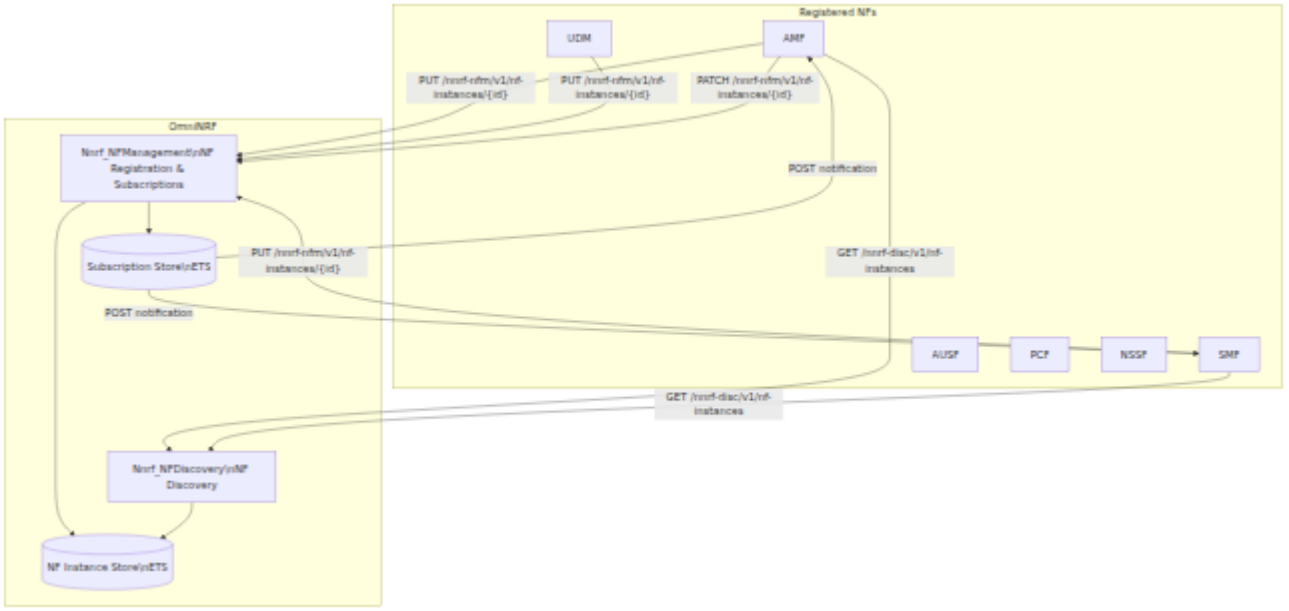
# OmniNRF Operations

## 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 | Succ |
|---|---|---|---|
| PUT | `/nnrf-nfm/v1/nf-instances/{nfInstanceId}` | Register or re-register an NF | 201 Creat / 200 |
| PATCH | `/nnrf-nfm/v1/nf-instances/{nfInstanceId}` | Heartbeat (empty body) or JSON Patch update | 204 N Conte / 200 |
| DELETE | `/nnrf-nfm/v1/nf-instances/{nfInstanceId}` | Deregister NF | 204 N Conte |
| GET | `/nnrf-nfm/v1/nf-instances/{nfInstanceId}` | Retrieve NF profile | 200 C |
| GET | `/nnrf-nfm/v1/nf-instances` | List all NF instances | 200 C |
| POST | `/nnrf-nfm/v1/subscriptions` | Subscribe to NF status notifications | 201 Creat |
| DELETE | `/nnrf-nfm/v1/subscriptions/{subscriptionId}` | Unsubscribe | 204 N Conte |

## 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 |
|---|---|---|
| `target-nf-type` | Yes | NF type being searched for (e.g. `UDM`, `SMF`) |
| `requester-nf-type` | Yes | NF type making the request (e.g. `AMF`) |
| `service-names` | No | Comma-separated list of service names to filter by |
| `plmn-id` | No | JSON-encoded PLMN ID, e.g. `{"mcc":"999","mnc":"70"}` |
| `limit` | 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

