OmniTWAG 构

述

OmniTWAG 是一个根据 3GPP TS 23.402 第 16 节定义的受信 WLAN 接入网关 (TWAG)。它将受信的 WLAN 接入点与演进分组核心 (EPC) 连接起来，提供身份验证、会话管理和数据平面隧道功能。

目录

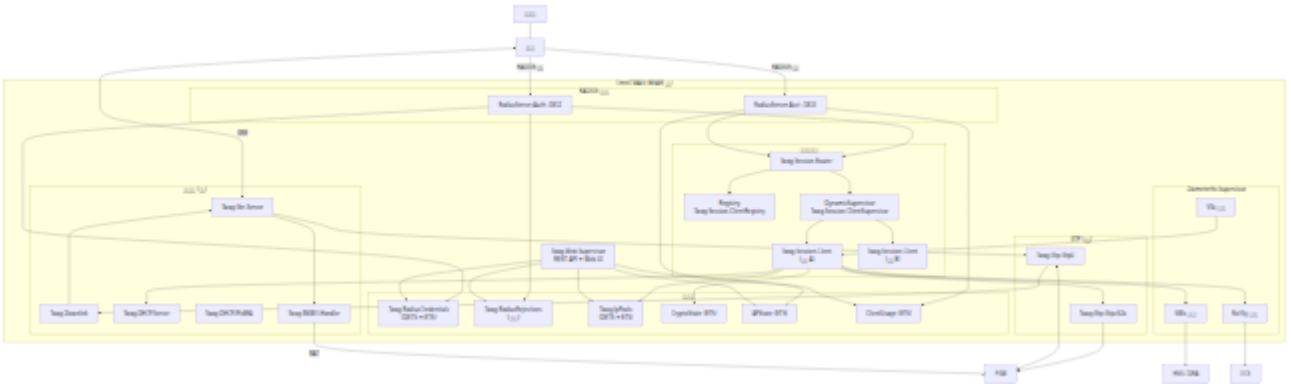
- 组件和过程架构
- 监督树
- 组件参考
- 操作模式
- AP 集成 (GRE 隧道)
- DHCP 行为
- 身份验证
- 会话生命周期 (家庭路由)
- Diameter 接口
- 配置摘要
- 密钥派生
- 相关页面

组件和过程构

OmniTWAG 作为一个单一的 BEAM (Erlang VM) 应用程序运行。顶级监督者 启动一组固定的共享服务，并根据配置启动一组可选的数据路径和接口子进程。每个身份的每个用户状态在一个 GenServer 中存在，根据需要创建。

主要组别包括:

- **Web** 和 **OAM**: REST API 和 Web 界面 (`Twag.Web.Supervisor`)。
- 每个身份会话: 每个用户的一个 `Twag.Session.Client` GenServer, 通过 `Registry` 查找并由 `DynamicSupervisor` 启动。 `Twag.Session.Router` 负责查找或创建。



- 共享存储：`Twag.RadiusCredentials`（持久共享密钥），`Twag.RadiusRejections`（内存中被拒绝的来源），`Twag.IpPools`（地址池），以及 ETS 支持的 `CryptoState`、`APState` 和 `ClientUsage` 存储。
- **RADIUS** 监听器：`RadiusServer.Auth`（端口 1812）和 `RadiusServer.Acct`（端口 1813）。
- **Diameter**：`DiameterEx.Supervisor` 处理 SWx（认证向量）、STa（重新认证和中止）和 Ro/Gy（在线计费）对等体。
- **GTP**：`Twag.Gtp.GtpcS2a`（控制）和 `Twag.Gtp.GtpU`（用户平面），仅在 `gtp.enabled` 为真时启动。
- 数据路径：`Twag.Gre.Server`、`Twag.Downlink`、`Twag.DHCP.Server`、`Twag.DHCP.IPv6RA` 和 `Twag.NSWO.Handler`，每个仅在其配置块启用时启动。

监督树

根监督者使用 `:one_for_one` 策略（在 60 秒内最多 10 次重启）。子进程按以下顺序启动，以便在第一个 RADIUS 数据包到达之前解析每个源 IP 的密钥：

标记为 `*` 的子进程仅在其配置块启用时启动（请参见 配置摘要）。

组件参考

组件	类型	启动时机
`Twag.Web.Supervisor`	监督者	始终
`DiameterEx.Supervisor`	监督者	始终
`Twag.Session.ClientRegistry`	注册表	始终
`Twag.Session.ClientSupervisor`	DynamicSupervisor	始终
`Twag.Session.Router`	模块	按需
`Twag.Session.Client`	GenServer	每个身份
`Twag.RadiusCredentials`	GenServer	始终
`Twag.RadiusRejections`	GenServer	始终
`Twag.IpPools`	GenServer	始终

组件	类型	启动时机
`RadiusServer.Auth`	GenServer	始终
`RadiusServer.Acct`	GenServer	始终
`CryptoState`	GenServer (ETS)	始终
`APState`	GenServer (ETS)	始终
`ClientUsage`	GenServer (ETS)	始终
`Twag.Gtp.GtpcS2a`	GenServer	`gtp.enabled`
`Twag.Gtp.GtpU`	GenServer	`gtp.enabled`
`Twag.Downlink`	GenServer	`downlink.enabled`
`Twag.Gre.Server`	GenServer	`gre.enabled`
`Twag.DHCP.Server`	GenServer	`dhcp.enabled`
`Twag.DHCP.IPv6RA`	GenServer	`ipv6_ra.enabled`



组件	类型	启动时机
`Twag.NSWO.Handler`	GenServer	`nswo.enabled`

操作模式

OmniTWAG 支持两种数据平面模式。会话使用的模式根据匹配的 AP 凭据解析（请参见 RADIUS 共享密钥），全局 `gtp` / `nswo` 配置作为默认值。

家庭路由模式（GTP 隧道）

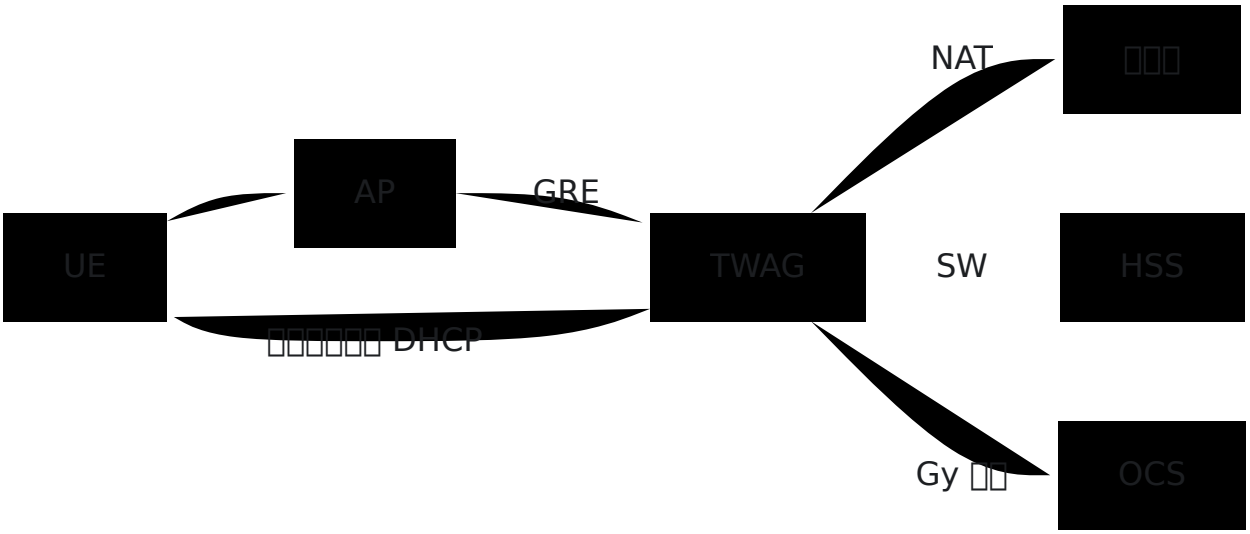
流量通过 GTP-C/GTP-U (S2a 接口) 隧道到 PGW。PGW 分配 UE IP 地址，应用策略/计费，并提供互联网访问。

Get an estima

- **UE IP**: 在创建会话响应 (PAA IE) 中由 PGW 分配。
- **IP** 交付: TWAG 通过 GRE 为 UE 提供 DHCP，分配 PGW 分配的 IP。
- 上行: AP 通过 GRE 发送 UE 流量。TWAG 将其封装到 GTP-U 发送到 PGW。
- 下行: PGW 通过 GTP-U 发送。TWAG 解封装并通过 GRE 发送到 AP，然后到 UE。
- 计费: 由 PGW (PCEF) 通过 Gy 处理到 OCS。
- 配置: `gtp.enabled: true`，`gre.enabled: true`。

本地突破模式（NSWO）

流量直接从 TWAG 退出到互联网。没有 PGW 参与。TWAG 执行 NAT/伪装，并可选择通过 Gy 自行进行在线计费。

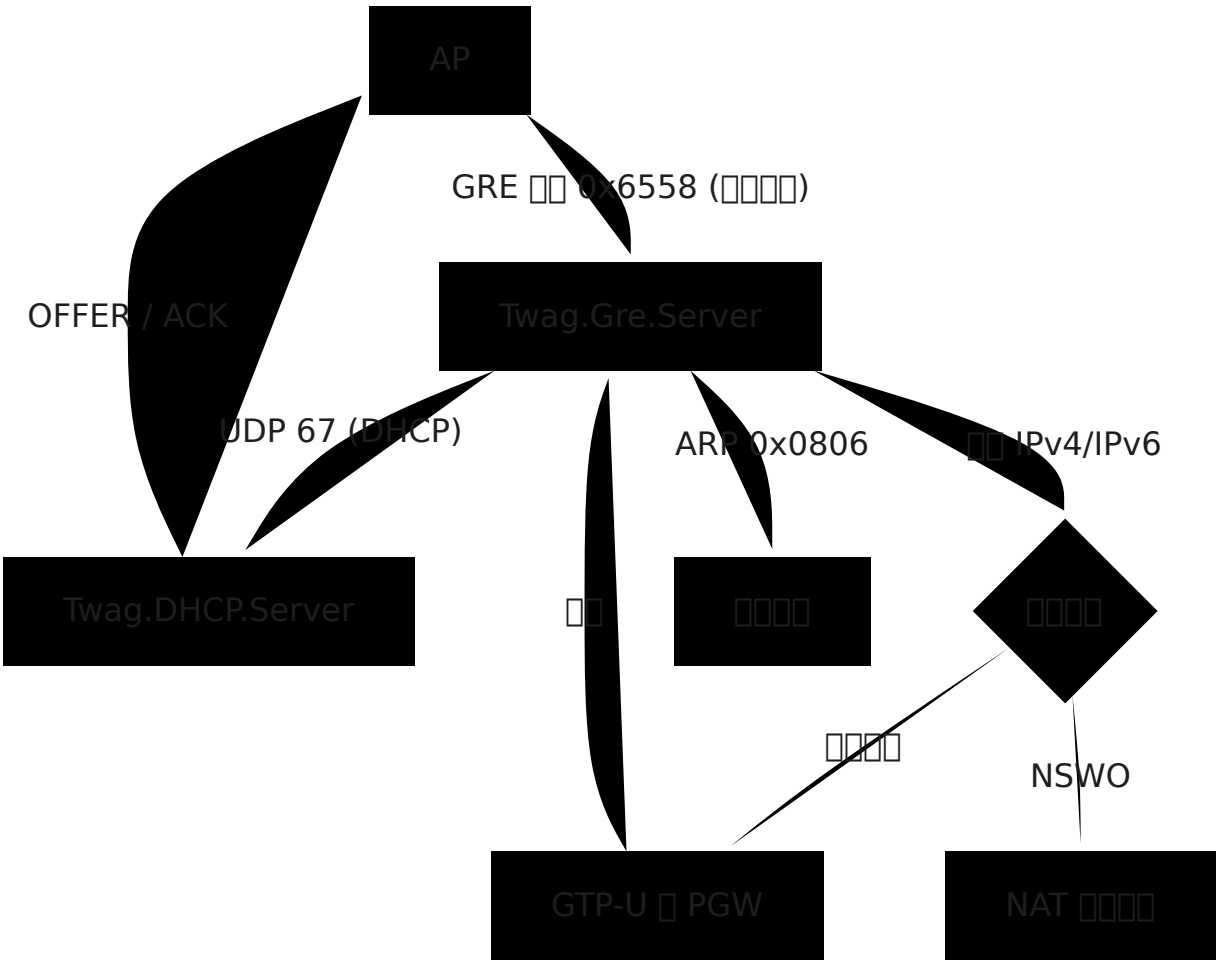


- **UE IP**: 从 TWAG 本地 IP 池 (`Twag.IpPools`) 中分配。
- **IP** 交付: TWAG 通过 GRE 为 UE 提供 DHCP，从池中分配。
- 上行: AP 通过 GRE 发送。TWAG 对互联网应用 NAT。
- 下行: 互联网路径通过 TWAG NAT 返回，然后通过 GRE 到 AP，然后到 UE。
- 计费: TWAG 通过 Gy 作为 PCEF 到 OCS（可选，默认禁用）。请参见 在线计费。
- 配置: `nswo.enabled: true`, `gtp.enabled: false`。

本地突破通过设置 `nswo.enabled: true`（并且 `gtp.enabled: false`）或通过凭据 `breakout_type` 按 AP 选择。当启用时，NSWO 处理程序启动，会话从本地池分配地址，而不是 PGW 分配的地址，上行直接 NAT/伪装到互联网。请参见 用户平面和数据路径 以获取数据路径的详细信息。

AP 集成（GRE 隧道）

AP 通过 GRE (IP 协议 47) 将 UE L2 帧隧道到 TWAG。这提供了 根据 TS 23.402 第 16.1.2 节所需的点对点链接。



Cisco AP 配置

```
dot11 ssid ons
   tunnel CWAG
   authentication open eap eap_methods
   authentication key-management wpa
   accounting acct_methods

dot11 tunnel CWAG
    type gre
    source {AP_IP}
    destination {TWAG_IP}
    mss 1410
    mtu 1500
```

GRE 数据流

AP
GRE 协议 0x6558 (以太网桥)
OFFER / ACK
Twag.Gre.Server
UDP 67 (DHCP)
ARP 0x0806
□□ IPv4/IPv6
Twag.DHCP.Server
下□
□□□□
□□□□
家□□由
NSWO
GTP-U □ PGW
NAT □□□□

DHCP 行为

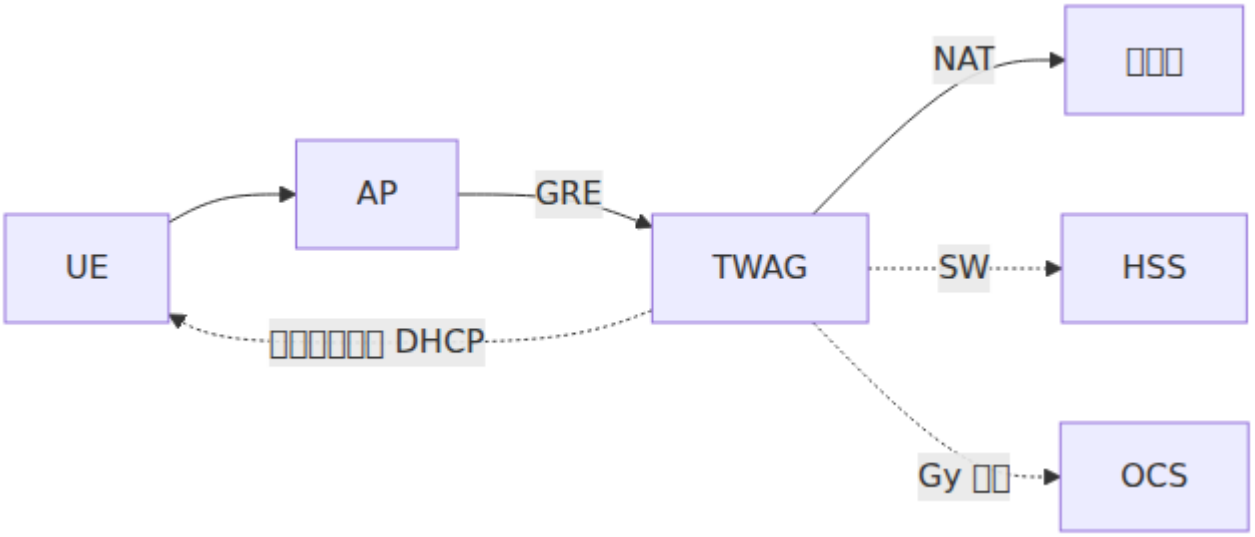
模式	IP 来源	DHCP 服务器	备注
家庭路由	PGW (创建会话响应中的 PAA)	TWAG 预分配 PGW IP，通过 GRE 提供 DHCP	PGW PCO 的 DNS
本地突破	TWAG 本地池 (`Twag.IpPools`)	TWAG 在 DISCOVER 时从绑定池中分配 (池分配)	池配置中的 DNS

TWAG DHCP 服务器:

1. 通过 GRE 隧道从 UE 接收 DHCP DISCOVER。
2. 提供 OFFER。在家庭路由模式下，这是会话预分配的 PGW 分配的 IP。在本地突破模式下，当 MAC 绑定到池时，服务器从池中分配一个新地址。
3. 处理 REQUEST/ACK 完成交换。
4. 通过 GRE 隧道将所有 DHCP 流量传递到 AP 和从 AP。

身份验证

OmniTWAG 作为一个组合的 TWAG 和 3GPP AAA 服务器，执行 EAP-AKA 本地身份验证，并通过 Diameter SWx 从 HSS 获取认证向量。每个用户的 EAP 状态机在其自己的 `Twag.Session.Client` 中运行。



认证流程

在处理任何数据包之前，RADIUS 监听器会根据 `Twag.RadiusCredentials` 授权源 IP。没有匹配凭据子网的源将被丢弃，并记录为 `unauthorized_source`。匹配但未通过消息验证的源记录为 `bad_secret`。请参见 RADIUS 拒绝。

认证模式

模式	配置	向量来源	**EAP** 处理
`:swx`	`auth_mode: :swx`	通过 SWx MAR/MAA 从 HSS	本地 (TWAG)
`:local`	`auth_mode: :local`	本地测试密钥 (Milenage)	本地 (TWAG)

EAP 方法

可通过 `eap_method` 配置。默认值为 `:eap_aka`。如果 UE 拒绝提供的方法，TWAG 会协商 UE 支持的首选方法。还支持 EAP-SIM 和 EAP 重新认证 (ERP)。请参见 身份验证。

会话生命周期（家庭路由）


EAP □□ -> Access-Accept (MSK □ AP)
AP □□ Accounting-Start
□ Twag.Session.Client □□ □□□□□
GTP-C □□□□□□ -> PGW
PGW □□ IP□□□□□□□
□□ GTP-U □□ (□□ + □□ TEID)
□□ DHCP □□ UE MAC □□ □ PGW IP
UE DHCP Discover -> OFFER □□ PGW IP





会话因多种原因被拆除，每种原因都记录在 `twag.session.teardown` 指标中：`accounting_stop`、`admin_disconnect`、`network_disconnect`（STa ASR）、`identity_timeout`（认证开始但从未完成）和 `liveness_timeout`。请参见 指标参考 和 会话。

Diameter 接口

接口	应用 ID	方向	目的
SWx	16777265	TWAG -> HSS	获取认证向量 (MAR/MAA)
STa	16777250	入站到 TWAG	RAR (重新认证)、ASR (中止)、STR (终止)
Ro/Gy	4	TWAG -> OCS	在线计费 (仅 NSWO)

DRA 路由

DRA 必须将 SWx 路由到 HSS：

```
%{rule_name: "swx_to_hss", match: :all,
  filters: [{:application_id, 16777265}, {:avp, {296, "
{realm}"}}],
  route: %{peers: ["{hss_host}"]}}
```

配置摘要

每个参数在以下块的表中都有文档。

```
config :omnitwag,
  auth_mode: :swx,
  eap_method: :eap_aka,
  identity_timeout_ms: 600_000,

  gtp: %{
    enabled: true,
    pgw_ip: "10.179.1.21",
    local_ip: "10.179.1.41",
    pgw_port: 2123,
    gtpu_port: 2152,
    echo_enabled: true
  },

  gre: %{
    enabled: true,
    local_ip: "10.179.1.41"
  },

  downlink: %{
    enabled: true
  },

  dhcp: %{
    enabled: true,
    interface: "gre"
  },

  ipv6_ra: %{
    enabled: false
  },

  nswo: %{
    enabled: false,
    local_pool: "10.100.0.0/16",
    nat_interface: "eth0"
  },

  online_charging: %{
    mode: :disabled
  },

  rate_limit: %{
```

```
    max_failures: 10,
    window_ms: 60_000
  },

  prometheus: %{
    port: 9568
  }
```

参数	默认值	描述
`auth_mode`	`:swx`	`:swx` 从 HSS 获取向量；`:l` Milenage 测试密钥。
`eap_method`	`:eap_aka`	提供的 EAP 方法（`:eap_ak` `:eap_aka_prime`）。
`identity_timeout_ms`	`600000`	会话在身份阶段可以停留的时间，超过[因 `identity_timeout`
`gtp.enabled`	`true`	启动 GTP-C/GTP-U 子进程[
`gtp.pgw_ip`	`127.0.0.1`	GTP-C/GTP-U 的 PGW [
`gtp.local_ip`	none	GTP 的本地 TWAG 地址。
`gtp.pgw_port`	`2123`	PGW GTP-C 端口。
`gtp.gtpu_port`	`2152`	本地 GTP-U 端口。
`gtp.echo_enabled`	`true`	向 PGW 发送 GTP 回[
`gre.enabled`	`false`	启动 AP 隧道的 GRE 服务器。
`gre.local_ip`	`0.0.0.0`	GRE 服务器绑定的本地地址。
`downlink.enabled`	`true`	启动下行转发器。
`dhcp.enabled`	`false`	启动 DHCPv4 服务器。
`dhcp.interface`	none	隧道模式下为 `"gre"`，或直接交[
`ipv6_ra.enabled`	`false`	启动 IPv6 路由器广告和 DHCF
`nswo.enabled`	`false`	启动 NSWO 本地突破处理程序。

参数	默认值	描述
`nswo.local_pool`	none	本地突破地址的后备 CIDR。
`nswo.nat_interface`	none	NAT/伪装的出口 NIC。
`online_charging.mode`	`:disabled`	Gy 模式: `:disabled`、`:auth` 或 `:online_charging`
`rate_limit.max_failures`	`10`	在丢弃 Access-Requests 的认证失败次数。
`rate_limit.window_ms`	`60000`	以毫秒为单位的速率限制窗口。
`prometheus.port`	`9568`	Prometheus 指标端口。
`radius_config.acct_interim_interval`	`300`	在 Access-Accept 中向 A 间隔（秒）。

密钥派生

对于 EAP-AKA', 根据 3GPP TS 33.402 第 6.2 节和附录 A.2, 以及 RFC 5448:

步骤	节点（`:swx` 模式）	结果
1	TWAG	SWx MAR，`SIP-Authentication-Scheme` = `EAP-AKA'` 和 `ANID` = 网络名称
2	HSS	RAND，AUTN，AMF 分离位 1，XRES
3	HSS	`CK'`，`IK'` = KDF(`CK`，`IK`，FC 0x20，网络名称，SQN XOR AK)
4	TWAG	`MK` = PRF'(`IK'`，`CK'`，"EAP-AKA'"，身份)
5	TWAG	从 `MK` 派生 `K_aut`，`MSK`，`EMSK`

在 `:local` 模式下，TWAG 也运行步骤 2 和 3，使用配置的测试密钥。请参见 身份验证 以获取完整的 派生过程。

对于 EAP-AKA (RFC 4187)：

```
HSS 供: RAND, AUTN, XRES, CK, IK
TWAG 派生: MK = SHA1(身份 || IK || CK)
               K_encr, K_aut, MSK, EMSK 从 PRF(MK)
```

网络名称 (ANID) 根据 TS 24.302 第 8.1.1.2 节为 `WLAN`。请参见 网络名称。

相关页面

- 身份验证
- RADIUS 共享密钥
- RADIUS 拒绝
- IP 池
- 会话
- 每个 AP 的计费
- 用户平面和数据路径
- 在线计费
- 指标参考

认证 (EAP-AKA / EAP-AKA' / EAP-SIM)

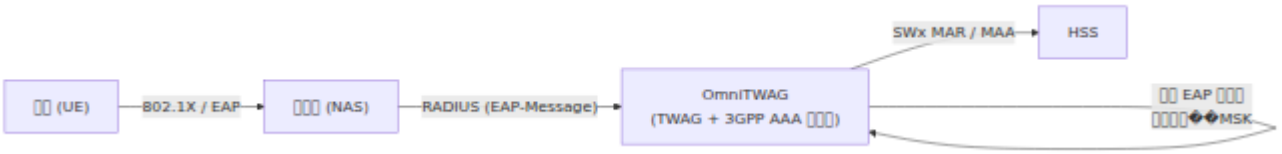
概述

OmniTWAG 使用 EAP 方法通过 RADIUS 在接入点和 TWAG 之间对每个用户进行身份验证。TWAG 作为一个结合了 TWAG 和 3GPP AAA 服务器的角色。它运行 EAP 状态机，并通过 Diameter SWx 从 HSS 获取认证向量，或从本地测试密钥获取。在成功后，TWAG 将主会话密钥 (MSK) 作为 MPPE 密钥通过 RADIUS Access-Accept 发送给接入点。接入点随后与用户完成 802.11 四次握手。

有关更广泛的系统上下文，请参见 架构。有关每个接入点如何解析 RADIUS 共享密钥的信息，请参见 RADIUS 共享密钥。认证首先运行。每个接入点的分流和计费策略在认证成功后应用。有关已认证会话及其状态生命周期的信息，请参见 会话。有关 TWAG 在认证后创建的 S2a GTP-C 会话的信息，请参见 S2a GTP-C。有关成功认证后计费的信息，请参见 在线计费 (Gy)。

目录

- 概述
- 支持的 EAP 方法
- EAP 状态机和阶段
- 认证序列
- 认证向量来源
- 身份和 NAI 处理
- 密钥派生和交付
- 重新同步
- 快速重新认证 (ERP)
- 故障处理和结果代码
- 配置
- 可观察性
- 故障排除



概述

接入点与用户启动 IEEE 802.1X EAP 交换。接入点将每个 EAP 消息封装在 RADIUS `EAP-Message` 属性中，并将其发送到 TWAG。TWAG 为每个用户身份运行一个认证状态机。状态机将 EAP 交换驱动到结果，然后将三种结果之一返回给 RADIUS 层。

结果	**RADIUS** 回复	意义
挑战	`Access-Challenge`	交换需要另一个 EAP 回合。
接受	`Access-Accept`	认证成功。回复携带 MSK 作为 MPPE 密钥。
拒绝	`Access-Reject`	认证失败。回复携带 EAP-Failure。

TWAG 为每个身份保持加密状态。它在 EAP 回合之间保持预期响应 (XRES)、认证密钥 (`K_aut`)、RAND、MSK 和扩展 MSK (EMSK)。在 TWAG 交付 MPPE 密钥后，TWAG 会将敏感密钥材料清零。

支持的 **EAP** 方法

TWAG 支持三种 EAP 方法。每种方法使用相同的 TLV 属性格式，但具有不同的 EAP 类型值和不同的密钥层次结构。

方法	**EAP** 类型	规范	向量	**K_aut**	**MAC**	密钥派生
EAP-AKA'	50 (`0x32`)	RFC 5448	UMTS 五元组	32 字节	HMAC-SHA-256, 截断为 16 字节	`CK'`/`IK'` 然后 PRF' (HMAC-SHA-256)
EAP-AKA	23 (`0x17`)	RFC 4187	UMTS 五元组	16 字节	HMAC-SHA1, 截断为 16 字节	`MK = SHA1(Identity | IK | CK)` 然后 FIPS 186-2 PRF
EAP-SIM	18 (`0x12`)	RFC 4186	2 或 3 GSM 三元组	16 字节	HMAC-SHA1, 截断为 16 字节	`MK = SHA1(Identity | Kc... | NONCE_MT | version)` 然后 FIPS 186-2 PRF

方法选择和协商

TWAG 提供默认方法。使用 `eap_method` 配置键设置该方法。默认值为 `:eap_aka_prime` (EAP-AKA')。

用户可以通过 EAP-NAK (EAP 类型 3) 拒绝所提供的方法。NAK 列出了用户首选的 EAP 类型。TWAG 读取该列表并选择第一个它支持的类型。TWAG 在 NAK 中支持这些类型: `50` (EAP-AKA')、`23` (EAP-AKA) 和 `18` (EAP-SIM)。

- 如果 NAK 列出了一个受支持的类型，TWAG 将使用该方法重新启动交换。对于 EAP-AKA' 和 EAP-AKA, TWAG 发送带有 `AT_PERMANENT_ID_REQ` 的 AKA-Identity 请求。对于 EAP-SIM, TWAG 发送 EAP-SIM/Start 请求。
- 如果 NAK 没有列出受支持的类型，TWAG 将拒绝用户。

EAP 状态机和阶段

TWAG 在每个用户身份内运行一个状态机，位于每个身份的客户端进程中。状态机跟踪 EAP 阶段。该阶段驱动 TWAG 期望的下一个 EAP 消息以及它发出的遥测。

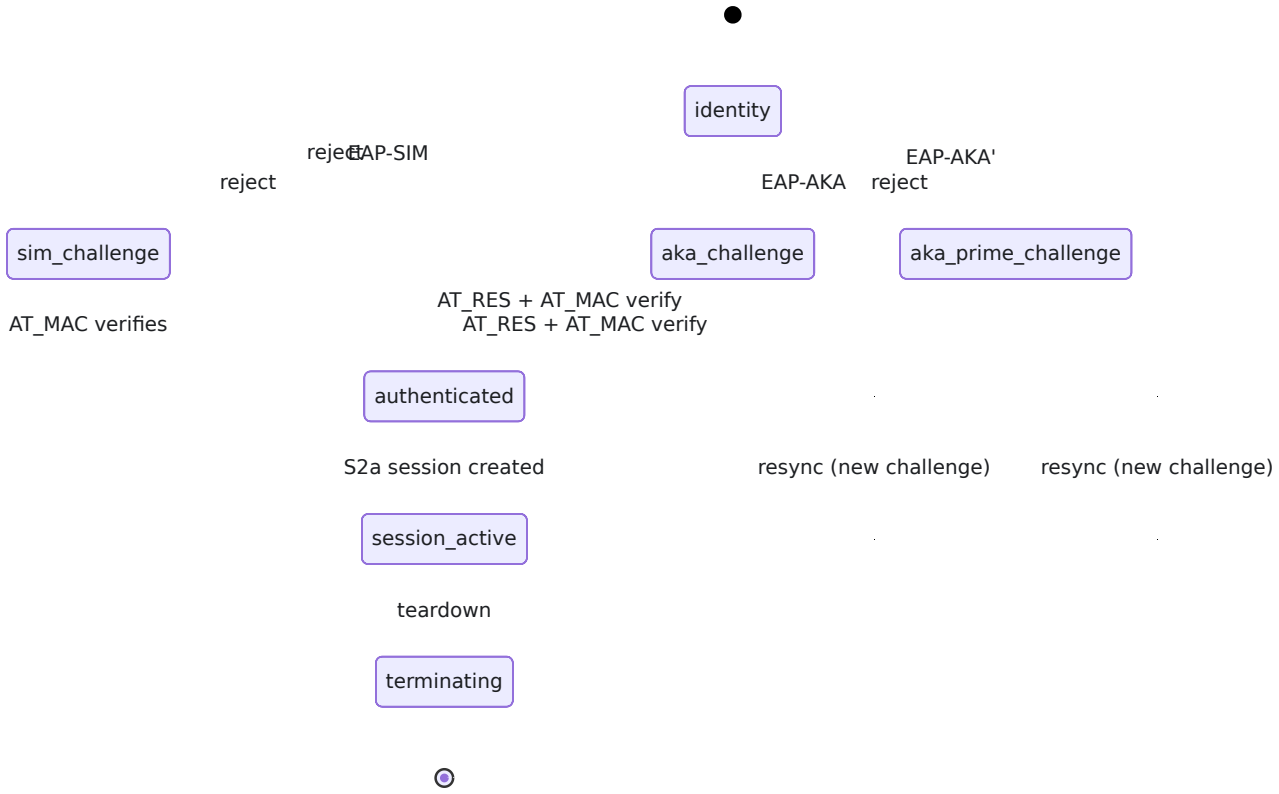
阶段（`eap_phase`）	意义
`identity`	初始阶段。TWAG 期望一个 EAP-Identity，或第一个 AKA-Identity 或 EAP-SIM/Start 回合。失败的回合将状态返回到 `identity`。
`aka_challenge`	TWAG 发送了一个 EAP-AKA 挑战并等待 AKA-Challenge 响应。
`aka_prime_challenge`	TWAG 发送了一个 EAP-AKA' 挑战并等待 AKA-Challenge 响应。
`sim_challenge`	TWAG 发送了一个 EAP-SIM 挑战并等待 SIM/Challenge 响应。
`authenticated`	EAP 交换成功。TWAG 拥有 MSK，并即将建立 S2a 会话。
`session_active`	S2a 会话已建立。用户可以传递用户数据。RADIUS 计费也可以将会话提升到此阶段。

代码中还存在两个进一步的阶段值：

- `resyncing` 保留为正在进行的认证阶段。在实践中，重新同步回合重建挑战并将阶段返回到 `aka_challenge` 或 `aka_prime_challenge`，因此状态不会停留在 `resyncing`。
- `terminating` 标记正在拆除的会话。

在身份超时后，停留在进行中的认证阶段（`identity`、`aka_challenge`、`aka_prime_challenge`、`sim_challenge` 或 `resyncing`）的会话将被清除。使用 `identity_timeout_ms` 设置超时。请参见 配置。

会话 页面将这些阶段映射到 UI 和 API 中显示的会话认证状态。



认证序列

完整序列包含身份阶段和挑战阶段。下面的示例显示 EAP-AKA'。EAP-AKA 具有相同的结构。EAP-SIM 使用 Start 回合代替 AKA-Identity 回合，并且不发送 `AT_RES`。

身份阶段

1. 用户发送一个 EAP-Identity 响应，携带网络接入标识符 (NAI)。
2. 对于 EAP-AKA' 或 EAP-AKA，TWAG 回复一个携带 `AT_PERMANENT_ID_REQ` 的 AKA-Identity 请求。这请求用户提供永久身份。
3. 对于 EAP-SIM，TWAG 回复一个携带 `AT_VERSION_LIST` (仅版本 1) 的 EAP-SIM/Start 请求。
4. 用户返回永久身份 (AKA-Identity 和 `AT_IDENTITY`)，或对于 EAP-SIM 的 Start 响应携带 `AT_NONCE_MT`。

挑战阶段

1. TWAG 从身份中提取 IMSI 并获取认证向量。请参见 认证向量来源。
2. TWAG 派生会话密钥，包括 MSK。请参见 密钥派生和交付。
3. TWAG 构建挑战。对于 EAP-AKA'，挑战携带 `AT_RAND`、`AT_AUTN`、`AT_KDF_INPUT` (网络名称)、`AT_KDF`、`AT_BIDDING` 和 `AT_MAC`。对于 EAP-AKA，挑战携带 `AT_RAND`、`AT_AUTN` 和 `AT_MAC`。对于 EAP-SIM，挑战携带 `AT_RAND` (2 或 3 个 RAND) 和 `AT_MAC`。
4. 用户验证网络 (AUTN) 和 MAC，然后返回响应。
5. TWAG 验证响应。请参见 响应验证。
6. 在成功���，TWAG 返回 EAP-Success 和 MSK。在失败时，TWAG 返回 EAP-Failure。

响应验证

TWAG 以固定顺序验证挑战响应。EAP-AKA 和 EAP-AKA' 的顺序相同。

1. 状态检查。如果存储的 `K_aut` 或 XRES 缺失，TWAG 拒绝。
2. 属性检查。如果 `AT_RES` 或 `AT_MAC` 缺失，TWAG 拒绝。
3. **MAC** 检查。 TWAG 将 `AT_MAC` 字段清零，使用 `K_aut` 重新计算数据包的 MAC，并进行比较。EAP-AKA 使用 HMAC-SHA1。EAP-AKA' 使用 HMAC-SHA-256。两者都截断为 16 字节。
4. 响应检查。 TWAG 将 `AT_RES` 与存储的 XRES 进行比较。比较是 常量时间 的，以防止时间攻击。TWAG 使用长度检查和常量时间字节比较。
5. 结果。如果 MAC 和响应都匹配，TWAG 接受。否则，TWAG 拒绝。

对于 EAP-SIM，用户不发送 `AT_RES`。`AT_MAC` 检查是用户持有正确密钥的唯一证明。如果 MAC 验证成功，TWAG 接受。

认证向量来源

TWAG 从两个来源之一获取认证向量。使用 `auth_mode` 配置键设置源。

模式	**`auth_mode`**	向量来源	使用
SWx	`:swx`	HSS，通过 Diameter SWx (MAR/MAA)	生产
本地	任何非 `:swx` 的值（默认 `:local`）	从预配置测试密钥生成的本地 Milenage	测试

在这两种模式下，TWAG 都在本地运行 EAP 状态机。只有向量的来源不同。任何不是 `:swx` 的 `auth_mode` 值都会选择本地 Milenage 路径。

> **STa** 直通是一个单独的路径。 `Twag.Auth.StaAuth` 提供程序可以通过 Diameter STa (DER/DEA) 将原始 EAP 有效负载中继到 3GPP AAA 服务器，因此 AAA 服务器运行 EAP-AKA 交换并返回 MSK。该提供程序存在，但不在实时 RADIUS 认证路径上。实时路径通过 `:swx` 或 `:local` 获取向量，并在 TWAG 上运行 EAP 状态机。

SWx 到 HSS

在 `:swx` 模式下，TWAG 通过 SWx 接口向 HSS 发送多媒体认证请求 (MAR) (3GPP TS 29.273，应用 ID `16777265`)。MAR 携带：

- `User-Name`：IMSI。
- `SIP-Auth-Data-Item`，其中 `SIP-Authentication-Scheme` 设置为正在使用的 EAP 方法：`EAP-AKA'` 或 `EAP-AKA`。对于重新同步，该项还携带 `SIP-Authorization`。这包含 RAND 后跟 AUTS。
- `SIP-Number-Auth-Items`：1。
- `ANID`：网络名称。默认值为 `WLAN`。对于 `EAP-AKA'`，HSS 使用它来派生 `CK'`/`IK'`。请参见 3GPP TS 29.273 第 8.1.2.1 条。
- `RAT-Type`：0。
- `Auth-Session-State`：1 (NO_STATE_MAINTAINED)。
- `Vendor-Specific-Application-Id`，其中 SWx Auth-Application-Id 为 `16777265`，Vendor-Id 为 `10415`。

HSS 返回多媒体认证应答 (MAA)。当 `Result-Code` 为 `2001` (成功) 时，TWAG 读取第一个 `SIP-Auth-Data-Item` 并提取向量：

- `SIP-Authentication-Scheme` = 向量的方案：`EAP-AKA'` 或 `EAP-AKA`。
- `SIP-Authenticate` = RAND (16 字节) 后跟 AUTN (16 字节)。
- `SIP-Authorization` = XRES。
- `Confidentiality-Key` = CK 对于 `EAP-AKA`，或 CK' 对于 `EAP-AKA'`。
- `Integrity-Key` = IK 对于 `EAP-AKA`，或 IK' 对于 `EAP-AKA'`。

如果 `Result-Code` 不是 `2001`，或者向量字段缺失，TWAG 将该应答视为认证拒绝。TWAG 用 EAP-Failure 拒绝用户，并记录拒绝，而不是崩溃。

如果 TWAG 请求 `EAP-AKA'` 而向量不是 EAP-AKA' 向量，TWAG 也会拒绝用户。这适用于方案为 `EAP-AKA`，或 AUTN 中的 AMF 分离位为 0。请参见 EAP-AKA' 密钥层次结构。

SWx 错误信号 (Experimental-Result)

SWx 在分组的 `Experimental-Result` AVP (Vendor-Id `10415`) 中信号大多数错误，而不是在基础 `Result-Code` 中。因此，失败的 MAA 可能根本不携带基础 `Result-Code`。TWAG 也读取 `Experimental-Result-Code`，并记录可读的名称，以便操作员看到真正的原因。有关代码的信息，请参见 故障排除。

一个常见的情况是 `DIAMETER_ERROR_USER_UNKNOWN` (5001) 或 `DIAMETER_ERROR_USER_NO_NON_3GPP_SUBSCRIPTION` (5450): 用户未在 SWx 上启用非 3GPP 访问，因此认证失败，即使同一用户在 S6a 上可能已知。

