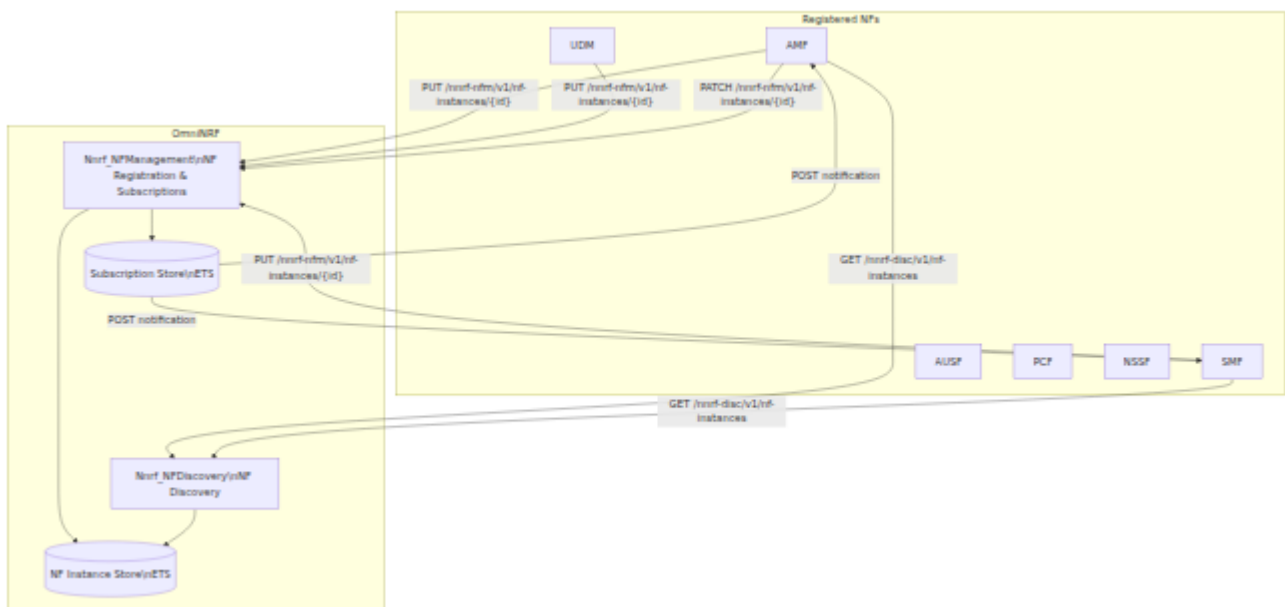


OmniNRF Operations

1. Component Overview

OmniNRF is the Network Repository Function (NRF) for the Omnitouch 5G core. It provides centralised NF registration, discovery, and status-change notification services to all other NFs in the 5GC. All NF instances register with the NRF on startup and maintain liveness via periodic heartbeats. Consumers query the NRF to resolve NF endpoints before issuing SBI calls.



2. 3GPP Role and Spec References

Aspect	Reference
NRF functional definition	TS 23.501 Section 6.2.6
Nnrf_NFManagement service	TS 29.510 Section 5.2
Nnrf_NFDiscovery service	TS 29.510 Section 5.3
NFProfile data model	TS 29.510 Section 6.1
NF status notification	TS 29.510 Section 5.2.2.8
Heartbeat procedure	TS 29.510 Section 5.2.2.4
SearchResult data model	TS 29.510 Section 6.2

OmniNRF implements the SBI producer role for `Nnrf_NFManagement` and `Nnrf_NFDiscovery` as defined in TS 29.510.

3. SBI Endpoints

All endpoints are HTTP/1.1 with `Content-Type: application/json`.

Nnrf_NFManagement (TS 29.510 Section 5.2)

Method	Path	Description	Success
PUT	/nnrf-nfm/v1/nf-instances/{nfInstanceId}	Register or re-register an NF	201 Created / 200 OK
PATCH	/nnrf-nfm/v1/nf-instances/{nfInstanceId}	Heartbeat (empty body) or JSON Patch update	204 No Content / 200 OK
DELETE	/nnrf-nfm/v1/nf-instances/{nfInstanceId}	Deregister NF	204 No Content
GET	/nnrf-nfm/v1/nf-instances/{nfInstanceId}	Retrieve NF profile	200 OK
GET	/nnrf-nfm/v1/nf-instances	List all NF instances	200 OK
POST	/nnrf-nfm/v1/subscriptions	Subscribe to NF status notifications	201 Created
DELETE	/nnrf-nfm/v1/subscriptions/{subscriptionId}	Unsubscribe	204 No Content

