← Visão Geral

Referência de Configuração do OmniNRF

OmniNRF lê sua configuração de operador de `config/runtime.exs`, que mapeia variáveis de ambiente para o ambiente da aplicação `:omninrf`. Cada chave voltada para o operador está listada abaixo com sua variável de ambiente e padrão.

Configuração em Tempo de Execução

```
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
```

Tabela de Parâmetros

Parâmetro	Tipo	Requerido	Padrão
`sbi_scheme`	string	Não	`http`
`sbi_addr`	string	Não	`127.0.0.1`
`sbi_port`	integer	Não	`7777`

Parâmetro	Tipo	Requerido	Padrão
`mcc`	string	Não	`999`
`mnc`	string	Não	`70`
`heartbeat_timeout`	integer (ms)	Não	`30000`

Parâmetro	Tipo	Requerido	Padrão
`cleanup_interval`	integer (ms)	Não	`10000`
`notification_max_attempts`	integer	Não	`4`
`notification_backoff_ms`	integer (ms)	Não	`500`
`discovery_validity_period`	integer (s)	Não	`3600`

Parâmetro	Tipo	Requerido	Padrão
`plmn_list`	list	Não	`[%{mcc: mcc, mnc:`
`prometheus_metrics_port`	integer	Não	`9568`
`logger format`	tuple	Não	`{OmniLogger.JsonF :format}`
`logger metadata`	atom/list	Não	`:all`

Parâmetro	Tipo	Requerido	Padrão

> `SBI_ADDR` tem como padrão `127.0.0.1` (loopback). Em qualquer implantação multi-host ou em contêiner, isso deve ser substituído por um endereço que os outros NFs possam alcançar, caso contrário, nenhum NF poderá se registrar ou descobrir.

Semântica de Envelhecimento de Heartbeat

O NRF **recebe** heartbeats de outros NFs; ele não os envia. Cada vez que um NF envia um heartbeat PATCH (ou qualquer registro/atualização) para sua instância, o NRF registra um novo timestamp de "última vista" contra esse perfil de NF.

Uma varredura em segundo plano é executada a cada `cleanup_interval` milissegundos e remove qualquer NF cujo timestamp de última vista seja mais antigo que `heartbeat_timeout` milissegundos. Quando um NF é removido dessa forma, ele é desregistrado do armazenamento e uma notificação de status `NF_DEREGISTERED` é enviada para quaisquer assinantes correspondentes, para que os consumidores parem de receber um endpoint morto.

No registro, se um NF não incluir seu próprio `heartBeatTimer`, o NRF preenche um igual a `heartbeat_timeout / 1000` segundos como uma dica de com que frequência espera ser heartbeatado. Os operadores devem garantir que os NFs façam heartbeat bem dentro de `heartbeat_timeout` (um valor confortavelmente menor que o timeout) para que a jitter transitória não cause a expiração de um NF ativo.

O intervalo da varredura em segundo plano (`cleanup_interval`) e o limite de obsolescência (`heartbeat_timeout`) são independentes. Um NF é removido apenas na primeira varredura que é executada *após* sua idade de última vista já ter ultrapassado `heartbeat_timeout`, então o tempo real no pior caso que

um NF permanece após ficar em silêncio é de até `heartbeat_timeout + cleanup_interval`. Mantenha `cleanup_interval` bem abaixo de `heartbeat_timeout` (os padrões são 10 s e 30 s) para que esse excesso permaneça pequeno; um `cleanup_interval` maior que `heartbeat_timeout` deixaria NFs mortos anunciados por mais tempo do que o pretendido.

plmn_list

`plmn_list` é o conjunto de PLMNs **que este NRF atende**, expresso como uma lista de mapeamentos `%{mcc, mnc}`. É distinto da `plmnList` contida no próprio perfil de cada NF registrado. Essa lista por NF é o que os filtros de descoberta utilizam, enquanto `plmn_list` descreve a(s) rede(s) atendida(s) pelo NRF.

Subcampo	Tipo	Descrição
`mcc`	string	Código do País Móvel (3 dígitos)
`mnc`	string	Código da Rede Móvel (2 a 3 dígitos)

Se não definido, `plmn_list` tem como padrão uma única entrada construída a partir das chaves de nível superior `mcc`/`mnc`. Configure várias entradas para um NRF compartilhado / multi-operador, por exemplo:

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

A correspondência de descoberta (o parâmetro de consulta `plmn-id`) é avaliada em relação à `plmnList` de cada NF, não em relação a `plmn_list`. De acordo com a TS 29.510, um NF registrado que não anuncia **nenhuma** `plmnList` é assumido como pertencente à PLMN do NRF e, portanto, corresponde a qualquer filtro `plmn-id`.

API de Gerenciamento / OAM

Juntamente com o SBI 5G, o OmniNRF expõe uma API REST de gerenciamento voltada para o operador para inspeção e controle em tempo de execução. Ela é servida sobre **HTTPS** no listener de gerenciamento api_ex (porta padrão `8443`). Os corpos de requisição e resposta são JSON.

