Tag a device through the REST API

Published: 2020-03-21

Tags can help you classify devices that share a common characteristic from among potentially hundreds of discovered devices on your network.

You might want to tag devices by their role on your network, such as the devices that make up your development and production servers. For example, if you are running multiple AWS instances in your environment, it is critical to size them for their workload. An undersized instance can result in poor performance; an oversized instance is needlessly costly. If you tag your AWS instances, you can easily set up device groups by instance size, and then create a dashboard to monitor usage and performance metrics.

In this guide, you will learn how create a tag, find the device you want to tag, and then add the tag to the device. An example script is provided at the end, which adds a given device tag to all IP addresses read from a CSV file.

Before you begin:

- You must log in to the ExtraHop system with an account that has unlimited privileges to generate an API key.
- You must have a valid API key to make changes through the REST API and complete the procedures below. (See Generate an API key.)
- Familiarize yourself with the ExtraHop REST API Guide to learn how to navigate the ExtraHop REST API Explorer.

Create a tag

If you already have a tag on the system, you can skip this step. The example script at the bottom of this guide will check for a tag and create a new tag only if necessary.

1. In a browser, navigate to the REST API Explorer.

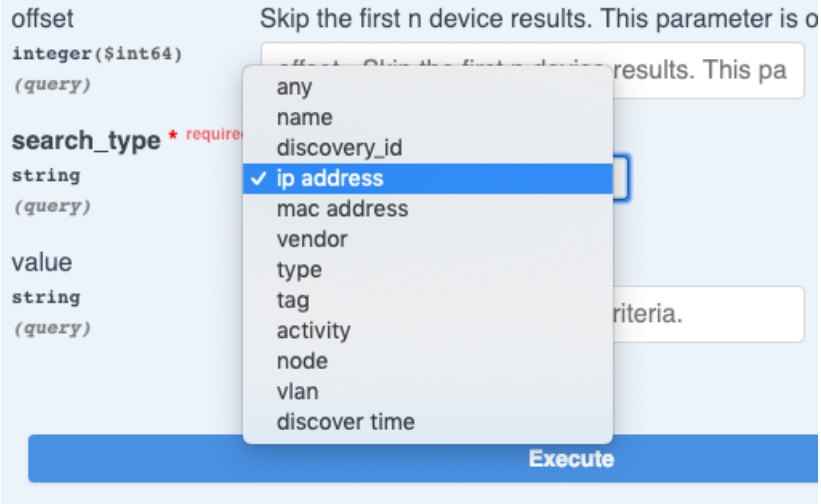
   The URL is the hostname or IP address of your ExtraHop Discover or Command appliance, followed by `/api/v1/explore/`. For example, if your hostname is seattle-eda, the URL is `https://seattle-eda/api/v1/explore/`.
2. Click **Enter API Key** and then paste or type your API key into the **API Key** field.
3. Click **Authorize** and then click **Close**.
4. Click **Tag** and then click **POST/tags**.
5. Click **Try it out**.
   The JSON schema is automatically added to the body parameter text box.
6. In the `name` field, replace `string` with the new tag name.
7. Click **Send Request** to create the tag.

Retrieve devices that match your criteria

In this step you will search for the devices that you want to tag and note the device ID. You must have the device ID before you can tag devices.

1. Scroll up the page and click **Device** to display device operations.
2. Click **GET /devices** to display GET parameters.
3. Click **Try it out**.
4. Click the **search_type** drop-down and select a search filter, then enter the corresponding information in the **value** field.

   For example, select **ip address** and then type the IP address for a specific device in the **value** field.



5. Click **Send Request**.
   In the Server response section, the Response body displays information about each device that matches your search criteria, including the device ID.

Assign the tag to a device or device group

In this step, you will assign a tag to a device by the device ID you found in the previous step.

1. Scroll down the page and click **Tag** to display tag operations.
2. Click **POST /tags/{id}/devices/{child-id}**.
3. Click **Try it out**.
4. In the child-id field, type the ID of the device you want to tag.
5. In the id field, type the ID of the tag you want to assign.
6. Click **Send Request** to assign the tag to the device.