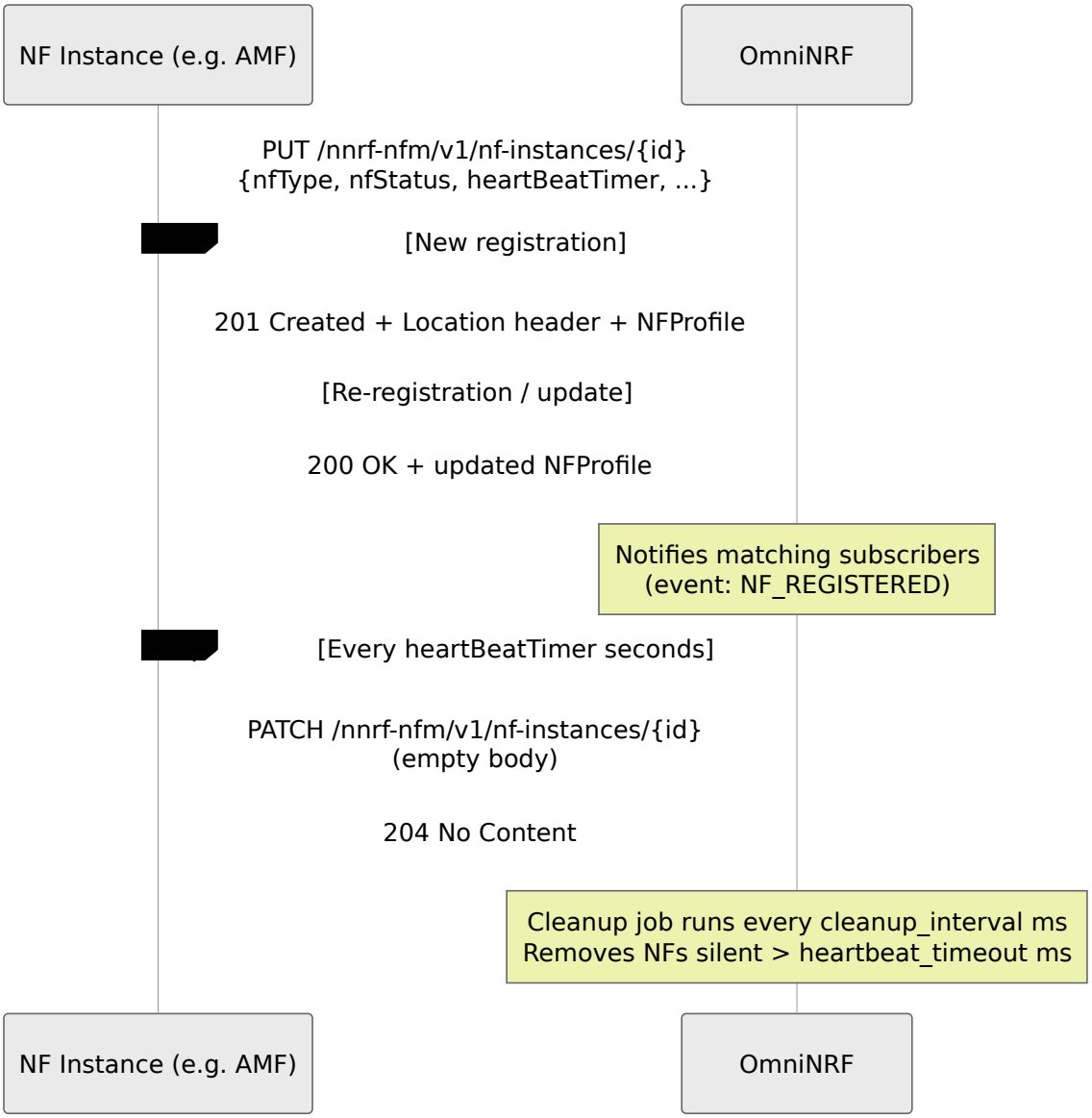
```
config :omninrf,
  sbi_scheme: "http",
  sbi_addr: "127.0.0.10",
  sbi_port: 7777,
  mcc: "999",
  mnc: "70",
  heartbeat_timeout: 30_000,
  cleanup_interval: 10_000,
  plmn_list: [%{mcc: "999", mnc: "70"}]
```