Método	Caminho	Descrição
GET	`/api/oam/config`	Visualizar a configuração atual em tempo de execução (`sbi_addr`, `sbi_port`, `heartbeat_timeout`, `cleanup_interval`, `plmn_list`, `log_level`)
PATCH	`/api/oam/config/:id`	Atualizar a configuração em tempo de execução. Apenas `heartbeat_timeout`, `cleanup_interval` e `log_level` podem ser alterados; todos os valores fornecidos são validados antes de serem aplicados (400 em um valor inválido)
POST	`/api/oam/log_level`	Alterar o nível de log em tempo de execução. Corpo `{"level": "..."}`; níveis válidos são `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug` (400 caso contrário)
POST	`/api/oam/heartbeat`	Atualizar manualmente o timestamp de heartbeat para um NF. Corpo `{"nf_instance_id": "..."}` (404 se a instância não estiver registrada)
GET	`/api/oam/nf_uptime/:id`	Mostrar idade de registro/heartbeat e status para uma instância NF (404 se não registrada)
POST	`/api/oam/nf_purge`	Acionar uma varredura imediata de purga de instâncias obsoletas do

Método	Caminho	Descrição
		registro; relata quantas instâncias NF foram removidas
GET	`/api/oam/resources`	Mostrar uso de recursos da VM BEAM (memória, contagem de processos/portas, contagem de agendadores, tempo de atividade, versões OTP/BEAM)
GET	`/api/oam/diagnostics`	Mostrar diagnósticos da VM BEAM (nó, tempo de atividade, nós conectados, contagem de processos, fila de execução, estatísticas de I/O e GC)
PATCH	`/api/oam/nf_status/:id`	Atualizar o status armazenado de uma instância NF

O mesmo listener também serve endpoints de status e inventário somente leitura: `/api/status/api`, `/api/status/license`, `/api/status/nrf`, `/api/status/nf`, `/api/api/nf_instances`, `/api/api/subscriptions`, e `/api/api/statistics`.

Inspecionando e purgando instâncias NF

`GET /api/oam/nf_uptime/:id` é a maneira direta de ver quão próximo um NF está de ser removido. Ele relata:

Campo	Significado
`last_heartbeat_ms_ago`	Idade do timestamp de última vista do NF, em milissegundos
`heartbeat_timeout`	O limite de obsolescência atual, em milissegundos
`nf_type`, `nf_status`	O tipo e o status armazenado do NF

Um NF é um candidato para remoção uma vez que `last_heartbeat_ms_ago` exceda `heartbeat_timeout`; a diferença entre os dois é aproximadamente quanto mais tempo o NF pode ficar em silêncio antes que a varredura o expire.

`POST /api/oam/nf_purge` executa a mesma varredura de instâncias obsoletas que o temporizador em segundo plano, imediatamente e sob demanda, e retorna quantas instâncias foram removidas. Ele remove apenas NFs que já estão obsoletos (última vista mais antiga que `heartbeat_timeout`); não força a remoção de NFs saudáveis. Use-o para recuperar o registro de uma só vez, em vez de esperar pela próxima varredura programada. Para remover forçosamente um NF ativo específico, independentemente de sua idade de heartbeat, desregistre-o diretamente com `DELETE /api/api/nf_instances/:id`.

Registro

OmniNRF emite **logs JSON estruturados** através do formatador `OmniLogger.JsonFormatter`, com todos os metadados do logger incluídos (`metadata: :all`). Cada linha de log carrega metadados por requisição, como o ID da instância NF e o procedimento sendo tratado, o que torna fácil correlacionar eventos de registro, heartbeat, descoberta e notificação para um dado NF.

O nível de log ativo pode ser alterado em tempo de execução sem reinicialização, seja através de `POST /api/oam/log_level` ou via a chave

`log_level` de `PATCH /api/oam/config/:id`.

← Visão Geral

Métricas do OmniNRF

As métricas são expostas em `http://{host}:{prometheus_metrics_port}/metrics` (porta padrão `9568`) no formato de texto do Prometheus.

Métricas do Serviço NRF

Métrica	Tipo	Descrição	
`omni_nrf_nf_registered_count`	Gauge	Número atual de instâncias NF registradas, por tipo	`nf_t`
`omni_nrf_active_nf_instances_count`	Gauge	Número atual de instâncias NF ativas, por tipo. Emitido em cada registro, cancelamento de registro e expiração de heartbeat, portanto reflete o conjunto ativo ao vivo	`nf_t`
`omni_nrf_registration_requests_count`	Counter	Total de solicitações de registro NF recebidas (todos os tipos)	(nen
`omni_nrf_nf_registrations_total`	Counter	Total de registros NF, por tipo	`nf_t`
`omni_nrf_nf_deregistrations_total`	Counter	Total de cancelamentos	`nf_t`

