← Overview

OmniSCP Configuration

All operator-tunable settings are supplied through `config/runtime.exs`, each backed by an environment variable so values can be overridden at deploy time without editing the file. Values shown below are the defaults applied when the corresponding environment variable is unset.

```
import Config

config :omniscp,
  sbi_scheme: System.get_env("SBI_SCHEME", "http"),
  sbi_addr: System.get_env("SBI_ADDR", "127.0.0.17"),
  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"),
  heartbeat_interval:
String.to_integer(System.get_env("HEARTBEAT_INTERVAL", "10000")),
  prometheus_metrics_port:
String.to_integer(System.get_env("PROMETHEUS_PORT", "9568")),
  discovery_cache_ttl:
String.to_integer(System.get_env("DISCOVERY_CACHE_TTL", "60000")),
  lb_strategy: String.to_atom(System.get_env("LB_STRATEGY",
"round_robin")),
  max_retries: String.to_integer(System.get_env("MAX_RETRIES",
"2")),
  upstream_timeout:
String.to_integer(System.get_env("UPSTREAM_TIMEOUT", "5000"))

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

`:omniscp` parameters

Parameter	Type	Required	Default
`sbi_scheme`	String	No	`"http"`
`sbi_addr`	String	No	`"127.0.0.17"`
`sbi_port`	Integer	No	`7777`
`sbi_certfile`	String	No	*(unset)*
`sbi_keyfile`	String	No	*(unset)*

Parameter	Type	Required	Default
`nrf_uri`	String	No	`"http://127.0.0.1:7777`
`mcc`	String	No	`"999"`
`mnc`	String	No	`"70"`
`heartbeat_interval`	Integer (ms)	No	`10000`
`prometheus_metrics_port`	Integer	No	`9568`

Parameter	Type	Required	Default
`discovery_cache_ttl`	Integer (ms)	No	`60000`
`lb_strategy`	Atom	No	`round_robin`
`max_retries`	Integer	No	`2`

Parameter	Type	Required	Default
`upstream_timeout`	Integer (ms)	No	`5000`

NRF profile capacity/priority

These two values are part of the SCP's own NRF profile. They are **not** set in `runtime.exs` and have no dedicated environment variable, so they normally take their defaults, but they are runtime-tunable via the management API (`PATCH /api/oam/config/{id}`), which also triggers an NRF re-registration so the new value is advertised.

Parameter	Type	Required	Default	Env Var	Description
`nf_capacity`	Integer (0-65535)	No	`100`	(none, OAM only)	Capacity value advertised in the SCP's NRF profile. Consumers/peers may use it when weighting the SCP.
`nf_priority`	Integer (0-65535)	No	`0`	(none, OAM only)	Priority value advertised in the SCP's NRF profile (lower = higher priority).

> **Compiled fallback defaults.** The application also carries compiled fallback defaults used only if the `:omniscp` key is entirely absent, and these differ from the `runtime.exs` environment defaults above (for example `sbi_addr` falls back to `127.0.0.200`, `nrf_uri` to `http://127.0.0.10:7777`, and `max_retries` to `1`). In normal operation `runtime.exs` supplies every key, so the values in the tables above are the effective defaults.

Load-balancer and overload tunables

These `:omniscp` application-config keys govern automatic failure-based health tracking and overload-control shedding. They have **no environment variable** and are not exposed through the OAM API, so they are set via application config only. Each is read live at runtime, so a change to the running application config takes effect on the next request-path use without a re-compile. The defaults shown apply when the key is unset.

Parameter	Type	Required	Default	Description
`health_check_interval`	Integer (ms)	No	`15000`	Interval of the background health/recove sweep over producer instances.
`failure_threshold`	Integer	No	`3`	Consecutive forwarding failures before an instance is marked unhealthy and taken out of rotation.
`recovery_cooldown_ms`	Integer (ms)	No	`30000`	Cooldown after which an unhealthy instance is eligible to return to rotation.
`oci_reduction_threshold`	Integer	No	`50`	Overload-control threshold governing when an overloaded producer (per its `3gpp-Sbi-Oci` reduction metric) is she from the

Parameter	Type	Required	Default	Description
				candidate set before load-balancer selection. See Operations → Routing and Discovery.

SBI listener tuning

The SBI proxy is served over HTTP/2. The following transport parameters are fixed in the application (not environment-tunable) and are documented here because they govern the SCP's concurrency behaviour under load:

Setting	Value	Description
`max_concurrent_streams`	`5000`	Maximum concurrent HTTP/2 streams per inbound connection. Caps how many in-flight SBI requests a single consumer connection can multiplex through the proxy.
`num_acceptors`	`100`	Size of the SBI listener's acceptor pool for accepting new inbound TCP connections.

Outbound (upstream) requests to producers use a shared Finch connection pool provided by the common Omni5G runtime, reused across all forwarded traffic and keyed by producer host. Its pool sizing is not exposed as an OmniSCP setting, so upstream concurrency is governed by that shared pool together with `upstream_timeout` (which bounds each individual forwarded request) and the inbound `max_concurrent_streams` limit above. To scale a busy SCP, the levers you control are `upstream_timeout` (shorter timeouts free connections back to the pool sooner) and `discovery_cache_ttl` (a higher hit ratio removes NRF

round-trips from the request path); `max_concurrent_streams` caps how much a single consumer connection can multiplex through the proxy.

Management / OAM API

OmniSCP exposes a management/OAM API over **HTTPS on port 8443** for read-only status and live operational control. These endpoints do not carry subscriber traffic; restrict access to the operations network. All paths below are served under the `/api` base path of the management listener.

Status endpoints (read-only)

Method	Path	Description
`GET`	`/api/status/api`	API service status.
`GET`	`/api/status/license`	License status.
`GET`	`/api/status/nrf`	NRF registration status for the SCP.
`GET`	`/api/status/nf`	SCP NF status: SBI address/port, discovery cache size, active `lb_strategy`, and per-instance load-balancer health.
`GET`	`/api/api/cache`	Discovery cache contents: per entry the target NF type, service name, instance count, and remaining TTL.
`GET`	`/api/api/health`	Load-balancer health: per tracked instance the healthy flag and failure count, plus healthy/unhealthy totals.
`GET`	`/api/api/statistics`	SCP statistics: discovery cache size and tracked/healthy/unhealthy instance counts.
`GET`	`/api/oam/config`	Current effective runtime configuration (SBI, NRF, cache TTL, `lb_strategy`, `max_retries`, `upstream_timeout`, `nf_capacity`, `nf_priority`, log level).
`GET`	`/api/oam/cache`	Discovery cache view (as `/api/api/cache`).

Method	Path	Description
`GET`	`/api/oam/upstream/drain`	Lists currently drained upstream NF instances.

Control endpoints

Method	Path	Description
`PATCH`	`/api/oam/config/{id}`	Update selected runtime settings live: `discovery_cache_ttl`, `lb_strategy`, `max_retries`, `upstream_timeout`, `nf_capacity`, `nf_priority`, `log_level`. Values are validated (`lb_strategy` must be one of `round_robin`, `weighted`, `priority`; the integer keys must be integers). Changing `nf_capacity`/`nf_priority` also triggers an NRF re-registration.
