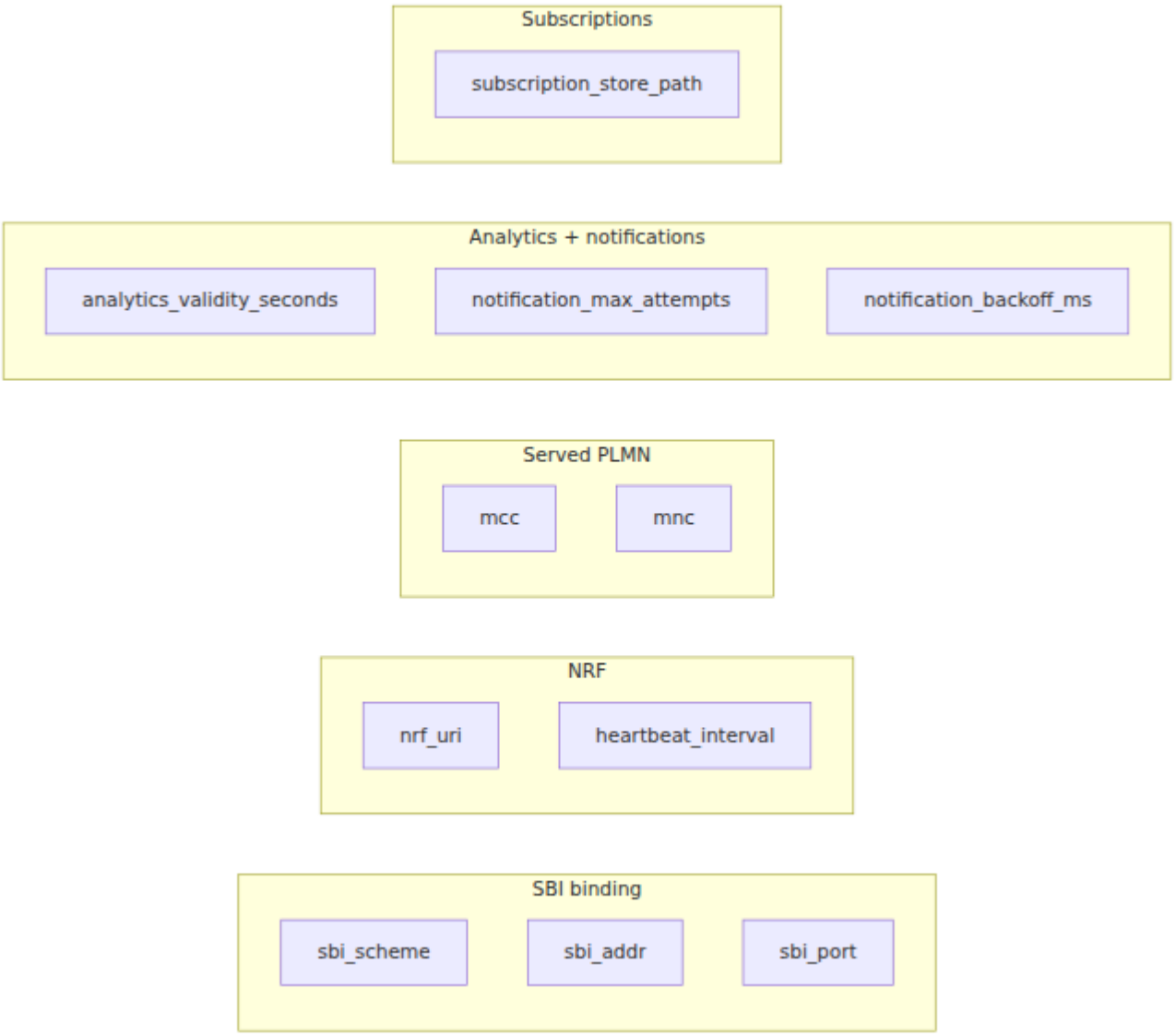
OmniNWDAF Configuration Reference

Overview | Operations | Metrics | Troubleshooting

This reference documents every OmniNWDAF configuration parameter. All parameters live under the `:omninwdaf` application scope. Environment variables apply only in the production runtime configuration (`runtime.exs`); the file-only tuning knobs have no environment variable and must be set in the application config file.

