规则：


IFC □□
□□□□
□□□ XML
{{imsi}}
□□ IMSI
{{msisdns}}
□□□□□□
{{mcc}}
□□□□
{{mnc}}
□□□□□
□□□ S-CSCF

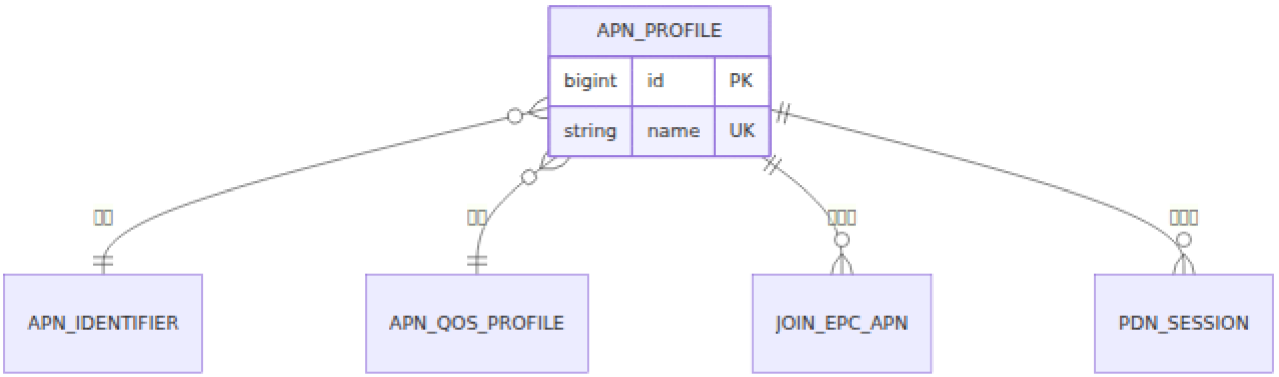
规则��估：


□□□□□□□
MCC: 310, MNC: 410
□□□□?
找到匹□
□有匹配
□□□□□□
□□□□□□
允许
拒绝
如果默认＝允许则允许
如果默认＝拒绝则拒绝

状态实体

用户状态

用户状态跟踪实时用户状态。



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` - 当前服务用户的 MME
- `last_seen_realm` - MME 的直径域
- `last_seen_rat_type` - 无线接入技术（LTE、5G 等）

IMS 信息：

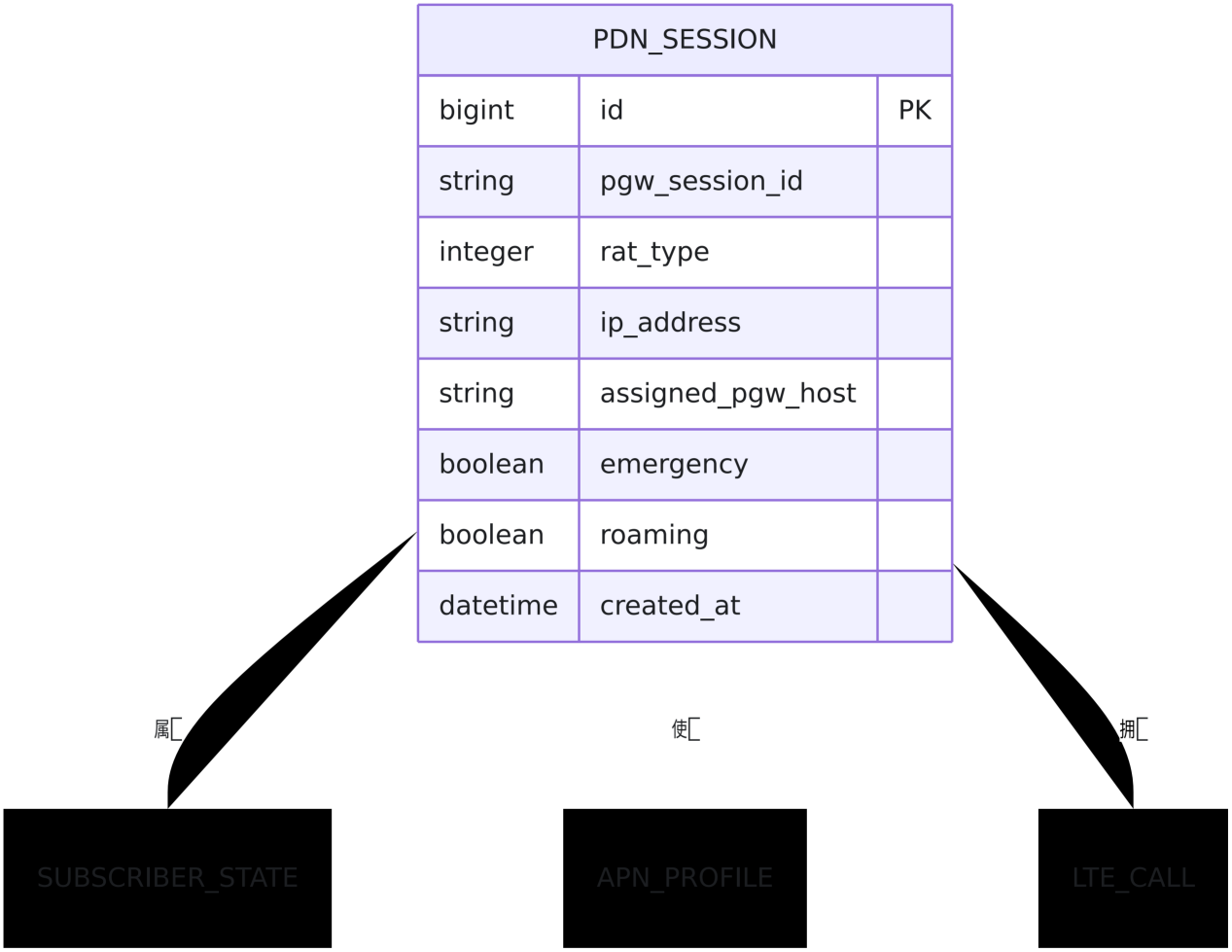
- `assigned_scscf` - 当前服务用户的 S-CSCF
- `ims_public_identity` - SIP URI（例如，sip:+14155551234@ims.example.com）
- `sh_repository_data` - 自定义 IMS 配置文件数据

时间戳：

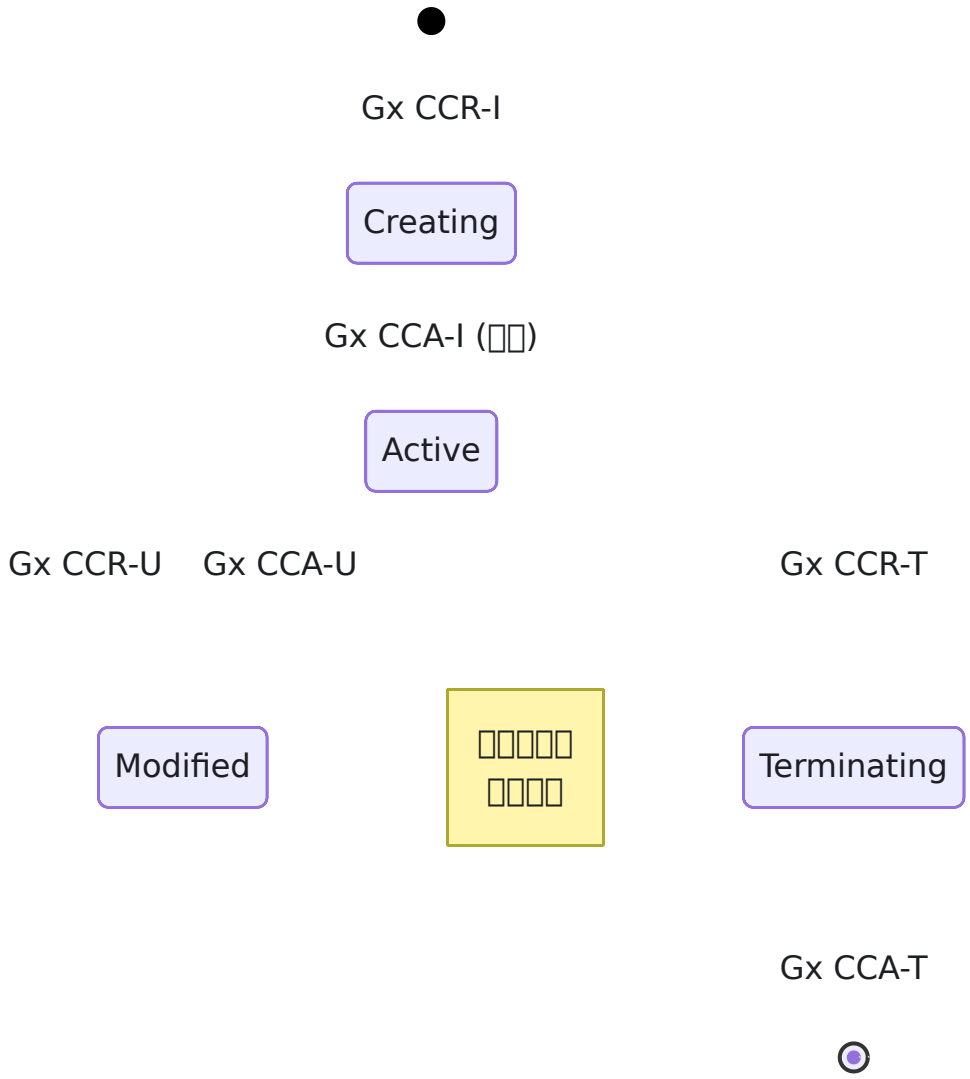
- `last_seen_at` - 最后接收到的直径消息
- 各种 `last_*_at` 时间戳用于不同的过程

PDN 会话

PDN 会话表示一个活跃的数据连接。

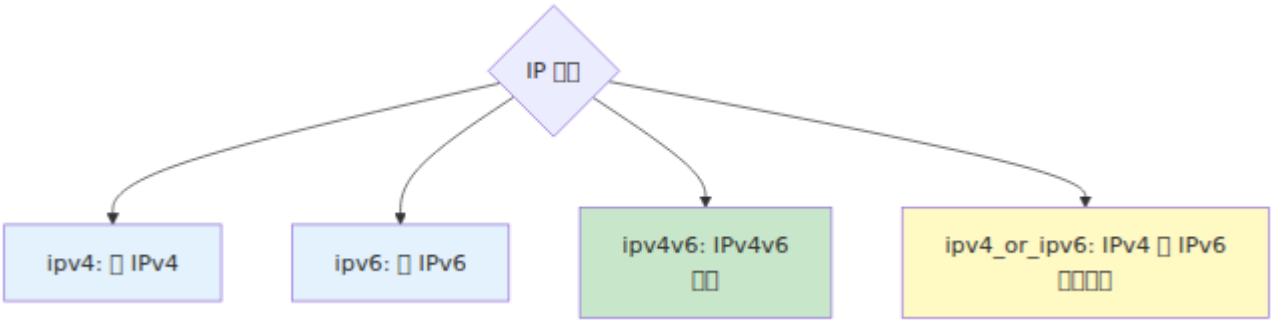


PDN 会话生命周期：

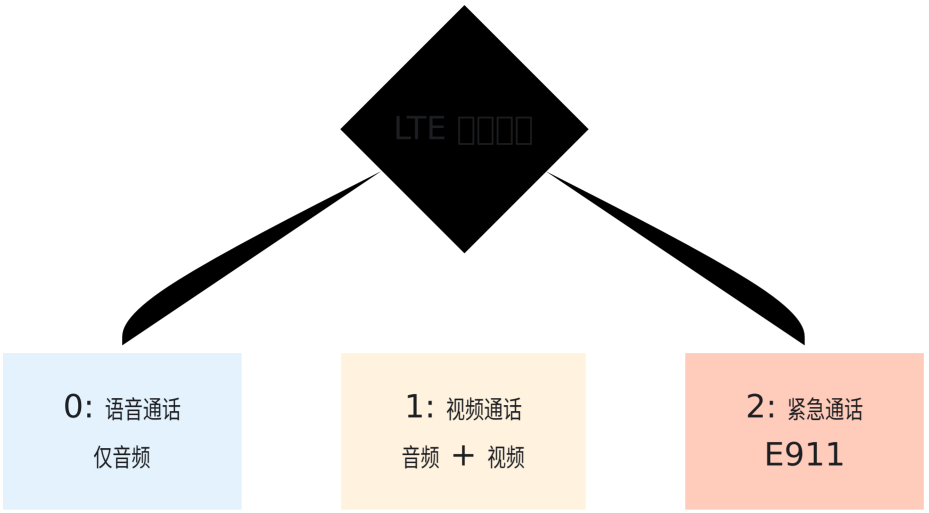


LTE □□

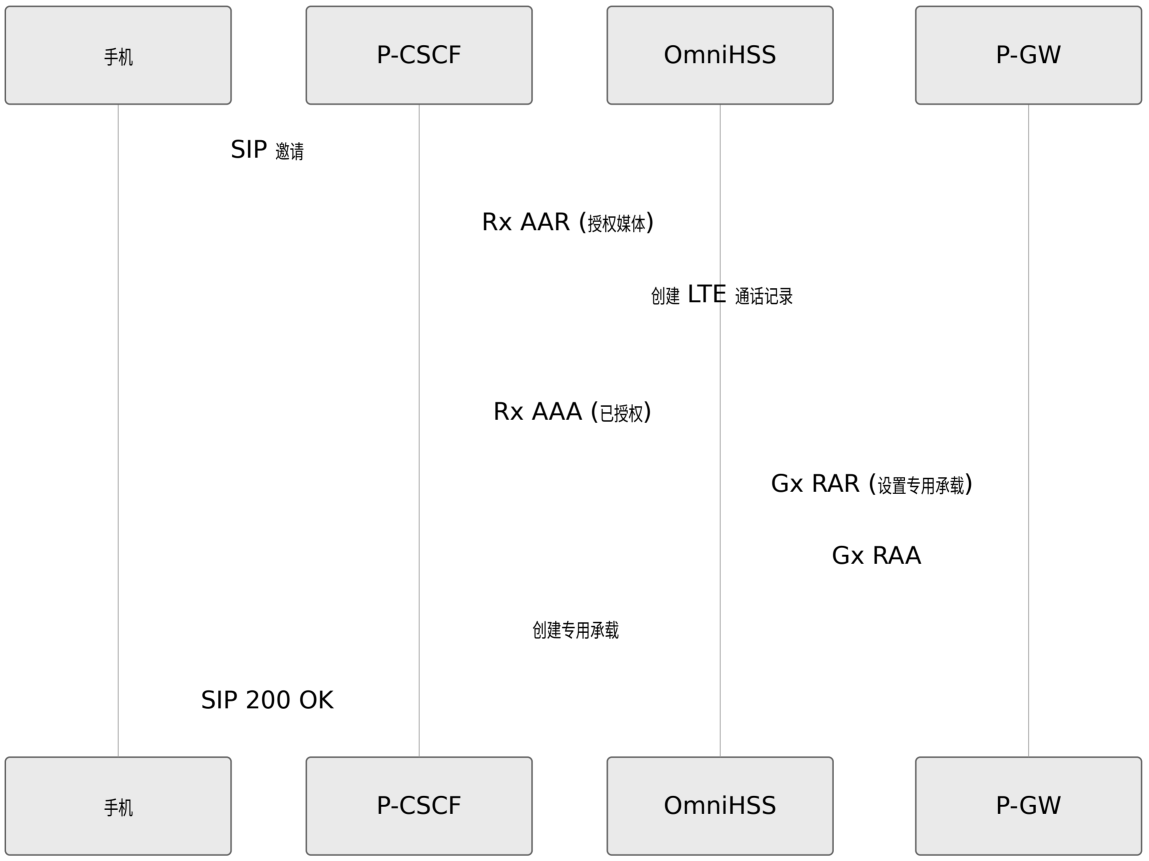
LTE □□□□□□□□□□ VoLTE □□/□□□□□



□□□□:

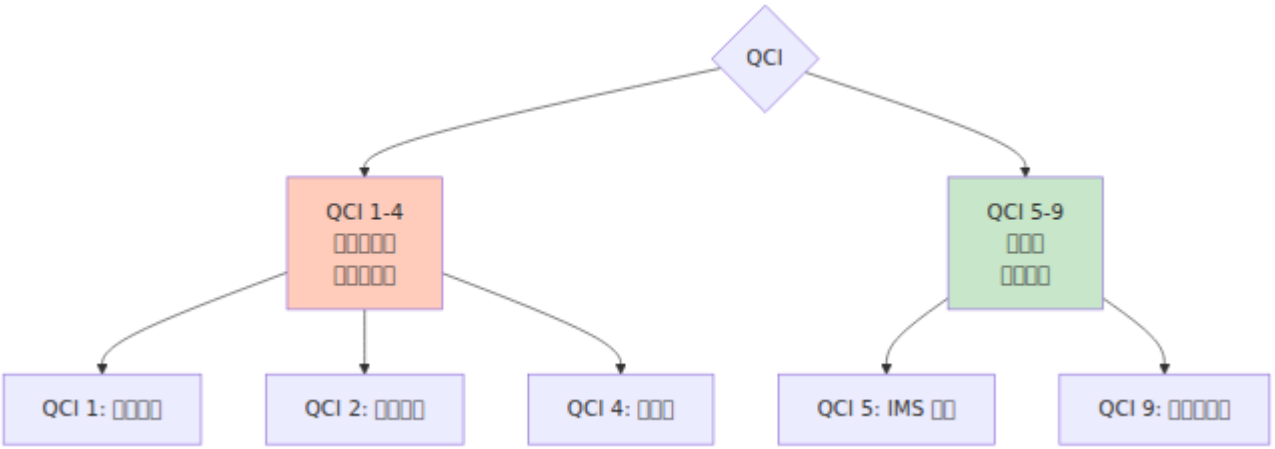

LTE □□□□
0: 语音通话
仅音频
1: 视频通话
音频 + 视频
2: 紧急通话
E911

VoLTE 通话流程：


手机
P-CSCF
OmniHSS
P-GW
SIP 邀请
Rx AAR (授权媒体)
创建 LTE 通话记录
Rx AAA (已授权)
Gx RAR (设置专用承载)
Gx RAA
创建专用承载
SIP 200 OK
手机
P-CSCF
OmniHSS
P-GW

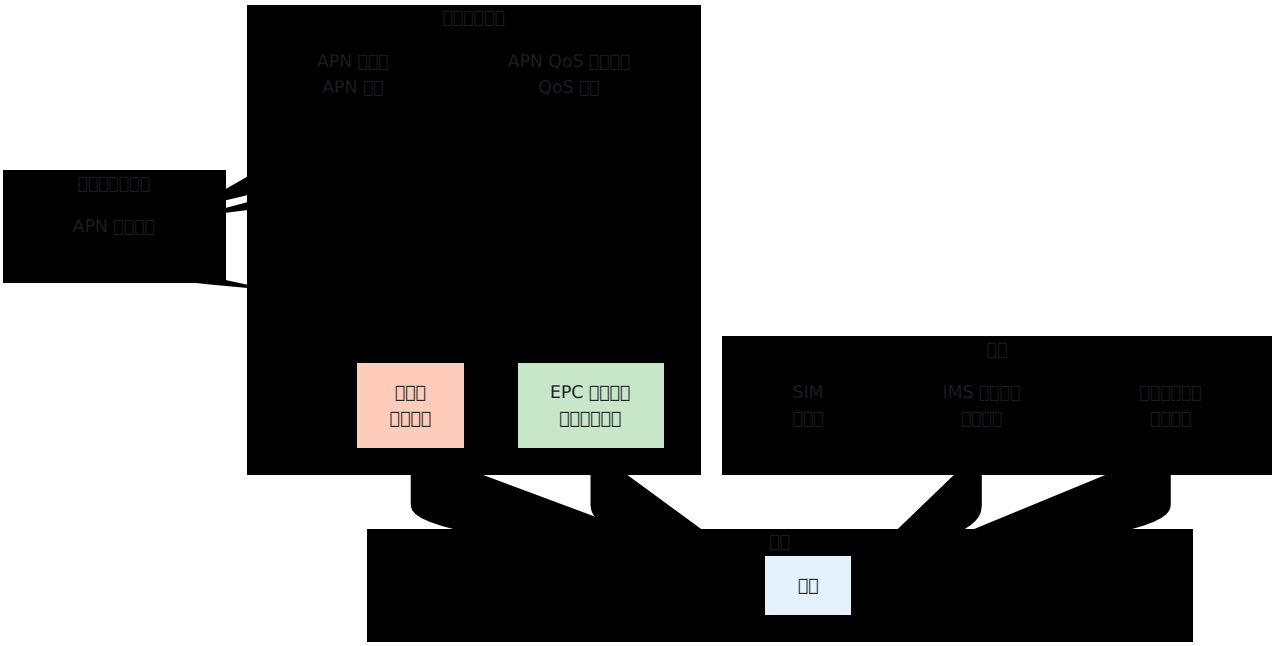
□□□□□

□□□□□□

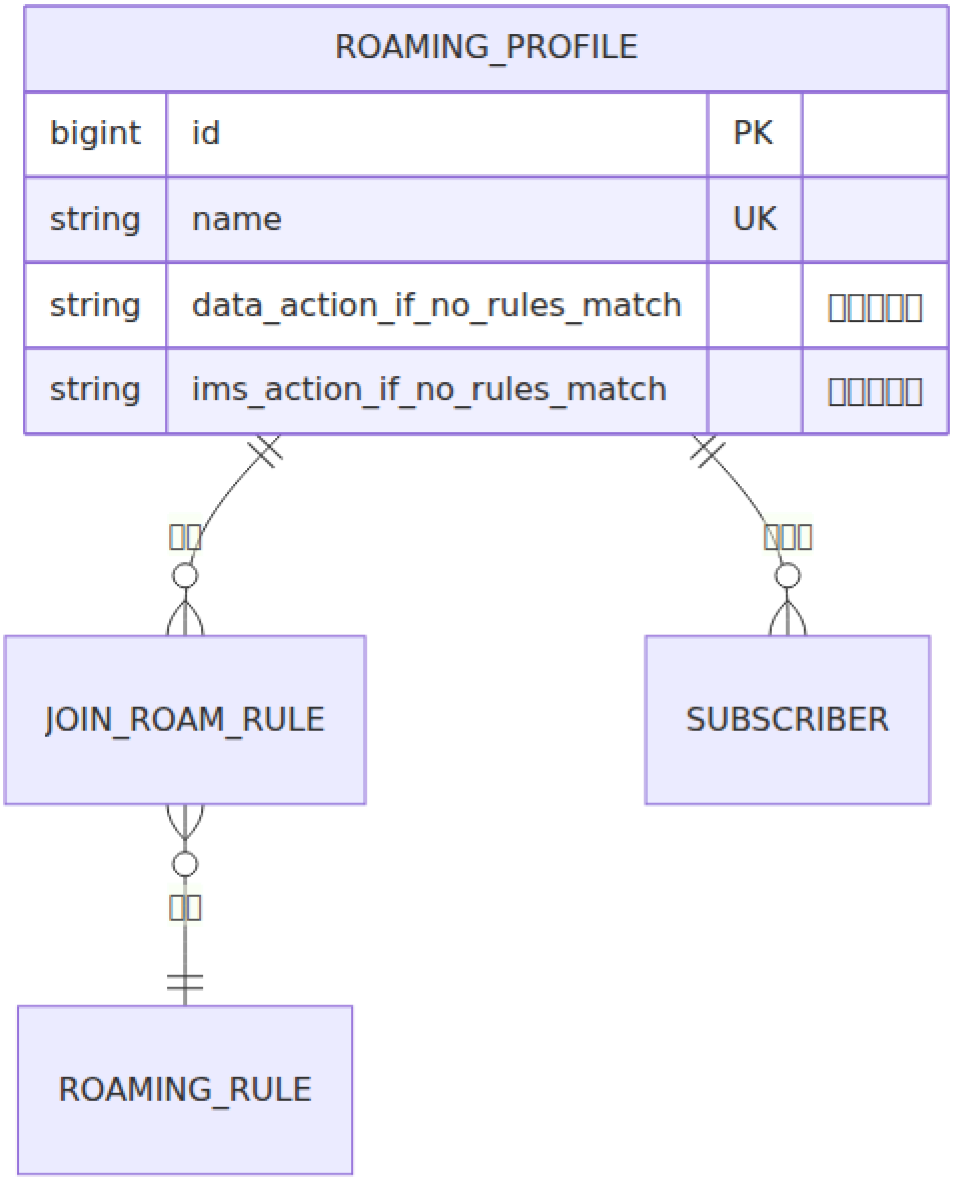

QCI
QCI 1-4
□□□□□
□□□□□
QCI 5-9
□□□
□□□□
QCI 1: □□□□
QCI 2: □□□□
QCI 4: □□□
QCI 5: IMS □□
QCI 9: □□□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□□

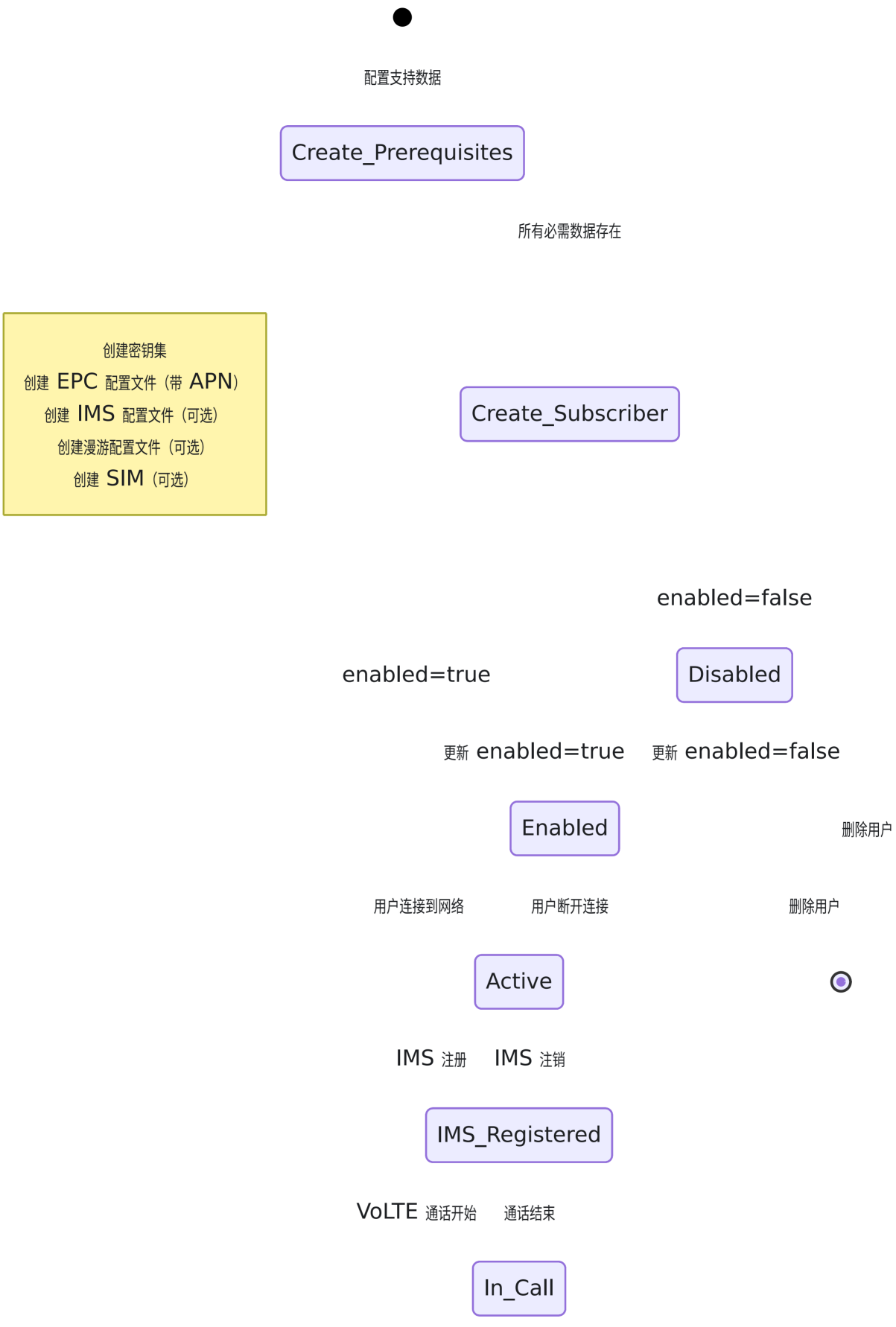

□□□□□□□
APN □□□
APN □□
APN QoS □□□□
QoS □□
□□□□□□□□
APN □□□□
□□□
□□□□
EPC □□□□
□□□□□□□
□□
SIM
□□□
IMS □□□□
□□□□
□□□□□□□
□□□□
□□
□□

会话状态关系

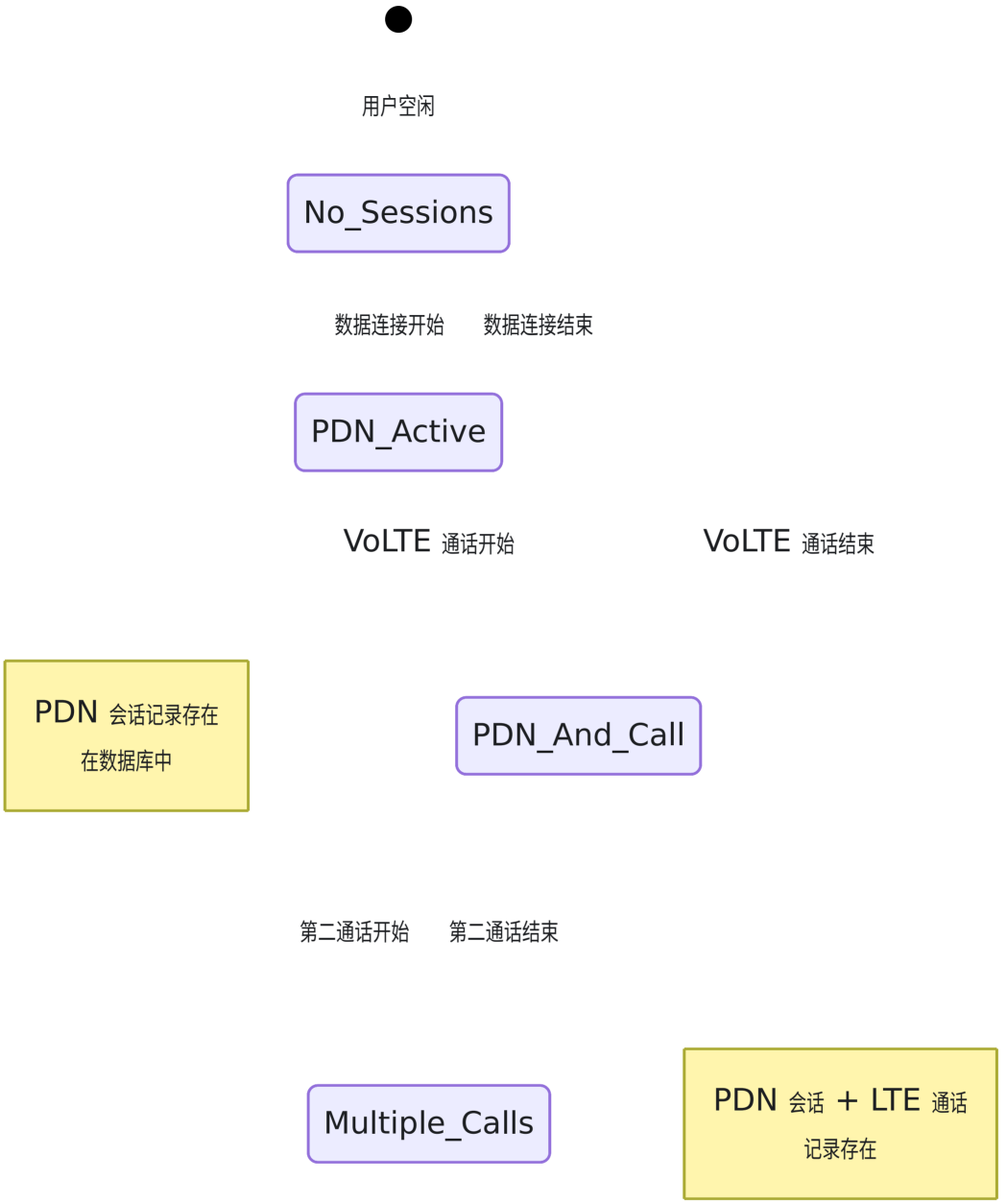

ROAMING_PROFILE
bigint
id
PK
string
name
UK
string
data_action_if_no_rules_match
string
ims_action_if_no_rules_match
JOIN_ROAM_RULE
SUBSCRIBER
ROAMING_RULE

实体生命周期

用户配置生命周期

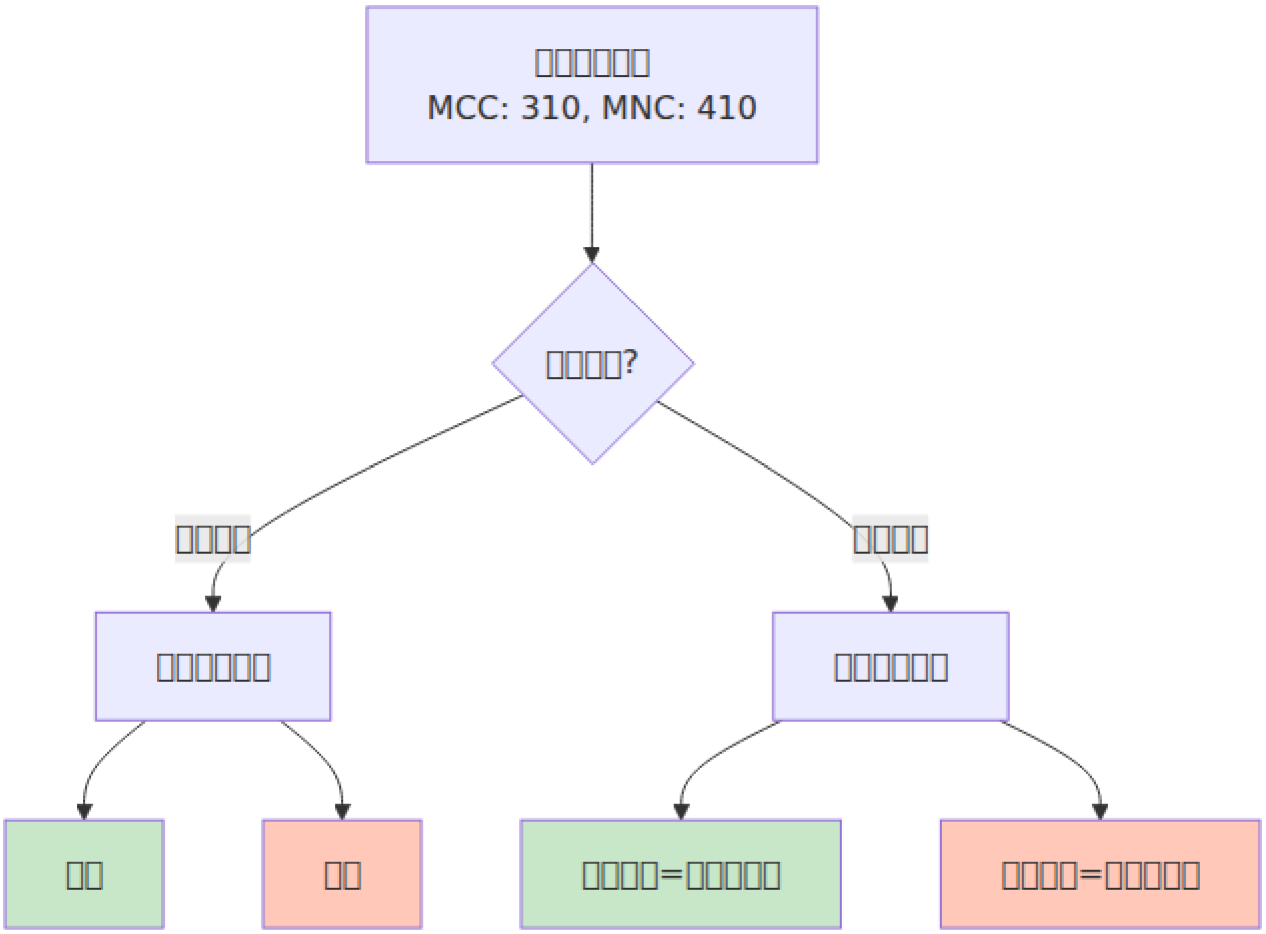

配置支持数据
Create_Prerequisites
所有必需数据存在
创建密钥集
创建 EPC 配置文件（带 APN）
创建 IMS 配置文件（可选）
创建漫游配置文件（可选）
创建 SIM（可选）
Create_Subscriber
enabled=false
enabled=true
Disabled
更新 enabled=true
更新 enabled=false
Enabled
删除用户
用户连接到网络
用户断开连接
删除用户
Active
IMS 注册
IMS 注销
IMS_Registered
VoLTE 通话开始
通话结束
In_Call

会话生命周期


用户空闲
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

数据流模式

身份验证流程



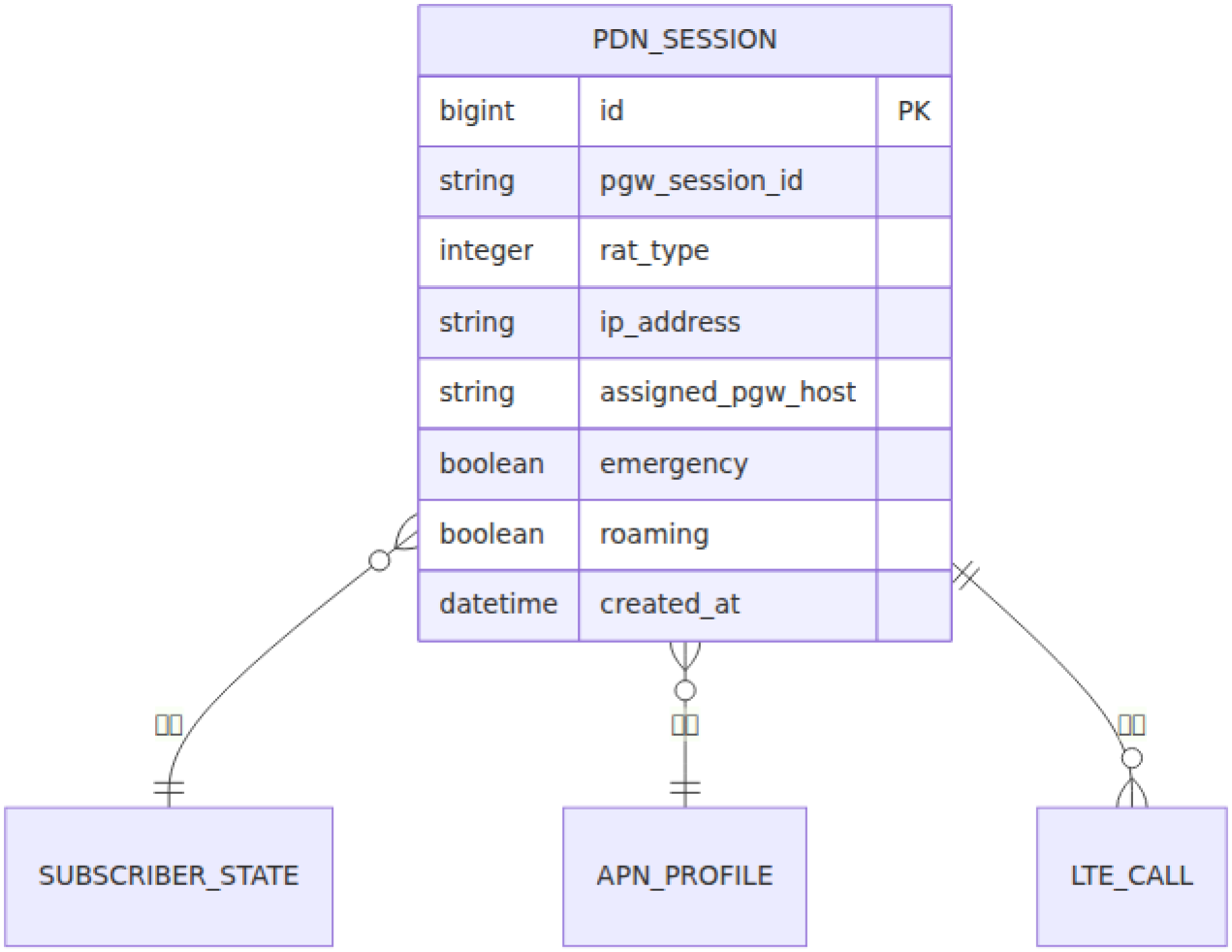
位置更新流程


S6a ULR 请求
□□ IMSI □□□□
加载 EPC 配置文件
+ APN 配置文件
更新用户状态
位置、MME 等
□□□□□□
AMBR□APN□QoS
S6a ULA 响应