Nnrf_NFDiscovery (TS 29.510 Section 5.3)

Method	Path	Description	Success
GET	/nnrf-disc/v1/nf-instances	Discover NF instances by type and filters	200 OK

Discovery Query Parameters

Parameter	Required	Description
<code>target-nf-type</code>	Yes	NF type being searched for (e.g. <code>UDM</code> , <code>SMF</code>)
<code>requester-nf-type</code>	Yes	NF type making the request (e.g. <code>AMF</code>)
<code>service-names</code>	No	Comma-separated list of service names to filter by
<code>plmn-id</code>	No	JSON-encoded PLMN ID, e.g. <code>{"mcc": "999", "mnc": "70"}</code>
<code>limit</code>	No	Maximum number of results to return

4. Configuration Reference

OmniNRF is configured via Elixir application environment under the `:omninrf` key. In a release, these values are typically set in `config/runtime.exs` or via environment variables.

Example Configuration

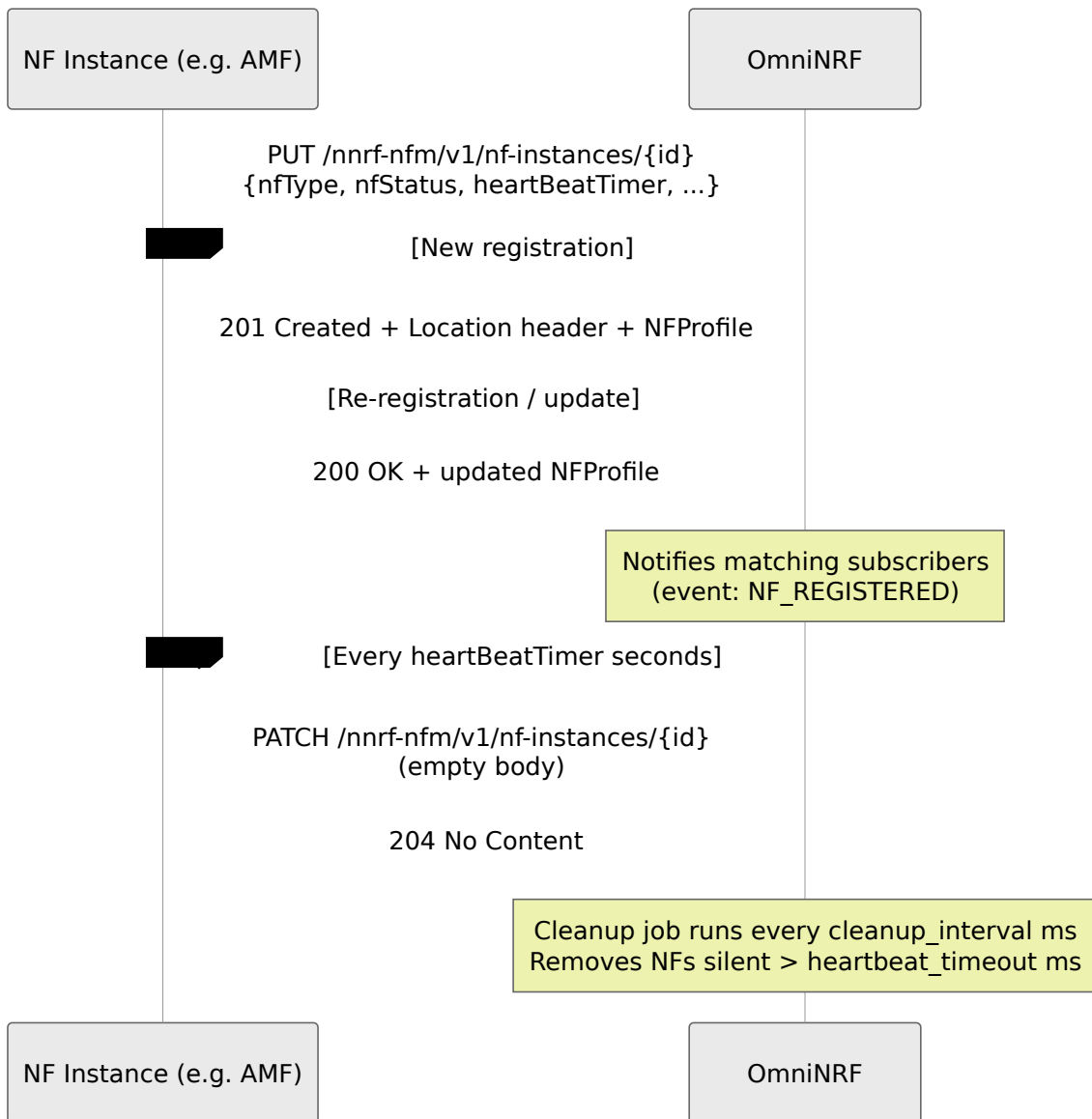
```
config :omninrf,  
  sbi_scheme: "http",  
  sbi_addr: "127.0.0.10",  
  sbi_port: 7777,  
  mcc: "999",  
  mnc: "70",  
  heartbeat_timeout: 30_000,  
  cleanup_interval: 10_000,  
  plmn_list: [%{mcc: "999", mnc: "70"}]
```

