

# OmniSMF Configuration Reference

Operator-writable configuration is supplied through `config/runtime.exs`, which is driven largely by OS environment variables. Most values below have a built-in default and an environment variable that overrides it before the SMF starts. A few keys are set through application configuration only and have no environment variable (`chf_uri`, the `charging` block, `pgw_c_addr`, and the `pco` block); these are called out where they appear. The compiled application environment key is `:omnismf`.

```
config :omnismf,
  sbi_scheme: "http",           # SBI_SCHEME
  sbi_addr:   "127.0.0.4",      # SBI_ADDR
  sbi_port:   7777,             # SBI_PORT
  nrf_uri:    "http://127.0.0.1:7777", # NRF_URI
  udm_uri:    "http://127.0.0.12:7777", # UDM_URI
  pcf_uri:    "http://127.0.0.13:7777", # PCF_URI
  amf_uri:    "http://127.0.0.5:7777", # AMF_URI
  upf_addr:   "10.179.1.24",     # UPF_ADDR
  upf_pfcps_port: 8805,         # UPF_PFCP_PORT
  pfcps_addr: "0.0.0.0",        # PFCP_ADDR
  mcc: "999",                  # MCC
  mnc: "70",                   # MNC
  dns: ["8.8.8.8", "8.8.4.4"],  # DNS
  dns6: ["2001:4860:4860::8888"], # DNS6
  mtu: 1400,                   # MTU
  prometheus_metrics_port: 9568, # PROMETHEUS_PORT
  heartbeat_interval: 10_000,    # HEARTBEAT_INTERVAL
  ue_ip_pool: [subnet_map: %{default: ["10.45.0.0/16"]} ] #
UE_IP_SUBNET_MAP / UE_IP_POOL
```

# SBI, Peer and PLMN Parameters

| Parameter  | Type    | Required | Default                  |
|------------|---------|----------|--------------------------|
| sbi_scheme | string  | No       | "http"                   |
| sbi_addr   | string  | No       | "127.0.0.4"              |
| sbi_port   | integer | No       | 7777                     |
| nrf_uri    | string  | No       | "http://127.0.0.1:7777"  |
| udm_uri    | string  | No       | "http://127.0.0.12:7777" |
| pcf_uri    | string  | No       | "http://127.0.0.13:7777" |
| amf_uri    | string  | No       | "http://127.0.0.5:7777"  |

| Parameter          | Type            | Required | Default       |
|--------------------|-----------------|----------|---------------|
|                    |                 |          |               |
| upf_addr           | string          | No       | "10.179.1.24" |
| upf_pfcp_port      | integer         | No       | 8805          |
| pfcp_addr          | string          | No       | "0.0.0.0"     |
| mcc                | string          | No       | "999"         |
| mnc                | string          | No       | "70"          |
| heartbeat_interval | integer<br>(ms) | No       | 10000         |

# N26 Interworking

| Parameter  | Type   | Required | Default      | Env Var                           | Description                                                                                                                                                                                                                                                |
|------------|--------|----------|--------------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pgw_c_addr | string | No       | "127.0.0.11" | None<br>(application config only) | The paired standalone PGW-C's S5/S8-C address, returned the EPS v during N: interwork so the MME/SGW Create Session lands on PGW-C th fronts the same UP anchor at this SMF. via the :omnism1 applicatio environn there is r environn variable t this key. |

# N40 Converged Charging (Nchf, TS 32.290 / TS 32.291)

The SMF can act as a converged-charging consumer toward the CHF over N40 (Nchf\_ConvergedCharging). When enabled, it opens a Charging Data Request as each PDU session is established, updates it during the session from N4 URR usage reports, and closes it on release. The CHF is resolved via NRF discovery first, falling back to the configured `chf_uri` when discovery does not resolve one. Converged charging is disabled by default; while disabled the charging path is a no-op and no CHF calls are made.

The top-level `chf_uri` key and the `charging` block are set through the `:omnismf` application configuration only; neither has an environment variable.

```
config :omnismf, :charging,  
  enabled: false,  
  chf_uri: "http://127.0.0.20:7777"
```

| Parameter                     | Type    | Required | Default                                         | Env Var                           |
|-------------------------------|---------|----------|-------------------------------------------------|-----------------------------------|
| <code>chf_uri</code>          | string  | No       | <code>"http://127.0.0.20:7777"</code>           | None<br>(application config only) |
| <code>charging enabled</code> | boolean | No       | <code>false</code>                              | None<br>(application config only) |
| <code>charging chf_uri</code> | string  | No       | (falls back to top-level <code>chf_uri</code> ) | None<br>(application config only) |

| Parameter | Type | Required | Default | Env Var |
|-----------|------|----------|---------|---------|
|           |      |          |         |         |

## Session-AMBR toward the PCF

When creating the SM policy association, the subscribed Session-AMBR (read from the UDM DNN configuration `sessionAmbr` field) is sent to the PCF under **both** the spec field name `subsSessAmbr` (TS 29.512 `SmPolicyContextData`) and the legacy key `sessionAmbr`. Sending only `sessionAmbr` leaves `subsSessAmbr` unset, causing the PCF to fall back to its ~1000 Kbps default and cap every session. No operator configuration governs this; it is documented here so operators understand which field the PCF must honour.

## UE-Facing PCO Defaults (flat keys)

These flat keys populate the Protocol Configuration Options returned to the UE in the PDU Session Establishment Accept. They are superseded by the structured `:pco` block below when that is present.

| Parameter         | Type            | Required | Default                               | Env Var           |                                  |
|-------------------|-----------------|----------|---------------------------------------|-------------------|----------------------------------|
| <code>dns</code>  | list of strings | No       | <code>["8.8.8.8", "8.8.4.4"]</code>   | <code>DNS</code>  | IP<br>s<br>a<br>t<br>(<br>s<br>t |
| <code>dns6</code> | list of strings | No       | <code>["2001:4860:4860::8888"]</code> | <code>DNS6</code> | IP<br>s<br>a<br>t<br>(<br>s<br>t |
| <code>mtu</code>  | integer         | No       | <code>1400</code>                     | <code>MTU</code>  | IP<br>M<br>a<br>t                |

# UE IP Pool

| Parameter  | Type                    | Required | Default                         | Env Var                       |
|------------|-------------------------|----------|---------------------------------|-------------------------------|
| ue_ip_pool | keyword<br>(subnet_map) | No       | %{default:<br>["10.45.0.0/16"]} | UE_IP_SUBNET_MAP / UE_IP_POOL |

The `UE_IP_SUBNET_MAP` JSON is keyed by exact DNN name, `^regex` pattern, or `"default"`; each value is a list of IPv4 and/or IPv6 CIDRs. Matching precedence is **exact DNN** → **regex** → **default**. Example:

```
{
  "internet":    ["10.45.0.0/16", "2001:db8:45::/64"],
  "ims":         ["10.47.0.0/24"],
  "^enterprise": ["10.99.0.0/16"],
  "default":     ["10.45.0.0/16"]
}
```

If neither `UE_IP_SUBNET_MAP` nor `UE_IP_POOL` is set, the default pool is `10.45.0.0/16`. See [UE IP Address Allocation](#) for the allocation and static-IP reservation behaviour.

