Triggers

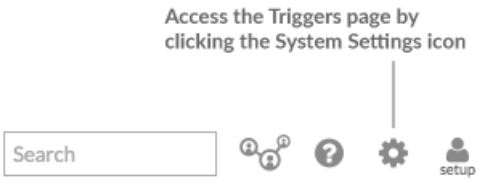
Published: 2018-11-07

Triggers are composed of user-defined code that automatically runs on system events through the ExtraHop Trigger API. You can write a trigger, which is a block of JavaScript, through the trigger API to extract, store, and visualize custom wire data events and metrics that are specific to your business, infrastructure, network, clients, and business applications.

Some of the most common workflows that you can perform through triggers include the following operations:

- Create an application container in which metrics are collected for specific devices. Application containers augment the device-based views that the ExtraHop system constructs by default.
- Create custom metrics and save them to the ExtraHop datastore. For example, user agent data generated by an HTTP request is not a metric built into the ExtraHop system. However, the ExtraHop Trigger API provides a user agent HTTP property, which enables you to write a trigger that collects user agent data as a custom metric.
- Generate records and write them to the ExtraHop Explore appliance for long-term storage and retrieval.
- Send data to syslog consumers, such as Splunk, or to third party databases, such as MongoDB or Kafka, through an open data stream.
- Perform universal payload analysis (UPA) to access and parse TCP and UDP payloads from unsupported protocols.
- Initiate packet captures to record individual flows based on user-specified criteria. Your ExtraHop system must be licensed for packet capture to access this feature.

In the ExtraHop Web UI, the Triggers page lists information about available triggers and provides access to the Trigger Configuration window, where you can write or modify triggers.



Plan a trigger

Writing a trigger to collect custom metrics is a powerful way to monitor your application and network performance. However, triggers consume system resources and can affect system performance, and a poorly-written trigger can cause unnecessary system load. Before you build a trigger, evaluate what you want your trigger to accomplish, identify which events and devices are needed to extract the data you need, and determine whether a solution already exists.

- Identify the specific information you need to collect, by asking the following types of questions:
  - When will my SSL certificates expire?
  - Is my network getting connections on non-authorized ports?
  - How many slow transactions is my network experiencing?
  - What data do I want to send to Splunk through an open data stream?
- Review the Metric Catalog to determine whether a built-in metric already exists that extracts the data you need. Built-in metrics do not create additional load on the system.
- Identify which system events produce the data that you want to collect. For example, a trigger that monitors cloud application activity in your environment might run on HTTP responses and on the open and close of SSL connections. For a complete list of system events, see the ExtraHop Trigger API Reference.

- Familiarize yourself with the API methods and properties available in the ExtraHop Trigger API Reference. For example, before you get too far in planning your trigger, check the reference to make sure that the property you want to extract is available, or to find out what properties are collected in a default CIFS record.
- Determine how you want to visualize or store data collected by the trigger. For example, you can view metrics on a dashboard or by protocol, you can send records to the ExtraHop Explore appliance, or you can send data to another third-party system, such as Splunk.
- Determine if a trigger already exists that meets your needs or might be easily modified; always start with a pre-existing trigger whenever possible. Search the following resources for an existing trigger:
  - Existing triggers on the Triggers page
  - The ExtraHop Solution Bundles Gallery
  - The ExtraHop Community Forums

Building triggers

If you determine that you need to build a new trigger, familiarize yourself with the following tasks that must be completed:

- Configure the trigger to provide details such as the trigger name and whether debugging is enabled. Most importantly, specify which system events the trigger will run on. For example, if you want your trigger to run each time an SSH connection is opened, you will specify `SSH_OPEN` as the trigger event.
- Write the trigger script, which specifies the instructions the trigger will carry out when a system event configured for the trigger occurs. The trigger script can provide instructions for a simple task such as creating a custom device count metric called "slow_rsp" or a more complex effort such as monitoring and collecting statistics about the cloud applications accessed in your environment.
- Assign the trigger to the devices the trigger will collect data from. For example, if the trigger is configured to run on `SSH_OPEN` events, the trigger will run only when those events occur on assigned devices. The trigger cannot run until it has been assigned to at least one device.

After the trigger is complete and running, it is important to check that the trigger is performing as expected.

- View the runtime log for expected output from debug statements in the trigger script. The log also displays any runtime errors and exceptions that you must fix.
- Monitor the performance cost by tracking the number of cycles consumed by the trigger.
- Check System Health charts for trigger exceptions, drops from the trigger queue, and unexpected activity.
- Check that the trigger script adheres to coding best practices.

Navigate triggers

The Triggers page contains a list of current triggers with the following information:

Name
: The user-defined name of the trigger.

Author
: The name of the user who wrote the trigger. Default triggers display ExtraHop for this field.

Events
: The system events that cause the trigger to run, such as `HTTP_RESPONSE`.

Type
: The type of metric source for the trigger, such as a device or a network.

Debug Mode
: Whether debugging is enabled. If debugging is enabled, output from debug statements in the trigger script are logged in the runtime log output.

ECA

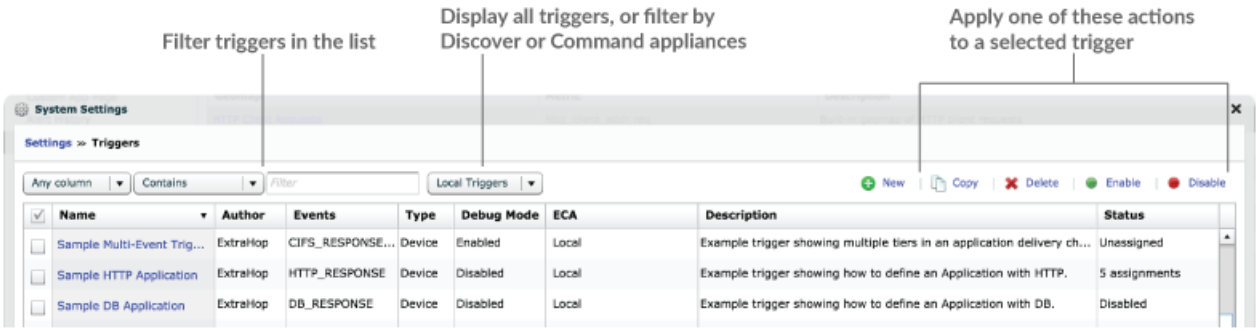
The appliance where the trigger was written. If the trigger was created on an ExtraHop Command appliance, the Command appliance name is displayed. Otherwise, this field displays Local to indicate that the trigger was written on the local Discover appliance. This column is only available from a Discover appliance that is connected to a Command appliance.

Description

The user-defined description of the trigger.

Status

Whether the trigger is enabled. If the trigger is enabled, the number of device assignments also displays.



Related topics

Check out the following guides and resources that are designed to familiarize new users with our top features.

- Build a trigger
- Learn how to build a trigger to collect custom metrics
- Learn how to build a trigger to monitor responses to NTP monlist requests