`DELETE`	`/api/oam/cache/{id}`	Flush the discovery cache for a specific NF type, or pass `all` to flush the entire cache.
`POST`	`/api/oam/upstream/drain`	Drain an upstream so the proxy stops selecting it. Body: `{"nf_instance_id": "..."}` (alias `"target"`), optional `"reason"`.
`DELETE`	`/api/oam/upstream/drain/{id}`	Undrain an upstream by NF instance ID, making it eligible for selection again.
`POST`	`/api/oam/log_level`	Change the active log level at runtime. Body: `{"level":`

Method	Path	Description
		`"..."}` (`emergency`, `alert`, `critical`, `error`, `warning`/`warn`, `notice`, `info`, `debug`).
`POST`	`/api/oam/nrf/reregister`	Force an immediate re-registration of the SCP with the NRF.

Upstream drain is administrative state, separate from automatic failure-based health tracking. Drained instances are filtered out of the candidate list *before* load-balancer selection, so a drained producer never receives forwarded traffic. If **every** candidate for a service is drained, the proxy has no usable next hop and requests to that service return `504 NF_DISCOVERY_FAILURE` (see Operations → Proxy error responses and Troubleshooting).

Logging

Parameter	Type	Required	Default	Descript
`format`	Tuple	No	`{OmniLogger.JsonFormatter, :format}`	Formats lines as structure JSON via shared `OmniLogg` formatte suitable f ingestion a log pipeline.
`metadata`	Atom / List	No	`:all`	Log metadata emit. `:al` includes every metadata key attached a log ent

The active log level can be changed at runtime without a restart via the management API, either `POST /api/oam/log_level` or `PATCH /api/oam/config/{id}` with a `log_level` value.

← Overview

OmniSCP Metrics

Metrics are exposed on the Prometheus listener (`prometheus_metrics_port`, default `9568`) at `/metrics`. Result-tag values on proxy metrics are: `success` (2xx/3xx), `client_error` (4xx), `server_error` (5xx), `error` (connection/timeout).

Proxy metrics

Metric: `omni_scp_proxy_requests_count` **Type**: Counter **Description**: Per-request proxy outcome count. **Labels**: `target_nf_type`, `result`

Metric: `omni_scp_proxy_requests_duration_ms` **Type**: Summary **Description**: Per-request proxy duration in milliseconds. **Labels**: `target_nf_type`

Metric: `omni_scp_proxy_requests_total` **Type**: Counter **Description**: Total proxied requests by target NF type and result. **Labels**: `target_nf_type`, `result`

Metric: `omni_scp_proxy_request_duration_ms` **Type**: Histogram (distribution) **Description**: Proxy request duration. Buckets: 1, 5, 10, 25, 50, 100, 250, 500, 1000 ms. **Labels**: `target_nf_type`

The SCP is a stateless forwarding proxy and keeps no per-association state, so it exposes no "active associations" metric. Live activity is observed through the proxy request counter above (rate, error ratio) and the load-balancer health figures on the OAM API.

```
# Proxied request rate by target NF type
sum by (target_nf_type) (rate(omni_scp_proxy_requests_total[5m]))

# Server-error ratio across all producers
sum(rate(omni_scp_proxy_requests_total{result="server_error"}
[5m]))
  / sum(rate(omni_scp_proxy_requests_total[5m]))

# p95 proxy latency by NF type
histogram_quantile(0.95, sum by (le, target_nf_type)
(rate(omni_scp_proxy_request_duration_ms_bucket[5m])))
```

Discovery cache metrics

Metric: `omni_scp_discovery_cache_hits` **Type**: Counter **Description**: Discovery cache hits, per service. **Labels**: `target_nf_type`, `service_name`

Metric: `omni_scp_discovery_cache_misses` **Type**: Counter **Description**: Discovery cache misses, per service. **Labels**: `target_nf_type`, `service_name`

Metric: `omni_scp_cache_hits_total` **Type**: Counter **Description**: Aggregate discovery cache hits. **Labels**: none

Metric: `omni_scp_cache_misses_total` **Type**: Counter **Description**: Aggregate discovery cache misses. **Labels**: none

```
# Overall cache hit ratio (target > 0.9 in a stable topology)
sum(rate(omni_scp_cache_hits_total[5m]))
  / (sum(rate(omni_scp_cache_hits_total[5m])) +
sum(rate(omni_scp_cache_misses_total[5m])))

# Miss rate per service (high values mean frequent NRF re-
discovery)
sum by (target_nf_type, service_name)
(rate(omni_scp_discovery_cache_misses[5m]))
```

NRF registration metric

Metric: `omni_scp_nrf_registration_status` **Type**: Gauge **Description**: SCP registration status with the NRF (`1` = registered, `0` = not registered). **Labels**: `nf_type`

```
# Alert if the SCP loses its NRF registration
min(omni_scp_nrf_registration_status) < 1
```

BEAM VM metrics

The runtime also exports host/VM gauges: `beam_memory_total`, `beam_memory_processes`, `beam_memory_processes_used`, `beam_memory_system`, `beam_memory_atom`, `beam_memory_atom_used`, `beam_memory_binary`, `beam_memory_code`, `beam_memory_ets` (all bytes), and `beam_processes_count`, `beam_ports_count`, `beam_atom_count`, `beam_vm_uptime` (seconds). These are useful for capacity and leak monitoring of the SCP process itself.

> Metric names above use Prometheus dot-to-underscore normalization; histogram series expose the usual `_bucket`, `_sum`, and `_count` suffixes.

← Overview

OmniSCP Operations

This guide covers the SCP's 3GPP role, the interfaces and endpoints it exposes, its request-routing and discovery behaviour, and the key runtime procedures with sequence diagrams. For every tunable setting see Configuration; for metrics see Metrics; for failure handling see Troubleshooting.

3GPP Role and Specification References

The SCP provides **indirect communication** as defined in TS 23.501. In **Model C**, the consumer performs discovery itself (via the NRF) but delegates request forwarding to the SCP. In **Model D**, the consumer delegates both discovery and forwarding to the SCP: it sends the request with `3gpp-Sbi-Discovery-*` parameters and the SCP performs the NRF discovery on its behalf. OmniSCP supports both models, plus direct forwarding when the consumer already knows the target.

Specification	Section	Relevance
TS 23.501	§6.3.1, Annex E	SCP role and the indirect communication models (C and D) in the 5G SBA.
TS 23.501	§7.3	SCP request-handling behaviour within the SBA.
TS 29.500	§6.10	SBI routing via the SCP; the `3gpp-Sbi-*` routing/discovery headers.
TS 29.500	§6.10.3	`3gpp-Sbi-Discovery-*` header set carried on delegated-discovery requests.
TS 29.500	§6.10.3.2	`3gpp-Sbi-Target-apiRoot` header (direct routing target); primary service name selection.
TS 29.500	§6.10.3.3	`3gpp-Sbi-Producer-Id` response header (consumer-to-producer binding).
