← نظرة عامة

مرجع تكوين OmniNRF

تقرأ OmniNRF تكوين المشغل من `config/runtime.exs`، والذي يربط متغيرات البيئة ببيئة تطبيق `:omninrf`. يتم سرد كل مفتاح يواجه المشغل أدناه مع متغير البيئة الافتراضي الخاص به.

تكوين وقت التشغيل

```
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
```

جدول المعلمات

المعلمة	النوع	مطلوب	الافتراضي
`sbi_scheme`	سلسلة	لا	`http`
`sbi_addr`	سلسلة	لا	`127.0.0.1`
`sbi_port`	عدد صحيح	لا	`7777`
`mcc`	سلسلة	لا	`999`

المعلمة	النوع	مطلوب	الافتراضي
`mnc`	سلسلة	لا	`70`
`heartbeat_timeout`	عدد صحيح (مللي ثانية)	لا	`30000`
`cleanup_interval`	عدد صحيح (مللي ثانية)	لا	`10000`
`notification_max_attempts`	عدد صحيح	لا	`4`

المعلمة	النوع	مطلوب	الافتراضي
`notification_backoff_ms`	عدد صحيح (مللي ثانية)	لا	`500`
`discovery_validity_period`	عدد صحيح (ثواني)	لا	`3600`
`plmn_list`	قائمة	لا	`[%{mcc: mcc, mnc: mnc}]`
`prometheus_metrics_port`	عدد صحيح	لا	`9568`
`logger format`	مجموعة	لا	`{OmniLogger.JsonFormatt :format}`

المعلمة	النوع	مطلوب	الافتراضي
`logger metadata`	ذرة/قائمة	لا	`:all`

الافتراضي هو `127.0.0.1` (حلقة العودة). في أ�ً نشر متعدد المضيفين أو `SBI_ADDR` الأخرى، وإلا فلن تتمكن أي NFs حاويات، يجب تجاوز هذا إلى عنوان يمكن أن تصل إليه من التسجيل أو الاكتشاف.

دلالات عمر النبض

يتلقى NRF من NFs **نبضات** PATCH (أو أي) نبضة NF الأخرى؛ لا يرسلها. في كل مرة يرسل فيها NF طابع زمني "آخر رؤية" جديد ضد ملف تعريف ذلك NRF لمثيله، يسجل (تسجيل/تحديث).

يكون طابع الزمن NF مللي ثانية وتزيل أي `cleanup_interval` تعمل عملية تنظيف خلفية كل بهذه الطريقة، يتم NF مللي ثانية. عندما يتم إزالة `heartbeat_timeout` "آخر رؤية" له أقدم من إلى أي مشتركين مطابقين، `NF_DEREGISTERED` إلغاء تسجيله من المتجر ويتم إرسال إشعار حالة حتى يتوقف المستهلكون عن تلقي نقطة نهاية ميتة.

بملء واحدة تساوي NRF يقوم، `heartBeatTimer` خاصته NF عند التسجيل، إذا لم يتضمن ثواني كإشارة لمدى تكرار توقعه أن يتم نبضه. يجب على `heartbeat_timeout / 1000` قيمة أقصر بشكل) `heartbeat_timeout` تقوم بنبض جيد داخل NFs المشغلين التأكد من أن حي NF حتى لا يتسبب الاهتزاز العابر في انتهاء (مريح من المهلة.

مستقلان. (`heartbeat_timeout`) وحدود القدم (`cleanup_interval`) فاصل تنظيف الخلفية فقط في أول عملية تنظيف تجري بعد أن يتجاوز عمر آخر رؤية له NF تتم إزالة `heartbeat_timeout`، بعد أن يصبح صامتًا هو حتى NF لذا فإن أسوأ وقت حقيقي يبقى فيه `cleanup_interval` أقل بكثير `heartbeat_timeout + cleanup_interval`. احتفظ بـ `heartbeat_timeout` (القيم الافتراضية هي 10 ثواني و30 ثانية) حتى تبقى هذه الزيادة من `heartbeat_timeout` أكبر من `cleanup_interval` سيترك NFs ميتة معلن صغيرة؛ فإن عنها لفترة أطول من المقصود.

plmn_list

`plmn_list` هو مجموعة من PLMNs **التي يخدمها هذا NRF**، معبرًا عنها كقائمة من خرائط `%{mcc, mnc}`. تلك القائمة NF الموجود داخل ملف تعريف كل `plmnList` إنه متميز عن مسجل. الشبكة (أو الشبكات) التي `plmn_list` هي ما يتم تصفية الاكتشاف ضده، بينما يصف NF لكل يخدمها NRF.

الحقل ال��رعي	النوع	الوصف
`mcc`	سلسلة	رمز الدولة المحمول (3 أرقام)
`mnc`	سلسلة	رمز الشبكة المحمولة (2 إلى 3 أرقام)

العليا. `mcc`/`mnc` الافتراضي هو إدخال واحد مبني من مفاتيح `plmn_list` إذا لم يتم تعيينه، فإن مشترك / متعدد المشغلين، على سبيل المثال NRF قم بتكوين إدخالات متعددة لـ:

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

وليس NF، الخاصة بكل `plmnList` ضد (`plmn-id` معامل الاستعلام) يتم تقييم مطابقة الاكتشاف ينتمي `plmnList` المسجل الذي يعلن **لا** NF يُفترض أن، TS 29.510 وفقًا لـ. `plmn_list` ضد إلى PLMN الخاص بـ NRF وبالتالي يتطابق مع أي فلتر `plmn-id`.

