OmniTWAG Operations and Deployment Guide

Created by Omnitouch

This guide is for network operators, system administrators, and customers deploying OmniTWAG.

Table of Contents

Introduction

OmniTWAG (Trusted WiFi Access Gateway) is a standards-compliant implementation of a 3GPP TWAG that enables mobile network operators to securely offload subscriber traffic from cellular networks to WiFi access points while maintaining secure, SIM-based authentication.

The TWAG authenticates WiFi subscribers using their SIM credentials via EAP-AKA (Extensible Authentication Protocol - Authentication and Key Agreement), the same authentication mechanism used in cellular networks. This provides seamless, secure WiFi access for mobile subscribers without requiring separate WiFi passwords.

Key Benefits

For End Users:

- **Zero Configuration**: Works out of the box with compatible SIM
- **Seamless Experience**: Automatic connection like cellular
- **Secure**: Always uses encrypted WiFi (WPA2)
- **No Passwords**: SIM-based authentication

For Mobile Operators:

- **Network Capacity Relief**: Reduces load on cellular base stations
- **Controlled Offload**: Only authorized subscribers can connect
- **Improved User Experience**: WiFi typically offers higher bandwidth
- **Cost Efficiency**: WiFi infrastructure is less expensive than cellular
- **Consistent Identity**: Same IMSI used for WiFi and cellular
- **Billing Integration**: Can charge for WiFi usage if desired

For Venues/Enterprises:

- **Operator-Grade Security**: No risk of password sharing
- **Scalability**: Support thousands of users without manual provisioning
- **Simplified Management**: No need to distribute WiFi passwords

---

What is WiFi Offload?

WiFi offload allows mobile network operators to redirect subscriber data traffic from congested cellular networks to WiFi networks.

How TWAG Enables Offload

The TWAG acts as the authentication gateway between:

- **WiFi Access Points** (via RADIUS protocol)
- **Mobile Core Network** HSS/HLR (via Diameter SWx interface)

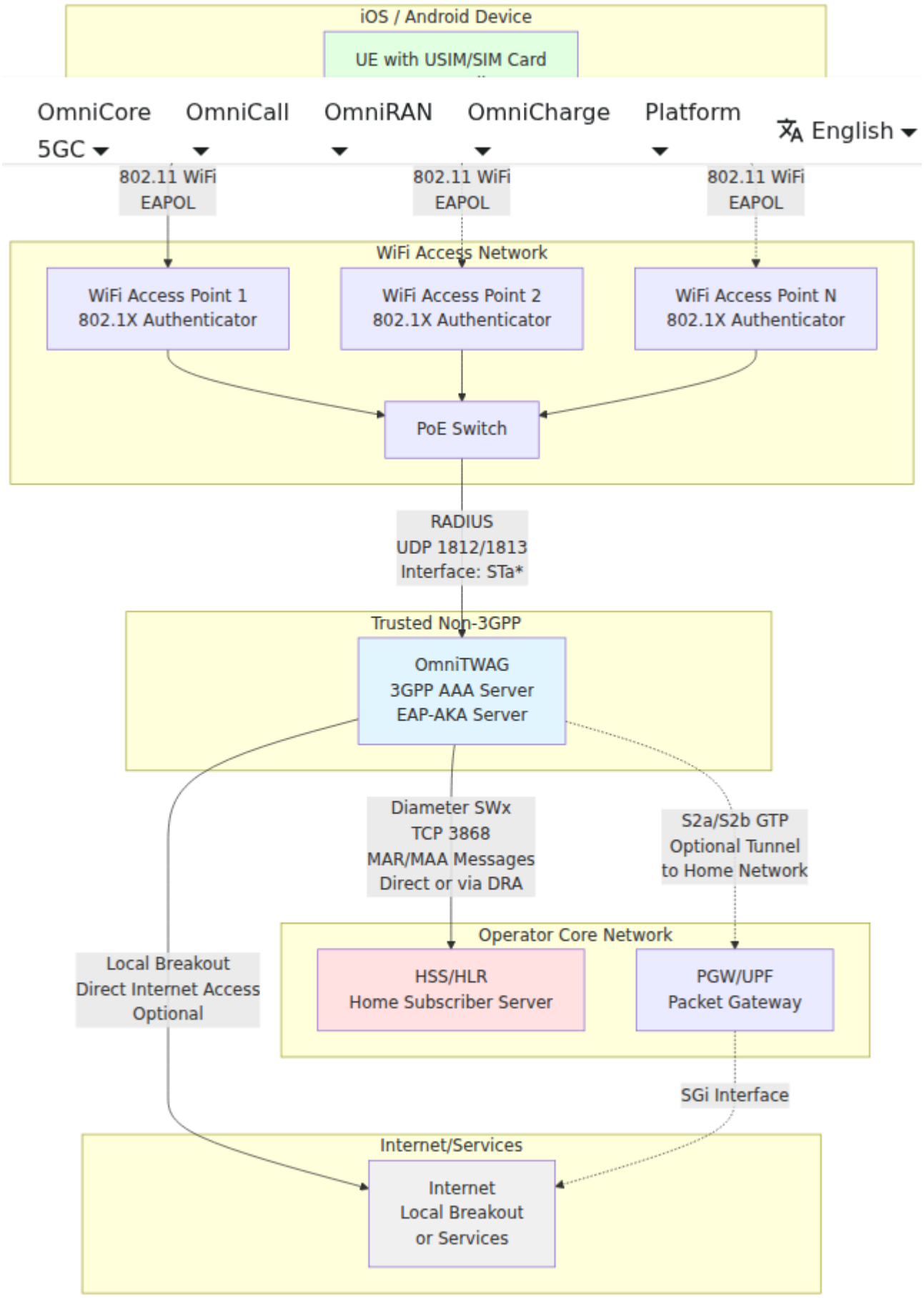
When a subscriber's device connects to a WiFi AP configured for offload:

1. The device identifies itself using its IMSI (from the SIM card)
2. The WiFi AP forwards authentication requests to the TWAG via RADIUS
3. The TWAG communicates with the operator's HSS to retrieve authentication vectors
4. EAP-AKA challenge-response authentication occurs between the device and TWAG
5. Upon successful authentication, the device is granted WiFi access
6. Optionally, traffic can be tunneled back to the mobile core or break out locally

---

Deployment Architecture

Network Topology

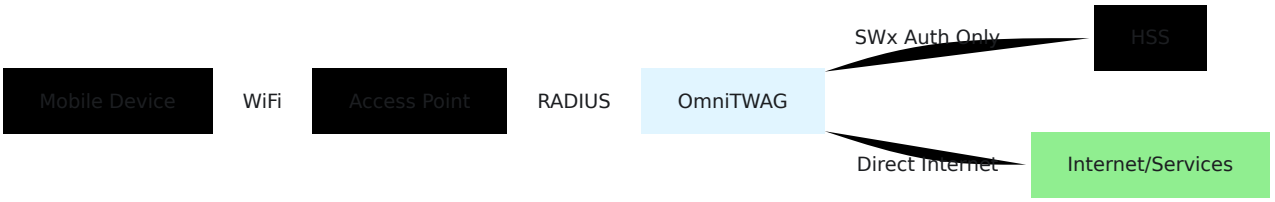


Interface Legend:

- **STa***: RADIUS/Diameter interface between WiFi AP and TWAG (non-3GPP to AAA)
- **SWx**: Diameter interface between TWAG (3GPP AAA Server) and HSS
- **S2a/S2b**: GTP tunnel interface for backhaul to home network (optional)
- **SGi**: Interface to external packet data networks (Internet)
- **802.11**: WiFi radio interface
- **EAPOL**: EAP over LAN (802.1X authentication)

Deployment Scenarios

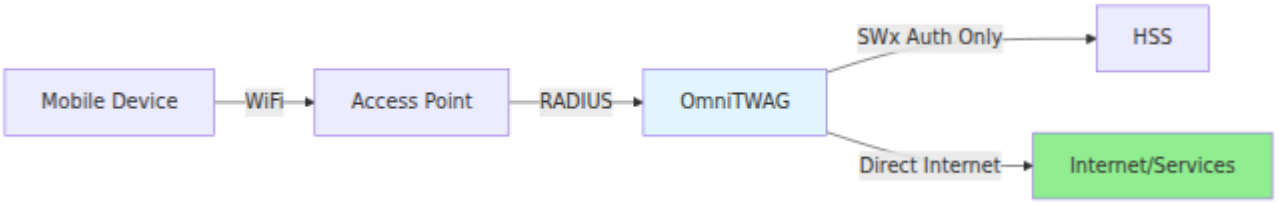
Scenario 1: Local Breakout (Recommended for Performance)



Benefits:

- Lower latency (no hairpinning to core)
- Reduced core network load
- Better user experience for high-bandwidth applications
- Cost savings on backhaul capacity

Scenario 2: Home Network Routing (GTP Tunnel)



Benefits:

- Consistent policy enforcement
- Centralized charging/billing
- Corporate VPN/security policies apply
- Seamless mobility between WiFi and cellular

SWx Connection Options

Option 1: Direct Connection to HSS

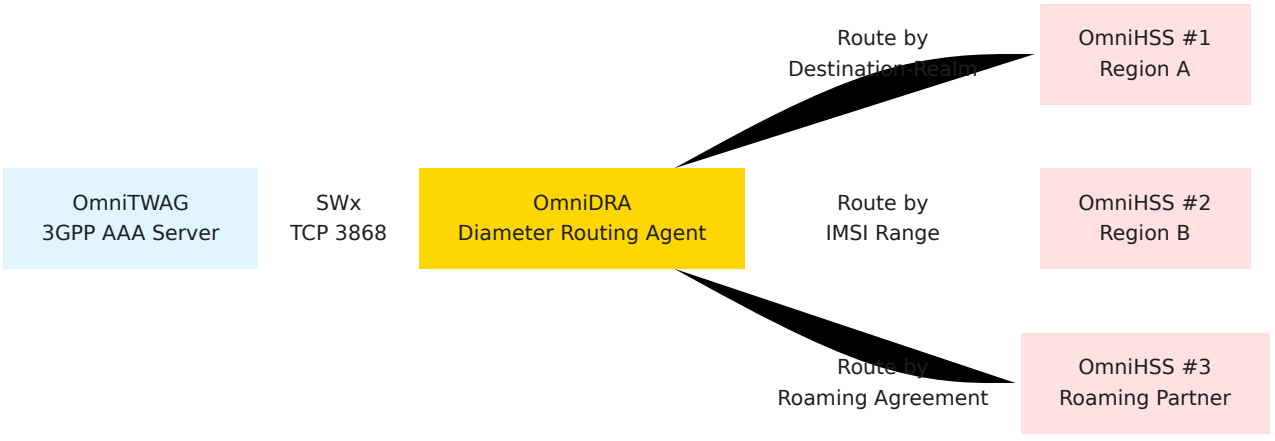
OmniTWAG 3GPP AAA Server	Direct SWx TCP 3868 MAR/MAA	OmniHSS Subscriber Database

Use Case: Simple deployments, lab environments, single HSS

Benefits:

- Lower latency (no hop through DRA)
- Simplified configuration
- Easier troubleshooting

Option 2: Via DRA (Diameter Routing Agent)



Use Case: Multi-HSS deployments, roaming scenarios, large-scale networks

Benefits:

- Centralized routing logic
- Load balancing across multiple HSS
- Roaming support (routes to home HSS)
- Redundancy and failover
- Session stickiness

Charging Flow

The TWAG can be fully integrated to send Diameter `Gy` based online charging requests to an Online Charging System (OCS).

This allows for accounting of all data consumed on WiFi, against the balance of the customer, and is delivered via the AP on RADIUS and converted to `Gy` by the TWAG and forwarded to the DRA/OCS.

In all modes, usage is traced by the TWAG metrics.

Charging Modes

The TWAG supports three online charging modes:

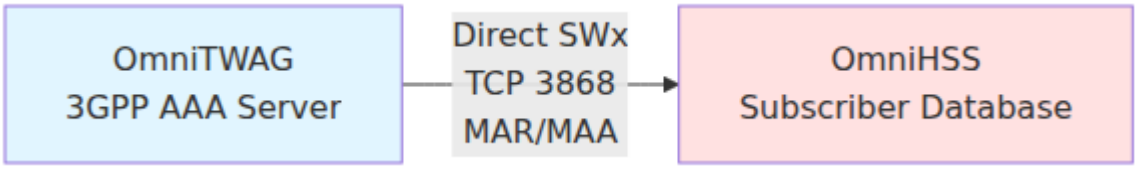
1. Charging Disabled

No credit control requests are sent. No authorization of balance is performed.

Use Cases:

- Open/free WiFi networks
- Lab/testing environments
- Networks with offline charging only (RADIUS accounting to billing)

Flow:



