BROCADE

API REFERENCE GUIDE

Brocade Flow Optimizer REST API Guide

Supporting Flow Optimizer 1.2

53-1004295-01
9 May 2016

Contents

Preface

Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Brocade technical documentation.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used in the flow of the text to highlight specific words or phrases.

Format	Description
bold text	Identifies command names
	Identifies keywords and operands
	Identifies the names of user-manipulated GUI elements
	Identifies text to enter at the GUI
italic text	Identifies emphasis
	Identifies variables
	Identifies document titles
`Courier font`	Identifies CLI output
	Identifies command syntax examples

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
value	In Fibre Channel products, a fixed value provided as input to a command option is printed in plain text, for example, **--show** WWN.
[ ]	Syntax components displayed within square brackets are optional.
	Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
	In Fibre Channel products, square brackets may be used instead for this purpose.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.

Convention	Description
...	Repeat the previous element, for example, *member*[*member*...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION
A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER
A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Brocade resources

Visit the Brocade website to locate related documentation for your product and additional Brocade resources.

You can download additional publications supporting your product at www.brocade.com. Select the Brocade Products tab to locate your product, then click the Brocade product name or image to open the individual product page. The user manuals are available in the resources module at the bottom of the page under the Documentation category.

To get up-to-the-minute information on Brocade products and resources, go to MyBrocade. You can register at no cost to obtain a user ID and password.

Release notes are available on MyBrocade under Product Downloads.

White papers, online demonstrations, and data sheets are available through the Brocade website.

Contacting Brocade Technical Support

As a Brocade customer, you can contact Brocade Technical Support 24x7 online, by telephone, or by e-mail. Brocade OEM customers contact their OEM/Solutions provider.

Brocade customers

For product support information and the latest information on contacting the Technical Assistance Center, go to http://www.brocade.com/services-support/index.html.

If you have purchased Brocade product support directly from Brocade, use one of the following methods to contact the Brocade Technical Assistance Center 24x7.

Online	Telephone	E-mail
Preferred method of contact for non-urgent issues: • My Cases through MyBrocade • Software downloads and licensing tools • Knowledge Base	Required for Sev 1-Critical and Sev 2-High issues: • Continental US: 1-800-752-8061 • Europe, Middle East, Africa, and Asia Pacific: +800-AT FIBREE (+800 28 34 27 33) • For areas unable to access toll free number: +1-408-333-6061 • Toll-free numbers are available in many countries.	support@brocade.com Please include: • Problem summary • Serial number • Installation details • Environment description

Brocade OEM customers

If you have purchased Brocade product support from a Brocade OEM/Solution Provider, contact your OEM/Solution Provider for all of your product support needs.

- OEM/Solution Providers are trained and certified by Brocade to support Brocade® products.
- Brocade provides backline support for issues that cannot be resolved by the OEM/Solution Provider.
- Brocade Supplemental Support augments your existing OEM support contract, providing direct access to Brocade expertise. For more information, contact Brocade or your OEM.
- For questions regarding service levels and response times, contact your OEM/Solution Provider.

Document feedback

To send feedback and report errors in the documentation you can use the feedback form posted with the document or you can e-mail the documentation team.

Quality is our first concern at Brocade and we have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission, or you think that a topic needs further development, we want to hear from you. You can provide feedback in two ways:

- Through the online feedback form in the HTML documents posted on www.brocade.com.
- By sending your feedback to documentation@brocade.com.

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

Overview of the Flow Optimizer REST API

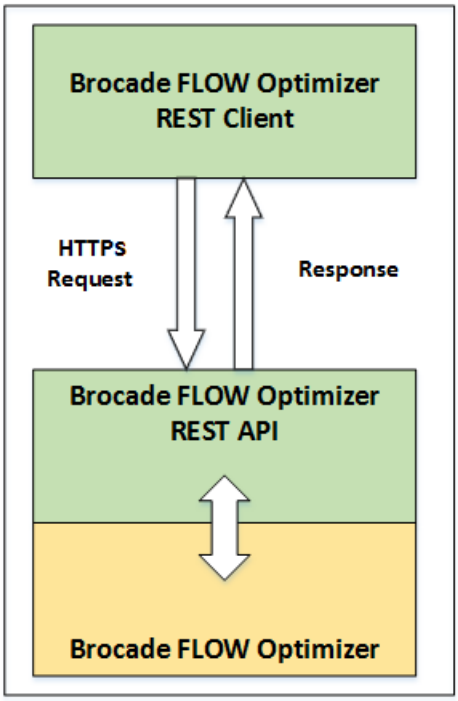
Introduction

This document details the REST API, which is a programmatic interface supported in the Brocade Flow Optimizer (BFO) for Release 1.2. This interface is built for third party consumption as well as BFO clients.

The Brocade Flow Optimizer REST API exposes services perform CRUD operation for profiles, and other higher level services, including but not limited to retrieval of the following data:

- Performance data
- Events

This document covers details on the Brocade REST API interface, including basics to higher level functionality within the scope of this release.



Support Requirements and Objectives

This topic covers a list of support requirements and objectives for the Brocade Flow Optimizer (BFO) REST API.

Requirements

BFO support requirements include but are not limited to the following.

- Support REST interface for getting the list of Profiles.
- Support REST interface for Create / Edit / Delete / Enable / Disable / Change Priority of the Custom Profiles.

- Support REST interface to get list of EVENTS with filter options.
- Support REST interface for getting different settings in BFO.
- Support REST interface for Add / Edit / Delete different settings in BFO.
- Support REST interface for Edit / Delete user defined custom flows operations in BFO.
- Support REST interface for obtaining the Historical and Real time performance data.
- Support for both XML and JSON content types.

Objectives

BFO support objectives include but are not limited to the following.

- Design a REST API model.
- Define the URIs in accordance with this model for those services within the scope of this release.
- Define and generate the response schema for the request inputs and response outputs.
- Support both XML and JSON payloads based on user input for all REST based requests.
- Support authentication of every web service request.

URI Design

Brocade Flow Optimizer (BFO) URIs were designed hierarchically with HTTPS architecture, and specified with HTTP **Accept** and **WSToken**.

BFO URIs have following objectives:

- The URIs follow a hierarchical structure to support containment and also retrieve any specific object or service related to that object.
- All operations are HTTPS requests (Get / Post / Put / Delete).
- The HTTP request header **Accept** must be specified to indicate the content type of the response payload.
- The HTTP request header **WSToken** must be specified with the token received from login.
- The request and response schema will be provided as part of the build.
- URIs are case sensitive.

HTTPS Base URI

The Brocade Flow Optimizer (BFO) base URI protocol support is exclusively supported for HTTPS, and not for HTTP. REST API HTTPS requests are supported.

All REST operations will be provided at the following base context path.

TABLE 1 HTTPS Protocol Support

HTTPS Default Port	HTTPS URL
8089	https://<BFO Server IP>:8089

Getting Started

Authentication and Session Management

Authentication and Session Management Overview

Brocade Flow Optimizer (BFO) authenticated users can access web services for authentication and session management, requiring a login (consisting of a valid non-empty password.)

The REST interface provides two operations:

- Login to a client session.
- Logout of a client session.

Login

This topic covers steps for logging in to the Brocade Flow Optimizer (BFO) server.

Resource URL
`<Base URI>/collector/login`

All REST operations will be provided at the following base context path in the URI field of your REST client tool.

To login to the BFO server, complete the following steps.

1. Login to the BFO server by entering the following URI in the URL field of your REST client tool: https://localhost:8089/collector/login
2. Define the login parameters.
3. Set the HTTPS request method to POST.

Login Parameters

This topic covers login parameters for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Username	The valid BFO user name.
Password	Valid clear text password.

NOTE

The POST operation is supported.

Login Examples

This topic covers login examples for the Brocade Flow Optimizer (BFO) server.

Login examples for BFO POST operation for login include:

- BFO URI
- BFO Login request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>/collector/login
```

Example of Login Request Headers Sent to BFO

The following is an example of login request headers sent.

```
JSON

        Request header

        {
        Accept - Application/json
        Content-Type - Application/json
        Request payload
        }

        Request body

        {
        "username": "Administrator",
        "password": "pass"
        }

        Response body

        {
        "token": "Administrator_1426620456948",
        "status": "Login success"
        }

XML:

        Request header

        Accept - Application/xml
        Content-Type - Application/xml

        Request payload

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <login>
        <username>Administrator</username>
        <password>pass</password>
        </login>

        Response body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <login>
        <token>Administrator_1426270596791</token>
        <status>Login success</status>
        </login>
```

TABLE 2 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
401 Unauthorized	FAILED, reason included in HTTPS error response.

If the login request is successful, BFO will send the authorized token, which will be used for the other REST operations.

NOTE

Login object contains the token that can be used for authorization of other REST requests.

Logout

This topic covers steps for logging out of the Brocade Flow Optimizer (BFO) server.

Resource URL
<Base URI>/collector/logout

All REST operations will be provided at the following base context path in the URI field of your REST client tool.

To logout from the BFO server, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/logout
2. Define the logout parameters.
3. Set the HTTPS request method to POST.

Logout Parameters

This topic covers logout parameters for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.

NOTE
The POST operation is supported.

Logout Examples

Logout examples for the Brocade Flow Optimizer (BFO) server.

Logout examples for BFO POST operation for logout include:

- BFO URI
- BFO Logout request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>:8089/collector/logout
```

Example of Login Request Headers Sent to BFO

The following is an example of logout request headers sent.

```
JSON

        Request header

        {
        Accept - Application/json
        Content-Type - Application/json
        Authorization - Administrator_1426270596791
        }

        Response body

        {
        "status": "Session Administrator_1426270596791 removed"
        }

XML

        Request header

        Accept - Application/xml
        Content-Type - Application/xml
        Authorization - Administrator_1426270596791

        Response body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <logout>
        <status>Session Administrator_1426270596791 removed</status>
        </logout>
```

TABLE 3 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
401 Unauthorized	FAILED, reason included in HTTPS error response

User Management

User Management Overview

User Management URIs allow users to return a list of users, create a new user, edit a user, and delete a user in the Brocade Flow Optimizer (BFO).

The REST interface provides these operations:

- Returning a list of users.
- Creating new users.
- Editing new users.
- Deleting new users.

Get List of Users

Get the list of users created in the Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/users

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To get the list of users created in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/users
2. Define the list of users parameters to get.
3. Set the HTTPS request method to GET.

Get List of Users Parameters

This topic covers retrieving the list of users parameters for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE

The GET operation is supported.

Get List of Users Examples

Get list of users examples for BFO GET operation include:

- BFO URI
- BFO get users request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>:8089/collector/users
```

Example of Login Request Headers Sent to BFO

The following is an example of get list of users headers sent.

JSON

```
        Request header

        {
        Authorization - Administrator_1426270596791
        Accept - Application/json
        }

        Response body

        {
        "user": [
        {
        "name": "Administrator",
        "access_privilege": -1,
        "created_on": "Tue Dec 23 09:24:50 PST 2014",
        "last_modified_on": "Tue Dec 23 09:24:50 PST 2014"
        },
        {
        "name": "Guest",
        "access_privilege": 0,
        "created_on": "Fri Mar 13 10:31:33 PDT 2015",
        "last_modified_on": "Fri Mar 13 10:31:33 PDT 2015"
        }
        ]
        }
```

XML

```
        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/xml

        Response body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <users>
        <user>
        <name>Administrator</name>
        <access_privilege>-1</access_privilege>
        <created_on>Tue Dec 23 09:24:50 PST 2014</created_on>
        <last_modified_on>Tue Dec 23 09:24:50 PST 2014</last_modified_on>
        </user>
        <user>
        <name>Guest</name>
        <access_privilege>0</access_privilege>
        <created_on>Fri Mar 13 10:31:33 PDT 2015</created_on>
        <last_modified_on>Fri Mar 13 10:31:33 PDT 2015</last_modified_on>
        </user>
        </users>
```

TABLE 4 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS, on successful code the list of users are returned
401 Unauthorized	FAILED, reason included in HTTPS error response

Create a New User

This topic covers the process of creating a new user in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/users/<user name>

All REST operations will be provided at the base context path in the URI field of your REST client tool.

Before creating a new user in the BFO, criteria for creating a user are as follows:

- Only a user with Admin privileges (access_privilege=1) is allowed to create a new user.
- Valid username name and password are required as input.
- Access privilege is optional, and if not provided, created user will have operational privileges (access_privilege=0).
- A user cannot be created with same name as another user.

NOTE

BFO is limited to one default administrator. The default administrator is the only BFO user with administrative privileges. The default administrator cannot assign administrative privileges to other users.

To create a new user in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/users/<user name>
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Creating New User Parameters

This topic covers parameters for creating a new user for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE

The POST operation is supported.

Creating New User Examples

This topic covers creating a new user examples for the Brocade Flow Optimizer (BFO) server.

Creating a new user examples for BFO POST operation for login include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>:8089/collector/users
```

Example of Headers Sent to BFO

The following is an example of creating a new user request headers sent.

```
JSON

        Request header

        {
        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        }

        Request body

        {
        "name": "GuestTwo",
        "password": "guest_two"
        "access_privilege": 1
        }

XML

        Request header

        Authorization - Guest_1426270596791
        Accept - Application/xml

        Request payload

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <user>
        <name>GuestTwo</name>
        <password>guest_two</password>
        <access_privilege>0</access_privilege>
        </user>
```

TABLE 5 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS, on successful code the list of users are returned
401 Unauthorized	FAILED, reason included in HTTPS error response

On successful completion, new user will created in the BFO.

Edit a User

Updating a single user in the Brocade Flow Optimizer (BFO). Only a user with Admin privileges (access_privilege=1) is allowed to update an existing user. The user's password and access_privilege can be updated.

Resource URL
<Base URI>/collector/users/<user name>

All REST operations will be provided at the base context path in the URI field of your REST client tool.

Before editing a user in the BFO, criteria for editing a user are as follows:

- Only a user with Admin privileges (access_privilege=1) is allowed to edit a user.
- Valid username name and password are optional as input.

NOTE
The root user by the name Administrator cannot be updated.

To edit the existing user in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/users/<user name>
2. Define the updated user parameters.
3. Set the HTTPS request method to PUT.

Editing User Parameters

This topic covers parameters for editing a user for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Editing User Examples

This topic covers editing a user examples for the Brocade Flow Optimizer (BFO) server.

Editing a user examples for BFO PUT operation for login include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/users/<username>
```

Example of Headers Sent to BFO

The following is an example of edit user request headers sent.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/json

        Request payload

        {
        "password": "guest_two_updated",
        "access_privilege": 0
        }

        Request Body

        {
        "password": "guest_two_updated",
        "access_privilege": 0
        }

XML

        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/xml

        Request payload

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <user>
        <password>guest_two_updated</password>
        <access_privilege>0</access_privilege>
        </user>

        Request body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <user>
        <password>guest_two_updated</password>
        <access_privilege>0</access_privilege>
        </user>
```

TABLE 6 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS, on successful code the list of users are returned
401 Unauthorized	FAILED, reason included in HTTPS error response

On successful completion, new user is edited in the BFO.

Delete a User

This topic covers the process of deleting a single user. Only a user with Admin privileges (access_privilege=1) is allowed to delete an existing user.

Resource URL
<Base URI>/collector/users/<user name>

All REST operations will be provided at the base context path in the URI field of your REST client tool.

NOTE
User by the name Administrator is the root user and cannot be deleted.

To delete the existing user in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/users/<username>
2. Define the deleted user parameters.
3. Set the HTTPS request method to DELETE.

Deleting User Parameters

This topic covers parameters for deleting a user of the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE
The DELETE operation is supported.

Deleting User Examples

This topic covers examples for deleting a user of the Brocade Flow Optimizer (BFO) server.

Deleting a user examples for BFO PUT operation for login include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/users/<username>
```

Example of Headers Sent to BFO

The following is an example of the delete user request headers sent.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json

XML

       Request header

       Authorization - Guest_1426270596791
       Accept - Application/xml
```

TABLE 7 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS, on successful code the list of users are returned
401 Unauthorized	FAILED, reason included in HTTPS error response

On successful completion, user is deleted in the BFO.