OAM إدارة / واجهة برمجة التطبيقات

لإدارة المشغل لعمليات الفحص REST واجهة برمجة تطبيقات OmniNRF، تعرض 5Gبجانب SBI والتحكم في وقت التشغيل. يتم تقديمها عبر **HTTPS** على مستمع api_ex (المنفذ للإدارة `8443` الافتراضي). طلبات وأجسام الاستجابة هي JSON.

الطريقة	المسار	الوصف
GET	`/api/oam/config`	عرض تكوين وقت التشغيل الحالي (`sbi_addr`, `sbi_port`, `heartbeat_timeout`, `cleanup_interval`, `plmn_list`, `log_level`)
PATCH	`/api/oam/config/:id`	تحديث تكوين وقت التشغيل. يمكن تغيير فقط `heartbeat_timeout` و `log_level` يتم؛ `cleanup_interval` و التحقق من جميع القيم المقدمة قبل تطبيق أي منها (400 في حالة قيمة غير صالحة)
POST	`/api/oam/log_level`	تغيير مستوى السجل في وقت التشغيل. الجسم `{"level": "..."}` المستويات؛ الصالحة هي `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug` (400 خلاف ذلك)
POST	`/api/oam/heartbeat`	تحديث طابع زمن النبض يدويًا لـ NF. الجسم `{"nf_instance_id": "..."}` (404 إذا لم يكن المثيل مسجلاً)
GET	`/api/oam/nf_uptime/:id`	عرض عمر التسجيل/النبض وحالة مثيل NF واحد (404 إذا لم يكن مسجلاً)
POST	`/api/oam/nf_purge`	بدء عملية تنظيف فورية للمثيلات القديمة من NF التي تمت السجل؛ تقارير عن عدد مثيلات إزالتها
GET	`/api/oam/resources`	عرض استخدام موارد BEAM VM (الذاكرة، عدد العمليات/المنافذ، عدد المجدول، وقت التشغيل، إصدارات OTP/BEAM)
GET	`/api/oam/diagnostics`	عرض تشخيصات BEAM VM (العقدة، وقت التشغيل، العقد المتصلة، عدد العمليات، قائمة

الطريقة	المسار	الوصف
		التشغيل، إحصائيات الإدخال/الإخراج و GC)
PATCH	`/api/oam/nf_status/:id`	تحديث الحالة المخزنة لمثيل NF

يقدم نفس المستمع أيضًا نقاط نهاية للحالة والمخزون للقراءة فقط: `/api/status/api`، `/api/status/license`، `/api/status/nrf`، `/api/status/nf`، `/api/api/nf_instances`، `/api/api/subscriptions`، و `/api/api/statistics`.

فحص وإزالة مثيلات NF

`GET /api/oam/nf_uptime/:id` هو الطريقة المباشرة لرؤية مدى قرب NF من أن يتم إزالته. إنه يقدم:

الحقل	المعنى
`last_heartbeat_ms_ago`	عمر طابع الزمن "آخر رؤية" لـ NF، بالمللي ثانية
`heartbeat_timeout`	الحد الحالي للقدم، بالمللي ثانية
`nf_type`، `nf_status`	نوع NF وحالته المخزنة

يكون NF صامتًا قبل أن تنتهي عملية التنظيف. الفجوة بين الاثنين هي تقريبًا مدى بقاء `last_heartbeat_ms_ago` مرشحًا للإ��الة بمجرد أن يتجاوز `heartbeat_timeout`.

`POST /api/oam/nf_purge` يشغل نفس عملية التنظيف للمثيلات القديمة كما هو الحال مع مؤقت الخلفية، على الفور وعند الطلب، ويعيد عدد المثيلات التي تمت إزالتها. إنه يزيل فقط NFs الصحية على NFs لا يجبر ;(`heartbeat_timeout` آخر رؤية أقدم من) التي هي *بالفعل* قديمة الخروج. استخدمه لاستعادة السجل دفعة واحدة بدلاً من الانتظار لعملية التنظيف المجدولة التالية. لإزالة NF حي معين بشكل قسري بغض النظر عن عمر نبضه، قم بإلغاء تسجيله مباشرة باستخدام `DELETE /api/api/nf_instances/:id`.

التسجيل

تقوم OmniNRF بإصدار **سجلات** **JSON** **منظمة** عبر مُنسق `OmniLogger.JsonFormatter`، مع تضمين جميع بيانات التعريف الخاصة بالسجل (`metadata: :all`). يحمل كل سطر سجل بيانات تعريف لكل طلب مثل معرف مثيل NF والإجراء الذي يتم التعامل معه، مما يجعل من السهل ربط التسجيل، والنبض، والاكتشاف، وأحداث الإشعار لـ NF معين.

يمكن تغيير مستوى السجل النشط في وقت التشغيل دون إعادة تشغيل، إما من خلال `POST /api/oam/log_level` أو عبر مفتاح `log_level` من `PATCH /api/oam/config/:id`.

← نظرة عامة

مقاييس OmniNRF

تُعرض المقاييس على `http://{host}:{prometheus_metrics_port}/metrics` (المنفذ الافتراضي `9568`) بتنسيق نصي Prometheus.