Configuration Structure



Complete Configuration Example

```
config :omninwdaf,
  # SBI binding and advertised endpoint
  sbi_scheme: "http",
  sbi_addr: "127.0.0.23",
  sbi_port: 7777,

  # NRF registration target and heartbeat cadence
  nrf_uri: "http://127.0.0.10:7777",
  heartbeat_interval: 10_000,

  # Served PLMN (override the 999-70 test default per deployment)
  mcc: "999",
  mnc: "70",

  # Validity window applied to returned analytics expiry (seconds)
  analytics_validity_seconds: 3600,

  # Notification delivery retry tuning
  notification_max_attempts: 4,
  notification_backoff_ms: 500,

  # Durable subscription store file (defaults under the system
temp directory)
  subscription_store_path: "/var/lib/omninwdaf/subscriptions.dets"
```

The production `runtime.exs` reads the environment-backed subset from the environment, falling back to the defaults below:

```
config :omninwdaf,
  sbi_scheme: System.get_env("NWDAF_SBI_SCHEME", "http"),
  sbi_addr: System.get_env("NWDAF_SBI_ADDR", "127.0.0.23"),
  sbi_port: String.to_integer(System.get_env("NWDAF_SBI_PORT",
"7777")),
  nrf_uri: System.get_env("NWDAF_NRF_URI",
"http://127.0.0.10:7777"),
  mcc: System.get_env("NWDAF_MCC", "999"),
  mnc: System.get_env("NWDAF_MNC", "70")
```

SBI Binding Parameters

These control the address and port OmniNWDAF binds for its SBI services, and the scheme it advertises to the NRF and to consumers.

Parameter	Type	Required	Default	Env Var	De
`sbi_scheme`	String	No	`http`	`NWDAF_SBI_SCHEME`	UR for en an ad NF
`sbi_addr`	String	No	`127.0.0.23`	`NWDAF_SBI_ADDR`	IP On bir SB an ad to an co
`sbi_port`	Integer	No	`7777`	`NWDAF_SBI_PORT`	TC On list for rec

NRF Parameters

These control registration with the NRF and the heartbeat cadence. See Operations: NRF Registration and Heartbeat. NRF interaction follows TS 29.510.

Parameter	Type	Required	Default	
`nrf_uri`	String	No	`http://127.0.0.10:7777`	`NW`
`heartbeat_interval`	Integer (ms)	No	`10000`	(n

Served PLMN Parameters

These identify the PLMN OmniNWDAF advertises in its NF profile.

Parameter	Type	Required	Default	Env Var	Description
`mcc`	String	No	`999`	`NWDAF_MCC`	Mobile Country Code of the served PLMN. The `999` default is a test value: **override per deployment.**
`mnc`	String	No	`70`	`NWDAF_MNC`	Mobile Network Code of the served PLMN. The `70` default is a test value: **override per deployment.**

Analytics Parameters

Parameter	Type	Required	Default	Env Var	
`analytics_validity_seconds`	Integer (s)	No	`3600`	(none)	

Notification Retry Parameters

These tune best-effort delivery of each events-subscription notification (creation-time, recurring, and post-modify). See Operations: Notification Delivery and Retry.

Parameter	Type	Required	Default	Env Var	D
`notification_max_attempts`	Integer	No	`4`	(none)	M nu de at no be dr C or er va
`notification_backoff_ms`	Integer (ms)	No	`500`	(none)	B be re m Th gr ex pe C or er va

Subscription Store Parameters

These control where events subscriptions are durably persisted. Subscriptions are held in a hot in-memory ETS table mirrored to a DETS file, and reloaded on restart. See Operations: Nnwdaf_EventsSubscription.

Parameter	Type	Required	Default
`subscription_store_path`	String (path)	No	`omninwdaf_subscriptions` under the system temp directory

Notes on File-Only Parameters

`heartbeat_interval`, `analytics_validity_seconds`, `notification_max_attempts`, `notification_backoff_ms`, and `subscription_store_path` have **no environment variable** and are set in the application config file.

OmniNWDAF Metrics and Monitoring

Overview | Operations | Configuration | Troubleshooting

Observability: Structured Logs

Observability is provided through **structured application logs** and standard request logging. Operators should monitor the OmniNWDAF logs for the following signals.

Subscription Lifecycle

Log entries cover the events-subscription lifecycle:

- Creation of a subscription and its assigned subscription id.
- Reload of durable subscriptions at startup, including how many were restored from the store file.
- Rejection of an invalid subscription (missing `notificationURI` or no supported event).
- Deletion of a subscription, including deletion attempts for an unknown id.

Notification Delivery

Log entries cover delivery of each notification (creation-time, recurring, and post-modify; see Operations: Notification Delivery and Retry):

- Each delivery attempt to a subscriber's `notificationURI`.
- Each retry, following the bounded exponential-backoff schedule.
- The final give-up after the last attempt, logged as a **warning**.