DRA 必须将 SWx 应用程序路由到 HSS。请参见 架构: DRA 路由。

本地 Milenage

在 `:local` 模式下，TWAG 使用从预配置测试密钥 (`test_keys`) 生成的 Milenage 算法 (3GPP TS 35.206) 来生成向量。密钥为 `ki`、`opc`、`amf` 和 `sqn`。TWAG 在过程中计算 RAND、AUTN、XRES、CK 和 IK。仅在实验室测试中使用此模式。

EAP-SIM 的向量

EAP-SIM 需要 2 或 3 个 GSM 三元组。TWAG 从相同的 Milenage 向量中派生三元组。对于每个向量，TWAG 取 RAND，取 XRES 的前 4 字节作为 SRES，并使用 c3 转换函数从 CK 和 IK 派生 GSM `Kc` (3GPP TS 33.102 附录 C.3): `Kc = CK[0..7] XOR CK[8..15] XOR IK[0..7] XOR IK[8..15]`。

身份和 NAI 处理

用户身份是网络接入标识符 (NAI)。NAI 的形式为 `{username}@{realm}`，例如 `0{IMSI}@wlan.mnc{MNC}.mcc{MCC}.3gppnetwork.org`。

IMSI 提取

TWAG 通过两个步骤从 NAI 中提取 IMSI:

1. TWAG 获取用户名，即 `@` 前的部分。
2. TWAG 删除前导身份类型数字。前导 `0` (永久身份，EAP-AKA/AKA') 或前导 `6` 映射到裸 IMSI。剩余的数字即为 IMSI。

TWAG 接受 14 或 15 位的 IMSI。IMSI 是 SWx MAR 中的 `User-Name`。TWAG 还使用 IMSI 派生 GTP-C 会话的 MCC 和 MNC: MCC 是前 3 位，MNC 是接下来的 2 位。

网络名称 **(EAP-AKA')**

EAP-AKA' 将密钥绑定到网络名称。3GPP 将此值称为 接入网络标识符 **(ANID)**。TWAG 在 `AT_KDF_INPUT` 中将网络名称发送给用户。TWAG 在 SWx MAR 的 `ANID` AVP 中将相同的字符串发送给 HSS。HSS 和用户都使用它来派生 `CK'`/`IK'`，因此这两个值必须相同。

默认值为 `WLAN`。这是 3GPP TS 24.302 第 8.1.1.2 条 中 WLAN 接入网络的 ANID，没有附加的 ANID 字符串。`network_name` 配置键覆盖默认值。除非 HSS 期望不同的 ANID，否则请保持该键未设置。

密钥派生和交付

每种方法派生一个 64 字节的主会话密钥 (MSK) 和一个 64 字节的扩展 MSK (EMSK)。TWAG 将 MSK 交付给接入点。TWAG 保留 EMSK 以用于快速重新认证。

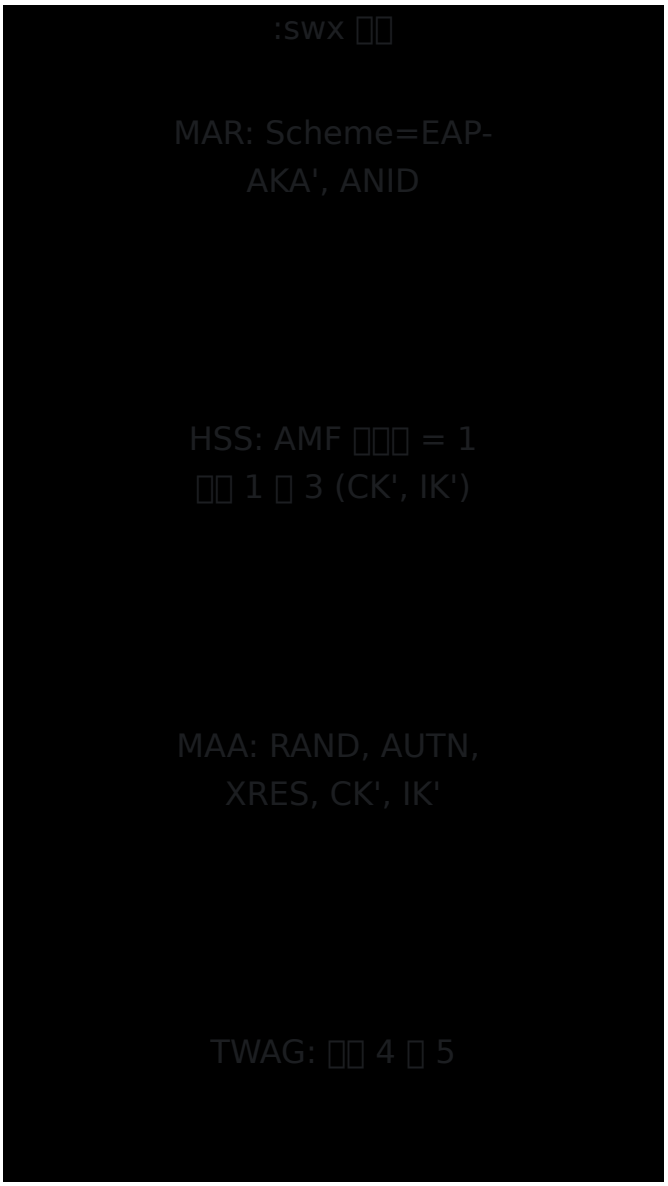
EAP-AKA' 密钥层次结构

根据 RFC 5448 和 3GPP TS 33.402 第 6.2 条和附录 A:

1. `Key = CK \| IK`。
2. `S = FC(0x20) \| network_name \| len(network_name) \| (SQN XOR AK) \| len(SQN XOR AK)`。
3. `CK' \| IK' = HMAC-SHA-256(Key, S)`。`CK'` 是前 16 字节。`IK'` 是最后 16 字节。
4. `MK = PRF'(IK' \| CK', "EAP-AKA'" \| Identity)`，208 字节。PRF' 使用 HMAC-SHA-256。
5. TWAG 将 `MK` 拆分为: `K_encr` (16)、`K_aut` (32)、`K_re` (32)、`MSK` (64)、`EMSK` (64)。

向量来源决定步骤 1 到 3 的运行位置:





在 `:swx` 模式下，HSS 执行步骤 1 到 3。TWAG 在 MAR 中请求 EAP-AKA' 向量。HSS 将 AMF 分离位设置为 1，并使用 ANID 派生 `CK'`/`IK'`。MAA 在 `Confidentiality-Key` 中携带 `CK'`，在 `Integrity-`

Key 中携带 IK' 。TWAG 使用这些密钥并执行步骤 4 和 5。对于 AKA' 向量，CK 和 IK 不会离开 HSS。请参见 3GPP TS 33.402 第 6.2 条，步骤 10。

在 :local 模式下，TWAG 是向量来源。TWAG 使用配置的测试密钥运行所有五个步骤。

如果 HSS 用方案 EAP-AKA 或 AMF 分离位 0 回应 EAP-AKA' 请求，TWAG 将用 EAP-Failure 拒绝用户。日志行以 HSS 向量无法用于 EAP-AKA' 开头，根据 TS 33.402 第 6.2 条第 15 步，用户在任何情况下都会拒绝此类挑战。早期拒绝将 HSS 作为原因，而不是来自手持设备的 Client-Error 。

EAP-AKA 密钥层次结构

根据 RFC 4187:

1. MK = SHA1(Identity \| IK \| CK) 。
2. TWAG 使用 FIPS 186-2 PRF 将 MK 扩展到 160 字节。
3. TWAG 拆分输出: K_encr (16)、K_aut (16)、MSK (64)、EMSK (64)。

EAP-SIM 密钥层次结构

根据 RFC 4186:

1. MK = SHA1(Identity \| Kc1 \| Kc2 \| [Kc3] \| NONCE_MT \| Version_List \| Selected_Version) 。
2. TWAG 使用 FIPS 186-2 PRF 将 MK 扩展到 160 字节。
3. TWAG 拆分输出: K_encr (16)、K_aut (16)、MSK (64)、EMSK (64)。

MSK 到 MPPE 到接入点

TWAG 将 64 字节的 MSK 通过 RADIUS Access-Accept 交付给接入点。TWAG 拆分 MSK 并将每一半编码为 Microsoft MPPE 密钥 (RFC 2548, RFC 3748):

- MS-MPPE-Recv-Key = MSK[0..31] (前32个字节) 。
- MS-MPPE-Send-Key = MSK[32..63] (后32个字节) 。

TWAG使用RADIUS共享密钥和请求身份验证器对每个密钥进行加密，遵循RFC 2548 。TWAG为每个密钥使用一个新鲜的高位设置的2字节盐。TWAG使用共享密钥对Access-Accept进行签名。有关TWAG如何解析密钥的详细信息，请参见RADIUS Shared Secrets 。

接入点解密MPPE密钥，恢复MSK，并将MSK用作与用户的802.11四路握手的成对主密钥。

Access-Accept需要有效的64字节MSK。如果MSK缺失或大小不正确，TWAG将发送Access-Reject。

重新同步

用户可以报告其序列号（SQN）与网络不同步。用户发送AKA-Synchronization-Failure（子类型`0x04`）和`AT_AUTS`。`AT_AUTS`为14字节：6字节的隐蔽SQN和8字节的MAC-S。

TWAG处理重新同步的步骤如下：

1. TWAG构建重新同步信息：存储的RAND与14字节的AUTS连接在一起。
2. TWAG运行重新同步。这恢复用户的SQN，使用Milenage f5*函数，验证MAC-S，并返回更新的序列号。
3. 成功时，TWAG生成一个带有更新SQN的新向量，构建一个新的挑战，并发送新的Access-Challenge。
4. 失败时，TWAG以EAP-Failure拒绝用户。

TWAG同时处理EAP-AKA和EAP-AKA'的重新同步。重新同步使用本地Milenage密钥。`AT_AUTS`值必须恰好为14字节，否则TWAG将引发错误，轮次失败。

快速重新认证（**ERP**）

TWAG支持EAP重新认证协议（ERP），遵循RFC 6696。ERP允许用户在没有完整EAP交换的情况下重新认证。节省来自于TWAG在第一次完整认证期间从EMSK派生的密钥链。

在完整认证后，TWAG派生并存储重新认证密钥：

- `rRK` = 重新认证根密钥，来自EMSK。
- `rIK` = 重新认证完整性密钥，来自rRK。

所有ERP密钥派生使用HMAC-SHA-256。

当用户发送EAP-Initiate/Re-auth消息（EAP类型41）时，TWAG执行以下操作：

1. 如果TWAG没有存储的`rRK`或`rIK`，TWAG将回退到完整认证。TWAG发送一个新的AKA-Identity请求。
2. 如果TWAG有密钥，TWAG解析消息并使用`rIK`验证身份验证标签。
3. 如果标签验证成功，TWAG从`rRK`和消息序列号派生重新认证MSK（`rMSK`），构建EAP-Finish/Re-auth响应（EAP类型42），并接受。`rMSK`成为MPPE密钥的新MSK。

4. 如果标签未验证，或解析失败，TWAG将回���到完整认证。

失败处理和结果代码

EAP结果

条件	TWAG操作
SWx MAR失败，或向量不完整	向量获取被拒绝。TWAG干净地拒绝用户，返回EAP-Failure，并记录拒绝而不是崩溃。
请求EAP-AKA'，但MAA方案为`EAP-AKA`	Access-Reject和EAP-Failure。日志行以`HSS vector unusable for EAP-AKA'`开头。HSS不支持EAP-AKA'向量。
请求EAP-AKA'，但AUTN中的AMF分离位为0	Access-Reject和EAP-Failure。相同的日志行。HSS发出了用户必须拒绝的向量（TS 33.402条款6.2第15步）。
`AT_MAC`验证失败	Access-Reject和EAP-Failure。
`AT_RES`与XRES不匹配	Access-Reject和EAP-Failure。
缺少`K_aut`或XRES状态	Access-Reject和EAP-Failure。
缺少`AT_RES`或`AT_MAC`属性	Access-Reject和EAP-Failure。
用户发送AKA-Authentication-Reject（子类型`0x02`）	Access-Reject。用户拒绝了网络。
用户发送AKA-Client-Error（子类型`0x0E`）	Access-Reject。
EAP-NAK没有支持的方法	Access-Reject。
Accept时MSK缺失或不是64字节	Access-Reject。
未处理的EAP消息	Access-Reject和EAP-Failure。

SWx Diameter结果代码

成功代码为基本`Result-Code`。下面的错误代码在`Experimental-Result` AVP中到达，而不是基`Result-Code`。请参见SWx错误信号说明和Troubleshooting。

代码	名称	意义
2001	DIAMETER_SUCCESS	MAA携带有效向量。
5001	DIAMETER_ERROR_USER_UNKNOWN	HSS没有该IMSI的非3GPP/SWx订阅。用户可能仍然在S6a/EPC上被识别。
5004	DIAMETER_ERROR_NOT_SUPPORTED_USER_DATA	请求的用户数据不被支持。
5420	DIAMETER_ERROR_UNKNOWN_EPS_SUBSCRIPTION	该用户没有EPS订阅。
5450	DIAMETER_ERROR_USER_NO_NON_3GPP_SUBSCRIPTION	用户没有非3GPP订阅。
其他	（Diameter错误）	向量获取失败。轮次被拒绝。

参考：RFC 6733第7.1节，3GPP TS 29.273，3GPP TS 29.229。

网络发起的事件（STa）

3GPP AAA服务器可以通过STa接口向TWAG发送事件（3GPP TS 29.273，应用ID `16777250`）。TWAG对每个事件的响应为`Result-Code` `2001`。

消息	触发	**TWAG**操作
ASR（Abort-Session-Request）	AAA服务器希望断开连接。	TWAG断开用户会话，然后回答ASA。
STR（Session-Termination-Request）	HSS发起的分离。	TWAG断开用户会话，然后回答STA。
RAR（Re-Auth-Request）	AAA服务器希望重新认证。	TWAG以成功��答RAA并记录请求。尚未从RAR触发实时会话的完整重新认证。

所有三种回答都携带来自Diameter身份的Origin-Host和Origin-Realm。请参见Configuration。

配置

在`:omnitwag`应用程序配置中设置身份验证参数。Diameter身份（`host`和`realm`）来自`:diameter_ex`配置，TWAG将其用作SWx和STa上的Origin-Host和Origin-Realm。

```
config :omnitwag,
  # 向量来源: :swx（生产）或: local（测试）
  auth_mode: :swx,

  # 供的EAP方法: :eap_aka_prime（默认），:eap_aka或:eap_sim
  eap_method: :eap_aka_prime,

  # EAP-AKA'网络名称（ANID）。TWAG在AT_KDF_INPUT和SWx MAR中发送它。默认值为"WLAN"（TS 24.302条款8.1.1.2）。在正常部署中保持未设置。
  network_name: "WLAN",

  # 在经过此许多毫秒后，重新收割仍停留在进行中的身份验证阶段的会话。默认值600000（10分钟）。
  identity_timeout_ms: 600_000,

  # TWAN标识符字段，在设置时添加到STa DER/AAR中
  twan_ssid: "ons",
  twan_bssid: "00:11:22:33:44:55",

  # 本地测试密钥（仅在auth_mode: :local时使用）
  test_keys: %{
    ki: "11111111111111111111111111111111",
    opc: "22222222222222222222222222222222",
    amf: "8000",
    sqn: 1200
  }
```

参数	类型	必 ???	默认	描述
`auth_mode`	原子	否	`:local`	向量来源。`:swx`从HSS获取[量。`:local`从`test_ke` 在生产中设置`:swx`。
`eap_method`	原子	否	`:eap_aka_prime`	提供的EAP方法。可以 是`:eap_aka_prime`、[或`:eap_sim`。用户可以通过 协商不同的支持方法。
`network_name`	字符串	否	`WLAN`	EAP-AKA'网络名称（ANID 在`AT_KDF_INPUT`中发送[SWx MAR中的`ANID` A 是TS 24.302条款8.1.1. WLAN接入值。
`identity_timeout_ms`	整数	否	`600000`	会话在TWAG重新收割之前可以[份验证阶段的时间（毫秒）。
`twan_ssid`	字符串	否	-	WLAN SSID。当设置时，T STa DER和AAR消息中添[符。
`twan_bssid`	字符串	否	-	接入点BSSID。与`twan_s` TWAN标识符。
`test_keys`	映射	否	内置默认值	本地Milenage密钥。仅在`au` 为`:local`时使用。请参见Te parameters。

测试密钥参数

TWAG仅在`:local`模式下读取`test_keys`。请勿在生产中使用`test_keys`。

参数	类型	必需	默认	描述
`ki`	字符串（十六进制）	是（本地模式）	`1111...`	用户密钥`Ki`，16字节作为32个十六进制字符。
`opc`	字符串（十六进制）	是（本地模式）	`2222...`	操作员变体密钥`OPc`，16字节作为32个十六进制字符。
`amf`	字符串（十六进制）	是（本地模式）	`8000`	身份验证管理字段，2字节作为4个十六进制字符。
`sqn`	整数	是（本地模式）	`1200`	起始序列号。

可观察性

TWAG通过EAP状态机和STa处理程序发出遥测计数器。Prometheus端点公开计数器。有关端点和查询示例，请参见Metrics Reference。

EAP指标

指标	意义
`eap_aka_identity_count`	EAP-AKA/AKA'身份交换（第一阶段）。
`eap_aka_challenge_count`	EAP-AKA/AKA'挑战交换（第二阶段）。
`eap_aka_sync_failure_count`	AKA同步失败（重新同步事件）。
`eap_aka_auth_success_count`	成功的EAP-AKA/AKA'认证。
`eap_aka_auth_reject_count`	被拒绝的EAP-AKA/AKA'认证。

TWAG还发出EAP-SIM（`start`、`auth_success`、`auth_reject`）、AKA `auth_reject`和`client_error`以及ERP（`initiate`、`success`、`fallback`、`mac_failure`）的遥测事件。请参见

Metrics Reference。

RADIUS回复指标

指标	意义
`radius_access_challenge_count`	Access-Challenge回复（EAP轮次进行中）。
`radius_access_accept_count`	Access-Accept回复（成功认证）。
`radius_access_reject_count`	Access-Reject回复（认证失败）。

STa事件指标

STa处理程序为每个网络发起的事件发出遥测计数器：`[:sta, :asr, :received, :count]`、`[:sta, :str, :received, :count]`和`[:sta, :rar, :received, :count]`，以及会话`disconnect`和`reauth`计数器。

日志上下文

TWAG为每个会话设置日志元数据`identity`和`imsi`。使用这些字段跟踪一个用户通过身份验证流程。

故障排除

症状	可能原因	
身份验证失败，日志显示`DIAMETER_ERROR_USER_UNKNOWN (5001)`	HSS没有该IMSI的非3GPP/SWx订阅。用户可能在S6a/EPC上被识别，但未启用非3GPP访问。	在HSS上[
身份验证失败，日志显示`DIAMETER_ERROR_USER_NO_NON_3GPP_SUBSCRIPTION (5450)`	用户配置文件没有非3GPP订阅。	在HSS上[
日志在失败的MAA上显示`Result-Code=nil`	SWx在`Experimental-Result` AVP中返回错误，而不是基本`Result-Code`。	阅读同一日[`Experi Code`。T
身份验证从未完成，会话被收割	用户在进行中的身份验证阶段停滞，超过身份验证超时。	检查交换的U `identi`
用户发送AKA-Authentication-Reject	UE未能认证网络（AUTN检查）。	检查网络名[否与`:loc` HSS向量[
重复的AKA-Synchronization-Failure	用户SQN与网络SQN漂移。	TWAG[存在，请检[Resync
预期有效EAP-Success的Access-Reject	Accept时MSK缺失或不是64字节。	确认向量来[MAA或[

症状	可能原因	
EAP-AKA'认证在挑战之前失败，日志显示`HSS vector unusable for EAP-AKA'`	HSS用EAP-AKA向量回答MAR：方案为`EAP-AKA`，或AMF分离位为0。HSS不支持EAP-AKA'向量，或忽略了`ANID`。	升级或配置H 29.273 AKA'向[`eap_me` 为`:eap_`
手机用AKA-Client-Error回答EAP-AKA'挑战	手机无法验证挑战。通常原因是网络名称不匹配：HSS使用与`AT_KDF_INPUT`值不同的ANID派生`CK'`/`IK'`。	确认HSS[持`netwo` 双方都使用

对于在EAP交换之前或周围发生的RADIUS层拒绝（共享密钥错误、未知接入点），请参见RADIUS Shared Secrets和Sessions状态参考。

呼叫流程

述

本页面展示了OmniTWAG的端到端消息流程。每个流程是用户设备（UE）、WiFi接入点（AP）、TWAG和核心网络对等方（HSS（SWx）、PGW（S2a）和OCS（Gy））之间的时序图。

请阅读每个功能页面以获取每个步骤的详细信息：认证、S2a GTP-C、用户平面、IP地址管理、IP池、在线计费、RADIUS凭证和会话。

目录

- 参与者和接口
- 成功附加 - 隧道，计费
- 附加 - NSWO与IP池
- 附加 - NSWO与DHCP分配
- 附加 - 本地突破（AP数据平面）
- 认证失败 - 订阅者未为SWx配置
- 计费和使用报告
- 配额耗���和重新授予
- 拆除 - 订阅者断开连接
- 拆除 - 操作员从API中踢出
- 拆除 - 存活超时
- 计费模式变体

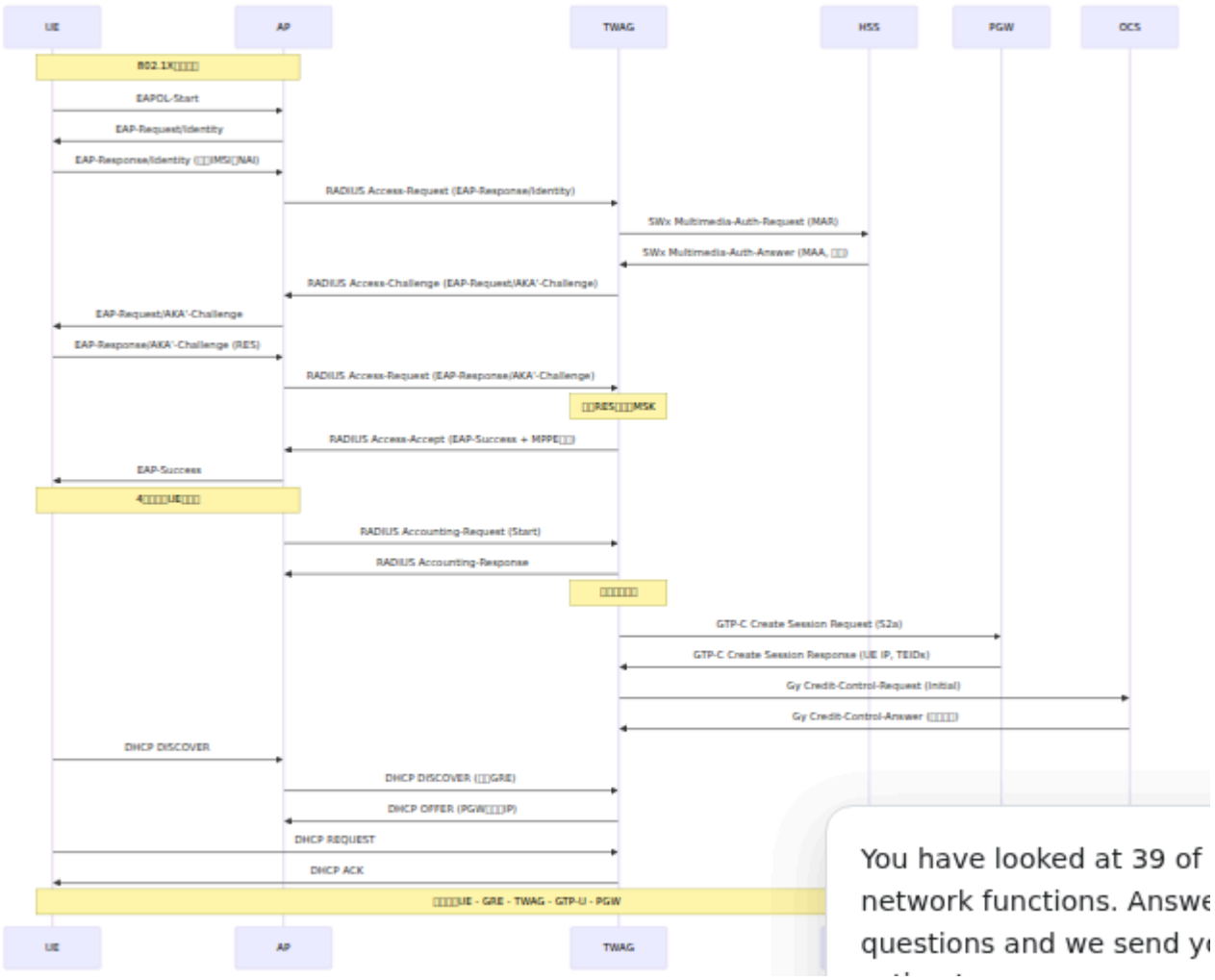
参与者和接口

参与者	角色	与**TWAG**的接口
UE	带有SIM的订阅设备	802.11 / EAPOL（通过AP）
AP	WiFi接入点和RADIUS客户端	RADIUS（UDP 1812认证，1813计费）
TWAG	可信WiFi接入网关和3GPP AAA服务器	-
HSS	家庭订阅服务器	Diameter SWx（认证向量）
PGW	数据包网关	GTP-C/GTP-U通过S2a（仅限隧道）
OCS	在线计费系统	Diameter Gy（仅限计费）

会话的突破类型和计费模式来自AP凭证。请参见RADIUS凭证。

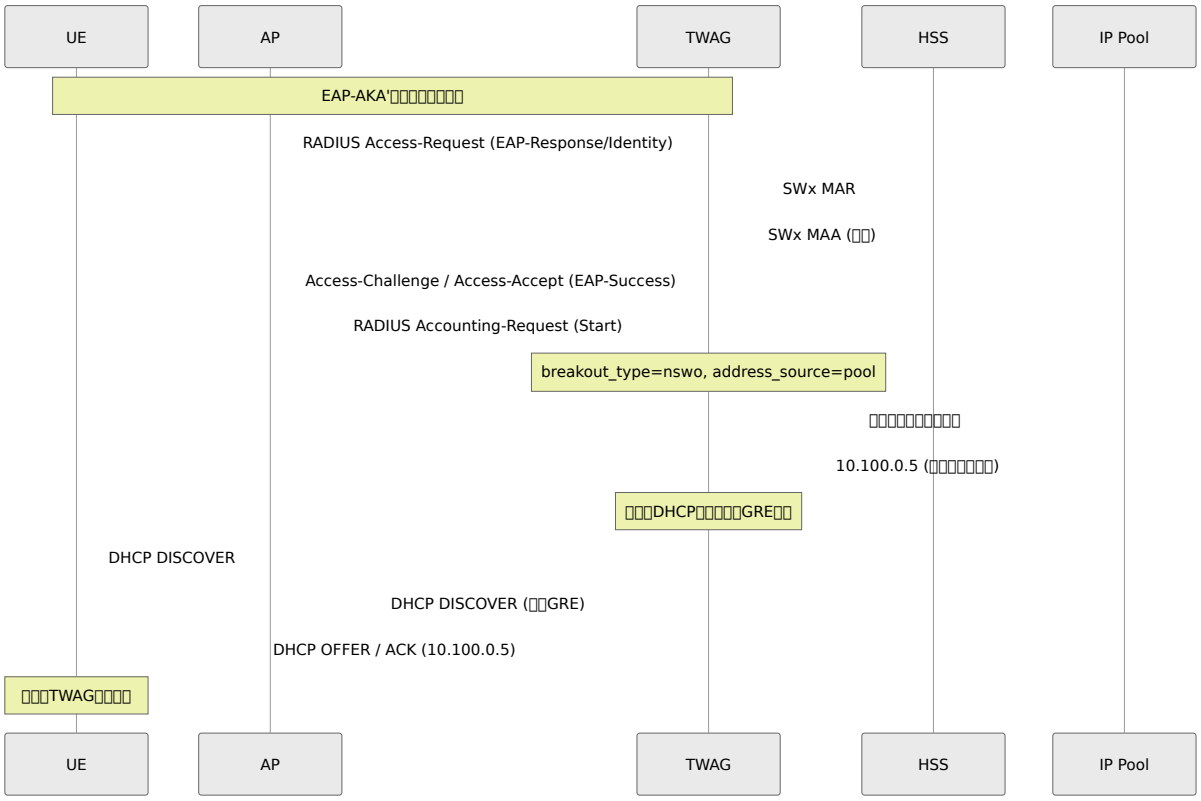
成功附加 - 隧道，计费

这是`breakout_type: tunneled`和`charging_mode: charged`的完整成功路径。TWAG通过EAP-AKA'对订阅者进行身份验证，建立与PGW的GTP会话，通过DHCP传递PGW分配的IP，并打开Gy计费会话。



附加 - NSWO与IP池

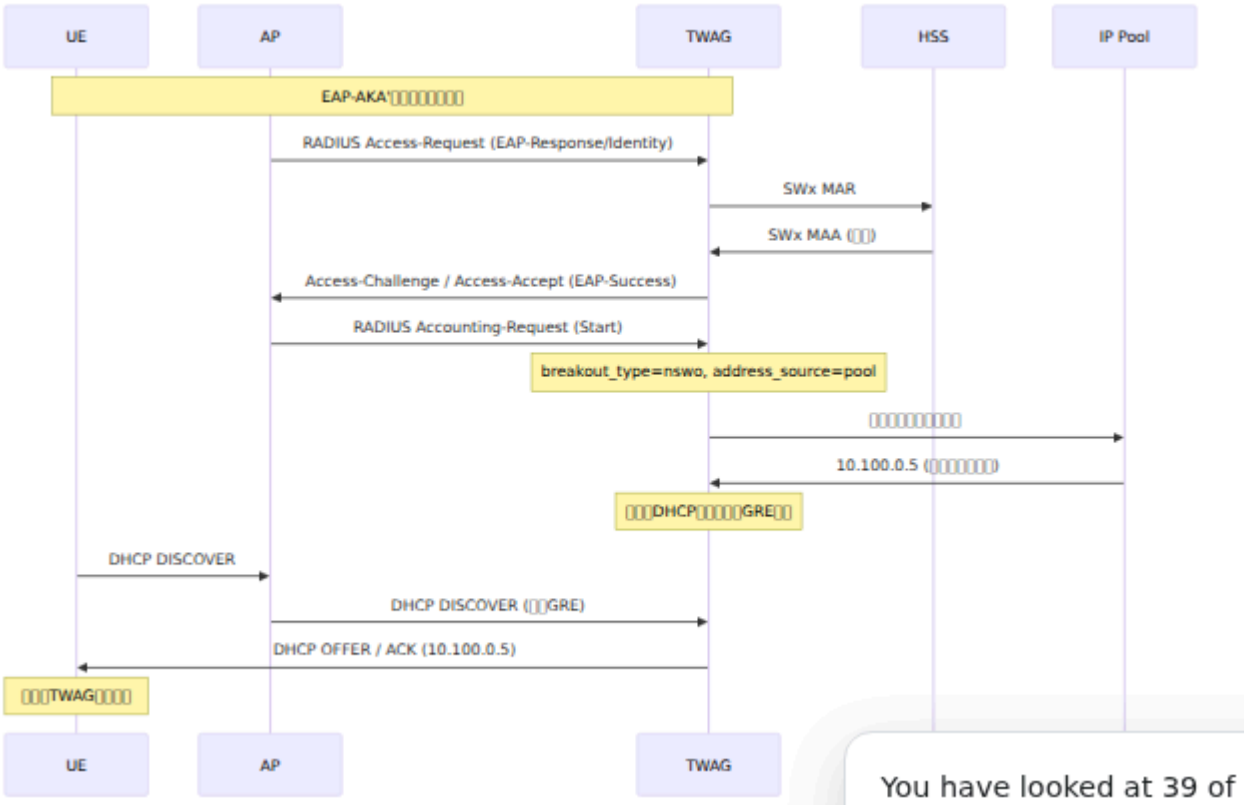
对`breakout_type: nswo`和`address_source: pool`，TWAG不联系PGW。TWAG从关联的IP池中预分配一个固定地址，并通过DHCP传递。订阅者流量在本地突破。

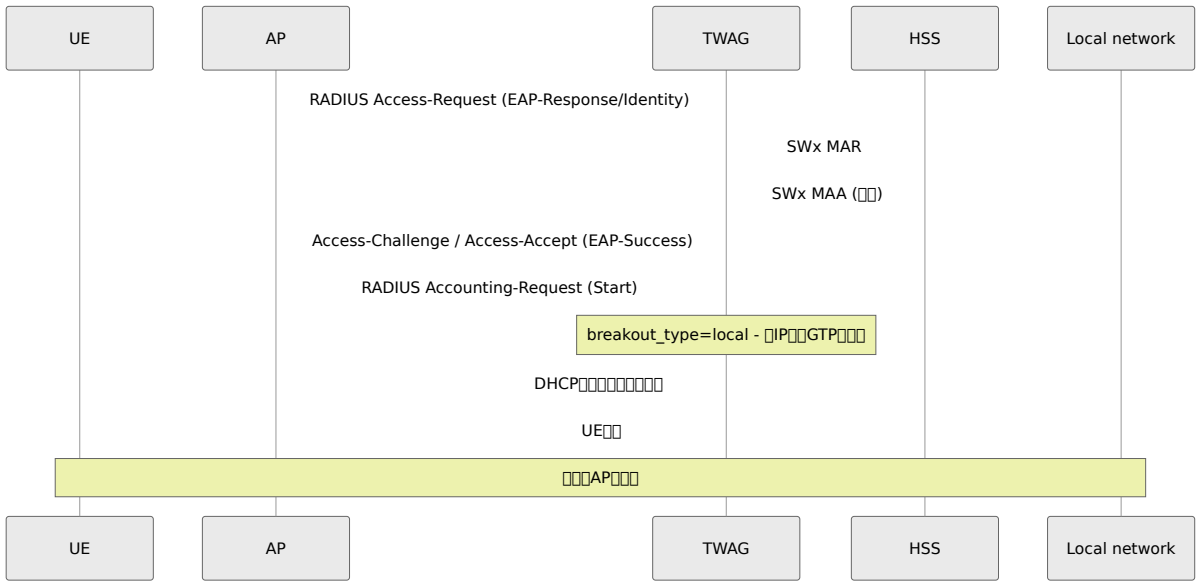


UE
AP
TWAG
HSS
IP Pool
EAP-AKA'认证（如上所述）
RADIUS Access-Request (EAP-Response/Identity)
SWx MAR
SWx MAA (向量)
Access-Challenge / Access-Accept (EAP-Success)
RADIUS Accounting-Request (Start)
breakout_type=nswo, address_source=pool
为此会话分配一个地址
10.100.0.5 (第一个空闲主机)
预加载DHCP租约，注册GRE下行
DHCP DISCOVER
DHCP DISCOVER (通过GRE)
DHCP OFFER / ACK (10.100.0.5)
流量在TWAG本地突破
UE
AP
TWAG
HSS
IP Pool

附加 - NSWO与DHCP分配

对于`breakout_type: nswo`和`address_source: dhcp`，TWAG将客户端MAC绑定到池。当UE发送DISCOVER时，DHCP服务器从池范围中分配下一个空闲地址。请参见IP池。





UE
AP
TWAG
HSS
IP Pool
EAP-AKA'
RADIUS Access-Request (EAP-Response/Identity)
SWx MAR
SWx MAA
Access-Challenge / Access-Accept (EAP-Success)
RADIUS Accounting-Request (Start)
breakout_type=nswo, address_source=pool
10.100.0.5
DHCP GRE
DHCP DISCOVER
DHCP DISCOVER GRE
DHCP OFFER / ACK (10.100.0.5)
TWAG
UE
AP
TWAG
You have looked at 39 of

附加 - 本地突破（AP数据平面）

对于`breakout_type: local`，AP拥有数据平面。TWAG对订阅者进行身份验证且不分配IP。UE从本地网络获取其地址。TWAG不建立GTP会话，也不建立DHCP租约。

UE
AP
TWAG
HSS
Local network
RADIUS Access-Request (EAP-Response/Identity)
SWx MAR
SWx MAA (向量)
Access-Challenge / Access-Accept (EAP-Success)
RADIUS Accounting-Request (Start)
breakout_type=local - 无IP, 无GTP, 无池
DHCP (由本地网络提供)
UE地址
数据在AP处突破
UE
AP
TWAG
HSS
Local network

认证失败 - 订阅者未为SWx配置

HSS在分组的`Experimental-Result` AVP中发出SWx错误，而不是基本的`Result-Code`。在S6a上已知的订阅者如果HSS没有IMSI的非3GPP订阅，仍然可能SWx失败。TWAG拒绝会话。请参见认证。

计费和使用报告

AP发送RADIUS计费。TWAG将流量合并到会话中（请参见会话），并在`charged`模式下，在配额阈值或有效性计时器到期时向OCS报告使用情况，而不是在每次中间更新时。请参见在线计费。

配额耗尽和重新授予

当OCS不再授予配额或设置最终单位操作时，TWAG阻止UE数据路径。TWAG保留会话、租约和计费，以便充值可以解除对UE的阻止。请参见在线计费。

UE
TWAG
OCS
Gy CCR-Update (已使用配额)
Gy CCA (Final-Unit-Action, 不再有配额)
阻止UE GRE数据路径（会话保留）
数据被阻止
订阅者充值
Gy CCR-Update
Gy CCA (新配额)
解除对UE数据路径的阻止
数据恢复
UE
TWAG
OCS

拆除 - 订阅者断开连接

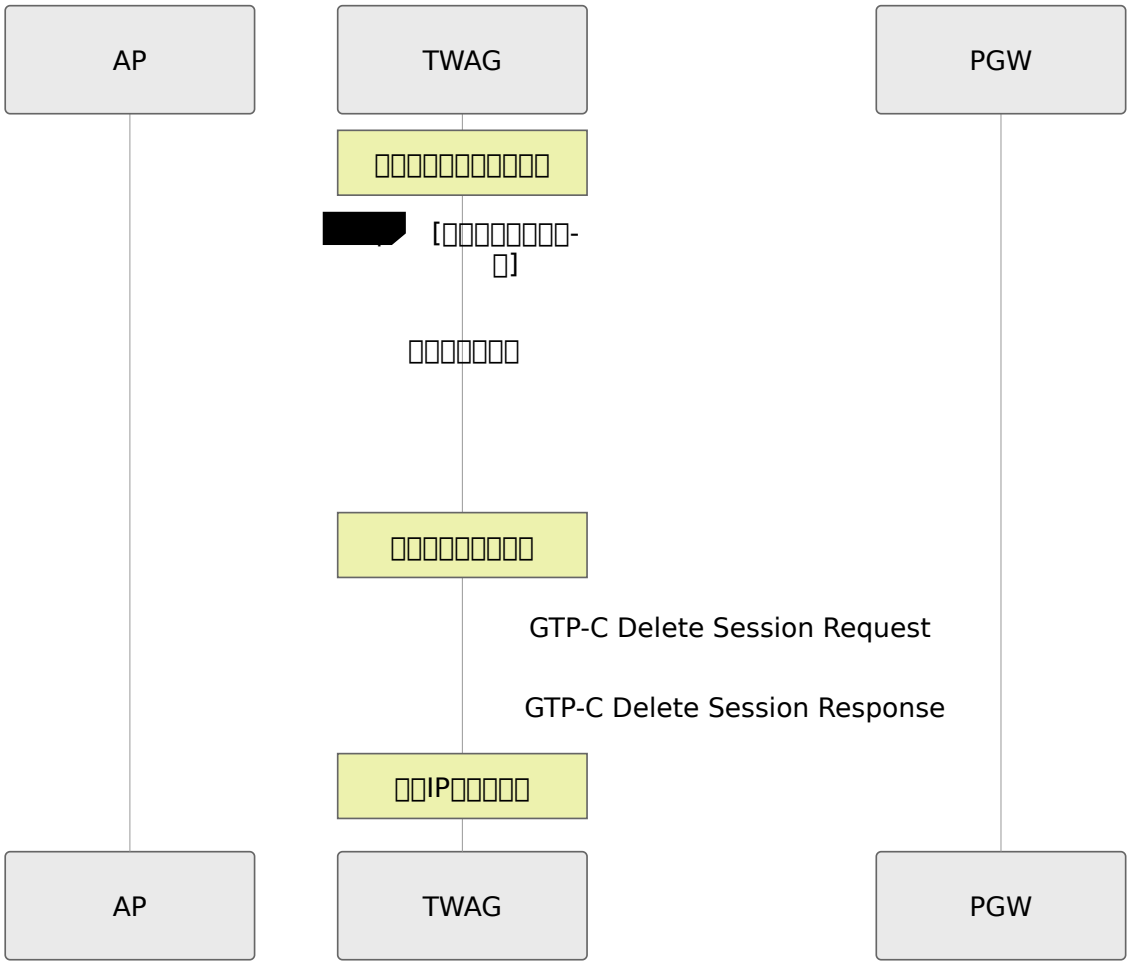
当UE断开连接时，AP发送Accounting-Stop。TWAG拆除会话：它删除GTP会话（隧道），发送Gy CCR-Terminate（计费），释放地址并清除会话状态。