Dashboard

Dashboard Overview

Dashboard is a monitoring section of the Brocade Flow Optimizer (BFO). The Dashboard REST API URIs retrieve the list of Large Flows that BFO has identified, based on user configured profiles.

Dashboard REST URIs provide some of the vital statistics listed below.

- Total traffic BFO is processing or receiving.
- Total traffic (Mbps) dropped.
- Total traffic (Mbps) passed. This include the traffic BFO has not marked as large flow.
- List of Large Flows identified by Default and Custom Profiles.

Dashboard Overall Traffic Graph

This topic covers the process of accessing the overall traffic graph. This API returns the Overall traffic utilization (Dropped + Passed) for the last one (1) minute. Each API call returns the last 1 minute of data (Passed Traffic and Dropped Traffic).

Resource URL
`<Base URI>/collector/utilization/overallTraffic`

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To access the overall traffic graph in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/utilization/overallTraffic
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to GET.

Dashboard Overall Traffic Graph Parameters

This topic covers parameters for the overall traffic graph for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE
The GET operation is supported.

Overall Traffic Graph Examples

This topic covers overall traffic graph examples for the Brocade Flow Optimizer (BFO) server.

Overall traffic graph examples for BFO GET operation for overall traffic include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>:8089/collector/utilization/overallTraffic
```

Example of Headers Sent to BFO

The following is an example of overall traffic graph request headers sent.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
          "utilization":[
            {
              "passed":1383,
              "blocked":0
            },
            {
              "passed":1376,
              "blocked":0
            },
            {
              "passed":1369,
              "blocked":0
            },
            {
              "passed":1372,
              "blocked":0
            },
            {
              "passed":1375,
              "blocked":0
            }
          [
        }

XML

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
```

TABLE 8 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

The utilization data of Blocked Traffic and Passed Traffic will be returned in the response. The data will cover the last thirty (30) minutes, with one (1) minute granularity on initial load. Every 1 minute, the utilization for the current minute will be updated.

Dashboard Historical Drill Down of Overall Traffic Graph

This topic covers the process of returning the historical data of the overall traffic graph. The Brocade Flow Optimizer (BFO) supports historical data retained for the past thirty (30) minutes, one (1) hour, one (1) day, one (1) week, or thirty (30) days.

Resource URL
`<Base URI>/collector/utilization/detailOverallTraffic`

The dashboard historical drill down of overall traffic graph API returns the top five (5) large flows (by BW utilized) for the time range selected. The user can retrieve the flows for dropped and passed traffic.

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To retrieve the historical overall traffic data in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/utilization/detailOverallTraffic
2. Define the return list of parameters.
3. Set the HTTPS request method to POST.

Dashboard Historical Drill Down of Overall Traffic Graph Parameters

This topic covers parameters for the historical drill down of overall traffic graph for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Dashboard Historical Drill Down of Overall Traffic Graph Examples

This topic covers historical drill down of overall traffic graph examples for the Brocade Flow Optimizer (BFO) server.

Historical drill down of overall traffic graph examples for BFO POST operation for overall traffic include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://<BFO Server IP>:8089/collector/utilization/detailOverallTraffic
```

Example of Headers Sent to BFO

The following is an example of dashboard historical drill down of overall traffic graph request headers sent.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Content-Type:application/json

        Request Body

        {
          "startTime": 0,
          "endTime": 0,
          "duration": 1800000,
          "mitigationType": "NONE"
        }

        Response Body

        {
          "startTime":1440161280000,
          "endTime":1440163079000,
          "granularity_ms":60000,
          "mitigationType":"NONE",
          "topFlows":[
            {
              "id":296657552,
              "rank":1,
              "percentage":100,
              "totalUtil": 542,
              "utilization":[
                1371,
                1379,
                1395,
                1368,
                1363,
                1343,
                1395,
                1368,
                1373,
                1347,
                1382,
                1370,
                1359,
                1387,
                1368,
                1375,
                1372,
                1376,
                1364,
                1376,
                1351,
                1395,
                1382,
                1370,
                1373,
                1386,
                1368,
                1375,
                1354,
                0
              ],
              "srcIp":"10.10.10.4",
              "destIp":"30.30.30.4",
              "srcMac":"00:24:38:7c:7e:00",
              "destMac":"00:24:38:ae:cb:00",
              "srcPort":53,
              "destPort":60,
```

```
            "inVlan":20
          },
          {
          }
        ],
        "otherFlows":[
        ]
      }
```

XML

```
      Request header

      Authorization - Administrator_1426270596791
      Content-Type - Application/xml

      Request Body

      <?xml version="1.0" encoding="UTF-8"?>
      <detailOverallTraffic>
        <duration>1800000</duration>
        <endTime>0</endTime>
        <mitigationType>NONE</mitigationType>
        <startTime>0</startTime>
      </detailOverallTraffic>
```

TABLE 9 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

The utilization data for the specified duration will be retrieved in the response.

Dashboard Current Large Flows

This topic covers the process of returning the list of Large flows identified for default profiles and custom profiles in the Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/utilization/profileTraffic

The dashboard current large flows will return the list of the flows sorted by BW utilization.

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To retrieve the list of the flows sorted by BW utilization in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/utilization/profileTraffic
2. Define the return list of parameters.
3. Set the HTTPS request method to GET.

Dashboard Current Large Flows Parameters

This topic covers parameters for the current large flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE

The GET operation is supported.

Dashboard Current Large Flows Examples

This topic covers examples of current large flows for the Brocade Flow Optimizer (BFO) server.

Examples of current large flows for BFO GET operation for overall traffic include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/utilization/profileTraffic
```

Example of Headers Sent to BFO

The following is an example of dashboard current large flows request headers sent.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
          "profileUtilization":[
            {
              "id":1,
              "name":"NTP_Reflection",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":2,
              "name":"DNS_Reflection",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":4,
              "name":"ICMP_Flood",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":5,
              "name":"CharGen",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":6,
              "name":"Quote_Of_The_Day",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":7,
              "name":"Simple_Service_Discovery_Protocol",
              "type":0,
              "action":"NONE",
              "threshold":null,
              "utilization":null,
              "attacks":null
            },
            {
              "id":3,
              "name":"UDP_Flood",
```

```
        "type":0,
        "action":"NONE",
        "threshold":null,
        "utilization":null,
        "attacks":null
      },
      {
        "id":101,
        "name":"Custom Profile",
        "type":1,
        "action":"METER",
        "threshold":null,
        "utilization":null,
        "attacks":null
      }
    ]
  }
```


TABLE 10 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Flows

Flows Overview

Flows is a URI to fetch learned and programmed flows with utilization as Mbps.

Flows REST URI provides some of the features listed below.

- Learned flows with utilization as Mbps.
- Programmed flows with utilization as Mbps.

Learned and Programmed Flows

This topic covers the process of learned and programmed flows. This API returns learned and programmed flows with utilization as Mbps. The response returns active flows for last 15 seconds.

Resource URL
`<Base URI>/collector/flows/largeFlows`

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To access learned and programmed flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/flows/largeFlows
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Learned and Programmed Flows Parameters

This topic covers parameters for learned and programmed flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Learned and Programmed Flows Examples

This topic covers the fetch request for learned and programmed flows for the Brocade Flow Optimizer (BFO) server.

Learned and programmed flows fetch request examples for BFO POST operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/flows/largeFlows
```

Example of Headers Sent to BFO

The following is an example of learned and programmed flows fetch request headers sent.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Request Body

        {
          "page_size": "1",
          "page_range": "1-2",
          "filters": {
            "network_attribute": [
              {
                "name": "SRC_MAC",
                "value": "00:24:38:80:8e:00-V4"
              },
              {
                "name": "V4_DEST_ADDR",
                "value": "10.1.1.4"
              },
              {
                "name": "IN_VLAN",
                "value": "10"
              }
              ],
              "profile_name": "Custom Profile", // to filter by profile name
                "profile_type": "0", // to filter by profile type 0- Default, 1- Custom Profile
              "action": "None" // to filter by action
            }
        }

        Response Body

        {
            "page_size": 1,
            "total_pages": 250,
            "timestamp": 1440505824364,
            "page": [
              {
                    "number": 1,
                    "large_flow": [
                      {
                            "id": "FlowKey_1",
                            "utilization": 25000,
                            "identifiedOn": 1440505824,
                            "profileId": 1,
                            "profileName": "Custom Profile_0",
                            "profileType": 1,
                            "profileAction": "None",
                            "l2_network_attributes": [
                              {
                                    "name": "SRC_MAC",
                                    "value": "00:24:38:80:8e:00-V4",
                                    "label": "Source Mac"
                              },
                              {
                                    "name": "DEST_MAC",
                                    "value": "00:0a:95:9d:68:16-V4",
                                    "label": "Destination Mac"
                              }
                          ],
                            "l3_network_attributes": [
                              {
                                    "name": "V4_DEST_ADDR",
                                    "value": "10.10.1.1",
                                    "label": "Destination IP"
```

```
                        }
                    ],
                      "l4_network_attributes": [
                        {
                              "name": "UDP_SRC_PORT",
                              "value": "123",
                              "label": "UDP Source Port"
                        }
                    ]
                }
            ]
        },
        {
              "number": 2,
              "large_flow": [
                {
                      "id": "FlowKey_2",
                      "utilization": 12500,
                      "identifiedOn": 1440505824,
                      "profileId": 2,
                      "profileName": "Custom Profile_1",
                      "profileType": 1,
                      "profileAction": "None",
                      "l2_network_attributes": [
                        {
                              "name": "SRC_MAC",
                              "value": "00:24:38:80:8e:00-V4",
                              "label": "Source Mac"
                        },
                        {
                              "name": "DEST_MAC",
                              "value": "00:0a:95:9d:68:16-V4",
                              "label": "Destination Mac"
                        }
                    ],
                      "l3_network_attributes": [
                        {
                              "name": "V4_DEST_ADDR",
                              "value": "10.10.1.1",
                              "label": "Destination IP"
                        }
                    ],
                      "l4_network_attributes": [
                        {
                              "name": "UDP_SRC_PORT",
                              "value": "123",
                              "label": "UDP Source Port"
                        }
                    ]
                }
            ]
        }
    ],
      "filters": {
          "network_attribute": [
            {
                  "name": "SRC_MAC",
                  "value": "00:24:38:80:8e:00-V4"
            },
            {
                  "name": "V4_DEST_ADDR",
                  "value": "10.1.1.4"
            },
            {
                  "name": "IN_VLAN",
                  "value": "10"
            }
        ],
          "profile_name": "Custom Profile",
          "action": "None"
    }
}
```

TABLE 11 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Learned Flows

This topic covers how to retrieve a list of all active flows. By default, the details (utilization in Mbps) is fetched for the last 30 minutes. The Filter criteria is set based on learned flows entries.

Resource URL
`<Base URI>/collector/flows/allFlows`

The learned flows REST URI provides some of the features listed below.

- Bandwidth utilization.
- Network attributes.

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To access learned and programmed flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/flows/allFlows
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Learned Flows Parameters

This topic covers parameters for learned flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Learned Flows Examples

This topic covers learned flows fetch request for the Brocade Flow Optimizer (BFO) server.

Learned flows fetch request examples for BFO POST operation for flows include:

- BFO URI
- BFO request headers sent

Request data criteria includes the following.

- Pagination Support is available. The response data is populated based on page size and page range.
- Filters the data based on network attribute. See the Example of BFO URI.
- Filtering based on Bandwidth utilization. A value of zero is optional as it will be excluded.

Equals 500 Mbps:

```
"minUtil" : 500,
“maxUtil” : 500
```

Greater than 300 Mbps:

```
“minUtil” : 300 ,
“maxUtil” : “0”
```

Less than 250 Mbps:

```
“minUtil” : 0 ,
“maxUtil” : 250
```

- Retrieve all values or based on filter criteria.

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/flows/allFlows
```

Example of Headers Sent to BFO

The following is an example of learned flows fetch request headers sent.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Request Body

        {
            "page_size": "1",
            "page_range": "1-2",
            "filters": {
                "network_attribute": [
                        {
                                        "name": "FLOW_NAME",
                                        "value": "SJ_DC_FLOW"
                                    },
                                    {
                                        "name": "SRC_MAC",
                                        "value": "00:00:82:8a:e6:2f"
                                    },
                                    {
                                        "name": "DEST_MAC",
                                        "value": "00:24:38:93:f7:00"
                                    },
                                    {
                                        "name": "V4_SRC_ADDR",
                                        "value": "218.218.218.201"
                                    },
                        {
                                        "name": "V4_DEST_ADDR",
                                        "value": "219.219.219.201"
                                    },
                                    {
                                        "name": " L3_PROTOCOL",
                                        "value": "UDP"
                                    },
                                    {
                                        "name": "TCP_SRC_PORT",
                                        "value": "*"
                                    },
                        {
                                        "name": "UDP_SRC_PORT",
                                        "value": "*"
                                    },
                                    {
                                        "name": "TCP_DEST_PORT",
                                        "value": "123"
                                    },
                                    {
                                        "name": "UDP_DEST_PORT",
                                        "value": "63"
                                    },
                        {
                                           "name": "IN_VLAN",
                                        "value": "218"
                                    },
                        {
                                           "name": "VLAN_PRIORITY",
                                        "value": "10"
                                    }
                                ],
                    ],
                            "duration": "1800",
                         "minUtil" : 50,
                         "maxUtil" : -1
                }
```

```
}
```

Response

The Traffic Flow details along with utilization will be sent in the response.

Response Body

```
{
    "page_size": 20,
    "total_pages": 3,
    "timestamp": 1443509790954,
    "page": [
      {
         "number": 1,
         "traffic_flow": [
            {
               "id": 60345244,
                  “flow_name”:SF_DC_WebServer_Flow,
               "utilization": 46.1,
               "source_mac": "00:24:38:93:f7:00",
               "destination_mac": "00:00:f3:ae:4f:6a",
               "in_vlan": 30,
               "in_priority": 0,
               "ip_tos": 0,
               "source_ip": "10.10.10.2",
               "destination_ip": "25.25.25.3",
               "ip_protocol": "UDP",
               "source_port": 63,
               "destination_port": 63,
               "action": "NONE",
               "mpls_label": 16,
               "mpls_traffic_class": 0,
               "mpls_stack": 1,
               "mpls_ttl": 62,
               "vni": null,
               "inner_source_mac": "",
               "inner_destination_mac": "",
               "inner_ip_fragment": 0,
               "inner_source_ip": "",
               "inner_destination_ip": "",
               "inner_ip_protocol": "",
               "inner_is_ipv6_flow": false,
               "esp_sec_param_indx”:”2",
               "esp_sec_number”:”101",
               "ah_next_header”:"",
               "ah_length":””,
               "ah_sec_param_indx":””,
               "ah_sec_number":””,
               "ah_auth_data":””
             },
             {
               "id": 70765469,
               "utilization": 46.03,
               "source_mac": "00:24:38:93:f7:00",
               "destination_mac": "00:00:f3:ae:4f:6a",
               "in_vlan": 30,
               "in_priority": 0,
               "ip_tos": 0,
               "source_ip": "10.10.10.2",
               "destination_ip": "25.25.25.4",
               "ip_protocol": "UDP",
               "source_port": 63,
               "destination_port": 63,
               "action": "NONE",
               "mpls_label": 16,
               "mpls_traffic_class": 0,
               "mpls_stack": 1,
               "mpls_ttl": 62,
               "vni": null,
               "inner_source_mac": "",
```

```
                "inner_destination_mac": "",
                "inner_ip_fragment": 0,
                "inner_source_ip": "",
                "inner_destination_ip": "",
                "inner_ip_protocol": "",
                "inner_is_ipv6_flow": false
              },……
            ……
            ……
        {
           "id": 3408120787,
           "utilization": 43.6,
           "source_mac": "00:24:38:93:f7:00",
           "destination_mac": "00:00:f3:ae:4f:6a",
           "in_vlan": 30,
           "in_priority": 0,
           "ip_tos": 0,
           "source_ip": "10.10.10.2",
           "destination_ip": "25.25.25.27",
           "ip_protocol": "UDP",
           "source_port": 63,
           "destination_port": 63,
           "action": "NONE",
           "mpls_label": 16,
           "mpls_traffic_class": 0,
           "mpls_stack": 1,
           "mpls_ttl": 62,
           "vni": null,
           "inner_source_mac": "",
           "inner_destination_mac": "",
           "inner_ip_fragment": 0,
           "inner_source_ip": "",
           "inner_destination_ip": "",
           "inner_ip_protocol": "",
           "inner_is_ipv6_flow": false,
               "esp_sec_param_indx”:”2",
               "esp_sec_number”:”101",
               "ah_next_header”:”",
               "ah_length":””,
               "ah_sec_param_indx":””,
               "ah_sec_number":””,
               "ah_auth_data":””
           }
         ]
       }
     ],
     "filters": {
        "network_attribute": [
           {
             "name": "SRC_MAC",
             "value": "00:24:38:93:f7:00"
           },
           {
            "name": "L3_PROTOCOL",
            "value": "ALL"
          }
     ],
     "action": null,
     "duration": null,
     "minUtil": null,
     "maxUtil": null
   }
}
```


TABLE 12 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response

TABLE 12 Status Code (continued)

HTTPS Status Code	Description
500 Internal Server Error	FAILED, reason included in HTTPS error response

Learned Flows: Updating Flow Name

This topic covers how to update flow names.