Request Logging

Standard HTTP request logging covers the SBI endpoints, including the AnalyticsInfo retrievals, the subscription create, read, modify, and delete calls, and the liveness status endpoint. Use this to confirm that consumers are reaching OmniNWDAF and to observe response status codes such as `400 MANDATORY_QUERY_PARAM_MISSING`, `404 UNSUPPORTED_EVENT`, and `400 MANDATORY_IE_MISSING`.

Recommended Monitoring Approach

Until NF-specific metrics are added, monitor OmniNWDAF via its logs and its liveness endpoint:

Signal	Where	What it tells you
Liveness	`GET /nnwdaf-analyticsinfo/v1/status` returning `200 {"status":"UP"}`	The service is up and serving SBI requests.
NRF registration	Logs at startup	Whether registration with the NRF succeeded.
Notification give-ups	Warning logs after the final delivery attempt	Subscribers whose `notificationURI` is unreachable.
Request errors	Request logs with `4xx` status	Malformed or unsupported analytics requests and subscriptions.

Health Checking

The liveness endpoint is the recommended readiness and liveness probe target:

- **Endpoint:** `GET /nnwdaf-analyticsinfo/v1/status`
- **Healthy response:** `200` with body `{"status":"UP"}`

A healthy status confirms the SBI listener is serving requests. It does not, by itself, confirm that the NRF registration is current: check the startup and heartbeat logs for that.

OmniNWDAF Operations Guide

Overview | Configuration | Metrics | Troubleshooting

This guide describes how OmniNWDAF behaves at run time and how to operate it. OmniNWDAF is the 5G Core **Network Data Analytics Function** (3GPP TS 23.501 §6.2.18, TS 23.288), exposing the Nnwdaf services of TS 29.520.

Table of Contents

- Analytics Coverage
- Architecture Overview
- Service Endpoints
- NRF Registration and Heartbeat
- Nnwdaf_AnalyticsInfo: Request / Response
- Nnwdaf_EventsSubscription: Subscribe / Notify
- Notification Delivery and Retry
- Analytics Event ID Reference
- Operational Behaviour Summary

Analytics Coverage

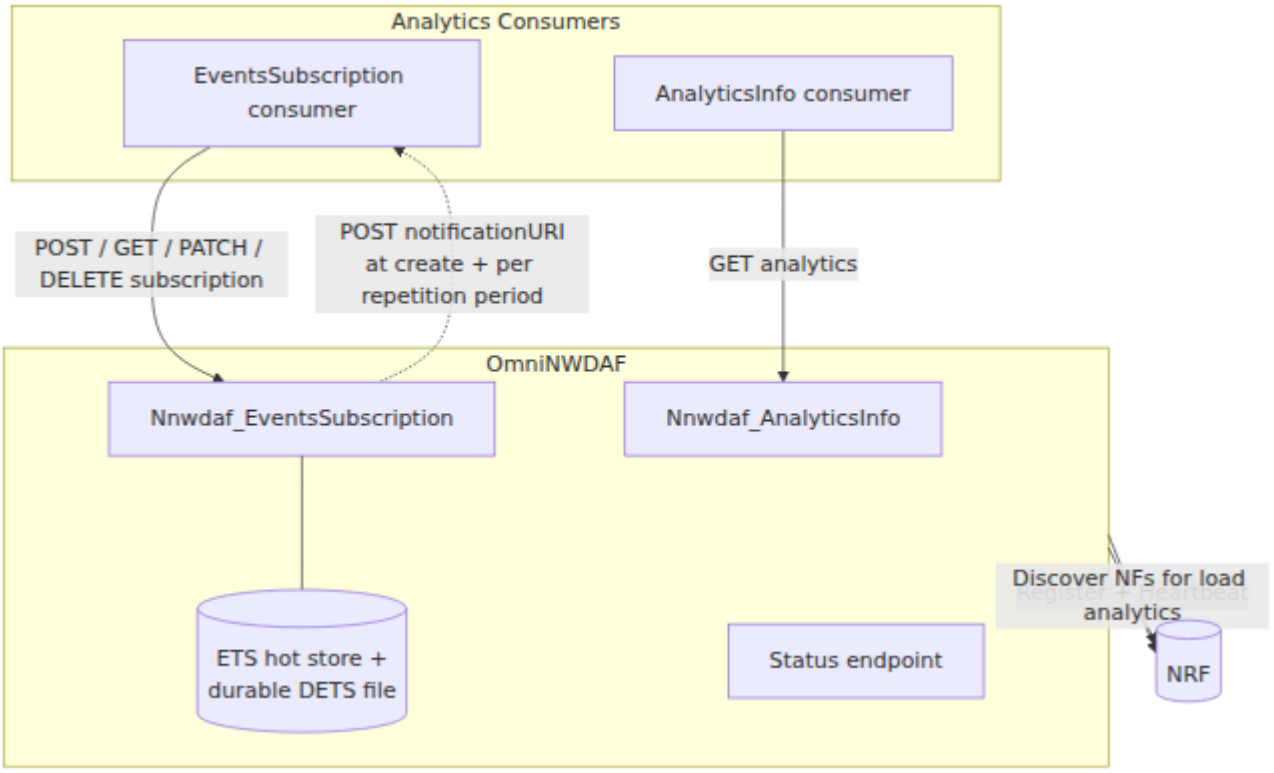
The load-oriented events are computed from live data OmniNWDAF discovers from the NRF, namely each NF's registration status and its reported `load` (0-100, TS 29.510 NFProfile):

