← Vue d'ensemble

Référence de configuration OmniNRF

OmniNRF lit sa configuration d'opérateur à partir de `config/runtime.exs`, qui associe les variables d'environnement à l'environnement d'application `:omninrf`. Chaque clé destinée à l'opérateur est listée ci-dessous avec sa variable d'environnement et sa valeur par défaut.

Configuration d'exécution

```
import Config

config :omninrf,
  sbi_addr: System.get_env("SBI_ADDR", "127.0.0.1"),
  sbi_port: String.to_integer(System.get_env("SBI_PORT", "7777")),
  mcc: System.get_env("MCC", "999"),
  mnc: System.get_env("MNC", "70"),
  heartbeat_timeout:
String.to_integer(System.get_env("HEARTBEAT_TIMEOUT", "30000")),
  prometheus_metrics_port:
String.to_integer(System.get_env("PROMETHEUS_PORT", "9568"))

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

Tableau des paramètres

Paramètre	Type	Requis	Par défaut
`sbi_scheme`	chaîne	Non	`http`
`sbi_addr`	chaîne	Non	`127.0.0.1`
`sbi_port`	entier	Non	`7777`

Paramètre	Type	Requis	Par défaut
`mcc`	chaîne	Non	`999`
`mnc`	chaîne	Non	`70`
`heartbeat_timeout`	entier (ms)	Non	`30000`

Paramètre	Type	Requis	Par défaut
`cleanup_interval`	entier (ms)	Non	`10000`
`notification_max_attempts`	entier	Non	`4`
`notification_backoff_ms`	entier (ms)	Non	`500`

Paramètre	Type	Requis	Par défaut
`discovery_validity_period`	entier (s)	Non	`3600`
`plmn_list`	liste	Non	`[%{mcc: mcc, mnc: r`

Paramètre	Type	Requis	Par défaut
`prometheus_metrics_port`	entier	Non	`9568`
`logger format`	tuple	Non	`{OmniLogger.JsonFo :format}`
`logger metadata`	atome/liste	Non	`:all`

> `SBI_ADDR` par défaut à `127.0.0.1` (boucle locale). Dans tout déploiement multi-hôtes ou conteneurisé, cela doit être remplacé par une adresse que les autres NF peuvent atteindre, sinon aucun NF ne pourra s'enregistrer ou se découvrir.

Sémantique d'âge du heartbeat

Le NRF **reçoit** des heartbeats d'autres NF ; il ne les envoie pas. Chaque fois qu'un NF envoie un heartbeat PATCH (ou toute inscription/mise à jour) pour son instance, le NRF enregistre un nouvel horodatage "dernière vue" contre ce profil NF.

Un balayage en arrière-plan s'exécute toutes les `cleanup_interval` millisecondes et supprime tout NF dont l'horodatage de dernière vue est plus ancien que `heartbeat_timeout` millisecondes. Lorsqu'un NF est éliminé de cette manière, il est désenregistré du magasin et une notification d'état

`NF_DEREGISTERED` est envoyée à tous les abonnés correspondants, afin que les consommateurs ne reçoivent plus un point de terminaison mort.

Lors de l'enregistrement, si un NF n'inclut pas son propre `heartBeatTimer`, le NRF en remplit un égal à `heartbeat_timeout / 1000` secondes comme indice de la fréquence à laquelle il s'attend à être heartbeaté. Les opérateurs doivent s'assurer que les NF heartbeatent bien à l'intérieur de `heartbeat_timeout` (une valeur confortablement plus courte que le timeout) afin que des variations transitoires ne provoquent pas l'expiration d'un NF actif.

L'intervalle de balayage en arrière-plan (`cleanup_interval`) et le seuil de vieillissement (`heartbeat_timeout`) sont indépendants. Un NF n'est supprimé que lors du premier balayage qui s'exécute *après* que son âge de dernière vue ait déjà dépassé `heartbeat_timeout`, donc le temps réel maximum qu'un NF reste silencieux après être devenu inactif est jusqu'à `heartbeat_timeout + cleanup_interval`. Gardez `cleanup_interval` bien en dessous de `heartbeat_timeout` (les valeurs par défaut sont 10 s et 30 s) afin que cet excès reste faible ; un `cleanup_interval` supérieur à `heartbeat_timeout` laisserait des NF morts annoncés plus longtemps que prévu.

plmn_list

`plmn_list` est l'ensemble des PLMN **que ce NRF sert**, exprimé sous forme de liste de mappages `%{mcc, mnc}`. Il est distinct de la `plmnList` portée à l'intérieur du profil de chaque NF enregistré. Cette liste par NF est celle contre laquelle la découverte filtre, tandis que `plmn_list` décrit le(s) réseau(x) servi(s) par le NRF.

Sous-champ	Type	Description
`mcc`	chaîne	Code de pays mobile (3 chiffres)
`mnc`	chaîne	Code de réseau mobile (2 à 3 chiffres)

S'il n'est pas défini, `plmn_list` par défaut à une seule entrée construite à partir des clés `mcc`/`mnc` de premier niveau. Configurez plusieurs entrées pour un NRF partagé / multi-opérateur, par exemple :

```
config :omninrf,
  plmn_list: [%{mcc: "999", mnc: "70"}, %{mcc: "001", mnc: "01"}]
```

Le matching de découverte (le paramètre de requête `plmn-id`) est évalué par rapport à la `plmnList` de chaque NF, et non par rapport à `plmn_list`. Selon la TS 29.510, un NF enregistré qui n'annonce **aucune** `plmnList` est supposé appartenir au PLMN du NRF et correspond donc à tout filtre `plmn-id`.