Métrica	Tipo	Descrição	L
		de registro NF, por tipo. O rótulo `nf_type` é resolvido a partir do perfil NF armazenado	
`omni_nrf_heartbeats_total`	Counter	Total de solicitações de heartbeat NF, por tipo. O rótulo `nf_type` é resolvido a partir do perfil NF armazenado	`nf_t`
`omni_nrf_discovery_requests_count`	Counter	Total de solicitações de descoberta recebidas (todos os tipos de destino)	(nen
`omni_nrf_discovery_requests_total`	Counter	Total de solicitações de descoberta, por tipo de NF de destino	`targ`

Métricas do BEAM VM

Métrica	Tipo	Descrição
`beam_memory_total`	Gauge	Memória total do BEAM (bytes)
`beam_memory_processes`	Gauge	Memória alocada para processos Erlang
`beam_memory_processes_used`	Gauge	Memória realmente usada pelos processos
`beam_memory_system`	Gauge	Memória do sistema (não-processo)
`beam_memory_atom`	Gauge	Memória total de átomos
`beam_memory_atom_used`	Gauge	Memória de átomos usada
`beam_memory_binary`	Gauge	Memória binária
`beam_memory_code`	Gauge	Memória de código
`beam_memory_ets`	Gauge	Memória da tabela ETS (contém os armazéns de NF e assinatura)
`beam_processes_count`	Gauge	Número de processos Erlang
`beam_ports_count`	Gauge	Número de portas Erlang
`beam_atom_count`	Gauge	Número de átomos
`beam_vm_uptime`	Gauge	Tempo de atividade da VM (segundos)

PromQL Útil

Instâncias NF registradas por tipo:

```
omni_nrf_nf_registered_count
```

Taxa de solicitações de registro ao longo de 5 minutos:

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

Taxa de solicitações de descoberta por tipo de NF de destino:

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

Taxa de heartbeat por tipo de NF (uma queda em direção a zero alerta sobre NFs que pararam de enviar heartbeat). O rótulo `nf_type` é resolvido a partir do perfil armazenado, portanto a divisão por tipo é significativa:

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

Churn líquido de registro (registros menos cancelamentos de registro), por tipo de NF. Ambos os contadores carregam um `nf_type` real, portanto o churn pode ser calculado por tipo:

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

Registros por tipo de NF (este contador carrega um `nf_type` real):

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

> Renderização do nome da métrica: Nomes de telemetria como `omni_nrf.nf.registered.count` são exportados com pontos substituídos por sublinhados. Confirme os nomes exportados exatos contra uma coleta ao vivo de `/metrics`, pois eles podem variar com a configuração do exportador.

← Visão Geral

Guia de Operações do OmniNRF

OmniNRF implementa a Função de Repositório de Rede (NRF) do Core 5G. O NRF é o registro central e diretório de serviços da Arquitetura Baseada em Serviços (SBA): cada outra Função de Rede (NF) registra seu perfil no NRF na inicialização, mantém esse registro ativo com batimentos cardíacos periódicos e consulta o NRF para descobrir os endpoints das NFs com as quais precisa se comunicar.

OmniNRF expõe dois serviços SBI:

- **Nnrf_NFManagement** (`nnrf-nfm/v1`): registro, atualização, batimento cardíaco, desregistro, recuperação e notificações/assinaturas de mudança de status da NF.
- **Nnrf_NFDiscovery** (`nnrf-disc/v1`): resolução de instâncias de NF alvo por tipo, serviço, S-NSSAI, DNN e PLMN.

Todos os perfis de NF e assinaturas são mantidos em armazenamentos em memória (ETS). Não há banco de dados externo; o registro é reconstruído à medida que as NFs se registram novamente após uma reinicialização do NRF. Uma varredura em segundo plano continuamente remove instâncias de NF que pararam de enviar batimentos cardíacos, de modo que o registro só anuncia NFs que estão demonstravelmente ativas.

Referências de Papel e Especificação da 3GPP

Especificação	Relevância
TS 23.501	Arquitetura do sistema, papel do NRF na Arquitetura Baseada em Serviços (cláusula 6.2.6)
TS 29.510	Serviços Nnrf_NFManagement e Nnrf_NFDiscovery, modelos de dados NFProfile / SearchResult / SubscriptionData
TS 29.510 §5.2	Nnrf_NFManagement: registro, atualização, batimento cardíaco (`heartBeatTimer`), desregistro, assinaturas e notificações de status
TS 29.510 §5.3	Nnrf_NFDiscovery: consulta de descoberta de instância de NF e SearchResult
TS 29.510 §6.2.3.2.3.1	Parâmetros de consulta de descoberta, ordenação de resultados e paginação

OmniNRF é o **produtor** SBI para `Nnrf_NFManagement` e `Nnrf_NFDiscovery`. Ele não se registra com nenhum outro NRF; é o registro que tudo mais aponta, portanto não há um `nrf_uri` externo em sua configuração de operador.