مقاييس خدمة NRF

المقياس	النوع	الوصف	ك
`omni_nrf_nf_registered_count`	Gauge	العدد الحالي لحالات NF المسجلة، حسب النوع	`nf_t`
`omni_nrf_active_nf_instances_count`	Gauge	العدد الحالي لحالات NF النشطة، حسب النوع. يتم إصدارها في كل تسجيل NF، deregister، وانتهاء نبض القلب، لذا فهي تعكس المجموعة النشطة الحية	`nf_t`
`omni_nrf_registration_requests_count`	Counter	إجمالي طلبات تسجيل NF المستلمة (جميع الأنواع)	شيء)
`omni_nrf_nf_registrations_total`	Counter	إجمالي تسجيلات NF، حسب النوع	`nf_t`
`omni_nrf_nf_deregistrations_total`	Counter	إجمالي deregistrations NF، حسب النوع. يتم حل تسمية `nf_type` من ملف NF المخزن	`nf_t`
`omni_nrf_heartbeats_total`	Counter	إجمالي طلبات نبض القلب NF،	`nf_t`

المقياس	النوع	الوصف	ف
		حسب النوع. يتم حل تسمية من `nf_type` ملف NF المخزن	
`omni_nrf_discovery_requests_count`	Counter	إجمالي طلبات الاكتشاف المستلمة (جميع أنواع الأهداف)	شيء)
`omni_nrf_discovery_requests_total`	Counter	إجمالي طلبات الاكتشاف، حسب نوع NF المستهدف	`targ`

مقاييس BEAM VM

المقياس	النوع	الوصف
`beam_memory_total`	Gauge	إجمالي ذاكرة BEAM (بايت)
`beam_memory_processes`	Gauge	الذاكرة المخصصة لعمليات Erlang
`beam_memory_processes_used`	Gauge	الذاكرة المستخدمة فعليًا بواسطة العمليات
`beam_memory_system`	Gauge	ذاكرة النظام (غير العمليات)
`beam_memory_atom`	Gauge	إجمالي ذاكرة الذرات
`beam_memory_atom_used`	Gauge	ذاكرة الذرات المستخدمة
`beam_memory_binary`	Gauge	ذاكرة ثنائية
`beam_memory_code`	Gauge	ذاكرة التعليمات البرمجية
`beam_memory_ets`	Gauge	ذاكرة جدول ETS (تحتفظ بمتاجر NF والاشتراكات)
`beam_processes_count`	Gauge	عدد عمليات Erlang
`beam_ports_count`	Gauge	عدد منافذ Erlang
`beam_atom_count`	Gauge	عدد الذرات
`beam_vm_uptime`	Gauge	وقت تشغيل VM (ثواني)

استعلامات PromQL المفيدة

حالات NF المسجلة حسب النوع:

```
omni_nrf_nf_registered_count
```

معدل طلبات التسجيل على مدار 5 دقائق:

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

معدل طلبات الاكتشاف حسب نوع NF المستهدف:

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

معدل نبض القلب حسب نوع NF (انخفاض نحو الصفر يحذر من NFs التي توقفت عن إرسال نبض القلب). يتم حل تسمية `nf_type` من الملف المخزن، لذا فإن التفصيل حسب النوع له معنى:

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

تغير التسجيل الصافي (التسجيلات ناقص deregistrations)، حسب نوع NF. كلا العدادات تحمل `nf_type` حقيقي، لذا يمكن حساب التغير حسب النوع:

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

التسجيلات حسب نوع NF (هذا العداد يحمل `nf_type` حقيقي):

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

> عرض اسم المقياس: يتم تصدير أسماء القياس مثل `omni_nrf.nf.registered.count` مع استبدال النقاط بشرطات سفلية. تأكد من الأسماء المصدرة الدقيقة مقابل عملية سحب `/metrics` حية، حيث يمكن أن تختلف مع تكوين المصدر.

← نظرة عامة

دليل عمليات OmniNRF

تقوم OmniNRF بتنفيذ وظيفة مستودع الشبكة (NRF) 5 في النواة 5G. يعتبر NRF السجل المركزي لكل وظيفة شبكة (NF) أخرى ودليل الخدمات للهندسة المعمارية المعتمدة على الخدمة (SBA): عند بدء التشغيل، وتحافظ على تلك التسجيلات حية من خلال NRF تسجل ملفها الشخصي مع نبضات قلب دورية، وتستفسر عن NRF لاكتشاف نقاط النهاية الخاصة بـ NFs التي تحتاج إلى التحدث إليها.

تقدم OmniNRF خدمتين SBI:

- **Nnrf_NFManagement** (`nnrf-nfm/v1`): تسجيل ملف NF، نبض القلب، إلغاء التسجيل، الاسترجاع، والاشتراكات/الإشعارات لتغيير حالة NF.
- **Nnrf_NFDiscovery** (`nnrf-disc/v1`): حل مثيلات NF المستهدفة حسب النوع، الخدمة، S-NSSAI، DNN، و PLMN.

لا يوجد قاعدة بيانات خارجية؛ يتم تُحتفظ جميع ملفات NF والاشتراكات في مخازن الذاكرة (ETS). يقوم مسح خلفي بتقادم مثيلات NFs التي توقفت عن إرسال نبضات القلب، لذا فإن السجل يعلن فقط عن NF التي هي حية بشكل واضح.

ومراجع المواصفات GPPدور 3

المواصفة	الصلة
TS 23.501	في الهندسة المعمارية المعتمدة على الخدمة NRF بنية النظام، دور (البند 6.2.6)
TS 29.510	خدمات Nnrf_NFDiscovery، نماذج Nnrf_NFManagement و بيانات NFProfile / SearchResult / SubscriptionData
TS 29.510 §5.2	Nnrf_NFManagement: التسجيل، التحديث، نبض القلب (`heartBeatTimer`)، إلغاء التسجيل، الاشتراكات والإشعارات للحالة
TS 29.510 §5.3	Nnrf_NFDiscovery: استعلام اكتشاف مثيل NF و SearchResult
TS 29.510 §6.2.3.2.3.1	معلمات استعلام الاكتشاف، ترتيب النتائج والتقسيم

تعتبر OmniNRF **منتج** SBI لـ `Nnrf_NFManagement` و `Nnrf_NFDiscovery`. نها لا�� آخر؛ إنها السجل الذي تشير إليه كل الأشياء الأخرى، لذا لا يوجد NRF تسجل نفسها مع أي `nrf_uri` خارجي في تكوين مشغلها.

