

FIWARE-Netfloc Open Specification

DATE: 11 September 2015

View in Apiary 

Editors

- Denis Baudinot, ZHAW
- Irena Trajkovska, ZHAW

Copyright

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to the publisher, addressed "Attention: Permissions Coordinator," at the address below.

Abstract

This specification defines the FIWARE-Netfloc GE API. The APIs expose the network filtered per tenant. This includes retrieval of network graph, interfaces and network path - all filtered per tenant. It also includes APIs to define network flow patterns and perform CRUD operations on such flow patterns over the network paths.

Status of this document

This is a work in progress and is changing on a daily bases. This specification is licensed under the [FIWARE Open Specification License](http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Open_Specification_Legal_Notice_%28implicit_patents_license%29) (http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Open_Specification_Legal_Notice_%28implicit_patents_license%29) .

Table of Contents

FIWARE-Netfloc Open Specification	1
Editors	1
Copyright	1
Abstract	1
Status of this document	1
Table of Contents	2
API Summary	3
Netfloc	4
API Specification	5
Default	5
Network Graph	5
Get the Network Graph	5
Tenant filtered Network Graph	5
Get the tenant filtered Network Graph	5
Tenants	6
Get all tenants	6
Host Ports	6
Get all Host Ports	6
Network Path	6
Get a Network Path between source and destination ports	6
Flow Patterns	7
Get all flow patterns	7
Create a new flow pattern	7
Flow Pattern	7
Get flow pattern by id	7
Update flow pattern by id	8
Delete flow pattern by id	8
Install and delete Flow Patterns on Network Paths	9
Post a flow pattern onto a network path	9
Delete a flow pattern on a network path	9
Examples	9
Default	10
Network Graph	10
Get the Network Graph	10
Tenant filtered Network Graph	10
Get the tenant filtered Network Graph	10
Tenants	11
Get all tenants	11
Host Ports	11
Get all Host Ports	11
Network Path	12
Get a Network Path between source and destination ports	12
Flow Patterns	13
Get all flow patterns	13
Create a new flow pattern	14
Flow Pattern	16
Get flow pattern by id	16
Update flow pattern by id	17
Delete flow pattern by id	19
Install and delete Flow Patterns on Network Paths	20
Post a flow pattern onto a network path	20
Delete a flow pattern on a network path	22
References	24

API Summary

- Default
 - Network Graph
 - GET - Get the Network Graph [/networkGraph]
 - Tenant filtered Network Graph
 - GET - Get the tenant filtered Network Graph [/networkGraph/{tenant}]
 - Tenants
 - GET - Get all tenants [/tenant]
 - Host Ports
 - GET - Get all Host Ports [/hostPort]
 - Network Path
 - GET - Get a Network Path between source and destination ports
[/networkPath/{src}/{dst}]
 - Flow Patterns
 - GET - Get all flow patterns [/flowPattern]
 - POST - Create a new flow pattern [/flowPattern]
 - Flow Pattern
 - GET - Get flow pattern by id [/flowPattern/{id}]
 - PUT - Update flow pattern by id [/flowPattern/{id}]
 - DELETE - Delete flow pattern by id [/flowPattern/{id}]
 - Install and delete Flow Patterns on Network Paths
 - POST - Post a flow pattern onto a network path
[/flowPattern/{id}/{src}/{dst}]
 - DELETE - Delete a flow pattern on a network path
[/flowPattern/{id}/{src}/{dst}]

Netfloc

Netfloc application API.

API Specification

Default

Network Graph [/networkGraph]

Get the Network Graph

GET /networkGraph

Response 200 (application/json)

[Go to example](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/network-graph/get-the-network-graph>)

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](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/tenant-filtered-network-graph/get-the-tenant-filtered-network-graph>)

Tenants [/tenant]

Get all tenants

GET /tenant

Response 200 (application/json)

[Go to example](#)

[View in Apiary \(http://docs.Netfloc.apiary.io/#reference/default/tenants/get-all-tenants\)](http://docs.Netfloc.apiary.io/#reference/default/tenants/get-all-tenants)

Host Ports [/hostPort]

Get all Host Ports

GET /hostPort

Response 200 (application/json)

[Go to example](#)

[View in Apiary \(http://docs.Netfloc.apiary.io/#reference/default/host-ports/get-all-host-ports\)](http://docs.Netfloc.apiary.io/#reference/default/host-ports/get-all-host-ports)

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](#)

[View in Apiary \(http://docs.Netfloc.apiary.io/#reference/default/network-path/get-a-network-path-between-source-and-destination-ports\)](http://docs.Netfloc.apiary.io/#reference/default/network-path/get-a-network-path-between-source-and-destination-ports)

Flow Patterns [/flowPattern]

[Get all flow patterns](#)

GET /flowPattern

Response 200 (application/json)

[Go to example](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/flow-patterns/get-all-flow-patterns>)

[Create a new flow pattern](#)

POST /flowPattern

Request (application/json)

Response 201 (application/json)

Response 400 (application/json)

[Go to example](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/flow-patterns/create-a-new-flow-pattern>)

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](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/flow-pattern/get-flow-pattern-by-id>)

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](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/flow-pattern/update-flow-pattern-by-id>)

Delete flow pattern by id

DELETE /flowPattern/{id}

Response 200 (application/json)

Response 409 (application/json)

Response 404 (application/json)

[Go to example](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/flow-pattern/delete-flow-pattern-by-id>)

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](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/install-and-delete-flow-patterns-on-network-paths/post-a-flow-pattern-onto-a-network-path>)

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](#)

View in Apiary (<http://docs.Netfloc.apiary.io/#reference/default/install-and-delete-flow-patterns-on-network-paths/delete-a-flow-pattern-on-a-network-path>)

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}]

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"
      ]
    }
  }
]
}

```

[Go to specification](#)

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"
}
```

[Go to specification](#)

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](#)

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)

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 allready 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](#)

References

- [FIWARE Open Specification License](http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Open_Specification_Legal_Notice_%28implicit_patents_license%29)
([http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Open_Specification_Legal_Notice_%28implicit_patents_license%29](http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Open_Specification_Legal_Notice_%28implicit_patents_license%29))