Endpoints SBI

O ouvinte SBI é servido sobre **HTTP/2 (texto claro, h2c)** em `http://{sbi_addr}:{sbi_port}`. Os corpos de requisição e resposta são JSON. Os corpos PATCH usam `application/json-patch+json`; a lista de instâncias de NF é retornada como `application/3gppHal+json`.

Nnrf_NFManagement (`nnrf-nfm/v1`)

Método	Caminho	Descrição	Suc
PUT	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Registrar ou registrar novamente uma NF (corpo completo do NFProfile)	201 C (novo) OK (atual
PATCH	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Batimento cardíaco (corpo vazio) ou atualização de perfil JSON-Patch	204 S Conte (batim cardía 200 O (patch
DELETE	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Desregistrar uma NF	204 S Conte
GET	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	Recuperar um único NFProfile	200 O
GET	`/nnrf-nfm/v1/nf-instances`	Listar instâncias de NF (UriList; suporta `nf-type`, `limit`, `page-`	200 O

Método	Caminho	Descrição	Suc
		`number`, `page-size`)	
POST	`/nnrf-nfm/v1/subscriptions`	Assinar notificações de status da NF	201 C
GET	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Ler uma única assinatura	200 O
PATCH	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Atualizar uma assinatura (JSON-Patch)	200 O
DELETE	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	Cancelar assinatura	204 S Conte

Respostas bem-sucedidas de PUT/PATCH/GET em uma única instância de NF carregam um cabeçalho `ETag`. Os clientes podem fornecer `If-Match` em PUT/PATCH para tornar a atualização condicional (412 Falha de Pré-condição em caso de discrepância).

Nnrf_NFDiscovery (`nnrf-disc/v1`)

Método	Caminho	Descrição	Sucesso
GET	`/nnrf-disc/v1/nf-instances`	Descobrir instâncias de NF por tipo e filtros opcionais	200 OK

Quando um resultado de descoberta é paginado (um `limit` foi fornecido e mais resultados permanecem), a resposta carrega um cabeçalho `Link: <...>;`

`rel="next"` apontando para a próxima página.

Parâmetros de Consulta de Descoberta

Parâmetro	Obrigatório	Descrição
`target-nf-type`	Sim	Tipo de NF sendo pesquisado (por exemplo, `UDM`, `SMF`, `PCF`)
`requester-nf-type`	Sim	Tipo de NF que faz a solicitação (por exemplo, `AMF`)
`service-names`	Não	Nomes de serviços separados por vírgula; uma NF corresponde se oferecer qualquer serviço listado
`plmn-id`	Não	ID de PLMN codificado em JSON, por exemplo, `{"mcc":"999","mnc":"70"}`. Uma NF sem `plmnList` corresponde a qualquer PLMN
`snssais`	Não	S-NSSAI codificado em JSON (objeto ou array), por exemplo, `[{"sst":1,"sd":"000001"}]`. Correspondido contra os `sNssais` da NF, `allowedNssais` e `allowedNssais` por serviço
`dnn`	Não	Nome da Rede de Dados; correspondido contra `allowedDnns` por serviço
`limit`	Não	Número máximo de resultados nesta página
`offset`	Não	Índice inicial no conjunto de resultados (ordenado), usado para paginação

Os resultados são retornados em um `SearchResult` (`validityPeriod`, `nfInstances`, `numNfInstComplete`). Apenas NFs com `nfStatus = REGISTERED`

são retornadas.

Ordenação de Resultados

As NFs correspondentes são ordenadas de modo que a instância mais preferida apareça primeiro, permitindo que um consumidor selecione `nfInstances[0]` e apenas recorra a entradas posteriores. As chaves de ordenação, em ordem, vêm do próprio perfil registrado de cada NF:

Chave	Direção	Significado para um operador
`priority`	ascendente (menor primeiro)	Preferência relativa definida no perfil da NF. Menor vence; NFs sem `priority` são ordenadas por último (tratadas como `65535`)
`capacity`	descendente (maior primeiro)	Capacidade anunciada da NF. Entre NFs de igual prioridade, a de maior capacidade é preferida
`load`	ascendente (menor primeiro)	Carga atual relatada pela NF. Entre NFs de outra forma iguais, a menos carregada é preferida

Para direcionar consumidores a um produtor específico, defina uma `priority` mais baixa (ou uma `capacity` mais alta) no perfil registrado dessa NF; essas são propriedades da NF alvo, não do NRF.

Regras de Correspondência de Filtros

- **`service-names`**: uma NF corresponde se oferecer *qualquer um* dos serviços listados.
- **`snssais`**: correspondido contra a união dos `sNssais` da NF, seus `allowedNssais` de nível superior e `allowedNssais` de cada serviço. Um `sd` só é exigido para corresponder quando a consulta fornece um.
- **`plmn-id`** e **`dnn`**: correspondidos contra o `plmnList` da NF e `allowedDnns` por serviço, respectivamente.

Para cada filtro opcional, uma NF que anuncia *nenhum* atributo desse tipo é tratada como correspondente a qualquer valor para ele (por exemplo, uma NF sem `plmnList` corresponde a qualquer `plmn-id`). Esta regra de "não restrito significa universal" segue a TS 29.510 e mantém NFs minimamente provisionadas descobertas.