نقاط نهاية SBI

يتم تقديم مستمع SBI عبر **HTTP/2 (h2c، نص عادي)** على `http://{sbi_addr}:{sbi_port}`. تكون أجسام الطلب والاستجابة بتنسيق JSON. تستخدم أجسام PATCH `application/json-patch+json`؛ يتم إرجاع قائمة مثيلات NF كـ `application/3gppHal+json`.

Nnrf_NFManagement (`nnrf-nfm/v1`)

الطريقة	المسار	الوصف	النجاح
PUT	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	تسجيل أو إعادة تسجيل NF (جسم NFProfile كامل)	201 تم الإنشاء (جديد) / 200 موافق (تحديث)
PATCH	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	نبض القلب (جسم فارغ) أو تحديث ملف تعريف JSON-Patch	204 لا محتوى (نبض القلب) / 200 موافق (تصحيح)
DELETE	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	إلغاء تسجيل NF	204 لا محتوى
GET	`/nnrf-nfm/v1/nf-instances/{nfInstanceId}`	استرجاع ملف NFProfile واحد	200 موافق
GET	`/nnrf-nfm/v1/nf-instances`	قائم�� مثيلات NF (UriList؛ يدعم `nf-type`، `limit`، `page-number`،	200 موافق

الطريقة	المسار	الوصف	النجاح
		`page-size`)	
POST	`/nnrf-nfm/v1/subscriptions`	الاشتراك في إشعارات حالة NF	201 تم الإنشاء
GET	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	قراءة اشتراك واحد	200 موافق
PATCH	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	تحديث اشتراك (JSON-Patch)	200 موافق
DELETE	`/nnrf-nfm/v1/subscriptions/{subscriptionId}`	إلغاء الاشتراك	204 لا محتوى

تحتوي استجابات PUT/PATCH/GET الناجحة على مثيل NF واحد على رأس `ETag`. يمكن للعملاء تقديم `If-Match` على PUT/PATCH لجعل التحديث مشروطًا (412 فشل الشرط عند عدم التطابق).

Nnrf_NFDiscovery (`nnrf-disc/v1`)

الطريقة	المسار	الوصف	النجاح
GET	`/nnrf-disc/v1/nf-instances`	اكتشاف مثيلات NF حسب النوع والفلاتر الاختيارية	200 موافق

تحمل الاستجابة رأس `Link: <...>; rel="next"` يشير إلى الصفحة التالية عندما تكون نتيجة الاكتشاف مقسمة (وما زالت هناك نتائج أخرى `limit` تم تقديم).

معلمات استعلام الاكتشاف

المعلمة	مطلوب	الوصف
`target-nf-type`	نعم	نوع NF المطلوب البحث عنه (مثل `UDM`، `SMF`، `PCF`)
`requester-nf-type`	نعم	نوع NF الذي يقوم بالطلب (مثل `AMF`)
`service-names`	لا	إذا كان يقدم NF أسماء الخدمات مفصولة بفواصل؛ يتطابق أي خدمة مدرجة
`plmn-id`	لا	معرف PLMN بتنسيق JSON، مثل `{"mcc":"999","mnc":"70"}`. يتطابق NF الذي لا يحتوي على `plmnList` مع أي PLMN
`snssais`	لا	S-NSSAI بتنسيق JSON (كائن أو مصفوفة)، مثل `[{"sst":1,"sd":"000001"}]`. يتطابق مع `sNssais` و `allowedNssais` لكل خدمة
`dnn`	لا	اسم شبكة البيانات؛ يتطابق مع `allowedDnns` لكل خدمة
`limit`	لا	الحد الأقصى لعدد النتائج في هذه الصفحة
`offset`	لا	الفهرس الابتدائي في مجموعة النتائج (المرتبة)، المستخدمة للتقسيم

تُرجع النتائج في `SearchResult` (`validityPeriod`، `nfInstances`، `numNfInstComplete`). يتم إرجاع NFs فقط التي لديها `nfStatus = REGISTERED`.

ترتيب النتائج

تُرتب NFs المتطابقة بحيث يظهر المثيل الأكثر تفضيلًا أولاً، مما يسمح للمستهلك باختيار والعودة فقط إلى الإدخالات اللاحقة. تأتي مفاتيح الترتيب، بالترتيب، من كل `nfInstances[0]` ملف شخصي مسجل خاص بـ NF:

المفتاح	الاتجاه	المعنى لمشغل
`priority`	تصاعدي (الأدنى أولاً)	تفضيل نسبي محدد على ملف NF. الأدنى يفوز؛ يتم ترتيب NFs في النهاية (تعامل على `priority` التي لا تحتوي على أنها `65535`)
`capacity`	تنازلي (الأعلى أولاً)	ذات الأولوية المتساوية، يُفضل NFs بين NF. السعة المعلنة لـ الأعلى سعة NF
`load`	تصاعدي (الأدنى أولاً)	المتساوية NFs بين NF. الحمل الحالي المبلغ عنه من قبل الأقل تحميلًا NF بخلاف ذلك، يُفضل

على ملف (أعلى `capacity` أو) أقل `priority` لتوجيه المستهلكين نحو منتج معين، قم بتعيين NF المستهدف، وليست NF المسجل؛ هذه خصائص NRF.

قواعد تطابق الفلاتر

- **`service-names`**: يتطابق NF إذا كان يقدم *أي واح**ة من الخدمات المدرجة.
- **`snssais`**: يتطابق مع اتحاد `allowedNssais` على المستوى و NF، لـ `sNssais` المطابقة فقط عندما يقدم `sd` لكل خدمة. يتطلب `allowedNssais` العلوي، و الاستعلام واحدًا.
- **`plmn-id`** و **`dnn`**: يتطابق مع `plmnList` و NF لـ `allowedDnns` على لكل خدمة التوالي.

