← Vue d'ensemble

Référence de configuration OmniCHF

La configuration de l'opérateur est lue à partir de la clé d'environnement de l'application `:omnichf`, peuplée au démarrage par `config/runtime.exs`. Chaque clé peut être remplacée par une variable d'environnement du système d'exploitation. L'exemple ci-dessous montre la configuration complète à l'exécution avec les valeurs appliquées lorsque la variable d'environnement correspondante n'est pas définie.

```
config :omnichf,
  # Écouteur SBI (interface basée sur le service)
  sbi_scheme: "http",
  sbi_addr:   "127.0.0.14",
  sbi_port:   7777,

  # Enregistrement NRF / battement de cœur
  nrf_uri:    "http://127.0.0.1:7777",

  # Identité PLMN servie
  mcc: "999",
  mnc: "70",

  # Observabilité
  prometheus_metrics_port: 9568,

  # Cadence de battement de cœur NRF
  heartbeat_interval: 10_000,

  # Intégration CGRateS pour la tarification / le solde
  cgrates_enabled: true,
  cgrates_url:     "http://localhost:2080/jsonrpc",
  cgrates_tenant:  "cgrates.org",
  cgrates_timeout: 5000

# Journalisation JSON structurée
config :logger, :default_formatter,
  format: {OmniLogger.JsonFormatter, :format},
  metadata: :all
```

Paramètres principaux

Paramètre	Type	Requis	Par défaut
`sbi_scheme`	Chaîne	Non	`"http"`
`sbi_addr`	Chaîne	Non	`"127.0.0.14"`
`sbi_port`	Entier	Non	`7777`
`nrf_uri`	Chaîne	Non	`"http://127.0.0.1:7777"`
`mcc`	Chaîne	Non	`"999"`

Paramètre	Type	Requis	Par défaut
`mnc`	Chaîne	Non	`"70"`
`prometheus_metrics_port`	Entier	Non	`9568`
`heartbeat_interval`	Entier (ms)	Non	`10000`

Paramètres CGRateS

Ces clés contrôlent l'intégration avec le moteur de tarification et de solde externe CGRateS. Voir Intégration CGRateS pour des détails comportementaux.

Paramètre	Type	Requis	Par défaut
`cgrates_enabled`	Booléen	Non	`true` (`"true"`)
`cgrates_url`	Chaîne	Non	`"http://localhost:2080/jsonrpc"`

Paramètre	Type	Requis	Par défaut
`cgrates_tenant`	Chaîne	Non	`"cgrates.org"`
`cgrates_timeout`	Entier (ms)	Non	`5000`

> **Par défaut lorsque désactivé au moment de la construction :** le défaut compilé (utilisé si l'environnement de l'application n'est jamais peuplé par `runtime.exs`) est `cgrates_enabled: false`. Dans un déploiement normal, `runtime.exs` le définit, et là le défaut est `true`. Le profil de test force `cgrates_enabled: false`.

Clés avancées / optionnelles

Ces clés ne sont pas peuplées par le `runtime.exs` par défaut mais sont lues à partir de l'environnement `:omnichf` si présent. Elles ajustent la résolution des groupes de tarification, la tarification hors ligne et la génération d'ID de

tarification. Elles n'ont pas de variable d'environnement dédiée et sont définies directement dans le bloc de configuration `:omnichf`.

Paramètre	Type	Requis	Par défaut
`rating_groups`	Carte	Non	`%{}`
`default_rating_group`	Entier	Non	`1`
`offline_charging_enabled`	Booléen	Non	`false`
`cdr_output_dir`	Chaîne	Non	`"/var/log/omnichf/cdr"`
`node_id`	Entier	Non	`1`

Paramètre	Type	Requis	Par défaut

Groupes de tarification

`rating_groups` associe un **DNN** (la clé de la carte, une chaîne) au **groupe de tarification** (la valeur, un entier) appliqué aux sessions de ce DNN. Le groupe de tarification résolu apparaît dans le champ `ratingGroup` de `multipleUnitInformation` retourné au consommateur et dans le champ `rating_group` du CDR. La résolution est :

1. Si le DNN de la session est une clé dans `rating_groups`, utilisez cette valeur.
2. Sinon, utilisez `default_rating_group`.

```
config :omnichf,
  rating_groups: %{
    "internet" => 10,
    "ims"      => 20,
    "iot"      => 30
  },
  default_rating_group: 1
```

La correspondance se fait sur la chaîne DNN exacte. Un DNN sans entrée (et sans `default_rating_group`) revient au groupe de tarification `1`. Une valeur `rating_groups` mal formée (pas une carte) est ignorée et le défaut est utilisé, donc une mauvaise configuration ne bloque jamais la tarification.

Journalisation

La journalisation JSON structurée est configurée via le `:logger` `:default_formatter`.

Paramètre	Type	Par défaut	Description
`format`	Tuple	`{OmniLogger.JsonFormatter, :format}`	Formateur de ligne de journal. Émet du JSON structuré, adapté à l'ingestion par un expéditeur de journaux.
`metadata`	Atome / Liste	`:all`	Métadonnées de journal incluses dans chaque ligne. `:all` émet toutes les métadonnées attachées (SUPI, DNN, ID de session PDU, ID de tarification, procédure, etc.).

