ExtraHop 6.1
ExtraHop REST API Guide

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

Published: 2018-10-26

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

Contents

Introduction to ExtraHop REST API

The ExtraHop REST application programming interface (API) enables you to automate administration and configuration tasks on your ExtraHop Discover, Command, and Explore appliances. 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.

You can automate administration tasks, such as configuring LDAP authentication, saving customizations, applying SSL decrypt keys, and managing support packs. And, you can automate configuration tasks, such as creating alerts, or writing triggers.

When a REST API request is sent over HTTPS to an ExtraHop appliance, 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.

The Discover and Command appliances provide access to the built in ExtraHop API Explorer tool, which enables you to view all of the available system resources, methods, properties, and parameters. The API Explorer tool also enables you to test out API calls directly on your ExtraHop appliance.

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 coding against the ExtraHop REST API or performing operations through the ExtraHop API Explorer, you must meet the following requirements:

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

Get started

If you have a user account for your ExtraHop appliance, you can connect to the ExtraHop API Explorer and begin browsing through the available resources.

1. From the ExtraHop Web UI, click the User icon, and then select **API Access**.
2. On the API Access page, click **REST API Explorer**.
3. Locate a resource you want and click **List Operations** to view all operations that you can perform on that resource.
4. Click an operation name to view implementation information such as parameters, response class and messages, and JSON model and schema that are appicable to the operation.

Next steps

Access and authenticate to the ExtraHop REST API

Enable CORS for the ExtraHop REST API

Learn about the ExtraHop REST API Explorer

Learn about the ExtraHop REST API

View ExtraHop REST API examples

Access and authenticate to the ExtraHop REST API

Administrators, or users with full system 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 then users generate their own API key. Users can delete API keys for their own account, and users with full system privileges can delete API keys for any user. For more information, see the Users section in the ExtraHop Admin UI Guide.

After you generate an API key, you must append the key to your request headers. The following example shows a request that would retrieve all of the alerts set on the ExtraHop system:

```
curl -i -X GET --header "Authorization: ExtraHop apikey=39284639207" \
--header "Accept: application/json" \
"https://<hostname-or-IP-of-your-ExtraHop-appliance>/api/v1/alerts"
```

Permission levels

The permission level set for a user specifies what ExtraHop Web UI and ExtraHop Admin UI tasks the user can perform through the ExtraHop REST API.

Permissions levels for users are set through the `granted_roles` property available from the following endpoints:

- POST /users
- PATCH /users/{username}

You can view the permission levels for users through the `granted_roles` and `effective_roles` properties returned by the following endpoints:

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

The `granted_roles` and `effective_roles` properties support the following values:

- "system": "full"
- "system": null
- "write": "full"
- "write": "limited"
- "write": "personal"
- "write": null
- "packets": "full"
- "packets": null

The following table describes what actions you can perform on the ExtraHop appliance at each permission level:

Permission level	Actions allowed
Full system privileges	• Enable or disable API key generation for the ExtraHop appliance. • Generate an API key. • View the last four digits and description for any API key on the system.

Permission level	Actions allowed
	• 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 Admin UI task available through the REST API. • Perform any Web UI task available through the REST API.
Full write privileges	• Generate your own API key. • View or delete your own API key. • Change your own password, but you cannot perform any other Admin UI tasks through the REST API. • Perform any Web UI task available through the REST API.
Limited write privileges	• Generate an API key. • View or delete their own API key. • Change your own password, but you cannot perform any other Admin UI 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 that you own. • Perform metric and record queries.
Personal write privileges Read-only privileges	• Generate an API key. • View or delete your own API key. • Change your own password, but you cannot perform any other Admin UI tasks through the REST API. • Perform all GET operations through the REST API. • Delete dashboards that you own. • Perform metric and record queries.
View and download packets privileges	• View and download packets from an ExtraHop Discover appliance through the `GET/packetcaptures/{id}` operation. This additional privilege can be granted to a user with full write, limited write, personal write, or read-only privileges.

Manage API key access

Users with full system privileges can manage which users are able to generate API keys on the ExtraHop appliance.

1. Log in to the ExtraHop Admin UI through the following URL: https://<hostname-or-IP-of-your-ExtraHop-appliance>/admin
2. In the Access Settings section, click **API Access**.
3. In the Manage Access section, select one of the following options:
   - **Allow All User Generated API Keys**: Local and remote users can generate API keys.
   - **Local Users Only**: Only local users can generate API keys.
   - **No API Keys Allowed**: No API keys can be generated by any user.
4. Click **Save Settings**.

Generate an API key

After you log into the ExtraHop appliance, if API key generation is enabled, you can generate an API key.

1. In the Access Settings section, click **API Access**.
2. In the API Keys section, type a description for the key, and then click **Generate**.
3. Copy the API key and paste the key into the REST API Explorer or append the key to a request header.

Delete an API Key

You can delete an API key from the ExtraHop appliance.

1. In the Access Settings section, click **API Access**.
2. In the Keys section, click the delete (X) icon next to the API key you want to delete.
3. Click **OK**.

Enable CORS for the ExtraHop REST API

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 full system privileges can view and edit CORS settings.

View CORS settings

You can view CORS settings from the ExtraHop Admin UI.

1. Log into the ExtraHop Admin UI through the following URL: https://<hostname-or-IP-of-your-ExtraHop-appliance>/admin
2. In the Access Settings section, click **API Access**.
   The CORS Settings section displays the following settings:
   - The list of URLs that can access the REST API.
   - The status of the **Allow API requests from any Origin** option.

Add an allowed origin

You can configure one or more allowed origins or you can allow access to the ExtraHop REST API from any origin.

1. Log into the ExtraHop Admin UI.
2. In the Access Settings section, click **API Access**.
3. 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.
4. Click **Save Settings**.

Delete an allowed origin

You can delete a URL from the list of allowed origins or disable access from all origins.

1. Log into the ExtraHop Admin UI.
2. In the Access Settings section, click **API Access**.
3. In the CORS Settings section, modify one of the following access configurations.
   - To delete a specific URL, click the delete (X) icon next to the origin you want to delete.
   - To disable access from any URL, clear the **Allow API requests from any Origin** checkbox.
4. Click **Save Settings**.

Learn about the ExtraHop REST API Explorer

The ExtraHop 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, which are performed on your ExtraHop and return information about your network.

Note: Be cautious when clicking the **Try it out!** button, because the operation is performed on your ExtraHop appliance.

Learn about the ExtraHop REST API

View resource information

Click on any resource group in the ExtraHop REST API Explorer to view information about the available methods and the expected URL syntax for the resource.

The following options enable you to manage the information displayed on the main page.

- **Show/Hide:** Expands and collapses information about the resource.
- **List Operations:** Expands information about the resource operations.
- **Expand Operations:** Expands information about all of the resource operations. Clicking the method or path of the expanded operation will collapse the additional information.

View operation information

From the ExtraHop REST API Explorer, you can click on any operation to view additional 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
Implementation Notes	Provides all of the fields for the request body and supported values for each field.
Response Class	Provides the response code and type for successful requests.
Parameters	Provides information about the available query parameters.
Response Messages	Provides additional information about the possible HTTP status codes for the resource.
Model	Provides the JSON body objects and descriptions.
Model Schema	Provides the JSON body schema. Red text indicates user-defined text values. Green text indicates Boolean and number values.

