NOKIA

NSP
Network Services Platform

Network Functions Manager - Packet (NFM-P)
Network Resource Controller - Flow (NRC-F)
Network Resource Controller - Packet (NRC-P)
Network Resource Controller - Transport (NRC-T)
Network Services Director - (NSD)

Release 17.3

API Programmer Guide

3HE-11987-AAAA-TQZZA

Issue 1

March 2017

Contents

About this document

Purpose

The *NSP API Programmer Guide* provides information to assist OSS developers with the creation of business applications that interact with NSP modules and applications to perform various network management functions.

Document support

Customer documentation and product support URLs:

- Customer Documentation Welcome Page
- Technical support

How to comment

- Documentation feedback

Part I: Getting started

Overview

Purpose

This volume provides an overview of OSS application development and Open Interfaces Professional Services.

Contents

1 Getting started

1.1 Overview

1.1.1 About this guide

The *NSP API Programmer Guide* provides information to assist OSS developers with the creation of business applications that interact with NSP modules and applications to perform various network management functions.

1.1.2 Before you begin

Nokia recommends that you perform the following before you begin using the NSP APIs:

- review the license requirements and platform size considerations for your NSP system. See the *NSP Deployment Overview* for more information.
- understand how to install and configure the devices and NSP software to deploy a network for your development environment. See the *NSP NFM-P Installation and Upgrade Guide* and the *NSP NSD and NRC Installation and User Guide* for more information.
- understand OSS interfaces and have knowledge of the following technologies, which are beyond the scope of this guide:
  - HTTP(S) – *HyperText Transfer Protocol (Secure)*
  - REST architecture – *Representational State Transfer*
  - JSON – *Javascript Object Notation*
  - web applications

1.1.3 Open Interfaces Professional Services

The Nokia Open Interfaces Professional Services (OIPS) portfolio provides OSS developers with support on how to integrate with the NSP suite of applications.

OSS integration initiatives include project review, design consultation, development support, and training for integration projects.

The key elements of each initiative include:

- Design consultation and high-level project reviews. Extensive software design consultation provides an architectural review of a proposed design and guidance on the suitability of the design methodology to the proposed application.
- software development support and technical guidance
- sample code that provides examples to facilitate rapid OSS integration
- OSS product training to minimize project downtime and optimize development efficiency

Nokia OIPS can customize the support content and contract length to meet the unique development requirements of service providers.

Part II: NSP APIs

Overview

Purpose

This volume describes the NSP modules.

Contents

2 NSP API overview

2.1 Overview

2.1.1 Purpose

The Network Service Platform, or NSP, comprises different modules. This chapter introduces the NSP modules that expose their functionality through REST API services in the following functional categories:

- NSP REST gateway
- NSD and NRC
  - — Service fulfillment
  - — Optimization
  - — Notification
- Network and service assurance
- Network Functions Management

See the *NSP Deployment Overview* for more information about the NSP and its modules.

2.1.2 Contents

NSP API modules

2.2 Overview

2.2.1 NSP REST gateway

The NSP REST Gateway API provides the entry point for API clients to locate and authenticate REST client requests to gain access to the various NSP modules that have registered API services.

The REST gateway online documentation is located at https://<server>/rest-gateway/api-docs

where <server> is the hostname of the NSP server that hosts the nspOS server.

See Chapter 3, “NSP REST API Gateway” for more information.

2.2.2 NSD and NRC

The NSD and NRCs modules provide a common set of northbound RESTful APIs that exposes a simplified view of the network. The APIs support queries, service creation, deletion, and modification requests, and allow northbound OSS clients to receive notifications.

Table 1 SDN Modules

Module	Description	API online documentation
Service Fulfillment	The Network Services Director, or NSD, is the multi-vendor service fulfillment module of the NSP SDN Platform. It automates IP/MPLS, Carrier Ethernet and optical service provisioning by mapping abstract service definitions to detailed service templates using operator-defined policies. It provides provisioning for complex multi-technology services across multi-domain and multi-vendor networks.	https://<server>:8543/sdn/doc https://<server>:8543/task-scheduler/doc where <server> is the hostname or IP address of the NSP server that hosts the API module

Table 1 SDN Modules (continued)

Module	Description	API online documentation
Optimization	The Network Resource Controller (NRC) module performs multi-domain and multi-layer WAN path computations across IP/MPLS, optical or hybrid IP/optical multi-vendor networks. The NRC serves path connection requests from the NSD, OSSs and orchestration systems, and physical/virtual network elements. The NRC is standards-based on a stateful PCE architecture. It employs various path optimization algorithms to ensure the best path placement for services and load-balancing for path distribution across the network.	https://<server>:8543/sdn/doc https://<server>:8543/task-scheduler/doc where <server> is the hostname or IP address of the NSP server that hosts the API module
Notification	The NSP northbound interface allows clients to receive notifications whenever the operational or administrative state of an NSP-managed service or endpoint changes. This simplifies synchronization with the NSP by removing the need for periodic polling of the REST API. HTTPS Server Side Events (SSE) is used for the transport of these notifications, which are encoded in JSON according to the IETF RESTCONF protocol.	https://<server>:8543/ean/doc where <server> is the hostname or IP address of the NSP server that hosts the API module

See Chapter 4, “NSD and NRC” for more information.