Interfaces de gestion

OmniCHF expose des surfaces de gestion destinées aux opérateurs aux côtés du SBI. Ces ports sont pertinents pour la surveillance et l'OAM mais ne font pas partie du service Nchf :

Interface	Port	Schéma	But
SBI (Nchf)	`sbi_port` (par défaut `7777`)	HTTP	Trafic de tarification inter-NF 5G.
Métriques Prometheus	`prometheus_metrics_port` (par défaut `9568`)	HTTP	Point de terminaison de récupération des métriques.
API de gestion	`8443`	HTTPS	Lectures d'état, lectures de session/statistiques/état de CGRateS, recherche de CDR et actions OAM (configuration, vidage de CDR, niveau de journal, interruption de session, réenregistrement NRF).
Panneau de contrôle	`7443`	HTTPS	Interface Web pour les ressources, la configuration, la licence et les journaux.

Points de terminaison de l'API de gestion / OAM

L'API de gestion sur le port `8443` (HTTPS) expose les surfaces suivantes pour les opérateurs :

Méthode	Chemin	But
`GET`	`/api/status/api`	Lire l'état de l'API/service.
`GET`	`/api/status/license`	Lire l'état de la licence.
`GET`	`/api/status/nrf`	Lire l'état de l'enregistrement NRF.
`GET`	`/api/status/nf`	Lire l'état du NF (nœud).
`GET`	`/api/sessions`	Lister les sessions de tarification actives.
`GET`	`/api/sessions/{id}`	Inspecter une seule session de tarification par `charging_data_ref`.
`GET`	`/api/statistics`	Lire les statistiques de tarification (nombre de sessions actives, volume total facturé, accessibilité CGRateS).
`GET`	`/api/health/cgrates`	Lire l'état de connectivité CGRateS.
`GET`	`/api/oam/config`	Lire la configuration à l'exécution.
`PATCH`	`/api/oam/config/{id}`	Mettre à jour une clé de configuration à l'exécution.
`POST`	`/api/oam/charging_session`	Interrompre une session de tarification par `charging_data_ref`.

Méthode	Chemin	But
`GET`	`/api/oam/cdr`	Rechercher des CDR complétés. Paramètres de requête : `supi`, `dnn`, `date_from`, `date_to` (tous optionnels).
`POST`	`/api/oam/cdr/flush`	Forcer le vidage de l'utilisation accumulée en tant que CDR pour une session (`{"charging_data_ref": "<ref>"}`) ou toutes les sessions suivies (`{}` / `"all"`).
`POST`	`/api/oam/log_level`	Changer le niveau de journal à l'exécution.
`POST`	`/api/oam/nrf/reregister`	Déclencher le réenregistrement NRF.

> **Visibilité du stockage de session :** les surfaces de gestion orientées session (`/api/sessions`, `/api/statistics`, `/api/oam/charging_session`, `/api/oam/cdr/flush`) présentent chaque session active. Le chemin de tarification SBI en direct maintient chaque session dans un processus **worker** par session, et chaque session worker est reflétée dans le **stockage Context** en mémoire lors de la création et de la mise à jour et supprimée à l'arrêt. Les surfaces de gestion interrogent d'abord le registre des workers et reviennent au Context, donc une session active est toujours visible et actionnable. Voir Stockage de session et visibilité de gestion pour ce que cela signifie opérationnellement. La recherche de CDR (`/api/oam/cdr`) lit les fichiers CDR hors ligne et est indépendante des deux magasins.

← Vue d'ensemble

Référence des métriques OmniCHF

Les métriques sont exposées sur `prometheus_metrics_port` (par défaut `9568`) à `/metrics`.

Métriques de Chargement

Métrique: `omni_chf_charging_initial_count`
Type: Compteur
Description: Requêtes INITIAL (Créer) de session de chargement reçues.
Étiquettes: `supi`

Métrique: `omni_chf_charging_update_count`
Type: Compteur
Description: Requêtes UPDATE de session de chargement reçues.
Étiquettes: `ref`

Métrique: `omni_chf_charging_release_count`
Type: Compteur
Description: Requêtes RELEASE de session de chargement reçues.
Étiquettes: `ref`

Métrique: `omni_chf_charging_creates_total`
Type: Compteur
Description: Résultats des opérations de création.
Étiquettes: `result` (`success` / `failure`)

Métrique: `omni_chf_charging_updates_total`
Type: Compteur
Description: Résultats des opérations de mise à jour.
Étiquettes: `result` (`success` / `failure`)

Métrique: `omni_chf_charging_releases_total`
Type: Compteur
Description: Résultats des opérations de libération. Libéré via le SBI avec un CGRateS terminate réussi (`success`), libéré via le SBI où le `TerminateSession` de CGRateS a échoué mais le CDR a été finalisé (`failure`, affiché au consommateur comme un `500`), ou finalisé de force via le point de terminaison OAM CDR-flush (`oam_flush`).
Étiquettes: `result` (`success` / `failure` / `oam_flush`)

Métrique: `omni_chf_sessions_active_count`
Type: Jauge
Description: Sessions de chargement actives actuellement conservées en mémoire. Émis à chaque création et suppression de contexte, ce qui inclut désormais chaque session de travail en direct miroir, donc la jauge suit le véritable nombre de sessions actives.
Étiquettes: aucune

```
# Taux de création de session de chargement
rate(omni_chf_charging_creates_total{result="success"}[5m])

# Ratio d'échec de création
sum(rate(omni_chf_charging_creates_total{result="failure"}[5m]))
  / sum(rate(omni_chf_charging_creates_total[5m]))

# Sessions actuellement actives
omni_chf_sessions_active_count
```

Métriques CGRateS

Métrique: `omni_chf_cgrates_requests_total`
Type: Compteur
Description: Requêtes JSON-RPC CGRateS.
Étiquettes: `operation` (méthode SessionS), `result` (`success` / `failure`)

Métrique: `omni_chf_cgrates_request_duration_ms`
Type: Histogramme
Description: Durée des requêtes JSON-RPC CGRateS en millisecondes.