GET requests

GET requests retrieve information about the objects in the associated resource. You can request information about all of the objects in a resource or you can specify an object ID to retrieve detailed information about only that object.

GET example: Retrieve a list of devices

Retrieve a list of devices on the ExtraHop system through the ExtraHop API Explorer.

1. Click **Device**, and then click **GET /devices**.
2. In the Parameters section, modify the fields to build your query. For example, the default limit parameter restricts the number of devices returned to 100.
3. Click the **Try it out!** button. A successful GET request returns a response body with the number of devices specified in the limit parameter. A failed GET request returns the response body with an error.

GET example: Retrieve specific device information

Retrieve information only about a specific device. You must have a device ID, which you can retrieve from the previous example.

1. Click **Device**, and then click **GET /devices/{id}**.
2. In the Parameters section, in the **id** field, type the device ID.
3. Click the **Try it out!** button. A successful GET request returns a response body with information about the specified device. A failed GET request returns the response body with an error.

POST requests

POST requests create objects and queries for the associated resource.

POST example: Create a custom dynamic FTP device group

Create a custom dynamic FTP device group on the ExtraHop system through the ExtraHop API Explorer.

1. Click **Device Group**, and then click **POST /devicegroups**.
2. Click on **Model** to see descriptions for optional and required fields.
3. Click on **Model Schema**, and then click in the box to automatically add the schema to the body parameter.
4. Edit the JSON fields. In the following example, the dynamic parameter is set to true, the field parameter is set to type and the value parameter is set to /^extrahop.device.ftp_servers$/.

```
{
 "description": "FTP Example",
 "dynamic": true,
 "field": "type",
 "include_custom_devices": true,
 "name": "",
 "value": "/^extrahop.device.ftp_server$/"
}
```

5. Click **Try it out!**

Note: To see available parameter options, perform a GET request on the resource. The response body for the GET request displays possible values for the fields.

A 201 status is returned upon success, there is no response body, and the response headers display the location URL (/api/v1/devicegroups/151) and ID (151) for the device group as follows:

```
{
 "date": "Mon, 30 Nov 2015 19:00:28 GMT",
 "via": "1.1 localhost",
 "server": "Apache",
 "content-type": "text/plain; charset=utf-8",
 "location": "/api/v1/devicegroups/151",
 "cache-control": "private, max-age=0",
 "connection": "Keep-Alive",
 "keep-alive": "timeout=45, max=99",
 "content-length": "0"
}
```

Patch requests

PATCH requests update existing objects with modified or missing information.

PATCH example: Modify a device group

Modify a custom dynamic FTP device group on the ExtraHop system through the ExtraHop API Explorer.

1. Click **Device Group** and then click **PATCH /devicegroups/{id}**.
2. In the id parameter, type the ID for the device group. From the POST example above, the device group ID is 151.
3. Add and modify only the fields that you want to change to the body parameter. For example, to add a name for the device group that was created in the example above, type:

   ```
   {
    "name": "FTP Servers"
   }
   ```
4. Click **Try it out!**

DELETE requests

DELETE requests remove objects from the system. You must have an object ID to perform a DELETE operation.

DELETE example: Delete a device group

Delete a custom dynamic FTP device group on the ExtraHop system through the ExtraHop API Explorer.

1. Click **Device Group**, and then click **DELETE /devicegroups/{id}**.
2. In the id parameter, type the ID of the device group you want to delete. In the POST example above, the device group ID is 151.
3. Click **Try it out!**

The device group is deleted from your ExtraHop appliance. A status code of 204 is returned upon success.

PUT requests

For limited operations, you can erase and replace the content in a resource with a PUT request.

PUT example: Overwrite a product key

Erase and replace the license product key on the ExtraHop system through the ExtraHop API Explorer.

1. Click **Licenses** and then click **PUT /licenses/productkey**.
2. Click in the **Model Schema** box to add the schema to the product_key parameter.
3. Replace the string with your product key.
4. Click **Try it out!**

A status code of 204 is returned upon success.

Learn about the ExtraHop REST API

The ExtraHop REST API enables you to automate tasks for the ExtraHop Web UI and Admin UI. In addition, you can view and try all of the available resources through the ExtraHop REST API Explorer and perform operations directly on your ExtraHop appliance.

Learn about the ExtraHop REST API Explorer

Identify objects on the ExtraHop system

View ExtraHop REST API examples

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 ExtraHop REST API Explorer.

Activity group

Activity groups classify devices automatically based on their network traffic.

You can retrieve IDs for all activity groups and then perform additional operations on a group that is associated with a single ID.

For example, activity group IDs can be added to Metric queries to retrieve metrics simultaneously for a group of devices.

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

Operation	Description
GET /activitygroups	Retrieve all activity groups from the ExtraHop appliance.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Alert

Alerts are system notifications that are generated upon an event. Default alerts are available in the system, or you can create a custom alert.

Threshold alerts can be set to alert you if a metric crosses the value defined in the alert. Trend alerts cannot be configured through the REST API.

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

Operation	Description
GET /alerts	Retrieve all alerts.
POST /alerts	Create a new alert with specified values.
DELETE /alerts{id}	Delete a specific alert.
GET /alerts{id}	Retreive a specific alert.
PATCH /alerts{id}	Apply updates to a specific alert.

Operation	Description
GET /alerts{id}/applications	Retrieve all applications that have a specific alert assigned.
POST /alerts{id}/applications	Assign and unassign a specific alert to applications.
DELETE /alerts{id}/applications/{child-id}	Unassign an application from a specific alert.
POST /alerts{id}/applications/{child-id}	Assign an application to a specific alert.
GET /alerts/{id}/devicegroups	Retrieve all device groups that have a specific alert assigned.
POST /alerts/{id}/devicegroups	Assign and unassign a specific alert to device groups.
DELETE /alerts/{id}/devicegroups/{child-id}	Unassign a device group from a specific alert.
POST /alerts/{id}/devicegroups/{child-id}	Assign a device group to a specific alert.
GET /alerts/{id}/devices	Retrieve all devices that have a specific alert assigned.
POST /alerts/{id}/devices	Assign and unassign a specific alert to devices.
DELETE /alerts/{id}/devices/{child-id}	Unassign a device from a specific alert.
POST /alerts/{id}/devices/{child-id}	Assign a device to a specific alert.
GET /alerts/{id}/emailgroups	Retrieve all email groups that have a specific alert assigned.
POST /alerts/{id}/emailgroups	Assign and unassign a specific alert to email groups.
DELETE /alerts/{id}/emailgroups/{child-id}	Unassign a email group from a specific alert.
POST /alerts/{id}/emailgroups/{child-id}	Assign a email group to a specific alert.
GET /alerts/{id}/exclusionintervals	Retrieve all exclustion intervals that have a specific alert assigned.
POST /alerts/{id}/exclusionintervals	Assign and unassign a specific alert to exclustion intervals.
DELETE /alerts/{id}/exclusionintervals/{child-id}	Unassign an exclustion interval from a specific alert.
POST /alerts/{id}/exclusionintervals/{child-id}	Assign an exclustion interval to a specific alert.
GET /alerts/{id}/networks	Retrieve all networks that have a specific alert assigned.
POST /alerts/{id}/networks	Assign and unassign a specific alert to networks.
DELETE /alerts/{id}/networks/{child-id}	Unassign a network from a specific alert.
POST /alerts/{id}/networks/{child-id}	Assign a network to a specific alert.
GET /alerts/{id}/stats	Retrieve all additional statistics for a specific alert.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Application

Applications are user-defined groups that collect metrics identified through triggers across multiple types of traffic.

