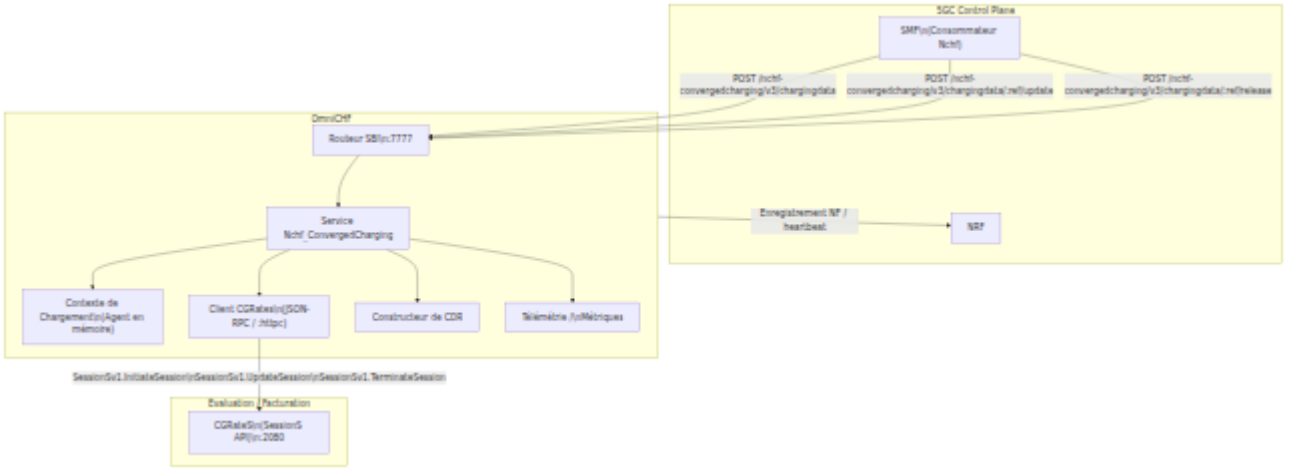
Guide des Opérations OmniCHF

Table des Matières

Aperçu des Composants

OmniCHF implémente la fonction de réseau de Chargement (CHF) définie dans 3GPP TS 32.291. Le CHF fournit un chargement en ligne et hors ligne convergent pour les sessions PDU 5G via le service Nchf_ConvergedCharging. Il traduit les demandes de chargement 5G en appels JSON-RPC SessionS CGRateS pour l'autorisation de crédit et la gestion de session, et génère des enregistrements de détails d'appel (CDRs) lors de la libération de session.



Cycle de Vie de la Session de Chargement

Chaque session PDU correspond à une session de chargement, suivie par un `chargingDataRef` (UUID). L'état de la session est conservé dans un Agent en mémoire et n'est pas persistant. Un redémarrage perd tout l'état de session actif.

État	Déclencheur	Action de stockage
Créé	POST /chargingdata	Contexte créé, CGRateS InitiateSession appelé
Mis à jour	POST /chargingdata/:ref/update	Contexte mis à jour (utilisation accumulée, séquence incrémentée)
Libéré	POST /chargingdata/:ref/release	CDR construit et enregistré, CGRateS TerminateSession appelé, contexte supprimé

Rôle 3GPP et Références de Spécifications

Élément	Référence
Définition NF CHF	3GPP TS 23.501 Section 6.2.16
Service Nchf_ConvergedCharging	3GPP TS 32.291
Procédure de Création de Données de Chargement	3GPP TS 32.291 Section 6.1.3.2.1
Procédure de Mise à Jour des Données de Chargement	3GPP TS 32.291 Section 6.1.3.2.2
Procédure de Libération des Données de Chargement	3GPP TS 32.291 Section 6.1.3.2.3
Modèle de données ChargingDataRequest / Response	3GPP TS 32.291 Section 6.1.6
Format CDR pour les sessions PDU 5G	3GPP TS 32.290
Cadre commun SBI	3GPP TS 29.500
Enregistrement NF avec NRF	3GPP TS 29.510

