← Vue d'ensemble

Référence de configuration OmniUDR

OmniUDR lit sa configuration d'opérateur à partir de la clé d'environnement de l'application `:omniudr`, remplie au démarrage à partir de `config/runtime.exs`. Chaque clé d'exécution a une substitution par variable d'environnement et une valeur par défaut ; le bloc ci-dessous est la surface complète de l'opérateur.

Configuration d'exécution

```
import Config

config :omniudr,
  # API HSS backend
  hss_api_base_url: System.get_env("HSS_API_BASE_URL",
"https://127.0.0.1:8443"),

  # Configuration SBI
  sbi_scheme: System.get_env("SBI_SCHEME", "http"),
  sbi_addr: System.get_env("SBI_ADDR", "127.0.0.22"),
  sbi_port: String.to_integer(System.get_env("SBI_PORT", "7777")),

  # Configuration NRF
  nrf_uri: System.get_env("NRF_URI", "http://127.0.0.1:7777"),

  # Identité PLMN
  mcc: System.get_env("MCC", "999"),
  mnc: System.get_env("MNC", "70"),

  # Métriques Prometheus
  prometheus_metrics_port:
String.to_integer(System.get_env("PROMETHEUS_PORT", "9568")),

  # Intervalle de heartbeat (ms)
  heartbeat_interval:
String.to_integer(System.get_env("HEARTBEAT_INTERVAL", "10000")),

  # Abonnés de test - aide à la mise en service du laboratoire.
Carte JSON de l'IMSI → {ki, opc, amf}.
  test_subscribers: System.get_env("TEST_SUBSCRIBERS")

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

Tableau des paramètres

Paramètre	Type	Requis	Par défaut
`hss_api_base_url`	chaîne (URL)	Oui	`https://127.0.0.1:8443`
`sbi_scheme`	chaîne	Non	`http`
`sbi_addr`	chaîne (IP)	Non	`127.0.0.22`

Paramètre	Type	Requis	Par défaut
`sbi_port`	entier	Non	`7777`
`nrf_uri`	chaîne (URL)	Non	`http://127.0.0.1:7777`
`mcc`	chaîne	Non	`999`
`mnc`	chaîne	Non	`70`
`prometheus_metrics_port`	entier	Non	`9568`
`heartbeat_interval`	entier (ms)	Non	`10000`
`test_subscribers`	chaîne (JSON) | nil	Non	`nil`

Paramètre	Type	Requis	Par défaut
`default_am_nssai`	carte	Non	eMBB / URLLC / mIoT tranches de laboratoire (voir ci-dessous)

Paramètre	Type	Requis	Par défaut
`dnn_snssai_map`	carte	Non	`%{"iot" => %{"sst" => 3, "sd" => "000003"}}`

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

> Les clés `default_am_nssai` et `dnn_snssai_map` existent pour faire le pont entre le modèle de données HSS 4G (sans concept de tranche) et les structures Nudr 5G. Alignez-les avec le plan de tranche réel du déploiement et avec le NSSAI servi par l'AMF/SMF. Les valeurs par défaut AM/SM (eMBB `sst 1`, URLLC `sst 2 / sd 000002`, mIoT `sst 3 / sd 000003`) sont des valeurs de laboratoire.

Formes de la carte de synthèse de tranche (`default_am_nssai`, `dnn_snssai_map`)

Les deux clés sont des cartes structurées d'objets S-NSSAI. Un **S-NSSAI** a deux sous-champs :

Sous-champ	Type	Requis	Signification
`sst`	entier (0–255)	Oui	Type de tranche/service. Valeurs standardisées : `1` = eMBB, `2` = URLLC, `3` = mIoT (selon TS 23.501).
`sd`	chaîne (6 chiffres hexadécimaux)	Non	Différenciateur de tranche. Distingue deux tranches ayant le même `sst`. Omettre pour une tranche bare-`sst` ; lorsqu'il est présent, il doit correspondre exactement à la valeur servie par l'AMF/SMF.

`default_am_nssai` est le NSSAI d'accès et de mobilité, avec deux listes :

Liste	Signification
`defaultSingleNssais`	Les S-NSSAIs que l'UE obtient lorsqu'elle s'enregistre sans demander une tranche spécifique - l'ensemble de tranches par défaut du réseau pour l'abonné.
`singleNssais`	L'ensemble complet des S-NSSAIs que l'abonné est autorisé à utiliser (un sur-ensemble de `defaultSingleNssais`). Une UE demandant un S-NSSAI qualifié SD non listé ici est rejetée par l'AMF avec la cause 5GMM #62.

```
config :omniudr, :default_am_nssai, %{
  "defaultSingleNssais" => [%{"sst" => 1}, %{"sst" => 1, "sd" =>
"000001"}],
  "singleNssais" => [
    %{"sst" => 1},
    %{"sst" => 1, "sd" => "000001"},
    %{"sst" => 2, "sd" => "000002"},
    %{"sst" => 3, "sd" => "000003"}
  ]
}
```

`dnn_snssai_map` mappe une **chaîne DNN** au S-NSSAI unique sur lequel vivent ses sessions PDU. Tout DNN non présent dans la carte revient à eMBB (`%{"sst" => 1, "sd" => "000001"}`). Gardez chaque S-NSSAI référencé ici à l'intérieur de `default_am_nssai.singleNssais`, sinon le SMF placera la session sur une tranche à laquelle l'abonné n'est pas abonné.

```
config :omniudr, :dnn_snssai_map, %{
  "iot" => %{"sst" => 3, "sd" => "000003"},
  "internet" => %{"sst" => 1, "sd" => "000001"}
}
```

À propos de `hss_api_base_url` (le backend REST HSS)

C'est le paramètre d'opérateur le plus important. OmniUDR est une façade - il n'a pas de base de données. Lorsqu'un consommateur demande à OmniUDR l'abonnement d'authentification d'un abonné, les données AM, les données SM ou les données de politique, OmniUDR émet des requêtes REST en direct à `hss_api_base_url` (principalement `GET /api/subscriber/imsi/{imsi}` et `GET /api/key_set/{id}`, et pour les vecteurs d'authentification / SQN / localisation les points de terminaison `/api/hlr/*`), puis mappe la réponse HSS dans la structure 3GPP. Chaque requête HSS a un délai d'attente de 5 secondes (3 secondes pour la vérification de l'état). Si ce backend est inaccessible, les lectures Nudr correspondantes retournent `404`/`5xx` - voir Dépannage.

À propos de `test_subscribers`

`test_subscribers` remplit un ensemble d'abonnés statiques en code (deux IMSIs intégrés - vecteurs de test Milenage bien connus - plus tout ce qui est fourni via la carte JSON) qui retourne des données d'authentification, AM et SM fixes sans aucun appel HSS. C'est une **aide de laboratoire** pour la mise en service et les auto-tests de conformité où la mise en place d'un HSS complet n'est pas justifiée. Dans la version actuelle, le chemin de données Nudr_DR en direct lit toujours à partir du backend HSS ; l'ensemble d'abonnés de test n'est pas câblé en tant que solution de secours automatique lorsque le HSS est en panne. Traitez cette clé comme une surface de laboratoire/test uniquement et laissez-la non définie (`nil`) dans tout déploiement soutenu par un véritable HSS. Une valeur JSON invalide revient aux deux valeurs par défaut intégrées.

Identité PLMN et profil NRF

`mcc` et `mnc` sont annoncés dans le profil NF NRF comme le PLMN de service. La capacité et la priorité NF annoncées au NRF sont gérées par la configuration NRF Omni5G partagée et peuvent être modifiées à l'exécution via l'API OAM (`nf_capacity`, `nf_priority`).