The default All Activity application contains all collected metrics.

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

Operation	Description
GET /applications	Retrieve all applications that were active within a specific timeframe.
GET /applications/{id}	Retreive a specific application.
PATCH /applications/{id}	Update a specific application.
GET /applications/{id}/activity	Retrieve all activity for a specific application.
GET /applications/{id}/alerts	Retrieve all alerts that are assigned to a specific application.
POST /applications/{id}/alerts	Assign and unassign alerts to a specific application.
DELETE /applications/{id}/alerts/{child-id}	Unassign an alert from a specific application.
POST /applications/{id}/alerts/{child-id}	Assign an alert to a specific application.
GET /applications/{id}/flexgrids	Retrieve all flex grids that are assigned to a specific application.
POST /applications/{id}/flexgrids	Assign and unassign flex grids to a specific application.
DELETE /applications/{id}/flexgrids/{child-id}	Unassign a flex grid from a specific application.
POST /applications/{id}/flexgrids/{child-id}	Assign a flex grid to a specific application.
GET /applications/{id}/geomaps	Retrieve all geomaps that are assigned to a specific application.
POST /applications/{id}/geomaps	Assign and unassign geomaps to a specific application.
DELETE /applications/{id}/geomaps/{child-id}	Unassign a geomap from a specific application.
POST /applications/{id}/geomaps/{child-id}	Assign a geomap to a specific application.
GET /applications/{id}/pages	Retrieve all pages that are assigned to a specific application.
POST /applications/{id}/pages	Assign and unassign pages to a specific application.
DELETE /applications/{id}/pages/{child-id}	Unassign a page from a specific application.
POST /applications/{id}/pages/{child-id}	Assign a page to a specific application.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Audit log

The audit log displays a record of all recorded system administration and configuration activity, such as the time of the activity, the user who performed the activity, the operation, operation details, and system component..

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

Operation	Description
GET /auditlog	Retrieve all audit log messages.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Bundle

Bundles are JSON-formatted documents that contain information about selected system configuration, such as triggers, dashboards, applications, or alerts.

You can create a bundle and then transfer those configurations to another ExtraHop appliance, or save the bundle as a backup. Bundles can also be downloaded from ExtraHop Solution Bundles and applied through the REST API.

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

Operation	Description
GET /bundles	Retrieve metadata about all bundles on the ExtraHop appliance.
POST /bundles	Upload a new bundle to the ExtraHop appliance.
DELETE /bundles/{id}	Delete a specific bundle.
GET /bundles/{id}	Retrieve a specific bundle export.
POST /bundles/{id}/apply	Apply a saved bundle to the ExtraHop appliance.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Custom device

You can create a custom device by defining a set of rules.

For example, you can create a custom device that has an IP address on a specified VLAN. By default, all IP addresses outside of the locally-monitored broadcast domains are aggregated behind a router. To identify devices that are behind that router, you can create a custom device, and then collect metrics from the device.

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

Operation	Description
GET /customdevices	Retrieve all custom devices.
POST /customdevices	Create a custom device.
DELETE /customdevices/{id}	Delete a specific custom device.
GET /customdevices/{id}	Retrieve a specific custom device.

Operation	Description
PATCH /customdevices/{id}	Update a specific custom device.
GET /customdevices/{id}/criteria	Retreive all criteria fro the specific custom device.
POST /customdevices/{id}/criteria	Create a new criterion for a specific custom device.
DELETE /customdevices/{id}/criteria/{child-id}	Delete a criterion for a specific custom device.
GET /customdevices/{id}/criteria/{child-id}	Retrieve a single custom device criterion.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Customization

Similar to bundles, customizations enable you to save ExtraHop configurations for backup. Unlike with bundles, however, you cannot select the information that is contained in a Customization.

All of the major system changes are saved as a JSON-formatted document and can be uploaded to a restored or new ExtraHop appliance. Customizations are accessible only to users with Full System Permissions.

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

Operation	Description
GET /customizations	Retrieve all customizations.
POST /customizations	Create a backup of the customizations on the ExtraHop appliance.
DELETE /customizations/{id}	Delete a specific customizations.
GET /customizations/{id}	Retrieve a specific customizations.
POST /customizations/{id}/apply	Apply a backup of the customizations on the ExtraHop appliance.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Dashboards

Dashboards are built-in or customized views of your ExtraHop metrics information.

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

Operation	Description
GET /dashboards	Retrieve all dashboards.
DELETE /dashboards/{id}	Delete a specific dashboard.
GET /dashboards/{id}	Retrieve a specific dashboard.
PATCH /dashboards/{id}	Update ownership of a specific dashboard.
GET /dashboards/{id}/sharing	Retrieve the users and their sharing permissions for a specific dashboard.

Operation	Description
PATCH /dashboards/{id}/sharing	Update the users and their sharing permissions for a specific dashboard.
PUT /dashboards/{id}/sharing	Replace the users and their sharing permissions for a specific dashboard.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Device

Devices are objects on your network that have been identified and classified by your ExtraHop appliance.

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

Operation	Description
GET /devices	Retrieve all devices that were active within a specific time period.
GET /devices/{id}	Retrieve a specific device.
PATCH /devices/{id}	Update a specific device.
GET /devices/{id}/activity	Retrieve all activity for a device.
GET /devices/{id}/alerts	Retrieve all alerts that are assigned to a specific device.
POST /devices/{id}/alerts	Assign and unassign a specific device to alerts.
DELETE /devices/{id}/alerts/{child-id}	Unassign an alert from a specific device.
POST /devices/{id}/alerts/{child-id}	Assign an alert to a specific device.
GET /devices/{id}/devicegroups	Retrieve all device groups that are assigned to a specific device.
POST /devices/{id}/devicegroups	Assign and unassign a specific device to device groups.
DELETE /devices/{id}/devicegroups/{child-id}	Unassign a device group from a specific device.
POST /devices/{id}/devicegroups/{child-id}	Assign a device group to a specific device.
GET /devices/{id}/flexgrids	Retrieve all flex grids that are assigned to a specific device.
POST /devices/{id}/flexgrids	Assign and unassign a specific device to flex grids.
DELETE /devices/{id}/flexgrids/{child-id}	Unassign a flex grid from a specific device.
POST /devices/{id}/flexgrids/{child-id}	Assign a flex grid to a specific device.
GET /devices/{id}/geomaps	Retrieve all geomaps that are assigned to a specific device.
POST /devices/{id}/geomaps	Assign and unassign a specific device to geomaps.
DELETE /devices/{id}/geomaps/{child-id}	Unassign a geomap from a specific device.
POST /devices/{id}/geomaps/{child-id}	Assign a geomap to a specific device.