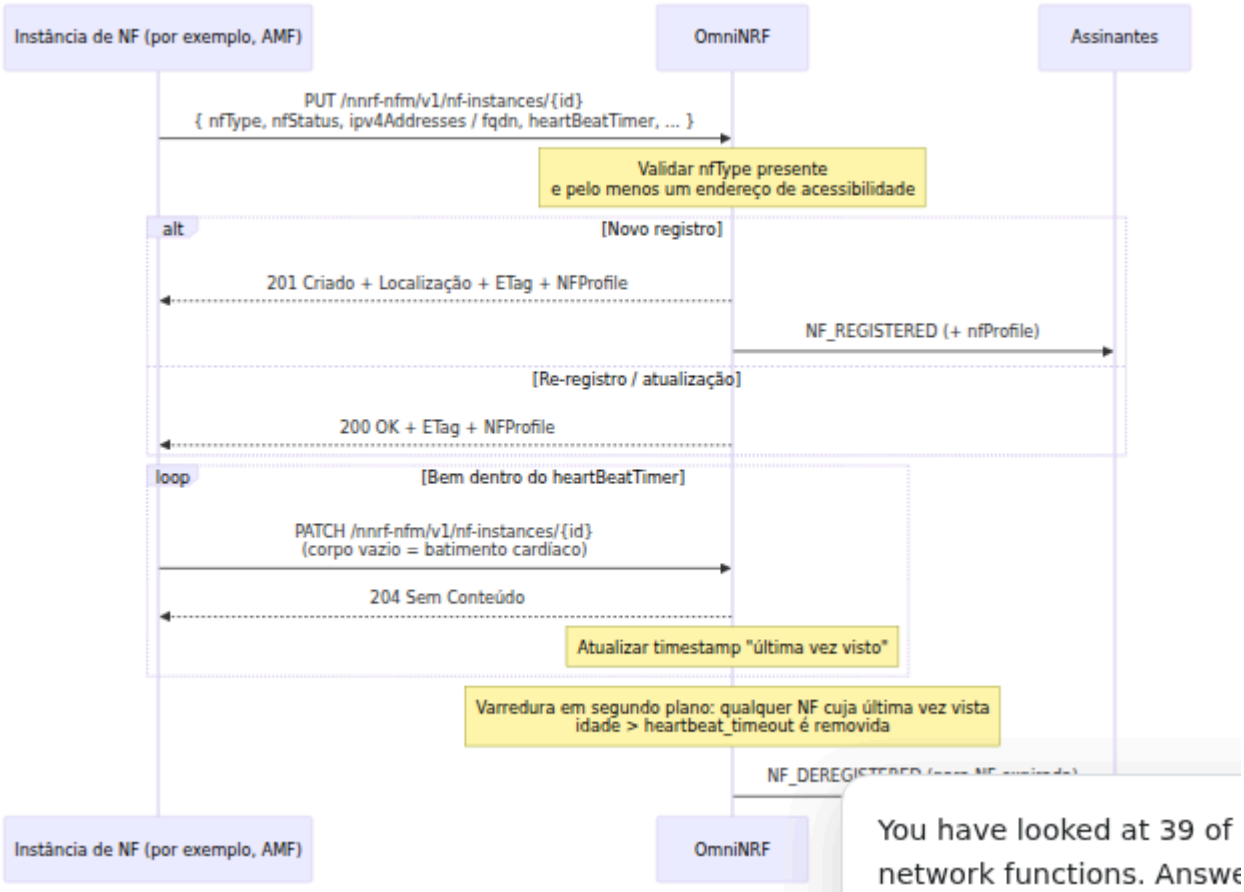
Paginação

`limit` limita o número de instâncias retornadas em uma página; `offset` é o índice inicial no conjunto de resultados ordenados. Quando mais resultados permanecem além da página, a resposta carrega um cabeçalho `Link: <...>; rel="next"` cujo `offset` aponta para a próxima página. `numNfInstComplete` sempre reflete a contagem total de correspondências antes da paginação, para que um consumidor possa paginar até que `offset + len(nfInstances)` a alcance.

> Esta paginação de descoberta SBI (`offset`/`limit` + `Link`) é separada da paginação do endpoint de *lista* de instâncias de NF sob Nnrf_NFManagement, que em vez disso usa `limit`/`page-number`/`page-size` sobre um `3gppHal+json` UriList.