Livraison de notification

`notification_max_attempts` et `notification_backoff_ms` régissent comment OmniUDR livre les notifications DataChangeNotify Nudr_DR aux abonnés de changement de données (le service `subs-to-notify`, abonnements TS 29.504). Lorsqu'une ressource surveillée change, OmniUDR publie une notification à l'URI de rappel de chaque abonné correspondant. Si une livraison échoue (le rappel est inaccessible ou retourne un statut non-2xx), OmniUDR réessaie jusqu'à `notification_max_attempts` fois, attendant `notification_backoff_ms` pour le premier essai et doublant le délai à chaque nouvelle tentative. Une fois les tentatives épuisées, la notification est abandonnée et l'événement est enregistré. La livraison est faite au mieux et n'affecte jamais la réponse SBI retournée au consommateur qui a causé le changement. Les deux clés sont uniquement configurables par l'application (pas de substitution par variable d'environnement) : configurez-les sous `config :omniudr, ...`.

API de gestion et OAM

Au-delà de l'écouteur SBI, OmniUDR expose une API de gestion HTTPS orientée opérateur sur le port **8443** (TLS, `enable_tls: true`). Une interface utilisateur de panneau de contrôle web est également servie sur HTTPS sur le port **7443**.

Surface	Port	Objectif
SBI (Nudr_DR)	`sbi_port` (par défaut `7777`)	Interface de service 3GPP
Métriques Prometheus	`prometheus_metrics_port` (par défaut `9568`)	Point de terminaison de collecte de métriques
API de gestion / OAM	`8443` (HTTPS, monté sous `/api`)	État, santé HSS, proxy d'abonné, actions OAM
Interface utilisateur du panneau de contrôle	`7443` (HTTPS)	Ressources d'application, configuration, licence et pages de log

L'API de gestion est montée sous le préfixe de chemin `/api`. Elle fournit un état en lecture seule, une vérification de connectivité HSS, un proxy d'abonné en lecture seule vers le HSS, et des actions OAM :

Méthode	Chemin	Objectif
GET	`/api/status/api`	État de l'API/service
GET	`/api/status/nrf`	État d'enregistrement NRF
GET	`/api/status/nf`	Profil / état NF
GET	`/api/status/license`	État de la licence
GET	`/api/health/hss`	Vérification de connectivité HSS (appelle `GET /api/status/api` sur le HSS)
GET	`/api/subscribers`	Liste d'abonnés paginée en lecture seule proxifiée depuis le HSS (`page`, `page_size`)
GET	`/api/subscribers/{imsi}`	Détails complets de l'abonné proxifiés depuis le HSS
GET	`/api/oam/config`	Voir la configuration d'exécution en direct
PATCH	`/api/oam/config/{id}`	Mettre à jour `hss_api_base_url`, `nrf_uri`, `nf_capacity`, `nf_priority` ou `log_level` à l'exécution
POST	`/api/oam/nrf/reregister`	Forcer la réinscription NRF
DELETE	`/api/oam/subscriber-cache/{id}`	Vider le contexte AMF/SMF mis en cache pour un abonné (`id=all` vide chaque abonné)
POST	`/api/oam/log_level`	Changer le niveau de log à l'exécution

Notes sur la mise à jour de la configuration OAM (`PATCH /api/oam/config/{id}`) :

- Seuls `hss_api_base_url`, `nrf_uri`, `nf_capacity`, `nf_priority` et `log_level` sont acceptés ; toute autre clé est ignorée, et une requête sans clé valide retourne 400.
- `nf_capacity` et `nf_priority` doivent être des entiers ; `log_level` doit être l'un des `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug`.
- Changer `nf_capacity` / `nf_priority` persiste à la fois la valeur et pousse un profil mis à jour vers le NRF.
- Le proxy d'abonné est en lecture seule. Si le HSS est inaccessible, `/api/subscribers` et `/api/subscribers/{imsi}` retournent `502` ; un abonné désactivé ou absent retourne `404`.

Journalisation

OmniUDR émet des journaux JSON structurés via `OmniLogger.JsonFormatter` avec des métadonnées complètes. Le niveau de journalisation à l'exécution peut être changé sans redémarrage via l'API OAM (`POST /api/oam/log_level` ou `PATCH /api/oam/config/{id}` avec `log_level`). Les actions OAM (mises à jour de configuration, vidages de cache, réinscriptions) sont enregistrées au niveau `info` avec un préfixe `[OAM]` ; les échecs de requêtes HSS sont enregistrés au niveau `warning`/`error`.

← Vue d'ensemble

Métriques OmniUDR

OmniUDR expose des métriques Prometheus sur le point de terminaison des métriques (`prometheus_metrics_port`, par défaut `9568`). Les noms des métriques ci-dessous utilisent la forme d'exposition Prometheus (les points deviennent des underscores). Les étiquettes et leurs valeurs sont en minuscules.

Métriques UDR / HSS

Métrique	Type	Étiquettes	Desc
`omni_udr_nrf_registration_status`	gauge	`nf_type`	État d'enregis NRF (1 = enregistr non)
`omni_udr_hss_health`	gauge	-	État de connectiv d'OmniHS opération hors serv à jour à c vérificati `/health/`
`omni_udr_hss_requests_total`	counter	`endpoint`, `result`	Nombre c REST au HSS, étiq chemin d (`endpoin` `result` (`success`
`omni_udr_hss_request_duration_ms`	distribution	`endpoint`	Latence c requêtes en millise Seaux : 5 50, 100, 1000, 25

Métriques BEAM VM

Métrique	Type	Description
`beam_memory_total`	gauge	Mémoire totale BEAM en octets
`beam_memory_processes`	gauge	Mémoire utilisée par les processus Erlang
`beam_memory_system`	gauge	Mémoire système (ETS, atomes, code, binaires)
`beam_processes_count`	gauge	Nombre de processus Erlang
`beam_vm_uptime`	gauge	Temps de fonctionnement de la VM en secondes

PromQL utiles

- Disponibilité du backend HSS : `omni_udr_hss_health`
- UDR enregistré avec NRF : `omni_udr_nrf_registration_status`
- Taux d'erreur HSS (5 min) : `sum(rate(omni_udr_hss_requests_total{result="failure"}[5m])) / sum(rate(omni_udr_hss_requests_total[5m]))`
- Latence p95 HSS (5 min) : `histogram_quantile(0.95, sum(rate(omni_udr_hss_request_duration_ms_bucket[5m])) by (le, endpoint))`
- Taux de requêtes HSS par point de terminaison : `sum(rate(omni_udr_hss_requests_total[1m])) by (endpoint)`
- Nombre de processus BEAM : `beam_processes_count`

← Vue d'ensemble

Guide des opérations OmniUDR

OmniUDR est le Référentiel de Données Unifié (UDR) du cœur Omnitouch 5G. Il implémente le service Nudr_DataRepository (TS 29.504) et expose des données structurées sur les abonnés, les politiques et le contexte au UDM et au PCF via l'interface basée sur les services (SBI).

OmniUDR ne possède pas de données sur les abonnés. C'est une façade Nudr_DR conforme aux normes sur le **backend de provisioning REST OmniHSS** : chaque lecture du référentiel de données est traduite au moment de la demande en un ou plusieurs appels REST OmniHSS, et la réponse HSS est mappée dans le modèle de données Nudr 3GPP avant d'être renvoyée au consommateur. OmniUDR est la seule fonction de réseau 5G Core qui communique directement avec l'API REST OmniHSS - le UDM (et, à travers lui, l'AUSF) accède aux données des abonnés en appelant OmniUDR via le Nudr standard, et non en appelant le HSS.

Étant donné que le HSS est le système de référence, OmniUDR ne conserve presque aucun état persistant. Son seul état local est constitué de trois petits magasins en mémoire (ETS) : un cache de contexte d'enregistrement AMF et SMF (utilisé pour décider si une écriture de contexte est une création ou une mise à jour), un magasin d'abonnement aux changements de données contenant les abonnements actifs Nudr_DR `subs-to-notify`, et un magasin de données d'application contenant les familles de données `application-data` Nudr_DR (PFDs, influence sur le trafic, BDT). Aucun de ces éléments n'est persistant ; tous sont effacés au redémarrage. Le cache de contexte d'enregistrement peut être vidé via l'API de gestion.

