← Visão Geral

Referência de Configuração do OmniNSSF

A configuração do operador é lida a partir da chave de ambiente da aplicação `:omnissf`. A configuração em tempo de execução é fornecida via `config/runtime.exs`, que mapeia variáveis de ambiente do sistema operacional para chaves de configuração na inicialização. A topologia do slice (lista NSI, NSSAI permitido, mapeamento de conjunto AMF, NSSAI configurado) é fornecida através da configuração estrutural da aplicação (veja Configuração da Topologia do Slice) e ajustada em tempo de execução através da API OAM.

Configuração em Tempo de Execução (`config/runtime.exs`)

```
config :omnissf,
  sbi_scheme: System.get_env("SBI_SCHEME", "http"),
  sbi_addr: System.get_env("SBI_ADDR", "127.0.0.14"),
  sbi_port: String.to_integer(System.get_env("SBI_PORT", "7777")),
  nrf_uri: System.get_env("NRF_URI", "http://127.0.0.1:7777"),
  mcc: System.get_env("MCC", "999"),
  mnc: System.get_env("MNC", "70"),
  prometheus_metrics_port:
String.to_integer(System.get_env("PROMETHEUS_PORT", "9568")),
  heartbeat_interval:
String.to_integer(System.get_env("HEARTBEAT_INTERVAL", "10000"))

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

Parâmetro	Tipo	Requerido	Padrão
`sbi_scheme`	string	Não	`"http"`
`sbi_addr`	string	Não	`"127.0.0.14"`
`sbi_port`	inteiro	Não	`7777`
`nrf_uri`	string	Não	`"http://127.0.0.1:777`
`mcc`	string	Não	`"999"`

Parâmetro	Tipo	Requerido	Padrão
`mnc`	string	Não	`"70"`
`prometheus_metrics_port`	inteiro	Não	`9568`

Parâmetro	Tipo	Requerido	Padrão
`heartbeat_interval`	inteiro (ms)	Não	`10000`

> Nota: o padrão compilado para `nrf_uri` na configuração em tempo de execução é `http://127.0.0.1:7777`, enquanto o padrão do acessador interno é `http://127.0.0.10:7777`. Quando `runtime.exs` é aplicado, o padrão efetivo é `http://127.0.0.1:7777`. Defina `NRF_URI` explicitamente para evitar ambiguidade.

Comportamento e Configurações de Notificação

As seguintes chaves de configuração da aplicação `:omnissf` ajustam o comportamento de seleção de slice e notificação. Nenhuma delas possui uma variável de ambiente `runtime.exs`, portanto, são definidas através da configuração da aplicação em vez de uma variável de ambiente do sistema operacional.

Parâmetro	Tipo	Requerido	Padrão	
`dynamic_nrf_discovery`	Booleano	Não	`false`	Qu NS NR sli de pa via Nn em ap `ns` est ca de sli co rai NS e r da co qu de de ina nã
`strict_availability`	Booleano	Não	`false`	Mo ap dis Qu (pe TA de dis

Parâmetro	Tipo	Requerido	Padrão	
				ex for pe co Qu (fa TA NS é r
`nssf_notification_max_attempts`	Inteiro	Não	`3`	Má ter en no mu dis an de fal reg
`nssf_notification_backoff_ms`	Lista de inteiros	Não	`[200, 1000]`	Cr ba ter mi pa de dis O é a da ter se da últ reu

Parâmetro	Tipo	Requerido	Padrão	
				qu ter ad

Logger

Parâmetro	Tipo	Requerido	Padrão	A
`format`	tupla de formatador	Não	`{OmniLogger.JsonFormatter, :format}`	-
`metadata`	átomo / lista	Não	`:all`	-

O nível de log ativo pode ser alterado em tempo de execução sem reinicialização através da API OAM (`PATCH /oam/config/{id}` com `log_level`, ou `POST /oam/log_level`). Os níveis válidos são `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, e `debug`.

Configuração da Topologia do Slice

A topologia do slice é carregada no armazenamento `SliceConfig` na inicialização a partir das seguintes chaves de configuração da aplicação `:omnissf`. Cada uma tem um padrão embutido que modela um único slice eMBB (SST 1, SD `0x000001`). Essas chaves são estruturais (mapas e listas) e são definidas através da configuração da aplicação em vez de uma única variável de ambiente; elas também podem ser alteradas em tempo de execução através da API OAM (veja Gerenciamento de Slice em Tempo de Execução).

```
config :omnissf,
  nsi_list: [
    %{s_nssai: %{sst: 1, sd: "0x000001"}, nrf_uri:
"http://127.0.0.10:7777", nsi_id: "1"}
  ],
  allowed_nssai: %{
    "999-70" => [%{sst: 1, sd: "0x000001"}]
  },
  amf_set_mapping: %{
    "1-0x000001" => ["1"]
  },
  configured_nssai: %{}
```

`nsi_list` - Entradas de Instância de Slice de Rede

Cada entrada vincula um S-NSSAI ao NRF que serve a descoberta de NF específica do slice para aquele slice. O NSSF compara o S-NSSAI solicitado com esta lista; uma correspondência fornece as informações NSI retornadas ao AMF, e nenhuma correspondência resulta em `403`.

Parâmetro	Tipo	Requerido	Padrão	Var (Ambie
`nsi_list`	lista de mapas	Não	entrada eMBB única (abaixo)	-
`s_nssai.sst`	inteiro	Sim	`1`	-
`s_nssai.sd`	string	Não	`"0x000001"`	-
`nrf_uri`	string	Sim	`"http://127.0.0.10:7777"`	-
`nsi_id`	string	Não	`"1"`	-

A correspondência de SD é sensível a curingas: um SD `nil` em qualquer uma das entradas configuradas ou na solicitação corresponde a qualquer SD, e SDs não nulos são comparados sem distinção de maiúsculas e minúsculas.