TS 29.500	§5.2.2.2	`User-Agent` header conveys the requester's NF type.
TS 29.510	§6.2	Nnrf_NFDiscovery, the NRF discovery service the SCP consumes.
TS 29.510	§6.1	Nnrf_NFManagement, SCP self-registration and NF status notifications.

Reference: TS 23.501, TS 29.500, TS 29.510.

Interfaces and Listeners

OmniSCP is a catch-all HTTP reverse proxy. NF consumers point their SBI traffic at the SCP's SBI listener instead of directly at producers. For each request, the

SCP resolves a target producer (from a request header, from a cached NRF discovery result, or by querying the NRF), selects one instance via the configured load-balancing strategy, forwards the request, and on failure retries against a different instance.

Listener	Default Port	Scheme	Purp
SBI proxy	`7777` (`sbi_port`)	HTTP / HTTPS (`sbi_scheme`)	Accepts SBI re consumers an NRF status no non-notificatic proxied to pro `sbi_scheme=h` `sbi_certfile` to terminate T listener.
Prometheus metrics	`9568` (`prometheus_metrics_port`)	HTTP	Serves the `/m` scrape endpoi Metrics.
Management / OAM API	`8443`	HTTPS	Read-only stat control endpoi Configuration Management A

The SBI listener is served over HTTP/2 with a per-connection stream limit and a fixed acceptor pool; those transport-tuning values are described in Configuration → SBI listener tuning.

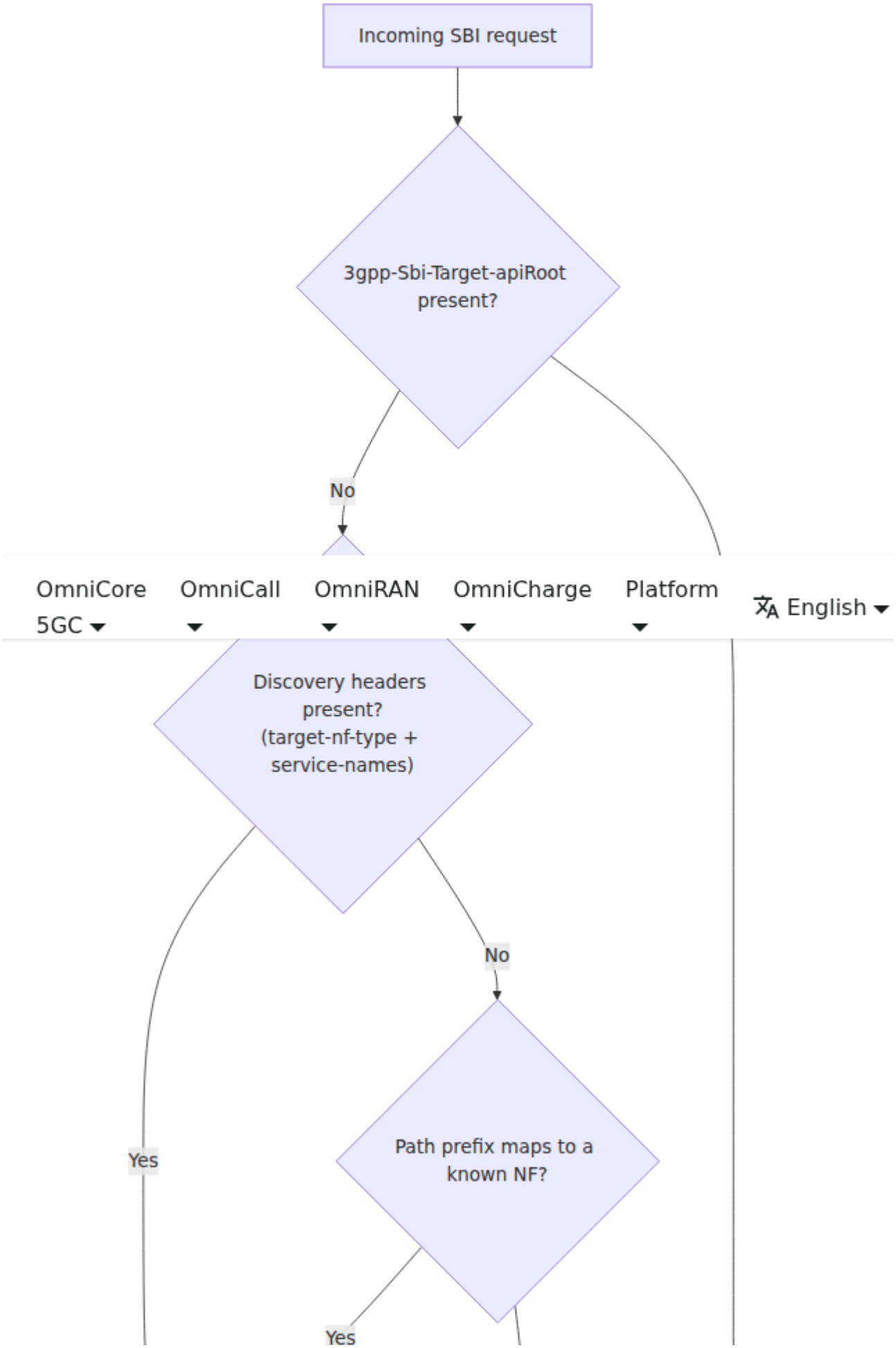
SBI endpoint table

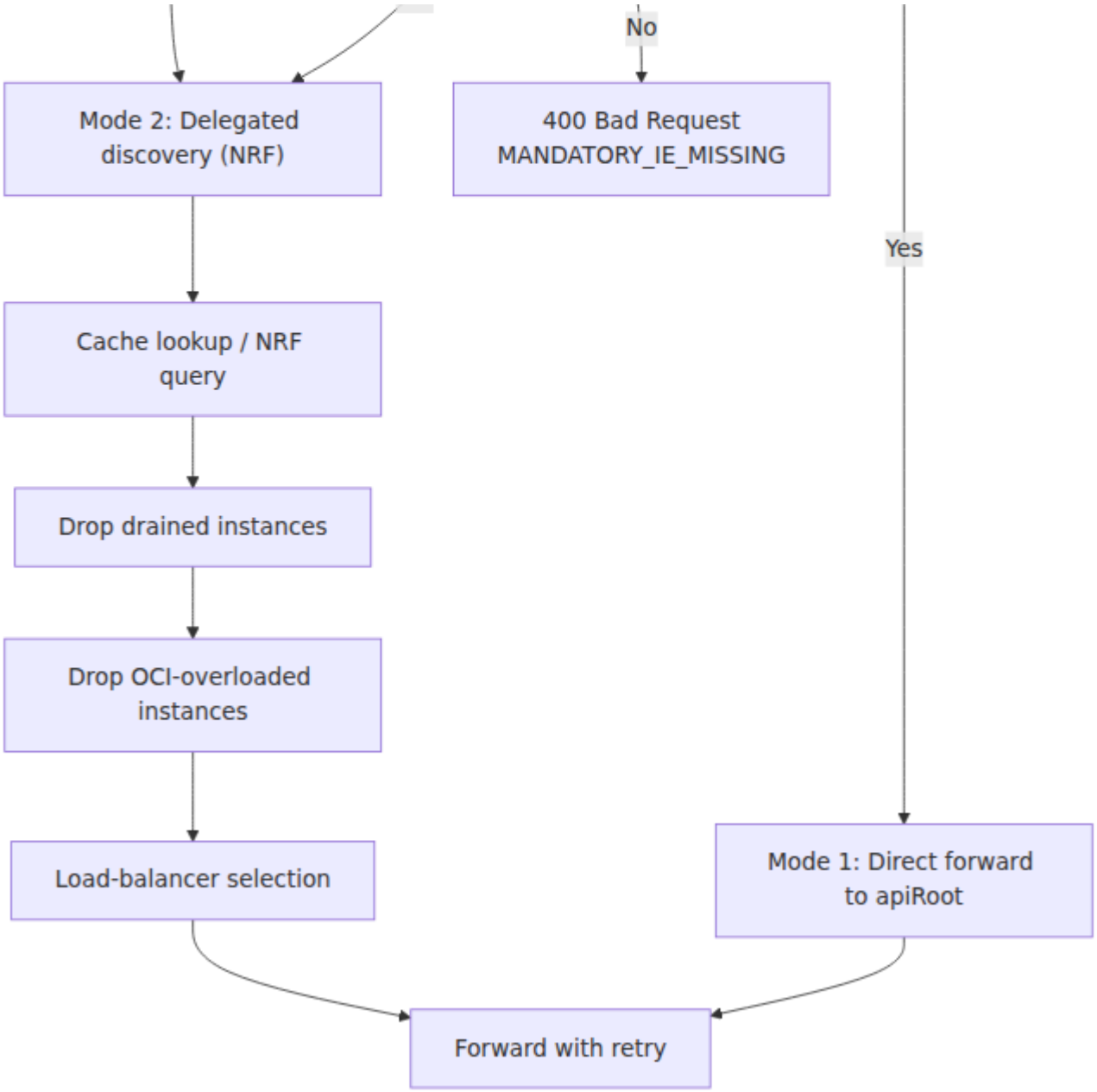
OmniSCP is transparent: exactly one path is handled locally, and every other request is proxied.

Method	Path	Handled locally	Description
`POST`	`/nnrf-nfm/v1/nf-status-notify`	Yes	Receives NRF NF status notifications. On `NF_DEREGISTERED` or `NF_PROFILE_CHANGED`, the discovery cache is invalidated for the single affected instance identified by its `nfInstanceId` (falling back to eviction of one NF type only if no instance id is present, and doing nothing if neither is present). The whole cache is never cleared. Returns `204 No Content`.
`*`	all other paths	No (proxied)	Forwarded to the resolved producer per the active routing mode.

Routing and Discovery

On every incoming request the SCP inspects the request headers and path to decide **how** to resolve the target producer. Three routing modes are evaluated in priority order:


Incoming SBI request
3gpp-Sbi-Target-apiRoot present?
No
OmniCore 5GC
OmniCall
OmniRAN
OmniCharge
Platform
English
Discovery headers present? (target-nf-type + service-names)
No
Yes
Path prefix maps to a known NF?
Yes



Mode 1: Direct forwarding

When the consumer sends `3gpp-Sbi-Target-apiRoot`, it already knows the producer's API root. The SCP strips the SCP-specific headers and forwards the request straight to that base URI with no NRF lookup, cache, or load balancing.

Mode 2: Delegated discovery (Models C/D)

When the consumer sends `3gpp-Sbi-Discovery-target-nf-type` and `3gpp-Sbi-Discovery-service-names`, the SCP resolves the producer set itself:

1. **Cache lookup**: keyed by `{target NF type, primary service name}`. The first name in `3gpp-Sbi-Discovery-service-names` is treated as the primary