Rôle 3GPP et références de spécifications

Spécification	Pertinence
TS 23.501	Architecture du système 5G - rôle de l'UDR et sa relation avec l'UDM, le PCF et la couche de données UDM
TS 29.504	Service Nudr_DataRepository - cadre des ressources, données d'abonnement, données de contexte, données de politique
TS 29.505	Structures de données d'abonnement - abonnement à l'authentification, accès et mobilité, gestion de session, sélection SMF
TS 29.519	Structures de données de politique - données de politique AM et formes de ressources de données de politique SM
TS 29.503	Services Nudm - le consommateur UDM dont les procédures UEAU/SDM/UECM pilotent les lectures UDR et les écritures de contexte
TS 29.571	Types de données communs - `PatchResult`, `ProblemDetails`, encodages PLMN et S-NSSAI

Interfaces

Interface	Liaison	Objectif
SBI (Nudr_DR)	`{sbi_scheme}://{sbi_addr}:{sbi_port}` (par défaut `http://127.0.0.22:7777`)	Sert les ressources `nudr-dr/v2` consommées par l'UDM et le PCF
Backend REST HSS	`hss_api_base_url` (par défaut `https://127.0.0.1:8443`)	Appels REST sortants vers OmniHSS qui remplissent chaque lecture/mise à jour
NRF	`nrf_uri`	Enregistrement NF et heartbeat périodique
API de gestion / OAM	`:8443` (HTTPS, monté sous `/api`)	Statut, santé du HSS, proxy d'abonnés et actions OAM - voir Référence de configuration
Métriques Prometheus	`prometheus_metrics_port` (par défaut `9568`)	Point de terminaison de collecte de métriques - voir Métriques

---

Ensembles de données servis

OmniUDR sert les ensembles de données Nudr_DR suivants. Chacun est dérivé au moment de la lecture du dossier d'abonné OmniHSS ; la colonne "source HSS" nomme les données REST sous-jacentes à partir desquelles la valeur est mappée.

Ensemble de données	Famille de ressources Nudr	Accès	Consc
Abonnement à l'authentification	`subscription-data/.../authentication-data/authentication-subscription`	Lecture / mise à jour	UDM (Nudm
Vecteurs d'authentification	`subscription-data/.../authentication-data/authentication-vectors`	Générer (propriétaire)	OmniU
Données d'accès et de mobilité (AM)	`subscription-data/.../provisioned-data/am-data`	Lecture	UDM (Nudm
Données de gestion de session (SM)	`subscription-data/.../provisioned-data/sm-data`	Lecture	UDM (Nudm
Données de sélection SMF	`subscription-data/.../provisioned-data/smf-selection-subscription-data`	Lecture	UDM (Nudm
Contexte d'enregistrement AMF	`subscription-data/.../context-data/amf-3gpp-access`	Écriture	UDM (Nudm
Contexte d'enregistrement SMF	`subscription-data/.../context-data/smf-registrations/{pduSessionId}`	Écriture	UDM (Nudm

Ensemble de données	Famille de ressources Nudr	Accès	Consc
Données de politique AM	`policy-data/ues/{ueId}/am-data`	Lecture	PCF
Données de politique SM	`policy-data/ues/{ueId}/sm-data`	Lecture	PCF

| Données d'application | `application-data/{family}/{id}` (familles `pfds`, `influenceData`, `bdtData`) | Créer / lire / remplacer / supprimer | PCF, NEF | Aucune - conservée dans un magasin en mémoire (ETS) local, non soutenue par le HSS et non persistante |

> Les données d'application sont servies par un magasin générique en mémoire plutôt que par le HSS. Elles ne sont pas persistées : un redémarrage les efface. Voir Points de terminaison SBI → Données d'application ci-dessous.

Points de terminaison SBI

Tous les points de terminaison Nudr_DR sont servis sur HTTP à `{sbi_scheme}://{sbi_addr}:{sbi_port}` (par défaut `http://127.0.0.22:7777`) avec `Content-Type: application/json`. Les échecs sont renvoyés sous forme de `ProblemDetails` TS 29.571.

Le segment de chemin `{ueId}` est un SUPI au format `imsi-<chiffres>` (un IMSI nu est également accepté). Le segment `{servingPlmnId}` sur les lectures de données provisionnées est validé : il doit s'agir d'un PLMN bien formé (5 ou 6 chiffres) et doit correspondre au PLMN servant configuré à partir de `mcc`/`mnc`, sinon la lecture est rejetée avec `400 MANDATORY_IE_INCORRECT`. Cette version sert un seul enregistrement par abonné.

Données d'abonnement - Authentification (TS 29.504 / 29.505)

Méthode	Chemin	Description	Succès
GET	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-subscription`	Récupérer l'abonnement à l'authentification (méthode, Ki, OPc, AMF, SQN)	200
PATCH	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-subscription`	Mettre à jour l'abonnement à l'authentification (SQN)	204, ou 20 + `PatchResu` en cas d'échec partiel
POST	`/nudr-dr/v2/subscription-data/{ueId}/authentication-data/authentication-vectors`	Générer des vecteurs d'authentification (voir note)	200

> **Point de terminaison non standard.** Le POST `authentication-vectors` est une extension propriétaire d'Omni 5GC ; TS 29.504/29.505 ne définissent pas de telle ressource. La génération de vecteurs et la gestion de SQN/resync sont déléguées au UDR afin qu'elles soient gérées de manière centralisée au HSS. Elle est consommée uniquement par OmniUDM ; un consommateur tiers conforme aux normes ne l'appellera jamais.

Données d'abonnement - Provisionnées (TS 29.505)

Méthode	Chemin	Description
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/am-data`	Récupérer les données d'abonnement d'accès et de mobilité
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/sm-data`	Récupérer les données d'abonnement de gestion de session
GET	`/nudr-dr/v2/subscription-data/{ueId}/{servingPlmnId}/provisioned-data/smf-selection-subscription-data`	Récupérer les données d'abonnement de sélection SMF

Données d'abonnement - Contexte (TS 29.504)

Méthode	Chemin	Description	Succès
PUT	`/nudr-dr/v2/subscription-data/{ueId}/context-data/amf-3gpp-access`	Stocker le contexte d'enregistrement d'accès 3GPP AMF	201 + `Location` (nouveau) ou 204 (mise à jour)
PUT	`/nudr-dr/v2/subscription-data/{ueId}/context-data/smf-registrations/{pduSessionId}`	Stocker le contexte d'enregistrement SMF	201 + `Location` (nouveau) ou 204 (mise à jour)

Un corps vide sur l'un ou l'autre des PUT de contexte renvoie 400 `ProblemDetails` (`cause: MANDATORY_IE_MISSING`).

Données de politique (TS 29.519)

Méthode	Chemin	Description	Succès
GET	`/nudr-dr/v2/policy-data/ues/{ueId}/am-data`	Récupérer les données de politique AM (`subscCats`, `praInfos`)	200
GET	`/nudr-dr/v2/policy-data/ues/{ueId}/sm-data`	Récupérer les données de politique SM (`smPolicySnssaiData` par DNN)	200

Données d'application (TS 29.504)

CRUD générique sur les familles `application-data` Nudr_DR. Le segment de chemin `{family}` est l'un de `pfds`, `influenceData` ou `bdtData` ; le segment `{id}` est l'identifiant de ressource de la famille (`appId` / `influenceId` / `bdtReferenceId`). Les enregistrements sont conservés dans un magasin en mémoire (ETS) local, pas dans le HSS, et ne sont pas persistés lors d'un redémarrage. Toute autre valeur `{family}` tombe dans le 404 catch-all.

Méthode	Chemin	Description	Succès
GET	`/nudr-dr/v2/application-data/{family}`	Lister chaque ressource dans la famille (collection GET)	200 (tableau)
GET	`/nudr-dr/v2/application-data/{family}/{id}`	Récupérer une ressource	200, ou 404 si inconnue
PUT	`/nudr-dr/v2/application-data/{family}/{id}`	Créer ou remplacer une ressource	201 + `Location` (nouveau) ou 200 (remplacer)
DELETE	`/nudr-dr/v2/application-data/{family}/{id}`	Supprimer une ressource	204, ou 404 si inconnue

Un corps vide sur le PUT renvoie 400 `ProblemDetails` (`cause: MANDATORY_IE_MISSING`).