`allowed_nssai` - NSSAI Permitido por PLMN

Vincula uma chave PLMN `"mcc-mnc"` à lista de S-NSSAIs permitidos para UEs desse PLMN. Quando um PLMN de consulta não possui entrada, o NSSF recorre

ao PLMN doméstico (`mcc-mnc` da configuração em tempo de execução). A lista resultante é retornada como `allowedNssaiList` (tipo de acesso `3GPP_ACCESS`) e, quando um TAI é fornecido, é filtrada para os S-NSSAIs reportados disponíveis naquele TAI.

Parâmetro	Tipo	Requerido	Padrão	Var de Ambiente	Desc
`allowed_nssai`	mapa	Não	`{"999-70" => [...]}`	-	Tabela NSSAI permit indexa por PLI
chave do mapa	string `"mcc-mnc"`	Sim	`"999-70"`	-	Identifi do PLM qual o conjun permit aplica
`sst`	inteiro	Sim	`1`	-	SST de S-NSSA permit
`sd`	string	Não	`"0x000001"`	-	SD de NSSAI permit

`amf_set_mapping` - Conjunto AMF alvo por S-NSSAI

Vincula uma chave S-NSSAI aos conjuntos AMF que podem servir aquele slice. Quando presente, o primeiro conjunto AMF é retornado como `targetAmfSet`, informando ao AMF para redirecionar se não puder servir o slice por conta própria.

Parâmetro	Tipo	Requerido	Padrão	Var de Ambiente	Des
`amf_set_mapping`	mapa	Não	`{"1-0x000001" => ["1"]}`	-	Tabel conju alvo i por S-
chave do mapa	string `"sst"` ou `"sst-sd"`	Sim	`"1-0x000001"`	-	Chave NSSA mape se ap
valor	lista de strings	Sim	`["1"]`	-	Identi do co AMF a o slice

> A correspondência de chaves para `amf_set_mapping` e `configured_nssai` é **exata**, ao contrário de `nsi_list`. A chave de pesquisa é construída a partir do S-NSSAI na consulta: `"sst"` quando o S-NSSAI solicitado não possui SD, `"sst-sd"` quando possui. Uma consulta que carrega um SD corresponde apenas a uma chave `"sst-sd"`; uma chave bare `"sst"` não corresponderá a ela, e vice-versa. Não há curinga de SD aqui. Se você serve um slice com um SD, use a forma `"sst-sd"` (por exemplo, `"1-0x000001"`); se o S-NSSAI solicitado não possui SD, use a forma bare `"sst"` (por exemplo, `"1"`).

`configured_nssai` - Mapeamento de NSSAI Configurado (roaming)

Vincula uma chave S-NSSAI a um valor de NSSAI Configurado retornado ao AMF para cenários de roaming/mapeamento. Vazio por padrão; quando definido, o valor é retornado literalmente como `configuredNssai`. Isso é usado para fornecer a um UE em roaming o mapeamento entre um S-NSSAI do PLMN que está servindo (visitado) e o correspondente S-NSSAI de seu PLMN doméstico,

para que o UE possa apresentar um S-NSSAI do PLMN doméstico quando se mover posteriormente. O NSSF não interpreta o valor - ele passa através de qualquer objeto que você configurar - portanto, deve ser estruturado exatamente como o AMF espera de acordo com a TS 29.531.

Parâmetro	Tipo	Requerido	Padrão	Var de Ambiente	De
`configured_nssai`	mapa	Não	`%{}` (vazio)	-	Tabela config index NSSA
chave do mapa	string `"sst"` ou `"sst-sd"`	Não	(nenhum)	-	Chave qual c mape config aplica (corre exata nota a
valor	objeto	Não	(nenhum)	-	Valor Config retorr literal como `confi`

Exemplo - retornar um mapeamento de S-NSSAI do PLMN doméstico para o slice do PLMN visitado `1-0x000001`:

```
config :omnissf,
  configured_nssai: %{
    "1-0x000001" => %{"sst" => 1, "sd" => "0x0000ff"}
  }
```

Deixe `configured_nssai` vazio (`%{}`) para implantações não roaming; a resposta então omite completamente `configuredNssai`.

Definindo um slice de ponta a ponta

Um único slice não é descrito por uma única chave de configuração - as quatro chaves acima são tabelas independentes, e uma resposta de seleção de slice é montada a partir de qualquer uma delas que contenha uma entrada correspondente. Para tornar um novo slice totalmente utilizável, normalmente você preenche duas, e opcionalmente todas as quatro:

Chave	Necessária para o funcionamento do slice?	O que controla na resposta
`nsi_list`	**Sim**	Se o S-NSSAI resolve ou não. Uma solicitação sem entrada NSI correspondente é rejeitada com `403 SNSSAI_NOT_FOUND`. Fornece `nsiInformation` (NRF alvo + ID NSI opcional)
`allowed_nssai`	**Sim**, por PLMN que está servindo	Se o UE é informado de que pode usar o slice. Sem uma entrada PLMN correspondente (e sem fallback para PLMN doméstico), a resposta omite `allowedNssaiList`, então o AMF não tem NSSAI Permitido para fornecer ao UE
`amf_set_mapping`	Opcional	Adiciona `targetAmfSet` para que o AMF possa redirecionar quando não puder servir o slice por conta própria. Omitir para implantações de conjunto único de AMF
`configured_nssai`	Opcional (apenas roaming)	Adiciona `configuredNssai` para mapeamento de S-NSSAI de visitado para doméstico

Exemplo prático - adicionar slice SST `2` (URLLC), SD `0x000010`, servido pelo NRF `http://10.0.0.20:7777`, permitido para PLMN doméstico `999-70`, redirecionável para conjunto AMF `2`:

```
config :omnissf,
  nsi_list: [
    %{s_nssai: %{sst: 2, sd: "0x000010"}, nrf_uri:
"http://10.0.0.20:7777", nsi_id: "2"}
  ],
  allowed_nssai: %{
    "999-70" => [%{sst: 2, sd: "0x000010"}]
  },
  amf_set_mapping: %{
    "2-0x000010" => ["2"]
  }
```

Lembre-se da assimetria na correspondência de SD: `nsi_list` e `allowed_nssai` comparam SDs sem distinção de maiúsculas e minúsculas, com `nil` atuando como um curinga, enquanto `amf_set_mapping` e `configured_nssai` são indexados por uma string exata `"sst-sd"`. Mantenha a forma do SD consistente (por exemplo, sempre `"0x000010"`) em todas as quatro chaves para que as chaves se alinhem.

