Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Network Security Controller (NSC) APIs

[Document subtitle]

Table of Contents

[1 Background 1](#_Toc419409537)

[2 Proposal 2](#_Toc419409538)

[3 APIs 2](#_Toc419409539)

[3.1 API Version 2](#_Toc419409540)

[3.2 Service Function Ports 2](#_Toc419409541)

[3.2.1 List Service Function Ports 3](#_Toc419409542)

[3.2.2 Create (Onboard) Service Function Port 4](#_Toc419409543)

[3.2.3 Get Service Function Port (or query by Port Id) 4](#_Toc419409544)

[3.2.4 Delete (Off-board) Service Function Port 6](#_Toc419409545)

[3.3 Port Chains 6](#_Toc419409546)

[3.3.1 List Port Chains 6](#_Toc419409547)

[3.3.2 Create Port Chain 7](#_Toc419409548)

[3.3.3 Get Port Chain (or query by Inspected Port) 8](#_Toc419409549)

[3.3.4 Update Port Chain 9](#_Toc419409550)

[3.3.5 Delete Port Chain 10](#_Toc419409551)

# Background

SFC (Service Function Chaining) aim to address number of problem spaces and there are already number of active community initiatives to address them. Just to name a few main ones: 1) Definition of a service and service chain, 2) Service deployment (placement, resource allocation, and life cycle management), 3) Endpoint Grouping, 4) Chain realization/rendering.

SDN controller originally been designed to delegate logic and programmability to application. Recently, we see a trend which deviate from original intent and also try to deliver a ‘whole-in-one’ SFC solution as part of SDN controller.

As there are many use cases and permutations that would dictate different type of solution to address main challenges, it may be impractical to assume that SDN controller would be able to accommodate them all. Instead, it would be more practical to let SDN controller to do what it does best – network programmability. To that extent, the following proposed API attempt to define the lowest common denominator that any SFC must implemented regardless of solution choice or use case.

At the heart of SFC, an end point’s traffic needs to be steered to one or more network introspection service(s), in a certain order and optionally examined using a certain service policy. Any other SFC aspect mentioned earlier should be delegated to an ‘orchestration application’ to do the right thing per use case.

# Proposal

Below Diagram and API defines the notion of ‘Service Function Port’, ‘Inspected Port’ and ‘Port Chain’. ‘Service Function Port’ is a network interface designated to perform a service function by inspecting traffic. ‘Inspected Port’ is a workload end point port on which behalf traffic is being inspected. ‘Port Chain’ defines an association of Service Function Ports and Inspected Ports, along with other characteristics such as inspection policy tag (number), encapsulation type (VLAN, MPLS, VXLAN, NSH, GRE), encapsulation format (SCH), order (within the chain), and chaining failure type (Fail-open, Fail-close).

All ports identifiers are Neutron ports. This proposal’s scope is limited to the APIs (verbs and nouns) and does not discuss where to introduce the API and which component should be augmented. One possible suggestion would be to extend Neutron to expose these APIs and optionally allow the ML2 driver plugin to implement them. ML2 plugins such as OVS Agent, ODL and others can be enhanced to support this API.







# APIs

## API Version

List information for all NSC API versions and show details about API v1

|  |  |  |
| --- | --- | --- |
| **GET** | **/nsc** | **List API Versions** |

|  |  |  |
| --- | --- | --- |
| **GET** | **/v1.0** | **Show API V1.0 details** |

## Service Function Ports

List, show information for, create and delete Service Function Ports

### List Service Function Ports

|  |  |  |
| --- | --- | --- |
| **GET** | **/v1.0/nsc/servicefunctionports** | **List Service Function Ports** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | unauthorized (401) |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The Service Function Ports id (aka iface-id) |
| port\_chains | plain | xsd:dict | The associated Port Chains |
| id | Plain | csapi:uuid | The id of the Service Function Port |

Port Chains parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| service\_function\_port\_id | Plain | csapi:uuid |  |
| inspected\_port\_id | Plain | csapi:uuid |  |
| id | Plain | Csapi:uuid |  |
| order | Plain | xsd:integer |  |
| tag | Plain | xsd:integer |  |
| chain\_failure\_type | plain | xsd:string |  |