## P-CSCF and Protocol Configuration Options (:pco)

For VoNR/IMS the SMF returns the P-CSCF address (and the other PCO items: DNS, NBNS, link MTU) to the UE inside the PDU Session Establishment Accept ePCO. This block uses the **same layout and selection logic as the 4G PGW-**

**C**, so an operator configures both nodes identically. When present, `:pco` values take precedence over the flat `dns/dns6/mtu` keys above. This block is set in the `:omnismf` application configuration only and has no environment variable.

```
config :omnismf, :pco,  
  p_cscf_ipv4_address_list: ["10.179.4.165"],  
  p_cscf_ipv6_address_list: [],  
  p_cscf_discovery_enabled: true,  
  p_cscf_discovery_fqdn:  
    "pcscf.ims.mnc070.mcc999.3gppnetwork.org",  
  p_cscf_discovery_dns_server: "10.179.2.10",  
  p_cscf_discovery_timeout_ms: 5000,  
  primary_dns_server_address: "8.8.8.8",  
  secondary_dns_server_address: "8.8.4.4",  
  primary_nbns_server_address: nil,  
  secondary_nbns_server_address: nil,  
  ipv4_link_mtu_size: 1400
```

| Parameter                             | Type            | Required |                   |
|---------------------------------------|-----------------|----------|-------------------|
| <code>p_cscf_ipv4_address_list</code> | list of strings | No       | <code>[]</code>   |
| <code>p_cscf_ipv6_address_list</code> | list of strings | No       | <code>[]</code>   |
| <code>p_cscf_discovery_enabled</code> | boolean         | No       | <code>true</code> |

| Parameter                                 | Type            | Required |                              |
|-------------------------------------------|-----------------|----------|------------------------------|
|                                           |                 |          |                              |
| <code>p_cscf_discovery_fqdn</code>        | string          | No       | <code>pcscf.mnc{MNC}.</code> |
|                                           |                 |          |                              |
| <code>p_cscf_discovery_dns_server</code>  | string          | No       | system resolver              |
|                                           |                 |          |                              |
| <code>p_cscf_discovery_timeout_ms</code>  | integer<br>(ms) | No       | <code>5000</code>            |
|                                           |                 |          |                              |
| <code>primary_dns_server_address</code>   | string          | No       | (flat <code>dns</code> )     |
|                                           |                 |          |                              |
| <code>secondary_dns_server_address</code> | string          | No       | (flat <code>dns</code> )     |

| Parameter                                  | Type            | Required |                          |
|--------------------------------------------|-----------------|----------|--------------------------|
| <code>primary_nbns_server_address</code>   | string  <br>nil | No       | <code>nil</code>         |
| <code>secondary_nbns_server_address</code> | string  <br>nil | No       | <code>nil</code>         |
| <code>ipv4_link_mtu_size</code>            | integer         | No       | (flat <code>mtu</code> ) |

## Legacy flat P-CSCF fallback keys

The flat keys `pcscf` and `pcscf6` are legacy fallbacks retained for parity with older PGW-C configurations. They are only consulted when the structured `:pco` P-CSCF address lists (`p_cscf_ipv4_address_list` / `p_cscf_ipv6_address_list`) are not set. The structured `:pco` form is preferred for new deployments. Each key accepts a single address string or a list of address strings. These keys are set through the `:omnismf` application configuration only and have no environment variable.

| Parameter           | Type                      | Required | Default         | Env Var                        | Description                                                                                                      |
|---------------------|---------------------------|----------|-----------------|--------------------------------|------------------------------------------------------------------------------------------------------------------|
| <code>pcscf</code>  | string or list of strings | No       | <code>[]</code> | None (application config only) | Legacy static IP fallback, used if <code>p_cscf_ip_v4</code> is unset. Preferred structured <code>:pcscf</code>  |
| <code>pcscf6</code> | string or list of strings | No       | <code>[]</code> | None (application config only) | Legacy static IP fallback, used if <code>p_cscf_ip_v6</code> is unset. Preferred structured <code>:pcscf6</code> |

# Metrics and Management Ports

| Parameter                                    | Type    | Required | Default           | Env Var                   |
|----------------------------------------------|---------|----------|-------------------|---------------------------|
| <code>prometheus_metrics_port</code>         | integer | No       | <code>9568</code> | <code>PROMETHEUS_F</code> |
| management API port ( <code>:api_ex</code> ) | integer | No       | <code>8443</code> | <code>API_PORT</code>     |
| Control Panel HTTPS port                     | integer | No       | <code>7443</code> | <code>CONTROL_PANE</code> |

The management API and Control Panel are served over TLS using the certificate at `priv/cert/omnitouch.crt` and key at `priv/cert/omnitouch.pem`. Only the listening ports are runtime-tunable via the environment variables above.

## Management / OAM API

The management API (TLS, default port `8443`, env `API_PORT`) exposes the following endpoints for operations and diagnostics. All management endpoints are mounted under the `/api` prefix.