service (TS 29.500 §6.10.3.2). On a hit within TTL, the cached instance list is used and no NRF query is made.

2. **NRF discovery**: on a miss, the SCP queries the NRF (Nnrf_NFDiscovery), passing through the discovery parameters it received: requester NF type, target PLMN list, requester S-NSSAI list, NF set ID, and target NF instance ID. The result is cached under the same key for `discovery_cache_ttl`.
3. **Drain filtering**: any instance whose `nfInstanceId` is administratively drained (see Configuration → Management API) is removed from the candidate set before selection.
4. **Overload-control shedding**: any instance currently signalling overload via its `3gpp-Sbi-Oci` (Overload Control Information) response header at or above `oci_reduction_threshold` for its validity period is removed from the candidate set (TS 29.500 §6.3). This is best-effort: if every candidate is overloaded the full set is kept rather than dropping the request.
5. **Load-balancer selection**: one instance is chosen from the remaining candidates using `lb_strategy`.
6. **Forward with retry**: the request is forwarded; on failure a different instance is tried, up to `max_retries`. On each response the producer's advertised `3gpp-Sbi-Oci` is recorded so subsequent requests can shed load from it.

Mode 3: Path-based inference

When no routing headers are present, the SCP infers the target NF type from the service-name prefix of the request path (path format `/<service-`

`name>/<version>/...`) and then proceeds exactly as Mode 2.

Path prefix	NF type
`nudm-`	UDM
`nausf-`	AUSF
`namf-`	AMF
`nsmf-`	SMF
`npcf-`	PCF
`nudr-`	UDR
`nnssf-`	NSSF
`nbsf-`	BSF
`nnrf-`	NRF
`nchf-`	CHF
`nnef-`	NEF
`naf-`	AF

If the path prefix is not in this table, the SCP returns `400 Bad Request` with cause `MANDATORY_IE_MISSING`.

Load-balancing strategies

The selection strategy is set by `lb_strategy` and applies to Mode 2/3 (discovered instance sets):

Strategy	Selection logic
`round_robin`	Cycles through the healthy candidate instances in order. A separate rotation counter is kept per `{nfType, serviceName}` group so different services do not interfere. This is the default.
`weighted`	Selects the instance with the best (lowest) `load - capacity` score from its NRF profile, preferring high-capacity, low-load producers. Missing `load` defaults to 50 and missing `capacity` to 100.
`priority`	Selects the instance with the lowest `priority` value (highest priority) from its NRF profile. Missing `priority` defaults to 65535. Suited to active/standby deployments.

> **Where the weights come from.** The `weighted` and `priority` strategies read `load`, `capacity`, and `priority` from **each producer's own NRF profile**; the SCP has no per-producer weighting configuration of its own. To bias selection you tune those values on the producers (or in the NRF), not on the SCP. The SCP's `nf_capacity`/`nf_priority` settings (Configuration → NRF profile capacity/priority) are a separate concept: they describe *the SCP itself* in its own profile so peers can weight the SCP, and have no effect on how the SCP weights producers.