Example JSON response

{

"service\_function\_ports": [

{

"port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"port\_chains": [

{

"enc\_type": "vlan",

"enc\_format": "sch",

"service\_function\_port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"inpected\_port\_id": "5306720d-d3d6-46be-8c93-58a7a78333a1",

"id": "82c393b9-f127-4d7d-808f-371da48d13fa",

"order": 0,

"tag": 200,

"chain\_failure\_type": "FAIL\_OPEN"

}

**...**

],

"id": "0c7a9437-3354-4c7b-a793-8ec69487a6c7"

}

**...**

]

}

### Create (Onboard) Service Function Port

|  |  |  |
| --- | --- | --- |
| **POST** | **/v1.0/nsc/servicefunctionports** | **Create Service Function Port** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | badRequest(400), unauthorized (401) |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The port id to create the Service Function Port for |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The Service Function Port id (aka iface-id) |
| id | Plain | csapi:uuid | The id of the Service Function Port |

Example JSON request

{

"service\_function\_port": {

"port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40"

}

}

Example JSON response

{

"service\_function\_port": {

“port\_id”: "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"inspected\_ports": [],

"id": "109f96a7-038e-439a-bd02-c4c386a65a9f"

}

}

### Get Service Function Port (or query by Port Id)

|  |  |  |
| --- | --- | --- |
| **GET** | **/v1.0/nsc/servicefunctionports?port\_id=<port-id>** | **Show Service Function Port** |
| **GET** | **/v1.0/nsc/servicefunctionports/<id>** | **Show Service Function Port** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | unauthorized (401), itemNotFound(404) |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The port id to show inspection data for |
| id | Plain | csapi:uuid |  |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The Service Function Ports id (aka iface-id) |
| inspected\_ports | plain | xsd:dict | The associated Inspected Ports |
| Id | Plain | csapi:uuid | The id of the Service Function Port |

Inspected Port parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| port\_id | Plain | csapi:uuid |  |
| id | Plain | csapi:uuid |  |
| order | Plain | xsd:integer |  |
| tag | Plain | xsd:integer |  |

Example JSON response

{

"service\_function\_port": {

"port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"port\_chain": [

{

"enc\_type": "vlan",

"enc\_format": "sch",

"service\_function\_port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"inspected\_port\_id": "5306720d-d3d6-46be-8c93-58a7a78333a1",

"id": "82c393b9-f127-4d7d-808f-371da48d13fa",

"order": 0,

"tag": 200,

"chain\_failure\_type": "FAIL\_OPEN"

}

**...**

],

"id": "0c7a9437-3354-4c7b-a793-8ec69487a6c7"

}

}

### Delete (Off-board) Service Function Port

|  |  |  |
| --- | --- | --- |
| **DELETE** | **/v1.0/nsc/servicefunctionports/<port\_id>** | **Delete Service Function Port** |

|  |  |
| --- | --- |
| Normal response codes | 204 |
| Error response codes | unauthorized (401), itemNotFound(404), conflict(409) |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| port\_id | plain | csapi:uuid | The port id to delete |

## Port Chains

List, show information for, create, update and delete Port Chains

### List Port Chains

|  |  |  |
| --- | --- | --- |
| **GET** | **/v1.0/nsc/portchains** | **List Port Chains** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | unauthorized (401) |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| id | plain | csapi:uuid |  |
| inspected\_port\_id | plain | csapi:uuid |  |
| service\_function\_port\_id | plain | csapi:uuid |  |
| order | plain | xsd:integer |  |
| tag | plain | xsd:integer |  |
| chain\_failure\_type | plain | xsd:string |  |

Example JSON response

{

"port\_chains": [

{

"enc\_type": "vlan",

"enc\_format": "sch",

"id": "82c393b9-f127-4d7d-808f-371da48d13fa",

"inspected\_port\_id": "5306720d-d3d6-46be-8c93-58a7a78333a1",

"service\_function\_port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"order": 0,

"tag": 200,

"chain\_failure\_type": "FAIL\_OPEN"

}

**...**

]