## Parameter Table

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

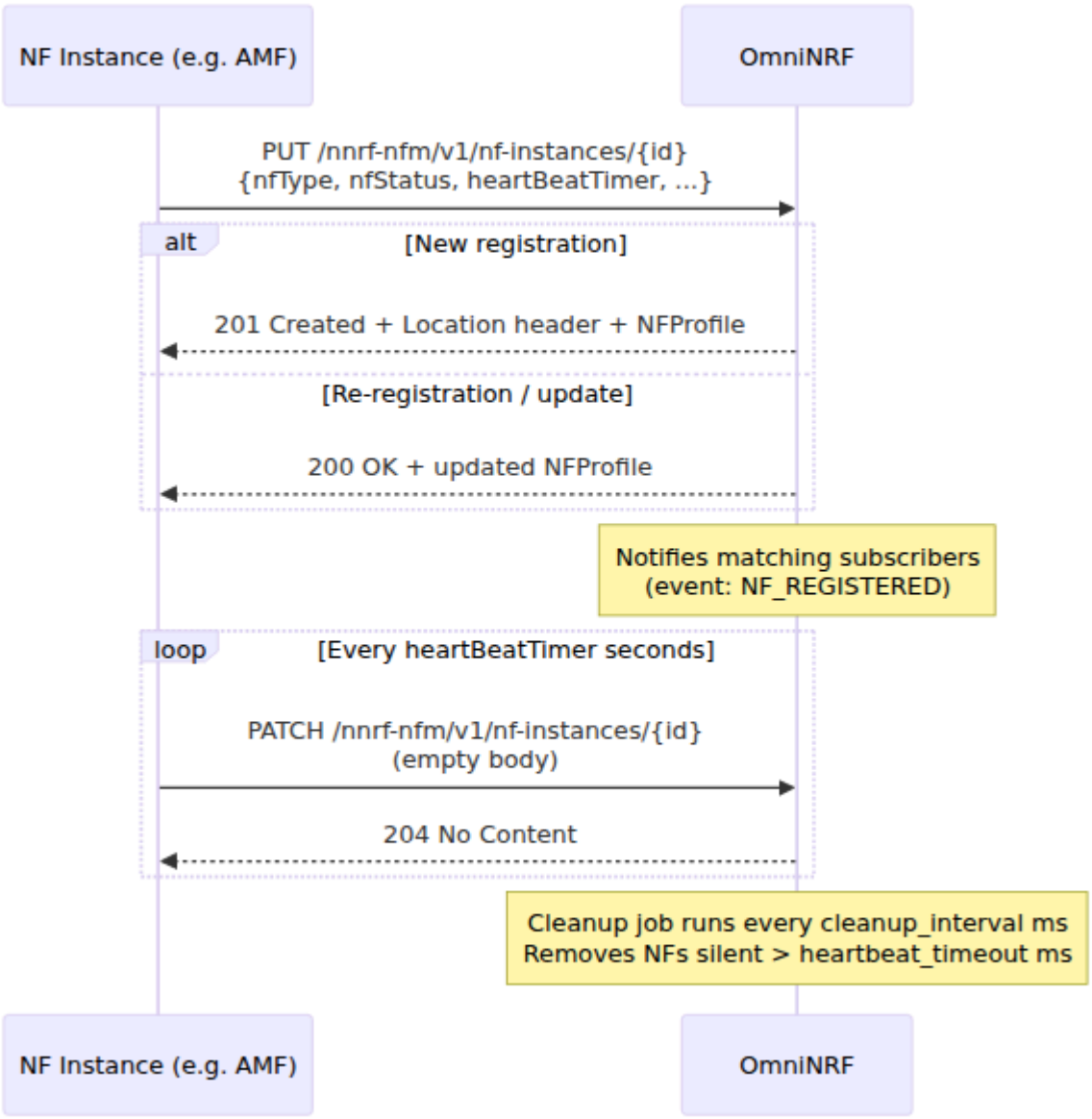
# 5. Key Procedures

## 5.1 NF Registration and