ExtraHop 8.2
ExtraHop Explore REST API Guide

For more documentation, see https://docs.extrahop.com/.

Published: 2021-09-04

ExtraHop Networks
Seattle, WA 98101
877-333-9872 (US)
+44 (0)203 7016850 (EMEA)
+65-31585513 (APAC)
www.extrahop.com

Contents

Introduction to the ExtraHop REST API 4
ExtraHop API requirements 4

Access and authenticate to the ExtraHop REST API 5
Privilege levels 5
Manage API key access 7
Generate an API key 8
Configure cross-origin resource sharing (CORS) 8

Learn about the REST API Explorer 9
Open the REST API Explorer 9
View operation information 9
Identify objects on the ExtraHop system 9

ExtraHop API resources 12
Appliance 12
APIKey 12
ExtraHop 13
License 14
Running config 14

ExtraHop REST API examples 15
Upgrade ExtraHop firmware through the REST API 15
Upgrade ExtraHop firmware with cURL 15
Python script example 16
Upgrading Explore appliances 17

Introduction to the ExtraHop REST API

The ExtraHop REST API enables you to automate administration and configuration tasks on your ExtraHop system. You can send requests to the ExtraHop API through a Representational State Transfer (REST) interface, which is accessed through resource URIs and standard HTTP methods.

When a REST API request is sent over HTTPS to an ExtraHop system, that request is authenticated and then authorized through an API key. After authentication, the request is submitted to the ExtraHop system and the operation completes.

Each ExtraHop system provides access to the built-in ExtraHop REST API Explorer, which enables you to view all of the available system resources, methods, properties, and parameters. The REST API Explorer also enables you to send API calls directly to your ExtraHop system.

Note: This guide is intended for an audience that has a basic familiarity with software development and the ExtraHop system.

ExtraHop API requirements

Before you can begin writing scripts for the ExtraHop REST API or performing operations through the REST API Explorer, you must meet the following requirements:

- Your ExtraHop system must be configured to allow API key generation for the type of user you are (remote or local).
- You must generate a valid API key.
- You must have a user account on the ExtraHop system with appropriate privileges set for the type of tasks you want to perform.

Access and authenticate to the ExtraHop REST API

Administrators, or users with unlimited privileges, control whether users can generate API keys. For example, you can prevent remote users from generating keys or you can disable API key generation entirely. When this functionality is enabled, API keys are generated by users and can be viewed only by the user who generated the key.

Note: Administrators set up user accounts and assign privileges, but then users generate their own API keys. Users can delete API keys for their own account, and users with unlimited privileges can delete API keys for any user. For more information, see Users and user groups.

After you generate an API key, you must append the key to your request headers. The following example shows a request that retrieves metadata about the firmware running on the ExtraHop system:

```
curl -i -X GET --header "Accept: application/json" \
--header "Authorization: ExtraHop apikey=2bc07e55971d4c9a88d0bb4d29ecbb29" \
"https://<hostname-or-IP-of-your-ExtraHop-system>/api/v1/extrahop"
```

Privilege levels

User privilege levels determine which ExtraHop system and administration tasks the user can perform through the ExtraHop REST API.

You can view the privilege levels for users through the `granted_roles` and `effective_roles` properties. The `granted_roles` property shows you which privilege levels are explicitly granted to the user. The `effective_roles` property shows you all privilege levels for a user, including those received outside of the granted role, such as through a user group.

The `granted_roles` and `effective_roles` properties are returned by the following operations:

- GET /users
- GET /users/{username}

The `granted_roles` and `effective_roles` properties support the following privilege levels. Note that the type of tasks for each ExtraHop system vary by the available resources listed in the REST API Explorer.

Privilege level	Actions allowed
"system": "full"	• Enable or disable API key generation for the ExtraHop system. • Generate an API key. • View the last four digits and description for any API key on the system. • Delete API keys for any user. • View and edit cross-origin resource sharing. • Transfer ownership of any non-system dashboard to another user. • Perform any administration task available through the REST API. • Perform any ExtraHop system task available through the REST API.
"write": "full"	• Generate your own API key. • View or delete your own API key. • Change your own password, but you cannot perform any other administration tasks through the REST API.