Points de Terminaison SBI

Chemin de base : `/nchf-convergedcharging/v3`

Méthode	Chemin	Description
`POST`	`/chargingdata`	Créer une sessior de chargement (demande initiale). Alloue ur `chargingDataRef` démarre une session CGRateS, et retourne les unités accordées.
`POST`	`/chargingdata/{chargingDataRef}/update`	Mettre à jour une session de chargement (demande intermédiaire). Signale l'utilisation actuelle et demande un crédit supplémentaire.
`POST`	`/chargingdata/{chargingDataRef}/release`	Libérer une session de chargement (demande finale) Signale l'utilisation finale, génère un CDR, termine la sessior CGRateS.

ChargingDataRequest — Champs Clés

Champ	Type	Utilisé dans	Description
`subscriberIdentifier`	string	Créer, Mettre à jour, Libérer	SUPI (par exemple, `imsi-999700000000001`). Utilisé comme identifiant de compte CGRateS.
`nfConsumerIdentification`	object	Créer	Infos sur le consommateur NF. Source de secours pour SUPI si `subscriberIdentifie` est absent.
`pDUSessionChargingInformation`	object	Créer, Mettre à jour, Libérer	Détails de la session PDU : DNN, S-NSSAI, type RAT, QoS, ID et type de session PDU.
`multipleUnitUsage`	array	Mettre à jour, Libérer	Conteneurs d'utilisation signalés. Le `usedUnitContainer` du premier élément est utilisé pour l'extraction de volume et de durée.
`requestType`	string	Tous	`INITIAL_REQUEST`, `UPDATE_REQUEST`, ou `TERMINATION_REQUEST`

ChargingDataResponse — Champs Clés

Champ	Type	Présent	Description
`invocationSequenceNumber`	integer	Créer, Mettre à jour	Numéro de séquence pour cette réponse. Codé en dur à `1` lors de la création (voir CHF-M1). Incrémenté à chaque mise à jour.
`invocationResult`	object	Créer, Mettre à jour	Toujours `{"resultCode": "SUCCESS"}` sur le chemin heureux.
`sessionId`	string	Créer, Mettre à jour	Le `chargingDataRef` (UUID) alloué pour cette session.
`multipleUnitInformation`	array	Créer, Mettre à jour	Unités accordées. Contient une entrée avec `resultCode`, `grantedUnit` (totalVolume, temps), et `ratingGroup` (codé en dur à `1`, voir CHF-L2).

Référence de Configuration

Tous les paramètres sont définis via l'environnement de l'application (typiquement `config/runtime.exs`).

```
config :omnichf,
  sbi_scheme:       "http",
  sbi_addr:         "127.0.0.14",
  sbi_port:         7777,
  nrf_uri:          "http://127.0.0.10:7777",
  mcc:              "999",
  mnc:              "70",
  heartbeat_interval: 10_000,
  cgrates_enabled:  false,
  cgrates_url:      "http://localhost:2080/jsonrpc",
  cgrates_tenant:   "cgrates.org",
  cgrates_timeout:  5000
```

Tableau des Paramètres

Paramètre	Par Défaut	Type	
`sbi_scheme`	`"http"`	string	Sche pour
`sbi_addr`	`"127.0.0.14"`	string	Adre serv lie.
`sbi_port`	`7777`	integer	Port serv écou
`nrf_uri`	`"http://127.0.0.10:7777"`	string	URI utilis l'enr le he
`mcc`	`"999"`	string	Cod Utili sour le nc serv
`mnc`	`"70"`	string	Cod Utili sour le nc serv
`heartbeat_interval`	`10_000`	integer (ms)	Inte dem hear