Abonnements aux changements de données (TS 29.504 subscriptions-to-notify)

Méthode	Chemin	Description	Succès
POST	`/nudr-dr/v2/subscription-data/subs-to-notify`	Créer un abonnement aux changements de données (le corps nomme les URIs de ressources surveillées et l'URI de rappel de notification)	201 + `Location`, ou 400 en cas de corps incorrect
DELETE	`/nudr-dr/v2/subscription-data/subs-to-notify/{subsId}`	Supprimer un abonnement aux changements de données	204, ou 404 si l'abonnement est inconnu

Voir Abonnements aux changements de données et DataChangeNotify pour le comportement de diffusion.

Tout chemin qui ne correspond pas aux ressources ci-dessus est répondu avec un 404 `ProblemDetails`.

Appels REST du backend HSS

Chaque opération SBI est satisfaite par un ou plusieurs appels au backend REST OmniHSS à `hss_api_base_url`. La relation est :

Opération Nudr_DR	Appel(s) REST HSS	Remarques
Lecture de l'abonnement à l'authentification	`GET /api/subscriber/imsi/{imsi}`, puis `GET /api/key_set/{id}` si le matériel clé est référencé par id	Le matériel clé en ligne dans le dossier d'abonné évite le second appel
Génération de vecteurs d'authentification	`POST /api/hlr/generate_auth_vectors`	Corps `{imsi, auts}` ; `auts` est `rand‖auts` hex sur resync
Mise à jour de SQN (PATCH)	`POST /api/hlr/update_sqn`	Corps `{imsi, sqn}`
Lecture AM / SM / sélection SMF	`GET /api/subscriber/imsi/{imsi}`	Récupération d'un seul abonné, mappé localement
Écriture de contexte AMF	`PUT /api/hlr/circuit_session/{imsi}`	Mise à jour de localisation/session de circuit ; également mis en cache localement
Écriture de contexte SMF	(aucun)	Cache ETS local uniquement
Lecture de données de politique (AM / SM)	`GET /api/subscriber/imsi/{imsi}`	Réutilise le mappage AM/SM et le reformate
Vérification de santé	`GET /api/status/api`	Pilote le point de terminaison de

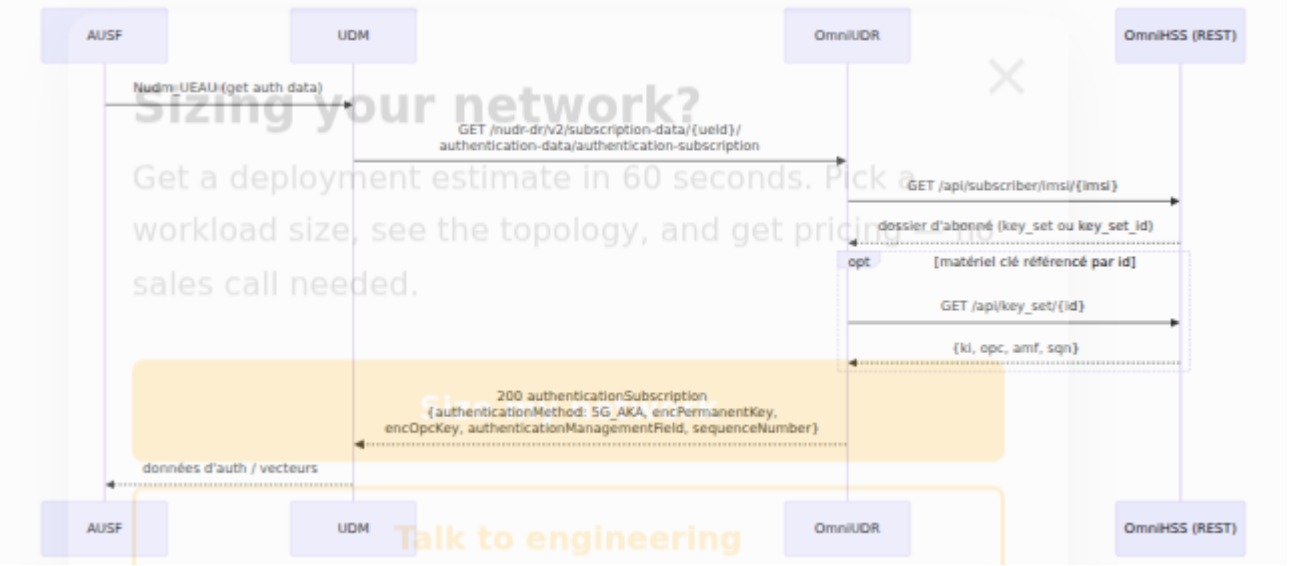
Opération Nudr_DR	Appel(s) REST HSS	Remarques
		gestion `/health/hss` et le gauge de santé

> Les points de terminaison `/api/hlr/*` (`generate_auth_vectors`, `update_sqn`, `circuit_session/{imsi}`) sont l'interface REST spécialement conçue du HSS pour les opérations de vecteurs d'authentification, SQN et localisation d'OmniUDR, aux côtés des points de terminaison d'abonné/ensemble de clés (`/api/subscriber/*`, `/api/key_set/*`) et de la sonde de santé (`/api/status/api`). Un abonné désactivé (`enabled: false`) est traité comme non trouvé.

Procédures clés

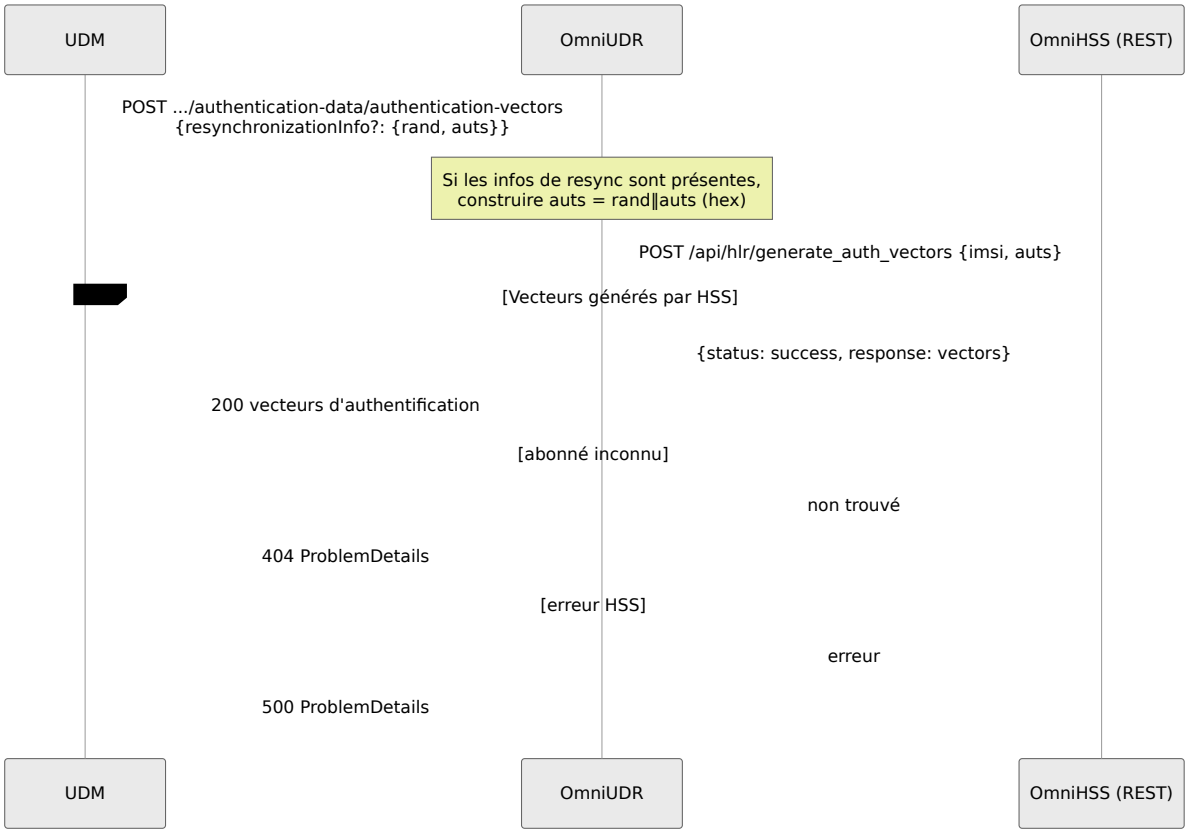
Lecture de l'abonnement à l'authentification (UDM → UDR → HSS)

Le flux principal UDR : le UDM (servant l'AUSF pendant 5G-AKA) demande à OmniUDR l'abonnement à l'authentification d'un abonné ; OmniUDR lit l'abonné et le matériel clé depuis le HSS et le mappe à la structure Nudr.

