← Overview

OmniUDR Configuration Reference

OmniUDR reads its operator configuration from the application environment key `:omniudr`, populated at boot from `config/runtime.exs`. Every runtime key has an environment-variable override and a default; the block below is the complete operator surface.

Runtime Configuration

```
import Config

config :omniudr,
  # HSS API backend
  hss_api_base_url: System.get_env("HSS_API_BASE_URL",
"https://127.0.0.1:8443"),

  # SBI configuration
  sbi_scheme: System.get_env("SBI_SCHEME", "http"),
  sbi_addr: System.get_env("SBI_ADDR", "127.0.0.22"),
  sbi_port: String.to_integer(System.get_env("SBI_PORT", "7777")),

  # NRF configuration
  nrf_uri: System.get_env("NRF_URI", "http://127.0.0.1:7777"),

  # PLMN identity
  mcc: System.get_env("MCC", "999"),
  mnc: System.get_env("MNC", "70"),

  # Prometheus metrics
  prometheus_metrics_port:
String.to_integer(System.get_env("PROMETHEUS_PORT", "9568")),

  # Heartbeat interval (ms)
  heartbeat_interval:
String.to_integer(System.get_env("HEARTBEAT_INTERVAL", "10000")),

  # Test subscribers - lab bring-up aid. JSON map of IMSI → {ki,
opc, amf}.
  test_subscribers: System.get_env("TEST_SUBSCRIBERS")

config :logger, :default_formatter,
  format: {OmniLogger.JsonFormatter, :format},
  metadata: :all
```

Parameter Table

Parameter	Type	Required	Default
`hss_api_base_url`	string (URL)	Yes	`https://127.0.0.1:84`
`sbi_scheme`	string	No	`http`
`sbi_addr`	string (IP)	No	`127.0.0.22`

Parameter	Type	Required	Default
`sbi_port`	integer	No	`7777`
`nrf_uri`	string (URL)	No	`http://127.0.0.1:777`
`mcc`	string	No	`999`
`mnc`	string	No	`70`
`prometheus_metrics_port`	integer	No	`9568`
`heartbeat_interval`	integer (ms)	No	`10000`
`test_subscribers`	string (JSON) | nil	No	`nil`

Parameter	Type	Required	Default
`default_am_nssai`	map	No	eMBB / URLLC / mIoT la slices (see below)
`dnn_snssai_map`	map	No	`%{"iot" => %{"sst" =` `3, "sd" => "000003"}`

Parameter	Type	Required	Default
`notification_max_attempts`	integer	No	`4`
`notification_backoff_ms`	integer (ms)	No	`500`

> Both `default_am_nssai` and `dnn_snssai_map` exist to bridge the 4G HSS data model (no slice concept) onto the 5G Nudr structures. Align them with the deployment's real slice plan and with the AMF/SMF served NSSAI. The AM/SM defaults (eMBB `sst 1`, URLLC `sst 2 / sd 000002`, mIoT `sst 3 / sd 000003`) are lab values.

Slice synthesis map shapes (`default_am_nssai`, `dnn_snssai_map`)

Both keys are structured maps of S-NSSAI objects. An **S-NSSAI** has two sub-fields:

Sub-field	Type	Required	Meaning
`sst`	integer (0–255)	Yes	Slice/Service Type. Standardised values: `1` = eMBB, `2` = URLLC, `3` = mIoT (per TS 23.501).
`sd`	string (6 hex digits)	No	Slice Differentiator. Distinguishes two slices with the same `sst`. Omit it for a bare-`sst` slice; when present it must match the AMF/SMF served value exactly.

`default_am_nssai` is the access & mobility NSSAI, with two lists:

List	Meaning
`defaultSingleNssais`	The S-NSSAIs the UE gets when it registers without requesting a specific slice - the network's default slice set for the subscriber.
`singleNssais`	The full set of S-NSSAIs the subscriber is allowed to use (a superset of `defaultSingleNssais`). A UE requesting an SD-qualified S-NSSAI not listed here is rejected by the AMF with 5GMM cause #62.

```
config :omniudr, :default_am_nssai, %{
  "defaultSingleNssais" => [%{"sst" => 1}, %{"sst" => 1, "sd" =>
"000001"}],
  "singleNssais" => [
    %{"sst" => 1},
    %{"sst" => 1, "sd" => "000001"},
    %{"sst" => 2, "sd" => "000002"},
    %{"sst" => 3, "sd" => "000003"}
  ]
}
```

`dnn_snssai_map` maps a **DNN string** to the single S-NSSAI its PDU sessions live on. Any DNN not present in the map falls back to eMBB (`%{"sst" => 1, "sd" => "000001"}`). Keep every S-NSSAI referenced here inside `default_am_nssai.singleNssais`, or the SMF will place the session on a slice the subscriber is not subscribed to.

```
config :omniudr, :dnn_snssai_map, %{
  "iot" => %{"sst" => 3, "sd" => "000003"},
  "internet" => %{"sst" => 1, "sd" => "000001"}
}
```

About `hss_api_base_url` (the HSS REST backend)

This is the most important operator setting. OmniUDR is a facade - it has no database. When a consumer asks OmniUDR for a subscriber's authentication subscription, AM data, SM data, or policy data, OmniUDR issues live REST requests to `hss_api_base_url` (chiefly `GET /api/subscriber/imsi/{imsi}` and `GET /api/key_set/{id}`, and for auth vectors / SQN / location the `/api/hlr/*` endpoints), then maps the HSS response into the 3GPP structure. Each HSS request has a 5-second timeout (3 seconds for the health check). If this backend is unreachable, the corresponding Nudr reads return `404`/`5xx` - see Troubleshooting.

About `test_subscribers`

`test_subscribers` populates a static, in-code subscriber set (two built-in IMSIs - well-known Milenage test vectors - plus any provided via the JSON map) that returns fixed authentication, AM, and SM data without any HSS call. It is a **lab aid** for bring-up and conformance self-tests where standing up a full HSS is not warranted. In the current build the live Nudr_DR data path always reads from the HSS backend; the test-subscriber set is not wired in as an automatic fallback when the HSS is down. Treat this key as a lab/testing surface only and leave it unset (`nil`) in any deployment backed by a real HSS. An invalid JSON value falls back to the two built-in defaults.

PLMN identity and NRF profile

`mcc` and `mnc` are advertised in the NRF NF profile as the serving PLMN. The NF capacity and priority advertised to the NRF are managed through the shared Omni5G NRF configuration and can be changed at runtime through the OAM API (`nf_capacity`, `nf_priority`).

Notification delivery

`notification_max_attempts` and `notification_backoff_ms` govern how OmniUDR delivers Nudr_DR DataChangeNotify notifications to data-change subscribers (the `subs-to-notify` service, TS 29.504 subscriptions-to-notify). When a monitored resource changes, OmniUDR posts a notification to each matching subscriber's callback URI. If a delivery fails (the callback is unreachable or returns a non-2xx status), OmniUDR retries up to `notification_max_attempts` times, waiting `notification_backoff_ms` for the first retry and doubling the delay on each subsequent retry. Once the attempts are exhausted the notification is dropped and the event is logged. Delivery is best-effort and never affects the SBI response returned to the consumer that caused the change. Both keys are application-config only (no environment-variable override): configure them under `config :omniudr, ...`.

Management and OAM API

Beyond the SBI listener, OmniUDR exposes an operator-facing HTTPS management API on port **8443** (TLS, `enable_tls: true`). A Control Panel web UI is also served over HTTPS on port **7443**.

