← 述

OmniNSSF 置参考

操作员配置从应用程序环境键 `:omnissf` 中读取。运行时配置通过 `config/runtime.exs` 提供，该文件在启动时将操作系统环境变量映射到配置键。切片拓扑（NSI 列表、允许的 NSSAI、AMF 集映射、配置的 NSSAI）通过结构化应用程序配置提供（请参见切片拓扑配置），并通过 OAM API 在运行时进行调整。

运行时置（`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
```

□□	□□	□□	□□	□
`sbi_scheme`	□□□	□	`"http"`	`SBI_SCHE`
`sbi_addr`	□□□	□	`"127.0.0.14"`	`SBI_ADDR`
`sbi_port`	□□	□	`7777`	`SBI_PORT`
`nrf_uri`	□□□	□	`"http://127.0.0.1:7777"`	`NRF_URI`
`mcc`	□□□	□	`"999"`	`MCC`
`mnc`	□□□	□	`"70"`	`MNC`

参数	类型	必需	默认	[
`prometheus_metrics_port`	整数	否	`9568`	`PROMETHE`
`heartbeat_interval`	整数（毫秒）	否	`10000`	`HEARTBEA`

> 注意：运行时配置中 `nrf_uri` 的编译默认值为 `http://127.0.0.1:7777`，而内部访问器的默认值为 `http://127.0.0.10:7777`。当应用 `runtime.exs` 时，有效默认值为 `http://127.0.0.1:7777`。请显式设置 `NRF_URI` 以避免歧义。

行为和通知设置

以下 `:omnissf` 应用程序配置键调整切片选择和通知行为。没有 `runtime.exs` 环境变量，因此通过应用程序配置而不是操作系统环境变量进行设置。

参数	类型	必需	默认	描述
`dynamic_nrf_discovery`	布尔值	否	`false`	当为 `true` 时，NSSF 在运行时通过 Nnrf_NFDiscovery 从根 NRF 发现切片服务 NRF，而不是仅使用静态 `nsi_list`。在每次切片选择请求中，它查询根 NRF 以获取请求的 S-NSSAI，并在发现被禁用、不可达或返回空时回退到配置的 `nsi_list` NRF
`strict_availability`	布尔值	否	`false`	可用性强制模式。当为 `false`（宽松）时，未显式可用性记录的 TAI 仍会返回配置的允许 NSSAI。当为 `true`（失败关闭）时，此类 TAI 被拒绝，允许 NSSAI 返回为空
`nssf_notification_max_attempts`	整数	否	`3`	每个可用性变化通知的最大投递尝试次数，超过后将被丢弃并记录失败
`nssf_notification_backoff_ms`	整数列表	否	`[200, 1000]`	每次重试的可用性变化通知的退避计划（毫秒）。第一个值在第二次尝试之前应用，第二个值在第三次之前应用，最后一个值在任何进一步尝试中重复使用

日志记录

参数	类型	必需	默认	环境变量	描述
`format`	格式化元组	否	`{OmniLogger.JsonFormatter, :format}`	-	默认日志记录器的结构化 JSON 日志格式化程序
`metadata`	原子 / 列表	否	`:all`	-	每个条目中包含的日志元数据（`:all` 会输出每个元数据字段）

活动日志级别可以通过 OAM API 在运行时更改而无需重启（`PATCH /oam/config/{id}` 带 `log_level`，或 `POST /oam/log_level`）。有效级别为 `emergency`、`alert`、`critical`、`error`、`warning`、`warn`、`notice`、`info` 和 `debug`。

切片拓扑置

切片拓扑在启动时从以下 `:omnissf` 应用程序配置键加载到 `SliceConfig` 存储中。每个键都有一个内置默认值，建模为单个 eMBB 切片（SST 1，SD `0x000001`）。这些键是结构性的（映射和列表），通过应用程序配置设置，而不是单个环境变量；它们也可以通过 OAM API 在运行时更改（请参见 运行时切片管理）。

```
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` - 网络切片实例条目

每个条目将 S-NSSAI 绑定到为该切片提供切片特定 NF 发现的 NRF。NSSF 将请求的 S-NSSAI 与此列表进行匹配；匹配时返回 NSI 信息给 AMF，不匹配时返回 `403`。

参数	类型	必需	默认	环境变量	描述
`nsi_list`	映射列表	否	单个 eMBB 条目（如下）	-	启动时加载的完整 NSI 条目集
`s_nssai.sst`	整数	是	`1`	-	切片的切片/服务类型
`s_nssai.sd`	字符串	否	`"0x000001"`	-	切片区分符。当省略时，该条目与的任何 SD 匹配
`nrf_uri`	字符串	是	`"http://127.0.0.10:7777"`	-	为该切片提供服务的 NRF 的URI，返回给 AMF 作为 `nsiInformation.r`
`nsi_id`	字符串	否	`"1"`	-	网络切片实例标识符，返回为 `nsiInformation.r` 未设置时从响应中省略