الذي يعلن *عدم وجود* تلك السمة على أنه يتطابق مع أي NF لكل فلتر اختياري، يتم التعامل مع تتبع هذه القاعدة (`plmn-id` مثل) قيمة لها `plmnList` يتطابق مع أي الذي لا يحتوي على NF المجهزة بأدنى حد NFs وتحافظ على إمكانية اكتشاف TS 29.510 ""غير المقيدة تعني عالمية.

التقسيم

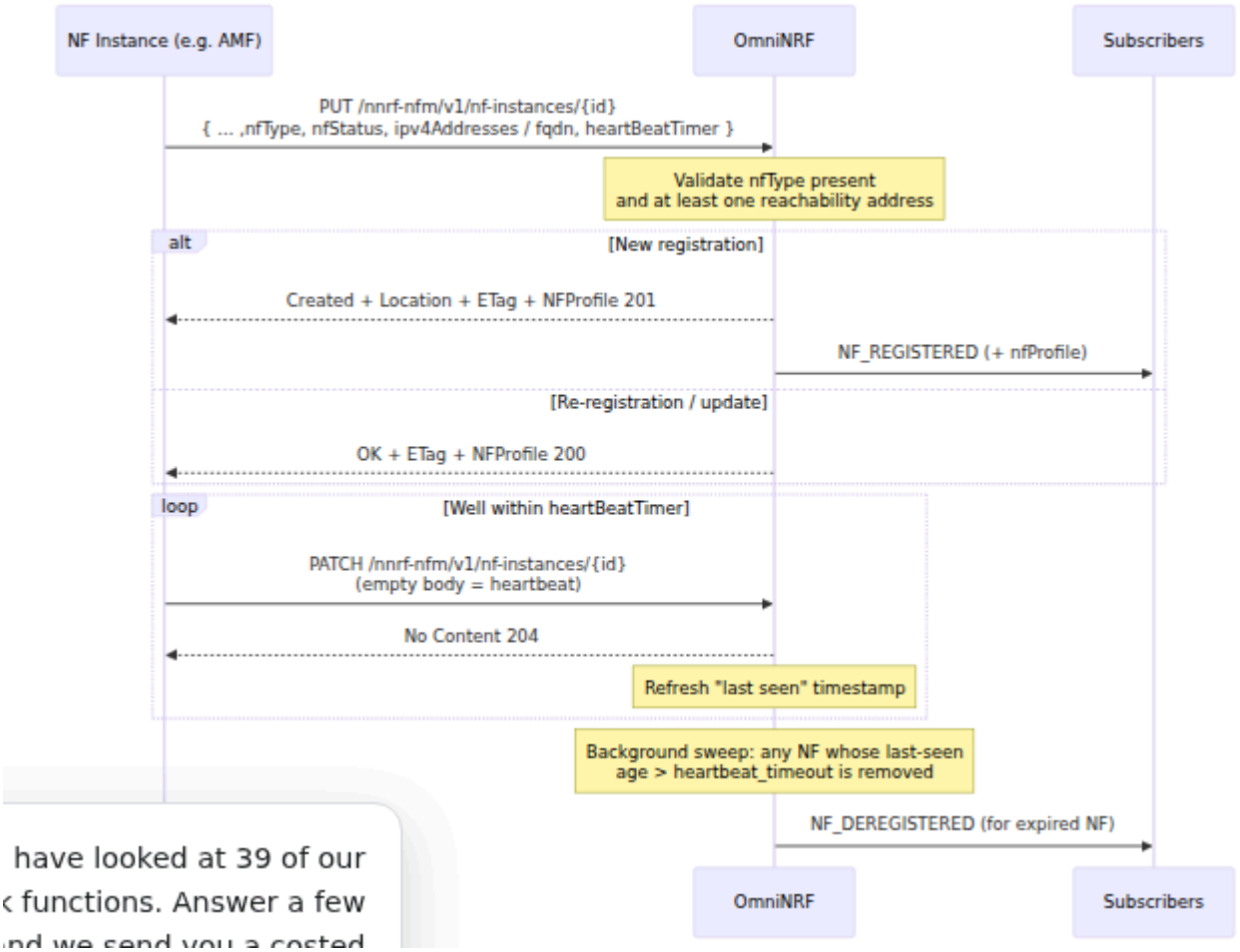
`limit` يحدد `offset` عدد المثيلات المعادة في مجموعة هو الفهرس الابتدائي في النتائج المرتبة. عندما تبقى نتائج أكثر بعد الصفحة، تحمل الاستجابة رأس `Link: <...>; rel="next"` الذي يشير إلى الصفحة التالية. تعكس `numNfInstComplete` دائمًا العدد الكامل للتطابق قبل التقسيم، لذا يمكن للمستهـ**ك تقسيم النتائج حتى يصل `offset + len(nfInstances)` إليها.

> يتم تقسيم اكتشاف SBI هذا (`offset`/`limit` + `Link`) بشكل منفصل عن تقسيم نقطة Nnrf_NFManagement، التي تستخدم بدلاً من ذلك *قائمة* مثيلات NF تحت نهاية

`limit`/`page-number`/`page-size` `3` على`gppHal+json` UriList.