API de gestion / OAM

Aux côtés du SBI 5G, OmniNRF expose une API REST de gestion destinée aux opérateurs pour l'inspection et le contrôle en temps réel. Elle est servie sur **HTTPS** via le listener de gestion api_ex (port par défaut `8443`). Les corps de requête et de réponse sont en JSON.

Méthode	Chemin	Description
GET	`/api/oam/config`	Voir la configuration d'exécution actuelle (`sbi_addr`, `sbi_port`, `heartbeat_timeout`, `cleanup_interval`, `plmn_list`, `log_level`)
PATCH	`/api/oam/config/:id`	Mettre à jour la configuration d'exécution. Seuls `heartbeat_timeout`, `cleanup_interval` et `log_level` peuvent être modifiés ; toutes les valeurs fournies sont validées avant d'être appliquées (400 en cas de valeur invalide)
POST	`/api/oam/log_level`	Changer le niveau de log à l'exécution. Corps `{"level": "..."}` ; les niveaux valides sont `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug` (400 sinon)
POST	`/api/oam/heartbeat`	Actualiser manuellement l'horodatage du heartbeat pour un NF. Corps `{"nf_instance_id": "..."}` (404 si l'instance n'est pas enregistrée)
GET	`/api/oam/nf_uptime/:id`	Afficher l'âge d'enregistrement/heartbeat et l'état pour une instance NF (404 si non enregistrée)
POST	`/api/oam/nf_purge`	Déclencher un balayage immédiat de purge des instances obsolètes

Méthode	Chemin	Description
		du registre ; rapporte combien d'instances NF ont été supprimées
GET	`/api/oam/resources`	Afficher l'utilisation des ressources de la VM BEAM (mémoire, comptes de processus/ports, nombre de planificateurs, temps de fonctionnement, versions OTP/BEAM)
GET	`/api/oam/diagnostics`	Afficher les diagnostics de la VM BEAM (nœud, temps de fonctionnement, nœuds connectés, nombre de processus, file d'attente d'exécution, statistiques I/O et GC)
PATCH	`/api/oam/nf_status/:id`	Mettre à jour l'état stocké d'une instance NF

Le même listener sert également des points de terminaison d'état et d'inventaire en lecture seule : `/api/status/api`, `/api/status/license`, `/api/status/nrf`, `/api/status/nf`, `/api/api/nf_instances`, `/api/api/subscriptions`, et `/api/api/statistics`.

Inspection et purge des instances NF

`GET /api/oam/nf_uptime/:id` est le moyen direct de voir à quel point un NF est proche d'être éliminé. Il rapporte :

Champ	Signification
`last_heartbeat_ms_ago`	Âge de l'horodatage de dernière vue du NF, en millisecondes
`heartbeat_timeout`	Le seuil de vieillissement actuel, en millisecondes
`nf_type`, `nf_status`	Le type et l'état stocké du NF

Un NF est un candidat à la suppression une fois que `last_heartbeat_ms_ago` dépasse `heartbeat_timeout` ; l'écart entre les deux est à peu près combien de temps le NF peut rester silencieux avant que le balayage ne l'expire.

`POST /api/oam/nf_purge` exécute le même balayage d'instances obsolètes que le minuteur en arrière-plan, immédiatement et à la demande, et retourne combien d'instances ont été supprimées. Il ne supprime que les NF qui sont *déjà* obsolètes (dernière vue plus ancienne que `heartbeat_timeout`) ; il ne force pas la sortie des NF sains. Utilisez-le pour récupérer le registre d'un coup plutôt que d'attendre le prochain balayage programmé. Pour supprimer de force un NF actif spécifique, quelle que soit son ancienneté de heartbeat, désenregistrez-le directement avec `DELETE /api/api/nf_instances/:id`.

Journalisation

OmniNRF émet des **logs JSON structurés** via le formateur `OmniLogger.JsonFormatter`, avec toutes les métadonnées du logger incluses (`metadata: :all`). Chaque ligne de log porte des métadonnées par requête telles que l'ID d'instance NF et la procédure en cours de traitement, ce qui facilite la corrélation des événements d'inscription, de heartbeat, de découverte et de notification pour un NF donné.

Le niveau de log actif peut être changé à l'exécution sans redémarrage, soit via `POST /api/oam/log_level`, soit via la clé `log_level` de `PATCH /api/oam/config/:id`.

← Vue d'ensemble

Métriques OmniNRF

Les métriques sont exposées à `http://{host}:{prometheus_metrics_port}/metrics` (port par défaut `9568`) au format texte Prometheus.