拆除 - 操作员从API中踢出

操作员可以通过API删除会话。TWAG向AP发送RADIUS CoA-Disconnect（Disconnect-Message），AP将UE踢出，然后拆除会话。请参见会话。

拆除 - 存活超时

TWAG通过存活计时器监控已建立的会话。计费中间更新重置计时器。如果AP在配置的间隔数内没有发送更新，TWAG将声明会话已死并拆除它。未完成身份验证的会话将由单独的身份超时清除。请参见会话���



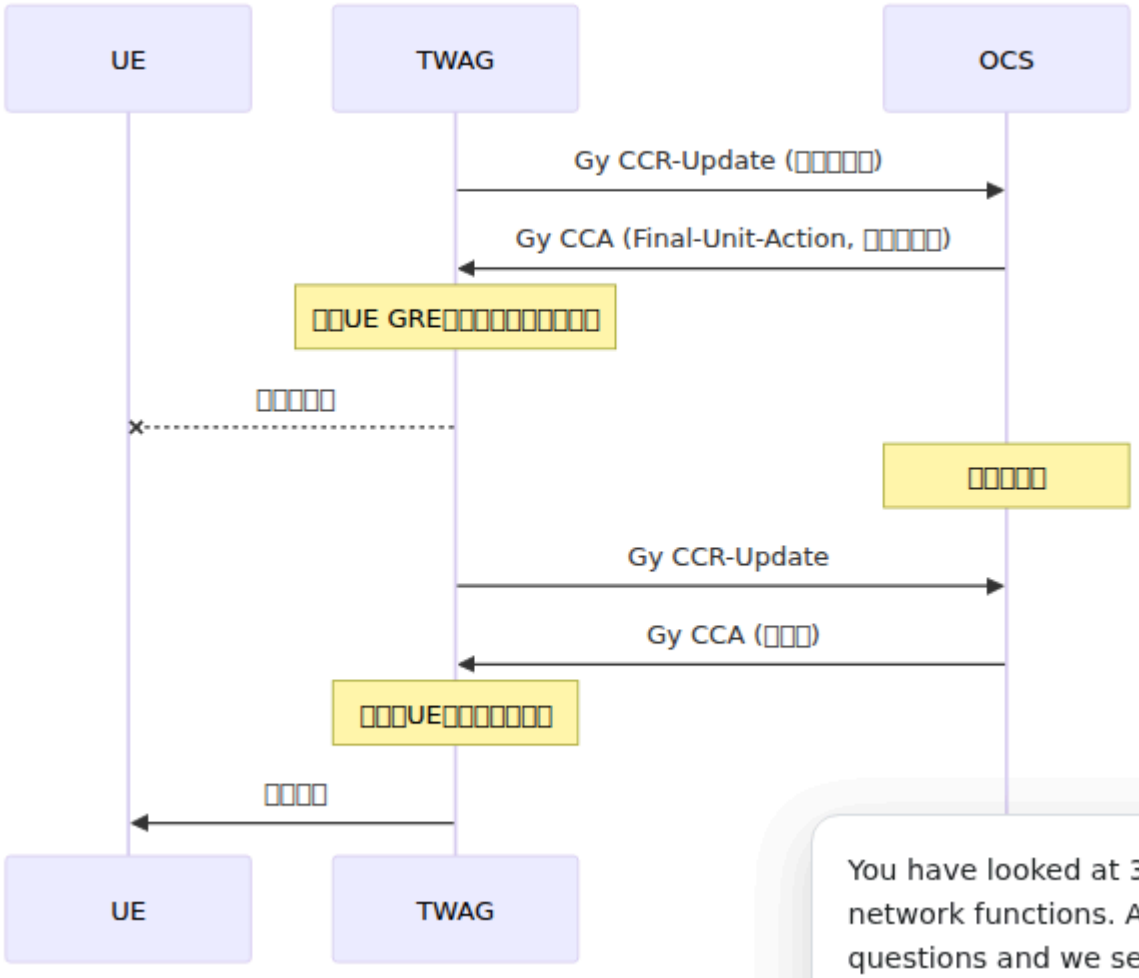
AP
TWAG
PGW
在建立时启动存活计时器
[redacted]
[每个没有计费的间隔]
增加未命中间隔
未命中间隔达到阈值
GTP-C Delete Session Request
GTP-C Delete Session Response
释放IP，移除会话
AP
TWAG
PGW

计费模式变体

AP凭证上的`charging_mode`改变接受后的步骤。请参见RADIUS凭证。

charging_mode	数据会话	**Gy**
`charged`	已建立（按`breakout_type`）	CCR-Initial / Update / Terminate
`uncharged`	已建立（按`breakout_type`）	无Gy会话
`auth_only`	未建立	无Gy会话

对于`uncharged`，流程是上述隧道或NSWO附加，但TWAG不发送Gy消息。对于`auth_only`，流程在Access-Accept和Accounting-Start后停止：TWAG不建立数据会话。


UE
TWAG
OCS
Gy CCR-Update (□□□□□□)
Gy CCA (Final-Unit-Action, □□□□□□)
□□UE GRE□□□□□□□□□□□□
□□□□□□
□□□□□□
Gy CCR-Update
Gy CCA (□□□)
□□□UE□□□□□□□□
□□□□□
UE
TWAG
You have looked at 39 of
network functions. Answe
questions and we send y

□□

连接模式、WLCP 和移动性

概述

本指南解释了 TWAG 连接模式、无线局域网控制协议 (WLCP) 和切换行为。它描述了 OmniTWAG 软件今天的功能。它还标记了每个部分或尚未集成的区域，以便运营商不会围绕代码未运行的行为进行规划。

TWAG 遵循 3GPP TS 23.402 第 16 节中的受信 WLAN 模型。有关这些模式所构建的数据平面、认证和 S2a 信令，请参见：

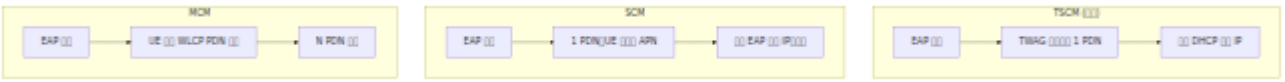
- 架构：数据平面、会话生命周期和接口
- 认证：EAP-AKA / EAP-AKA' 和密钥材料
- S2a GTP-C：创建会话、删除会话和承载信令
- 会话：已认证会话及其状态生命周期
- 用户平面：GRE、GTP-U 和 DHCP 交付
- RADIUS 共享密钥：每个 AP 的分解和在认证后应用的计费

目录

- 概述和模式比较
- 模式协商
- 透明单连接模式 (TSCM)
- 单连接模式 (SCM)
- 多连接模式 (MCM) 和 WLCP
- 切换和移动性
- 配置
- 可观察性
- 实施状态

概述和模式比较

TWAG 支持三种连接模式，供通过受信 WLAN 访问 EPC 的 UE 使用。该模式控制 UE 如何学习其 IP 地址、UE 是否可以信令 APN，以及 UE 是否可以保持多个 PDN 连接。



属性	TSCM	SCM	MCM
模式代码	`0`	`1`	`2`
全名	透明单连接模式	单连接模式	多连接模式
每个 UE 的 PDN 连接	一个	一个	多个
IP 地址交付	DHCP	EAP	EAP / WLCP
UE 信令 APN 选择	否	是	是（每个连接）
IP 保持的切换	否	是	是
专用承载	否	是	是
UE 控制协议	无	无	WLCP (TS 24.244)
UE 了解 S2a	否	是	是

TSCM 是默认模式。TSCM 匹配预 Release-12 行为，其中 UE 不信令网络。TWAG 选择 APN，创建一个 PDN 连接，并通过 DHCP 交付 PGW 分配的 IP 地址。

模式代码 `0`、`1` 和 `2` 遵循 3GPP TS 29.273 中的 TWAN-Connection-Mode AVP。

模式协商

连接模式在认证期间通过携带 EAP 的 STa 交换进行协商。有关 EAP 流本身，请参见 认证。设计如下：

1. TWAG 向 AAA 服务器通告其支持的模式。通告的集合来自 `connection_modes` 配置键。
2. AAA 服务器在 STa DEA 响应中返回所选模式，在 TWAN-Connection-Mode AVP 中。
3. 如果 AAA 服务器未指示模式，则 TWAG 使用 TSCM 作为默认模式。

如果 AAA 服务器选择了 TWAG 不支持的模式，则 TWAG 会回退。回退顺序优先考虑 TSCM，然后是 SCM，再然后是 MCM。TWAG 从该顺序中选择第一个支持的模式。

TWAG 从 AAA 响应中的以下任一键中检索所选模式，顺序为：`twan_connection_mode`、`TWAN-Connection-Mode`、`connection_mode` 或 `Connection-Mode`。该值可以是数字代码（`0`/`1`/`2`）或原子（`:tscm`/`:scm`/`:mcm`）。未识别的值默认为 TSCM。

在 STa 路径上，STa DER 在 `TWAN-Connection-Mode` AVP 中通告 TWAG 首选的支持模式。STa DEA 解析读取协商的模式，并在 DEA 不携带时默认为 TSCM。

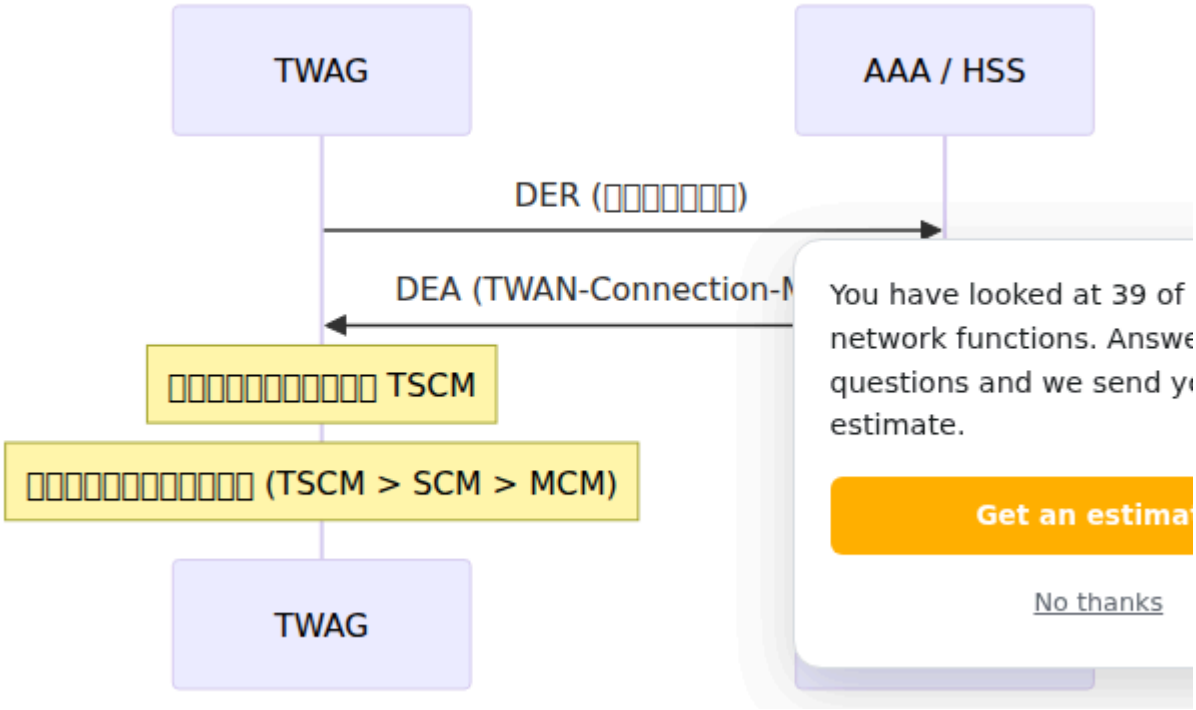
> 状态：协商的模式未在实时会话路径上强制执行。 STa 客户端（`StaClient`）确实在 DER 中通告支持的模式，并从 DEA 中读取协商的模式。连接模式模块存在并经过单元测试。然而，STa 直通认证路径不是实时 RADIUS 认证路径，实时客户端从未应用协商的模式：`store_connection_mode` 入口点在附加路径上没有调用者，附加和数据路径始终表现为 TSCM（通过 DHCP 获取 IP，无 WLCP，无 UE 信令切换或 APN）。今天每个实时会话都遵循 TSCM 路径。请参见 实施状态。

透明单连接模式（TSCM）

TSCM 是默认模式，也是今天实时会话路径运行的行为。UE 不信令网络。TSCM 在 架构 会话生命周期中进行了描述，并在此处进行了总结。

在 TSCM 中：

- UE 保持一个 PDN 连接。



- TWAG 选择 APN。UE 不能请求 APN。
- UE 通过 DHCP 接收其 IP 地址，由 TWAG 通过 GRE 隧道提供。请参见 用户平面。
- 没有 WLCP 信令。
- IP 保持的切换、专用承载和 UE 信令 APN 选择不可用。

TWAG 在认证后自动创建 S2a 会话。它向 PGW 发送创建会话请求，Handover Indication 标志清除。然后，它在其 DHCP 服务器中为 UE MAC 地址预分配 PGW 分配的 IP 地址。有关创建会话的详细信息，请参见 S2a GTP-C，有关结果会话及其状态生命周期，请参见 会话。启用时，会话的计费通过 Gy 进行。请参见 在线计费 (Gy)。

单连接模式（SCM）

SCM 每个 UE 保持一个 PDN 连接，但 UE 了解 S2a。SCM 在 TSCM 的基础上增加了这些功能：

- UE 通过 EAP 接收其 IP 地址，而不是 DHCP。
- UE 可以选择非默认 APN。
- UE 在 3GPP 到 WLAN 和 WLAN 到 3GPP 的切换中保持其 IP 地址。
- PGW 可创建专用承载。

连接模式模块报告这些 SCM 功能。它将 SCM 分类为通过 EAP 交付 IP、支持切换、支持 APN 选择和支持专用承载的模式。

> 状态：**SCM** 还不是一个独立的运行时路径。 SCM 的能力模型存在，但会话建立代码尚未根据协商的模式进行分支。AAA 服务器将 UE 放置在 SCM 中时，仍通过 TSCM 路径提供服务（自动创建 PDN，通过 DHCP 获取 IP）。在此版本中，请勿依赖 EAP 交付的 IP 或 UE 信令 APN 选择。

多连接模式（MCM）和 WLCP

MCM 允许 UE 同时保持多个 PDN 连接。UE 使用无线局域网控制协议（WLCP）管理每个连接，该协议在 3GPP TS 24.244 中定义。认证后，TWAG 不会自动创建 PDN 连接。相反，UE 发送 WLCP PDN 连接请求。TWAG 将每个 PDN 连接映射到通往 PGW 的单独 S2a GTP-C 会话。

WLCP 传输

TWAG 在 UDP 端口 `2156` 上运行 WLCP 服务器，这是标准 WLCP 端口。服务器将每个数据包路由到每个 UE 的 WLCP 会话处理程序，基于 UE 身份进行键控。服务器在认证后将 UE 源 IP 和端口映射到身份。

在生产中，WLCP 信令根据 TS 23.402 第 16.1.4A.3.1 节通过 DTLS 运行。当前服务器仅处理普通 UDP 层。DTLS 尚未应用。

WLCP 消息格式

每个 WLCP 消息都有一个固定的头部，后面跟着信息元素（IE）的列表。

```
消息类型（1 字节）| 长度（2 字节）| 事务 ID（1 字节）| IEs...
```

每个 IE 的格式如下：

```
IE 类型（1 字节）| IE 长度（2 字节）| IE 值（可变）
```

`长度`字段计算整个消息，包括 4 字节头部。

WLCP 消息类型

代码	消息	方向	目的
`0x01`	PDN 连接请求	UE 到 TWAG	请求新的 PDN 连接
`0x02`	PDN 连接接受	TWAG 到 UE	连接已创建，带有 IP / DNS / QoS
`0x03`	PDN 连接拒绝	TWAG 到 UE	连接被拒绝，带有原因
`0x04`	PDN 断开请求	UE 到 TWAG	释放 PDN 连接
`0x05`	PDN 断开接受	TWAG 到 UE	连接已释放
`0x06`	下行数据通知	TWAG 到 UE	通知 UE 有待处理的下行数据

WLCP 信息元素

代码	**IE**	备注
`0x01`	APN	接入点名称字符串
`0x02`	PDN 类型	`1` IPv4, `2` IPv6, `3` IPv4v6
`0x03`	PCO	协议配置选项
`0x04`	PDN 地址	分配的 UE IP 地址 (IPv4、IPv6 或 IPv4v6)
`0x05`	承载 QoS	QCI 和 MBR / GBR 值
`0x06`	TFT	流量流模板
`0x07`	原因	结果原因值
`0x08`	承载 ID	EPS 承载标识 (EBI)
`0x09`	DNS 地址	一个或多个 DNS 服务器地址

WLCP 原因值

代码	原因	意义
`0x10`	request_accepted	请求被接受
`0x20`	apn_not_allowed	此 UE 不允许 APN
`0x21`	no_resources	没有可用资源
`0x22`	unknown_apn	APN 未知
`0x23`	user_auth_failed	用户认证失败
`0x24`	network_failure	网络故障（例如 S2a 故障）
`0x25`	pdn_type_not_supported	请求的 PDN 类型不受支持
`0x30`	request_rejected	请求被拒绝

PDN 连接设置

当 PDN 连接请求到达时，WLCP 会话处理程序执行以下步骤：

1. 它从消息中读取 APN、PDN 类型和 PCO。如果 APN 缺失，则使用 `internet`。如果 PDN 类型缺失，则使用 IPv4。
2. 它检查 APN 是否被允许。请参见 配置 中的 `allowed_apns`。
3. 如果 UE 没有已知的 IMSI，它将以 `user_auth_failed` 原因拒绝请求。
4. 它为该 APN 创建一个通往 PGW 的 S2a GTP-C 会话，RAT 类型为 `3`（WLAN），并且 Handover Indication ��志清除。
5. 成功后，它注册 GTP-U 隧道并存储 PDN 连接，以 EBI 为键。然后，它发送 PDN 连接接受，带有 UE IP 地址，以及（如有）DNS 服务器、承载 QoS 和 EBI。
6. 在 S2a 故障时，它发送 PDN 连接拒绝，原因是 `network_failure`。

UE
WLCP 服务器 / 会话
PGW
EAP-AKA 认证 (MCM)
PDN 连接请求 (APN=internet)
创建会话请求
创建会话响应 (IP, DNS)
PDN 连接接受 (IP, DNS, EBI)
PDN 连接请求 (APN=ims)
创建会话请求
创建会话响应 (IP, DNS)
PDN 连接接受 (IP, DNS, EBI)
PDN 断开请求 (EBI=5)
删除会话请求
删除会话响应
PDN 断开接受
UE
WLCP 服务器 / 会话
PGW

PDN 连接拆除

当 PDN 断开请求到达时，处理程序通过 EBI 查找 PDN 连接。如果 EBI 未知，它会回复 PDN 连接拒绝和原因 `request_rejected`。如果 EBI 已知，它会删除 S2a GTP-C 会话，注销 GTP-U 隧道，移除连接，并回复 PDN 断开接受。当 UE 没有剩余的 PDN 连接时，WLCP 会话处理程序停止。

如果 PGW 删除承载，处理程序会���除匹配的 PDN 连接并释放本地 GTP-U 隧道。在这种情况下，它不会发送删除会话，因为 PGW 已经释放了承载。

WLCP 密钥派生

WLCP 密钥 `K_WLCP` 保护 WLCP 信令。它是从 EAP 认证生成的主会话密钥 (MSK) 派生的。派生遵循 3GPP TS 33.402 第 6.2 节和附录 A。

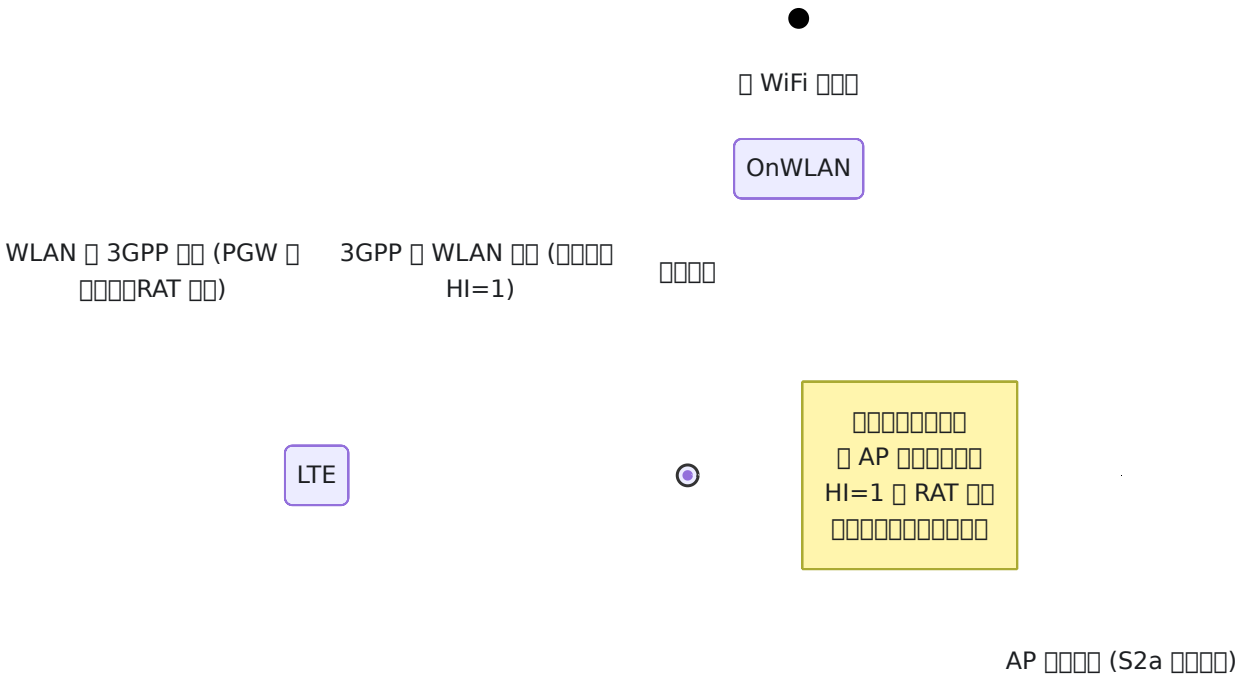
```
K_WLCP = HMAC-SHA-256(MSK, FC | P0 | L0)

FC = 0x21
P0 = TWAG IP 地址 (IPv4 为 4 字节, IPv6 为 16 字节)
L0 = P0 的长度 (2 字节, 大端)
```

输出为 32 字节（256 位）。TWAG 通过 `P0` 参数将密钥绑定到其自身的 IP 地址。当 TWAG IP 尚未知道时，简化变体从 MSK 和固定标签派生密钥。

> 状态：**MCM** 和 **WLCP** 是部分实现，尚未集成。 WLCP 编解码器、WLCP UDP 服务器、每个 UE 的会话处理程序和密钥派生都存在并经过单元测试。它们未由应用程序启动，也未由认证流驱动。以下内容尚未实现：
>
> - WLCP 的 DTLS 保护。仅处理普通 UDP。
> - MCM 内的专用承载管理。编解码器携带 QoS 和 TFT IEs，但处理程序未对 PGW 发起的专用承载创建采取行动。
> - MCM 内的切换。
> - 下行数据通知消息在编解码器中定义，但未由会话处理程序生成。
>
> 将 MCM 视为预览。请勿为生产 UE 启用它。

切换和移动性

TWAG 根据 TS 23.402 建模三种移动场景。今天，只有 AP 之间的移动性在实时会话路径上处于活动状态。



在 WiFi 上附加
OnWLAN
WLAN 到 3GPP 切换 (PGW 删除承载, RAT 更改)
3GPP 到 WLAN 切换 (创建会话 HI=1)
会话拆除
LTE
今天的实时路径:
仅 AP 之间的漫游。
HI=1 和 RAT 更改
分支在附加时未被执行。
AP 之间漫游 (S2a 隧道保持)

3GPP 到 WLAN 切换

当 UE 从 LTE 移动到 WiFi 时，TWAG 必须保持现有的 IP 会话。为此，TWAG 发送创建会话请求，Handover Indication (HI) 标志设置。PGW 然后保留现有的 IP 地址和承载上下文，而不是分配新地址。有关创建会话和指示 IE 的详细信息，请参见 S2a GTP-C。

S2a 创建会话请求支持 Handover Indication 标志。切换处理程序可以将切换事件路由到 UE 客户端进程，并设置切换标志。

> 状态：主会话路径未设置切换标志。自动会话建立始终发送创建会话请求，Handover Indication 标志清除。因此，3GPP 到 WLAN 切换中的 IP 保持未在此版本的正常附加路径中执行。

WLAN 到 3GPP 切换

当 UE 从 WiFi 移动到 LTE 时，PGW 发送删除承载请求。原因值指示 RAT 更改。TWAG 将这些原因值视为 RAT 更改切换：

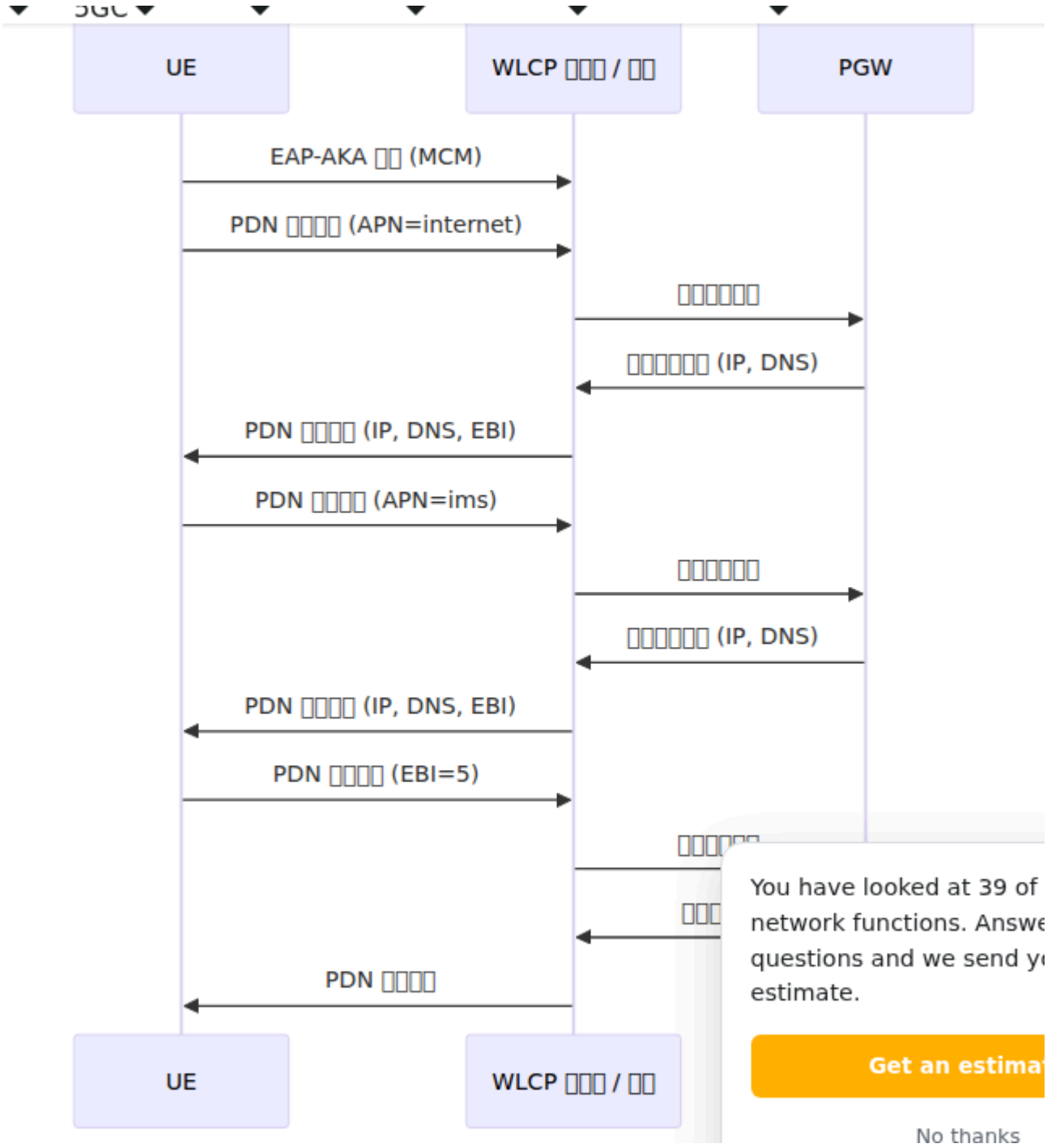
原因代码	名称
`4`	RAT 更改（3GPP 到非 3GPP）
`5`	ISR 停用
`10`	访问从非 3GPP 更改为 3GPP

在 RAT 更改删除承载时，TWAG 会在本地清理 WiFi 会话。它不会发送 GTP-C 删除会话，因为 PGW 已经释放了承载。UE 在 LTE 侧保持其 IP 地址。

> 状态：删除承载处理程序今天未根据原因进行分支。 UE 客户端进程通过运行本地清理然后停止，处理 PGW 删除承载，无论原因如何。RAT 更改分类作为辅助存在，但客户端不使用原因选择不同的拆除路径。

AP 之间的移动性

当 UE 在同一 TWAN 内的 WiFi AP 之间漫游时，S2a GTP 隧道保持开启。UE 保持相同的身份，但到达时具有不同的 NAS-IP-Address 和 Called-Station-Id。TWAG 更新 UE 客户端进程中的 AP 跟踪数据，并且不会拆除会话。这使得 UE IP 地址在 AP 更改之间保持稳定。



AP 之间的移动性是实时会话代码运行的唯一移动路径。它更新 UE 客户端进程字段以获取 NAS-IP 和 Called-Station-Id，并记录新 AP 在 AP 跟踪状态中。请参见 每 AP 计费 以获取 AP 跟踪数据。

配置

连接模式

使用 `connection_modes` 键设置 TWAG 通告的模式。该值是模式原子的列表。默认值为 `[:tscm]`。

```
config :omnitwag,
  # 此 TWAG 通告给 AAA 服务器的连接模式。
  connection_modes: [:tscm, :scm]
```

参数	类型	必需	默认	描述
`connection_modes`	列表	否	`[:tscm]`	TWAG 通告的模式。使用 `:tscm`、`:scm` 和 `:mcm`。顺序无关紧要。通告的代码在使用前会被排序。

> 注意。通告的集合由连接模式模块读取，STa 客户端在 STa DER 中发送首选模式。实时 RADIUS 会话路径尚未对协商的模式采取行动。请参见 实施状态。在 SCM 和 MCM 集成之前，请将默认值 `[:tscm]` 保持为生产环境。

WLCP 服务器（MCM）

WLCP 服务器设置位于 `:wlcp` 键下。WLCP 服务器仅与 MCM 相关。

```
config :omnitwag,
  wlcp: %{
    enabled: true,
    port: 2156,
    allowed_apns: :all
  }
```

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	启用 WLCP 服务器。除非您测试 MCM，否则请保持为 `false`。
`port`	整数	否	`2156`	WLCP 的 UDP 端口。`2156` 是标准 WLCP 端口。
`allowed_apns`	原子或列表	否	`:all`	WLCP PDN 请求的 APN 允许列表。使用 `:all` 以允许所有 APN，或使用 APN 字符串列表，例如 `["internet", "ims"]`。请求未在列表中的 APN 将被拒绝，原因是 `apn_not_allowed`。

> 注意。 WLCP 服务器在此版本中未由应用程序监视器启动。`enabled` 键为将来使用而定义。

相关配置

MCM PDN 设置路径重用其 S2a 会话的 GTP 设置。有关 `gtp` 块的配置摘要，请参见 架构，包括 `pgw_ip`、`local_ip`、`mcc` 和 `mnc`。

可观察性

WLCP 服务器和会话处理程序会发出遥测事件。这些事件为指标管道提供数据。有关指标目录，请参见 指标。

遥测事件	发出时
`[:wlcp, :packet, :received]`	WLCP 数据包到达。携带 `bytes`。
`[:wlcp, :packet, :malformed]`	数据包解码失败。
`[:wlcp, :packet, :unregistered]`	来自未映射源地址的数据包到达。
`[:wlcp, :pdn, :request]`	收到 PDN 连接请求。
`[:wlcp, :pdn, :accept]`	建立 PDN 连接。
`[:wlcp, :pdn, :reject]`	PDN 连接被拒绝。携带原因。
`[:wlcp, :pdn, :disconnect_request]`	收到 PDN 断开请求。
`[:wlcp, :pdn, :disconnect_accept]`	释放 PDN 连接。
`[:twag, :handover, :in]`	路由 3GPP 到 WLAN 的切换事件。携带 `source`。
`[:twag, :handover, :out]`	处理 WLAN 到 3GPP 的 RAT 更改。携带 `cause`。
`[:twag, :handover, :ap_change]`	处理 AP 之间的移动事件。

WLCP 服务器和会话处理程序还在信息级别记录日志。它们记录每个 PDN 连接请求、接受、拒绝和断开，包含 UE 身份、APN 和 EBI。切换处理程序记录每个切换进入、RAT 更改和 AP 更改，包含 UE 身份。

实施状态

此表总结了实时会话路径上运行的内容、存在但未集成的内容以及未实现的内容。使用它来围绕真实行为进行规划。

功能	状态
TSCM（默认，自动创建 PDN，通过 DHCP 获取 IP）	实时
AP 之间的移动性（S2a 隧道保持）	实时
S2a 创建会话与 Handover Indication 标志	在 S2a 编码器中受支持，但附加路径始终清除它
连接模式协商逻辑和回退	作为模块存在，经过单元测试
STa DER 通告支持的模式	已连接（STa DER 携带首选的 TWAN-Connection-Mode，以及 SSID / BSSID）
STa DEA 读取所选模式	已连接（StaClient 解析协商的模式，默认为 TSCM）
协商的模式应用于实时会话	未连接（`store_connection_mode` 在附加路径上没有调用者）
SCM 作为独立的运行时路径（EAP IP，APN 选择）	未集成（作为 TSCM 提供服务）
WLCP 编解码器（所有消息类型和 IEs）	存在，经过单元测试
WLCP UDP 服务器和每个 UE 会话处理程序	存在，但未由应用程序启动
WLCP 密钥派生（`K_WLCP`，FC=0x21）	存在，经过单元测试
MCM 多个并发 PDN 连接	在处理程序中存在，但未端到端集成
WLCP 的 DTLS 保护	未实现（仅普通 UDP）
MCM 中的专用承载管理	未实现
WLCP 下行数据通知生成	未实现（仅编解码器）

功能	状态
MCM 中的切换	未实现
附加路径上的 3GPP 到 WLAN IP 保持	未执行（附加清除 HI 标志）
WLAN 到 3GPP 的拆除由 RAT 更改原因	未分支（删除承载清理与原因无关）

IP 地址管理（DHCP / DHCPv6 / IPv6 RA / PCO）

概述

OmniTWAG 为每个 UE 分配一个 IP 地址及其相关的网络参数。该子系统涵盖 DHCPv4、DHCPv6、IPv6 路由器广告 (SLAAC) 以及 DNS 和 P-CSCF 地址的传递。该子系统不会随机生成地址。它提供的地址是系统其他部分已经选择的。源地址取决于每个 AP 模式。

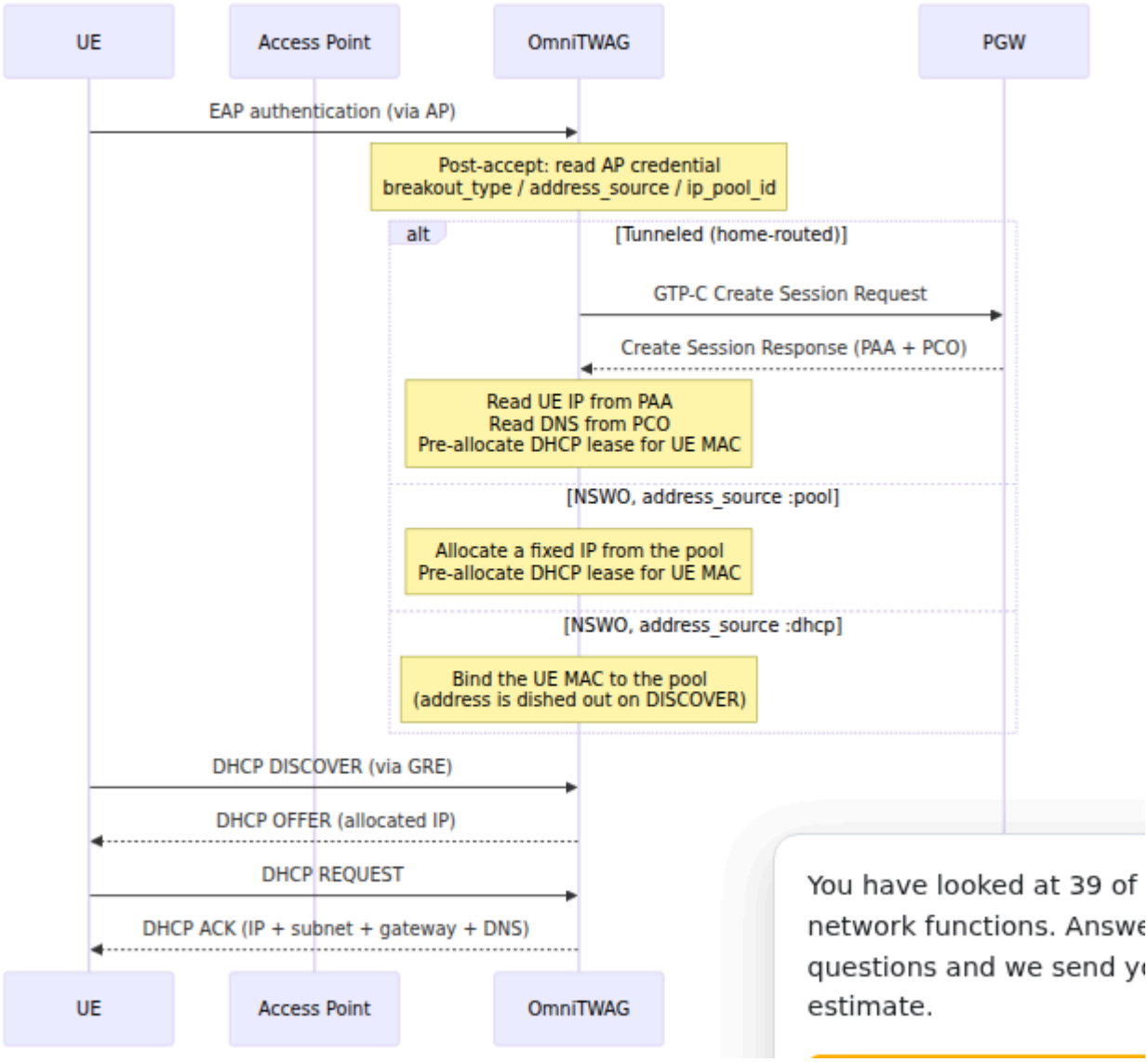
- 在 隧道（家庭路由）模式 下，PGW 通过 S2a GTP 分配地址，OmniTWAG 将相同的地址传递给 UE。该设计遵循 3GPP TS 23.402 第 16.2 节。
- 在 **NSWO**（本地突破）模式 下，没有 PGW。OmniTWAG 从运营商定义的本地 **IP** 池 中自行分配地址。请参见 IP 池。

模式、地址源和池都来自与 RADIUS 源匹配的 AP 凭证。有关 AP 如何映射到 `breakout_type`、`address_source` 和 `ip_pool_id` 的信息，请参见 RADIUS 凭证。

有关操作模式和 GRE 数据路径的信息，请参见 用户平面 参考。有关承载 PAA 和 PCO 的 GTP-C 控制平面的信息，请参见 S2a GTP-C。有关驱动分配的每个 UE 状态机的信息，请参见 会话。

目录

- 概述：谁分配地址
- 地址源
- 隧道会话：PGW 分配的地址
- NSWO 会话：池和 DHCP
- DHCPv4
- DHCPv6
- IPv6 路由器广告 (SLAAC)
- DNS 和 PCO 传递
- 地址池和耗尽
- 配置
- 可观察性



- 故障排除

概述：谁分配地址

OmniTWAG 运行一个轻量级的 DHCPv4 服务器、一个无状态和有状态的 DHCPv6 服务器，以及一个 IPv6 路由器广告服务器。这些服务器在隧道情况下不会扫描范围以寻找可用地址。它们提供 PGW 已经分配的地址。在 NSWO 模式下，同一个 DHCPv4 服务器提供 OmniTWAG 从本地 IP 池中获取的地址。

地址流在 UE 认证时开始。在 Access-Accept 后，每个 UE 会话过程运行一个后接受建立步骤。该步骤读取 AP 凭证以决定模式，然后通过正确的路径分配地址。

You have looked at 39 of
network functions. Answe
questions and we send y
estimate.