Privilege level	Actions allowed
	• Perform any ExtraHop system task available through the REST API.
"write": "limited"	• Generate an API key. • View or delete their own API key. • Change your own password, but you cannot perform any other administration tasks through the REST API. • Perform all GET operations through the REST API. • Modify the sharing status of dashboards that you are allowed to edit. • Delete dashboards and activity maps that you own. • Perform metric and record queries.
"write": "personal"	• Generate an API key. • View or delete your own API key. • Change your own password, but you cannot perform any other administration tasks through the REST API. • Perform all GET operations through the REST API. • Delete dashboards and activity maps that you own. • Perform metric and record queries.
"metrics": "full"	• Generate an API key. • View or delete your own API key. • Change your own password, but you cannot perform any other administration tasks through the REST API. • View dashboards and activity maps shared with you. • Perform metric and record queries.
"metrics": "restricted"	• Generate an API key. • View or delete your own API key. • Change your own password, but you cannot perform any other administration tasks through the REST API. • View dashboards and activity maps shared with you.
"packets": "full"	• View and download packets from an ExtraHop system through the `GET/packetcaptures/{id}` operation. This is an add-on privilege that can be granted to a user with one of the following privilege levels: • "write": "full" • "write": "limited" • "write": "personal" • "write": null • "metrics": "full" • "metrics": "restricted"
"packets": "full_with_keys"	• View and download packets from an ExtraHop system through the `GET/packetcaptures/{id}` operation. This is an add-on privilege that can be granted to a user with one of the following privilege levels:

Privilege level	Actions allowed
	• "write": "full" • "write": "limited" • "write": "personal" • "write": null • "metrics": "full" • "metrics": "restricted"
"detections": "full"	• View detections in the ExtraHop system. This is an add-on privilege that can be granted to a user with one of the following privilege levels: • "write": "full" • "write": "limited" • "write": "personal" • "write": null • "metrics": "full" • "metrics": "restricted"
"detections": "none"	• No access to detections. This is an add-on privilege that can be granted to a user with one of the following privilege levels: • "write": "full" • "write": "limited" • "write": "personal" • "write": null • "metrics": "full" • "metrics": "restricted"

Manage API key access

Users with unlimited privileges can configure whether users can generate API keys for the ExtraHop system. You can allow only local users to generate keys, or you can also disable API key generation entirely.

Users must generate an API key before they can perform operations through the ExtraHop REST API. Keys can be viewed only by the user who generated the key or system administrators with unlimited privileges. After a user generates an API key, they must append the key to their request headers.

1. Log in to the Administration settings on the ExtraHop system through `https://<extrahop-hostname-or-IP-address>/admin`.
2. In the Access Settings section, click **API Access**.
3. In the Manage API Access section, select one of the following options:
   - **Allow all users to generate an API key**: Local and remote users can generate API keys.
   - **Only local users can generate an API key**: Remote users cannot generate API keys.
   - **No users can generate an API key**: No API keys can be generated by any user.
4. Click **Save Settings**.

Generate an API key

You must generate an API key before you can perform operations through the ExtraHop REST API. Keys can be viewed only by the user who generated the key or by system administrators with unlimited privileges. After you generate an API key, add the key to your request headers or the ExtraHop REST API Explorer.

Before you begin
Make sure the ExtraHop system is configured to allow API key generation.

1. In the Access Settings section, click **API Access**.
2. In the Generate an API Key section, type a description for the new key, and then click **Generate**.
3. Scroll down to the API Keys section, and copy the API key that matches your description.

You can paste the key into the REST API Explorer or append the key to a request header.

Configure cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) allows you to access the ExtraHop REST API across domain-boundaries and from specified web pages without requiring the request to travel through a proxy server.

You can configure one or more allowed origins or you can allow access to the ExtraHop REST API from any origin. Only administrative users with unlimited privileges can view and edit CORS settings.

1. In the **Access Settings** section, click **API Access**.
2. In the CORS Settings section, specify one of the following access configurations.
   - To add a specific URL, type an origin URL in the text box, and then click the plus (+) icon or press ENTER.

     The URL must include a scheme, such as `HTTP` or `HTTPS`, and the exact domain name. You cannot append a path; however, you can provide a port number.
   - To allow access from any URL, select the Allow API requests from any Origin checkbox.

     Note: Allowing REST API access from any origin is less secure than providing a list of explicit origins.
3. Click **Save Settings** and then click **Done**.

Learn about the REST API Explorer

The REST API Explorer is a web-based tool that enables you to view detailed information about the ExtraHop REST API resources, methods, parameters, properties, and error codes. Code samples are available in Python, cURL, and Ruby for each resource. You also can perform operations directly through the tool.

Open the REST API Explorer

You can open the REST API Explorer from the Administration settings or through the following URL:

```
https://<extrahop-hostname-or-ip-address>/api/v1/explore/
```

1. Log in to the Administration settings on the ExtraHop system through `https://<extrahop-hostname-or-IP-address>/admin`.
2. From the Access Setting section, click **API Access**.
3. On the API Access page, click **REST API Explorer**.
   The REST API Explorer opens in your browser.

View operation information

From the REST API Explorer, you can click any operation to view configuration information for the resource.

The following table provides information about the sections available for resources in the REST API Explorer. Section availability varies by HTTP method. Not all methods have all of the sections listed in the table.