Métriques du service NRF

Métrique	Type	Description
`omni_nrf_nf_registered_count`	Gauge	Nombre actuel d'instances NF enregistrées, par type
`omni_nrf_active_nf_instances_count`	Gauge	Nombre actuel d'instances NF actives, par type. Émis à chaque enregistrement, désenregistrement et expiration de heartbeat NF, donc il reflète l'ensemble actif en direct
`omni_nrf_registration_requests_count`	Counter	Total des demandes d'enregistrement NF reçues (tous types)
`omni_nrf_nf_registrations_total`	Counter	Total des enregistrements NF, par type
`omni_nrf_nf_deregistrations_total`	Counter	Total des désenregistrements NF, par type. L'étiquette `nf_type` est résolue à partir du profil NF stocké

Métrique	Type	Description
`omni_nrf_heartbeats_total`	Counter	Total des demandes de heartbeat NF, par type. L'étiquette `nf_type` est résolue à partir du profil NF stocké
`omni_nrf_discovery_requests_count`	Counter	Total des demandes de découverte reçues (tous types cibles)
`omni_nrf_discovery_requests_total`	Counter	Total des demandes de découverte, par type NF cible

Métriques BEAM VM

Métrique	Type	Description
`beam_memory_total`	Gauge	Mémoire totale BEAM (octets)
`beam_memory_processes`	Gauge	Mémoire allouée aux processus Erlang
`beam_memory_processes_used`	Gauge	Mémoire réellement utilisée par les processus
`beam_memory_system`	Gauge	Mémoire système (non-processus)
`beam_memory_atom`	Gauge	Mémoire totale des atomes
`beam_memory_atom_used`	Gauge	Mémoire des atomes utilisée
`beam_memory_binary`	Gauge	Mémoire binaire
`beam_memory_code`	Gauge	Mémoire de code
`beam_memory_ets`	Gauge	Mémoire de la table ETS (contient les magasins NF et d'abonnement)
`beam_processes_count`	Gauge	Nombre de processus Erlang
`beam_ports_count`	Gauge	Nombre de ports Erlang
`beam_atom_count`	Gauge	Nombre d'atomes
`beam_vm_uptime`	Gauge	Temps de fonctionnement de la VM (secondes)

PromQL utiles

Instances NF enregistrées par type :

```
omni_nrf_nf_registered_count
```

Taux de demande d'enregistrement sur 5 minutes :

```
rate(omni_nrf_registration_requests_count[5m])
```

Taux de demande de découverte par type NF cible :

```
sum by (target_nf_type)
(rate(omni_nrf_discovery_requests_total[5m]))
```

Taux de heartbeat par type NF (une chute vers zéro avertit des NF qui ont cessé de heartbeat). L'étiquette `nf_type` est résolue à partir du profil stocké, donc la répartition par type est significative :

```
sum by (nf_type) (rate(omni_nrf_heartbeats_total[5m]))
```

Churn net d'enregistrement (enregistrements moins désenregistrements), par type NF. Les deux compteurs portent un vrai `nf_type`, donc le churn peut être calculé par type :

```
sum by (nf_type) (rate(omni_nrf_nf_registrations_total[5m]))
  - sum by (nf_type) (rate(omni_nrf_nf_deregistrations_total[5m]))
```

Enregistrements par type NF (ce compteur porte un vrai `nf_type`) :

```
sum by (nf_type) (rate(omni_nrf_nf_registrations_total[5m]))
```

> Rendu du nom de la métrique : Les noms de télémétrie tels que `omni_nrf.nf.registered.count` sont exportés avec des points remplacés par des underscores. Confirmez les noms exacts exportés contre un scrape `/metrics` en direct, car ils peuvent varier selon la configuration de l'exportateur.

← Vue d'ensemble

Guide des opérations OmniNRF

OmniNRF implémente la fonction de dépôt de réseau (NRF) du cœur 5G. Le NRF est le registre central et le répertoire de services de l'architecture basée sur les services (SBA) : chaque autre fonction de réseau (NF) enregistre son profil auprès du NRF au démarrage, maintient cette inscription en vie avec des signaux de vie périodiques et interroge le NRF pour découvrir les points de terminaison des NF avec lesquels elle doit communiquer.

OmniNRF expose deux services SBI :

- **Nnrf_NFManagement** (`nnrf-nfm/v1`) : enregistrement, mise à jour, signal de vie, désinscription, récupération et abonnements/notifications de changement d'état de NF.
- **Nnrf_NFDiscovery** (`nnrf-disc/v1`) : résolution des instances de NF cibles par type, service, S-NSSAI, DNN et PLMN.

Tous les profils de NF et abonnements sont conservés dans des magasins en mémoire (ETS). Il n'y a pas de base de données externe ; le registre est reconstruit lorsque les NF se r��inscrivent après un redémarrage du NRF. Un balayage en arrière-plan élimine continuellement les instances de NF qui ont cessé d'envoyer des signaux de vie, de sorte que le registre ne fait jamais que promouvoir des NF qui sont manifestement vivants.