- `NF_LOAD` - one `NfLoadLevelInformation` entry per discovered NF, carrying its reported average and peak load and its status. When a single `nfType` filter is supplied, only that type is discovered.
- `SLICE_LOAD_LEVEL` and `NSI_LOAD_LEVEL` - the average reported load across the NFs that serve the requested S-NSSAI.

- `NETWORK_PERFORMANCE` - network-wide aggregates: the ratio of registered NFs (`GNB_ACTIVE_RATIO`) and the average reported load (`GNB_COMPUTING_USAGE`, `GNB_MEMORY_USAGE`).

For these events, `confidence` reflects how much live data backed the result (the count of NFs with a usable `load`) and is `0` when no data was available.

Architecture Overview



The subscription store keeps a hot in-memory ETS table backed by a **durable DETS file**. Subscriptions survive a restart: they are reloaded on start and their recurring notifications are re-armed.

Service Endpoints

OmniNWDAF exposes the following HTTP SBI endpoints. All paths are served under the SBI binding configured by `sbi_scheme`, `sbi_addr`, and `sbi_port` (see Configuration).

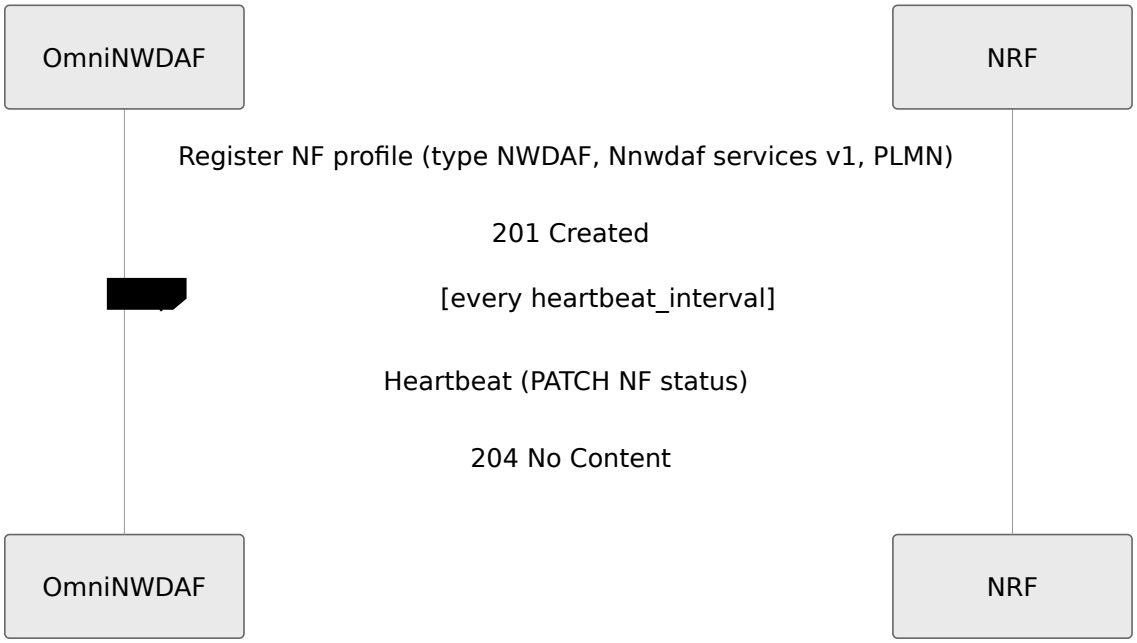
Method	Path	
GET	`/nnwdaf-analyticsinfo/v1/analytics`	Retr one by `e` quer filter
POST	`/nnwdaf-eventssubscription/v1/subscriptions`	Crea subs carri `not:` and `ever`
GET	`/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}`	Read subs
PATCH	`/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}`	Mod subs mer store
DELETE	`/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}`	Dele subs
GET	`/nnwdaf-analyticsinfo/v1/status`	Live

These map to the Nnwdaf service operations in TS 29.520. Create, read, modify (PATCH), and delete of an individual subscription are all provided.