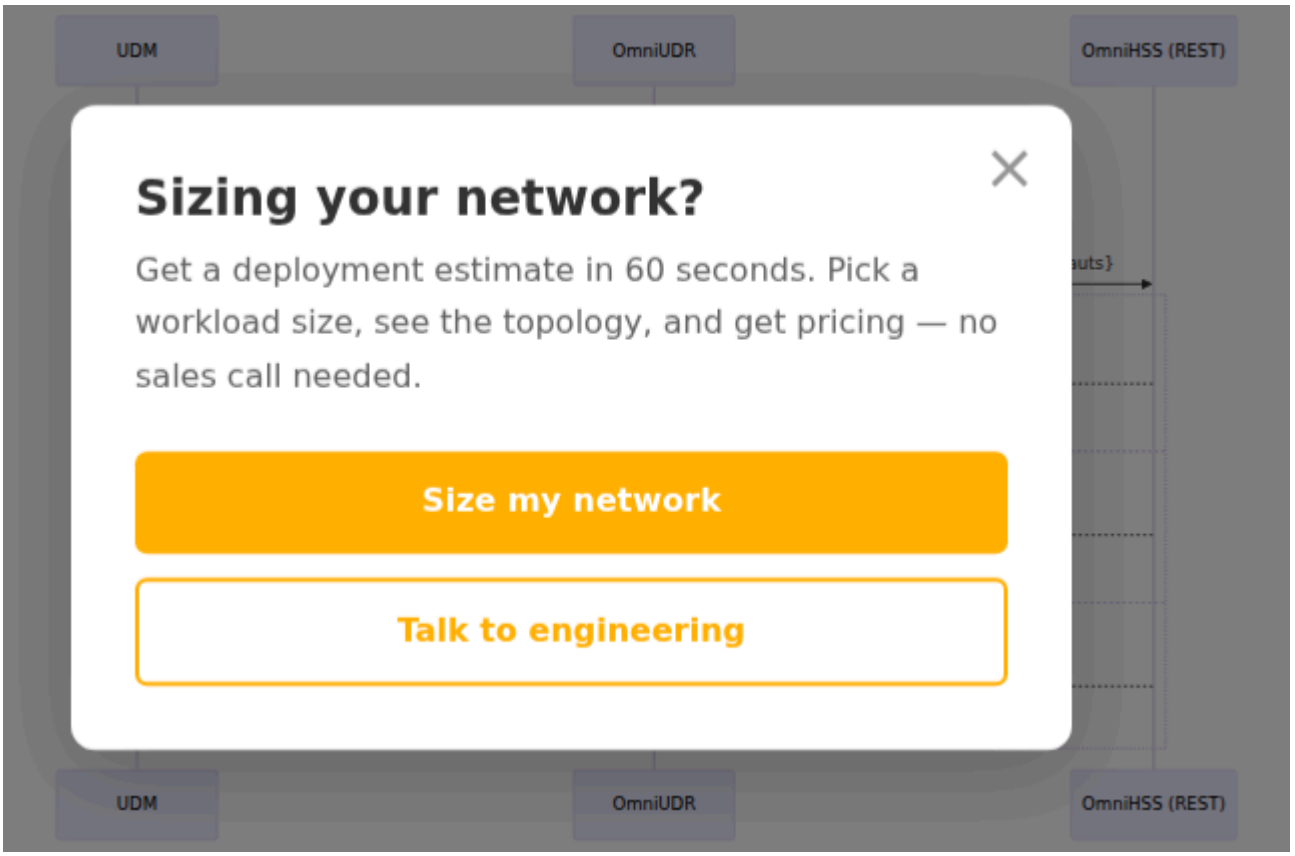


Génération de vecteurs d'authentification (propriétaire, avec resync)

OmniUDM appelle le POST non standard `authentication-vectors` afin que la génération de vecteurs Milenage et la gestion de SQN/resync soient gérées de manière centralisée au HSS. Lors d'une resynchronisation, le UDM passe `resynchronizationInfo` (`rand` + `auts`), qu'OmniUDR concatène dans le hex `rand‖auts` attendu par le HSS.



Lecture de données provisionnées (AM / SM / sélection SMF)



Pour les données AM, le mappage dérive plutôt `subscribedUeAmbr` de l'UE-AMBR EPC et attache un NSSAI synthétisé (voir `default_am_nssai`); pour les données de sélection SMF, il liste les APN de l'abonné comme informations DNN sous une seule clé S-NSSAI souscrite.

Comment le profil APN HSS devient des données d'abonnement SM 5G

L'enregistrement HSS 4G n'a pas de structures de gestion de session 5G, donc OmniUDR synthétise une entrée `dnnConfigurations` par APN provisionné (`epc_profile.apn_profiles[]`). Chaque champ est dérivé comme suit. Lorsque le champ HSS nommé est absent, le fallback est servi - un fallback généralisé signifie que les noms de champs du profil QoS APN ne correspondent pas (voir Dépannage → Les données SM servent un AMBR fixe).

Champ SM 5G	Dérivé de (champ HSS)
`singleNssai`	DNN → S-NSSAI via `dnn_snssai_map`
Clé DNN dans `dnnConfigurations`	Nom APN (`apn_identifier.apn`)
`pduSessionTypes`	APN `apn_identifier.ip_version` (voir t dessous)
`sscModes`	fixe : `SSC_MODE_1` par défaut, autorisé `SS` + `SSC_MODE_2`
`5gQosProfile.5qi`	`apn_qos_profile.qci`
`5gQosProfile.arp.priorityLevel`	`apn_qos_profile.allocation_retentio`
`5gQosProfile.arp.preemptCap`	`apn_qos_profile.pre_emption_capabil`
`5gQosProfile.arp.preemptVuln`	`apn_qos_profile.pre_emption_vulnera` (bool)
`sessionAmbr.uplink`	`apn_qos_profile.apn_ambr_ul_kbps`
`sessionAmbr.downlink`	`apn_qos_profile.apn_ambr_dl_kbps`
`chargingRules`	APN `pcrf_profile.charging_rules[]` (r priorité, 5QI/QCI, ARP, GBR/MBR, infos de

Types de session PDU. Le type de session PDU 5G est le jumeau 5G du type PDN 4G, dérivé de `ip_version` de l'APN, donc un APN IPv6 ou à double pile produit une session capable d'IPv6 au lieu du défaut uniquement IPv4. `defaultSessionType` est ce vers quoi le SMF revient lorsque l'UE ne demande pas un type spécifique autorisé.