Rôle 3GPP et références de spécifications

Spécification	Pertinence
TS 23.501	Architecture système, rôle du NRF dans l'architecture basée sur les services (clause 6.2.6)
TS 29.510	Services Nnrf_NFManagement et Nnrf_NFDiscovery, modèles de données NFProfile / SearchResult / SubscriptionData
TS 29.510 §5.2	Nnrf_NFManagement : enregistrement, mise à jour, signal de vie (`heartBeatTimer`), désinscription, abonnements et notifications d'état
TS 29.510 §5.3	Nnrf_NFDiscovery : requête de découverte d'instance de NF et SearchResult
TS 29.510 §6.2.3.2.3.1	Paramètres de requête de découverte, ordre des résultats et pagination

OmniNRF est le **producteur** SBI pour `Nnrf_NFManagement` et `Nnrf_NFDiscovery`. Il ne s'enregistre pas lui-même auprès d'un autre NRF ; il est le registre vers lequel tout le monde pointe, donc il n'y a pas d'`nrf_uri` sortant dans sa configuration d'opérateur.

Points de terminaison SBI

L'écouteur SBI est servi sur **HTTP/2 (texte clair, h2c)** à `http://{sbi_addr}:{sbi_port}`. Les corps de requête et de réponse sont en JSON. Les corps PATCH utilisent `application/json-patch+json` ; la liste des instances de NF est retournée sous `application/3gppHal+json`.

Nnrf_NFManagement (`nnrf-nfm/v1`)

Méthode	Chemin	Description	Su
PUT	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Enregistrer ou réenregistrer un NF (corps complet NFProfile)	201 (nou / 20 (mis jour
PATCH	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Signal de vie (corps vide) ou mise à jour du profil JSON-Patch	204 de cont (sig vie) OK (pat
DELETE	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Désinscrire un NF	204 de cont
GET	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Récupérer un seul NFProfile	200
GET	`/nnrf-nfm/v1/nf-instances`	Lister les instances de NF (UriList ; prend en charge `nf-type`, `limit`, `page-number`, `page-size`)	200

Méthode	Chemin	Description	Su
POST	`/nnrf-nfm/v1/subscriptions`	S'abonner aux notifications d'état de NF	201
GET	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Lire un abonnement unique	200
PATCH	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Mettre à jour un abonnement (JSON-Patch)	200
DELETE	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Se désabonner	204 de cont

Les réponses PUT/PATCH/GET réussies sur une seule instance de NF portent un en-tête `ETag`. Les clients peuvent fournir `If-Match` sur PUT/PATCH pour rendre la mise à jour conditionnelle (412 Échec de la précondition en cas de non-correspondance).

Nnrf_NFDiscovery (`nnrf-disc/v1`)

Méthode	Chemin	Description	Succès
GET	`/nnrf-disc/v1/nf-instances`	Découvrir des instances de NF par type et filtres optionnels	200 OK

Lorsqu'un résultat de découverte est paginé (un `limit` a été fourni et d'autres résultats restent), la réponse porte un en-tête `Link: <...>; rel="next"` pointant vers la page suivante.

Paramètres de requête de découverte

Paramètre	Requis	Description
`target-nf-type`	Oui	Type de NF recherché (par exemple `UDM`, `SMF`, `PCF`)
`requester-nf-type`	Oui	Type de NF faisant la demande (par exemple `AMF`)
`service-names`	Non	Noms de services séparés par des virgules ; un NF correspond s'il offre un service listé
`plmn-id`	Non	ID PLMN encodé en JSON, par exemple `{"mcc":"999","mnc":"70"}`. Un NF sans `plmnList` correspond à tout PLMN
`snssais`	Non	S-NSSAI encodé en JSON (objet ou tableau), par exemple `[{"sst":1,"sd":"000001"}]`. Correspond à `sNssais`, `allowedNssais` et `allowedNssais` par service du NF
`dnn`	Non	Nom du réseau de données ; correspond à `allowedDnns` par service
`limit`	Non	Nombre maximum de résultats dans cette page
`offset`	Non	Index de départ dans l'ensemble de résultats (trié), utilisé pour la pagination

Les résultats sont retournés dans un `SearchResult` (`validityPeriod`, `nfInstances`, `numNfInstComplete`). Seuls les NF avec `nfStatus = REGISTERED` sont retournés.

Ordre des résultats

Les NF correspondants sont triés de sorte que l'instance la plus préférée apparaisse en premier, permettant à un consommateur de sélectionner

`nfInstances[0]` et de ne revenir qu'aux entrées suivantes si nécessaire. Les clés de tri, dans l'ordre, proviennent de chaque profil enregistré de NF :

Clé	Direction	Signification pour un opérateur
`priority`	croissant (plus bas en premier)	Préférence relative définie sur le profil NF. Plus bas gagne ; les NF sans `priority` sont triés en dernier (traités comme `65535`)
`capacity`	décroissant (plus élevé en premier)	Capacité annoncée du NF. Parmi les NF de même priorité, celui avec la capacité la plus élevée est préféré
`load`	croissant (plus bas en premier)	Charge actuelle rapportée par le NF. Parmi les NF autrement égaux, celui avec la charge la plus faible est préféré

Pour orienter les consommateurs vers un producteur particulier, définissez une `priority` plus basse (ou une `capacity` plus élevée) sur le profil enregistré de ce NF ; ce sont des propriétés du NF cible, pas du NRF.

Règles de correspondance des filtres

- **`service-names`** : un NF correspond s'il offre *au moins un* des services listés.
- **`snssais`** : correspond à l'union des `sNssais` du NF, de ses `allowedNssais` de niveau supérieur et des `allowedNssais` de chaque service. Un `sd` n'est requis que si la requête en fournit un.