Surface	Port	Purpose
SBI (Nudr_DR)	`sbi_port` (default `7777`)	3GPP service interface
Prometheus metrics	`prometheus_metrics_port` (default `9568`)	Metrics scrape endpoint
Management / OAM API	`8443` (HTTPS, mounted under `/api`)	Status, HSS health, subscriber proxy, OAM actions
Control Panel UI	`7443` (HTTPS)	Application resources, configuration, license, and log pages

The management API is mounted under the `/api` path prefix. It provides read-only status, an HSS connectivity check, a read-only subscriber proxy to the HSS, and OAM actions:

Method	Path	Purpose
GET	`/api/status/api`	API/service status
GET	`/api/status/nrf`	NRF registration status
GET	`/api/status/nf`	NF profile / status
GET	`/api/status/license`	License status
GET	`/api/health/hss`	HSS connectivity check (calls `GET /api/status/api` on the HSS)
GET	`/api/subscribers`	Paginated read-only subscriber list proxied from the HSS (`page`, `page_size`)
GET	`/api/subscribers/{imsi}`	Full subscriber detail proxied from the HSS
GET	`/api/oam/config`	View live runtime configuration
PATCH	`/api/oam/config/{id}`	Update `hss_api_base_url`, `nrf_uri`, `nf_capacity`, `nf_priority`, or `log_level` at runtime
POST	`/api/oam/nrf/reregister`	Force NRF re-registration
DELETE	`/api/oam/subscriber-cache/{id}`	Flush cached AMF/SMF context for a subscriber (`id=all` flushes every subscriber)
POST	`/api/oam/log_level`	Change the runtime log level

Notes on the OAM config update (`PATCH /api/oam/config/{id}`):

- Only `hss_api_base_url`, `nrf_uri`, `nf_capacity`, `nf_priority`, and `log_level` are accepted; any other key is ignored, and a request with no valid key returns 400.
- `nf_capacity` and `nf_priority` must be integers; `log_level` must be one of `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug`.
- Changing `nf_capacity` / `nf_priority` both persists the value and pushes an updated profile to the NRF.
- The subscriber proxy is read-only. If the HSS is unreachable, `/api/subscribers` and `/api/subscribers/{imsi}` return `502`; a disabled or absent subscriber returns `404`.

Logging

OmniUDR emits structured JSON logs through `OmniLogger.JsonFormatter` with full metadata. The runtime log level can be changed without a restart through the OAM API (`POST /api/oam/log_level` or `PATCH /api/oam/config/{id}` with `log_level`). OAM actions (config updates, cache flushes, re-registration) are logged at `info` with an `[OAM]` prefix; HSS request failures are logged at `warning`/`error`.

← Overview

OmniUDR Metrics

OmniUDR exposes Prometheus metrics on the metrics endpoint (`prometheus_metrics_port`, default `9568`). Metric names below use the Prometheus-exposition form (dots become underscores). Labels and their values are lower-case.

UDR / HSS Metrics

Metric	Type	Labels	Descri
`omni_udr_nrf_registration_status`	gauge	`nf_type`	NRF regist status (1 = registered not)
`omni_udr_hss_health`	gauge	-	OmniHSS connectivi (1 = up, 0 down); up each `/hea` check
`omni_udr_hss_requests_total`	counter	`endpoint`, `result`	Count of R to the HSS backend, t by request (`endpoint` `result` (`success`/
`omni_udr_hss_request_duration_ms`	distribution	`endpoint`	HSS REST latency in millisecon Buckets: 5 50, 100, 2 1000, 250

BEAM VM Metrics

Metric	Type	Description
`beam_memory_total`	gauge	Total BEAM memory in bytes
`beam_memory_processes`	gauge	Memory used by Erlang processes
`beam_memory_system`	gauge	System memory (ETS, atoms, code, binaries)
`beam_processes_count`	gauge	Number of Erlang processes
`beam_vm_uptime`	gauge	VM uptime in seconds

Useful PromQL

- HSS backend availability: `omni_udr_hss_health`
- UDR registered with NRF: `omni_udr_nrf_registration_status`
- HSS error ratio (5 min): `sum(rate(omni_udr_hss_requests_total{result="failure"}[5m])) / sum(rate(omni_udr_hss_requests_total[5m]))`
- HSS p95 latency (5 min): `histogram_quantile(0.95, sum(rate(omni_udr_hss_request_duration_ms_bucket[5m])) by (le, endpoint))`
- HSS request rate by endpoint: `sum(rate(omni_udr_hss_requests_total[1m])) by (endpoint)`
- BEAM process count: `beam_processes_count`

← Overview

OmniUDR Operations Guide

OmniUDR is the Unified Data Repository (UDR) of the Omnitouch 5G Core. It implements the Nudr_DataRepository service (TS 29.504) and exposes structured subscriber, policy, and context data to the UDM and PCF over the service-based interface (SBI).

OmniUDR does not own subscriber data. It is a thin, standards-compliant Nudr_DR facade over the **OmniHSS REST provisioning backend**: every data-repository read is translated at request time into one or more OmniHSS REST calls, and the HSS response is mapped into the 3GPP Nudr data model before being returned to the consumer. OmniUDR is the sole 5G Core network function that talks to the OmniHSS REST API directly - the UDM (and, through it, the AUSF) reach subscriber data by calling OmniUDR over standard Nudr, not by calling the HSS.

Because the HSS is the system of record, OmniUDR holds almost no persistent state. Its only local state is three small in-memory (ETS) stores: a cache of AMF and SMF registration context (used to decide whether a context write is a create or an update), a data-change subscription store holding the active Nudr_DR `subs-to-notify` subscriptions, and an application-data store holding the Nudr_DR `application-data` families (PFDs, traffic influence, BDT). None of these are persisted; all are cleared on restart. The registration-context cache can be flushed through the management API.

3GPP Role and Specification References

Specification	Relevance
TS 23.501	5G system architecture - UDR role and its relationship to UDM, PCF, and the UDM data layer
TS 29.504	Nudr_DataRepository service - resource framework, subscription data, context data, policy data
TS 29.505	Subscription data structures - authentication subscription, access & mobility, session management, SMF selection
TS 29.519	Policy data structures - AM policy data and SM policy data resource shapes
TS 29.503	Nudm services - the UDM consumer whose UEAU/SDM/UECM procedures drive UDR reads and context writes
TS 29.571	Common data types - `PatchResult`, `ProblemDetails`, PLMN and S-NSSAI encodings

---

Interfaces

Interface	Bind	Purpose
SBI (Nudr_DR)	`{sbi_scheme}://{sbi_addr}:{sbi_port}` (default `http://127.0.0.22:7777`)	Serves the `nudr-dr/v2` resources consumed by the UDM and PCF
HSS REST backend	`hss_api_base_url` (default `https://127.0.0.1:8443`)	Outbound REST calls to OmniHSS that fulfil every read/update
NRF	`nrf_uri`	NF registration and periodic heartbeat
Management / OAM API	`:8443` (HTTPS, mounted under `/api`)	Status, HSS health, subscriber proxy, and OAM actions - see Configuration Reference
Prometheus metrics	`prometheus_metrics_port` (default `9568`)	Metrics scrape endpoint - see Metrics

---

Data Sets Served

OmniUDR serves the following Nudr_DR data sets. Each is derived at read time from the OmniHSS subscriber record; the "HSS source" column names the underlying REST data the value is mapped from.