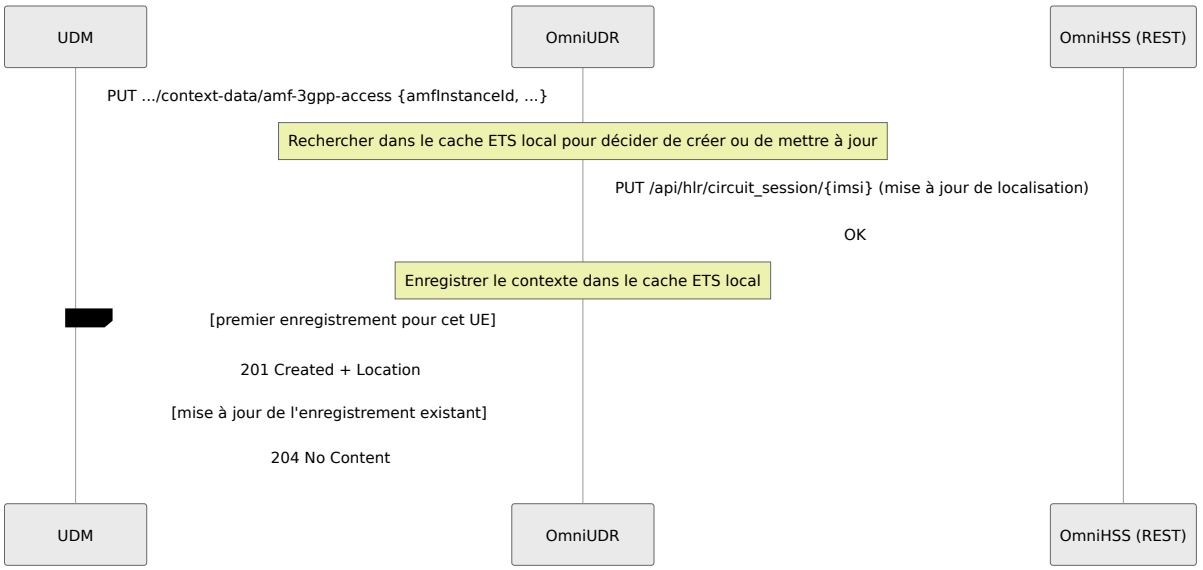
APN `ip_version`	`defaultSessionType`	`allowedSessionTypes`
`ipv4`	`IPV4`	`IPV4`
`ipv6`	`IPV6`	`IPV6`
`ipv4v6` / `ipv4_or_ipv6`	`IPV4V6`	`IPV4V6`, `IPV4`, `IPV6`
non défini / autre	`IPV4`	`IPV4`, `IPV4V6`

Les modes SSC ne sont pas provisionnables par abonné : chaque DNN est annoncé avec `SSC_MODE_1` par défaut et autorisé `SSC_MODE_1`/`SSC_MODE_2`. Selon TS 29.503 / 29.571, `SscModes.allowedSscModes` a `maxItems: 2` (le défaut plus au maximum une alternative), donc l'ensemble ne peut pas être élargi.

Les données d'abonnement de sélection SMF listent chaque APN provisionné comme un DNN sous une seule clé S-NSSAI souscrite `01`, indépendamment du mappage par DNN dans `dnn_snssai_map`. Dans un déploiement multi-slice où les DNN résident sur différents S-NSSAIs, utilisez les données d'abonnement AM/SM (qui portent le S-NSSAI par DNN correct) pour les décisions de slice ; la sélection SMF annonce uniquement la liste des DNN.

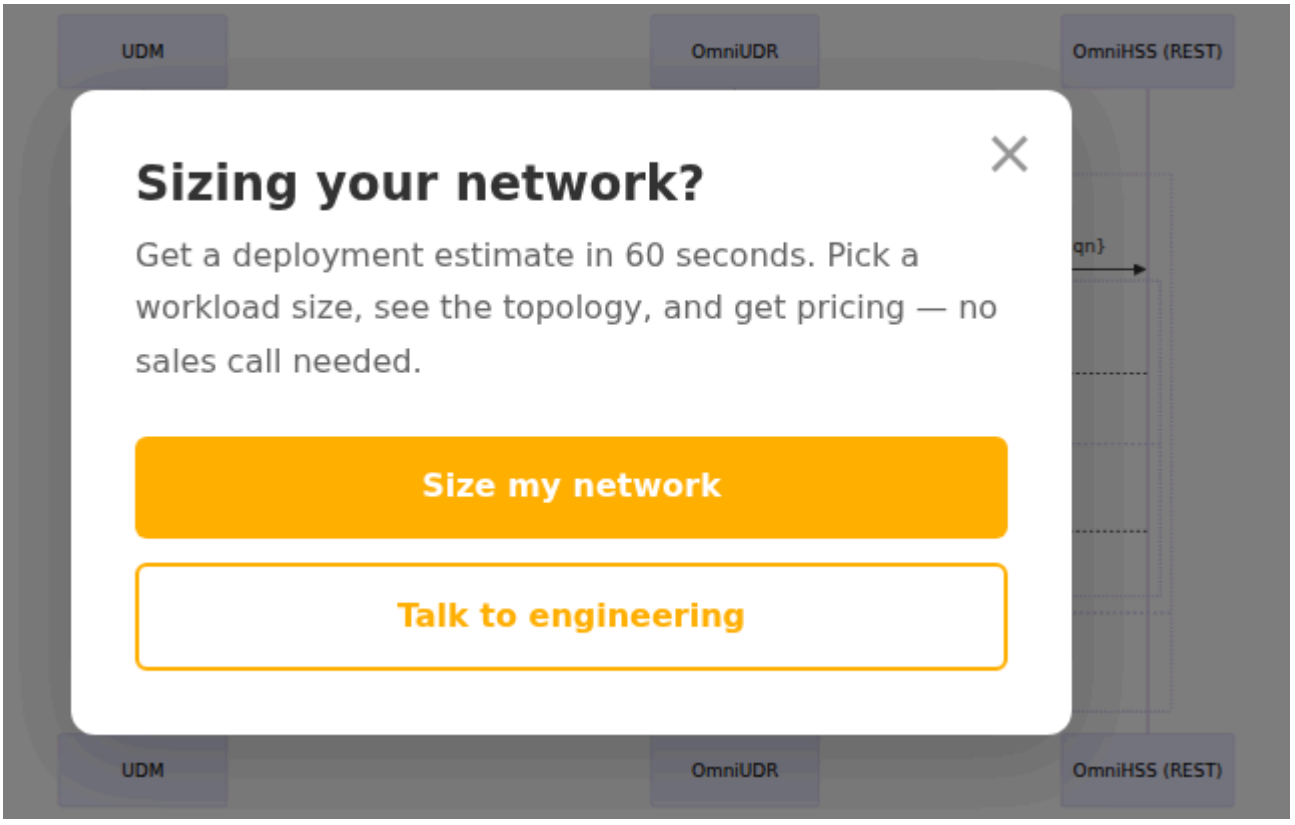
Mise à jour de SQN (PATCH abonnement à l'authentification)

Écriture du contexte d'enregistrement AMF



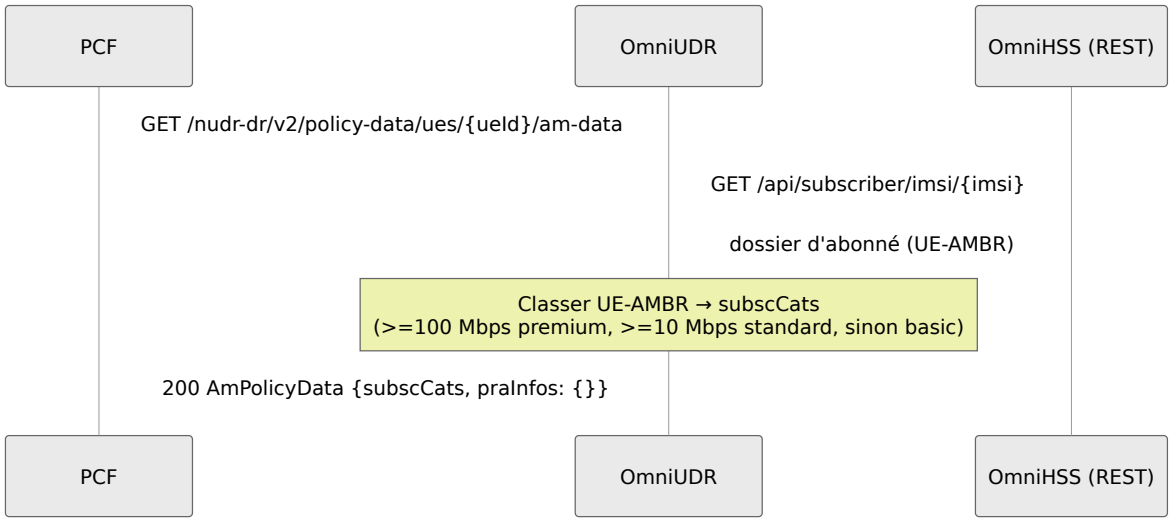
Écriture du contexte d'enregistrement SMF (cache local uniquement)

Le contexte d'enregistrement SMF est stocké uniquement dans le cache ETS local, indexé par `{ueId}/{pduSessionId}` ; il n'est jamais persistant au HSS et est perdu lors d'un redémarrage. La présence d'une entrée de cache existante décide de créer ou de mettre à jour.