Operation	Description
GET /devices/{id}/pages	Retrieve all pages that are assigned to a specific device.
POST /devices/{id}/pages	Assign and unassign a specific device to pages.
DELETE /devices/{id}/pages/{child-id}	Unassign a page from a specific device.
POST /devices/{id}/pages/{child-id}	Assign a page to a specific device.
GET /devices/{id}/tags	Retrieve all tags that are assigned to a specific device.
POST /devices/{id}/tags	Assign and unassign a specific device to tags.
DELETE /devices/{id}/tags/{child-id}	Unassign a tag from a specific device.
POST /devices/{id}/tags/{child-id}	Assign a tag to a specific device.
GET /devices/{id}/triggers	Retrieve all triggers that are assigned to a specific device.
POST /devices/{id}/triggers	Assign and unassign a specific device to triggers.
DELETE /devices/{id}/triggers/{child-id}	Unassign a trigger from a specific device.
POST /devices/{id}/triggers/{child-id}	Assign a trigger to a specific device.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Device group (or custom group)

Device groups can be either static or dynamic.

A static device group is user-defined; you create a device group and then manually identify and assign each device to that group. A dynamic device group is defined and automatically managed by a set of configured rules.

For example, you can create a device group and then set a rule to classify all devices within a certain IP address range to be added to that group automatically.

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

Operation	Description
GET /devicegroups	Retrieve all device groups that were active within a specific time period.
POST /devicegroups	Create a new device group.
DELETE /devicegroups/{id}	Delete a device group.
GET /devicegroups/{id}	Retrieve a specific device group.
PATCH /devicegroups/{id}	Update a specific device group.
GET /devicegroups/{id}/alerts	Retrieve all alerts that are assigned to a specific device group.
POST /devicegroups/{id}/alerts	Assign and unassign a specific device group to alerts.
DELETE /devicegroups/{id}/alerts/{child-id}	Unassign an alert from a specific device group.

Operation	Description
POST /devicegroups/{id}/alerts/{child-id}	Assign an alert to a specific device group.
GET /devicegroups/{id}/devices	Retrieve all devices in the device group that are active within a specific time window.
POST /devicegroups/{id}/devices	Assign and unassign a devices to a specific static device group.
DELETE /devicegroups/{id}/devices/{child-id}	Unassign a device from a specific static device group.
POST /devicegroups/{id}/devices/{child-id}	Assign a device to a specific static device group.
GET /devicegroups/{id}/flexgrids	Retrieve all flex grids that are assigned to a specific device group.
POST /devicegroups/{id}/flexgrids	Assign and unassign a specific device group to flex grids.
DELETE /devicegroups/{id}/flexgrids/{child-id}	Unassign a flex grid from a specific device group.
POST /devicegroups/{id}/flexgrids/{child-id}	Assign a flex grid to a specific device group.
GET /devicegroups/{id}/geomaps	Retrieve all geomaps that are assigned to a specific device group.
POST /devicegroups/{id}/geomaps	Assign and unassign a specific device group to geomaps.
DELETE /devicegroups/{id}/geomaps/{child-id}	Unassign a geomap from a specific device group.
POST /devicegroups/{id}/geomaps/{child-id}	Assign a geomap to a specific device.
GET /devicegroups/{id}/pages	Retrieve all pages that are assigned to a specific device group.
POST /devicegroups/{id}/pages	Assign and unassign a specific device to pages group.
DELETE /devicegroups/{id}/pages/{child-id}	Unassign a page from a specific device group.
POST /devicegroups/{id}/pages/{child-id}	Assign a page to a specific device group.
GET /devicegroups/{id}/tags	Retrieve all tags that are assigned to a specific device group.
POST /devicegroups/{id}/tags	Assign and unassign a specific device group to tags.
DELETE /devicegroups/{id}/tags/{child-id}	Unassign a tag from a specific device group.
POST /devicegroups/{id}/tags/{child-id}	Assign a tag to a specific device group.
GET /devicegroups/{id}/triggers	Retrieve all triggers that are assigned to a specific device group.
POST /devicegroups/{id}/triggers	Assign and unassign a specific device group to triggers.
DELETE /devicegroups/{id}/triggers/{child-id}	Unassign a trigger from a specific device group.
POST /devicegroups/{id}/triggers/{child-id}	Assign a trigger to a specific device group.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Email group

You can add individual or group email addresses to an email group and assign them to a system alert. When that alert is triggered, the system sends an email to all of the addresses in the email group.

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

Operation	Description
GET /emailgroups	Retrieve all email groups.
POST /emailgroups	Create a new email group.
DELETE /emailgroups/{id}	Delete a email group by a unique identifier.
GET /emailgroups/{id}	Retrieve a specific email group by a unique identifier.
PATCH /emailgroups/{id}	Apply updates to a specific email group.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Exclusion intervals

An exclusion interval can be created to set a time period to suppress an alert.

For example, if you do not want to be notified about alerts after hours or on the weekends, an exclusion interval can create a rule to suppress the alert during that time period.

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

Operation	Description
GET /exclusionintervals	Retrieve all exclusion intervals.
POST /exclusionintervals	Create a new exclusion interval.
DELETE /exclusionintervals/{id}	Delete a specific exclusion interval.
GET /exclusionintervals/{id}	Retrieve a specific exclusion interval.
PATCH /exclusionintervals/{id}	Apply updates to a specific exclusion interval.

Implementation information and instructions for each operation are documented in the ExtraHop 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 appliance, such as the firmware version or if the appliance is a Command appliance.

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 appliance.

Operation	Description
GET /extrahop/platform	Retrieve the platform name of the ExtraHop appliance.
GET /extrahop/processes	Retrieve a list of processes running on the ExtraHop appliance.
GET /extrahop/processes/{process}/restart	Restart a process running on the ExtraHop appliance.
POST /extrahop/sslcert	Regenerate the SSL certificate on the ExtraHop appliance.
PUT /extrahop/sslcert	Replace the SSL certificate on the ExtraHop appliance.
GET /extrahop/version	Retrieve the version of the firmware running on the ExtraHop appliance.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Flex Grid

A Flex Grid provides a table view of metrics information about devices.

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

Operation	Description
GET /flexgrids	Retrieve all flex grids.
GET /flexgrids/{id}/applications	Retrieve all applications that have a flex grid assigned.
POST /flexgrids/{id}/applications	Assign and unassign a specific flex grid to applications.
DELETE /flexgrids/{id}/applications/{child-id}	Unassign an application from a specific flex grid.
POST /flexgrids/{id}/applications/{child-id}	Assign an application to a specific flex grid.
GET /flexgrids/{id}/devicegroups	Retrieve all device groups that are assigned to a flex grid.
POST /flexgrids/{id}/devicegroups	Assign and unassign a specific flex grid to device groups.
DELETE /flexgrids/{id}/devicegroups/{child-id}	Unassign a device group from a specific flex grid.
POST /flexgrids/{id}/devicegroups/{child-id}	Assign a device group to a specific flex grid.
GET /flexgrids/{id}/devices	Retrieve all devices that have a flex grid assigned.
POST /flexgrids/{id}/devices	Assign and unassign a specific flex grid to devices.
DELETE /flexgrids/{id}/devices/{child-id}	Unassign a device from a specific flex grid.
POST /flexgrids/{id}/devices/{child-id}	Assign a device to a specific flex grid.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Geomap

Geomaps display metrics across a global map, which indicates where metrics activity has occurred.