O mesmo slice também pode ser adicionado em tempo de execução com `POST /oam/slice` (que escreve apenas `nsi_list`); tabelas de NSSAI permitido, conjunto AMF e NSSAI configurado são apenas de configuração e não são editáveis através da API OAM. Adições de NSI em tempo de execução reverterão na reinicialização.

Portas de Gerenciamento e Métricas

Os seguintes ouvintes voltados para o operador são iniciados juntamente com o endpoint SBI:

Ouvinte	Padrão	Var de Ambiente	Descrição
Gerenciamento / API OAM	TCP `8443` (HTTPS)	-	API REST para status, inspeção de slice, estatísticas e ações OAM em tempo de execução
Métricas do Prometheus	TCP `9568`	`PROMETHEUS_PORT`	Endpoint de coleta do Prometheus expondo métricas do NSSF e do BEAM

API de Gerenciamento / OAM

Uma API de gerenciamento/OAM REST é servida sobre HTTPS na porta `8443` (vincular `0.0.0.0`). Ela fornece status e estatísticas somente leitura, além de controles OAM em tempo de execução. Todas as respostas são JSON, envolvidas em um envelope padrão `success` / `error`. Os caminhos abaixo são relativos à base da API.

Status e Inspeção

Método	Caminho	Descrição
GET	`/status/api`	Liveness do serviço API
GET	`/status/nrf`	Status de registro do NRF
GET	`/status/nf`	Status do NF NSSF
GET	`/slice_config`	Instantâneo da configuração atual do slice
GET	`/api/slices`	S-NSSAIs configurados com mapeamentos NSI, conjuntos AMF alvo, NSSAI permitido por PLMN e mapeamento de conjunto AMF
GET	`/api/nssai_availability`	Disponibilidade de NSSAI por AMF atualmente armazenada
GET	`/api/statistics`	Estatísticas operacionais do NSSF
GET	`/oam/config`	Configuração efetiva em tempo de execução (endereço/porta SBI, URI do NRF, MCC/MNC, capacidade/prioridade do NF, nível de log)

`GET /api/statistics` retorna um instantâneo de contagens útil para uma verificação rápida de saúde/capacidade sem coletar do Prometheus:

Campo	Significado
`total_slices`	Número de entradas NSI atualmente carregadas (tamanho de `nsi_list`)
`total_allowed_nssai_plmns`	Número de PLMNs com uma entrada `allowed_nssai`
`total_amf_set_mappings`	Número de chaves S-NSSAI em `amf_set_mapping`
`nssai_availability_count`	Número de AMFs que relataram disponibilidade por TAI
`nssai_availability_subscription_count`	Número de assinaturas ativas de mudança de disponibilidade

Gerenciamento de Slice em Tempo de Execução

Método	Caminho	Descrição
POST	`/oam/slice`	Adicionar ou atualizar um mapeamento NSI. Corpo: `sst` (requerido), `nrf_uri` (requerido), `sd` opcional, `nsi_id`
DELETE	`/oam/slice/{id}`	Remover um mapeamento NSI. `id` é a chave S-NSSAI `"sst"` ou `"sst-sd"`. Retorna `404` se nenhuma NSI correspondente existir
DELETE	`/oam/nssai_availability/{id}`	Invalidar a disponibilidade de NSSAI armazenada para um AMF (`id` = `nfId`)

Mudanças de slice em tempo de execução se aplicam imediatamente a consultas de seleção de slice subsequentes, mas não são persistidas; elas reverterão para a topologia configurada na reinicialização.

Controles de Configuração em Tempo de Execução

Método	Caminho	Descrição
PATCH	`/oam/config/{id}`	Atualizar configuração em tempo de execução. Chaves permitidas: `nrf_uri`, `nf_capacity`, `nf_priority`, `log_level`. Chaves/valores inválidos retornam `400`
POST	`/oam/nrf/reregister`	Forçar re-registro do NRF
POST	`/oam/log_level`	Alterar nível de log. Corpo: `level` (um dos `emergency`, `alert`, `critical`, `error`, `warning`, `warn`, `notice`, `info`, `debug`)

Para `PATCH /oam/config/{id}`, `nf_capacity` e `nf_priority` devem ser inteiros e `log_level` deve ser uma string de nível válida; caso contrário, a solicitação retorna `400`.