Heartbeat



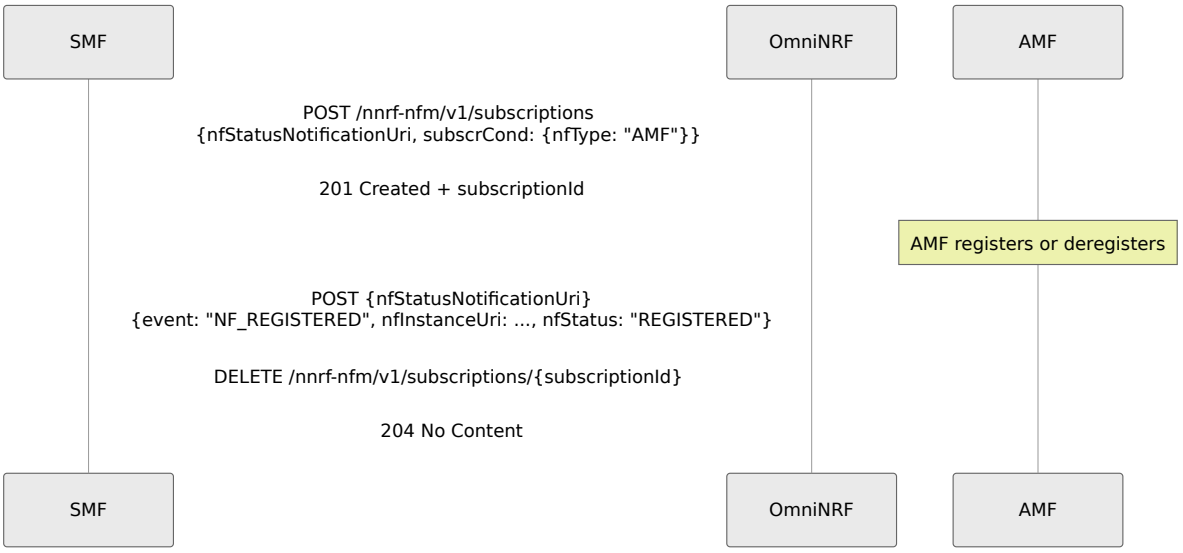

After a stale NF is removed, the NRF sends a `NF_DEREGISTERED` notification to all matching subscribers.