الإجراءات الرئيسية

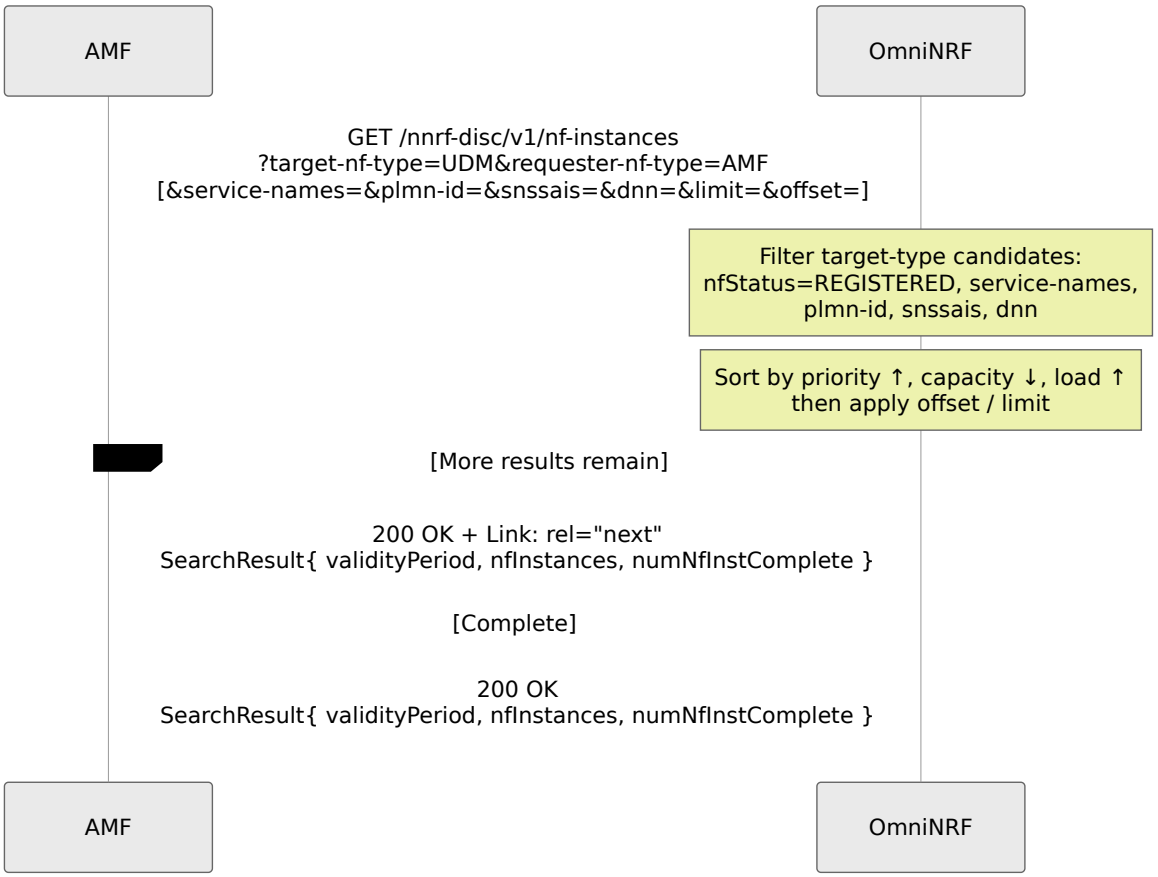
تسجيل NF ونبض القلب



يتطلب التسجيل `nfType` و `nfStatus` في الجسم (يتم رفض الجسم الذي يفتقر إلى أي منهما) **و** على الأقل عنوان وصول واحد (`fqdn`، `ipv4Addresses`، أو `ipv6Addresses`). يتم رفض الجسم الذي لا يحتوي على عنوان وصول مع `MANDATORY_IE_MISSING` مع 400 (طلب غير صحيح).