Failure-based health tracking. Independently of the strategy, the load balancer tracks per-instance outcomes. An instance is marked unhealthy after **3 consecutive failures** and is excluded from selection; a background health sweep runs every **15 seconds** and returns an instance to service once a **30-second cooldown** has elapsed since its last failure. If every candidate is unhealthy, the SCP falls back to the full list rather than failing outright (favouring availability over strict exclusion). This automatic health tracking is distinct from operator-driven drain state.

These three thresholds (failure count, sweep interval, and cooldown) are not exposed as environment variables and default as shown above, but they are real `:omniscp` application-config keys (`failure_threshold`, `health_check_interval`, `recovery_cooldown_ms`) that are read live at runtime.

No OAM endpoint changes them at runtime. See Configuration → Load-balancer and overload tunables. To take a producer out of rotation on a schedule you control (rather than waiting for the failure threshold), use administrative drain instead. A failure for this purpose is a 5xx response or a connection/timeout error; any response with status below 500 (including 4xx) counts as a success and resets the instance's failure counter.

SBI / Proxy Behaviour

Headers consumed (and stripped before forwarding)

These control routing and are never forwarded to the producer.

Header	Purpose
`3gpp-Sbi-Target-apiRoot`	Mode 1 direct routing target.
`3gpp-Sbi-Discovery-target-nf-type`	NF type to discover (e.g. `UDM`).
`3gpp-Sbi-Discovery-service-names`	Comma-separated service names; the first is the primary (used as the cache key and NRF query service).
`3gpp-Sbi-Discovery-requester-nf-type`	Requester NF type; scopes the NRF query.
`3gpp-Sbi-Discovery-target-plmn-list`	Target PLMN list; passed through to the NRF query.
`3gpp-Sbi-Discovery-requester-snssai-list`	Requester S-NSSAI list; passed through to the NRF query.
`3gpp-Sbi-Discovery-nf-set-id`	NF set ID filter; passed through to the NRF query.
`3gpp-Sbi-Discovery-target-nf-instance-id`	Specific target NF instance ID; passed through to the NRF query.
`3gpp-Sbi-Discovery-requester-nf-instance-id`	Requester instance ID. Consumed and stripped, but not currently included in the NRF discovery query.

The `User-Agent` header is also inspected: per TS 29.500 §5.2.2.2 it begins with the requester's NF type (e.g. `AMF-...`), which is used as the requester NF type if no explicit discovery header supplies one. If neither is present, the SCP defaults the requester NF type to `SCP` for the NRF query.

Headers produced

Header	Purpose
`3gpp-Sbi-Producer-Id`	Added to the response, carrying the `nfInstanceId` of the producer that served the request, so the consumer can bind subsequent requests to the same producer (TS 29.500 §6.10.3.3).
`3gpp-Sbi-Lci`	Producer Load Control Information, relayed end to end from the producer's response so the consumer receives its load hint through the SCP (TS 29.500 §6.3). The SCP relays the producer value and does not synthesize its own.
`3gpp-Sbi-Load`	The `3gpp-Sbi-Lci` load companion header, relayed end to end from the producer's response together with `3gpp-Sbi-Lci` (TS 29.500 §6.3).

Hop-by-hop response headers (`transfer-encoding`, `connection`, `content-length`) are not forwarded back to the consumer.

Proxy error responses

When the SCP cannot complete an operation it returns an RFC 7807 `ProblemDetails` body per TS 29.500.

HTTP status	Cause	Condition
`400 Bad Request`	`MANDATORY_IE_MISSING`	No routing information: no target-apiRoot, no discovery headers, and the path prefix maps to no known service.
`502 Bad Gateway`	`TARGET_NF_NOT_REACHABLE`	No SBI URI could be resolved from a discovered NF profile.
`504 Gateway Timeout`	`NF_DISCOVERY_FAILURE`	Discovery (NRF + cache) yielded no usable NF instances, including when **every** candidate instance for the service is administratively drained.
`500 Internal Server Error`	`SYSTEM_FAILURE`	Genuinely unexpected internal proxy error not covered by the more specific causes above.

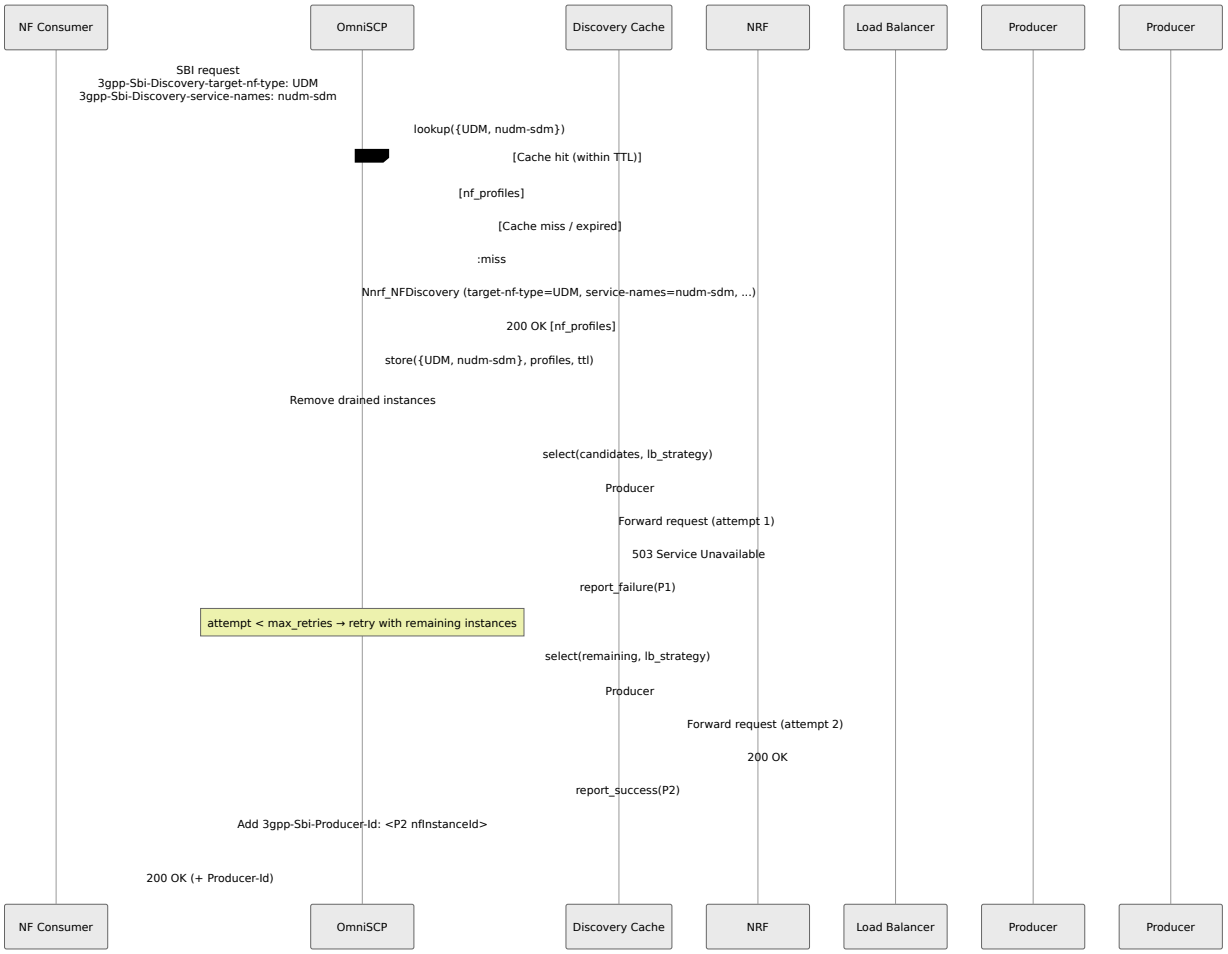
> **Note on the all-drained case.** If every candidate for a service is drained, the proxy has no usable next hop and returns `504 NF_DISCOVERY_FAILURE` (not `500`). Undrain at least one instance to restore service. See Troubleshooting.