## 5.2 NF Discovery



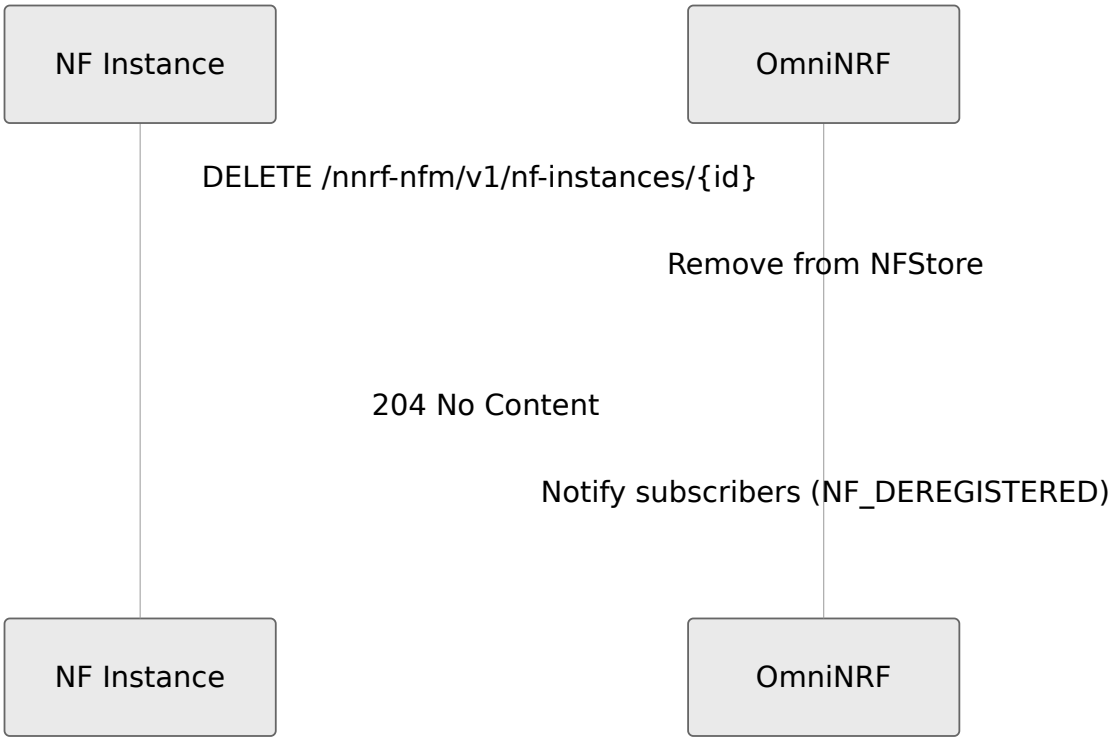

The returned `validityPeriod` is hardcoded to `3600` seconds (see NRF-L5 in Known Limitations).

## 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 | De |
|---|---|---|---|
| `omni_nrf.nf.registered.count` | gauge | `nf_type` | Nur reg inst typ |
| `omni_nrf.discovery.requests.count` | counter | -- | Dis req |
| `omni_nrf.registration.requests.count` | counter | -- | Reg req |
| `omni_nrf.nf_registrations.total` | counter | `nf_type` | Tota reg |
| `omni_nrf.nf_deregistrations.total` | counter | `nf_type` | Tota der |
| `omni_nrf.discovery_requests.total` | counter | `target_nf_type` | Tota disc req targ |
| `omni_nrf.heartbeats.total` | counter | `nf_type` | Tota hea req |
| `omni_nrf.active_nf_instances.count` | gauge | `nf_type` | Nur act inst typ |

## 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.processes_used` | gauge | Memory actually used by processes |
| `beam.memory.system` | gauge | System memory |
| `beam.memory.atom` | gauge | Total atom memory |
| `beam.memory.atom_used` | gauge | Used atom memory |
| `beam.memory.binary` | gauge | Binary memory |
| `beam.memory.code` | gauge | Code memory |
| `beam.memory.ets` | gauge | ETS table memory |
| `beam.processes.count` | gauge | Number of Erlang processes |
| `beam.ports.count` | gauge | Number of Erlang ports |
| `beam.atom.count` | gauge | Number of atoms |
| `beam.vm.uptime` | 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 `target-nf-type`, `requester-nf-type`, `service-names`, `plmn-id`, and `limit`. Filters for `snssai`, `dnn`, `tai`, `guami`, `preferred-collocated-nf-types`, 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 `NfProfile` object and `subscriptionId` field in the body per TS 29.510 Section 5.2.2.8 |
| NRF-H3 | Subscription expiry | Subscriptions have no `validityTime` or `reqPeriodicRegTimer` enforcement. Subscriptions never expire |
| NRF-H4 | NFProfile validation | Only `nfType` is validated on registration. Fields such as `nfInstanceId`, `nfStatus`, PLMN lists, and capacity are not validated |
| NRF-H5 | Subscription PATCH | There is no `PATCH /nnrf-nfm/v1/subscriptions/{subscriptionId}` endpoint for updating an existing subscription |
| NRF-M1 | JSON Patch | The PATCH handler applies only flat-key operations (`replace`, `add`, `remove`). Nested JSON Pointer paths (e.g. `/nfServices/0/serviceStatus`) 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 `GET /nnrf-nfm/v1/subscriptions/{subscriptionId}` endpoint |

| ID | Area | Description |
|---|---|---|
| NRF-L5 | Discovery validityPeriod | `validityPeriod` in discovery responses is hardcoded to `3600` seconds rather than derived from NF profile data or NRF configuration |
| NRF-L6 | Discovery pagination | No `Link` 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}`.