Procedimentos Chave

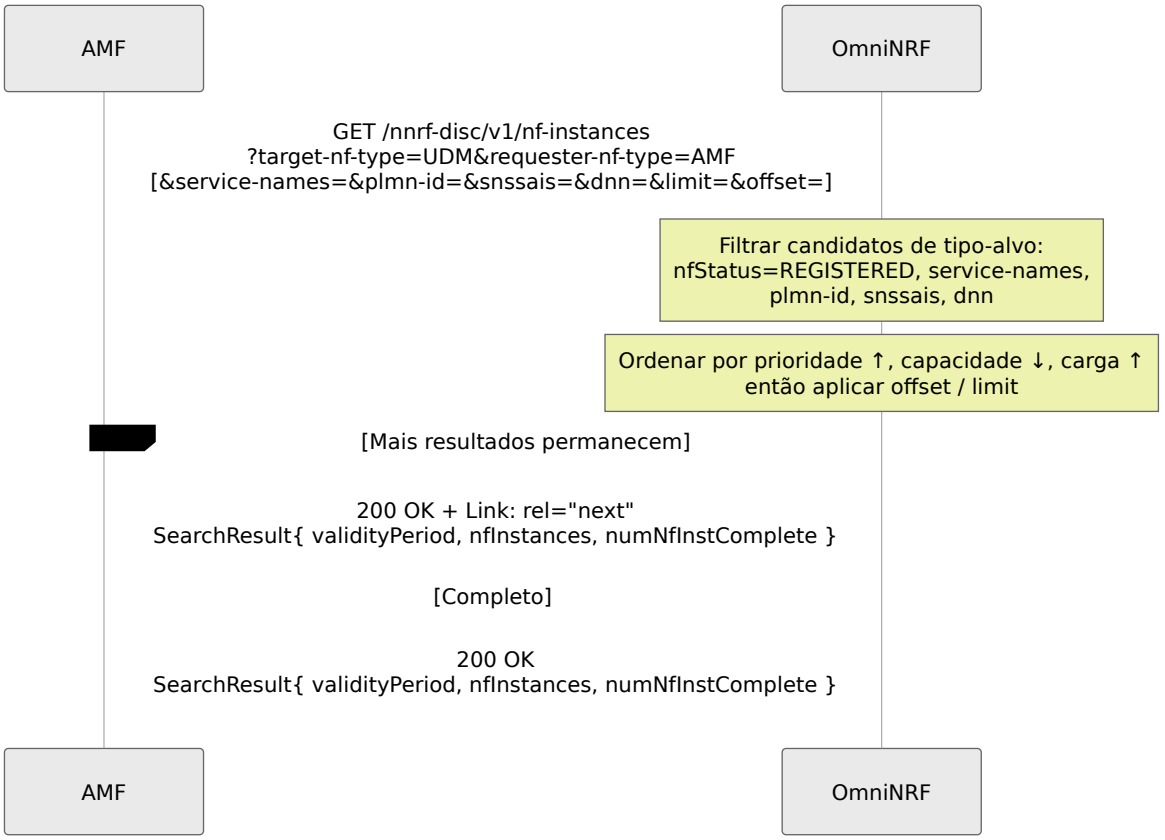
Registro de NF e Batimento Cardíaco



O registro requer `nfType` e `nfStatus` no corpo (um corpo que falta qualquer um é rejeitado com 400 `MANDATORY_IE_MISSING`) **e** pelo menos um endereço de acessibilidade (`fqdn`, `ipv4Addresses` ou `ipv6Addresses`). Um corpo sem endereço de acessibilidade é rejeitado com 400 (solicitação inválida).