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

Operation	Description
GET /geomaps	Retrieve all geomaps.
POST /geomaps	Create a geomap.
DELETE /geomaps/{id}	Delete a single geomap by unique identifier.
GET /geomaps/{id}	Retrieve a single geomap by unique identifier.
PATCH /geomaps/{id}	Update a single geomap by unique identifier.
GET /geomaps/{id}/applications	Retrieve all applications that are assigned to a specific geomap.
POST /geomaps/{id}/applications	Assign and unassign a specific geomap to applications.
DELETE /geomaps/{id}/applications/{child-id}	Unassign an application from a specific geomap.
POST /geomaps/{id}/applications/{child-id}	Assign an application to a specific geomap.
GET /geomaps/{id}/devicegroups	Retrieve all device groups that are assigned to a specific geomap.
POST /geomaps/{id}/devicegroups	Assign and unassign a specific geomap to device groups.
DELETE /geomaps/{id}/devicegroups/{child-id}	Unassign a device group from a specific geomap.
POST /geomaps/{id}/devicegroups/{child-id}	Assign a device group to a specific geomap.
GET /geomaps/{id}/devices	Retrieve all devices that are assigned to a specific geomap.
POST /geomaps/{id}/devices	Assign and unassign a specific geomap to devices.
DELETE /geomaps/{id}/devices/{child-id}	Unassign a device from a specific geomap.
POST /geomaps/{id}/devices/{child-id}	Assign a device to a specific geomap.

Implementation information and instructions for each operation are documented in the ExtraHop 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 appliance.
PUT /license	Apply and register a new license to the ExtraHop appliance.
GET /license/productkey	Retrieve the product key to this ExtraHop appliance.

Operation	Description
PUT /license/productkey	Apply the specified product key to the ExtraHop appliance and register the license.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Metrics

Metrics information is collected about every object identified by the ExtraHop appliance.

Note that metrics are retrieved through the POST method, which creates a query to collect the requested information through the API.

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

Operation	Description
POST /metrics	Perform a metric query.
GET /metrics/next/{xid}	Retrieve the next chunked results from a metrics query request. This request is only valid on a Command appliance.
POST /metrics/total	Perform a metric query for total values.
POST /metrics/totalbyobject	Perform a metric query for total values that are grouped by object.

For example, if you want to see all HTTP responses that occurred on the network in the last 30 seconds, enter the following request schema into the `POST /metrics` operation:

```
{
  "cycle": "auto",
  "from": -1800000,
  "metric_category": "http",
  "metric_specs": [
    {
      "name": "rsp"
    }
  ],
  "object_ids": [
    0
  ],
  "object_type": "application",
  "until": 0
}
```

The response body returns a list of HTTP responses and the time of each event, similar to the following output:

```
{
  "stats": [
    {
      "oid": 0,
      "time": 1494539640000,
      "duration": 30000,
      "values": [
        354
      ]
```

```
    },
    {
      "oid": 0,
      "time": 1494539640000,
      "duration": 30000,
      "values": [
        354
      ]
    },

    {
      "oid": 0,
      "time": 1494539640000,
      "duration": 30000,
      "values": [
        354
      ]
    },
  ],
  "cycle": "30sec",
  "node_id": 0,
  "clock": 1494541440000,
  "from": 1494539640000,
  "until": 1494541440000
}
```

Enter the same request schema into the `POST /metrics/total` operation to retrieve a count of all HTTP responses that occurred on the network in the last 30 seconds. The response body is similar to the following output:

```
{
  "stats": [
    {
      "oid": -1,
      "time": 1494541380000,
      "duration": 1800000,
      "values": [
        33357
      ]
    }
  ],
  "cycle": "30sec",
  "node_id": 0,
  "clock": 1494541440000,
  "from": 1494539640000,
  "until": 1494541440000
}
```

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Network

Networks are correlated to the network interface card that receives input from all of the objects identified by the ExtraHop appliance.

On an ExtraHop Command appliance, each connected appliance is identified as a network capture that is looking at the traffic for each ExtraHop Discover appliance that is connected to the Command appliance.

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

Operation	Description
GET /networks	Retrieve all networks.
GET /networks/{id}	Retreive a specific network by ID.
PATCH /networks/{id}	Update a specific network by ID.
GET /networks/{id}/alerts	Retrieve all alerts that are assigned to a specific network.
POST /networks/{id}/alerts	Assign and unassign alerts to a specific network.
DELETE /networks/{id}/alerts/{child-id}	Unassign an alert from a specific network.
POST /networks/{id}/alerts/{child-id}	Assign an alert to a specific network.
GET /networks/{id}/pages	Retrieve all pages that are assigned to a specific network.
POST /networks/{id}/pages	Assign and unassign pages to a specific network.
DELETE /networks/{id}/pages/{child-id}	Unassign a page from a specific network.
POST /networks/{id}/pages/{child-id}	Assign a page to a specific network.
GET /networks/{id}/vlans	Retrieve all VLANS assigned to a specific network.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Node

A node is defined by its relationship to an ExtraHop Command appliance. The environment which contains Discover nodes and a Command appliance is called a Command cluster.

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

Operation	Description
GET /nodes	Retrieve all Discover nodes connected to this Command appliance.
GET /nodes/{id}	Retrieve a specific Discover node that is connected to this Command appliance.
PATCH /nodes/{id}	Update a specific Discover node that is connected to this Command appliance.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Packet capture

Packet captures store packets from a network traffic flow.

You must write a trigger to identify the information you want to generate. For example, you can write a trigger to collect all of the packets going to a particular device that is generating a high volume of errors. Then, you can download or delete that information.

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

Operation	Description
GET /packetcaptures	Retrieve metadata about all packet captures stored on this ExtraHop appliance.
DELETE /packetcaptures/{id}	Permanently remove a specific packet capture from the ExtraHop appliance.
GET /packetcaptures/{id}	Download a specific packet capture in PCAP format.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Page

Pages provide a template for creating a customized view of built-in metrics or metrics collected from triggers.

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

Operation	Description
GET /pages	Retrieve all pages.
POST /pages	Create a page.
DELETE /pages/{id}	Delete a single page.
GET /pages/{id}	Retrieve a single page.
PATCH /pages/{id}	Update a single page.
GET /pages/{id}/applications	Retrieve all applications that have a specific page assigned.
POST /pages/{id}/applications	Assign and unassign a specific page to applications.
DELETE /pages/{id}/applications/{child-id}	Unassign an application from a specific page.
POST /pages/{id}/applications/{child-id}	Assign an application to a specific page.
GET /pages/{id}/devicegroups	Retrieve all device groups that are assigned to a specific page.
POST /pages/{id}/devicegroups	Assign and unassign a specific page to device groups.
DELETE /pages/{id}/devicegroups/{child-id}	Unassign a device group from a specific page.
POST /pages/{id}/devicegroups/{child-id}	Assign a device group to a specific page.
GET /pages/{id}/devices	Retrieve all devices that have a specific page assigned.
POST /pages/{id}/devices	Assign and unassign a specific page to devices.
DELETE /pages/{id}/devices/{child-id}	Unassign a device from a specific page.
POST /pages/{id}/devices/{child-id}	Assign a device to a specific page.
GET /pages/{id}/networks	Retrieve all networks that have a specific page assigned.

