Extract the device list through the REST API

The ExtraHop REST API enables you to extract the list of devices discovered by the ExtraHop system. By extracting the list with a REST API script, you can export the list in a format that can be read by third-party applications, such as a configuration management database (CMDB). In this topic, we show methods for extracting a list through both the cURL command and a Python script.

Before you begin

* You must log in to the ExtraHop system with an account that has full write privileges to generate an API key.
* You must have a valid API key to retrieve devices through the REST API and complete the procedures below. (See Generate an API key ⤤.)

Retrieve the device list with the cURL command

The device list includes all device metadata, such as MAC addresses and device IDs. However, you can filter the list of devices with a JSON parser to extract the specific information you want to export. In this example, the device list is retrieved and then filtered with the jq parser to only extract the display name of each device.

Before you begin

* The cURL tool must be installed on your machine.
* The jq parser must be installed on your machine. For more information, see https://stedolan.github.io/jq/ ⤤.

Open a terminal application and run the following command, where `YOUR_KEY` is the API for your user account, `HOSTNAME` is the hostname of your ExtraHop system, and `MAX_DEVICES` is a number large enough to be more than the total number of devices discovered by your system:

```
curl -s -X GET --header "Accept: application/json" --header
  "Authorization: ExtraHop apikey=YOUR_KEY"  "https://HOSTNAME/api/v1/
devices?active_from=1&active_until=0&limit=MAX_DEVICES" | jq -r '.[]
  | .display_name'
```

📄 **Note:** If the command returns no results, make sure that a trusted certificate has been added to your ExtraHop system ⤤. Alternatively, you can add the `--insecure` option to retrieve the device list from an ExtraHop system without a trusted certificate; however, this method is not secure and is not recommended.

💡 **Tip:** You can append the `select(.analysis == "LEVEL")` option to filter results by analysis level. For example, the following command limits the results to include only devices that are selected for advanced analysis:

```
curl -s -X GET --header "Accept: application/json" --header
  "Authorization: ExtraHop apikey=YOUR_KEY"  "https://HOSTNAME/
api/v1/devices?active_from=1&active_until=0&limit=10000000000"
  | jq -r '.[] | select(.analysis == "advanced") | .display_name'
```

💡 **Tip:** You can append the `select(.critical == BOOLEAN)` option to filter results by the critical field. For example, the following command limits the results to include only devices that are identified as critical by the ExtraHop system:

```
curl -s -X GET --header "Accept: application/json" --header
  "Authorization: ExtraHop apikey=YOUR_KEY"  "https://HOSTNAME/
```

```
     api/v1/devices?active_from=1&active_until=0&limit=10000000000"
       | jq -r '.[] | select(.critical == true) | .display_name'
```

Tip: You can append the `select(.cloud_instance_name != null)` option to filter results by the cloud instance name field. For example, the following command limits the results to include only devices with a cloud instance name:

```
curl -s -X GET --header "Accept: application/json" --header
  "Authorization: ExtraHop apikey=YOUR_KEY"  "https://HOSTNAME/
api/v1/devices?active_from=1&active_until=0&limit=10000000000"
   | jq -r '.[] | select(.cloud_instance_name != null)
   | .cloud_instance_name'
```

Python script example

The following example Python script extracts the device list, including all device metadata, and writes the list to a CSV file in the same directory as the script. The script includes the following configuration variables that you must replace with information from your environment:

• **HOST:** The IP address or hostname of the ExtraHop system

• **APIKEY:** The API key

• **FILENAME:** The file that output will be written to

• **LIMIT:** The maximum number of devices to retrieve with each GET request

• **SAVEL2:** Retrieves L2 parent devices. This variable is valid only if you have enabled the ExtraHop system to discover devices by IP address.

• **ADVANCED_ONLY:** Retrieves only devices that are currently under advanced analysis

• **CRITICAL_ONLY:** Retrieves only devices that have been identified as critical by the ExtraHop system

```python
#!/usr/bin/python3

import http.client
import json
import csv
import datetime
import ssl
import sys

HOST = 'extrahop.example.com'
APIKEY = '123456789abcdefghijklmnop'
FILENAME = 'devices.csv'
LIMIT = 1000
SAVEL2 = False
ADVANCED_ONLY = False
CRITICAL_ONLY = False

headers = {}
headers['Accept'] = 'application/json'
headers['Authorization'] = 'ExtraHop apikey='+APIKEY

def getDevices(offset):
    conn = http.client.HTTPSConnection(HOST)
    conn.request('GET', '/api/v1/devices?limit=%d&offset=
%d&search_type=any'%(LIMIT,offset), headers=headers)
    resp = conn.getresponse()
    if resp.status == 200:
        devices = json.loads(resp.read())
        conn.close()
```

```
            return devices
     else:
         print("Error retrieving Device list")
         print(resp.status, resp.reason)
         resp.read()
         dTable = None
         conn.close()
         sys.exit()

continue_search = True
offset = 0
dTable = []
while (continue_search):
     new_devices = getDevices(offset)
     offset += LIMIT
     dTable += new_devices
     if (len(new_devices) > 0):
         continue_search = True
     else:
         continue_search = False

if (dTable != None):
     print (" - Saving %d devices in CSV file" % len(dTable))
     with open(FILENAME, 'w') as csvfile:
         csvwriter = csv.writer(csvfile,dialect='excel')
         csvwriter.writerow(list(dTable[0].keys()))
         w = 0
         s = 0
         for d in dTable:
             if ADVANCED_ONLY == False or (ADVANCED_ONLY == True and
 d['analysis'] == 'advanced'):
                 if CRITICAL_ONLY == False or (CRITICAL_ONLY == True and
 d['critical'] == True):
                     if d['is_l3'] | SAVEL2:
                         w += 1
                         d['mod_time'] =
 datetime.datetime.fromtimestamp(d['mod_time']/1000.0)
                         d['user_mod_time'] =
 datetime.datetime.fromtimestamp(d['user_mod_time']/1000.0)
                         d['discover_time'] =
 datetime.datetime.fromtimestamp(d['discover_time']/1000.0)
                         csvwriter.writerow(list(d.values()))
                     else:
                         s += 1
                 else:
                     s += 1
             else:
                 s += 1
         print(" - Wrote %d devices, skipped %d devices " % (w,s))
```

> **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 ExtraHop system ⬈. 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))
```