← Visão Geral

Métricas do OmniNSSF

As métricas são expostas no formato Prometheus na porta de métricas (padrão `9568`, `PROMETHEUS_PORT`). As métricas específicas do NSSF estão listadas abaixo; as métricas de VM/memória do BEAM (`beam.memory.*`, `beam.processes.count`, `beam.ports.count`, `beam.atom.count`, `beam.vm.uptime`) também são exportadas.

Métricas do NSSF

Métrica	Tipo	R
`omni_nssf_nsselection_requests_count`	Contador	`re` `nf`
`omni_nssf_ns_selection_requests_total`	Contador	`re`
`omni_nssf_nssai_availability_update_count`	Contador	`nf`
`omni_nssf_nssai_availability_delete_count`	Contador	`nf`
`omni_nssf_nssai_availability_subscribe_count`	Contador	(n
`omni_nssf_nssai_availability_unsubscribe_count`	Contador	(n
`omni_nssf_nssai_availability_updates_total`	Contador	`nf`
`omni_nssf_nssai_availability_patch_total`	Contador	`nf`
`omni_nssf_nssai_availability_patch_subscription_total`	Contador	(n

Métrica	Tipo	R
`omni_nssf_nrf_registration_status`	Gauge	`nf`

> Existem dois contadores de seleção de slice quase idênticos que não devem ser somados. `omni_nssf_nsselection_requests_count` usa valores de `result` `success`/`error` e também carrega `nf_type`, enquanto `omni_nssf_ns_selection_requests_total` usa valores de `result` `success`/`failure` e não tem `nf_type`. Escolha um como sua fonte de verdade para evitar a contagem dupla das mesmas solicitações.

> `omni_nssf_nssai_availability_updates_total` (plural `updates`) e `omni_nssf_nssai_availability_update_count` (singular) estão ambos conectados ao mesmo evento de atualização PUT, então eles rastreiam juntos. Qualquer um pode ser usado para o volume total de atualização de disponibilidade.

Métricas da VM BEAM

Métrica	Tipo	Descrição
`beam_memory_total`	Gauge	Memória total alocada pela VM (bytes)
`beam_memory_processes` / `beam_memory_processes_used`	Gauge	Memória alocada para / usada por processos (bytes)
`beam_memory_system`	Gauge	Memória alocada para o sistema da VM (bytes)
`beam_memory_atom` / `beam_memory_atom_used`	Gauge	Memória alocada para / usada pela tabela de átomos (bytes)
`beam_memory_binary`	Gauge	Memória alocada para binários (bytes)
`beam_memory_code`	Gauge	Memória alocada para código carregado (bytes)
`beam_memory_ets`	Gauge	Memória alocada para tabelas ETS (bytes)
`beam_processes_count`	Gauge	Número de processos ativos
`beam_ports_count`	Gauge	Número de portas abertas
`beam_atom_count`	Gauge	Número de átomos na tabela de átomos

Métrica	Tipo	Descrição
`beam_vm_uptime`	Gauge	Tempo de atividade da VM (segundos)

Exemplo PromQL

Taxa de erro de seleção de slice ao longo de 5 minutos:

```
sum(rate(omni_nssf_nsselection_requests_count{result="error"}
[5m]))
  / sum(rate(omni_nssf_nsselection_requests_count[5m]))
```

Taxa de solicitação de seleção de slice por tipo de NF solicitado:

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

Saúde do registro NRF (alerta quando não registrado):

```
min(omni_nssf_nrf_registration_status) < 1
```

Taxa de atualização de disponibilidade NSSAI por AMF:

```
sum by (nf_id)
(rate(omni_nssf_nssai_availability_update_count[5m]))
```

← Visão Geral

Guia de Operações do OmniNSSF

OmniNSSF implementa a Função de Seleção de Fatiamento de Rede (NSSF) do Core 5G. Ele resolve consultas de seleção de fatiamento do AMF em uma resposta de seleção de fatiamento autorizada e coleta e autoriza a disponibilidade de fatiamentos por Área de Rastreamento reportada pelos AMFs.

A topologia de fatiamento (mapeamentos de S-NSSAI para NRF, NSSAI permitidos por PLMN, mapeamentos de conjunto AMF e NSSAI configurados) é mantida em um armazenamento `SliceConfig` em memória que é carregado a partir da configuração do aplicativo na inicialização e pode ser ajustado em tempo de execução através da API de gerenciamento OAM. A disponibilidade de NSSAI por AMF e as assinaturas de disponibilidade são mantidas em um armazenamento `Context` em memória. Este é o estado em tempo de execução: ele é reconstruído a partir da configuração e dos relatórios ao vivo do AMF após uma reinicialização.

OmniNSSF se registra no NRF na inicialização e mantém seu registro com batimentos cardíacos periódicos. Por padrão, os alvos NRF específicos de fatiamento são configurados pelo operador em `nsi_list` e retornados ao AMF como informações NSI. A descoberta dinâmica do NRF também está disponível como um comportamento opcional: quando `dynamic_nrf_discovery` está habilitado, a cada solicitação de seleção de fatiamento, o NSSF resolve o NRF que serve o fatiamento em tempo de execução a partir do NRF raiz via Nnrf_NFDiscovery, voltando para o `nsi_list` configurado estaticamente quando a descoberta não está acessível ou não retorna nada. Isso está desabilitado por padrão.