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

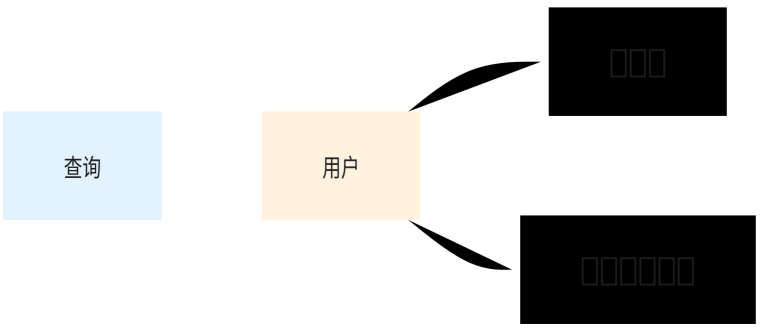
会话建立流程



查询优化模式

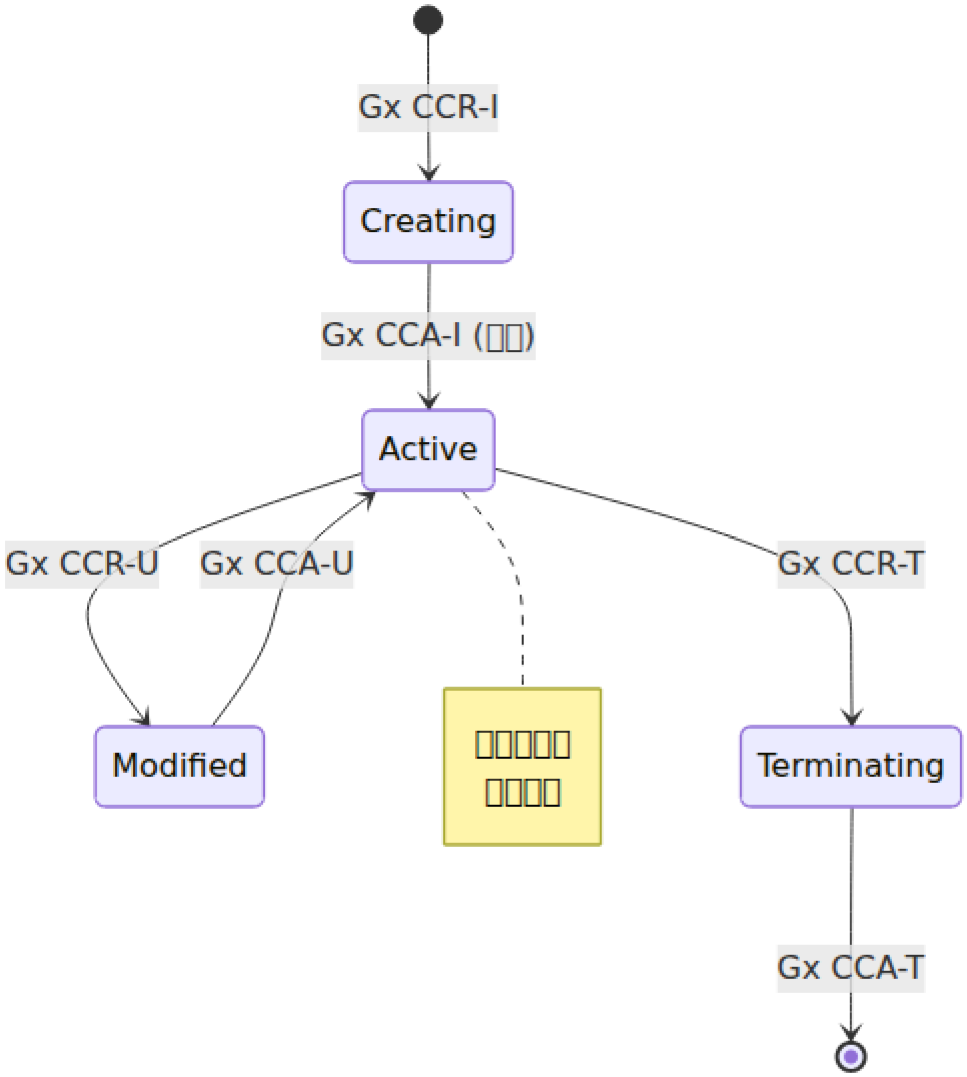
OmniHSS 通过选择性预加载每个操作所需的关联来优化数据库查询：

最小查询（身份验证）



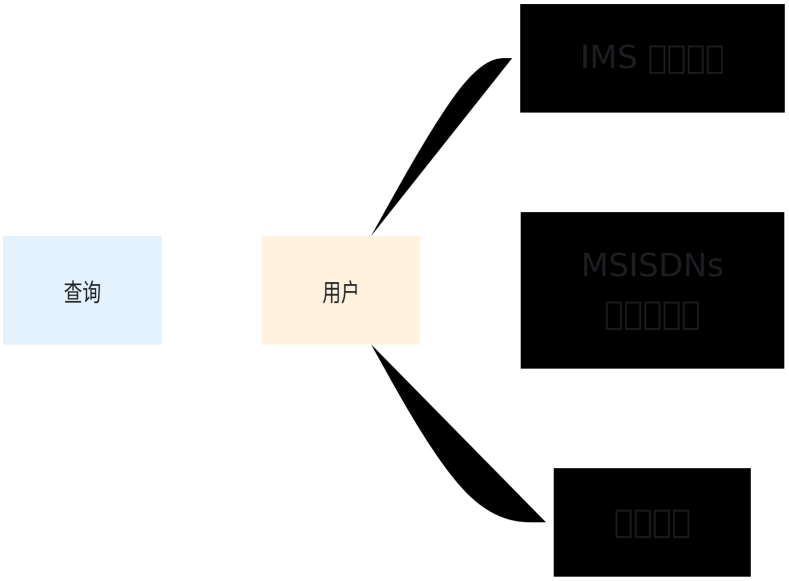
用例: S6a AIR - 仅需要加密密钥和漫游规则

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 →

Galera 数据库复制

← 返回操作指南

目录

数据库后端

OmniHSS 基于 Elixir 构建，使用 **Ecto** 作为其数据库抽象层。Ecto 支持多种关系数据库后端，允许在数据库选择上具有灵活性。**MariaDB** 与 Galera 集群□支持的配置之一，文档在此。

可以根据您的基础设施要求和操作偏好使用其他关系数据库后端。与 **Omnitouch** 网络服务 **(ONS)** 的集成团队合作，以确定最适合您环境的数据库后端和复制策略。

MariaDB 与 Galera

数据库	复制选项
MariaDB 10.6+	Galera 集群（本文档）

选择正确的方法

最佳的数据库和复制策略取决于您的环境：

- 现有的数据库基础设施和操作专业知识
- HSS 节点的数量和地理分布
- 可用性和故障转移要求
- 节点之间的网络延迟

您的 ONS 集成团队可以提供建议：

- 哪种数据库后端适合您的基础设施
- 适合您可用性要求的复制拓扑
- 针对您的用户量进行性能调优
- 与您现有的监控和备份系统集成

概述

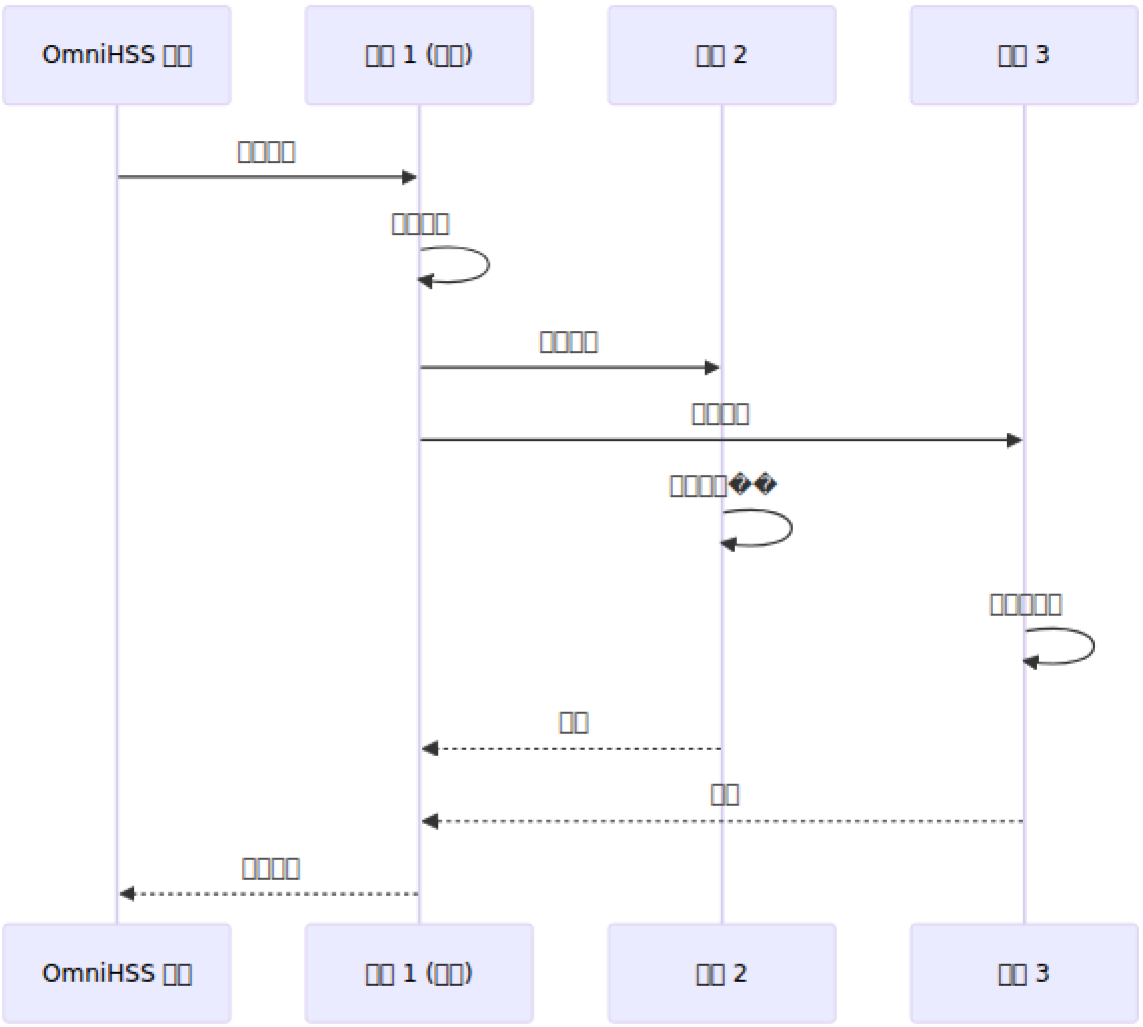
本文档涵盖 **MariaDB Galera** 集群，这□高可用性 OmniHSS 部署的主要复制选项。Galera 提供同步多主复制，确保所有 HSS 节点共享相同的用户数据，并具有自动故障转移功能。

主要优点

- 同步复制：所有节点始终具有一致的数据
- 多主：任何节点都可以接受读写操作
- 自动故障转移：如果某个节点失败，其他节点继续运行
- 自动节点恢复：返回的节点会自动重新同步
- 无脑裂：基于认证的复制防止冲突

Galera 的工作原理

同步复制流程



写集复制 (WSREP)

每个数据库事务遵循以下流程：

1. 事务执行：客户端在本地节点上执行 SQL
2. 写集创建：节点将更改打包成“写集”
3. 认证：所有节点验证写集以检查冲突
4. 提交：如果认证通过，所有节点原子性提交
5. 确认：只有在所有节点提交后，成功才返回给客户端

状态转移方法

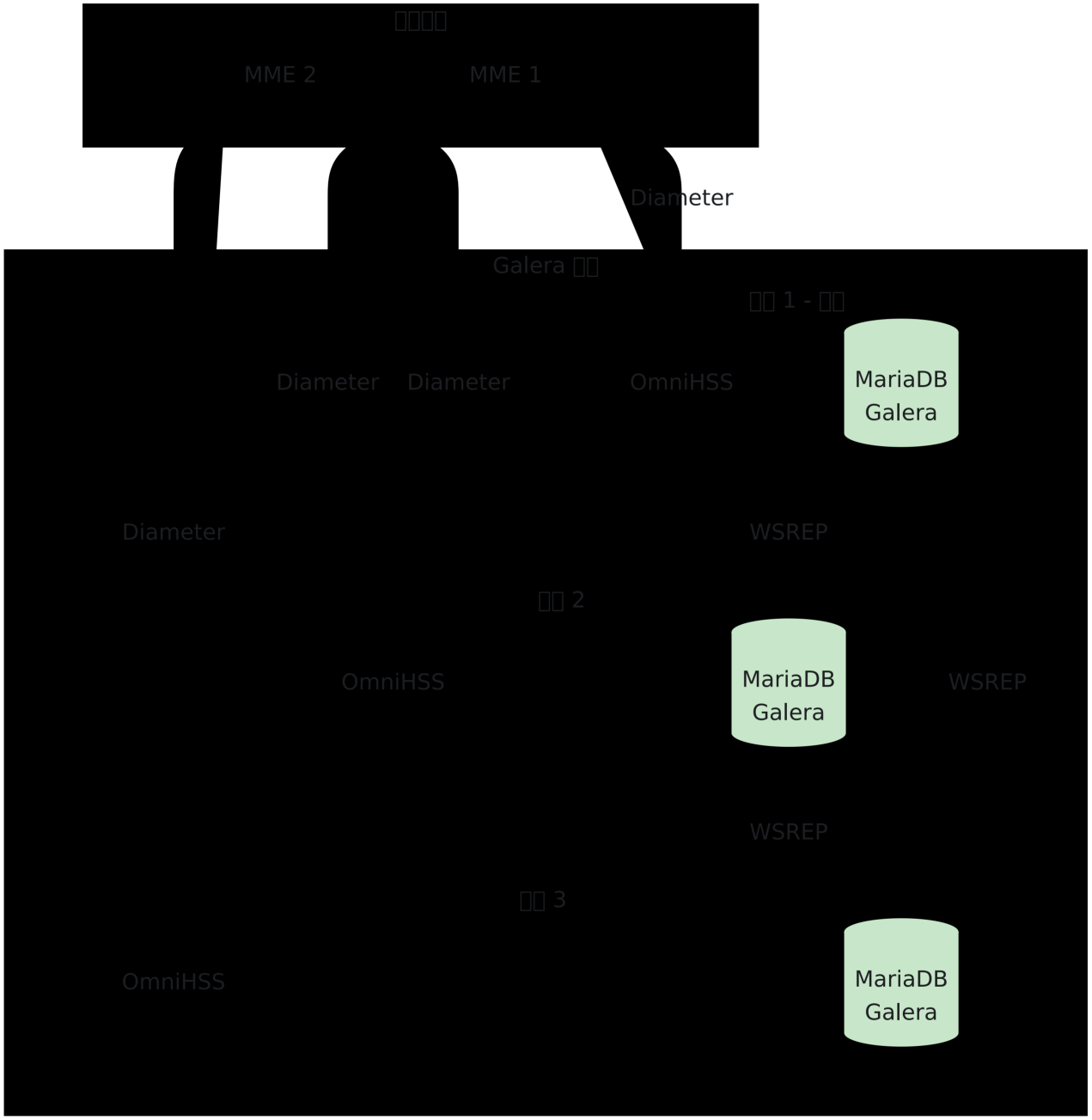
方法	类型	用例
IST (增量)	增量同步	节点短暂断开，赶上错过的事务
SST (快照)	完全同步	新节点或长时间断开的节点，接收完整的数据库副本

OmniHSS 使用 `rsync` 进行 SST:

```
wsrep_sst_method=rsync
```

部署架构

多节点集群


MME 2
MME 1
Diameter
Galera
1 -
Diameter
Diameter
OmniHSS
MariaDB
Galera
Diameter
WSREP
2
OmniHSS
MariaDB
Galera
WSREP
WSREP
3
OmniHSS
MariaDB
Galera

网络要求

端口	协议	目的
3306	TCP	MySQL 客户端连接
4567	TCP/UDP	Galera 集群通信
4568	TCP	增量状态转移 (IST)
4444	TCP	状态快照转移 (SST)

防火墙配置

```
# 在 Galera 节点之间
ufw allow from <node2_ip> to any port 3306,4567,4568,4444 proto
tcp
ufw allow from <node2_ip> to any port 4567 proto udp
ufw allow from <node3_ip> to any port 3306,4567,4568,4444 proto
tcp
ufw allow from <node3_ip> to any port 4567 proto udp
```

配置参考

Ansible 变量

在您的库存的 group_vars 中配置 Galera:

```yaml
omnihss:
  database_host: "localhost"
  database_username: "hss"
  database_password: "secure_password"
  mysql:
    replication_mode: "galera"          # 启用 Galera
    bootstrap_host: "hss01"             # 启动集群的第一个节点
    run_bootstrap: false                # 仅在初始设置时设置为 true
    reinstall: false                    # 设置为 true 以重新安装 MariaDB
```

Galera 配置文件

Galera 配置模板为 `/etc/mysql/my.cnf`:

```
[mysqld]
# 核心设置
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
datadir         = /var/lib/mysql
log-error       = /var/log/mysql/error.log

# Galera 所需
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera 提供者
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# 集群配置
wsrep_cluster_name="omnihss_galera"
wsrep_cluster_address="gcomm://10.4.10.140,10.4.10.141,10.4.10.142"

# 状态转移
wsrep_sst_method=rsync

# 节点身份
wsrep_node_address="10.4.10.140"
wsrep_node_name="hss01"
```

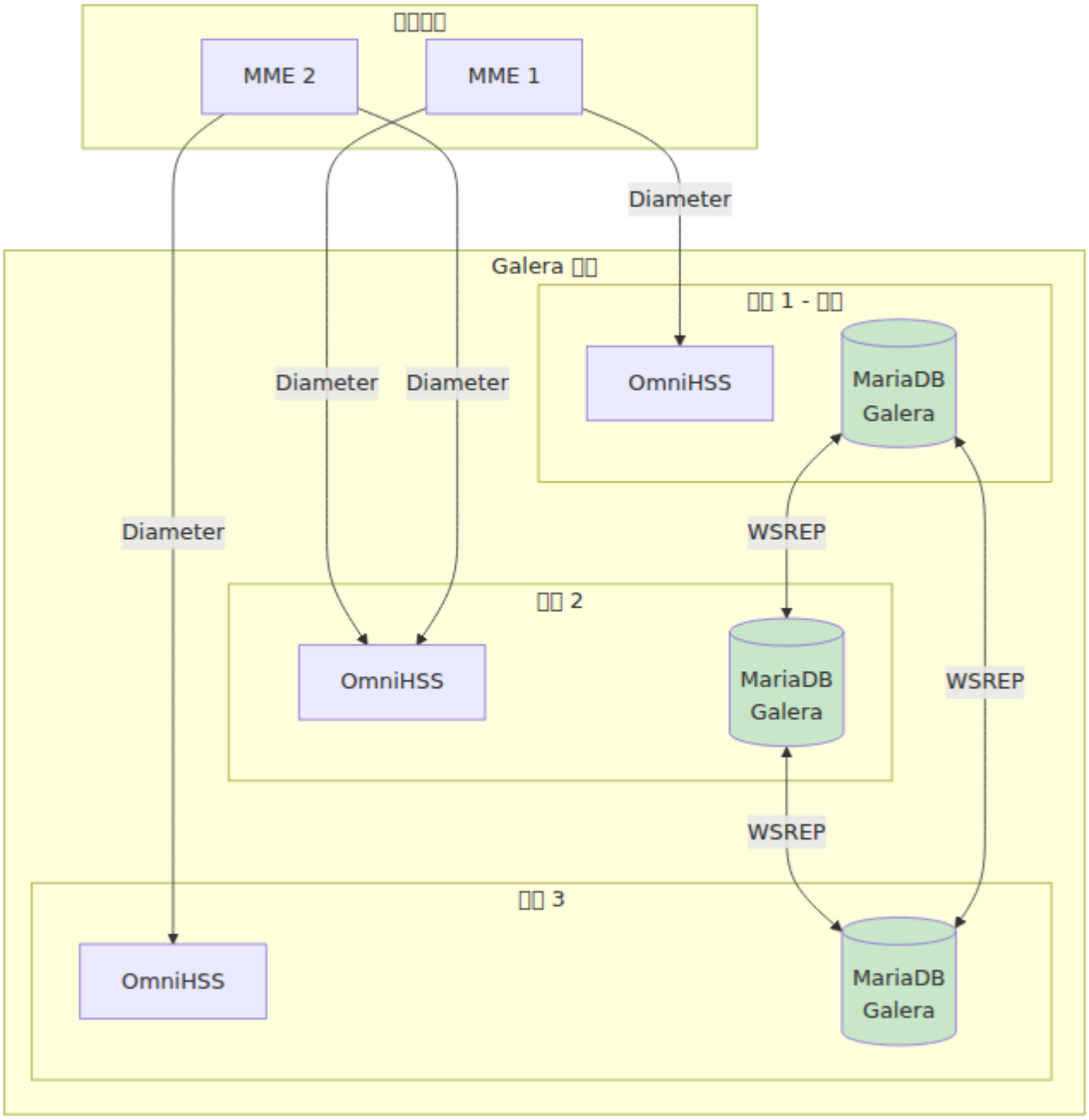
配置参数解释

参数	值	目的
`binlog_format`	`ROW`	必需 - 行级日志用于复制
`innodb_autoinc_lock_mode`	`2`	必需 - 允许并发自增
`wsrep_on`	`ON`	启用 WSREP 复制
`wsrep_provider`	libgalera 的路径	Galera 库位置
`wsrep_cluster_name`	`"omnihss_galera"`	所有节点必须使用相同名称
`wsrep_cluster_address`	`gcomm://ip1,ip2,ip3`	所有集群节点 IP 的列表
`wsrep_sst_method`	`rsync`	完整状态转移方法
`wsrep_node_address`	节点 IP	此节点的集群 IP
`wsrep_node_name`	节点主机名	此节点的标识符

引导过程

初始集群设置

引导过程创建一个新的 Galera 集群:



[illegible]

1. [illegible]

```
omnihss:
  mysql:
    replication_mode: "galera"
    bootstrap_host: "hss01"
    run_bootstrap: true
```

2. 运行 **Ansible** 剧本

```
ansible-playbook -i hosts/your_site/inventory.ini
services/omnihss.yml
```

3. 发生了什么:

   - 禁用 AppArmor（Galera 所需）
   - 在所有节点上安装 Galera 软件包
   - 引导节点在 `/var/lib/mysql/grastate.dat` 中设置 `safe_to_bootstrap=1`
   - 引导节点运行 `mysqld_bootstrap` 命令
   - 其他节点重启 MariaDB 并通过 `gcomm://` 地址加入
   - 数据库迁移仅在引导节点上运行（更改会复制）

4. 引导后

```yaml
omnihss:
  mysql:
    run_bootstrap: false    # 禁用未来运行的引导
```

Grastate 文件

集群状态记录在 `/var/lib/mysql/grastate.dat` 中:

```
# GALERA 保存的状态
version: 2.1
uuid:    abc12345-6789-def0-1234-567890abcdef
seqno:   1234567
safe_to_bootstrap: 0
```

- `uuid`: 集群唯一标识符
- `seqno`: 最后提交的事务序列号
- `safe_to_bootstrap`: 仅在应启动集群的节点上设置为 `1`

□作

添加新节点

1. 在库存中使用 `hss` 组配置新节点
2. 更新 `wsrep_cluster_address` 以包含所有节点
3. 运行 OmniHSS 剧本 - 节点将自动:
   - 安装 Galera 软件包
   - 获取带有集群地址的配置
   - 通过 SST 加入集群

移除节点

1. 停止要移除的节点上的 OmniHSS 和 MariaDB
2. 从库存中移除该节点
3. 更新剩余节点上的 `wsrep_cluster_address`
4. 重启剩余节点上的 MariaDB

受控重启

进行维护时，逐个重启节点：

```
# 在每个节点上，逐个进行
systemctl stop omnihss
systemctl stop mysql
# 执行维护
systemctl start mysql
systemctl start omnihss
```

在重启下一个节点之前，等待每个节点完全重新加入。

紧急恢复

如果整个集群停止（停电等）：

1. 识别最近的节点

```
# 检查每个节点上的 seqno
cat /var/lib/mysql/grastate.dat
```

2. 从最近的节点引导

```
# 在具有最高 seqno 的节点上
sed -i "/safe_to_bootstrap/s/0/1/" /var/lib/mysql/grastate.dat
mysqld_bootstrap
```

3. 启动其他节点

```
# 在其他节点上
systemctl start mysql
```

监控

集群状态

在任何节点上查询集群状态：

```
-- 集群大小（节点数量）
SHOW STATUS LIKE 'wsrep_cluster_size';

-- 集群状态
SHOW STATUS LIKE 'wsrep_cluster_status';

-- 节点状态
SHOW STATUS LIKE 'wsrep_local_state_comment';

-- 所有 WSREP 变量
SHOW STATUS LIKE 'wsrep_%';
```

关键指标

指标	健康值	描述
`wsrep_cluster_size`	预期节点数量	集群中的节点数量
`wsrep_cluster_status`	`Primary`	集群具有法定人数
`wsrep_local_state`	`4`	节点已同步
`wsrep_local_state_comment`	`Synced`	节点状态描述
`wsrep_ready`	`ON`	节点接受查询
`wsrep_connected`	`ON`	节点已连接到集群

节点状态

状态	值	描述
加入中	1	节点正在加入集群
供体/不同步	2	节点正在向其他节点提供 SST
已加入	3	节点已加入，正在同步
已同步	4	节点已完全同步

Prometheus 指标

在使用 MariaDB Galera 时，OmniHSS 通过标准指标端点公开 Galera 指标。

故障排除

节点无法加入集群

症状：节点启动但未加入集群

检查：

```
# 查看 MariaDB 错误日志
tail -f /var/log/mysql/error.log

# 检查 wsrep 是否正在运行
mysql -e "SHOW STATUS LIKE 'wsrep_on';"
```

常见原因：

- 防火墙阻止端口 4567、4568、4444
- `wsrep_cluster_address` 中的 IP 错误
- AppArmor 仍然启用
- 集群 UUID 不匹配

修复：

```
# 确保 AppArmor 已禁用
systemctl status apparmor
# 如果处于活动状态：systemctl stop apparmor && systemctl disable apparmor

# 验证端口是否打开
ss -tlnp | grep -E '4567|4568|4444|3306'
```

脑裂 / 非主状态

症状：`wsrep_cluster_status` 显示 `non-Primary`

发生原因：

- 集群失去法定人数（大多数节点宕机）

- 网络分区隔离节点

恢复：

```
-- 在数据最新的节点上
SET GLOBAL wsrep_provider_options='pc.bootstrap=YES';
```

SST 失败

症状：新节点无法完成状态转移

检查：

```
# 检查供体和加入者的磁盘空间
df -h /var/lib/mysql

# rsync 进程
ps aux | grep rsync
```

常见原因：

- 磁盘空间不足
- rsync 未安装
- 防火墙阻止端口 4444

节点在供体后不同步

症状：`wsrep_local_state_comment` 显示 `Donor/Desynced`

在 **SST** 期间这是正常的。节点在完成对加入节点的状态转移后会恢复正常状态。

如果卡住：

```
# 检查是否有卡住的 rsync
ps aux | grep rsync
# 如果卡住，终止
pkill rsync
systemctl restart mysql
```

Grastate 损坏

症状：MariaDB 无法启动，关于 grastate 的错误

修复：

```
# 删除损坏的 grastate
rm /var/lib/mysql/grastate.dat

# 作为新节点启动□将从现有集群进行 SST）
systemctl start mysql
```

性能下降

症状：写入缓慢，高 `wsrep_local_send_queue`

检查：

```
SHOW STATUS LIKE 'wsrep_local_send_queue%';
SHOW STATUS LIKE 'wsrep_flow_control%';
```

常见原因：

- 节点之间的网络延迟
- 某个节点显著较慢（磁盘 I/O）
- 非常大的事务

缓解措施：

- 确保节点之间的低延迟网络
- 使用相似的硬件配置所有节点

- 避免非常大的批量操作

Diameter 响应数据映射

← 返回操作指南

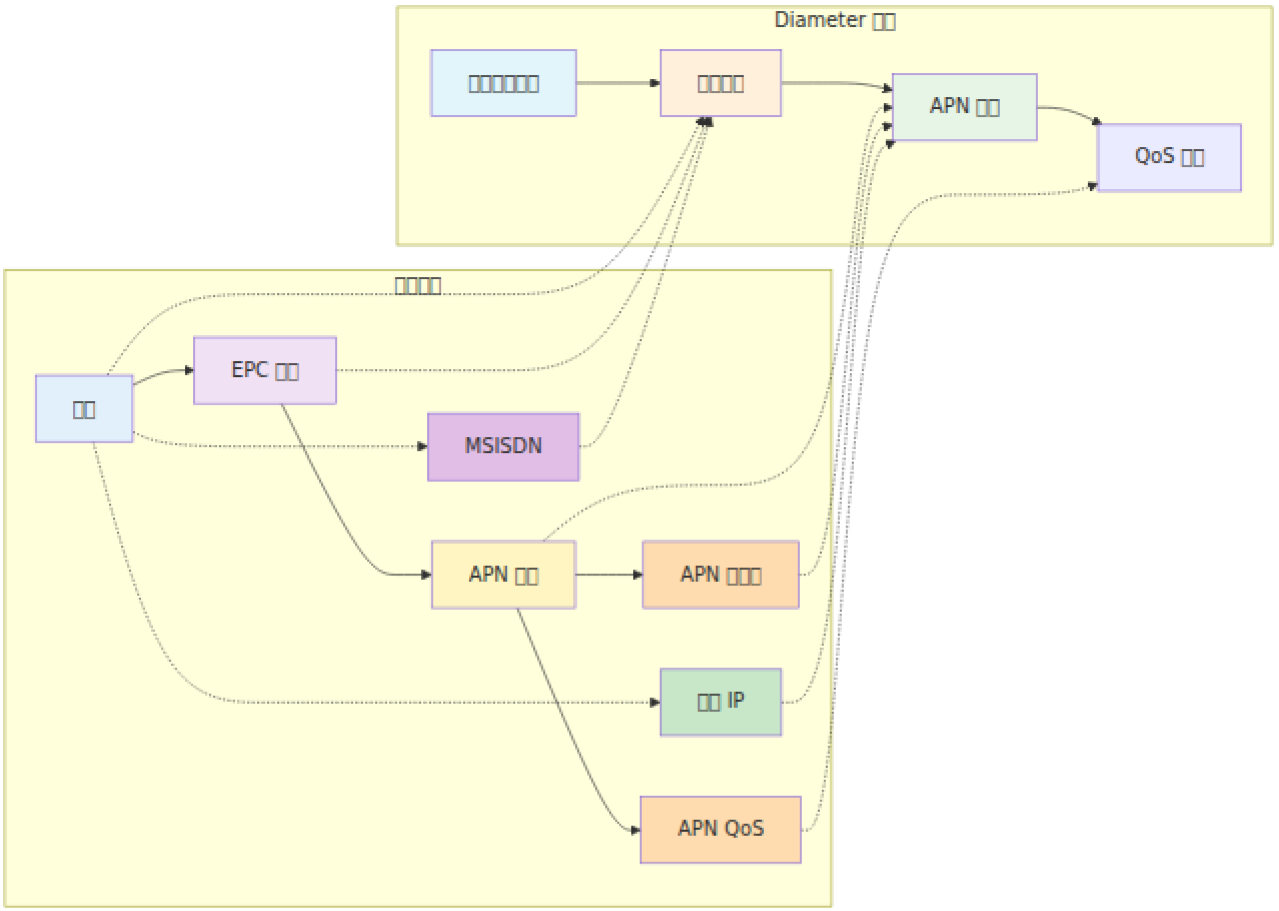
本文档提供详细的 mermaid 图表，显示 Diameter 协议响应中每个字段在 OmniHSS 系统中的来源。

目录

更新位置应答 (S6a ULA)

更新位置应答由 HSS 在 LTE 附加过程中发送给 MME。该图显示了从数据库表到 Diameter AVP 的完整数据流。

数据源映射



详细字段映射

数据库来源	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

数据库来源	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：为配置中的每个 APN 进行顺序编号（0, 1, 2...）
5. 静态 **IP**：仅在通过 `static_ips` 多对多关系分配时包含

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：从 CK||IK 使用 KDF 根据 TS 33.401 派生密钥

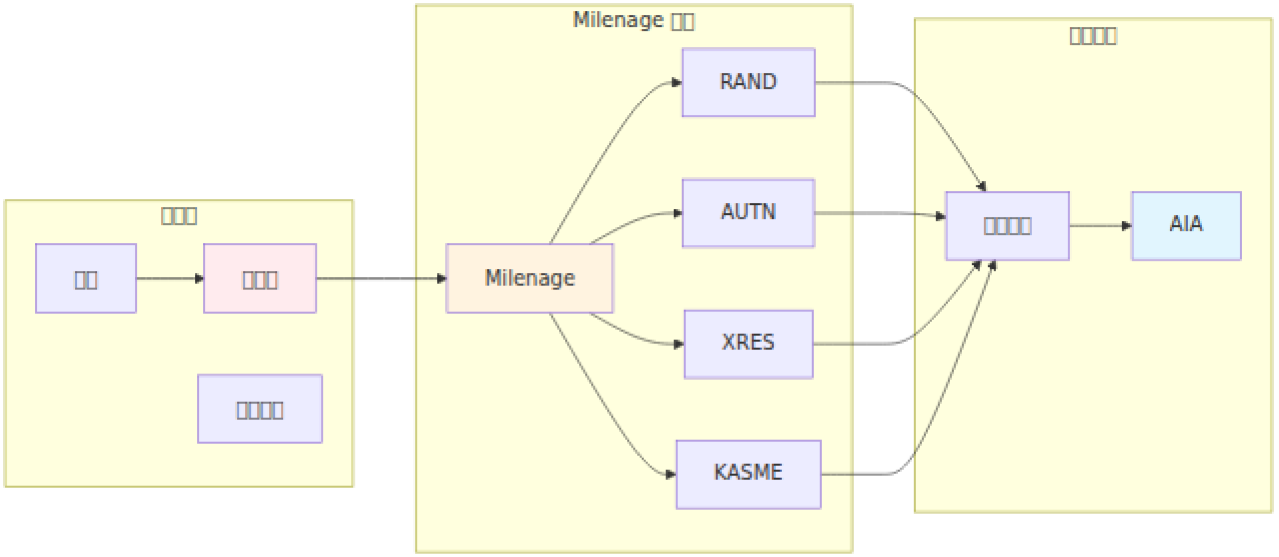
安全特性：

- 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 包含序列号（SQN）和 AMF 进行网络认证
- Milenage 算法提供 UE 和网络之间的相互认证

服务器分配应答 (Cx SAA)

服务器分配应答由 HSS 在 IMS 注册期间发送给 S-CSCF。

数据源映射



关键特性：

1. **IFC** 模板：XML 模板存储在 `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}`

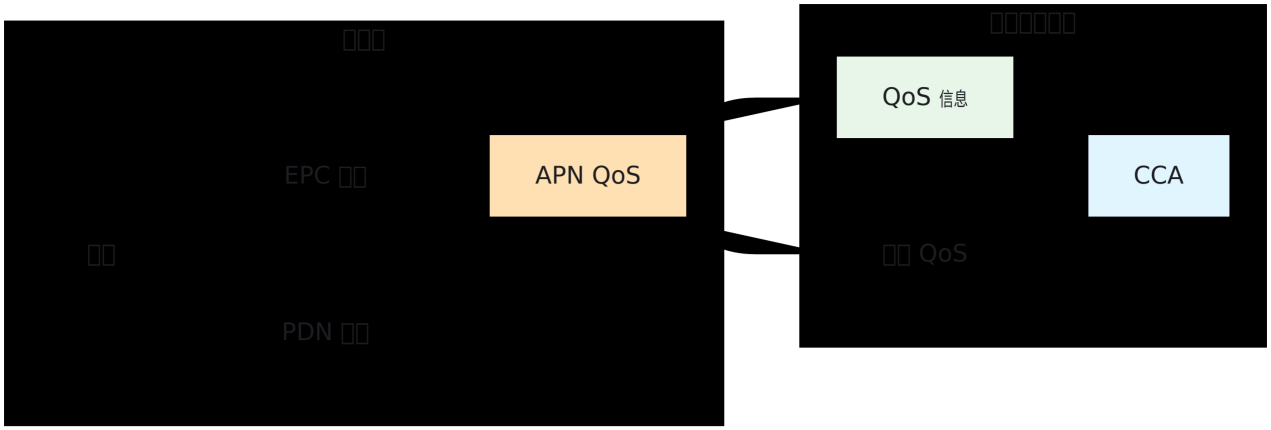
IFC 模板参数：

- `{{msisdn}}` - 来自用户的第一个 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）

信用控制应答 (Gx CCA)

信用控制应答由 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** 强制执行：提供来自 APN QoS 配置的 QCI 和带宽限制
3. 计费规则：返回默认计费规则以进行计费集成
4. **CC-**请求类型：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

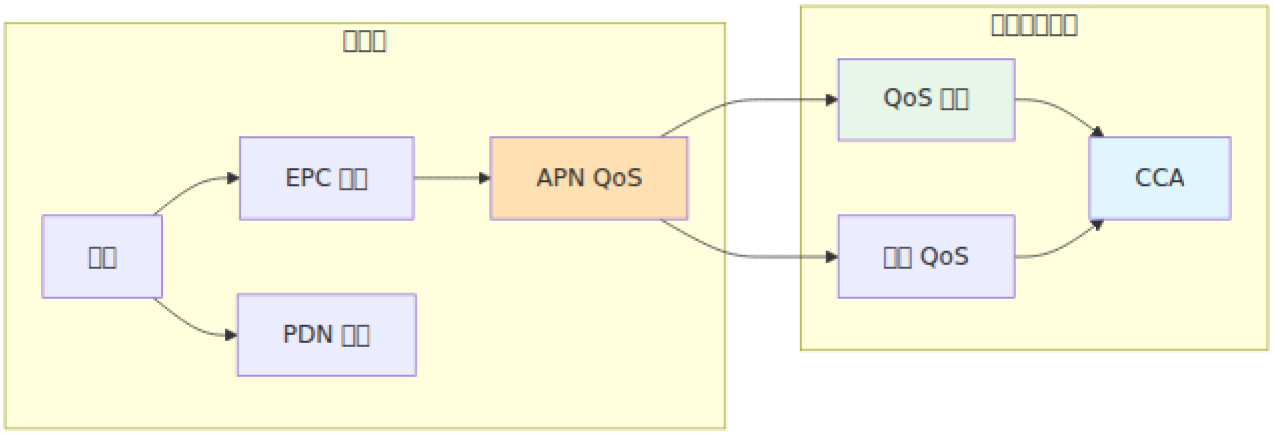
关键特性：

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：支持引用和透明数据模式

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：

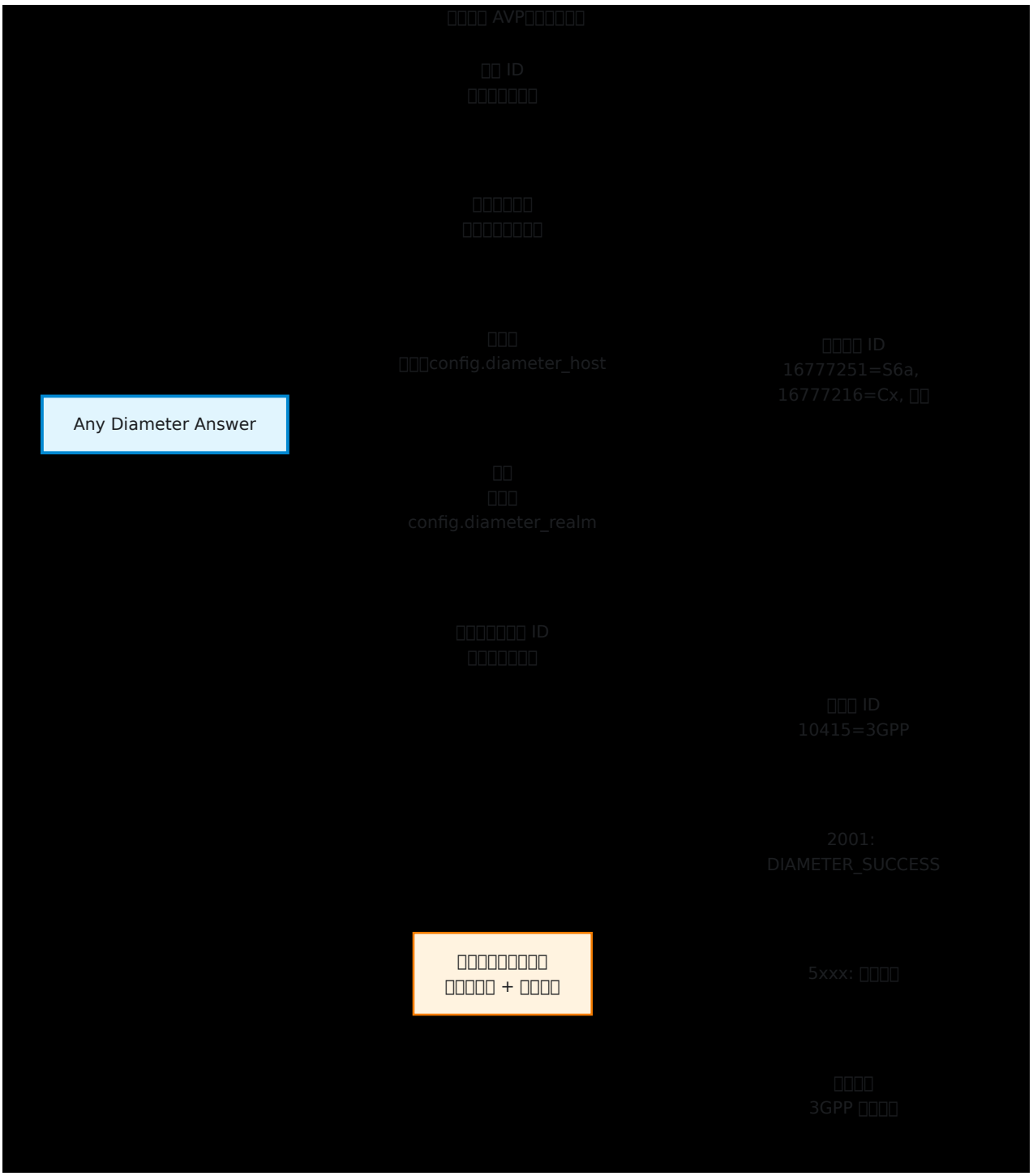
1. **IMEI** 正则匹配：规则使用正则表达式进行灵活匹配
2. 基于 **TAC** 的规则：可以匹配类型分配代码（前 8 位数字）
3. 默认行为：对于未知 IMEI 可配置（接受或拒绝）
4. 设备状态值：
   - `0` = 白名单（明确允许）
   - `1` = 黑名单（被盗/阻止）
   - `2` = 灰名单（允许但监控）
   - `5` = 未知（没有匹配规则）