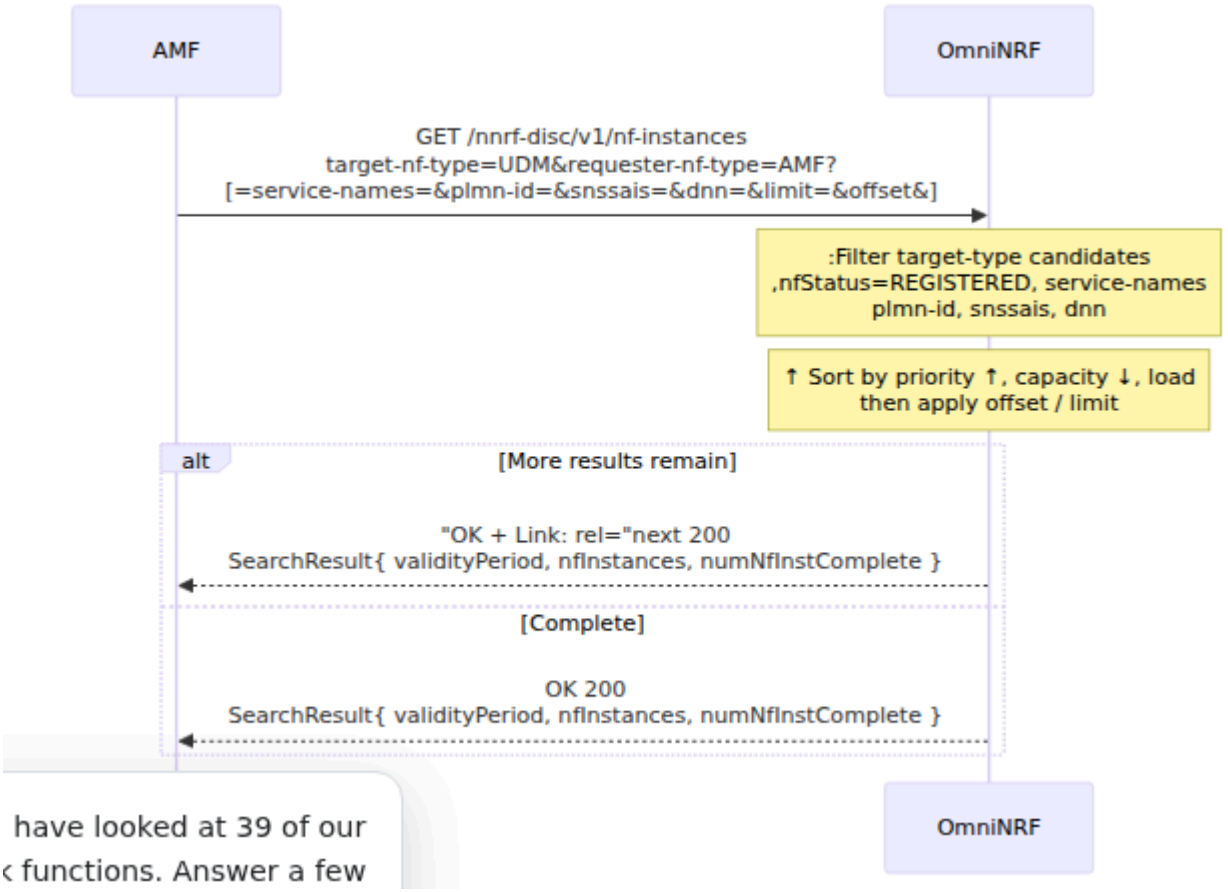
يتلقى NRF من NFs **نبضات القلب** الأخرى؛ فهو لا يرسلها. في كل مرة يرسل فيها NF طابع زمن "آخر ظهور" جديد ضد ملف NRF لمثيله، يسجل (أو أي تسجيل/تحديث) PATCH. يكون طابع الزمن "آخر ظهور" له أقدم من NF يعمل مسح خلفي بشكل مستمر ويزيل أي NF بهذه الطريقة، يتم إلغاء تسجيله من `heartbeat_timeout` مللي ثانية. عندما يتم تقادم `NF_DEREGISTERED` إلى أي مشتركين متطابقين، لذا يتوقف المخزن ويتم إرسال إشعار حالة المستهلكون عن تسليم نقطة نهاية ميتة.

اكتشاف NF



إلزاميان؛ يؤدي حذف أي منهما إلى 400. `requester-nf-type` و `target-nf-type` كلا من استجابة صحيحة تعني عدم (`numNfInstComplete: 0` مع) الفارغة `nfInstances` تعتبر قائمة تطابق أي شيء.

اشتراكات وإشعارات حالة NF



المشترك (المجمع) مع إعادة محاولة محدودة: يتم SBI HTTP/2 تُسلم الإشعارات عبر عميل مرات (الافتراضي 4)، باستخدام `notification_max_attempts` محاولة كل إشعار حتى الافتراضي 500 مللي ثانية، يتضاعف) `notification_backoff_ms` التراجع الأسّي مع قاعدة أو يكون غير متاح لفترة xxيتم إعادة محاولة المشترك الذي يجيب برمز غير 2 .(مع كل محاولة عن المحاولة NRF قصيرة بدلاً من إسقاطه بعد محاولة واحدة؛ بعد المحاولة الفاشلة الأخيرة، يتخلى يتم تعيين أنواع NRF. ويسجل تحذيرًا. يتم تسجيل المشترك غير المتاح ولكن لا يتسبب في تعطل الأحداث (من 5.2.2.8 البند TS 29.510) من حالة NF: `REGISTERED` إلى `NF_REGISTERED` (يتم تضمين `nfProfile` الكامل)، و `DEREGISTERED`/`SUSPENDED` إلى `NF_DEREGISTERED`، وتغييرات الحالة الأخرى إلى `NF_PROFILE_CHANGED`. الاشتراك مع `subscrCond.nfType` يتم (ISO 8601)`validityTime` إذا تم تقديم .NF يتلقى فقط الإشعارات لذلك النوع من حذف الاشتراك تلقائيًا عند انتهاء صلاحيته.

انتقالات حالة NF وكيف تصل إلى المشتركين

المحفز	الإشعار الناتج	هل تمت إزالته أيضًا من السجل؟
تسجيل NF (PUT، جديد)	`NF_REGISTERED` (مع `nfProfile`)	لا، تمت الإضافة
تغييرات ملف NF (PATCH / تحديث إلى حالة غير نهائية OAM)	`NF_PROFILE_CHANGED`	لا
إلغاء تسجيل NF (DELETE)	`NF_DEREGISTERED`	نعم
تم تقادمه (فقدان نبضات القلب، NF مسح خلفي)	`NF_DEREGISTERED`	نعم
تطهير يدوي (`POST /api/oam/nf_purge`) لـ NF قديم	`NF_DEREGISTERED`	نعم

المسجلة بشكل صريح لا يمكن تمييزهما للمشتركين: كلاهما يظهر كـ NF **المتقادمة** و NF متعمد DELETE وكلاهما يغادر السجل. الفرق هو السبب فقط: الإلغاء هو `NF_DEREGISTERED` الذي توقف عن نبض القلب. يتم الإبلاغ عن NRF NF عندما يزيل NF نفسه، بينما يتم تقادم NF من NF **المعلق** (`nfStatus = SUSPENDED` كـ `NF_DEREGISTERED`) لأغراض الإشعار، ولأ يتم إرجاعها من الاكتشاف، فهي مخفية فعليًا عن المستهلكين بينما يبقى `REGISTERED` فقط NFs ملفها الشخصي في المخزن.