Section	Description
Body Parameters	Provides all of the fields for the request body and supported values for each field.
Parameters	Provides information about the available query parameters.
Responses	Provides information about the possible HTTP status codes for the resource. If you click **Send Request**, this section also includes the response from the server and the cURL, Python, and Ruby syntax required to send the specified request.

Identify objects on the ExtraHop system

Objects on the ExtraHop system can be identified by any unique value, such as the IP address, MAC address, name, or system ID. However, to perform API operations on a specific object, you must locate the object ID. You can easily locate the object ID through the following methods in the REST API Explorer.

- The object ID is provided in the headers returned from a POST request. For example, if you send a POST request to create a page, the response headers display a location URL.

The following request returned the location for the newly created page as `/api/v1/pages/221` and the ID for the page as `221`.

```
{
 "date": "Wed, 25 Nov 2015 17:39:06 GMT",
 "via": "1.1 localhost",
 "server": "Apache",
 "content-type": "text/plain; charset=utf-8",
 "location": "/api/v1/pages/221",
 "cache-control": "private, max-age=0",
 "connection": "Keep-Alive",
 "keep-alive": "timeout=45, max=89",
 "content-length": "0"
}
```

- The object ID is provided for all objects returned from a GET request. For example, if you perform a GET request on all devices, the response body contains information for each device, including the ID.

  The following response body displays an entry for a single device, with an ID of 10212:

```
{
 "mod_time": 1448474346504,
 "node_id": null,
 "id": 10212,
 "extrahop_id": "test0001",
 "description": null,
 "user_mod_time": 1448474253809,
 "discover_time": 1448474250000,
 "vlanid": 0,
 "parent_id": 9352,
 "macaddr": "00:05:G3:FF:FC:28",
 "vendor": "Cisco",
 "is_l3": true,
 "ipaddr4": "10.10.10.5",
 "ipaddr6": null,
 "device_class": "node",
 "default_name": "Cisco5",
 "custom_name": null,
 "cdp_name": "",
 "dhcp_name": "",
 "netbios_name": "",
 "dns_name": "",
 "custom_type": "",
 "analysis_level": 1
},
```

- The object ID is provided in the URL for most objects. For example, in the ExtraHop system, click on **Assets**, and then **Devices**. Select any device and view the URL. In the following example, the URL for the device page shows Oid=10180.

```
https://10.10.10.205/extrahop/#/Devices?details=true&device
Oid=10180&from=6&interval_type=HR&until=0&view=l2stats
```

  To perform specific requests for that device, add 10180 to the id field in the REST API Explorer or to the body parameter in your request.

  The URL for dashboards displays a short_code, which appears after /Dashboard. When you add the short_code to the REST API Explorer or to your request, you must prepend a tilde to the short code.

In the following example, kmC9Y is the short_code. To perform requests for this dashboard, add ~kmC9Y as the value for the short_code field.

```
https://10.10.10.205/extrahop/#/Dashboard/kmC9Y/?from=6&interval_
type=HR&until=0
```

You can also find the short_code and dashboard ID in the Dashboard Properties for any dashboard, which can be accessed from the command menu ⋮. Some API operations, such as DELETE, require the dashboard ID.

ExtraHop API resources

You can perform operations on the following resources through the ExtraHop REST API. You also can view more detailed information about these resources, such as available HTTP methods, query parameters, and object properties in the REST API Explorer.

Appliance

The ExtraHop system consists of a network of connected appliances that perform tasks such as monitoring traffic, analyzing data, storing data, and identifying detections.

You can retrieve information about ExtraHop appliances connected to the local appliance and establish new connections to remote ExtraHop appliances.

Note: You can only establish a connection to a remote ExtraHop appliance that is licensed for the same edition as the local ExtraHop appliance.

The following table displays all of the operations you can perform on this resource:

Operation	Description
GET /appliances	Retrieve all remote ExtraHop appliances connected to the local appliance.
POST /appliances	Establish a new connection to a remote ExtraHop appliance.
GET /appliances/{id}	Retrieve a specific remote ExtraHop appliance connected to the local appliance.
GET /appliances/{id}/productkey	Retrieve the product key of the specified appliance.

Implementation information and instructions for each operation are documented in the REST API Explorer. You can click on any operation in the REST API Explorer to view implementation information such as parameters, response class and messages, and JSON model and schema.

APIKey

An API key enables a user to perform operations through the ExtraHop REST API.

You can generate the initial API key for the setup user account through the REST API. All other API keys are generated through the API Access page in the Administration settings.

The following table displays all of the operations you can perform on this resource:

Operation	Description
GET /apikeys	Retrieve all API keys.
POST /apikeys	Create the initial API key for the setup user account.
GET /apikeys/{keyid}	Retrieve information about a specific API key.