2.2.3 Network and service assurance

Comprehensive network and service assurance from the NSP assurance functions is an integrated option that is available for all of the NSP modules. Network and service assurance supports IP, optical and integrated IP/optical networks and services. NSP provides these functions using browser-based web applications for network fault monitoring and troubleshooting purposes.

The following table describes the NSP web applications that use REST APIs to publish a set of URLs that point to resources, or web services, managed by them. Each domain application documents the URLs that are available to developers.

Table 2 Assurance applications

Application	Description	API availability	API online documentation
Fault Management	Provides alarm monitoring, correlation, and troubleshooting for the unhealthiest NEs in the network, allowing you to filter, identify root causes, and determine the impact of alarms	being deprecated	https://<server>:8543/FaultManagement/api-docs http://<server>/FaultManagement/api-docs where <server> is the hostname or IP address of the NSP server that hosts the API module
Service Supervision OAM	Provides API Clients with access to execute OAM tests on an NSP managed network for the purposes of doing service validation and troubleshooting that verifies compliance with SLAs..	yes	https://<server>:8543/ServiceSupervision/api-docs http://<server>/ServiceSupervision/api-docs where <server> is the hostname or IP address of the NSP server that hosts the API module

See the NSP online help for more information about these applications.

See Chapter 5, “Network and service assurance” for more information.

2.2.4 Network Functions Management

Network Functions Management, or NFM, consists of two modules that provides base fault, configuration, accounting, performance and security (FCAPS) management with advanced extensions for network deployment automation, service templates and assurance.

Table 3 Network Functions Management applications

Application	Description	API availability	API online documentation
NFM-P	Allows access to OSS applications to perform IP network and service management across all IP/MPLS and carrier Ethernet network domains—access aggregation, metro, and core.	yes	https://<server>:8543/samrestfulservice/api-docs http://<server>/FaultManagement/api-docs where <server> is the hostname or IP address of the NSP server that hosts the API module

Table 3 Network Functions Management applications (continued)

Application	Description	API availability	API online documentation
NFM-T	Provides a common northbound REST API that enables OSS integration for the end-to-end optical management functions that include service provisioning over multi-technology optical transport networks such as SDH/SONET, carrier Ethernet, WDM, ROADM, OTN and packet	yes	Contact your Nokia representative for more information
PCMP	PCMP (Protocol Convertor and Mediation Platform) is an NFM-P component that interacts with the 7750 SR running as a vRGW (Virtual Residential Gateway). The PCMP translates REST API calls from a BRGC (Bridged Residential Gateway Controller) into the configuration required to modify the operational behavior of BRGs connected downstream from the 7750 SR	yes	http://<pcmp_ip_address>: 8080/pcmp/api-docs/ *PCMP API Reference Guide*

See Chapter 6, "Network Functions Manager " for more information.

Common API features

2.3 General information

2.3.1 REST operations

The NSP APIs implement a REST architecture that uses the simple HTTP(S) protocol to send requests and receive responses between an OSS client application and NSP. The NSP REST interfaces offer the CRUD (Create/ Read /Update/Delete) operations that translate to POST, GET, PUT/PATCH and DELETE HTTP(S) requests.

Both PUT and PATCH operations can be used to update/modify resources. However, when performing a PUT operation on an object or class, the request must contain the modified properties along with all of the other properties for that class. For PATCH operations, the request only needs to contain the modified properties.

The PATCH operation has been implemented for complex attributes in the NSD and NRC modules of the NSP, such as siteServiceQosProfile in L2EndpointRequest, or routeTargets, routingBgp, routingStatic, and secondaryAddresses in L3EndpointRequest. See Chapter 4, “NSD and NRC” for more information.

2.3.2 Versioning

Up to three versions of the NSP REST APIs are maintained for each major release. The latest API version may introduce additional request attributes if there is a fully backward-compatible default value, and users must be prepared to either accept additional response attributes, or ignore any that are unknown. The latest API versions may also introduce additional functions, but the behavior of existing functions do not change.

See Chapter 7, “Backwards compatibility” for more information about the differences between API versions.

2.3.3 Filtering

When using any of the NSP REST APIs, filtering can be used to limit the number of objects returned by a GET request. Filters are defined in the query parameter of the URI within GET requests, and only objects with attributes that match the defined expression or expressions are returned. The URI syntax supports expressions for many object attributes and allows for logical combinations of more than one filter criteria. A filter is comprised of a series of attributes, identified by name, that are assigned values for which to filter.

Multiple filters can be separated by the '&' character (for AND). Values can be preceded by the '!' character for NOT. Multiple values are separated by commas. Parenthesis are required around values when using NOT.