}

### Create Port Chain

|  |  |  |
| --- | --- | --- |
| **POST** | **/v1.0/nsc/portchains** | **Create Port Chain** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | badRequest(400), unauthorized (401) |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| inspected\_port\_id | plain | csapi:uuid |  |
| service\_function\_port\_id | plain | csapi:uuid |  |
| order | plain | xsd:integer |  |
| tag | plain | xsd:integer |  |
| chain\_failure\_type | plain | xsd:string |  |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| Id | plain | csapi:uuid | Newly created Port Chain id |

Example JSON request

{

"port\_chain" : {

"service\_function\_port\_id" : "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"inspected\_port\_id" : "5306720d-d3d6-46be-8c93-58a7a78333a1",

"tag" : 200,

"order" : 0,

"enc\_type" : "vlan",

"enc\_format": "sch",

"chain\_failure\_type": "FAIL\_OPEN"

}

}

Example JSON response

{

"port\_chain": {

“id”: "81fedbd3-78ad-4307-8414-c147b5533bf3"

}

}

### Get Port Chain (or query by Inspected Port)

|  |  |  |
| --- | --- | --- |
| **GET** | **/v1.0/nsc/portchains/<id>** | **Show Port Chain** |
| **GET** | **/v1.0/nsc/portchains?inspected\_port\_id=<port-id>&service\_function\_port\_id=<port-id>** | **List Port Chains by Inspected Port and optionally by Service Function Port** |

|  |  |
| --- | --- |
| Normal response codes | 200 |
| Error response codes | unauthorized (401), itemNotFound(404) |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| id | plain | csapi:uuid | The Port Chain id to show information for |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| id | plain | csapi:uuid |  |
| inspected\_port\_id | plain | csapi:uuid |  |
| service\_function\_port\_id | plain | csapi:uuid |  |
| order | plain | xsd:integer |  |
| tag | plain | xsd:integer |  |
| chain\_failure\_type | plain | xsd:string |  |

Example JSON response

{

"port\_chain": {

"enc\_type": "gre",

"enc\_format": "sch",

“id”: "81fedbd3-78ad-4307-8414-c147b5533bf3",

"inspected\_port\_id": "5306720d-d3d6-46be-8c93-58a7a78333a1",

"service\_function\_port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"order": 50,

"tag": 210,

"chain\_failure\_type": "FAIL\_OPEN"

}

}

### Update Port Chain

|  |  |  |  |
| --- | --- | --- | --- |
| **PUT** | **/v1.0/nsc/portchains/<id>** | | **Update Port Chain** |
| Normal response codes | | 200 | |
| Error response codes | | badRequest(400), unauthorized (401), itemNotFound(404) | |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string | The new encoding type |
| enc\_format | Plain | xsd:string |  |
| order | plain | xsd:integer | The new order |
| tag | plain | xsd:integer | The new tag |

Response parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| enc\_type | plain | xsd:string |  |
| enc\_format | Plain | xsd:string |  |
| id | plain | csapi:uuid |  |
| inspected\_port\_id | plain | csapi:uuid |  |
| service\_function\_port\_id | plain | csapi:uuid |  |
| order | plain | xsd:integer |  |
| tag | plain | xsd:integer |  |
| chain\_failure\_type | plain | xsd:string |  |

Example JSON request

{

"port\_chain": {

"tag" : 210,

"order" : 50,

"enc\_type" : "gre",

"chain\_failure\_type": "FAIL\_OPEN"

}

}

Example JSON response

{

"port\_chain": {

"enc\_type": "gre",

"enc\_format": "sch",

“id”: "81fedbd3-78ad-4307-8414-c147b5533bf3",

"inspected\_port\_id": "5306720d-d3d6-46be-8c93-58a7a78333a1",

"service\_function\_port\_id": "d4d066c8-16d3-4fd1-b7ec-6c4d34790b40",

"order": 50,

"tag": 210,

"chain\_failure\_type": "FAIL\_OPEN"

}

}

### Delete Port Chain

|  |  |  |  |
| --- | --- | --- | --- |
| **DELETE** | **/v1.0/nsc/portchains/<id>** | | **Delete Port Chain** |
| Normal response codes | | 204 | |
| Error response codes | | unauthorized (401), itemNotFound(404), conflict(409) | |

Request parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| id | plain | csapi:uuid | The Port Chain id to delete |