- **`plmn-id`** et **`dnn`** : correspondent respectivement à `plmnList` du NF et à `allowedDnns` par service.

Pour chaque filtre optionnel, un NF qui ne fait *aucune* publicité de cet attribut est traité comme correspondant à n'importe quelle valeur pour celui-ci (par exemple, un NF sans `plmnList` correspond à n'importe quel `plmn-id`). Cette règle "non contraint signifie universel" suit le TS 29.510 et permet de découvrir des NF à provisionnement minimal.

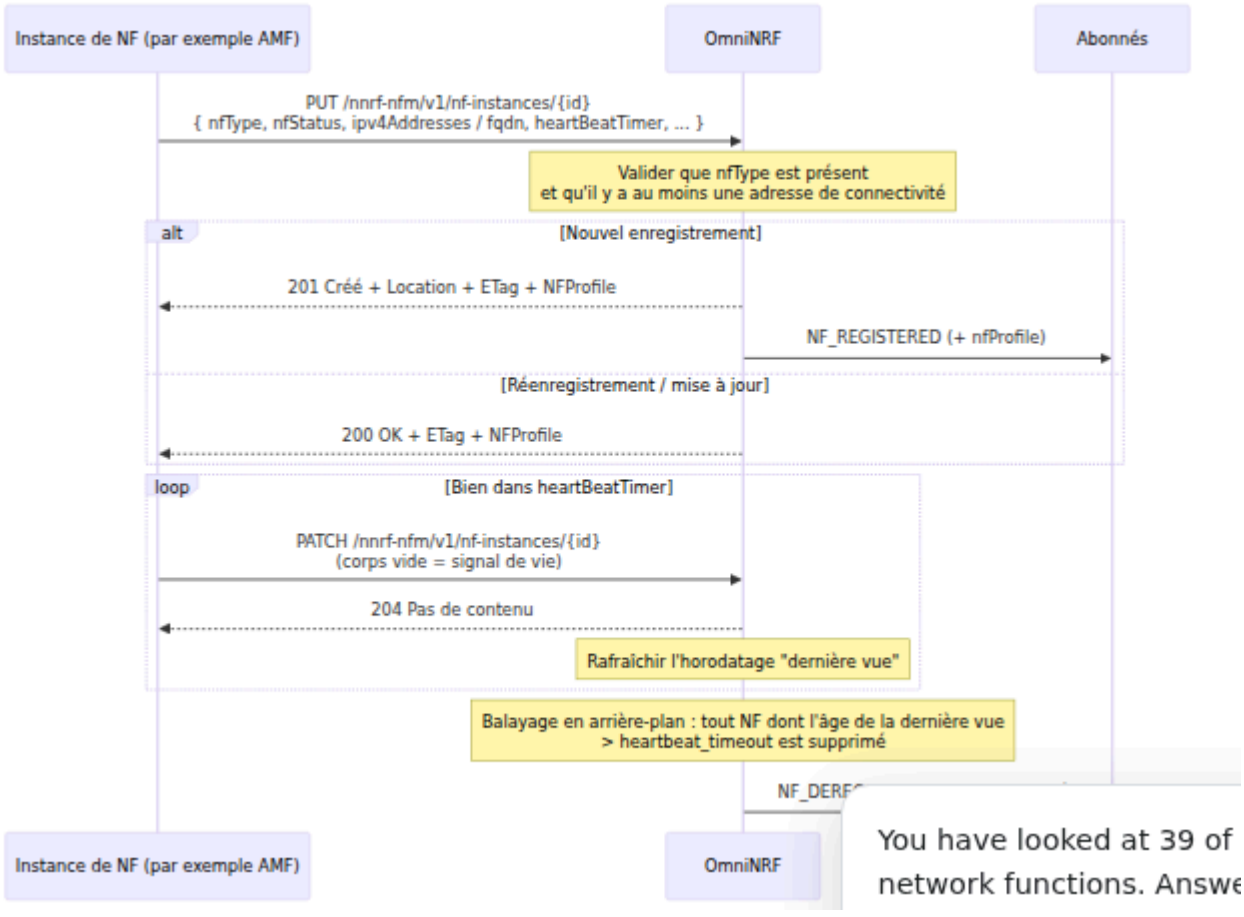
Pagination

`limit` limite le nombre d'instances retournées dans une page ; `offset` est l'index de départ dans l'ensemble de résultats triés. Lorsque d'autres résultats restent au-delà de la page, la réponse porte un en-tête `Link: <...>; rel="next"` dont l'`offset` pointe vers la page suivante. `numNfInstComplete` reflète toujours le nombre total de correspondances avant la pagination, permettant à un consommateur de paginer jusqu'à ce que `offset + len(nfInstances)` l'atteigne.

> Cette pagination de découverte SBI (`offset`/`limit` + `Link`) est distincte de la pagination de l'*endpoint* de liste d'instances de NF sous Nnrf_NFManagement, qui utilise plutôt `limit`/`page-number`/`page-size` sur un `3gppHal+json` UriList.