The formal filtering notation is:

```
filter = expression {"&" expression}

expression = attributeName "=" ["!("] value {"," value} [")"]

| "sortBy=" [-]attributeName

attributeName = string

value = string|number
```

The following notable information applies to filtering:
- both attributeName= and attributeName=() specify that the value of attributeName is an empty string
- attributeName=!() specifies that the value of attributeName is not an empty string
- attributeName=(value1) and attributeName=value1 are equivalent
- Only the first sortBy expression in a filter will take effect. Additional sortBy expressions are ignored.
- object attributes that have a null value or a boolean value cannot be filtered out of a list and always appear in the results
- Object attributes that are strings must not contain commas (,) as these are used as separators in an OR statement. String values are case sensitive.
- A filtered search ignores any unrecognized object attributes that are specified. A filtered search that contains a recognized numeric attribute, but an incomplete condition, returns an error. A filtered search that contains a recognized string attribute, but an incomplete condition, returns no results.

Note: Response filtering in the request is not currently available.

2.3.4 Pagination

The NSP REST APIs support pagination by a JSON object that includes the values “startRow”, “endRow”, “totalRows”, and “status”. The APIs described in this document do not use pagination, as the NSP server always returns a complete response.

2.3.5 Server Response Status Codes

The HTTP(S) protocol is used between the OSS application and the NSP applications. As such, the server replies using standard HTTP(S) response codes. All error codes must be handled accordingly.

2XX codes

2XX codes indicate that the request was processed.

- 200: Everything is OK (content is in the body)
- 204: Everything is OK, but there is no content in the body (this is the standard response for an update or a deletion)

4XX codes

- 400: Bad request: something is wrong in the request
- 401: Not authenticated (wrong password or wrong Token)
- 403: Not authorized (request for something to which you do not have rights)
- 404: Not found (request for something that does not exist)
- 409: Conflict (the requested change conflicts with existing configuration)

5XX codes

- 500: Internal server error (error is explained in the body)

2.3.6 API collections

Sample NSP REST API requests for a variety of functional use cases are provided in both Postman and HTML collections, as follows:

Postman collections

NFM-P REST API examples and NSD and NRC REST API examples are each available in Postman collections that are provided alongside their respective suites of documentation. Postman can be downloaded from https://www.getpostman.com/apps. The collections of REST API examples must be imported into Postman in order to be browsed. The collections have been organized so that folders contain individual workflows. The examples within the folders are presented in the order that they should be executed.

Example requests make use of Postman environments so that they can be executed against a running instance. The following variables can be configured:

- *urlHost* - the hostname of the NFM-P or NSD and NRC server
- *token* - the token returned by the authorization API

In order for the example requests to be successfully executed, their UUIDs may need to be modified so as to reflect applicable values for the system that is being used. In most cases, the workflows have been organized so that UUIDs from earlier requests can be used in subsequent requests. Comments within each example also describe any prerequisites that may be necessary. For further details on using Postman, see the documentation at https://www.getpostman.com/docs.

HTML collection

NSP REST API examples are available in an HTML collection that is provided alongside the full suite of NFM-P and NSD and NRC documentation. This collection has been organized so that sections contain individual workflows. The examples within the sections are presented in the order that they should be executed. In order for the example requests to be successfully executed, their UUIDs may need to be modified so as to reflect applicable values for the system that is being used. In most cases, the workflows have been organized so that UUIDs from earlier requests can be used in subsequent requests. Comments within each example also describe any prerequisites that may be necessary.

Part III: NSP API Modules

Overview

Purpose

This volume describes NSP API functions by providing sample use cases and workflows on the NSP API modules. It also includes information on API backwards compatibility.

Contents

3 NSP REST API Gateway

3.1 Introduction

3.1.1 NSP REST Gateway API

The NSP Rest Gateway API provides the initial entry point to access registered NSP API services of NSP Modules. These APIs allows OSS Clients to perform the following functions:

- List the URL location of registered NSP API services which includes their Base URL and documentation URL that are available to OSS clients.
- Provide Access to these services using an Authorization bearer token along with the ability to refresh the token
- Terminate active sessions

Note: Currently, only the NFM-P module and assurance applications use the NSP Rest Gateway API services for authentication. The other API modules will be supported in future releases of NSP.

Note: The API service “VNFManager” may be returned when the location function is executed. This API service will be changing in future releases of NSP. Contact your Nokia representative if you want to integrate this API service.

3.1.2 Prerequisites

The following are the prerequisites to using the NSP REST Gateway API.

- The NSP module whose API services you want to access using the REST Gateway must be installed and licensed. See the *NSP NFM-P Installation and Upgrade Guide* for more information.
- A valid user account with a username and password needs to be created on the NSP server with access rights to the NSP module. See the *NSP NFM-P System Administrator Guide* for more information.

3.1.3 Retrieve location services

When the NSP modules are installed they are registered with the nspOS which consists of a set of platform services which is used by all NSP modules. OSS clients can retrieve the list of root URL locations for all registered API services.

```
https://<server>/rest-gateway/rest/api/v1/location/services
```

where <server> is the hostname or IP address of the NSP server that hosts the API module

See NSP API postman collection for a sample request for retrieving location services.

3.1.4 Authentication Functions

The following are the different authentication functions.