用例：

- 通过精确 IMEI 阻止被盗��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

常见响应元素

所有 Diameter 响应共享这些共同的 AVP:

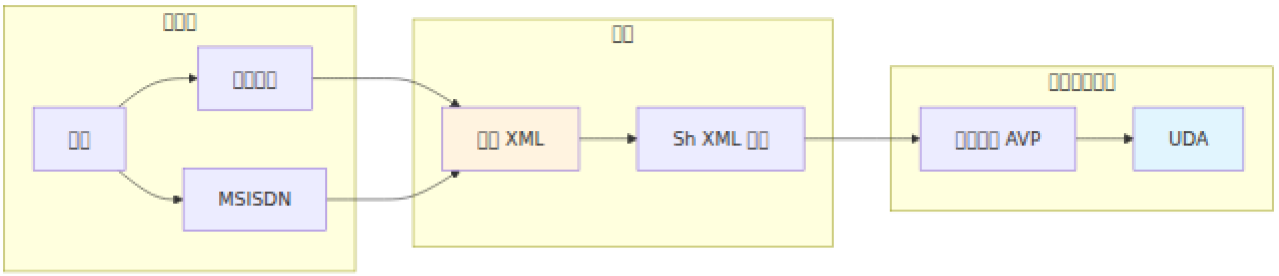


□□□□□

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

数��流摘要

请求处理管道



实施说明

协议处理器

系统实现了以下 Diameter 协议的处理器:

- **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���
- **SWx** - WiFi/IMS 接口用于非 3GPP 访问认证

数据模型

数据库架构包括以下核心实体:

- 用户 - 核心用户记录，包含 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 - 包含 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

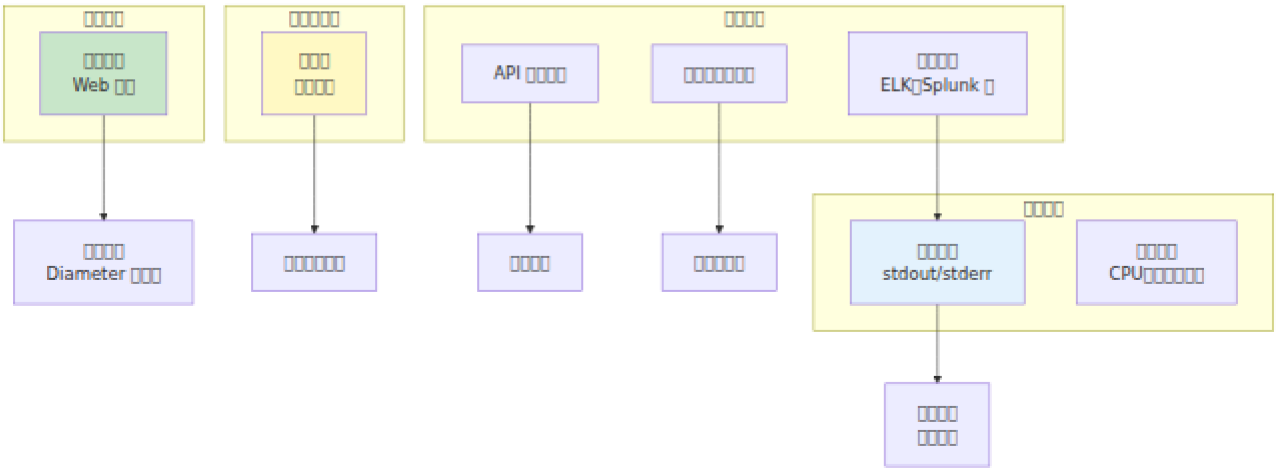
← 返回操作指南

目录

监控概述

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

监控层



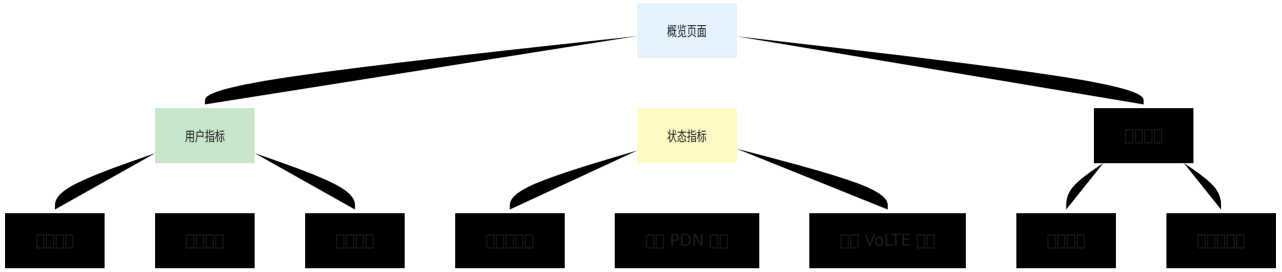
控制面板监控

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	意义
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 会话数 > 0	活动数据连接
IMS 注册	S-CSCF 分配	语音服务准备就绪
通话中	活动通话数 > 0	VoLTE 通话进行中

从概览中提取指标

虽然控制面板不直接导出指标，但您可以：

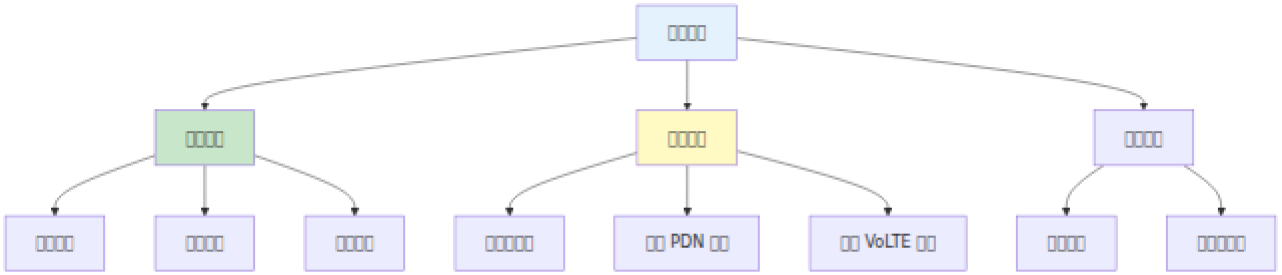
1. 计算可见行数以获取总用户数
2. 扫描绿色勾号以计算启用的用户
3. 查看扩展详细信息以获取状态信息

4. 记录最后一次看到的时间戳以获取响应性

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	重要性	如果断开连接的影响
MME	高	无法进行新的 LTE 附加
P-GW	高	无数据会话
S-CSCF	高	无 IMS 注册
P-CSCF	高	无 VoLTE 通话
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
进程数量	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：

用户数量

查询数据库以检索：

- 所有用户的总数
- 启用用户的数量
- IMS 启用用户的数量

会话统计

查询数据库以检索：

- 活动 PDN 会话的数量
- 活动 VoLTE 通话的数量
- 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）分组的用户数量
- 当前漫游的用户数量（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数量
- 按服务 MME 划分的用户分布
- 最后用户活动的时间戳分析

数据库健康监控

通过查询监控数据库健康：

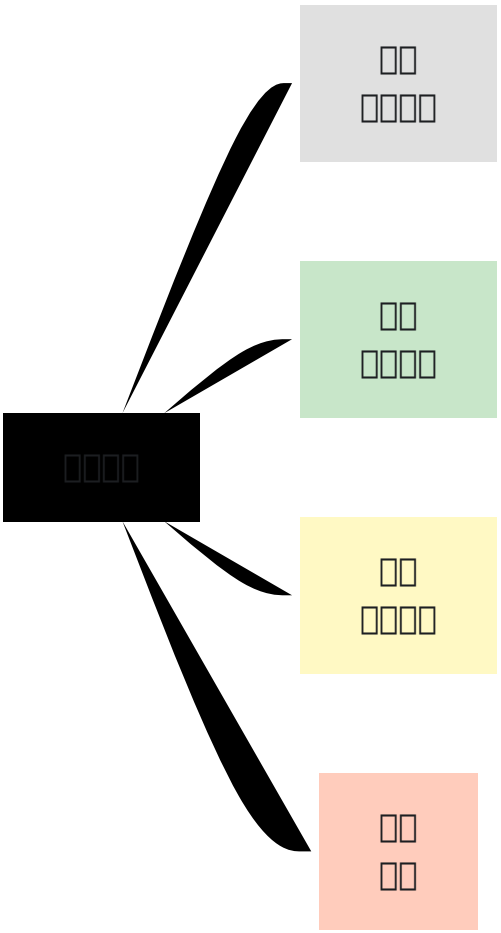
- 数据库总大小和增长趋势
- 各个表��大小和行数
- 当前数据库连接数
- 查询性能和资源使用情况

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
```

□□□□□:

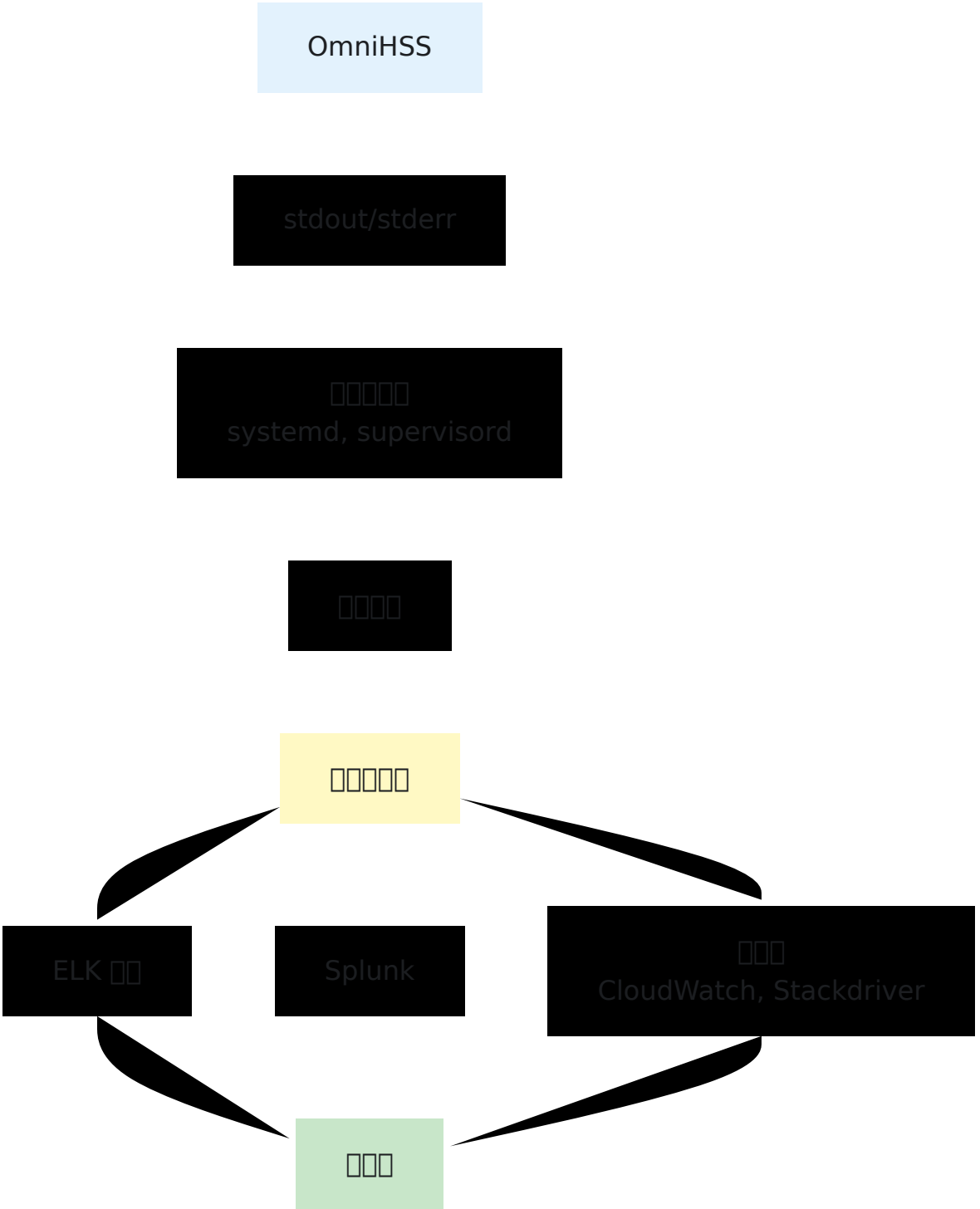
```
[info] Database connection established
[error] Database connection lost: timeout
[error] Database query failed: deadlock detected
```

□□□□:

```
[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

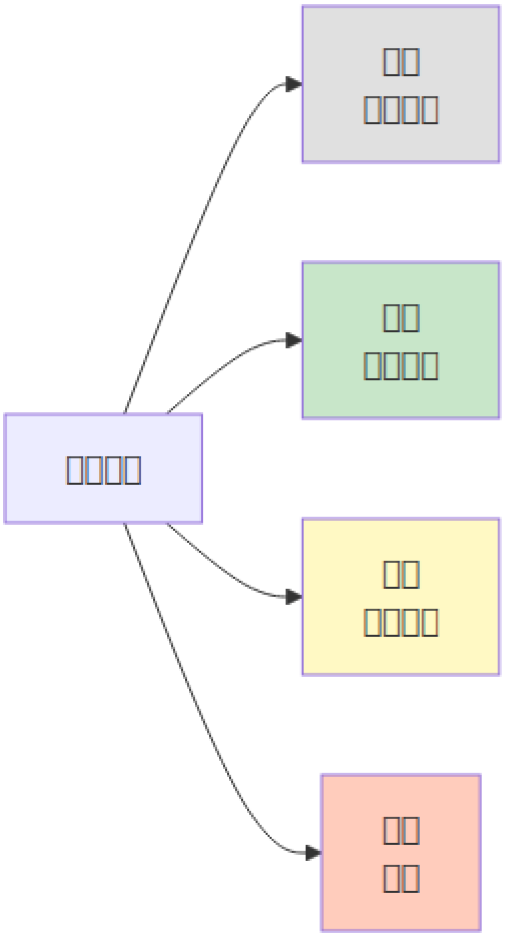
可以创建自定义出口将 OmniHSS 指标导出到 Prometheus，通过查询 API 和数据库。

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

□作 KPI



推荐的 KPI 阈值

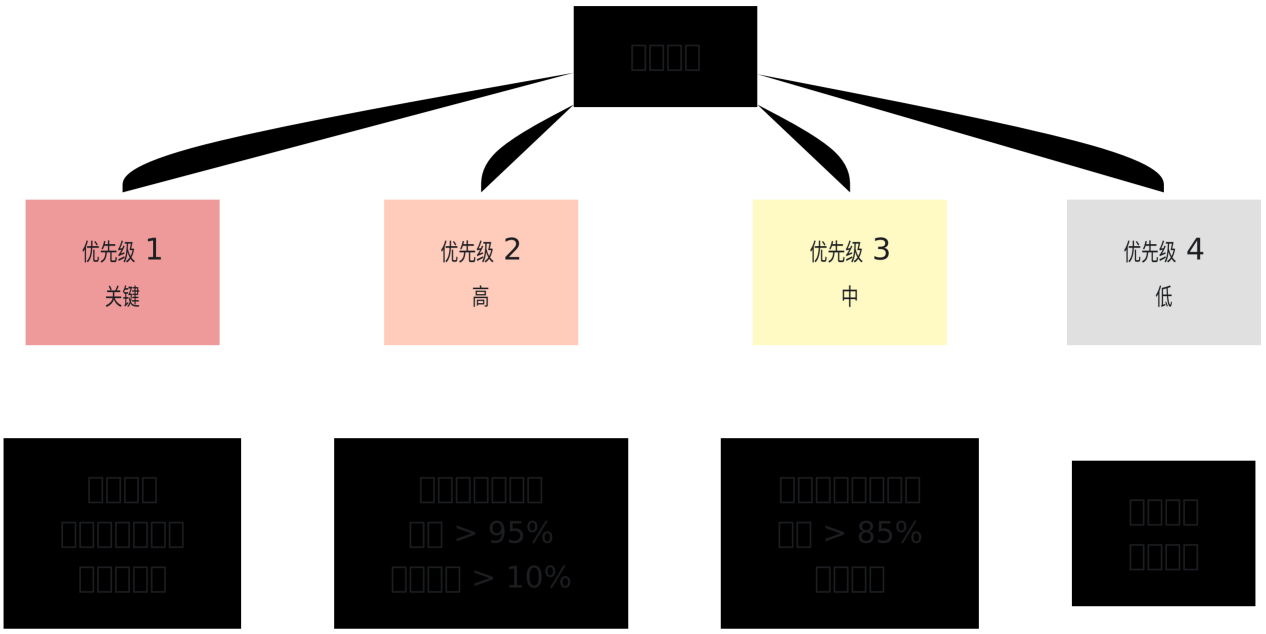
KPI	目标	警告	严重
系统运行时间	99.99%	< 99.95%	< 99.9%
Diameter 对等体运行时间	99.9%	< 99.5%	< 99%
认证成功率	> 99%	< 99%	< 95%
Diameter 响应时间	< 100ms	> 200ms	> 500ms
数据库查询时间	< 50ms	> 100ms	> 500ms
错误率	< 0.1%	> 0.5%	> 1%

容量 KPI

指标	监控	计划行动阈值
总用户数	当前数量	80% 的预期容量
并发 PDN 会话	活动会话	70% 的预期最大值
数据库大小	已使用 MB	80% 的分配存储
数据库连接	活动连接	80% 的池大小

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，必要时重启

数据库宕机：

- 无法连接到 SQL 数据库

- 行动：调查数据库服务器，必要时重启

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：查看日志，识别根本原因

低优先级警报 **(P4)**

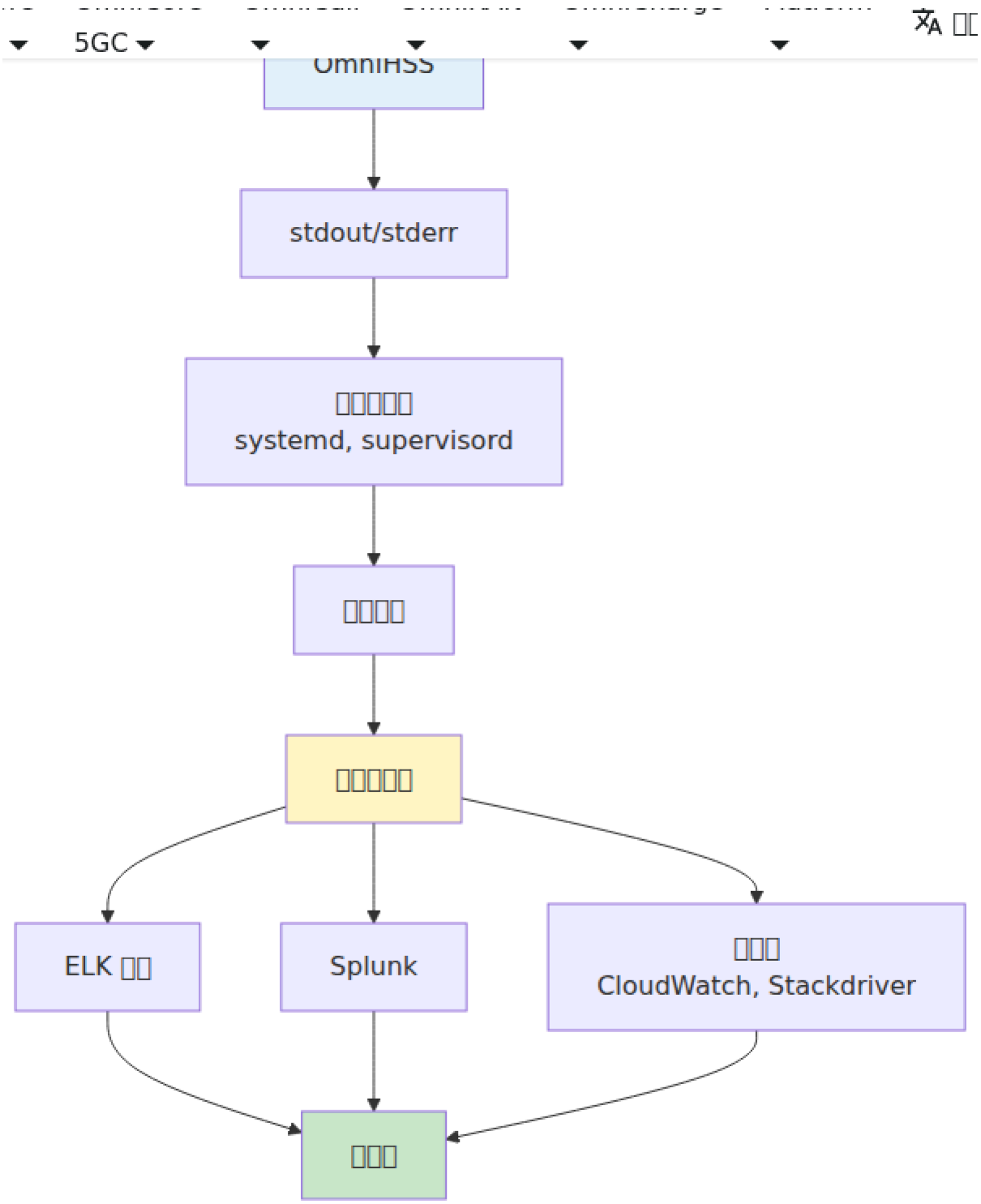
容量警告：

- 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

- [ ] 查看控制面板概览 - 用户数量正常
- [ ] 查看 Diameter 页面 - 所有关键对端体已连接
- [ ] 查看应用页面 - 内存和进程在限制范围内
- [ ] 检查错误日志 - 过去 24 小时内无关键错误
- [ ] 验证备份成功完成

每周检查

- [ ] 查看容量趋势 - 用户增长
- [ ] 查看性能趋势 - 响应时间
- [ ] 查看数据库大小 - 增长率可接受
- [ ] 查看错误率 - 识别模式
- [ ] 测试警报通知 - 确保正常工作

每月检查

- [ ] 容量规划审查 - 预测 6 个月后
- [ ] 性能优化审查 - 识别慢查询
- [ ] 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

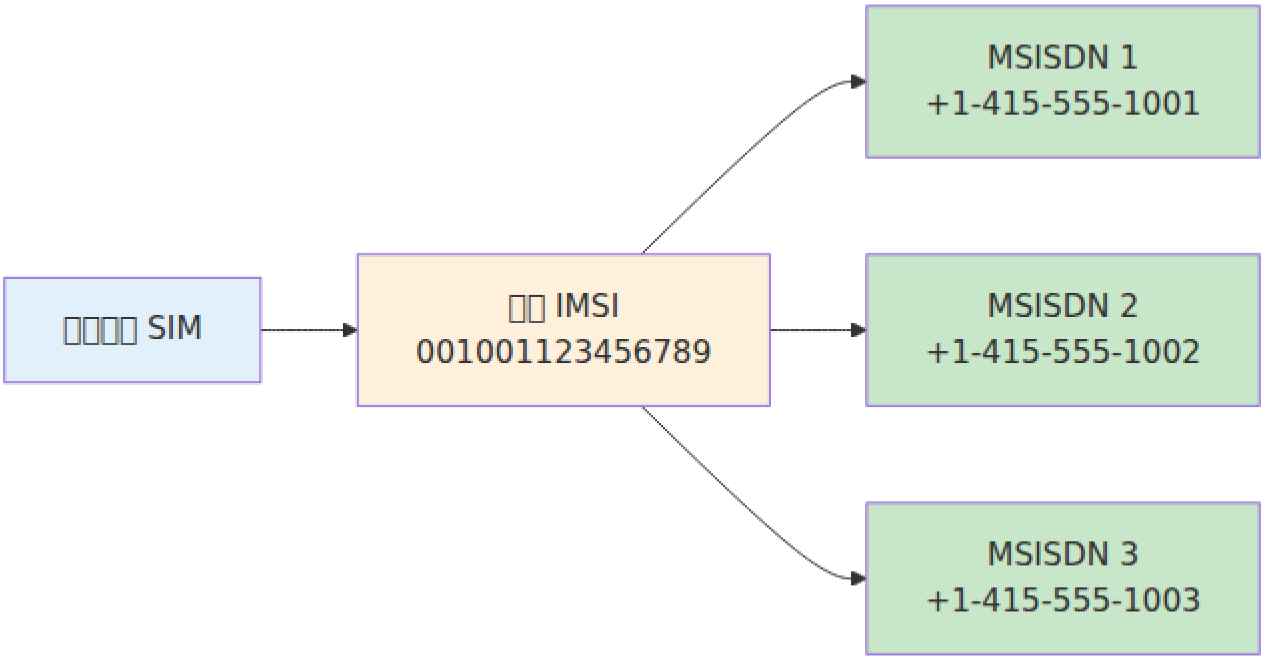
概述

OmniHSS 支持先进的配置能力，使灵活的服务配置成为可能：

多 MSISDN 支持

一个 **IMSI** → 多个电话号码

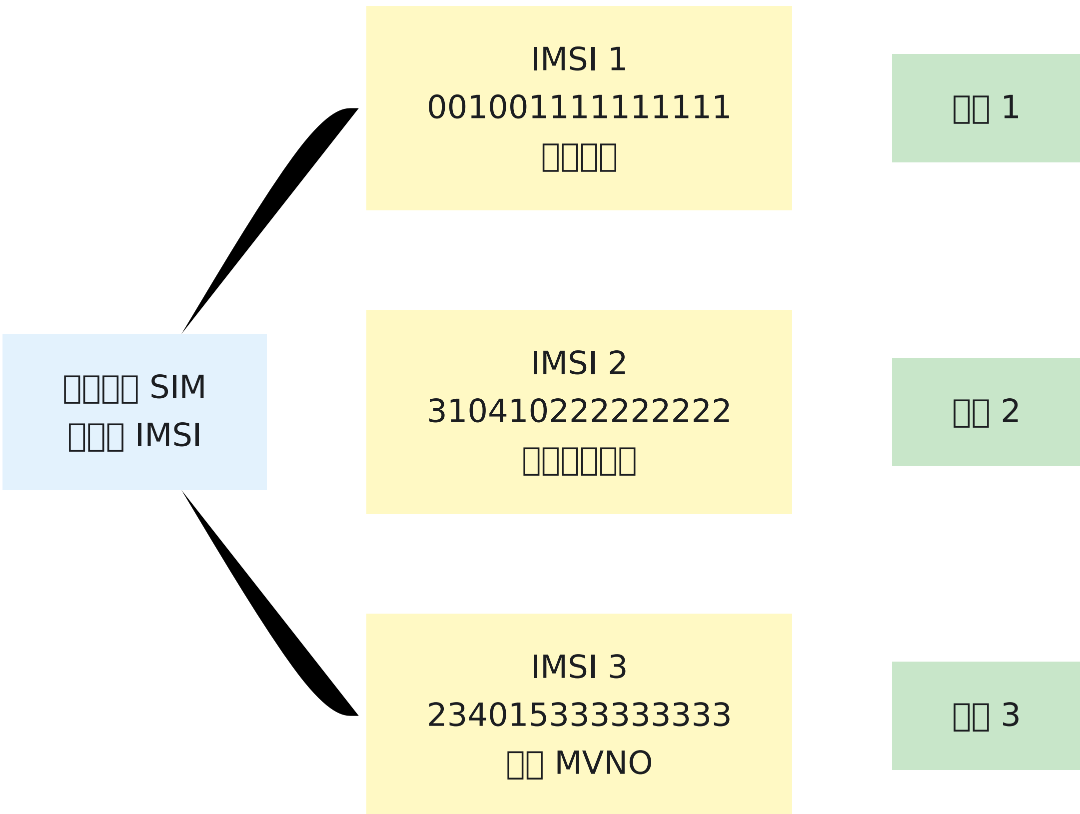
单个用户（通过一个 IMSI 识别）可以分配多个 MSISDN（电话号码）。所有号码在同一设备上响铃，并共享相同的服务配置文件。



□ IMSI SIM □□

□□ SIM → □□ IMSI

□□□□ SIM □□□□□□□ IMSI□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ MVNO □□□□□□□



多 **MSISDN**：多个电话号码

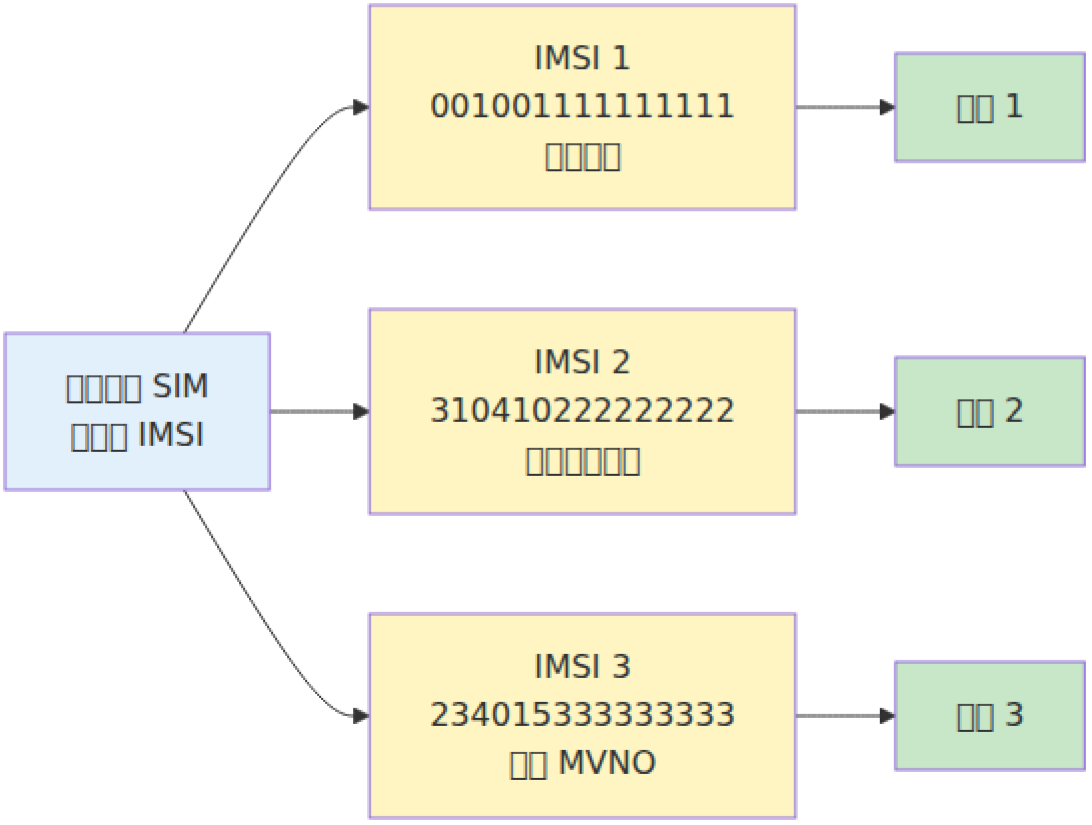
工作原理

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 都包含在 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** - 用户与其 SIM 卡绑定的单个 IMSI
- 多个 **MSISDN** - 用户可以拥有多个电话号码
- **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��件（EPC、IMS、漫游）

数据模型

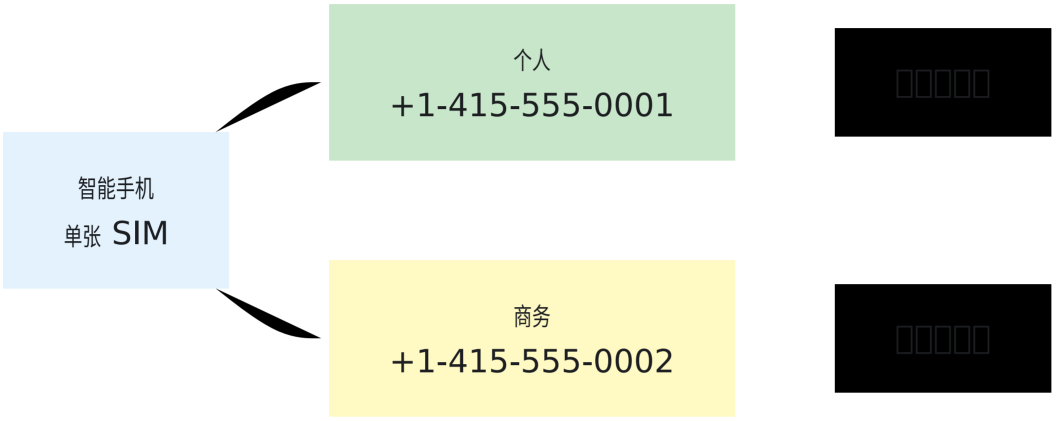


重要提示：一个 MSISDN 只能同时分配给一个用户。然而，一个用户可以拥有多个 MSISDN。

使用案例

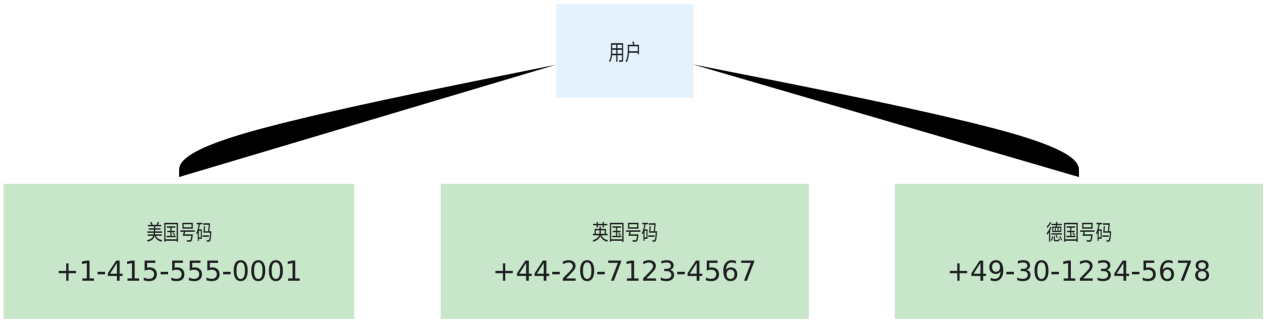
1. 商务和个人线路

一个用户在同一设备上拥有商务和个人电话号码：



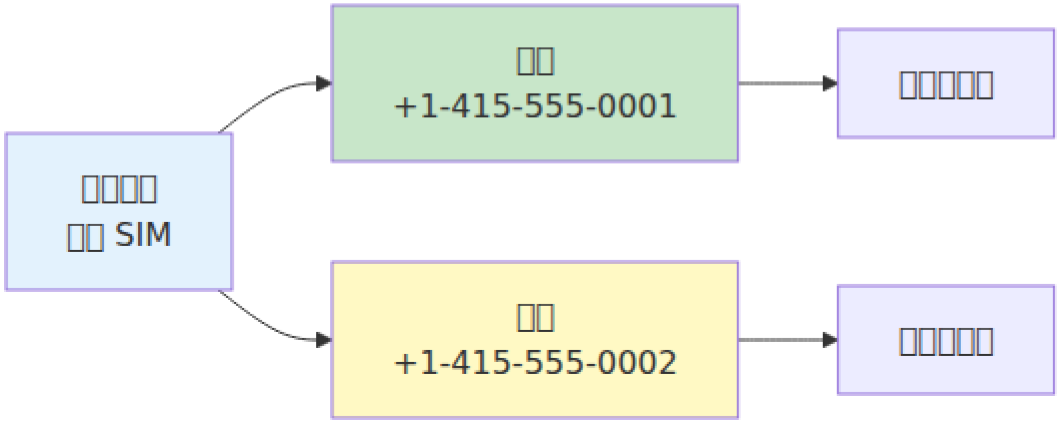
2. 国际号码

一个经常旅行的用户在多个国家拥有号码：



3. 家庭计划

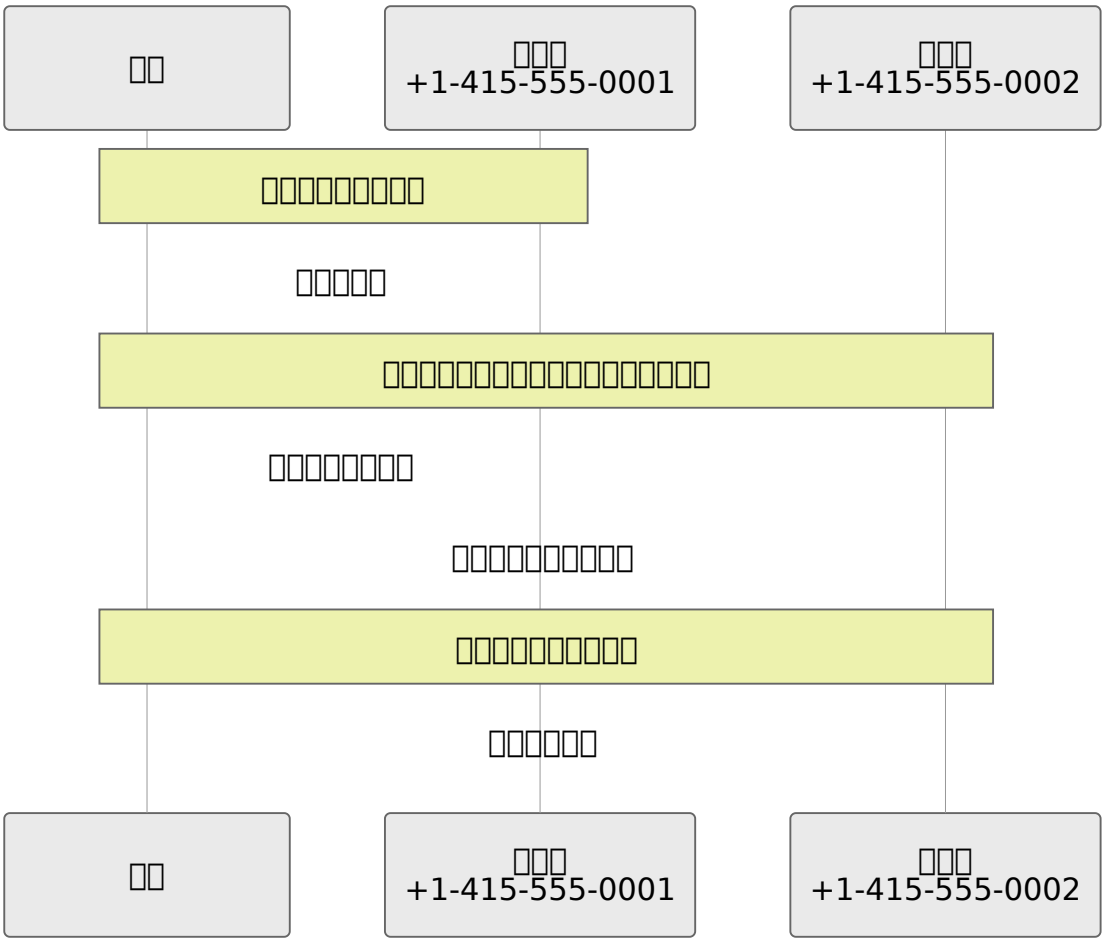
一位家长管理多个家庭成员的号码：



□□□ □ 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

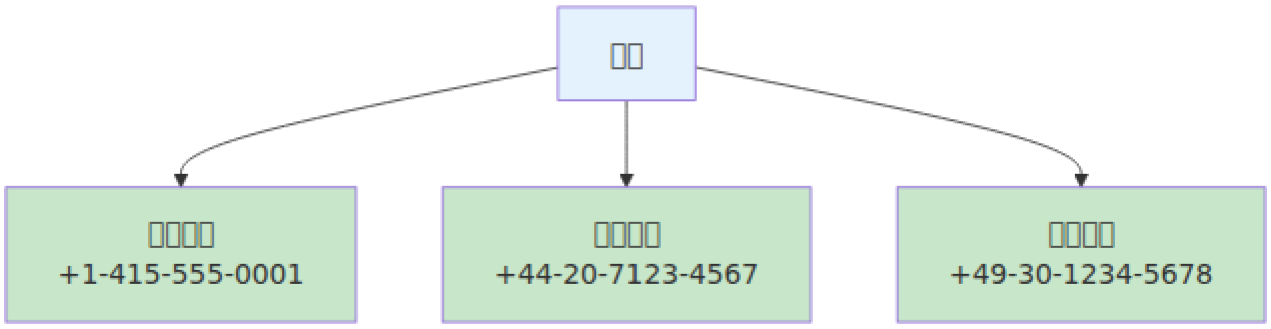
分配通过数据库中的连接表完成。

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. 在连接表中插入记录，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