地址源

AP 凭证通过两个字段设置地址源：

- `breakout_type`：`:tunneled`、`:nswo` 或 `:local`。
- `address_source`：`:pool` 或 `:dhcp`。这仅在 `breakout_type` 为 `:nswo` 时适用。

模式	凭证	谁选择 **UE** 地址	如何到达 **UE**
隧道（家庭路由）	`breakout_type: :tunneled`	PGW，在 GTP-C 创建会话响应 PAA 中。	预分配的 DHCPv4 租约、DHCPv6 或 IPv6 RA。
NSWO，固定	`breakout_type: :nswo`，`address_source: :pool`	OmniTWAG，从 `ip_pool_id` 池中，在建立时。	预分配的 DHCPv4 租约。
NSWO，分配	`breakout_type: :nswo`，`address_source: :dhcp`	OmniTWAG，从 `ip_pool_id` 池中，在 DHCP DISCOVER 时。	在 DISCOVER 时从 DHCPv4 池中分配。
AP-本地	`breakout_type: :local`	AP 后面的本地网络。	OmniTWAG 不分配地址。

有关凭证字段以及 RADIUS 源如何映射到凭证的信息，请参见 RADIUS 凭证。有关池存储的信息，请参见 IP 池。

隧道会话：**PGW** 分配的地址

对于隧道会话，OmniTWAG 通过 GTP-C 创建会话请求与 PGW 建立 S2a 会话。PGW 以创建会话响应进行回复。该响应携带该子系统使用的两个内容：

- **PDN** 地址分配 **(PAA)**。这是 UE 的 IP 地址。
- 协议配置选项 **(PCO)**。这携带 DNS 服务器、P-CSCF 地址和链路 MTU。

OmniTWAG 从响应中读取 PAA 和 PCO。然后，OmniTWAG 为 UE MAC 地址预分配一个 DHCP 租约。当 UE 发送 DHCP 请求时，OmniTWAG 用这个预分配的地址进行回复。UE 永远不会收到 PGW 未分配的地址。有关创建会话交换的信息，请参见 S2a GTP-C。

PAA 类型设置 OmniTWAG 交付的地址。

PAA 类型	交付给 **UE** 的地址
`ipv4`	来自 PAA 的 IPv4 地址。
`ipv6`	来自 PAA 的 IPv6 地址。
`ipv4v6`	来自 PAA 的 IPv4 地址。

如果创建会话响应没有 PAA，OmniTWAG 不会预分配租约。在这种情况下，UE 不会收到地址。

NSWO 会话：池和 DHCP

NSWO 会话没有 PGW，因此 OmniTWAG 拥有地址空间。地址来自凭证 `ip_pool_id` 指定的 IP 池。一个池包含一个 IPv4 CIDR、一个网关、一个 DNS 服务器列表和一个租约时间。请参见 IP 池。

凭证的 `address_source` 字段选择池地址如何到达 UE。

`address_source: :pool`（固定）

在后接受建立时，OmniTWAG 从池中分配一个地址，以订阅者身份为键。该分配是幂等的，因此相同的身份在重新建立期间保持相同的地址。OmniTWAG 为 UE MAC 预分配一个 DHCPv4 租约，使用该地址、池网关、池 DNS 服务器和从池 CIDR 派生的子网掩码。此时，它还在 GRE 数据路径中注册 UE。当 UE 发送 DISCOVER 时，服务器提供预分配的地址。

`address_source: :dhcp`（在 DISCOVER 时分配）

在建立时，OmniTWAG 尚未分配地址。它绑定 UE MAC 到池。在该 MAC 的第一次 DHCP DISCOVER 时，DHCPv4 服务器从池范围中分配下一个可用地址，记录租约并提供它。这是新的动态分配路径。地址是在 DISCOVER 时选择的，而不是在建立时选择的。

在会话拆除时，OmniTWAG 释放池分配，并且对于 `:dhcp`，移除池绑定和动态租约。如果池缺失或耗尽，或者 AP 没有 `ip_pool_id`，OmniTWAG 不分配地址，UE 自行寻址。

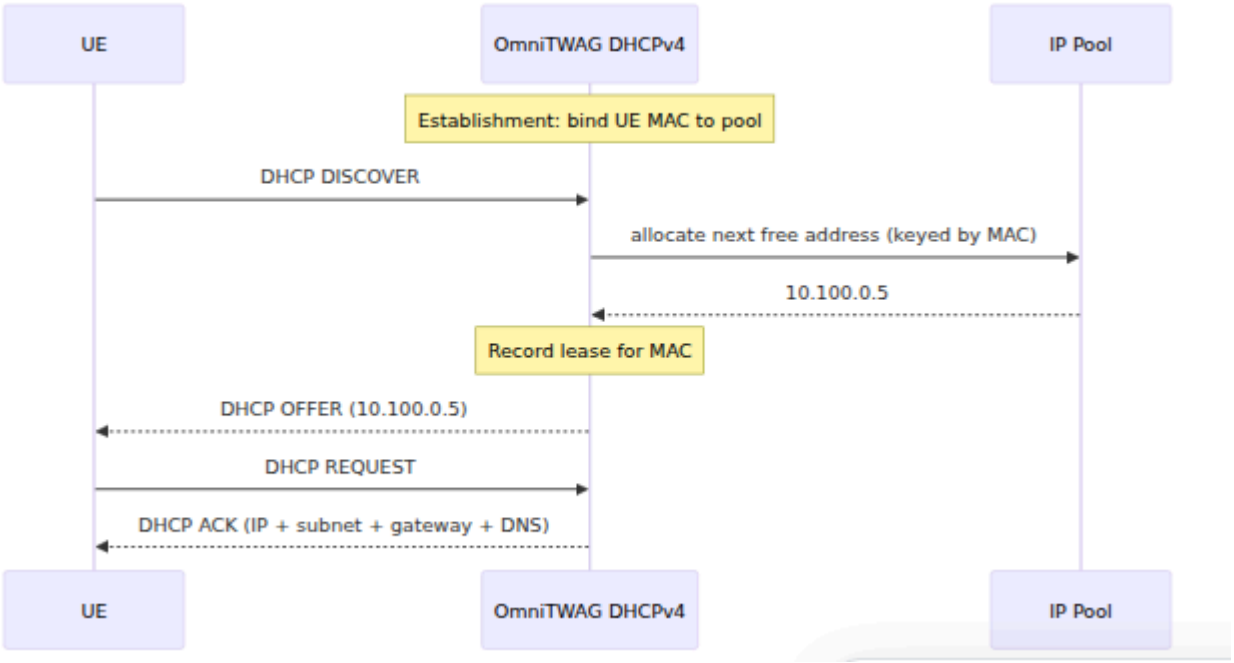
DHCPv4

DHCPv4 服务器监听 UDP 端口 `67`。它将每个租约存储在一个内存中的 ETS 表中，以 UE MAC 地址为键。租约通过两条路径之一到达表中。

- 预分配。当 PGW 分配地址（隧道）或 OmniTWAG 分配固定池地址（NSWO `address_source: :pool`）时，会话过程插入租约。
- 在 **DISCOVER** 时分配。对于 NSWO `address_source: :dhcp`，会话过程将 UE MAC 绑定到池。服务器在内存中持有每个 MAC 的绑定。在没有租约的绑定 MAC 的第一次 DISCOVER 时，服务器从池中分配下一个可用地址，记录租约并提供它。

租约生命周期

对于预分配的租约，服务提供请求 MAC 已经存在于表中的地址。对于绑定的 MAC，服务器在 DISCOVER 时通过从池中分配来创建租约。



OmniTWAG 处理这些 DHCP 消息类型。

消息	类型代码	**OmniTWAG** 行为
DISCOVER	1	如果 MAC 有租约，OmniTWAG 发送 OFFER。如果没有租约但 MAC 绑定到池，OmniTWAG 分配下一个可用池地址，记录租约并发送 OFFER。如果没有租约且没有绑定，OmniTWAG 记录警告并不发送任何内容。
REQUEST	3	如果租约存在且请求的 IP (选项 50) 与租约匹配，或者请求没有请求的 IP，OmniTWAG 发送 ACK。如果请求的 IP 不匹配，OmniTWAG 发送 NAK。如果没有租约，OmniTWAG 发送 NAK。
RELEASE	7	OmniTWAG 将租约标记为 `released`���OmniTWAG 保留租约记录。PDN 连接保持开启，遵循 TS 23.402 §16.1.5.2 NOTE 2。
DECLINE	4	OmniTWAG 记录警告。OmniTWAG 保留租约记录。
INFORM	8	OmniTWAG 发送仅包含配置的 ACK。ACK 携带网关、子网掩码和 DNS，但不包含地址 (`yiaddr` 为 `0.0.0.0`)。

对于 INFORM，OmniTWAG 使用配置中的 `default_gateway` 和 `default_subnet_mask`。OmniTWAG 在 INFORM 回复中使用 `8.8.8.8` 作为 DNS 服务器。

租约内容

每个租约包含 UE IP、子网掩码、网关、DNS 服务器和租约时间。对于隧道会话，会话过程从 PGW 响应中设置 UE IP 和 DNS 服务器，从 DHCP 配置中设置子网掩码、网关和租约时间。对于 NSWO 会话，这些值来自 IP 池：来自池范围的 IP、来自池 CIDR 前缀的子网掩码，以及来自池记录的网关、DNS 服务器和租约时间。请参见 IP 池 和 配置。

租约记录还具有 `allocated_at` 时间和 `expires_at` 时间。`expires_at` 时间为 `allocated_at` 加上租约时间。OmniTWAG 使用这些值报告租约状态。后台清理程序每 60 秒运行一次，删除任何 `expires_at` 已过期的租约。租约时间为零的租约没有 `expires_at`，且永远不会过期。当会话拆除时，���约也会被移除。

回复中的 **DHCP** 选项

OmniTWAG 使用以下 DHCP 选项构建每个 OFFER 和 ACK。

选项	代码	内容
消息类型	53	OFFER 或 ACK。
服务器标识符	54	DHCP `server_ip`。
子网掩码	1	租约子网掩码。
路由器	3	租约网关。
域名服务器	6	租约 DNS 服务器。如果列表为空，则省略。
IP 地址租约时间	51	租约时间（以秒为单位）。如果租约时间为零，则省略。
广播地址	28	从 IP 和子网掩码计算得出。仅在 UE 在其参数请求列表（选项 55）中列出此选项时发送。

通过 **GRE** 的 **DHCP** 传输

使用 GRE 隧道的接入点在 GRE 封装的以太网帧中转发 UE DHCP 请求。GRE 服务器提取 DHCP 有效负载并将其传递给 DHCPv4 服务器。OmniTWAG 用源 AP 地址标记数据包。当 OmniTWAG 构建回复时，它将 DHCP 有效负载封装在

UDP/IP/以太网帧中，并通过 GRE 隧道将帧发送回同一 AP。源为 `10.0.0.1:67`，目标为客户端端口 `68` 的广播地址。

直接到达 UDP 套接字的 DHCP 请求（而不是通过 GRE）在 UDP 套接字上进行回复。如果源端口不是标准客户端端口 `68`，OmniTWAG 会回复源地址和端口。此路径支持测试工具和��继。否则，OmniTWAG 遵循 RFC 2131：如果广播标志被设置或 `ciaddr` 为零，则使用广播地址；如果该地址被设置，则使用 `ciaddr`。

DHCPv6

DHCPv6 服务器在 IPv6 上监听 UDP 端口 `547`。它支持无状态模式和有状态模式。

- 无状态模式 通过信息请求 (类型 11) 回复信息回复 (类型 12)。回复携带 DNS 服务器。这遵循 RFC 3736 和 RFC 3646。当 UE 从其 RA 处理设置其他配置 (O) 标志时，它使用此模式。
- 有状态模式 通过请求 (类型 1) 回复广告 (类型 2)，并通过请求 (类型 3) 回复 (类型 7)。此模式在 IA_NA 选项中提供预分配的 IPv6 地址。这遵循 RFC 3315。

服务器为每个客户端 DUID 预分配一个有状态地址。GTP 控制平面插入分配。服务器在内存表中保存这些分配，以客户端 DUID 为键。

DHCPv6 消息处理

消息	类型	**OmniTWAG** 行为
信息请求	11	OmniTWAG 发送信息回复，包含服务器 DUID、回显的客户端 ID 和 DNS 服务器。
请求	1	OmniTWAG 发送广告。如果客户端 DUID 有分配，广告携带 IA_NA 地址。如果没有，广告携带 `NoAddrsAvail` 状态（代码 2）。
请求	3	OmniTWAG 发送回复。回复确认客户端 DUID 的 IA_NA 地址。
其他类型	-	OmniTWAG 忽略该消息。

每个回复回显客户端事务 ID，遵循 RFC 3315 §15。每个包含分配地址的回复根据租约时间设置 DHCPv6 定时器。T1 为租约时间的一半。T2 为租约时间的四分之五。

服务器 DUID

OmniTWAG 构建类型为 DUID-EN（企业编号）的服务器 DUID。它使用 3GPP 企业编号 `10415`。标识符源自节点名称，因此对于给定节点是稳定的。

回复中的 DHCPv6 选项

选项	代码	内容
客户端标识符	1	从请求中回显的客户端 ID（如果存在）。
服务器标识符	2	服务器 DUID。
非临时地址的身份关联 (IA_NA)	3	分配的 IPv6 地址，或 `NoAddrsAvail` 状态。仅限有状态模式。
DNS 递归名称服务器	23	配置的 DNS 服务器。如果列表为空，则省略。
域搜索列表	24	配置的搜索列表。仅在设置了 `dhcpv6_domain_list` 时发送。

IPv6 路由器广告 (SLAAC)

IPv6 RA 服务器通过无状态地址自配置 (SLAAC) 向 UE 传递 PGW 分配的 IPv6 前缀。它根据 RFC 4861 实现 ICMPv6 路由器广告。

GTP 控制平面为每个接口添加一个前缀。服务器将前缀保存在内存中，按接口名称分组。服务器在两种情况下发送路由器广告：

1. 在周期性定时器上。默认间隔为 `30` 秒。
2. 按需，当控制平面请求在接口上进行 RA 时。

服务器还解析路由器请求 (ICMPv6 类型 133) 消息。

路由器广告内容

每个路由器广告携带这些字段和选项。

字段或选项	值
当前跳数限制	来自 `hop_limit`。默认 `64`。
路由器生命周期	来自 `router_lifetime`。默认 `1800` 秒。
M 和 O 标志	均为 `0`。RA 不会信号管理地址配置或其他配置。
源链路层地址（选项 1）	源 MAC（如果是 6 字节值）。
MTU（选项 5）	仅在提供 MTU 时包含。
前缀信息（选项 3）	每个前缀一个。L（链路）和 A（自主）标志均被设置。
递归 DNS 服务器，RDNSS（选项 25）	来自前缀的 DNS 服务器，遵循 RFC 8106。如果没有 DNS 服务器，则省略。

每个前缀信息选项携带前缀、前缀长度��有效生命周期（默认 `86400` 秒）和首选生命周期（默认 `14400` 秒）。由于 A 标志被设置，UE 从前缀形成自己的 IPv6 地址。RDNSS 生命周期是所有前缀中最大的有效生命周期。

DNS 和 PCO 传递

OmniTWAG 通过多种方式传递 DNS 和 P-CSCF 信息。源是 PGW PCO（如果可能），否则为配置。

PGW 的 PCO

PCO 解析器从创建会话响应中读取协议配置选项，遵循 3GPP TS 24.008。解析器返回以下字段。

字段	**PCO** 协议 **ID**	含义
`dns_ipv4`	`0x000D`	IPv4 DNS 服务器地址。
`dns_ipv6`	`0x0003`	IPv6 DNS 服务器地址。
`pcscf_ipv4`	`0x000C`	IPv4 P-CSCF 地址。
`pcscf_ipv6`	`0x0001`	IPv6 P-CSCF 地址。
`mtu`	`0x0010`	IPv4 链路 MTU。
`ip_alloc_nas`	`0x000A`	一个标志。如果 PGW 通过 NAS 信令信号 IP 分配，则为 `true`。

解析器还将每个解析的选项保存在 `raw` 列表中。如果 PCO 二进制在某个地方损坏，解析器将返回它已读取的字段。它不会使整个解析失败。

OmniTWAG 还可以为创建会话请求构建 PCO 请求。该请求列出 OmniTWAG 希望 PGW 返回的选项，例如 IPv4 DNS 服务器和链路 MTU。

DNS 传递��� **UE**

DNS 服务器通过 UE 使用的传输到达 UE。

传输	**DNS** 源	备注
DHCPv4 (选项 6)	来自 PCO 的 IPv4 DNS 服务器。	如果 PCO 没有 IPv4 DNS 服务器，OmniTWAG 使用 `8.8.8.8`。
DHCPv6 (选项 23)	DHCPv6 DNS 服务器列表。	从 PGW PCO 或 `dhcpv6_dns_servers` 设置。如果未设置，则默认为 Google 公共 DNS IPv6。
IPv6 RA (RDNSS，选项 25)	附加到广告前缀的 IPv6 DNS 服务器。	随前缀一起传递。

DHCPv4 租约从 PCO `dns_ipv4` 列表中获取 IPv4 DNS 服务器。如果该列表为空，OmniTWAG 将回退到 `8.8.8.8` 作为租约。

> 关于 **P-CSCF** 的说明。 PCO 解析器从 PGW 读取 P-CSCF 地址。当前的 DHCP、DHCPv6 和 RA 传递路径不添加 P-CSCF DHCP 选项或 RA 选项。P-CSCF 值被解析并可用，但它们不会通过审计代码中的此子系统传递给 UE。

地址池和耗尽

耗尽行为取决于模式。

- 隧道（家庭路由）。没有本地地址池。PGW 拥有地址空间。OmniTWAG 精确传递 PGW 为每个会话分配的地址。没有本地范围可供耗尽。如果 PGW 不返回 PAA，UE 将不接收地址，OmniTWAG 记录该情况。
- **NSWO**。OmniTWAG 从本地 IP 池中分配，因此池可能耗尽。池是最多为 `/30` 的 IPv4 CIDR分配器跳过网络和广播地址，并分配第一个可用主机。当每个主机都在使用时，分配器返回 `pool_exhausted`。对于 `address_source: :pool`，UE 然后自我寻址。对于 `address_source: :dhcp`，DISCOVER 不会得到回复，OmniTWAG 记录耗尽。分配在内存中，并在会话重新建立时重建，类似于 DHCP 租约。有关池定义和容量的信息，请参见 IP 池。

`Twag.Cidr` 帮助程序为平台执行子网数学。它解析 IPv4 和 IPv6 CIDR 字符串，并测试地址是否在子网内。RADIUS 子系统使用此帮助程序进行源 IP 授权。IP 池存储使用它来推导池子网掩码、可用主机范围和默认网关。

配置

IP 管理服务器从 `:omnitwag` 应用程序配置中读取其配置。主要块为 `dhcp` 和 `ipv6_ra`。DHCPv6 读取一小组顶级键。

```
config :omnitwag,
  # DHCPv4 服务器
  dhcp: %{
    enabled: true,
    interface: "wlan0",
    server_ip: "10.0.0.1",
    default_lease_time: 3600,
    default_subnet_mask: "255.255.255.0",
    default_gateway: "10.0.0.1"
  },

  # IPv6 路由器广告
  ipv6_ra: %{
    enabled: true,
    interface: "wlan0",
    ra_interval: 30,
    hop_limit: 64,
    router_lifetime: 1800
  },

  # DHCPv6（可选顶级键）
  dhcpv6_dns_servers: ["2001:4860:4860::8888",
"2001:4860:4860::8844"],
  dhcpv6_domain_list: ["example.com"]
```

DHCPv4 参数（`dhcp`）

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`true`	启用 DHCPv4 服务器。设置为 `false` 以禁用它，例如在测试环境中。
`interface`	字符串	否	`wlan0`	DHCP 服务的网络接口。对于 GRE 隧道模式使用 `gre`。
`server_ip`	字符串	否	`10.0.0.1`	DHCP 服务器标识符。OmniTWAG 在 DHCP 选项 54 中发送此值，并作为 GRE 隧道回复的源。
`default_lease_time`	整数	否	`3600`	租约时间（以秒为单位）。OmniTWAG 将此值应用于每个预分配的租约。
`default_subnet_mask`	字符串	否	`255.255.255.0`	租约的子网掩码，以及 RFC 2131 广播计算的子网掩码。
`default_gateway`	字符串	否	`10.0.0.1`	租约的默认网关（DHCP 选项 3）。

IPv6 RA 参数（`ipv6_ra`）

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`true`	启用 IPv6 RA 服务器。设置为 `false` 以禁用它。
`interface`	字符串	否	`wlan0`	路由器广告的网络接口。
`ra_interval`	整数	否	`30`	定期路由器广告之间的间隔（以秒为单位）。
`hop_limit`	整数	否	`64`	路由器广告中的当前跳数限制字段。
`router_lifetime`	整数	否	`1800`	路由器广告中的路由器生命周期（以秒为单位）。

DHCPv6 参数（顶级）

参数	类型	必需	默认	描述
`dhcpv6_dns_servers`	字符串列表	否	Google 公共 DNS IPv6（`2001:4860:4860::8888`，`2001:4860:4860::8844`）	DHCPv6 回复的 IPv6 DNS 服务器。PGW PCO 也可以在运行时设置此列表。
`dhcpv6_domain_list`	字符串列表	否	（未设置）	DHCPv6 回复的 DNS 搜索列表（选项 24）。如果未设置，OmniTWAG 将省略该选项。

可观察性

Web 界面

控制面板在 `/dhcp` 处有一个 **DHCP** 池页面。该页面列出了每个 DHCPv4 租约，包括 UE MAC 地址、IP 地址、租约开始时间、租约到期时间和状态。状态为 `Active` 或 `Expired`。该页面显示每个活动租约的实时倒计时。选择租约行以查看完整的租约详细信息：子网掩码、网关、DNS 服务器、租约持续时间和时间戳。该页面每秒刷新一次。如果 DHCPv4 服务器未运行，该页面将显示 `Server Not Running`。

REST API

REST API 在 `DHCP` 标签下提供 DHCPv4 租约数据。

方法	路径	描述
GET	`/dhcp` (索引)	列出所有 DHCPv4 租约及其摘要计数。
GET	`/dhcp/{id}` (显示)	通过 MAC 地址获取一个租约。`id` 是以冒号分隔的 MAC。

索引响应具有 `enabled` 标志、`leases` 列表、总 `count`、`active` 计数和 `expired` 计数。如果 DHCPv4 服务器未运行，索引返回 `enabled: false` 和一个空列表。显示端点返回 `404` 如果没有租约与 MAC 匹配，以及 `503` 如果服务器未运行。

每个租约对象具有以下字段。

字段	描述
`mac`	UE MAC 地址，以冒号分隔。
`ip`	分配的 IPv4 地址。
`subnet_mask`	租约子网掩码。
`gateway`	租约网关。
`dns_servers`	DNS 服务器列表。
`lease_time`	租约持续时间（以秒为单位）。
`allocated_at`	租约开始时间，自 Unix 纪元以来的秒数。
`expires_at`	租约到期时间，自 Unix 纪元以来的秒数。
`status`	`active` 或 `expired`。
`time_remaining`	距到期的秒数，或 `0` 如果已过期。

监测事件

IP 管理服务器会发出监测事件。操作员可以将这些事件附加到指标导出器。有关 Prometheus 端点的信息，请参见 指标参考。

事件	源	含义
`[:dhcp, :lease, :allocate]`	DHCPv4	为一个 MAC 预分配了一个租约。
`[:dhcp, :lease, :deallocate]`	DHCPv4	在拆除时移除了一个租约。
`[:dhcp, :lease, :reap]`	DHCPv4	一个过期的租约被后台清理程序移除。
`[:dhcp, :discover]`	DHCPv4	一个 DISCOVER 被现有租约的 OFFER 回复。
`[:dhcp, :discover, :pool_allocated]`	DHCPv4	从池绑定的 MAC 的 DISCOVER 被分配了一个新的池地址（NSWO `:dhcp`）。
`[:dhcp, :discover, :no_allocation]`	DHCPv4	一个没有租约且没有池绑定的 MAC 收到的 DISCOVER。
`[:dhcp, :ack]`	DHCPv4	一个 REQUEST 被回复为 ACK。
`[:dhcp, :nak]`	DHCPv4	一个 REQUEST 被回复为 NAK。
`[:dhcp, :release]`	DHCPv4	一个客户端释放了其租约。
`[:dhcp, :decline]`	DHCPv4	一个客户端拒绝了其租约。
`[:dhcp, :inform]`	DHCPv4	一个 INFORM 被回复。
`[:dhcp, :packet, :malformed]`	DHCPv4	收到一个格式错误的 DHCP 数据包。
`[:dhcpv6, :info_request, :received]`	DHCPv6	收到一个信息请求。
`[:dhcpv6, :solicit, :received]`	DHCPv6	收到一个请求。

事件	源	含义
`[:dhcpv6, :request, :received]`	DHCPv6	收到一个请求。
`[:dhcpv6, :info_reply, :sent]`	DHCPv6	发送了一个信息回复。
`[:dhcpv6, :advertise, :sent]`	DHCPv6	发送了一个广告。
`[:dhcpv6, :reply, :sent]`	DHCPv6	发送了一个回复。
`[:ipv6_ra, :prefix, :add]`	IPv6 RA	一个前缀被添加到一个接口。
`[:ipv6_ra, :prefix, :remove]`	IPv6 RA	一个前缀从一个接口中移除。
`[:ipv6_ra, :advertisement, :sent]`	IPv6 RA	发送了一个路由器广告。

故障排除

UE 未获得 IP 地址

症状：UE 认证成功，但未获得 IPv4 地址。DHCP 池页面未显示 UE MAC 的租约。触发了 `[:dhcp, :discover, :no_allocation]` 事件。

可能的原因（隧道）：

- GTP 创建会话响应没有 PAA，因此 OmniTWAG 未预分配租约。
- GTP-C 会话未建立，因此未创建租约。

可能的原因（**NSWO**）：

- AP 凭证没有 `ip_pool_id`，因此 OmniTWAG 不分配地址。

- 池已耗尽，因此无法分配可用地址。
- 对于 `address_source: :dhcp`，DISCOVER 在 MAC 绑定到池之前到达，或者绑定未运行，因为会话尚未有 MAC。

常见原因：

- DHCPv4 服务器未运行。

解决方案：

1. 检查 OmniTWAG 日志以获取建立路径（隧道与 NSWO）和 UE IP。
2. 对于隧道会话，确认 PGW 为 APN 返回 PAA。
3. 对于 NSWO 会话，确认 AP 凭证命名有效的 `ip_pool_id`，并且池有可用容量。请参见 IP 池 和 RADIUS 凭证。
4. 确认 DHCP 池页面显示 DHCPv4 服务器正在运行。

UE 获得地址但没有 DNS

症状： UE 有一个 IPv4 地址，但无法解析名称。

可能的原因：

- PGW PCO 未返回 IPv4 DNS 服务器。在这种情况下，OmniTWAG 使用 `8.8.8.8`。
- UE 无法访问 PGW 提供的 DNS 服务器。

解决方案：

1. 在 DHCP 池页面或通过 REST API 阅读租约 DNS 服务器。
2. 确认 PGW PCO 返回预期的 DNS 服务器。
3. 确认 DNS 服务器可以从 UE 数据路径访问。

UE 收到 NAK

症状： UE 发送 REQUEST 并收到 NAK。触发了 `[:dhcp, :nak]` 事件。

可能的原因：

- UE 请求的 IP（选项 50）与其预分配的租约不匹配。
- 租约已被移除，例如在会话拆除后。

解决方案:

1. 确认 UE MAC 有租约。
2. 确认 UE 请求的 IP 与 PGW 或池分配的 IP 相同。
3. 如果会话已拆除，让 UE 在重新认证后重新启动 DHCP 交换。

相关页面

- □述
- IP 池
- RADIUS 凭证
- 会话
- S2a GTP-C
- 用户平面
- 在线计费 (Gy)

□述

IP 池

□述

OmniTWAG 从 **IP** 池 中为 NSWO（非无缝 WLAN 卸载）会话分配 IPv4 地址。NSWO 是以 TWAG 为基础的本地突破：OmniTWAG 终止用户流量并在本地进行突破。IP 池定义了一个 IPv4 CIDR 块、一个网关、DNS 服务器和租约时间。操作员可以通过 REST API 或控制面板 Web 界面按需添加、更改和删除池。

池定义是持久的。OmniTWAG 将它们存储在 DETS 文件中，因此池在重启后仍然存在。地址分配是在内存中进行的。OmniTWAG 在重启后会在会话重新建立时重建分配。

目录

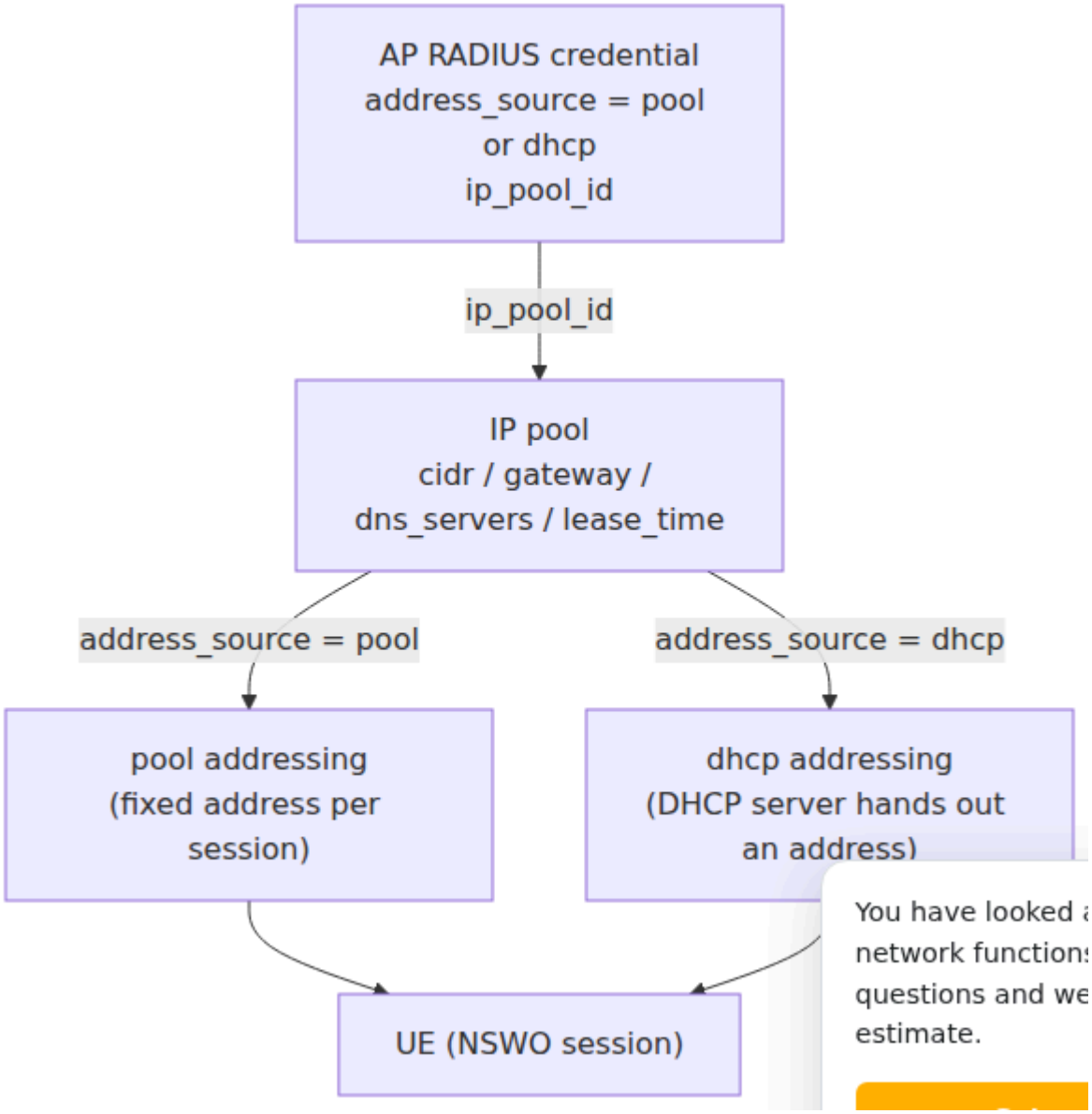
- □念
- 会话如何获取地址
- 地址分配
- 池模型
- REST API
- Web 界面
- 故障排除

概念

池是一组来自一个 IPv4 CIDR 块的地址。每个池都有一个网关、一组 DNS 服务器和一个租约时间。

IP 池通过 AP 的 RADIUS 凭证与接入点相□□□。凭证中有一个 `ip_pool_id` 字段。该字段保存池的 `id`。有关凭证模型，请参见 RADIUS 共享密钥。来自该接入点的 NSWO 会话从链接的池中获取地址。

每个凭证还有一个 `address_source` 字段。该字段选择 OmniTWAG 如何将地址分配给 UE（用户设备）。有两种地址来源：



地址来源	行为
`pool`	OmniTWAG 从池中为每个会话预分配一个固定地址。该地址在会话建立时选择。
`dhcp`	OmniTWAG DHCP 服务器在回复 UE 的 DHCP DISCOVER 时从池范围中分配一个地址。

会话如何获取地址

OmniTWAG 从与 AP 凭证链接的池中为每个 NSWO 会话分配地址。步骤如下:

1. 接入点通过其 RADIUS 凭证进行身份验证。

2. OmniTWAG 读取凭证上的 `ip_pool_id` 字段以查找池。
3. OmniTWAG 读取凭证上的 `address_source` 字段。
4. 如果 `address_source` 是 `pool`，OmniTWAG 从池中为会话预分配一个固定地址。
5. 如果 `address_source` 是 `dhcp`，OmniTWAG 等待 UE 的 DHCP DISCOVER。然后 DHCP 服务器从池范围中分配一个地址。
6. OmniTWAG 将地址、网关和来自池的 DNS 服务器提供给 UE。

地址分配

OmniTWAG 从池 CIDR 中分配主机。分配时跳过网络地址和广播地址。第一个可用主机是网络地址加一。

池 CIDR 前缀必须为 `/30` 或更大才能有可用主机。一个 `/30` 块有 2 个可用主机。更小的前缀可用主机更少，因此 OmniTWAG 会拒绝它。

当每个可用主机都被分配时，OmniTWAG 返回 `pool_exhausted`。来自链接接入点的后续会话无法获取地址，直到释放一个分配。

池上的 `allocated` 字段保持当前分配的计数。OmniTWAG 从内存中的分配计算此计数。

池模型

每个池都有以下字段。

字段	类型	描述
`id`	字符串	唯一标识符（UUID）。服务器设置此值。
`name`	字符串	可读标签，例如 `Melbourne guest WiFi`。
`cidr`	字符串	池的 IPv4 CIDR 块，例如 `10.60.0.0/24`。前缀必须为 `/30` 或更大才能有可用主机。
`gateway`	字符串	分配给 UE 的 IPv4 网关地址。如果省略，则默认为 CIDR 的第一个可用主机。
`dns_servers`	字符串列表	分配给 UE 的 IPv4 DNS 服务器地址。
`lease_time`	整数	租约时间（以秒为单位）。默认为 `3600`。
`default`	布尔值	当为 `true` 时，此池为 DHCP 默认范围。DHCP 服务器为任何没有预分配租约和没有池绑定的客户端提供此范围。一次只能有一个池是默认的。默认为 `false`。
`allocated`	整数	当前分配的计数。服务器计算此值。
`inserted_at`	整数	池创建的时间，自 Unix 纪元��来的毫秒数。服务器设置此值。
`updated_at`	整数	池最后更改的时间，自 Unix 纪元以来的毫秒数。服务器设置此值。

操作员设置 `name`、`cidr`、`gateway`、`dns_servers` 和 `lease_time`。服务器设置 `id`、`allocated`、`inserted_at` 和 `updated_at`。

REST API

OmniTWAG REST API 在配置的 API 端口（默认是 `8444`）上通过 HTTPS 提供这些端点。基本路径是 `/ip_pools`。

方法	路径	操作	描述
GET	`/ip_pools`	index	列出所有池。结果是分页的，并可以通过 `name` 或 `cidr` 搜索。
GET	`/ip_pools/{id}`	show	通过 `id` 获取一个池。
POST	`/ip_pools`	create	创建一个池。
PUT	`/ip_pools/{id}`	update	更改一个池。
DELETE	`/ip_pools/{id}`	delete	删除一个池。

请求体

`POST` 和 `PUT` 接受一个 JSON 体，包含以下字段。

字段	类型	必需	描述
`name`	字符串	否	可读标签。
`cidr`	字符串	是	IPv4 CIDR 块。前缀必须为 `/30` 或更大。
`gateway`	字符串	否	IPv4 网关地址。如果省略，则默认为 CIDR 的第一个可用主机。
`dns_servers`	字符串列表	否	IPv4 DNS 服务器地址。
`lease_time`	整数	否	租约时间（以秒为单位）。默认为 `3600`。
`default`	布尔值	否	将此池标记为 DHCP 默认范围。设置它会清除任何其他池上的标志。

`PUT` 仅更改请求体中的字段。请求体中没有的字段保持其当前值。

响应

状态	含义
`200 OK`	请求成功（索引、显示、更新、删除）。
`201 Created`	池已创建。
`404 Not Found`	没有具有给定 `id` 的池。
`422 Unprocessable Entity`	`cidr` 不是有效的 IPv4 CIDR 块，或者前缀太小以至于没有可用主机。

示例：列出池

请求：

```
GET /ip_pools
```

响应：

```
{
  "count": 1,
  "ip_pools": [
    {
      "id": "0d1e2f3a-4b5c-6d7e-8f90-1a2b3c4d5e6f",
      "name": "Melbourne guest WiFi",
      "cidr": "10.60.0.0/24",
      "gateway": "10.60.0.1",
      "dns_servers": ["8.8.8.8", "8.8.4.4"],
      "lease_time": 3600,
      "allocated": 12,
      "inserted_at": 1734512400000,
      "updated_at": 1734512400000
    }
  ]
}
```

示例：创建一个池

请求：

```
POST /ip_pools
```

```
{
  "name": "Melbourne guest WiFi",
  "cidr": "10.60.0.0/24",
  "gateway": "10.60.0.1",
  "dns_servers": ["8.8.8.8", "8.8.4.4"],
  "lease_time": 3600
}
```

成功的请求返回 `201 Created`，并包含新池及其 `id`。如果省略 `gateway`，池将使用 CIDR 的第一个可用主机（在此示例中为 `10.60.0.1`）。

示例：**CIDR** 太小

请求：

```
{
  "name": "Broken pool",
  "cidr": "10.60.0.0/31"
}
```

响应（`422 Unprocessable Entity`）：

```
{
  "error": "cidr too small: 10.60.0.0/31 has no usable hosts"
}
```

默认池

一个池可以是 DHCP 默认范围。DHCP 服务器为任何没有预分配租约和没有明确池绑定的客户端从默认池中分配一个地址。使用默认池为基础 DHCP 服务提供 API 定义的范围，而不仅仅是 NSWO 会话。

通过 `default` 字段或网页上的复选框设置默认值。一次只能有一个池是默认的。当您将一个池标记为默认时，OmniTWAG 会清除先前默认池上的标志。

每个活动租约记录其来源。请参见 IP 租约 视图：租约显示 `PGW (tunneled)`、命名的 `NSWO pool` 或 `default pool`。

Web 界面

控制面板有一个 **IP** 池 管理页面。该页面列出了每个池及其名称、CIDR、网关和分配计数。该页面有添加、编辑和删除池的控件。

以前的 **DHCP** 池 页面现在标记为 **IP** 租约。IP 租约页面显示活动租约。

故障排除

池已耗尽

症状：来自接入点的新 NSWO 会话无法获取地址。OmniTWAG 返回 `pool_exhausted`。

可能原因：

- 池 CIDR 中的每个可用主机都已分配。
- 池 CIDR 对于活动会话的数量来说太小。

解决方案：

1. 检查 `allocated` 计数与 CIDR 中可用主机的数量。
2. 扩大池 CIDR 以添加更多可用主机，或通过结束空闲会话释放过期分配。
3. 确认租约时间没有过长，以至于释放的地址仍然保持分配状态。