Operation	Description
POST /pages/{id}/networks	Assign and unassign a specific page to networks.
DELETE /pages/{id}/networks/{child-id}	Unassign a network from a specific page.
POST /pages/{id}/networks/{child-id}	Assign a network to a specific page.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Record Log

Records are structured flow and transaction information about events on your network. After you connect an ExtraHop Discover appliance to an ExtraHop Explore appliance, you can generate and send record information to the Explore appliance for storage, and you can query records to retrieve stored information about any object on your network.

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

Operation	Description
GET /records/cursor/{cursor}	Retrieve records starting at a specified cursor.
POST /records/search	Perform a record log query.

Implementation information and instructions for each operation are documented in the ExtraHop 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 appliance.

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 ExtraHop 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.

SSL decrypt key

This resource enables you to add a decryption key for your network traffic.

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

Operation	Description
GET /ssldecryptkeys	Retrieve all SSL decryption keys.
POST /ssldecryptkeys	Create a new SSL decryption key.
DELETE /ssldecryptkeys/{id}	Remove an SSL key from the ExtraHop appliance.
GET /ssldecryptkeys/{id}	Retrieve an SSL PEM and metadata.
PATCH /ssldecryptkeys/{id}	Update an existing SSL decryption key.
GET /ssldecryptkeys/{id}/protocols	Retrieve all protocols assigned to an SSL decryption key.
POST /ssldecryptkeys/{id}/protocols	Create a new protocol for an SSL decryption key.
DELETE /ssldecryptkeys/{id}/protocols/{child-id}	Delete a protocol from an SSL decryption key.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Support pack

A support pack is a file that contains configuration adjustments provided by ExtraHop Support.

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

Operation	Description
GET /supportpacks	Retrieve metadata about all support packs.
POST /supportpacks/execute	Run a new support pack.
GET /supportpacks/queue/{id}	Check on the status of an in-progress, running support pack.
GET /supportpacks/{filename}	Downlolad an existing support pack by filename.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Tag

Device tags enable you to associate a device or group of devices by some characteristic. For example, you might tag all of your HTTP servers or tag all of the devices that are in a common subnet.

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

Operation	Description
GET /tags	Retrieve all tags.
POST /tags	Create a a new tag.
DELETE /tags/{id}	Delete a specific tag.
GET /tags/{id}	Retrieve a specific tag.
PATCH /tags/{id}	Apply updates to a specific tag.
GET /tags/{id}/devices	Retrieve all devices that are assigned to a specific tag.

Operation	Description
POST /tags/{id}/devices	Assign and unassign a specific tag to devices.
DELETE /tags/{id}/devices/{child-id}	Unassign a device from a specific tag.
POST /tags/{id}/devices/{child-id}	Assign a device to a specific tag.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Trigger

Triggers are custom scripts that perform an action upon a pre-defined event.

For example, you can write a trigger to record a custom metric every time an HTTP request occurs, or classify traffic for a particular server as an Application server. For more information, see the Trigger API Reference. For supplemental implementation notes about advanced options, see Advanced trigger options.

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

Operation	Description
GET /triggers	Retrieve all triggers.
POST /triggers	Create a new trigger.
DELETE /triggers/{id}	Delete a specific identifier.
GET /triggers/{id}	Retrieve a specific trigger by unique identifier.
PATCH /triggers/{id}	Update an existing trigger.
GET /triggers/{id}/devicegroups	Retrieve all device groups that are assigned to a specific trigger.
POST /triggers/{id}/devicegroups	Assign and unassign a specific trigger to device groups.
DELETE /triggers/{id}/devicegroups/{child-id}	Unassign a device group from a specific trigger.
POST /triggers/{id}/devicegroups/{child-id}	Assign a device group to a specific trigger.
GET /triggers/{id}/devices	Retrieve all devices that are assigned to a specific trigger.
POST /triggers/{id}/devices	Assign and unassign a specific trigger to devices.
DELETE /triggers/{id}/devices/{child-id}	Unassign a device from a specific trigger.
POST /triggers/{id}/devices/{child-id}	Assign a device to a specific trigger.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

User

The user resource enables you to create and manage the list of users who have access to the ExtraHop appliance and the permission levels for those users.

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

Operation	Description
GET /users	Retrieve all users.
POST /users	Create a new user.
DELETE /users/{username}	Delete a specific user.
GET /users/{username}	Retrieve a specific user.
PATCH /users/{username}	Update settings for a specific user.
GET /users/{username}/apikeys	Retrieve all API keys for a specific user.
GET /users/{username}/apikeys/{keyid}	Retrieve information about a specific API key and user.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

User group

The user group resource enables you to manage and update groups of users and their dashboard sharing associations.

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

Operation	Description
GET /usergroups	Retrieve all user groups.
POST /usergroups/refresh	Query LDAP for the most recent user memberships for all remote user groups.
GET /usergrups/{id}	Retrieve a specific user group.
PATCH /usergroups/{id}	Update a specific user group.
DELETE /usergroups/{id}/associations	Delete all dashboard sharing associations with a specific user group.
GET /usergroups/{id}/members	Retrieve all members of a specific user group.
POST /usergroups/{id}/refresh	Query LDAP for the most recent user membership of a specific remote user group.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

VLAN

Virtual LANs are logical groupings of traffic or devices on the network.

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

Operation	Description
GET /vlans	Retrieve all VLANs
PATCH /vlans/{id}	Update a specific VLAN.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

Whitelist

A whitelist prioritizes devices that are added after the device limit is reached. When the device limit is reached, any new devices that are added are placed in limited analysis mode.

If you want to prioritize a device, you can add that device to the whitelist. Devices are removed from the ExtraHop device list based on seniority; the latest devices are removed first.

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

Operation	Description
DELETE /whitelist/device/{id}	Remove a device from the whitelist.
POST /whitelist/device/{id}	Add a device to the whitelist.
GET /whitelist/devices	Retrieve all devices that are in the whitelist.
POST /whitelist/devices	Add or remove devices from the whitelist.

Implementation information and instructions for each operation are documented in the ExtraHop 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.

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. To perform API operations on a specific object, you must locate the object ID.

You can locate an object ID through the following methods:

- 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, such as:

```
{
 "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 location for the newly created page is /api/v1/pages/221 and the ID for the page is 221.
- 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
},
```

To perform further requests on a specific device, add the ID in the request for that device.