Étiquettes: `operation`
Seaux: 5, 10, 25, 50, 100, 250, 500, 1000, 2500

Métrique: `omni_chf_cgrates_health`
Type: Jauge
Description: État de connectivité CGRateS (`1` = accessible, `0` = hors ligne/désactivé).
Étiquettes: aucune

```
# Latence p99 de CGRateS par opération
histogram_quantile(0.99,
  sum by (le, operation)
(rate(omni_chf_cgrates_request_duration_ms_bucket[5m])))

# Taux d'erreur de CGRateS
sum(rate(omni_chf_cgrates_requests_total{result="failure"}[5m]))

# Accessibilité de CGRateS (alerte lorsque 0)
omni_chf_cgrates_health
```

Métriques NRF

Métrique: `omni_chf_nrf_registration_status`
Type: Jauge
Description: État d'enregistrement NRF (`1` = enregistré, `0` = non enregistré).
Étiquettes: `nf_type`

```
omni_chf_nrf_registration_status
```

Métriques BEAM VM

Les jauges de santé d'exécution standard sont également exportées.

Métrique	Type	Description
`beam_memory_total`	Jauge	Mémoire totale BEAM (octets)
`beam_memory_processes` / `beam_memory_processes_used`	Jauge	Mémoire de processus allouée / utilisée
`beam_memory_system`	Jauge	Mémoire système (non-processus)
`beam_memory_atom` / `beam_memory_atom_used`	Jauge	Mémoire d'atome allouée / utilisée
`beam_memory_binary`	Jauge	Mémoire binaire
`beam_memory_code`	Jauge	Mémoire de code
`beam_memory_ets`	Jauge	Mémoire de table ETS
`beam_processes_count`	Jauge	Nombre de processus Erlang
`beam_ports_count`	Jauge	Nombre de ports Erlang
`beam_atom_count`	Jauge	Nombre d'atomes
`beam_vm_uptime`	Jauge	Temps de fonctionnement de la VM (secondes)

← Vue d'ensemble

Guide des opérations OmniCHF

Table des matières

1. Vue d'ensemble
2. Cycle de vie de la session de facturation
3. Rôle 3GPP et références de spécification
4. Points de terminaison SBI
5. Intégration CGRateS
6. Procédures clés

Vue d'ensemble

OmniCHF implémente la **Fonction de Facturation (CHF)** du cœur 5G. Le CHF fournit une facturation convergente en ligne et hors ligne pour les sessions PDU 5G via le service **Nchf_ConvergedCharging** (3GPP TS 32.290 / TS 32.291 / TS 32.255).

Un consommateur de facturation, typiquement le **SMF**, crée une session de facturation pour chaque session PDU, rapporte l'utilisation pendant la durée de cette session, et enfin la libère. OmniCHF traduit chaque demande de facturation 5G en un appel JSON-RPC de SessionS **CGRateS** pour l'autorisation de crédit en temps réel (facturation en ligne / gestion de quota) et génère un **Enregistrement de Données de Facturation (CDR)** lors de la libération de la session (facturation hors ligne).

Chaque session de facturation est détenue par un **ChargingWorker** GenServer dédié sous un `DynamicSupervisor` (architecture de processus par session). Tout l'état d'une session (SUPI, DNN, S-NSSAI, volume/durée accumulés, groupe de

tarification, état de session CGRateS, ID de facturation) est conservé dans ce processus de travail. L'échec d'une session ne nuit pas aux autres. L'état de session est conservé en mémoire et est reconstruit à mesure que les consommateurs créent de nouvelles sessions de facturation après un redémarrage.

Responsabilités principales :

- Servir le service `nchf-convergedcharging/v3` sur le SBI.
- Allouer un `chargingDataRef` par session et gérer son cycle de vie INITIAL → UPDATE → TERMINATE.
- Accorder des unités de service (GSU) en réponse aux Unités de Service Demandées (RSU), retournant le volume/temps accordé et les **déclencheurs** de gestion de quota au consommateur.
- Faire le lien avec CGRateS pour la tarification et la gestion des soldes (facturation en ligne).
- Émettre des CDR dans le journal d'application et, en option, dans des fichiers de facturation hors ligne par jour.
- S'enregistrer auprès du NRF et maintenir un heartbeat.

Cycle de vie de la session de facturation

Chaque session PDU correspond à une session de facturation, suivie par un `chargingDataRef` (UUID).

État	Déclencheur	Action
Créé	`POST /chargingdata`	ChargingWorker créé, CGRateS `InitiateSession` appelé, quota initial accordé
Mis à jour	`POST /chargingdata/{ref}/update`	CGRateS `UpdateSession` appelé, utilisation accumulée, numéro de séquence incrémenté, nouveau quota accordé
Libéré	`POST /chargingdata/{ref}/release`	CGRateS `TerminateSession` appelé, CDR construit et émis, worker arrêté

Stockage de session et visibilité de gestion

OmniCHF conserve l'état des sessions actives dans **deux** magasins en mémoire qui sont synchronisés, de sorte que les surfaces de gestion voient et peuvent agir sur chaque session active :

Magasin	Support	Contient	Rôle
Registre des workers	`ChargingWorker` GenServer par session sous un `DynamicSupervisor`, indexé dans un `Registry`	État complet par session (volume/durée accumulés, numéro de séquence, état CGRateS)	Source de vérité pour une session active. La création génère un worker ; la mise à jour et la libération résolvent d'abord le worker
Magasin de contexte	Une seule carte `Agent` en mémoire indexée par `chargingDataRef`	Un miroir des mêmes attributs de session	Modèle de lecture pour l'observabilité et les surfaces OAM

Sur le SBI, **Créer** génère toujours un worker, et **Mettre à jour** et **Libérer** résolvent d'abord ce worker, ne tombant sur le magasin de contexte que si aucun worker n'est trouvé (par exemple après un blip de superviseur). Chaque session de worker active est reflétée dans le magasin de contexte lors de la création et à chaque mise à jour, et est retirée lorsque le worker s'arrête (libération normale, abandon ou crash). Les surfaces de gestion interrogent d'abord le registre des workers et tombent ensuite sur le Contexte, de sorte que les deux magasins présentent une vue cohérente.