接入点未使用池

症状：来自接入点的会话没有从任何 IP 池中获取地址。

可能原因：

- 接入点使用的是接入点拥有的本地突破，而不是以 TWAG 为基础的 NSWO。一个本地的、接入点拥有的突破接入点为其自己的客户端提供地址，而不使用 OmniTWAG 池。
- AP 凭证没有 `ip_pool_id`，因此没有链接池。

解决方案：

1. 确认接入点使用的是以 TWAG 为基础的 NSWO。只有 NSWO 会话使用池。
2. 如果接入点应该使用池，请在其 RADIUS 凭证上设置 `ip_pool_id` 为池的 `id`。请参见 RADIUS 共享密钥。

UE 未获取地址

症状：UE 连接但未接收到 IPv4 地址。

可能原因：

- 凭证的 `address_source` 与 UE 请求地址的方式不匹配。使用 `dhcp` 时，UE 必须发送 DHCP DISCOVER。使用 `pool` 时，OmniTWAG 预分配地址。
- 链接的池已耗尽。
- 凭证上的 `ip_pool_id` 指向一个不存在的池。

解决方案：

1. 确认凭证上的 `address_source` 与 UE 的行为匹配。
2. 确认链接的池有可用地址。检查 `allocated` 计数。
3. 确认凭证上的 `ip_pool_id` 指向一个有效的池。

述

TWAG 指标参考

述

TWAG 在端口 **9568**（由 `prometheus.port` 设置）上公开 Prometheus 指标。仪表指标每 5 秒由轮询器刷新。计数器指标从触发它们的代码路径中发出。

目录

- 指标端点
- RADIUS 认证指标
- RADIUS 授权和速率限制指标
- RADIUS 计费指标
- EAP 指标
- 客户端、AP 和会话仪表
- 数据吞吐量指标
- TWAG 会话和数据路径指标
- Gy 在线计费指标
- STa Diameter 指标
- Diameter 消息指标
- NSWO 指标
- DHCP 指标
- DHCPv6 和 IPv6 RA 指标
- WLCP 指标
- Erlang VM 指标
- 示例 Grafana 查询
- 警报规则
- 相关页面

指标端点

```
http://<twag-host>:9568/metrics
```

Prometheus 将每个事件名称扁平化为指标名称，通过用下划线替换点。例如事件 `[:radius, :access, :request, :count]` 被公开为 `radius_access_request_count`。

RADIUS 认证指标

这些计数器跟踪每个 RFC 2865 的 RADIUS Access-Request/Response 流。

指标	类型	标签	描述
`radius_access_request_count`	计数器	无	接收到的 RADIUS Access-Request 数据包。
`radius_access_accept_count`	计数器	无	发送的 RADIUS Access-Accept 数据包（成功认证）。
`radius_access_reject_count`	计数器	无	发送的 RADIUS Access-Reject 数据包（认证失败）。
`radius_access_challenge_count`	计数器	无	发送的 RADIUS Access-Challenge 数据包（EAP 交换进行中）。

示例查询：

```
# 每秒请求速率
rate(radius_access_request_count[5m])

# 挑战速率（活动 EAP 协商）
rate(radius_access_challenge_count[5m])

# 认证成功比率
sum(rate(radius_access_accept_count[5m])) /
  (sum(rate(radius_access_accept_count[5m])) +
sum(rate(radius_access_reject_count[5m])))
```

RADIUS 授权和速率限制指标

这些计数器跟踪在认证之前或期间被丢弃或标记的数据包。请参见 RADIUS 拒绝 和 RADIUS 共享密钥。

指标	类型	标签	
`radius_auth_unauthorized_source_count`	计数器	`source`	来自未匹配凭据子 AP）。该数据包
`radius_auth_bad_secret_count`	计数器	`source`, `credential_id`	来自匹配凭据但未 包（错误的共享密
`radius_access_rate_limited_count`	计数器	`nas_ip`	因 NAS 在窗 `rate_lim` 而被丢弃的 Acc

示例查询：

```
# 出现未知 AP（为这些添加凭据）
increase(radius_auth_unauthorized_source_count[1h])

# 共享密钥错误的 AP
sum by (source) (increase(radius_auth_bad_secret_count[1h]))

# 速率限制的 NAS 设备
sum by (nas_ip) (increase(radius_access_rate_limited_count[15m]))
```

RADIUS 计费指标

跟踪每个 RFC 2866 的 RADIUS 计费消息。

radius_accounting_request_count

类型：计数器

标签	描述
`status_type`	计费状��类型（见下文）。

状态类型值：`start`，`stop`，`interim_update`，`accounting_on`，`accounting_off`。

示例查询：

```
# 每分钟会话开始
rate(radius_accounting_request_count{status_type="start"}[5m]) *
60

# 每分钟会话停止
rate(radius_accounting_request_count{status_type="stop"}[5m]) * 60
```

EAP 指标

这些计数器跟踪每个 RFC 4187 的 EAP-AKA 阶段，以及 EAP-SIM、EAP 重新认证（ERP）和通用 EAP 失败。

指标	类型	描述
`eap_aka_identity_count`	计数器	EAP-AKA 身份交换（阶段 1）。
`eap_aka_challenge_count`	计数器	EAP-AKA 挑战交换（阶段 2）。
`eap_aka_sync_failure_count`	计数器	EAP-AKA 同步失败（SQN 不同步，触发重新同步）。
`eap_aka_auth_success_count`	计数器	成功的 EAP-AKA 认证。
`eap_aka_auth_reject_count`	计数器	被拒绝的 EAP-AKA 认证（UE 未能认证网络 / AUTN 失败）。
`eap_aka_client_error_count`	计数器	从 UE 接收到的 EAP-AKA 客户端错误通知。
`eap_auth_rejected_count`	计数器	预期的认证拒绝（例如 HSS USER_UNKNOWN），干净地拒绝 UE。
`eap_process_crash_count`	计数器	处理 EAP 消息���发生的意外错误（UE 被拒绝，会话被保留）。
`eap_sim_start_count`	计数器	EAP-SIM 启动交换。
`eap_sim_auth_success_count`	计数器	成功的 EAP-SIM 认证。
`eap_sim_auth_reject_count`	计数器	被拒绝的 EAP-SIM 认证。
`eap_erp_initiate_count`	计数器	EAP ERP 重新认证的发起。

指标	类型	描述
`eap_erp_success_count`	计数器	成功的 EAP ERP 重新认证。
`eap_erp_mac_failure_count`	计数器	EAP ERP MAC 验证失败。
`eap_erp_fallback_count`	计数器	EAP ERP 回退到完全认证。标签 `reason`。

示例查询：

```
# 同步失败（应该很少 - 如果 > 0 则警报）
increase(eap_aka_sync_failure_count[1h])

# 每分钟成功的 AKA 认证
rate(eap_aka_auth_success_count[5m]) * 60

# EAP 处理崩溃（应该保持平稳）
increase(eap_process_crash_count[1h])
```

客户端、AP 和会话仪表

这些仪表每 5 秒由轮询器采样。

指标	类型	标签	描述
`radius_active_clients_count`	仪表	无	当前认证的客户端（活动的 `Twag.Session.Client` 进程）。
`radius_access_points_count`	仪表	无	注册的接入点（`APState` 条目）。
`radius_clients_per_ap_count`	仪表	`nas_ip`, `ssid`	每个接入点连接的客户端。
`radius_active_sessions_count`	仪表	`status`	按计费状态分组的会话（`start`, `stop`, `interim_update`）。
`radius_active_sessions_total`	仪表	无	状态为 `start` 或 `interim_update` 的会话。

示例查询：

```
# 前 5 个最繁忙的 AP
topk(5, radius_clients_per_ap_count)

# 按 SSID 的客户端
sum by (ssid) (radius_clients_per_ap_count)

# 对活动会话的突然下降发出警报
delta(radius_active_sessions_total[5m]) < -10
```

数据吞吐量指标

这些仪表是从 RADIUS 计费计数器中汇总的所有跟踪会话。

指标	类型	单位	描述
`radius_data_bytes_in`	仪表	字节	跨会话接收到的总字节（Acct-Input-Octets）。
`radius_data_bytes_out`	仪表	字节	跨会话发送的总字节（Acct-Output-Octets）。
`radius_session_total_time`	仪表	秒	跨会话的总会话时间（Acct-Session-Time）。

示例查询：

```
# 总吞吐量（MB）
(radius_data_bytes_in + radius_data_bytes_out) / 1024 / 1024

# 平均会话持续时间
radius_session_total_time / radius_active_sessions_total
```

TWAG 会话和数据路径指标

这些计数器跟踪由 `Twag.Session.Client` 发出的每个会话生命周期事件。

□□	□□	□□	□□
`twag_session_teardown_count`	□□□□	`reason`	□□□□□□□□□ `accounting_stop` `admin_disconnec` `network_disconn` `identity_timeou` `liveness_timeou`
`twag_bearer_modify_command_count`	□□□□	□	□□□□□□ PGW Modi Bearer-Command
`twag_bearer_update_count`	□□□□	□	□□□□□□ PGW Upda Bearer □□□
`twag_detach_cleanup_start_count`	□□□□	□	□□□□□□□
`twag_detach_cleanup_complete_count`	□□□□	□	□□□□□□□
`twag_detach_pgw_delete_count`	□□□□	□	□□□□□ PGW Delete Session □□□
`twag_detach_liveness_timeout_count`	□□□□	□	□□□□□□□□□□□□□
`twag_liveness_arp_probe_success_count`	□□□□	□	□□□□□ ARP □□□

指标	类型	标签	描述
`twag_liveness_arp_probe_failure_count`	计数器	无	失败的存活 ARP 探测。
`twag_handover_in_count`	计数器	`source`	Handover-in 事件。
`twag_handover_out_count`	计数器	`cause`	Handover-out 事件。
`twag_handover_ap_change_count`	计数器	无	会话内 AP 更改事件。

示例查询：

```
# 最近一小时的拆除原因
sum by (reason) (increase(twag_session_teardown_count[1h]))

# 因未完成认证而被收回的会话
increase(twag_session_teardown_count{reason="identity_timeout"}
[1h])

# 因 UE 沉默而被丢弃的会话
increase(twag_detach_liveness_timeout_count[1h])
```

> 注意：Gy 配额事件 `twag_gy_quota_blocked` 和 `twag_gy_quota_unblocked` 是由 `Twag.Session.Client` 发出的，但目前未声明为 Prometheus 指标，因此不会出现在 `/metrics` 端点上。它们在结构化日志中可见。



Gy 在线计费指标

这些计数器跟踪 Diameter Ro/Gy 计费控制，仅用于 NSWO 本地突破。请参见 在线计费。

指标	类型	标签	描述
`twag_gy_ccr_sent_count`	计数器	`type`	发送到 OCS 的计费控制请求。`type` 为 `initial`、`update` 或 `terminate`。
`twag_gy_ccr_error_count`	计数器	`type`	CCR 发送或构建失败。
`twag_gy_cca_received_count`	计数器	`type`、`result`	从 OCS 接收到的计费控制应答。

示例查询：

```
# 按类型的 CCR 发送速率
sum by (type) (rate(twag_gy_ccr_sent_count[5m]))

# CCR 失败
sum by (type) (increase(twag_gy_ccr_error_count[1h]))

# CCA 结果
sum by (result) (rate(twag_gy_cca_received_count[5m]))
```

STa Diameter 指标

这些计数跟踪 TWAG 处理的 STa 入站请求。

□□	□□	□□	□□
`sta_asr_received_count`	□□□□	□	□□□□ Abort-Session-Requests□
`sta_rar_received_count`	□□□□	□	□□□□ Re-Auth-Requests□
`sta_str_received_count`	□□□□	□	□□□□ Session-Termination-Requests□
`sta_session_disconnect_count`	□□□□	`reason`	STa □□□□□
`sta_session_reauth_count`	□□□□	`re_auth_type`	STa □□□□□□□

Diameter □□□□

diameter_message_count

□□□ □□□

标签	描述
`application`	Diameter 应用程序（例如 `swx`, `sta`）。
`command`	命令名称（例如 `MAR`, `MAA`）。
`direction`	`inbound` 或 `outbound`。

示例查询：

```
# 按应用程序的 Diameter 消息速率
sum by (application) (rate(diameter_message_count[5m]))
```

NSWO 指标

指标	类型	描述
`nswo_allocate_count`	计数器	NSWO 本地 IP 分配。
`nswo_release_count`	计数器	NSWO 本地 IP 释放。

DHCP 指标

这些计数器跟踪 DHCPv4 服务器。`discover.pool_allocated` 事件涵盖 NSWO 分配路径，其中服务器从绑定池中分配一个新地址。

指标	类型	描述
`dhcp_lease_allocate_count`	计数器	租约分配。
`dhcp_lease_deallocate_count`	计数器	租约取消分配。
`dhcp_discover_count`	计数器	从预分配租约提供的 DISCOVER 数据包。
`dhcp_discover_no_allocation_count`	计数器	没有租约且没有绑定池的 DISCOVER（没有地址可提供）。
`dhcp_discover_pool_allocated_count`	计数器	通过从绑定池中分配新地址（NSWO）提供的 DISCOVER。
`dhcp_ack_count`	计数器	发送的 ACK 数据包。
`dhcp_nak_count`	计数器	发送的 NAK 数据包。
`dhcp_release_count`	计数器	接收到的 RELEASE 数据包。
`dhcp_decline_count`	计数器	接收到的 DECLINE 数据包。
`dhcp_inform_count`	计数器	接收到的 INFORM 数据包。
`dhcp_packet_malformed_count`	计数器	格式错误的 DHCP 数据包。

示例查询：

```
# 池分配速率（NSWO 地址分配）
rate(dhcp_discover_pool_allocated_count[5m])

# 无法□供的 DISCOVER
increase(dhcp_discover_no_allocation_count[15m])
```

> 注意：DHCP 服务器在收回过期租约时还会发出 `dhcp_lease_reap`。该事件目前未声明为 Prometheus 指标，因此不会出现在 `/metrics` 端点上。

DHCPv6 和 IPv6 RA 指标

指标	类型	描述
`dhcpv6_info_request_received_count`	计数器	接收到的 DHCPv6 信息请求数据包。
`dhcpv6_solicit_received_count`	计数器	接收到的 DHCPv6 请求数据包。
`dhcpv6_request_received_count`	计数器	接收到的 DHCPv6 请求数据包。
`dhcpv6_info_reply_sent_count`	计数器	发送的 DHCPv6 信息回复数据包。
`dhcpv6_advertise_sent_count`	计数器	发送的 DHCPv6 广告数据包。
`dhcpv6_reply_sent_count`	计数器	发送的 DHCPv6 回复数据包。
`ipv6_ra_prefix_add_count`	计数器	IPv6 RA 前缀添加。
`ipv6_ra_prefix_remove_count`	计数器	IPv6 RA 前缀移除。
`ipv6_ra_advertisement_sent_count`	计数器	发送的 IPv6 路由器广告。

WLCP 指标

这些计数器跟踪用于多连接模式的 WLAN 控制协议。

指标	类型	单位	描述
`wlcp_packet_received_count`	计数器	无	接收到的 WLCP 数据包。
`wlcp_packet_received_bytes`	总和	字节	接收到的 WLCP 数据包的总字节数。
`wlcp_packet_malformed_count`	计数器	无	格式错误的 WLCP 数据包。
`wlcp_packet_unregistered_count`	计数器	无	来自未注册客户端的 WLCP 数据包。
`wlcp_pdn_request_count`	计数器	无	PDN 连接请求。
`wlcp_pdn_accept_count`	计数器	无	PDN 连接接受。
`wlcp_pdn_reject_count`	计数器	无	PDN 连接拒绝。
`wlcp_pdn_disconnect_request_count`	计数器	无	PDN 断开请求。
`wlcp_pdn_disconnect_accept_count`	计数器	无	PDN 断开接受。

Erlang VM 指标

这些指标提供对 BEAM 运行时的可见性。

内存

指标	描述
`vm_memory_total`	总分配内存（字节）。
`vm_memory_processes`	Erlang 进程使用的内存。
`vm_memory_processes_used`	进程实际使用的内存。
`vm_memory_system`	运行时使用的内存。
`vm_memory_atom`	原子使用的内存。
`vm_memory_atom_used`	原子实际使用的内存。
`vm_memory_binary`	二进制使用的内存。
`vm_memory_code`	加载代码使用的内存。
`vm_memory_ets`	ETS 表（会话和 AP 状态）使用的内存。

系统

指标	描述
`vm_system_info_process_count`	当前 Erlang 进程数量。
`vm_system_info_port_count`	当前端口数量。
`vm_system_info_atom_count`	当前原子数量。
`vm_system_info_schedulers`	��调度程序线程。
`vm_system_info_schedulers_online`	在线调度程序线程。

调度程序

指标	描述
`vm_statistics_run_queue`	总运行队列长度。
`vm_total_run_queue_lengths_total`	总调度程序运行队列。
`vm_total_run_queue_lengths_cpu`	CPU 调度程序运行队列。
`vm_total_run_queue_lengths_io`	IO 调度程序运行队列。

示例查询：

```
# 总内存（MB）
vm_memory_total / 1024 / 1024

# 运行队列饱和度（应该较低）
avg_over_time(vm_statistics_run_queue[5m])
```

示例 **Grafana** 查询

认证概述

```
# 每秒接受数
sum(rate(radius_access_accept_count[5m]))

# 每秒拒绝数
sum(rate(radius_access_reject_count[5m]))

# 成功比率百分比
100 * sum(rate(radius_access_accept_count[5m])) /
  (sum(rate(radius_access_accept_count[5m])) +
sum(rate(radius_access_reject_count[5m])))
```

每 AP 统计

```
# 按 AP 的请求（Access-Request 没有 nas_ip 标签；请使用计费代替）
sum by (nas_ip) (rate(radius_accounting_request_count[5m]))

# 按 AP 的客户端
radius_clients_per_ap_count
```

拒绝和速率限制分析

```
# 按原因类别的认证拒绝
sum(increase(radius_access_reject_count[1h]))
sum(increase(radius_auth_unauthorized_source_count[1h]))
sum(increase(radius_auth_bad_secret_count[1h]))

# 拆除原因
sum by (reason) (increase(twag_session_teardown_count[1h]))
```

数据使用

```
# 总吞吐量（MB）
(radius_data_bytes_in + radius_data_bytes_out) / 1024 / 1024
```

警报规则

关键警报

```
groups:
  - name: twag_critical
    rules:
      - alert: HighAuthenticationFailureRate
        expr: >
          sum(rate(radius_access_reject_count[5m])) /
          (sum(rate(radius_access_accept_count[5m])) +
sum(rate(radius_access_reject_count[5m]))) > 0.1
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "高认证失败率 (> 10%)"

      - alert: EAPSyncFailures
        expr: increase(eap_aka_sync_failure_count[5m]) > 0
        for: 1m
        labels:
          severity: warning
        annotations:
          summary: "检测到 EAP-AKA 同步失败"

      - alert: UnknownRadiusSources
        expr: increase(radius_auth_unauthorized_source_count[15m])
> 0
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "来自没有匹配凭据的源的 RADIUS 数据包"

      - alert: BadRadiusSecret
        expr: increase(radius_auth_bad_secret_count[15m]) > 0
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "因错误的共享密钥而拒绝的 RADIUS 数据包"
```

```
      - alert: NoActiveClients
        expr: radius_active_clients_count == 0
        for: 15m
        labels:
          severity: warning
        annotations:
          summary: "15 分钟内没有活动客户端"
```

资源警报

```
groups:
  - name: twag_resources
    rules:
      - alert: HighMemoryUsage
        expr: vm_memory_total > 1073741824
        for: 10m
        labels:
          severity: warning
        annotations:
          summary: "TWAG 内存使用超过 1GB"

      - alert: HighRunQueue
        expr: avg_over_time(vm_statistics_run_queue[5m]) > 10
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "高调度程序运行队列指示 CPU 饱和"
```

相关页面

- 架构
- RADIUS 拒绝
- RADIUS 共享密钥
- 会话
- 在线计费

- □ AP □□

□□

在线计费（Gy）

□述

OmniTWAG 可以实时对在线计费系统（OCS）进行数据会话计费。OmniTWAG 使用 Diameter Ro (Gy) 接口发送信用控制请求（CCR）并接收信用控制应答（CCA）。这使得运营商能够对 TWAG 提供的 WLAN 流量应用预付费信用控制。

在线计费适用于 TWAG 拥有数据路径的会话，即 **NSWO**（本地突破）和 TWAG 锚定流量。在这种情况下，TWAG 是流量退出互联网的点，因此 TWAG 充当计费执行点。在 家庭路由（**GTP**）模式 下，PGW 执行计费，因此运营商通常将全局计费模式设置为 `:disabled`。有关数据平面模式的更多信息，请参见 用户平面 参考。

两个控制决定了 Gy 是否在给定的 UE 上运行：

- 全局计费模式（`online_charging.mode`）。这决定了 OmniTWAG 可以发送哪些信用控制请求。
- 每个 **AP** 的计费模式 在 AP 凭证上（`charging_mode`）。这决定了 OmniTWAG 是否为该 UE 打开 Gy 会话。请参见 RADIUS 凭证。

两个控制都必须允许 CCR-Initial 请求发送。请参见 每个 AP ���计费模式。

目录

- □念
- 计费模式
- 每个 AP 的计费模式
- 信用控制流程
- 配额、使用和重新授权
- 配额执行
- 计费组和服务单元
- 配置
- 操作行为和故障处理
- 可观察性
- 参考表

概念

Gy 接口是一个 Diameter 应用程序，用于在网络元素和 OCS 之间传递信用控制消息。OmniTWAG 使用 3GPP Ro 字典，Auth-Application-Id 为 `4`，如 RFC 4006 和 3GPP TS 32.299 中定义。OmniTWAG 使用 Service-Context-Id `32251@3gpp.org`，这是 3GPP PS 计费上下文。

OmniTWAG 在会话期间发送三种类型的信用控制请求：

- **CCR-Initial (CCR-I)**：在会话开始时。OmniTWAG 请求 OCS 开始对订阅者的信用控制。
- **CCR-Update (CCR-U)**：在会话期间。OmniTWAG 报告已使用的数据并请求更多配额。
- **CCR-Termination (CCR-T)**：在会话结束时。OmniTWAG 报告最终使用的数据并关闭信用控制会��。

OCS 对每个请求以信用控制应答 (CCA) 进行回复。CCA 包含一个 `Result-Code`、一个可选的配额授予 (`Granted-Service-Unit`) 和一个可选的 `Validity-Time`。

OmniTWAG 为每个订阅者会话保持一个单独的 Gy 状态。Gy 状态保存 Diameter `Session-Id`、当前 `CC-Request-Number`、累积使用计数器和 OCS 授予的最后配额。OmniTWAG 在发送 CCR-I 时创建此状态，并在会话结束时丢弃该状态。

计费模式

`online_charging.mode` 参数是全局计费模式。它选择三种模式之一，并控制 OmniTWAG 可以发送哪些信用控制请求。每个 AP 的 `charging_mode`（请参见每个 AP 的计费模式）然后决定对于每个 UE，OmniTWAG 是否打开 Gy 会话。

模式	会话开始时 **CCR-I**	会话期间 **CCR-U**	会话结束时 **CCR-T**
`:disabled`	否	否	否
`:authorize_only`	是	否	是
`:online_charging`	是	是	是

`:disabled`

OmniTWAG 不联系 OCS。这是默认设置。对于家庭路由（GTP）操作，请使用此模式，其中 PGW 执行计费。

`:authorize_only`

OmniTWAG 在会话开始时发送 CCR-I，并在会话结束时发送 CCR-T。OmniTWAG 不发送临时 CCR-U 消息。

��用此模式在开始时检查订阅者余额，并在结束时报告总使用量，而不进行实时配额管理。

`:online_charging`

OmniTWAG 在开始时发送 CCR-I，在每次计费临时更新时发送 CCR-U，并在结束时发送 CCR-T。使用此模式进行全面的配额管理，在此期间 OCS 授予和刷新配额。

每个 AP 的计费模式

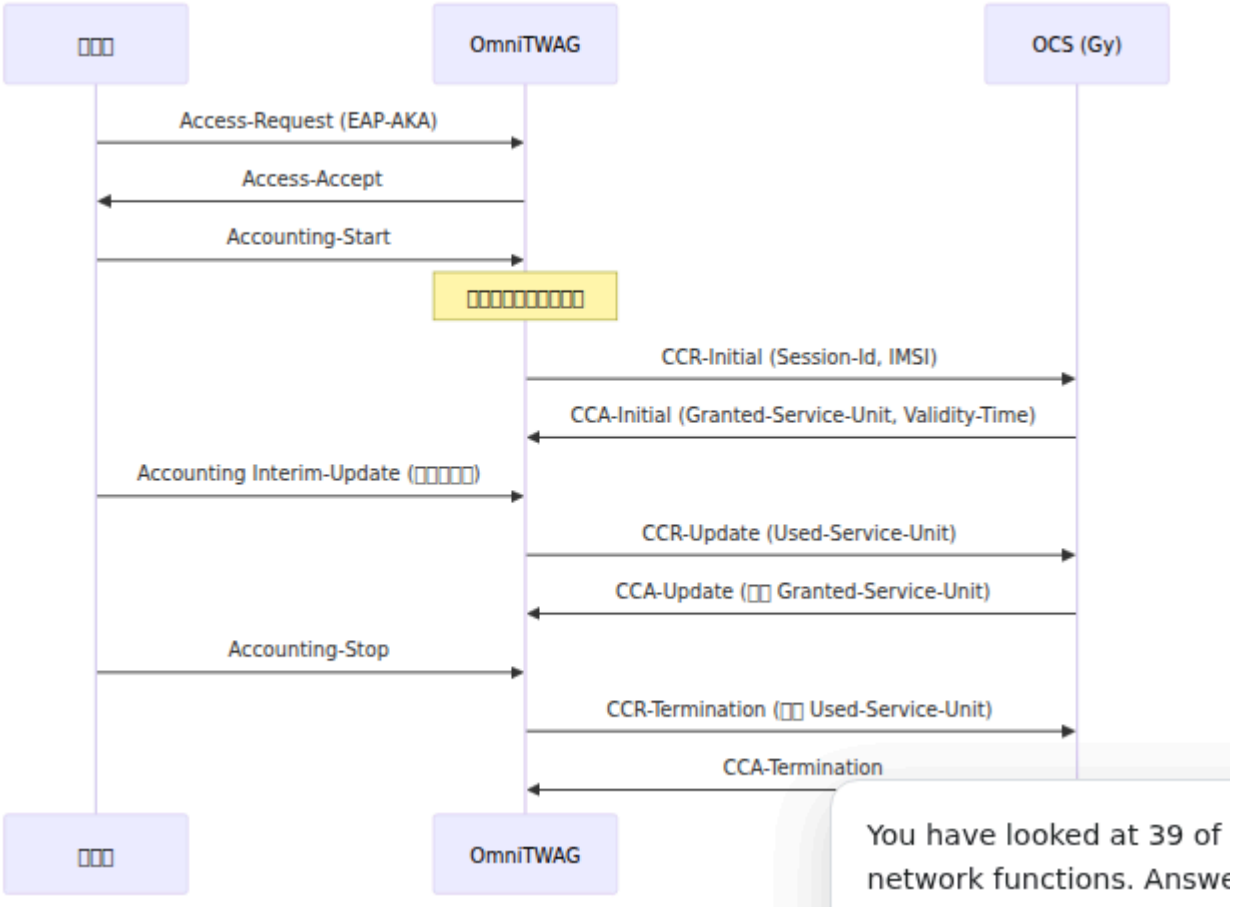
每个 AP 凭证携带一个 `charging_mode` 字段。这与全局 `online_charging.mode` 是分开的。OmniTWAG 从会话的 RADIUS 源解析凭证，因此每个接入点可以有自己的行为。请参见 RADIUS 凭证。

每个 **AP** 的 `charging_mode`	含义	对 **Gy** 的影响
`:charged`	对该 AP 的流量进行计费。这是默认设置。	当全局模式也允许时，OmniTWAG 为 UE 打开 Gy 会话 (CCR-Initial)。
`:uncharged`	服务该 AP，但不对其计费。	OmniTWAG 建立数据会话，但不打开 Gy 会话。没有发送 CCR。
`:auth_only`	仅进行身份验证，不建立数据会话。	OmniTWAG 进行订阅者身份验证并停止。它不建立数据会话，也不打开 Gy 会话。

只有在满足 所有 这些条件时，OmniTWAG 才会发送 CCR-Initial:

- 全局 `online_charging.mode` 不是 `:disabled`（Gy 客户��已启用）。
- 会话具有 IMSI。
- AP 凭证 `charging_mode` 为 `:charged`。

如果任何条件失败，OmniTWAG 将不为 UE 打开 Gy 会话，并且不会发送后续的 CCR-U 或 CCR-T。因此，每个 AP 的模式独立于全局模式控制 Gy：即使全局模式设置为 `:online_charging`，标记为 `:uncharged` 的 AP 也不会产生 Gy 流量。



信用控制流程

信用控制会话跟随订阅者会话。OmniTWAG 从 RADIUS 会话生命周期中的特定点触发每个信用控制请求。

会话开始 **(CCR-Initial)**

OmniTWAG 在建立会话后发送 CCR-I。会话建立在接受后步骤中运行，RADIUS 服务器在 Access-Accept 和 Accounting-Start 之后触发。只有在全局计费模式启用、会话具有 IMSI，并且 AP 凭证 `charging_mode` 为 `:charged` 时，OmniTWAG 才会发送 CCR-I。请参见 每个 AP 的计费模式。

CCR-I 在 `Subscription-ID` AVP 中携带订阅者 IMSI，类型为 `1`（END_USER_IMSI）。CCR-I 将 `Multiple-Services-Indicator` 设置为 `1`，并包含一个具有空 `Requested-Service-Unit` 和配置的 `Rating-Group` 的 `Multiple-Services-Credit-Control` (MSCC) 组。

临时更新 **(CCR-Update)**

在 `:online_charging` 模式下，每个 RADIUS 计费临时更新将报告的字节和会话时间折叠到 Gy 使用计数器中。OmniTWAG 不会在每个临时更新时发送 CCR-U。它仅在以下触发条件之一触发时发送 CCR-U：

- 使用的流量超过授予流量的配置比例（`quota.update_threshold`，默认值为 `0.8`）。
- 授予的配额的 `Validity-Time` 已过期。

未触发条件的临时更新仅在本地记录使用情况，因此 OCS 不会在每个临时更新时进行咨询。当 CCR-U 发送时，OmniTWAG 从计费属性中读取使用的字节和会话时间，并在 `Used-Service-Unit` AVP 中报告它们。CCR-U 还包括一个空的 `Requested-Service-Unit`，以便 OCS 可以授予更多配额。

当配额有效性计时器[illegible]发时，也会在计费路径上发送 CCR-U。请参见 配额执行。

OmniTWAG 为 CCR-U 读取以下 RADIUS 计费属性：

RADIUS 属性	**Gy** 字段
`Acct-Input-Octets`	`CC-Input-Octets`
`Acct-Output-Octets`	`CC-Output-Octets`
`Acct-Session-Time`	`CC-Time`

OmniTWAG 计算 `CC-Total-Octets` 为输入和输出字节的总和。

会话结束 **(CCR-Termination)**

OmniTWAG 在会话拆除时发送 CCR-T，例如在 RADIUS Accounting-Stop 时。OmniTWAG 在 `Used-Service-Unit` AVP 中报告最终的累积使用情况，并将 `Termination-Cause` 设置为 `1`（DIAMETER_LOGOUT）。OmniTWAG 在 `:authorize_only` 模式和 `:online_charging` 模式下都发送 CCR-T。

配额、使用和重新授权

授予的配额

每个 CCA 可以在第一个 MSCC 组中携带一个 `Granted-Service-Unit`。OmniTWAG 在 Gy 状态中存储授予的值：

- `CC-Total-Octets`、`CC-Input-Octets`、`CC-Output-Octets`：授予的数据量。

- `CC-Time`：授予的时间。
- `Validity-Time`：授予保持有效的秒数。OmniTWAG 将其转换为绝对截止日期，并对其进行计时。

OmniTWAG 还从每个 CCA 中读取 MSCC 的 `Final-Unit-Indication` 和 MSCC 级别的 `Result-Code`。`Final-Unit-Action` 为 TERMINATE、REDIRECT 或 RESTRICT_ACCESS，或 MSCC 的 `Result-Code` 为 `4010`（END_USER_SERVICE_DENIED）或 `4012`（CREDIT_LIMIT_REACHED），将作为信用耗尽信号存储。请参见 配额执行。

OmniTWAG 在每个成功的 CCA-I 和 CCA-U 上更新存储的授予。

使用报告

OmniTWAG 在 Gy 状态中保持累积使用计数器。RADIUS 计费计数器是会话的累积值，因此每个临时更新报告到目前为止的总数。OmniTWAG 记录报告的计数器并将其作为 `Used-Service-Unit` 发送。CCR-U 将 `Reporting-Reason` 设置为 `4`（VALIDITY_TIME）。CCR-T 将 `Reporting-Reason` 设置为 `2`（FINAL）。

重新授权

OmniTWAG 在两个路径上重新授权：

- 基于流量。超过配额阈值或发现有效性过期的 RADIUS 计费临时更新会触发 CCR-U。临时间隔是接入点的一个属性，因此较短的临时间隔可以提供更精细的计量。
- 基于时间。 OmniTWAG 从授予的 `Validity-Time` 启动配额有效性计时器。如果计时器在几乎没有流量的情况下触发，OmniTWAG 仍然会发送 CCR-U 以重新授权。如果 OCS 不发送 `Validity-Time`，OmniTWAG 可以退回到配置的 `quota.validity_time`。如果两者都不存在，则不会启动有效性计时器。

在每个成功的 CCA 之后，`CC-Request-Number` 增加 1。此值���信用控制会话中的每个请求提供唯一的序列号，如 RFC 4006 所要求。

配额执行

OmniTWAG 在 `:online_charging` 模式下执行信用耗尽。TWAG 没有单独的用户平面功能，因此其执行杠杆是 GRE 数据路径：它将 UE 从数据路径中注销，从而中断该 UE 的上行和下行流量。

当 Gy 状态显示信用耗尽时，CCR-U 响应会触发执行，即当 OCS：

- 发出任何 `Final-Unit-Action`（TERMINATE、REDIRECT 或 RESTRICT_ACCESS），或
- 返回 MSCC 的 `Result-Code` 为 `4010` 或 `4012`。

由于 TWAG 无法在其数据路径上执行 REDIRECT 或 RESTRICT_ACCESS，因此任何 `Final-Unit-Action` 都被视为停止。当执行触发时，OmniTWAG 阻止 UE 的数据路径，但保持会话、IP 租约和计费处于活动状态。稍后的重新授权授予可用配额会恢复数据路径。

CCA 授予配额
Serving
CCA 授予更多配额（如果被阻止则解除阻止）
达到阈值 / 有效性过期
Reauth
后续 CCA 重新授予配额
CCR-T 在拆除时
使用低于阈值且有效性新鲜
Final-Unit-Action / 信用被拒绝 (4010, 4012)
Blocked
CCR-T 在拆除时

执行是尽力而为，并在传输错误时失败开放：在 Diameter 层失败的 CCR-U 从不阻止 UE。阻止仅在 OCS 显式拒绝信用或在 CCA MSCC 中发出耗尽信号时发生。请参见 操作行为和故障处理。

计费组和服务单元

OmniTWAG 为整个会话使用一个计费组。`rating_group` 参数设置 CCR-I 中 MSCC 的 `Rating-Group` 值。默认值为 `1`。OmniTWAG 不会将流量拆分为多个计费组或服务标识符。OmniTWAG 请求一个带有空 `Requested-Service-Unit` 的组合量授予，因此 OCS 决定授予。

配置

在 `:omnitwag` 应用程序下设置在线计费参数，在 `config/runtime.exs` 的 `online_charging` 块中。

```
config :omnitwag,
  online_charging: %{
    # 全局计费模式：:disabled、:authorize_only 或 :online_charging
    mode: :disabled,

    # OCS Diameter 领域（CCR 的 Destination-Realm）
    ocs_realm: "epc.mnc001.mcc001.3gppnetwork.org",

    # 多服务信用控制组的计费组
    rating_group: 1,

    # 配额管理（在 :online_charging 模式下使用）
    quota: %{
      # 达到授予量的比例时触发 CCR-Update
      update_threshold: 0.8,

      # 当 OCS 不发送 Validity-Time 时的回退有效性时间（秒）
      validity_time: 3600
    }
  }
```

参数

参数	类型	必需	默认	描述
`mode`	原子	否	`:disabled`	全局计费模式。可以是 `:disabled`、`:authorize_on` 或 `:online_charging`。请参见 [式。
`ocs_realm`	字符串	否	Diameter 领域	OCS 的 Diameter 领域。OmniTWAG 在每个 CCR 中将其用作 `Destination-Realm`。如果未设置 OmniTWAG 将使用本地 Diamete 域。
`rating_group`	整数	否	`1`	OmniTWAG 在 CCR-Initial 的 MSCC 中放置的 `Rating-Group` [
`quota.update_threshold`	浮点数	否	`0.8`	达到授予量的比例（0.0 到 1.0），在此时触[时更新 CCR-Update。
`quota.validity_time`	整数	否	（未设置）	回退配额有效性时间（秒），用于在 OCS 不发 `Validity-Time` 时启动有效性计时器[果未设置，则不启动回退计时器。

每个 AP 的 `charging_mode` 不在此块中。它存在于每个 AP 凭证上。请参见 RADIUS 凭证。

Gy 会话的 Diameter `Origin-Host` 和 `Origin-Realm` 来自 `:diameter_ex` 配置中的 Diameter 身份（`host` 和 `realm`）。`Destination-Host` 未配置。OmniTWAG 从第一个 CCA 的 `Origin-Host` 学习 `Destination-Host`，并将此 AVP 添加到后续的 CCR-U 和 CCR-T 消息中。

Diameter 传输和 Ro 应用程序

Gy 客户端通过共享的 Diameter 栈发送。将 OCS 配置为 Diameter 对等体，并在 `:diameter_ex` 配置中启用 `:ro` 应用程序。