| Method | Path                  | Purpose                                                                |
|--------|-----------------------|------------------------------------------------------------------------|
| GET    | /api/status/api       | API service status.                                                    |
| GET    | /api/status/nrf       | NRF registration status.                                               |
| GET    | /api/status/nf        | NF (SMF) status summary.                                               |
| GET    | /api/pdu_session      | List active PDU sessions.                                              |
| POST   | /api/pdu_session      | Create a PDU session (operational testing).                            |
| GET    | /api/pdu_session/{id} | Show a single PDU session.                                             |
| DELETE | /api/pdu_session/{id} | Delete a single PDU session.                                           |
| GET    | /api/upf              | UPF / PCF association view.                                            |
| POST   | /api/eps_handover     | Drive N26 EPS interworking (adopt / switch / lookup).                  |
| GET    | /api/ip_pool          | Overall IPv4 capacity plus a per-DNN pool summary.                     |
| GET    | /api/ip_pool/detail   | Per-DNN pools with each range's CIDR, totals, and allocated addresses. |

| Method    | Path                       | Purpose                                         |
|-----------|----------------------------|-------------------------------------------------|
| GET       | /api/statistics            | Session and operational statistics.             |
| GET       | /api/oam/config            | Read runtime configuration.                     |
| PATCH/PUT | /api/oam/config/{id}       | Update a runtime configuration value.           |
| GET       | /api/oam/resources         | Resource utilisation snapshot.                  |
| GET       | /api/oam/diagnostics       | Diagnostics snapshot.                           |
| GET       | /api/oam/peers             | SBI peer (UDM/PCF/AMF/NRF) status.              |
| GET       | /api/oam/pfcp_peers        | PFCP (UPF) peer status.                         |
| GET       | /api/oam/pfcp_session/{id} | Inspect a PFCP session.                         |
| POST      | /api/oam/upf               | Manage the UPF association.                     |
| POST      | /api/oam/pfcp/reassociate  | Re-trigger PFCP association setup with the UPF. |
| POST      | /api/oam/cache/clear       | Clear internal caches.                          |
| POST      | /api/oam/log_level         | Change the runtime log level without a restart. |

| Method | Path                                         | Purpose                                 |
|--------|----------------------------------------------|-----------------------------------------|
| POST   | <code>/api/oam/nrf/reregister</code>         | Force NRF re-registration.              |
| DELETE | <code>/api/oam/session/{id}</code>           | Drain a single session.                 |
| DELETE | <code>/api/oam/session/by_dnn/{id}</code>    | Drain sessions by DNN.                  |
| DELETE | <code>/api/oam/session/by_supi/{id}</code>   | Drain sessions by SUPI.                 |
| DELETE | <code>/api/oam/session/by_snssai/{id}</code> | Drain sessions by S-NSSAI.              |
| DELETE | <code>/api/oam/session/by_upf/{id}</code>    | Drain sessions by UPF.                  |
| POST   | <code>/api/oam/session/modify</code>         | Modify a session (operational testing). |

## Logging

The SMF emits structured JSON logs. The default log formatter is configured in `runtime.exs` and is not intended to be edited per-deployment:

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

| Parameter             | Type      | Required | Default                               | Description                                                                |
|-----------------------|-----------|----------|---------------------------------------|----------------------------------------------------------------------------|
| <code>format</code>   | formatter | No       | <code>OmniLogger.JsonFormatter</code> | Emits a log line in JSON o for ing by log pipelin                          |
| <code>metadata</code> | atom/list | No       | <code>:all</code>                     | Include Logger metadata (SUPI, session SM cor ref, DN proced in each line. |

The runtime log level can be changed without a restart through the management API endpoint `POST /api/oam/log_level`. Filter logs by SUPI to trace a subscriber's full session lifecycle.

[← Overview](#)

# OmniSMF Metrics

Metrics are exposed at `GET /metrics` on `prometheus_metrics_port` (default `9568`, env `PROMETHEUS_PORT`). All metric names below are prefixed as shown by the exporter.

# PDU Session Metrics

| Metric                                           | Type    | Labels                                 | Description                                                                                                            |
|--------------------------------------------------|---------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <code>omni_smf_pdu_session_create_count</code>   | counter | <code>result</code>                    | PDU session create operation                                                                                           |
| <code>omni_smf_pdu_session_modify_count</code>   | counter | <code>result</code>                    | PDU session modify operation                                                                                           |
| <code>omni_smf_pdu_session_release_count</code>  | counter | <code>result</code>                    | PDU session release operation                                                                                          |
| <code>omni_smf_pdu_session_creates_total</code>  | counter | <code>result</code> , <code>dnn</code> | Total PDU session creates by result and DNN.                                                                           |
| <code>omni_smf_pdu_session_releases_total</code> | counter | <code>result</code>                    | Total PDU session releases by result.                                                                                  |
| <code>omni_smf_active_pdu_sessions_count</code>  | gauge   | <code>dnn</code>                       | Number of active PDU sessions by DNN. Emits on session establishment and release from the Context Session (the source) |

| Metric                                         | Type         | Labels | Description                                                                                                                       |
|------------------------------------------------|--------------|--------|-----------------------------------------------------------------------------------------------------------------------------------|
|                                                |              |        | truth for active sessions)                                                                                                        |
| <code>omni_smf_session_duration_seconds</code> | distribution | none   | PDU session duration seconds, emitted on session release and session establishment (buckets: 5, 15, 30, 300, 900, 3600, 86400 s). |

Example (establishment success rate):

```
sum(rate(omni_smf_pdu_session_creates_total{result="success"}[5m]))
/ sum(rate(omni_smf_pdu_session_creates_total[5m]))
```

Example (active sessions per DNN):

```
sum by (dnn) (omni_smf_active_pdu_sessions_count)
```

# PFCP / UPF Metrics

| Metric                                                  | Type    | Labels                 |                             |
|---------------------------------------------------------|---------|------------------------|-----------------------------|
| <code>omni_smf_pfcpsessions_total</code>                | counter | <code>operation</code> | T<br>S<br>o<br>(<br>/       |
| <code>omni_smf_pfcpassociations_total</code>            | counter | <code>result</code>    | T<br>a<br>S<br>t<br>(<br>\$ |
| <code>omni_smf_pfcphheartbeats_total</code>             | counter | <code>result</code>    | T<br>h<br>o<br>t<br>(<br>\$ |
| <code>omni_smf_upfhealth</code>                         | gauge   | none                   | L<br>a<br>h<br>0            |
| <code>omni_smf_n4usage_reports_total</code>             | counter | none                   | N<br>r<br>r<br>t<br>2       |
| <code>omni_smf_n4usage_report_volume_bytes_total</code> | counter | none                   | T<br>u                      |

| Metric | Type | Labels |                  |
|--------|------|--------|------------------|
|        |      |        | v<br>b<br>a<br>u |

The two `omni_smf_n4_*` metrics are the SMF's only user-plane volume metrics: they populate from the URR usage reports the UPF returns over N4, so they only advance when the UPF has usage-reporting rules installed and is delivering reports.