Paramètre	Par Défaut	Type	

`cgrates_enabled`	`false`	boolean	Inte
			pou
			CGR
			`fal`
			CGR
			et u
			de 8
			reto
			`tru`
			lors
			CGR
			disp

`cgrates_url`	`"http://localhost:2080/jsonrpc"`	string	URL
			term
			pou
			CGR
			uniq
			`cgr`
			`tru`

`cgrates_tenant`	`"cgrates.org"`	string	Nom CGR tous Sess char corr loca dans
`cgrates_timeout`	`5000`	integer (ms)	Déla la re les a CGR de v

Paramètre	Par Défaut	Type	

			sant
			`min`
			`3000`
			bloq
			term

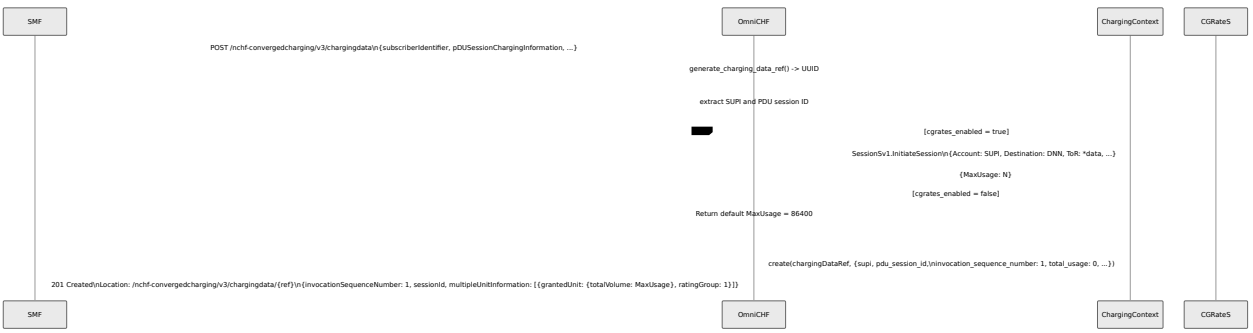
Notes d'Intégration CGRateS

Lorsque `cgrates_enabled` est `false`, OmniCHF fonctionne en **mode de contournement** : toutes les demandes de chargement sont acceptées et accordées 86 400 unités (temps ou volume) sans aucune évaluation ou autorisation. Cela convient pour les tests en laboratoire et d'intégration lorsque CGRateS n'est pas disponible.

La communication CGRateS utilise le client HTTP intégré `:httpc` d'Erlang (voir limitation CHF-M5). Ce client ne prend pas en charge le regroupement de connexions. Sous une charge élevée, chaque demande CGRateS ouvre et ferme une nouvelle connexion HTTP, ce qui peut devenir un goulot d'étranglement.