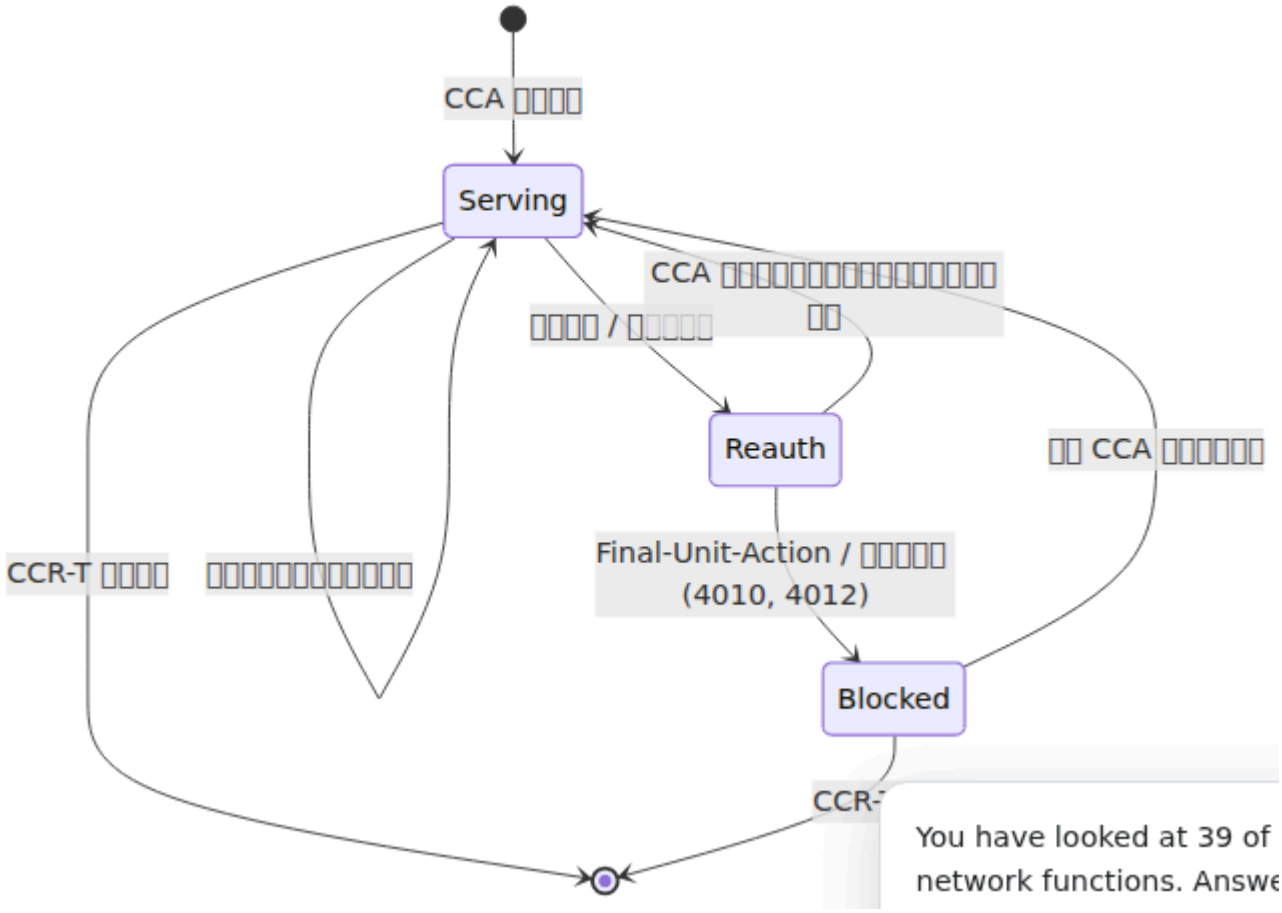
```
config :diameter_ex,
  diameter: %{
    service_name: :omnitouch_twag,
    host: "amanaki",
    realm: "epc.mnc001.mcc001.3gppnetwork.org",
    request_timeout: 5000,
    applications: [
      %{
        application_name: :ro,
        application_dictionary: :diameter_gen_3gpp_ro,
        vendor_specific_application_ids: [
          %{vendor_id: 0, auth_application_id: 4,
acct_application_id: nil}
        ]
      }
    ],
    peers: [
      %{
        host: "ocs.epc.mnc001.mcc001.3gppnetwork.org",
        realm: "epc.mnc001.mcc001.3gppnetwork.org",
        ip: "10.179.2.233",
        port: 3868,
        transport: :diameter_tcp,
        initiate_connection: true
      }
    ]
  }
```

参数	类型	必需	默认	描述
`service_name`	原子	是	-	本地 Diameter 服务的名称。OmniTWAG 通过此服务发送 G 请求。
`host`	字符串	是	-	本地 Diameter 身份（主机）。OmniTWAG 从 `host` 和 `realm` 构建 Gy 的 `Origin Host`。
`realm`	字符串	是	-	本地 Diameter 领域。OmniTWAG 将其用作 `Origin-Realm`，并在未设置 `ocs_realm` 时用作 `Destination-Realm`。
`request_timeout`	整数	否	5000	等待 CCA 的时间（毫秒），超时后请[败。
`application_name`	原子	是	-	Diameter 应用程序的名称。设置[`:ro` 用于 Gy。
`application_dictionary`	原子	是	-	应用程序的 Diameter 字典。设置 `:diameter_gen_3gpp_r` 用于 Gy。
`vendor_specific_application_ids`	列表	是	-	厂商特定的应用程序 ID。对于 Ro，使 `vendor_id: 0` 和 `auth_application_id: 4`。
`peers`	列表	是	-	Diameter 对等体的列表。在此处[OCS 添加为对等体。请参见 对等体参数。

对等体参数

参数	类型	必需	默认	描述
`host`	字符串	是	-	对等 Diameter 身份（FQDN）。必须与 OCS 身份匹配。
`realm`	字符串	是	-	对等 Diameter 领域。设置为 OCS 领域。
`ip`	字符串	是	-	OCS 的 TCP 连接 IP 地址。
`port`	整数	否	3868	OCS Diameter 端口。标准端口为 3868。
`transport`	原子	否	`:diameter_tcp`	传输协议。使用 `:diameter_tcp` 进行 TCP。
`initiate_connection`	布尔	否	false	当为 `true` 时，OmniTWAG 启动与对等体的连接。

> 注意：在随附的 `config/runtime.exs` 中，`online_charging` 块还显示 `ocs_host`、`service_identifier`、顶层 `request_timeout` 和 `quota` 键 `initial_request` 和 `subsequent_request`。当前的计费代码不读取这些键。它读取 `mode`、`ocs_realm`、`rating_group`、`quota.update_threshold` 和 `quota.validity_time`。Diameter 请求超时来自 `:diameter_ex` 的 `request_timeout`。OmniTWAG 请求配额时使用空的 `Requested-Service-Unit`，因此 `initial_request` 和 `subsequent_request` 目前不会影响 CCR。请在代码支持这些未读取的键之前，不要依赖它们。



操作行为和故障处理

失败开放行为

OmniTWAG 将 Gy 传输故障视为非致命。如果信用控制请求在 Diameter 层失败，订阅者会话将继续，OmniTWAG 不会阻止流量。

- **OCS** 无法访问或超时：CCR 在 Diameter 传输层失败。OmniTWAG 记录警告。对于 CCR-I，OmniTWAG 仍然保留 Gy 状态，因此可以稍后发送 CCR-T。对于 CCR-U，OmniTWAG 保留先前的 Gy 状态，并且不阻止 UE。对于 CCR-T，OmniTWAG 记录警告并完成拆除。
- **CCA** 错误结果代码：OCS 返回的命令级 `Result-Code` 不是 `2001`（DIAMETER_SUCCESS）。OmniTWAG 记录错误并将请求视为失败。OmniTWAG 保持会话处于活动状态。

执行与传输故障是分开的。在 `:online_charging` 模式下，OmniTWAG 确实会在 OCS 明确拒绝信用或在 CCA MSCC 中发出最终操作信号时阻止 UE 的数据路径。请参见 配额执行。它从不在传输故障时阻止，也从不在 `:authorize_only` 模式下拒绝访问。请考虑到这一点来规划 OCS 集成。

信用控制会话生命周期摘要

结果代码处理

OmniTWAG 在两个级别读取结果代码。

- 命令级 **`Result-Code`**。OmniTWAG 将 `2001` 视为成功。它还将缺失的 `Result-Code` 视为成功，因为某些对等响应以原始记录的形式到���，而没有解码的结果代码。任何其他命令级值都是错误：OmniTWAG 记录它，不重试，并保持会话处于活动状态。
- **MSCC** 级别 **`Result-Code`**。在第一个 MSCC 组中，`Result-Code` 为 `4010`（END_USER_SERVICE_DENIED）或 `4012`（CREDIT_LIMIT_REACHED）被视为信用拒绝信号。在 `:online_charging` 模式下，这驱动执行并阻止 UE 的数据路径。请参见 配额执行。

可观察性

OmniTWAG 为每个信用控制请求和应答发出遥测事件。这些事件携带 `type` 元数据值 `:initial`、`:update` 或 `:terminate`。

遥测事件	何时	元数据
`[:twag, :gy, :ccr, :sent]`	OmniTWAG 发送 CCR	`type`
`[:twag, :gy, :cca, :received]`	OmniTWAG 接收 CCA	`type`、`result`（`:success` 或 `:error`）
`[:twag, :gy, :ccr, :error]`	CCR 在传输时失败	`type`
`[:twag, :gy, :quota, :blocked]`	UE 数据路径因信用耗尽而被阻止	`identity`
`[:twag, :gy, :quota, :unblocked]`	UE 数据路径在重新授予后恢复	`identity`

> 注意：当前构建中，这些 Gy 遥测事件未注册为 Prometheus 指标。指标参考 尚未导出它们。要查看今天的 Gy 行为，请使用应用程序日志。Gy 客户端在 INFO 级别记录��个 CCR-I、CCR-U 和 CCR-T，以及在 WARN 或 ERROR 级别记录每个失败。日志行携带 Diameter `Session-Id`，并且对于 CCR-I，携带 IMSI。

参考表

CC-Request-Type 值

值	名称	消息
1	INITIAL_REQUEST	CCR-Initial
2	UPDATE_REQUEST	CCR-Update
3	TERMINATION_REQUEST	CCR-Termination

值在 RFC 4006 §8.3 中定义。

使用的 Reporting-Reason 值

值	名称	**OmniTWAG** 使用它的地方
2	FINAL	CCR-Termination
4	VALIDITY_TIME	CCR-Update

Result-Code

代码	级别	名称	**OmniTWAG** 行为	参考
2001	命令	DIAMETER_SUCCESS	视为成功。Gy 状态前进。	RFC 6733 §7.1.1
(缺失)	命令	-	视为成功，适用于原始记录响应。	-
任何其他	命令	(错误)	记录为错误。会话继续。	RFC 4006 §9
4010	MSCC	END_USER_SERVICE_DENIED	视为信用被拒绝。在 `:online_charging` 模式下执行（阻止 UE）。	RFC 4006 §9
4012	MSCC	CREDIT_LIMIT_REACHED	视为信用被拒绝。在 `:online_charging` 模式下执行（阻止 UE）。	RFC 4006 §9

Final-Unit-Action 值

MSCC `Final-Unit-Indication` 携带 `Final-Unit-Action`。OmniTWAG 没有用户平面功能，因此无法在数据路径上执行 REDIRECT 或 RESTRICT_ACCESS。它将任何 Final-Unit-Action 视为停止并阻止 UE。

值	名称	**OmniTWAG** 行为
0	TERMINATE	阻止 UE 数据路径（会话保留）。
1	REDIRECT	阻止 UE 数据路径（不支持重定向）。
2	RESTRICT_ACCESS	阻止 UE 数据路径（不支持限制访问）。

Final-Unit-Indication 在 3GPP TS 32.299 和 RFC 4006 §5.6 中定义。

Diameter 应用程序

ID	接口	字典	方向	参考
4	Ro / Gy	`diameter_gen_3gpp_ro`	TWAG 到 OCS	RFC 4006、3GPP TS 32.299

规范参考

主题	规范
Diameter 信用控制应用程序	RFC 4006
Diameter 计费应用程序 (Ro)	3GPP TS 32.299
计费架构	3GPP TS 32.240
TWAG / 非 3GPP 访问 (S2a, NSWO)	3GPP TS 23.402
Diameter 基础协议	RFC 6733

相关页面

- 概述
- RADIUS 凭证
- IP 池
- IP 地址管理
- 会话
- S2a GTP-C
- 用户平面

概述

每个接入点的计费

□述

OmniTWAG 为每个接入点聚合 RADIUS 计费数据。操作员可以看到每个接入点的总流量和会话计数。这些数据与每个会话的计费记录是分开的。OmniTWAG 通过接入点的 `NAS-IP-Address` 来键入每个接入点的总计。当接入点省略 `NAS-IP-Address`（例如某些 UBNT 型号）时，OmniTWAG 会回退到 RADIUS 数据包的 UDP 源 IP，因此计费附加到身份验证创建的相同接入点条目上。

目录

- □念
- 总计如何计算
- 记录的字段
- 活动客户端计数
- 数据查看位置
- 与每会话数据的关系
- 故障排除
- 相关页面

概念

接入点为每个客户端会话发送 RADIUS Accounting-Request 数据包。数据包具有一个 `Acct-Status-Type`，表示事件：

- 开始（`1`）：会话开始。
- 临时更新（`3`）：会话处于活动状态，当前计数器。
- 停止（`2`）：会话停止，最终计数器。

OmniTWAG 从每个数据包中读取八位字节和数据包计数器以及 `NAS-IP-Address`。OmniTWAG 将这些值添加到该接入点的运行总计中。这为操作员提供了每个接入点的流量和会话视图。

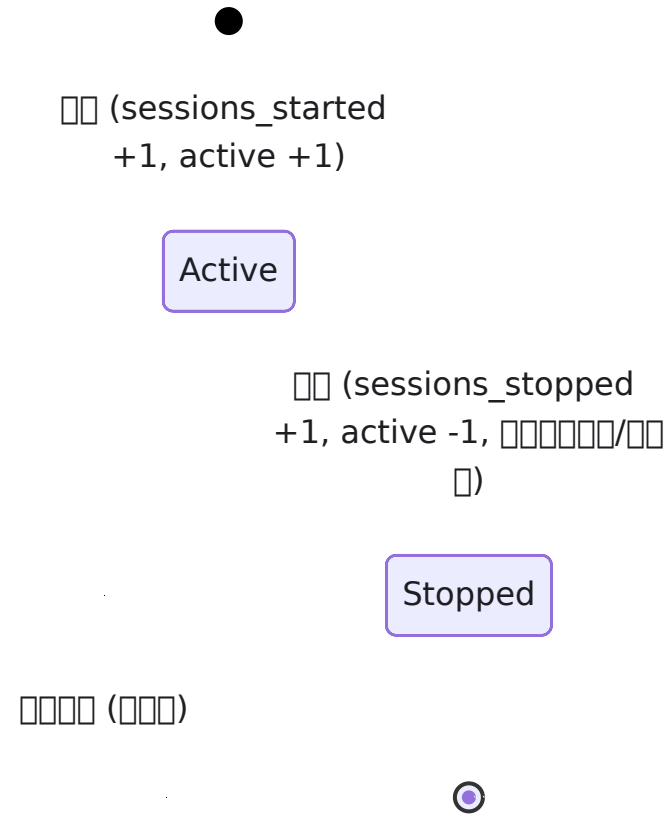
总计如何计算

RADIUS 八位字节和数据包计数器是 每个会话的累积。临时更新和停止都报告会话到目前为止的总计，而不是自上一个数据包以来的变化。如果 OmniTWAG 从每个数据包中添加计数器，它将多次计算相同的流量。

为了避免重复计数，OmniTWAG 仅在 停止 时将八位字节和数据包计数器添加到每个接入点的总计中。停止报告已完成会话的最终总计。规则如下：

- 在 开始 时：将 `acct_sessions_started` 增加 1。将 `acct_active_sessions` 增加 1。
- 在 停止 时：将 `acct_sessions_stopped` 增加 1。将 `acct_active_sessions` 减少 1，但 ���低于零。将最终的八位字节和数据包计数器添加到每个接入点的总计中。
- 在 临时更新 时：仅更新最后的计费时间。不要更改总计。

如果 OmniTWAG 收到一个它没有看到开始的会话的停止，活动会话计数不会低于零。OmniTWAG 仍然添加最终的八位字节和数据包。如果 OmniTWAG 收到一个它之前没有看到的接入点的 Accounting-Request，OmniTWAG 会创建每个接入点条目。这意味着仅用于计费的接入点仍然被跟踪。



开始 (sessions_started +1, active +1)
Active
停止 (sessions_stopped +1, active -1, 添加八位字节/数据包)
Stopped
临时更新（仅时间）

记录的字段

OmniTWAG 为每个接入点记录这些字段，除了接入点元数据，例如 SSID 和 `Called-Station-Id`。

字段	类型	描述
`acct_input_octets`	整数	停止会话的总输入八位字节。
`acct_output_octets`	整数	停止会话的总输出八位字节。
`acct_input_packets`	整数	停止会话的总输入数据包。
`acct_output_packets`	整数	停止会话的总输出数据包。
`acct_sessions_started`	整数	在此接入点上开始的���话数量。
`acct_sessions_stopped`	整数	在此接入点上停止的会话数量。
`acct_active_sessions`	整数	基于计费的开放会话计数：`started` 减去 `stopped`，永远不低于零。它仅反映 RADIUS 计费报告的内容，因此在接入点错过停止数据包（例如重启后）时可能会过度计数，并在停止到达或条目过期之前保持过时。这 不是 在接入点页面上显示的“活动客户端”数字 — 请参见 活动客户端计数。
`last_accounting_at`	整数	最后一个计费数据包的时间，自 Unix 纪元以来的毫秒数。

八位字节和数据包计数器的方向遵循 RADIUS 在 RFC 2866 中的定义。输入是来自客户端的流量进入网络。输出是来自网络到客户端的流量。

活动客户端计数

接入点页面上的 活动客户端 数字，以及接入点 API 上的 `client_count` 字段，报告当前有多少客户端 连接。这个值不是计费计数器 `acct_active_sessions`。它是从 OmniTWAG 为每个经过身份验证的客户端运行的实时内存会话进程中得出的，按接入

□□ `NAS-IP-Address` □□□

□□□

故障排除

总计没有变化

症状：一个接入点的八位字节和数据包总计保持为零，即使客户端正在使用网络。

可能的原因：

- 接入点没有发送停止数据包。OmniTWAG 仅在停止时添加八位字节和数据包。
- 接入点根本没有发送计费。

解决方案：

1. 确认接入点向 OmniTWAG 发送 Accounting-Request 数据包。
2. 确认接入点在会话结束时发送停止。
3. 确认接入点上的计费共享密钥与文件中的凭证匹配。请参见 RADIUS 共享密钥。

活动会话计数看起来过高

症状：`acct_active_sessions` 值高于实际活动客户端的数量。

可能的原因：

- 接入点发送了开始数据包，但没有发送匹配的停止数据包，例如在接入点重启后。

解决方案：

1. 确认接入点为每个会话发送停止。
2. 在接入点上设置临时更新间隔，以便 OmniTWAG 看到活动会话。
3. 随着新停止数据包的到达，活动计数会自行修正。计数不会低于零。

这仅影响 `acct_active_sessions`。接入点页面上的 活动客户端 数字（以及 `client_count` API 字段）不受影响，因为它是实时会话进程的总计，而不是计费计数器。请参见 活动客户端计数。

接入点未出现

症状：一个发送计费的接入点未出现在列表中。

可能的原因：

- OmniTWAG 丢弃了数据包，因为源 IP 与任何凭证子网不匹配。

解决方案：

1. 检查 `radius_auth_unauthorized_source_count` 指标。请参见 指标参考 和 RADIUS 拒绝。
2. 为接入点的子网添加凭证。请参见 RADIUS 共享密钥。

相关页面

- 述
- 会话
- RADIUS 共享密钥
- RADIUS 拒绝
- 指标参考
- 架构

述

OmniTWAG 性能基准

受信 WLAN 接入网关的控制平面身份验证和信令吞吐量。

测试环境

CPU	Intel(R) Core(TM) i5-5250U CPU @ 1.60GHz (2 cores)
Runtime	Elixir 1.17.0 / Erlang OTP 27 (JIT) (erts 15.0.1)
Measurement	1 s warmup + 3 s measure + 1 s memory per op
Commit	`main` @ `01ec4e8`

吞吐量以 **ips**（每秒迭代次数：越高越快）报告。**Avg** 和 **Median** 是每次调用的延迟（微秒）；中位数是典型稳态延迟的更好估计。**Mem/op** 是每次调用分配的堆内存。

完整身份验证（端到端）

顶线数字：每个场景链式处理完整的每次附加序列（EAP 身份解码、身份验证向量处理、EAP-AKA' 密钥层次、挑战构建、MAC、响应解码和 MAC 验证），因此其吞吐量（**ips**）是 每秒每核心的完整附加，即会话。□□□□ 在盒子上计算身份验证向量（Milenage）；*STa* □□ 从 HSS ��过 SWx/STa 接收它。

场景	吞吐量 **(ips)**	平均 **(µs)**	中位数 **(µs)**	**Mem/op**
完整 EAP-AKA' 附加，STa 模式（向量来自 HSS）	17,311	57.77	52.90	7,048 B
完整 EAP-AKA' 附加，本地模式（在盒子上计算向量）	12,493	80.05	73.31	10,560 B

加密

每次身份验证的加密操作：SIM 身份验证向量生成（Milenage）、EAP-AKA' 密钥层次（CK'/IK'、会话密钥、MAC）、ERP 快速重新身份验证密钥，以及通用的 3GPP 密钥派生函数。

操作	吞吐量 **(ips)**	平均 **(µs)**	中位数 **(µs)**	**Mem/op**
WLCP 密钥派生（EMSK -> WLCP 密钥）	262,975	3.80	3.54	72 B
EAP-AKA' 构建挑战数据包（编码）	257,991	3.88	0.87	456 B
EAP-AKA' 派生 CK'/IK'	225,191	4.44	3.93	328 B
EAP-AKA' 计算 MAC（HMAC-SHA-256/16）	223,503	4.47	4.11	72 B
3GPP 通用 KDF（HMAC-SHA-256）	198,596	5.04	4.32	248 B
ERP 派生 rMSK（每次重新身份验证）	113,269	8.83	8.01	376 B
ERP 完整快速重新身份验证（rRK -> rIK -> rMSK）	61,496	16.26	14.97	520 B
Milenage compute_all（RES、CK、IK、AK、MAC-A）	52,451	19.07	16.11	3,560 B
EAP-AKA' derive_keys（MK -> 所有会话密钥）	31,696	31.55	28.93	1,552 B
EAP-AKA' 完整身份验证加密（CK'/IK' + 所有密钥）	28,380	35.24	32.59	1,808 B

操作	功能
EAP-AKA' 构建挑战数据包	编码 EAP-Request/AKA'-Challenge（AT_RAND、AT_AUTN、AT_KDF、AT_MAC）。
WLCP 密钥派生	从 EMSK 派生 WLCP 保护密钥（多连接模式）。
3GPP 通用 KDF	在密钥层次中使用的 HMAC-SHA-256 密钥派生函数。
EAP-AKA' 计算 MAC	对 EAP 数据包进行 HMAC-SHA-256 AT_MAC，截断为 16 字节。
EAP-AKA' 派生 CK'/IK'	将 CK/IK 转换为绑定到接入网络名称的 CK'/IK'。
ERP 派生 rMSK	用于 EAP 重新身份验证的每次重新身份验证会话密钥（RFC 6696）。
ERP 完整快速重新身份验证	快速重新身份验证的完整 rRK → rIK → rMSK 派生链。
Milenage compute_all	一个 AKA 身份验证向量：RES、CK、IK、AK、MAC-A。
EAP-AKA' derive_keys	将主密钥扩展为 K_encr、K_aut、K_re、MSK、EMSK。
EAP-AKA' 完整身份验证加密	CK'/IK' 派生加上完整的会话密钥扩展。

数据包编解码器

信令平面编码/解码操作：EAP 数据包解析、WLCP（TS 24.244）多连模式消息处理，以及在每个 RADIUS Access-Accept 上构建的 MPPE 密钥属性。

操作	吞吐量 (ips)	平均 (µs)	中位数 (µs)	Mem/op
WLCP 解码（PDN 连接请求）	4,229,558	0.24	0.20	448 B
WLCP 编码（PDN 连接请求）	2,492,441	0.40	0.24	160 B
WLCP 编码 + 解码往返	1,989,407	0.50	0.40	608 B
EAP 解码身份数据包	336,816	2.97	2.66	1,136 B
MPPE 构建 VSA（Send-Key，RC4 + MD5）	135,026	7.41	5.32	1,296 B

操作	功能
WLCP 解码	从网络解析 WLCP PDN 连接请求。
WLCP 编码	将 WLCP PDN 连接请求序列化到网络。
WLCP 编码 + 解码往返	对一个 WLCP 消息进行编码然后解码。
EAP 解码身份数据包	解析 EAP 身份响应（NAI 提取）。
MPPE 构建 VSA	构建携带 MSK 的 MS-MPPE-Send-Key RADIUS VSA（RC4 + MD5）。

RADIUS 共享密钥（按源 IP 组）

概述

OmniTWAG 使用共享密钥对每个 RADIUS 客户端（接入点或 NAS）进行身份验证。此功能允许操作员管理 多个 共享密钥，每个密钥与一个或多个源 IP 子网相关联。共享同一子网的接入点组也共享一个密钥。操作员可以通过 REST API 或控制面板 Web 界面按需添加、修改和删除这些凭据。存储是持久的，因此凭据在重启后仍然有效。

目录

- 概念
- 身份验证如何使用凭据
- 凭据模型
- 按 AP 分流和计费
- REST API
- Web 界面
- 配置
- 从单一静态密钥迁移
- 指标
- 故障排除

概念

凭据 将一个共享密钥与一个或多个 源子网 关联。源子网是一个 CIDR 块。子网可以是 IPv4、IPv6 或两者的混合。

源 **IP** 是发送 RADIUS 数据包的接入点的 IP 地址。OmniTWAG 从接收到的 UDP 数据包中读取源 IP。OmniTWAG 不使用 `NAS-IP-Address` 属性进行密钥选择，因为攻击者可以将该属性设置为任何值。

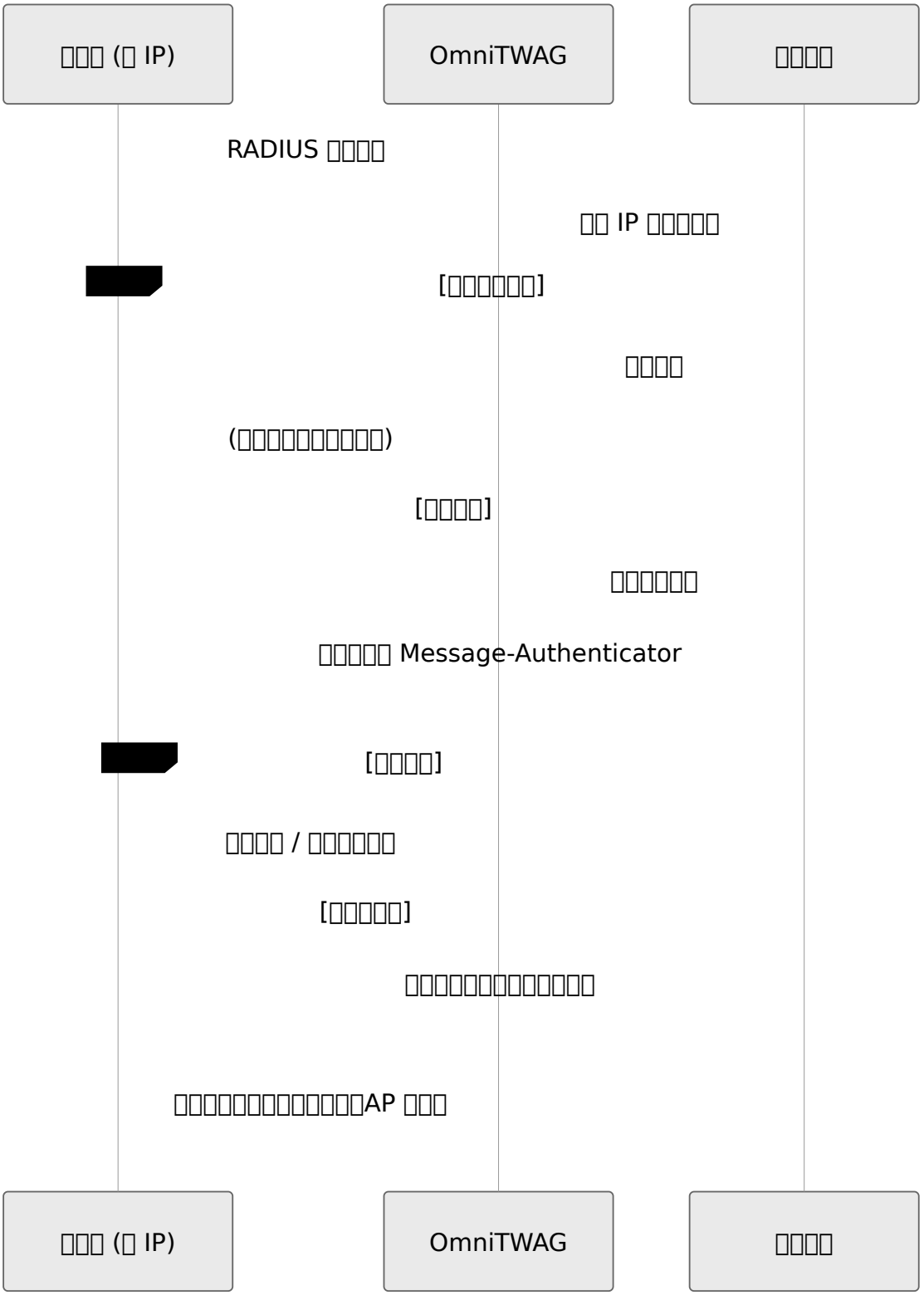
凭据子网还 授权 接入点。如果源 IP 与凭据子网匹配，则接入点被授权。如果源 IP 与任何凭据子网都不匹配，OmniTWAG 将丢弃该数据包。这意味着当操作员为新接入点添加凭据时，接入点会同时被授权。

身份验证如何使用凭据

OmniTWAG 为每个传入的 RADIUS 数据包解析共享密钥。步骤如下：

1. OmniTWAG 读取数据包的源 IP。
2. OmniTWAG 查找与源 IP 匹配的凭据子网。如果多个子网匹配，OmniTWAG 使用 最长前缀（最具体的匹配）的子网。
3. 如果没有子网匹配，OmniTWAG 将丢弃数据包。源未被授权。
4. 如果有子网匹配，OmniTWAG 使用该凭据的密钥解码数据包，检查 `Message-Authenticator`，加密 MPPE 密钥，并签署回复。

如果子网匹配但 `Message-Authenticator` 不验证，接入点持有的密钥与文件中的不同。OmniTWAG 记录此事件并增加一个指标。OmniTWAG 仍然处理该数据包。回复是用文件中的密钥签署的，因此接入点拒绝回复，身份验证失败。这种行为使得错误配置的密钥容易被发现。它不会丢弃数据包，因此有效的边缘情况不会被错误丢弃。



接入点（源 IP）
OmniTWAG
凭据存储
RADIUS 访问请求
将源 IP 匹配到子网
[没有子网匹配]
没有凭据
(数据包丢弃，事件记录)
[子网匹配]
凭据（密钥）
用密钥验证 Message-Authenticator
[密钥匹配]
访问挑战 / 接受（签名）
[密钥不匹配]
记录错误密钥事件，增加指标
用文件中的密钥签署的回复（AP 拒绝）
接入点（源 IP）
OmniTWAG
凭据存储

凭据模型

每个凭据都有以下字段。

字段	类型	描述
`id`	字符串	唯一标识符（UUID）。由服务器设置此值。
`name`	字符串	人类可读的标签，例如 `墨尔本 CBD APs`。
`secret`	字符串	RADIUS 共享密钥。
`subnets`	字符串列表	一个或多个 CIDR 块（IPv4、IPv6 或混合），例如 `["10.10.0.0/24", "2001:db8:a::/48"]`。
`breakout_type`	字符串	该接入点会话的数据平面处理：`tunneled`、`nswo` 或 `local`。默认值为 `tunneled`。请参见 按 AP 分流和计费。
`address_source`	字符串	对于 `nswo` 分流，UE IP 的分配方式：`pool` 或 `dhcp`。默认值为 `pool`。请参见 按 AP 分流和计费。
`charging_mode`	字符串	Gy 在线计费行为：`charged`、`uncharged` 或 `auth_only`。默认值为 `charged`。请参见 按 AP 分流和计费。
`ip_pool_id`	字符串	可选的 IP 池 的 `id`，用于处理来自该接入点的 `nswo` 会话。`null` 表示没有池。
`inserted_at`	整数	创建凭据的时间，自 Unix 纪元以来的毫秒数。由服务器设置此值。
`updated_at`	整数	上次更改凭据的时间，自 Unix 纪元以来的毫秒数。由服务器设置此值。

操作员设置 `name`、`secret`、`subnets`、`breakout_type`、`address_source`、`charging_mode` 和 `ip_pool_id`。服务器设置 `id`、`inserted_at` 和 `updated_at`。

按 AP 分流和计费

每个凭据还控制该组接入点的数据平面和计费。默认值保留标准行为：通过 GTP 隧道连接到 PGW 进行在线计费。

breakout_type

`breakout_type` 设置订阅者流量如何离开 TWAG。

值	描述
`tunneled`	OmniTWAG 建立与 PGW 的 GTP-C/GTP-U 会话，并通过隧道转发订阅者流量。这是默认值。请参见 S2a GTP-C 接口。
`nswo`	非无缝 WLAN 卸载。OmniTWAG 在本地锚定会话，而不隧道到 PGW。OmniTWAG 从 IP 池中分配 UE IP。订阅者流量经过 TWAG 并卸载到本地网络。
`local`	接入点拥有数据平面。OmniTWAG 对订阅者进行身份验证，但不分配 IP。UE 从本地网络获取其地址。

address_source

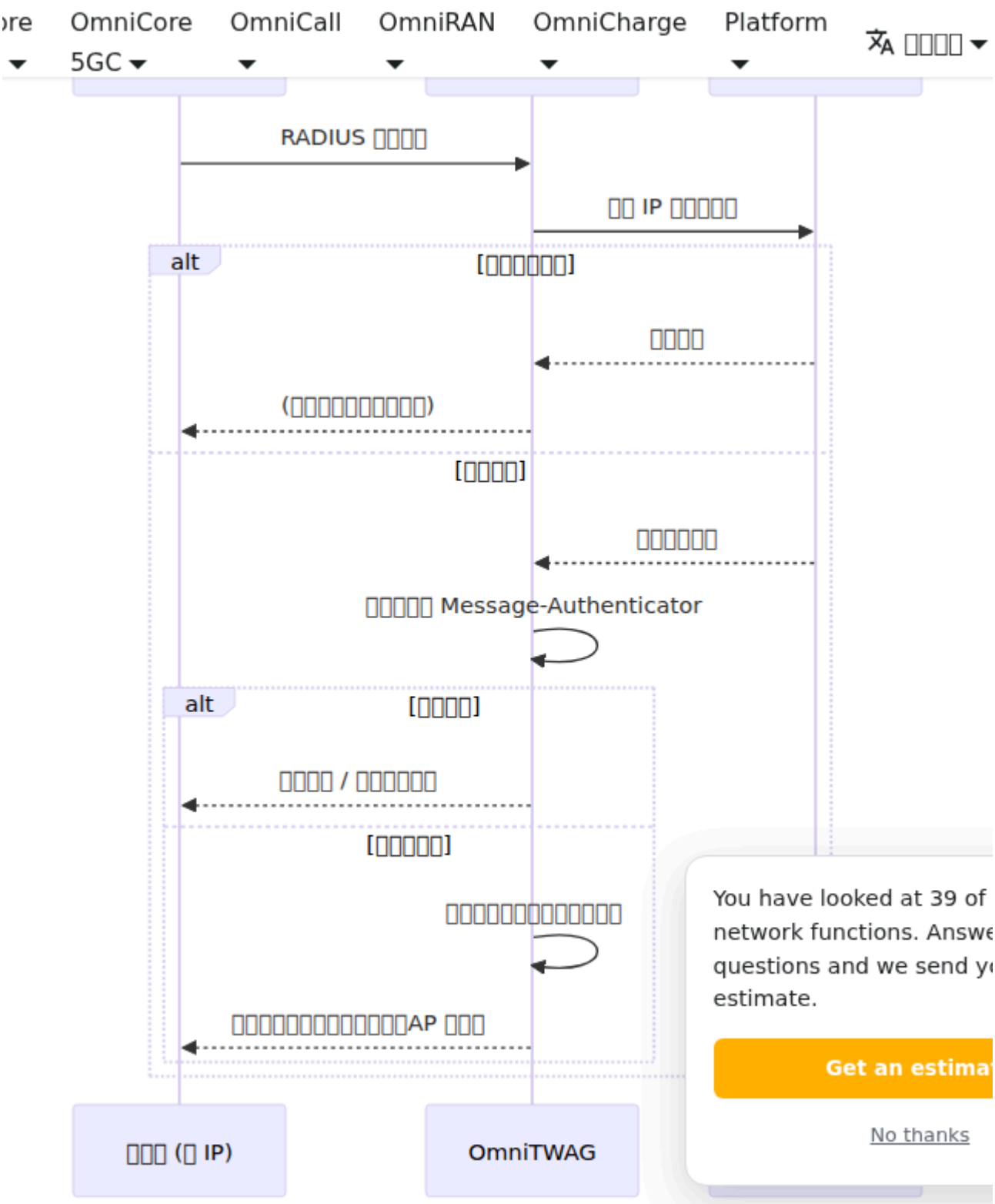
`address_source` 仅在 `breakout_type` 为 `nswo` 时适用。它设置 UE IP 的分配方式。请参见 IP 池。

值	描述
`pool`	OmniTWAG 为每个会话从关联的 IP 池中预分配一个固定地址。这是默认值。
`dhcp`	当 UE 发送 DHCP DISCOVER 时，OmniTWAG DHCP 服务器从池范围中分配一个地址。

charging_mode

`charging_mode` 设置 Gy 在线计费行为。请参见 在线计费。

值	描述
`charged`	OmniTWAG 打开 Gy 在线计费会话并向 OCS 报告使用情况。这是默认值。
`uncharged`	OmniTWAG 建立数据会话，但不打开 Gy 会话。对于固定费率或非计费访问，请使用此值。
`auth_only`	OmniTWAG 对订阅者进行身份验证，但不建立数据会话。当另一个系统提供连接时，请使用此值。



字段之间的交互

`charging_mode: auth_only` 优先级最高。当凭据使用 `auth_only` 时，OmniTWAG 不建立数据会话，因此 `breakout_type` 和 `address_source` 没有影响。

REST API

OmniTWAG REST API 通过 HTTPS 在配置的 `api_ex` 端口（默认 `8444`）提供这些端点。基本路径为 `/radius_credentials`。

方法	路径	操作	描述
GET	`/radius_credentials`	index	列出所有凭据。
GET	`/radius_credentials/{id}`	show	通过 `id` 获取一个凭据。
POST	`/radius_credentials`	create	创建一个凭据。
PUT	`/radius_credentials/{id}`	update	修改一个凭据。
DELETE	`/radius_credentials/{id}`	delete	删除一个凭据。

请求体

`POST` 和 `PUT` 接受具有以下字段的 JSON 主体。

字段	类型	必需	描述
`name`	字符串	否	人类可读的标签。
`secret`	字符串	是	RADIUS 共享密钥。不能为空。
`subnets`	字符串列表	是	一个或多个有效的 CIDR 块。不能为空。
`breakout_type`	字符串	否	`tunneled`、`nswo` 或 `local`。默认值为 `tunneled`。请参见 按 AP 分流和计费。
`address_source`	字符串	否	`pool` 或 `dhcp`（对于 `nswo`）。默认值为 `pool`。
`charging_mode`	字符串	否	`charged`、`uncharged` 或 `auth_only`。默认值为 `charged`。
`ip_pool_id`	字符串	否	用于 `nswo` 地址的 IP 池 的 `id`。空字符串表示没有池。

`PUT` 仅更改主体中的字段。主体中未包含的字段保持其当前值。

响应

状态	意义
`200 OK`	请求成功（索引、显示、更新、删除）。
`201 Created`	凭据已创建。
`404 Not Found`	没有具有给定 `id` 的凭据。
`422 Unprocessable Entity`	`secret` 为空，子网不是有效的 CIDR 块，或 `breakout_type` / `address_source` / `charging_mode` 不是有效值。

示例：列出凭据

请求：

```
GET /radius_credentials
```

响应：

```
{
  "count": 1,
  "radius_credentials": [
    {
      "id": "0d1e2f3a-4b5c-6d7e-8f90-1a2b3c4d5e6f",
      "name": "墨尔本 CBD APs",
      "secret": "s3cr3t",
      "subnets": ["10.10.0.0/24", "2001:db8:a::/48"],
      "inserted_at": 1734512400000,
      "updated_at": 1734512400000
    }
  ]
}
```

示例：创建凭据

请求：

```
POST /radius_credentials
```

```
{
  "name": "墨尔本 CBD APs",
  "secret": "s3cr3t",
  "subnets": ["10.10.0.0/24", "2001:db8:a::/48"]
}
```

成功的请求返回 `201 Created`，并包含凭据及其 `id`。

示例：无效子网

请求：

```
{
  "secret": "s3cr3t",
  "subnets": ["10.10.0.0/24", "not-a-subnet"]
}
```

响应（`422 Unprocessable Entity`）：

```
{
  "error": "无效的 CIDR 子网(s): not-a-subnet"
}
```

Web 界面

控制面板在 `/radius_credentials` 处有一个 **RADIUS** 密钥 页面。该页面列出了每个凭据及其名称、子网、掩码密钥和最后更改时间。该页面有一个表单用于添加凭据，以及编辑或删除凭据的控件。在子网字段中输入一个或多个子网。用逗号或换行符分隔子网。

配置

RADIUS 凭据存储将其数据保存在 DETS 文件中。在 `radius_config` 块中设置文件路径。

```
config :omnitwag,
  radius_config: %{
    # 持久 RADIUS 凭据存储的路径。
    credentials_dets_path: "priv/radius_credentials.dets",
    # 旧版单密钥字段（见下文“迁移”）。
    allowed_source_subnets: [],
    secret: "123456"
  }
```