Example (alert when the UPF association drops):

```
omni_smf_upf_health == 0
```

Example (user-plane volume throughput across all sessions):

```
sum(rate(omni_smf_n4_usage_report_volume_bytes_total[5m]))
```

Example (rate of usage reports received from the UPF):

```
sum(rate(omni_smf_n4_usage_reports_total[5m]))
```

Example (PFCP heartbeat failures toward the UPF):

```
sum(rate(omni_smf_pfcpc_heartbeats_total{result="failure"}[5m]))
```

# N40 Charging Metrics

| Metric                                   | Type    | Labels                 | Description                                                                                |
|------------------------------------------|---------|------------------------|--------------------------------------------------------------------------------------------|
| <code>omni_smf_n40_charging_total</code> | counter | <code>operation</code> | Total N40 Nchf converged-charging operations toward the CHF by charging event (TS 32.290). |

Example (rate of converged-charging operations by event):

```
sum by (operation) (rate(omni_smf_n40_charging_total[5m]))
```

# NRF Metrics

| Metric                                        | Type  | Labels               | Description                                        |
|-----------------------------------------------|-------|----------------------|----------------------------------------------------|
| <code>omni_smf_nrf_registration_status</code> | gauge | <code>nf_type</code> | NRF registration status (1 = registered, 0 = not). |

Example:

```
omni_smf_nrf_registration_status == 0
```

# BEAM VM Metrics

| Metric                                  | Type  | Description                           |
|-----------------------------------------|-------|---------------------------------------|
| <code>beam_memory_total</code>          | gauge | Total BEAM memory (bytes).            |
| <code>beam_memory_processes</code>      | gauge | Memory allocated to Erlang processes. |
| <code>beam_memory_processes_used</code> | gauge | Memory actually used by processes.    |
| <code>beam_memory_system</code>         | gauge | System memory.                        |
| <code>beam_memory_atom</code>           | gauge | Total atom memory.                    |
| <code>beam_memory_atom_used</code>      | gauge | Used atom memory.                     |
| <code>beam_memory_binary</code>         | gauge | Binary memory.                        |
| <code>beam_memory_code</code>           | gauge | Code memory.                          |
| <code>beam_memory_ets</code>            | gauge | ETS table memory.                     |
| <code>beam_processes_count</code>       | gauge | Number of Erlang processes.           |
| <code>beam_ports_count</code>           | gauge | Number of Erlang ports.               |
| <code>beam_atom_count</code>            | gauge | Number of atoms.                      |
| <code>beam_vm_uptime</code>             | gauge | VM uptime (seconds).                  |

The active-process count doubles as a session-scale signal, since each PDU session is a dedicated worker process. A sustained divergence between `beam_processes_count` and `omni_smf_active_pdu_sessions_count` can indicate orphaned session workers (see [Troubleshooting](#)).

# OmniSMF Operations Guide

OmniSMF is the Session Management Function (SMF) of the 5G Core. It owns the full lifecycle of PDU sessions: address allocation, policy association, user-plane programming and teardown. This guide covers the 3GPP role, the SBI and N4 interfaces it exposes, and the key procedures it drives, each with a sequence diagram.

# 3GPP Role and Specification References

| Specification         | Relevance                                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| TS 23.501             | System architecture: SMF role, PDU session concept, QoS model                                                     |
| TS 23.502             | Procedures: PDU Session Establishment (§4.3.2), Modification (§4.3.3), Release (§4.3.4), Service Request (§4.2.3) |
| TS 29.502             | Nsmf_PDUSession service API                                                                                       |
| TS 29.244             | N4 interface: PFCP protocol between SMF and UPF                                                                   |
| TS 29.503             | Nudm_SubscriberDataManagement (sm-data) and Nudm_UEContextManagement (smf-registrations)                          |
| TS 29.512             | Npcf_SMPolicyControl: SM policy association and notification toward PCF                                           |
| TS 29.518             | Namf_Communication: N1N2MessageTransfer toward AMF                                                                |
| TS 32.290 / TS 32.291 | Nchf_ConvergedCharging (N40): converged charging toward the CHF                                                   |
| TS 24.501             | 5GS NAS (5GSM): PDU session establishment/modification, session type and 5GSM cause handling (§6.4.1.3)           |
| TS 29.571             | Common data types: <code>IpAddress</code> , S-NSSAI                                                               |

# Interfaces

| Interface | Peer  | Protocol                      | Purpose                                                                                                                                                          |
|-----------|-------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| N11       | AMF   | Nsmf_PDUSession (SBI, HTTP/2) | SM context create / retrieve update / release, N1N2MessageTransfer                                                                                               |
| N7        | PCF   | Npcf_SMPolicyControl (SBI)    | SM policy association create + policy-notify callback                                                                                                            |
| N10       | UDM   | Nudm_SDM + Nudm_UECM (SBI)    | Session subscription data, S registration                                                                                                                        |
| N4        | UPF   | PFCP (UDP)                    | PFCP association + session establishment/modification/c                                                                                                          |
| N40       | CHF   | Nchf_ConvergedCharging (SBI)  | Converged charging: open/update/close a Charging Request over the PDU sessio lifecycle (CHF resolved via N discovery with a static fallback disabled by default) |
| N26       | PGW-C | Internal interworking         | Adopt / switch / lookup for 4G mobility                                                                                                                          |
| n/a       | NRF   | Nnrf_NFManagement (SBI)       | NF registration + heartbeat outbound discovery)                                                                                                                  |

## SBI Endpoints

All SBI endpoints are served under the base URL `{SBI_SCHEME}://{SBI_ADDR}:{SBI_PORT}`.

| Method | Path                                                    | Service         | Path                                                 |
|--------|---------------------------------------------------------|-----------------|------------------------------------------------------|
| POST   | /nsmf-pdusession/v1/sm-contexts                         | Nsmf_PDUSession | Create SM Context (PDU Session establishment)        |
| POST   | /nsmf-pdusession/v1/sm-contexts/{smContextRef}/retrieve | Nsmf_PDUSession | Retrieve SM Context (uses FQDN GET)                  |
| POST   | /nsmf-pdusession/v1/sm-contexts/{smContextRef}/modify   | Nsmf_PDUSession | Update SM Context UP state, handover, flow release   |
| POST   | /nsmf-pdusession/v1/sm-contexts/{smContextRef}/release  | Nsmf_PDUSession | Release SM Context (teardown)                        |
| POST   | /nsmf-callback/sm-policy-notify/{smContextRef}          | Npcf callback   | Receive SmPolicy update from PCF (control QoS flows) |

## Create SM Context

Mandatory request fields: supi, sNssai, servingNetwork, dnn, smContextStatusUri, pduSessionId, anType, servingNfId. Returns 201 Created with Location: {base}/nsmf-pdusession/v1/sm-contexts/{ref}. Error mapping: 400 MANDATORY\_IE\_MISSING, 404 DNN\_DENIED, 500 SYSTEM\_FAILURE, 503 NF\_DISCOVERY\_FAILURE (UDM unreachable).