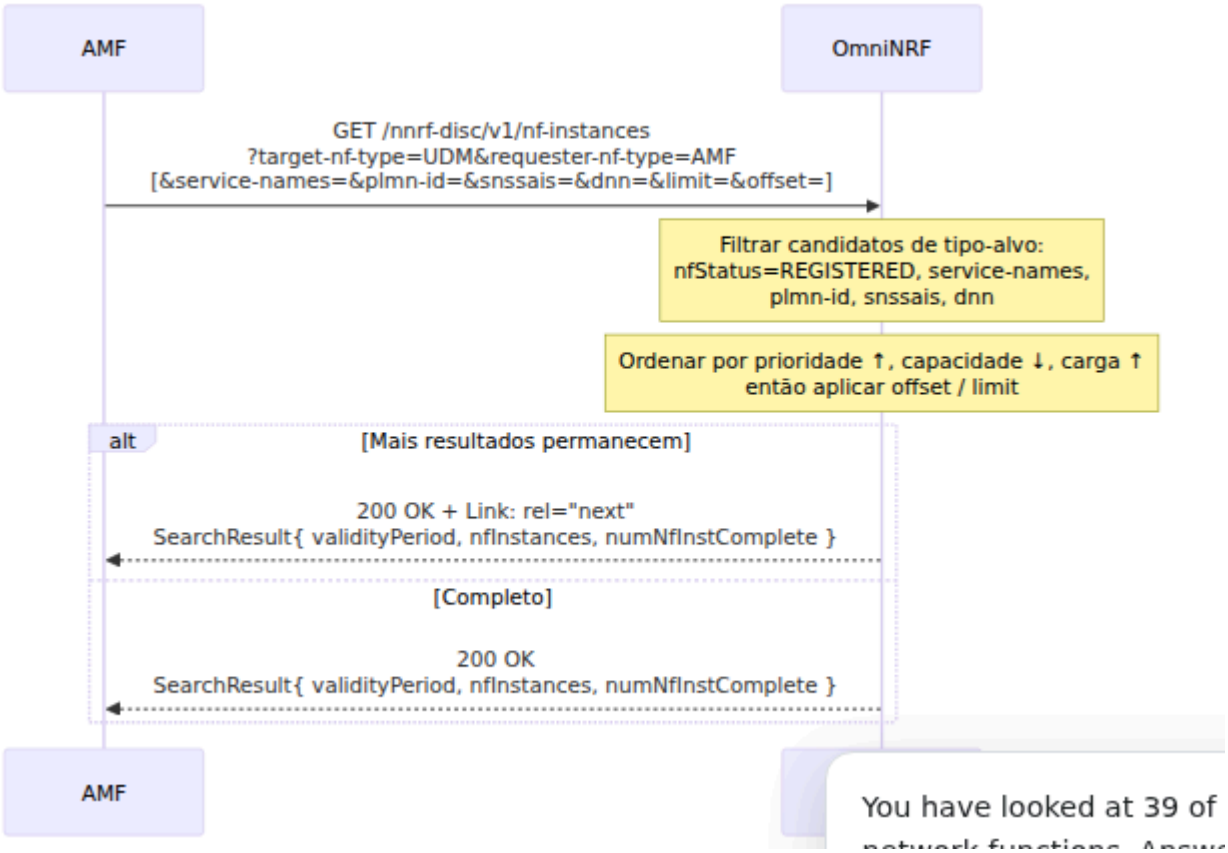
O NRF **recebe** batimentos cardíacos de outras NFs; ele não os envia. Cada vez que uma NF envia um batimento cardíaco PATCH (ou qualquer registro/atualização) para sua instância, o NRF registra um novo timestamp "última vez visto" contra esse perfil de NF. Uma varredura em segundo plano é executada continuamente e remove qualquer NF cujo timestamp de última vez visto seja mais antigo que `heartbeat_timeout` milissegundos. Quando uma NF é removida dessa forma, ela é desregistrada do armazenamento e uma notificação de status `NF_DEREGISTERED` é enviada a quaisquer assinantes correspondentes, para que os consumidores deixem de receber um endpoint inativo.

Descoberta de NF



Tanto `target-nf-type` quanto `requester-nf-type` são obrigatórios; omitir qualquer um resulta em 400. Uma lista `nfInstances` vazia (com `numNfInstComplete: 0`) é uma resposta válida significando que nada correspondeu.

Assinaturas e Notificações de Status da NF



As notificações são entregues através do cliente HTTP/2 SBI compartilhado (em pool) com uma tentativa limitada: cada notificação é tentada até `notification_max_attempts` vezes (padrão 4), usando retrocesso exponencial com base em `notification_backoff_ms` (padrão 500 ms, dobrando a cada tentativa). Um assinante que responde com um código não-2xx ou que está brevemente inacessível é refeito em vez de ser descartado após uma única tentativa; após a última tentativa falhada, o NRF desiste e registra um aviso. Um assinante inacessível é registrado, mas nunca derruba o NRF. Os tipos de eventos (cláusula 5.2.2.8 da TS 29.510) são mapeados a partir do status da NF: `REGISTERED` para `NF_REGISTERED` (o `nfProfile` completo é incluído), `DEREGISTERED`/`SUSPENDED` para `NF_DEREGISTERED`, outras mudanças de status para `NF_PROFILE_CHANGED`. Uma assinatura com `subscrCond.nfType` só recebe notificações para aquele tipo de NF. Se `validityTime` (ISO 8601) for fornecido, a assinatura é automaticamente excluída quando expira.

Transições de status da NF e como elas alcançam os assinantes

Gatilho	Notificação resultante	Também removido do registro?
NF registra (PUT, novo)	`NF_REGISTERED` (com `nfProfile`)	Não, adicionado
Mudanças de perfil da NF (PATCH / atualização OAM para um status não-terminal)	`NF_PROFILE_CHANGED`	Não
NF desregistra (DELETE)	`NF_DEREGISTERED`	Sim
NF removida (batimentos cardíacos perdidos, varredura em segundo plano)	`NF_DEREGISTERED`	Sim
Purga manual (`POST /api/oam/nf_purge`) de uma NF obsoleta	`NF_DEREGISTERED`	Sim

Uma NF **removida** e uma NF **desregistrada** explicitamente são indistinguíveis para os assinantes: ambas aparecem como `NF_DEREGISTERED` e ambas deixam o registro. A diferença é puramente a causa: o desregistro é um DELETE deliberado da própria NF, enquanto a remoção é o NRF retirando uma NF que parou de enviar batimentos cardíacos. Uma NF **suspensa** (`nfStatus = SUSPENDED`) é relatada como `NF_DEREGISTERED` para fins de notificação e, porque apenas NFs `REGISTERED` são retornadas pela descoberta, é efetivamente oculta dos consumidores enquanto seu perfil permanece no armazenamento.