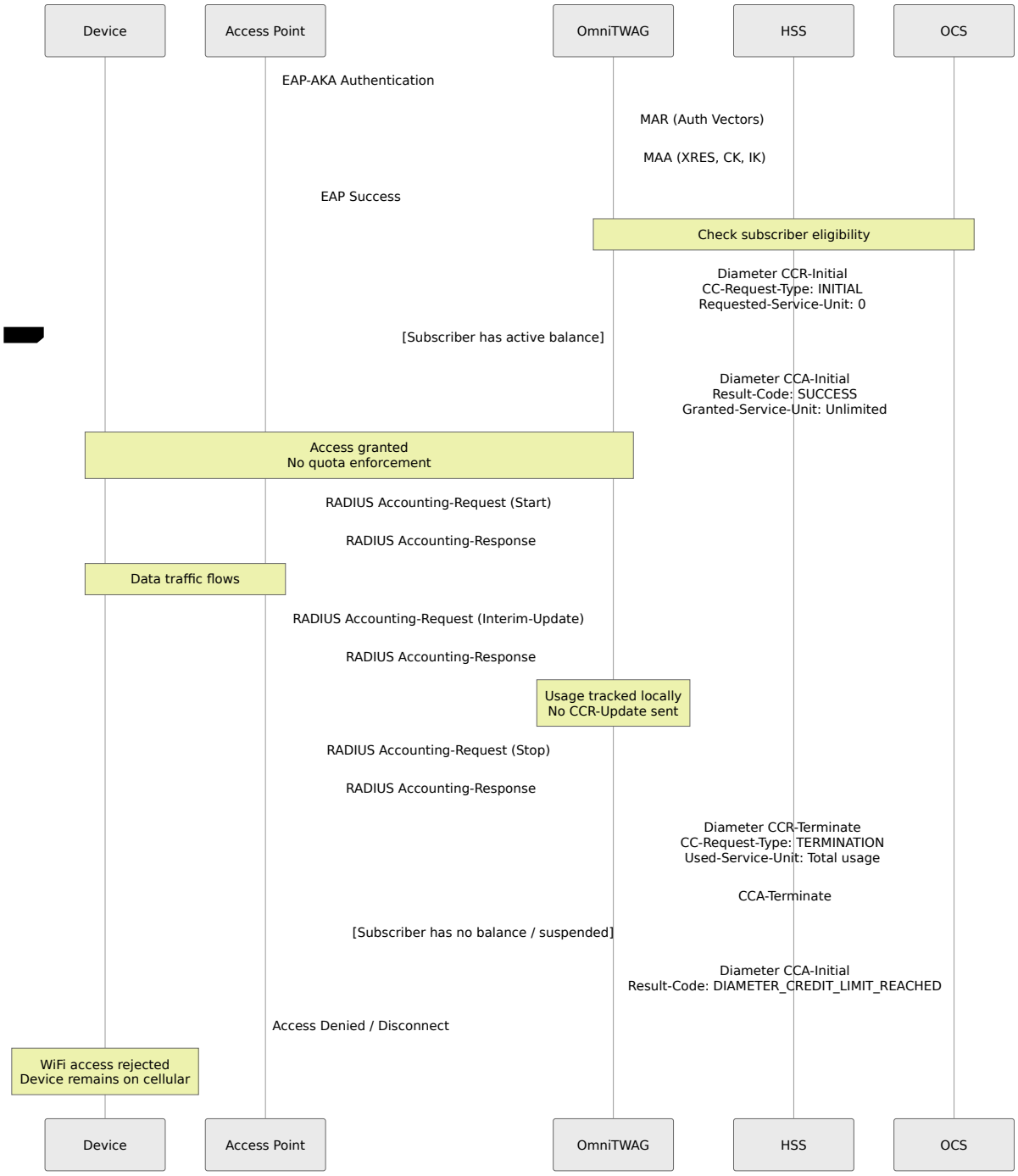
2. Authorization Only

A CCR-Initial (Credit-Control-Request) is sent to the OCS at the start of the WiFi session to validate the subscriber has balance, but the balance is not drawn down during the session.

Use Cases:

- Validate subscriber has active account/balance
- Prevent WiFi access for suspended accounts
- Check service eligibility without quota tracking
- Allow WiFi as bonus/unlimited service for paying customers

Flow:



Configuration:

- OCS is queried at session start (CCR-I) and end (CCR-T)
- No CCR-Update messages sent during session
- Subscriber authorized based on account status, not quota
- Usage reported at end of session for informational purposes only

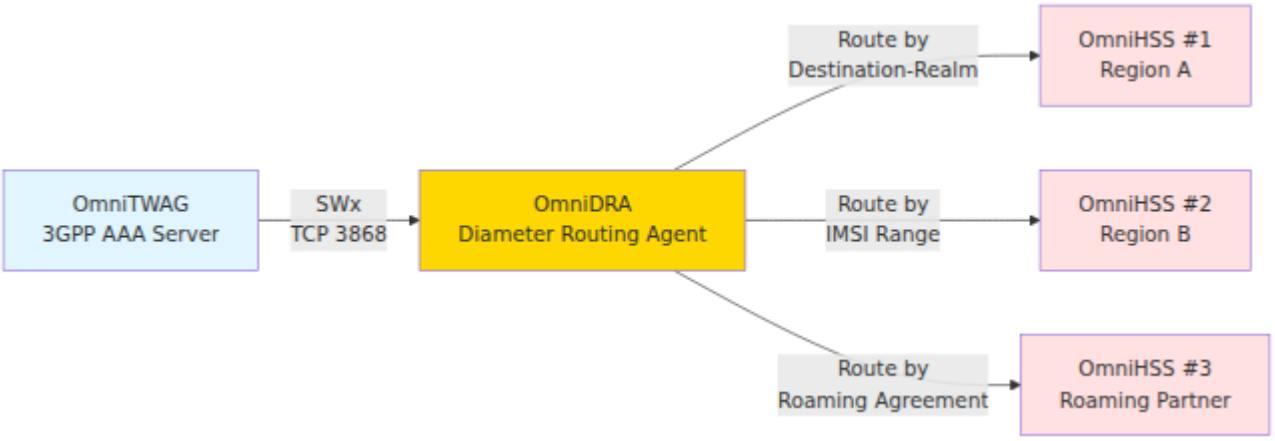
3. Fully Gy Online Charging (Full Implementation)

Standard 3GPP online charging flow is followed. All usage on WiFi is passed to the OCS for charging, with the subscriber cut off once they have exceeded their quota.

Use Cases:

- Prepaid data services
- Pay-per-use WiFi
- Quota-based plans (e.g., 10GB monthly allowance)
- Real-time charging and cutoff

Flow:

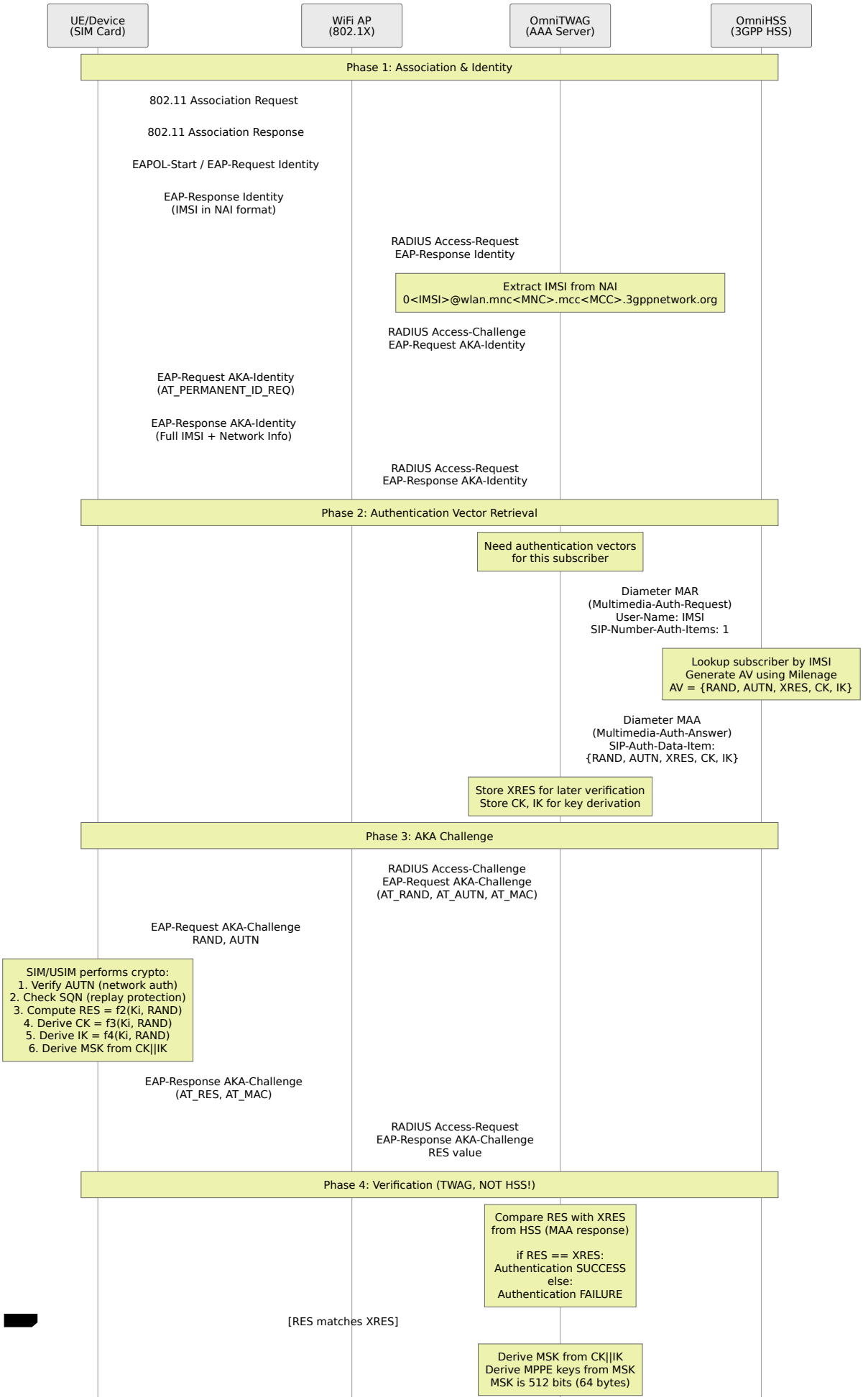


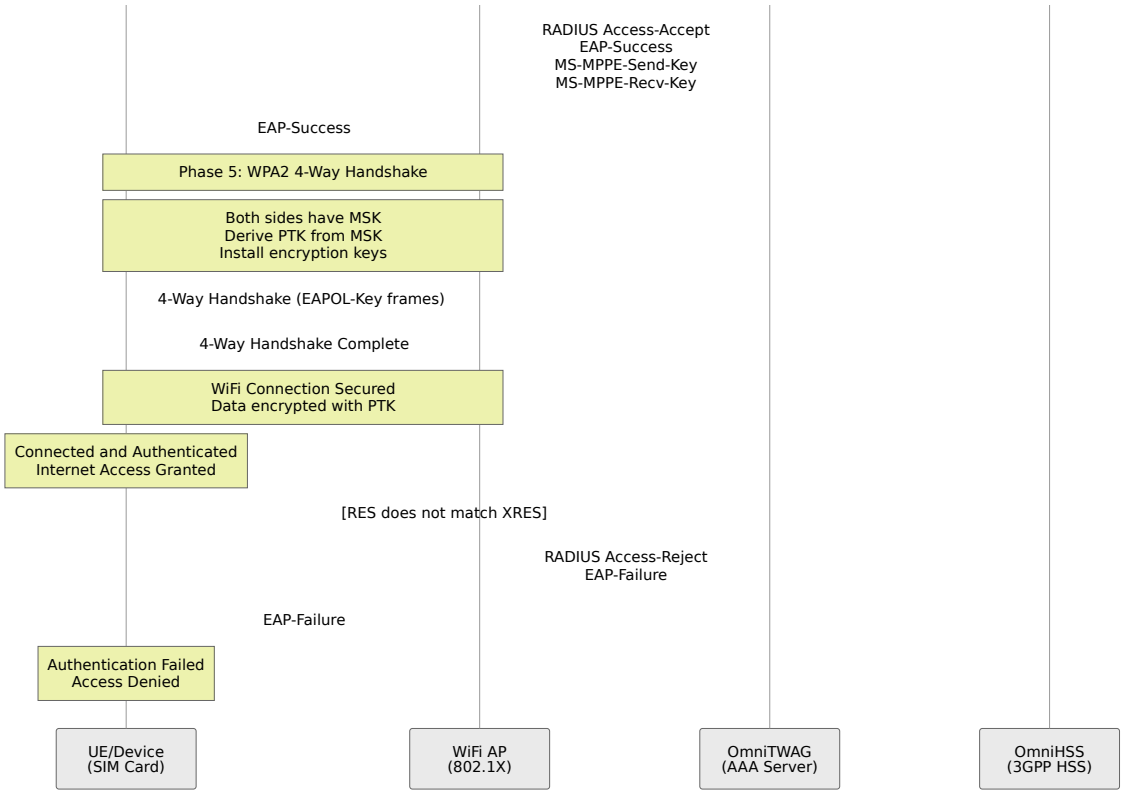
Configuration:

- OCS queried at session start (CCR-I), during session (CCR-U), and at end (CCR-T)
- Quota requested in configurable chunks (e.g., 10MB, 50MB, 100MB)
- CCR-Update triggered at configurable threshold (e.g., 80% of granted quota)
- Validity timer triggers re-authorization if quota not exhausted
- Forced disconnection when quota exhausted
- Real-time balance deduction

Authentication Flow

Complete EAP-AKA Authentication Sequence


UE/Device
(SIM Card)
WiFi AP
(802.1X)
OmniTWAG
(AAA Server)
OmniHSS
(3GPP HSS)
Phase 1: Association & Identity
802.11 Association Request
802.11 Association Response
EAPOL-Start / EAP-Request Identity
EAP-Response Identity
(IMSI in NAI format)
RADIUS Access-Request
EAP-Response Identity
Extract IMSI from NAI
0<IMSI>@wlan.mnc<MNC>.mcc<MCC>.3gppnetwork.org
RADIUS Access-Challenge
EAP-Request AKA-Identity
EAP-Request AKA-Identity
(AT_PERMANENT_ID_REQ)
EAP-Response AKA-Identity
(Full IMSI + Network Info)
RADIUS Access-Request
EAP-Response AKA-Identity
Phase 2: Authentication Vector Retrieval
Need authentication vectors
for this subscriber
Diameter MAR
(Multimedia-Auth-Request)
User-Name: IMSI
SIP-Number-Auth-Items: 1
Lookup subscriber by IMSI
Generate AV using Milenage
AV = {RAND, AUTN, XRES, CK, IK}
Diameter MAA
(Multimedia-Auth-Answer)
SIP-Auth-Data-Item:
{RAND, AUTN, XRES, CK, IK}
Store XRES for later verification
Store CK, IK for key derivation
Phase 3: AKA Challenge
RADIUS Access-Challenge
EAP-Request AKA-Challenge
(AT_RAND, AT_AUTN, AT_MAC)
EAP-Request AKA-Challenge
RAND, AUTN
SIM/USIM performs crypto:
1. Verify AUTN (network auth)
2. Check SQN (replay protection)
3. Compute RES = f2(Ki, RAND)
4. Derive CK = f3(Ki, RAND)
5. Derive IK = f4(Ki, RAND)
6. Derive MSK from CK||IK
EAP-Response AKA-Challenge
(AT_RES, AT_MAC)
RADIUS Access-Request
EAP-Response AKA-Challenge
RES value
Phase 4: Verification (TWAG, NOT HSS!)
Compare RES with XRES
from HSS (MAA response)
if RES == XRES:
Authentication SUCCESS
else:
Authentication FAILURE
[RES matches XRES]
Derive MSK from CK||IK
Derive MPPE keys from MSK
MSK is 512 bits (64 bytes)



Key Points in Authentication Flow

1. **MAR/MAA is the end of HSS communication**: After receiving the MAA (Multimedia-Auth-Answer) with XRES, the TWAG handles all subsequent verification locally.