□□□询数据库以检索用户及其所有链接的 MSISDN：

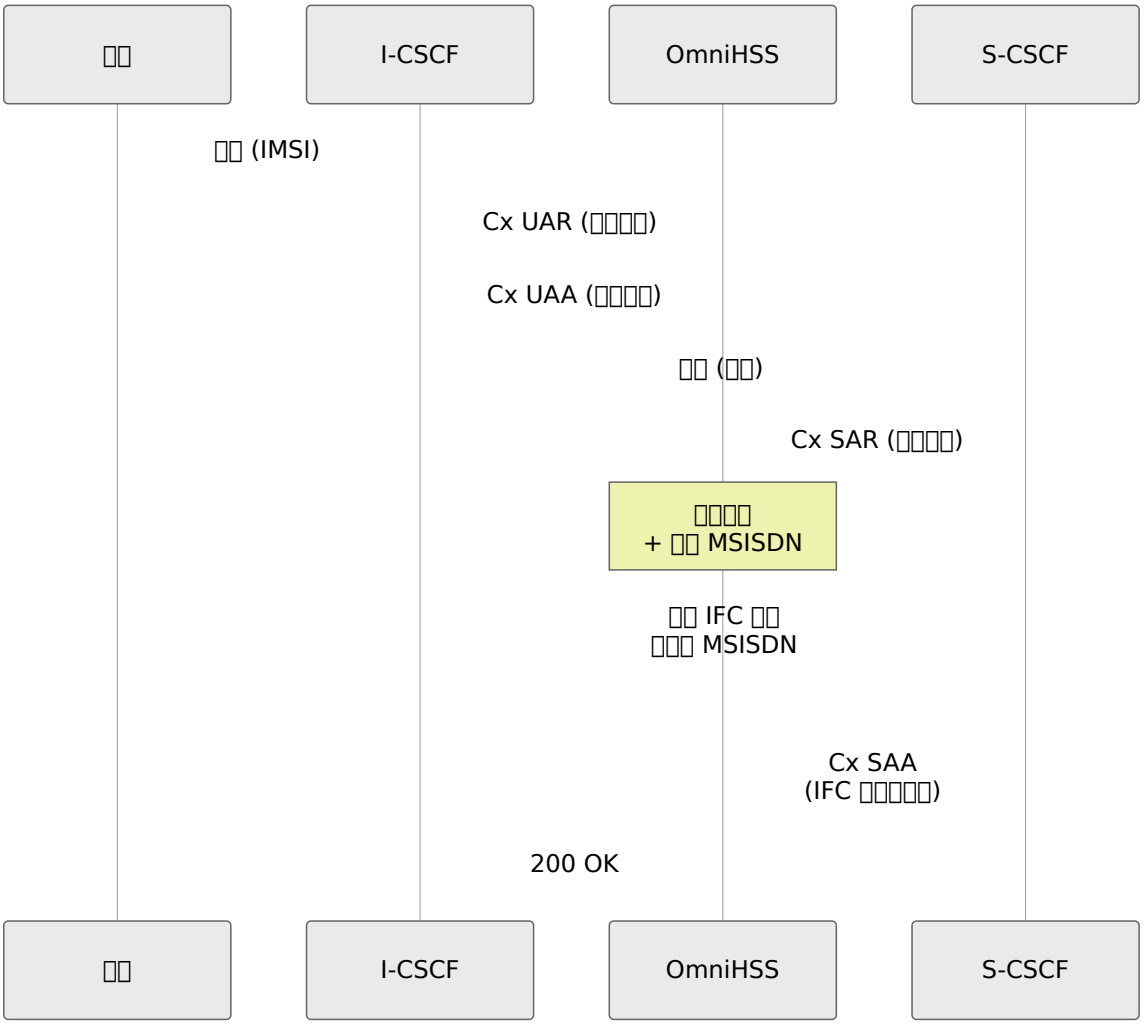
- 将用户表与连接表连接
- 将连接表与 msisdn 表连接
- 按用户分组结果，以查看所有电话号码

□□□□□□□ ID□IMSI □□□□□□□ MSISDN □□□

IMS □□

IMS □□

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



IFC □□□□

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

□□ **IFC □□□**

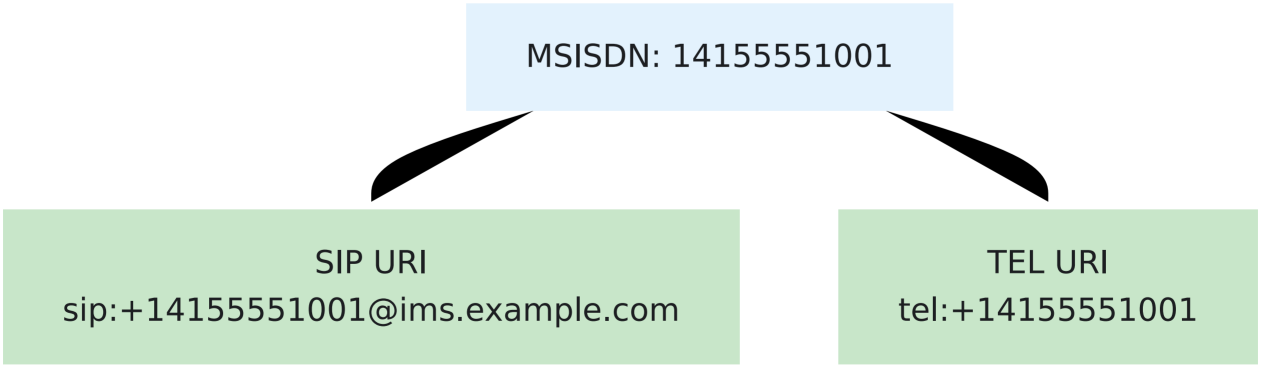
```
<ServiceProfile>
  <PublicIdentity>
    <Identity>sip:
{{imsi}}@ims.mnc{{mnc}}.mcc{{mcc}}.3gppnetwork.org</Identity>
  </PublicIdentity>
  <!-- 为每个 MSISDN 重复 -->
  <PublicIdentity>
    <Identity>sip:+14155551001@ims.example.com</Identity>
  </PublicIdentity>
  <PublicIdentity>
    <Identity>tel:+14155551001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

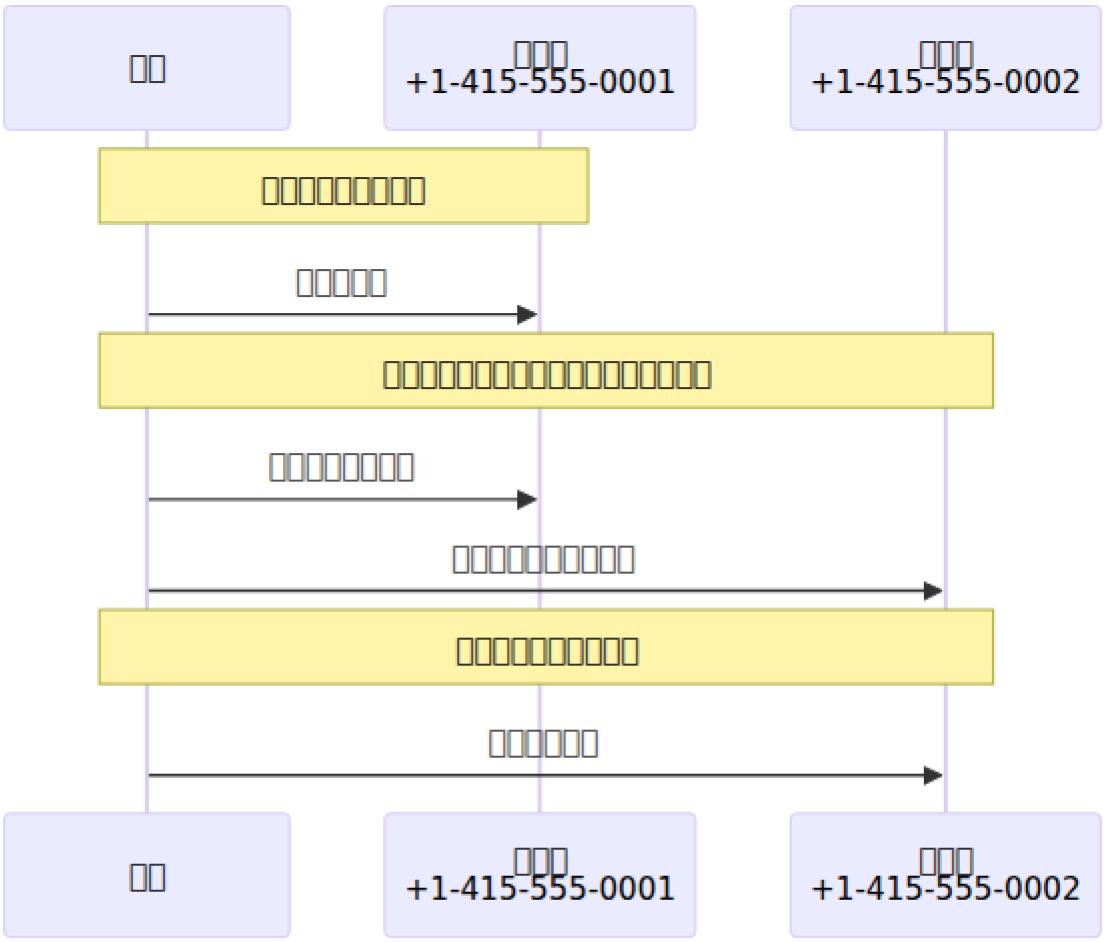
公共身份

每个 MSISDN 通常会生成两个 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` 头部指示正在使用的用户号码。

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
- 对第二个号码的呼叫失败

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 日志以获取注册的公共身份

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 是否已链接到其他用户
    - 解决方案：首先删除现有分配，然后创建新的分配
2. **MSISDN** 不存在：

- 查询数据库以验证 MSISDN 记录□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
- 拨打次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□否适用于所有身份
   - 检查□否需要特定号码的特殊路由
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：通过 **MSISDN** 的 **API** 查找返回错误的用户

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。如果返回多个或错误的用户，则连接表中的数据不正确，��要进行更正。

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

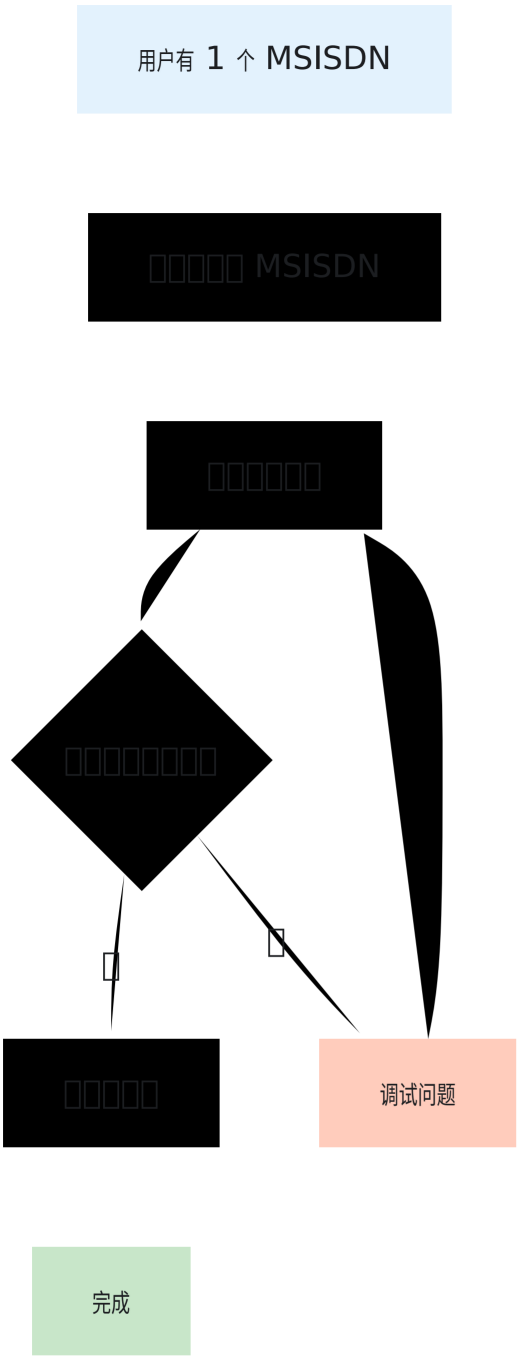
- 在用户自定义属性中记录主号码与次号码
- 在迁移时顺序迁移号码，以避免服务中断
- 在交付给客户之前测试所有号码

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

从单个 MSISDN 迁移到多 MSISDN 时:



多 **IMSI SIM**：多个网络身份

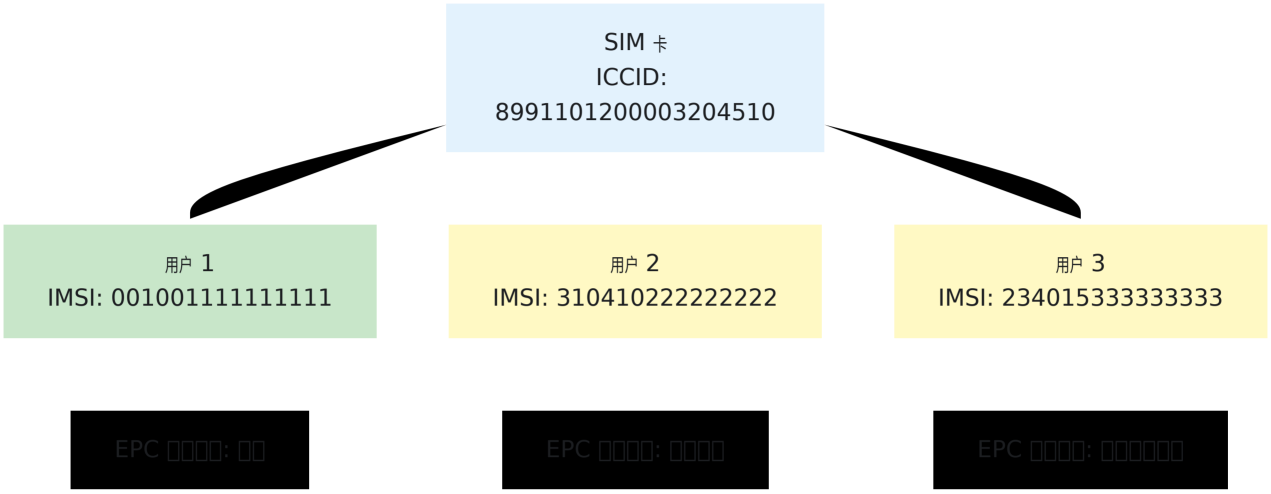
工作原理

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，通常根据位置或网络可用性自动进行切换。

重要提示：任何时候只能激活一个 **IMSI**。当备在同一 SIM 卡上切换到不同的 IMSI 时，HSS 将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：



使用案例

1. 国际漫游优化

- 本地 IMSI: 001-001（本地网络费率）
- 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

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

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/低成本备用
- 关键设备保持连接

4. 合规性

- 不同的 IMSI 用��不同的合规区域
- 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
- 每个网络不同的安全凭证

独立服务配置文件

- 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 引用同一张 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

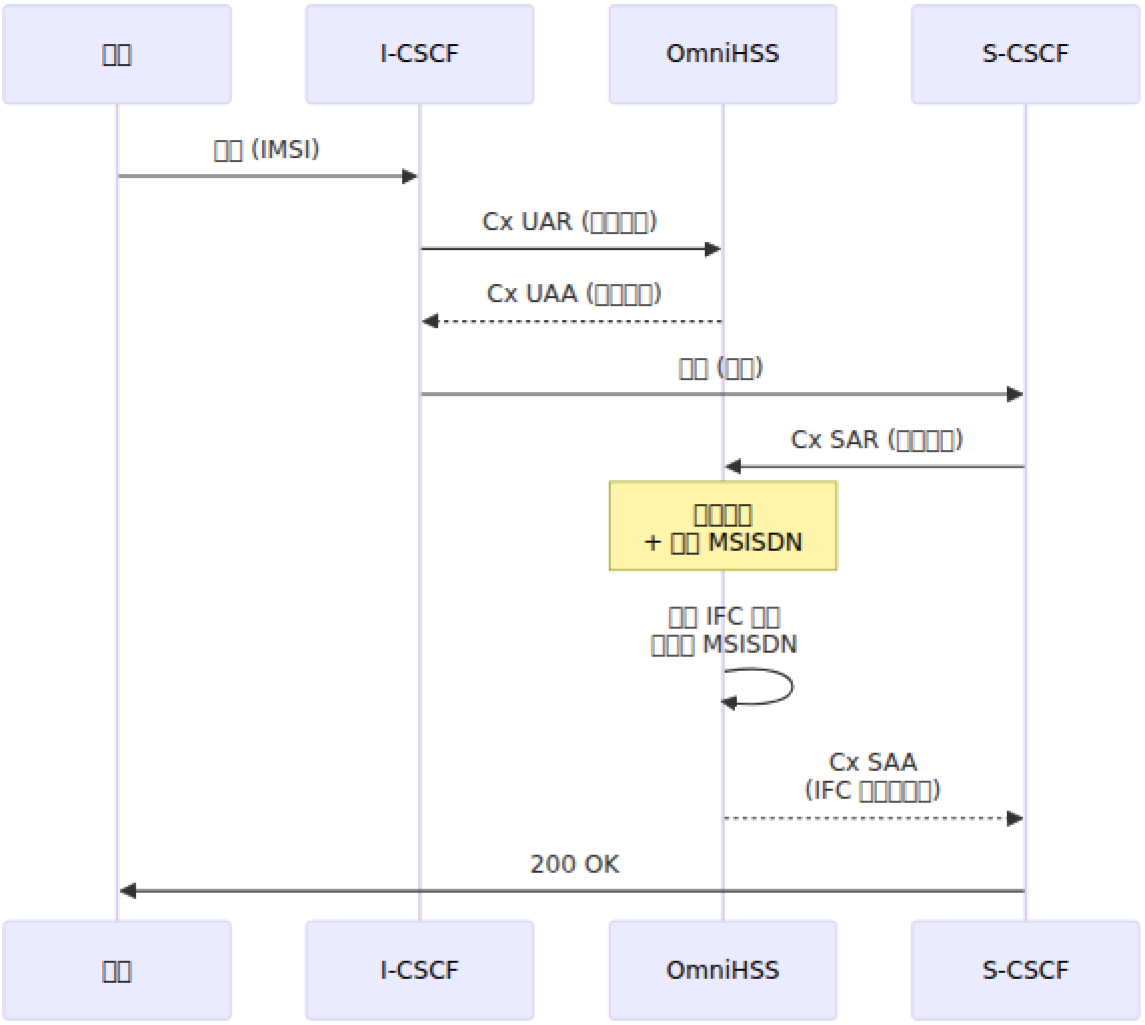
- 设备或 SIM 卡决定使用哪个 IMSI
- 基于可用网络、位置、策略
- HSS 认证设备呈现的任何 IMSI

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（本地网络）
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（本地 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\": 1
  }}"

# 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\": 2
  }}"

# 6. 为 SIM 上的其他 IMSI 重复此过程...
```

认证流程

当多 IMSI 设备连接时:



HSS 不需要知道这□一个多 IMSI SIM——它只需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 通过向 HSS 发送取消位置请求（**CLR**）来自动处理此操作，以注销先前激活的 IMSI。

单个活动 **IMSI** 规则

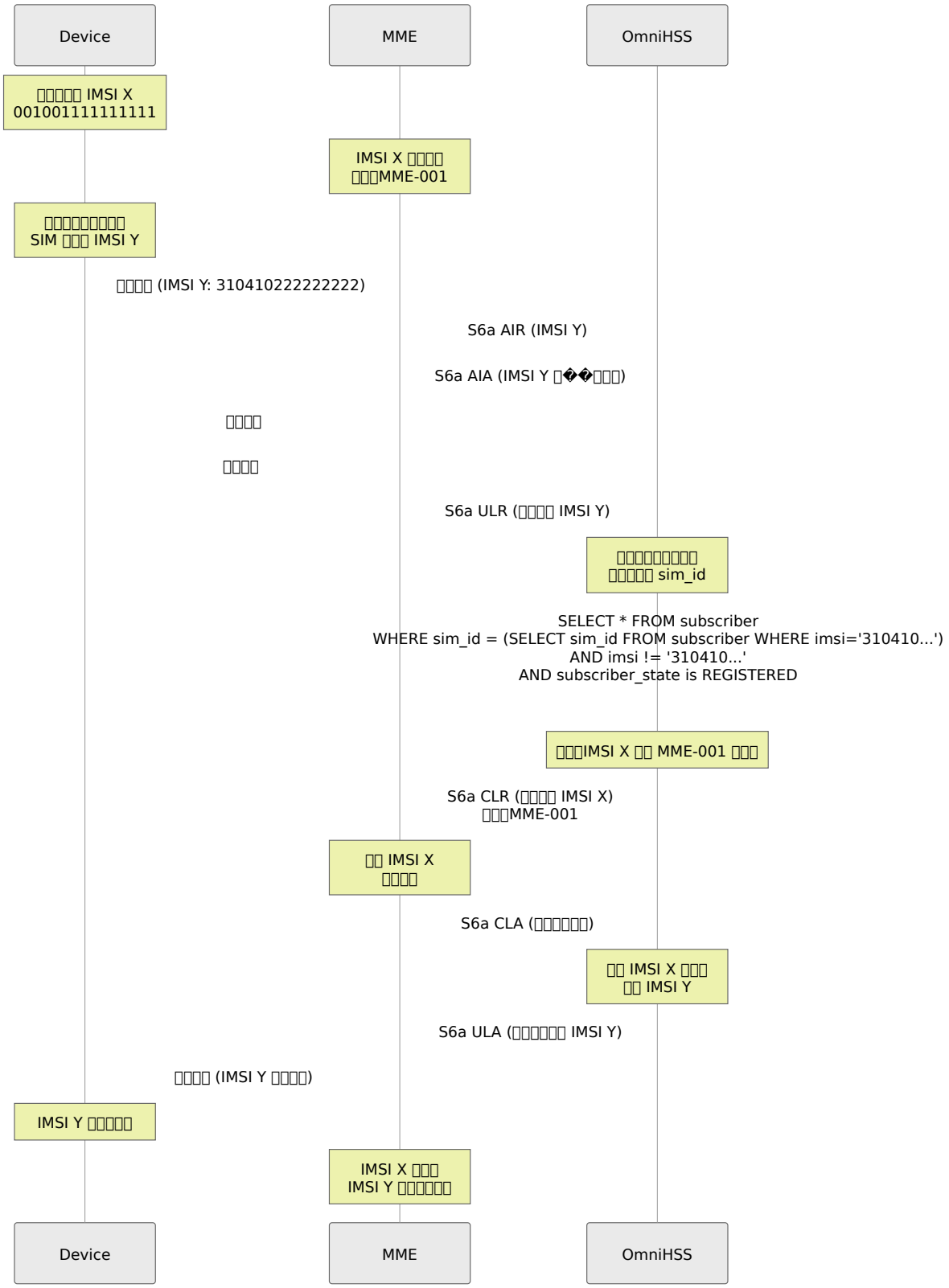
关键概念：每张 SIM 卡只能有一个用户（IMSI）在任何时候处于活动状态。

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

- HSS □□□□□□ □□□□□□□ □□□ **IMSI X**

□□□□ IMSI □□□□□□□□□□□□□□□□□□□□□□□□

IMSI □□□□



这很重要的原因

网络完整性:

- 防止同一物理 SIM 的重复注册
- 确保网络资源被正确释放
- 维护准确的用户位置信息

计费准确性:

- 任何时候只有一个 IMSI 被收取网络访问费用
- 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���准确

IMSI 切换触发器

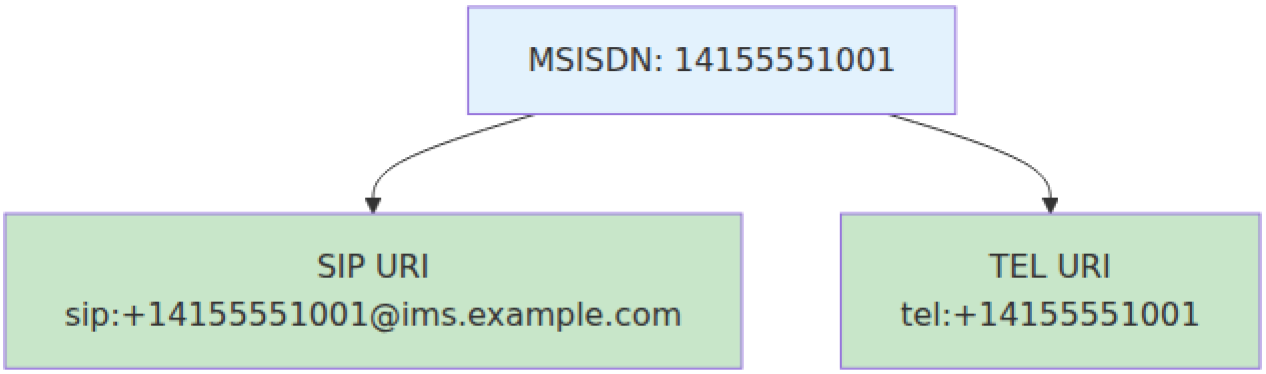
设备/SIM 决定何时切换 IMSI，基于:

1. 网络可用性
    - 本地 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

- 切换到漫游费用较低的 IMSI
- 使用本地 IMSI 避免漫游费用

IMS 考虑

相同的取消位置请求行为适用于 IMS 注册：



□□影响

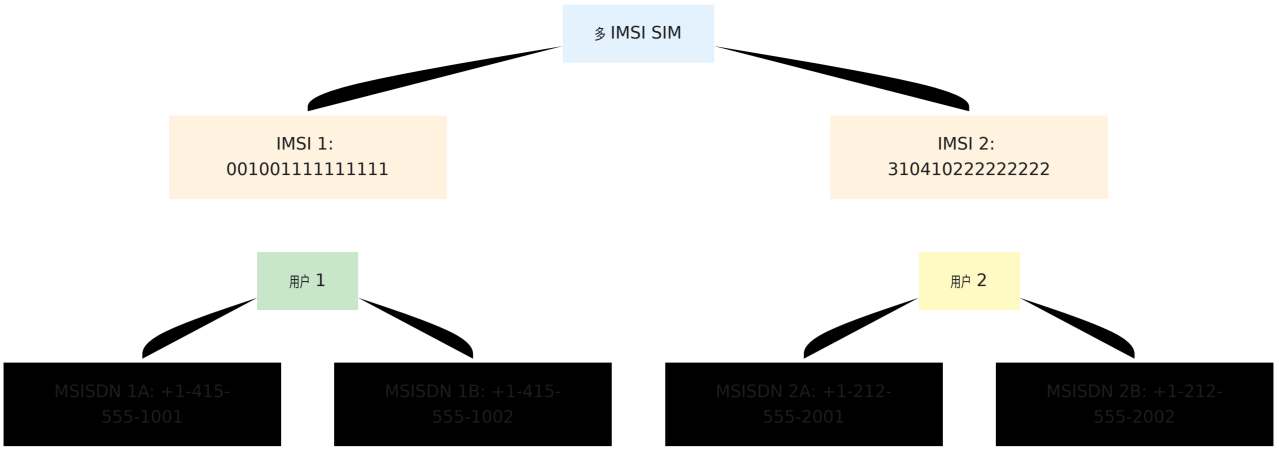
对于□作人员：

1. 用户显示离线：当 IMSI 切换时，旧 IMSI 将在 HSS 中显示为“注销”。这□正常行为。
2. 一张 **SIM** 的两个用户记录：多 IMSI SIM 将有多个用户记录共享相同的 `sim_id`。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 IMSI
   - 验证当前网络使用的正确 IMSI
   - 确认每张 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：一张 SIM 上多个 IMSI，每个 IMSI 有多个 MSISDN。



示例用例：

- ###地网络（**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

当设备在本国时，使用 IMSI 1 及其 MSISDN。当在美国漫游时，切换到 IMSI 2 及其针对美国网络优化的不同 MSISDN。

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
```

响应包括所有链接的 MSISDN。

故障排除多 **IMSI**

设备无法使用第二个 **IMSI** 附加：

1. 验证该 IMSI 否存在第二个用户记录
2. 检查该 IMSI 的 key_set 否配置正确
3. 验证 EPC 配置文件否已分配
4. 确认漫游规则否允许附加

设备意外切换 **IMSI**：

- 这由设备/SIM 逻辑控制的，而不 HSS
- HSS 认证呈现的任何 IMSI
- 检查设备 IMSI 选择设置

故障排除多 **MSISDN**

第二个号码不响铃：

1. 验证 MSISDN 否在连接表中链接
2. 检查 IMS 配置文件模板否包含 `{{msisdns}}` 变量
3. 确认 IMS 注册否包含所有公共身份
4. 查看 S-CSCF 日志以获取注册的身份

出站呼叫仅显示一个号码：

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

优势总结

多 **MSISDN** 优势

- ✓ 一张 SIM，多条电话号码
- ✓ 分开的商务和个人线路
- ✓ 国际本地存在
- ✓ 简化设备管理
- ✓ 所有号码共享相同的数据服务
- ✓ 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
- ✓ 合规性
- ✓ 跨网络的服务连续性

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

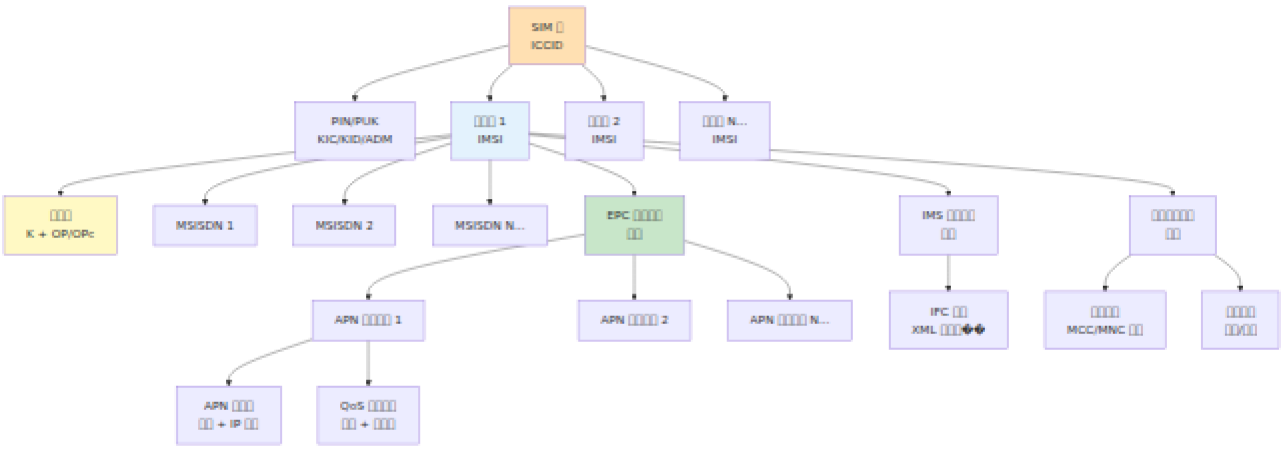
← 返回操作指南

OmniHSS 配置文件管理

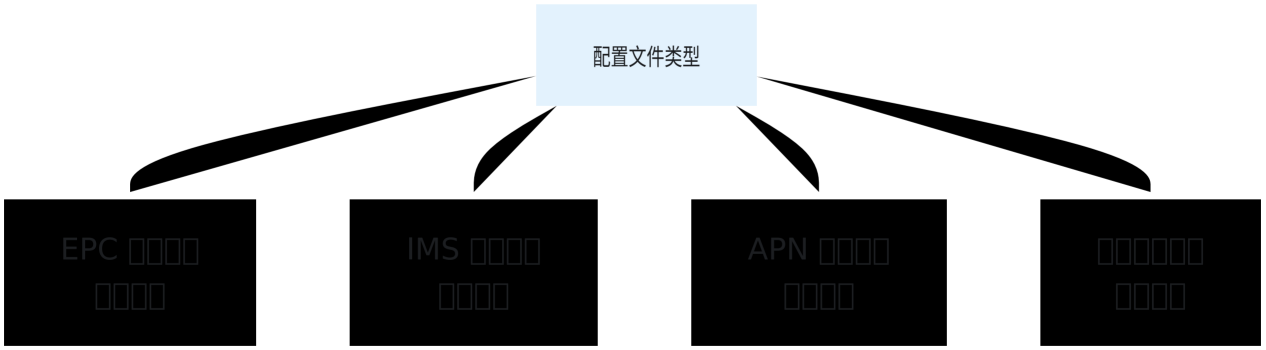
← 返回操作指南

概述

OmniHSS 使用 配置文件 来定义订阅者的服务特性。配置文件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，从而简化配置并确保一致性。



配置文件类型



EPC 配置文件

EPC（演进分组核心）配置文件定义 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 定时器	54 秒（典型）

创建 EPC 配置文件

```
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 配置文件模板

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 配置文件

IMS 配置文件定义语音服务特性，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>"
    }
  }'
```

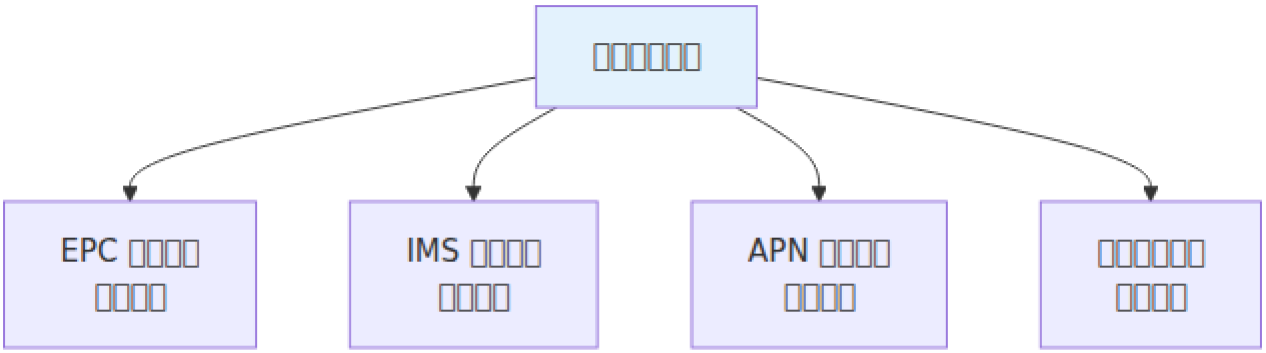
IFC □□□□

```
<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 配置文件

APN（接入点名称）配置文件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 配置文件

定义服务质量参数。

QCI（**QoS** 类标识符）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

创建完整 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 配置文件
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 配置文件
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 配置文件

APN 通过 `join_epc_profile_to_apn_profile` 表与 EPC 配置文件关联。

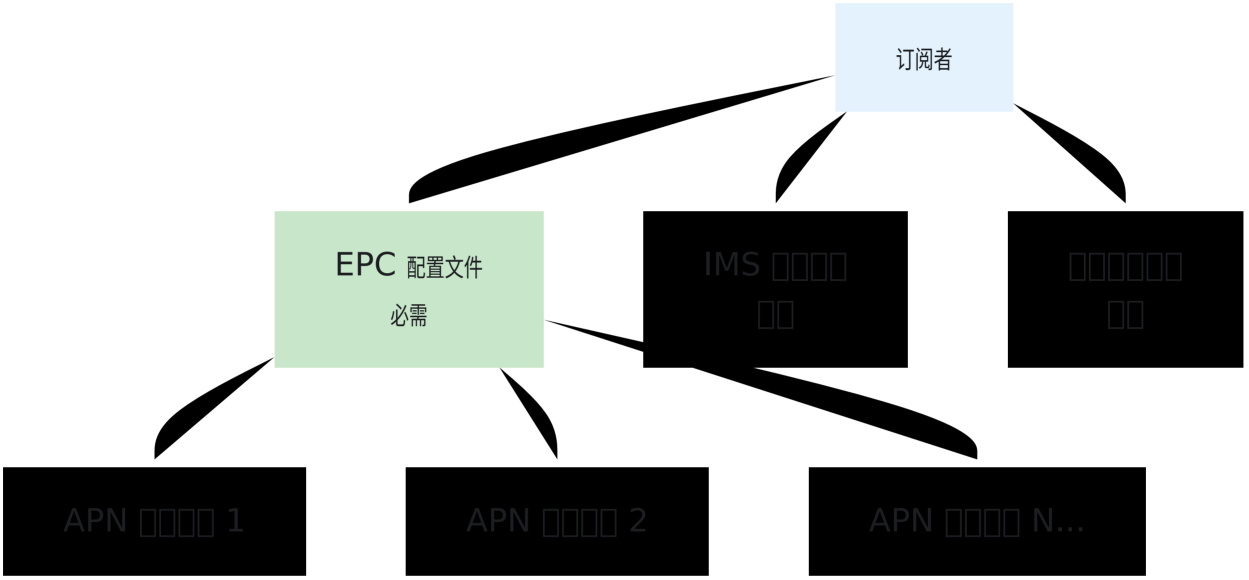
将记录插入连接表，以将 APN 配置文件 ID 链接到 EPC 配置文件 ID。多个 APN 配置文件可以分配给一个 EPC 配置文件。

漫游配置文件

请参阅 漫游控制指南 中的详细文档。

配置文件分配

订阅者配置文件关系



将配置文件分配给订阅者

```
# 在创建订阅者时分配 EPC 和 IMS 配置文件
curl -k -X POST https://hss.example.com:8443/api/subscriber \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "imsi": "001001123456789",
      "key_set_id": 1,
      "epc_profile_id": 1,
      "ims_profile_id": 1,
      "roaming_profile_id": 1
    }
  }'

# 更新订阅者配置文件
curl -k -X PUT https://hss.example.com:8443/api/subscriber/1 \
  -H "Content-Type: application/json" \
  -d '{
    "subscriber": {
      "epc_profile_id": 2
    }
  }'
```

配置文件管理最佳实践

设计原则

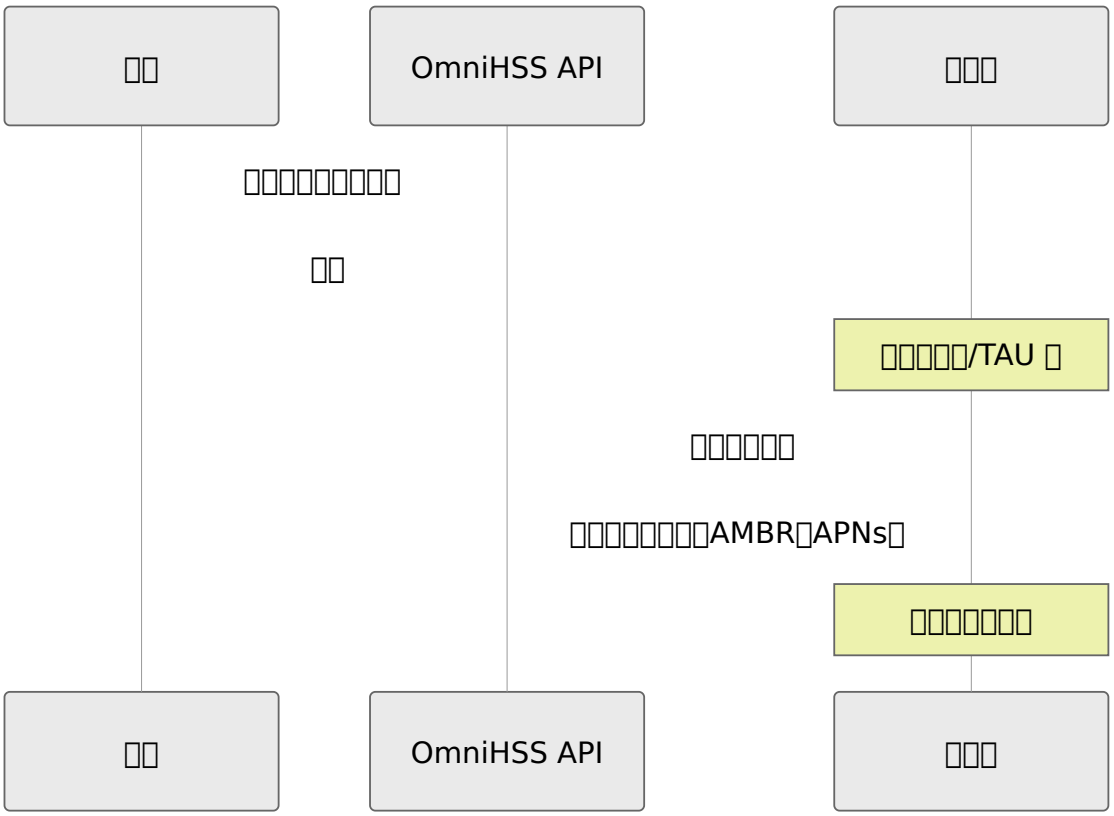
1. 创建标准配置文件 - 定义常见服务层级（基础��标准、高级）
2. 重用配置文件 - 将相同配置文件分配给多个订阅者
3. 记录变更 - 跟踪配置文件修改
4. 生产前测试 - 首先验证配置文件与测试订阅者的兼容性