Procédures clés

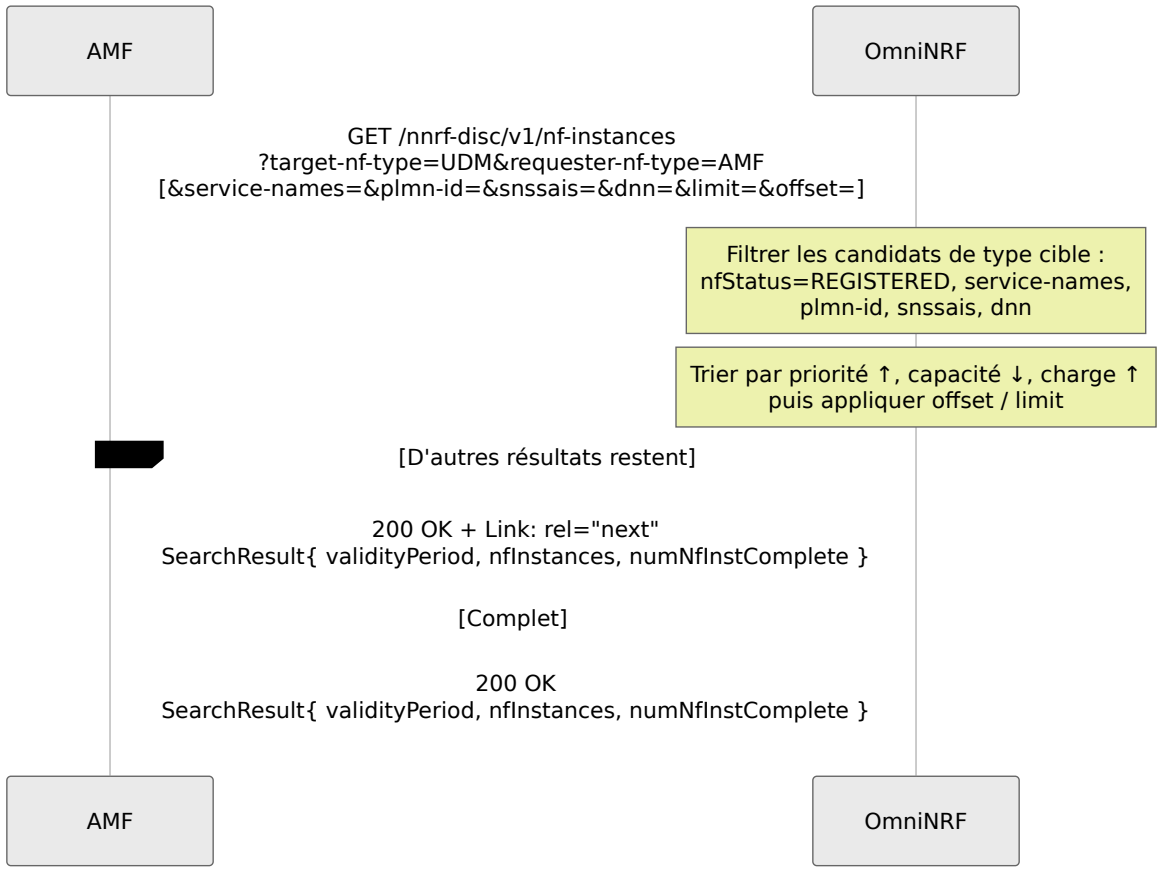
Enregistrement de NF et signal de vie



L'enregistrement nécessite `nfType` et `nfStatus` dans le corps (un corps manquant l'un ou l'autre est rejeté avec 400 `MANDATORY_IE_MISSING`) **et** au moins une adresse de connectivité (`fqdn`, `ipv4Addresses` ou `ipv6Addresses`). Un corps sans adresse de connectivité est rejeté avec 400 (mauvaise requête).

Le NRF **reçoit** des signaux de vie d'autres NF ; il ne les envoie pas. Chaque fois qu'un NF envoie un signal de vie PATCH (ou tout enregistrement/mise à jour) pour son instance, le NRF enregistre un nouvel horodatage "dernière vue" contre ce profil NF. Un balayage en arrière-plan fonctionne en continu et supprime tout NF dont l'horodatage de dernière vue est plus ancien que `heartbeat_timeout` millisecondes. Lorsqu'un NF est éliminé de cette manière, il est désinscrit du magasin et une notification d'état `NF_DEREGISTERED` est envoyée à tous les abonnés correspondants, afin que les consommateurs ne reçoivent plus un point de terminaison mort.