Initial Authentication

Prior to sending any REST requests, the OSS application must first authenticate the user. This is done by sending a POST request, using Basic authentication (base64 encoded credentials in the form of username:password), to the /rest-gateway /authentication/rest/api/v1/token endpoint to obtain a Bearer token which will be used for future communications:

```
https://<server>/rest-gateway/rest/api/v1/auth/token
```

where the "grant_type" must be specified as “client_credentials”

and where <server> is the hostname or IP address of the NSP server that hosts the API module.

Authentication Refresh

The Bearer token has an expiration time. To avoid expiration, a port request with the “refresh_token” generates a new bearer token that is valid for another “expire_in” seconds, using Basic authentication (base64 encoded credentials in the form of username:password)

. The previous tokens are invalidated upon refresh token generation.

```
https://<server>/rest-gateway/rest/api/v1/auth/token
```

the "grant_type" can be specified as “refresh_token”

and where <server> is the hostname or IP address of the NSP server that hosts the API module.

Revocation

A bearer token can be terminated by sending a POST request to the rest-gateway endpoint with the token or client User_name details.

```
https://<server>/rest-gateway/rest/api/v1/auth/revocation
```

where <server> is the hostname or IP address of the NSP server that hosts the API module.

See NSP API postman collection for sample requests for performing authorisation functions.

3.2 Redundancy overview

3.2.1 Redundancy

For a redundant NSP server deployment, the Rest Gateway API service provides redirect functionality. This means that an API client can perform a location service request or perform authentication on either of the redundant servers, regardless of which is active or standby. It is recommended that the API client store the IP address or hostnames of both the active and standby servers.

In the case of single server failure, the API client may receive an error status code of “404 Not Found” when a service request is sent to the server which is operationally down. We recommend that the API client send a location service request to one of the NSP servers at a time until the API client receives a response with status code “200 OK”. The response will contain the updated URI for all the registered REST API services. An API Client may resume normal functions once authenticated with the current active NSP server.

See NSP API postman collection for sample requests for performing redundancy functions.

See the *NSP Deployment Overview* for more information on redundancy.

Note: Redirecting of API service interfaces such as alarms or service supervision OAM requests are not included as part of the current release. An API client may request these services per Rest API URIs obtained from the location service response. See 3.1.3 “Retrieve location services” (p. 23) and 3.1.4 “Authentication Functions” (p. 24) for more information.

3.3 Sample workflow

3.3.1 Sample workflow

The following example provides a workflow that outlines the steps that an API client can use to gain initial access to the NSP REST Gateway, which provides a token that can be used by registered API services from other NSP API modules. For this example, the API client uses the NSP REST Gateway to authorize access to perform a simple GET operation on one of the Service Supervision API services.

- retrieve the list of available NSP API services using the NSP REST Gateway; see 3.1.3 “Retrieve location services” (p. 23)
- authenticate the user using the NSP REST Gateway API service; see 3.1.4 “Authentication Functions” (p. 24)

- use the bearer token returned from the authentication service to access the Service Supervision API service that performs a simple Get operation to retrieve the test results for a VCCV ping; see 5.2.8 “OSS use case: On-demand OAM diagnostic testing” (p. 41)
- revoke the token to terminate the user’s access; see 3.1.4 “Authentication Functions” (p. 24)

4 NSD and NRC

4.1 Introduction

4.1.1 NSD and NRC applications

The NSD and NRC modules provide service creation and network optimization functions using browser-based applications. Each of these applications can be accessed by any authorized user. To view a dashboard of all the available applications, go to:

```
https://<server>:8543
```

where *server* is the hostname or IP address of your installed NSD and NRC server.

4.2 NSD and NRC API

4.2.1 General information

This section provides a list of API Use Case collection for all the supported SDN Service Provisioning and Optimization functions that are available through the NSD and NRCs APIs.

For more information on NSD and NRC, see the *NSP NSD and NRC Installation and User Guide*.

4.2.2 Authorization

All NSD and NRC API clients are required to login and authenticate with the NRC and NSD module using bearer tokens. This is achieved by sending a JSON-formatted REST request using the standard HTTP(S) protocol to the NSD and NRC server. The server processes the request and returns a JSON-formatted response through the same HTTP protocol.

See the NSD and NRC API postman collection for a sample request for retrieving the token required for authorization as follows:

Authorization – retrieves a bearer token for the API client to use

HTTP headers

All API request uses HTTP headers for authentication. Each API call must be authenticated, so the Authorization header must be sent with each request (with the exception of the GET token request):

Content-type

- The server response is formatted in JSON. Content-Type is a required header field for HTTP POST, PUT, and PATCH as mentioned below to either create or modify an entity:

  PUT /whatever HTTP/1.1

  Content-Type: application/json

  Authorization: $AUTHORIZATION_STRING

Authorization (required)

- Each API call must be authenticated, so the Authorization header must be sent with each request (with the exception of the GET token request):

  GET /whatever HTTP/1.1

  Authorization: $AUTHORIZATION_STRING

  $AUTHORIZATION_STRING is the token returned by the GET token request.

4.2.3 Tenant management

Tenant Management API operations allows an API client to assign users to tenants. A tenant is a logical group that contains users assigned to network resources (ports or nodes). The assignment also includes user roles that determines a user’s access rights to the resources that are assigned to their tenant. Creating, deleting and assigning tenants to users and their roles can be done using this API service.