□□□□□□□□□

```
[□□□□□]-[□□]-[□□]

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

□□□□□□

□□□□□□□□□□□□□□



□□: □□□□□□□□□□□□□□□□

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

故障排除配置文件问题

订阅者未获得预期速度：

1. 检查分配的 EPC 配置文件 AMBR 值
2. 检查 APN QoS 配置文件 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. 验证是否分配了 IMS 配置文件
2. 检查 IFC 模板 XML 的有效性
3. 查看 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 配置文件是否链接到 EPC 配置文件
2. 检查 APN 标识符是否与网络请求匹配
3. 查看 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

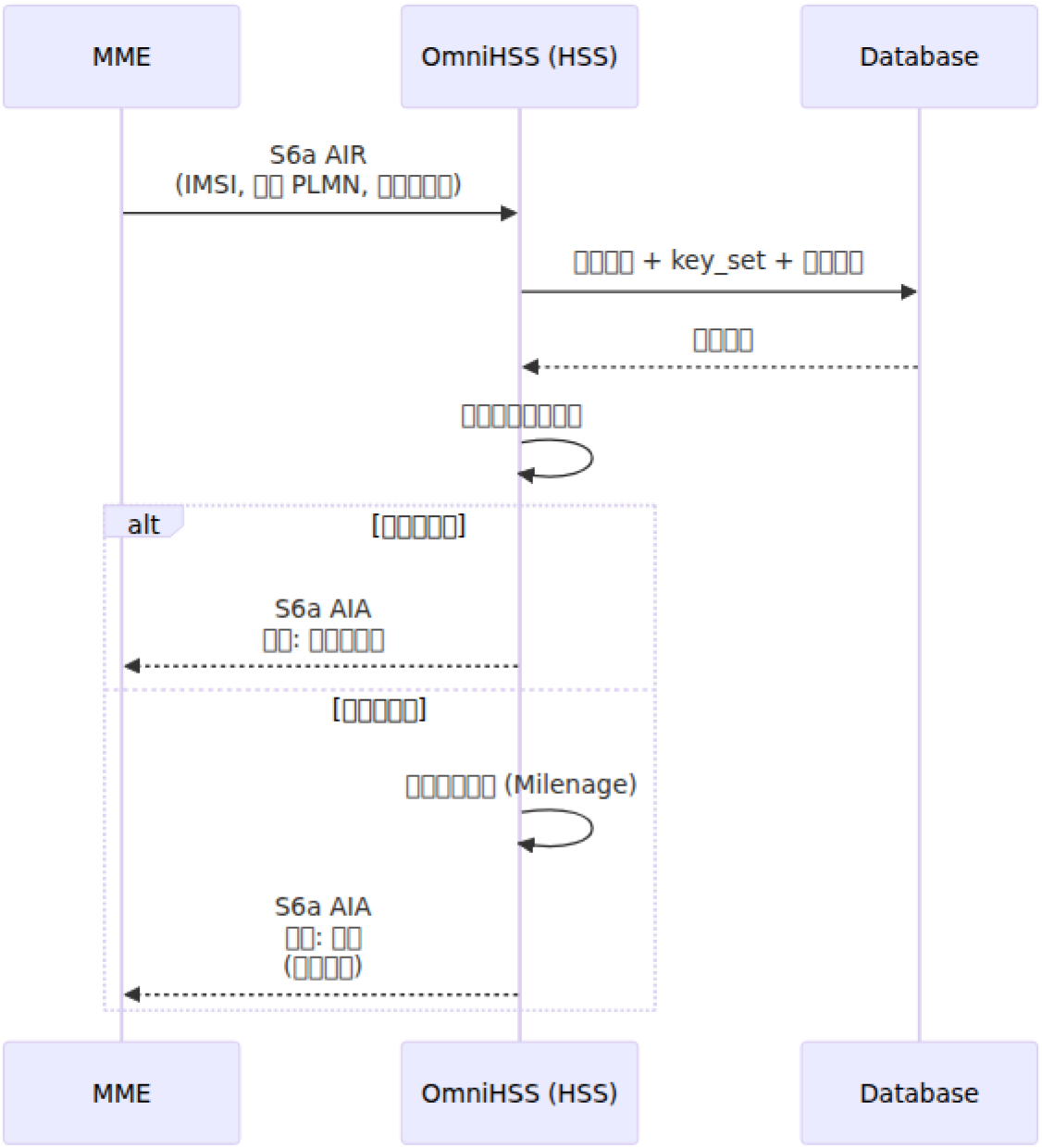
概述

本文档详细说明了 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。

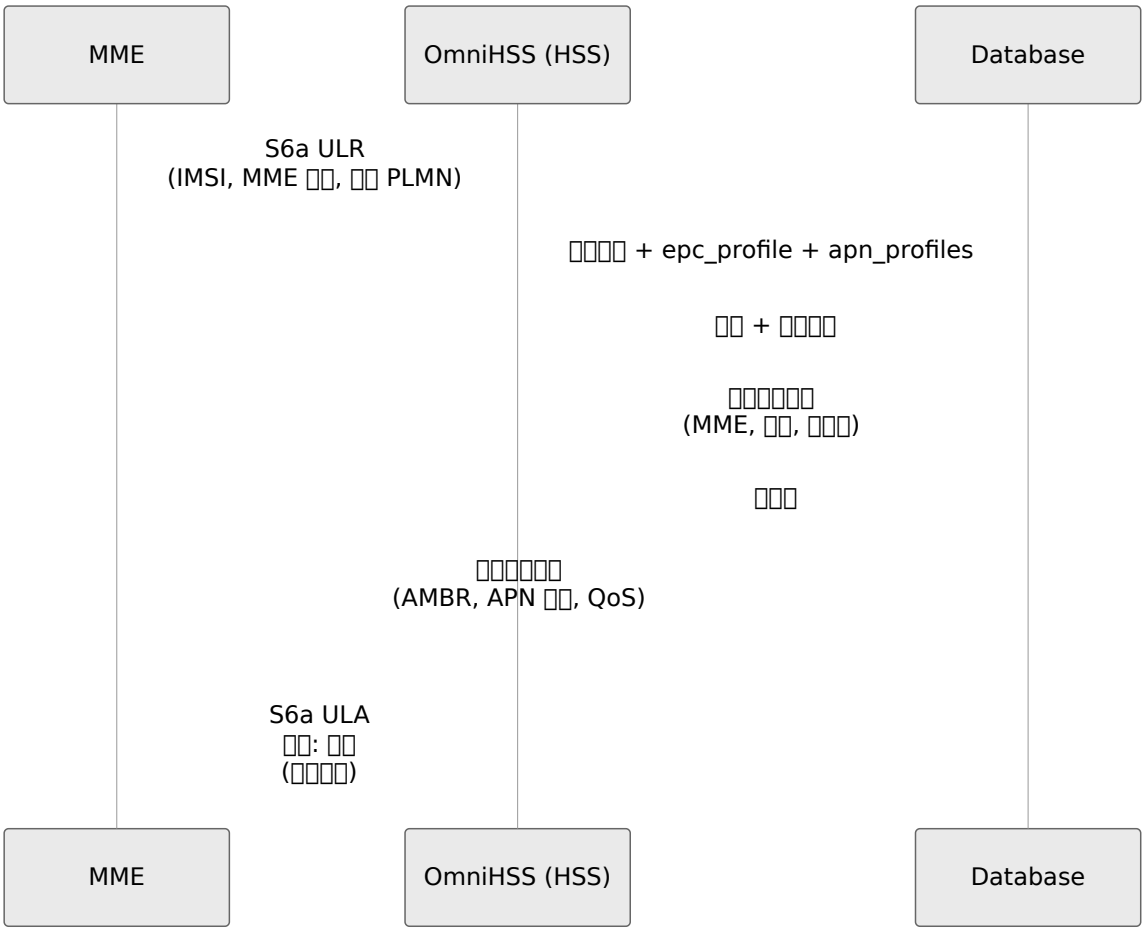


关键 **AVP:**

- 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。

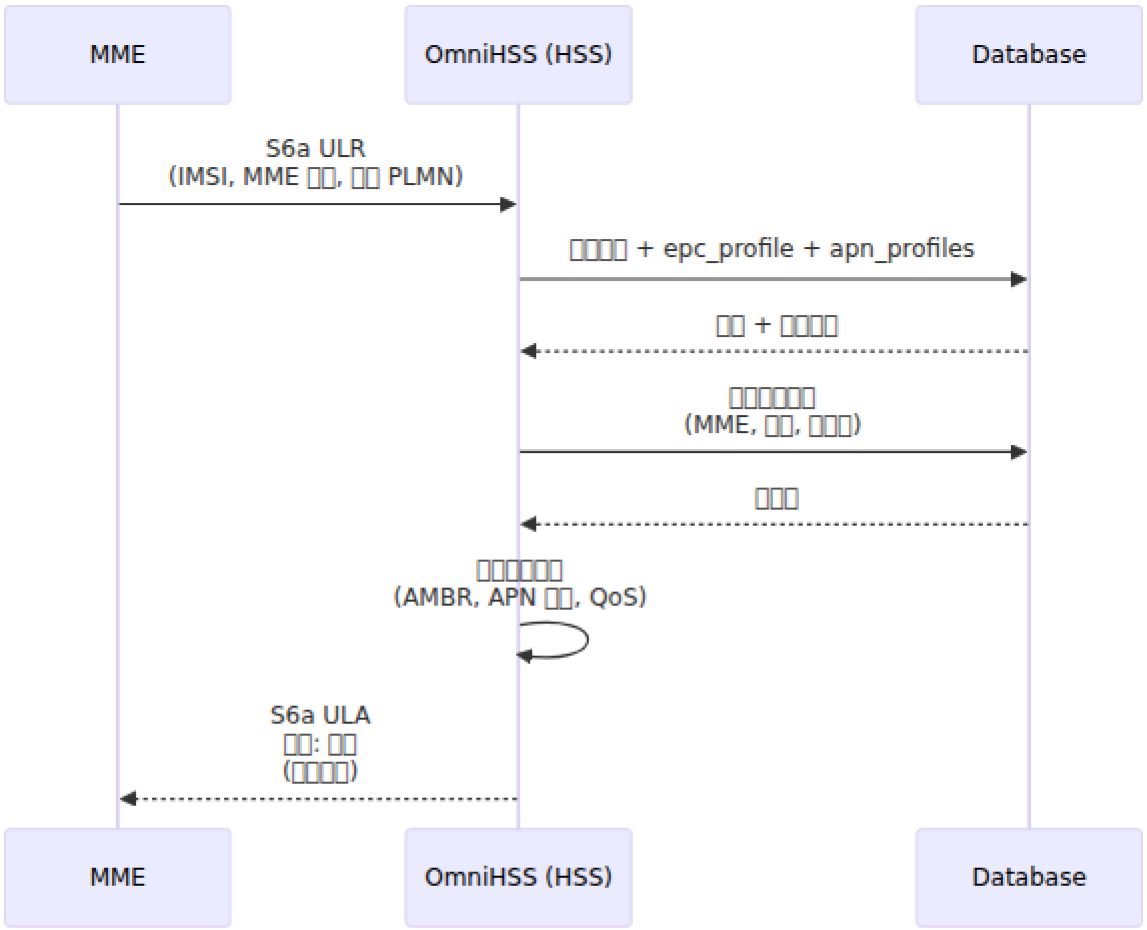


□□ **AVP:**

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

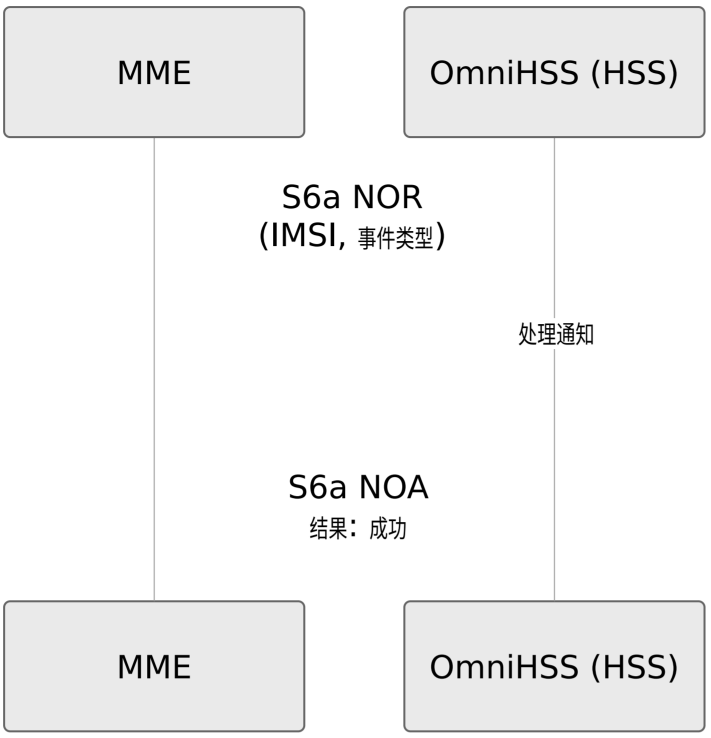
□□ UE □□ (PUR/PUA)

MME □□ HSS □□□□□□□□□□□□



通知请求 **(NOR/NOA)**

MME 通知 HSS 各种事件。

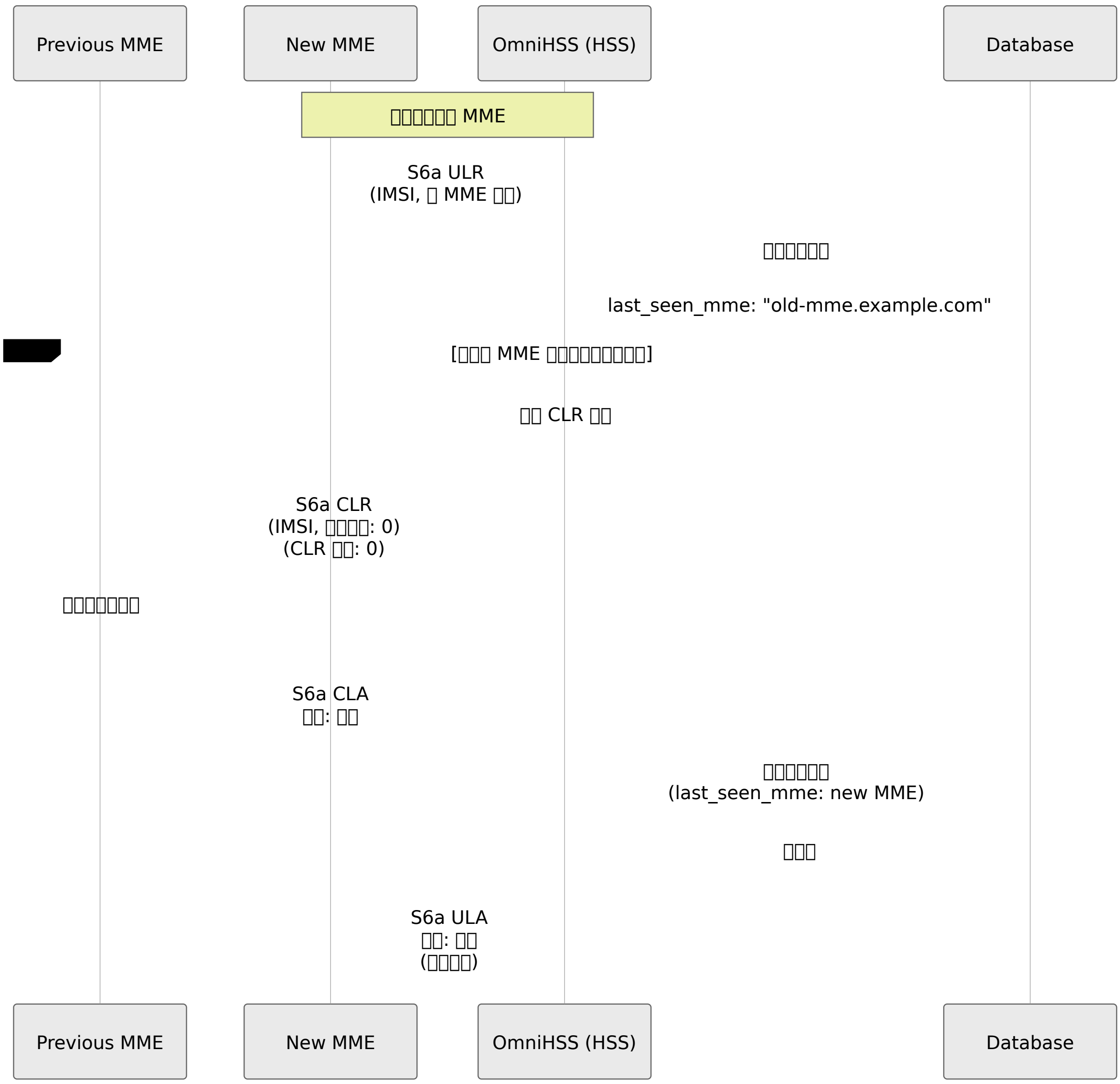


取消位置请求 **(CLR/CLA)**

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**)**

当用户从新的 MME 执行更新位置请求时，OmniHSS 会自动删除之前的 MME 发送 CLR 以清理过时的注册。

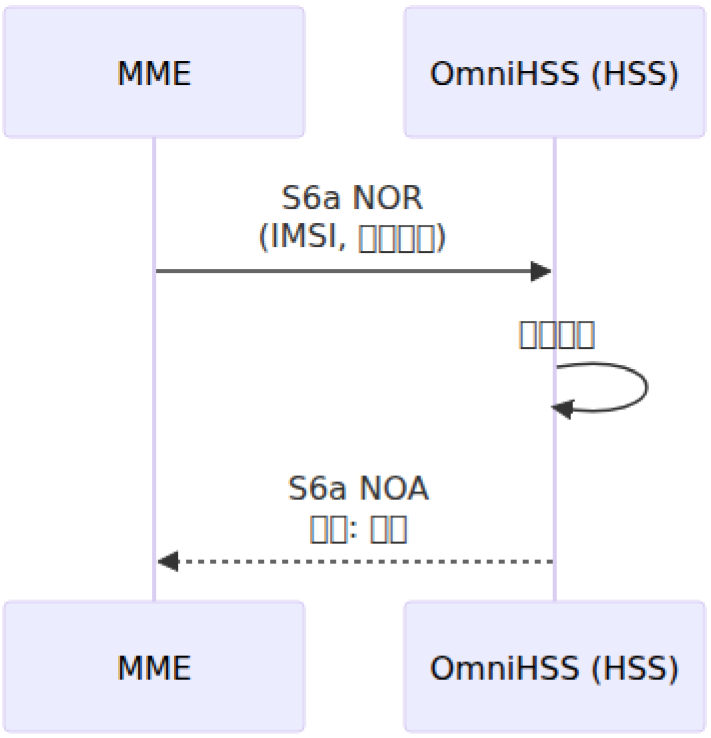


□□ AVP (□□ CLR):

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

□□□ CLR (API □□)

□□□□□□□□□□ API □□ CLR□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



关键 **AVP (**程序化 **CLR):**

- 用户名: 用户的 IMSI
- 目标主机: 最近看到的 MME 主机名
- 目标域: 最近看到的 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 支持按 3GPP TS 29.272 的多种取消类型:

类型	值	描述	用例
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 一个位掩码，具有以下字段：

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

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 按每个用户 **(IMSI)** 跟踪 MME 注册，而不按 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"
```

