Upgrade ExtraHop firmware through the REST API

Published: 2020-04-06

You can automate upgrades to the firmware on your ExtraHop appliances through the ExtraHop REST API. This guide includes methods for both the cURL command and a Python script.

While the firmware upgrade process is similar across all ExtraHop appliances, some appliances have additional considerations or steps that you must address before you install the firmware in your environment. If you need assistance with your upgrade, contact ExtraHop Support.

All appliances must meet the following requirements:

- The firmware version must be compatible with your appliance model.
- The firmware version on your appliance must be supported by the upgrade version.
- Command appliances must be running firmware that is greater than or equal to their connected appliances.
- Discover appliances must be running firmware that is greater than or equal to Explore and Trace appliances.

If your deployment only includes a Discover appliance, proceed to the cURL or Python upgrade instructions.

If your deployment includes additional appliance types, you must address the following dependencies before proceeding with the upgrade instructions.

If your deployment includes...	Pre-upgrade tasks	Upgrade order
Command appliances	Reserve a maintenance window of an hour for Command appliances managing 50,000 devices or more.	• Command appliance • Discover appliances • All Explore appliances (master nodes, then data nodes) • Trace appliances
Explore appliances	See Upgrading Explore appliances.	
Trace appliances	None	

Upgrade ExtraHop firmware with cURL

You can upgrade the firmware on an ExtraHop appliance through the cURL command.

Before you begin

- The cURL tool must be installed on your machine.
- The appliance firmware .tar file must be downloaded on your machine.

1. Open a terminal application.
2. Upload the firmware file.

   Run the following command, where `YOUR_KEY` is the API key for your user account, `HOSTNAME` is the hostname of your ExtraHop appliance, and `FILE_PATH` is the relative file path of the appliance firmware .tar file:

   ```
   curl -X POST https://HOSTNAME/api/v1/extrahop/firmware --data-binary
    @FILE_PATH -H "Content-Type:application/vnd.extrahop.firmware" -H
    "Authorization: ExtraHop apikey=YOUR_KEY"
   ```

3. Upgrade the appliance firmware.

Run the following command, where `YOUR_KEY` is the API key for your user account, and `HOSTNAME` is the hostname of your ExtraHop appliance:

```
curl -X POST "https://HOST/api/v1/extrahop/firmware/latest/upgrade" -H
 "accept: application/json" -H "Authorization: ExtraHop apikey=YOUR_KEY"
 -H "Content-Type: application/json" -d "{ \"restart_after\": true}"
```

4. Verify that the appliance has been successfully upgraded.

   Run the following command, where `YOUR_KEY` is the API key for your user account, and `HOSTNAME` is the hostname of your ExtraHop appliance:

   ```
   curl -X GET https://HOST/api/v1/extrahop -H "Authorization: ExtraHop
    apikey=YOUR_KEY"
   ```

   The command displays an object that contains information about the firmware currently running on the appliance. Verify that the version field matches the firmware version you are upgrading to. If the above command does not display the correct version number, wait a few minutes, and then try again. It might take several minutes for the upgrade to complete.

Python script example

The following example Python script upgrades multiple appliances by reading the appliance URLs, API keys, and firmware file paths from a CSV file.

Each row of the CSV file must contain the following columns in the specified order:

Appliance HTTPS URL	API key	Firmware file path

The script includes the following configuration variable that you must replace with information from your environment:

- **APPLIANCE_LIST:** The relative file path of the CSV file.

Note: The script does not automatically disable record ingest for Explore appliances. You must manually disable record ingest before running the script for an Explore appliance.

```
#!/usr/bin/python3

import os
import requests
import csv

APPLIANCE_LIST = 'appliances.csv'

# Retrieve URLs, API keys, and firmware file paths
appliances = []
with open(APPLIANCE_LIST, 'rt', encoding='ascii') as f:
    reader = csv.reader(f)
    for row in reader:
        appliance = {
                      'host': row[0],
                      'api_key': row[1],
                      'firmware': row[2]
                    }
        appliances.append(appliance)

# Upload firmware to appliance
def uploadFirmware(host, api_key, firmware):
    headers = {
```

```
        'Authorization': 'ExtraHop apikey=%s' % api_key,
        'Content-Type': 'application/vnd.extrahop.firmware'
    }
    url = host + 'api/v1/extrahop/firmware'
    file_path = os.path.join(firmware)
    data = open(file_path, 'rb')
    r = requests.post(url, data=data, headers=headers)
    if r.status_code == 201:
        print('Uploaded firmware to ' + host)
        return True
    else:
        print('Failed to upload firmware to ' + host)
        print(r.text)
        return False

# Upgrade firmware on appliance
def upgradeFirmware(host, api_key):
    headers = {'Authorization': 'ExtraHop apikey=%s' % api_key}
    url = host + 'api/v1/extrahop/firmware/latest/upgrade'
    r = requests.post(url, headers=headers)
    print(r.status_code)
    if r.status_code == 202:
        print('Upgraded firmware on ' + host)
        return True
    else:
        print('Failed to upgrade firmware on ' + host)
        print(r.text)
        return False

# Upgrade firmware for each appliance
for appliance in appliances:
    host = appliance['host']
    api_key = appliance['api_key']
    firmware = appliance['firmware']
    upload_success = uploadFirmware(host, api_key, firmware)
    if upload_success:
        upgradeFirmware(host, api_key)
```

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 `verify=False` option to bypass certificate verification. However, this method is not secure and is not recommended. The following code sends an HTTP GET request without certificate verification:

```
requests.get(url, headers=headers, verify=False)
```

Upgrading Explore appliances

Pre-upgrade tasks

Before upgrading an Explore appliance, you must halt record ingest. You can halt record ingest for all of the nodes in a cluster from a single node.

Note: The message `Could not determine ingest status on some nodes` and `Error` might appear on the Cluster Data Management page in the Admin UI of the upgraded nodes until all nodes in the cluster are upgraded. These errors are expected and can be ignored.

1. Open a terminal application.

2. Run the following command, where `YOUR_KEY` is the API for your user account, and `HOSTNAME` is the hostname of your Explore appliance:

```
curl -X PATCH "https://HOST/api/v1/extrahop/cluster" -H "accept:
 application/json" -H "Authorization: ExtraHop apikey=YOUR_KEY" -H
 "Content-Type: application/json" -d "{ \"ingest_enabled\": false}"
```

Post-upgrade tasks

After you have upgraded all of the nodes in the Explore cluster, enable record ingest.

1. Open a terminal application.
2. Run the following command, where `YOUR_KEY` is the API for your user account, and `HOSTNAME` is the hostname of your Explore appliance:

```
curl -X PATCH "https://HOST/api/v1/extrahop/cluster" -H "accept:
 application/json" -H "Authorization: ExtraHop apikey=YOUR_KEY" -H
 "Content-Type: application/json" -d "{ \"ingest_enabled\": false}"
```