See the NSD and NRC API postman collection for sample requests to perform the following:

Tenant and User Management Workflow

1. Create a tenant and a user.
2. Assign the user to the tenant.
3. Reassign resources to the tenant.

4.2.4 External applications notifications

The NSD and NRC modules' northbound interface allows its clients to receive notifications whenever the operational or administrative state of an NSP-managed service or endpoint changes. Clients can also receive notifications when the Object Life Cycle of a service changes. This simplifies synchronization with the NSD and NRC modules by removing the need for as periodic polling of the REST API. HTTPS Server

Side Events (SSE) is used for the transport of these notifications, which are encoded in JSON according to the IETF RESTCONF protocol. See the following examples:

Notification for admin state change (service):

```
{
    "data": {
        "ietf-restconf:notification": {
            "eventTime": "2016-10-19T16:33:24.338Z",
            "nsp-service:service-state-change": {
                "admin-state": {
                    "old-value": "UP",
                    "new-value": "DOWN"
                },
                "uuid": "7095-86bc248d-f75a-438c-8de4-
d1221fe8711a"
            }
        }
    }
}
```

Notification for operational state change (endpoint):

```
{
    "data": {
        "ietf-restconf:notification": {
            "eventTime": "2016-10-19T16:33:24.425Z",
            "nsp-service:endpoint-state-change": {
                "operational-state": {
                    "old-value": "UP",
                    "new-value": "DOWN"
                },
                "uuid": "7107-4445bd7a-c741-4fdb-a9c5-
3141b85facb4"
            }
        }
    }
}
```

Notification for operational state change (service):

```
{
    "data": {
        "ietf-restconf:notification": {
            "eventTime": "2016-10-19T16:33:24.448Z",
            "nsp-service:service-state-change": {
                "operational-state": {
                        "old-value": "UP",
                        "new-value": "DOWN"
                    },
                    "uuid": "7095-86bc248d-f75a-438c-8de4-
d1221fe8711a"
                }
            }
        }
}
```

Notifications for Object Life Cycle change (service):

```
data:{
              "ietf-restconf:notification":{
                  "eventTime":"2017-03-21T08:56:20.549Z",
                  "nsp-service:object-life-cycle-change":{
                      "uuid":"3304-fa5c4151-f7f2-447a-8cb7-
d9b347b83ada",
                      "life-cycle-state":{
                          "new-value":"WaitingForDeployment",
                          "old-value":"Deployed"
                    }
               }
          }
}

data:{
              "ietf-restconf:notification":{
                  "eventTime":"2017-03-21T08:56:20.549Z",
                  "nsp-service:object-life-cycle-change":{
                      "uuid":"2632-b7d518a8-1556-4f01-961e-
5a0d0a529722",
                      "life-cycle-reason":{
                          "new-value":"Waiting for underlying
layer to deploy",
                          "old-value":"MPLS path deployed"
                    }
               }
          }
}
```

Note: A security token is required in order to subscribe to the notification stream. See 4.2.2 “Authorization ” (p. 27) for more information about obtaining a token.

cURL, a generally-available tool for retrieving URLs, can be used to connect to the notification stream. The following command is used to subscribe (cURL uses HTTP GET method by default):

```
curl -k https://<server>:8543/ean/api/v3/notifications/
subscriber -H    'Authorization:<token>'
```

where

server is the IP address of your NSD and NRC server

token is a previously-obtained authorization token

4.2.5 Inventory Management

The NSD and NRC APIs include REST CRUD operations that allows API clients to retrieve and where applicable create, update and delete infrastructure objects such as nodes, ports, TE links, IGP links, and applications that are associated with services and their related object information.

See the NSD and NRC API postman collection for sample requests to perform the following:

- Inventory management - retrieving Nodes, Physical links, service and Tunnels
- Filtering –retrieving services and nodes based on specific filtering criteria
- IETF TE –retrieving TE Networks, TE Links, Nodes and Termination Points

4.2.6 Service provisioning

API operations allows for the provisioning, modification, deletion and retrieval of both IP and Optical services that the NSD and NRCs support. These include services such as clines, elans, elines, el3-vpn, OCH/ODU, LAG and Service Tunnels. See the *NSP NSD and NRC Installation and User Guide* for more information on the service types and description of the IP and Optical services that NSP supports.

See the API postman collection for sample requests to perform the following IP and Optical Service provisioning examples:

E-Line Service Workflow

- retrieve list of tenants, retrieve ports for tenant which is then used to provision the E-Line service, finally verify service is successfully deployed.

Optical Service Workflow

- Retrieve ODU ports required to create ODU optical service and create an underlying OCH service if required followed by verifying service deployment and retrieving the OCH service created. Also included is a modification of the ODU Optical services parameters, such as latency and objective COST.

LAG Service Workflow

- retrieve ports that support LAG and then use those ports to create the LAG service, verify the service was deployed and modify the LAG service objective.

4.2.7 Template and policy provisioning

Both templates and policies can be created using the NSD and NRC APIs.