NRF Registration and Heartbeat

On startup OmniNWDAF registers with the NRF at `nrf_uri` as NF type `NWDAF`, advertising the two Nnwdaf services (`nnwdaf-analyticsinfo` and `nnwdaf-eventssubscription`, both `v1`) and the served PLMN (`mcc`/`mnc`). It then sends a periodic heartbeat every `heartbeat_interval` milliseconds to keep the profile alive. NRF registration and heartbeat follow TS 29.510.

Beyond registration, OmniNWDAF also uses the NRF as a **data source**: when a load-oriented analytics request or notification is served, it discovers the relevant NF instances from the NRF and reads their reported `load` and status to compute the result (see Analytics Coverage). Discovery is best-effort: a discovery failure yields an empty NF set and the affected analytics return a `confidence` `0` result rather than erroring.



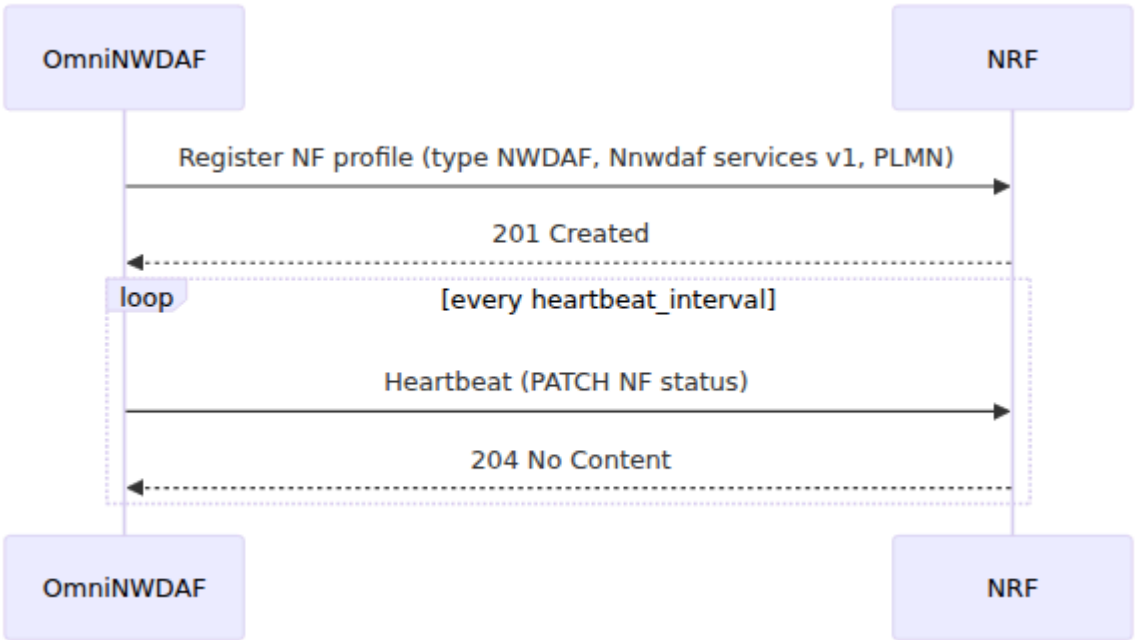
In addition to registration and heartbeat, OmniNWDAF performs NRF discovery to read the load and status of other NFs for the load-oriented analytics. It does not consume any other producer interfaces.

Nnwdaf_AnalyticsInfo: Request / Response

A consumer requests analytics for a single event by calling `GET /nnwdaf-analyticsinfo/v1/analytics` with an `event-id` query parameter. Additional query parameters (for example S-NSSAI, DNN, SUPI, NF type) act as filters.