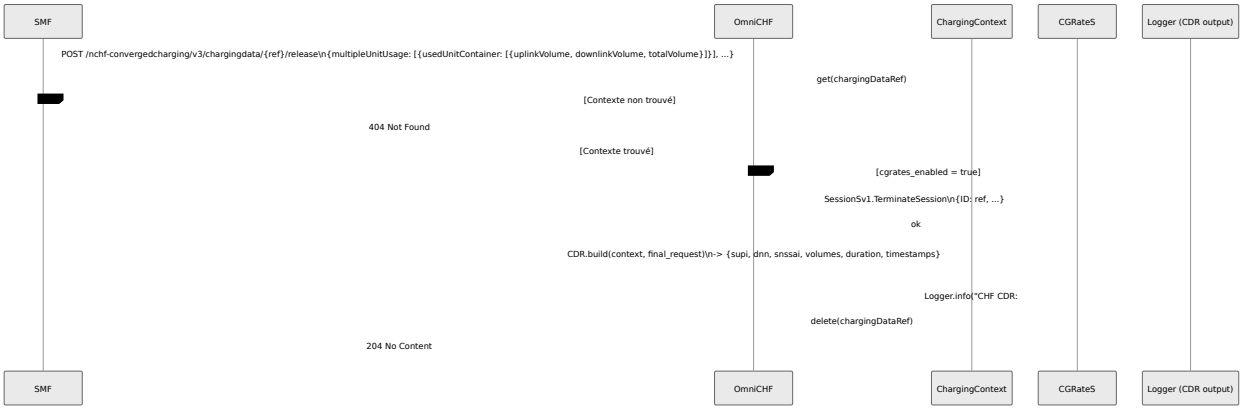
Procédures Clés

Création de Session de Chargement (Initiale)



Mise à Jour de Session de Chargement (Intermédiaire)

Libération de Session de Chargement (Finale)



Mappage des Événements CGRateS

OmniCHF mappe les champs de chargement 5G aux champs d'événements SessionS CGRateS comme suit :

Champ CGRateS	Source	
`Account`	`subscriberIdentifier` (SUPI)	Fallback s `nfConsum`
`Subject`	`subscriberIdentifier` (SUPI)	Identique
`Destination`	`pduSessionInformation.dnnId` ou `.dnn`	Nom du R
`ToR`	`pduSessionInformation.pduType`	Toujours PDU
`RequestType`	`requestType`	Toujours r
`Usage`	`usedUnitContainer.totalVolume` ou somme de uplink+downlink ou `time`	La premiè l'emporte
`OriginID`	`chargingDataRef`	UUID unic
`OriginHost`	statique `"OmniCHF"`	
`SUPI`	`subscriberIdentifier`	Champ d'
`DNN`	`pduSessionInformation.dnnId`	Champ d'
`S-NSSAI_SST`	`pduSessionInformation.sNSSAI.sst`	Champ d'
`S-NSSAI_SD`	`pduSessionInformation.sNSSAI.sd`	Champ d'
`5QI`	`pduSessionInformation.qoSInformation.5qI`	Champ d'
`RATType`	`pduSessionInformation.ratType`	Par défau
`PDUSessionID`	`pduSessionInformation.pduSessionID`	Champ d'
`PDUSessionType`	`pduSessionInformation.pduType`	Par défau

Champs CDR

Les CDR sont construits lors de la libération de session et émis dans le journal de l'application au niveau INFO. La carte CDR contient :

Champ	Source
`record_type`	Statique : `"5G_PDU_SESSION"`
`supi`	Contexte de chargement
`dnn`	`pduSessionInformation.dnnId` ou `.dnn`
`snssai`	`{sst, sd}` de `pduSessionInformation.sNSSAI`
`qos_5qi`	`pduSessionInformation.qoSInformation.5qI`
`rat_type`	`pduSessionInformation.ratType`
`pdu_session_id`	Contexte de chargement
`pdu_session_type`	`pduSessionInformation.pduType`
`volume_uplink`	`usedUnitContainer.uplinkVolume`
`volume_downlink`	`usedUnitContainer.downlinkVolume`
`volume_total`	`usedUnitContainer.totalVolume` (ou uplink+downlink)
`duration`	`usedUnitContainer.time` (ou différence d'horloge si zéro)
`start_time`	Horodatage `created_at` de la session
`end_time`	Horloge au moment de la libération
`charging_data_ref`	UUID de la session

Observabilité

Événements de Télémétrie

Événement	Mesures	Tags	Description
`[:omnichf, :charging, :initial]`	`count`	`supi`	Déclenché à chaque demande de création
`[:omnichf, :charging, :update]`	`count`	`ref`	Déclenché à chaque demande de mise à jour
`[:omnichf, :charging, :release]`	`count`	`ref`	Déclenché à chaque demande de libération
`[:omnichf, :charging, :creates]`	`count`	`result` (`success`/`failure`)	Résultat de l'opération de création
`[:omnichf, :charging, :updates]`	`count`	`result`	Résultat de l'opération de mise à jour
`[:omnichf, :charging, :releases]`	`count`	`result`	Résultat de l'opération de libération
`[:omnichf, :cgrates, :request]`	`count`, `duration_ms`	`operation`, `result`	Par appel JSON-RPC CGRateS
`[:omnichf, :cgrates,`	`status` (1/0)	—	Santé de la connectivité