Data set	Nudr resource family	Access	Consun
Authentication subscription	`subscription-data/.../authentication-data/authentication-subscription`	Read / update	UDM (Nudm_UI
Authentication vectors	`subscription-data/.../authentication-data/authentication-vectors`	Generate (proprietary)	OmniUDM
Access & mobility (AM) data	`subscription-data/.../provisioned-data/am-data`	Read	UDM (Nudm_SI
Session management (SM) data	`subscription-data/.../provisioned-data/sm-data`	Read	UDM (Nudm_SI
SMF selection data	`subscription-data/.../provisioned-data/smf-selection-subscription-data`	Read	UDM (Nudm_SI
AMF registration context	`subscription-data/.../context-data/amf-3gpp-access`	Write	UDM (Nudm_UI
SMF registration context	`subscription-data/.../context-data/smf-registrations/{pduSessionId}`	Write	UDM (Nudm_UI
AM policy data	`policy-data/ues/{ueId}/am-data`	Read	PCF

Data set	Nudr resource family	Access	Consun
SM policy data	`policy-data/ues/{ueId}/sm-data`	Read	PCF

| Application data | `application-data/{family}/{id}` (families `pfds`, `influenceData`, `bdtData`) | Create / read / replace / delete | PCF, NEF | None - held in a local in-memory (ETS) store, not HSS-backed and not persisted |

> Application data is served by a generic in-memory store rather than the HSS. It is not persisted: a restart clears it. See SBI Endpoints → Application Data below.

SBI Endpoints

All Nudr_DR endpoints are served over HTTP at `{sbi_scheme}://{sbi_addr}:{sbi_port}` (default `http://127.0.0.22:7777`) with `Content-Type: application/json`. Failures are returned as TS 29.571 `ProblemDetails`.

The `{ueId}` path segment is a SUPI in `imsi-<digits>` form (a bare IMSI is also accepted). The `{servingPlmnId}` segment on the provisioned-data reads is validated: it must be a well-formed PLMN (5 or 6 digits) and must match the serving PLMN configured from `mcc`/`mnc`, otherwise the read is rejected with `400 MANDATORY_IE_INCORRECT`. This build serves a single record per subscriber.

Subscription Data - Authentication (TS 29.504 / 29.505)

Method	Path	Description	Success
GET	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-subscription`	Retrieve authentication subscription (method, Ki, OPc, AMF, SQN)	200
PATCH	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-subscription`	Update authentication subscription (SQN)	204, or 200 + `PatchResult` on partial failure
POST	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-vectors`	Generate authentication vectors (see note)	200

> **Non-standard endpoint.** The `authentication-vectors` POST is a proprietary Omni 5GC extension; TS 29.504/29.505 define no such resource. Vector generation and SQN/resync management are delegated to the UDR so they are handled centrally at the HSS. It is consumed by OmniUDM only; a standards-compliant third-party consumer will never call it.

Subscription Data - Provisioned (TS 29.505)

Method	Path	Description	
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/am-data`	Retrieve access & mobility subscription data	2 `M` o m `{`
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/sm-data`	Retrieve session management subscription data	2 `M` o m `{`
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/smf-selection-subscription-data`	Retrieve SMF selection subscription data	2 `M` o m `{`

Subscription Data - Context (TS 29.504)

Method	Path	Description	Success
PUT	`/nudr-dr/v2/subscription-data/{ueId}/context-data/amf-3gpp-access`	Store AMF 3GPP-access registration context	201 + `Location` (new) or 204 (update)
PUT	`/nudr-dr/v2/subscription-data/{ueId}/context-data/smf-registrations/{pduSessionId}`	Store SMF registration context	201 + `Location` (new) or 204 (update)

An empty body on either context PUT returns 400 `ProblemDetails` (`cause: MANDATORY_IE_MISSING`).

Policy Data (TS 29.519)

Method	Path	Description	Success
GET	`/nudr-dr/v2/policy-data/ues/{ueId}/am-data`	Retrieve AM policy data (`subscCats`, `praInfos`)	200
GET	`/nudr-dr/v2/policy-data/ues/{ueId}/sm-data`	Retrieve SM policy data (`smPolicySnssaiData` per DNN)	200

Application Data (TS 29.504)

Generic CRUD over the Nudr_DR `application-data` families. The `{family}` path segment is one of `pfds`, `influenceData`, or `bdtData`; the `{id}` segment is the family's resource id (`appId` / `influenceId` / `bdtReferenceId`). Records

are held in a local in-memory (ETS) store, not the HSS, and are not persisted across a restart. Any other `{family}` value falls through to the 404 catch-all.

Method	Path	Description	Success
GET	`/nudr-dr/v2/application-data/{family}`	List every resource in the family (collection GET)	200 (array)
GET	`/nudr-dr/v2/application-data/{family}/{id}`	Retrieve one resource	200, or 404 if unknown
PUT	`/nudr-dr/v2/application-data/{family}/{id}`	Create or replace a resource	201 + `Location` (new) or 200 (replace)
DELETE	`/nudr-dr/v2/application-data/{family}/{id}`	Delete a resource	204, or 404 if unknown

An empty body on the PUT returns 400 `ProblemDetails` (`cause: MANDATORY_IE_MISSING`).

Data-Change Subscriptions (TS 29.504 subscriptions-to-notify)

Method	Path	Description	Success
POST	`/nudr-dr/v2/subscription-data/subs-to-notify`	Create a data-change subscription (body names the monitored resource URIs and the notification callback URI)	201 + `Location`, or 400 on a bad body
DELETE	`/nudr-dr/v2/subscription-data/subs-to-notify/{subsId}`	Remove a data-change subscription	204, or 404 if the subscription is unknown

See Data-Change Subscriptions and DataChangeNotify for the fan-out behaviour.

Any path that does not match the resources above is answered with a 404 `ProblemDetails`.

HSS Backend REST Calls

Every SBI operation is fulfilled by one or more calls to the OmniHSS REST backend at `hss_api_base_url`. The relationship is:

Nudr_DR operation	HSS REST call(s)	Notes
Authentication subscription read	`GET /api/subscriber/imsi/{imsi}`, then `GET /api/key_set/{id}` if key material is referenced by id	Key material inline in the subscriber record skips the second call
Authentication vector generation	`POST /api/hlr/generate_auth_vectors`	Body `{imsi, auts}`; `auts` is `rand‖auts` hex on resync
SQN update (PATCH)	`POST /api/hlr/update_sqn`	Body `{imsi, sqn}`
AM / SM / SMF-selection read	`GET /api/subscriber/imsi/{imsi}`	Single subscriber fetch, mapped locally
AMF context write	`PUT /api/hlr/circuit_session/{imsi}`	Location/circuit-session update; also cached locally
SMF context write	(none)	Local ETS cache only
Policy data read (AM / SM)	`GET /api/subscriber/imsi/{imsi}`	Reuses the AM/SM mapping and re-shapes it
Health check	`GET /api/status/api`	Drives the `/health/hss` management