Parameter Table

Parameter	Type	Default	Description
<code>sbi_scheme</code>	string	<code>"http"</code>	URI scheme for the SBI listener (<code>http</code> or <code>https</code>)
<code>sbi_addr</code>	string	<code>"127.0.0.10"</code>	IP address the SBI HTTP server binds to
<code>sbi_port</code>	integer	<code>7777</code>	TCP port the SBI HTTP server listens on
<code>mcc</code>	string	<code>"999"</code>	Mobile Country Code for the served PLMN
<code>mnc</code>	string	<code>"70"</code>	Mobile Network Code for the served PLMN
<code>heartbeat_timeout</code>	integer (ms)	<code>30000</code>	Time after the last heartbeat before an NF instance is considered stale and removed. The NRF also uses this value to set <code>heartBeatTimer</code> in the NFProfile if the registering NF does not supply one
<code>cleanup_interval</code>	integer (ms)	<code>10000</code>	How often the stale-registration cleanup job runs
<code>plmn_list</code>	list of maps	<code>[%{mcc: mcc, mnc: mnc}]</code>	List of PLMNs served by this NRF, used for PLMN filtering in discovery

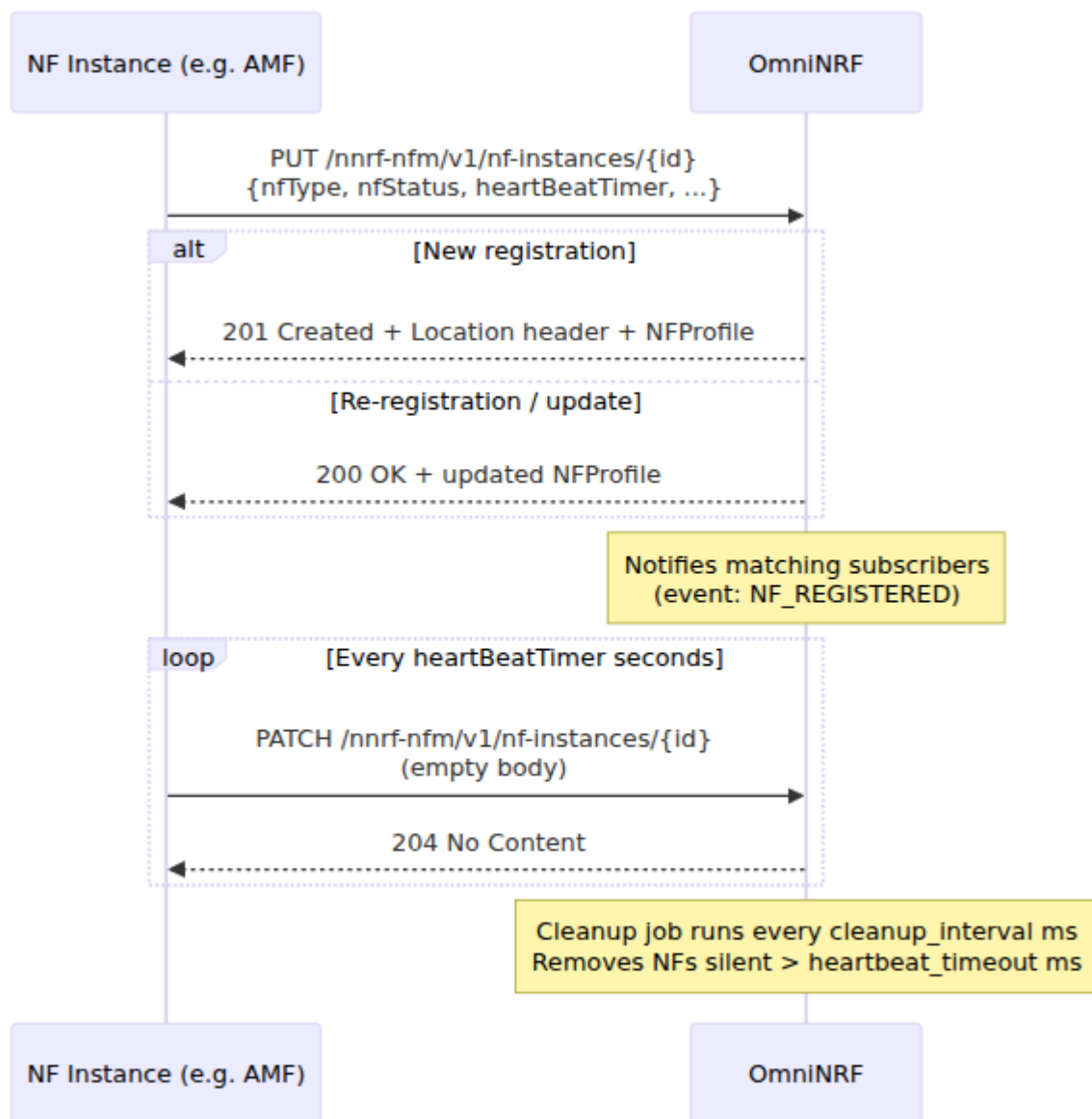
5. Key Procedures

5.1 NF Registration and Heartbeat



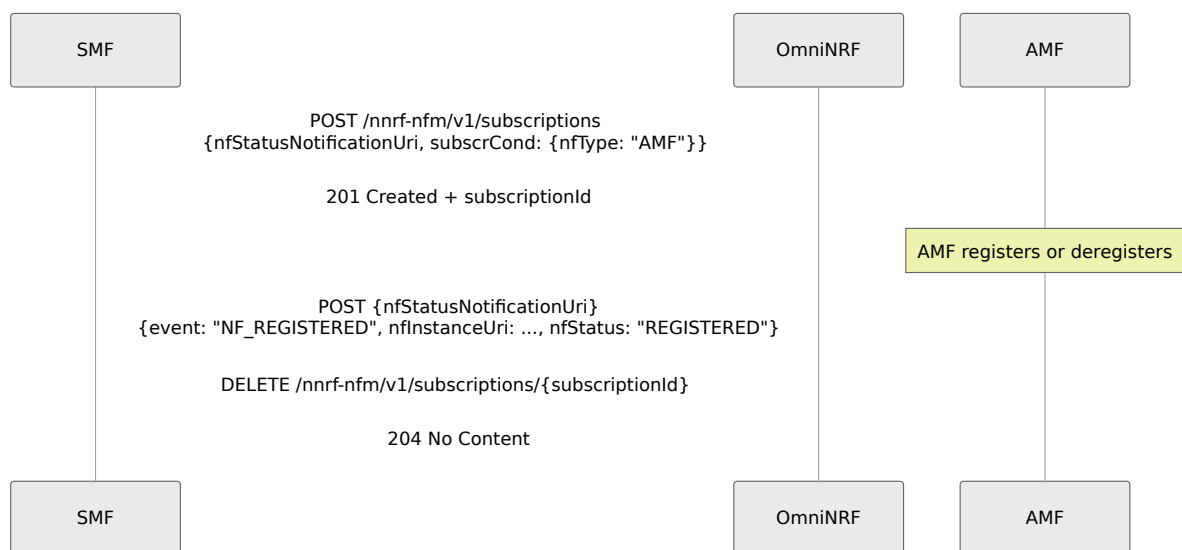
After a stale NF is removed, the NRF sends a **NF_DEREGISTERED** notification to all matching subscribers.