Referências de Papel e Especificação da 3GPP

Especificação	Relevância
TS 23.501 §5.15	Conceitos de fatiamento de rede - S-NSSAI, tipos de NSSAI (Solicitado / Permitido / Configurado / Assinado), Instância de Fatiamento de Rede, papel do NSSF
TS 23.502	Procedimentos - seleção de fatiamento durante Registro e Estabelecimento de Sessão PDU
TS 29.531	APIs de serviço Nnssf_NSSelection e Nnssf_NSSAIAvailability (HTTP/2 SBI)
TS 23.003	Formato S-NSSAI - SST (1 octeto) e SD opcional (3 octetos)

Conceitos de S-NSSAI e NSSAI

Uma **Informação de Assistência de Seleção de Fatiamento de Rede Única (S-NSSAI)** identifica um fatiamento de rede. De acordo com o TS 23.003, consiste em:

- **SST (Tipo de Fatiamento/Serviço)** - obrigatório, um valor de 8 bits. Valores padronizados incluem `1` = eMBB, `2` = URLLC, `3` = MIoT, `4` = V2X.
- **SD (Diferenciador de Fatiamento)** - opcional, um valor de 24 bits que distingue múltiplos fatiamentos do mesmo SST. Comumente escrito como uma string de 6 dígitos hexadecimais, como `0x000001`. Quando o SD está ausente, o S-NSSAI se aplica a qualquer/todos os diferenciadores desse SST.

Um **NSSAI** é um conjunto de S-NSSAIs. O sistema 5G distingue vários tipos de NSSAI, todos os quais o NSSF utiliza:

Tipo de NSSAI	Significado	Onde o OmniNSSF o utiliza
NSSAI Solicitado	S-NSSAIs que o UE pede para usar, enviados pelo AMF na consulta de seleção de fatiamento	Entrada - extraído do registro ou solicitação de informações de fatiamento da sessão PDU
NSSAI Assinado	S-NSSAIs que o UE tem direito a usar de acordo com sua assinatura	Entrada - fallback quando nenhum NSSAI Solicitado está presente em uma consulta de registro
NSSAI Permitido	S-NSSAIs que o UE pode realmente usar na PLMN/área de serviço	Saída - calculado a partir do conjunto permitido por PLMN e filtrado pela disponibilidade de TAI
NSSAI Configurado	S-NSSAIs configurados para o UE em uma PLMN (usado para roaming/mapeamento)	Saída - retornado quando um mapeamento de NSSAI Configurado é definido para o S-NSSAI

O NSSF resolve uma consulta da seguinte forma: ele compara o S-NSSAI solicitado com a lista de **Instâncias de Fatiamento de Rede (NSI)** configuradas para encontrar o NRF específico do fatiamento, calcula o NSSAI Permitido para a PLMN de serviço (opcionalmente filtrado pela disponibilidade de TAI reportada) e adiciona um conjunto de AMF alvo e um mapeamento de NSSAI Configurado quando estes estão configurados para o fatiamento.

Endpoints SBI

Todos os endpoints SBI são servidos sobre a URL base `{sbi_scheme}://{sbi_addr}:{sbi_port}` (padrão `http://127.0.0.14:7777`).

Nnssf_NSSelection (TS 29.531)

Método	Caminho	Descrição
GET	`/nnssf-nsselection/v2/network-slice-information`	Resolve a seleção de fatiamento para o AMF. Retorna um objeto `AuthorizedNetworkSliceInfo`

Parâmetros de consulta obrigatórios: `nf-type`, `nf-id`. Um parâmetro obrigatório ausente retorna `400` com causa `MANDATORY_IE_MISSING`. A entrada de fatiamento é fornecida através de uma das `slice-info-request-for-pdu-session`, `slice-info-request-for-registration`, ou um parâmetro direto `snssai`. Parâmetros opcionais incluem `tai`, `home-plmn-id`, e `supported-features`.

Códigos de resposta:

Código	Significado
`200`	`AuthorizedNetworkSliceInfo` (informações NSI, e onde aplicável NSSAI Permitido, conjunto AMF alvo, NSSAI Configurado)
`400`	Parâmetro obrigatório ausente ou nenhuma informação de fatiamento fornecida
`403`	Nenhuma NSI configurada para o S-NSSAI solicitado (causa `SNSSAI_NOT_FOUND`)

Nnssf_NSSAIAvailability (TS 29.531)

Método	Caminho	De
PUT	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	Armazena a lis suportados por `AuthorizedNss`
PATCH	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	Atualiza parcia disponibilidade usando JSON P `application/j`
DELETE	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	Remove a disp armazenada pa `204`
POST	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions`	Cria uma assin de disponibilid com um cabeç
PATCH	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/{subscriptionId}`	Atualiza uma a JSON Patch (RF
DELETE	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/{subscriptionId}`	Remove uma a `204`

Um corpo de disponibilidade PUT/PATCH deve conter um array `supportedNssaiAvailabilityData` não vazio, cada entrada carregando um objeto `tai` e uma `supportedSnssaiList` não vazia. Um PATCH na disponibilidade armazenada (ou em uma assinatura) que referencia um `{nfId}` / `{subscriptionId}` desconhecido retorna `404`. Um corpo de assinatura deve conter `nfNssaiAvailabilityUri`, uma `taiList` não vazia, e um `event`. Qualquer caminho SBI não correspondido retorna `404`.