SD 匹配是支持通配符的：配置条目或请求中的 `nil` SD 匹配任何 SD，非 `nil` SD 不区分大小写进行比较。

`allowed_nssai` - 每个 PLMN 的允许 NSSAI

将 PLMN 键 `"mcc-mnc"` 映射到该 PLMN 的 UE 允许的 S-NSSAIs 列表。当查询的 PLMN 没有条目时，NSSF 回退到 home PLMN（来自运行时配置的 `mcc-mnc`）。结果列表作为 `allowedNssaiList` 返回（访问类型 `3GPP_ACCESS`），并且在提供 TAI 时，会过滤到在该 TAI 上报告可用的 S-NSSAIs。

参数	类型	必需	默认	环境变量	描述
`allowed_nssai`	映射	否	`{"999-70" => [...]}`	-	PLMN 键控的允许 NSSAI 表
映射键	字符串 `"mcc-mnc"`	是	`"999-70"`	-	允许集适用的 PLMN 标识符
`sst`	整数	是	`1`	-	允许的 S-NSSAI 的 SST
`sd`	字符串	否	`"0x000001"`	-	允许的 S-NSSAI 的 SD

`amf_set_mapping` - 每个 S-NSSAI 的目标 AMF 集

将 S-NSSAI 键映射到能够为该切片服务的 AMF 集。当存在时，第一个 AMF 集作为 `targetAmfSet` 返回，告知 AMF 如果无法为切片服务则重新路由。

参数	类型	必需	默认	环境变量	描述
`amf_set_mapping`	映射	否	`{"1-0x000001" => ["1"]}`	-	S-NSSAI 键控的目标 AMF 集表
映射键	字符串 `"sst"` 或 `"sst-sd"`	是	`"1-0x000001"`	-	映射适用的 S-NSSAI 键
值	字符串列表	是	`["1"]`	-	切片的目标 AMF 集标识符

> `amf_set_mapping` 和 `configured_nssai` 的键匹配是 精确 的，与 `nsi_list` 不同。查找键是从查询中的 S-NSSAI 构建的：当请求的 S-NSSAI 没有 SD 时为 `"sst"`，当有时为 `"sst-sd"`。仅携带 SD 的查询只匹配 `"sst-sd"` 键；裸 `"sst"` 键将不匹配它，反之亦然。这里没有 SD 通配符。如果您提供一个带有 SD 的切片，请

> 使用 `"sst-sd"` 形式（例如 `"1-0x000001"`）；如果请求的 S-NSSAI 没有 SD，请使用裸 `"sst"` 形式（例如 `"1"`）。

`configured_nssai` - 配置的 NSSAI 映射（漫游）

将 S-NSSAI 键映射到返回给 AMF 的配置 NSSAI 值，用于漫游/映射场景。默认情况下为空；设置时，值将原样返回为 `configuredNssai`。这用于为漫游 UE 提供服务（访问）PLMN 的 S-NSSAI 与其 home PLMN 的对应 S-NSSAI 之间的映射，以便 UE 在后续移动时可以呈现 home-PLMN S-NSSAI。NSSF 不解释该值 - 它传递您配置的任何对象 - 因此它必须完全符合 AMF 根据 TS 29.531 的期望。

参数	类型	必需	默认	环境变量	描述
`configured_nssai`	映射	否	`%{}`（空）	-	S-NSSAI 键控的配置 NSSAI 表
映射键	字符串 `"sst"` 或 `"sst-sd"`	否	（无）	-	配置映射适用的 S-NSSAI 键（精确匹配，见上面注释）
值	对象	否	（无）	-	原样返回的配置 NSSAI 值作为 `configuredNssai`

示例 - 返回访问 PLMN 切片 `1-0x000001` 的 home-PLMN S-NSSAI 映射：

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

对于非漫游部署，请保持 `configured_nssai` 为空（`%{}`）；响应将完全省略 `configuredNssai`。

定义端到端切片

单个切片不是由一个配置键描述的 - 上述四个键是独立的表，切片选择响应是从包含匹配条目的任意键组装而成。要使新切片完全可用，通常需要填充两个键，并可选地填充所有四个：

键	切片正常工作所需?	控制响应中的内容
`nsi_list`	是	S-NSSAI 是否解析。没有匹配 NSI 条目的请求将被拒绝，返回 `403 SNSSAI_NOT_FOUND`。提供 `nsiInformation`（目标 NRF + 可选 NSI ID）
`allowed_nssai`	是，每个提供 PLMN	UE 是否被告知可以使用该切片。没有匹配的 PLMN 条目（且没有 home-PLMN 回退）时，响应将省略 `allowedNssaiList`，因此 AMF 没有允许的 NSSAI 提供给 UE
`amf_set_mapping`	可选	添加 `targetAmfSet`，以便 AMF 在无法为切片服务时可以重新路由。对于单 AMF 集部署省略
`configured_nssai`	可选（仅漫游）	为访问到 home S-NSSAI 映射添加 `configuredNssai`