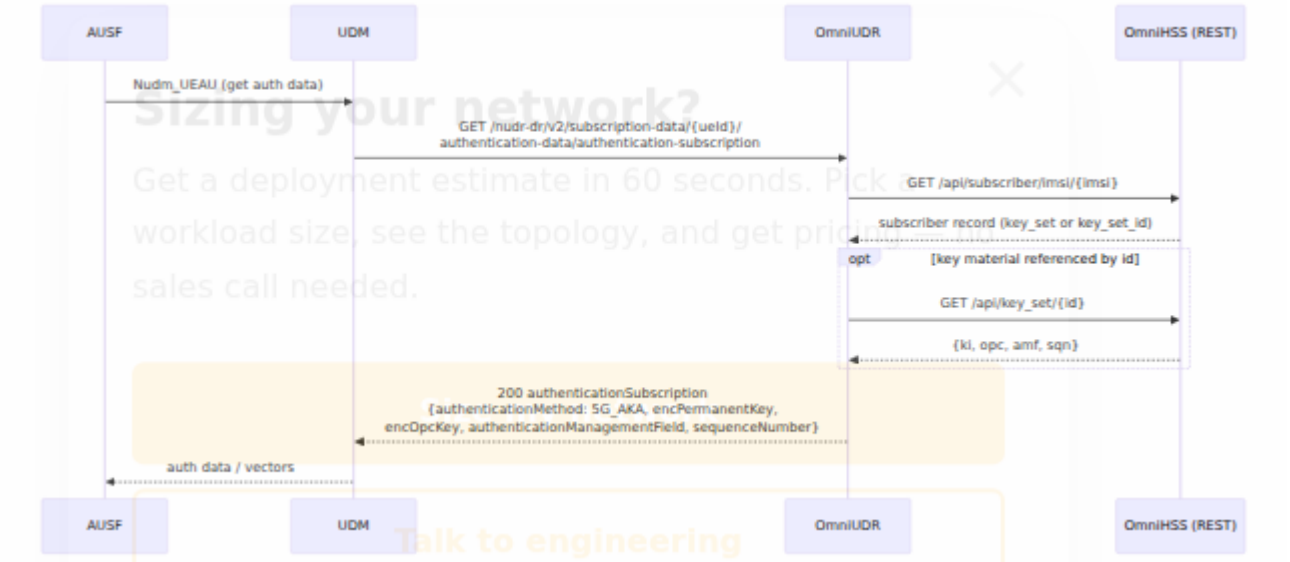
Nudr_DR operation	HSS REST call(s)	Notes
		endpoint and the health gauge

> The `/api/hlr/*` endpoints (`generate_auth_vectors`, `update_sqn`, `circuit_session/{imsi}`) are the HSS's purpose-built REST interface for OmniUDR's auth-vector, SQN, and location operations, alongside the subscriber/key-set endpoints (`/api/subscriber/*`, `/api/key_set/*`) and the health probe (`/api/status/api`). A disabled subscriber (`enabled: false`) is treated as not found.

Key Procedures

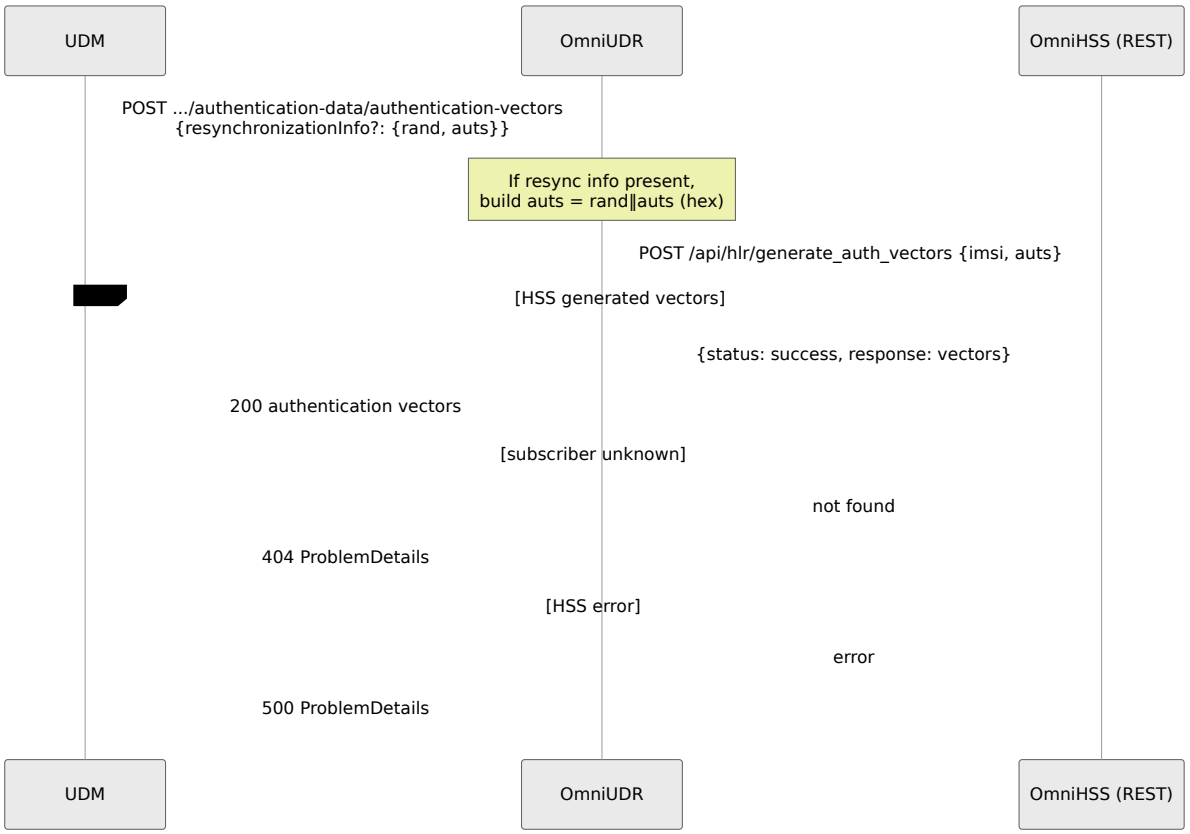
Authentication Subscription Read (UDM → UDR → HSS)

The core UDR flow: the UDM (serving the AUSF during 5G-AKA) asks OmniUDR for a subscriber's authentication subscription; OmniUDR reads the subscriber and key material from the HSS and maps it to the Nudr structure.