□□	□□	□□	□□	□□
`credentials_dets_path`	□□□	□	`priv/radius_credentials.dets`	□□□□□ DETS □□□ □□□□ □□□□ □□□□ □□□□ □□□
`secret`	□□□	□	`123456`	□□□□ □□□ □□□□ □□□□ □□□□ □□□ □□□□ □ □□ □□□□ □□□□
`allowed_source_subnets`	□□□□□	□	`[]`	□□□ IP □□ □□□□ □□□□ □□□□ □□□□ □□□□ □ □□ □□□□ □□□□

从单一静态密钥迁移

早期版本使用一个静态 `secret` 和一个可选的 `allowed_source_subnets` 允许列表。在首次启动时，如果凭据存储为空，OmniTWAG 将从这些旧版字段中初始化一个凭据。仅在以下两个条件都为真时，初始化会运行：

- 存储中没有凭据。
- 设置了 `secret`，并且 `allowed_source_subnets` 不是空的。

初始化的凭据使用旧版 `secret` 和旧版子网。这使现有部署在升级后继续正常工作。OmniTWAG 在初始化凭据时记录一条消息。

如果 `allowed_source_subnets` 为空，OmniTWAG 不会初始化凭据。在这种情况下，OmniTWAG 将拒绝所有 RADIUS 请求，直到操作员添加凭据。通过 REST API 或 Web 界面 添加凭据。

逐包身份验证不读取 `secret` 或 `allowed_source_subnets`。这些字段仅作为首次启动的初始化。通过 API 或 Web 界面管理实时凭据。

指标

OmniTWAG 在 Prometheus 端���上公开这些计数器。有关端点和查询示例，请参见 指标参考。

radius_auth_unauthorized_source_count

类型：计数器

描述：因为源 IP 不匹配任何凭据子网而丢弃的 RADIUS 数据包数量。

标签	描述
`source`	被丢弃数据包的源 IP。

radius_auth_bad_secret_count

类型：计数器

描述：源 IP 匹配凭据，但 `Message-Authenticator` 未验证的 RADIUS 数据包数量。接入点持有的密钥与文件中的不同。

标签	描述
`source`	接入点的源 IP。
`credential_id`	匹配凭据的 `id`。

故障排除

接入点没有响应

症状：接入点发送 RADIUS 请求但未收到回复。`radius_auth_unauthorized_source_count` 指标增加。

可能的原因：

- 没有凭据子网包含接入点的源 IP。
- 接入点从不同的 IP 发送，例如 NAT 地址。

解决方案：

1. 在 OmniTWAG 日志中查找接入点的源 IP。
2. 确认凭据子网包含该 IP。
3. 如果没有凭据覆盖该 IP，请为正确的子网添加凭据。

接入点已授权但身份验证失败

症状：接入点的 `radius_auth_bad_secret_count` 指标增加。客户端无法进行身份验证。

可能的原因：

- 接入点持有的共享密钥与文件中的凭据不同。

解决方案：

1. 阅读指标上的 `credential_id` 标签，或在日志中查找错误密钥消息。
2. 将接入点上的密钥与凭据中的密钥进行比较。
3. 更正接入点上的密钥，或更新凭据以匹配。

重启后凭据丢失

症状：重启后凭据列表为空。OmniTWAG 拒绝所有请求。

可能的原因：

- `credentials_dets_path` 指向不可写的目录，或指向在重启时被清除的路径。

解决方案：

1. 将 `credentials_dets_path` 设置为可写的持久数据目录。
2. 确认 OmniTWAG 进程可以写入该路径。
3. 通过 API 或 Web 界面再次添加凭据。

RADIUS 拒绝

□述

OmniTWAG 跟踪被拒绝的 RADIUS 源的包。这使得操作员能够一目了然地发现未知或配置错误的接入点。此功能是 RADIUS 共享密钥 的镜像：凭证存储记录 OmniTWAG 接受 的源，而拒绝跟踪器记录 OmniTWAG 拒绝 的源。

跟踪器将其数据保存在内存中。数据不是持久的。跟踪器在重启时重置。这是设计使然。请参见 故障排除。

目录

- □念
- 数据包如何到达跟踪器
- 拒绝原因
- 条目模型
- 存储行为
- REST API
- 指标
- 故障排除

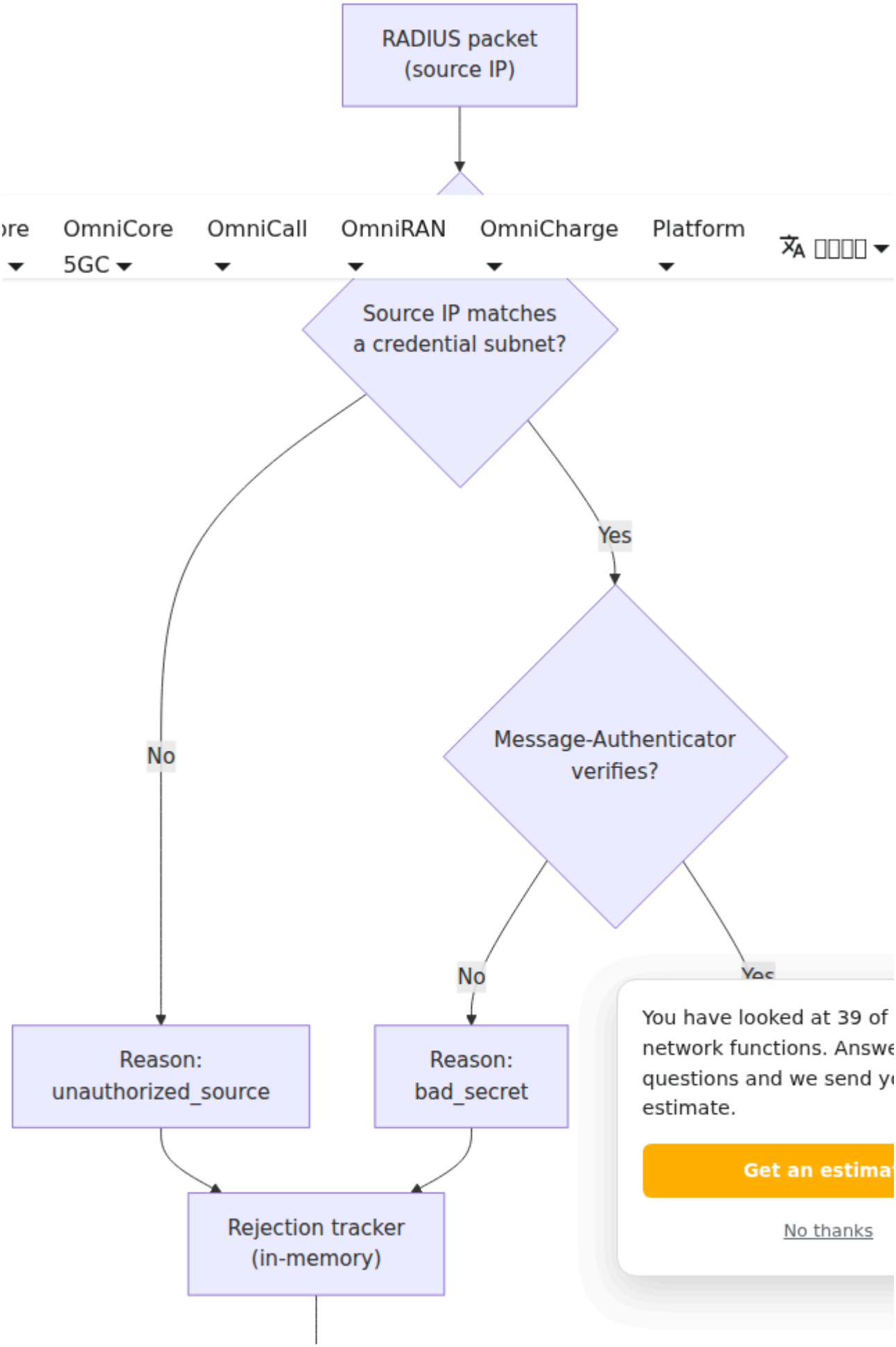
概念

拒绝源 是发送了 OmniTWAG 未接受的 RADIUS 数据包的 IP 地址。跟踪器为每对唯一的 源 和 原因 记录一个条目。如果同一源因相同原因多次被拒绝，OmniTWAG 会在现有条目上增加一个计数器。OmniTWAG 不会添加新行。

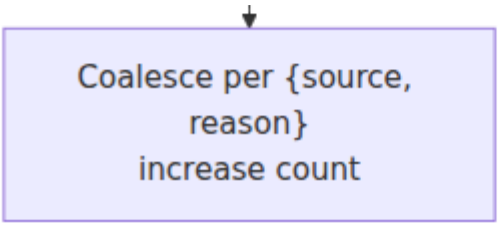
源 **IP** 是发送 RADIUS 数据包的 IP 地址。OmniTWAG 从接收到的 UDP 数据包中读取源 IP。OmniTWAG 不使用 `NAS-IP-Address` 属性，因为攻击者可以将该属性设置为任何值。

数据包如何到达跟踪器

OmniTWAG 将每个传入的 RADIUS 数据包与凭证子网进行匹配。当数据包未通过两个检查之一时，它将被拒绝。跟踪器记录每个被拒绝数据包的源和原因。



You have looked at 39 of
network functions. Answe
questions and we send y
estimate.
Get an estima
No thanks



拒绝原因

源因以下两种原因之一被拒绝。

原因	意义
`unauthorized_source`	源 IP 未匹配任何凭证子网。OmniTWAG 丢弃了该数据包。这是未知接入点的情况。
`bad_secret`	匹配了凭证子网，但数据包的 `Message-Authenticator` 未通过验证。接入点持有的共享密钥与文件中的不同。

对于 `bad_secret` 条目，跟踪器还记录匹配的凭证。这告诉操作员需要检查哪个凭证。

条目模型

每个条目具有以下字段。

字段	类型	描述
`source`	字符串	被拒绝数据包的源 IP。此值是条目标识符。
`reason`	字符串	拒绝原因：`unauthorized_source` 或 `bad_secret`。
`count`	整数	OmniTWAG 因该原因拒绝此源的次数。
`first_seen`	整数	源首次被拒绝的时间，自 Unix 纪元以来的毫秒数。
`last_seen`	整数	源最后被拒绝的时间，自 Unix 纪元以来的毫秒数。
`credential_id`	字符串	匹配凭证的 `id`。仅在 `reason` 为 `bad_secret` 时设置。
`credential_name`	字符串	匹配凭证的 `name`。仅在 `reason` 为 `bad_secret` 时设置。

对于 `unauthorized_source` 条目，`credential_id` 和 `credential_name` 字段为空，因为没有凭证匹配。

存储行为

跟踪器是一个内存存储。它具有以下属性。

- 非持久性。存储在重启时重置。它不会在进程重启后存活。
- 合并。OmniTWAG 为每对 `source` 和 `reason` 记录一个条目。重复拒绝会增加 `count` 并更新 `last_seen`。它不会添加新行。
- 有界。存储保持最大数量的源。默认最大值为 256。
- 最近最少使用驱逐。当存储满时，OmniTWAG 删除 `last_seen` 值最旧的条目以为新源腾出空间。存储是有界的，因为源 IP 可以被伪造。伪造的源 IP 洪水不能无限制地增长存储。

REST API

OmniTWAG REST API 在配置的 API 端口（默认是 `8444`）提供以下端点。基本路径是 `/radius_rejections`。

方法	路径	操作	描述
GET	`/radius_rejections`	index	列出被拒绝的源。列表是分页的。按 `source` 或 `reason` 搜索。
DELETE	`/radius_rejections/{id}`	delete	忘记一个源。`{id}` 是源 IP。

使用 `DELETE` 忘记一个源，在您解决它之后，例如在您添加了覆盖源 IP 的凭证之后。

响应

状态	意义
`200 OK`	请求成功。
`404 Not Found`	没有具有给定源 IP 的条目。

示例：列出被拒绝的源

请求：

```
GET /radius_rejections
```

响应：

```
{
  "count": 2,
  "radius_rejections": [
    {
      "source": "203.0.113.7",
      "reason": "unauthorized_source",
      "count": 42,
      "first_seen": 1734512400000,
      "last_seen": 1734512999000,
      "credential_id": null,
      "credential_name": null
    },
    {
      "source": "10.10.0.15",
      "reason": "bad_secret",
      "count": 3,
      "first_seen": 1734512500000,
      "last_seen": 1734512890000,
      "credential_id": "0d1e2f3a-4b5c-6d7e-8f90-1a2b3c4d5e6f",
      "credential_name": "Melbourne CBD APs"
    }
  ]
}
```

示例：忘记一个源

请求：

```
DELETE /radius_rejections/203.0.113.7
```

响应：

```
{
  "deleted": "203.0.113.7"
}
```

指标

拒绝跟踪器补充了凭证功能文档中的两个 Prometheus 计数器。每个计数器在 OmniTWAG 记录拒绝时同时增加。

计数器	拒绝原因
`radius_auth_unauthorized_source_count`	`unauthorized_source`
`radius_auth_bad_secret_count`	`bad_secret`

计数器提供了随时间变化的总数。跟踪器提供了每个源的实时视图。请参见 RADIUS 共享密钥 以获取计数器定义，以及 指标参考 以获取 Prometheus 端点和查询示例。

故障排除

列表中出现未知接入点

症状：一个源出现，`reason` 设置为 `unauthorized_source`。该源的 `radius_auth_unauthorized_source_count` 指标增加。

可能原因：

- 没有凭证子网包含��? IP。
- 接入点位于 NAT 后面。OmniTWAG 看到的 RADIUS 源 IP 与接入点本身的地址不同。

解决方案：

1. 阅读条目的 `source` 字段。
2. 确认这是一个已知的接入点。
3. 添加一个覆盖 OmniTWAG 实际看到的源 IP 的凭证子网。对于 NAT 情况，使用转换后的地址，而不是接入点的地址。请参见 RADIUS 共享密钥。
4. 使用 `DELETE /radius_rejections/{id}` 删除条目以忘记该源。

已知接入点出现 **bad_secret**

症状：一个已知源出现，`reason` 设置为 `bad_secret`。该源的 `radius_auth_bad_secret_count` 指标增加。

可能原因：

- 接入点持有的共享密钥与文件中的凭证不同。

解决方案：

1. 阅读条目的 `credential_id` 和 `credential_name` 字段。这些字段命名匹配的凭证。
2. 比较接入点上的密钥与该凭证中的密钥。
3. 更正接入点上的密钥，或更新凭证以匹配。请参见 RADIUS 共享密钥。
4. 使用 `DELETE /radius_rejections/{id}` 删除条目以忘记该源。

重启后列表为空

症状：重启后拒绝列表为空，即使在重启之前列出了源。

原因：

- 这是预期的行为。���踪器将其数据保存在内存中。数据不是持久的。存储在重启时重置。

解决方案：

- 无需采取任何措施。新的拒绝将重新填充列表。使用 `radius_auth_unauthorized_source_count` 和 `radius_auth_bad_secret_count` 指标获取在重启后仍然有效的历史记录。

□述

S2a / GTP-C 接口到 PGW

述

S2a 接口将 OmniTWAG 连接到数据包数据网络网关 (PGW)。它为受信 WLAN 接入网络 (TWAN) 提供基于 GTP 的 PDN 连接。OmniTWAG 充当 TWAN GTP-C 端点。这个角色类似于 S2b 接口上的 ePDG，但它使用 S2a 接口类型和 TWAN 标识符。

S2a 接口有两个平面：

- **GTP-C**（控制平面）：创建、修改和删除 PDN 会话。它还进行路径管理。本文档涵盖 GTP-C。
- **GTP-U**（用户平面）：在 GTP 隧道中承载 UE 数据。请参见 用户平面 指南。

S2a 上的 GTP-C 遵循 3GPP TS 29.274（GTPv2-C）。TWAG 使用 GTPv2-C 消息类型和信息元素（IE）与 S2a 特定接口类型。

目录

- S2a 的位置：三种分流模式
- 会话生命周期中的角色
- S2a 接口类型
- 会话建立
  - 触发
  - 创建会话请求
  - 默认承载
  - 创建会话响应
  - 紧急会话
- 承载管理
- 会话拆除
- 路径管理
- 配置
- 参考表
- 可观察性
- 故障排除

- 3GPP 参考

S2a 的位置：三种分流模式

OmniTWAG 根据与 AP 匹配的 RADIUS 凭据为每个接入点 (AP) 解析数据平面模式。凭据上的 `breakout_type` 值选择三种模式之一。请参见 RADIUS 凭据 以获取 `breakout_type`、`address_source` 和 `ip_pool_id` 设置。

模式 (`breakout_type`)	谁分配 **UE IP**	数据平面	使用的 **S2a / GTP**
`tunneled`	PGW（在 PDN 地址分配 IE 中）	从 AP 到 TWAG 的 GRE，然后是 S2a 上的 GTP-U 到 PGW，然后由 PGW 连接到互联网	是的。本文档。
`nswo`	TWAG，来自 IP 池	从 AP 到 TWAG 的 GRE，然后在 TWAG 处本地分流到互联网（NAT）	否。没有 PGW 会话。
`local`	AP 的本地网络	AP 拥有数据平面。TWAG 不承载流量。	否。没有 PGW 会话，没有 IP。

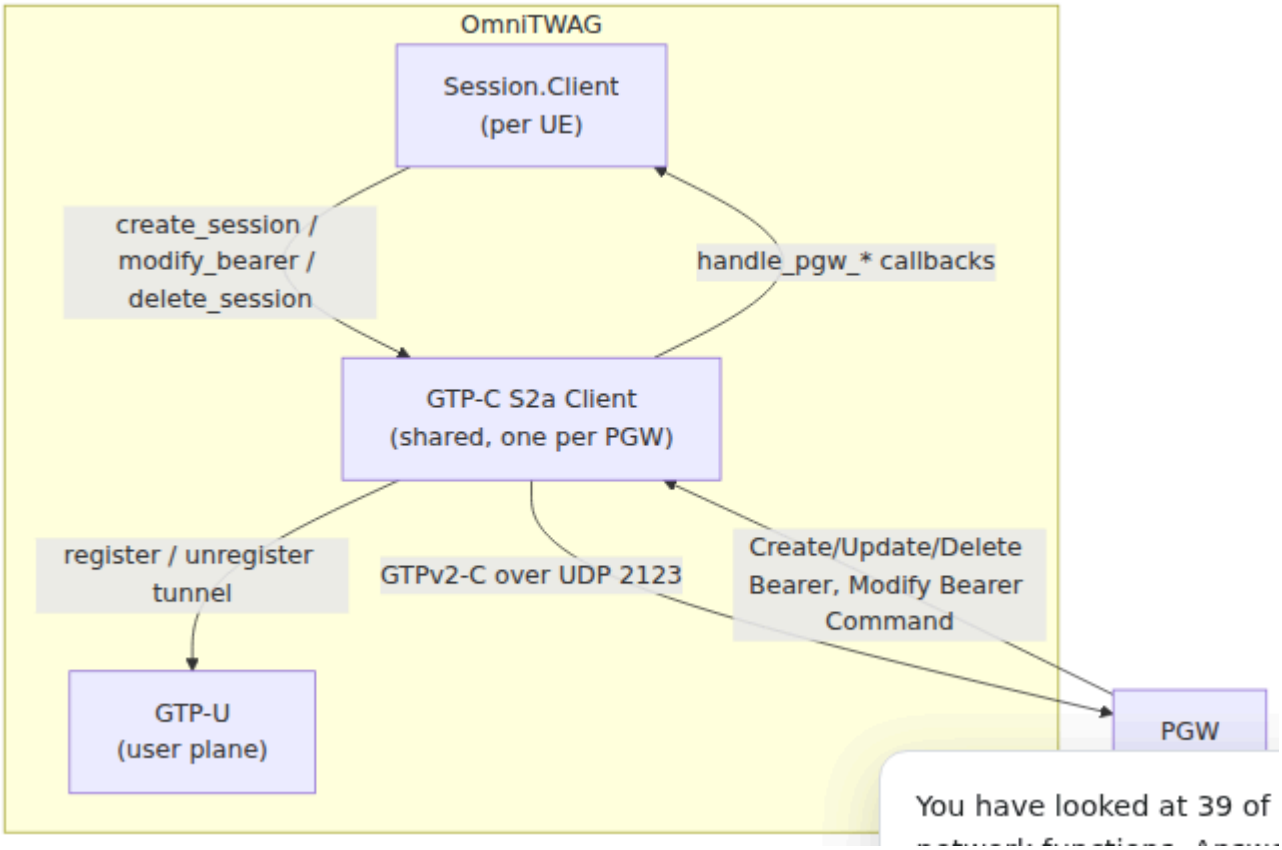
S2a GTP-C 仅适用于 `tunneled` 模式。在 `tunneled` 模式下，PGW 分配 UE IP 地址并应用策略和计费。在 `nswo` 和 `local` 模式下，TWAG 不创建 PDN 会话，因此不发送 S2a 消息。

在 `tunneled` 模式下，计费由 PGW 完成，而不是 TWAG。TWAG 侧的 Gy 在线计费在 在线计费 中适用于本地分流模式。

> 注意：一个单独的全局非无缝 WLAN 卸载 (NSWO) 处理程序 (`Twag.NSWO.Handler`，通过 `nswo.enabled: true` 启用) 为每个会话提供本地分流，当不使用 GTP 时。请参见用户平面指南的 本地分流 部分。上面描述的每个 AP 的 `nswo` 分流类型是凭据驱动模型，并且是首选的。

会话生命周期中的角色

GTP-C 客户端是一个单一的共享进程 (`Twag.Gtp.GtpcS2a`)。它处理针对一个配置的 PGW 的所有会话。每个经过身份验证的 `tunneled` UE 会话驱动一个创建会话交换。每个会话的 `Session.Client` 进程调用 GTP-C 客户端。它本身不发送 GTPv2-C 消息。



S2a 接口类型

TWAG 使用 S2a F-TEID 接口类型。这些与 S2b（ePDG）不同，后者使用类型 30 和 31。值遵循 TS 29.274 表 8.22-1。

接口类型	值	方向	用途
`s2a_twan_gtp_c`	35	TWAG 控制	TWAG F-TEID 在创建会话请求中
`s2a_twan_gtp_u`	34	TWAG 用户	TWAG 承载 F-TEID（GTP-U）
`s2a_pgw_gtp_c`	36	PGW 控制	PGW F-TEID 在创建会话响应中
`s2a_pgw_gtp_u`	37	PGW 用户	PGW 承载 F-TEID（GTP-U）

RAT 类型始终为 WLAN（值 3）。

会话建立

触发

成功的 EAP 身份验证启动会话。AP 发送 Accounting-Start，这触发后接受处理。`Session.Client` 解析 AP 分流模式。对于 `tunneled` 模式，它调用 GTP-C 客户端以创建 PDN 会话。如果 `gtp.enabled` 参数为 `false`，TWAG 将跳过 S2a。如果会话没有 IMSI，TWAG 不会创建会话。对于 `nswo` 和 `local` 模式，TWAG 不发送任何 S2a 消息。请参见 会话 以获取完整的会话生命周期。

创建会话请求

TWAG 向 PGW 发送创建会���请求（消息类型根据 TS 29.274 §7.2.1）。TWAG 为请求生成一个新的、不可预测的控制平面 TEID 和一个用户平面 TEID。它使用以下信息元素构建请求：

信息元素	来源 / 值	备注
IMSI	UE 订阅	如果不存在则省略。
服务网络 (MCC/MNC)	`gtp.mcc` / `gtp.mnc`，或 IMSI 的前几位数字	如果任一值缺失则省略。
RAT 类型	WLAN (3)	对于 S2a 固定。
F-TEID (控制)	接口类型 35，TWAG 控制 TEID，TWAG 本地 IP	标识 TWAG GTP-C 端点。
接入点名称 (APN)	`gtp.apn`，默认 `internet`。如果请求则为紧急 APN。	请参见 紧急会话。
选择模式	2（网络提供，未验证订阅）	APN 在 TWAN 中始终为网络提供。
PDN 地址分配 (PAA)	请求类型，地址为零	TWAG 请求一个地址。PGW 分配它。
聚合最大比特率 (APN-AMBR)	默认 256000 上行 / 256000 下行	如果提供速率则覆盖。
创建的承载上下文	默认 EBI 5，承载 F-TEID（类型 34），QoS	请参见 默认承载。
APN 限制	2（正常），或 0（紧急）	
ME 身份 (IMEISV)	UE 设备身份	如果不存在则省略。
恢复	TWAG 恢复计数器	用于重启检测。
计费特性	`0x0800`	固定值。
TWAN 标识符	从 `twan_ssid` 获取的 SSID，来自 `twan_bssid` 的可选 BSSID	标识 WLAN。
UE 时区	从主机本地时间偏移派生	

信息元素	来源 / 值	备注
指示	切换标志	切换标志在切换中设置。

PDN 类型默认为 IPv4。可以通过 `gtp.pdn_type` 参数设置为 `:ipv4`、`:ipv6` 或 `:ipv4v6`。请求的 PAA 地址始终为零，因此 PGW 分配真实地址。

> 注意：附加协议配置选项 (APCO) IE 未包含。TWAG 目前不通过 APCO 请求 DNS 或 P-CSCF 地址。如果 PGW 提供 DNS，则从 PCO 或附加 PCO IE 中读取。

默认承载

TWAG 每个会话创建一个默认承载。默认 EBI 为 5。承载没有 TFT 过滤器，因此承载所有流量。请求中的承载级 QoS 使用以下固定值：

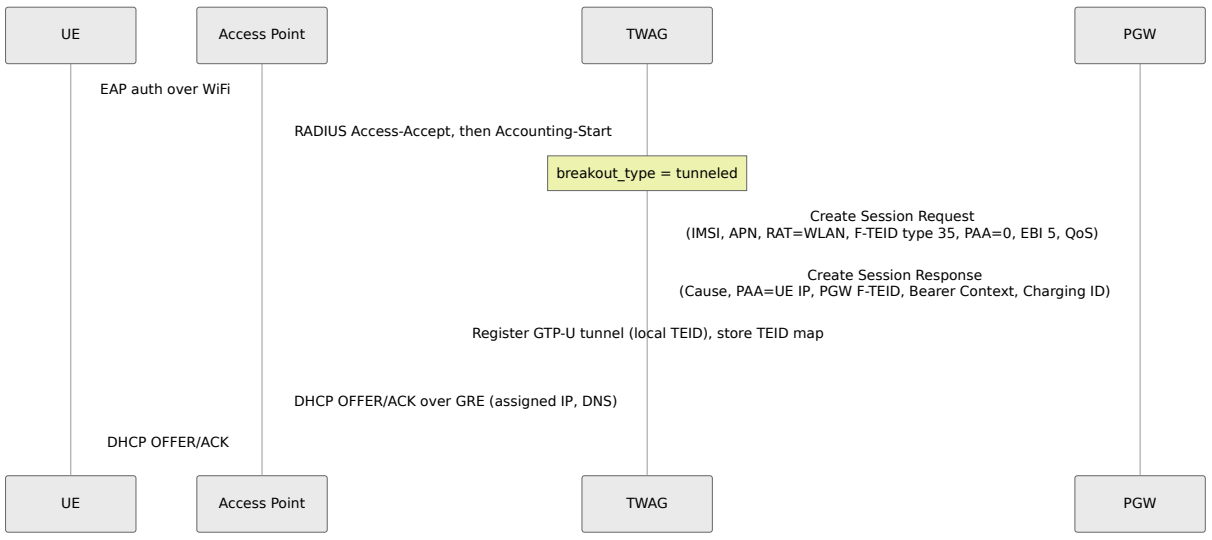
QoS 参数	值
QCI	9
优先级级别	10
抢占能力	禁用
抢占脆弱性	启用
GBR / MBR（上行和下行）	0

TWAG 不支持专用承载。请参见 承载管理。

创建会话响应

TWAG 根据序列号将响应与待处理请求匹配。它检查原因 IE。如果原因是 `request_accepted`、`new_pdp_type_due_to_network_preference` 或 `new_pdp_type_due_to_single_address_bearer_only`，则请求成功。任何其他原因返回带有该原因值的 `gtpc_error`。

成功时，TWAG 从响应中读取以下值：



响应元素	提取值	用途
PDN 地址分配 (PAA)	分配的 UE IP 地址	通过 DHCP 通过 GRE 传递给 UE。请参见 IP 地址管理。
F-TEID（PGW 控制）	PGW 控制 TEID 和 PGW IP	用作后续控制消息的目的地。
承载上下文	承载 F-TEID（PGW GTP-U TEID 和 IP）、EBI、计费 ID	用于注册 GTP-U 隧道。
APN-AMBR	会话速率限制	存储在会话中。
PCO / 附加 PCO	DNS 服务器地址	通过 DHCP 传递给 UE。

TWAG 存储从其本地控制 TEID 到 PGW 控制 TEID 和 UE 身份的映射。此映射将后续 PGW 发起的消息路由到正确的会话。如果没有承载 F-TEID，TWAG 将回退到用于用户平面隧道的 PGW F-TEID。

成功后，`Session.Client` 注册 GTP-U 隧道（生成 TWAG 侧本地用户平面 TEID），使用分配的 IP 和 DNS 预分配 DHCP 租约，并注册 UE 进行 GRE 下行路由。请参见 用户平面 以获取隧道和下行路径。

紧急会话

如果会话请求紧急服务，TWAG 使用紧急 APN。`emergency_apn` 参数设置此 APN。默认值为 `sos`。对于紧急会话，APN 限制设置为 0（没有限制）。

承载管理

仅默认承载

OmniTWAG 仅支持默认承载。它不支持专用承载。这遵循 TWAN 透明单连接模式 (TSCM) 配置文件。承载管理器可以跟踪以 EBI 为键的承载表，但 S2a 仅建立默认承载。

修改承载（**TWAG** 发起）

TWAG 可以发送修改承载请求（TS 29.274 §7.2.7）。此请求携带一个承载上下文，包含默认 EBI、TWAG 用户平面 F-TEID（类型 34）和 TWAG 本地 IP。目标 TEID 是 PGW 控制 TEID。TWAG 根据序列号匹配修改承载响应，并检查原因 IE 是否成功。切换时重用创建会话路径，并设置切换标志，而不是单独的修改承载。

PGW 发起的程序

PGW 可以启动承载程序。TWAG 在其 GTP-C 套接字上接收这些消息。消息中的目标 TEID 是 TWAG 本地控制 TEID。TWAG 使用存储的 TEID 映射查找 PGW 控制 TEID 和 UE 身份。然后，它通知匹配的 `Session.Client` 进程。

PGW 消息	**TWAG** 响应	行为
Create Bearer Request	Create Bearer Response, cause `request_rejected`	TWAG 拒绝所有专用承载（TSCM）。没有承载被创建。
Modify Bearer Command	Modify Bearer Request back to the PGW	TWAG 确认新参数（APN-AMBR、F-TEID）。如果发送失败，则发送带有原因 `system_failure` 的修改承载失败指示。
Update Bearer Request	Update Bearer Response, cause `request_accepted` per bearer	TWAG 接受 QoS 更新并通知会话。
Delete Bearer Request	Delete Bearer Response, cause `request_accepted`	请参见 会话拆除。

对于修改承载命令，TWAG 读取 APN-AMBR 和承载上下文（EBI、新 F-TEID、QoS）。它通知会话进程，然后向 PGW 发送修改承载请求以确认。如果承载上下文携带新的 GTP-U F-TEID，则会���使用新的远程值重新注册 GTP-U 隧道。对于更新承载请求，TWAG 读取 APN-AMBR 和每个承载上下文（EBI、QoS），通知会话，然后接受响应中的每个承载。

会话拆除

删除会话（**TWAG** 发起）

当 UE 断开连接时，TWAG 删除 PDN 会话。`Session.Client` 拆除调用 GTP-C 客户端。TWAG 向 PGW 控制 TEID 发送删除会话请求（TS 29.274 §7.2.9）。请求携带：

- EPS 承载 ID（默认 EBI，默认 5）
- 原因（默认 `local_detach`）

TWAG 根据序列号匹配删除会话响应，并检查原因 IE。然后，它删除该 PGW 控制 TEID 的 TEID 映射条目。如果会话没有 PGW 控制 TEID（例如 `nswo` 或 `local` 会话），TWAG 将跳过 GTP-C 删除。删除后，`Session.Client` 注销 GTP-U 隧道，释放 DHCP 租约和任何 IP 池地址，注销 UE 的 GRE，并清理会话。

删除承载（**PGW** 发起）

PGW 可以通过删除承载请求启动拆除。TWAG 读取链接的 EBI，EBI 列表和原因。它通知匹配的 `Session.Client` 进程，该进程进行本地清理。在这种情况下，TWAG 不会发送删除会话请求，因为 PGW 已经删除了承载。然后，TWAG 发送带有原因 `request_accepted` 的删除承载响应。如果链接的 EBI 是默认承载，则整个 PDN 连接被拆除。

路径管理

回声

TWAG 通过回声请求和回声响应运行 GTPv2-C 路径管理（TS 29.274 §6）。`echo_enabled` 参数开启此功能。默认情况下是开启的。TWAG 每 60 秒向 PGW 发送一次回声请求。每个回声请求携带 TWAG 序列号。每个回声响应携带恢复计数器。

当 PGW 发送回声请求时，TWAG 回复带有 TWAG 恢复计数器的回声响应。

路径状态

TWAG 跟踪到 PGW 的路径状态。成功的回声响应标记路径为 UP，并重置失败计数。连续的回声失败会增加计数器。连续 3 次失败后，TWAG 将路径标记为 DOWN。

PGW 重启检测

TWAG 存储来自每个回声响应的 PGW 恢复计数器。如果计数器发生变化，TWAG 检测到 PGW ���启。然后，它清理所有受影响的会话。它通知每个会话进程，原因是 `pgw_restart`，并清除 TEID 映射。

请求超时和重传

TWAG 使用 T3-RESPONSE 定时器和 N3-REQUESTS 重试控制请求。

定时器 / 计数器	值	含义
T3-RESPONSE	3000 ms	等待响应的时间，然后进行重传。
N3-REQUESTS	3	请求失败之前的最大重传次数。

如果请求超时，TWAG 将重传相同的消息，使用相同的序列号。它重新启动定时器。在 3 次重试后没有响应，TWAG 将请求标记为失败。创建会话失败返回 `timeout` 给调用者。回声重传遵循相同的定时器和重试限制。回声失败计入路径 DOWN 阈值。

TWAG 在启动时打开 GTP-C UDP 套接字。如果套接字无法打开，TWAG 每 5 秒重试一次。

配置

S2a 在 `gtp` 映射下进行配置。该映射是 `runtime.exs` 中 `omnitwag` 应用程序配置的一部分。

```
config :omnitwag,
  gtp: %{
    # 启用 S2a 接口（需要用于隧道分流）
    enabled: true,

    # PGW GTP-C 端点
    pgw_ip: "10.5.198.1",
    pgw_port: 2123,

    # TWAG 本地 IP 用于 GTP-C 和 GTP-U F-TEIDs
    local_ip: "10.5.198.200",

    # GTP-U 用户平面端口
    gtpu_port: 2152,

    # 路径管理（回声）
    echo_enabled: true,

    # 创建会话请求默认值
    apn: "internet",
    pdn_type: :ipv4
  }
```

GTP □□

□□	□□	□□	□□	□□
`enabled`	□□□	□	`false`	□□ S2a □□□□□□ `true` □□□□□□□□□□ □ `false` □□TWAG □□□ GTP-C □□ □□□□□□□□□□ S2a□
`pgw_ip`	□□□	□□□□□	`127.0.0.1`	PGW GTP-C IP □□□TWAG □□□ GTPv2-C □□□□□□□□□
`pgw_port`	□□	□	`2123`	PGW GTP-C UDP □□□□□ GTP-C □□ □ 2123□
`local_ip`	□□□	□	□□□□	TWAG □□ IP□□□ F-TEIDs□□□□ GTP-C □ GTP-U □□□□□□□□□TWAG □□□□□□□ IP□
`gtpu_port`	□□	□	`2152`	□□□□□ GTP-U UDP □□□□□□ □□□□□
`echo_enabled`	□□□□	□	`true`	□□ GTPv2-C □□□□□□□ PGW□
`apn`	□□□□	□	`internet`	PDN □□□□□□□□□
`pdn_type`	□□	□	`:ipv4`	PDN □□□`:ipv4`□`:ipv6` □ `:ipv4v6`□
`mcc`	□□□□	□	IMSI □□ 3 □□	□□□□ IE □□□□□□□□

参数	类型	必需	默认	描述
`mnc`	字符串	否	IMSI 的第 4-5 位数字	服务网络 IE 的移动网络代码。

紧急和 TWAN 参数

以下参数从 `omnitwag` 应用程序环境读取，而不是从 `gtp` 映射中读取。

参数	类型	默认	描述
`emergency_apn`	字符串	`sos`	用于紧急会话的 APN。对于这些会话，APN 限制设置为 0。
`twan_ssid`	字符串	`TWAG`	放置在 TWAN 标识符 IE 中的 SSID。
`twan_bssid`	二进制	（未设置）	TWAN 标识符 IE 的可选 6 字节 BSSID。如果不是有效的 6 字节值，则省略。

> 注意：GTP-C 客户端连接到单个 PGW。目前没有按 APN 或多 PGW 选择。要更改 PGW，请更新 `pgw_ip`。

参考表

GTPv2-C 消息类型（S2a）

消息	方向	目的
Echo Request / Response	双向	路径管理。
Create Session Request / Response	TWAG 到 PGW	建立 PDN 会话。
Modify Bearer Request / Response	TWAG 到 PGW	更新承载 F-TEID 或 QoS。
Delete Session Request / Response	TWAG 到 PGW	拆除 PDN 会话。
Create Bearer Request / Response	PGW 到 TWAG	专用承载。被 TWAG 拒绝。
Update Bearer Request / Response	PGW 到 TWAG	更新 QoS。被 TWAG 接受。
Modify Bearer Command	PGW 到 TWAG	PGW 请求 TWAG 更新承载参数。
Modify Bearer Failure Indication	TWAG 到 PGW	当无法满足修改承载命令时发送。
Delete Bearer Request / Response	PGW 到 TWAG	PGW 发起的拆除。

GTPv2-C 原因代码

TWAG 使用这些原因值。这些代码遵循 TS 29.274 §8.4。

代码	名称	含义
2	本地分离	TWAG 发起的删除会话的默认原因。
13	网络故障	对等方报告的网络故障。
16	请求已接受	成功。
64	找不到上下文	对等方没有匹配的上下文。
72	系统故障	在修改承载失败指示中使用。
92	用户身份验证失败	身份验证被拒绝。
93	APN 访问被拒绝	不允许该 APN。
94	请求被拒绝	当 TWAG 拒绝创建承载请求时使用。
96	IMSI/IMEI 不明	订阅者未知。

可观察性

TWAG 发出承载和拆除活动的遥测事件。这些事件驱动 指标参考 中的指标。

事件	何时
`[:twag, :bearer, :modify_command]`	TWAG 接收到 PGW 修改承载命令。
`[:twag, :bearer, :update]`	TWAG 接收到 PGW 更新承载请求。
`[:twag, :detach, :pgw_delete]`	TWAG 接收到 PGW 删除承载请求。
`[:twag, :session, :teardown]`	会话被拆除。
`[:twag, :detach, :cleanup_start]`	会话拆除开始。
`[:twag, :detach, :cleanup_complete]`	会话拆除完成。

客户端状态

GTP-C 客户端报告状态快照。此状态显示 S2a 路径的健康状况。

字段	含义
`connected`	GTP-C UDP 套接字已打开。
`pgw_ip` / `pgw_port`	配置的 PGW 端点。
`pending_requests`	正在进行的 GTP-C 请求计数。
`path_up`	到 PGW 的回声路径状态。
`echo_failures`	连续回声失败计数。
`recovery_counter`	TWAG 恢复计数器。

日志记录

GTP-C 客户端记录每个过程。关键日志行包括：

- 创建会话、删除会话和修改承载的发送和响应。
- 路径状态变化：`Path to PGW ... is UP` 和 `Path to PGW ... is DOWN`。
- PGW 重启检测，包含旧的和新的恢复计数器。
- PGW 发起的创建、更新、修改和删除承载消息。

故障排除

创建会话失败

症状：会话未建立。日志显示 `GTP Create Session failed`。

可能的原因：

- PGW 以非成功原因拒绝请求。
- 到 PGW 的路径为 DOWN。
- APN 未在 PGW 上配置。
- IMSI 在核心中未知。

解决方案：

1. 检查返回的原因代码与 原因代码 表。
2. 检查客户端状态中的路径状态（`path_up`）。
3. 确认 APN 在 PGW 上已配置。
4. 确认订阅者在 HSS 中已配置。