Retry semantics. A producer response with status **< 500** (including 4xx) is treated as a valid answer and returned to the consumer as-is. A **5xx** response or a **connection/timeout error** counts as a failure: the failing instance is reported to the load balancer and removed from the candidate set, and a different instance is tried. Retrying continues until either a non-failure response is obtained, `max_retries` is reached, or the candidate list is exhausted, at which point the last response (or error) is returned.

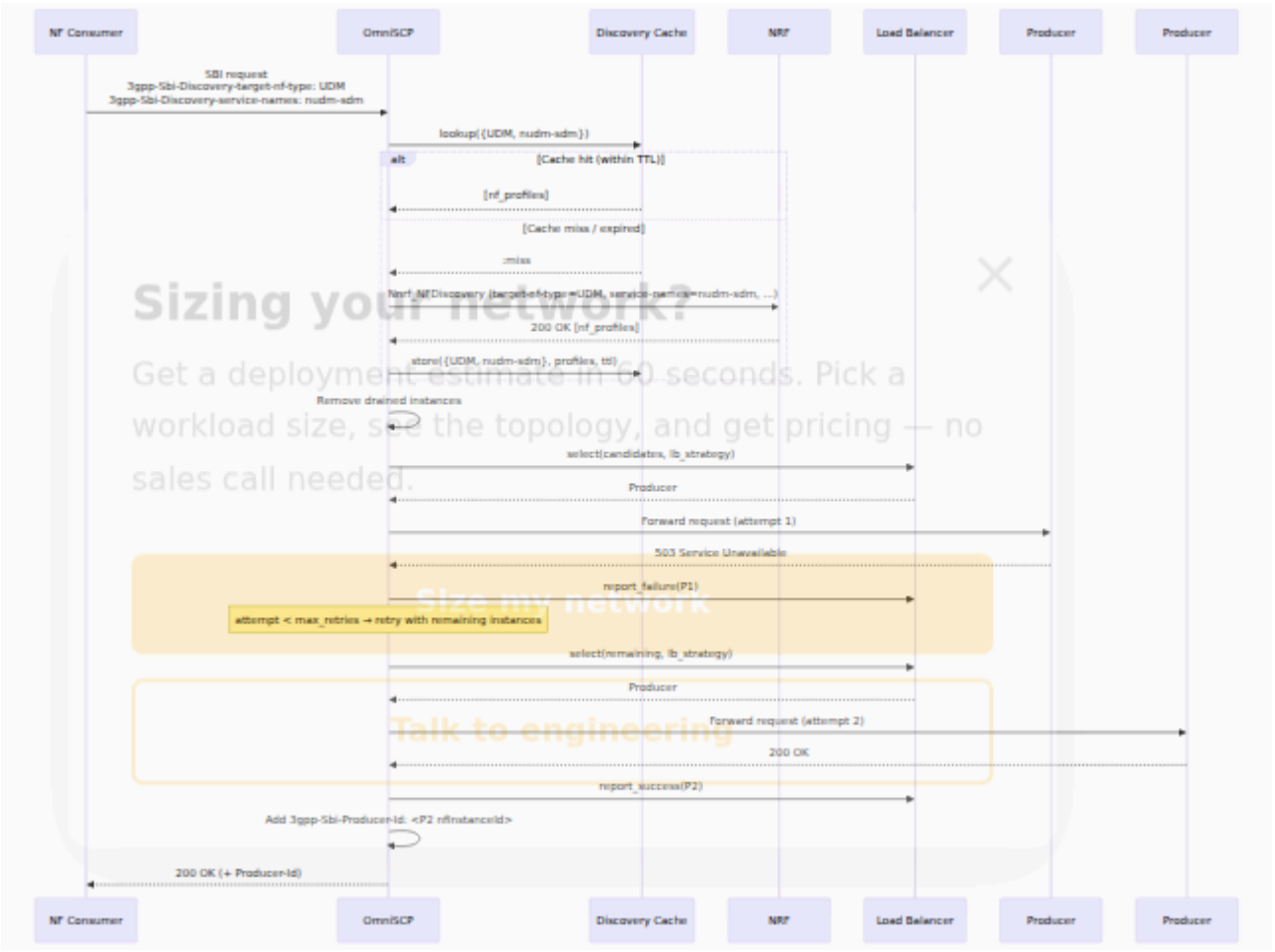
Key Procedures

Proxied, load-balanced request with a retry (Mode 2/3)

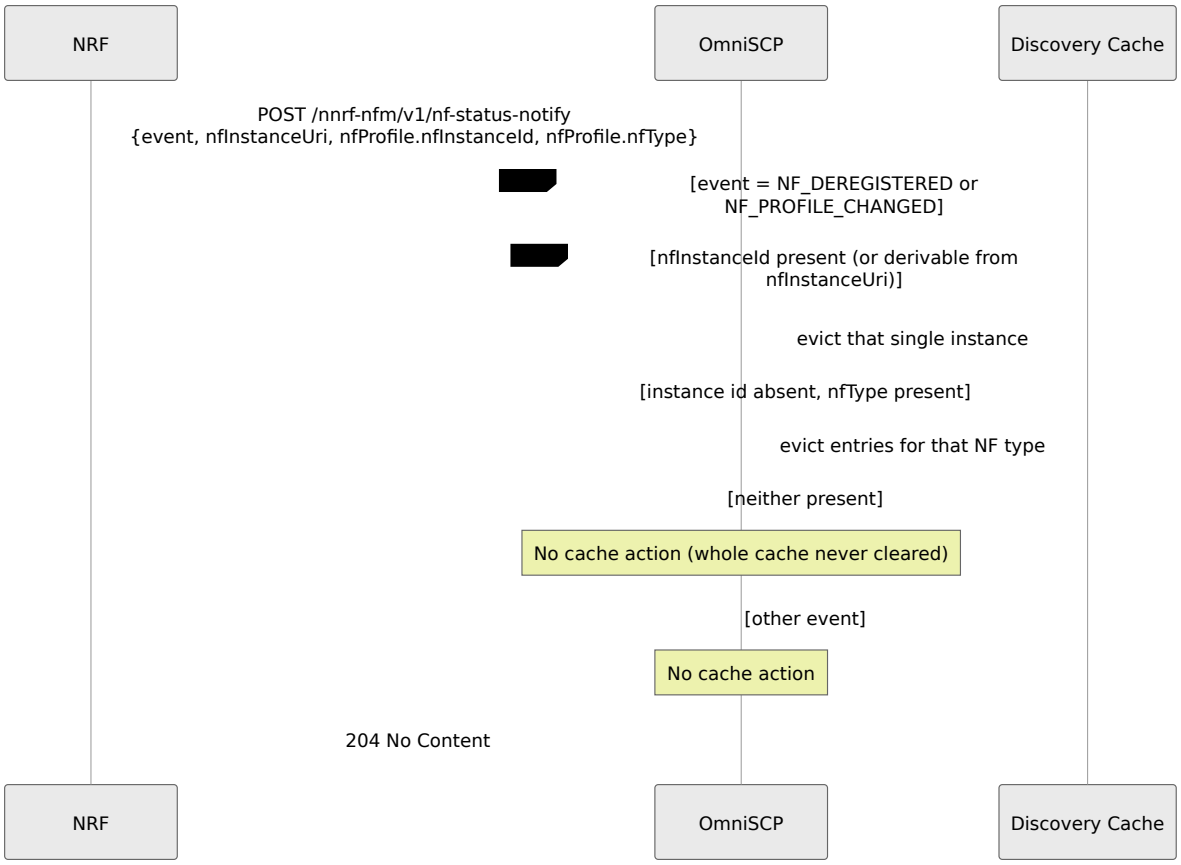
The sequence below shows delegated discovery, load-balancer selection, a first attempt that fails with a 5xx, and a successful retry against a second instance.