Service templates

Templates is another method by which an API client application can provision NSP-managed services. They simplify the configuration of the supported NSP IP and Optical Services that can be reused for service activation use-cases. See the *NSP NSD and NRC Installation and User Guide* for more information.

Note: If a service is provisioned using a template the parameters can be overridden by GUI and abstract API requests.

Policies

Optimization of the network can be achieved by using policies. This is done by enabling policy-driven behavior by managing the rules that allow the NSD and NRC API to customise network policies for routing algorithms selection and execution. The API also provides an interface to service definitions that exposes services in an abstract, customizable way while supporting the mediation of these definitions to the low-level network elements and resources.

The NSD and NRC API policy services allows API Clients to query, define and update the policies for tunnel selection, traffic steering, and RD/RT range modification for L3VPN.

See the following API postman collection for sample requests to perform the following Template and Policy provisioning examples:

L3 VPN Service Workflow Using Template

- Create a QoS Template that uses an existing (Generice QoS Profile) GQP on the NFM-P server.
- Create a L3 VPN service template that includes the previously created QoS template.
- Provision the L3VPN service by associating with the previously created service template and finally verify the service was successfully deployed.

- Clean up by deleting the Service and QoS templates.

Template Modification Workflow

- Identify permissible actions on selected tunnels based on constraints and objectives.

Rd/RT Policies

- Query and Modify Route Distinguisher/Route Target range (min and max values) for L3VPN.

E-Line Service Creation with Steering Parameters

- Create a steering parameter policy that will be used to mark the tunnel to be selected, retrieve bi-directional tunnels and assign the steering parameter to them.
- Create a tunnel selection policy with the steering parameters created previously.
- Create an E-Line service with the specific tunnel selection policy.

E-LAN Service Workflow Using GQP and QoS Overrides

- Create an E-LAN Service with the ID of an existing Generic QOS Profile (GQP) from the NFM-P. Verify the service is up,then update the Service with QoS Override. Verify the service once more.

4.2.8 Custom attributes

Custom attributes can be used to extend the NSD models and change the standard provisioning behavior towards devices. In create or modify requests for services or endpoints, the custom attributes can be provided as attributeName/attributeValue string lists. The NSD then passes these parameters to the device mediation layer, which can use them when configuring services and service endpoints. Custom attributes are supported for mediation by the NFM-P.

In a default NSP configuration, custom attributes can reference the NFM-P object model for services and endpoints. This allows a client of the NSP NBI to manipulate service and endpoint parameters in the NFM-P mediation layer, even if the parameters are not defined in the abstract NSD models, and thus, deploy highly customized services.

Additional processing logic for custom attributes can be implemented by installing scripts in the NFM-P mediation layer. These system scripts can realize even complex provisioning operations, and they can be uploaded on the fly without restarting the NFM-P. The combination of custom attributes and scripts is therefore a very powerful method to achieve flexible and customized service provisioning in NSD. The NFM-P scripts can be further fine-tuned by professional service teams for any future needs.

Custom attributes are supported on service and endpoint objects for all IP services, including E-Line, E-LAN, and L3 VPN. Custom attributes can be set and modified by REST API users. Custom attributes can also be persistently configured in NSD custom attribute templates. Custom attributes cannot be configured from the Service Fulfillment application.

Note: Values should not be supplied for custom attributes that are natively-configured by the NSP, such as the displayedName attribute. Providing values for custom attributes can cause some API requests to fail.

Note: Custom attributes are only supported on 5620 SAM Releases 14.0 and later, including NFM-P Release 17.3.

See the API postman collection for sample requests that includes custom attribute.

L3 VPN Service Workflow

- Retrieve ports that will be used to provision a L3 VPN service with custom attributes. In this example, the custom attributes include a description and the physical address for the service. This workflow continues by showing how to use the REST patch operation to modify an endpoint followed by adding an additional endpoint to the service. Followed by deleting the endpoint and finally deleting the service.

4.2.9 Task scheduling management

The Task Scheduling API service enables API clients to perform CRUD operations with respect to scheduling bandwidth modification requests/tasks on an existing E-Line service. API Clients can schedule single or repeatable tasks. After scheduling tasks, the API client can subsequently retrieve, modify, or delete existing tasks. In the case of modification, the user can change both the start date and the task execution intervals. The user can view their current requests and the state of those requests (Scheduled / Running / Disabled).

See the NSD and NRC API postman collection for sample requests for task scheduling:

Task Scheduler Workflow

1. Create scheduled tasks to increase bandwidth of E-Line service.
2. Schedule to revert the bandwidth to the original bandwidth.
3. Verify that tasks completed successfully.
4. Delete the scheduled task.

4.2.10 AS-based traffic optimization using steering parameters

The NSD and NRC module includes API services that allows API clients to perform traffic optimization by steering traffic on monitored routers, on a per-destination-AS-basis, to alternate next hops. Steering per destination AS implies that steering will be performed for all prefixes associated with a given destination AS. The NRC-F will automatically correlate the destination AS number to the set of prefixes associated with it. Steering is accomplished using the NRC-F Openflow controller, by automatically adding an Openflow flow rule per destination subnet. This allows the user to offload high traffic usage from the uplinks onto alternate paths on a per-AS-basis.