示例 - 添加切片 SST `2`（URLLC），SD `0x000010`，由 NRF `http://10.0.0.20:7777` 提供，允许 home PLMN `999-70`，可重新路由到 AMF 集 `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"]
  }
```

请记住 SD 匹配的不对称性：`nsi_list` 和 `allowed_nssai` 在比较 SD 时不区分大小写，`nil` 作为通配符，而 `amf_set_mapping` 和 `configured_nssai` 是通过精确的 `"sst-sd"` 字符串键控。保持所有四个键中 SD 形式的一致性（例如始终使用 `"0x000010"`），以确保键对齐。

同样的切片也可以通过 `POST /oam/slice` 在运行时添加（仅写入 `nsi_list`）；允许 NSSAI、AMF 集和配置 NSSAI 表仅为配置项，不能通过 OAM API 编辑。运行时 NSI 添加在重启时恢复。

管理和指标端口

以下操作员面向的��听器与 SBI 端点一起启动：

监听器	默认	环境变量	描述
管理 / OAM API	TCP `8443`（HTTPS）	-	用于状态、切片检查、统计信息和运行时 OAM 操作的 REST API
Prometheus 指标	TCP `9568`	`PROMETHEUS_PORT`	暴露 NSSF 和 BEAM 指标的 Prometheus 抓取端点

管理 / OAM API

REST 管理/OAM API 通过 HTTPS 在端口 `8443` 上提供（绑定 `0.0.0.0`）。它提供只读状态和统计信息以及运行时 OAM 控制。所有响应都是 JSON，包装在标准的 `success` / `error` 信封中。以下路径相对于 API 基础路径。

状态和检查

方法	路径	描述
GET	`/status/api`	API 服务存活性
GET	`/status/nrf`	NRF 注册状态
GET	`/status/nf`	NSSF NF 状态
GET	`/slice_config`	当前切片配置快照
GET	`/api/slices`	配置的 S-NSSAIs 及其 NSI 映射、目标 AMF 集、每个 PLMN 的允许 NSSAI 和 AMF 集映射
GET	`/api/nssai_availability`	当前存储的每 AMF NSSAI 可用性
GET	`/api/statistics`	NSSF 操作统计信息
GET	`/oam/config`	有效的运行时配置（SBI 地址/端口、NRF URI、MCC/MNC、NF 容量/优先级、日志级别）

`GET /api/statistics` 返回一个计数快照，便于快速健��/容量检查，而无需抓取 Prometheus:

字段	意义
`total_slices`	当前加载的 NSI 条目数量（`nsi_list` 大小）
`total_allowed_nssai_plmns`	具有 `allowed_nssai` 条目的 PLMN 数量
`total_amf_set_mappings`	`amf_set_mapping` 中的 S-NSSAI 键数量
`nssai_availability_count`	报告每 TAI 可用性的 AMF 数量
`nssai_availability_subscription_count`	活动可用性变化订阅的数量

运行时切片管理

方法	路径	描述
POST	`/oam/slice`	添加或更新 NSI 映射。主体：`sst`（必需）、`nrf_uri`（必需）、可选 `sd`、`nsi_id`
DELETE	`/oam/slice/{id}`	删除 NSI 映射。`id` 是 S-NSSAI 键 `"sst"` 或 `"sst-sd"`。如果没有匹配的 NSI，则返回 `404`