Opérationnellement, cela signifie :

- Une session créée normalement est entièrement visible à travers les surfaces de gestion : elle apparaît dans `GET /api/sessions` et dans les totaux statistiques, et peut être abandonnée (`POST /api/oam/charging_session`) et vidée de CDR (`POST /api/oam/cdr/flush`) via les points de terminaison OAM pendant qu'elle est active.
- Le compteur de sessions actives, les statistiques et les listes de sessions reflètent le véritable ensemble de sessions actives, et pas seulement un sous-ensemble de secours.

- Les deux magasins sont conservés en mémoire et sont repopulés à mesure que les consommateurs créent de nouvelles sessions après un redémarrage. Comme les sessions sont conservées dans un registre local, un `chargingDataRef` est détenu par l'instance qui l'a créé.

Le compteur `omni_chf_sessions_active_count` et les compteurs `omni_chf_charging_creates_total` / `omni_chf_charging_releases_total` corroborent le compte de sessions actives rapporté par `GET /api/sessions`.

Rôle 3GPP et références de spécification

Spécification	Pertinence
TS 23.501	Architecture du système 5G : rôle du CHF, concept de facturation convergente
TS 32.240	Architecture et principes de facturation
TS 32.290	Architecture de facturation du système 5G et flux de messages
TS 32.291	Facturation du système 5G : API du service **Nchf_ConvergedCharging** (HTTP/2 SBI), modèle de données ChargingDataRequest/Response
TS 32.255	Facturation du domaine de connectivité de données 5G (facturation de session PDU)
TS 32.251	Format de l'ID de facturation (`chargingId`), clause 5.2.1.6 du TS 32.251
TS 32.298	Descriptions et encodage des paramètres CDR
TS 29.500 / 29.501	Cadre commun SBI 5G et conception API
TS 29.510	Enregistrement NF et heartbeat avec le NRF

Références de procédure Nchf_ConvergedCharging (TS 32.291) :

Procédure	Clause
Création de données de facturation (INITIAL)	6.1.3.2.1
Mise à jour des données de facturation (UPDATE)	6.1.3.2.2
Libération des données de facturation (TERMINATE)	6.1.3.2.3
Notification de facturation (REAUTHORIZATION / ABORT_CHARGING)	6.1.6.2

Points de terminaison SBI

Tous les points de terminaison Nchf sont servis sous l'URL de base `{sbi_scheme}://{sbi_addr}:{sbi_port}` et le chemin de base `/nchf-convergedcharging/v3`.

Méthode	Chemin	Description
`POST`	`/chargingdata`	Créer une session de facturation (**INITIAL**). Alloue un `chargingDataRef` démarre une session CGRateS, accorde un quota initial.
`POST`	`/chargingdata/{chargingDataRef}/update`	Mettre à jour une session de facturation (**UPDATE**). Rapporte l'utilisation et demande un quota supplémentaire.
`POST`	`/chargingdata/{chargingDataRef}/release`	Libérer une session de facturation (**TERMINATE**). Rapporte l'utilisation finale, génère un CDR, termine la sessio CGRateS.
`POST`	`/notify`	Déclencheur de notification de facturation. Instruire OmniCH d'envoyer une

Méthode	Chemin	Description
		notification `REAUTHORIZATION` ou `ABORT_CHARGING` à l'`notifyUri` stocké du consommateur.

Lors de la libération, un `500` (cause `SYSTEM_FAILURE`, titre "Échec de la terminaison de la facturation") est retourné lorsque le `TerminateSession` de CGRateS échoue. Le CDR est toujours finalisé et écrit, mais l'échec est signalé afin qu'une incohérence dans le backend de facturation (une session CGRateS laissée ouverte) ne soit pas cachée derrière un faux `204`. Voir Les retours de libération 500.

Validation des demandes

Chaque corps de demande est validé pour les éléments d'information obligatoires suivants avant traitement. Un IE manquant retourne `400 Bad Request` avec la cause `MANDATORY_IE_MISSING` :

IE obligatoire	Type
`nfConsumerIdentification`	objet
`invocationTimeStamp`	chaîne (ISO 8601)
`invocationSequenceNumber`	entier

ChargingDataRequest : Champs clés consommés

Champ	Type	Utilisé dans	Descri
`subscriberIdentifier`	chaîne	Tous	SUPI (par exemple `999700000000001`) le `Account` de CG à `nfConsumerIdenti` si absent.
`notifyUri`	chaîne	Créer	URI de rappel stoc notification de fac ultérieure (REAUT ABORT_CHARGING
`pDUSessionChargingInformation`	objet	Tous	Détails de la sessi `pduSessionInform` ID de session PDU type RAT, `qoSInfo` `networkSlicingIn`
`multipleUnitUsage`	tableau	Mise à jour, Libération	Conteneurs d'utilis rapportés. Les ent `usedUnitContaine` élément sont addi le volume/durée a Transporte égalem `ratingGroup` dem
`requestType`	chaîne	Tous	`INITIAL_REQUEST` `UPDATE_REQUEST`, `TERMINATION_REQU`

