
Abstract

Netfloc application API.

Table of Contents

Abstract	1
Table of Contents	2
API Specification	4
Default	4
Network Graph [/networkGraph]	4
Get the Network Graph - GET /networkGraph	4
Tenant filtered Network Graph [/networkGraph/{tenant}]	4
Get the tenant filtered Network Graph - GET /networkGraph/{tenant}	4
Tenants [/tenant]	4
Get all tenants - GET /tenant	4
Host Ports [/hostPort]	5
Get all Host Ports - GET /hostPort	5
Network Path [/networkPath/{src}/{dst}]	5
Get a Network Path between source and destination ports - GET /networkPath/{src}/{dst}	5
Flow Patterns [/flowPattern]	5
Get all flow patterns - GET /flowPattern	5
Create a new flow pattern - POST /flowPattern	5
Flow Pattern [/flowPattern/{id}]	6
Get flow pattern by id - GET /flowPattern/{id}	6
Update flow pattern by id - PUT /flowPattern/{id}	6
Delete flow pattern by id - DELETE /flowPattern/{id}	6
Install and delete Flow Patterns on Network Paths [/flowPattern/{id}/{src}/{dst}]	7
Post a flow pattern onto a network path - POST /flowPattern/{id}/{src}/{dst}	7
Delete a flow pattern on a network path - DELETE /flowPattern/{id}/{src}/{dst}	7
Examples	7
Default	7
Network Graph [/networkGraph]	7
Get the Network Graph - GET /networkGraph	7
Tenant filtered Network Graph [/networkGraph/{tenant}]	8
Get the tenant filtered Network Graph - GET /networkGraph/{tenant}	8
Tenants [/tenant]	8
Get all tenants - GET /tenant	9
Host Ports [/hostPort]	9
Get all Host Ports - GET /hostPort	9
Network Path [/networkPath/{src}/{dst}]	10
Get a Network Path between source and destination ports - GET /networkPath/{src}/{dst}	10
Flow Patterns [/flowPattern]	11
Get all flow patterns - GET /flowPattern	11
Create a new flow pattern - POST /flowPattern	12
Flow Pattern [/flowPattern/{id}]	13
Get flow pattern by id - GET /flowPattern/{id}	13
Update flow pattern by id - PUT /flowPattern/{id}	15
Delete flow pattern by id - DELETE /flowPattern/{id}	16
Install and delete Flow Patterns on Network Paths [/flowPattern/{id}/{src}/{dst}]	17
Post a flow pattern onto a network path - POST /flowPattern/{id}/{src}/{dst}	18
Delete a flow pattern on a network path - DELETE /flowPattern/{id}/{src}/{dst}	19

API Specification

Default

Network Graph [/networkGraph]

Get the Network Graph - GET /networkGraph

Response 200 (application/json)

[Go to example](#)

Tenant filtered Network Graph [/networkGraph/{tenant}]

Parameters

tenant (Required, string)

Name of the tenant

Get the tenant filtered Network Graph - GET /networkGraph/{tenant}

Response 200 (application/json)

[Go to example](#)

Tenants [/tenant]

Get all tenants - GET /tenant

Response 200 (application/json)

[Go to example](#)

Host Ports [/hostPort]

Get all Host Ports - GET /hostPort

Response 200 (application/json)

[Go to example](#)

Network Path [/networkPath/{src}/{dst}]

Get a Network Path between source and destination ports - GET /networkPath/{src}/{dst}

Response 200 (application/json)

[Go to example](#)

Flow Patterns [/flowPattern]

Get all flow patterns - GET /flowPattern

Response 200 (application/json)

[Go to example](#)

Create a new flow pattern - POST /flowPattern

Request (application/json)

Response 201 (application/json)

Response 400 (application/json)

[Go to example](#)

Flow Pattern [/flowPattern/{id}]

Get flow pattern by id - GET /flowPattern/{id}

Response 200 (application/json)