DELETE	`/oam/nssai_availability/{id}`	使存储的 NSSAI 可用性无效，针对 AMF（`id` = `nfId`）

运行时切片更改立即应用于后续的切片选择查询，但不会持久化；它们在重启时恢复到配置的拓扑。

运行时□置控制

方法	路径	描述
PATCH	`/oam/config/{id}`	更新运行时配置。允许的键：`nrf_uri`、`nf_capacity`、`nf_priority`、`log_level`。无效的键/值返回 `400`
POST	`/oam/nrf/reregister`	强制 NRF 重新注册
POST	`/oam/log_level`	更改日志级别。主体：`level`（`emergency`、`alert`、`critical`、`error`、`warning`、`warn`、`notice`、`info`、`debug` 之一）

对于 `PATCH /oam/config/{id}`，`nf_capacity` 和 `nf_priority` 必须是整数，`log_level` 必须是有效的级别字符串；否则请求返回 `400`。

← 概述

OmniNSSF 指标

指标以 Prometheus 格式在指标端口上公开（默认 `9568`，`PROMETHEUS_PORT`）。NSSF 特定指标如下所示；BEAM VM/内存指标（`beam.memory.*`，`beam.processes.count`，`beam.ports.count`，`beam.atom.count`，`beam.vm.uptime`）也会导出。

NSSF □□

□□	□□	□□
`omni_nssf_nsselection_requests_count`	□□□	`result`, `nf_type`
`omni_nssf_ns_selection_requests_total`	□□□	`result`
`omni_nssf_nssai_availability_update_count`	□□□	`nf_id`
`omni_nssf_nssai_availability_delete_count`	□□□	`nf_id`
`omni_nssf_nssai_availability_subscribe_count`	□□□	(□)
`omni_nssf_nssai_availability_unsubscribe_count`	□□□	(□)
`omni_nssf_nssai_availability_updates_total`	□□□	`nf_id`
`omni_nssf_nssai_availability_patch_total`	□□□	`nf_id`

指标	类型	标签
`omni_nssf_nssai_availability_patch_subscription_total`	计数器	(无)
`omni_nssf_nrf_registration_status`	测量	`nf_type`

> 存在两个几乎相同的切片选择计数器，且不应将它们相加。`omni_nssf_nsselection_requests_count` 使用 `result` 值 `success`/`error`，并且还携带 `nf_type`，而 `omni_nssf_ns_selection_requests_total` 使用 `result` 值 `success`/`failure`，且没有 `nf_type`。选择一个作为你的真实来源，以避免重复计算相同的请求。

> `omni_nssf_nssai_availability_updates_total`（复数 `updates`）和 `omni_nssf_nssai_availability_update_count`（单数）都连接到同一个 PUT 更新事件，因此它们一起跟踪。可以使用任一项作为可用性更新量的总量。

BEAM VM □□

□□	□□	□□