OmniNWDAF returns an `AnalyticsData` object:

- `start` and `timeStampGen` are set to the current time.
- `expiry` is set to the current time plus `analytics_validity_seconds`.
- For the four load-oriented events (`NF_LOAD`, `SLICE_LOAD_LEVEL`, `NSI_LOAD_LEVEL`, `NETWORK_PERFORMANCE`), OmniNWDAF discovers NF instances from the NRF and computes the analytics from their reported load and status. `confidence` reflects the number of NFs with usable load data (`0` when none).
- For the remaining events, the caller's filter values are echoed back inside the response.



How it works: the load-oriented events reflect the live NF population reported by the NRF. For the remaining events, the response carries the standard timestamps and echoes the caller's filter values.

Nnwdaf_EventsSubscription: Subscribe / Notify

A consumer creates a subscription by POSTing to `/nnwdaf-eventssubscription/v1/subscriptions` with a body containing a `notificationURI` and one or more `eventSubscriptions`.

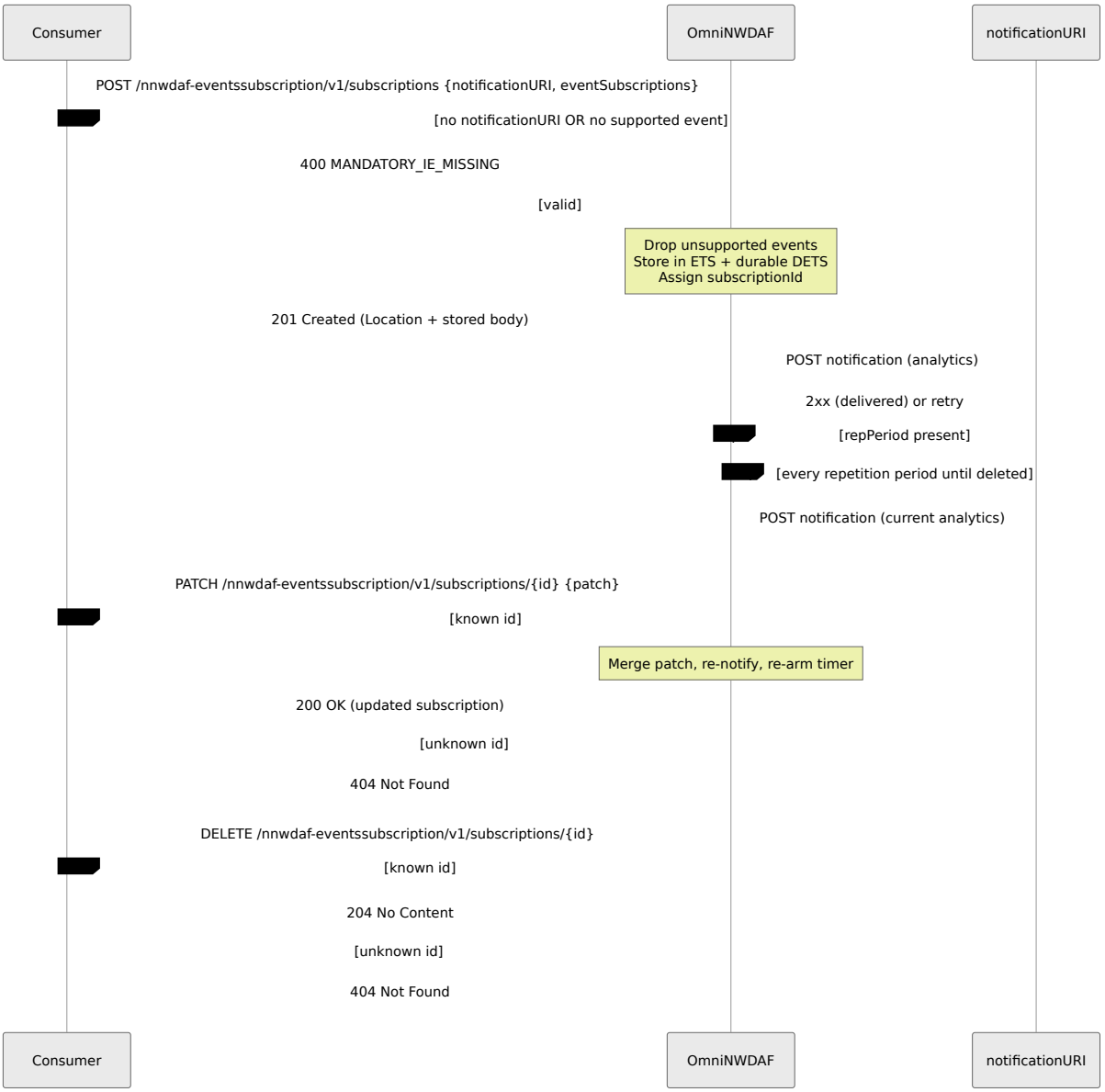
Behaviour on create:

- The subscription is stored in the hot ETS table and mirrored to the **durable DETS file**, and assigned a subscription id.
- A `201 Created` is returned with a `Location` header pointing at the new subscription and the stored body in the response.
- **Unsupported events in the subscription are silently dropped.** A subscription that contains **no supported event** is rejected with `400 MANDATORY_IE_MISSING`. A missing `notificationURI` is also rejected with `400 MANDATORY_IE_MISSING`.
- An **immediate notification** is delivered to the `notificationURI` at creation (one `eventNotification` per supported subscribed event).