## Retrieve SM Context

The AMF reads the current SM context by referencing it (TS 29.502 §5.2.2.6 uses `POST .../sm-contexts/{smContextRef}/retrieve`, not `GET`). The SMF returns `200 OK` with the current SM context view for a known reference, or `404 CONTEXT_NOT_FOUND` for an unknown one.

## Update SM Context

The modify handler dispatches on the first recognized key in the body:

| Body key                                               | Scenario                                                                                                                                             |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>n2SmInfo</code> + <code>n2SmInfoType</code>      | N2 SM information from gNB ( <code>PDU_RES_SETUP_RSP</code> activates the tunnel; <code>PDU_RES_REL_RSP</code> deactivates it)                       |
| <code>upCnxState:</code><br><code>"DEACTIVATED"</code> | AN release / UE idle entry (downlink FAR to BUFFER)                                                                                                  |
| <code>upCnxState:</code><br><code>"ACTIVATING"</code>  | Service request; response carries N2 <code>PDU_RES_SETUP_REQ</code>                                                                                  |
| <code>hoState</code>                                   | Handover state transition: <code>PREPARING</code> , <code>PREPARED</code> , <code>COMPLETED</code> (re-activates the tunnel), <code>CANCELLED</code> |
| <code>qosFlows</code>                                  | Post-establishment QoS-flow add/modify (returns <code>204 No Content</code> ; changes conveyed to the gNB via N2 SM info)                            |
| <code>release: true</code>                             | AMF-initiated release (returns <code>204 No Content</code> )                                                                                         |
| <code>servingNfId</code>                               | AMF instance change during mobility (returns <code>204 No Content</code> )                                                                           |

## Release SM Context

The body may include `ueLocation` for final location recording. Returns `204 No Content`.

## Key Procedures

## PDU Session Establishment (TS 23.502 §4.3.2)



The N1N2MessageTransfer is sent asynchronously so the 201 Created returns to the AMF first. The N1 SM message is a NAS PDU Session Establishment Accept and the N2 SM info is an NGAP

PDU Session Resource Setup Request Transfer, carried as a multipart/related body (TS 29.518) with the binary NAS/NGAP parts referenced by contentId.

## DNN Configuration Sourcing

A DNN's behaviour is defined in two places, and an operator must set both consistently for a DNN to come up correctly:

| DNN attribute           | Sourced from                                         | How it is provisioned                                                                                                                      |
|-------------------------|------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Allowed S-NSSAI         | AMF/UDM subscription                                 | The AMF supplies <code>sNssai</code> on the Create SM Context; the DNN must exist under a matching S-NSSAI in the subscriber's UDM SM data |
| Session-AMBR            | UDM<br><code>dnnConfiguration.sessionAmbr</code>     | Provisioned in the HSS → UDM/UDR                                                                                                           |
| Default QoS (5QI + ARP) | UDM<br><code>dnnConfiguration.5gQosProfile</code>    | Provisioned in the HSS → UDM/UDR                                                                                                           |
| PDU session type(s)     | UDM<br><code>dnnConfiguration.pduSessionTypes</code> | <code>defaultSessionType</code> + <code>allowedSessionTypes</code>                                                                         |
| SSC mode                | UDM <code>dnnConfiguration.sscModes</code>           | <code>defaultSscMode</code> + <code>allowedSscModes</code> (TS 23.501 §5.6.9.2)                                                            |

| DNN attribute             | Sourced from                            | How it is provisioned            |
|---------------------------|-----------------------------------------|----------------------------------|
| Static UE IP              | UDM<br>dnnConfiguration.staticIpAddress | Provisioned in the HSS → UDM/UDR |
| UE IP pool (dynamic)      | <b>SMF</b> ue_ip_pool subnet_map        | SMF configuration                |
| DNS / MTU / P-CSCF (ePCO) | <b>SMF</b> dns / dns6 / mtu / :pco      | SMF configuration                |

The rule of thumb: **per-subscriber and per-DNN policy (S-NSSAI, AMBR, default QoS, session type, SSC mode, static IP) lives in the UDM subscription; the network resources handed to the UE (dynamic IP pool, DNS, MTU, P-CSCF) live in SMF configuration.** A DNN that is present in the SMF subnet\_map but absent from the subscriber's UDM SM data is rejected with 404 DNN\_DENIED; a DNN present in the UDM but with no matching subnet\_map entry and no default pool fails IP allocation with :no\_pool\_for\_dnn.

## UE IP Address Allocation

Every PDU session is assigned an IP address (IPv4, IPv6, or both) before the PFCP session is established with the UPF. The IP pool mirrors the 4G PGW-C allocation model. Sources are tried in priority order:

1. **N26 adopted address.** For a 4G-originated session moving to 5G over N26 (an adoptedIpv4 on the create request), the UE IP the standalone PGW-C already allocated is reused verbatim; the address belongs to the

PGW-C's pool, so it is preserved across the handover, not re-allocated. For a dual-stack session the IPv6 portion is then allocated dynamically.

2. **Static IP (subscriber-pinned).** When the UDM SM subscription data contains a `staticIpAddress` for the requested DNN, that fixed address is honoured for every session on that DNN. This mirrors the 4G HSS/PGW-C static-IP behaviour and is ultimately provisioned in the HSS and surfaced through the UDM/UDR. For a dual-stack session the IPv6 portion is allocated dynamically.
3. **Dynamic pool.** Otherwise an address is allocated from the DNN's configured pool. Allocation picks a random free host address (excluding network/broadcast for IPv4), retries on collision, and is deduplicated across all active sessions.

The UE IP is returned to the pool on **any** worker exit (explicit release, crash, or supervisor shutdown), so a CPE stuck in an establish/release/retry loop cannot slowly leak pool addresses.

### DNN-aware pools ( `subnet_map` )

Pools are selected per-DNN via the `subnet_map`. Keys are exact DNN names, `^regex` patterns, or `:default`; values are lists of IPv4 and/or IPv6 CIDRs. Matching precedence is **exact DNN** → **regex** → `:default`. For an `IPv4V6` session the SMF allocates from both the IPv4 and IPv6 ranges of the matched pool. See [UE IP Pool](#) in the Configuration Reference.