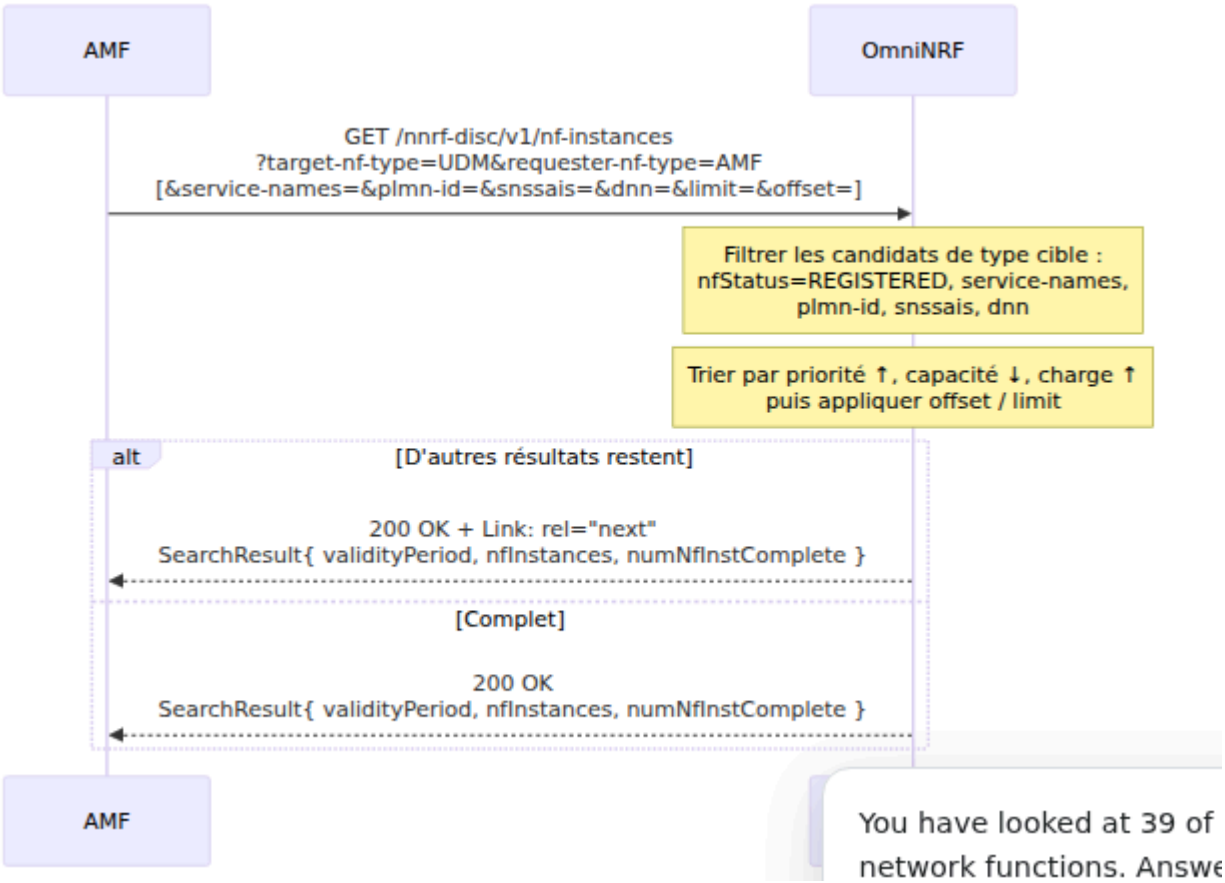
Découverte de NF



Les deux `target-nf-type` et `requester-nf-type` sont obligatoires ; omettre l'un ou l'autre entraîne un 400. Une liste `nfInstances` vide (avec

`numNfInstComplete: 0`) est une réponse valide signifiant qu'aucune correspondance n'a été trouvée.

Abonnements et notifications d'état de NF



Les notifications sont livrées via le client HTTP/2 SBI partagé (piscine) avec une tentative de réessai limitée : chaque notification est tentée jusqu'à `notification_max_attempts` fois (par défaut 4), en utilisant un backoff exponentiel avec une base de `notification_backoff_ms` (par défaut 500 ms, doublant à chaque réessai). Un abonné qui répond avec un code non-2xx ou qui est brièvement inaccessible est réessayé plutôt que supprimé après une seule tentative ; après la dernière tentative échouée, le NRF abandonne et enregistre un avertissement. Un abonné inaccessible est enregistré mais ne fait jamais planter le NRF. Les types d'événements (TS 29.510 clause 5.2.2.8) sont mappés à partir de l'état de NF : `REGISTERED` à `NF_REGISTERED` (le `nfProfile` complet est inclus), `DEREGISTERED`/`SUSPENDED` à `NF_DEREGISTERED`, d'autres changements d'état à `NF_PROFILE_CHANGED`. Un abonnement avec `subscrCond.nfType` ne reçoit que des notifications pour ce type de NF. Si `validityTime` (ISO 8601) est fourni, l'abonnement est automatiquement supprimé lorsqu'il expire.

Transitions d'état de NF et comment elles atteignent les abonnés

Déclencheur	Notification résultante	Également supprimé du registre ?
NF s'enregistre (PUT, nouveau)	`NF_REGISTERED` (avec `nfProfile`)	Non, ajouté
Changements de profil NF (PATCH / mise à jour OAM vers un état non terminal)	`NF_PROFILE_CHANGED`	Non
NF se désinscrit (DELETE)	`NF_DEREGISTERED`	Oui
NF éliminé (signaux de vie manqués, balayage en arrière-plan)	`NF_DEREGISTERED`	Oui
Purge manuelle (`POST /api/oam/nf_purge`) d'un NF obsolète	`NF_DEREGISTERED`	Oui

Un NF **éliminé** et un NF **désinscrit** explicitement sont indiscernables pour les abonnés : tous deux apparaissent comme `NF_DEREGISTERED` et quittent tous deux le registre. La différence est purement la cause : la désinscription est un DELETE délibéré de la part du NF lui-même, tandis que l'élimination est le NRF qui retire un NF qui a cessé d'envoyer des signaux de vie. Un NF **suspendu** (`nfStatus = SUSPENDED`) est signalé comme `NF_DEREGISTERED` à des fins de notification et, parce que seuls les NF `REGISTERED` sont retournés par la découverte, il est effectivement caché des consommateurs tant que son profil reste dans le magasin.