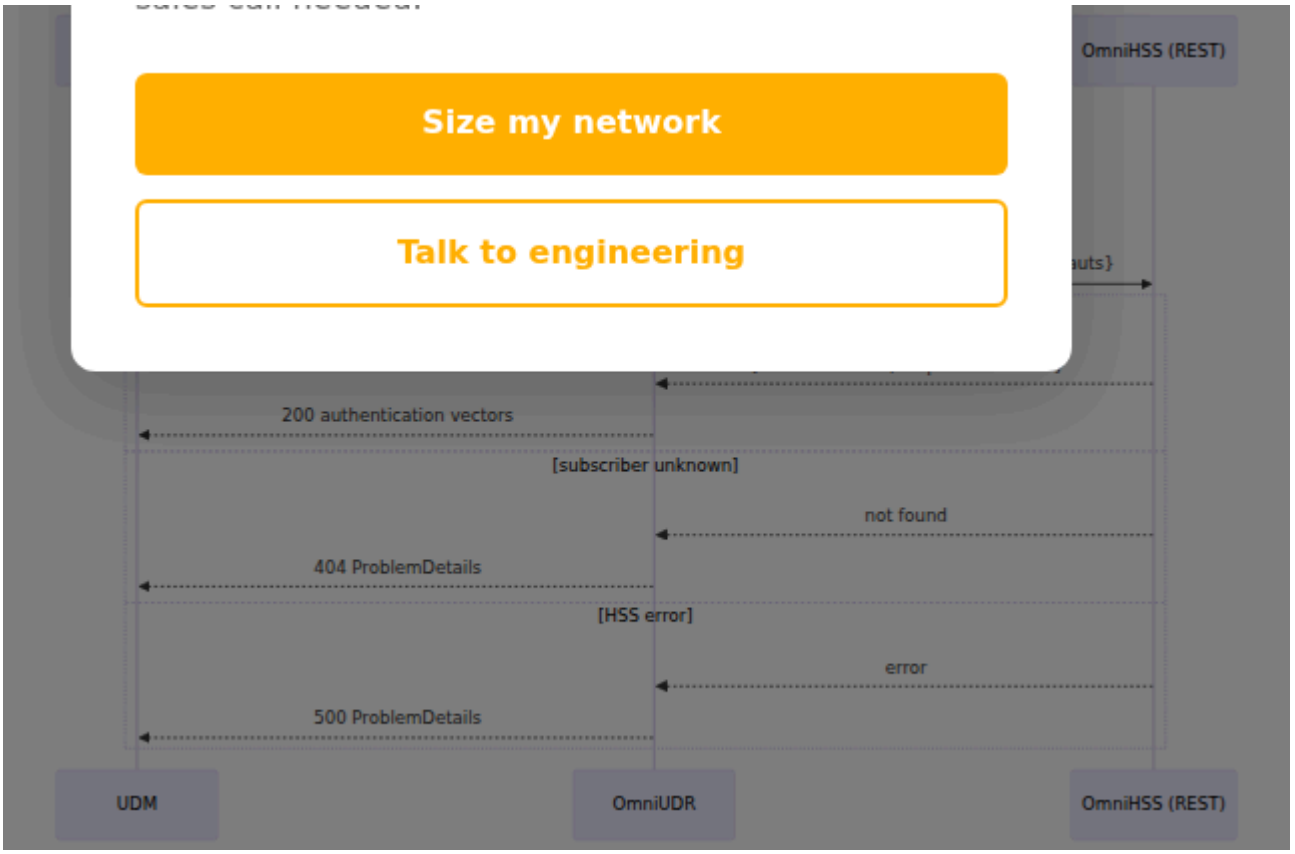


Authentication Vector Generation (proprietary, with resync)

OmniUDM calls the non-standard `authentication-vectors` POST so that Milenage vector generation and SQN/resync are handled centrally at the HSS. On a resynchronisation, the UDM passes `resynchronizationInfo` (`rand` + `auts`), which OmniUDR concatenates into the hex `rand‖auts` the HSS expects.



Provisioned Data Read (AM / SM / SMF-selection)



For AM data the mapping instead derives `subscribedUeAmbr` from the EPC UE-AMBR and attaches a synthesised NSSAI (see `default_am_nssai`); for SMF-selection data it lists the subscriber's APNs as DNN infos under a single subscribed S-NSSAI.

How the HSS APN profile becomes 5G SM subscription data

The 4G HSS record has no 5G session-management structures, so OmniUDR synthesises one `dnnConfigurations` entry per provisioned APN (`epc_profile.apn_profiles[]`). Each field is derived as follows. When the named HSS field is absent, the fallback is served - a fallback across the board means the APN QoS profile field names are not matching (see Troubleshooting → SM data serves a fixed AMBR).

5G SM field	Derived from (HSS field)
`singleNssai`	DNN → S-NSSAI via `dnn_snssai_map`
DNN key in `dnnConfigurations`	APN name (`apn_identifier.apn`)
`pduSessionTypes`	APN `apn_identifier.ip_version` (see ta
`sscModes`	fixed: default `SSC_MODE_1`, allowed `SSC_M` `SSC_MODE_2`
`5gQosProfile.5qi`	`apn_qos_profile.qci`
`5gQosProfile.arp.priorityLevel`	`apn_qos_profile.allocation_retentio`
`5gQosProfile.arp.preemptCap`	`apn_qos_profile.pre_emption_capabil`
`5gQosProfile.arp.preemptVuln`	`apn_qos_profile.pre_emption_vulnera` (bool)
`sessionAmbr.uplink`	`apn_qos_profile.apn_ambr_ul_kbps`
`sessionAmbr.downlink`	`apn_qos_profile.apn_ambr_dl_kbps`
`chargingRules`	APN `pcrf_profile.charging_rules[]` (r precedence, 5QI/QCI, ARP, GBR/MBR, flo

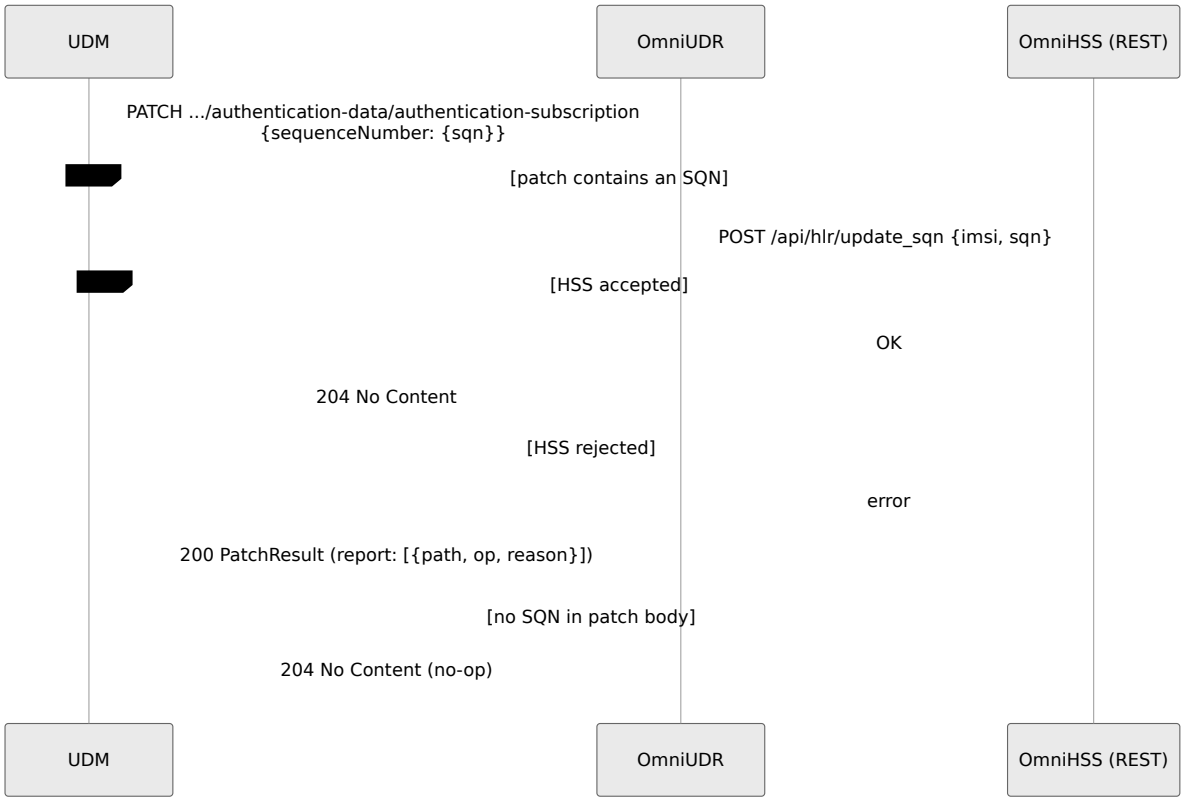
PDU session types. The 5G PDU session type is the 5G twin of the 4G PDN-Type, derived from the APN's `ip_version` so an IPv6 or dual-stack APN yields an IPv6-capable session instead of the IPv4-only default. `defaultSessionType` is what the SMF falls back to when the UE does not request a specific allowed type.