Implementation information and instructions for each operation are documented in the REST API Explorer. You can click on any operation in the REST API Explorer to view implementation information such as parameters, response class and messages, and JSON model and schema.

ExtraHop

This resource provides metadata about the ExtraHop system.

The following table displays all of the operations you can perform on this resource:

Operation	Description
GET /extrahop	Retrieve metadata about the firmware running on the ExtraHop system.
GET /extrahop/cluster	Retrieve Explore cluster configuration settings.
PATCH /extrahop/cluster	Update Explore cluster configuration settings.
PUT /extrahop/detections/access	Update detections access control settings.
GET /extrahop/edition	Retrieve the edition of the ExtraHop system.
POST /extrahop/firmware	Upload a new firmware image to the ExtraHop system. For more information, see Upgrade ExtraHop firmware through the REST API.
POST /extrahop/firmware/latest/upgrade	Upgrade the ExtraHop system to the most recently uploaded firmware image.
GET /extrahop/idrac	Retrieve the iDRAC IP address of the ExtraHop system.
GET /extrahop/platform	Retrieve the platform name of the ExtraHop system.
GET /extrahop/processes	Retrieve a list of processes running on the ExtraHop system.
POST /extrahop/processes/{process}/restart	Restart a process running on the ExtraHop system.
GET /extrahop/services	Retrieve settings for all services.
PATCH /extrahop/services	Update the settings for services.
POST /extrahop/restart	Restart the ExtraHop system.
POST /extrahop/sslcert	Regenerate the SSL certificate on the ExtraHop system. For more information, see Create a trusted SSL certificate through the REST API
PUT /extrahop/sslcert	Replace the SSL certificate on the ExtraHop system.
POST /extrahop/sslcert/signingrequest	Create an SSL certificate signing request. For more information, see Create a trusted SSL certificate through the REST API.
GET /extrahop/ticketing	Retrieve the ticketing integration status.
PATCH /extrahop/ticketing	Enable or disable ticketing integration.
GET /extrahop/version	Retrieve the version of the firmware running on the ExtraHop system.

Implementation information and instructions for each operation are documented in the REST API Explorer. You can click on any operation in the REST API Explorer to view implementation information such as parameters, response class and messages, and JSON model and schema.

License

This resource enables you to retrieve and set product keys or to retrieve and set a license.

The following table displays all of the operations you can perform on this resource:

Operation	Description
GET /license	Retrieve the license applied to this ExtraHop system.
PUT /license	Apply and register a new license to the ExtraHop system.
GET /license/productkey	Retrieve the product key to this ExtraHop system.
PUT /license/productkey	Apply the specified product key to the ExtraHop system and register the license.

Implementation information and instructions for each operation are documented in the REST API Explorer. You can click on any operation in the REST API Explorer to view implementation information such as parameters, response class and messages, and JSON model and schema.

Running config

The running configuration file is a JSON document that contains core system configuration information for the ExtraHop system.

The following table displays all of the operations you can perform on this resource:

Operation	Description
GET /runningconfig	Retrieve the current running configuration file.
PUT /runningconfig	Replace the current running configuration file. Configuration file changes are not automatically saved.
POST /runningconfig/save	Save the current changes to the running configuration file.
GET /runningconfig/saved	Retrieve the saved running configuration file.

Implementation information and instructions for each operation are documented in the REST API Explorer. You can click on any operation in the REST API Explorer to view implementation information such as parameters, response class and messages, and JSON model and schema.

ExtraHop REST API examples

The following examples demonstrate common REST API operations.

- Change a dashboard owner through the REST API
- Extract the device list through the REST API
- Create and assign a device tag through the REST API
- Query for metrics about a specific device through the REST API
- Create, retrieve, and delete an object through the REST API
- Query the record log

Upgrade ExtraHop firmware through the REST API

You can automate upgrades to the firmware on your ExtraHop system 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 system through the cURL command.

Before you begin

- The cURL tool must be installed on your machine.
- The system 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 system, and FILE_PATH is the relative file path of the system 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 system firmware.

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

   ```
   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 system 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 system:

   ```
   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 system. 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 ExtraHop systems by reading 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:

System HTTPS URL	API key	Firmware file path

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

- **SYSTEM_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

SYSTEM_LIST = 'systems.csv'

# Retrieve URLs, API keys, and firmware file paths
systems = []
with open(SYSTEM_LIST, 'rt', encoding='ascii') as f:
    reader = csv.reader(f)
    for row in reader:
        system = {
                        'host': row[0],
```

```
                    'api_key': row[1],
                    'firmware': row[2]
                }
        systems.append(system)

# Function that uploads firmware to system
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

# Function that upgrades firmware on system
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 system
for system in systems:
    host = system['host']
    api_key = system['api_key']
    firmware = system['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 ExtraHop system. 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 Administration settings 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}"
```