Événement	Mesures	Tags	Description
`:health]`			CGRateS
`[:omnichf, :sessions, :active]`	`count`	—	Mesure : sessions de chargement actives
`[:omni5g, :nrf, :registration]`	`status` (1/0)	`nf_type`	État d'enregistrement NRF

Métriques Prometheus

Métriques de Chargement

Métrique	Type	Tags	Description
`omni_chf.charging.initial.count`	compteur	`supi`	Créations de sessions de chargement
`omni_chf.charging.update.count`	compteur	`ref`	Mises à jour de sessions de chargement
`omni_chf.charging.release.count`	compteur	`ref`	Libérations de sessions de chargement
`omni_chf.charging.creates.total`	compteur	`result`	Total des créations de sessions de chargement
`omni_chf.charging.updates.total`	compteur	`result`	Total des mises à jour de sessions de chargement
`omni_chf.charging.releases.total`	compteur	`result`	Total des libérations de sessions de chargement
`omni_chf.sessions.active.count`	jauge	--	Nombre de sessions de chargement actives

Métriques CGRateS

Métrique	Type	Tags	Des
`omni_chf.cgrates.calls.count`	compteur	`method`, `result`	App JSO CGR
`omni_chf.cgrates.latency.milliseconds`	jauge	--	Late app CGR
`omni_chf.cgrates.health`	jauge	--	San con CGR (1= 0=c
`omni_chf.cgrates.requests.total`	compteur	`operation`, `result`	Tota den JSO CGR
`omni_chf.cgrates.request.duration_ms`	distribution	`operation`	Dur den CGR ms 5, 1 50, 250 100

Métriques NRF

Métrique	Type	Tags	Description
`omni_chf.nrf.registration.status`	jauge	`nf_type`	État d'enregistrement NRF (1=registré, 0=non)

Métriques BEAM VM

Métrique	Type	Description
`beam.memory.total`	jauge	Mémoire totale BEAM en octets
`beam.memory.processes`	jauge	Mémoire utilisée par les processus Erlang
`beam.memory.processes_used`	jauge	Mémoire réellement utilisée par les processus
`beam.memory.system`	jauge	Mémoire système
`beam.memory.atom`	jauge	Mémoire totale des atomes
`beam.memory.atom_used`	jauge	Mémoire des atomes 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 en secondes

Modèles de Journal

Niveau	Modèle	Signification
`info`	`CHF Create: ref=<UUID> supi=<SUPI> pdu_session=<N>`	Création réussie initiée
`info`	`CHF Update: ref=<UUID>`	Demande de mise à jour reçue
`info`	`CHF Release: ref=<UUID>`	Demande de libération reçue
`info`	`CHF CDR: %{...}`	CDR émis lors de la libération
`info`	`Initiating CGRateS session for <ref>, account: <SUPI>`	Session CGRateS InitiateSession envoyée
`info`	`CGRateS authorized <N> units for session <ref>`	Crédit accordé
`info`	`CGRateS session <ref> terminated successfully`	CGRateS TerminateSession OK
`warning`	`CGRateS integration disabled, returning default authorization`	Mode de contournement actif
`warning`	`CHF Update: unknown ref=<UUID>`	Mise à jour pour une session inexistante
`warning`	`CHF Release: unknown ref=<UUID>`	Libération pour une session inexistante
`error`	`CHF Create failed: <reason>`	Échec de l'opération de création