ChargingDataResponse : Champs clés retournés

Champ	Type	Description
`invocationSequenceNumber`	entier	`1` lors de la création ; incrémenté à chaque mise à jour.
`invocationTimeStamp`	chaîne	Horodatage ISO 8601 de la réponse.
`multipleUnitInformation`	tableau	Unités accordées : une entrée avec `resultCode: "SUCCESS"`, `grantedUnit` (`totalVolume`, `time`) et le `ratingGroup` résolu.
`triggers`	tableau	Déclencheurs de gestion de quota instruisant le consommateur sur le moment de revenir : un `QUOTA_THRESHOLD` (IMMEDIATE_REPORT à la moitié du volume accordé) et une `TIME_LIMIT` (DEFERRED_REPORT à 3600 s).

Intégration CGRateS

CGRateS est le moteur de tarification et de solde externe. OmniCHF agit comme un client SessionS, mappant chaque opération de facturation Nchf sur une méthode JSON-RPC SessionS :

Opération Nchf	Méthode SessionS CGRateS
Créer (INITIAL)	`SessionSv1.InitiateSession`
Mettre à jour (UPDATE)	`SessionSv1.UpdateSession`
Libérer (TERMINATE)	`SessionSv1.TerminateSession`
Vérification de santé	`SessionSv1.GetActiveSessions` (limite 1)

La valeur `MaxUsage` que CGRateS retourne est directement renvoyée au consommateur comme le `grantedUnit` (à la fois `totalVolume` et `time`).

Comportement activé vs désactivé

Aspect	`cgrates_enabled: true`	`cgrates_enabled: false` **(mode de contournement)**
Autorisation de crédit	Réel : CGRateS tarifie le compte et retourne `MaxUsage`	Aucun : un octroi fixe de **86400** unités est retourné pour chaque création et mise à jour
Impact sur le solde	Le solde de l'abonné est débité/réservé dans CGRateS	Aucun solde n'est touché
Terminer	`TerminateSession` finalise la session CGRateS et le CDR	Pas d'opération
Vérification de santé	Probe en direct contre `cgrates_url`	Rapporte toujours `reachable: false`, `cgrates_url: "disabled"`
Signature de journal	`Initiating CGRateS session for ...`	`Intégration CGRateS désactivée, retour à l'autorisation par défaut` (avertissement)
Cas d'utilisation	Facturation en ligne en production	Test en laboratoire / intégration sans moteur de tarification

> En mode de contournement, OmniCHF alloue toujours des `chargingDataRef`, suit les sessions, incrémente les numéros de séquence, émet des déclencheurs et génère des CDR ; seule le contrôle de crédit est simulé. Cela le rend sûr d'exercer le flux complet SMF↔CHF sans déploiement CGRateS.

La configuration de l'intégration CGRateS (`cgrates_enabled`, `cgrates_url`, `cgrates_tenant`, `cgrates_timeout`) est documentée dans la Référence de configuration.

Cartographie des événements 5G vers CGRateS

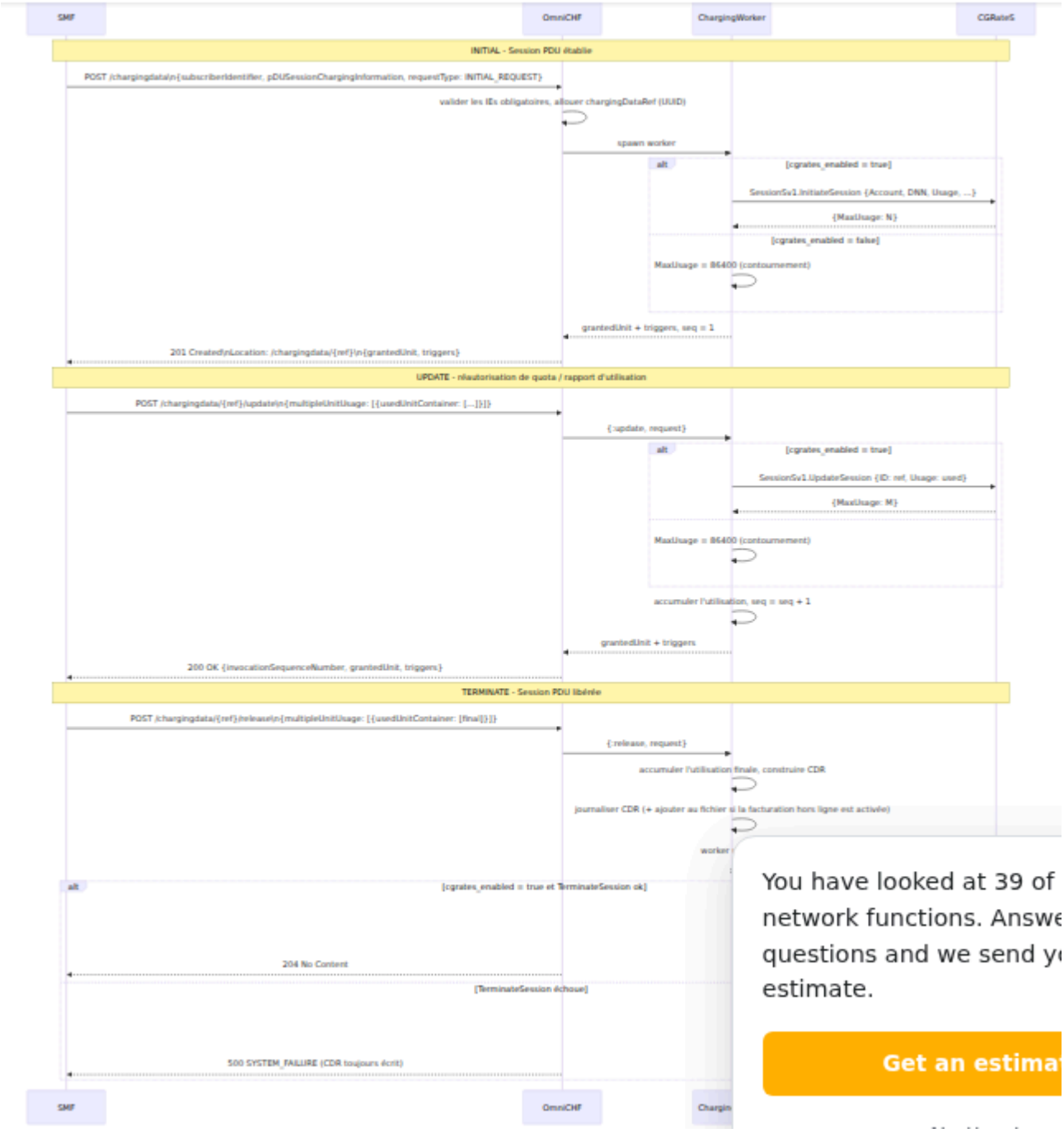
Chaque appel SessionS porte une carte `Event` construite à partir du ChargingDataRequest :