### Static IP field and reservation

The `staticIpAddress` field follows TS 29.503 (`DnnConfiguration`) and TS 29.571 (`IpAddress`): an array of objects, each optionally carrying `ipv4Addr` (a single object is also accepted). A static address is reserved in the pool so it is never handed out dynamically:

- A static address **inside** a configured range is counted against that pool and excluded from dynamic allocation.
- A static address **outside** every configured range is tracked for duplicate detection only and does not consume pool capacity (static IPs may live on their own routed subnet).

- If the static address is **already in use** by another live session, the new PDU session is rejected (`:static_ip_in_use`) rather than issuing a duplicate, matching the 4G PGW-C. This should only occur if the same static IP is misconfigured against two subscribers.

To provision a static IP, set `staticIpAddress` on the subscriber's DNN configuration (via the HSS → UDM/UDR); no SMF configuration change is required.

### **PDU session type reconciliation (TS 24.501 §6.4.1.3)**

When a dual-stack (`IPV4V6`) session is requested but the matched pool only yields one address family, the SMF **reconciles the signalled session type down to the family it actually allocated** before it builds the N1 Establishment Accept and the N2 `PDUSessionResourceSetupRequestTransfer`:

- IPv4 allocated, no IPv6 → signal `IPV4` with 5GSM cause **#50 (PDU session type IPv4 only allowed)**.
- IPv6 allocated, no IPv4 → signal `IPV6` with 5GSM cause **#51 (PDU session type IPv6 only allowed)**.
- Both families present → signal `IPV4V6` unchanged.

This is required for interoperability: advertising `IPV4V6` while carrying only an IPv4 address makes the RAN reject the N2 resource setup. Operators serving dual-stack subscriptions from IPv4-only pools will therefore see those sessions come up as IPv4 with the corresponding NAS cause, rather than failing.

### **Monitoring**

- `GET /api/ip_pool`: overall IPv4 capacity plus a per-DNN pool summary.
- `GET /api/ip_pool/detail`: per-DNN pools with each range's CIDR, total/allocated/available, and the full list of currently-allocated addresses (dynamic + static).

**IPv6 note:** IPv6 addresses are allocated, reserved, released, and reported by the pool.

# P-CSCF Discovery and Advertisement (VoNR / IMS)

For an IMS/VoNR PDU session the UE must learn a P-CSCF to register against. The SMF returns it in the **ePCO** of the PDU Session Establishment Accept (P-CSCF IPv4 container `0x000C`, IPv6 container `0x0001`), one container per address, identical to the 4G PGW-C. The P-CSCF is only returned when the DNN begins with `ims`. Selection order:

1. **DNS discovery + health check (preferred).** When `p_cscf_discovery_enabled`, the SMF resolves `_sip._tcp.<fqdn>` (SRV) to A/AAAA, then health-checks each candidate with a SIP `OPTIONS` (TCP then UDP). Only servers that answer are advertised, and the set is refreshed on a fixed 60-second cycle, so a P-CSCF that stops answering `OPTIONS` is dropped from the advertised set within roughly one minute, and a recovered one is re-added on the next cycle. The refresh runs independently of session establishment; new sessions always read the latest healthy set.
2. **Static fallback.** If discovery is disabled or returns nothing healthy, `p_cscf_ipv4_address_list` / `p_cscf_ipv6_address_list` are used.

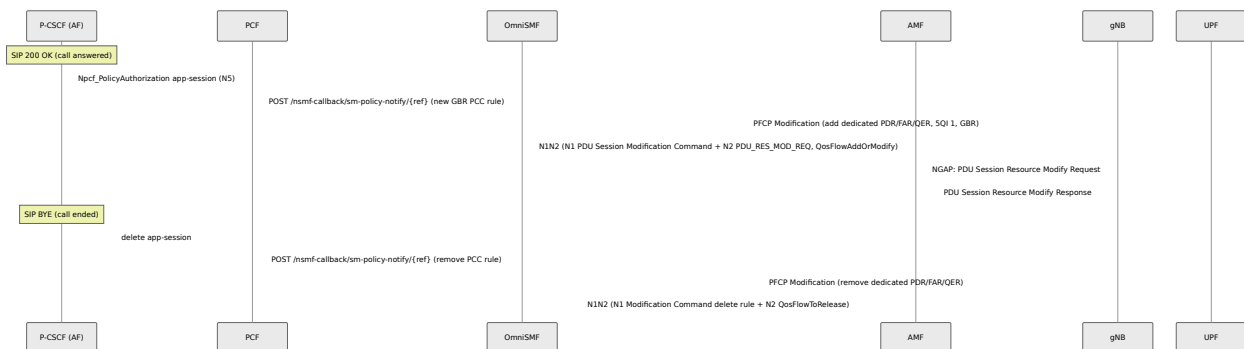
This mirrors the PGW-C exactly, so a mixed 4G/5G IMS deployment configures P-CSCF the same way on both nodes. See [P-CSCF and Protocol Configuration Options](#).

## Dedicated QoS Flow / GBR Bearer (VoNR, 5QI-1)

VoNR needs a dedicated GBR QoS flow (5QI 1) for the voice media, alongside the default non-GBR flow. Both a **static** (provisioned) and a **dynamic** (call-driven) path are supported.

**Static:** the GBR rule is provisioned in policy (the HSS PCRF charging-rule to UDR `sm-policy-data` to PCF PCC rule) and returned on the `sm-policies` decision. It is stashed and installed once the PFCP session exists, so the UPF marks the flow with its 5QI/GBR from the moment the session comes up.

**Dynamic (call-driven):** the flow is added when the call is answered and removed on BYE, driven by the PCF policy-notify callback:

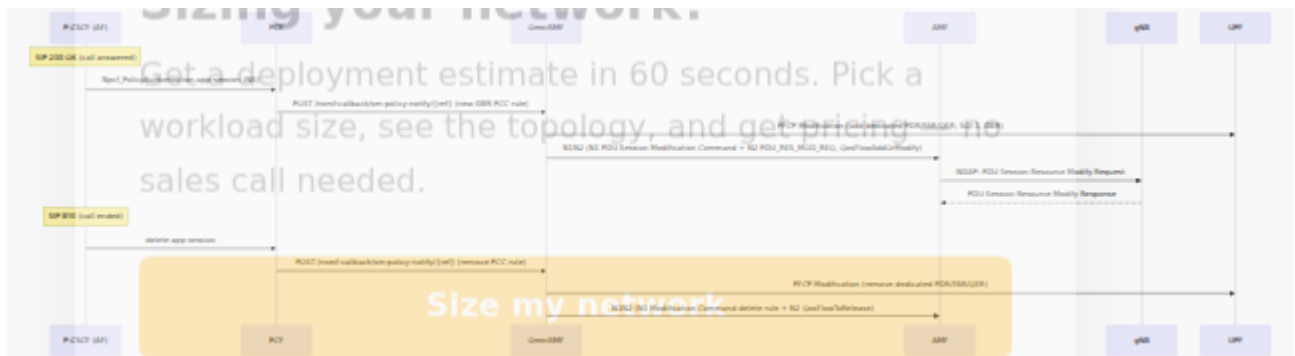


The notification is delivered to `/nsmf-callback/sm-policy-notify/{smContextRef}` (the SMF reads `smPolicyDecision` from the body): a non-null PCC rule installs the flow, a null-valued rule removes it (TS 29.512, a null-valued PCC rule is a rule to remove). The AF app-session to SM-policy matching (by UE IP) is performed on the PCF.