Recurring notification. When a subscription carries a reporting requirement with a `repPeriod` (seconds), either at the subscription level (`evtReq`) or per event (`eventReq`), OmniNWDAF re-delivers the notification on that period until the subscription is removed. The smallest positive `repPeriod` found is used. Without a `repPeriod` the subscription is one-shot: only the creation-time notification is sent. There is no threshold or data-change trigger.

Read and modify. A stored subscription can be read with `GET .../subscriptions/{id}` and modified with `PATCH .../subscriptions/{id}`. A modify shallow-merges the patch body into the stored subscription (preserving the `subscriptionId`), re-notifies immediately with the updated definition, and re-arms the repetition timer.

Durability. Subscriptions and their recurring notifications survive a restart: the store is reloaded from the DETS file on start and the repetition timers are re-armed.



Notification Delivery and Retry

Every notification (the creation-time one, each recurring one, and the one following a modify) is delivered **best-effort** with a bounded, exponential-backoff retry:

- Up to `notification_max_attempts` delivery attempts are made.
- The delay before each retry starts at `notification_backoff_ms` and grows exponentially per attempt.
- If the subscriber has not accepted the notification after the final attempt, OmniNWDAF gives up and logs a warning. The subscription remains stored; it is not automatically removed.

See Configuration for the retry parameters, and Metrics for how delivery attempts, retries, and give-ups surface in the logs.

Analytics Event ID Reference

OmniNWDAF advertises and accepts all 16 standardised analytics event IDs from the `NwdafEvent` enumeration in TS 29.520.

Event ID	Analytics	Reference
`NF_LOAD`	Load analytics for a specific NF	TS 29.520
`SLICE_LOAD_LEVEL`	Load level information for a network slice	TS 29.520
`NSI_LOAD_LEVEL`	Network slice instance load level	TS 29.520
`NETWORK_PERFORMANCE`	Network performance analytics	TS 29.520
`SERVICE_EXPERIENCE`	Service experience analytics	TS 29.520
`UE_MOBILITY`	UE mobility analytics	TS 29.520
`UE_COMMUNICATION`	UE communication analytics	TS 29.520
`QOS_SUSTAINABILITY`	QoS sustainability analytics	TS 29.520
`ABNORMAL_BEHAVIOUR`	Abnormal behaviour analytics	TS 29.520
`USER_DATA_CONGESTION`	User data congestion analytics	TS 29.520
`SM_CONGESTION`	Session management congestion analytics	TS 29.520
`DISPERSION`	Data and mobility dispersion analytics	TS 29.520
`RED_TRANS_EXP`	Redundant transmission experience analytics	TS 29.520
`WLAN_PERFORMANCE`	WLAN performance analytics	TS 29.520
`DN_PERFORMANCE`	Data Network performance analytics	TS 29.520

Event ID	Analytics	Reference
`PDU_SESSION_TRAFFIC`	PDU Session traffic analytics	TS 29.520

An `event-id` outside this set returns `404 UNSUPPORTED_EVENT` on AnalyticsInfo, and is silently dropped when it appears inside an events subscription.

Operational Behaviour Summary

Behaviour	Detail
Load-oriented analytics	`NF_LOAD`, `SLICE_LOAD_LEVEL`, `NSI_LOAD_LEVEL`, `NETWORK_PERFORMANCE` computed from live NRF data; `confidence` = count of NFs with usable load
Other analytics content	Filters echoed back
Analytics timestamps	`start`/`timeStampGen` = now; `expiry` = now + `analytics_validity_seconds`
Subscription storage	Hot ETS table mirrored to a durable DETS file; reloaded on restart
Subscription operations	Create, read (GET), modify (PATCH), delete
Subscription notifications	Immediate notification at creation; recurring notification when a `repPeriod` is present; re-notify on modify; no threshold or data-change trigger
Unsupported events	Silently dropped in a subscription; `404 UNSUPPORTED_EVENT` on AnalyticsInfo
Empty subscription	Rejected with `400 MANDATORY_IE_MISSING`
Notification delivery	Best-effort, bounded exponential-backoff retry; logged give-up after the final attempt
NRF	Registration, periodic heartbeat, and NF discovery for the load-oriented analytics

Behaviour	Detail
Data collection	NRF discovery (per-NF load + status) feeds the load-oriented analytics

OmniNWDAF Troubleshooting

This guide covers common issues and their resolutions.