2. **TWAG performs RES verification**: The HSS provides the expected response (XRES), but the TWAG compares it against the actual RES from the UE. The HSS is NOT involved in this comparison.

3. **Authentication happens at TWAG**: This is different from some diagrams that show HSS doing verification—in the actual 3GPP architecture, the AAA server (TWAG) performs the comparison.

Identity Format

The device responds with its permanent identity (IMSI) in NAI format:

```
505570000000000001@wlan.mnc057.mcc505.3gppnetwork.org
```

Format: `0<IMSI>@wlan.mnc<MNC>.mcc<MCC>.3gppnetwork.org`

Note - The first digit,before the IMSI is the identity, this is generally 0 but may be another single digit number for multi-IMSI SIMs / handsets.

Master Session Key (MSK)

The Master Session Key (MSK) is a 512-bit (64-byte) cryptographic key derived during EAP-AKA authentication. It serves as the root key material for securing the WiFi connection.

MSK Derivation:

1. Both UE and TWAG independently derive the same MSK
2. UE derives from CK/IK computed by SIM
3. TWAG derives from CK/IK received from HSS
4. MSK = PRF'(CK || IK, "Full Authentication", IMSI, ...)

MSK Usage:

1. **PMK Derivation**: PMK = first 256 bits (32 bytes) of MSK
2. **WPA2 4-Way Handshake**: Both UE and AP use PMK to derive PTK
3. **Data Encryption**: All WiFi data frames encrypted with Temporal Key (TK) from PTK

Why MSK is Critical:

- **Confidentiality**: Without MSK, WiFi traffic would be unencrypted
- **Integrity**: Prevents tampering with WiFi frames
- **Authentication Binding**: Links EAP authentication to WiFi encryption
- **Replay Protection**: Fresh MSK prevents replay attacks
- **Perfect Forward Secrecy**: Compromise of one MSK doesn't affect others

Resynchronization Recovery

If the device detects a sequence number mismatch (SQN out of sync), it initiates resynchronization:

1. Device computes AUTS (Authentication Token - Synchronization)
2. Sends EAP-AKA Synchronization-Failure with AT-AUTS

3. TWAG forwards AUTS to HSS
4. HSS resyncs sequence number and generates new vectors
5. Authentication retried with fresh vectors

This is transparent to the end user and requires no operator intervention.

Configuration Guide

The TWAG is configured via Elixir configuration files in the `config/` directory. The main runtime configuration is in `config/runtime.exs`.

For production deployments, configuration is centrally managed. The below is a reference only, any values changed on a production node will be lost next time the automated orchestration is run.

Diameter Configuration

Located in `config :diameter_ex`:

```
config :diameter_ex,
  diameter: %{
    # Service name for the Diameter stack
    service_name: :omnitouch_twag,

    # Local IP address to bind Diameter service
    listen_ip: "10.5.198.200",

    # Local port for Diameter connections (standard is 3868)
    listen_port: 3868,

    # Diameter Origin-Host
    host: "omnitwag",

    # Diameter Origin-Realm (matches your network realm)
    realm: "epc.mnc057.mcc505.3gppnetwork.org",

    # Diameter peers (HSS, DRA, AAA servers)
    peers: [
      %{
        # Peer Diameter Origin-Host
        host: "omni-hss01.epc.mnc057.mcc505.3gppnetwork.org",

        # Peer Diameter Origin-Realm
        realm: "epc.mnc057.mcc505.3gppnetwork.org",

        # Peer IP address (can be HSS directly or DRA)
        ip: "10.179.2.140",

        # Peer port (standard is 3868)
        port: 3868,

        # Use TLS for transport security
        tls: false,

        # Transport protocol (:diameter_tcp or :diameter_sctp)
        transport: :diameter_tcp,

        # Initiate connection to peer (true) or wait for peer to
connect (false)
        initiate_connection: true
      }
```

```
    ]
  }
```

Realm Format follows 3GPP TS 23.003:

```
epc.mnc<MNC>.mcc<MCC>.3gppnetwork.org
```

Where:

- MNC = Mobile Network Code (e.g., 057)
- MCC = Mobile Country Code (e.g., 505 for Australia)

Note on DRA Usage: To use OmniDRA, configure the peer IP to point to the DRA instead of directly to the HSS. The DRA will then route messages to the appropriate HSS based on routing rules (Destination-Realm, IMSI range, etc.).

RADIUS Configuration

Located in `config :omnitwag`:

```
config :omnitwag,
  radius_config: %{
    # List of allowed source IP subnets for RADIUS clients
    # Empty list = allow all (not recommended for production)
    allowed_source_subnets: ["10.7.15.0/24", "192.168.1.0/24"],

    # Shared secret for RADIUS clients
    # All APs must use this secret
    secret: "YOUR_STRONG_SECRET_HERE"
  }
```

Security Best Practices:

- Use strong RADIUS shared secrets (20+ characters)
- Configure `allowed_source_subnets` to restrict AP access
- Use firewall rules to further restrict access to ports 1812/1813

Example subnet configuration:

```
allowed_source_subnets: ["10.7.15.0/24", "192.168.1.0/24"]
```

If empty, all sources are allowed (only suitable for lab/testing).

Prometheus Monitoring Configuration

Located in `config :omnitwag`:

```
config :omnitwag,
  prometheus: %{
    # Port for Prometheus metrics endpoint
    port: 9568
  }
```

Access metrics at: `http://<twag-ip>:9568/metrics`

Port Summary

Port	Protocol	Purpose
1812	UDP	RADIUS Authentication
1813	UDP	RADIUS Accounting
3868	TCP	Diameter (SWx to HSS/DRA)
443	TCP	HTTPS Web Dashboard
8444	TCP	HTTPS REST API
9568	TCP	Prometheus Metrics

Access Point Setup

Supported Access Points

OmniTWAG works with any WiFi AP that supports:

- **WPA2-Enterprise** (802.1X authentication)
- **RADIUS client** functionality
- **EAP-AKA** authentication method

Tested platforms: Cisco Aironet, Aruba, Ubiquiti UniFi, Ruckus, hostapd-based APs

General AP Configuration Requirements

1. **WPA2-Enterprise (802.1X)** security mode
2. **RADIUS server** pointing to TWAG IP address
3. **RADIUS authentication port**: 1812
4. **RADIUS accounting port**: 1813 (optional but recommended)
5. **RADIUS shared secret**: Must match TWAG configuration
6. **EAP method**: EAP-AKA (or "All")

Cisco AP Configuration Example

CLI Configuration:

```
! Configure RADIUS server
radius-server host 10.5.198.200 auth-port 1812 acct-port 1813 key
YOUR_SHARED_SECRET

! Configure SSID with 802.1X
dot11 ssid OPERATOR-WIFI
   vlan 10
   authentication open eap eap_methods
   authentication network-eap eap_methods
   authentication key-management wpa version 2

! Associate SSID with radio interface
interface Dot11Radio0
   encryption mode ciphers aes-ccm
   ssid OPERATOR-WIFI
```

Web Interface:

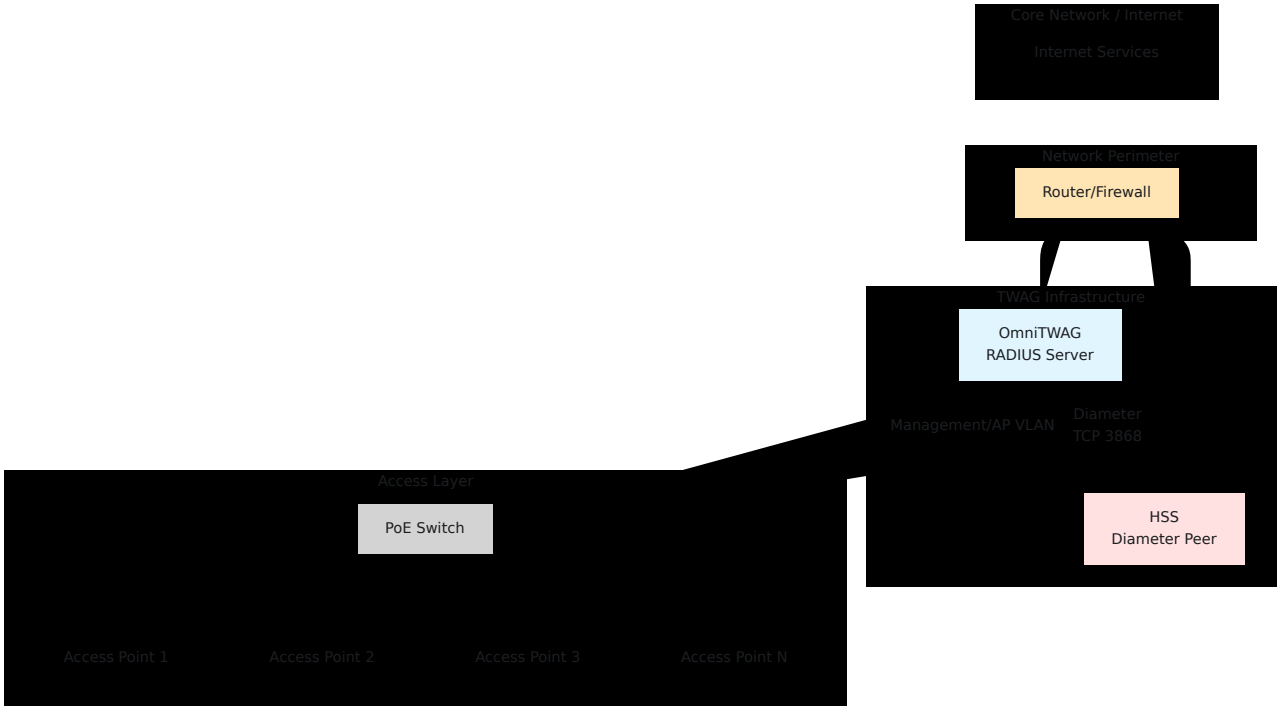
1. Navigate to **Security** → **AAA** → **RADIUS Server**
2. Add RADIUS server: `10.5.198.200:1812` with shared secret
3. Navigate to **WLAN** configuration
4. Set Security to **WPA2-Enterprise**
5. Set EAP method to **EAP-AKA** or **All**
6. Assign RADIUS server group

hostapd Configuration Example

For Linux-based APs (OpenWrt, embedded systems):

```
# /etc/hostapd/hostapd.conf

interface=wlan0
driver=nl80211
ssid=OPERATOR-WIFI

# WPA2-Enterprise
wpa=2
wpa_key_mgmt=WPA-EAP
wpa_pairwise=CCMP
ieee8021x=1

# RADIUS configuration
auth_server_addr=10.5.198.200
auth_server_port=1812
auth_server_shared_secret=YOUR_SHARED_SECRET

acct_server_addr=10.5.198.200
acct_server_port=1813
acct_server_shared_secret=YOUR_SHARED_SECRET

# EAP configuration
eap_server=0

# Hotspot 2.0 (Optional - for automatic offload)
interworking=1
internet=1
anqp_3gpp_cell_net=505,057
domain_name=wlan.mnc057.mcc505.3gppnetwork.org
nai_realm=0,wlan.mnc057.mcc505.3gppnetwork.org,0,21[2:1][5:7]
roaming_consortium=505057
hs20=1
```

Network Architecture Best Practices



Important: Place APs and TWAG on trusted network segments. Use firewall rules to:

- Allow only APs to reach TWAG ports 1812/1813
- Allow TWAG to reach HSS port 3868
- Restrict management access to TWAG dashboard (port 443)

Hotspot 2.0 Integration

Hotspot 2.0 (Passpoint) Overview

Hotspot 2.0 (also called Passpoint or 802.11u) is a WiFi Alliance standard that enables automatic, secure WiFi network discovery and connection without user interaction. It's the key technology for seamless WiFi offload.

Key Features:

- **Automatic Network Discovery**: Device finds compatible networks based on criteria

- **Automatic Authentication**: Uses SIM credentials (EAP-AKA) without user input
- **Encrypted Initial Association**: OSEN (OSU Server-only Authentication) for secure provisioning
- **Roaming Agreements**: Supports visited networks (like cellular roaming)
- **Prioritization**: Device prefers operator-owned networks

Hotspot 2.0 AP Configuration

Requirements for AP:

1. **802.11u Support**: ANQP query/response capability
2. **WPA2-Enterprise**: 802.1X authentication
3. **EAP-AKA Support**: Must support EAP-AKA method
4. **ANQP Configuration**: Advertise correct operator information

Example Configuration (hostapd-based AP):

```
# Hotspot 2.0 / Passpoint Configuration
interworking=1
internet=1
asra=0
esr=0
uesa=0

# ANQP Configuration
anqp_3gpp_cell_net=505,057
domain_name=omnitouchns.com,wlan.mnc057.mcc505.3gppnetwork.org

# NAI Realm configuration
nai_realm=0,wlan.mnc057.mcc505.3gppnetwork.org,0,21[2:1][5:7]
# Format: <encoding>,<realm>,<eap-method>[auth-id:auth-val]
# 21 = EAP-AKA
# 2:1 = Credential Type: SIM
# 5:7 = Tunneled EAP Method: None (direct EAP-AKA)

# Roaming Consortium
roaming_consortium=505057
# MCC=505 (USA), MNC=057 (operator specific)

# Venue Information (optional)
venue_group=1
venue_type=8
venue_name=eng:Operator Public WiFi

# WPA2-Enterprise Configuration
wpa=2
wpa_key_mgmt=WPA-EAP
rsn_pairwise=CCMP
ieee8021x=1

# RADIUS Configuration (points to OmniTWAG)
auth_server_addr=10.5.198.200
auth_server_port=1812
auth_server_shared_secret=YOUR_SHARED_SECRET

acct_server_addr=10.5.198.200
acct_server_port=1813
acct_server_shared_secret=YOUR_SHARED_SECRET

# SSID Configuration
```

```
ssid=OperatorWiFi
utf8_ssid=1

# Hotspot 2.0 Indication
hs20=1
hs20_oper_friendly_name=eng:Operator WiFi Network
```

Automatic Offload Behavior

How Automatic Offload Works:

1. Device with Passpoint profile performs periodic WiFi scan
2. Sends ANQP query to detected APs
3. If ANQP response matches profile (MCC/MNC, roaming consortium):
   - Priority is HIGH (home network) or MEDIUM (roaming partner)
4. If priority ≥ threshold and signal > minimum:
   - Automatic EAP-AKA authentication
5. If authentication successful and priority > current connection:
   - Switch to WiFi, disconnect cellular data
6. Monitor signal quality and maintain connectivity