Champ d'événement CGRateS	Source	Re
`Account` / `Subject`	`subscriberIdentifier` (SUPI)	Revertit à `nfConsumerI` sinon `"unkno`
`Destination` / `DNN`	`pduSessionInformation.dnnId` (ou `.dnn`)	Nom du rése
`ToR`	dérivé de `pduType`	Toujours `*da` PDU.
`RequestType`	`requestType`	Mappé à `*pr`
`Category`	statique	`"data"`.
`Usage`	`usedUnitContainer`	Premier non- `totalVolume` `+ downlinkV`
`OriginID`	`chargingDataRef`	Unique par s
`OriginHost`	statique	`"OmniCHF"`.
`SUPI`	`subscriberIdentifier`	Champ d'ext
`S-NSSAI_SST` / `S-NSSAI_SD`	`pduSessionInformation.sNSSAI`	Valeurs par d `""`.
`5QI`	`qoSInformation.5qI` (ou `fiveQI`)	Valeur par dé
`RATType`	`pduSessionInformation.ratType`	Valeur par dé
`PDUSessionID`	`pduSessionInformation.pduSessionID`	Valeur par dé

Champ d'événement CGRateS	Source	Re
`PDUSessionType`	`pduSessionInformation.pduType`	Valeur par dé
`ChargingOperation`	opération	`Initial` / `Up`

Procédures clés

Session de facturation convergente (INITIAL → UPDATE → TERMINATE)



> Si le `TerminateSession` de CGRateS échoue lors de la libération, OmniCHF **finalise** et écrit toujours le CDR, de sorte que l'enregistrement hors ligne n'est pas perdu. Le consommateur reçoit alors `500` (cause `SYSTEM_FAILURE`) plutôt qu'un faux `204`, car la session côté CGRateS peut rester ouverte et le backend de facturation peut nécessiter une

> réconciliation. Lors d'une terminaison réussie, le consommateur reçoit `204 No Content`.

Gestion de quota (RSU / GSU et déclencheurs)

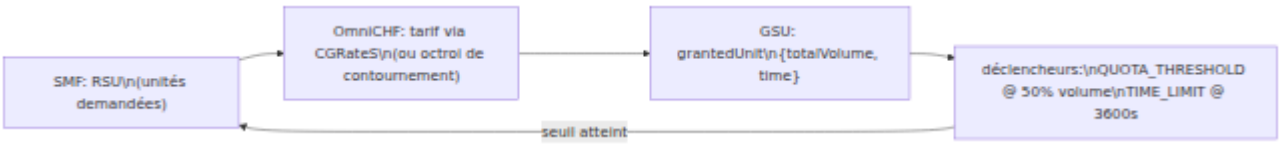
Selon le TS 32.291, le consommateur envoie des Unités de Service Demandées (RSU) et OmniCHF retourne des Unités de Service Accordées (GSU) plus des **déclencheurs** qui informent le consommateur quand revenir :



Le consommateur se réautorise (envoie une MISE À JOUR) lorsqu'il atteint le seuil de volume ou la limite de temps. OmniCHF peut également conduire proactivement la réautorisation ou la déconnexion via le point de terminaison `/notify` (`REAUTHORIZATION` / `ABORT_CHARGING`), qui poste à l'`notifyUri` fourni lors de la demande INITIAL.

Notification de facturation (REAUTHORIZATION / ABORT_CHARGING)

Selon la clause 6.1.6.2 du TS 32.291, OmniCHF pousse une `ChargingNotifyRequest` vers l'`notifyUri` stocké du consommateur. Le déclencheur est le point de terminaison `POST /nchf-convergedcharging/v3/notify` orienté opérateur, dont le corps porte le `chargingDataRef` cible et le `notificationType`. OmniCHF recherche la session, résout son `notifyUri`, et POSTe la notification au consommateur.



> La notification est faite au mieux vers la surface opérateur : un `notifyUri` rejeté ou inaccessible est enregistré mais ne change pas le résultat de `/notify` une fois la session et le `notifyUri` résolus. Une session sans `notifyUri` stocké produit une erreur interne `no_notify_uri`.