APN `ip_version`	`defaultSessionType`	`allowedSessionTypes`
`ipv4`	`IPV4`	`IPV4`
`ipv6`	`IPV6`	`IPV6`
`ipv4v6` / `ipv4_or_ipv6`	`IPV4V6`	`IPV4V6`, `IPV4`, `IPV6`
unset / other	`IPV4`	`IPV4`, `IPV4V6`

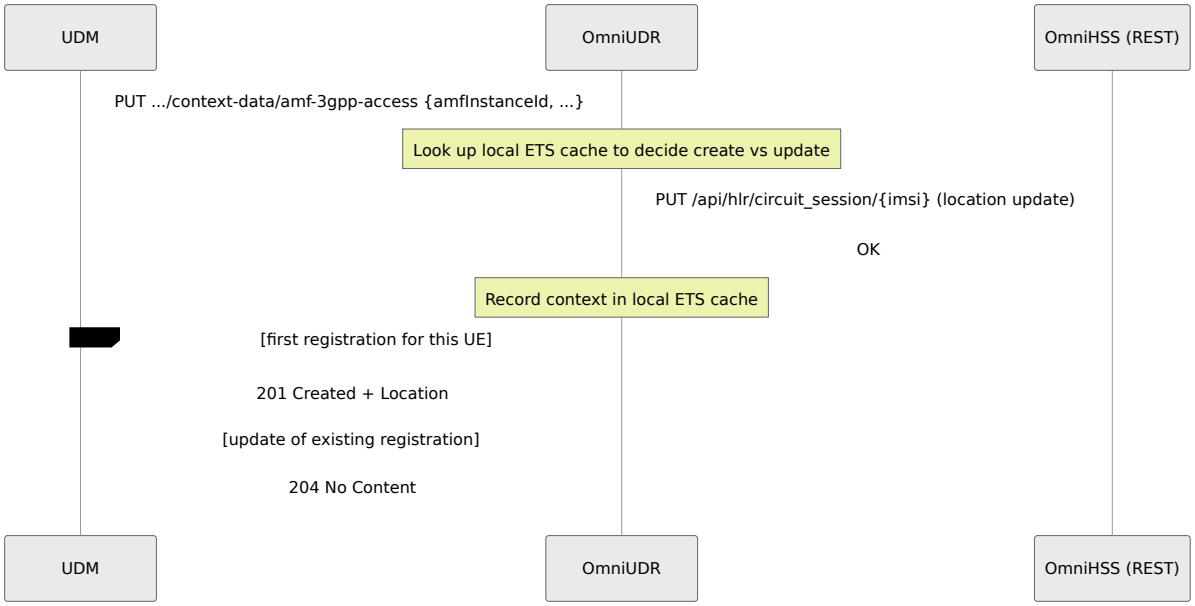
SSC modes are not provisionable per subscriber: every DNN is advertised with default `SSC_MODE_1` and allowed `SSC_MODE_1`/`SSC_MODE_2`. Per TS 29.503 / 29.571 `SscModes.allowedSscModes` has `maxItems: 2` (the default plus at most one alternative), so the set cannot be widened.

SMF-selection subscription data lists every provisioned APN as a DNN under a single subscribed S-NSSAI key `01`, regardless of the per-DNN mapping in `dnn_snssai_map`. In a multi-slice deployment where DNNs live on different S-NSSAIs, use the AM/SM subscription data (which carries the correct per-DNN S-NSSAI) for slice decisions; SMF selection advertises only the DNN list.

SQN Update (PATCH authentication subscription)

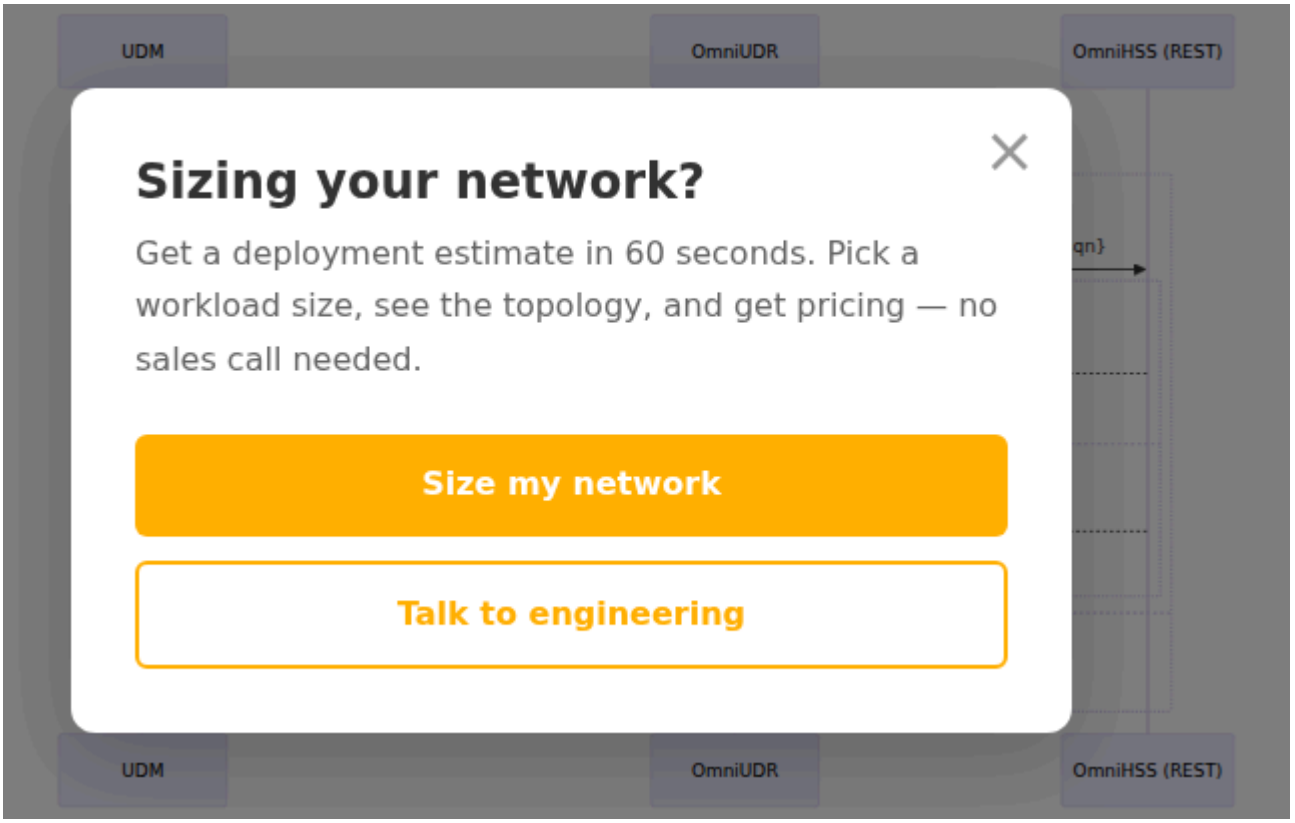


AMF Registration Context Write



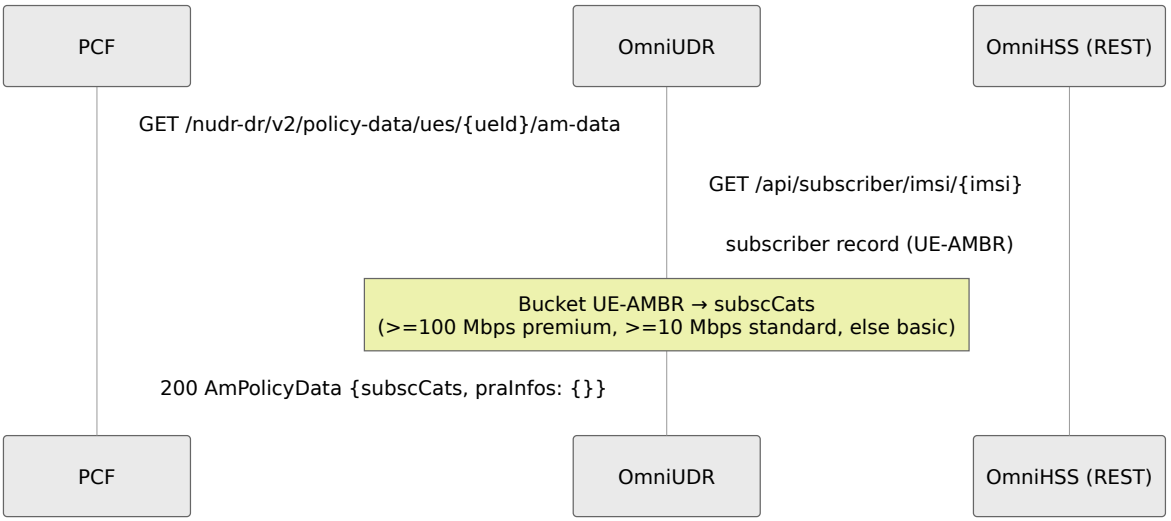
SMF Registration Context Write (local cache only)