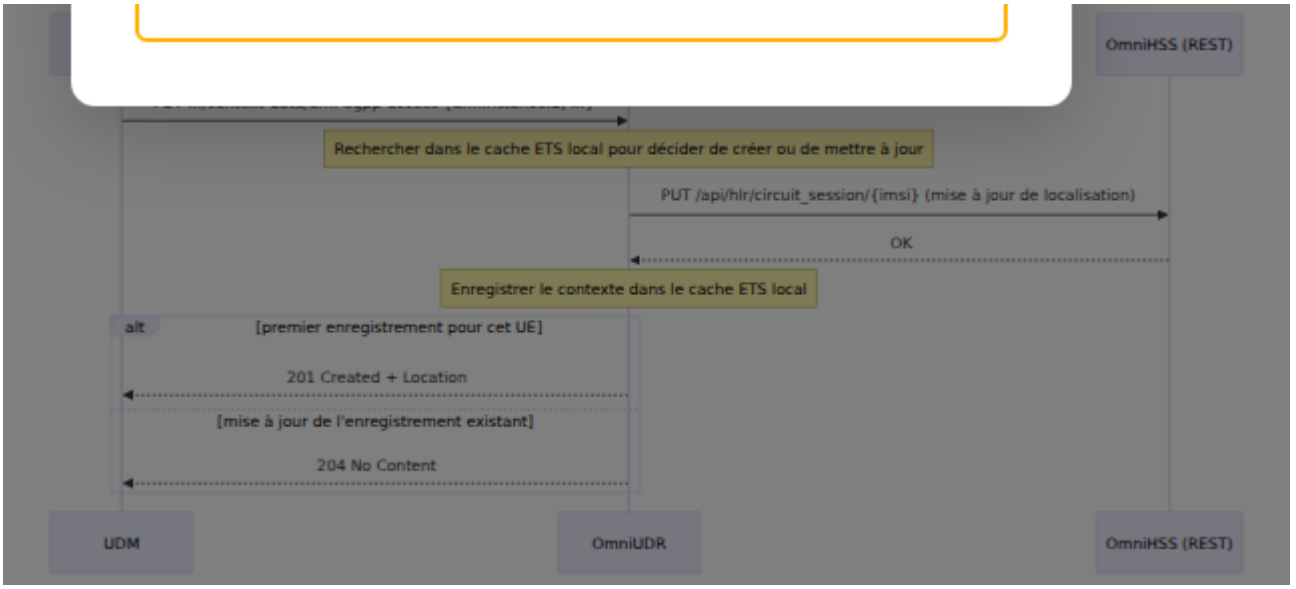


Lecture des données de politique AM (PCF → UDR → HSS)

Les données de politique AM sont grossières : OmniUDR lit l'UE-AMBR de l'abonné depuis le HSS et le classe dans une seule catégorie d'abonné (`premium` / `standard` / `basic`) ; `praInfos` est toujours vide.

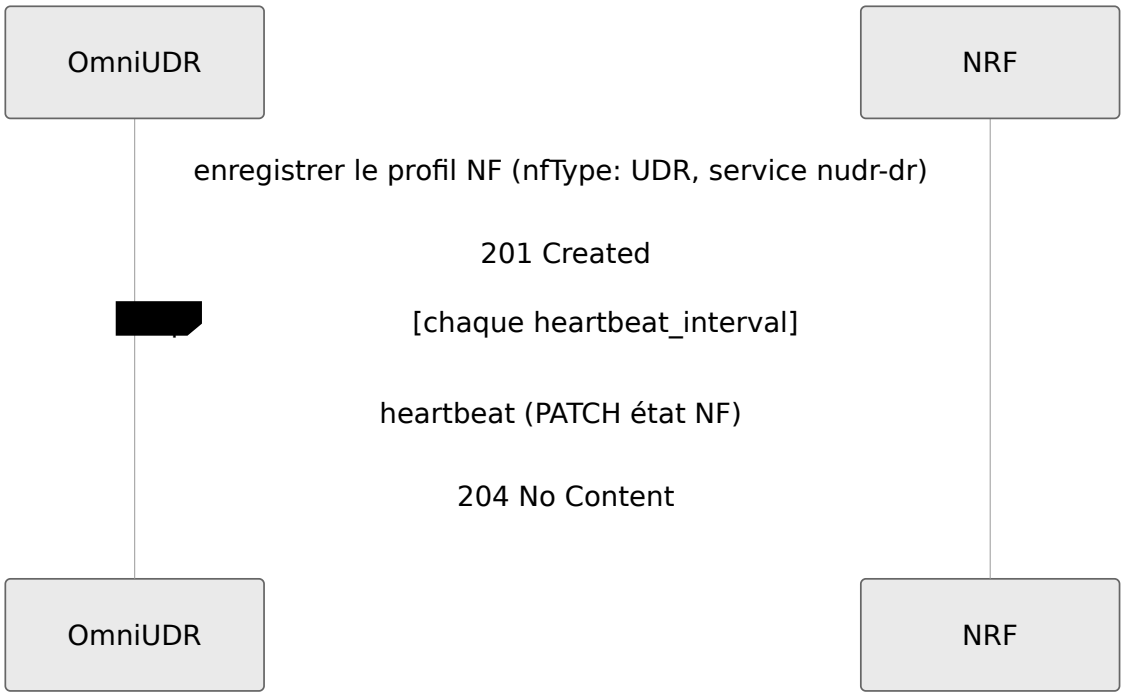


Lecture des données de politique SM (PCF → UDR → HSS)



Enregistrement et heartbeat NRF

Au démarrage, OmniUDR enregistre son profil NF auprès du NRF, puis envoie un heartbeat toutes les `heartbeat_interval` millisecondes afin que le registre continue de l'annoncer comme vivant. L'état d'enregistrement est exposé comme un gauge (voir Métriques), et une action OAM peut forcer la réinscription.