`beam_memory_total`	□□	VM □□□□□□□□□□□□
`beam_memory_processes` / `beam_memory_processes_used`	□□	□□□ / □□□□□□□□□□□□□□
`beam_memory_system`	□□	□□□ VM □□□□□□□□□□
`beam_memory_atom` / `beam_memory_atom_used`	□□	□□□ / □□□□□□□□□□□□□
`beam_memory_binary`	□□	□□□□□□□□□□□□□□□□□□
`beam_memory_code`	□□	□□□□□□□□□□□□□□□□□□
`beam_memory_ets`	□□	□□□ ETS □□□□□□□□□
`beam_processes_count`	□□	□□□□□□□□
`beam_ports_count`	□□	□□□□□□□□
`beam_atom_count`	□□	□□□□□□□□□□

指标	类型	描述
`beam_vm_uptime`	测量	VM 的正常运行时间（秒）

示例 PromQL

切片选择错误率在 5 分钟内：

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

按请求 NF 类型的切片选择请求率：

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

NRF 注册健康状况（未注册时警报）：

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

按 AMF 的 NSSAI 可用性更新率：

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

← 概述

OmniNSSF 操作指南

OmniNSSF 实现了 5G 核心的网络切片选择功能 (NSSF)。它将来自 AMF 的切片选择查询解析为授权的切片选择响应，并收集和授权 AMF 报告的每个跟踪区域切片可用性。

切片拓扑 (S-NSSAI 到 NRF 的映射、每个 PLMN 的允许 NSSAI、AMF 集映射和配置的 NSSAI) 存储在内存中的 `SliceConfig` 存储中，该存储在启动时从应用程序配置加载，并可以通过 OAM 管理 API 在运行时进行调整。每个 AMF 的 NSSAI 可用性和可用性订阅存储在内存中的 `Context` 存储中。这是运行时状态：在重启后，它是从配置和实时 AMF 报告重建的。

OmniNSSF 在启动时向 NRF 注册，并通过定期心跳保持其注册。默认情况下，切片特定的 NRF 目标在 `nsi_list` 中由运营商配置，并作为 NSI 信息返回给 AMF。动态 NRF 发现也可作为可选行为：当启用 `dynamic_nrf_discovery` 时，在每个切片选择请求中，NSSF 从根 NRF 通过 Nnrf_NFDiscovery 在运行时解析切片服务 NRF，当发现不可达或未返回任何内容时，回退到静态配置的 `nsi_list`。默认情况下，这个功能是禁用的。

3GPP 角色和规范参考

规范	相关性
TS 23.501 §5.15	网络切片概念 - S-NSSAI、NSSAI 类型（请求/允许/配置/订阅）、网络切片实例、NSSF 角色
TS 23.502	程序 - 在注册和 PDU 会话建立期间的切片选择
TS 29.531	Nnssf_NSSelection 和 Nnssf_NSSAIAvailability 服务 API (HTTP/2 SBI)
TS 23.003	S-NSSAI 格式 - SST (1 字节) 和可选 SD (3 字节)

S-NSSAI 和 NSSAI 概念

单一网络切片选择辅助信息 **(S-NSSAI)** 标识一个网络切片。根据 TS 23.003，它由以下部分组成：