- The object ID is provided in the URL for most objects. For example, in the ExtraHop Web UI, click on Metrics, and then Devices. Select any device. The URL for that device page displays an OIDD=<number>, such as:

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

To perform further requests for that device, add 10180 to the id field in the ExtraHop 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 ExtraHop API Explorer or to your request, you must prepend a tilde to the short code.

In the following example, kmC9Y is the short_code:

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

You can also find the short_code in Dashboard Properties in the command menu from any dashboard.

View ExtraHop REST API implementation notes

This section contains supplemental implementation notes to the ExtraHop REST API. These notes provide additional conceptual or reference information to help you configure certain parameters or options.

The following implementation notes are available:

- Advanced trigger options
- Supported time units
- Operand values in record queries

Advanced trigger options

Advanced trigger options are configuration options that you can set depending on the system events associated with the trigger. For example, you can configure the number of payload bytes to buffer on HTTP request events.

Advanced options are contained in the `hints` object of the trigger resource as shown in the following example:

```
"hints": {
"flowClientPortMin": null,
"flowClientBytes": 16384,
"flowClientPortMax": null,
"flowServerBytes": 16384,
"flowPayloadTurn": true,
"flowServerPortMin": 135,
"flowServerPortMax": 49155
}
```

The following table describes available advanced options and applicable events:

Option	Description	Applicable events
`"snaplen": string`	Specifies the number of bytes to capture per packet. Capture starts with the first bytes in the packet. Specify this option only if the trigger script performs packet capture. A value of 0 specifies that the capture should collect all bytes in each packet.	All events except: • ALERT_RECORD_COMMIT • METRIC_CYCLE_BEGIN • METRIC_CYCLE_END • FLOW_REPORT • NEW_APPLICATION • NEW_DEVICE • SESSION_EXPIRE
`"payloadBytes": number`	Specifies the number of captured payload bytes to buffer.	• CIFS_REQUEST • CIFS_RESPONSE • HTTP_REQUEST • HTTP_RESPONSE • ICA_TICK
`"clipboardBytes": number`	Specifies the number of bytes to buffer on a Citrix clipboard transfer.	• ICA_TICK

Option	Description	Applicable events
`"cycle": [30sec, 5min, 1hr, 24hr]`	Specifies the length of the metric cycle, expressed in seconds.	• METRIC_CYCLE_BEGIN • METRIC_CYCLE_END • METRIC_RECORD_COMMIT
`"metricTypes": string`	Specifies the metric type by the raw metric name such as extrahop.device.http_server.	• ALERT_RECORD_COMMIT • METRIC_RECORD_COMMIT
`"flowPayloadTurn": boolean`	Enables packet capture on each flow turn. Per-turn analysis continuously analyzes communication between two endpoints to extract a single payload data point from the flow. If this option is enabled, any values specified for the `flowClientString` and `flowServerString` options are ignored.	• SSL_PAYLOAD • TCP_PAYLOAD
`"flowClientPortMin": number`	Specifies the minimum port number of the client port range. Valid values are 0 to 65535. A value of 0 specifies matching of any port.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
`"flowClientPortMax": number`	Specifies the maximum port number of the client port range. Valid values are 0 to 65535. Any value specified for this option is ignored if the value of the `flowClientPortMin` option is 0.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
`"flowClientBytes": number`	Specifies the number of client bytes to buffer. The value of this option cannot be set to 0 if the value of the `flowServerBytes` option is also set to 0.	• SSL_PAYLOAD • TCP_PAYLOAD
`"flowClientString": string`	Specifies the format string of client data to process. Any value specified for this option is ignored if the `flowPayloadTurn` option is enabled.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
`"flowServerPortMin": number`	Specifies the minimum port number of the server port range. Valid values are 0 to 65535.	• SSL_PAYLOAD • TCP_PAYLOAD

Option	Description	Applicable events
	A value of 0 specifies matching of any port.	• UDP_PAYLOAD
`"flowServerPortMax": number`	Specifies the maximum port number of the server port range. Valid values are 0 to 65535. Any value specified for this option is ignored if the value of the `flowServerPortMin` option is 0.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
`"flowServerBytes": number`	Specifies the number of server bytes to buffer. The value of this option cannot be set to 0 if the value of the `flowClientBytes` option is also set to 0.	• SSL_PAYLOAD • TCP_PAYLOAD
`"flowServerString": string`	Specifies the format string of server data to process. Returns the entire packet upon a string match. Any value specified for this option is ignored if the `flowPayloadTurn` option is enabled.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
`"flowUdpAll": boolean`	Enables capture of all UDP datagrams.	• UDP_PAYLOAD
`"flowClassifyOnExpiration": boolean`	Enables running the event upon expiration in order to accumulate metrics for flows that were not classified before expiring.	• FLOW_CLASSIFY

Supported time units

For most parameters, the default unit for time measurement is milliseconds. Some parameters in GET and POST operations return or accept alternative time units such as minutes and hours.

The following table displays supported time units:

Time unit	Unit suffix
Year	y
Month	M
Week	w
Day	d
Hour	h
Minute	m

Time unit	Unit suffix
Second	s
Millisecond	ms

To specify a time unit other than milliseconds for a parameter, append the unit suffix to the value. For example, to request devices active in the last 30 minutes, specify the following parameter value:

```
GET /api/v1/devices?active_from=-30m
```

The following example specifies a search for HTTP records created between 1 and 2 hours ago:

```
{
    "from": "-2h",
    "until": "-1h",
    "types": ["~http"]
}
```

Operand values in record queries

The `operand` field in the `POST /records/search` method specifies the value that a record query attempts to match. You can specify only the value, or you can specify both the data type and the value. If you specify only the value, the query will refer to the record format associated with the `field` parameter to determine the data type of the value.

For example, if you want to search for an IP address, you can specify an IP address data type, and then provide the actual address as the value.

The following example explicitly specifies the data type and value of the operand:

```
{
    "from": -1000,
    "filter": {
        "field" : "senderAddr",
        "operator": "=",
        "operand" : { "type" : "ipaddr4", "value": "1.2.3.4" }
    }
}
```

The following example specifies only the value of the operand:

```
{
    "from": -1000,
    "filter": {
        "field" : "senderAddr",
        "operator": "=",
        "operand" : "1.2.3.4"
    }
}
```

You can explicitly specify the following data types in the `operand` field:

- ipaddr4
- ipaddr6
- device
- application
- string

- number
- boolean

View ExtraHop REST API examples

The following examples are available:

Example 1: Set up an SSL certificate

Before making requests to an ExtraHop appliance with a self-signed certificate, you must set up an SSL certificate for each user who will access the ExtraHop appliance from a particular computer.

In each of the following examples, replace {HOST} with the hostname of your ExtraHop system and replace {API KEY} with a valid API key from your ExtraHop system.

Note: The SSL certificate applies only to the user performing the command. Each user must run the command with their login credentials to set up the SSL certificate.