## PDU Session Modification: QoS Flow Update

An AMF may also drive a QoS-flow modification directly through the modify endpoint with a `qosFlows` list, rather than via the PCF. Each entry either updates an existing flow (matched by `qfi`) or allocates a new QFI. The UPF is re-programmed with the changed QoS, and because `SmContextUpdatedData` defines no `qosFlows` JSON member, the response is `204 No Content`; the QoS change is conveyed to the gNB via N2 SM information.

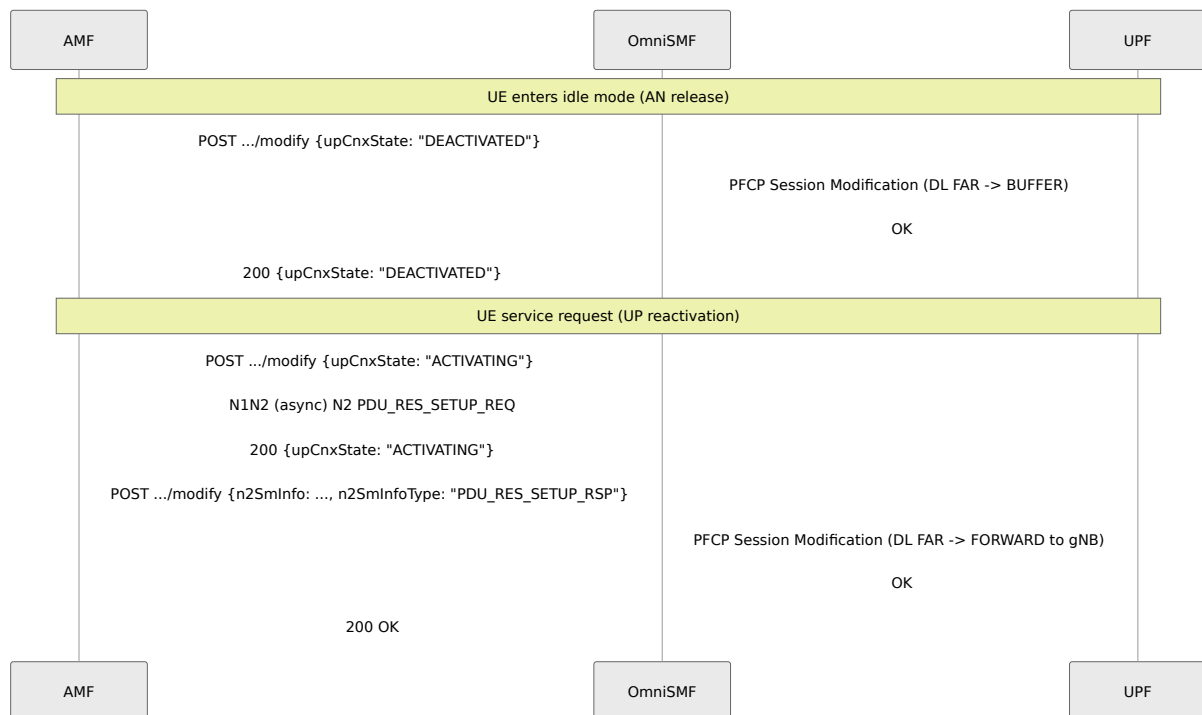
## PDU Session Release (TS 23.502 §4.3.4)



Teardown order is PCF policy delete → PCF session delete → UDM deregistration → SM context status notification (`resourceStatus: RELEASED`) to

the AMF callback; the worker then releases the UE IP back to the pool and stops. The same teardown runs for an AMF-initiated `release: true` modify.

## UE Idle / Service Request: UP Connection State (TS 23.502 §4.2.3)



On `ACTIVATING`, the SMF updates the PTI from the UE's latest establishment request and sends an N1N2MessageTransfer to the AMF to trigger the `PDUSessionResourceSetup` on the gNB; the `200` response carries `upCnxState: ACTIVATING`. The user plane is re-pointed at the gNB only once the gNB F-TEID arrives in the follow-up `PDU_RES_SETUP_RSP`.

## Handover State Transitions

An AMF-driven handover progresses the session through `hoState` values on the modify endpoint:

- `PREPARING` / `PREPARED`: state is recorded; the user plane is not changed.
- `COMPLETED`: the tunnel is re-activated (PFCP modification, downlink FAR to the target gNB) and the handover state cleared.
- `CANCELLED`: the handover state is cleared with no user-plane change.

## N4 PFCP Session Structure

Each PDU session installs the following PFCP IEs on the UPF (TS 29.244):

| IE             | Direction     | Purpose                                                                              |
|----------------|---------------|--------------------------------------------------------------------------------------|
| PDR (uplink)   | Access → Core | Match GTP-U traffic from gNB on the UPF N3 F-TEID                                    |
| PDR (downlink) | Core → Access | Match traffic from N6 by UE IP address                                               |
| FAR (uplink)   | Core          | Forward to N6 (no outer header)                                                      |
| FAR (downlink) | Access        | Initially BUFFER; updated to GTP-U FORWARD after <code>PDU_RES_SETUP_RSP</code>      |
| QER            | Both          | Enforce session AMBR (uplink and downlink MBR); dedicated QER added per GBR flow     |
| URR            | Both          | Time-based usage reporting (threshold derived from <code>heartbeat_interval</code> ) |

The single URR uses duration-based measurement with a time-threshold trigger. The threshold is derived from `heartbeat_interval` (six times its value, e.g. 60 s at the default 10 s heartbeat), so it is not independently tunable; raise `heartbeat_interval` to lengthen the reporting cadence. Usage reports the UPF sends back are recorded and surfaced as the `omni_smf_n4_usage_reports_total` and `omni_smf_n4_usage_report_volume_bytes_total` metrics (see [Metrics](#)), and, when N40 converged charging is enabled, forwarded to the CHF as interim Charging Data Requests.