Response 404 (application/json)

Response 410 (application/json)

[Go to example](#)

Update flow pattern by id - PUT /flowPattern/{id}

Request (application/json)

Response 200 (application/json)

Response 400 (application/json)

Response 404 (application/json)

Response 410 (application/json)

[Go to example](#)

Delete flow pattern by id - DELETE /flowPattern/{id}

Response 200 (application/json)

Response 409 (application/json)

Response 404 (application/json)

[Go to example](#)

Install and delete Flow Patterns on Network Paths [/flowPattern/{id}/{src}/{dst}]

Post a flow pattern onto a network path - POST /flowPattern/{id}/{src}/{dst}

Response 400 (application/json)

Response 201 (application/json)

Response 404 (application/json)

Response 410 (application/json)

Response 409 (application/json)

[Go to example](#)

Delete a flow pattern on a network path - DELETE /flowPattern/{id}/{src}/{dst}

Response 200 (application/json)

Response 404 (application/json)

Response 404 (application/json)

Response 410 (application/json)

[Go to example](#)

Examples

Default

Network Graph [/networkGraph]

Get the Network Graph - GET /networkGraph

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
```

```
"status": "Network Graphs retrieved",
"data": [
  {
    "linkPorts": [],
    "internalPorts": [],
    "hostPorts": []
  }
]
```

[Go to specification](#)

Tenant filtered Network Graph [/networkGraph/{tenant}]

Parameters

tenant (Required, string)

Name of the tenant

Get the tenant filtered Network Graph - GET /networkGraph/{tenant}

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "filtered Network Graph retrieved",
  "data": [
    {
      "externalPorts": [
        { "link": ??? }
      ],
      "internalPorts": [],
      "hostPorts": [
        { "tenant": tenant }
      ]
    }
  ]
}
```

[Go to specification](#)

Tenants [/tenant]

Get all tenants - GET /tenant

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "tenant list retrieved",
  "data": [
    "tenant1", "tenant2", "tenant3"
  ]
}
```

[Go to specification](#)

Host Ports [/hostPort]

Get all Host Ports - GET /hostPort

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "all host ports retrieved",
  "data": [
    {
      "ofPort": 1,
      "tenant": "tenant1",
      "l2Address": "ff:ff:ff:ff:ff:ff",
      "l3Addresses": [
        "192.168.0.1"
      ]
    }
  ]
}
```

[Go to specification](#)

Network Path [/networkPath/{src}/{dst}]

Get a Network Path between source and destination ports - GET /networkPath/{src}/{dst}

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{  "status": "Network path retrieved",
  "data": {
    "flowPatterns" [],
    "path" :[
      {
        "srcHostPort": {
          "ofPort": 1,
          "tenant": "tenant1",
          "l2Address": "00:00:00:00:00:01",
          "l3Addresses": [
            "192.168.0.1"
          ]
        },
        "nextLinkPort": {
          "ofPort": 3
        }
      },
      {
        "previousLinkPort": {
          "ofPort": 1
        },
        "nextLinkPort": {
          "ofPort": 3
        }
      },
      {
        "previousLinkPort": {
          "ofPort": 3
        },
        "dstHostPort": {
          "ofPort": 1,
          "tenant": "tenant1",
          "l2Address": "00:00:00:00:00:02",
          "l3Addresses": [
            "192.168.0.2"
          ]
        }
      }
    ]
  }
}
```

Flow Patterns [/flowPattern]

*Get all flow patterns - GET /flowPattern***Response 200** (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flow pattern retrieved",
  "data": [
    {
      "srcBridge": [
        {
          "match": {
            "in_port": "srcHostPort",
            "hw_src": "srcMAC",
            "hw_dst": "dstMAC"
          },
          "actions": {
            "output": "nextLinkPort"
          }
        }
      ],
      "linkBridge": [
        {
          "match": {
            "hw_dst": "dstMAC"
          },
          "actions": {
            "output": "nextLinkPort"
          }
        }
      ],
      "dstBridge": [
        {
          "match": {
            "hw_dst": "dstMAC"
          },
          "actions": {
            "output": "dstHostPort"
          }
        }
      ]
    }
  ]
}
```

```
}
```