Resource URL
`<Base URI>/collector/flows/allflows/<-Flow ID>?flowname=<NAME>`

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To update flow names in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/flows/allflows/<-Flow ID>?flowname=<NAME>
2. Set the HTTPS request method to PUT.

Learned Flows: Updating Flow Name Parameters

This topic covers parameters for updating flow names for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	TokenID
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Learned Flows: Updating Flow Name Examples

This topic covers updating flow name fetch request for the Brocade Flow Optimizer (BFO) server.

Updating flow name fetch request examples for BFO PUT operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/flows/allflows/<-Flow ID>?flowname=<NAME>
```

Example of Headers Sent to BFO

The following is an example of updating flow name request headers sent.

```
JSON

       Request header

       Authorization - TokenID
       Content-Type - Application/json
       Content-Type: Application/json
```

TABLE 13 Status Code

HTTPS Status Code	Description
204 No Content	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Bulk Deletion of Custom Flows

This topic covers how to bulk delete custom flows.

Resource URL
`<Base URI>/collector/customflows`

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To bulk delete custom flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/customflows
2. Set the HTTPS request method to DELETE.

Bulk Deletion of Custom Flows Parameters

This topic covers parameters for bulk deletion of custom flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	TokenID
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE

The DELETE operation is supported.

Bulk Deletion of Custom Flows Examples

This topic covers bulk deletion of custom flows fetch request for the Brocade Flow Optimizer (BFO) server.

Bulk deletion of custom flows fetch request examples for BFO DELETE operation for flows include:

- BFO URI

- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/customflows
```

Example of Headers Sent to BFO

The following is an example of bulk deletion of custom flows fetch request headers sent.

```
JSON

       Request header

       Authorization - TokenID
       Content-Type - Application/json
       Content-Type: Application/json

       Request Payload

       {object_id: ["BFO_UD_Pri_1461924921972", "BFO_UD_Pri_1461857084126"]}
```

TABLE 14 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

User Defined Flows

This topic covers the process of user defined flows, which is a GET operation for creating a `CustomFlows` search, returning a list of `CustomFlows`.

Resource URL
`<Base URI>/collector/customflows/search?startpage=<value>&noofpages=<value>&pagesize=<value>`

The `CustomFlows` object is a paginated resource containing a list of CustomFlow instances. Each `CustomFlows` object represents a page of the resource. The API provides the user with three query parameters `startpage`, `noofpages`, and `pagesize`.

- The `startpage` parameter indicates the starting number of the first page of data.
- The `noofpages` parameter indicates the number of pages to be returned which equates to the number of `CustomFlows` object instances.
- The `pagesize` parameter indicates the number of elements per page, which equates to the number of CustomFlow instances within the `CustomFlows` object instance.

If the startpage query parameter is null, then all `CustomFlows` object instances are returned. It startpage is not null and > 0 but `noofpages` and `pagesize` are null, then number of pages and pagesize are defaulted to 5 and 50 respectively.

All REST operations will be provided at the base context path in the URI field of your REST client tool.

To access learned and programmed flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/customflows/search?startpage=<value>&noofpages=<value>&pagesize=<value>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to GET.

User Defined Flows Parameters

This topic covers parameters for user defined flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

Query Parameter	Parameter Value
startpage	Start page.
noofpages	Number of pages to be returned.
pagesize	Number of records per page.

NOTE
The GET operation is supported.

User Defined Flows Examples

This topic covers the fetch request for user defined flows for the Brocade Flow Optimizer (BFO) server.

User defined flows fetch request examples for BFO GET operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/flows/largeFlows
```

Example of Headers Sent to BFO

The following is an example for retrieving user defined flows.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Payload

        {
           "flow_name":"APS_SPECIAL",
           "highest_priority":false,
           "flow_priority":1001,
           "source_mac":"00:24:38:80:e6:03",
           "ip_protocol":"UDP",
           "action":{
              "action_id":"REDIRECT",
              "action_parameters":{
                 "redirect":{
                    "redirect_action_parameter":[
                       {
                          "device_ip":"10.25.225.228",
                          "egress_ports":[
                             65
                          ],
                          "set_field_dest_mac":"",
                          "set_field_push_vlanid":""
                       }
                    ]
                 }
              }
           }
        }

        Response Payload

        [
           {
              "custom_flow":[
                 {
                    "flow_name":"APS_SPECIAL",
                    "flow_id":"BFO_UD_Pri_36001",
                    "flow_priority":36001,
                    "source_mac":"00:24:38:80:e6:00",
                    "ip_protocol":"UDP",
                    "action":{
                       "action_id":"REDIRECT",
                       "action_parameters":{
                          "redirect":{
                             "redirect_action_parameter":[
                                {
                                   "device_ip":"10.25.225.228",
                                   "egress_ports":[
                                      65
                                   ],
                                   "set_field_push_vlanid":0
                                }
                             ]
                          },
                          "mirror":null,
                          "meter":null
                       }
                    }
                 }
              ],
              "timestamp":1443792133626,
              "page_number":1,
```

```
            "page_size":1,
            "total_pages":1
        }
    ]
```

TABLE 15 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Create User Defined Flow

This topic covers the process of creating a custom open flow, by providing the match criteria and mitigation action.

NOTE

To create a new Remotely Triggered Black Hole (RTBH) user defined flow, see the topic *Create RTBH User Defined Flow*.

Resource URL
`<Base URI>/collector/customFlows`

The user will be able to provide the flow priority above or below the learned and programmed flows.

To access custom open flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/customFlows
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Create User Defined Flow Parameters

This topic covers parameters for creating user defined flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Content-Type	JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE

The POST operation is supported.

Create User Defined Flow Examples

This topic covers the fetch request for creating user defined flows for the Brocade Flow Optimizer (BFO) server.

The create user defined flows fetch request examples for BFO POST operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/customFlows
```

Example of Headers Sent to BFO

The following is an example of creating user defined flows.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/json
        Accept - Application/xml

        Request Payload

        {
           "flow_name":"APS_SPECIAL",
           "highest_priority":false,
           "flow_priority":1001,
           "source_mac":"00:24:38:80:e6:03",
           "ip_protocol":"UDP",
           "action":{
              "action_id":"REDIRECT",
              "action_parameters":{
                 "redirect":{
                    "redirect_action_parameter":[
                       {
                          "device_ip":"10.25.225.228",
                          "egress_ports":[
                             65
                          ],
                          "set_field_dest_mac":"",
                          "set_field_push_vlanid":""
                       }
                    ]
                 }
              }
           }
        }
```

TABLE 16 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Delete User Defined Flow

This topic covers the process of deleting a user defined flow, by providing the Flow ID.

Resource URL
<Base URI>/collector/customFlows/<Flow ID>

To access custom open flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/customFlows/<Flow ID>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to DELETE.

Delete User Defined Flow Parameters

This topic covers parameters for deleting user defined flows for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Content-Type	JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE

The DELETE operation is supported.

Delete User Defined Flow Examples

This topic covers the fetch request for deleting user defined flows for the Brocade Flow Optimizer (BFO) server.

The delete user defined flows fetch request examples for BFO DELETE operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/customFlows/<Flow ID>
```

Example of Headers Sent to BFO

The following is an example of deleting user defined flows.

```
JSON

       Request headers

       Authorization - Administrator_1426270596791
       Content-Type - Application/json
       Content-Type: Application/json
```

TABLE 17 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Profiles

Profiles Overview

Management of profiles involve URIs to add, edit, and delete profiles; change the priority of profiles; and enable or disable a profile.

Profiles REST URIs provides some of the features listed below.

- Add a new profile.
- Edit a profile.
- Enable a profile.
- Disable a profile.
- Change the priority of a profile.
- Delete a profile.

Create New Custom Profile

This topic covers the process of creating a new custom profile in the Brocade Flow Optimizer (BFO).

NOTE

To create a new Remotely Triggered Black Hole (RTBH) custom profile, see the topic *Create New RTBH Custom Profile*.

Resource URL
`<Base URI>/collector/profile`

To create a new custom profile in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profile
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Create New Custom Profile Parameters

This topic covers parameters for creating new custom profiles for the Brocade Flow Optimizer (BFO).

NOTE

To create new Remotely Triggered Black Hole (RTBH) custom profile parameters, see the topic *Create New RTBH Custom Profile Parameters*.

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE

The POST operation is supported.

Create New Custom Profile Examples

This topic covers the fetch request for creating new custom profiles for the Brocade Flow Optimizer (BFO).

NOTE

To see examples for creating a new Remotely Triggered Black Hole (RTBH) custom profile, see the topic *Create New RTBH Custom Profile Examples*.

Create new custom profile fetch request examples for BFO POST operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profile
```

Example of Headers Sent to BFO

The following is an example of creating a new custom profile.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/json
        Content-Type:application/json

        Request Body

        Support mitigation action for the profile includes
        NONE / DROP / REDIRECT / METER / REAMRK / MIRROR.

        Profile with NONE / DROP / REDIRECT / METER actions

        {
           "name":"Custom Profile_FlowTab",
           "type":1,
           "enabled":1,
           "priority":14,
           "last_modified_by":"Administrator",
           "observation_interval_ms":10000,
           "threshold":100,
           "network_attributes":{
              "network_attribute":[
                 {
                    "name":"V4_SRC_ADDR",
                    "value":"10.10.0.2/32"
                 },
                 {
                    "name":"SRC_MAC",
                    "value":"aa:eb:3c:f3:ee:a2/ff:ff:ff:ff:ff:ff"
                 },
                 {
                    "name":"V4_DEST_ADDR",
                    "value":"60.60.60.4/32"
                 }
              ]
           },
           "mitigation_actions":{
              "mitigation_action":[
        {
                    "action":"NONE",
                 },
                 {
                    "action":"DROP",
                    "redirect_nodes":{
                       "redirect_node":[
                          {
                             "id":"10.24.63.240",
                             "ports":"3",
                             "dest_mac":"",
                             "vlan_id":"",
                             "ingress_port":3,
                             "portsJson":[
                                {
                                   "id":"openflow:10195225427312640:3",
                                   "port-number":3,
                                   "name":"eth1/3"
                                }
                             ]
                          }
                       ]
                    }
                 },
                 {
                    "action":"REDIRECT",
                    "redirect_nodes":{
```

```
                        "redirect_node":[
                           {
                              "id":"10.24.63.240",
                              "ports":"4,50",
                              "dest_mac":"",
                              "vlan_action":"IGNORE",
                              "vlan_id":"",
                              "ingress_port":3,
                              "portsJson":[
                                 {
                                    "id":"openflow:10195225427312640:4",
                                    "port-number":4,
                                    "name":"eth1/4"
                                 },
                                 {
                                    "id":"openflow:10195225427312640:50",
                                    "port-number":50,
                                    "name":"eth2/2"
                                 }
                              ]
                           }
                        ]
                     }
                  },
                  {
                     "action":"METER",
                     "dscp_band_enabled":"false",
                     "rate_limit":2000,
                     "redirect_nodes":{
                        "redirect_node":[
                           {
                              "id":"10.24.63.240",
                              "vlan_id":10,
                              "ingress_port":3,
                              "portsJson":[
                                 {
                                    "id":"openflow:10195225427312640:3",
                                    "port-number":3,
                                    "name":"eth1/3"
                                 }
                              ]
                           }
                        ]
                     }
                  }
               ]
            }
         }
```

XML

Request Header

```
Authorization - Administrator_1426270596791
Content-Type - Application/xml
Accept - Application/xml
```

Request Body

```
<?xml version="1.0" encoding="UTF-8" ?>
<profile>
      <name>Custom Profile1</name>
      <type>1</type>
      <enabled>1</enabled>
      <priority>8</priority>
      <last_modified_by>Administrator</last_modified_by>
      <observation_interval_ms>15000</observation_interval_ms>
      <threshold>5000</threshold>
      <action>NONE</action>
      <network_attributes>
            <network_attribute>
                  <name>IN_VLAN</name>
```

```
                <value>20</value>
            </network_attribute>
        </network_attributes>
    </profile>
```

TABLE 18 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, a new custom profile is created in the BFO.

Edit Custom Profile

This topic covers the process of editing a custom profile in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profile?profileName=<Profile Name>`

To edit custom profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profile?profileName=<Profile Name>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to PUT.

Edit Profile Parameters

This topic covers parameters for editing custom profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Edit Custom Profile Examples

This topic covers the fetch request for editing custom profiles for the Brocade Flow Optimizer (BFO).

Edit custom profile fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profile?profileName=<Profile Name>
```

Example of Headers Sent to BFO

The following is an example of editing a custom profile.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/json
        Content-Type: application/json

        Request Body

        Support mitigation action for the profile include
        NONE / DROP / REDIRECT / METER / REAMRK / MIRROR.

        {
           "id":101,
           "name":"Custom Profile_FlowTab",
           "type":1,
           "enabled":1,
           "priority":14,
           "last_modified_by":"Administrator",
           "observation_interval_ms":10000,
           "threshold":100,
           "network_attributes":{
              "network_attribute":[
                 {
                    "name":"V4_SRC_ADDR",
                    "value":"10.10.0.2/32"
                 },
                 {
                    "name":"SRC_MAC",
                    "value":"aa:eb:3c:f3:ee:a2/ff:ff:ff:ff:ff:ff"
                 },
                 {
                    "name":"V4_DEST_ADDR",
                    "value":"60.60.60.4/32"
                 }
              ]
           },
           "mitigation_actions":{
              "mitigation_action":[
        {
                    "action":"NONE",
                 },
                 {
                    "action":"DROP",
                    "redirect_nodes":{
                       "redirect_node":[
                          {
                             "id":"10.24.63.240",
                             "ports":"3",
                             "dest_mac":"",
                             "vlan_id":"",
                             "ingress_port":3,
                             "portsJson":[
                                {
                                   "id":"openflow:10195225427312640:3",
                                   "port-number":3,
                                   "name":"eth1/3"
                                }
                             ]
                          }
                       ]
                    }
                 },
                 {
                    "action":"REDIRECT",
                    "redirect_nodes":{
                       "redirect_node":[
```

```
                        {
                           "id":"10.24.63.240",
                           "ports":"4,50",
                           "dest_mac":"",
                           "vlan_action":"IGNORE",
                           "vlan_id":"",
                           "ingress_port":3,
                           "portsJson":[
                              {
                                 "id":"openflow:10195225427312640:4",
                                 "port-number":4,
                                 "name":"eth1/4"
                              },
                              {
                                 "id":"openflow:10195225427312640:50",
                                 "port-number":50,
                                 "name":"eth2/2"
                              }
                           ]
                        }
                     ]
                  }
               },
               {
                  "action":"METER",
                              "dscp_band_enabled":"false",
                              "rate_limit":2000,
                  "redirect_nodes":{
                     "redirect_node":[
                        {
                           "id":"10.24.63.240",
                           "vlan_id":10,
                           "ingress_port":3,
                           "portsJson":[
                              {
                                 "id":"openflow:10195225427312640:3",
                                 "port-number":3,
                                 "name":"eth1/3"
                              }
                           ]
                        }
                     ]
                  }
               }
            ]
         }
      }

      NOTE

      For action [MIRROR] the payload is same as in add operation.