← Visão Geral

Solução de Problemas do OmniNRF

Um NF desapareceu da descoberta, mesmo estando em execução

Quase sempre por causa de batimentos perdidos. Se um NF não enviar um batimento PATCH dentro do `heartbeat_timeout` (padrão de 30 s), a varredura em segundo plano expira e ele desaparece tanto da recuperação quanto da descoberta. Verifique os logs do NRF para `NRF: Expiring stale NF instance {id}`, e confirme se o NF está enviando batimentos corretamente dentro do tempo limite. Aumentar o `HEARTBEAT_TIMEOUT` oferece mais flexibilidade; o NF também deve respeitar o `heartBeatTimer` retornado em sua resposta de registro.

Registro rejeitado com 400

O NFProfile está faltando um elemento obrigatório ou contém um elemento malformado. O NRF requer tanto `nfType` quanto pelo menos um endereço de acessibilidade (`fqdn`, `ipv4Addresses` ou `ipv6Addresses`). Um 400 com `MANDATORY_IE_MISSING` significa que `nfType`/`nfStatus` estavam ausentes do corpo do PUT; um 400 cujo detalhe menciona acessibilidade significa que nenhum endereço foi fornecido.

Além desses elementos obrigatórios, o NRF também impõe uma validação estrutural profunda do restante do perfil e retorna um 400 com a causa `MANDATORY_IE_INCORRECT` quando um elemento presente está malformado (de acordo com TS 29.510). Causas comuns:

- `nfServices`: cada serviço precisa de um `serviceInstanceId`, um `serviceName`, um `scheme` válido, um `nfServiceStatus` válido e um array

`versions` não vazio onde cada entrada contém `apiVersionInUri`.

- `sNssais`: cada entrada precisa de um inteiro `sst` na faixa de 0 a 255, e um `sd` (quando presente) de exatamente 6 dígitos hexadecimais.
- `plmnList`: as entradas devem ser objetos PLMN bem formados.
- `capacity` e `priority` devem ser inteiros na faixa de 0 a 65535; `load` deve ser um inteiro na faixa de 0 a 100.

Elementos opcionais ausentes são aceitos; apenas um elemento presente, mas malformado, é rejeitado. O 400 `detail` nomeia o campo ofensivo, apontando diretamente o que deve ser corrigido.

PATCH rejeitado com 415 Tipo de Mídia Não Suportado

O Content-Type do PATCH foi definido como algo diferente de `application/json-patch+json`. Envie esse tipo de mídia para atualizações JSON-Patch, ou envie um batimento verdadeiramente vazio (nenhum Content-Type é tolerado para batimentos).

Atualização/batimento/exclusão retorna 404

O `nfInstanceId` não está no registro, tipicamente porque o NF já foi expirado por batimentos perdidos, ou nunca completou o registro. Re-registre com um PUT antes de aplicar patch ou batimento.

PUT/PATCH retorna 412 Pré-condição Falhou

Um cabeçalho `If-Match` foi enviado cujo ETag não corresponde ao perfil atualmente armazenado (o perfil mudou sob o cliente). Re-`GET` a instância para obter o `ETag` atual e, em seguida, tente novamente.

A descoberta retorna uma lista `nfInstances` vazia

- Confirme se o NF alvo está registrado e seu `nfStatus` armazenado é `REGISTERED` (apenas NFs registrados são retornados).
- `target-nf-type` é comparado de forma sensível a maiúsculas e minúsculas com o `nfType` armazenado.
- Se filtros `plmn-id`, `snssais` ou `dnn` forem fornecidos, verifique se estão bem formados e se o perfil do NF alvo realmente os anuncia (um NF que não anuncia nenhum dos atributos dados não corresponde a nenhum valor para ele).
- Certifique-se de que `requester-nf-type` está presente. Sua ausência resulta em 400, não em uma lista vazia.

As notificações de status não estão sendo recebidas

- Confirme se `nfStatusNotificationUri` na assinatura é acessível a partir do host NRF.
- A entrega utiliza uma tentativa limitada com retrocesso exponencial (até `notification_max_attempts` tentativas, padrão 4, com um atraso base de `notification_backoff_ms`, padrão 500 ms, dobrando a cada tentativa). Uma nova tentativa é registrada como `NRF: Retrying status notification to ...`. Se cada tentativa falhar, o NRF desiste e registra `NRF: Giving up status notification to ... after N attempts`. Pesquise os logs do NRF por essas duas linhas para ver se a entrega foi tentada novamente e eventualmente abandonada.
- Se `subscrCond.nfType` estiver definido, apenas mudanças de status para aquele tipo de NF são entregues.

O registro está vazio após uma reinicialização do NRF

Esperado. Os armazéns de NF e assinatura estão na memória (ETS). Após uma reinicialização, o registro se repopula à medida que os NFs se re-registram em seu próximo ciclo de registro/batimento; não há persistência para recarregar.