Champs CDR

Un CDR est construit lors de la libération, enregistré au niveau INFO (`CHF CDR: %{...}`), et, lorsque `offline_charging_enabled` est `true`, ajouté comme une ligne JSON dans `cdr_YYYYMMDD.json` :

Champ	Source
`record_type`	Statique `"5G_PDU_SESSION"`
`supi`	Contexte de session
`dnn`	Contexte de session / `pduSessionInformation`
`snssai`	`{sst, sd}`
`qos_5qi`	`qoSInformation.5qI` (valeur par défaut `9`)
`rat_type`	`pduSessionInformation.ratType` (valeur par défaut `"NR"`)
`pdu_session_id` / `pdu_session_type`	Contexte de session / demande
`volume_uplink` / `volume_downlink` / `volume_total`	Accumulé à partir de tous les `usedUnitContainer`s
`duration`	Temps rapporté, sinon différence d'horloge murale
`start_time` / `end_time`	Horodatages de création / libération de session
`charging_data_ref`	UUID de session
`charging_id`	ID de facturation 32 bits 3GPP (TS 32.251)
`recording_entity`	URL de base SBI OmniCHF
`rating_group`	Résolu à partir de la cartographie DNN

La sortie CDR hors ligne (`offline_charging_enabled`, `cdr_output_dir`) et la composition de l'ID de facturation (`node_id`) sont configurées dans la Référence de configuration.

Exécution de la facturation hors ligne

Pour capturer les CDR dans un fichier pour la médiation/facturation :

1. Définir `offline_charging_enabled: true` et pointer `cdr_output_dir` vers un répertoire écrivable.
2. Les CDR s'accumulent un objet JSON par ligne dans un fichier par jour nommé `cdr_YYYYMMDD.json` (la date est la date de libération UTC). Un nouveau fichier est ouvert chaque jour UTC ; il n'y a pas de rotation ou de conservation automatique au-delà de la séparation du nom de fichier quotidien, donc éliminez les anciens fichiers avec votre propre entretien.
3. Chaque session libérée ajoute exactement un CDR. Les sessions en cours conservent leur utilisation dans le worker/contexte jusqu'à la libération ; elles ne sont pas écrites dans le fichier avant cela.

Si `offline_charging_enabled` est `false`, les CDR sont toujours construits et enregistrés au niveau INFO comme `CHF CDR: %{...}` ; seule l'écriture dans le fichier est ignorée. Pour capturer les CDR sans sortie de fichier, expédiez ces lignes de journal avec votre pipeline de journalisation. Voir CDRs manquants dans les fichiers hors ligne.

Localisation et recherche de CDR

L'API de gestion expose `GET /api/oam/cdr` pour rechercher les fichiers CDR écrits. Elle lit chaque fichier `cdr_*.json` sous `cdr_output_dir` et filtre sur les paramètres de requête fournis (tous optionnels, combinés avec AND) :

Paramètre de requête	Correspond à	Format
`supi`	CDR `supi` exact	par exemple `imsi-999700000000001`
`dnn`	CDR `dnn` exact	par exemple `internet`
`date_from`	Date de fichier CDR `>=` cette date	`YYYY-MM-DD` (traits d'union optionnels)
`date_to`	Date de fichier CDR `<=` cette date	`YYYY-MM-DD` (traits d'union optionnels)

Les filtres de date sont appliqués à la **date du nom de fichier**, et non aux horodatages des enregistrements individuels, donc l'ensemble du fichier d'une journée est inclus ou exclu en tant qu'unité. Sans paramètres, le point de terminaison retourne chaque CDR sur disque. Parce qu'il lit des fichiers, `GET /api/oam/cdr` ne retourne des enregistrements que lorsque `offline_charging_enabled` était activé au moment de la libération.

Forcer la vidange d'un CDR (OAM)

En fonctionnement normal, un CDR n'est produit que lorsqu'une session est libérée. `POST /api/oam/cdr/flush` permet à un opérateur d'émettre un CDR plus tôt pour toute session active (voir Stockage de session et visibilité de gestion), par exemple pour clôturer des sessions que le consommateur n'a jamais libérées :

Corps de la demande	Effet
`{"charging_data_ref": "<ref>"}`	Construire et écrire un CDR pour cette session, puis le finaliser. `404` si la référence n'est pas trouvée.
`{}` ou `{"charging_data_ref": "all"}`	Vider chaque session actuellement suivie.

La vidange construit le CDR à partir de l'utilisation accumulée de la session plus le dernier `ChargingDataRequest` et réutilise le même écrivain CDR que lors de la libération ; elle ne fait **pas** appel à CGRateS `TerminateSession`. La réponse rapporte `write_result` et `offline_charging_enabled` par session. Si la facturation hors ligne est désactivée, le CDR est construit et la session finalisée, mais rien n'est écrit dans le fichier (`write_result: "ok"` sans sortie de fichier). Chaque vidange incrémente `omni_chf_charging_releases_total{result="oam_flush"}`.

← Vue d'ensemble

Dépannage OmniCHF

400 Mauvaise demande sur toute demande de chargement

Symptômes : Le consommateur reçoit `400` avec la cause `MANDATORY_IE_MISSING`.

Causes possibles :

- Le corps de la requête manque `nfConsumerIdentification`, `invocationTimeStamp` ou `invocationSequenceNumber`.

Résolution :

1. Confirmer que le consommateur inclut les trois IEs obligatoires à chaque opération (Créer, Mettre à jour, Libérer).
2. Vérifier le champ `detail` de la réponse ProblemDetails ; il liste les noms des IEs manquants.

404 sur Mise à jour ou Libération

Symptômes : Le consommateur reçoit `404` avec la cause `CHARGING_DATA_NOT_FOUND`.