Tip: After you click **Send Request**, you can click the tabs to view scripts for the operation in Curl, Python 2.7, or Ruby.

Python script example

The following Python script reads IP addresses from a csv file, creates a device tag, and then assigns the tag to all devices with the specified IP addresses. The script creates a new tag only if the specified tag does not already exist. The script includes the following configuration variables that you must replace with information from your environment:

- **HOST:** The IP address or hostname of the Discover appliance
- **APIKEY:** The API key
- **TAG:** The name of the tag
- **DEVICE_LIST:** The file that contains the list of IP addresses

```
#!/usr/bin/python3

import http.client
import json
import csv
```

```
HOST = 'extrahop.example.com'
APIKEY = '123456789abcdefghijklmnop'
TAG = 'new-tag'
IP_LIST = 'ip_list.csv'

headers = {'Content-Type': 'application/json',
           'Accept': 'application/json',
           'Authorization': 'ExtraHop apikey=%s' % APIKEY}

#### Returns the ID of a tag ####
def get_tag_id(tag):
    tag_id = ''
    conn = http.client.HTTPSConnection(HOST, context=context)
    conn.request('GET', '/api/v1/tags', headers=headers)
    resp = conn.getresponse()
    tags = json.loads(resp.read())
    tag_index = 0
    while tag_id == '' and tag_index < len(tags):
        if tags[tag_index]['name'] == tag:
            tag_id = tags[tag_index]['id']
        tag_index += 1
    return tag_id

#### If the tag does not already exist, create it ####
tag_id = get_tag_id(TAG)
if tag_id != '':
    print(TAG + ' already exists')
else:
    print('Creating ' + TAG + ' tag')
    body = {"name": TAG}
    conn = http.client.HTTPSConnection(HOST, context=context)
    conn.request('POST', '/api/v1/tags', headers=headers,
 body=json.dumps(body))
    resp = conn.getresponse()
    if resp.status == 201:
        print(TAG + ' tag created successfully!')
    else:
        print('Error: could not create ' + TAG + ' tag')
        sys.exit()
    tag_id = get_tag_id(TAG)

#### Get IPs from IP_LIST file ####
device_ips = []
with open(IP_LIST, 'rt', encoding='ascii') as f:
    reader = csv.reader(f)
    for row in reader:
        for item in row:
            device_ips.append(item)

#### Get IDs for devices with the specified IPs ####
device_ids = []
tagged_devices = []
for ip in device_ips:
    conn = http.client.HTTPSConnection(HOST, context=context)
    url = '/api/v1/devices?limit=100&search_type=ip%20address&value=' + ip
    conn.request('GET', url, headers=headers)
    resp = conn.getresponse()
    try:
        devices = json.loads(resp.read())
    except:
        continue
    for device in devices:
        device_ids.append(device['id'])
        tagged_devices.append(device['display_name'])
```

```
#### Add tag to devices ####
body = {"assign": device_ids}
conn = http.client.HTTPSConnection(HOST, context=context)
url = '/api/v1/tags/%d/devices' % tag_id
conn.request('POST', url, headers=headers, body=json.dumps(body))
resp = conn.getresponse()
if resp.status == 204:
    print('Success! Assigned %s to the following devices:' % TAG)
    for name in tagged_devices:
        print('    ' + name)
    print('Assigned %s to %d devices' % (TAG, len(device_ids)))
else:
    print('Error! Failed to tag specified devices')
```

Note: If the script returns an error message that the SSL certificate verification failed, make sure that a trusted certificate has been added to your appliance. Alternatively, you can add the context option and send the request over TLSv1.2 to bypass certificate verification. However, this method is not secure and is not recommended. The following code creates an HTTP connection over TLSv1.2:

```
conn = httplib.HTTPSConnection(HOST,
 context=ssl.SSLContext(ssl.PROTOCOL_TLSv1_2))
```