XML

      Request Header

      Authorization - Administrator_1426270596791
      Content-Type - Application/xml
      Accept - Application/xml
```

TABLE 19 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the existing custom profile is updated in the BFO.

Delete Custom Profile

This topic covers the process of deleting a custom profile in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profile?profileName=<Profile Name>`

To delete custom profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profile?profileName=<Profile Name>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to DELETE.

Delete Custom Profile Parameters

This topic covers parameters for deleting custom profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE
The DELETE operation is supported.

Delete Custom Profile Examples

This topic covers the fetch request for deleting custom profiles for the Brocade Flow Optimizer (BFO).

Delete custom profile fetch request examples for BFO DELETE operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profile?profileName=<Profile Name>
```

Example of Headers Sent to BFO

The following is an example of deleting a custom profile.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Accept - Application/json
       Content-Type: application/json

XML

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
       Accept - Application/xml
```

TABLE 20 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful delete, existing custom profile is deleted in the BFO.

Get List of Profiles

This topic covers how to get a list of profiles in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/profiles

To get a list of profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Get List of Profiles Parameters

This topic covers parameters for getting a list of profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE
The GET operation is supported.

Get List of Profiles Examples

This topic covers the fetch request for getting a list of profiles for the Brocade Flow Optimizer (BFO).

Get list of profiles fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example for getting a list of profiles.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Accept - Application/json

        Response Body

        {
            "profile": [
                {
                    "id": 1,
                    "name": "NTP_Reflection",
                    "description": "This profile is used to monitor and detect
        unsolicited responses fromNTP servers.The NTP servers responds to get
        monlist requests from a source that typically cannot be traced. The
        traffic is amplified and results in a huge amount of monlist query
        responses that consume the target's network resources and disrupt the
        network.",
                    "observation_interval_ms": 30000,
                    "threshold": 100000,
                    "reduced_threshold": 90000,
                    "action": "NONE",
                    "last_modified_time": 1439360462491,
                    "last_modified_by": "Administrator",
                    "enabled": 1,
                    "type": 0,
                    "priority": 3,
                    "dscp_band_enabled": false,
                    "flow_timeout": 0,
                    "network_attributes": {
                        "network_attribute": [
                            {
                         "name": "L3_PROTOCOL",
                         "value": "UDP"
                      },
                      {
                         "name": "UDP_SRC_PORT",
                         "value": "123"
                      }
                   ]
                }
             },
             {
                "id": 2,
                "name": "DNS_Reflection",
                "description": "This profile is used to monitor and detect
        unsolicited DNS query responsescoming from third party systems
        (usually name servers). The traffic  is amplified and results
        in a huge amountof DNS query responses that consume the target's
        network resources and disrupt the network.",
                "observation_interval_ms": 15000,
                "threshold": 10000,
                "reduced_threshold": 9000,
                "action": "NONE",
                "last_modified_time": 1439360370207,
                "last_modified_by": "Administrator",
                "enabled": 1,
                "type": 0,
                "priority": 4,
                "dscp_band_enabled": false,
                "flow_timeout": 0,
                "network_attributes": {
                    "network_attribute": [
                        {
                            "name": "L3_PROTOCOL",
                            "value": "UDP_TCP"
```

```
                },
                {
                    "name": "UDP_SRC_PORT",
                    "value": "53"
                },
                {
                    "name": "TCP_SRC_PORT",
                    "value": "53"
                }
            ]
        }
    }
```

XML

Request Header

Authorization - Guest_1426270596791
Accept - Application/xml

Request Body

```
<?xml version="1.0" encoding="UTF-8" ?>
         <profiles>
               <profile>
                   <id>1</id>
                   <name>NTP_Reflection</name>
                   <description>profile description.</description>
                   <observation_interval_ms>30000</observation_interval_ms>
                   <threshold>100000</threshold>
                   <reduced_threshold>90000.0</reduced_threshold>
                   <action>NONE</action>
                   <last_modified_time>1439360462491</last_modified_time>
                   <last_modified_by>Administrator</last_modified_by>
                   <enabled>1</enabled>
                   <type>0</type>
                   <priority>3</priority>
                   <dscp_band_enabled>false</dscp_band_enabled>
                   <flow_timeout>0</flow_timeout>
                   <network_attributes>
                         <network_attribute>
                               <name>L3_PROTOCOL</name>
                               <value>UDP</value>
                         </network_attribute>
                         <network_attribute>
                               <name>UDP_SRC_PORT</name>
                               <value>123</value>
                         </network_attribute>
                   </network_attributes>
               </profile>
               <profile>
                     <id>2</id>
                         <name>DNS_Reflection</name>
                         <description> profile description.</description>
                         <observation_interval_ms>15000</observation_interval_ms>
                      <threshold>10000</threshold>
                      <reduced_threshold>9000.0</reduced_threshold>
                         <action>NONE</action>
                         <last_modified_time>1439360370207</last_modified_time>
                         <last_modified_by>Administrator</last_modified_by>
                      <enabled>1</enabled>
                      <type>0</type>
                         <priority>4</priority>
                         <dscp_band_enabled>false</dscp_band_enabled>
                         <flow_timeout>0</flow_timeout>
                     <network_attributes>
                         <network_attribute>
                               <name>L3_PROTOCOL</name>
                               <value>UDP_TCP</value>
                      </network_attribute>
                         <network_attribute>
                               <name>UDP_SRC_PORT</name>
```

```
                              <value>53</value>
                        </network_attribute>
                        <network_attribute>
                              <name>TCP_SRC_PORT</name>
                              <value>53</value>
                        </network_attribute>
                  </network_attributes>
            </profile>

            <profile>
                  <id>6</id>
                  <name>Quote of the day</name>
                  <description>Quote of the day</description>
                  <observation_interval_ms>15000</observation_interval_ms>
                  <threshold>5000</threshold>
                  <reduced_threshold>4500.0</reduced_threshold>
                  <action>NONE</action>
                  <last_modified_time>1439359968047</last_modified_time>
                  <last_modified_by>Administrator</last_modified_by>
                  <enabled>1</enabled>
                  <type>0</type>
                  <priority>7</priority>
                  <dscp_band_enabled>false</dscp_band_enabled>
                  <flow_timeout>0</flow_timeout>
                  <network_attributes>
                        <network_attribute>
                              <name>L3_PROTOCOL</name>
                              <value>UDP_TCP</value>
                        </network_attribute>
                       <network_attribute>
                              <name>UDP_SRC_PORT</name>
                              <value>17</value>
                       </network_attribute>
                       <network_attribute>
                              <name>TCP_SRC_PORT</name>
                              <value>17</value>
                       </network_attribute>
                  </network_attributes>
            </profile>

      </profiles>
```

TABLE 21 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful request, a list of profiles details will be retrieved from the BFO.

Bulk Enabling of Profiles

This topic covers the process of bulk enabling of profiles in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profiles`

To bulk enable profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles

2. Define the overall traffic user parameters.
3. Set the HTTPS request method to PUT.

Bulk Enabling of Profiles Parameters

This topic covers parameters for bulk enabling profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Bulk Enabling of Profiles Examples

This topic covers the fetch request for bulk enabling of profiles for the Brocade Flow Optimizer (BFO).

Bulk enabling of profiles fetch request examples for BFO PUT operation includes:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example of bulk enabling of profiles.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - application/json

        Request Body

        {"action": "enable", "name": ["NTP_Reflection", "DNS_Reflection", "ICMP_Flood"]}

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" ?>
        <profiles>
               <action>enable</action>
               <name>NTP_Reflection</name>
               <name>DNS_Reflection</name>
               <name>ICMP_Flood</name>
        </profiles>
```

TABLE 22 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the profiles status will be enabled in the BFO.

Bulk Disabling of Profiles

This topic covers the process of bulk disabling of profiles in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profiles`

To bulk disable profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to PUT.

Bulk Disabling of Profiles Parameters

This topic covers parameters for bulk disabling of profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Bulk Disabling of Profiles Examples

This topic covers the request headers sent for bulk disabling of profiles to the Brocade Flow Optimizer (BFO).

Bulk disabling of profiles fetch request examples for BFO PUT operation includes:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example of bulk disabling of profiles.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - application/json

        Request Body

        {"action": "disable", "name": ["NTP_Reflection", "DNS_Reflection", "ICMP_Flood"]}

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" ?>
        <profiles>
               <action>disable</action>
               <name>NTP_Reflection</name>
               <name>DNS_Reflection</name>
               <name>ICMP_Flood</name>
        </profiles>
```

TABLE 23 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the profiles status will be disabled in the BFO.

Bulk Deletion of Profiles

This topic covers the process of bulk deletion of profiles in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profiles`

To bulk delete profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to PUT.

Bulk Deletion of Profiles Parameters

This topic covers parameters for bulk deletion of profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Bulk Deletion of Profiles Examples

This topic covers the request headers sent for bulk deletion of profiles to the Brocade Flow Optimizer (BFO).

Bulk deletion of profiles fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example of bulk deletion of profiles.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - application/json

        Request Body

        {"action": "delete", "name": ["NTP_Reflection", "DNS_Reflection", "ICMP_Flood"]}

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" ?>
        <profiles>
               <action>delete</action>
               <name>NTP_Reflection</name>
               <name>DNS_Reflection</name>
               <name>ICMP_Flood</name>
        </profiles>
```

TABLE 24 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the profiles status will be deleted in the BFO.

Moving Profiles to the Top Priority

This topic covers how to move profiles to the top priority in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/profiles`

To move profiles to the top priority in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Moving Profiles to the Top Priority Parameters

This topic covers parameters for moving profiles to the top priority for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Moving Profiles to the Top Priority Examples

This topic covers the fetch request for moving profiles to the top priority for the Brocade Flow Optimizer (BFO).

Moving profiles to the top priority fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example for moving profiles to the top priority.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json
       Accept - Application/json

       Request Body

       {"action": "top", "name": ["NTP_Reflection", "DNS_Reflection",
       "ICMP_Flood"]}

XML

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
       Accept - Application/xml

       Request Body

       <?xml version="1.0" encoding="UTF-8" ?>
       <profiles>
             <action>top</action>
             <name>NTP_Reflection</name>
             <name>DNS_Reflection</name>
             <name>ICMP_Flood</name>
       </profiles>
```

TABLE 25 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the selected profiles will be moved to top priority in the BFO.

Moving Profiles to the Bottom Priority

This topic covers how to move profiles to the bottom priority in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/profiles

To move profiles to the bottom priority in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Moving Profiles to the Bottom Priority Parameters

This topic covers parameters for moving profiles to the bottom priority for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Moving Profiles to the Bottom Priority Examples

This topic covers the fetch request for moving profiles to the bottom priority for the Brocade Flow Optimizer (BFO).

Moving profiles to the bottom priority fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example for moving profiles to the bottom priority.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - Application/json

        Request Body

        {"action": "bottom", "name": ["NTP_Reflection", "DNS_Reflection",
        "ICMP_Flood"]}

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" ?>
        <profiles>
              <action>bottom</action>
              <name>NTP_Reflection</name>
              <name>DNS_Reflection</name>
              <name>ICMP_Flood</name>
        </profiles>
```

TABLE 26 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the selected profiles will be moved to bottom priority in the BFO.

Moving Profiles Up to the Next Priority Level

This topic covers how to move profiles above their current priority level in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/profiles

To move the profiles priority up (to the next level above the profile's present priority level) in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Moving Profiles Up to the Next Priority Level Parameters

This topic covers parameters for moving profiles above their current priority level in the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Moving Profiles Up to the Next Priority Level Examples

This topic covers the fetch request for moving profiles above their current priority level for the Brocade Flow Optimizer (BFO).

Moving profiles up to the next priority level fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example for moving profiles up to the next priority level.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json
       Accept - Application/json

       Request Body

       {"action": "up", "name": ["NTP_Reflection", "DNS_Reflection",
       "ICMP_Flood"]}

XML

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
       Accept - Application/xml

       Request Body

       <?xml version="1.0" encoding="UTF-8" ?>
       <profiles>
             <action>up</action>
             <name>NTP_Reflection</name>
             <name>DNS_Reflection</name>
             <name>ICMP_Flood</name>
       </profiles>
```

TABLE 27 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the selected profiles will be moved one priority above their current priority in the BFO.

Moving Profiles Down to a Lower Priority Level

This topic covers how to move profiles below their current priority level in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/profiles

To move the profiles priority down (to a lower level than the profile's present priority level) in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profiles
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Moving Profiles Down to a Lower Priority Level Parameters

This topic covers parameters for moving profiles below their current priority level in the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Moving Profiles Down to a Lower Priority Level Examples

This topic covers the fetch request for moving profiles below their current priority level for the Brocade Flow Optimizer (BFO).

Moving profiles down to a lower priority level fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profiles
```

Example of Headers Sent to BFO

The following is an example for moving profiles down to a previous priority level.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - Application/json

        Request Body

        {"action": "down", "name": ["NTP_Reflection", "DNS_Reflection",
        "ICMP_Flood"]}

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" ?>
        <profiles>
            <action>down</action>
            <name>NTP_Reflection</name>
            <name>DNS_Reflection</name>
            <name>ICMP_Flood</name>
        </profiles>
```

TABLE 28 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, the selected profiles will be moved one priority below their current priority in the BFO.

Settings

Settings Overview

Settings in the Brocade Flow Optimizer (BFO) involve URIs for setting controller, email notification, sFlow, SNMP profile management, and device management.

Settings REST URIs provides some of the features listed below.

- Set the controller.
- Set email notification.
- Set sFlow settings.
- Set SNMP profile management.
- Set device management.

Controller Settings

This topic covers controller settings in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/controller`

The URIs return the controller information, which operation any user can perform. Password information is only returned if the user performing the operation has Administrator privileges (`access_privilege=1`). The `connection_status` specifies if controller credentials are valid, hence allowing the establishment of a connection to the controller. If no controller is present, then an empty controller object is returned.

To access controller settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/controller
2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Controller Settings Parameters

This topic covers parameters for controller settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE

The GET operation is supported.

Controller Settings Examples

This topic covers the fetch request for getting controller settings for the Brocade Flow Optimizer (BFO).

Controller settings fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/controller
```

Example of Headers Sent to BFO

The following is an example for getting controller settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept - Application/json

        Response Body

        {
            "controller": {
            "url": "https://10.24.41.109:8181",
            "username": "admin",
            "password": "admin",
            "connection_status": 1
           }
        }

XML

        Request Header

        Authorization - Guest_1426270596791
        Content-Type - Application/xml
        Accept - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
        <controller>
        <url>https://10.24.49.138:8181</url>
        <username>admin</username>
        <password>admin</password>
        <connection_status>1</connection_status>
        </controller>
```

TABLE 29 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful request, controller details will be retrieved from the BFO.

Add Controller Settings

This topic covers adding controller settings in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/controller`

The URIs return the controller information, which operation any user can perform. Password information is only returned if the user performing the operation has Administrator privileges (`access_privilege=1`). The `connection_status` specifies if controller credentials are valid, hence allowing the establishment of a connection to the controller. If no controller is present, then an empty controller object is returned.

To add controller settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/controller
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Add Controller Settings Parameters

This topic covers parameters for adding controller settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Add Controller Settings Examples

This topic covers the fetch request for adding controller settings for the Brocade Flow Optimizer (BFO).