- **SST (**切片/服务类型**) -** 强制性，8 位值。标准化值包括 `1` = eMBB，`2` = URLLC，`3` = MIoT，`4` = V2X。
- **SD (**切片区分符**) -** 可选，24 位值，用于区分同一 SST 的多个切片。通常写作 6 个十六进制数字的字符串，例如 `0x000001`。当 SD 缺失时，S-NSSAI 适用于该 SST 的任何/所有区分符。

NSSAI 是一组 S-NSSAIs。5G 系统区分几种 NSSAI 类型，NSSF 与所有这些类型一起工作：

NSSAI 类型	含义	**OmniNSSF** 使用场景
请求的 **NSSAI**	UE 请求使用的 S-NSSAIs，由 AMF 在切片选择查询中发送	输入 - 从注册或 PDU 会话切片信息请求中提取
订阅的 **NSSAI**	UE 根据其订阅有权使用的 S-NSSAIs	输入 - 当注册查询中没有请求的 NSSAI 时的回退
允许的 **NSSAI**	UE 实际可以在服务 PLMN/区域中使用的 S-NSSAIs	输出 - 从每个 PLMN 允许的集合计算，并根据 TAI 可用性进行过滤
□置的 **NSSAI**	在 PLMN 中为 UE 配置的 S-NSSAIs（用于漫游/映射）	输出 - 当为 S-NSSAI 定义了配置的 NSSAI 映射时返回

NSSF 解析查询的方式如下：它将请求的 S-NSSAI 与配置的 网络切片实例 **(NSI)** 列表进行匹配，以找到切片特定的 NRF，计算服务 PLMN 的允许 NSSAI（可选地根据 TAI 的报告可用性进行过滤），并在为切片配置时添加目标 AMF 集和配置的 NSSAI 映射。

SBI 端点

所有 SBI 端点通过基础 URL `{sbi_scheme}://{sbi_addr}:{sbi_port}` 提供服务（默认 `http://127.0.0.14:7777`）。

Nnssf_NSSelection (TS 29.531)

方法	路径	描述
GET	`/nnssf-nsselection/v2/network-slice-information`	解析 AMF 的切片选择。返回 `AuthorizedNetworkSliceInfo` 对象

强制查询参数：`nf-type`，`nf-id`。缺少强制参数将返回 `400`，原因是 `MANDATORY_IE_MISSING`。切片输入通过 `slice-info-request-for-pdu-session`、`slice-info-request-for-registration` 或直接的 `snssai` 参数提供。可选参数包括 `tai`、`home-plmn-id` 和 `supported-features`。

响应代码：

代码	含义
`200`	`AuthorizedNetworkSliceInfo`（NSI 信息，以及在适用时的允许 NSSAI、目标 AMF 集、配置的 NSSAI）
`400`	缺少强制参数或未提供切片信息
`403`	请求的 S-NSSAI 没有配置 NSI（原因 `SNSSAI_NOT_FOUND`）

Nnssf_NSSAIAvailability (TS 29.531)

方法	路径	
PUT	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	存储 AMF 的每个 TAI `AuthorizedNss`
PATCH	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	使用 JSON Patch (`application/j` 更新存储的可用性
DELETE	`/nnssf-nssaiavailability/v1/nssai-availability/{nfId}`	删除 AMF 的存储可用性[
