ExtraHop 5.2
ExtraHop Trigger API Quick Start Guide

Published: 2018-02-06

Application Inspection triggers are composed of user-defined code that automatically executes on system events through the ExtraHop trigger API. By writing triggers, you can collect custom metric data about the activities on your network. In addition, triggers can perform operations on protocol messages (such as an HTTP request) before the packet is discarded.

The ExtraHop system monitors, extracts, and records a core set of Layer 7 (L7) metrics for devices on the network, such as response counts, error counts, and processing times. After these metrics are recorded for a given L7 protocol, the packets are discarded, freeing resources for continued processing.

Triggers enable you to:

- Generate and store custom metrics to the internal datastore of the ExtraHop system. For example, while the ExtraHop system does not collect information about which user agent generated an HTTP request, you can generate and collect that level of detail by writing a trigger and committing the data to the datastore. You can also view custom data that is stored in the datastore by creating custom metrics pages and displaying those metrics through the Metric Explorer and dashboards.
- Generate and send records to an Explore appliance for long-term storage and retrieval.
- Associate an arbitrary cross-section of monitored traffic with an ExtraHop application container to enable tier-by-tier application-based views of data. Application views augment the device-based views that the ExtraHop system constructs by default.
- Generate custom metrics and send the information to syslog consumers such as Splunk, or to third party databases such as MongoDB or Kafka.
- Initiate a packet capture to record individual flows based on user-specified criteria. You can download captured flows and process them through third-party tools. Your ExtraHop system must be licensed for packet capture to access this feature.

Related documentation

This guide is designed to provide a general foundation for writing triggers. For more information about writing triggers, refer to the following documentation:

- Sample triggers available at https://forums.extrahop.com/
- ExtraHop Trigger API Reference
- Planning a Trigger training
- Creating a Simple Trigger training
- Creating a Multi-Event Trigger training
- Creating and Using App Containers training

Plan a trigger

Before you create a trigger, you should have a clear idea about the information you want the trigger to collect. Start by answering the following three questions:

- Which devices require further processing?
- Which events on your target devices are you interested in further processing?
- How do you want to process these events?

For example, you might answer these questions with the following statement:

When web servers web01, web02, and web03 (devices) receive an HTTP request (event) from subnet 10.10.1.0/24, record the information to the ExtraHop datastore about the user agent (process).

One of the easiest ways to write a trigger is to start with a broad statement and narrow that statement down until you have a clear idea of exactly what you are looking for. For example, you could start out with a broad statement like:

I want to know which of my internal-facing web servers are returning an excess number of page not found errors.

Then, add the specific internal-facing web servers to the statement:

I want to know how many page not found errors are being returned by web01, web02, and web03.

Then, add the specific system event and process to the statement:

I want to know how many HTTP Response events from web01, web02, and web03 have http.statusCode of 404.

Create a trigger

ExtraHop triggers are written in a JavaScript-like syntax. You can access the Trigger Configuration settings through the ExtraHop Web UI. Custom metrics are captured based on triggers and, after the information is collected, those metrics are available for display on an HTML dashboard or custom page.

The Trigger Configuration window has three tabs (Configuration, Editor, and Assignment). After these three tabs are populated, two additional panels (Runtime Log and Performance) become available. These additional panels can assist with debugging and analyzing the performance of the trigger.

For more information, see the ExtraHop Trigger API Reference.

1. In the ExtraHop Web UI, click the **System Settings** icon, and then click **Triggers**.
2. Click **New**.
3. In the Configuration pane, provide the following information:
   - **Name**: Type a name for the trigger. ExtraHop recommends that you do not use a dynamic trigger name. Dynamic content should be placed in the key label fields, not in the name of the trigger itself.
   - **Author**: Type the author name for the trigger.
   - **Description**: Type a description for the trigger.
   - **Priority**: Select a priority to assign to the trigger. When multiple triggers fire on the same event, the order in which they fire is determined by the assigned priority value. Higher priority numbers represent higher priority. For example, a trigger with a priority value of 9 executes before a priority 0 trigger. If the priorities are the same, triggers are executed in alphabetical order by name.
   - **Status**: Click the **Disable Trigger** checkbox to make the trigger inactive on its assigned objects.
   - **Debugging**: Select the **Enable Debugging** checkbox to enable debugging for your trigger. Select this option while your trigger is being developed but clear this option after you have verified that the trigger is working. A large number of debug statements can cause both performance issues and excessive debug statements in the Runtime Log.
   - **Events**: Start typing an event name, and then select one or more events from the drop-down list that you want the trigger to process.
   - **Show advanced options**: Click this to set additional event-specific options. Not all events will have advanced options.
4. In the Editor pane, enter the source code for your trigger. Click the API Reference link to open the Trigger API Reference in a new browser tab.

   Tip: Click the **API Reference** link at the top of the window to open the ExtraHop Trigger API Reference in a new browser tab.
5. Click **Save**.