Niveau	Modèle	Signification
`error`	`CHF Update failed: <reason>`	Échec de l'opération de mise à jour
`error`	`CGRateS InitiateSession failed for <ref>: <reason>`	Erreur CGRateS lors de la création
`error`	`CGRateS HTTP error <status>: <body>`	Non-200 de CGRateS
`error`	`CGRateS HTTP request failed: <reason>`	Erreur réseau vers CGRateS

Limitations Connues

ID	Gravité	Description
CHF-M1	Moyenne	`invocationSequenceNumber` est codé en dur à `1` dans la réponse de Création (Initiale). Selon TS 32.291, le numéro de séquence doit commencer à 1 et s'incrémenter lors des réponses suivantes, ce qu'il fait lors de la Mise à jour. Le problème est que si un consommateur envoie une Création avec un `invocationSequenceNumber` dans la demande, cette valeur n'est pas inspectée ou validée.
CHF-M3	Moyenne	`invocationTimeStamp` est absent de `ChargingDataResponse`. Selon TS 32.291, ce champ est obligatoire dans le corps de la réponse. Les consommateurs stricts qui nécessitent ce champ recevront une réponse incomplète.
CHF-M5	Moyenne	Le client CGRateS utilise le client HTTP `:httpc` d'Erlang plutôt que Finch. `:httpc` ne prend pas en charge le regroupement de connexions ; chaque appel JSON-RPC ouvre et ferme une connexion TCP. Sous charge (de nombreuses sessions de chargement simultanées), la latence des appels CGRateS augmentera et le surcoût de mise en place de la connexion devient significatif. Surveillez `omni_chf.cgrates.request.duration_ms`.
CHF-L1	Faible	Aucun champ `triggers` n'est inclus dans `ChargingDataResponse`. Selon TS 32.291, les déclencheurs peuvent instruire le SMF d'envoyer une mise à jour intermédiaire sur des événements spécifiques (seuil de volume, seuil de temps, changement de QoS). Sans déclencheurs, le SMF utilise sa propre politique locale pour déterminer quand envoyer des mises à jour.

ID	Gravité	Description
CHF-L2	Faible	`ratingGroup` dans `multipleUnitInformation` est codé en dur à `1`. Les déploiements réels ont généralement plusieurs groupes de tarification par session PDU (un par flux de données de service). Toute utilisation est attribuée au groupe de tarification 1, indépendamment des valeurs `ratingGroup` dans l'utilisation `multipleUnitUsage` de la demande.
CHF-L3	Faible	Aucun `chargingId` au format 3GPP n'est généré. Selon TS 32.290, les enregistrements de chargement doivent porter un chargingId qui correspond à l'ID de session PDU attribué par le SMF. Le UUID `charging_data_ref` est utilisé à la place, ce qui peut causer des problèmes de corrélation dans les systèmes de facturation en aval qui s'attendent au format chargingId 3GPP.
CHF-L4	Faible	L'enregistrement CDR manque des champs `chargingID` et `recordingEntity` requis par TS 32.290. Les systèmes de médiation ou de facturation en aval s'attendant à ces champs devront tolérer leur absence ou être configurés pour les traiter comme optionnels.
CHF-L5	Faible	Le chargement hors ligne et la sortie de fichier CDR ne sont pas implémentés. Les CDR sont émis uniquement dans le journal de l'application via `Logger.info`. Il n'y a pas de sortie de fichier CDR, pas d'encodage ASN.1, et pas de transfert vers une passerelle de domaine de facturation. Pour le chargement hors ligne en production, un expéditeur de journal (par exemple, Fluentd, Vector) doit collecter les lignes de journal CDR CHF et les transformer pour le système de facturation.

Dépannage

404 sur Mise à Jour ou Libération

Le `chargingDataRef` ne correspond à aucune session active en mémoire. Causes :

1. OmniCHF redémarré entre Création et Mise à jour/Libération — tout l'état de session est en mémoire et est perdu lors du redémarrage.
2. Le SMF a envoyé le mauvais `chargingDataRef` dans le chemin.
3. Une Libération a été précédemment envoyée pour cette session, ce qui a supprimé le contexte.