See the NSD and NRC API postman collection for sample requests for using Steering parameters:

Steering AS Parameters

- creates, retrieves and deletes AS Steering parameters

Steering Monitored Routers

- creates and add ports to a monitored router, delete monitored router

Steering Next Hop Bundles

- creates, retrieves and deletes next hop bundles

Steering VIP Parameters

- creates VIP customer steering parameter, add subnet to VIP customer then retrieve VIP customers and overlapping a subnet IP address. Deletes subnets from VIP customer and delete VIP customer.

4.2.11 Basic flow placement

The NSD and NRC module includes API services that allows an API client to manipulate flow rules, by sending requests to the NRC-F Openflow controller to add or delete flow rules. The API client can also retrieve the flow tables from the Openflow switches of any 7x50 router within a network.

See the API postman collection for sample requests for Flow placement:

Openflow

- Retrieve all openflow switches then retrieve a specific openflow switch by router ID. Create a cookie for the flow, create a flow that will redirect to a next hop, retrieve the ports and flow tables associated with the flows, finally delete the flow.

4.2.12 Using Application IDs

The Application ID property is a highly searchable field that allows for integration with preexisting infrastructure, or other systems that use object IDs. When Application IDs are used, correlation of objects can be performed quickly and without the need of full discovery.

To set an Application ID on an object, you must either configure the *appId* property at the time of its creation, or use the generic/application-id/{id}/{appId} REST call to modify it later. It is recommended that these IDs be prefixed with an application-specific prefix.

4.2.13 System operations

The NSD and NRC modules include API services that allow an API client to perform the following system operations:

- Retrieve the version and status of the system
- Determine which NSD and NRC server is the master
- Trigger Data synchronization with the connected NMS

5 Network and service assurance

5.1 Overview

5.1.1 Purpose

This chapter contains information that a developer can use to help with the development of applications for the following OSS domains:

- Network and service assurance
  - — Fault Management
  - — Service Supervision OAM

5.1.2 Contents

Network and service assurance

5.2 Network and service assurance

5.2.1 Fault Management API

The Fault Management application provides a means of investigating faults in a network. A user can obtain information about faults, such as their causes and impacts. The Fault Management application provides a list of published URLs that can be launched from OSS applications and a set of REST APIs designed for Fault Management OSS applications.

The Fault Management service is being deprecated, please contact your local Nokia representative.

5.2.2 Fault Management API use cases

See NSP API postman collection for common Fault Management use cases for retrieving root cause and impacts, as well as other use cases, for alarms.

5.2.3 Fault Management cross-launch

The Fault Management application publishes a list of URLs that allows an OSS application to perform various functions on NFM-P alarms. The full list of public URLs can be viewed at:

https://<*server*>:8543/FaultManagement/api-docs

where <server> is the hostname or IP address of the NSP server that hosts the API module.

The following are examples of the published URLs:

<*server*>:8543/FaultManagement/

<*server*>:8543/FaultManagement/?view=<*viewName*>

where *viewName* is one of: unhealthyNE, alarmList, neInspector, topProblems.

<*server*>:8543/FaultManagement/?view=alarmListImpacts&objectFullName=<*alarm_objectFullName*>

where *alarm_objectFullName* is a double URL-encoded alarm objectFullName.

5.2.4 Service Supervision OAM API

The Service Supervision OAM REST API is designed to help users understand the concepts and perform the tasks associated with Service Supervision OAM operations.

You can perform on-demand OAM testing by using the existing OAM functionality in the NFM-P through established on the NFM-P server.

SLA and on-demand OAM testing are supported on 7750 SR NEs, the 7705 SAR, and MPLS-capable 7210 SAS NEs.

Use of the OAM testing REST API requires knowledge of the NFM-P. Before you perform OAM testing using the REST API, you must configure test policies, LSPs, services, and test suites in the NFM-P. See the *NSP NFM-P User Guide* for more information.

5.2.5 Use cases

The SLA OAM testing API uses the NFM-P Service Test Manager framework to monitor LSPs and to provide data that network operators can use in the OSS SLA application.

The SLA OAM testing API uses four major components of the NFM-P STM framework:
- test suites
- test policy and threshold
- NFM-P schedule
- scheduled tasks

The system can raise alarms and events during the test execution. If you need to process these events through the OSS interface, you must develop a JMS client application to register the events. For more information about how to develop JMS clients, see the *NSP NFM-P XML API Developer Guide*.

5.2.6 OSS use case: Listing of services

The alarms and events are processed depending on the application design and configuration of the JMS clients. The threshold-crossing alarm includes the NFM-P FDN of the LSP in the event. All of the alarm and event messages contain the FDN of the affected NFM-P object. The Service Supervision list methods (*/rest/api/v1/oam/lsps /{lspFdn}/services*) can be used to return information about affected services.