POST	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions`	创建可用性变化订阅。返回 头
PATCH	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/{subscriptionId}`	使用 JSON Patch (
DELETE	`/nnssf-nssaiavailability/v1/nssai-availability/subscriptions/{subscriptionId}`	删除订阅。返回 `204`

PUT/PATCH 可用性主体必须包含一个非空的 `supportedNssaiAvailabilityData` 数组，每个条目携带一个 `tai` 对象和一个非空的 `supportedSnssaiList`。对存储的可用性（或订阅）进行 PATCH，如果引用未知的 `{nfId}` / `{subscriptionId}`，则返回 `404`。订阅主体必须包含 `nfNssaiAvailabilityUri`、非空的 `taiList` 和 `event`。任何不匹配的 SBI 路径返回 `404`。

可用性数据形状和 **TAI** 匹

每个 `supportedNssaiAvailabilityData` 条目有两个必需部分：

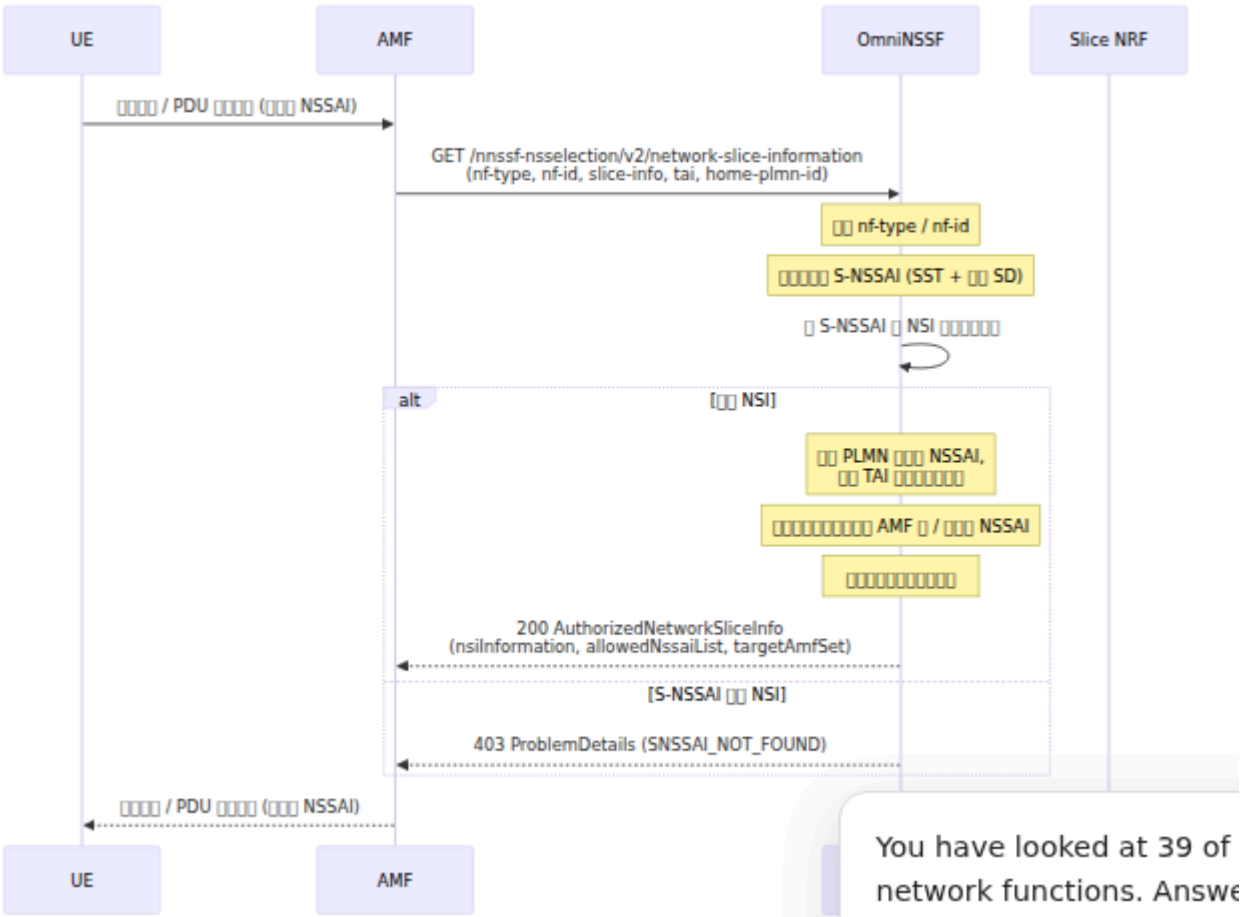
字段	形状	含义
`tai`	`{ "plmnId": { "mcc", "mnc" }, "tac" }`	可用性适用的跟踪区域
`supportedSnssaiList`	非空数组的 `{ "sst", "sd"? }`	报告 AMF 在该 TAI 中支持的 S-NSSAIs

当切片选择稍后通过查询 TAI 过滤允许 NSSAI 时（见 网络切片选择），匹配是在 **`mcc`**、**`mnc`** 和 **`tac`** 的所有三个字段上完全匹[]，只有当某个报告条目为该 TAI 列出时，S-NSSAI 才被视为可用（SD 完全匹配，或在查询省略 SD 时匹配任何 SD）。因此，切片可能看起来不可用，仅仅是因为 AMF 报告了一个与选择查询中的 `tac` 或 PLMN 不同的 TAI - 确保 AMF 报告相同的 TAI，以便后续查询时使用。

关键程序

网络切片选择 **(Nnssf_NSSelection)**

在 UE 注册和 PDU 会话建立期间由 AMF 查询。NSSF 将请求的 S-NSSAI 解析为授权的切片选择响应。

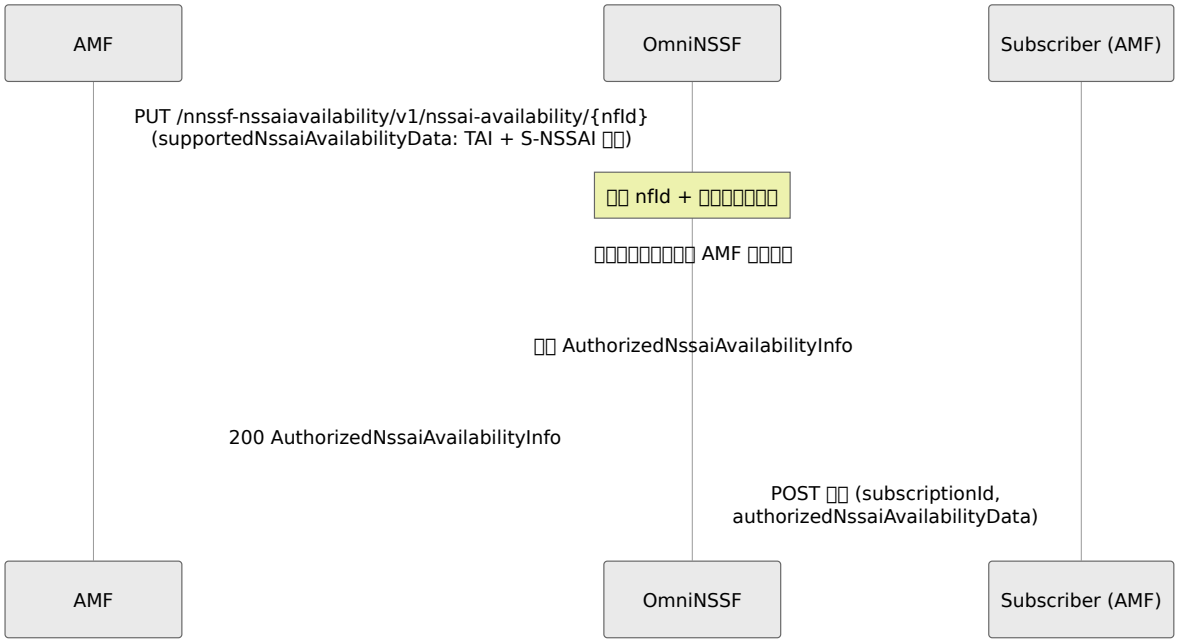


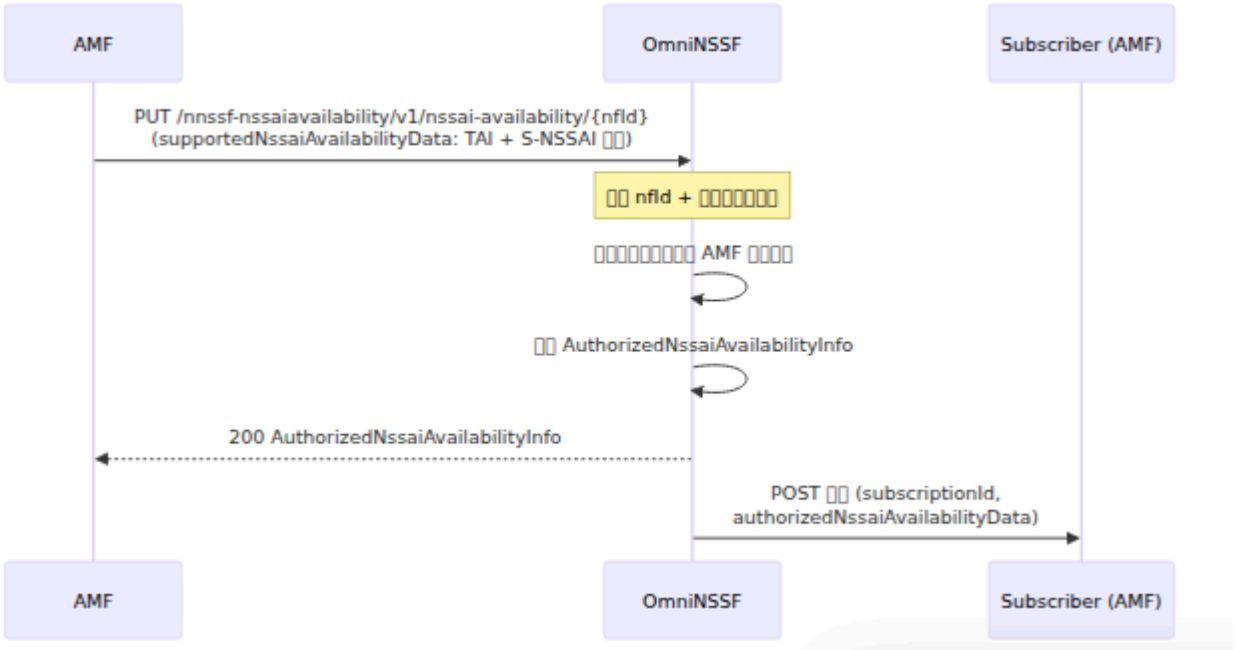
当查询中提供 TAI 并且一个或多个 AMF 报告了可用性时，NSSF 检查请求的 S-NSSAI 是否在该 TAI 可用，并相应地过滤允许的 NSSAI。如果尚未报告任何可用性，则所有配置的切片都被视为可用（宽松的默认设置；设置 `strict_availability: true` 以关闭失败）。当 AMF 提供 `supported-features` 值时，NSSF 返回与其自身功能集（当前为 NssaiAvailability 位）协商的交集。

作为 `nsiInformation` 返回的目标 NRF 通常来自匹配的 `nsi_list` 条目。当启用 `dynamic_nrf_discovery` 时，NSSF 会通过查询根 NRF 来在运行时解析切片服务 NRF，并在发现被禁用、不可达或未返回任何内容时回退到配置的 `nsi_list` NRF。有关配置键，请参见 配置参考。

NSSAI 可用性更新 (Nnssf_NSSAIAvailability)

AMF 推送它们每个 TAI 支持的 S-NSSAIs。NSSF 存储和授权这些信息，并通知任何订阅者变化。