Forma dos dados de disponibilidade e correspondência de TAI

Cada entrada de `supportedNssaiAvailabilityData` tem duas partes obrigatórias:

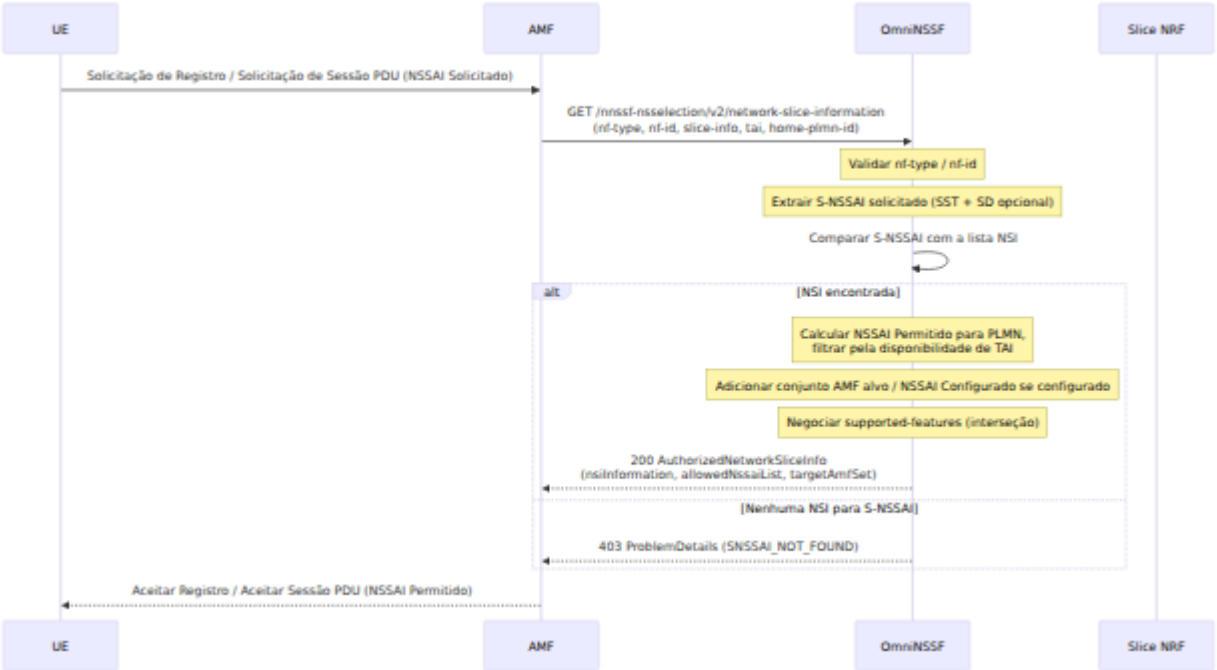
Campo	Forma	Significado
`tai`	`{ "plmnId": { "mcc", "mnc" }, "tac" }`	A Área de Rastreamento à qual a disponibilidade se aplica
`supportedSnssaiList`	array não vazio de `{ "sst", "sd"? }`	S-NSSAIs que o AMF reportante suporta nessa TAI

Quando a seleção de fatiamento filtra um NSSAI Permitido por uma TAI de consulta (veja Seleção de Fatiamento de Rede), a correspondência é **exata em todos os três de `mcc`, `mnc`, e `tac`**, e um S-NSSAI é considerado disponível apenas se alguma entrada reportada para essa TAI a listar (SD comparado exatamente, ou correspondendo a qualquer SD quando a consulta omite SD). Um fatiamento pode, portanto, parecer indisponível simplesmente porque o AMF reportou uma TAI cujo `tac` ou PLMN difere daquele na consulta de seleção - assegure-se de que o AMF reporte a mesma TAI que codifica em consultas posteriores.

Procedimentos Chave

Seleção de Fatiamento de Rede (Nnssf_NSSelection)

Consultado pelo AMF durante o Registro do UE e durante o Estabelecimento da Sessão PDU. O NSSF resolve o S-NSSAI solicitado em uma resposta de seleção de fatiamento autorizada.

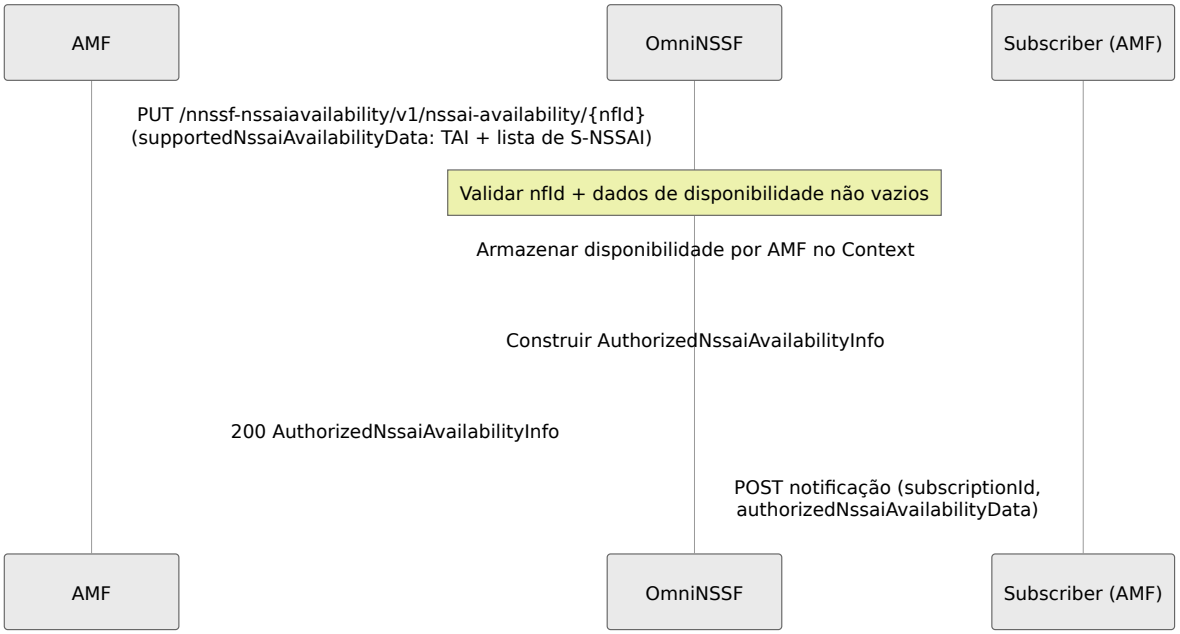


Quando uma TAI é fornecida na consulta e um ou mais AMFs relataram disponibilidade, o NSSF verifica se o S-NSSAI solicitado está disponível naquela TAI e filtra o NSSAI Permitido de acordo. Se nenhuma disponibilidade foi reportada ainda, todos os fatiamentos configurados são tratados como disponíveis (padrão permissivo; defina `strict_availability: true` para falhar fechado). Quando o AMF fornece um valor `supported-features`, o NSSF retorna a interseção negociada em relação ao seu próprio conjunto de recursos (atualmente o bit NssaiAvailability).