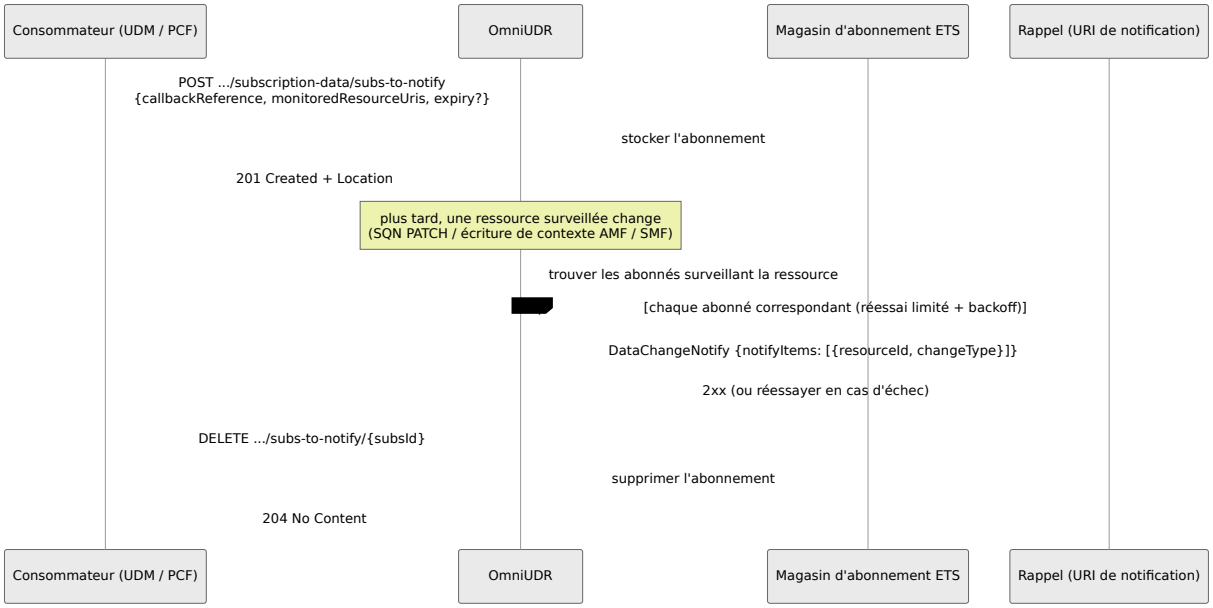
Abonnements aux changements de données et DataChangeNotify

OmniUDR implémente le service d'abonnement aux changements de données Nudr_DR (la ressource `subscription-data/subs-to-notify`, TS 29.504 subscriptions-to-notify). Un consommateur enregistre un abonnement qui nomme les URIs de ressources qu'il souhaite surveiller et une URI de rappel de notification, éventuellement avec une expiration. Les abonnements sont conservés dans un magasin d'abonnement en mémoire (ETS) ; une expiration, lorsqu'elle est fournie, est respectée par le magasin, qui supprime l'abonnement une fois qu'elle a expiré.

Lorsqu'une ressource surveillée change, OmniUDR émet un `DataChangeNotify` à l'URI de rappel de chaque abonné dont les URIs surveillées couvrent la ressource changée (une correspondance d'URI exacte, ou un préfixe de collection afin qu'un abonnement sur une ressource parent attrape ses ressources enfants). Chaque notification porte un type de changement : `ADD`, `REPLACE`, ou `REMOVE`. Dans cette version, les notifications se déclenchent lors d'une mise à jour de SQN (PATCH de l'abonnement à l'authentification) et lors

des écritures de contexte AMF et SMF (un premier enregistrement pour une ressource est un `ADD`, une écriture ultérieure est un `REPLACE`).

La livraison est faite sur la base du meilleur effort et n'affecte jamais la réponse SBI que le consommateur reçoit. Un rappel qui échoue ou qui est injoignable est réessayé avec un nombre d'essais limité et un backoff exponentiel (voir `notification_max_attempts` et `notification_backoff_ms`); une fois les tentatives épuisées, la notification est abandonnée et l'événement est enregistré.



← Vue d'ensemble

Dépannage OmniUDR

Tous les lectures Nudr retournent 404 / les recherches d'abonnés échouent

OmniUDR satisfait chaque lecture depuis le backend HSS. Si les lectures échouent :

1. Confirmez que `hss_api_base_url` est accessible depuis l'hôte OmniUDR et que le HSS est opérationnel. Utilisez la vérification intégrée : `GET /api/health/hss` sur l'API de gestion, et surveillez le compteur `omni_udr_hss_health` (1 = opérationnel, 0 = hors service).
2. Confirmez que l'abonné existe et est **activé** dans le HSS. Un abonné désactivé (`enabled: false`) est traité comme non trouvé (404 pour le consommateur).
3. Confirmez que le `{ueId}` utilise la forme `imsi-<chiffres>` (un IMSI nu est également accepté).
4. Surveillez `omni_udr_hss_requests_total{result="failure"}` et `omni_udr_hss_request_duration_ms` pour distinguer un HSS hors service/lent d'un abonné réellement absent.

Le consommateur reçoit 5xx / erreurs lorsque le HSS est inaccessible

Lorsque le backend HSS est hors service ou expire (5 s par requête, 3 s pour la vérification de santé), le client REST HSS retourne une erreur et OmniUDR la remonte au consommateur (typiquement un `404` sur les ressources de lecture, ou un `502` sur le proxy d'abonné de l'API de gestion). Cela est attendu -

OmniUDR n'a pas de magasin d'abonnés local sur lequel se reposer. Rétablissez l'accessibilité du HSS ; `omni_udr_hss_health` revient à 1 une fois que la prochaine vérification réussit. Notez que `test_subscribers` est un outil de laboratoire et **n'est pas** un repli automatique en cas de HSS hors service dans cette version.

PATCH pour la mise à jour SQN retourne 200 au lieu de 204

Un `200` avec un corps `PatchResult` signifie que le HSS a **rejeté** la mise à jour SQN ; le tableau `report` nomme le chemin échoué et la raison du HSS. Une mise à jour réussie retourne `204`. Vérifiez les journaux du HSS et le chemin `/api/hlr/update_sqn`. Un corps PATCH sans `sequenceNumber` est traité comme une opération sans effet réussie (`204`).

La génération de vecteurs d'authentification retourne 404 / 500

Le POST `authentication-vectors` délègue à l'endpoint HSS `/api/hlr/generate_auth_vectors`. Un `404` signifie que l'abonné est inconnu du HSS ; un `500` signifie que le HSS a retourné une erreur ou une forme de réponse inattendue. Lors d'une resynchronisation, assurez-vous que le UDM a fourni `resynchronizationInfo` avec à la fois `rand` et `auts` - OmniUDR les concatène en `rand||auts` hex que le HSS attend. Cet endpoint est propriétaire et seul OmniUDM devrait l'appeler.

Les données de politique semblent minces

Les données de politique AM retournent uniquement des `subscCats` grossiers dérivés du niveau UE-AMBR de l'abonné (`premium` ≥ 100 Mbps, `standard` ≥ 10 Mbps, sinon `basic`), et `praInfos` est vide. Le niveau utilise celui des UE-AMBR

montants ou descendants qui est le plus élevé. Lorsque le profil EPC HSS de l'abonné ne porte pas de UE-AMBR, OmniUDR sert un défaut de 1000 Kbps, donc un abonné non provisionné ou mal mappé tombe dans `basic` - si un abonné que vous vous attendez à être `premium`/`standard` montre `basic`, vérifiez que l'UE-AMBR EPC (`ue_ambr_ul_kbps` / `ue_ambr_dl_kbps`) est provisionné dans le HSS. Les données de politique SM sont construites à partir des profils APN/QoS et des règles de facturation PCRF de l'abonné. Il n'y a pas de magasin de politique 5G dédié ; c'est par conception dans l'architecture actuelle.

Mismatch de tranche de données AM (S-NSSAI)

L'enregistrement HSS 4G ne porte pas d'abonnement de tranche 5G, donc le NSSAI de données AM est synthétisé par OmniUDR et par défaut aux tranches de laboratoire standard (eMBB, URLLC, mIoT). Si un S-NSSAI S-D qualifié demandé par un UE est rejeté par l'AMF (cause 5GMM #62), alignez `default_am_nssai` (et, par DNN, `dnn_snssai_map`) avec le NSSAI servi par l'AMF pour le déploiement. Voir la Référence de Configuration.

Les données SM servent un AMBR fixe pour chaque abonné

Si les données d'abonnement SM montrent le même AMBR de session / QoS pour tous les abonnés, indépendamment du provisionnement, les noms de champs du profil QoS APN HSS ne sont pas appariés et OmniUDR revient aux valeurs par défaut. Le mappage attend les champs `ApnQosProfile` d'OmniHSS (`apn_ambr_dl_kbps`, `apn_ambr_ul_kbps`, `allocation_retention_priority`, `pre_emption_capability`, `pre_emption_vulnerability`, `qci`). Vérifiez que l'enregistrement d'abonné HSS porte ces champs.

Le cache de contexte semble obsolète après des changements

HSS hors bande

Le cache d'enregistrement AMF/SMF ne suit que les écritures de contexte et est utilisé pour la décision 201-vs-204 ; il ne met pas en cache les données de lecture. S'il doit être vidé (par exemple après un cycle de test), videz-le via `DELETE /api/oam/subscriber-cache/{imsi}` (ou `.../all`).