5.2 NF Discovery



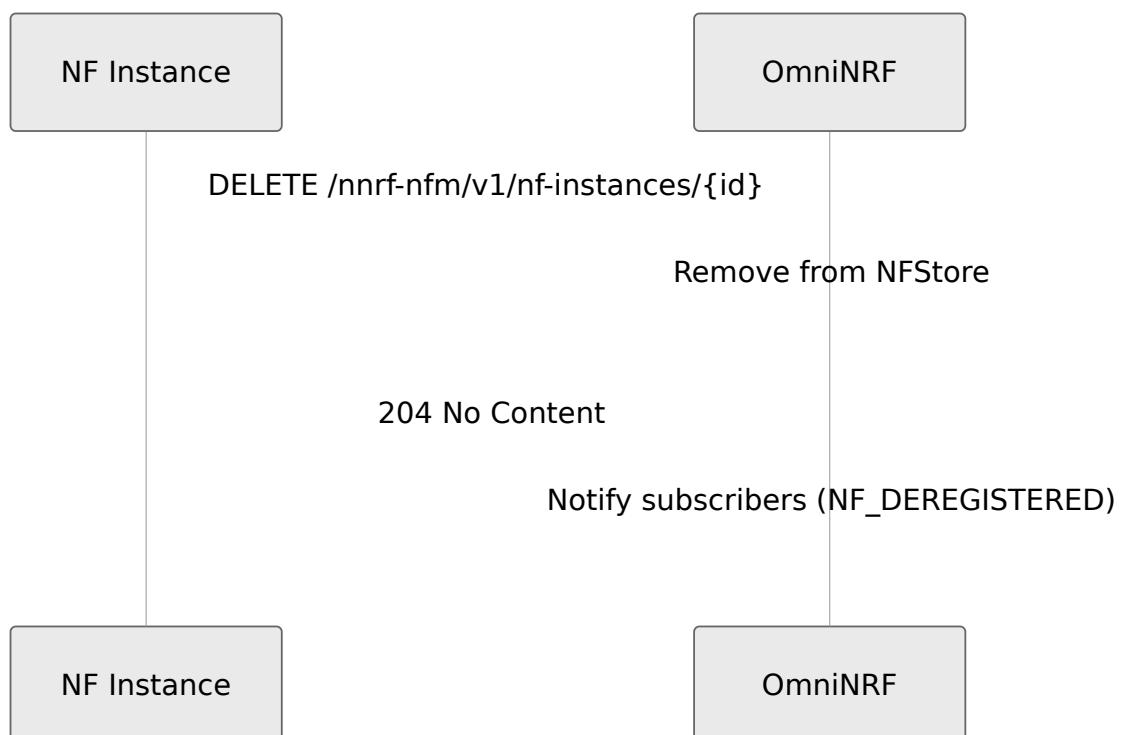
The returned `validityPeriod` is hardcoded to `3600` seconds (see NRF-L5 in Known Limitations).

5.3 NF Status Subscriptions



Notifications are fire-and-forget (no retry on delivery failure). Subscriptions can be filtered by `subscrCond.nfType`.

5.4 NF Deregistration



6. Prometheus Metrics

NRF Metrics

Metric	Type	Tags	Description
<code>omni_nrf.nf.registered.count</code>	gauge	<code>nf_type</code>	Number of registered instances by type
<code>omni_nrf.discovery.requests.count</code>	counter	--	Discovery requests
<code>omni_nrf.registration.requests.count</code>	counter	--	Registration requests
<code>omni_nrf.nf_registrations.total</code>	counter	<code>nf_type</code>	Total number of registrations
<code>omni_nrf.nf_deregistrations.total</code>	counter	<code>nf_type</code>	Total number of deregistrations
<code>omni_nrf.discovery_requests.total</code>	counter	<code>target_nf_type</code>	Total discovery requests by target type
<code>omni_nrf.heartbeats.total</code>	counter	<code>nf_type</code>	Total heartbeats received by type
<code>omni_nrf.active_nf_instances.count</code>	gauge	<code>nf_type</code>	Number of active instances by type