Controller settings fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/controller
```

Example of Headers Sent to BFO

The following is an example for adding controller settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
            "url": "https://10.24.49.138:8181",
            "username": "admin",
            "password": "admin",
            "connection_status": 0
        }

XML

        Request Header

        Authorization - Guest_1426270596791
        Content-Type - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
        <controller>
        <url>https://10.24.49.138:8181</url>
        <username>admin</username>
        <password>admin</password>
        <connection_status>0</connection_status>
        </controller>
```

TABLE 30 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful request, controller settings will be added to the BFO.

Edit Controller Settings

This topic covers editing controller settings in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/controller`

Users with Administrator privileges (`access_privilege=1`) may update the controller information, including the URL, username, and password.

To edit controller settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/controller
2. Define the new user parameters.

3. Set the HTTPS request method to PUT.

Edit Controller Settings Parameters

This topic covers parameters for editing controller settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Edit Controller Settings Examples

This topic covers the fetch request for editing controller settings for the Brocade Flow Optimizer (BFO).

Controller settings fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/controller
```

Example of Headers Sent to BFO

The following is an example for editing controller settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
            "url": "https://10.24.41.110:8181",
            "username": "admin",
            "password": "admin",
            "connection_status": 0
        }

XML

        Request Header

        Authorization - Guest_1426270596791
        Content-Type - Application/xml

        Response Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
        <controller>
        <url>https://10.24.41.109:8181</url>
        <username>admin</username>
        <password>admin</password>
        <connection_status>0</connection_status>
        </controller>
```

TABLE 31 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful request, existing controller will be updated in the BFO.

Email Test Settings

This topic covers email test settings in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/email`

To access email notification settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/email
2. Define the test email settings to test the email configuration, before adding email notification settings Brocade Flow Optimizer.
3. Define the new user parameters.
4. Set the HTTPS request method to POST.

Email Test Settings Parameters

This topic covers test for email notification settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE

The POST operation is supported.

Email Test Settings Examples

This topic covers the fetch request for email test settings for the Brocade Flow Optimizer (BFO).

Email test settings fetch request examples for BFO POST operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/email
```

Example of Headers Sent to BFO

The following is an example for email test settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
            "enabled": "true",
            "id": "",
            "password": "",
            "server": "smtp.brocade.com",
            "port": "25",
            "replyAddress": "vnepolea@brocade.com",
            "emailAddresses": "vnepolea@brocade.com",
            "testRecipients": "vnepolea@brocade.com",
            "testEmail": "true"
        }

XML

        Request Header

        Authorization - Guest_1426270596791
        Content-Type - Application/xml

        Request Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <email>
            <enabled>true</enabled>
            <id> ""</id>
            <password>""</password>
            <server>smtp.brocade.com</server>
            <port>25</port>
            <replyAddress>Updated@brocade.com</replyAddress>
            <emailAddresses>vnepolea@brocade.com</emailAddresses>
            <testRecipients>vnepolea@brocade.com</testRecipients>
            <testEmail>true</testEmail>
        </email>
```

TABLE 32 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful execution, a test mail will be send to the configured email address.

Add Email Settings

This topic covers adding the email configuration in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/email

To add email settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/email
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Add Email Settings Parameters

This topic covers parameters for adding email settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Add Email Settings Examples

This topic covers the fetch request for adding email settings for the Brocade Flow Optimizer (BFO).

Email settings fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/email
```

Example of Headers Sent to BFO

The following is an example for adding email settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Response Body

        {
            "enabled": "true",
            "id": "",
            "password": "",
            "server": "smtp.brocade.com",
            "port": "25",
            "replyAddress": "vnepolea@brocade.com",
            "emailAddresses": "vnepolea@brocade.com"
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Response Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <email>
            <enabled>true</enabled>
            <id> ""</id>
            <password>""</password>
            <server>smtp.brocade.com</server>
            <port>25</port>
            <replyAddress>Updated@brocade.com</replyAddress>
            <emailAddresses>vnepolea@brocade.com</emailAddresses>
        </email>
```

TABLE 33 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful request, email configuration will be added to the BFO.

Update Email Settings

This topic covers updating the email configuration in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/email

To update email settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/email

2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Update Email Settings Parameters

This topic covers parameters for updating email settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Update Email Settings Examples

This topic covers the fetch request for updating email settings for the Brocade Flow Optimizer (BFO).

Email settings fetch request examples for BFO PUT operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/email
```

Example of Headers Sent to BFO

The following is an example for updating email settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Request Body

        {
            "enabled": "true",
            "id": "",
            "password": "",
            "server": "smtp.brocade.com",
            "port": "25",
            "replyAddress": "vnepolea@brocade.com",
            "emailAddresses": "vnepolea@brocade.com"
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Response Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <email>
            <enabled>true</enabled>
            <id> ""</id>
            <password>""</password>
            <server>smtp.brocade.com</server>
            <port>25</port>
            <replyAddress>Updated@brocade.com</replyAddress>
            <emailAddresses>vnepolea@brocade.com</emailAddresses>
        </email>
```

TABLE 34 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, existing email settings will be updated in the BFO.

Retrieve sFlow Settings, SNMP and sFlow Registration

This topic covers retrieving the sFlow settings for SNMP and sFlow registration in the Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/sFlowSettings

To retrieve sFlow settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/sFlowSettings

2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Retrieve sFlow Settings, SNMP and sFlow Registration Parameters

This topic covers parameters to retrieve sFlow settings for SNMP and sFlow registration for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE
The GET operation is supported.

Retrieve sFlow Settings, SNMP and sFlow Registration Examples

This topic covers the fetch request to retrieve sFlow settings for SNMP and sFlow registration for the Brocade Flow Optimizer (BFO).

sFlow settings for SNMP and sFlow registration fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/sFlowSettings
```

Example of Headers Sent to BFO

The following is an example for retrieving sFlow settings for SNMP and sFlow registration.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
         selectedNetworkInterface:
        {
         inBand: "99.99.99.146"
         outBand: "99.99.99.146"
        }
        networkInterfaceList:
        {
         ipaddress: [12]
        0:  "192.168.122.1"
        1:  "99.99.99.146"
        2:  "2620:100:0:fe07:250:56ff:fe87:6109%eth0"
        3:  "2620:100:0:fe07:55bb:c699:b1f3:bfe7%eth0"
        4:  "2620:100:0:fe07:c892:9f5:cf2:e2a1%eth0"
        5:  "2620:100:0:fe07:41d0:b22e:a827:a5f%eth0"
        6:  "2620:100:0:fe07:24ed:21c9:f2c7:4037%eth0"
        7:  "2620:100:0:fe07:a8ca:27f4:3d2f:9886%eth0"
        8:  "2620:100:0:fe07:c4bf:e9ef:d514:b584%eth0"
        9:  "2620:100:0:fe07:7d91:fddc:22a5:9de8%eth0"
        10:  "2620:100:0:fe07:2c4c:a239:5203:f3c8%eth0"
        11:  "10.24.49.146"
         }
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept-Content - Application/xml
```

TABLE 35 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the sFlow settings details are retrieved in the BFO.

Add sFlow Settings

This topic covers adding the sFlow settings for SNMP and sFlow registration in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/sFlowSettings`

To add sFlow settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/sFlowSettings
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Add sFlow Settings Parameters

This topic covers parameters for adding sFlow settings for SNMP and sFlow registration for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Add sFlow Settings Examples

This topic covers the fetch request for adding sFlow settings for SNMP and sFlow registration for the Brocade Flow Optimizer (BFO).

Adding sFlow settings for SNMP and sFlow registration fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/sFlowSettings
```

Example of Headers Sent to BFO

The following is an example for adding sFlow settings for SNMP and sFlow registration.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
            "selectedNetworkInterface": {
                "inBand": "172.26.1.61",
                "outBand": "172.26.1.61"
            }
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Request Body
        <?xml version="1.0" encoding="UTF-8" ?>
        <sflowsettings>
           <selectedNetworkInterface>
               <inBand>172.26.1.61</inBand>
               <outBand>172.26.1.61</outBand>
           </selectedNetworkInterface>
        </sflowsettings>
```

TABLE 36 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the sFlow settings are added in the BFO.

Update Existing sFlow Settings

This topic covers updating the existing sFlow settings in Brocade Flow Optimizer (BFO).

Resource URL
<Base URI>/collector/sFlowSettings

To update sFlow settings in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/sFlowSettings
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Update Existing sFlow Settings Parameters

This topic covers parameters for updating the existing sFlow settings for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Update Existing sFlow Settings Examples

This topic covers the fetch request for updating existing sFlow settings for the Brocade Flow Optimizer (BFO).

Updating existing sFlow settings fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/sFlowSettings
```

Example of Headers Sent to BFO

The following is an example for updating existing sFlow settings.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
            "selectedNetworkInterface": {
                "inBand": "172.26.1.61",
                "outBand": "172.26.1.61"
            }
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Request Body
        <?xml version="1.0" encoding="UTF-8" ?>
        <sflowsettings>
           <selectedNetworkInterface>
               <inBand>172.26.1.61</inBand>
               <outBand>172.26.1.61</outBand>
           </selectedNetworkInterface>
        </sflowsettings>
```

TABLE 37 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the sFlow settings are updated in the BFO.

Retrieve SNMP Profiles

This topic covers retrieving the SNMP profiles in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/snmpProfiles`

To retrieve SNMP profiles in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/snmpProfiles
2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Retrieve SNMP Profiles

This topic covers parameters to retrieve SNMP profiles for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE

The GET operation is supported.

Retrieve SNMP Profiles Examples

This topic covers the fetch request to retrieve SNMP profiles for the Brocade Flow Optimizer (BFO).

SNMP profiles fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/snmpProfiles
```

Example of Headers Sent to BFO

The following is an example for retrieving SNMP profiles.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
          "snmp_profile": [
            {
              "name": "snmpsettingv1",
              "type": "V_1",
              "rw_community_string": "private",
              "order": 1,
              "last_modified_on": "Mon Aug 17 16:14:57 PDT 2015"
            },
            {
              "name": "snmpsettingv3",
              "type": "V_3",
              "username": "md5aesuser",
              "auth_protocol": "HMAC_MD_5",
              "auth_password": "md5password",
              "priv_protocol": "CFB_AES_128",
              "priv_password": "aespasswords",
              "order": 2,
              "last_modified_on": "Mon Aug 17 16:16:09 PDT 2015"
            }
          ]
        }
```

XML

```
        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept-Content - Application/xml

        Response Body
        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <snmp_profiles>
            <snmp_profile>
                <name>snmpsettingv1</name>
                <type>v1</type>
                <rw_community_string>private
                </rw_community_string>
                <order>1</order>
                <last_modified_on>Mon Aug 17 16:14:57 PDT 2015
                </last_modified_on>
            </snmp_profile>
            <snmp_profile>
                <name>snmpsettingv3</name>
                <type>v3</type>
                <username>md5aesuser</username>
                <auth_protocol>hmac_md5</auth_protocol>
                <auth_password>md5password</auth_password>
                <priv_protocol>cfb_aes_128</priv_protocol>
                <priv_password>aespasswords</priv_password>
                <order>2</order>
                <last_modified_on>Mon Aug 17 16:16:09 PDT 2015
                </last_modified_on>
            </snmp_profile>
        </snmp_profiles>
```

TABLE 38 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the SNMP profiles are retrieved in the BFO.

Create SNMP Profile

This topic covers creating the SNMP profile in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/snmpProfiles`

To create an SNMP profile in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/snmpProfiles
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Create SNMP Profile Parameters

This topic covers parameters for creating an SNMP profile for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Create SNMP Profile Examples

This topic covers the fetch request for creating an SNMP profile for the Brocade Flow Optimizer (BFO).

Creating SNMP profile fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/snmpProfiles
```

Example of Headers Sent to BFO

The following is an example for adding sFlow settings for SNMP and sFlow registration.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Request Body

        {
        "name": "snmpsettingv1_another",
        "type": "V_1",
        "rw_community_string": "something_private"
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Request Body
        <snmp_profile>
            <name>snmpsettingv3_another</name>
            <type>v3</type>
            <username>md5aesuser_another</username>
            <auth_protocol>hmac_md5</auth_protocol>
            <auth_password>md5password_another</auth_password>
            <priv_protocol>cfb_aes_128</priv_protocol>
            <priv_password>aespasswords_another</priv_password>
        </snmp_profile>
```

TABLE 39 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the new SNMP profile is added in the BFO.

Update Existing SNMP Profile

This topic covers updating an existing SNMP profile in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/snmpProfiles/<profile name>`

One profile is updated at a time, the name of which appears as a path parameter in the URI. The request payload is a `snmp_profile` object. Only user with Administrator privileges (`access_privilege=1`) can perform this operation. Order cannot be changed by this operation. There is a different API provided to perform a swap of order for two profiles.

To update an SNMP profile in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/snmpProfiles/<profile name>
2. Define the new user parameters.

3. Set the HTTPS request method to PUT.

Update Existing SNMP Profile Parameters

This topic covers parameters for updating an existing SNMP profile for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Update Existing SNMP Profile Examples

This topic covers the fetch request for updating an existing SNMP profile for the Brocade Flow Optimizer (BFO).

Updating existing SNMP profile fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/snmpProfiles/<profile name>
```

Example of Headers Sent to BFO

The following is an example for updating an existing SNMP profile.

```
JSON

        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json

        Request Body

        {
        "name": "snmpsettingv1_another",
        "type": "V_1",
        "rw_community_string": "new_password"
        }

XML

        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml

        Request Body
        <snmp_profile>
            <name>snmpsettingv3_another</name>
            <priv_protocol>cfb_aes_128</priv_protocol>
            <priv_password>new_aespasswords</priv_password>
        </snmp_profile>
```

TABLE 40 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, the existing SNMP profile is updated in the BFO.

Delete Existing SNMP Profile

This topic covers the process of deleting an existing SNMP profile in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/snmpProfiles/<profile name>`

One profile is deleted at a time, the name of which appears as a path parameter in the URI. Only user with Administrator privileges (`access_privilege=1`) can perform this operation.

To delete an existing SNMP profile in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/snmpProfile/<profile name>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to DELETE.

Delete Existing SNMP Profile Parameters

This topic covers parameters for deleting an existing SNMP profile for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE

The DELETE operation is supported.

Delete Existing SNMP Profile Examples

This topic covers the fetch request for deleting an existing SNMP profile for the Brocade Flow Optimizer (BFO).

Delete existing SNMP profile fetch request examples for BFO DELETE operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/snmpProfiles/<profile name>
```

Example of Headers Sent to BFO

The following is an example of deleting a custom profile.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type: application/json

XML

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
```

TABLE 41 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful operation, existing SNMP profile will be deleted in the BFO.

Device Management

Device Management

Device management in the Brocade Flow Optimizer (BFO) involves URIs for retrieving controller and managed devices, registering devices, updating registered managed devices, and de-registering managed devices.

Device management REST URIs provides some of the features listed below.

- Retrieving controller and managed devices.
- Registering devices.
- Updating registered managed devices.
- De-registering managed devices.

Retrieve Controller and Managed Devices

This topic covers retrieving controller and managed devices in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/managedDevices`
`<Base URI>/collector/managedDevices?ips={value}`

The list of retrieved controller and managed devices includes those devices currently managed by the application and those available in the controller. The returned object is a `managed_devices` object containing a list of `managed_device` instances.

- The managed property indicates whether the device is currently managed or not.
- The status and statusMessage relays information regarding the status of the managed device, like whether it became unreachable or was removed from the controller. Any user can perform this operation.

The API also provides the user with an optional query parameter ips. This parameter can be used to retrieve specific devices. Only devices matching the listed ips will be returned.

- In the event that the device has maxed out collector addresses (maximum of 4 collectors allowed on device), then the port list will be empty, as no more ports can be registered for sflow.
- If controller has not been configured then an appropriate error is returned. If SNMP profile has not been configured then an appropriate error is returned.

To retrieve controller and manage devices in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/managedDevices or https://localhost:8089/collector/managedDevices?ips={value}
2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Retrieve Controller and Managed Devices Parameters

This topic covers parameters to retrieve controller and managed devices for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
ips	The comma separated IP addresses of devices to be retrieved.
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE
The GET operation is supported.

Retrieve Controller and Managed Devices Examples

This topic covers the fetch request to retrieve controller and managed devices for the Brocade Flow Optimizer (BFO).

Retrieving controller and managed device fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/managedDevices
```

```
https://localhost:8089/collector/managedDevices?ips=(value}
```