当该用户移动到新的 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/转移期间:

```
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 仍然在 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** 用于多 **MSISDN** 用户

结果:

- 发送一个 **CLR** 到用户的 last_seen_mme
- 所有 **MSISDN** 与该 IMSI 关联的都被有效分离

- IMSI □跟踪 MME 注册的主键

重要注意事项

1. **IMSI** 是关键**:** CLR 操作始终□ 按 **IMSI**，而不□按 MSISDN。`subscriber_state` 表按用户 (IMSI) 跟踪 `last_seen_mme`。

2. 原子□作**:** 每个用户一次只能在一个 MME 注册。自动 CLR 确保通过清理旧注册来实现这一点。

3. 如果没有之前的 **MME** 则不发送 **CLR:** 如果 `last_seen_mme` 为 `nil`（用户从未注册），则在 ULR 期间不发送 CLR。

4. 包含订阅数据**:** 自动 CLR（在 ULR 期间）包括完整的 `Subscription-Data` AVP，以帮助旧 MME 正确清理上下文。

5. 异步**:** CLR □异步发送的（火并忘）。对新 MME 的 ULA 响应不等待��� MME 的 CLA。

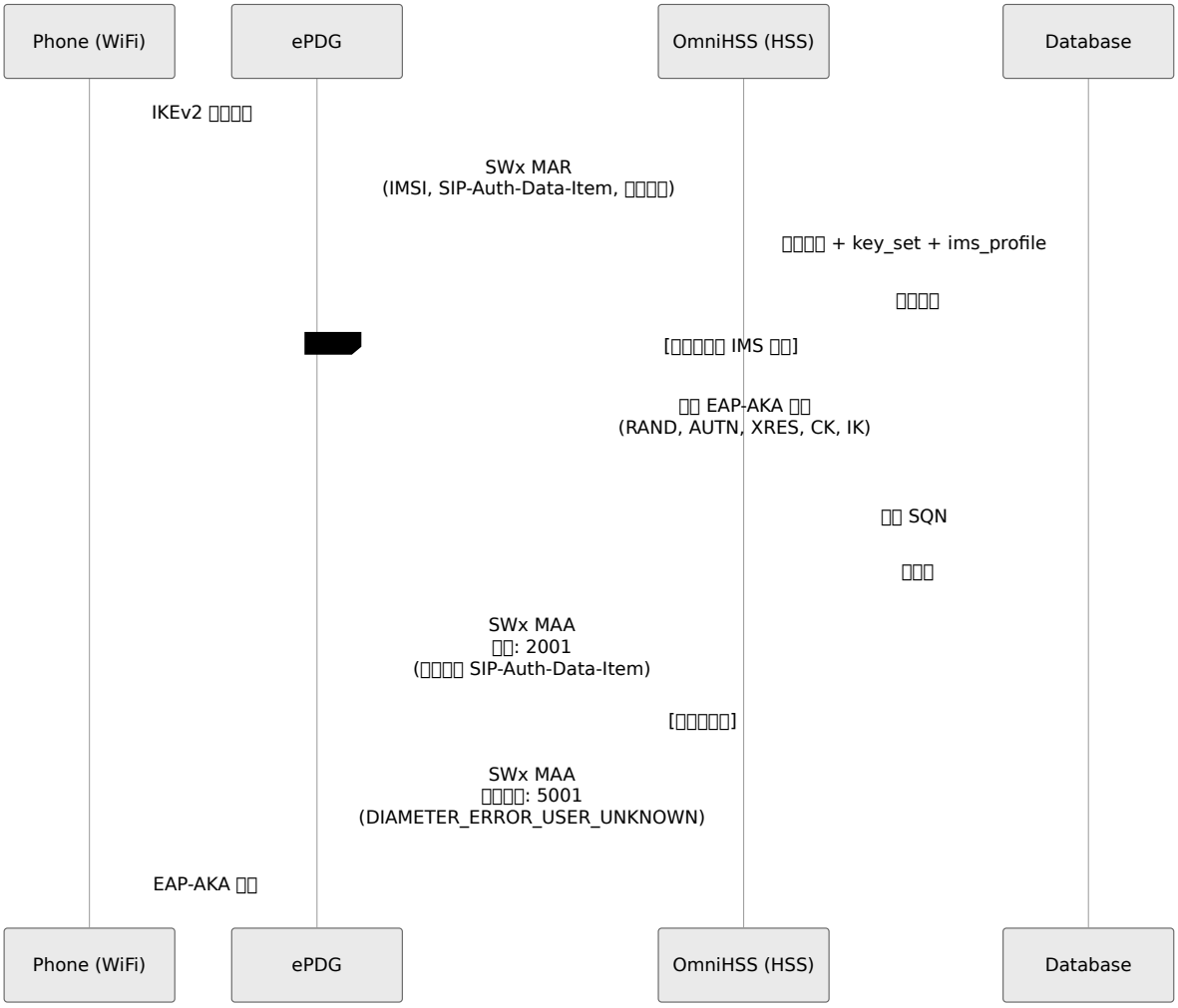
6. **CLA** 处理**:** OmniHSS 接收 CLA 响应，但当前会丢弃它们（在第 398 行为 `:discard`）。这防止了消息循环，并且□标准 HSS 行为。

SWx 接口 (非 3GPP 接入 / WiFi 通话)

SWx 接口通过将 ePDG（演进数据包网关）连接到 HSS，以实现 WiFi 通话的用户认证和授权。

多媒体认证请求 (MAR/MAA)

ePDG 请求 EAP-AKA 认证向量以进行 WiFi 通话注册。



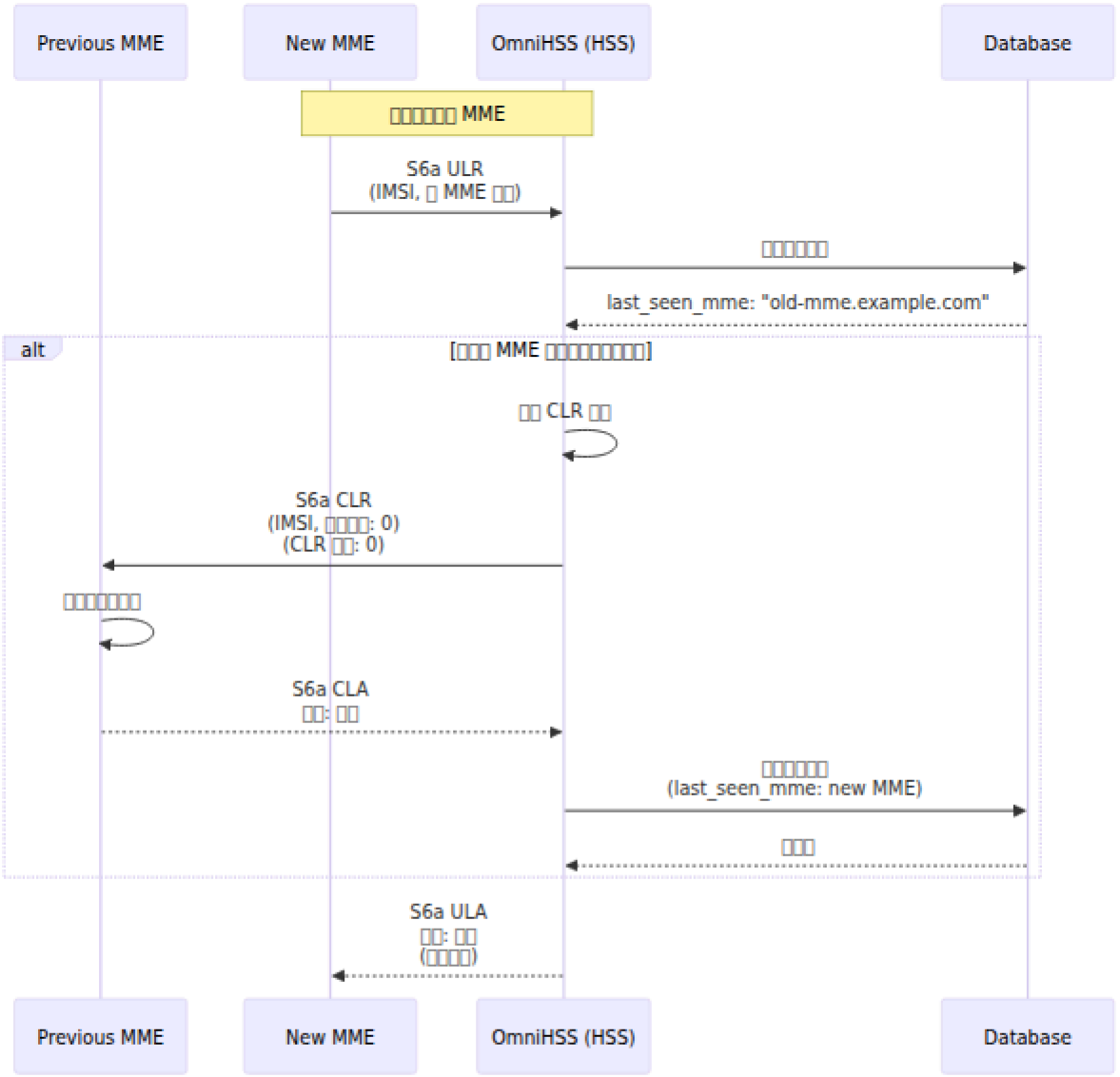
□□ AVP:

- □□: □□□ (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
S6a ULR
(IMSI, MME)
last_seen_mme: "old-mme.example.com"
alt
CLR
S6a CLR
(IMSI, : 0)
(CLR : 0)
S6a CLA
:
(last_seen_mme: new MME)
S6a ULA
:
(Previous MME)
Previous MME
New MME
OmniHSS (HSS)
Database

服务器分配类型：

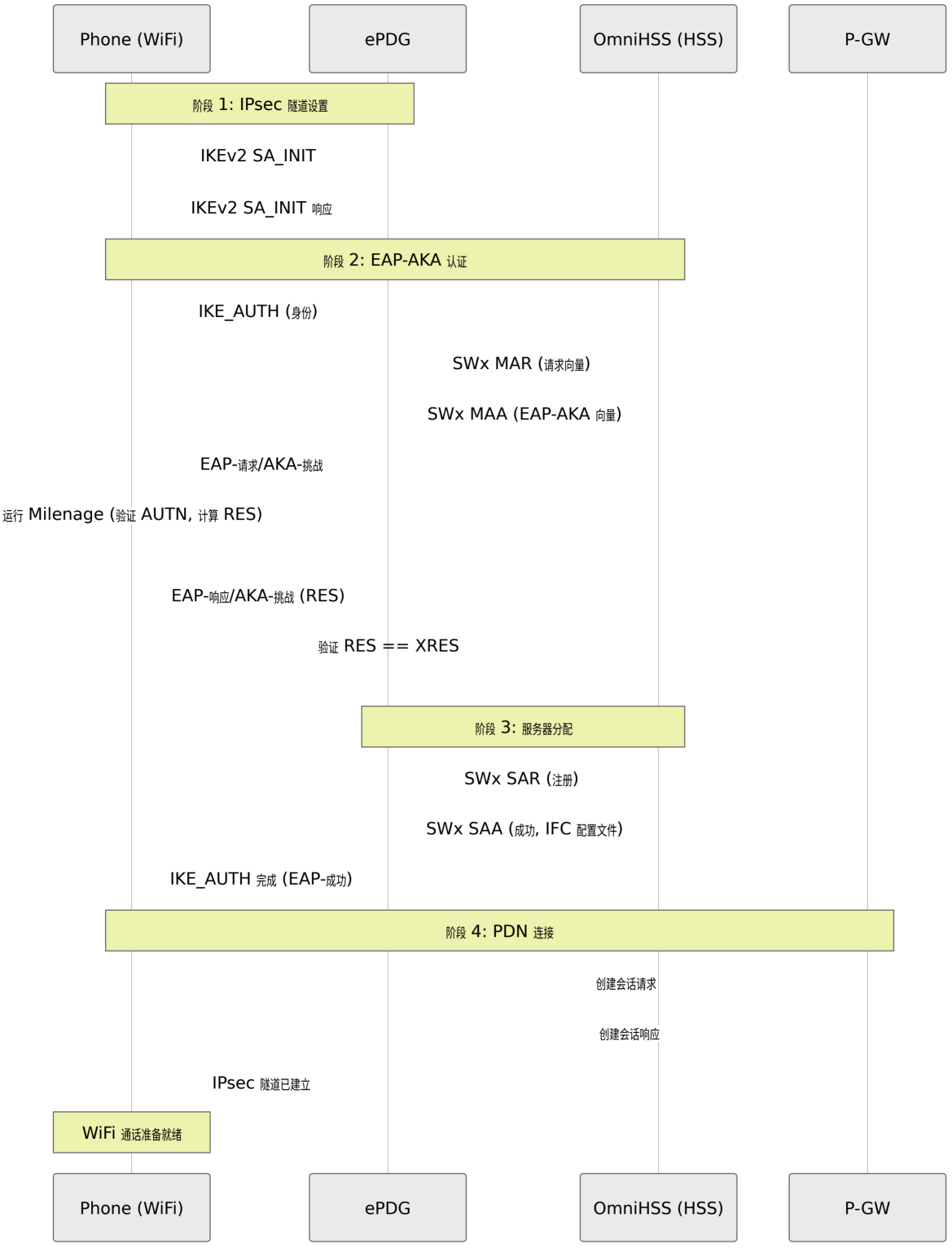
类型	值	描述
NO_ASSIGNMENT	0	仅查询，无状态变化
REGISTRATION	1	初始 WiFi 通话注册
RE_REGISTRATION	2	注册刷新
UNREGISTERED_USER	3	未注册用户的服务
TIMEOUT_DEREGISTRATION	4	注册超时
USER_DEREGISTRATION	5	用户发起的注销
AUTHENTICATION_FAILURE	7	认证失败，清除状态
ADMINISTRATIVE_DEREGISTRATION	8	管理员发起的注销

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 XML), 非 3GPP 用户数据

WiFi 通话注册流程

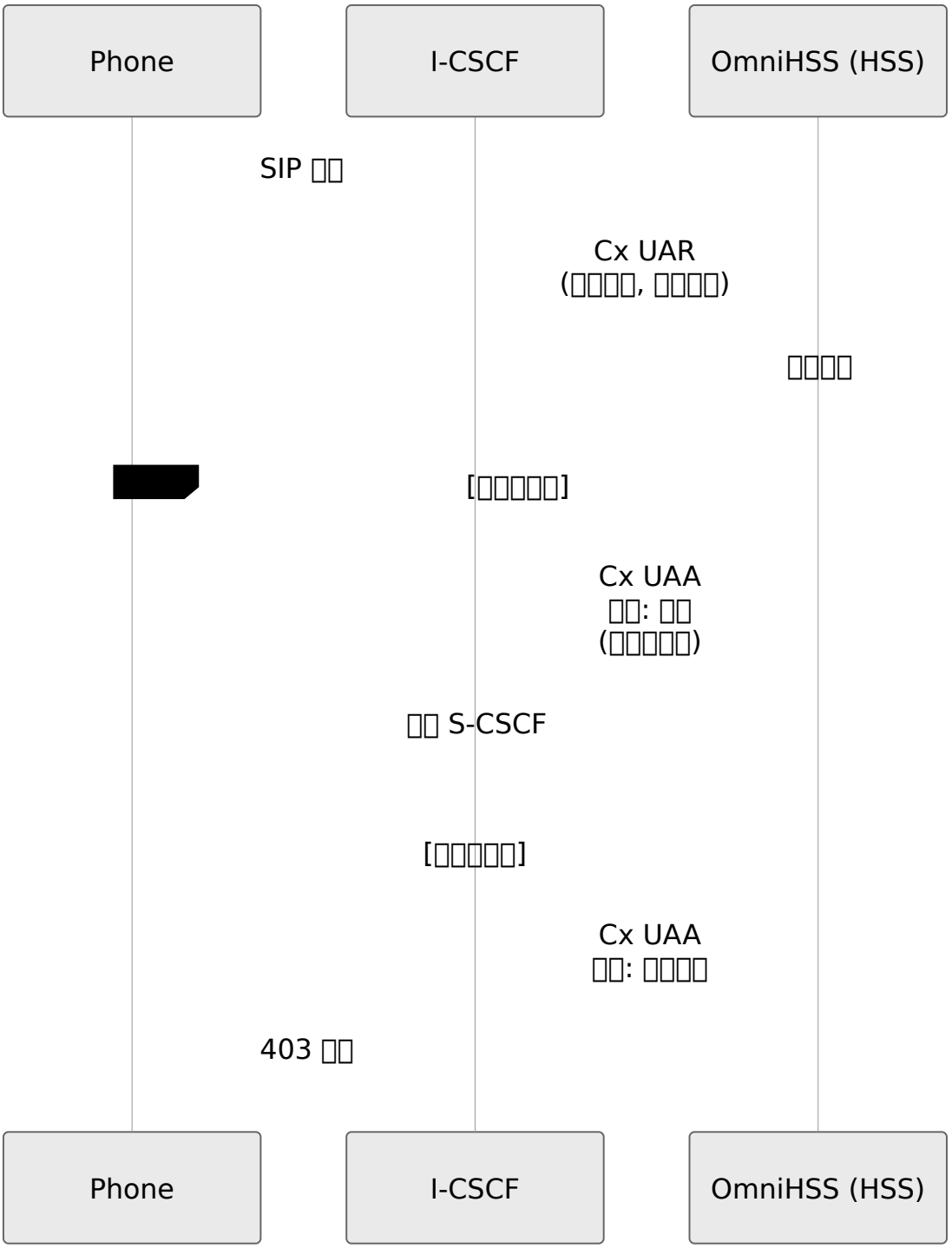
完整的端到端 WiFi 通话注册。


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)

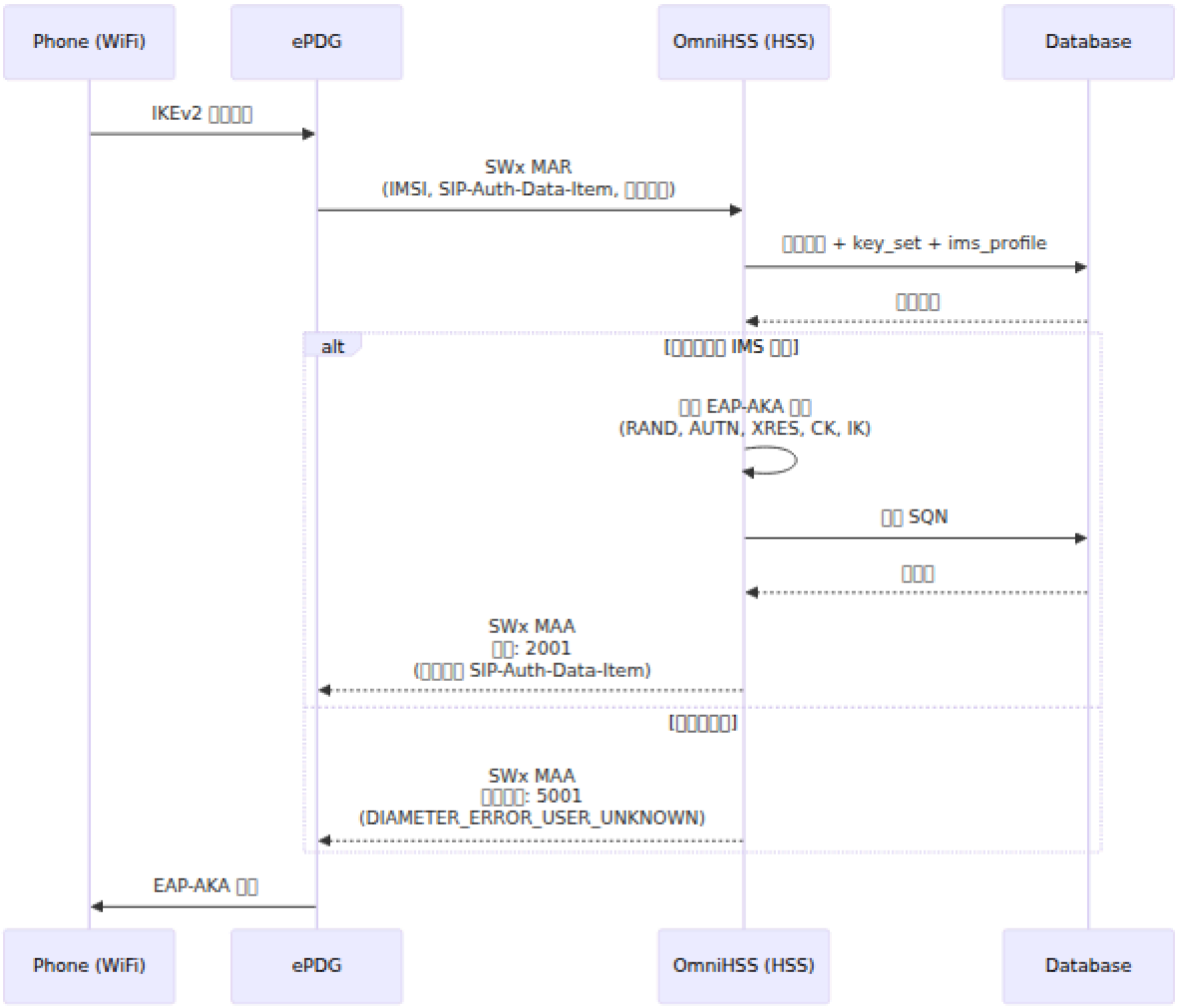
□□□□□□□ (UAR/UAA)

I-CSCF □□□□□□□□□□□□□



服务器分配请求 **(SAR/SAA)**

S-CSCF 注册/注销用户并检索 IMS 配置文件。

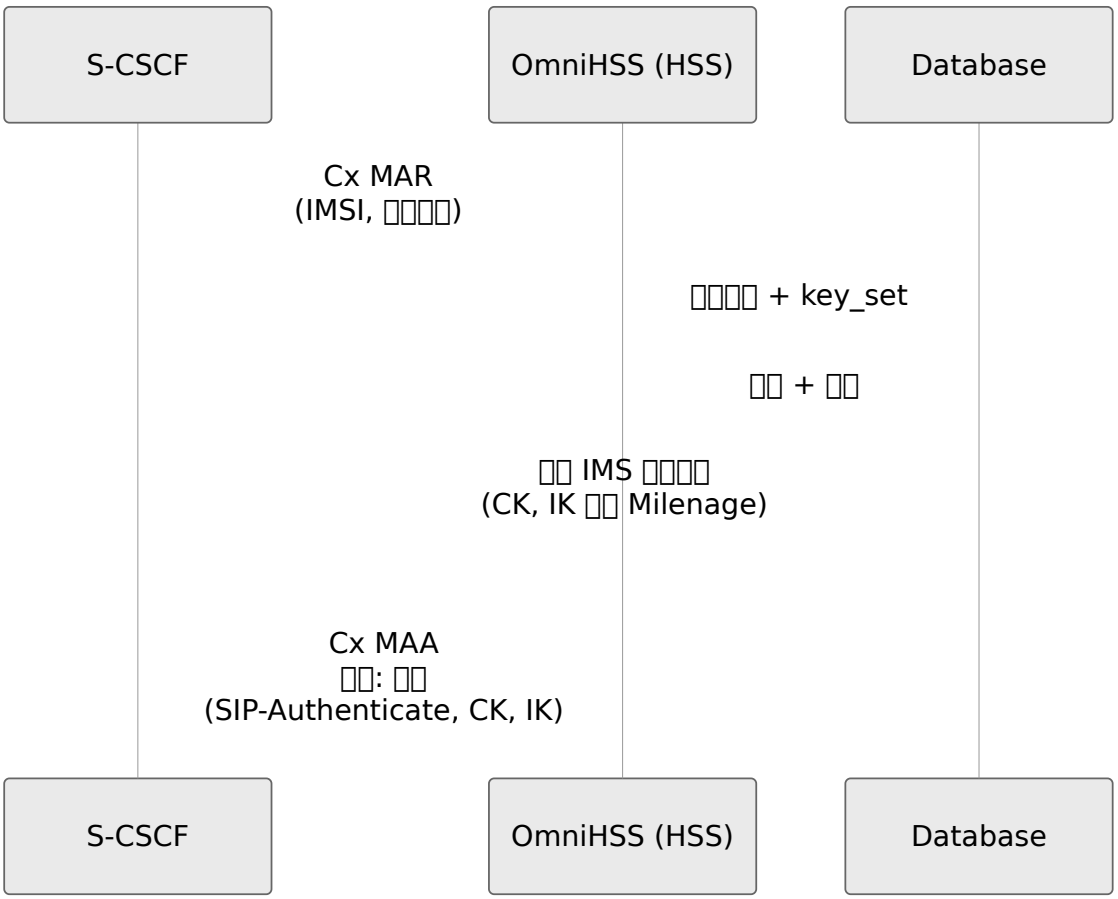


IFC 模板渲染**:**

- `{{imsi}}` → 实际 IMSI
- `{{msisdns}}` → 电话号码列表
- `{{mcc}}, {{mnc}}` → 本地 PLMN 代码

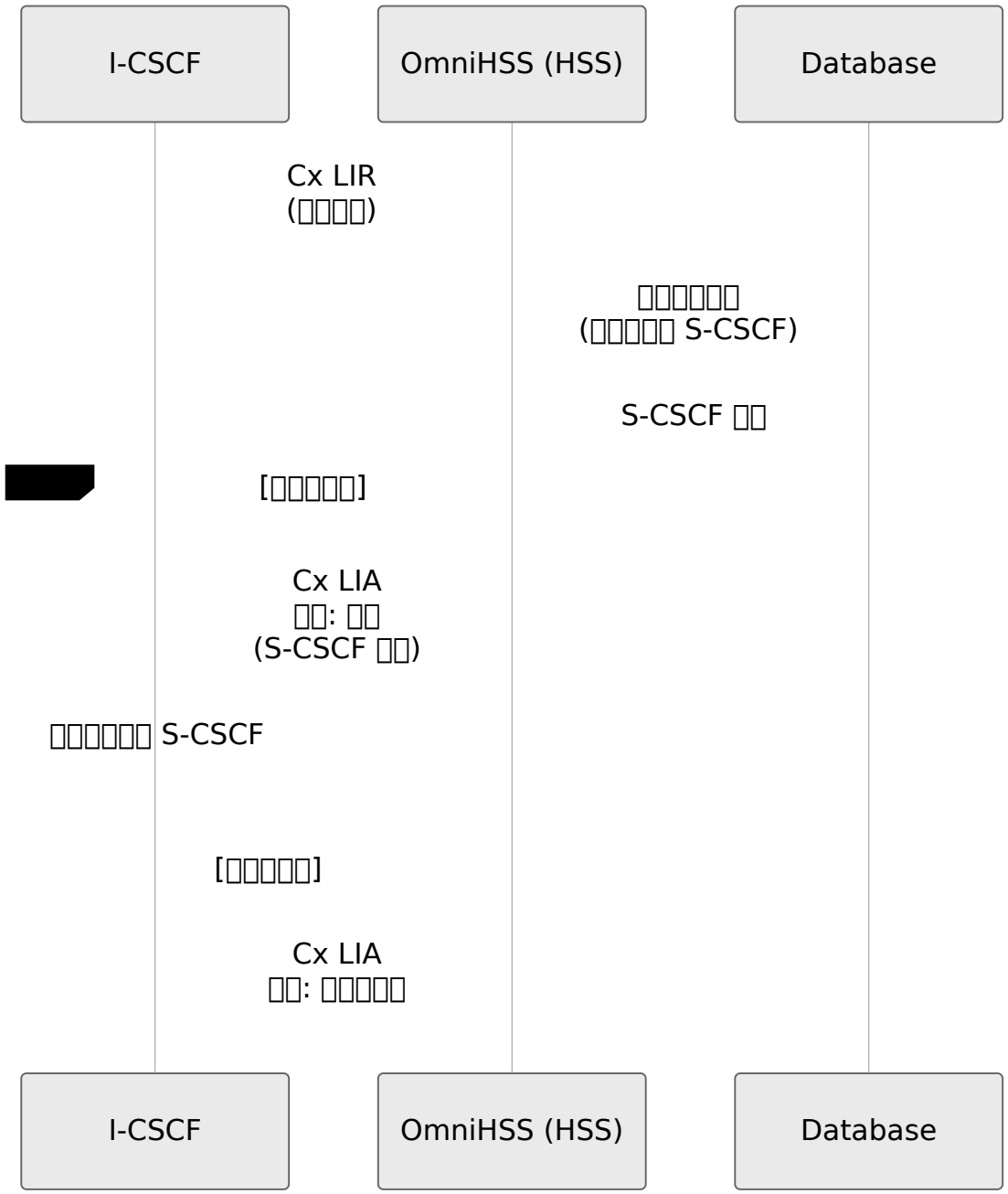
多媒体认证请求 **(MAR/MAA)**

S-CSCF 请求 IMS 注册的认证向量。



□□□□□□□ (LIR/LIA)

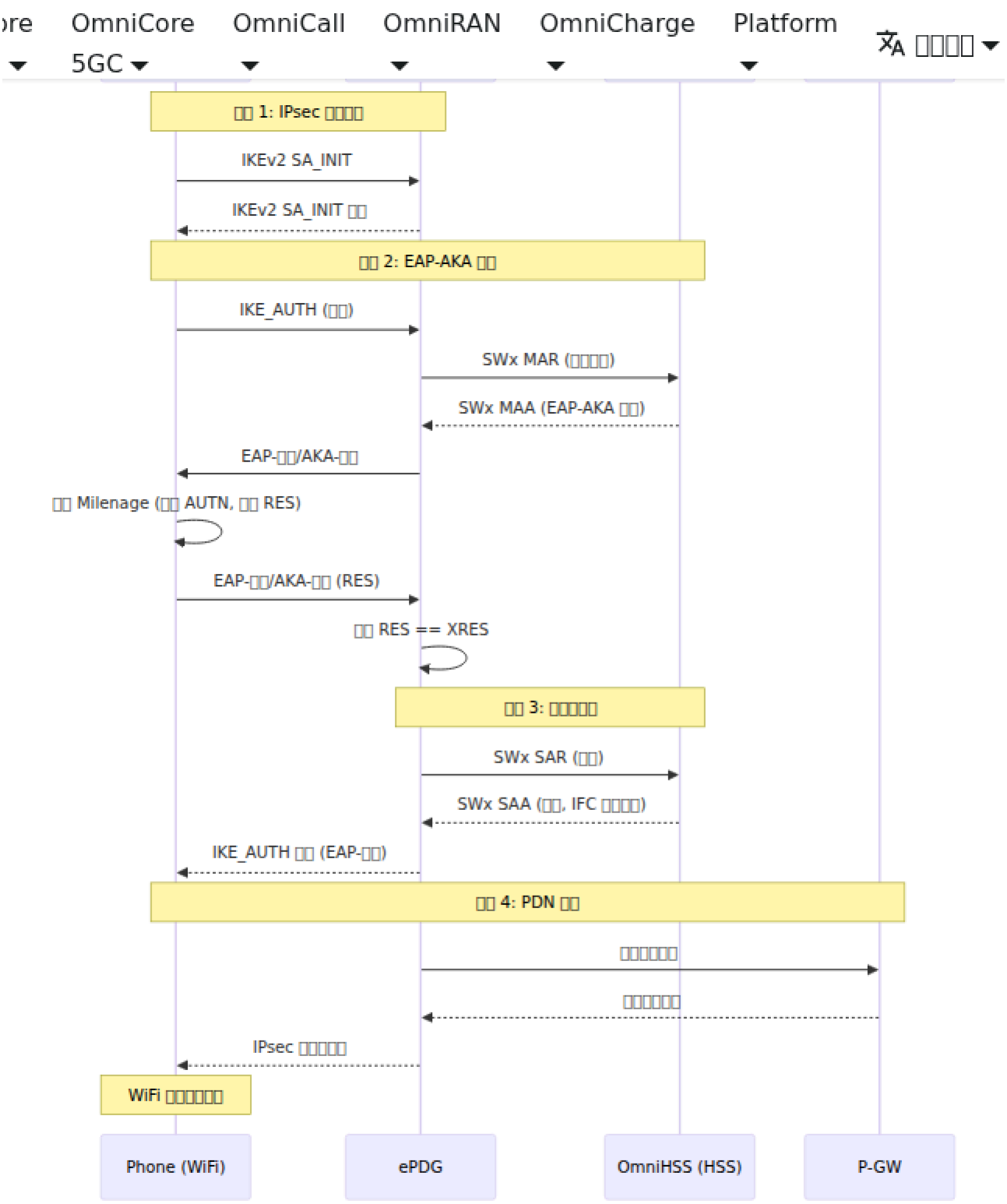
I-CSCF □□□□ S-CSCF □□□□□□□□□



Sh □□ (IMS □□□□□□)

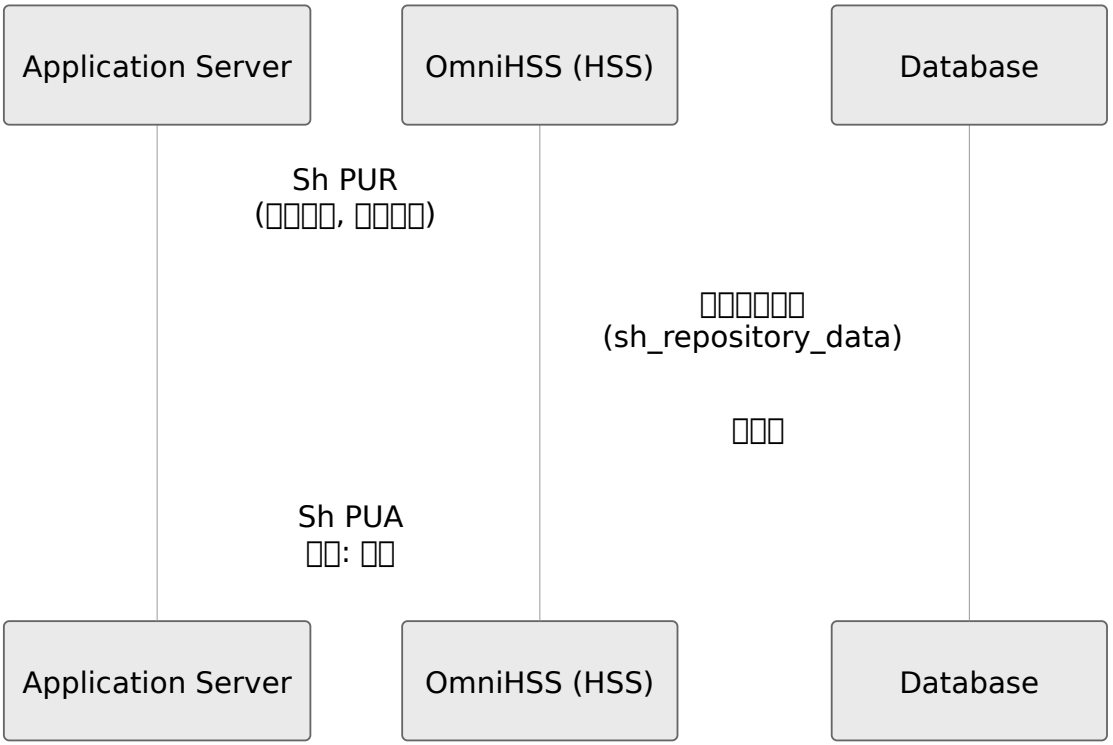
□□□□□□ (UDR/UDA)

□□□□□□□□□□□□□□□□□□□□



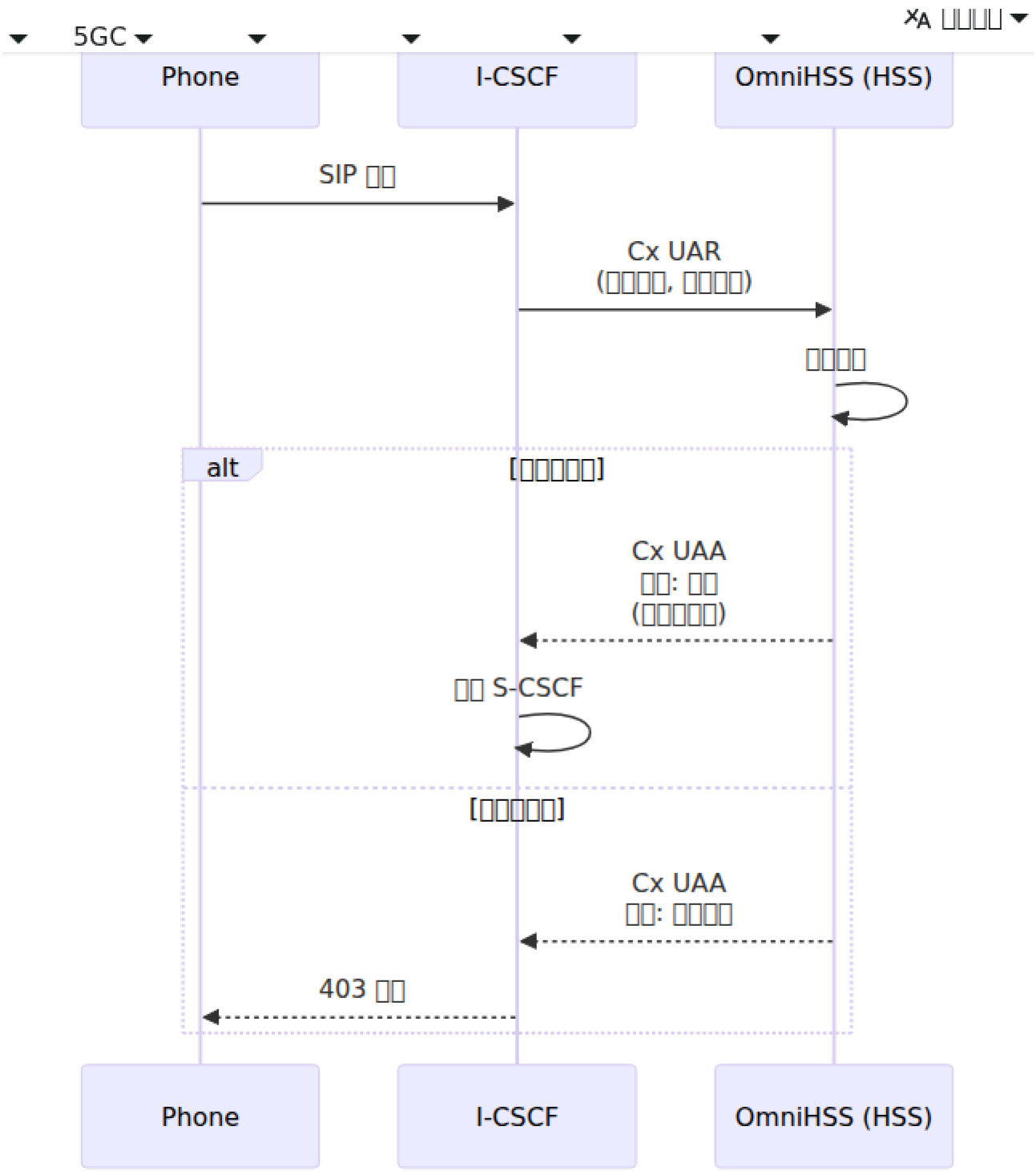
配置文件更新请求 **(PUR/PUA)**

应用服务器更新用户配置文件数据。


Application Server
OmniHSS (HSS)
Database
Sh PUR
(□□□□, □□□□)
□□□□□□□
(sh_repository_data)
□□□
Sh PUA
□□: □□
Application Server
OmniHSS (HSS)
Database

□□□□□□ (SNR/SNA)

□□□□□□□□□□□□□□□□



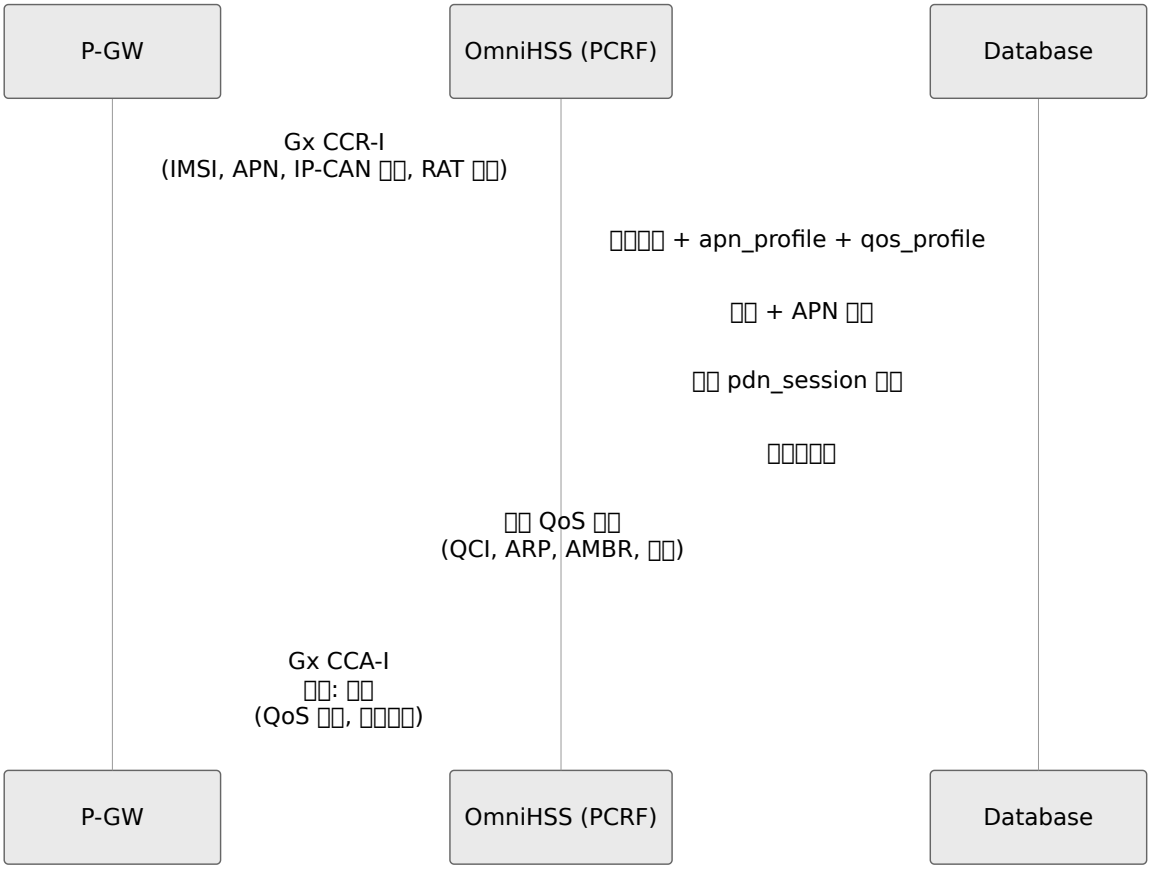
Gx 接口（策略控制）

OmniHSS 通过 Gx 接口作为 PCRF（策略和计费规则功能）运行。

请参阅 **PCRF** 文档以获取详细的架构、策略配置和 **QoS** 管理。

□□□□□□□ - □□ (CCR-I/CCA-I)

P-GW □□□ PDN □□□□□□□□□□□

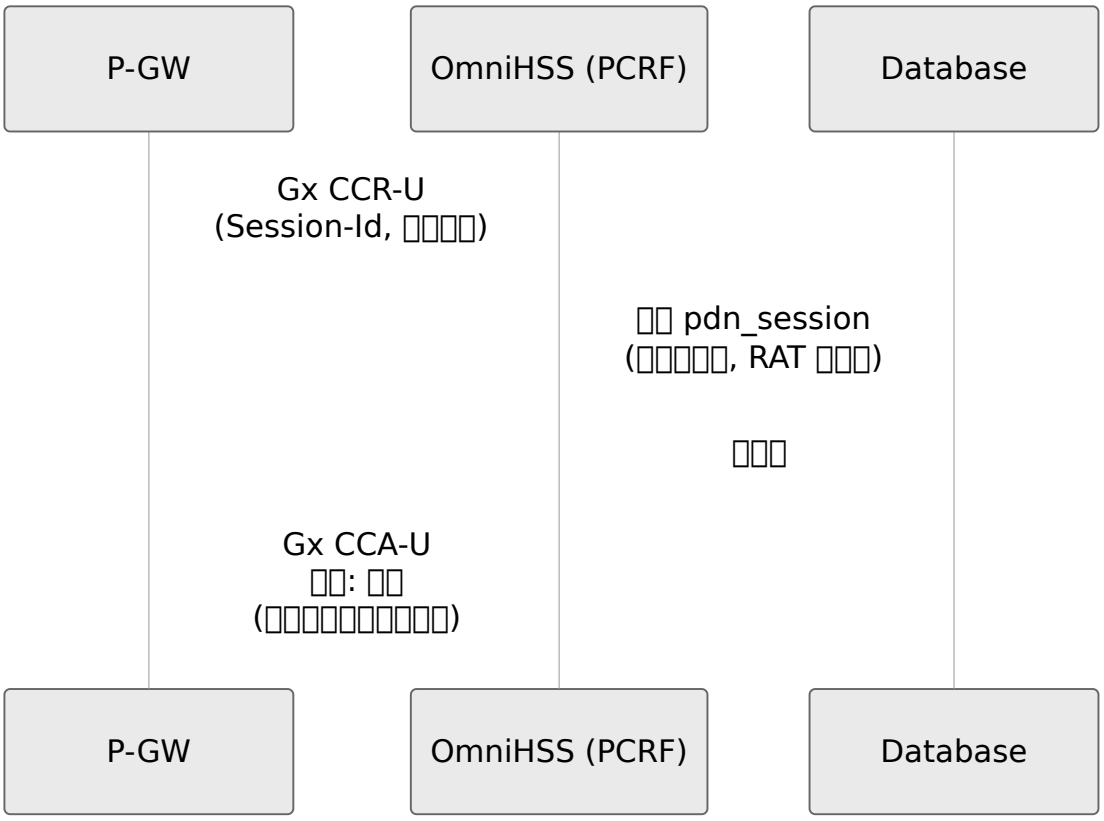


□□ **AVP:**

- □□: □□ ID (IMSI), □□□ ID (APN), RAT □□, IP-CAN □□
- □□: QoS □□ (QCI, ARP, AMBR), □□□□□□

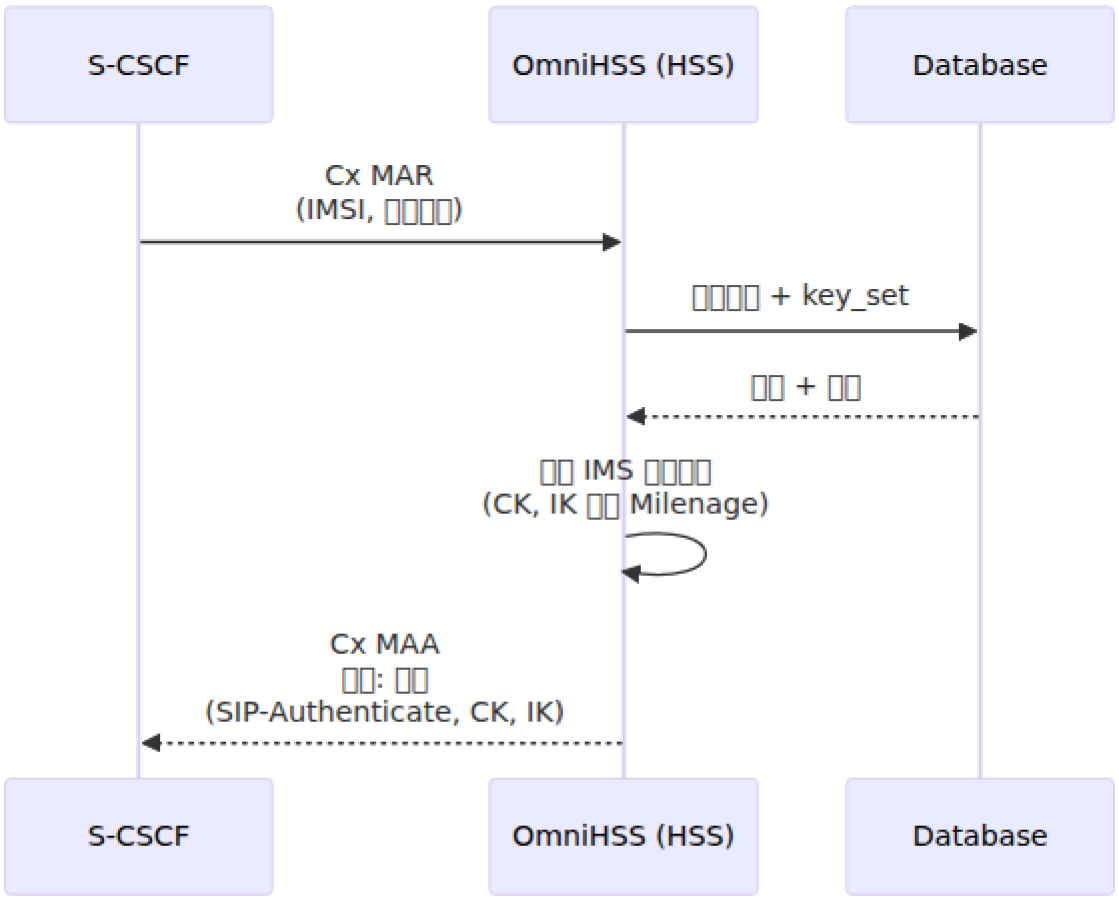
□□□□□□ - □□ (CCR-U/CCA-U)

P-GW □□□□□□□



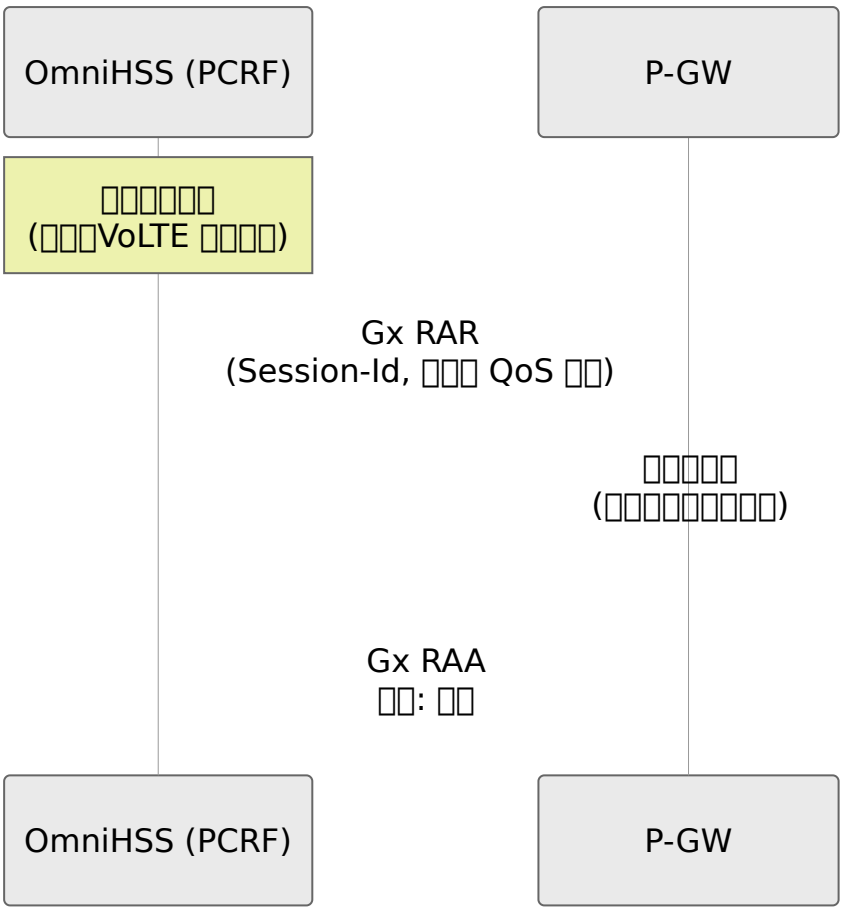
□□□□□□ - □□ (CCR-T/CCA-T)

P-GW □□ PDN □□□□□□



重新认证请求 **(RAR/RAA)**

OmniHSS (PCRF) 向 P-GW 发起策略更新。



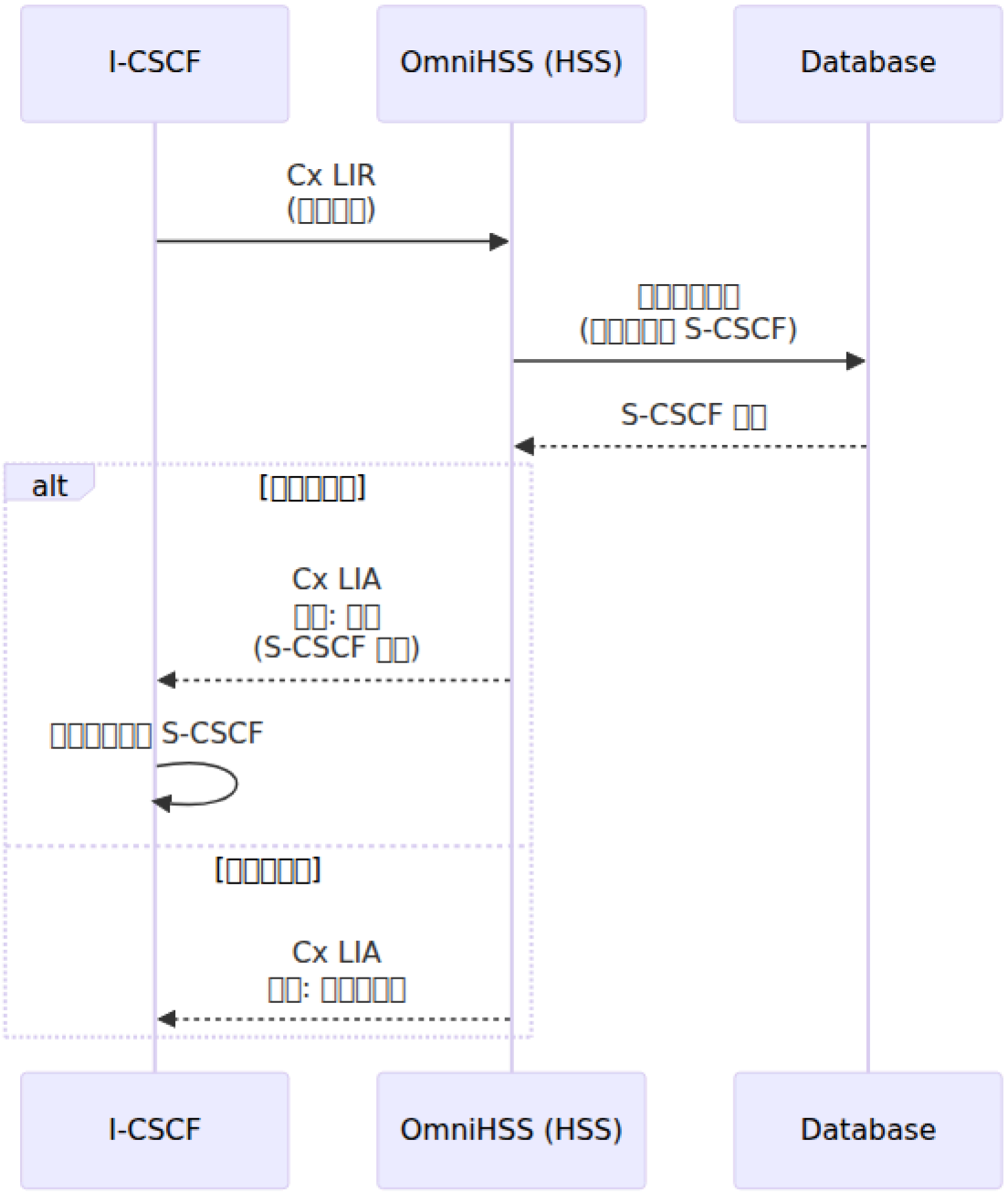
Rx □□ (IMS □□□□)

OmniHSS □□ Rx □□□□ PCRF □□□□□ IMS □□□□□

□□□ **PCRF □□** □□□□□□□ **VoLTE** □□□□□□□□□□□

AA □□ (AAR/AAA)

P-CSCF □□ IMS □□□□□□□□

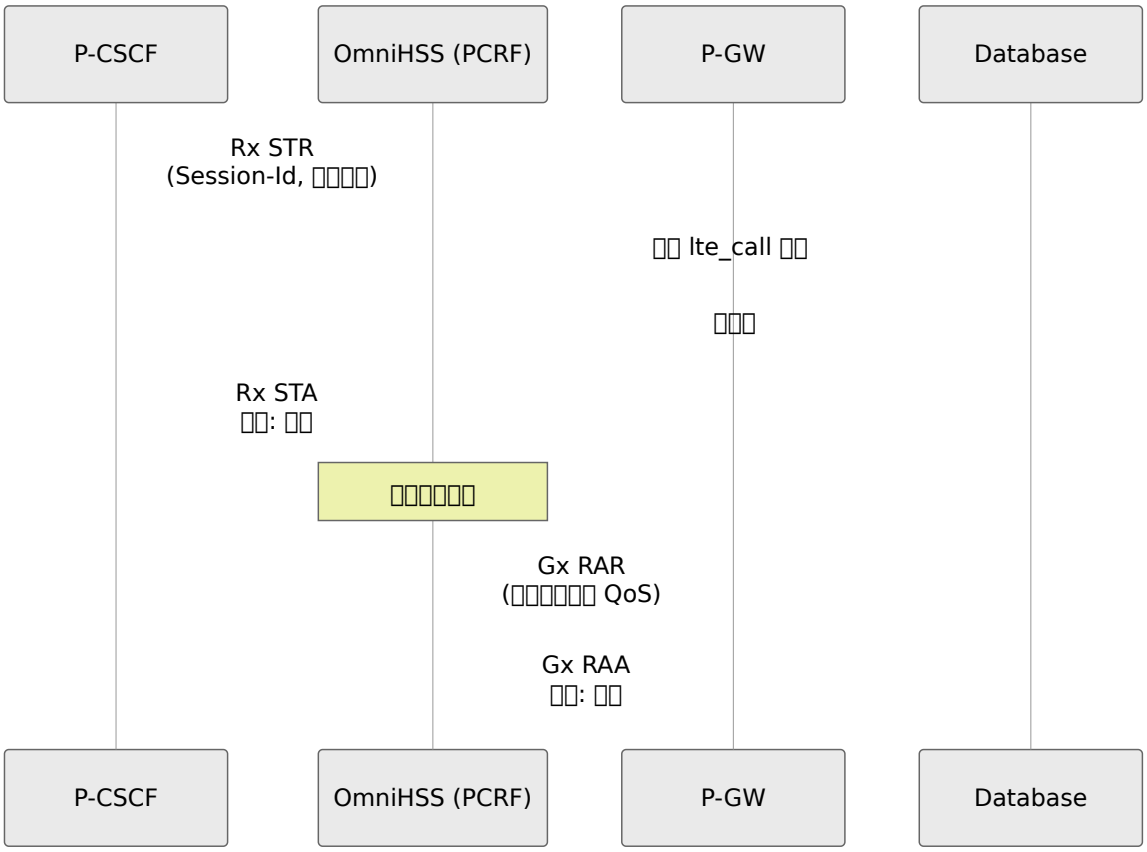


关键信息：

- 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。



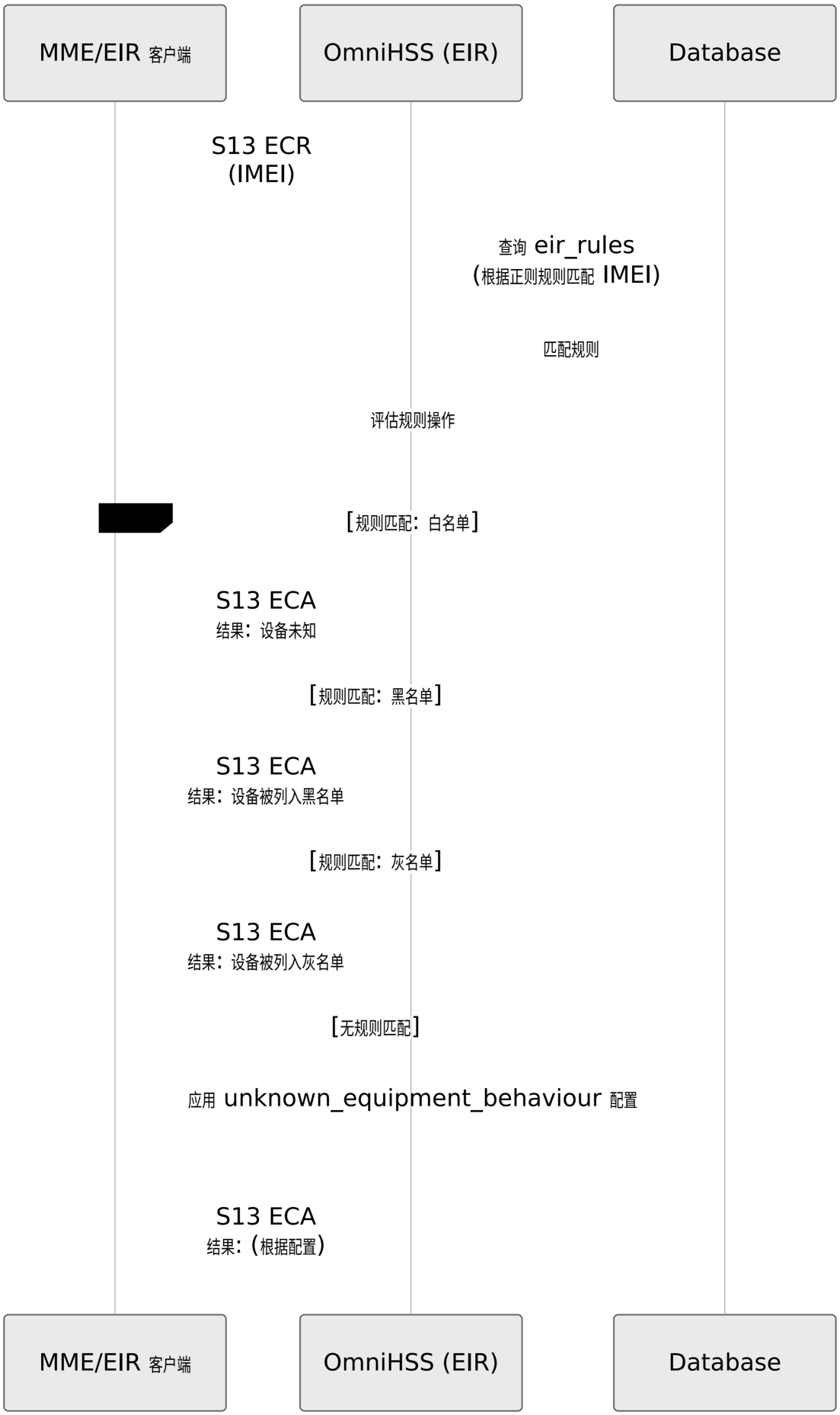
S13 □□ (EIR)

OmniHSS □□ S13 □□□□ EIR□□□□□□□□□□□□

□□□ **EIR □□** □□□□□□□□□□□□□□□**IMEI** □□□□□□□□□□

ME □□□□□□ (ECR/ECA)

□□ EIR □□□□□ MME□□□□□□□□

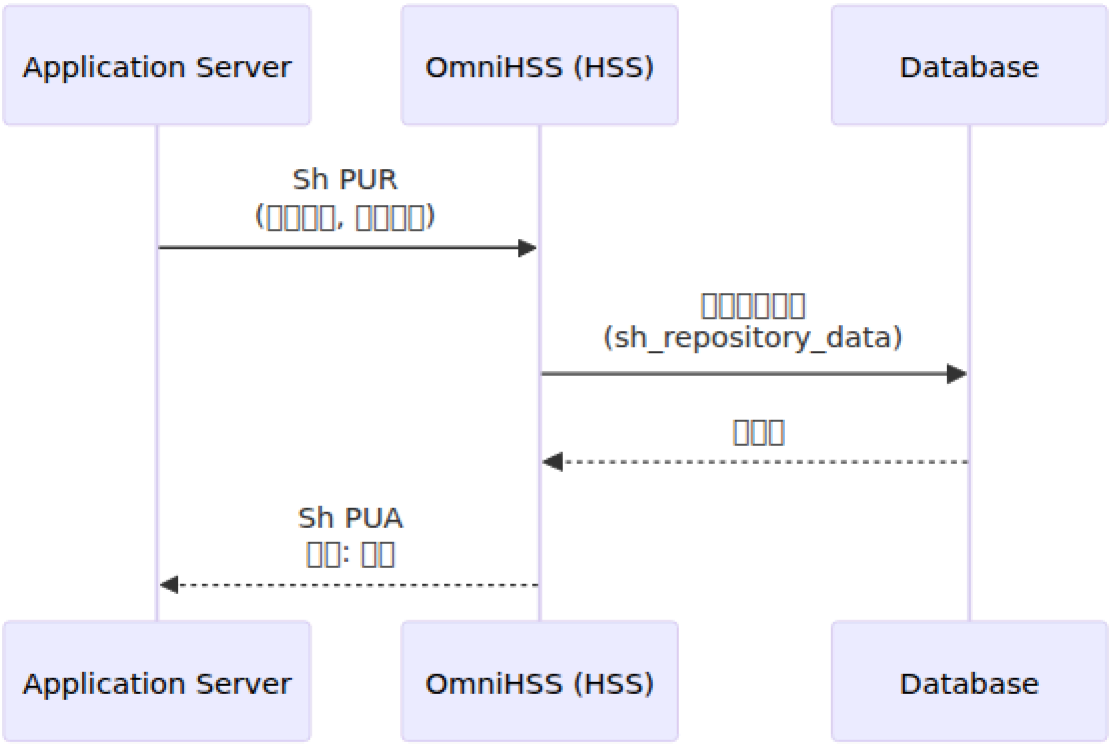

MME/EIR 客户端
OmniHSS (EIR)
Database
S13 ECR
(IMEI)
查询 eir_rules
(根据正则规则匹配 IMEI)
匹配规则
评估规则操作
[规则匹配：白名单]
S13 ECA
结果：设备未知
[规则匹配：黑名单]
S13 ECA
结果：设备被列入黑名单
[规则匹配：灰名单]
S13 ECA
结果：设备被列入灰名单
[无规则匹配]
应用 unknown_equipment_behaviour 配置
S13 ECA
结果：(根据配置)
MME/EIR 客户端
OmniHSS (EIR)
Database

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

完整呼叫流程：**VoLTE** 通话

端到端 VoLTE 通话设置，显示多个接口。

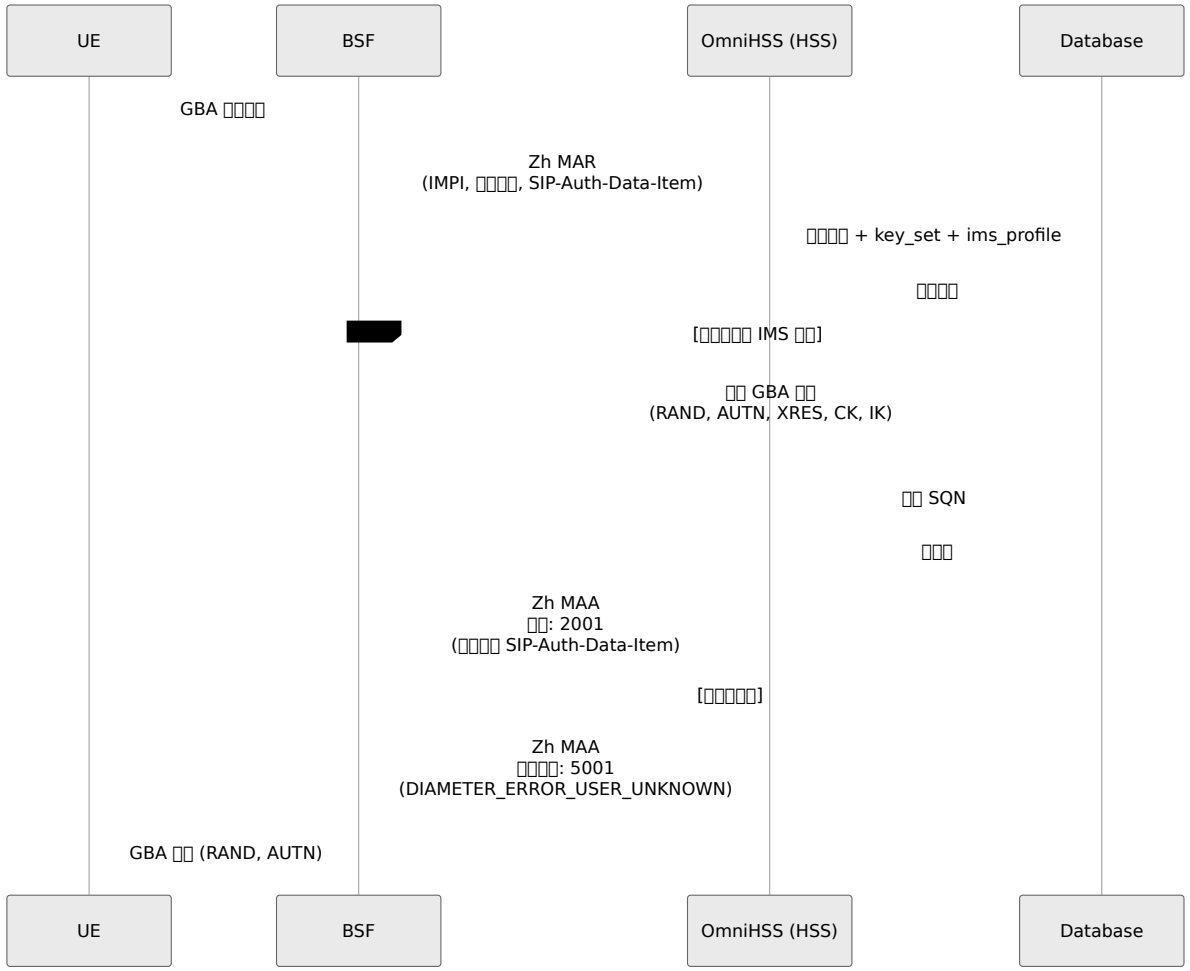


Zh 接口（通用引导架构）

Zh 接口通过将引导服务器功能 (BSF) 连接到 HSS，以实现通用引导架构 (GBA) 的用户认证和密钥派生。

多媒体认证请求 **(MAR/MAA)**

BSF 请求 GBA 认证向量以进行引导。



□□ AVP:

- □□: □□□ (IMPI), □□□□, SIP-Auth-Data-Item (□□□□), □□□
- □□: SIP-Auth-Data-Item (SIP-Authenticate, SIP-Authorization, □□□□, □□□□□), GBA-UserSecSettings

GBA □□:

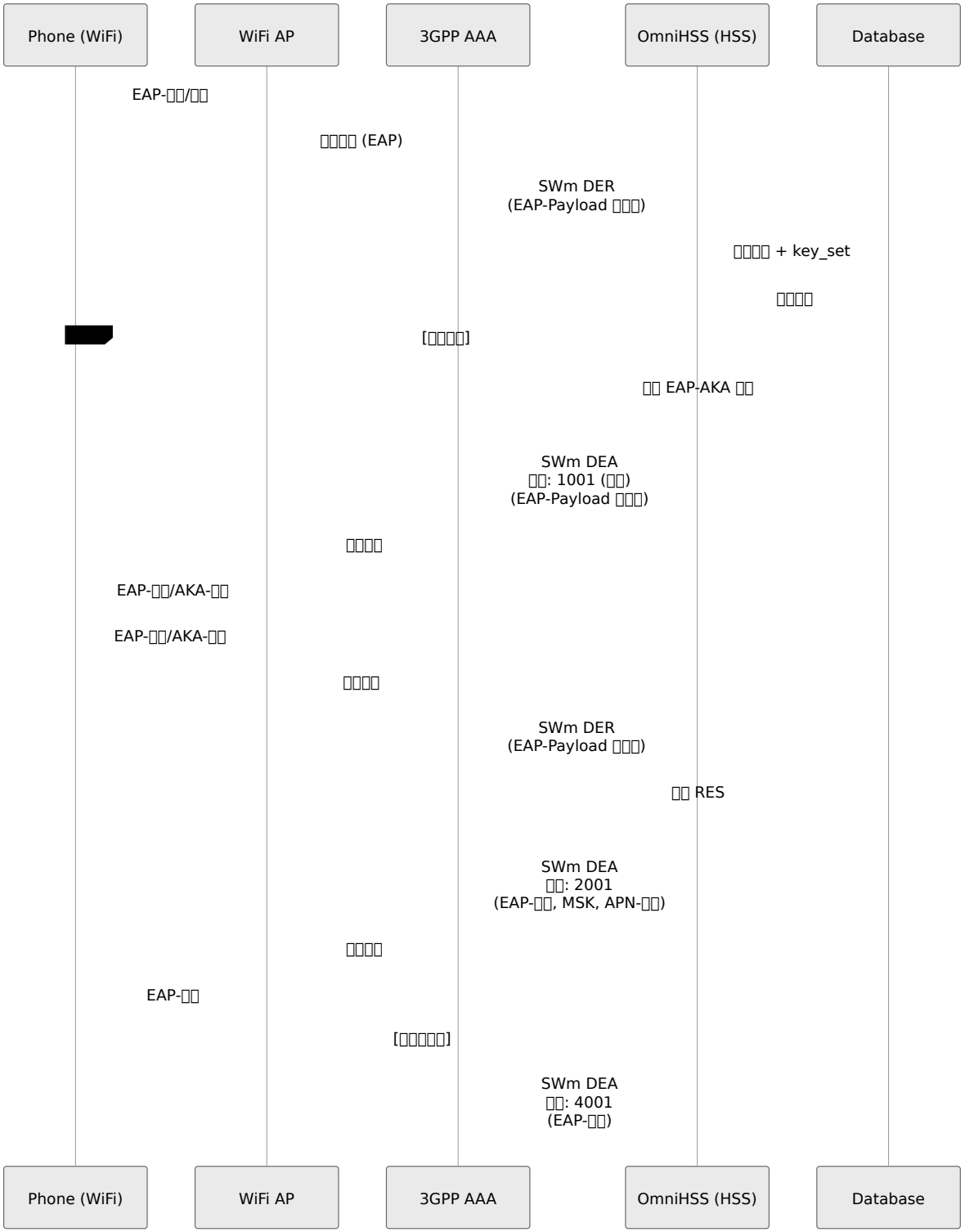
□□	□□
MBMS	□□□□□□□□□□□□□□
HTTPS	□□ GBA □□□□□
NAF	□□□□□□□□□□
Presence	□□□□□□□□□

SWm 接口（不受信任的非 3GPP 接入）

SWm 接口通过将 3GPP AAA 服务器连接到 HSS，以实现对不受信任的非 3GPP 接入网络（例如，通过 ePDG 的 WiFi）的 EAP-AKA 认证。

Diameter-EAP-请求 (DER/DEA)

EAP 认证交换以进行 WiFi 访问。

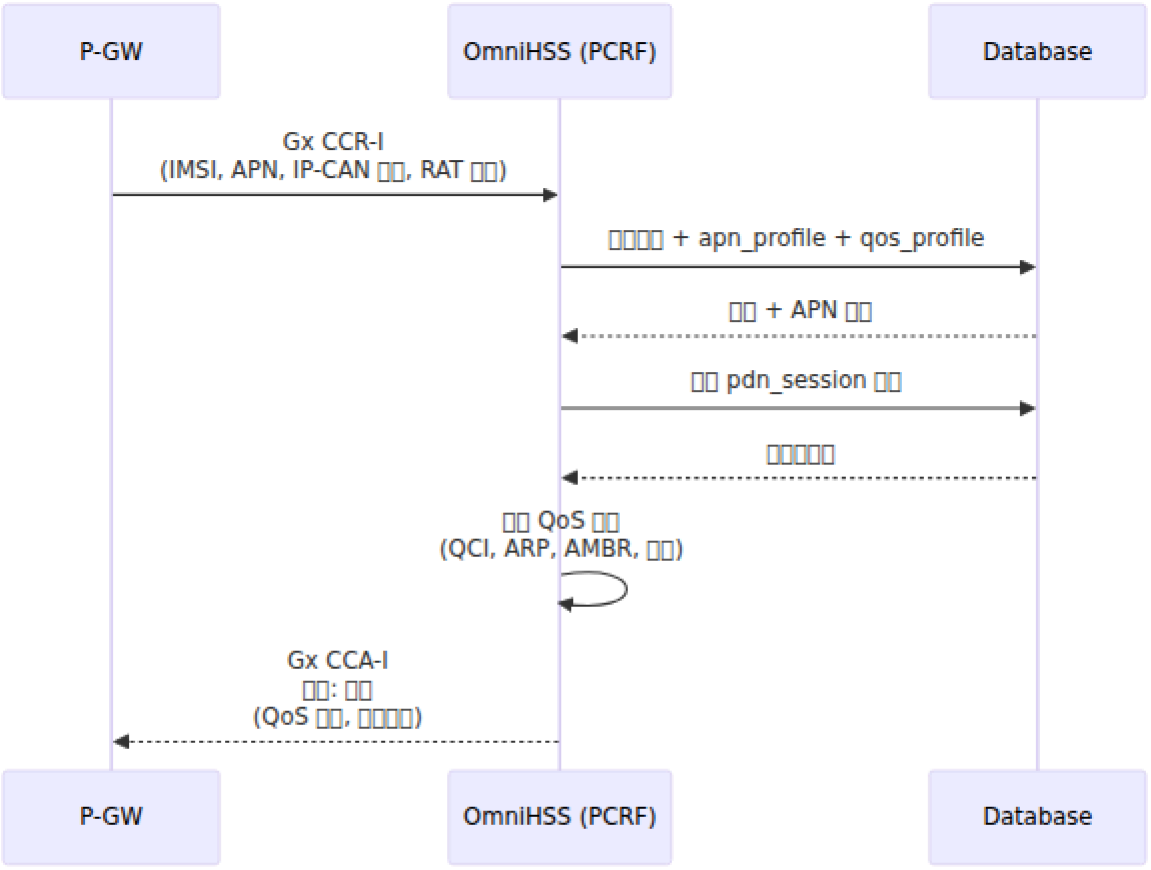


□□ AVP:

- □□: □□ ID, □□□□ ID, EAP-Payload, □□□□□□, □□□ (□□)
- □□: EAP-Payload, EAP □□□□□ (MSK), APN □□, □□□□

AA-请求（AAR/AAA）

成功 EAP 认证后的授权。

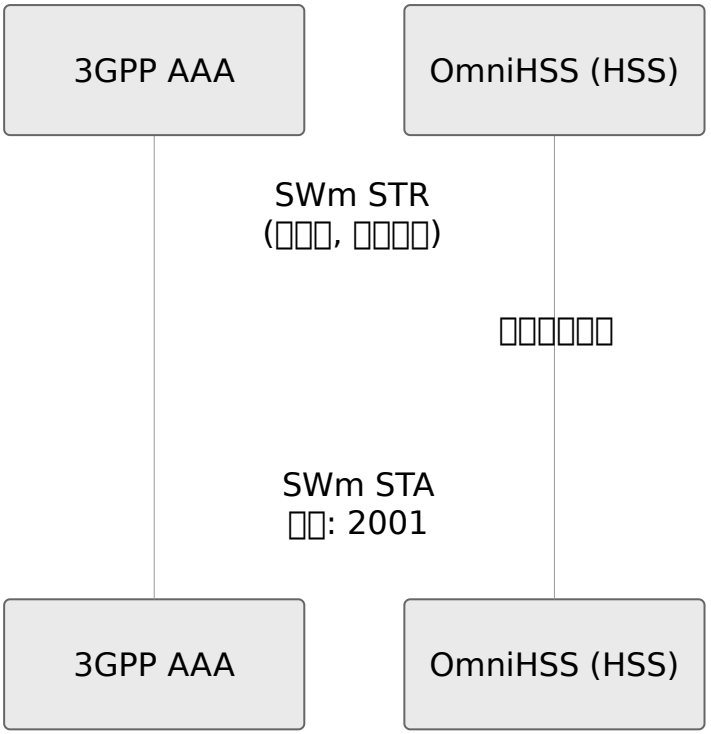


关键 **AVP:**

- 请求：会话 ID，认证应用 ID，用户名，认证请求类型，AAR 标志
- 响应：APN 配置，3GPP 计费特性，会话超时

会话终止请求（STR/STA）

WiFi 会话终止通知。



□□□□:

□□	□	□□
DIAMETER_LOGOUT	1	□□□□
DIAMETER_SERVICE_NOT_PROVIDED	2	□□□□□
DIAMETER_BAD_ANSWER	3	□□□□
DIAMETER_ADMINISTRATIVE	4	□□□□□
DIAMETER_LINK_BROKEN	5	□□□□
DIAMETER_AUTH_EXPIRED	6	□□□□
DIAMETER_USER_MOVED	7	□□□□□□□□□
DIAMETER_SESSION_TIMEOUT	8	□□□□

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. 漫游规则允许访问网络

位置更新失败 (S6a ULR)

检查：

1. EPC 配置文件存在并且已配置 APN
2. 数据服务允许漫游
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）

← □□□□□□

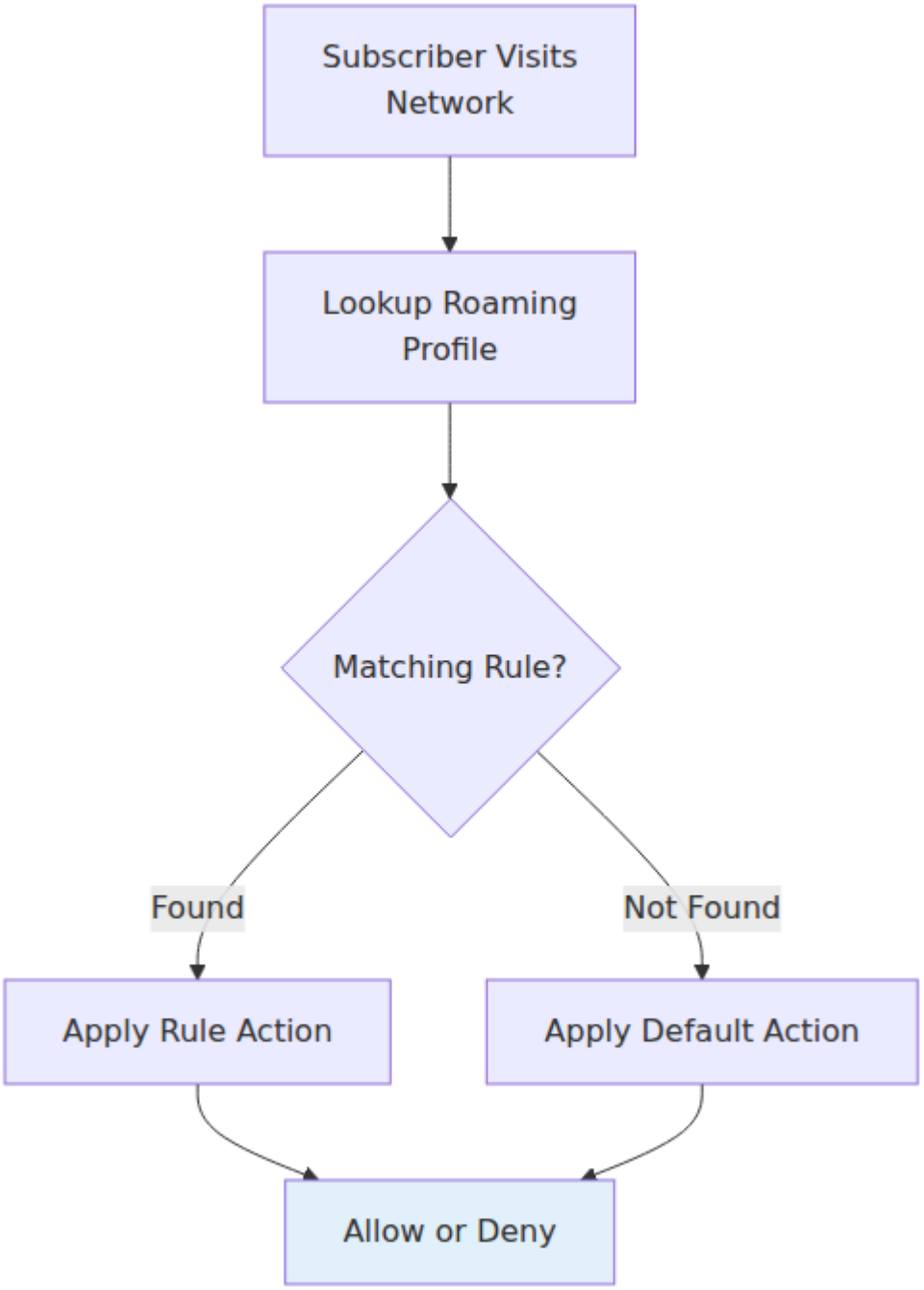
OmniHSS 漫游控制

← 返回操作指南

概述

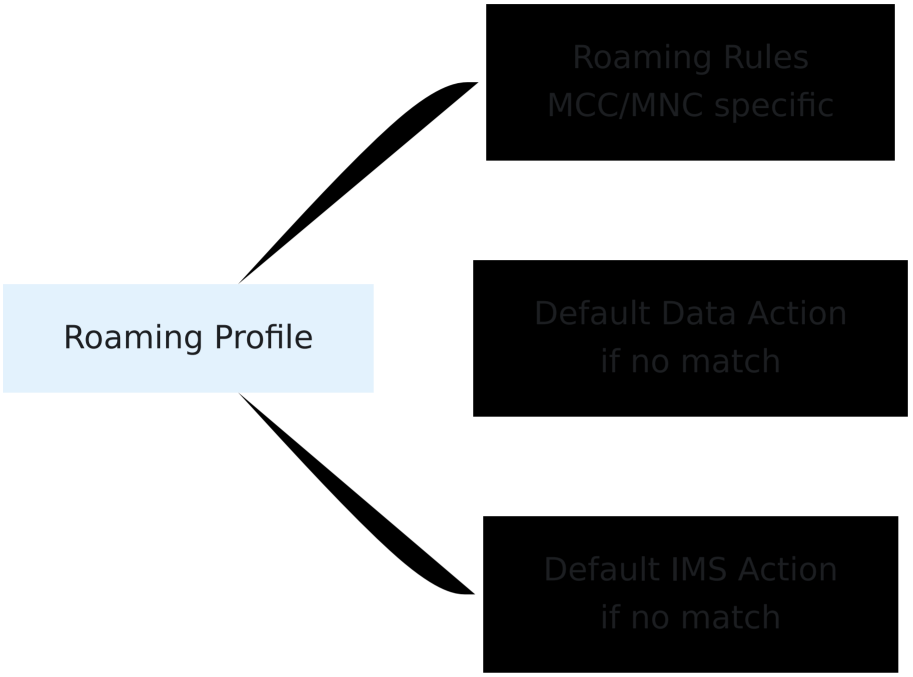
OmniHSS 提供细粒度的漫游控制，允许您定义订阅者在漫游时可以访问的数据和 IMS 服务的网络。

□□□□□□


Subscriber Visits Network
Lookup Roaming Profile
Matching Rule?
Found
Not Found
Apply Rule Action
Apply Default Action
Allow or Deny

漫游配置结构

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

定义默认行为并链接到规则。

字段：

- `name` - 配置名称
- `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
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_profile": {
      "name": "Allow All",
      "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
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_profile": {
      "name": "No Roaming",
      "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": "Allow 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": "Allow 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
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"roaming_profile\": {
      \"name\": \"US Carriers Only\",
      \"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": "Data Only - T-Mobile",
      "mcc": "310",
      "mnc": "260",
      "data_action": "allow",
      "ims_action": "deny"
    }
  }'
```