The SMF registration context is stored only in the local ETS cache, keyed by `{ueId}/{pduSessionId}`; it is never persisted to the HSS and is lost on restart. The presence of an existing cache entry decides create vs. update.

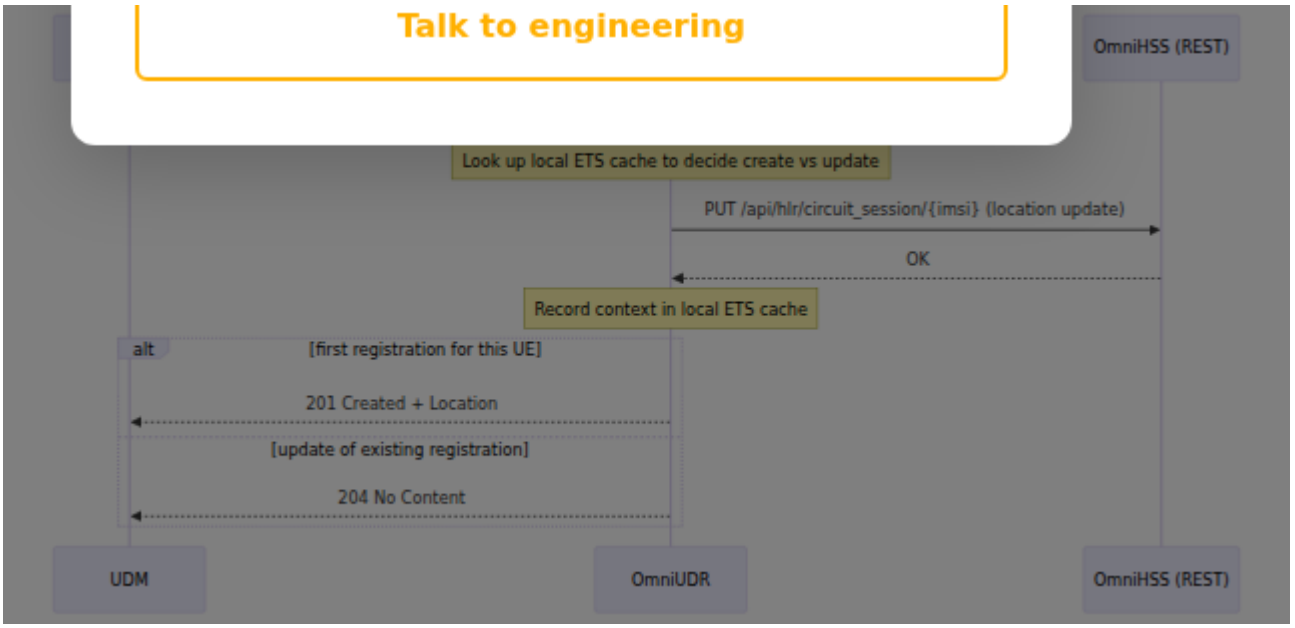


AM Policy Data Read (PCF → UDR → HSS)

AM policy data is coarse: OmniUDR reads the subscriber's UE-AMBR from the HSS and buckets it into a single subscriber category (`premium` / `standard` / `basic`); `praInfos` is always empty.

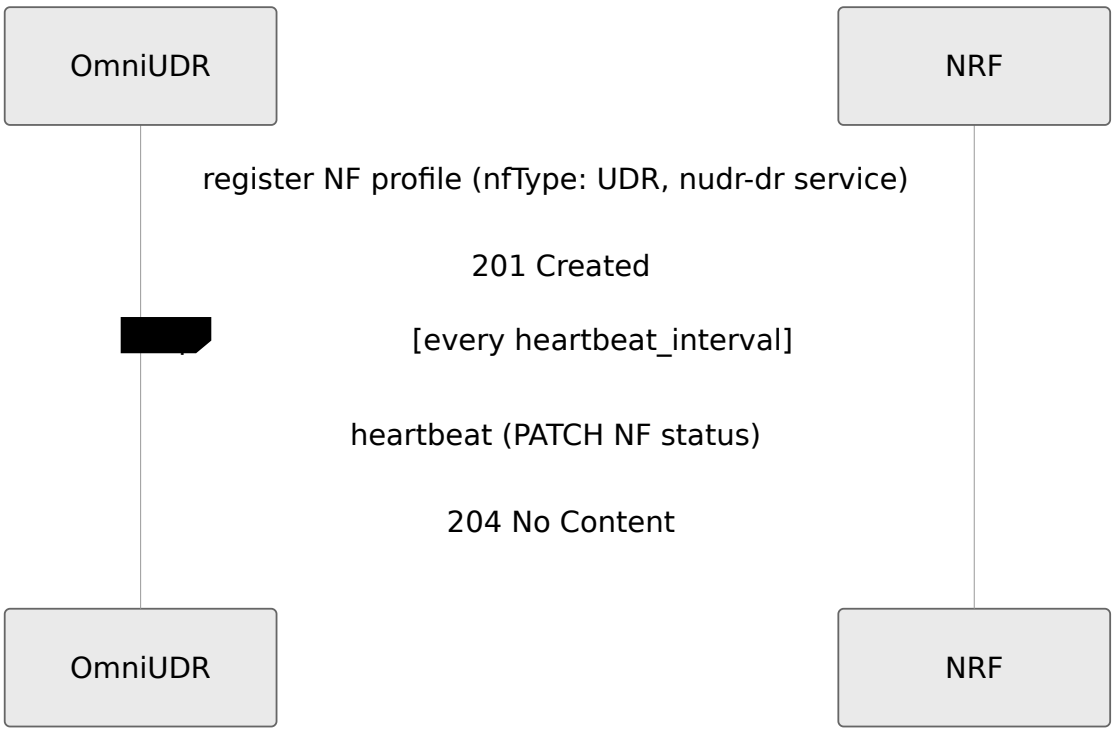


SM Policy Data Read (PCF → UDR → HSS)



NRF Registration and Heartbeat

At start-up OmniUDR registers its NF profile with the NRF and then heartbeats every `heartbeat_interval` milliseconds so the registry keeps advertising it as alive. Registration status is exposed as a gauge (see Metrics), and an OAM action can force re-registration.