Example of Headers Sent to BFO

The following is an example for retrieving sFlow settings for SNMP and sFlow registration.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
          "managed_device": [
            {
              "ip_address": "10.24.63.240",
              "device_type": "MLX",
              "status": 0,
              "statusMessage": "Success",
              "managed": 0,
              "snmp_profile_name": "",
              "last_modified_on": ""
              "is-trigger": "false"
            },
            {
              "ip_address": "10.24.63.241",
              "device_type": "MLX",
              "status": 0,
              "statusMessage": "Success",
              "managed": 0,
              "snmp_profile_name": "",
              "last_modified_on": ""
              "is-trigger": "true"
            },
            {
              "ip_address": "10.24.50.180",
              "device_type": "MLX",
              "status": 0,
              "statusMessage": "Success",
              "managed": 0,
              "snmp_profile_name": "",
              "last_modified_on": ""
              "is-trigger": "false"
            }
          ]
        }
```

XML

```
        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept-Content - Application/xml

        Response Body

        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <managed_devices>
            <managed_device>
                <ip_address>10.24.63.240</ip_address>
                <device_type>mlx</device_type>
                <status>0</status>
                <statusMessage>Success</statusMessage>
                <managed>0</managed>
                <snmp_profile_name></snmp_profile_name>
                <last_modified_on></last_modified_on>
                <is-trigger>true</is-trigger>
            </managed_device>
            <managed_device>
```

```
            <ip_address>10.24.63.241</ip_address>
            <device_type>mlx</device_type>
            <status>0</status>
            <statusMessage>Success</statusMessage>
            <managed>0</managed>
            <snmp_profile_name></snmp_profile_name>
            <last_modified_on></last_modified_on>
            <is-trigger>true</is-trigger>
        </managed_device>
        <managed_device>
            <ip_address>10.24.50.180</ip_address>
            <device_type>mlx</device_type>
            <status>0</status>
            <statusMessage>Success</statusMessage>
            <managed>0</managed>
            <snmp_profile_name></snmp_profile_name>
            <last_modified_on></last_modified_on>
            <is-trigger>true</is-trigger>
        </managed_device>
    </managed_devices>
```

TABLE 42 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Register Devices

This topic covers registering devices in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/managedDevices`

Allows the user to register with one or more devices for sFlow. Only user with Administrator privileges (`access_privilege=1`) can perform this operation. The request object is a `managed_devices` object containing a list of `managed_device` instances.

The user can select from list of available devices from the controller. Each device in the input list must have at least one port enabled for sFlow. Once registered, these devices are then managed devices, and thus monitored for flows by the application.

To register devices in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/managedDevices
2. Define the new user parameters.
3. Set the HTTPS request method to POST.

Register Devices Parameters

This topic covers parameters for registering devices for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json

Parameter Name	Parameter Value
	XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Register Devices Examples

This topic covers the fetch request for registering devices for the Brocade Flow Optimizer (BFO).

Registering devices fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/managedDevices
```

Example of Headers Sent to BFO

The following is an example for registering devices.

JSON

```
Request header

Authorization - Administrator_1426270596791
Content-Type - Application/json
Accept-Content - Application/json

Request Body

{
  "managed_device": [
    {
      "ip_address": "10.24.63.240",
      "device_type": "MLX",
      "intrfaces": {
        "intrface": [
          {
            "if_index": 1,
            "if_description": "1/1",
            "sflow_enabled": 1
          },
          {
            "if_index": 2,
            "if_description": "1/2",
            "sflow_enabled": 1
          }
        ]
      }
    }
  ]
}
```

XML

```
Request Header

Authorization - Administrator_1426270596791
Content-Type - Application/xml
Accept-Content - Application/json

Request Body
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<managed_devices>
    <managed_device>
        <ip_address>10.24.63.240</ip_address>
        <device_type>mlx</device_type>
        <intrfaces>
            <intrface>
                <if_index>2</if_index>
                <if_description>1/2</if_description>
                <sflow_enabled>1</sflow_enabled>
            </intrface>
    </managed_device>
    <managed_device>
        <ip_address>10.24.63.241</ip_address>
        <device_type>mlx</device_type>
        <intrfaces>
            <intrface>
                <if_index>2</if_index>
                <if_description>1/2</if_description>
                <sflow_enabled>1</sflow_enabled>
            </intrface>
        </intrfaces>
    </managed_device>
</managed_devices>
```

TABLE 43 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Update Registered Managed Devices

This topic covers updating registered managed devices in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/managedDevices`

Allows the user to register with one or more devices for sFlow. Only user with Administrator privileges (`access_privilege=1`) can perform this operation. The user edits managed ports of devices by selecting from a list of already managed devices.

To update registered managed devices in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/managedDevices
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Update Registered Managed Devices Parameters

This topic covers parameters for updating registered managed devices for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE
The PUT operation is supported.

Update Registered Managed Devices Examples

This topic covers the fetch request for updating registered managed devices for the Brocade Flow Optimizer (BFO).

Updating registered managed devices fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/managedDevices
```

Example of Headers Sent to BFO

The following is an example for updating registered managed devices.

JSON

```
        Request header

        Authorization - Administrator_1426270596791
        Content-Type - Application/json
        Accept-Content - Application/json

        Request Body

        {
          "managed_device": [
            {
              "ip_address": "10.24.63.240",
              "device_type": "MLX",
                     "intrfaces": {
                "intrface": [
                  {
                    "if_index": 1,
                    "if_description": "1/1",
                    "sflow_enabled": 1
                  },
                  {
                    "if_index": 2,
                    "if_description": "1/2",
                    "sflow_enabled": 1
                  }
                ]
              }
            }
          ]
        }
```

XML

```
        Request Header

        Authorization - Administrator_1426270596791
        Content-Type - Application/xml
        Accept-Content - Application/xml

        Request Body
        <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
        <managed_devices>
            <managed_device>
                <ip_address>10.24.63.240</ip_address>
                <device_type>mlx</device_type>
                <intrfaces>
                    <intrface>
                        <if_index>1</if_index>
                        <if_description>1/1</if_description>
                        <sflow_enabled>1</sflow_enabled>
                    </intrface>
                    <intrface>
                        <if_index>2</if_index>
                        <if_description>1/2</if_description>
                        <sflow_enabled>1</sflow_enabled>
                    </intrface>
                </intrfaces>
            </managed_device>
            <managed_device>
                <ip_address>10.24.63.241</ip_address>
                <device_type>mlx</device_type>
                <intrfaces>
                    <intrface>
                        <if_index>2</if_index>
                        <if_description>1/2</if_description>
                        <sflow_enabled>1</sflow_enabled>
```

```
                </intrface>
                <intrface>
                    <if_index>3</if_index>
                    <if_description>1/3</if_description>
                    <sflow_enabled>1</sflow_enabled>
                </intrface>
            </intrfaces>
        </managed_device>
    </managed_devices>
```

TABLE 44 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

De-register Managed Devices

This topic covers de-registering managed devices in Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/managedDevices/<device_IP>`
`<Base URI>/collector/managedDevices/<device_IP>?deleteOnly={value}`

To de-register managed devices in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/managedDevices/<device_IP> or https://localhost:8089/collector/managedDevices/<device_IP>?deleteOnly={value}
2. Define the new user parameters.
3. Set the HTTPS request method to DELETE.

De-register Managed Devices Parameters

This topic covers parameters for de-registering managed devices for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
deleteOnly	An integer value to force delete of the managed device from the application. Set to 1 to delete devices only.
Authorization	The session token returned after a successful login.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	DELETE

NOTE

The DELETE operation is supported.

De-register Managed Devices Examples

This topic covers the fetch request for de-registering managed devices for the Brocade Flow Optimizer (BFO).

De-registering managed devices fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

`https://localhost:8089/collector/managedDevices/<device_IP>` or `https://localhost:8089/collector/managedDevices/<device_IP>?deleteOnly={value}`

Example of Headers Sent to BFO

The following is an example for de-registering managed devices.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json
       Accept-Content - Application/json

XML

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
       Accept-Content - Application/json
```

TABLE 45 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Remotely Triggered Black Hole for BGP

RTBH for BGP Overview

The Remotely Triggered Black Hole (RTBH) drops undesirable traffic before it enters a protected network.

On detection of a DDoS attack by BFO, RTBH filtering can be used to selectively drop undesirable traffic destined to enter the attack destination. RTBH for BGP REST URIs provides some of the features listed below.

- RTBH filtering includes DDoS as one of its many applications.
- The Brocade Flow Optimizer (BFO) assists in detecting and mitigating DDoS attacks.

Create New RTBH Custom Profile

This topic covers the process of creating a new Remotely Triggered Black Hole (RTBH) custom profile in the Brocade Flow Optimizer (BFO).

NOTE

To create a new custom profile (without RTBH), see the topic *Create New Custom Profile*.

Resource URL
`<Base URI>/collector/profile`

To create a new RTBH custom profile in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/profile
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Create New RTBH Custom Profile Parameters

This topic covers parameters for creating new Remotely Triggered Black Hole (RTBH) custom profiles for the Brocade Flow Optimizer (BFO).

NOTE

To create new custom profile parameters (without RTBH), see the topic *Create New Custom Profile Parameters*.

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml

Parameter Name	Parameter Value
Operation	POST

NOTE
The POST operation is supported.

Create New RTBH Custom Profile Examples

This topic covers the fetch request for creating new Remotely Triggered Black Hole (RTBH) custom profiles for the Brocade Flow Optimizer (BFO).

NOTE
To see examples for creating a new custom profile (without RTBH), see the topic *Create New Custom Profile Examples*.

Create new RTBH custom profile fetch request examples for BFO POST operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/profile
```

Example of Headers Sent to BFO

The following is an example of creating a new RTBH custom profile.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json
       Accept-Content - Application/json

       Request Body

       <xs:complexType name="profile">
             <xs:sequence>
                   <xs:element type="xs:int" name="id" minOccurs="0" />
                   <xs:element type="xs:string" name="name" minOccurs="0" />
                   <xs:element type="xs:string" name="description" minOccurs="0" />
                   <xs:element type="xs:int" name="observation_interval_ms" minOccurs="0" />
                   <xs:element type="xs:int" name="threshold" minOccurs="0" />
                   <xs:element type="xs:float" name="reduced_threshold" minOccurs="0" />
                   <xs:element type="xs:string" name="action" minOccurs="0" />
                   <xs:element type="xs:long" name="last_modified_time" minOccurs="0" />
                   <xs:element type="xs:int" name="tag" minOccurs="0" />
             <xs:element type="xs:int" name="prefix_length" minOccurs="0" />
             <xs:element type="xs:string" name="last_modified_by" minOccurs="0" />
                   <xs:element type="xs:int" name="enabled" minOccurs="0" />
                   <xs:element type="xs:int" name="type" minOccurs="0" />
                   <xs:element type="xs:int" name="priority" minOccurs="0" />
                   <xs:element type="xs:boolean" name="dscp_band_enabled" minOccurs="0" />
                   <xs:element type="xs:int" name="rate_limit" minOccurs="0" />
                   <xs:element type="xs:int" name="dscp_rate_limit" minOccurs="0" />
                   <xs:element type="xs:int" name="dscp_prec_level" minOccurs="0" />
                   <xs:element type="xs:int" name="flow_timeout" minOccurs="0" />
                   <xs:element type="network_attributes" name="network_attributes"
                minOccurs="0" />
                   <xs:element type="redirect_nodes" name="redirect_nodes" minOccurs="0" />
               </xs:sequence>
       </xs:complexType>
```

TABLE 46 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

On successful update, a new custom profile is created in the BFO.

Create RTBH User Defined Flow

This topic covers the process of creating a custom Remotely Triggered Black Hole (RTBH) user defined flow, by providing the match criteria and mitigation action. Mitigating attack traffic using RTBH filtering is accomplished by defining a user defined flow.

NOTE

To create a new user defined flow (without RTBH), see the topic *Create User Defined Flow*.

Resource URL
`<Base URI>/collector/customFlows`

The user will be able to provide the RTBH flow priority above or below the learned and programmed flows.

To access RTBH custom open flows in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/customFlows
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Create RTBH User Defined Flow Parameters

This topic covers parameters for creating Remotely Triggered Black Hole (RTBH) user defined flows for the Brocade Flow Optimizer (BFO).

NOTE
To create new user defined flow parameters (without RTBH), see the topic *Create User Defined Flow Parameters*.

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Content-Type	JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE
The POST operation is supported.

Create RTBH User Defined Flow Examples

This topic covers the fetch request for creating Remotely Triggered Black Hole (RTBH) user defined flows for the Brocade Flow Optimizer (BFO) server.

NOTE
To see examples for creating a new user defined flow (without RTBH), see the topic *Create User Defined Flow Examples*.

The create RTBH user defined flows fetch request examples for BFO POST operation for flows include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/customFlows
```

Example of Headers Sent to BFO

The following is an example of creating user defined flows.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Accept - Application/json
       Accept - Application/xml

       Request Payload

       <xs:complexType name="custom_flow">
          <xs:sequence>
                   <!-- This property holds a user defined name for the flow. -->
                      <xs:element name="flow_name" type="xs:string" minOccurs="0" />
                      <!-- Please leave the Flow Id empty. We will generate the same and return
                in the response -->
                      <xs:element name="flow_id" type="xs:string" minOccurs="0" />
                      <!-- Please set this fields to true if you want to create a flow with
                priority higher than automatic programmed flows by BFO. If set false
                it will be set low priority -->
                      <xs:element name="highest_priority" type="xs:boolean" minOccurs="0" />
                      <!-- Please leave this field empty if you are using this REST URL out of
                BFO -->
                      <xs:element name="flow_checksum" type="xs:long" minOccurs="0" />
                      <xs:element name="source_mac" type="xs:string" minOccurs="0" />
                      <xs:element name="destination_mac" type="xs:string" minOccurs="0" />
                      <xs:element name="ether_type" type="xs:int" minOccurs="0" />
                      <xs:element name="vlan_id" type="xs:int" minOccurs="0" />
                      <xs:element name="vlan_id_present" type="xs:boolean" minOccurs="0" />
                      <xs:element name="vlan_priority" type="xs:int" minOccurs="0" />
                      <xs:element name="source_ip" type="xs:string" minOccurs="0" />
                      <xs:element name="destination_ip" type="xs:string" minOccurs="0" />
                      <xs:element name="ip_type" type="ip_address_format_type" minOccurs="0" />
                      <xs:element name="ip_protocol" type="l3_protocol_type" minOccurs="0" />
                      <xs:element name="ip_dscp" type="xs:int" minOccurs="0" />
                      <xs:element name="source_port" type="xs:int" minOccurs="0" />
                      <xs:element name="destination_port" type="xs:int" minOccurs="0" />
                      <xs:element name="action" type="action" minOccurs="1" maxOccurs="unbounded"/>
                      <xs:element type="xs:int" name="tag" minOccurs="0" />
                      <xs:element name="learned_flow" type="xs:boolean" minOccurs="0" />
                      <xs:element type="xs:long" name="byteCount" minOccurs="1" />
                      <xs:element type="xs:long" name="packetCount" minOccurs="1" />
            </xs:sequence>
      </xs:complexType>

      <xs:simpleType name="mitigation_action_type">
            <xs:annotation>
                  <xs:documentation>Enumeration of the various mitigation actions
                  </xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:string">
                  <xs:enumeration value="DROP" />
                  <xs:enumeration value="REDIRECT" />
                  <xs:enumeration value="METER" />
                  <xs:enumeration value="MIRROR" />
                  <xs:enumeration value="RTBH" />
               </xs:restriction>
      </xs:simpleType>
```

TABLE 47 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Select New RTBH Trigger Device

This topic covers the process of selecting a new Remotely Triggered Black Hole (RTBH) trigger device in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/manageddevice/{deviceip}?trigger=<true>&force=<true>`

To select a new RTBH trigger device in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/manageddevice/{deviceip}?trigger=<true>&force=<true>
2. Define the overall traffic user parameters.
3. Set the HTTPS request method to POST.