到 **PGW** 的路径为 **DOWN**

症状：日志显示 `Path to PGW ... is DOWN`。新会话失败。

可能的原因：

- PGW 在 UDP 2123 上不可达。
- 防火墙阻止 GTP-C。
- PGW 未运行。

解决方案：

1. 确认 `pgw_ip` 和 `pgw_port` 值正确。
2. 确认网络路径允许 UDP 2123 双向通信。
3. 确认 PGW GTP-C 服务正在运行。
4. 等待下一个回声。成功的回声响应将路径标记为 UP。

会话在 **PGW** 重启后掉线

症状：所有会话一起被拆除。日志显示 `PGW restart detected`。

可能的原因：

- PGW 重启并更改了其恢复计数器。

解决方案：

1. 这是预期的行为。TWAG 在 PGW 重启后清理过时的会话。
2. 确认 PGW 稳定。UE 重新进行身份验证并重新建立会话。

会话建立但没有数据流

症状：UE 获得 IP 地址，但流量未通过。

可能的原因：

- GTP-U 隧道或 GRE 路径未开启。

解决方案：

1. 确认 `gtp.enabled` 和 `gre.enabled` 都为 `true` 以进行隧道分流。
2. 检查 用户平面 指南中的 GTP-U 和 GRE 统计信息。

3GPP 参考

规范	标题
TS 29.274	演进的 GPRS 隧道协议控制平面 (GTPv2-C)。
TS 23.402	非 3GPP 接入的架构增强 (TWAN, S2a)。
TS 24.302	通过非 3GPP 接入网络访问 EPC。

述

会话

□述

一个 会话 代表在 OmniTWAG 上的一个经过身份验证的订阅者。OmniTWAG 通过订阅者身份（IMSI 或用户名）来标识会话。OmniTWAG 将会话保存在内存中。会话 API 提供实时会话状态。

会话视图还携带订阅者的流量。OmniTWAG 通过 `Acct-Session-Id` 将 RADIUS 计费记录合并到会话中。因此，会话是订阅者状态和订阅者流量的唯一来源。没有单独的计费端点。计费是会话的一部分。

目录

- □念
- 身份验证状态生命周期
- 会话模型
- 身份验证状态参考
- REST API
- 每会话操作
- JSON 示例
- Web 界面
- 故障排除

概念

每个经过身份验证的订阅者都有一个会话。当 RADIUS Access-Request 开始 EAP 身份验证时，OmniTWAG 创建会话。OmniTWAG 在身份验证���展和数据路径建立时更新会话。

会话跟踪订阅者身份、接入点、客户端 MAC、分配的 UE IP 地址、EAP 方法和阶段，以及 GTP 和 DNS 状态。当 RADIUS 计费到达订阅者时，OmniTWAG 将流量计数合并到会话中。

会话的形状是开放的。存在的键取决于身份验证方法和会话的进展程度。仅开始 EAP 的会话比达到 RADIUS Accept 并建立 GTP 和 DHCP 数据路径的会话具有更少的键。

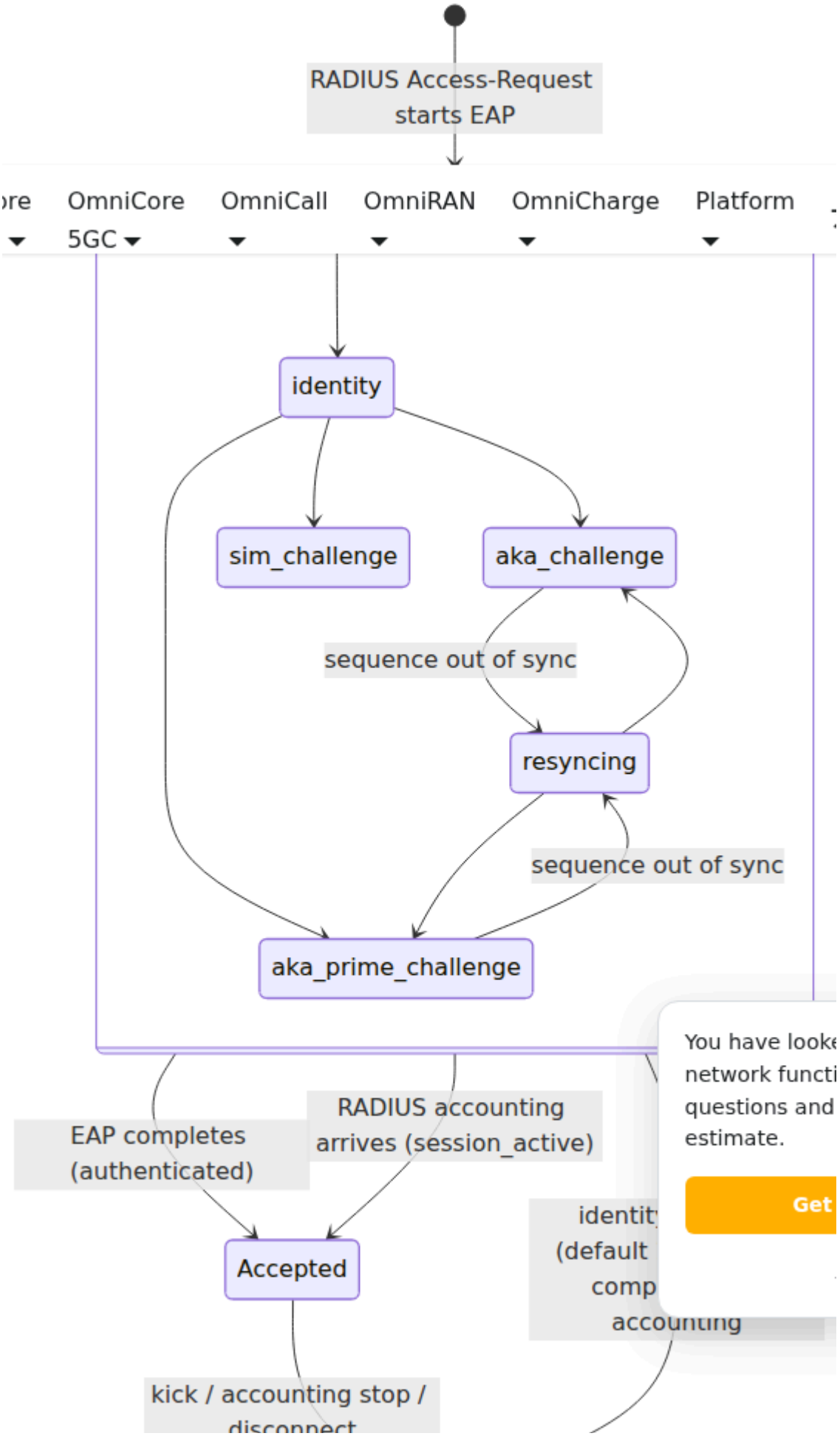
身份验证状态生命周期

会话具有身份验证状态。状态源自会话的 EAP 阶段（`eap_phase`）。

- 当 `eap_phase` 是进行中的身份验证阶段时，会话处于 身份验证中。进行中的阶段包括 `identity`、`aka_challenge`、`aka_prime_challenge`、`sim_challenge` 和 `resyncing`。
- 一旦会话经过身份验证，或一旦处于活动状态（`session_active`），会话为 已接受。

接收 RADIUS 计费的会话被提升为活动状态。计费仅在已经经过身份验证的订阅者上流动。因此，计费的到达证明订阅者处于活动状态，因此 OmniTWAG 将会话标记为已接受。

保持在进行中的身份验证阶段的会话会被清除。如果会话开始身份验证，但从未完成且从未接收计费，OmniTWAG 会在超时后拆除会话。这防止了半完成的身份验证尝试��远停留在身份验证中。默认超时为 10 分钟。


RADIUS Access-Request starts EAP
ore
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
identity
sim_challenge
aka_challenge
sequence out of sync
resyncing
sequence out of sync
aka_prime_challenge
EAP completes (authenticated)
RADIUS accounting arrives (session_active)
Accepted
kick / accounting stop /
You have looke
network functi
questions and
estimate.
Get

会话模型

会话具有以下字段。形状是开放的，因此会话也可能携带其他 GTP 或计费键。以下每个字段都呈现为可打印字符串、整数或列表。二进制值呈现为可打印字符串或十六进制。IP 元组呈现为点分四个字符串。

字段	类型	描述
`identity`	字符串	会话身份。这是 IMSI 或用户名。OmniTWAG 通过此值标识会话。
`imsi`	字符串	订阅者 IMSI。
`mac_address`	字符 `??`	UE MAC 地址。
`ue_ip`	字符串	分配的 UE IP 地址，作为点分四个字符串。
`nas_ip`	字符串	AP 源 IP（RADIUS `NAS-IP-Address`）。
`called_station_id`	字符串	接入点标识符（RADIUS `Called-Station-Id`）。
`calling_station_id`	字符串	客户端 MAC（RADIUS `Calling-Station-Id`）。
`acct_session_id`	字符串	RADIUS 计费会话 ID。OmniTWAG 通过此键合并计费记录。
`eap_phase`	字符串	当前 EAP 阶段。此值驱动身份验证状态。请参见 身份验证状态参考。
`eap_type`	字符串	使用的 EAP 方法，例如 `aka` 或 `aka_prime`。
`pgw_ip`	字符串	PGW IP 地址，作为点分四个字符串。
`bearer_id`	整数	EPS 承载 ID。
`dns_servers`	字符串列表	分配给 UE 的 DNS 服务器。
`created_at`	整数	会话创建时间，自 Unix 纪元以来的毫秒数。
`last_activity`	整数	最后活动时间，自 Unix 纪元以来的毫秒数。
`established_at`	整数	会话建立时间，自 Unix 纪元以来的毫秒数。

字段	类型	描述
`acct_input_octets`	整数	从 UE 接收的字节。合并自 RADIUS 计费。
`acct_output_octets`	整数	发送到 UE 的字节。合并自 RADIUS 计费。
`acct_input_packets`	整�?	从 UE 接收的数据包。合并自 RADIUS 计费。
`acct_output_packets`	整数	发送到 UE 的数据包。合并自 RADIUS 计费。
`acct_session_time`	整数	计费会话持续时间，以秒为单位。合并自 RADIUS 计费。
`acct_status_type`	整数	最新的计费状态。`1` 是开始，`2` 是停止，`3` 是临时。
`acct_last_update`	整数	最后计费更新时间，自 Unix 纪元以来的毫秒数。

`acct_*` 字段仅在订阅者的计费记录合并到会话后存在。

身份验证状态参考

OmniTWAG 将 `eap_phase` 值映射到显示状态。下表列出了映射关系。

`eap_phase`	显示状态	身份验证状态
`identity`	身份	身份验证中
`aka_challenge`	AKA 挑战	身份验证中
`aka_prime_challenge`	AKA' 挑战	身份验证中
`sim_challenge`	SIM 挑战	身份验证中
`resyncing`	重新同步	身份验证中
`authenticated`	已接受	已接受
`session_active`	已接受	已接受
`accepted`	已接受	已接受
`rejected`	被拒绝	被拒绝

任何其他 `eap_phase` 值都算作身份验证中。

REST API

OmniTWAG REST API 在配置的 API 端口（默认 `8444`）上通过 HTTPS 提供这些端点。��本路径为 `/sessions`。

方法	路径	操作	描述
GET	`/sessions`	index	列出经过身份验证的会话。响应是分页和可搜索的。
GET	`/sessions/{id}`	show	通过 `identity` 获取一个会话。
DELETE	`/sessions/{id}`	delete	移除会话并断开订阅者连接。
PATCH	`/sessions/{id}`	action	对会话执行操作。操作在 JSON 主体中。
PUT	`/sessions/{id}`	action	对会话执行操作。操作在 JSON 主体中。

`{id}` 路径参数是会话的 `identity`（IMSI 或用户名）。

搜索和分页

索引端点接受分页和搜索参数。

参数	类型	描述
`offset`	整数	在页面之前要跳过的会话数量。
`limit`	整数	页面中会话的最大数量。
`q`	字符串	搜索词。OmniTWAG 将该词与以下搜索字段匹配。

`q` 词与这些会话字段匹配。

搜索字段	匹配项
`identity`	会话身份（IMSI 或用户名）。
`imsi`	订阅者 IMSI。
`nas_ip`	AP 源 IP（`NAS-IP-Address`）。
`called_station_id`	接入点标识符。
`calling_station_id`	客户端 MAC。
`mac_address`	UE MAC 地址。
`ue_ip`	分配的 UE IP 地址。

操作请求主体

`PATCH` 和 `PUT` 接受一个包含一个字段的 JSON 主体。

字段	类型	必需	描述
`action`	字符串	是	要执行的操作。必须是 `kick`、`start_accounting` 或 `stop_accounting`。

有关每个操作的作用，请参见 每会话操作。

响应

状态	意义
`200 OK`	请求成功（索引、显示、删除、操作）。
`404 Not Found`	没有具有给定 `identity` 的会话。
`422 Unprocessable Entity`	操作不是 `kick`、`start_accounting` 或 `stop_accounting` 之一。

成功的操作返回一个包含 `status` 字段和 `identity` 的主体。删除返回 `{"status": "deleted", "identity": ...}`。

每会话操作

操作在实时会话上运行管理操作。将操作发送到 `/sessions/{id}` 的 `PATCH` 或 `PUT` 请求的 JSON 主体中。

操作	效果	结果 `status`
`kick`	向接入点发送 RADIUS CoA-Disconnect 并拆除会话。	`kicked`
`start_accounting`	将会话标记为活动并开始计费。这将会话提升为已接受。	`accounting_started`
`stop_accounting`	结束计费，就像接入点发送 Accounting-Stop 一样。这将拆除会话。	`accounting_stopped`

`kick` 和 `stop_accounting` 都会拆除会话。使用 `kick` 在接入点主动断开订阅者。使用 `stop_accounting` 结束会话，就像网络结束它一样。

JSON 示例

一个有流量的会话

请求：

```
GET /sessions/001010000000001
```

响应：

```
{
  "identity": "001010000000001",
  "imsi": "001010000000001",
  "mac_address": "AA:BB:CC:DD:EE:FF",
  "ue_ip": "10.45.0.2",
  "nas_ip": "192.0.2.10",
  "called_station_id": "AA-BB-CC-DD-EE-FF:WiFi-Calling",
  "calling_station_id": "11-22-33-44-55-66",
  "acct_session_id": "5A2E1F0C",
  "eap_phase": "session_active",
  "eap_type": "aka_prime",
  "pgw_ip": "192.0.2.20",
  "bearer_id": 5,
  "dns_servers": ["8.8.8.8", "8.8.4.4"],
  "created_at": 1712000000000,
  "last_activity": 1712000600000,
  "established_at": 1712000004000,
  "acct_input_octets": 1048576,
  "acct_output_octets": 5242880,
  "acct_input_packets": 1200,
  "acct_output_packets": 3400,
  "acct_session_time": 600,
  "acct_status_type": 3,
  "acct_last_update": 1712000600000
}
```

一个踢出操作

请求：

```
PATCH /sessions/001010000000001
```

```
{
  "action": "kick"
}
```

响应（`200 OK`）：

```
{
  "status": "kicked",
  "identity": "001010000000001"
}
```

一个未知操作

请求：

```
{
  "action": "reboot"
}
```

响应（`422 Unprocessable Entity`）：

```
{
  "error": "Unknown action \"reboot\" (expected
kick|start_accounting|stop_accounting)",
  "identity": "001010000000001"
}
```

Web 界面

控制面板有一个 **RADIUS** 客户端 页面。该页面列出了每个会话及其身份验证状态。该页面显示已接受、身份验证中和被拒绝会话的数量。使用该页面可以一目了然地查看实时会话。

故障排除

会话卡在 "身份验证中"

症状：会话显示身份验证中状态。状态未更改为已接受。会话停留在 EAP 阶段，例如 `identity`、`aka_challenge`、`aka_prime_challenge`、`sim_challenge` 或 `resyncing`。

可能原因：

- 订阅者开始了 EAP 身份验证，但从未完成。
- 接入点停止发送 EAP 响应。
- 订阅者身份验证失败且未重试。

如何清除：

1. 当会话进入进行中的身份验证阶段时，OmniTWAG 会启动超时。
2. 如果会话从未完成身份验证且从未接收计费，OmniTWAG 会在超时触发时拆除会话。默认超时为 10 分钟。
3. 完成份验证的会话将转移到已接受状态。接收 RADIUS 计费的会话被提升为活动状态并也转移到已接受状态。

要立即删除卡住的会话，请发送 `kick` 操作或向 `/sessions/{id}` 发送 `DELETE` 请求。

会话未显示流量

症状：会话已接受，但 `acct_*` 字段缺失或为零。

可能原因：

- 尚未收到订阅者的 RADIUS 计费记录。
- 计费记录的 `Acct-Session-Id` 与会话不同。

解决方案：

1. 确认接入点向 OmniTWAG 发送 RADIUS 计费。
2. 确认计费的 `Acct-Session-Id` 与会话的 `acct_session_id` 匹配。OmniTWAG 通过此键合并流量。
3. 等待下一个计费临时更新，然后再次读取会话。

述

用户平面和数据路径

述

OmniTWAG 将控制平面与用户平面分离。控制平面运行身份验证和会话信令。用户平面承载真实的 UE 流量。本页面描述用户平面。它解释了 OmniTWAG 如何移动 UE 数据包，如何建立和拆除 GTP-U 隧道，以及如何将下行数据包转发回 UE。

有关设置隧道数据路径的 GTP-C 会话信令，请参见 S2a / GTP-C。有关完整的系统视图，请参见 架构。

目录

- 控制平面和用户平面
- 数据路径模式
- GRE 数据路径
- GTP-U 隧道
- 下行转发
- 承载和流量流模板
- 错误处理
- MTU 和分片
- 配置
- 可观察性

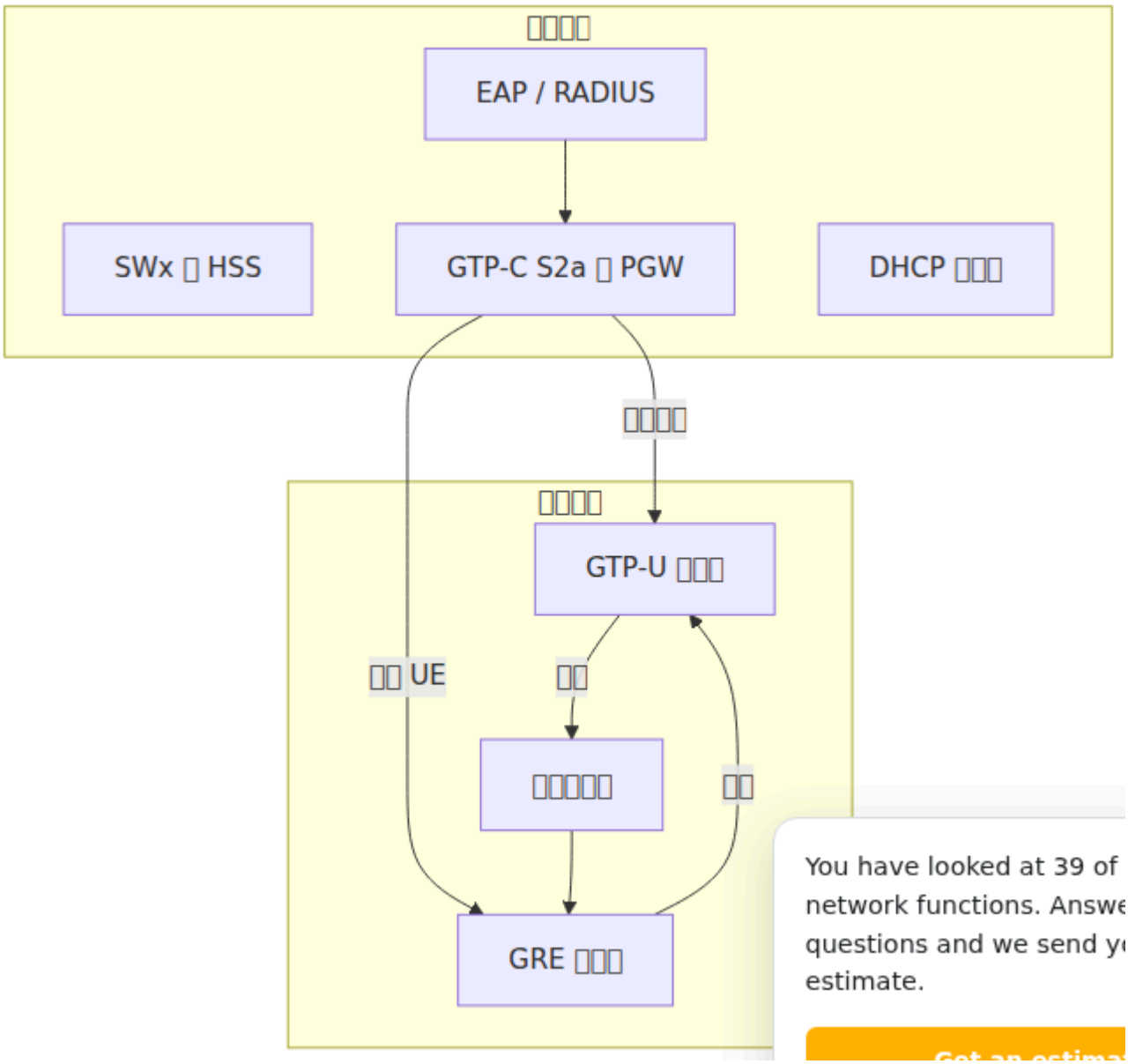
控制平面和用户平面

控制平面和用户平面作为独立进程运行。

控制平面 执行以下工作：

- ���接入点进行基于 RADIUS 的 EAP 身份验证。
- 通过 Diameter SWx 从 HSS 获取认证向量。
- 在 S2a 接口上与 PGW 进行 GTP-C 会话信令。这会创建和删除 PDN 连接和承载。请参见 S2a / GTP-C。
- 向 UE 提供 DHCP 地址。请参见 IP 地址管理。

用户平面 执行以下工作：



- 与接入点进行 GRE 隧道。
- 在 S2a-U 接口上与 PGW 进行 GTP-U 隧道，按照 3GPP TS 29.281（仅限隧道模式）。
- 向 UE 转发下行数据包。
- 将上行数据包分类到承载中。

控制平面设置用户平面的状态。在身份验证后，会话进程创建 GTP-C 会话（在隧道模式下），注册 GTP-U 隧道，并使用 GRE 服务器注册 UE。然后用户平面承载流量。

数据路径模式

OmniTWAG 根据与接入点（AP）匹配的 RADIUS 凭证解析数据平面模式。凭证上的 `breakout_type` 值选择三种模式之一。有关 `breakout_type`、`address_source` 和 `ip_pool_id` 设置，请参见 RADIUS 凭证。

模式 (`breakout_type`)	由谁分配 **UE IP**	**TWAG** 承载流量	路径
`tunneled`	PGW	是	从 AP 到 TWAG 的 GRE，然后在 S2a 上到 PGW 的 GTP-U。
`nswo`	TWAG，来自一个 IP 池	是	从 AP 到 TWAG 的 GRE，然后在 TWAG 处本地突破到互联网（NAT）。
`local`	AP 的本地网络	否	AP 拥有数据平面。TWAG 不承载数据包，也不分配 IP。

GRE 数据路径适用于 `tunneled` 和 `nswo` 突破。在这两种模式下，AP 将 UE 帧以 GRE 发送到 TWAG，TWAG 将下行数据包以 GRE 发送回 AP。不同之处在于 GRE 之后的上行：

- **`tunneled`**：TWAG 将上行数据包封装在 GTP-U 中并发送到 PGW。PGW 分配 UE IP 并应用策略和计费。
- **`nswo`**：TWAG 将上行数据包本地突破到互联网，使用 NAT 伪装规则。TWAG 从 IP 池中分配 UE IP。没有 PGW 或 GTP-C 会话。TWAG 侧的 Gy 在线计费 应用。

在 `local` 模式下，TWAG 既不分配 IP 也不承载流量，因此该 AP 不使用 GRE 或 GTP-U 路径。

隧道模式（**GTP-U** 到 **PGW**）

在隧道模式下，OmniTWAG 将 UE 流量隧道到 PGW。PGW 分配 UE IP 地址，应用策略和计费，并提供互联网访问。这是根据 3GPP TS 23.402 第 16 节的 S2a 接口。

上行路径为：

```
UE --WiFi--> AP --GRE--> TWAG --GTP-U S2a--> PGW --> Internet
```

下行路径是反向的：

```
Internet --> PGW --GTP-U S2a--> TWAG --GRE--> AP --WiFi--> UE
```

要使用此模式，请将 `gtp.enabled` 设置为 `true`，将 `gre.enabled` 设置为 `true`，并将 AP 凭证的 `breakout_type` 设置为 `tunneled`。请参见 配置。

本地突破（**NSWO**）

非无缝 WLAN 卸载（NSWO）是本地突破。UE 流量直接从 TWAG 退出到互联网。流量不经过 PGW。TWAG 分配 UE IP 并应用 NAT 伪装。这是根据 3GPP TS 23.402 第 16.2 节的规定。

OmniTWAG 有两种方式达到 NSWO：

1. 每 **AP** 突破（首选）：将 AP 凭证的 `breakout_type` 设置为 `nswo`。UE 地址来自 `ip_pool_id` 指定的 IP 池。凭证的 `address_source` 选择方式：
    - `pool` 从池中预分配一个固定地址，并立即注册 UE 以进行 GRE 下行。
    - `dhcp` 将 UE MAC 绑定到池，并让 TWAG DHCP 服务器在 DHCP DISCOVER 上分配一个地址。当地址已知时，注册 GRE 下行。
2. 全局 **NSWO** 处理程序：`Twag.NSWO.Handler` 为每个未使用 GTP 的会话提供本地突破。它通过 `nswo.enabled: true` 进行选择（并在 `gtp.enabled: false` 时应用）。该处理程序管理一个单一的本地 IP 池，并配置一个 `iptables` NAT 伪装规则。在会话建立时，TWAG 分配一个本地池地址，通过 DHCP 将其传递给 UE，并注册 UE 以进行 GRE 下行，而无需 PGW / GTP-C 会话。会话拆除时释放分配。

对于这两条路径，上行通过 NAT 伪装到互联网，下行通过 GRE 路径返回。有关寻址的信息，请参见 IP 地址管理 和 IP 池。

全局 NSWO 处理程序从 `nswo` 映射中读取这些参数：

参数	类型	默认值	描述
`enabled`	布尔值	`false`	启动全局 NSWO 处理程序并通过本地突破路由会话。
`local_pool`	字符串（CIDR）	`10.100.0.0/16`	分配 UE 地址的池。
`nat_interface`	字符串	`eth0`	NAT 伪装规则的出口接口。
`dns_servers`	列表	`["8.8.8.8", "8.8.4.4"]`	广播给 UE 的 DNS 服务器。本地突破没有 PGW PCO 来携带它们。

GRE 数据路径

GRE 服务器（`Twag.Gre.Server`）是通往接入点的用户平面端点。接入点根据 RFC 2784 和 RFC 2890 将 UE 帧以 GRE 发送到 TWAG。GRE 服务器还将下行数据包和 DHCP 响应发送回接入点。GRE 是 `tunneled` 和 `nswo` 突破的面向 AP 的路径。

GRE 服务器使用原始套接字处理 IP 协议 47（GRE），通过 `:socket.open/3` 打开。主机必须授予 `CAP_NET_RAW` 权限。如果套接字无法打开，服务器会记录错误并每 10 秒重试一次。GRE 默认关闭。要启用它，请将 `gre.enabled` 设置为 `true`。

入站 GRE 分类

GRE 服务器读取外部 IP 头，去除它，并解码 GRE 头。服务器支持两种 GRE 有效负载类型：

GRE 协议	意义	处理
`0x6558`	透明以太网桥接	提取内部以太网帧，然后按以太类型分类。
`0x0800`	IPv4	视为上行 IPv4 数据包。

对于透明以太网桥接帧，服务器读取内部以太类型：

以太类型	意义	处理
`0x0800`	IPv4	如果数据包是 DHCP，则将其传递给 DHCP 服务器。如果[illegible]是，则将其转发到上行路径。
`0x0806`	ARP	本地处理。不要转发。
`0x86DD`	IPv6	转发到上行路径。

服务器保持 UE MAC 地址到接入点 IP 的映射。服务器从每个入站帧的源 MAC 学习此映射。此映射使服务器能够将下行帧路由回正确的接入点。

上行处理

对于上行 IP 数据包，GRE 服务器读取源 IP。这是 UE IP。服务器在注册的 UE 表中查找 UE IP 以找到会话身份。它在注册表中找到会话进程并读取会话本地 TEID。然后，它异步将数据包发送到 GTP-U 服务器，使用该本地 TEID（在隧道模式下）。异步发送避免了 GRE 和 GTP-U 进程之间的死锁。

如果没有 UE IP 的会话，或者会话尚未具有 GTP-U 隧道，服务器将丢弃数据包。小于 20 字节的 IP 数据包也会被丢弃。

> 注意：对于 `nswo` 突破，本地 IP 池和 NAT 伪装将上行流量传递到互联网。GRE 服务器以与隧道模式相同的方式注册 UE 以进行下行路由。

UE 下行注册

当会话变为活动状态时，会话进程将 UE 注册到 GRE 服务器。注册保存 UE IP、UE MAC、接入点 IP 和会话身份。GRE 服务器使用这些数据来路由下行数据包。在会话拆除时，会话进程注销 UE。

GTP-U 隧道

GTP-U 服务器（`Twag.Gtp.GtpU`）是通往 PGW 的用户平面端点，处于隧道模式。它在端口 `2152` 上运行一个 UDP 套接字，并持有一个隧道表。服务器根据 3GPP TS 29.281 实现 GTPv1-U。

TEID 处理

每个隧道都有一个本地 TEID 和一个远程 TEID。远程 TEID 是 PGW TEID。TWAG 在发送到 PGW 的数据包头中使用远程 TEID。PGW 在发送到 TWAG 的数据包头中使用本地 TEID。TWAG 在隧道表中查找本地 TEID 以找到隧道。

当会话进程从 PGW 获取创建会话响应时，它注册隧道。GTP-U 服务器生成本地 TEID。TWAG 为本地 TEID 使用一个加密随机的 32 位值，并且从不使用零。这遵循 3GPP TS 29.274 中的不可预测 TEID 要求。

注册的隧道包含以下字段：

字段	描述
`local_teid`	TWAG 侧的 TEID。PGW 使用此值发送下行数据包。
`remote_teid`	PGW 侧的 TEID。TWAG 使用此值发送上行数据包。
`remote_ip`	PGW GTP-U IP 地址。
`remote_port`	PGW GTP-U UDP 端口。默认值 `2152`。
`session_id`	隧道的会话身份。

上行封装

对于每个上行 UE 数据包，GTP-U 服务器构建一个 GTPv1-U G-PDU 头，并将头放在内部 IP 数据前面。头部携带 GTP 版本（1）、协议类型（GTP）、消息类型（G-PDU，255）、有效负载长度和远程 TEID。服务器通过 UDP 套接字将数据包发送到 PGW IP 和端口。GRE 服务器在异步发送路径上调用 GTP-U 服务器，这避免了两个进程之间的死锁。

下行解封装

对于到达 GTP-U 套接字的每个数据包，服务器读取 GTP-U 头。服务器根据消息类型采取行动：

消息	类型	操作
G-PDU	255	根据本地 TEID 查找隧道，然后将内部 IP 数据包转发到下行转发器。
Echo Request	1	回复 Echo Response。
Echo Response	2	接受并忽略。
Error Indication	26	接受并忽略。
End Marker	254	接受并忽略。

对于 G-PDU，服务器去除 GTP-U 头和任何可选头字段（序列号、N-PDU 号和下一个扩展头类型）。然后服务器将内部 IP 数据包传递给 `Twag.Downlink`。请参见 下行转发。

如果 G-PDU 到达一个没有隧道的 TEID，服务器将丢弃数据包并向源发送 GTP-U 错误指示。请参见 错误处理。

下行转发

下行转发器（`Twag.Downlink`）接收来自 GTP-U 解封装的内部 IP 数据包。转发器将每个数据包发送到 UE。转发器有两条路径，首选 GRE。

GRE 路径（首选）

如果 GRE 服务器正在运行，转发器通过 GRE 发送数据包。转发器读取内部数据包的目标 IP。这是 UE IP。转发器调用 GRE 服务器。GRE 服务器在注册的 UE 表中查找 UE IP，将数据包包装在透明以太网桥接 GRE 帧中，并将其发送到接入点 IP。接入点通过 WiFi 将帧传递给 UE。

对于下行以太网帧，GRE 服务器使用 UE MAC 作为目标 MAC。服务器使用固定的本地管理 MAC（`02:00:00:00:00:01`）作为源 MAC。

原始套接字路径（回退）

如果 GRE 服务器未运行，或者 GRE 发送失败，转发器使用原始套接字。转发器注入原始 IP 数据包。内核随后将数据包从本地路由表路由到 UE。

转发器打开两个原始套接字：一个用于 IPv4，一个用于 IPv6。转发器设置 `IP_HDRINCL` 选项（对于 IPv6 设置 `IPV6_HDRINCL`）。此选项意味着转发器提供完整的 IP 头。在某些内核上，即使没有此选项，套接字仍然可以工作，因此未能设置该选项并不致命。

主机必须授予 `CAP_NET_RAW` 权限以使用原始套接字。如果套接字无法打开，转发器会记录警告并禁用该地址族。如果 GRE 和原始套接字都不可用，转发器会计数并丢弃数据包，并记录一次警告。

路径选择

`prefer_gre` 参数设置路径选择顺序。

- `prefer_gre: true`（默认）：转发器首先尝试 GRE 隧道，如果 GRE 不可用或 GRE 发送失败，则回退到原始套接字。
- `prefer_gre: false`：转发器首先尝试原始套接字进行数据包地址族，如果该原始套接字不可用，则仅使用 GRE。

数据包检查

转发器在以下情况下丢弃数据包：

- 转发器被配置禁用。
- 没有可用的套接字，并且 GRE 不可用。
- 数据包是小数据包（少于 20 字节）。
- 转发器无法读取目标 IP。

承载和流量流模板

承载管理器（`Twag.Bearer.Manager`）保存会话的承载。每个承载映射到一个 EPS 承载标识符（EBI）。一个承载保存 GTP-U TEID、流量流模板（TFT）数据包过滤器和 QoS 类（QCI）。一个承载是默认承载。这遵循 3GPP TS 23.402 第 16.1.6.1 节。

默认和专用承载

承载类型	TFT 过滤器	目的
默认	无	承载所有不匹配任何专用承载的流量。
专用	一个或多个	承载与其 TFT 数据包过滤器匹配的流量。

如果删除默认承载，承载管理器会发出完整的会话拆除信号。删除专用承载只会删除该承载。

TFT 数据包过滤器

TFT 是根据 3GPP TS 24.008 第 10.5.6.12 节定义的流量流模板。TFT 包含一个操作和一个数据包过滤器列表。每个数据包过滤器包含一个方向、一个优先级和一个组件列表。TFT 解析器支持以下组件类型：

组件	匹配
IPv4 远程地址和掩码	数据包目标地址。
IPv4 本地地址和掩码	数据包源地址。
IPv6 远程/本地地址和前缀	IPv6 地址。
协��标识符	IP 协议号。
单个本地端口 / 本地端口范围	UE 源端口。
单个远程端口 / 远程端口范围	目标端口。
服务类型 / 流量类	TOS 字节，带掩码。
安全参数索引	在 TWAN 上行中不匹配。
流标签	不匹配（仅限 IPv6）。

解析器支持 TFT 操作：创建、删除、添加过滤器、替换过滤器、删除过滤器和无操作。承载管理器在 PGW 发送更新承载请求时应用这些操作。

上行分类

上行分类器将每个上行数据包绑定到一个承载。规则如下：

1. 从数据包中读取源 IP、目标 IP、协议、端口和 TOS。
2. 将数据包与专用承载的 TFT 过滤器进行测试。
3. 过滤器仅在其所有组件匹配时才匹配。这是与逻辑。
4. 如果多个过滤器匹配，选择优先级数字最低的过滤器。优先级数字越低，优先级越高。
5. 如果没有专用承载匹配，选择默认承载。

□□□

配置

用户平面进程读取 `:omnitwag` 应用程序配置。相关块为 `gtp`、`gre` 和 `downlink`。AP 突破模式来自 RADIUS 凭证，而不是来自这些块。请参见 RADIUS 凭证。

```
config :omnitwag,
  # GTP-U 和 GTP-C 通往 PGW（S2a） - 隧道模式
  gtp: %{
    enabled: false,
    pgw_ip: "10.5.198.1",
    pgw_port: 2123,
    local_ip: "10.5.198.200",
    gtpu_port: 2152,
    echo_enabled: true
  },

  # GRE 通往接入点
  gre: %{
    enabled: false,
    local_ip: "10.5.198.200"
  },

  # 下行转发
  downlink: %{
    enabled: true,
    ip_hdr_incl: true,
    prefer_gre: true
  }
```

GTP 参数

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	启动 GTP-C 和 GTP-U 服务器。对于隧道模式设置为 `true`。
`pgw_ip`	字符串	否	`"10.5.198.1"`	GTP-C 和 GTP-U 的 PGW IP 地址。
`pgw_port`	整数	否	`2123`	PGW GTP-C UDP 端口。
`local_ip`	字符串	否	自动检测	GTP 的 TWAG 本地 IP。用作 GTP-U 数据包的源和错误指示对等地址。
`gtpu_port`	整数	否	`2152`	GTP-U 服务器的本地 UDP 端口。这是标准 GTP-U 端口。
`echo_enabled`	布尔值	否	`true`	向 PGW 运行 GTP-C 回声。请参见 S2a / GTP-C。

GRE 参数

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`false`	启动 GRE 服务器。设置为 `true` 以通过 GRE 从接入点接收 UE 帧。对于 `tunneled` 和 `nswo` 突破均需要。
`local_ip`	字符串	否	`"0.0.0.0"`	绑定 GRE 原始套接字的本地 IP。这是接入点发送 GRE 流量的目标 IP。

下行参数

参数	类型	必需	默认	描述
`enabled`	布尔值	否	`true`	启动下行转发器。当为 `false` 时，转发器丢弃所有下行数据包。
`ip_hdr_incl`	布尔值	否	`true`	在原始套接字上设置 `IP_HDRINCL` 和 `IPV6_HDRINCL` 选项，以便转发器提供完整的 IP 头。当为 `false` 时，不设置该选项。除非内核拒绝 `IP_HDRINCL`，否则请保持为 `true`。
`prefer_gre`	布尔值	否	`true`	路径选择顺序。当为 `true` 时，转发器首先尝试 GRE，然后在 GRE 不可用时回退到原始套接字。当为 `false` 时，转发器首先尝试原始套接字，仅在原始套接字不可用时使用 GRE。

主机能力

GRE 服务器和下行原始套接字路径均使用原始套接字。主机必须授予 `CAP_NET_RAW` 权限。没有此权限，原始套接字无法打开。GRE 服务器每 10 秒重试套接字。下行转发器回退或丢弃数据包，如 下行转发 中所述。

可观察性

用户平面进程保持内部统计信息。每个进程报告其自身状态，涵盖 GTP-U 隧道端点、GRE 服务器和下行转发器。有关 OmniTWAG 导出的 Prometheus 指标，请参见 指标参考。

GTP-U 状态

GTP-U 状态报告 `socket_open`、`local_ip`、`tunnel_count` 和嵌套的 `stats` 映射：

统计信息	描述
`packets_in`	在 GTP-U 套接字上接收到的数据包计数。
`packets_out`	在 GTP-U 套接字上发送的数据包计数。
`bytes_in`	接收的总字节数。
`bytes_out`	发送的总字节数。
`errors`	解码错误和未知隧道事件的计数。
`tunnel_count`	注册的隧道数量（顶级字段）。

GRE 状态

GRE 状态报告 `running`、`socket_open`、`local_ip`、`ap_tunnels`、`registered_ues` 和嵌套的 `stats` 映射：

统计信息	描述
`packets_in`	接收到的 GRE 数据包计数。
`packets_out`	发送的 GRE 数据包计数。
`bytes_in` / `bytes_out`	进出总字节数。
`dhcp_in`	从接入点接收到的 DHCP 数据包计数。
`uplink_forwarded`	转发到 GTP-U 的上行数据包计数。
`downlink_forwarded`	发送到接入点的下行数据包计数。
`unknown_protocol`	带有未知协议或以太类型的数据包计数。
`errors`	解码和发送错误的计数。
`registered_ues`	注册的 UE 数量（顶级字段）。

下行统计

统计信息	描述
`forwarded`	转发的 IPv4 数据包计数（GRE 和 IPv4 原始套接字）。
`forwarded_v6`	转发的 IPv6 数据包计数。
`dropped`	丢弃的数据包计数。
`errors`	发送错误的计数。
`bytes`	转发的总字节数。
`enabled`	转发器启用状态。
`socket_open`	原始 IPv4 套接字状态。

述