O NRF alvo retornado como `nsiInformation` normalmente vem da entrada correspondente da `nsi_list`. Quando `dynamic_nrf_discovery` está habilitado, o NSSF resolve o NRF que serve o fatiamento em tempo de execução consultando o NRF raiz via Nnrf_NFDiscovery para o S-NSSAI solicitado, e volta para o NRF do `nsi_list` configurado quando a descoberta está desabilitada, inacessível, ou não retorna nada. Veja a Referência de Configuração para a chave de configuração.

Atualização de Disponibilidade de NSSAI (Nnssf_NSSAIAvailability)

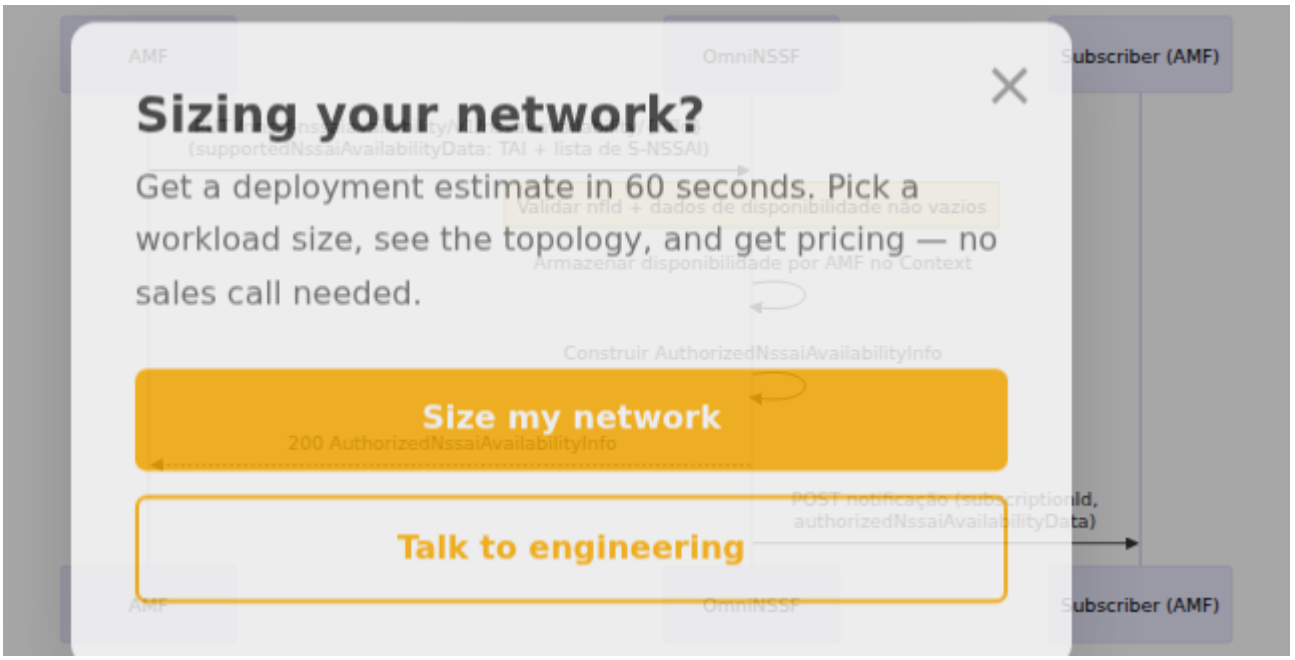
AMFs enviam os S-NSSAIs que suportam por TAI. O NSSF armazena e autoriza essa informação e notifica quaisquer assinantes da mudança.



Um PATCH na disponibilidade armazenada aplica as operações JSON Patch (RFC 6902) ao registro existente por AMF; um PATCH contra um `{nfId}` desconhecido retorna `404`. Tanto PUT quanto PATCH re-notificam os assinantes com os dados de disponibilidade autorizados.