Direct forwarding (Mode 1)

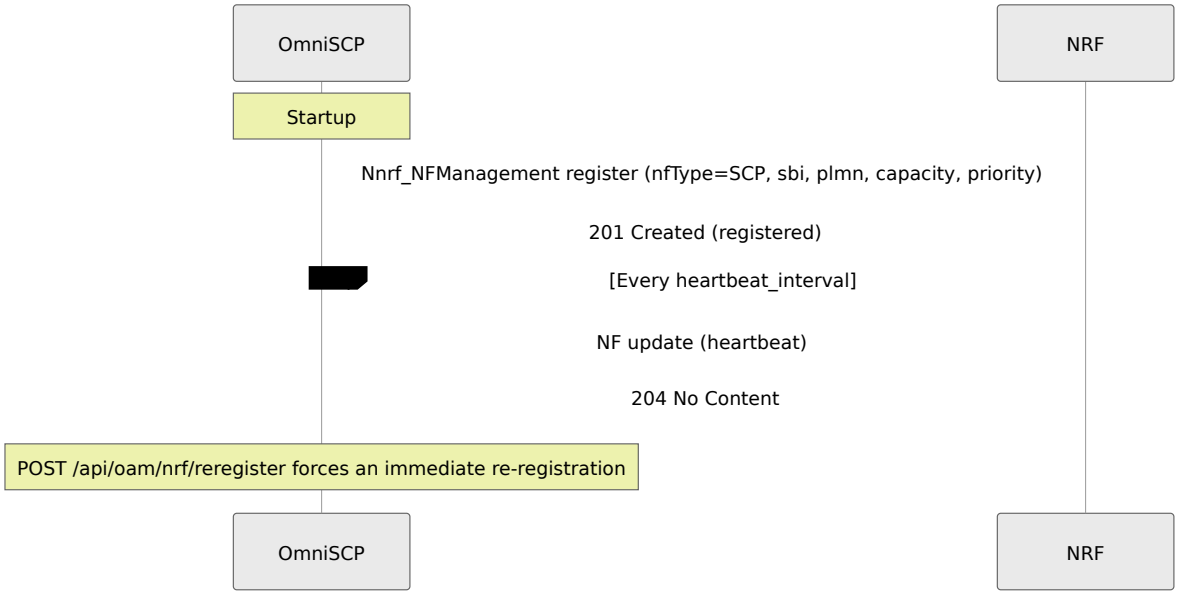


NRF status notification (cache invalidation)



NRF self-registration and heartbeat

The SCP registers itself with the NRF as an `SCP` NF at startup and keeps the registration alive with periodic heartbeats. The advertised profile carries the SBI address/port/scheme, the PLMN (`mcc`/`mnc`), and the NF `capacity`/`priority`. Registration can be forced immediately via the OAM API.

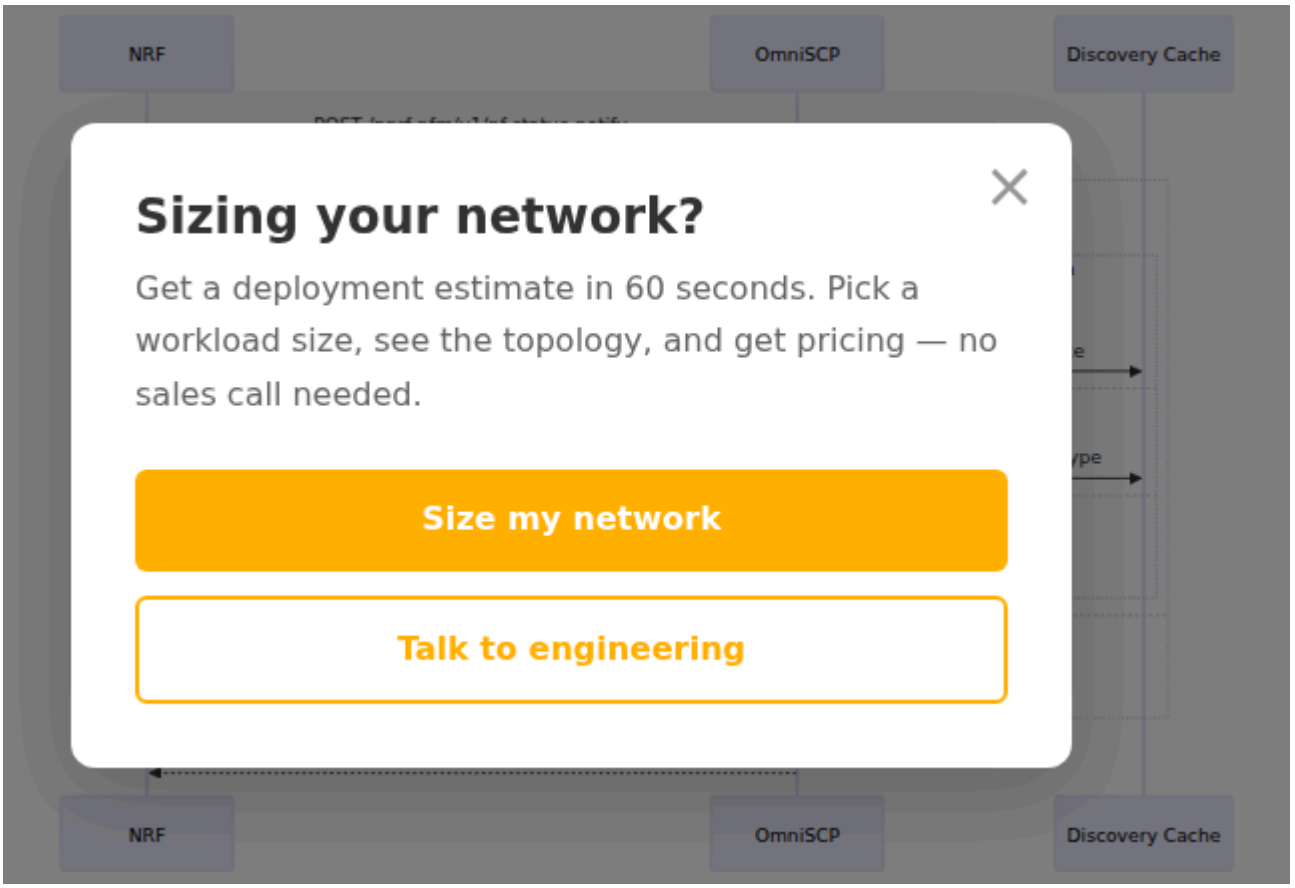


Administrative upstream drain

Drain is administrative state, separate from automatic failure-based health tracking. Drained instances are filtered out of the candidate list *before* load-balancer selection, so a drained producer never receives forwarded traffic. Use it to gracefully take a producer out of rotation (for maintenance) without deregistering it from the NRF.