BEAM VM Metrics

Metric	Type	Description
<code>beam.memory.total</code>	gauge	Total BEAM memory in bytes
<code>beam.memory.processes</code>	gauge	Memory used by Erlang processes
<code>beam.memory.processes_used</code>	gauge	Memory actually used by processes
<code>beam.memory.system</code>	gauge	System memory
<code>beam.memory.atom</code>	gauge	Total atom memory
<code>beam.memory.atom_used</code>	gauge	Used atom memory
<code>beam.memory.binary</code>	gauge	Binary memory
<code>beam.memory.code</code>	gauge	Code memory
<code>beam.memory.ets</code>	gauge	ETS table memory
<code>beam.processes.count</code>	gauge	Number of Erlang processes
<code>beam.ports.count</code>	gauge	Number of Erlang ports
<code>beam.atom.count</code>	gauge	Number of atoms
<code>beam.vm.uptime</code>	gauge	VM uptime in seconds

7. Known Limitations

These limitations reflect the current implementation state and represent gaps against the full TS 29.510 specification.

ID	Area	Description
NRF-C2	Discovery filters	Discovery supports only <code>target-nf-type</code> , <code>requester-nf-type</code> , <code>service-names</code> , <code>plmn-id</code> , and <code>limit</code> . Filters for <code>snssai</code> , <code>dnn</code> , <code>tai</code> , <code>guami</code> , <code>preferred-collocated-nf-types</code> , and others defined in TS 29.510 Section 6.2.3.2.3.1 are not implemented
NRF-H2	Notification body	Status change notifications are missing the full <code>NfProfile</code> object and <code>subscriptionId</code> field in the body per TS 29.510 Section 5.2.2.8
NRF-H3	Subscription expiry	Subscriptions have no <code>validityTime</code> or <code>reqPeriodicRegTimer</code> enforcement. Subscriptions never expire
NRF-H4	NFProfile validation	Only <code>nfType</code> is validated on registration. Fields such as <code>nfInstanceId</code> , <code>nfStatus</code> , PLMN lists, and capacity are not validated
NRF-H5	Subscription PATCH	There is no <code>PATCH /nnrf-nfm/v1/subscriptions/{subscriptionId}</code> endpoint for updating an existing subscription
NRF-M1	JSON Patch	The PATCH handler applies only flat-key operations (<code>replace</code> , <code>add</code> , <code>remove</code>). Nested JSON Pointer paths (e.g. <code>/nfServices/0/serviceStatus</code>) are not resolved correctly
NRF-M2	Optimistic concurrency	ETag / If-Match / If-None-Match conditional request handling is not implemented
NRF-M4	Subscription GET	There is no <code>GET /nnrf-nfm/v1/subscriptions/{subscriptionId}</code> endpoint

ID	Area	Description
NRF-L5	Discovery validityPeriod	<code>validityPeriod</code> in discovery responses is hardcoded to <code>3600</code> seconds rather than derived from NF profile data or NRF configuration
NRF-L6	Discovery pagination	No <code>Link</code> header pagination or cursor-based iteration is supported for large discovery result sets

8. Troubleshooting

NF registration fails with 400 Bad Request

The request body is missing `nfType`. All NFProfile bodies must include at least `nfType`. Check the payload of the PUT request.

NF disappears from registry unexpectedly

The NF is not sending heartbeats frequently enough. The default `heartbeat_timeout` is 30 seconds. Ensure the registering NF sends a PATCH heartbeat at an interval shorter than this value. The NRF sets `heartBeatTimer` in the registration response to `heartbeat_timeout / 1000` seconds as a hint.

Discovery returns an empty `nfInstances` list

- Verify the NF is registered: `GET /nnrf-nfm/v1/nf-instances/{id}` must return a profile with `nfStatus: "REGISTERED"`.
- Check that `target-nf-type` exactly matches the `nfType` in the stored NFProfile (case-sensitive).
- If `plmn-id` is specified, verify it is valid JSON and matches the `plmnList` in the NFProfile, or that the NFProfile has no `plmnList` (which causes it to match all PLMNs).

Status notifications are not received

- Confirm `nfStatusNotificationUri` in the subscription body is reachable from the NRF host.
- The notification delivery is fire-and-forget; check application logs for `Failed to send status notification to` warnings.
- If `subscrCond.nfType` is set, the notification is only sent when an NF of that type changes status.

High registration churn or frequent stale cleanup log messages

Lower `cleanup_interval` and `heartbeat_timeout` if aggressive cleanup is desired, or raise `heartbeat_timeout` if NFs are being wrongly expired. Stale cleanup events are logged at `info` level with the message `NRF: Expiring stale NF instance {id}`.