Vérifiez les journaux pour `CHF Update: unknown ref=` ou `CHF Release: unknown ref=` pour confirmer.

500 sur Création avec CGRateS activé

L'appel CGRateS a échoué. Vérifiez :

1. Est-ce que `cgrates_url` pointe vers une instance CGRateS accessible ?
2. Est-ce que `cgrates_tenant` est correct ? Des noms de locataires non correspondants provoquent une réponse d'erreur de CGRateS.
3. Vérifiez la métrique `omni_chf.cgrates.health` (1=up, 0=down).
4. Consultez les journaux pour `CGRateS InitiateSession failed for <ref>: <reason>` — la raison sera l'une de : `{:cgrates_error, message}`, `{:http_error, status}`, ou `{:http_error, reason}`.
5. Vérifiez que le service `SessionSv1` de CGRateS est activé dans la configuration de CGRateS.

Vérification de santé CGRateS échouant mais le démon est en cours d'exécution

La vérification de santé (via `SessionSv1.GetActiveSessions`) utilise un délai d'attente limité à 3 secondes. Si CGRateS est lent à répondre, la vérification de santé peut échouer alors que le service est techniquement disponible. Vérifiez

`cgrates_timeout` — la limite est `min(cgrates_timeout, 3000)`. Confirmez également que `cgrates_url` utilise HTTP (pas HTTPS) sauf si TLS est configuré.

CDRs n'apparaissant pas ou incomplets

Les CDRs sont écrits dans le journal de l'application au niveau INFO (voir limitation CHF-L5). Pour capturer les CDRs :

1. Assurez-vous que le niveau de journal de l'application est réglé sur `info` ou inférieur.
2. Filtrer les lignes de journal contenant `"CHF CDR:"` pour un post-traitement.
3. Notez que les CDRs manquent `chargingID`, `recordingEntity` (CHF-L4), et auront `ratingGroup: 1` pour tous les flux de données de service (CHF-L2).

Latence élevée des appels CGRateS

Parce que CGRateS utilise `:httpc` sans regroupement de connexions (CHF-M5), la latence augmente sous charge. Pour diagnostiquer :

1. Vérifiez l'histogramme `omni_chf.cgrates.request.duration_ms` pour la latence p99.
2. Si la latence est élevée sous charge concurrente, réduisez le nombre de sessions de chargement simultanées ou envisagez de déployer OmniCHF derrière un équilibreur de charge avec plusieurs instances.
3. En tant que solution de contournement, définissez `cgrates_timeout` à une valeur inférieure au temps de réponse CGRateS le plus défavorable attendu pour empêcher les appels CGRateS lents de bloquer le pool de processus Elixir.

Nombre de sessions actives ne diminuant pas après les libérations

Si `omni_chf.sessions.active.count` reste élevé après que les sessions auraient dû être libérées :

1. Vérifiez les réponses 404 sur les appels de Libération — si le SMF reçoit 404, il peut ne pas réessayer et le SMF considère la session libérée tandis

qu'OmniCHF peut encore avoir le contexte.

2. Dans le cas inverse, si OmniCHF a redémarré et perdu l'état de session, les contextes sont disparus mais le SMF peut encore envoyer des demandes de Mise à jour/Libération qui aboutissent à 404. C'est un comportement attendu.

Enregistrement NRF non maintenu

Vérifiez la métrique `omni_chf.nrf.registration.status`. Si elle indique 0 :

1. Vérifiez que `nrf_uri` est correct et que le NRF est accessible depuis l'adresse `sbi_addr` d'OmniCHF.
2. Vérifiez que `mcc` et `mnc` correspondent à la configuration PLMN du NRF.
3. Consultez les journaux de l'application au démarrage pour les erreurs d'enregistrement NRF.