The PFCP association with the UPF is established (with heartbeats) before any session request is sent. If the association is lost, it can be re-triggered through the management API (`POST /api/oam/pfcp/reassociate`).

## N26 EPS Interworking (4G ↔ 5G)

The SMF and the standalone PGW-C share a UPF and coordinate over an internal N26 interworking path so a UE keeps its IP address across a handover:

- **Adopt (4G → 5G):** a 4G-originated EPS session is adopted into 5GS, reusing the PGW-C's already-allocated UE IP. Re-adopting an existing session returns the existing 5G session rather than creating a duplicate.
- **Switch:** an existing session's access side is switched to `:eps` (5G → 4G) or `:fivegs` (4G → 5G) by updating the access tunnel via a PFCP modification (S-GW S5/S8-U TEID/address for EPS, gNB N3 TEID/address for 5GS) while keeping the same UE IP and the same PFCP session on the UPF. The switch returns an EPS view (preserved UE IP, DNN, session AMBR, default EPS bearer id, and the PGW-U F-TEID) that the PGW-C uses to answer the SGW.
- **Lookup:** a UE's active PDU sessions can be read as EPS PDN-connection views by SUPI/IMSI.

The management API exposes `POST /api/eps_handover` to drive interworking. The paired PGW-C's S5/S8-C address is advertised via the `pgw_c_addr` configuration key (see [Configuration Reference](#)).

# OmniSMF

## Troubleshooting

### PDU Session Establishment Fails with 404 DNN\_DENIED

- **Symptom:** AMF receives `404` with cause `DNN_DENIED`.
- **Cause:** the UDM SM subscription data does not include the requested DNN.
- **Resolution:** verify the UDM subscriber data for the SUPI contains a `dnnConfigurations` entry matching the requested DNN (matching is case-insensitive). The SMF log shows `dnn_not_found`.

### PDU Session Establishment Fails with 503 NF\_DISCOVERY\_FAILURE

- **Symptom:** AMF receives `503` with cause `NF_DISCOVERY_FAILURE`.
- **Cause:** the SMF could not reach the UDM at `udm_uri`.
- **Resolution:** verify connectivity from the SMF host to `udm_uri`, and that the UDM SBI port is reachable. The SMF log shows the underlying reason.

### PCF Session Establishment Rejected

- **Symptom:** the UPF returns a non-accepted cause in the PCF Session Establishment Response.
- **Causes / resolution:**
  - `upf_addr` / `upf_pfcf_port` incorrect: verify the UPF is listening on the PCF port.

- PFCP association not established: the SMF must complete PFCP Association Setup before sending session requests. Check startup logs for association messages; if needed, re-trigger via `POST /api/oam/pfcp/reassociate`.
- `pfcp_addr` not routable from the UPF: set it to an address the UPF can reach.

## UE Receives No IP Address

- **Symptom:** session fails during IP assignment.
- **Causes / resolution:**
  - `:no_pool_for_dnn`: the requested DNN has no matching `subnet_map` entry and no `default`; add one.
  - `:ipv4_pool_exhausted` / `:ipv6_pool_exhausted`: the matched range is full; enlarge or add a range.
  - `:static_ip_in_use`: the same `staticIpAddress` is provisioned against two subscribers; correct the subscription data.
  - `:static_ip_invalid`: the provisioned `staticIpAddress.ipv4Addr` is not a valid IPv4 address; correct the subscription data.
- **Dual-stack served as single-stack:** a dual-stack request from an IPv4-only pool intentionally comes up as IPv4 with 5GSM cause #50 (see [PDU session type reconciliation](#)).

## Session AMBR Capped at ~1 Mbps

- **Symptom:** every session's throughput is capped at roughly 1 Mbps regardless of the subscribed AMBR.
- **Cause:** the PCF received only a non-spec `sessionAmbr` key and left `subsSessAmbr` unset, so it fell back to its ~1000 Kbps default.
- **Resolution:** the SMF sends both `subsSessAmbr` (the TS 29.512 field name) and `sessionAmbr` on the SM policy association, and reads the subscribed AMBR from the UDM DNN configuration `sessionAmbr` field. Verify the UDM DNN configuration carries the correct `sessionAmbr`, and that the PCF honours `subsSessAmbr`. See [Session-AMBR toward the PCF](#).

# N1N2MessageTransfer to AMF Fails

- **Symptom:** the SM context and PFCP session are created but the UE never receives the Establishment Accept.
- **Cause:** the asynchronous N1N2MessageTransfer to the AMF failed; no automatic retry.
- **Resolution:** verify `amf_uri` and AMF reachability. To recover, release the SM context and retry establishment.

# Dedicated VoNR Flow Not Created

- **Symptom:** a voice call establishes but no dedicated 5QI-1 GBR flow appears.
- **Causes / resolution:**
  - The PCF policy-notify callback did not arrive: confirm the PCF is configured with the notify URI `/nsmf-callback/sm-policy-notify/{smContextRef}` and that the AF (P-CSCF) app-session reached the PCF.
  - No QFI available: the log shows `No QFI available for PCC rule ...`; too many dedicated flows already exist on the session.
  - The PCC rule carried no 5QI / GBR: only rules with a resolvable `5qi` and GBR values become dedicated flows; verify the charging rule / QoS decision in policy.

# High Session Count / Memory Growth

- **Symptom:** session worker count grows without matching releases.
- **Cause:** missed AMF/UE-initiated releases (e.g. after an AMF restart) leave orphaned session workers; there is no session timeout.
- **Resolution:** track the live active-session count with the `omni_smf_active_pdu_sessions_count` gauge (broken down by DNN), or read the `GET /api/statistics` and `GET /api/pdu_session` management endpoints (see [Metrics](#)). Sessions can be drained by DNN, SUPI, S-NSSAI, or

UPF via the `/api/oam/session/by_*` management endpoints. The UE IP is always returned to the pool when a worker exits, so draining orphaned workers reclaims their addresses.

## Log Correlation

Logs are structured JSON with metadata (SUPI, PDU session id, SM context ref, DNN, procedure). Filter by SUPI to trace a subscriber's session lifecycle. Adjust the runtime log level via `POST /api/oam/log_level` without a restart.