Priority Factors:

1. **Home vs. Roaming**: Home network (MCC/MNC match) preferred over roaming
2. **Signal Strength**: Stronger signal preferred
3. **Security**: WPA2-Enterprise preferred over open/WPA2-PSK
4. **Policy**: Operator can configure preferred networks
5. **User Override**: User can manually disable WiFi or prefer cellular

---

Monitoring and Management

Web Dashboard

Access the real-time monitoring dashboard at: `https://<twag-ip>/`

Features:

- **RADIUS Clients View**: Active subscribers, authentication status, session details
- **Access Points View**: Connected APs, client counts, SSID information
- **Client Usage View**: Accounting data, session time, data usage
- **Diameter Peers View**: HSS/DRA connection status

Prometheus Integration

Configure Prometheus to scrape TWAG metrics:

```yaml
# prometheus.yml
scrape_configs:
  - job_name: 'omnitwag'
    static_configs:
      - targets: ['10.5.198.200:9568']
    metrics_path: '/metrics'
    scrape_interval: 15s
```

Available Metrics:

RADIUS Server Metrics:

- `radius_access_request_count` - Total RADIUS Access-Request packets received
- `radius_access_accept_count` - Total Access-Accept packets sent
- `radius_access_reject_count` - Total Access-Reject packets sent
- `radius_access_challenge_count` - Total Access-Challenge packets sent
- `radius_accounting_request_count{status_type}` - Total Accounting-Request packets (tagged by status: start, stop, interim_update, accounting_on, accounting_off)
- `radius_active_clients_count` - Currently authenticated clients (polled every 5 seconds)
- `radius_access_points_count` - Registered access points (polled every 5 seconds)

EAP-AKA Authentication Metrics:

- `eap_aka_identity_count` - EAP-AKA Identity exchanges
- `eap_aka_challenge_count` - EAP-AKA Challenge exchanges
- `eap_aka_sync_failure_count` - Synchronization failures (SQN resync events)
- `eap_aka_auth_success_count` - Successful authentications
- `eap_aka_auth_reject_count` - Rejected authentications

Diameter Protocol Metrics:

- `diameter_message_count{application, command, direction}` - Total Diameter messages (tagged by application, command type, and direction)

Erlang VM Memory Metrics:

- `vm_memory_total` - Total amount of memory allocated (bytes)
- `vm_memory_processes` - Memory used by Erlang processes (bytes)
- `vm_memory_processes_used` - Memory used by Erlang processes excluding unused allocated memory (bytes)
- `vm_memory_system` - Memory used by the Erlang runtime system (bytes)
- `vm_memory_atom` - Memory used by atoms (bytes)
- `vm_memory_atom_used` - Memory used by atoms excluding unused allocated memory (bytes)
- `vm_memory_binary` - Memory used by binaries (bytes)
- `vm_memory_code` - Memory used by loaded code (bytes)
- `vm_memory_ets` - Memory used by ETS tables (bytes)

Erlang VM System Metrics:

- `vm_system_info_process_count` - Current number of Erlang processes
- `vm_system_info_port_count` - Current number of ports
- `vm_system_info_atom_count` - Current number of atoms
- `vm_system_info_schedulers` - Number of scheduler threads
- `vm_system_info_schedulers_online` - Number of schedulers currently online

Erlang VM Scheduler Metrics:

- `vm_statistics_run_queue` - Total length of all run queues
- `vm_total_run_queue_lengths_total` - Total length of all run queues (total schedulers)
- `vm_total_run_queue_lengths_cpu` - Total length of CPU scheduler run queues
- `vm_total_run_queue_lengths_io` - Total length of IO scheduler run queues

Metric Collection:

- RADIUS and EAP-AKA metrics are emitted in real-time as events occur
- Active clients and access points counts are polled every 5 seconds
- VM metrics are polled every 5 seconds from the Erlang runtime
- All metrics are exposed in Prometheus format at `http://<twag-ip>:9568/metrics`

Logging

The TWAG uses Elixir's Logger for structured logging.

View Logs (systemd):

```
# Real-time log tail
journalctl -u twag -f

# Last 100 lines
journalctl -u twag -n 100

# Logs since last boot
journalctl -u twag -b

# Logs for specific time range
journalctl -u twag --since "2025-10-12 10:00:00" --until "2025-10-12 11:00:00"
```

Key Log Messages:

- `RADIUS server listening on port 1812` - Server started
- `From {IP}: Access-Request received` - RADIUS request from AP
- `Phase 1: Identity Response` - Initial EAP identity
- `Phase 2: AKA Challenge` - Challenge sent to device
- `Authentication ACCEPTED` - Successful authentication
- `Authentication REJECTED` - Failed authentication
- `Registered AP: {IP}` - New AP detected

Performance Benchmarks

Control-plane authentication and signalling throughput (authentications per second, codec latency, and per-operation memory) is measured and published in the Performance Benchmarks reference. The figures cover the per-authentication cryptographic cost (Milenage vector generation, the EAP-AKA' key hierarchy, ERP fast re-authentication) and signalling-plane codec throughput (EAP and WLCP encode/decode).

Troubleshooting

Authentication Failures

Symptom: Client cannot connect to WiFi

Diagnostic Steps:

1. Check TWAG logs: `journalctl -u twag -f`
2. Verify RADIUS shared secret matches between AP and TWAG
3. Confirm RADIUS packets reaching TWAG: `tcpdump -i eth0 port 1812`
4. Check subscriber provisioning in HSS/configuration

Common Causes:

- Incorrect RADIUS shared secret
- Firewall blocking UDP 1812/1813

- RES/XRES mismatch (wrong SIM Ki or HSS configuration)
- Sequence number (SQN) out of sync (should auto-recover via resync)
- Network connectivity issues between AP and TWAG

Diameter Connection Issues

Symptom: Diameter peer not connecting to HSS/DRA

Diagnostic Steps:

1. Verify network connectivity: `telnet <hss-ip> 3868`
2. Check Diameter configuration (Origin-Host, Origin-Realm, peer IP)
3. Review HSS/DRA logs for connection attempts
4. Verify firewall allows TCP 3868

Common Causes:

- Incorrect peer IP/port in configuration
- Firewall blocking TCP 3868
- Origin-Host/Realm mismatch
- HSS/DRA not accepting connection from TWAG

Performance Issues

Symptom: Slow authentication (>5 seconds)

Diagnostic Steps:

1. Check HSS response time
2. Measure network latency: `ping <hss-ip>`, `mtr <hss-ip>`
3. Monitor TWAG resource usage: `top`, `htop`
4. Review Diameter request timeout settings

Common Causes:

- HSS query timeout or slow response
- High network latency

- TWAG resource exhaustion (CPU/memory)
- Too many concurrent authentications

Debug Tools

Packet Capture

```
# Capture RADIUS traffic
tcpdump -i eth0 -n port 1812 or port 1813 -w radius.pcap

# Capture Diameter traffic
tcpdump -i eth0 -n port 3868 -w diameter.pcap

# Capture from specific AP
tcpdump -i eth0 -n host 10.7.15.72 and port 1812 -w radius-
ap1.pcap
```

Analyze with Wireshark (supports RADIUS and Diameter dissectors).