Select New RTBH Trigger Device Parameters

This topic covers parameters for selecting a new Remotely Triggered Black Hole (RTBH) trigger device for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	POST

NOTE

The POST operation is supported.

Select New RTBH Trigger Device Examples

This topic covers the fetch request for selecting a trigger device for a Remotely Triggered Black Hole (RTBH) in the Brocade Flow Optimizer (BFO).

Select a new RTBH trigger device fetch request examples for BFO POST operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/manageddevice/{deviceip}?trigger=<true>&force=<true>
```

Example of Headers Sent to BFO

The following is an example for selecting a trigger device for a RTBH in the BFO.

```
JSON

       Request Header

       Authorization - Administrator_1426270596791
       Content-Type - Application/xml
       Accept-Content - Application/xml

       Request Body

       <xs:complexType name="managed_device">
          <xs:sequence>
             <xs:element type="xs:string" name="ip_address" />
             <xs:element type="device_type" name="device_type" default="mlx" />
             <xs:element type="intrfaces" name="intrfaces" />
             <xs:element type="intrfaces" name="openflow_intrfaces" />
             <xs:element type="xs:int" name="status" default="0" />
             <xs:element type="xs:string" name="statusMessage" />
             <xs:element type="xs:int" name="managed" default="0" />
             <xs:element type="xs:string" name="snmp_profile_name" />
             <xs:element type="xs:string" name="last_modified_on" />
             <xs:element type="xs:boolean" name="is_trigger" default="false" />
          </xs:sequence>
       </xs:complexType>
```

TABLE 48 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Adding SSH Credentials

This topic covers adding SSH credentials for the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/manageddevice/{deviceip}/credentials`

To use RTBH filtering support for adding SSH credentials provided by BFO for DDoS attack mitigation, complete the following steps:

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/manageddevice/{deviceip}/credentials
2. Define the new user parameters.
3. Set the HTTPS request method to PUT.

Adding SSH Credentials Parameters

This topic covers parameters for adding SSH credentials for the Remotely Triggered Black Hole (RTBH) for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
deviceIP	IP of the device where credentials are set.
Authorization	Authorized token.
Content-Type	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	PUT

NOTE

The PUT operation is supported.

Adding SSH Credentials Examples

This topic covers adding SSH credentials for the Remotely Triggered Black Hole (RTBH) for BGP for the Brocade Flow Optimizer (BFO).

Mitigation of attack traffic is achieved by configuring a static route on the trigger device by BFO. This is accomplished by programming a CLI command via SSH. In order to login to the device via SSH, we will need the device credentials.

Adding SSH credentials examples for the Remotely Triggered Black Hole (RTBH) for BGP BFO POST operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/manageddevice/{deviceip}/credentials
```

Example of Headers Sent to BFO

The following is an example for adding SSH credentials.

```
JSON

       Request header

       Authorization - Administrator_1426270596791
       Content-Type - Application/json

       XML Request sample

       <xs:complexType name="managed_device_credentials">
          <xs:sequence>
             <xs:element type="xs:string" name="ssh_username" />
             <xs:element type="xs:string" name="ssh_password" />
             <xs:element type="xs:string" name="ssh_enable_username" />
             <xs:element type="xs:string" name="ssh_enable_password" />
          <xs:sequence>
       <xs:complexType>
```

TABLE 49 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Events

Events Overview

Events in the Brocade Flow Optimizer (BFO) involve URIs for retrieving events.

Retrieve Events

This topic covers retrieving events in the Brocade Flow Optimizer (BFO).

Resource URL
`<Base URI>/collector/events`
`<Base URI>/collector/events?timeline={}&duration={}&offset={}&limit={}`

The returned object is an events object containing a list of event instances. Any user can perform this operation. The API provides a few query parameters to filter for the events. If none are specified, then the last 30 minutes of events are returned.

User can specify timeline and duration or offset and limit.

- By specifying timeline and duration, user can retrieve events for the specified duration from the specified timeline.
- By specifying the offset and limit, user can retrieve events in pages starting from a given offset and containing limit number of events per page.

Timeline takes precedence over offset.

- If both timeline and offset are null, then timeline is considered and is set to default.
- If both timeline and offset are specified and not null, then timeline is considered, offset is ignored.

To retrieve events in the BFO, complete the following steps.

1. Enter the following URI in the URL field of your REST client tool: https://localhost:8089/collector/events or https://localhost:8089/collector/events?timeline={}&duration={}&offset={}&limit={}
2. Define the new user parameters.
3. Set the HTTPS request method to GET.

Retrieve Events Parameters

This topic covers parameters to retrieve events for the Brocade Flow Optimizer (BFO).

Parameter Name	Parameter Value
Authorization	Authorized token.
Accept	The content type of the returned data. JSON format: Application/json XML format: Application/xml
Operation	GET

NOTE
The GET operation is supported.

Query Parameter Name	Value	Default Value
timeline	The time from which the duration is referenced in milliseconds.	Current time
duration	The duration from the timeline in milliseconds.	180,000 milliseconds (30 minutes)
offset	The numerical offset from 0 to get events in pages.	0
limit	The number of events per page.	1,000

Retrieve Events Examples

This topic covers the fetch request to retrieve events for the Brocade Flow Optimizer (BFO).

Retrieving events fetch request examples for BFO GET operation include:

- BFO URI
- BFO request headers sent

Example of BFO URI

The following is an example of the BFO URI.

```
https://localhost:8089/collector/events
```

```
https://localhost:8089/collector/events?timeline={}&duration={}&offset={}&limit={}
```

Example of Headers Sent to BFO

The following is an example for retrieving events.

JSON

```
Request header

Authorization - Administrator_1426270596791
Content-Type - Application/json
Accept-Content - Application/json

Request Body

{
  "timeline": 1439872962749,
  "duration": 1800000,
  "offset": null,
  "limit": null,
  "event": [
    {
      "description": "Profiles NTP_Reflection  disabled Successfully",
      "category": "AUDIT",
      "messageID": "1007",
      "timestamp": 1439872936431,
      "severity": "INFO",
      "profileName": null
    },
    {
      "description": "Profiles ICMP_Flood  disabled Successfully",
      "category": "AUDIT",
      "messageID": "1007",
      "timestamp": 1439872944487,
      "severity": "INFO",
      "profileName": null
    },
    {
      "description": "Profiles NTP_Reflection ,ICMP_Flood  enabled Successfully",
      "category": "AUDIT",
      "messageID": "1007",
      "timestamp": 1439872948528,
      "severity": "INFO",
      "profileName": null
    }
  ]
}
```

XML

```
Request Header

Authorization - Administrator_1426270596791
Content-Type - Application/xml
Accept-Content - Application/json

Response Body

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<events>
    <timeline>1439873035912</timeline>
    <duration>1800000</duration>
    <event>
        <description>Profiles NTP_Reflection
        disabled Successfully</description>
        <category>AUDIT</category>
        <messageID>1007</messageID>
        <timestamp>1439872936431</timestamp>
        <severity>INFO</severity>
    </event>
    <event>
        <description>Profiles ICMP_Flood
        disabled Successfully</description>
```

```
            <category>AUDIT</category>
            <messageID>1007</messageID>
            <timestamp>1439872944487</timestamp>
            <severity>INFO</severity>
        </event>
        <event>
            <description>Profiles NTP_Reflection
            ,ICMP_Flood enabled Successfully</description>
            <category>AUDIT</category>
            <messageID>1007</messageID>
            <timestamp>1439872948528</timestamp>
            <severity>INFO</severity>
        </event>
    </events>
```

TABLE 50 Status Code

HTTPS Status Code	Description
200 OK	SUCCESS
400 Bad Request Error	FAILED, reason included in HTTPS error response
500 Internal Server Error	FAILED, reason included in HTTPS error response

Protocol Support

Protocol Support Default HTTPS

The Brocade Flow Optimizer (BFO) protocol support is exclusively supported for HTTPS.

NOTE
BFO works on HTTPS only. All REST HTTPS requests will be rejected.

All REST operations will be provided at the following base context path.

TABLE 51 HTTPS Protocol Support

HTTPS Default Port	HTTPS URL
8089	https://<BFO Server IP>:8089

ContentType Support

ContentType Support REST Requests

All REST requests returning data support both XML and JSON formats.

NOTE
Depending on the content type requested by the user, the proper format of data is returned.

Accept HTTPS Request Header

For GET requests, the client must specify the data format of responses. This is done by filling in the HTTPS header information.

The user must add the header while forming the HTTPS GET request. Supported header values are shown in the following table.

TABLE 52 Accept HTTPS Request Header

Request Header Name	Request Header Value	Response Data Format
Accept	Application/xml	XML
Accept	Application/json	JSON

The content type for the response data is specified through the HTTPS request header named Accept. Conditions for Accept HTTPS request headers include the following:

- Value for the content type in the following format:

  `MEDIA type/MIME subtype;Version Identifier`

  The default version value is the latest version, unless the user specifies a specific version value in the form `vx`, where `x` is the latest version for the Accept HTTPS request header name.
- The default format value: `xml` (unless the user specifies a format value for the Accept HTTPS request header name).

Content-type HTTPS Request Header

In the case of POST requests, in addition to specifying the format of the data of the responses via the Accept header, clients must also specify the format of the data that they are sending through the input request payload by filling in the HTTPS *Content-type* header.

The content type for the request data is specified through the HTTPS request header named *Content-type* as stated below.

TABLE 53 Content-type HTTPS Request Header

Request Header Name	Request Header Value	Request Data Format
Content-type	Application/xml	XML
Content-type	Application/json	JSON

Error Handling

Error Handling

SdnException is the exception for all REST errors. This exception contains an integer errorCode and a string errorMessage.

The exception will be a string of the following format:

```
SdnException [errorCode=<int>, errorMsg=<string>]
```

REST operations are HTTPS requests, so the execution of an operation returns an HTTPS status code. For successful operations the status code will usually be 200 (OK) or 204 (No Content).

In the case of an error, depending on the reason for failure, any of the HTTPS status codes may be returned. But in the case of an API error, the HTTPS status code will be 500 (Internal Server Error). More details on the server error can be obtained from the SdnException embedded in the HTTPS response.

Stated below are the SdnException error codes.

TABLE 54 Error Codes

Error Code	Description
1000	Internal server error.
1001	Database exception
1002	Exception while getting Event Profile Info from database.
1003	Exception while creating Event in database.
1004	Name of profile is null or empty.
1005	Exception while getting Profiles from database.
1006	Failed to convert string to Errors object.
1008	Device SNMP communication failed, SNMP error is: {0}
2001	Invalid input parameters-granularity cannot be greater than duration.
2002	Size of utilizations in database are not equal for populating or aggregating data.
2003	User name cannot be empty. User name should not exceed 128 characters, valid characters alphanumeric, space, -, ., and ~
2004	Password cannot be empty, Password length should be at least 8 characters and should not exceed 75 characters.
2005	Invalid user name or password.
2006	User does not exist.
2007	Password encryption error -{0}
2008	User sessions have reached maximum limit.
2009	Invalid token.
2010	User does not have sufficient privileges.
2011	Root user account cannot be deleted.
2012	Duplicate user, the specified user already exists.
2013	Root user account cannot be updated.
2014	Invalid input parameters -start time is greater than end time.
2015	Invalid Request, Large flow Id is null or empty.

TABLE 54 Error Codes (continued)

Error Code	Description
2016	Invalid Request, Pro field is null or empty.
2017	Traffic flow details is null.
2018	Profile details is null.
2019	Controller URL is null or empty.
2020	Controller user name is null or empty.
2021	Controller password is null or empty.
2022	Controller already exists.
2023	Controller does not exist.
2024	Controller URL is invalid -{0}
2025	Invalid SNMP profile name, it is null or empty.
2026	Invalid SNMP version.
2027	Duplicate SNMP profile, the specified SNMP profile already exists.
2028	Invalid auth password, it is null or empty.
2029	Invalid priv password, it is null or empty.
2030	SNMP profile {0} does not exist.
2031	SDN Controller settings must be configured to get available devices.
2032	Invalid no of SNMP profiles, two are needed for swapping.
2033	Invalid device IP, IP address is null or empty.
2034	Device {0} is already managed.
2035	The re are no SNMP profiles to manage device, please configure at least one SNMP profile.
2036	Input port list for device {0} is empty, user has to specify at least one device port.
2037	SDN Controller {0} is not reachable, please check and update SDN Controller settings.
2038	Device {0} is not managed.
2039	Device {0} cannot be managed because it is missing in the controller. Please delete device.
2040	Cannot register on device, there are already 4 collectors on device.
2041	Device{0} cannot be managed because collector is missing on device. Please delete device and add again.
2042	Collector IP is missing, please configure collector.
2043	Username cannot be empty.
2044	Password cannot be empty.
2500	DB: Failed to insert the Large flow for key {0} checksum {1}
2501	DB: Failed to update the Large flow for key {0} checksum {1}
2502	DB: Failed to move the Large flow for key {0} to completion.
2503	DB: Failed to insert the Traffic flow details for checksum {0}
2504	DB: Failed to update Time series 1 sec data for checksum {0}
2505	DB: Failed to fetch the active Large flow details.
2506	DB: Failed to purge entries for {0} table.
3001	The action {0} for flow {1} configuration is not supported.
4001	Could not retrieve nodes from BVC.
4002	Nodes from BVCare null or empty.
5001	Profile Name cannot exceed more than 128 characters.
5002	Invalid observation interval value. Minimum observation interval valid is 5.

TABLE 54 Error Codes (continued)

Error Code	Description
5003	Invalid threshold value.
5004	Invalid profile type.
5005	Invalid profile status.
5006	Invalid username.
5007	Invalid mitigation action.
5008	Priority already set. Please use different priority.
5009	Network Attributes cannot be empty.
5010	Profile cannot have same network attribute twice.
5011	Destination MAC cannot be empty.
5012	Source MAC cannot be empty.
5013	Source VLAN cannot be empty.
5015	IPv4 source address cannot be empty.
5014	VLAN Priority cannot be empty.
5016	IPv4 destination address cannot be empty.
5017	IPv6 source address cannot be empty.
5018	IPv6 destination address cannot be empty.
5019	IP Protocol cannot be empty.
5020	DSCP cannot be empty.
5021	TCP source or dest port is invalid.
5022	TCP destination port cannot be empty.
5023	UDP source or dest port is invalid.
5024	UDP destination port cannot be empty.
5025	TCP Flags cannot be empty.
5026	IP Fragment cannot be empty.
5027	Invalid MAC format. Please provide the MAC address in format 11:22:33:44:55:66
5028	Invalid INVLAN string.
5029	Invalid VLANID. Valid Range: 1 to 4095.
5030	Invalid VLAN priority.
5031	Invalid VLAN priority. Valid Range: 0-7
5032	VLANID has to be selected for setting VLAN priority.
5033	Invalid IPv4 source address. Please enter valid IP address in CIDR format.
5034	Invalid IPv4 destination address. Please enter valid IP address in CIDR format.
5035	Invalid IPv6 source address. Please enter valid IP address in CIDR format.
5036	Invalid IPv6 destination address. Please enter valid IP address in CIDR format.
5037	Invalid IP Protocol. Valid values: TCP / UDP / ICMP
5038	Invalid DSCP. DSCP should be an integer value. Valid Range: 0-63
5039	IPv6 address cannot be selected when you want to set IPv4 source or destination.
5040	IPv4 address cannot be selected when you want to set IPv6 source or destination.
5041	The IP Protocol must be set to TCP when TCP Port is selected.
5042	UDP Port cannot be selected when TCP port is selected.
5043	The IP Protocol must be set to UDP when UDP Port is selected.