[Go to specification](#)

Create a new flow pattern - POST /flowPattern

Request (application/json)

Headers

Content-Type: application/json

Body

```
{
  "srcBridge": [
    {
      "match": {
        "in_port": "srcHostPort",
        "hw_src": "srcMAC",
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "nextLinkPort"
      }
    }
  ],
  "linkBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "nextLinkPort"
      }
    }
  ],
  "dstBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "dstHostPort"
      }
    }
  ]
}
```

Response 201 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern created",
  "id": 2
}
```

Response 400 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "request malformed"
}
```

[Go to specification](#)

Flow Pattern [/flowPattern/{id}]

Get flow pattern by id - GET /flowPattern/{id}

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flow pattern retrieved"
  "data": {
    "srcBridge": [
      {
        "match": {
          "in_port": "srcHostPort",
          "hw_src": "srcMAC",
          "hw_dst": "dstMAC"
        },
        "actions": {
```

```

        "output": "nextLinkPort"
      }
    }
  ],
  "linkBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "nextLinkPort"
      }
    }
  ],
  "dstBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "dstHostPort"
      }
    }
  ]
}

```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```

{
  "status": "flowPattern does not exist"
}

```

Response 410 (application/json)

Headers

Content-Type: application/json

Body

```

{
  "status": "flowPattern is deleted permanently"
}

```

Update flow pattern by id - PUT /flowPattern/{id}

Request (application/json)

Headers

Content-Type: application/json

Body

```
{
  "srcBridge": [
    {
      "match": {
        "in_port": "srcHostPort",
        "hw_src": "srcMAC",
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "nextLinkPort"
      }
    }
  ],
  "linkBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "nextLinkPort"
      }
    }
  ],
  "dstBridge": [
    {
      "match": {
        "hw_dst": "dstMAC"
      },
      "actions": {
        "output": "dstHostPort"
      }
    }
  ]
}
```

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern updated"
}
```

Response 400 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "request malformed"
}
```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern does not exist"
}
```

Response 410 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern has been deleted permanently"
}
```

[Go to specification](#)

Delete flow pattern by id - DELETE /flowPattern/{id}

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern is deleted successfully"
}
```

Response 409 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "transaction condition not met, flowPattern is installed on networkPath(s)",
  "data": {
    "networkPaths": [
    ]
  }
}
```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern does not exist"
}
```

[Go to specification](#)

Post a flow pattern onto a network path - POST /flowPattern/{id}/{src}/{dst}

Response 400 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "request malformed"
}
```

Response 201 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern is posted onto networkPath"
}
```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```
{
  "status": "flowPattern does not exist"
}
```

Response 410 (application/json)

Headers

Content-Type: application/json

Body

```
{
```

```
{  
  "status": "flowPattern has been deleted permanently"  
}
```

Response 409 (application/json)

Headers

Content-Type: application/json

Body

```
{  
  "status": "flowPattern is not already installed"  
}
```

[Go to specification](#)

Delete a flow pattern on a network path - DELETE /flowPattern/{id}/{src}/{dst}

Response 200 (application/json)

Headers

Content-Type: application/json

Body

```
{  
  "status": "flowPattern successfully deleted from network path"  
}
```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```
{  
  "status": "flowPattern not found on network path"  
}
```

Response 404 (application/json)

Headers

Content-Type: application/json

Body

```
{  
  "status": "flowPattern does not exist"  
}
```

Response 410 (application/json)

Headers

Content-Type: application/json

Body

```
{  
  "status": "flowPattern already deleted on network path"  
}
```

[Go to specification](#)