← نظرة عامة

استكشاف أخطاء OmniNRF

اختفى NF من الاكتشاف على الرغم من أنه يعمل

خلال PATCH نبضة قلب NF تقريبًا دائمًا ما تكون هناك نبضات قلب مفقودة. إذا لم يرسل `heartbeat_timeout` (30 ثانية بشكل افتراضي)، فإن المسح الخلفي ينتهي به ويختفي من كل للبحث عن NRF من الاسترجاع والاكتشاف. تحقق من سجلات `NRF: Expiring stale NF instance {id}`، وتأكد من أن NF رفع يرسل نبضات قلب بشكل جيد داخل المهلة. `heartBeatTimer` أيضًا احترام NF يمنح مزيدًا من المرونة؛ يجب على `HEARTBEAT_TIMEOUT` المعاد في استجابة التسجيل الخاصة به.

تم رفض التسجيل مع 400

تفتقر NFProfile كل من NRF إلى عنصر إلزامي أو تحتوي على عنصر مشوه. يتطلب `nfType` وعنوان وصول واحد على الأقل (`fqdn`، `ipv4Addresses`، أو `ipv6Addresses`). 400 تعني مع `MANDATORY_IE_MISSING` أن `nfType`/`nfStatus` كانت غائبة عن جسم PUT؛ بينما تعني التي تذكر التفاصيل الوصول أنه لم يتم تزويد أي عنوان 400.

أيضًا تحققًا هيكليًا عميقًا لبقية الملف الشخصي ويعيد NRF بجانب تلك العناصر الإلزامية، يفرض `MANDATORY_IE_INCORRECT` مع السبب 400 (عندما يكون عنصر موجود مشوهًا وفقًا لـ TS 29.510). الأسباب الشائعة:

- `nfServices`: تحتاج كل خدمة إلى `serviceInstanceId`، و `serviceName`، و `versions` غير فارغة حيث `nfServiceStatus` صالح، ومصفوفة `scheme` صالح، و `apiVersionInUri` يحمل كل إدخال.
- `sNssais`: يحتاج كل إدخال إلى عدد صحيح `sst` في النطاق من 0 إلى 255، و `sd` يتكون من 6 أرقام سداسية عشرية بالضبط (عند وجوده).
- `plmnList`: يجب أن تكون الإدخالات كائنات PLMN جيدة التكوين.

- أعداد صحيحة في النطاق من 0 إلى 65535؛ يجب أن تكون `priority` و `capacity` عددًا صحيحًا في النطاق من 0 إلى 100 يجب أن تكون `load`.

تُقبل العناصر الاخ��يارية الغائبة؛ فقط العنصر الموجود ولكن المشوه يتم رفضه. تُسمى 400 الحقل المخالف، لذا تشير مباشرة إلى ما يجب تصحيحه `detail`.

تم رفض PATCH مع 415 نوع وسائط غير مدعوم

تم تعيين نوع محتوى PATCH إلى شيء غير `application/json-patch+json`. إما إرسال هذا النوع من JSON-Patch، أو إرسال نبضة قلب فارغة تمامًا (لا يُسمح بنوع محتوى للنبضات).

404 تحديث/نبضة قلب/حذف يعيد

قد تم استبعاده بالفعل بسبب نبضات NF غير موجود في السجل، عادةً لأن `nfInstanceId` القلب المفقودة، أو أنه لم يكمل التسجيل أبدًا. أعد التسجيل باستخدام PUT قبل التعديل أو إرسال نبضة قلب.

412 فشل شرط مسبق يعيد PUT/PATCH

تم إرسال رأس `If-Match` الذي لا يتطابق مع ETag المخزنة حاليًا (تغير الملف الشخصي تحت العميل). أعد `GET` للحالة للحصول على `ETag` الحالي، ثم حاول مرة أخرى.

فارغة `nfInstances` يعيد الاكتشاف قائمة

- تأكد من أن NF المستهدف مسجل وأن `nfStatus` المخزنة هي `REGISTERED` (فقط NFs المسجلة تُعاد).
- يتم مطابقة `target-nf-type` بحساسية الحالة ضد `nfType` المخزنة.
- تحقق من أنها جيدة التكوين وأن، `dnn` أو، `snssais` إذا تم تزويد فلاتر `plmn-id`، الذي لا يعلن عن أي من السمة المعطاة (NF المستهدف يعلن عنها فعليًا NF ملف يتطابق مع أي قيمة لها).
- تأكد من وجود `requester-nf-type`. غيابه يؤدي إلى 400، وليس قائمة فارغة.

لا يتم تلقي إشعارات الحالة

- تأكد من أن `nfStatusNotificationUri` في الاشتراك يمكن الوصول إليه من مضيف NRF.
- تستخدم التسليم محاولة محدودة مع تراجع أسي (حتى محاولات، افتراضي 4، مع تأخير أساسي قدره `notification_max_attempts`، يتضاعف مع كل محاولة، `notification_backoff_ms` افتراضي 500 مللي ثانية). يتم تسجيل إعادة المحاولة كـ `NRF: Retrying status notification to ...`. إذا فشلت كل محاولة، يتخلى NRF عن الأمر ويسجل `NRF: Giving up status notification to ... after N attempts`. ابحث في NRF عن ��جلات هذين السطرين لمعرفة ما إذا تم إعادة محاولة التسليم وأخيرًا التخلي عنه.
- يتم تسليم تغييرات الحالة فقط لذلك النوع من NF، إذا تم تعيين `subscrCond.nfType`.

NRF السجل فارغ بعد إعادة تشغيل

بعد إعادة التشغيل، يتم إعادة ملء السجل (ETS). ومتاجر الاشتراك في الذاكرة NF متوقع. تخزن في دورة التسجيل/نبضة القلب التالية؛ لا يوجد استمرارية لإعادة التحميل NFs مع إعادة تسجيل.