对存储的可用性进行 PATCH 时，将 JSON Patch (RFC 6902) 操作应用于现有的每个 AMF 记录；对未知的 `{nfId}` 进行 PATCH 返回 `404`。PUT 和 PATCH 都会使用授权的可用性数据重新通知订阅者。

NSSAI 可用性订阅和通知

订阅者注册回调 URI；每当 PUT 或 PATCH 更新存储的可用性时，NSSF 会向每个订阅者发送变化通知。

通知是对每个订阅者的 `nfNssaiAvailabilityUri` 的 HTTP POST；通知携带 `subscriptionId` 和 `authorizedNssaiAvailabilityData`。每个通知在请求路径外交付，带有限重试：最多 `nssf_notification_max_attempts` 次尝试（默认 `3`），使用退避计划

`nssf_notification_backoff_ms`（默认 `200` 毫秒然后 `1000` 毫秒），之后通知将被丢弃并记录失败。由于交付是在请求路径外进行，因此慢或不可达的订阅者不会阻塞触发通知的可用性更新。

NRF 注册和心跳

在启动时，NSSF 向 NRF 注册其 NF 配置文件，宣传两个 SBI 服务，然后在定期心跳上重新注册。

注册状态反映在 `omni_nssf_nrf_registration_status` 指标中，并可以通过 OAM API 强制重新注册（`POST /oam/nrf/reregister`）。

← □□

OmniNSSF □□□□

常见问题

症状	可能原因	
切片选择返回 `400` `MANDATORY_IE_MISSING`	AMF 查询缺少 `nf-type` 或 `nf-id`	确认 AMF 发送了两
切片选择返回 `400` "未提供切片信息"	查询没有 `slice-info-request-for-pdu-session`、`slice-info-request-for-registration` 或 `snssai`	验证 AMF 为该过程
切片选择返回 `403` `SNSSAI_NOT_FOUND`	请求的 S-NSSAI 没有匹配的 NSI 条目	将 S-NSSAI 添加 `POST /oam/s`
响应中省略 `allowedNssaiList`	查询的 PLMN 没有 `allowed_nssai` 条目且没有家庭 PLMN 回退匹配	将 PLMN（或家庭 `allowed_nss`
切片在 TAI 上似乎不可用	对该 TAI 存在冲突的 AMF 可用性报告，且没有 S-NSSAI	检查 `GET /api/` 使用 `DELETE /oam/nssai_a` 使过期数据失效
响应中没有 `targetAmfSet`	对 S-NSSAI 键没有 `amf_set_mapping` 条目	将 `"sst-sd"` [
NRF 注册状态 `0`	NRF 无法访问或 `nrf_uri` 错误	验证 `NRF_URI`；`/oam/nrf/re`
可用性 PUT 被拒绝 `400`	缺少/空的 `supportedNssaiAvailabilityData`，或某个条目缺少 `tai` 或非空的 `supportedSnssaiList`	修正 AMF 的可用性
可用性 PATCH 返回 `404`	发送的 JSON Patch 针对没有存储可用性的 `nfId`	首先 PUT 可用性，[
订阅创建被拒绝 `400`	主体缺少 `nfNssaiAvailabilityUri`、`taiList` 或 `event`	提供所有三个必需的订[

症状	可能原因	
订阅 PATCH/DELETE 返回 `404`	未知的 `subscriptionId`	确认原始 `Locati` `subscriptio`
订阅者未被通知	回调 `nfNssaiAvailabilityUri` 无法访问	通知会在有限的退避时 （`nssf_notif` 次尝试，默认 `3`），[NSSF 日志以获取放