Set up SSL through Windows Powershell

```
Invoke-WebRequest "http://{HOST}/public.cer" -OutFile ($env:USERPROFILE +
"\ex.cer"); Import-Certificate ($env:USERPROFILE + "\ex.cer")
-CertStoreLocation Cert:\CurrentUser\Root
```

Set up SSL through OS X

```
curl -O http://{HOST}/public.cer; security add-trusted-cert -r trustRoot -k
~/Library/Keychains/login.keychain public.cer
```

Example 2: Create and assign a device tag

The following example shows how you can create a device tag and then assign that tag to all of the devices in a specified subnet.

```
#!/usr/bin/env python

import httplib
import urllib
import json
import sys

# Configuration Options:
host = "{HOST}"
apikey = "{API KEY}"
tag_name = "MyTestTag"
subnet = "10.20.0.[0-9]+"
batch_limit = 100
headers = {'Accept': 'application/json',
           'Authorization': "ExtraHop apikey=%s" % apikey}
conn = httplib.HTTPSConnection(host)
  def execute_req(method, path, expected_code, failure_message, body=None):
```

```
"""
   Returns the body of a successful request,
   otherwise prints error and terminates
"""

conn.request(method, "/api/v1" + path, headers=headers, body=body)
resp = conn.getresponse()
if resp.status is not expected_code:
    print(failure_message)
    print(resp.read())
    sys.exit(1)
return resp

def execute_get(path, expected_code, failure_message):
    resp = execute_req("GET", path, expected_code, failure_message)
    return json.loads(resp.read())

def execute_create(path, body, expected_code, failure_message):
    """Returns ID of newly created resource"""
    resp = execute_req("POST", path, expected_code, failure_message, body)
    resp.read() # drain the response
    return int(resp.getheader("location").split("/")[-1])

# First, search for the specified tag, by name
resp = execute_get("/tags", 200, "Unable to retrieve tags from ExtraHop")
tags = [tag for tag in resp if tag["name"] == tag_name]

if not tags:
    # tag is not found, create it
    body = json.dumps({"name": tag_name})
    tag_id = execute_create('/tags', body, 201, "Unable to create tag")
else:
    tag_id = tags[0]["id"]

query_params = {'limit': batch_limit,
                'search_type': 'ip address',
                'value': subnet}
query_string = urllib.urlencode(query_params)

# Paginate device results, building up a list of all devices to assign
device_ids = []
offset = 0

while True:
    path = "/devices?" + query_string + ("&offset=%d" % offset)
    resp = execute_get(path, 200, "Unable to retrieve devices")
    if not resp:
        break

    device_ids += [device["id"] for device in resp]
    offset += batch_limit

# Perform the assignments
resp = execute_req("POST", "/tags/%d/devices" % tag_id,
                   204, "Unable to perform assignments",
                   body=json.dumps({"assign": device_ids}))
resp.read() # drain the response

# Check that assignments were successful
resp = execute_get("/tags/%d/devices" % tag_id,
                   200, "Unable to retrieve tag assignments")
assigned_device_ids = [device["id"] for device in resp]

successful = set(device_ids).issubset(set(assigned_device_ids))
```

```
if successful:
    print("%d devices assigned to tag" % len(device_ids))
else:
    print("Unable to assign all devices to tag")
```

Example 3: Query for metrics about a specific device

The following request example shows how you can query for metrics from an HTTP client device with the ID 9363 and print the response.

```
import httplib

headers = {'Content-Type': 'application/json',
       'Accept': 'application/json',
       'Authorization': 'ExtraHop apikey={API KEY}’
body = r"""{
  "cycle": "auto",
  "from": -1800000,
  "until": 0,
  "metric_category": "http_client",
  "metric_specs": [
    {
       "name": "req"
    }
   ],
   "object_ids": [
      9363
   ],
   "object_type": "device"
}"""
conn = httplib.HTTPSConnection('{HOST}')
conn.request('POST', '/api/v1/metrics', headers=headers, body=body)
resp = conn.getresponse()
print resp.status, resp.reason
print resp.read()
```

The following response shows entries for the device with ID 9363:

```
{
 "date": "Thu, 19 Nov 2015 23:20:07 GMT",
 "via": "1.1 localhost",
 "server": "Apache",
 "vary": "Accept-Encoding",
 "content-type": "application/json; charset=utf-8",
 "cache-control": "private, max-age=0",
 "connection": "Keep-Alive",
 "content-encoding": "gzip",
 "keep-alive": "timeout=45, max=44",
 "content-length": "277"
}

{
 "stats": [
   {
    "oid": 9363,
    "time": 1447973460000,
    "duration": 30000,
    "values": [
     2
    ]
   },
```

```
   {
    "oid": 9363,
    "time": 1447973490000,
    "duration": 30000,
    "values": [
     0
    ]
   },
   {
    "oid": 9363,
    "time": 1447973520000,
    "duration": 30000,
    "values": [
      1
     ]
    },
    {
     "oid": 9363,
     "time": 1447973550000,
     "duration": 30000,
     "values": [
       2
      ]
     }
```

Example 4: Create, retrieve, and delete an object

This example shows how you can create and successfully retrieve information about a device tag. Then, after the device tags are deleted, the example shows how an attempt to retrieve information subsequently fails.

The following example shows how to create a device tag called my_test_tag.

```
curl -i -X POST --header "Content-Type: application/json" \
--header "Accept: application/json" \
--header "Authorization: ExtraHop apikey={API KEY}" \
-d "{
\"name\": \"my_test_tag\"
}" "https://{HOST}/api/v1/tags"
```

A 201 status returns upon success with the following response headers, which display that the tag was created, and provides the device tag location and ID of /api/v1/tags/1.

```
{
 "date": "Wed, 18 Nov 2015 20:24:13 GMT",
 "via": "1.1 localhost",
 "server": "Apache",
 "content-type": "text/plain; charset=utf-8",
 "location": "/api/v1/tags/1",
 "cache-control": "private, max-age=0",
 "connection": "Keep-Alive",
 "keep-alive": "timeout=45, max=88",
 "content-length": "0"
}
```

Next, the ID (1) is added to the following GET request, which returns a 200 status upon success and the JSON representation of the retrieved tag:

```
curl -i -X GET --header "Accept: application/json" \
--header "Authorization: ExtraHop apikey={API KEY}" \
```

```
"https://{HOST}/api/v1/tags/1"
{
 "mod_time": 1447878253953,
 "id": 1,
 "name": "my_test_tag"
}
```

Next, the following example shows a DELETE request to remove the device tag from the system, which returns a 204 status upon success:

```
curl -i -X DELETE --header "Accept: application/json" \
--header "Authorization: ExtraHop apikey={API KEY}" \
"https://{HOST}/api/v1/tags/1"
```

Finally, when another GET request is sent for that deleted device tag, the operation fails, and a 404 status is returned upon failure, indicating that the tag is no longer available.

```
curl -i -X GET --header "Accept: application/json" \
--header "Authorization: ExtraHop apikey={API KEY}" \
"https://{HOST}/api/v1/tags/1"
```

Example 5: Query the record log

The following example shows how you can query the record log to retrieve 100 HTTP records where the method is GET and the status code is 404.

```
{
  "filter": {
    "operator": "and",
    "rules": [
        {
            "field": "method",
            "operand": "GET",
            "operator": "="
        },
        {
            "field": "statusCode",
            "operand": "404",
            "operator": "="
        }
   ]
 },
 "from": -900000,
 "limit": 100,
 "types": [
   "~http"
 ]
}
```