Causes possibles :

- OmniCHF a redémarré entre la création et l'opération ultérieure ; l'état de session est conservé en mémoire, donc la session doit être recréée après un redémarrage.
- Le `chargingDataRef` dans le chemin est incorrect.
- La session a déjà été libérée (son worker a arrêté).

Résolution :

1. Grep les journaux pour `CHF Update: unknown ref=` / `CHF Release: unknown ref=`.
2. Confirmer qu'OmniCHF n'a pas redémarré depuis la création de la session.
3. Confirmer que le consommateur renvoie le `chargingDataRef` de l'en-tête `Location` de la création.

500 sur Création avec CGRateS activé

Symptômes : Le consommateur reçoit `500` avec la cause `CHARGING_FAILED`.

Causes possibles :

- `InitiateSession` de CGRateS a échoué : moteur inaccessible, mauvais locataire ou une erreur côté CGRateS.

Résolution :

1. Vérifier que `cgrates_url` pointe vers un point de terminaison CGRateS `/jsonrpc` accessible.
2. Vérifier que `cgrates_tenant` correspond au locataire CGRateS.
3. Vérifier le compteur `omni_chf_cgrates_health` (`1` = opérationnel).
4. Examiner les journaux pour `CGRateS InitiateSession failed ...` ; la raison est l'une de `{:cgrates_error, msg}`, `{:http_error, status}`, ou `{:http_error, reason}`.
5. Confirmer que le service `SessionSv1` est activé dans CGRateS.

Libération retourne 500

Symptômes : Le consommateur reçoit `500` avec la cause `SYSTEM_FAILURE` et le titre "Échec de la terminaison de chargement" sur `POST /chargingdata/{ref}/release`.

Causes possibles :

- `TerminateSession` de CGRateS a échoué lors de la libération (moteur inaccessible, mauvais locataire ou une erreur côté CGRateS).

Impact :

- Le CDR est toujours construit et écrit, donc l'enregistrement hors ligne n'est pas perdu.
- La session côté CGRateS peut rester ouverte, donc le backend de facturation peut être incohérent et nécessite une réconciliation.

Résolution :

1. Grep les journaux pour `CGRateS TerminateSession failed` sur le `ref` concerné.
2. Confirmer l'accessibilité de CGRateS avec le compteur `omni_chf_cgrates_health` (`1` = opérationnel).
3. Réconcilier toute session CGRateS laissée ouverte pour ce `ref` par rapport au CDR écrit.
4. Corréler avec `omni_chf_charging_releases_total{result="failure"}` pour voir le taux d'échec de terminaison.

Le contrôle de santé de CGRateS échoue alors que le démon est en cours d'exécution

Symptômes : `omni_chf_cgrates_health` lit `0` mais CGRateS est en cours d'exécution.

Causes possibles :

- CGRateS est lent : le contrôle de santé limite son délai d'attente à `min(cgrates_timeout, 3000)` ms.
- Mismatch de schéma (HTTP vs HTTPS) sur `cgrates_url`.

Résolution :

1. Confirmer que le schéma et le port de `cgrates_url` sont corrects.
2. Si CGRateS est légitimement lent, enquêter sur la charge de CGRateS plutôt que d'augmenter la limite.
3. Rechercher des avertissements `CGRateS unreachable at <url> ...`.

Latence élevée de CGRateS

Symptômes : Opérations de chargement lentes ; `omni_chf_cgrates_request_duration_ms` élevé.

Résolution :

1. Inspecter l'histogramme p95/p99 par `operation`.
2. Élargir CGRateS ou le chemin réseau entre OmniCHF et CGRateS.
3. Envisager de réduire `cgrates_timeout` afin que les appels lents échouent rapidement plutôt que de bloquer la session.

CDR manquants dans les fichiers hors ligne

Symptômes : Des lignes `CHF CDR:` apparaissent dans le journal, mais aucun fichier `cdr_YYYYMMDD.json` n'est écrit.

Causes possibles :

- `offline_charging_enabled` est `false` (par défaut) : les CDR sont seulement enregistrés.
- `cdr_output_dir` n'est pas accessible en écriture.

Résolution :

1. Définir `offline_charging_enabled: true`.
2. S'assurer que `cdr_output_dir` existe et est accessible en écriture ; surveiller les erreurs `[CDR] Failed to write CDR to file`.

3. Pour capturer les CDR sans sortie de fichier, expédier les lignes de journal `CHF CDR:` avec votre pipeline de journalisation.

Le nombre de sessions actives ne diminue pas

Symptômes : `omni_chf_sessions_active_count` reste élevé.

Causes possibles :

- Le consommateur n'envoie jamais de libération (les sessions ne se nettoient que lors de la libération ou d'un crash du worker).
- Les libérations retournent `404` (voir ci-dessus) donc le consommateur les considère comme terminées alors qu'OmniCHF peut ne pas l'être.

Résolution :

1. Corréler le nombre actif avec les taux de création par rapport aux taux de libération.
2. Enquêter sur le `404` lors de la libération, ce qui indique généralement un redémarrage antérieur.

L'enregistrement NRF n'est pas maintenu

Symptômes : `omni_chf_nrf_registration_status` lit `0`.

Résolution :

1. Vérifier que `nrf_uri` est accessible depuis l'`sbi_addr` d'OmniCHF.
2. Confirmer que `mcc` / `mnc` correspondent à la configuration PLMN NRF.
3. Confirmer que `sbi_addr` / `sbi_port` dans le profil annoncé sont accessibles par le NRF.
4. Examiner les journaux de démarrage pour les erreurs d'enregistrement NRF.