For example, the LSPDOWN alarm contains the LSP FDN *lsp:from-98.120.169.44-id-1*. To determine which services are affected by this LSPDOWN event, use the following API:

GET http://*server*:8543/ServiceSupervision/rest/api/v1/oam/lsps/lsp:from-98.120.169.44-id-1/services where *server* is the hostname or IP address of the NSP server that hosts the API module.

You can use the following URL to retrieve services using an SDP tunnel:

```
GET http://<server>:8543/ServiceSupervision/rest/api/v1/oam/
sdpTunnels/{tunnelFdn}/services
```

where *server* is the hostname or IP address of the NSP server that hosts the API module.

When you know the service FDN, you can retrieve LSPs being used by the service, as well as the SDP tunnels being used by the service, as follows:

http://*server*:8543/ServiceSupervision/rest/api/v1/oam/services/{serviceFdn}/lsps

http://*server*:8543/ServiceSupervision/rest/api/v1/oam/services/{serviceFdn}/sdpTunnels

5.2.7 OSS use case: On-demand Ethernet CFM testing

You can perform an on-demand OAM test on a service by using the oneTimeTest API. A NFM-P service FDN is required to run this test as well as the type of OAM test to perform, the CFM test level, and the number of test probes.

The following test types are supported:
- CFM link trace
- CFM loopback
- CFM two-way delay
- CFM two-way SLM

The oneTimeTest API is based on the NFM-P quick-run test functionality. See the *NSP NFM-P User Guide* for information about how to configure and run ETH-OAM tests contextually. Issue the following request to initiate the test:

```
POST http://<server>:8543/ServiceSupervision/rest/api/v1/oam/
ethernetCfm/services/{serviceFdn}/oneTimeTest
```

where *server* is the hostname or IP address of the NSP server that hosts the API module.

The response of the oneTimeTest request contains the test suite FDN. You can retrieve the generated test of that test suite by issuing the following request:

```
GET http://<server>:8543/ServiceSupervision/rest/api/v1/oam/
ethernetCfm/testSuites/{testSuiteFdn}/testResults
```

where *server* is the hostname or IP address of the NSP server that hosts the API module.

5.2.8 OSS use case: On-demand OAM diagnostic testing

You can manage the networks and customer SLAs with the packet-based OAM Diagnostic API. These oneTimeTest API allows you to monitor network performance and aids in troubleshooting activities. The following tests are supported:

- ICMP ping
- LSP ping
- VCCV ping
- VPRN ping
- MTU ping
- ICMP trace
- LSP trace
- VCCV trace
- VPRN trace

This API submits a request to run a VCCV Ping on a VLL, VPLS, or MVPLS , and returns a oneTimeTestIdentifier. The oneTimeTestIdentifier is used to retrieve results after the test is finished.

```
POST http://<server>:8543/ServiceSupervision/rest/api/v1/oam/
ping/vccv/oneTimeTests
```

where *server* is the hostname or IP address of the NSP server that hosts the API module.

The following API submits a request to retrieve all the test results for the VCCV Ping with the specified oneTimeTestidentifier.

```
GET http://<server>:8543/ServiceSupervision/rest/api/v1/oam/ping/
vccv/oneTimeTests/<oneTimeTestIdentifier>/results
```

where *server* is the hostname or IP address of the NSP server that hosts the API module.

See NSP API postman collection for sample requests for performing Service Supervision OAM functions.

6 Network Functions Manager

6.1 Overview

6.1.1 NFM-P platform API

The Network Functions Manager - Packet, or NFM-P, provides a set of REST northbound interfaces that allows OSS applications to retrieve, create, update, or delete information on the managed objects in an NFM-P-managed network.

6.1.2 Prerequisites

The following are the prerequisites for using the NFM-P platform API.

- All API clients must first authenticate their client using the NSP REST Gateway; see Chapter 3, “NSP REST API Gateway”. Once a token is returned, it is used in subsequent NFM-P API service requests.
- You must have an understanding of the NFM-P Managed Object information model, specifically of the format of the objects, their hierarchy and how they are identified by their fully distinguished name (FDN). See the *NSP NFM-P XML API Developer Guide* and *XML API Reference* for more information.

6.1.3 NFM-P Managed Object Operations

The NFM-P Platform API has Managed Object services that allows API clients to perform the following requests on an NFM-P managed network.

- retrieve a list of specified managedObjects by class, optional to include a filter
- retrieve the managed object details by Full Distinguished Name (Fdn)
- retrieve the children of the object represented by the managed object Fdn
- create a child of an existing object with the option to configure values for child or children, and grandchildren
- remove existing objects and modify existing objects

See the NSP API postman collection for sample requests of the operations described above.

7 Backwards compatibility

7.1 Overview

7.1.1 General information

The methods of the NSP REST APIs evolve with each new release. This may result in some methods becoming deprecated or obsolete. See the table below for more information about the API changes introduced in this release of the NSP.

API	Changes
NSD and NRC	See the *NSP NSD and NRC API Differences Guide*
NFM-P	Redirecting from previous releases of the API is supported
Fault Management	All previous API methods are deprecated, with the exception of cross-launch
Service Supervision OAM	The *api/v1/oam/testSuite/testEntities* and *api/v1/oam/testResultsFile* methods are obsolete