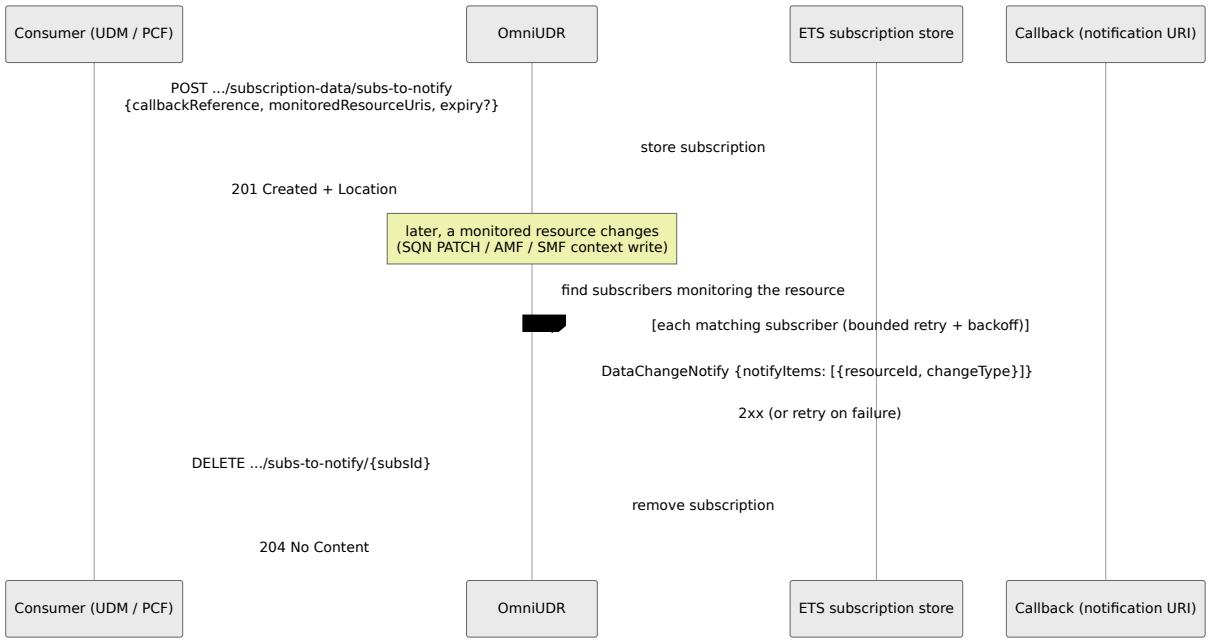


Data-Change Subscriptions and DataChangeNotify

OmniUDR implements the Nudr_DR data-change subscription service (the `subscription-data/subs-to-notify` resource, TS 29.504 subscriptions-to-notify). A consumer registers a subscription that names the resource URIs it wants monitored and a notification callback URI, optionally with an expiry. Subscriptions are held in an in-memory (ETS) subscription store; an expiry, when supplied, is honoured by the store, which drops the subscription once it elapses.

When a monitored resource changes, OmniUDR emits a `DataChangeNotify` to the callback URI of every subscriber whose monitored URIs cover the changed resource (an exact URI match, or a collection prefix so a subscription on a parent resource catches its child resources). Each notification carries a change type: `ADD`, `REPLACE`, or `REMOVE`. In this build notifications fire on an SQN update (PATCH of the authentication subscription) and on AMF and SMF context writes (a first registration for a resource is an `ADD`, a subsequent write is a `REPLACE`).

Delivery is best-effort and never affects the SBI response the consumer receives. A callback that fails or is unreachable is retried with a bounded number of attempts and an exponential backoff (see `notification_max_attempts` and `notification_backoff_ms`); once the attempts are exhausted the notification is dropped and the event is logged.



← Overview

OmniUDR Troubleshooting

All Nudr reads return 404 / subscriber lookups fail

OmniUDR fulfils every read from the HSS backend. If reads fail:

1. Confirm `hss_api_base_url` is reachable from the OmniUDR host and the HSS is up. Use the built-in check: `GET /api/health/hss` on the management API, and watch the `omni_udr_hss_health` gauge (1 = up, 0 = down).
2. Confirm the subscriber exists and is **enabled** in the HSS. A disabled subscriber (`enabled: false`) is treated as not found (404 to the consumer).
3. Confirm the `{ueId}` uses `imsi-<digits>` form (a bare IMSI is also accepted).
4. Watch `omni_udr_hss_requests_total{result="failure"}` and `omni_udr_hss_request_duration_ms` to distinguish a down/slow HSS from a genuinely absent subscriber.

Consumer receives 5xx / errors when the HSS is unreachable

When the HSS backend is down or times out (5 s per request, 3 s for the health check), the HSS REST client returns an error and OmniUDR surfaces it to the consumer (typically a `404` on the read resources, or a management-API `502` on the subscriber proxy). This is expected - OmniUDR has no local subscriber store to fall back to. Restore HSS reachability; `omni_udr_hss_health` returns to 1 once the next check succeeds. Note that `test_subscribers` is a lab aid and is **not** an automatic HSS-down fallback in this build.

PATCH for SQN update returns 200 instead of 204

A `200` with a `PatchResult` body means the HSS **rejected** the SQN update; the `report` array names the failed path and the HSS reason. A successful update returns `204`. Check the HSS logs and the `/api/hlr/update_sqn` path. A PATCH body with no `sequenceNumber` is treated as a successful no-op (`204`).

Authentication vector generation returns 404 / 500

The `authentication-vectors` POST delegates to the HSS `/api/hlr/generate_auth_vectors` endpoint. A `404` means the subscriber is unknown to the HSS; a `500` means the HSS returned an error or an unexpected response shape. On a resync, ensure the UDM supplied `resynchronizationInfo` with both `rand` and `auts` - OmniUDR concatenates them into the `rand||auts` hex the HSS expects. This endpoint is proprietary and only OmniUDM should call it.

Policy data looks thin

AM policy data returns only coarse `subscCats` derived from the subscriber's UE-AMBR tier (`premium` ≥ 100 Mbps, `standard` ≥ 10 Mbps, else `basic`), and `praInfos` is empty. The tier uses whichever of uplink or downlink UE-AMBR is higher. When the subscriber's HSS EPC profile carries no UE-AMBR, OmniUDR serves a 1000 Kbps default, so an unprovisioned or mis-mapped subscriber falls into `basic` - if a subscriber you expect to be `premium`/`standard` shows `basic`, verify the EPC UE-AMBR (`ue_ambr_ul_kbps` / `ue_ambr_dl_kbps`) is provisioned in the HSS. SM policy data is built from the subscriber's APN/QoS and PCRF charging-rule profiles. There is no dedicated 5G policy store; this is by design in the current architecture.

AM data slice (S-NSSAI) mismatch

The 4G HSS record carries no 5G slice subscription, so the AM-data NSSAI is synthesised by OmniUDR and defaults to the standard lab slices (eMBB, URLLC, mIoT). If a UE's requested SD-qualified S-NSSAI is rejected by the AMF (5GMM cause #62), align `default_am_nssai` (and, per-DNN, `dnn_snssai_map`) with the AMF's served NSSAI for the deployment. See the Configuration Reference.

SM data serves a fixed AMBR for every subscriber

If SM subscription data shows the same session AMBR / QoS for all subscribers regardless of provisioning, the HSS APN QoS profile field names are not being matched and OmniUDR is falling back to defaults. The mapping expects the OmniHSS `ApnQosProfile` fields (`apn_ambr_dl_kbps`, `apn_ambr_ul_kbps`, `allocation_retention_priority`, `pre_emption_capability`, `pre_emption_vulnerability`, `qci`). Verify the HSS subscriber record carries these fields.

Context cache appears stale after out-of-band HSS changes

The AMF/SMF registration cache only tracks context writes and is used for the 201-vs-204 decision; it does not cache read data. If it needs clearing (e.g. after test cycling), flush it via `DELETE /api/oam/subscriber-cache/{imsi}` (or `.../all`).