Common Issues

AnalyticsInfo returns 400 MANDATORY_QUERY_PARAM_MISSING

Symptoms: `GET /nnwdaf-analyticsinfo/v1/analytics` returns `400` with `MANDATORY_QUERY_PARAM_MISSING`.

Possible causes:

- The `event-id` query parameter is absent. It is mandatory: analytics are always retrieved for a single event.

Resolution:

1. Add an `event-id` query parameter set to one of the supported event IDs (see Operations: Analytics Event ID Reference).

AnalyticsInfo returns 404 UNSUPPORTED_EVENT

Symptoms: `GET /nnwdaf-analyticsinfo/v1/analytics` returns `404` with `UNSUPPORTED_EVENT`.

Possible causes:

- The `event-id` value is not one of the 16 standardised analytics event IDs.

Resolution:

1. Check the value against the Analytics Event ID Reference. Event IDs are case-sensitive and must match the TS 29.520 `NwdafEvent` names exactly.

Subscription create returns 400 MANDATORY_IE_MISSING

Symptoms: `POST /nnwdaf-eventssubscription/v1/subscriptions` returns `400` with `MANDATORY_IE_MISSING`.

Possible causes:

- The body has no `notificationURI`.
- The body's `eventSubscriptions` contains **no supported event**. Unsupported events are silently dropped, so a body that lists only unsupported events is treated as empty and rejected.

Resolution:

1. Ensure a `notificationURI` is present.
2. Ensure at least one `eventSubscriptions` entry references a supported event ID.

Subscriber never receives the notification

Symptoms: A subscription is created (`201`) but the subscriber's `notificationURI` never receives the callback.

Possible causes:

- The `notificationURI` is unreachable from OmniNWDAF, or the subscriber returned non-`2xx` for every attempt.
- All delivery attempts (up to `notification_max_attempts`) were exhausted; OmniNWDAF gave up and logged a warning.

Resolution:

1. Confirm the `notificationURI` is reachable from OmniNWDAF and that the subscriber returns a `2xx` on receipt.
2. Check the OmniNWDAF logs for the delivery attempts and the final give-up warning (see Metrics).
3. If needed, raise `notification_max_attempts` or `notification_backoff_ms` (see Configuration). A missed creation-time notification is retried per the delivery schedule; if the subscription carries a `repPeriod` it will also be re-attempted on each recurring cycle. Otherwise, modify (`PATCH`) or recreate the subscription to trigger a fresh delivery.

Subscriptions disappeared after a restart

Symptoms: Previously created subscriptions return `404` on read or delete, or no longer exist after a restart.

Possible causes:

- Subscriptions are persisted to a durable DETS file and reloaded on start. If they vanished, the store file was not preserved: it may have been on ephemeral storage (the default path is under the system temp directory) or the configured `subscription_store_path` changed between runs.

Resolution:

1. Point `subscription_store_path` at durable storage that persists across restarts, and keep the path stable between runs (see Configuration).
2. If the store file was lost, recreate the affected subscriptions.

OmniNWDAF does not appear in the NRF

Symptoms: Consumers cannot find OmniNWDAF, or the NRF shows no NWDAF profile.

Possible causes:

- `nrf_uri` points at the wrong NRF or an unreachable address.
- Registration failed at startup (check the logs).
- The heartbeat lapsed and the NRF expired the profile.

Resolution:

1. Verify `nrf_uri` is correct and reachable (see Configuration).
2. Check the startup logs for a successful registration.
3. Confirm the heartbeat is running; if `heartbeat_interval` is longer than the NRF's validity period, shorten it.

DELETE returns 404

Symptoms: `DELETE /nnwdaf-eventssubscription/v1/subscriptions/{id}` returns `404`.

Possible causes:

- The subscription id is unknown, either because it never existed, was already deleted, or the durable store file was not preserved across a restart.

Resolution:

1. Confirm the id matches one returned in a create `Location` header. If it existed before a restart, verify `subscription_store_path` points at durable storage (see Subscriptions disappeared after a restart).