Maintenance workflow. Obtain the target producer's `nfInstanceId` from the NRF (the value the producer registered with), then `POST /api/oam/upstream/drain` with that ID and an optional `reason`. In-flight requests already forwarded to the producer are not interrupted; only *new* selections skip it, so allow existing requests to complete before servicing the producer. Confirm it is out of rotation with `GET /api/oam/upstream/drain`, and when maintenance is done, `DELETE /api/oam/upstream/drain/{id}` to return it to service.

> **Drain state is in-memory.** Drain entries are held in an ETS table and are **not persisted across an SCP restart**: if the SCP process restarts while a producer is drained, that producer becomes selectable again and you must re-issue the drain. Plan maintenance windows accordingly, and re-check `GET /api/oam/upstream/drain` after any SCP restart.


NRF
OmniSCP
Discovery Cache
Sizing your network?
Get a deployment estimate in 60 seconds. Pick a workload size, see the topology, and get pricing — no sales call needed.
Size my network
Talk to engineering
NRF
OmniSCP
Discovery Cache

← Overview

OmniSCP Troubleshooting

Error causes and status codes referenced here are described in Operations → Proxy error responses. Configuration keys and OAM endpoints are in Configuration; metrics in Metrics.

400 Bad Request: MANDATORY_IE_MISSING

Symptoms: Consumer receives `400` with cause `MANDATORY_IE_MISSING`.

Possible causes:

- The request carried neither `3gpp-Sbi-Target-apiRoot` nor the `3gpp-Sbi-Discovery-target-nf-type` + `3gpp-Sbi-Discovery-service-names` pair.
- Relying on path-based inference (Mode 3) for a service prefix not in the built-in map.

Resolution:

1. Confirm the consumer sends a target-apiRoot or both discovery headers.
2. If depending on path inference, verify the path prefix is in the Mode 3 table. For a prefix not in that table, send explicit discovery headers.

504 Gateway Timeout: NF_DISCOVERY_FAILURE

Symptoms: Consumer receives `504` with cause `NF_DISCOVERY_FAILURE`.

Possible causes:

- The NRF returned no instances for the requested NF type/service.
- The NRF is unreachable.

Resolution:

1. Verify `nrf_uri` and network reachability to the NRF.
2. Confirm the target NF type is registered in the NRF.
3. Check whether a recent NRF status notification invalidated the cache for the NF type and the producer has not re-registered.

504 Gateway Timeout: NF_DISCOVERY_FAILURE (all instances drained)

Symptoms: Consumer receives `504` with cause `NF_DISCOVERY_FAILURE` for a service that has registered producers.

Possible causes:

- Every candidate instance for the service is administratively drained, so the proxy has no usable next hop. This condition maps to `504 NF_DISCOVERY_FAILURE`, the same status as discovery yielding no instances.

Resolution:

1. Check `GET /api/oam/upstream/drain`. If all instances for the service are drained, undrain the required ones via `DELETE /api/oam/upstream/drain/{id}`.
2. If no instances are drained and the service is registered, treat it as a discovery-failure case (see the NF_DISCOVERY_FAILURE section above).

500 Internal Server Error: SYSTEM_FAILURE

Symptoms: Consumer receives `500` with cause `SYSTEM_FAILURE`.

Possible causes:

- A genuinely unexpected internal proxy error not covered by the more specific causes (`400`, `502`, `504`) above.

Resolution:

1. Review the SCP logs for the internal error detail.

502 Bad Gateway: TARGET_NF_NOT_REACHABLE

Symptoms: Consumer receives `502` with cause `TARGET_NF_NOT_REACHABLE`.

Possible causes:

- No SBI URI could be resolved from a discovered NF profile.
- All selected producers returned connection errors after retries.

Resolution:

1. Confirm producers are running and reachable at the SBI addresses in their NRF profiles.
2. If producers are slow, increase `upstream_timeout`.
3. Review `max_retries`; `0` turns a single failure into an immediate error.
4. Check `GET /api/api/health` for instances marked unhealthy after repeated failures.

Upstream timeouts / high proxy latency

Symptoms: Elevated `omni_scp_proxy_request_duration_ms`; requests intermittently fail with `error` result.

Possible causes:

- Producers responding slower than `upstream_timeout`.
- Low cache hit ratio forcing an NRF round-trip on many requests.

Resolution:

1. Inspect the latency histogram and the per-NF-type error rate.
2. Compare `omni_scp_cache_hits_total` vs `omni_scp_cache_misses_total`; a high miss rate means frequent NRF discovery, so consider increasing `discovery_cache_ttl`.
3. Tune `upstream_timeout` to match realistic producer response times. Remember a timed-out attempt adds the full timeout to end-to-end latency before a retry.

Stale routing / discovery cache staleness

Symptoms: Requests are forwarded to producers that have moved or restarted, causing errors until entries age out.

Possible causes:

- A producer changed address or restarted without deregistering from the NRF, so the SCP holds a stale profile until TTL expiry.

Resolution:

1. Reduce `discovery_cache_ttl` to shorten the staleness window.
2. Ensure producers deregister from the NRF on shutdown so the SCP receives an `NF_DEREGISTERED` notification and invalidates the affected NF type.
3. Flush explicitly via `DELETE /api/oam/cache/{nf_type}` (or `DELETE /api/oam/cache/all`).

NRF registration not maintained

Symptoms: `omni_scp_nrf_registration_status` reads `0`.

Possible causes:

- Incorrect `nrf_uri` or NRF unreachable.
- PLMN mismatch (`mcc`/`mnc`) against NRF configuration.

Resolution:

1. Verify `nrf_uri` and connectivity.
2. Confirm `mcc`/`mnc` match the NRF PLMN configuration.
3. Force a re-registration with `POST /api/oam/nrf/reregister` and review startup logs.