Assinatura e Notificação de Disponibilidade de NSSAI

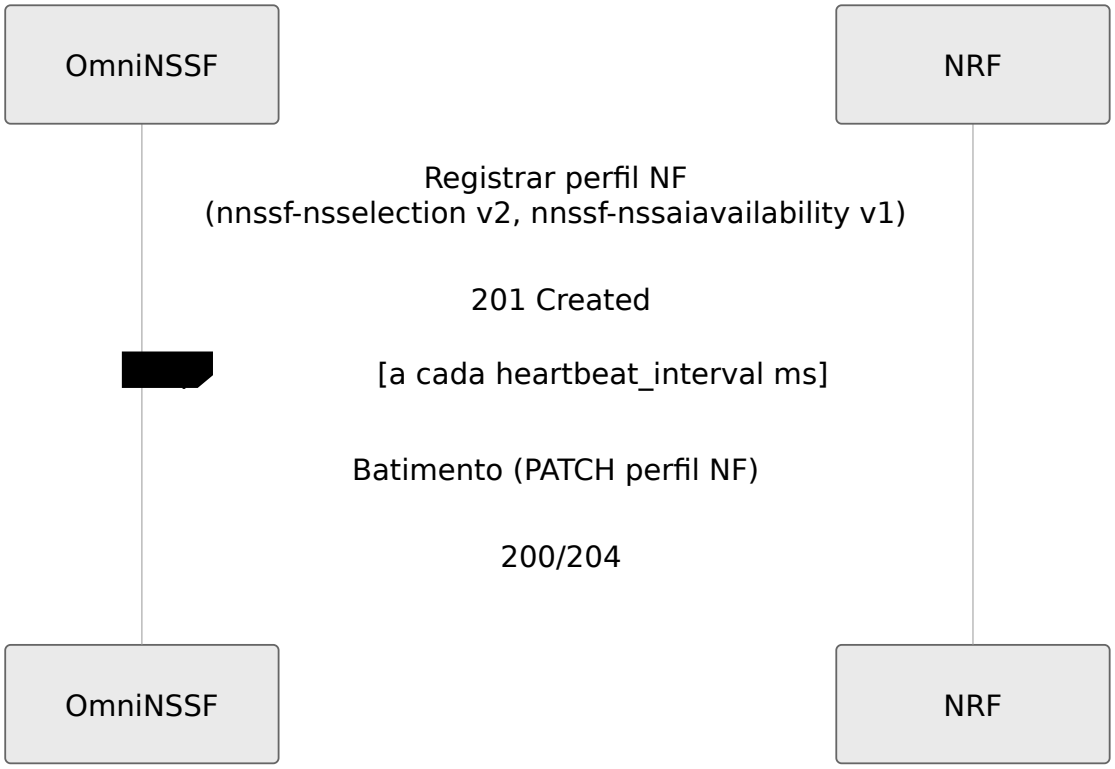
Os assinantes registram uma URI de callback; o NSSF dispara uma notificação de mudança para cada assinante sempre que um PUT ou PATCH atualiza a disponibilidade armazenada.



As notificações são POSTs HTTP para a `nfNssaiAvailabilityUri` de cada assinante; a notificação carrega o `subscriptionId` e os `authorizedNssaiAvailabilityData`. Cada notificação é entregue fora do caminho de solicitação com tentativas limitadas: até `nssf_notification_max_attempts` tentativas (padrão `3`) usando o cronograma de backoff `nssf_notification_backoff_ms` (padrão `200` ms e depois `1000` ms), após o qual a notificaç►►o é descartada e a falha é registrada. Como a entrega é fora do caminho de solicitação, um assinante lento ou inacessível não bloqueia a atualização de disponibilidade que acionou a notificação.

Registro e Batimento do NRF

Na inicialização, o NSSF registra seu perfil NF no NRF, anunciando ambos os serviços SBI, e então se registra novamente em um batimento cardíaco periódico.



O status de registro é refletido no gauge `omni_nssf_nrf_registration_status` e pode ser forçado a se registrar novamente através da API OAM (`POST /oam/nrf/reregister`).

← Visão Geral

Solução de Problemas do OmniNSSF

Problemas Comuns

Sintoma	Causa provável	
A seleção de slice retorna `400` `MANDATORY_IE_MISSING`	A consulta AMF não possui `nf-type` ou `nf-id`	Confirme se o parâmetros de obrigatórios
A seleção de slice retorna `400` "Nenhuma informação de slice fornecida"	A consulta não possui nenhum de `slice-info-request-for-pdu-session`, `slice-info-request-for-registration` ou `snssai`	Verifique se o slice-info para
A seleção de slice retorna `403` `SNSSAI_NOT_FOUND`	O S-NSSAI solicitado não possui uma entrada NSI correspondente	Adicione o S-N (config) ou via
A resposta omite `allowedNssaiList`	Nenhuma entrada `allowed_nssai` para o PLMN da consulta e nenhuma correspondência de fallback home-PLMN	Adicione o PLM `mnc`) a `allowe`
O slice parece indisponível em um TAI	Existe um relatório de disponibilidade AMF conflitante para aquele TAI sem o S-NSSAI	Inspecione `GE` `/api/nssai_av` invalide dados `DELETE` `/oam/nssai_av`
Sem `targetAmfSet` na resposta	Nenhuma entrada `amf_set_mapping` para a chave S-NSSAI	Adicione a cha `amf_set_mapp`
Status de registro NRF `0`	NRF inacessível ou `nrf_uri` incorreto	Verifique `NRF_` registro com `P` `/oam/nrf/rere`

Sintoma	Causa provável	
PUT de disponibilidade rejeitado `400`	Dados `supportedNssaiAvailabilityData` ausentes/vazios, ou uma entrada carece de `tai` ou de uma `supportedSnssaiList` não vazia	Corrija a forma disponibilidad
PATCH de disponibilidade retorna `404`	JSON Patch enviado para um `nfId` sem disponibilidade armazenada	Primeiro, faça disponibilidad `nfId` correto
Criação de assinatura rejeitada `400`	Corpo ausente `nfNssaiAvailabilityUri`, `taiList` ou `event`	Forneça os trê assinatura obr
PATCH/DELETE de assinatura retorna `404`	`subscriptionId` desconhecido	Confirme o `su` cabeçalho `Loc`
Assinantes não notificados	Callback `nfNssaiAvailabilityUri` inacessível	As notificaçõe novamente co (`nssf_notifi` tentativas, pa serem descart acessibilidade logs do NSSF desistência