TABLE 54 Error Codes (continued)

Error Code	Description
5044	TCP Port cannot be selected when UDP port is selected.
5045	The IP Protocol must be set to TCP when TCP Flag is selected.
5046	Invalid TCP flag. Valid values: URG / ACK / PSH / RST / SYN / FIN
5047	Only yes / No is allowed for IP fragment option.
5050	When redirect action selected, please provide the redirect node and port.
5051	Invalid Redirect node or port. Valid Format Node: 10.45.67.4; Port: 1, 2. Ingress port cannot be same as Mirror port.
5052	The profile name \"{0}\" from query parameter and profile name \"{1}\" from profile object does not match.
5053	Failed to search user name for given userId.
5054	Failed to insert profile {0}
5055	Failed to insert mitigation association {0} {1}
5056	Failed to insert profile attribute association {0} {1} {2}
5057	Failed to delete the profile {0}
5058	Failed to update the profile {0}
5059	Failed to delete profile mitigation association for profile {0}
5060	Failed to delete profile attribute association for profile {0}
5061	The node with IP: {0} is not discovered in BVC.
5062	Failed to create flow request. ProfileName: {0} Action: {1} FlowKey: {2}
5063	Failed to Program Flow for {0} on BVC for node: {1} for destination {2}
5064	Failed to create meter for {0} on BVC for node: {1} for VLAN: {2}
5065	Failed to Program Flow for {0} on BVC for node: {1} for VLAN: {2}
5066	Failed to delete meter for {0} on BVC for node: {1} for VLAN: {2}
5067	Failed to get configured nodes for programming flow: {0}
5068	Failed to create meter for {0} on BVC for node: {1}
5069	Failed to validate IP address {0} during the Large flow detection.
5070	Please select NONE as an action when any of the detection only parameters are selected.
5071	Ingress port is required for METER action.
5072	Please provide valid Ingress node and port for METER action.
5073	Only one Ingress node and port are allowed for METER / MIRROR action.
5074	VLANID is mandatory network attribute for metering the traffic.
5075	Invalid Rate limit value for meter.
5076	Invalid DSCP Rate limit value for meter.
5077	DSCP Remark rate limit should be less than Drop rate limit.
5078	Invalid Profile Name. Only Alphanumeric, Space and - / . / _ / ~ are allowed.
5079	Profile with name \"{0}\" already exists.
5080	VLAN ID is mandatory network attribute when you select MIRROR as an action.
5081	There are too many wildcard attributes for the profile {0}. Maximum is 2.
5082	Action is in valid for the profile {0} with wildcard attribute. Only NONE action is supported for a profile with wildcard attribute.
5083	Maximum of 50 profiles is allowed. Please remove one or more profiles before adding new profile.
5084	Mirror action is invalid for default profiles.
5085	Ingress and Mirror port a re-required for MIRROR action.
6001	Error while initializing the PBE security key.

TABLE 54 Error Codes (continued)

Error Code	Description
6002	Error while encrypting the text.
6003	Error while decrypting the text.
7001	Failed to retrieve the sFlow settings.
7002	Failed to retrieve the Network interfaces from Brocade Flow Optimizer host.
7003	Failed to inserts Flow settings.
7004	Failed to updates Flow settings.
7005	Invalid in-band or out-of-band address.
8001	Failed to send mail to the recipients.
8002	Invalid parameters for email configurations.

URI Return Behavior

This section explains in detail the error a user can expect when a URI fails.

Parsing of the URI follows a pattern and returns an error in the order stated below.

1. The URI is checked for correctness. If the URI is not valid (such as in the case of a misspelling), then the resource will not be found, and user should get an HTTPS status code of 404 (Not Found).
2. If the above PATH PARAM check succeeds, then the QUERY PARAM values (i.e. the query parameters) are checked for correctness. If the query parameters are invalid, the REST operation should fail with the HTTPS status code of 500 (Internal Server Error) and an SdnException. The value of the error code will depend on the exact error.
3. If the above QUERY PARAM check succeeds, then syntactically, the URI is correct. The URI is parsed from left to right. If any resource corresponding to the PATH PARAM is not present starting from the left, then the appropriate `Does Not Exist` or `Not Found` error is reported.

   Example

   ```
   <BASE-URI>/profiles/<profile Name>
   ```

   In the URI above, if the resourcegroup specified by path parameter rgkey does not exist, then the user should receive an appropriate error code. If rgkey is a valid key, then the existence of fcskey is checked, and so on.
4. The Not Found error code is returned if you are looking for a specific object.
5. In the case of a collection like fcswitches, the response will be either a populated or empty list. The return of an empty list, itself, suggests to the user that there are such instances within the requested collection.

Third-Party Bro Integration

Third-Party Bro Integration Overview

Bro IDS is open-source software capable of analyzing network traffic and raising events when a particular network activity is spotted (e.g. TCP connection has been established). The primary purpose of BFO-Bro integration is to give Bro users a way of calling the BFO REST API directly from a Bro script.

Through third-party Bro integration, Bro IDS users can use the Brocade Flow Optimizer (BFO) REST API to trigger network-level mitigation actions in a response to security threats. Such events come predefined within the Bro, and are usually handled by the user (e.g. enterprise security team) in an event handler coded in Bro scripting language.

Bro Installation

Bro Installation Directory

Bro stores its files in a single location.

By default it uses the following directories:

- `/opt/bro` if installing from a binary package.
- `/usr/local/bro` if installing from source.

The user is not limited to these directories, and may choose other directories during the installation process.

The Bro installation directory has a following top-level structure:

```
.
├── bin
├── etc
├── include
├── lib
├── logs
├── share
└── spool
```

Bro uses shared subdirectories to store its modules, where the BFO plugin should be installed.

Bro Plugin Installation

The user is provided with a tarball to perform the Bro plugin installation.

The tarball contains the following directory structure:

```
brocade
├── bfo.bro
└── bfo.py
```

The tarball must be extracted into the `share/bro` subdirectory of the Bro installation directory. Provided Bro has been installed in `/opt/bro`, the following command is used to untar the plugin:

```
tar –xf bfo-bro-plugin.tar.gz –C /opt/bro/share/bro
```

NOTE

If Bro has been installed in a different directory (e.g. `/usr/local/bro`), the path in the command above must be changed accordingly.

The file structure presented below shows in detail where the plugin should be placed after extracting the tarball:

```
.
├── bin
├── etc
├── include
├── lib
├── logs
├── share
│   ├── bro
│   │   ├── base
│   │   ├── broctl
│   │   ├── broxygen
│   │   ├── brocade
│   │   │   ├── bfo.bro
│   │   │   └── bfo.py
│   │   ├── policy
│   │   └── site
│   ├── broctl
│   └── man
└── spool
```

Perform the following steps to verify the installation was successful:

1. NOTE

   If the file `check.bro` already exists containing the command line `@load brocade/bfo` then skip to the next step.

   Add the module load directive `@load brocade/bfo` to a new file named `check.bro`.
   This Bro script verifies the plugin installation.

2. Run the following command from the Bro installation directory:

   ```
   ./bin/bro –a check.bro
   ```

   If the plugin is installed correctly, this command will execute without errors.

Bro Plugin Configuration

The plugin installation directory must be explicitly configured in order to execute an external Python script.

The user must set the *plugin_dir* constant to a string representing the path where the plugin is installed by appending two lines to the `site/local.bro` file, as shown below.

> **NOTE**
> The example assumes Bro is installed in `/opt/bro`. If other installation directory has been used, the file path must be changed accordingly.

File Path	File Content

`<install_dir>/share/bro/site/local.bro`	`(…)`
	`@load brocade/bfo`
	`redef BFO::plugin_dir = "/opt/bro/share/brocade"`

Bro Plugin Examples

This topic covers examples of third-party Bro integration through the Bro plugin used through a Bro scripts.

Examples of third-party Bro integration through Bro plugin scripts include the following:

- Drop Action
- Meter Action
- Mirror Action
- Redirect Action

Drop Action Example

This topic covers how third-party Bro integration through the Bro plugin is used through the Bro drop action script.

Example of Drop Action Script

The following example demonstrates the Bro plugin drop action script.

File: `test_drop.bro`

```
@load brocade/bfo

export {
  redef BFO::hostname = "itcs-vm-odl-01";
  redef BFO::tcp_port = 8089;
  redef BFO::username = “Administrator”;
  redef BFO::password = “password”;
}

event bro_init() {
  local match = BFO::Match(
    $l2 = BFO::L2Match(
      $src_mac = "AA:BB:CC:DD:EE:FF"
    ),
    $l3 = BFO::L3Match(
      $src_ip = 192.168.1.1/32,
      $dst_ip = 192.168.1.2/32,
      $protocol = icmp
    )
  );

  local drop_mitigation = BFO::Mitigation(
    $action = BFO::DropAction,
    $drop_params = BFO::DropParams(
      [$node="188.184.19.238", $ingress_port="1/1/1"],
      [$node="188.184.19.239", $ingress_port="1/1/2"]
    ),
    $hard_timeout = 15
  );

  when (local result = BFO::mitigate(match, drop_mitigation)) {
    if (result$success) {
      print "DROP Flow created successfully!";

      if (result ?$ flow_id) {
        print fmt("Flow ID: %s", result$flow_id);
      }
    } else {
      print fmt("Failed DROP mitigation. Error: %s", result$error);
    }
  }
}
```

Meter Action Example

This topic covers how third-party Bro integration through the Bro plugin is used through the Bro meter action script.

Example of Meter Action Script

The following example demonstrates the Bro plugin meter action script.

NOTE

The example below shows how a tagged VLAN match can be configured using the `$match_vlan` and `$vlan_tag` properties. In order to specify an untagged VLAN match, `$match_vlan` has to be set to `false` and `$vlan_tag` must be skipped, as shown below.

```
local match = BFO::Match(
    $l2 = BFO::L2Match(
      $match_vlan = F,
      (...)
    ),
    (...)
)
```

Please note however that untagged VLAN match is supported only for DROP and REDIRECT actions.

File: `test_meter.bro`

```
@load brocade/bfo

export {
  redef BFO::hostname = "itcs-vm-odl-01";
  redef BFO::tcp_port = 8089;
  redef BFO::username = "Administrator";
  redef BFO::password = "password";
}

event bro_init() {
  local match = BFO::Match(
    $l2 = BFO::L2Match(
      $match_vlan = T,
      $vlan_tag = 101
    ),
    $l3 = BFO::L3Match(
      $src_ip = 192.168.1.1/32,
      $dst_ip = 192.168.1.2/32,
      $protocol = tcp
    )
  );

  local meter_mitigation = BFO::Mitigation(
    $action = BFO::MeterAction,
    $meter_params = BFO::MeterParams(
      $node = "188.184.19.238",
      $ingress_port = "1/2/1",
      $rate_limit_mbps = 550,
      $dscp_remark = T,
      $dscp_rate_limit_mbps = 200,
      $dscp_precedence = 5
    )
    $hard_timeout = 15
  );

  when (local result = BFO::mitigate(match, meter_mitigation)) {
    if (result$success) {
      print "METER Flow created successfully!";

      if (result ?$ flow_id) {
        print fmt("Flow ID: %s", result$flow_id);
      }
    } else {
      print fmt("Failed METER mitigation. Error: %s", result$error);
    }
  }
}
```

Mirror Action Example

This topic covers how third-party Bro integration through the Bro plugin is used through the Bro mirror action script.

Example of Mirror Action Script

The following example demonstrates the Bro plugin mirror action script.

File: `test_mirror.bro`

```
@load brocade/bfo

export {
  redef BFO::hostname = "itcs-vm-odl-01";
  redef BFO::tcp_port = 8089;
  redef BFO::username = "Administrator";
  redef BFO::password = "password";
}

event bro_init() {
  local match = BFO::Match(
    $l2 = BFO::L2Match(
      $match_vlan = T,
      $vlan_tag = 101
    ),
    $l3 = BFO::L3Match(
      $protocol = icmp
    )
  );

  local mirror_mitigation = BFO::Mitigation(
    $action = BFO::MirrorAction,
    $mirror_params = BFO::MirrorParams(
      $node = "188.184.19.238",
      $ingress_port = "1/2/1",
      $mirror_port = "1/1/11"
    ),
    $hard_timeout = 15
  );

  when (local result = BFO::mitigate(match, mirror_mitigation)) {
    if (result$success) {
      print "MIRROR Flow created successfully!";

      if (result ?$ flow_id) {
        print fmt("Flow ID: %s", result$flow_id);
      }
    } else {
      print fmt("Failed MIRROR mitigation. Error: %s", result$error);
    }
  }
}
```

Redirect Action Example

This topic covers how third-party Bro integration through the Bro plugin is used through the Bro redirect action script.

Example of Redirect Action Script

The following example demonstrates the Bro plugin redirect action script.

File: `test_redirect.bro`

```
@load brocade/bfo

export {
  redef BFO::hostname = "itcs-vm-odl-01";
  redef BFO::tcp_port = 8089;
  redef BFO::username = "Administrator";
  redef BFO::password = "password";
}

event bro_init() {
  local match = BFO::Match(
    $l3 = BFO::L3Match(
      $src_ip = 192.168.1.1/32,
      $dst_ip = 192.168.1.2/32,
      $protocol = tcp
    ),
    $l4 = BFO::L4Match(
      $src_port = 8080
    )
  );

  local redirect_mitigation = BFO::Mitigation(
    $action = BFO::RedirectAction,
    $redirect_params = BFO::RedirectParams(
      [
        $node = "188.184.19.238",
        $egress_ports = vector("1/1/1", "1/1/11")
      ],
      [
        $node = "188.184.19.239",
        $egress_ports = vector("1/1/1"),
        $vlan_action = BFO::VlanPush,
        $vlan_id = 5,
        $dst_mac = "AA:C2:F1:2E:8F:0A"
      ]
    )
  );

  when (local result = BFO::mitigate(match, redirect_mitigation)) {
    if (result$success) {
      print "REDIRECT Flow created successfully!";

      if (result ?$ flow_id) {
        print fmt("Flow ID: %s", result$flow_id);
      }
    } else {
      print fmt("Failed REDIRECT mitigation. Error: %s", result$error);
    }
  }
}
```

Bro Plugin Reference

This topic covers the Bro plugin reference.

```
mitigate_connection: function(c: connection, action: Mitigation): FlowCreateResult;
mitigate: function(match: Match, mitigation: Mitigation): FlowCreateResult;
unmitigate: function(flow_id: string): FlowDeleteResult;

  type FlowCreateResult: record {
    flow_id: string &optional;
    success: bool;
    error: string &optional;
  };

  type FlowDeleteResult: record {
    success: bool;
    error: string &optional;
  };

  type Match: record {
    l2: L2Match &optional;
    l3: L3Match &optional;
    l4: L4Match &optional;
  };

  type L2Match: record {
    src_mac: string &optional;
    dst_mac: string &optional;
    match_vlan: bool &optional;
    ether_type: count &optional;
    vlan_tag: count &optional;
    vlan_priority: count &optional;
  };

  type L3Match: record {
    src_ip: subnet &optional;
    dst_ip: subnet &optional;
    dscp: count &optional;
    ip_type: IpType &default=IPV4;
    protocol: transport_proto &optional;
  };

  type L4Match: record {
    src_port: count &optional;
    dst_port: count &optional;
  };

  type IpType: enum {
    IPV4, IPV6
  };

  type Action: enum {
    DropAction, MeterAction, MirrorAction, RedirectAction
  };

  type VlanAction: enum {
    VlanIgnore, VlanPush, VlanPop, VlanModify
  };

  type Mitigation: record {
    action: Action &optional;
    hard_timeout: count &optional &default=0;
    mirror_params: MirrorParams &optional;
    meter_params: MeterParams &optional;
    redirect_params: RedirectParams &optional;
    drop_params: DropParams &optional;
  };


  type MirrorParams: record {
    node: string;
```

```
  ingress_port: string;
  mirror_port: string;
};

type MeterParams: record {
  rate_limit_mbps: count;
  dscp_remark: bool;
  dscp_rate_limit_mbps: count;
  dscp_precedence: count;
  node: string;
  ingress_port: string;
};

type RedirectParams: vector of RedirectRule;

type DropRule: record {
  node: string;
  ingress_port: string;
};

type DropParams: vector of DropRule;
```