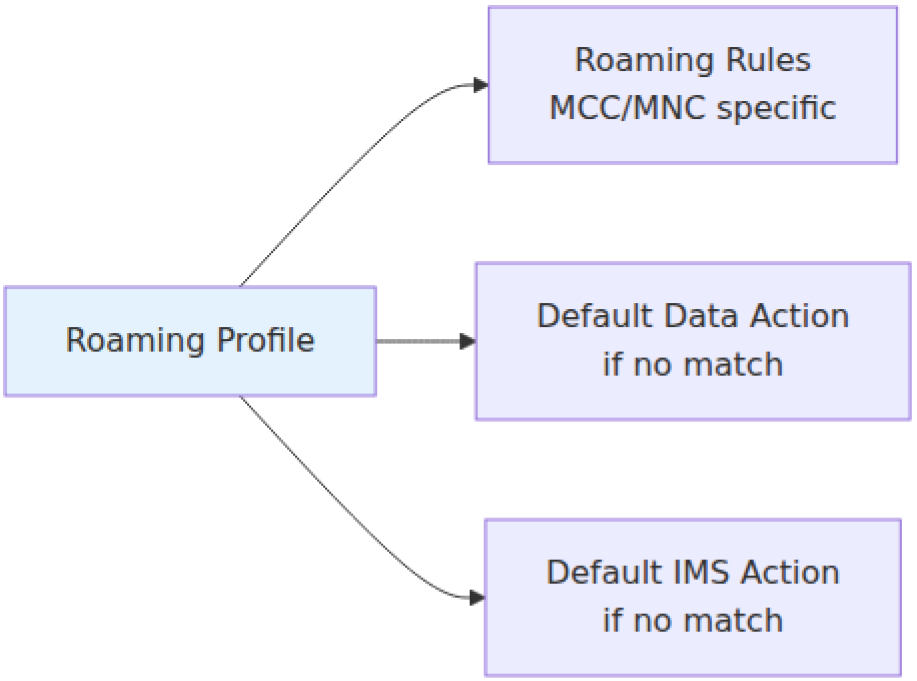
阻止特定网络（黑名单）

```
# 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": "Block Expensive Network",
      "mcc": "206",
      "mnc": "01",
      "data_action": "deny",
      "ims_action": "deny"
    }
  }' | jq -r '.response.id')

# 创建默认允许的配置
curl -k -X POST https://hss.example.com:8443/api/roaming/profile \
  -H "Content-Type: application/json" \
  -d "{
    \"roaming_profile\": {
      \"name\": \"Block Expensive Networks\",
      \"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：仅国内漫游

订阅者可以在本国漫游，但不能国际漫游。

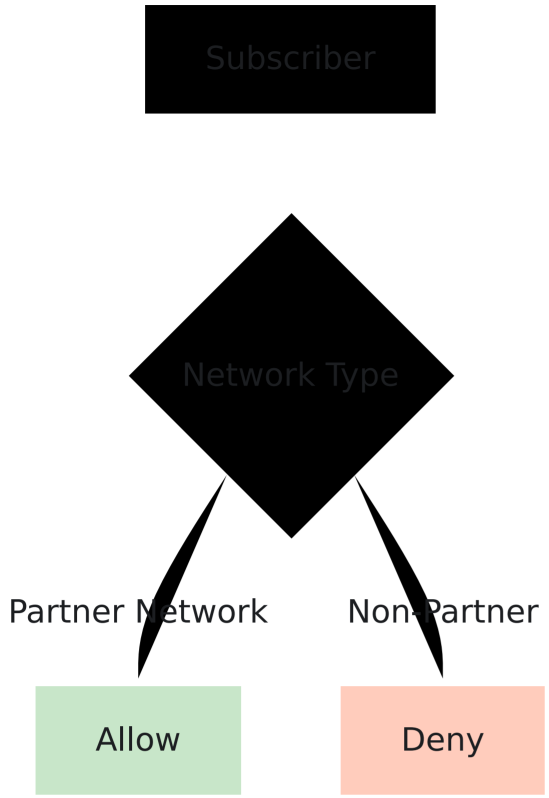


配置：

- 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：仅漫游合作伙伴

订阅者只能在与商业协议的网络上漫游。

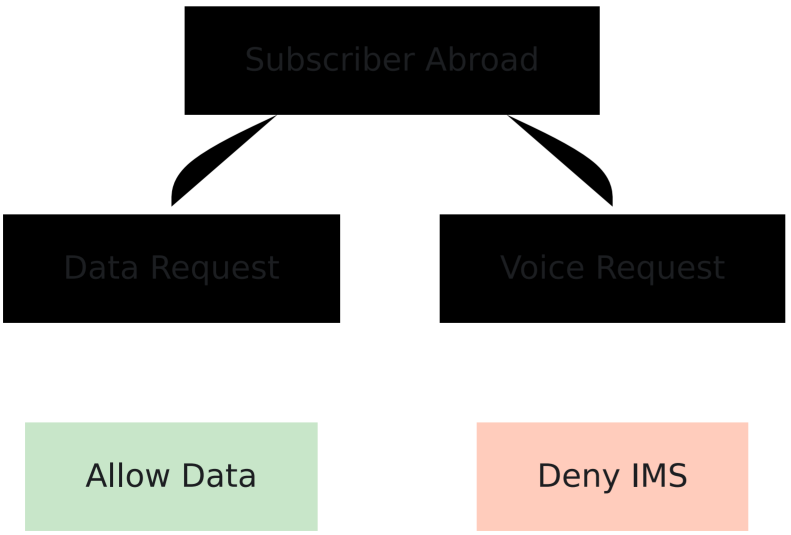


配置：

- 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，不允许语音漫游

订阅者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**）

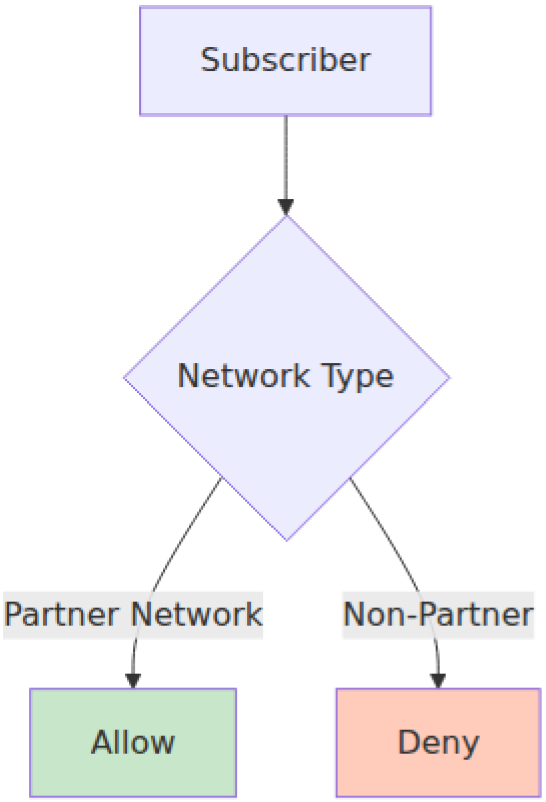
MCC	MNC	运营商
310	410	AT&T
311	480	Verizon
310	260	T-Mobile
310	120	Sprint
313	380	（示例测试网络）

完整列表：请参见 ITU-T E.212 或 MCC/MNC 数据库

漫游执行点

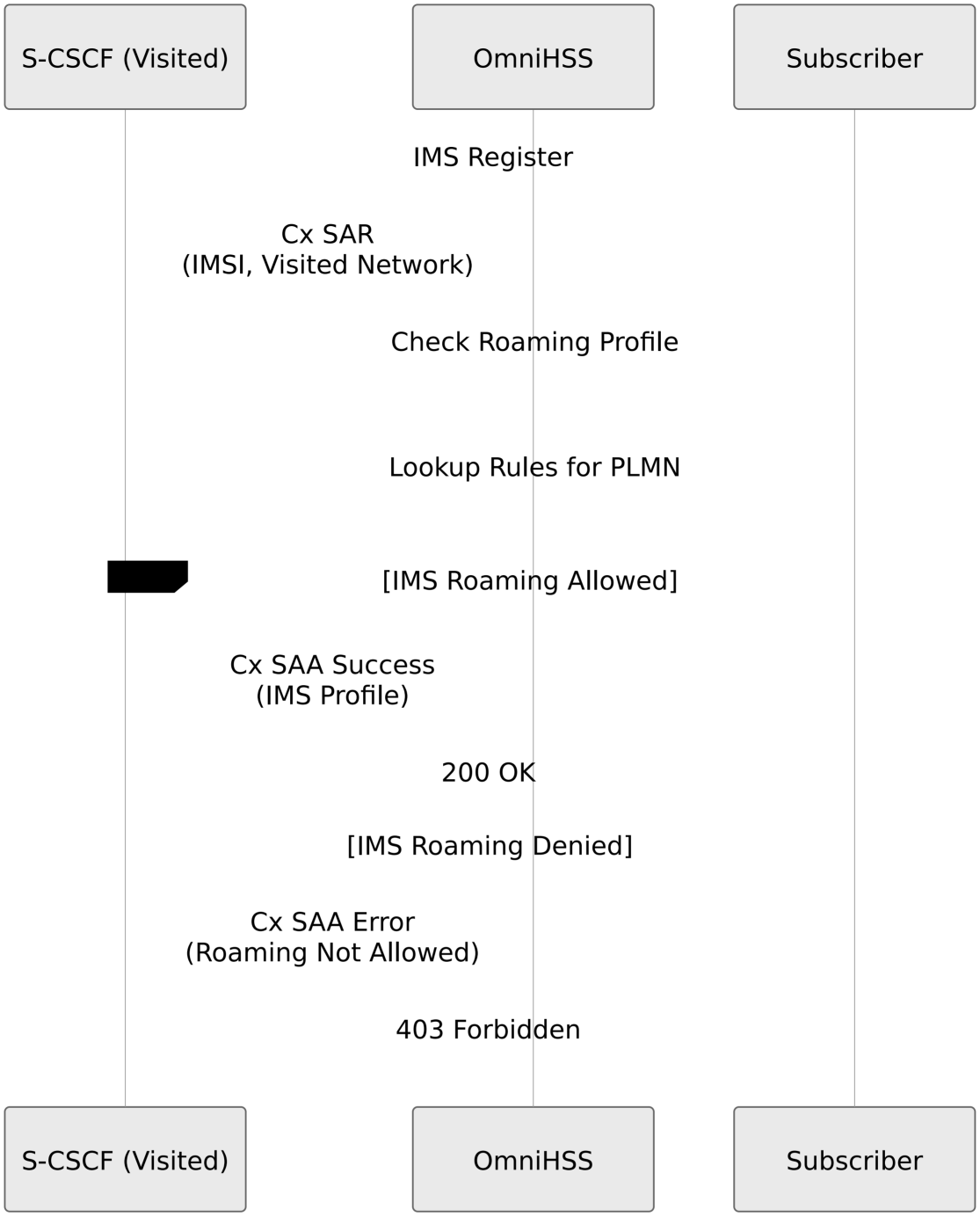
S6a 接口（数据）

当订阅者连接到访问网络时：



Cx 接口（IMS）

当订阅者在访问网络中注册 IMS 时：



排查漫游问题

订阅者无法在访问网络中连接

检查漫游配置分配：

- 查询数据库以查看订阅者分配的漫游配置

- 验证配置名称和默认操作设置

检查访问网络是否存在规则：

- 查询数据库以获取与访问网络的 MCC/MNC 匹配的漫游规则
- 验证订阅者的漫游配置中□否存在规则
- 检查该特定网络的 data_action 值

订阅者可以连接但无法注册 IMS

单独检查 **IMS** □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

查看漫游检查日志：

```
[info] Roaming check: IMSI 001001123456789, Visited PLMN 310-410
[info] Roaming rule matched: "Allow AT&T"
[info] Data action: allow, IMS action: allow
```

最佳实践

配置设计

1. 从限制开始 - 默认拒绝，明确允许合作伙伴
2. 彻底测试 - 在生产之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 "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 - 跟踪订阅者实际访问的网络

作程序

1. 紧急变更 - 有快速启用/禁用漫游的程序
2. 批量更新 - 计划更新多个订阅者的漫游配置
3. 报告 - 跟踪漫游使用情况和拒绝尝试
4. 客户沟通 - 通知客户漫游政策变更

← 返回操作指南 | 下一步：协议流程 →

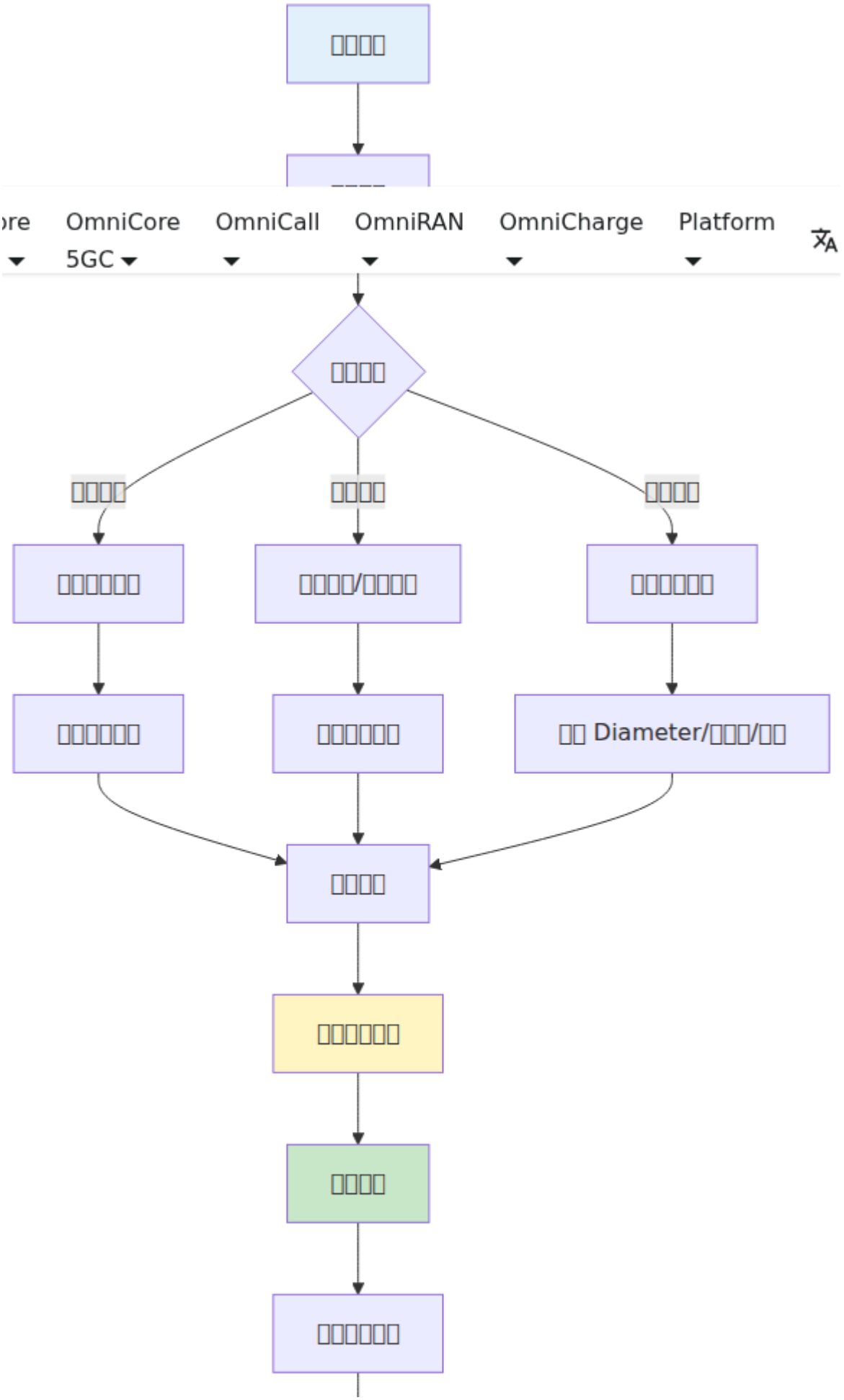
OmniHSS 故障排除指南

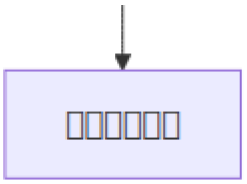
← 返回操作指南

目录

故障排除概述

一般故障排除方法


OmniCore
5GC
OmniCall
OmniRAN
OmniCharge
Platform
Diameter/



收集信息

在故障排除任何问题之前，请收集：

1. 用户信息（如果□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？
    - □间歇性还□持续性？
    - 上次成功操作的时间
3. 影响范围
    - 单个用户还□多个用户？
    - 特定网络还□所有网络？
    - 特定服务（数据/语音）还□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
- 对于具有相同配置文件的其他用户有效

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□否存在并具有正确的值:

   ```
   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:

- 认证在之前正常后失败
- 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. 在日志中查找与 SQN 相关的错误
3. 验证用户的密钥集 SQN 值

解决方案：

- 用户发送 AUTS 后，SQN 将自动重新同步
- 如果持续存在，重置密钥集中的 SQN 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：