Interactive Console

Attach to running TWAG for live debugging:

```
# Remote shell to running TWAG
iex --sname debug --remsh twag@hostname --cookie <cookie>
```

From IEx console:

```
# List all authenticated clients
CryptoState.keys()

# Get specific client state
CryptoState.get("050533805790000 1867@wlan.mnc057.mcc505.3gppnetwork.c

# List all APs
APState.list()

# List accounting sessions
ClientUsage.list()
```

Common Error Messages

Error Message	Meaning	Solution
`Message-Authenticator validation failed`	Shared secret mismatch	Verify RADIUS secret matches on AP and TWAG
`RES verification failed: expected &lt;XRES>, got &lt;RES>`	Authentication response incorrect	Check SIM Ki, verify HSS provisioning
`Diameter peer connection timeout`	Can't reach HSS	Check network, firewall, HSS configuration
`Failed to decode EAP message`	Malformed EAP packet	Check AP firmware, may need AP update
`Unknown EAP-AKA subtype`	Unsupported EAP-AKA message	Device using non-standard EAP-AKA variant
`Sequence number synchronization required`	SQN out of sync	Normal, device will resync automatically

Standards Compliance

OmniTWAG implements the following 3GPP and IETF specifications:

- **3GPP TS 23.402**: Architecture enhancements for non-3GPP accesses
- **3GPP TS 24.302**: Access to EPC via non-3GPP access networks
- **3GPP TS 29.273**: Diameter-based SWx/SWm interfaces
- **3GPP TS 33.402**: Security aspects of non-3GPP accesses
- **3GPP TS 35.206**: Milenage algorithm specification
- **RFC 2865**: RADIUS Authentication
- **RFC 2866**: RADIUS Accounting
- **RFC 3579**: RADIUS support for EAP
- **RFC 4187**: EAP-AKA authentication protocol
- **RFC 5448**: EAP-AKA' (enhanced version)

Measured control-plane performance against these protocols is published in the Performance Benchmarks reference.

Summary

OmniTWAG, created by Omnitouch, provides a complete, standards-compliant solution for 3GPP WiFi offload:

1. **Flexible Deployment**: Supports local breakout or home-routed traffic
2. **Standards-Based**: Implements 3GPP SWx, EAP-AKA, RADIUS protocols
3. **Secure Authentication**: SIM-based mutual authentication with automatic resync
4. **Strong Encryption**: MSK-derived keys provide WPA2 encryption
5. **Hotspot 2.0 Ready**: Enables fully automatic, zero-touch offload
6. **Operator Control**: Maintains identity, policy, and optionally billing
7. **Flexible Connectivity**: Direct HSS connection or via OmniDRA for routing/load balancing

OmniTWAG Performance Benchmarks

Control-plane authentication and signalling throughput of the Trusted WLAN Access Gateway.

Test environment

CPU	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz (16 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` @ `631cd77`

Throughput is reported as **ips** (iterations per second — higher is faster). **Avg** and **Median** are per-call latency in microseconds; the median is the better estimate of typical steady-state latency. **Mem/op** is the heap memory allocated per call.

Full authentication (end-to-end)

The topline figure: each scenario chains the complete per-attach sequence — EAP Identity decode, authentication-vector handling, the EAP-AKA' key hierarchy, challenge build, MAC, response decode, and MAC verification — so its throughput (**ips**) is **complete attaches, i.e. sessions, per second per core**. *Local mode* computes the authentication vector on-box (Milenage); *STa mode* receives it from the HSS over SWx/STa.

Scenario	Throughput (ips)	Avg (µs)	Median (µs)	Mem/op
Full EAP-AKA' attach — STa mode (vector from HSS)	17,062	58.61	53.26	7,048 B
Full EAP-AKA' attach — local mode (vector computed on-box)	12,211	81.89	76.75	10,560 B

Cryptography

Per-authentication cryptographic operations: SIM authentication-vector generation (Milenage), the EAP-AKA' key hierarchy (CK'/IK', session keys, MAC), ERP fast re-authentication keys, and the generic 3GPP key-derivation function.

Operation	Throughput (ips)	Avg (µs)	Median (µs)	Mem/op
EAP-AKA' build Challenge packet (encode)	206,550	4.84	1.92	456 B
WLCP key derivation (EMSK -> WLCP key)	203,706	4.91	3.11	72 B
EAP-AKA' compute MAC (HMAC-SHA-256/16)	192,032	5.21	3.77	72 B
EAP-AKA' derive CK'/IK'	185,969	5.38	3.91	328 B
3GPP generic KDF (HMAC-SHA-256)	168,479	5.94	3.81	248 B
ERP derive rMSK (per re-auth)	105,348	9.49	7.38	376 B
ERP full fast re-auth (rRK -> rIK -> rMSK)	65,456	15.28	13.12	520 B
Milenage compute_all (RES, CK, IK, AK, MAC-A)	52,195	19.16	15.63	3,544 B
EAP-AKA' derive_keys (MK -> all session keys)	35,700	28.01	25.35	1,552 B
EAP-AKA' full auth crypto (CK'/IK' + all keys)	26,840	37.26	31.61	1,808 B

Operation	What it does
EAP-AKA' build Challenge packet	Encodes the EAP-Request/AKA'-Challenge (AT_RAND, AT_AUTN, AT_KDF, AT_MAC).
WLCP key derivation	Derives the WLCP protection key from the EMSK (multi-connection mode).
3GPP generic KDF	HMAC-SHA-256 key-derivation function used across the key hierarchy.
EAP-AKA' compute MAC	HMAC-SHA-256 AT_MAC over an EAP packet, truncated to 16 bytes.
EAP-AKA' derive CK'/IK'	Transforms CK/IK into CK'/IK' bound to the access-network name.
ERP derive rMSK	Per-re-auth session key for EAP Re-authentication (RFC 6696).
ERP full fast re-auth	Full rRK → rIK → rMSK derivation chain for a fast re-auth.
Milenage compute_all	One AKA authentication vector: RES, CK, IK, AK, MAC-A.
EAP-AKA' derive_keys	Expands the master key into K_encr, K_aut, K_re, MSK, EMSK.
EAP-AKA' full auth crypto	CK'/IK' derivation plus the full session-key expansion.

Packet codecs

Signalling-plane encode/decode operations: EAP packet parsing, WLCP (TS 24.244) message handling for multi-connection mode, and the MPPE key

attribute built on every RADIUS Access-Accept.

Operation	Throughput (ips)	Avg (µs)	Median (µs)	Mem/op
WLCP decode (PDN Connection Request)	2,933,539	0.34	0.27	448 B
WLCP encode + decode round-trip	1,217,555	0.82	0.59	608 B
WLCP encode (PDN Connection Request)	1,210,184	0.83	0.40	160 B
EAP decode Identity packet	267,446	3.74	2.41	1,136 B
MPPE build VSA (Send-Key, RC4 + MD5)	89,186	11.21	7.64	1,296 B

Operation	What it does
WLCP decode	Parses a WLCP PDN Connection Request from the wire.
WLCP encode	Serialises a WLCP PDN Connection Request to the wire.
WLCP encode + decode round-trip	Encode then decode of one WLCP message.
EAP decode Identity packet	Parses an EAP Identity response (NAI extraction).
MPPE build VSA	Builds the MS-MPPE-Send-Key RADIUS VSA (RC4 + MD5) carrying the MSK.