← Vue d'ensemble

Dépannage OmniNRF

Un NF a disparu de la découverte bien qu'il soit en cours d'exécution

Presque toujours des battements de cœur manqués. Si un NF n'envoie pas un battement de cœur PATCH dans `heartbeat_timeout` (par défaut 30 s), le balayage en arrière-plan l'expire et il disparaît à la fois de la récupération et de la découverte. Vérifiez les journaux NRF pour `NRF: Expiring stale NF instance {id}`, et confirmez que le NF envoie des battements de cœur correctement dans le délai imparti. Augmenter `HEARTBEAT_TIMEOUT` donne plus de marge ; le NF doit également respecter le `heartBeatTimer` renvoyé dans sa réponse d'enregistrement.

Enregistrement rejeté avec 400

Le NFProfile manque d'un élément obligatoire ou en contient un malformé. Le NRF nécessite à la fois `nfType` et au moins une adresse de connectivité (`fqdn`, `ipv4Addresses`, ou `ipv6Addresses`). Un 400 avec `MANDATORY_IE_MISSING` signifie que `nfType`/`nfStatus` étaient absents du corps PUT ; un 400 dont le détail mentionne la connectivité signifie qu'aucune adresse n'a été fournie.

Au-delà de ces éléments obligatoires, le NRF impose également une validation structurelle approfondie du reste du profil et renvoie un 400 avec la cause `MANDATORY_IE_INCORRECT` lorsqu'un élément présent est malformé (selon TS 29.510). Causes courantes :

- `nfServices` : chaque service a besoin d'un `serviceInstanceId`, d'un `serviceName`, d'un `scheme` valide, d'un `nfServiceStatus` valide, et d'un tableau `versions` non vide où chaque entrée porte `apiVersionInUri`.
- `sNssais` : chaque entrée a besoin d'un entier `sst` dans la plage de 0 à 255, et d'un `sd` (lorsqu'il est présent) de exactement 6 chiffres

hexadécimaux.

- `plmnList` : les entrées doivent être des objets PLMN bien formés.
- `capacity` et `priority` doivent être des entiers dans la plage de 0 à 65535 ; `load` doit être un entier dans la plage de 0 à 100.

Les éléments optionnels absents sont acceptés ; seul un élément présent mais malformé est rejeté. Le 400 `detail` nomme le champ fautif, ce qui indique directement ce qu'il faut corriger.

PATCH rejeté avec 415 Type de média non pris en charge

Le type de contenu PATCH a été défini sur autre chose que `application/json-patch+json`. Envoyez soit ce type de média pour les mises à jour JSON-Patch, soit envoyez un battement de cœur véritablement vide (aucun type de contenu n'est toléré pour les battements de cœur).

Mise à jour/battement de cœur/suppression renvoie 404

Le `nfInstanceId` n'est pas dans le registre, généralement parce que le NF a déjà été expiré pour des battements de cœur manqués, ou il n'a jamais terminé l'enregistrement. Réenregistrez avec un PUT avant de patcher ou d'envoyer un battement de cœur.

PUT/PATCH renvoie 412 Échec de la précondition

Un en-tête `If-Match` a été envoyé dont l'ETag ne correspond pas au profil actuellement stocké (le profil a changé sous le client). Re-`GET` l'instance pour obtenir le `ETag` actuel, puis réessayez.

La découverte renvoie une liste `nfInstances` vide

- Confirmez que le NF cible est enregistré et que son `nfStatus` stocké est `REGISTERED` (seuls les NF enregistrés sont renvoyés).
- `target-nf-type` est comparé de manière sensible à la casse avec le `nfType` stocké.
- Si des filtres `plmn-id`, `snssais`, ou `dnn` sont fournis, vérifiez qu'ils sont bien formés et que le profil du NF cible les annonce réellement (un NF qui n'annonce aucun attribut donné ne correspond à aucune valeur pour celui-ci).
- Assurez-vous que `requester-nf-type` est présent. Son absence entraîne un 400, pas une liste vide.

Les notifications de statut ne sont pas reçues

- Confirmez que `nfStatusNotificationUri` dans l'abonnement est accessible depuis l'hôte NRF.
- La livraison utilise une tentative limitée avec un retour exponentiel (jusqu'à `notification_max_attempts` essais, par défaut 4, avec un délai de base de `notification_backoff_ms`, par défaut 500 ms, doublant à chaque nouvelle tentative). Une nouvelle tentative est enregistrée comme `NRF: Retrying status notification to ...`. Si chaque tentative échoue, le NRF abandonne et enregistre `NRF: Giving up status notification to ... after N attempts`. Recherchez dans les journaux NRF ces deux lignes pour voir si la livraison a été réessayée et finalement abandonnée.
- Si `subscrCond.nfType` est défini, seules les modifications de statut pour ce type de NF sont livrées.

Le registre est vide après un redémarrage du NRF

Prévisible. Les magasins NF et d'abonnement sont en mémoire (ETS). Après un redémarrage, le registre se remplit à nouveau lorsque les NF se réenregistrent lors de leur prochain cycle d'enregistrement/battement de cœur ; il n'y a pas de persistance à recharger.