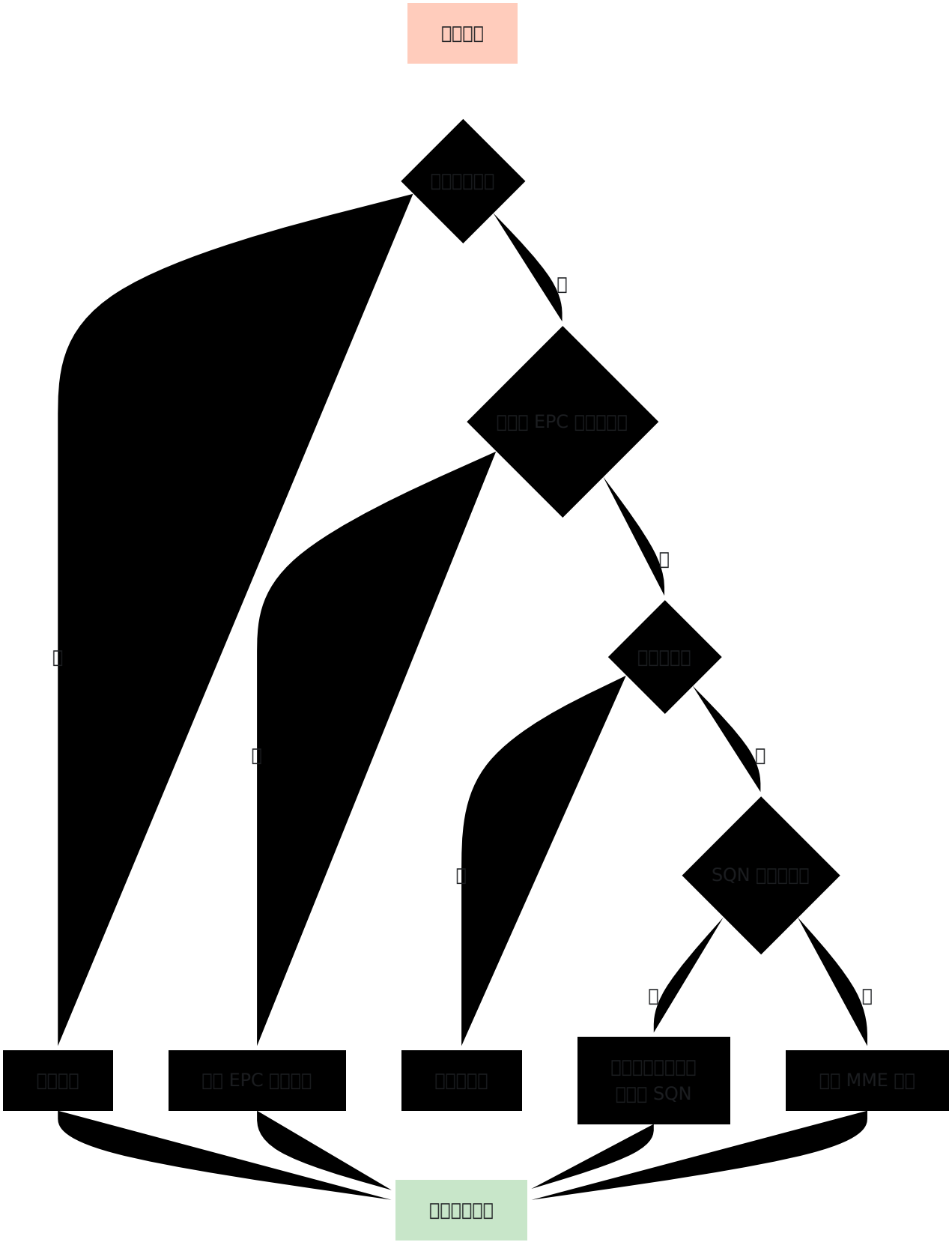
1. 检查用户的 `epc_profile_id` 字段
2. 验证 EPC 配置文件是否存在：

```
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: 网络连接问题

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□否可达:

   ```
   telnet [PEER_IP] 3868
   ```

3. 验证防火墙规则□否允许 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 Diameter 配置:
    - 验证对等体 origin_host □否与对等体的预期值匹配
    - 检查 origin_realm 配置
    - 验证对等体 IP 地址□否正确
2. 检查日志中的 CER/CEA 错误
3. 验证对等体的配置□否期望 OmniHSS 的 origin_host

解决方案:

- 使用正确的 Diameter 配置 更新 runtime.exs
- 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□否完整

4. 检查对等体的证书□否为互相 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

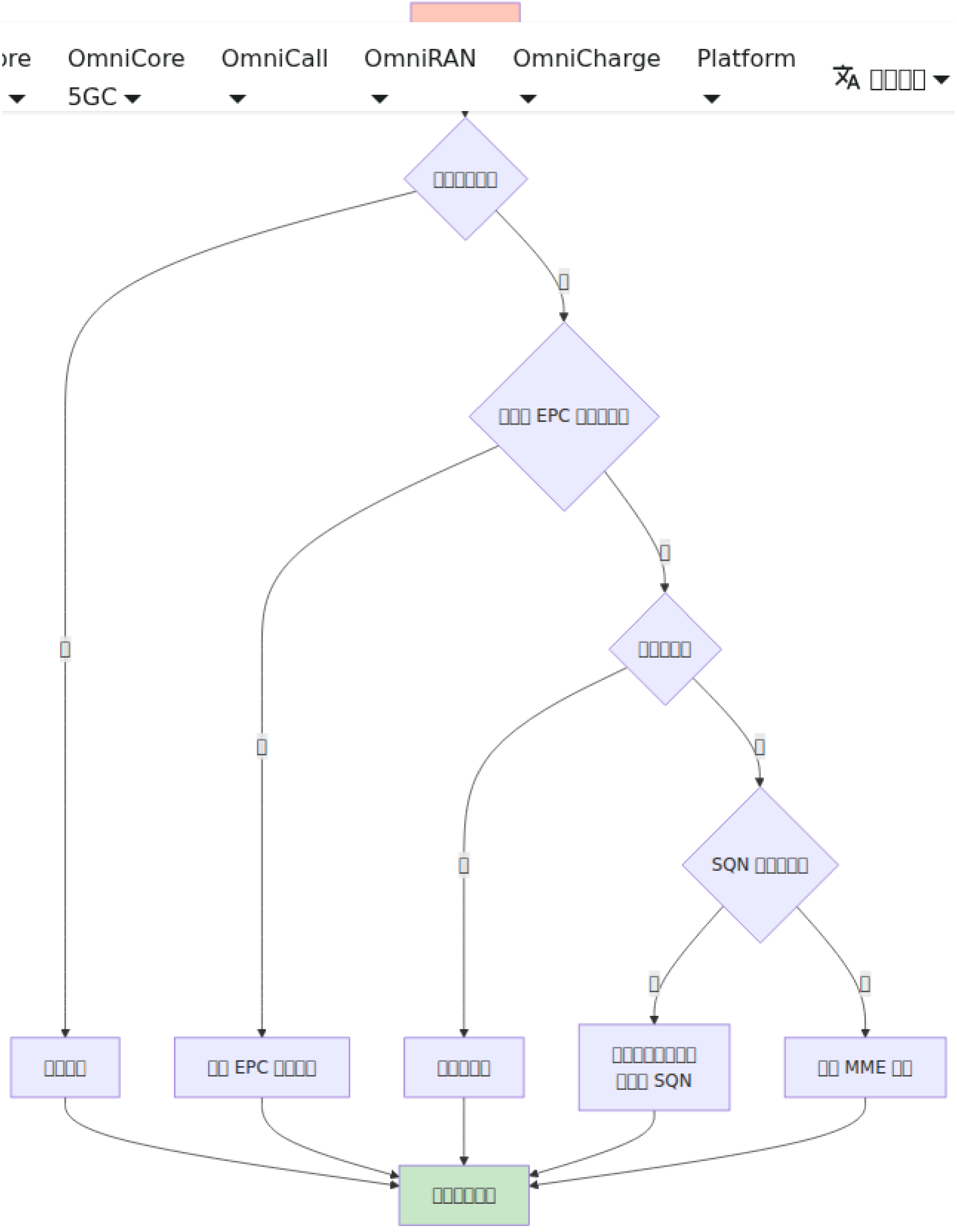
诊断步骤：

1. 检查 控制面板 Diameter 页面 中的对等应用程序
2. 验证对等体□否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□否包含所需的 Diameter 应用程序
- 检查对等体类型□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:

   ```
   # 如果使用 PostgreSQL
   psql -h [DB_HOST] -U [DB_USER] -d [DB_NAME]

   # 如果使用 MySQL
   mysql -h [DB_HOST] -u [DB_USER] -p [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
- 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计数
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，调查连接泄漏

- 如果负载持续较高，考虑数据库扩展

原因 **4**：慢查询

症状：

- API 响应非常慢
- 用户查找超时
- 数据库 CPU 高

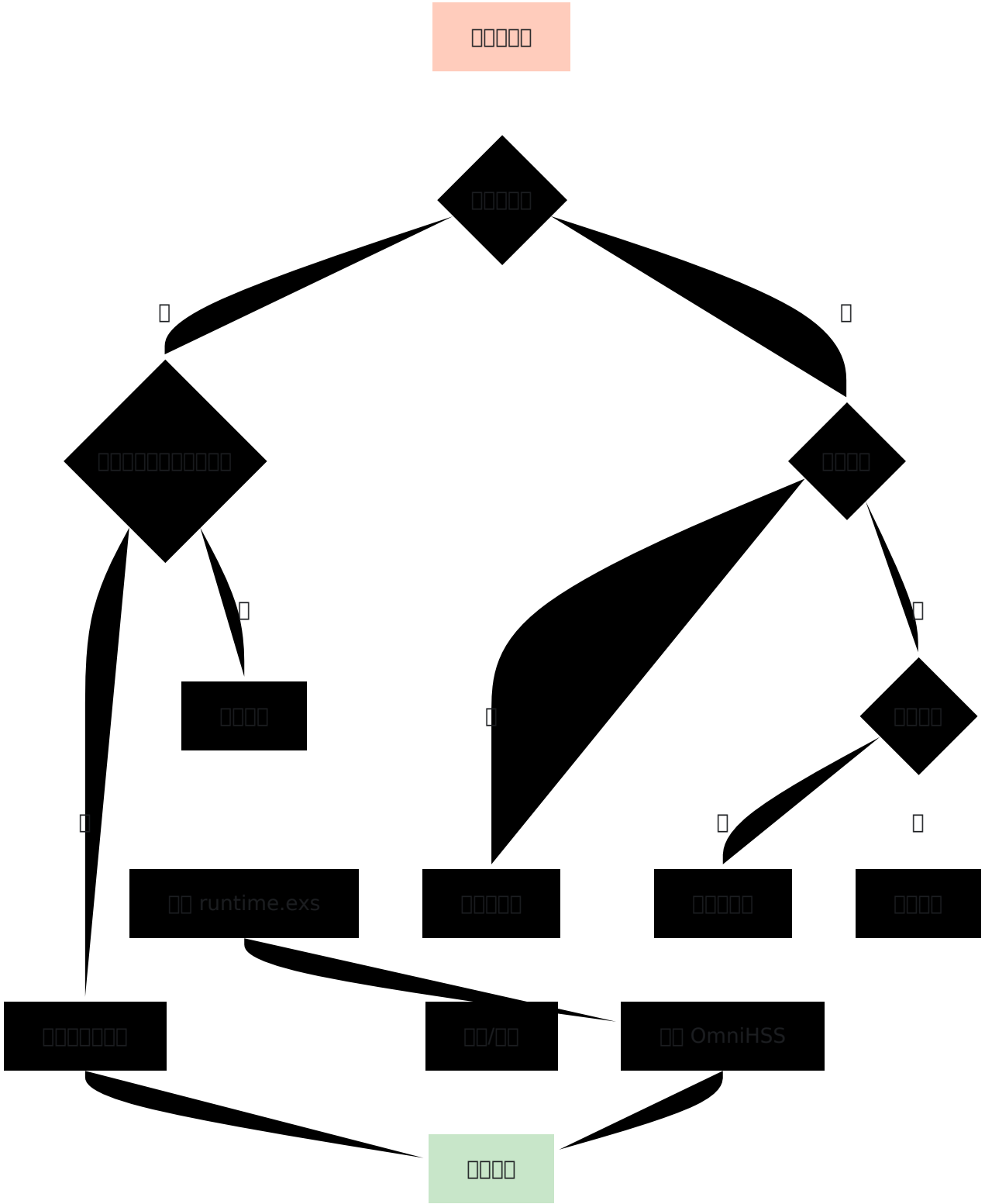
诊断步骤：

1. 查询数据库的慢查询日志
2. 确定特定的慢查询
3. 检查□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
- 如果需要，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
- 未建立 PDN 会话

常见原因和解决方案

原因 1: 漫游被拒绝

症状:

- 用户在本地网络上工作，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□否存在��游规则

解决方案:

- 为访问网络 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
- 请参阅 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□否链接了 APN 配置文件
2. 验证 APN 标识符□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□否存在

原因 **3**: **MME** 未连接

症状：

- 所有用户都无法连接
- 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 □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
- 脱离���重新连接失败

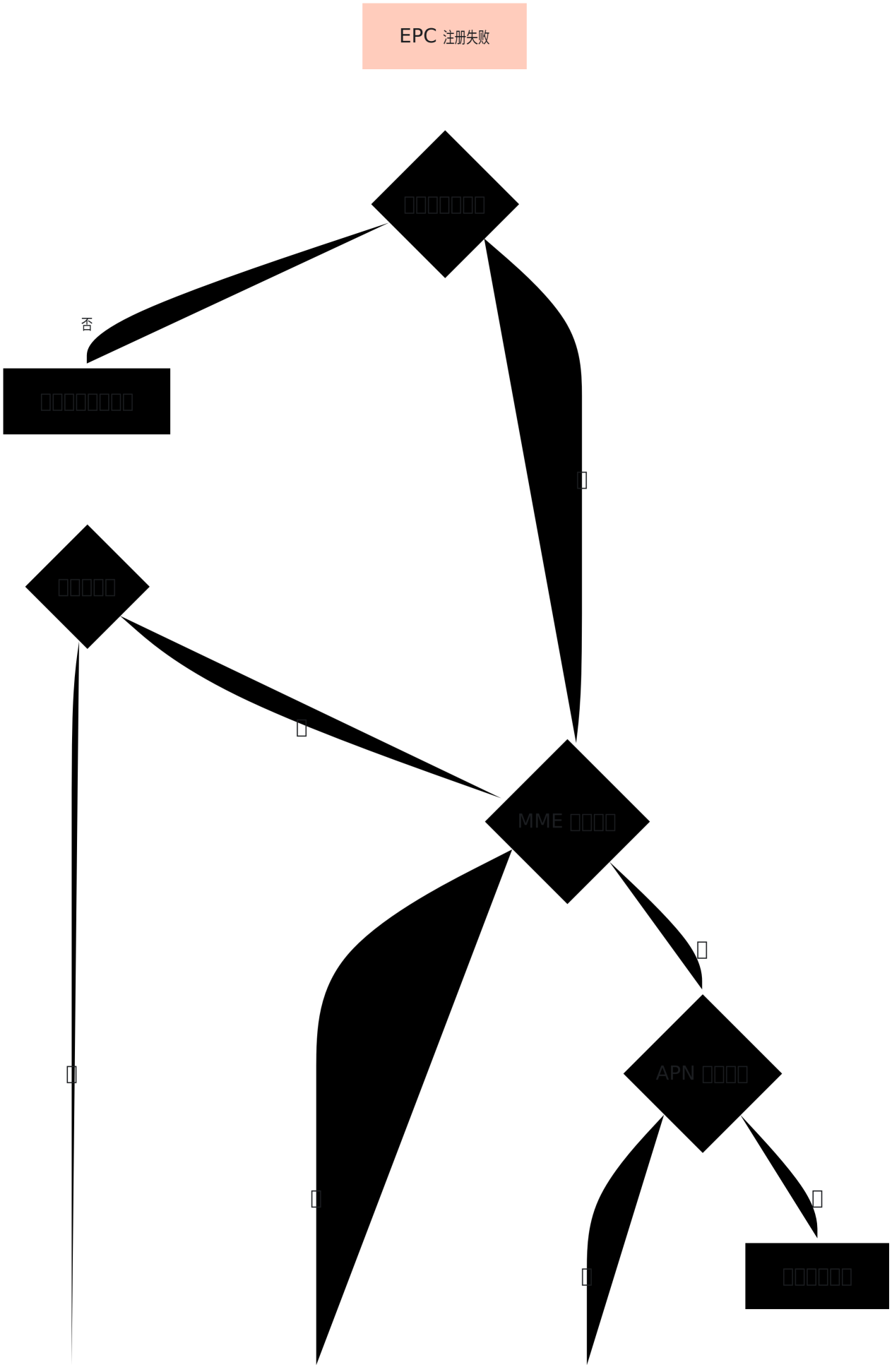
诊断步骤：

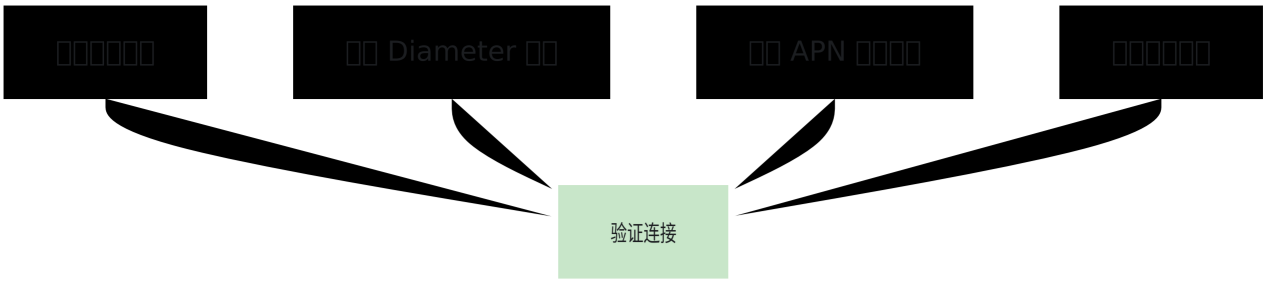
1. 从数据库查询用户状态
2. 检查□否存在过时的 MME 分配
3. 验证最后更新时间戳

解决方案：

- 清除用户状态（脱离程序）
- 重置用户状态中的服务 MME
- 可能需要用户断电重启

EPC □□□□□□□□□


EPC 注册失败
否



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□□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□否分配了 `ims_profile_id`

解决方案:

- 启用 IMS 为用户
- 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 □否支持 Cx 应用程序

解决方案：

- 修复 Diameter 连接问题 到 S-CSCF
- 验证 S-CSCF 配置

原因 **3**：缺少或无效的 **IFC** 模板

症状：

- 在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□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
- 请参阅 配置文件文档 以获取 IFC 示例

原因 **4**：**IMS** 漫游被拒绝

症状：

- IMS 在本地网络上工作
- 漫游时失败
- 数据漫游正常但 IMS 不正常

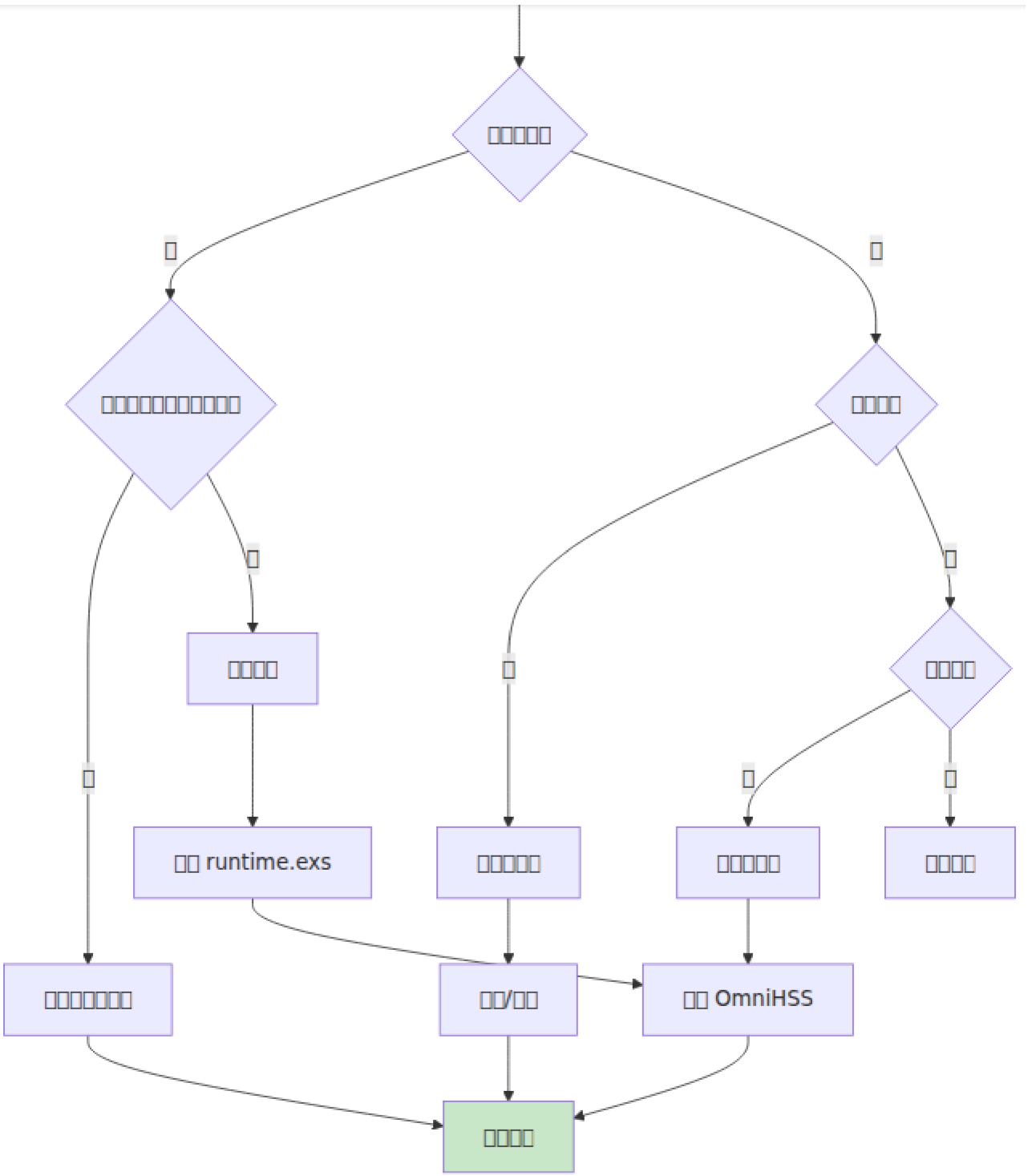
诊断步骤：

1. 检查漫游配置文件 IMS 操作
2. 验证漫游规则□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
- 呼叫立即掉线
- 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 ��连接

症状:

- 注册正常但呼叫失败
- 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 否支持 Rx 应用程序（OmniHSS PCRF 功能）

解决方案:

- 修复 Diameter 连接问题 到 P-CSCF
- 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 AAR（AA-请求）□否收到
3. 检查 AAA（AA-应答）响应

解决方案：

- 验证 P-CSCF □否发送 AAR 进行媒体授权
- 检查 OmniHSS Rx 应用程序配置
- 验证用户□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

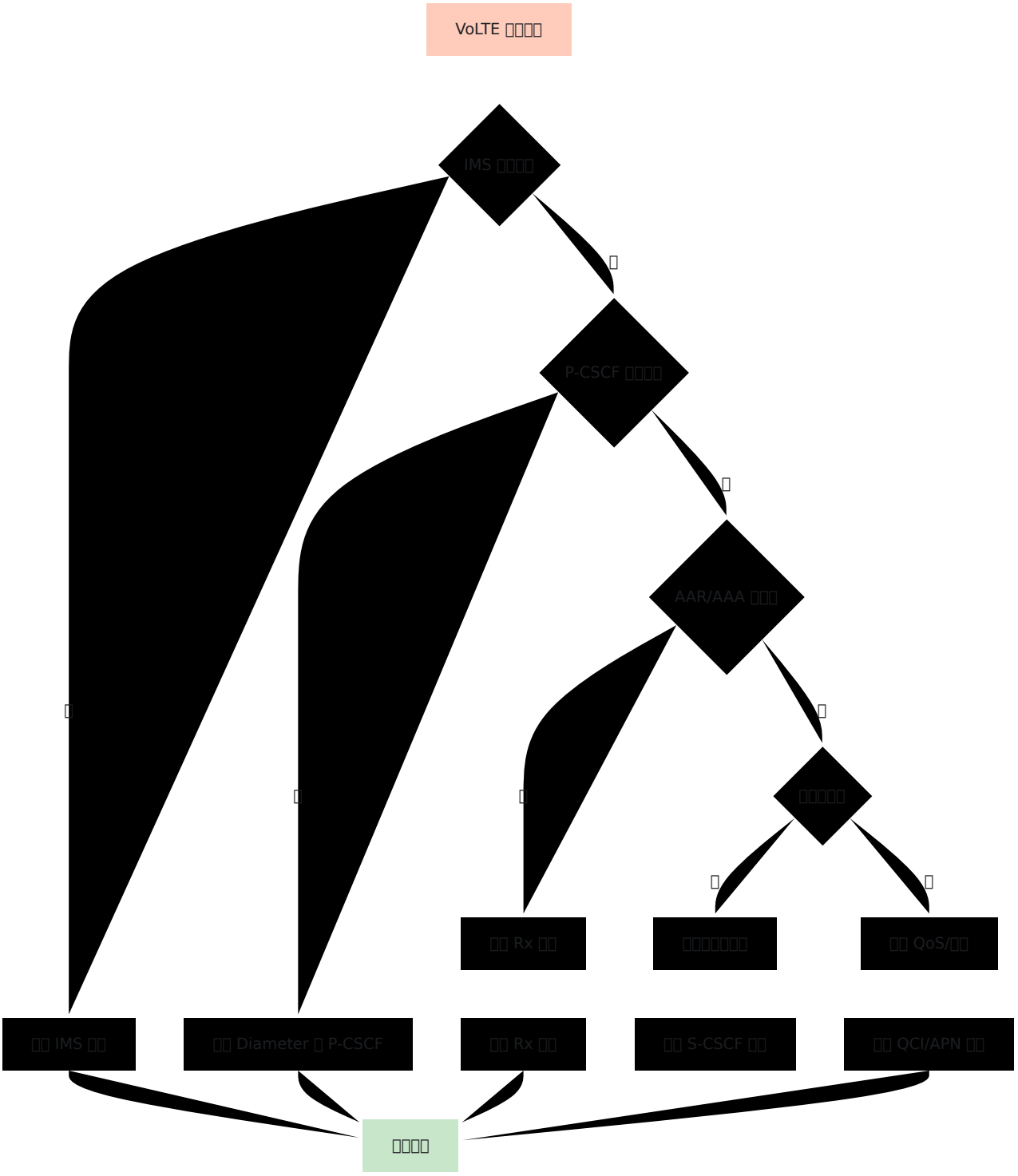
诊断步骤：

1. 检查语音 APN 的 APN QoS 配置文件
2. 验证 QCI □否正确设置（通常语音为 QCI 1）
3. 检查 P-GW □否连接 Gx（PCRF 功能）

解决方案：

- 验证 APN QoS 配置文件 □否适用于 IMS APN
- 确保语音承载配置为 QCI 1
- 如有需要，修复 Diameter 连接问题 到 P-GW

VoLTE □□□□□□□□□□


VoLTE □□□□
IMS □□□□
□
P-CSCF □□□□
□
AAR/AAA □□□
□
□
□
□□□□□
□
□
□□ Rx □□
□□□□□□□□
□□ QoS/□□
□□ IMS □□
□□ Diameter □ P-CSCF
□□ Rx □□
□□ S-CSCF □□
□□ QCI/APN □□
□□□□

漫游问题

症状

- 用户在本地工作但在漫游时不工作
- 某些漫游网络有效，其他无效
- 漫游数据正常但语音（或反之亦然）

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□否为空

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：

- 在特定网络上漫游始终失败
- 错误指示政策拒绝

诊断步骤：

1. 从用户设备或 MME 确定访问网络的 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 以允许访问网络：

```
curl -k -X POST https://hss.example.com:8443/api/roaming/rule \
  -H "Content-Type: application/json" \
  -d '{
    "roaming_rule": {
      "name": "允许访问网络",
      "mcc": "310",
      "mnc": "410",
      "data_action": "allow",
      "ims_action": "allow"
    }
  }'
```

原因 **3**：数据允许但 **IMS** 被拒绝

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: IMEI 正则表达式不正确

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（第一个匹配优先）

原因 2: MME 未发送 S13 请求

症状:

- EIR 检查从未发生
- 所有设备都被允许，无论规则如何

诊断步骤:

1. 检查 MME □否配置为使用 S13 接口
2. 验证 MME Diameter 对等体已连接
3. 检查 S13 应用程序支持
4. 查看 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 应用程序（16777252）
- 如有需要，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具有意外行为

诊断步骤:

1. 查询所有 EIR 规则
2. 检查□否存在捕获所有规则
3. 验证���则排序

解决方案:

- 添加默认规则，正则表达式为 `.*` 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□否缺少索引
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
- 请参阅 数据库问题

原因 **2**: 用户数量高

症状:

- 性能随时间下降
- 缓慢与用户增长相关
- 列表操作特别慢

诊断步骤:

1. 查询总用户数量

2. 检查表大小
3. 查看查询执行计划
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
- 考虑对大结果进行分页
- 如有需要，实施缓存

原因 3: Diameter 对等体问题

症状:

- Diameter 操作缓慢
- 对特定对等体超时
- 某些对等体快，其他慢

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□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□否存在内存泄漏
4. 查看 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，调查内存泄漏
- 在 runtime.exs 中调整 Erlang VM 内存设置
- 如果持续高，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未连接
- 过时的状态信息
- 位置信息不正确
- 无法脱离用户

常见原因和解决方案

原因 1: MME 崩溃/重启

症状：

- 用户显示的服务 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. 检查用户状态的服务 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 应在重启时发送取消位置

原因 **2**：未收到网络脱离

症状：

- 用户关闭电源但显示为已连接
- PDN 会话在数据库中保持
- 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）
3. 检查用户设备是否可达

解决方案：

- 状态将在用户再次连接时清除
- 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
- 外键违规
- 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：

- 识别并修复不一致的数据
- 可能需要手动清理数据库
- 如果损坏广泛，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□否□□在
4. 检查数据类型

解决方案:

- 修复请求数据以匹配 API 参考
- 确保包含所有必填字段
- 验证外键引用□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□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: 数据库连接问题

症状:

- 500 错误
- 所有 API 调用失败
- 数据库连接错误

解决方案：

- 请参阅 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：用户数量、活动会话、系统状态
2. **Diameter** 状态
    - URL: `https://[hostname]:7443/diameter`
    - 检查：所有关键对等体已连接
3. 应用健康
    - URL: `https://[hostname]:7443/application`
    - 检查：内存使用情况、进程数量、运行时��

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/1
```

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/1

# IMS 配置文件
curl -k https://hss.example.com:8443/api/ims/profile/1

# 漫游配置文件
curl -k https://hss.example.com:8443/api/roaming/profile/1
```

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：

```
# PostgreSQL
psql -h [DB_HOST] -U [DB_USER] -d [DB_NAME] -c "SELECT COUNT(*)
FROM subscriber;"

# MySQL
mysql -h [DB_HOST] -u [DB_USER] -p -e "SELECT COUNT(*) FROM
subscriber;" [DB_NAME]
```

日志分析

在日志中搜索特定 **IMSI**：

```
grep "001001123456789" /var/log/omnihss/omnihss.log
```

查找认���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

在以下情况下升级到工程/供应商支持：

1. 系统范围的故障，无法通过文档程序解决
2. 数据损坏或不一致的数据库状态
3. 怀疑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. 有关未记录行为的问题

提供的信息

升级时，请包括：

1. 详细症状 - 什么失败，何时，谁
2. 采取的步骤 - 您已经做了什么故障排除
3. 日志 - 显示问题的相关日志摘录
4. 配置 - runtime.exs 的相关部分（遮蔽敏感数据）
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）：

- 系统完全宕机
- 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）：

- 单个用户问题可以解决
- 可管理的性能下降
- 增强请求
- 文档问题

常见错误消息参考

认证错误

错误消息	原因	解决方案
"认证向量生成失败"	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"	对等体不支持所需应用程序	检查对等体应用程序
"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
"认证失败"	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 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

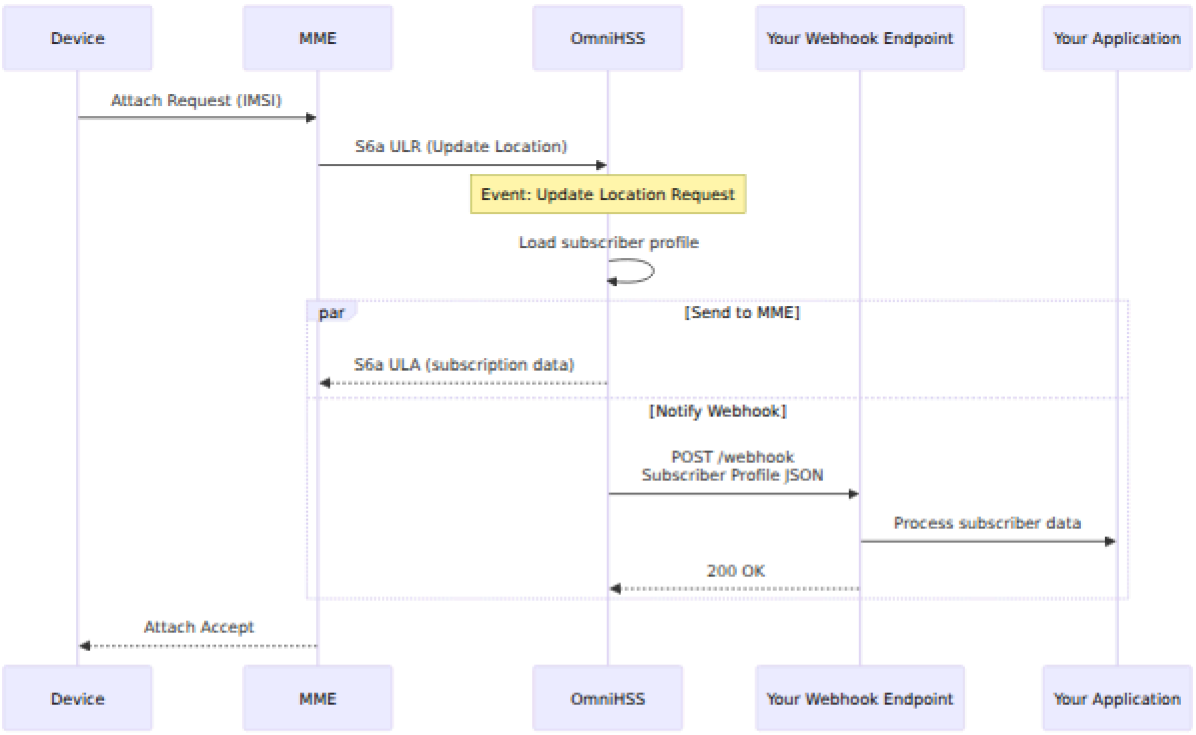
目录

概述

OmniHSS 支持 **webhooks** 以实时通知外部系统有关订阅者事件的信息。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，包含完整的订阅者配置文件数据。

什么是 **Webhooks**?

Webhooks HTTP 回调，允许 OmniHSS 在事件发生时将通知推送到您的应用程序，而不要求您的应用程序轮询 HSS API 以获取更改。



主要优势

- 实时通知 - 在订阅者事件发生时获取即时更新
- 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

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��返回 HTTP 200-299 以确认接收

交付保证

- 尽力而为的交付 - Webhooks 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 端点出现故障，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（设备关闭、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 配置文件

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": 1234,
    "imsi": "001001123456789",
    "enabled": true,
    "ims_enabled": true,
    "msisdns": [
      {"id": 1, "msisdn": "14155551001"},
      {"id": 2, "msisdn": "14155551002"}
    ],
    "sim": {
      "id": 5678,
      "iccid": "89911012000003204510",
      "is_esim": false
    },
    "key_set": {
      "id": 100,
      "amf": "8000"
    },
    "epc_profile": {
      "id": 1,
      "name": "Premium 100Mbps",
      "ue_ambr_dl_kbps": 100000,
      "ue_ambr_ul_kbps": 50000
    },
    "ims_profile": {
      "id": 1,
      "name": "Standard VoLTE"
    },
    "roaming_profile": {
      "id": 1,
      "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	额外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": 5678,
  "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 头部

头部	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": 1,
    "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/1 \
  -H "Content-Type: application/json" \
  -d '{
    "webhook": {
      "enabled": false
    }
  }'
```

删除 **Webhook**

```
curl -k -X DELETE https://hss.example.com:8443/api/webhook/1
```

Webhook 端点要求

您的 webhook 端点必须:

1. 接受 **POST** 请求，并带有 `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（见安全部分）

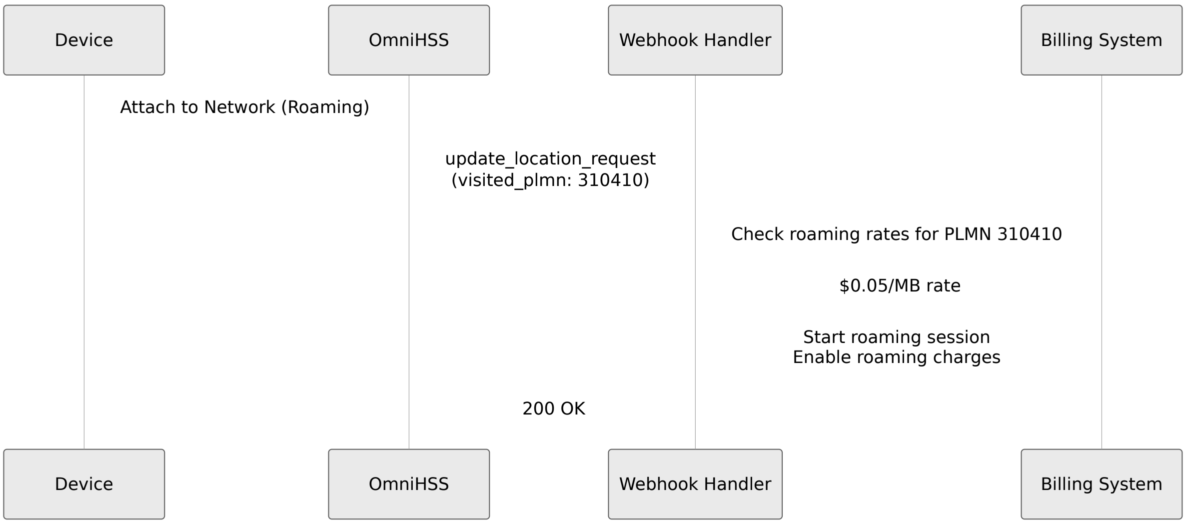
示例 **Webhook** 处理程序（**Node.js/Express**）：

```
const express = require('express');
const app = express();

app.post('/omnihss-webhook', express.json(), (req, res) => {
  const { event, subscriber, event_context } = req.body;

  console.log(`Received event: ${event}`);
  console.log(`Subscriber 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) {
  // 您的异步处理逻辑
  // 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
Attach to Network (Roaming)
update_location_request
(visited_plmn: 310410)
Check roaming rates for PLMN 310410
$0.05/MB rate
Start roaming session
Enable roaming charges
200 OK
Device
OmniHSS
Webhook Handler
Billing System

□□□

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

2. □□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□ □□□□□□□□□□

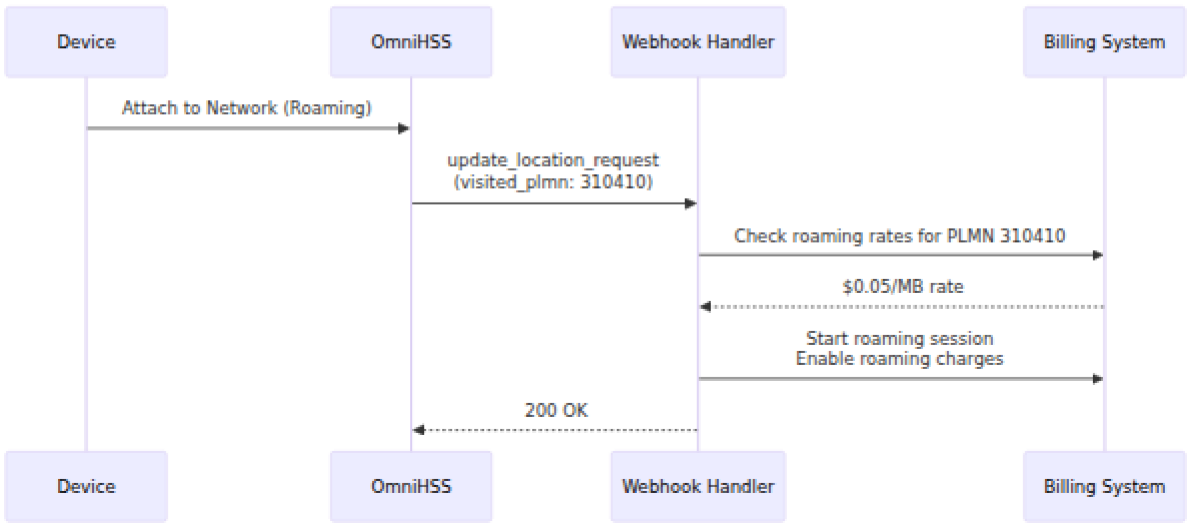
```
// Webhook 处理程序将数据馈送到分析平台
app.post('/omnihss-webhook', async (req, res) => {
  const { event, subscriber, event_context } = req.body;

  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（物理上不可能）
    - 操作：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
    - 操作：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）
    - 操作：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, 'Roaming to blocked
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, 'Impossible travel
detected')

    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。

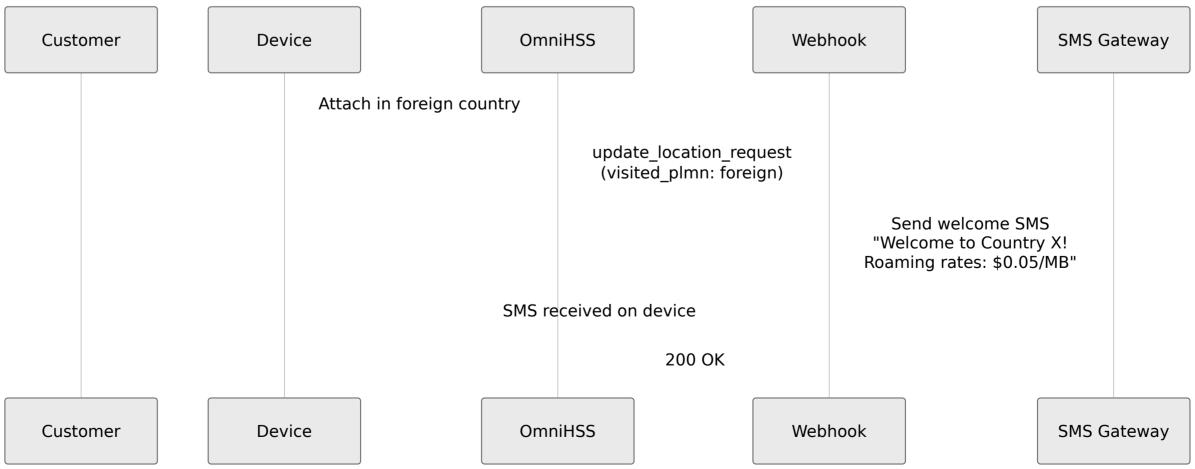
用例：当订阅者首次使用 VoLTE 时自动启用 IMS

```
app.post('/omnihss-webhook', async (req, res) => {
  const { event, subscriber } = req.body;

  if (event === 'ims_registration' && !subscriber.ims_enabled) {
    // 首次 IMS 用户 - 永久启用 IMS
    await omnihss.updateSubscriber(subscriber.id, {
      ims_enabled: true,
      custom_attributes: {
        ...subscriber.custom_attributes,
        volte_activated_at: new Date().toISOString()
      }
    });

    // 更新 CRM
    await crm.updateCustomer(subscriber.imsi, {
      features: ['volte']
    });
  }

  res.status(200).send();
});
```

5. 客户通知

向客户发送有关其服务的实时通知。

用例：当国际漫游时发送欢迎消息



示例通���:

- "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，接收他们切换 IMSI 时的通知。

```
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 切换以进行分析
    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(`Excessive IMSI switching: 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 - 将订阅者数据馈送到网络分析平台
- 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，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` 头部：

```
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: 'Invalid signature' });
  }

  // 处理 webhook...
  res.status(200).send();
});
```

最佳实践

1. 使用 **HTTPS** - 始终为 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. 安全凭证 - 在安全配置中存储 webhook 密钥（环境变量、密钥管理器）

数据隐私

Webhook 负载包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 - 使用 TLS 进行 webhook 传输
- 审计日志 - 记录所有 webhook 交付以确保合规

故障排除

Webhook 未接收

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：

- 确保事件类型包含在 webhook 的 `events` 列表中
- 示例：如果您想要 `ims_registration` 事件，请验证它在事件数组中

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 - 在响应后异步处理
app.post('/webhook', (req, res) => {
  res.status(200).send(); // 立即响应
  processData(req.body); // 异步处理
});
```

重复 **Webhooks**

症状:

- 相同事件多次交付
- `event_id` 对于重复交付□相同的

原因:

- 网络重试（尽管 OmniHSS 不重试，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:

使用 `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□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 头部
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
});
  }
});
```

缺少订阅者数据

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 - 某些配置文件可能□可选的（IMS、漫游）

2. 数据竞争条件 - 在事件触发和 webhook 发送之间更新了订阅者

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": 1,
  "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 →