6. Click **Cancel** to exit the Trigger Configuration window.

Your trigger is now saved and the corresponding custom metrics have been created in the Metric Catalog.

Next steps
You must refresh the page to see your new trigger in the trigger list.

Example: HTTP response trigger

The following example of a simple HTTP_RESPONSE trigger records a custom metric that counts all HTTP status codes greater than or equal to 400.

By default, the ExtraHop system records statistics about each status code individually, but not in this particular grouping. Therefore, for statistics about this particular grouping, a trigger and a custom metric are required.

```
if (HTTP.statusCode >= 400) {
Device.metricAddCount("http_4xx_5xx", 1);
}
```

Once captured, the custom metric is available for display on an HTML dashboard or on a custom page.

Assign trigger to devices

After you create a trigger, the trigger must be assigned to one or more devices before the trigger can begin gathering data.

You also can assign the trigger to a device group, which assigns the trigger to each device in the group.

Note: Avoid assigning any trigger to all devices. Only assign a trigger to devices you want to actively monitor with the trigger.

1. In the ExtraHop Web UI, click **Metrics** in the main navigation bar, then click **Sources** > **Devices** in the left pane.
2. Select the check box for each device you want to assign the trigger to.
3. In the **Select Action** drop-down list, select **Assign to Trigger**.
4. Select the check box for the trigger you want to assign to the selected devices.
5. Click **OK**.

The trigger will execute on the selected devices whenever the trigger event occurs.

Edit a custom metric in the metric catalog

After you create your trigger, you need to edit your entries in the Metric Catalog to add the following information:

- Update the display name to make the trigger more meaningful when used in a dashboard.
- Edit the description to improve understanding of the metric.
- Associate base and detail metrics.
- Add key labels to indicate how base metrics and detail metrics are related.

1. In the ExtraHop Web UI, click the **System Settings** icon, and then click **Metric Catalog**.
2. In the filter text box, enter the name of the metric you want to edit and select it from the search results.
   All custom metrics will begin with "Custom -".
3. In the Parameters section, do not modify any information as changes can potentially cause your custom metric to stop functioning correctly.
4. In the Display section, modify the following metric fields:

- **Name**: Update the name to make the trigger more meaningful when it is used in things like a dashboard. The name field is initially populated based on the trigger script you entered.
- **Units**: Ensure that the correct units are selected. If not, click the **Units** drop-down and select the correct units for your metric.
- **Description**: Enter a description for your metric. This is particularly important for complicated metrics. The default value for this field is seen on <xx-xx-xx> in trigger <trigger name> where <*xxxx- xx*> is the date your trigger was created and <*trigger name*> is the name you assigned to your trigger when you created it. This description is displayed in the Web UI
- **Key**: Enter a key label to help determine how the base metric and detail metrics are related. This field is only available for detail metrics.

5. In the Detail Relationships section, modify the following metric fields:

   - **Detail Metrics**: Click **Add Detail Metric** and select the appropriate custom detail metrics from the list. This field is only available for base metrics.
   - **Base Metric**: Click **Set Base Metric** and select the appropriate custom base metric from the list. This field is only available for detail metrics.

   You can only associate custom detail and custom base metrics to each other.
6. Click **Update** to save your changes to the metric catalog.

View your custom metric on a dashboard

You can add both base and detail custom metrics to a dashboard and view them alongside built-in metrics.

1. Create a new dashboard OR select **Dashboards** from the top menu, select the dashboard on which you want to place the widget for your custom metric from the dashboard Dock and click **Edit Layout** from the command menu.
2. Click the **Region** icon and drag in onto the dashboard. You can resize the region by dragging the lower right corner.
3. Click the **Chart** widget icon and drag it onto the region. You can resize the widget by dragging the lower right corner.
4. Click the command menu button in the top right corner of the widget and select **Edit**.
5. Optionally, click **Chart** in the title bar and rename the chart, and then press enter to save the name.
6. Click the **Add Metric Source** button and enter all or part of your custom metric name.
7. Select your metric from the list. A preview chart appears on the right.
8. In the upper left corner, click **Options** to change the units, show the metric as a rate, or change the suffix notation.
9. In the upper right corner, select the time interval to display the metrics for.
10. Select a chart type at the bottom to change the appearance of the chart.
11. Click **Save**.
12. Click **Exit Layout Mode** in the upper right corner to return to the Dashboards screen.

Example: View metrics with an application container

Application containers are pre-configured page templates that are added by triggers. Use the `application.commit()` method to add or populate an application container. The application container page displays data only for devices that have been committed to that application by a trigger.

The following example script and task adds and populates an application container page called Finance App, which includes all HTTP responses that contain the string "finance".

```
if (HTTP.uri.indexOf("finance") > -1) { Application("Finance
 App").commit(); }
```

In order for this trigger to run properly, the event must be set to `HTTP_RESPONSE` and the trigger must be assigned to any HTTP server that runs Finance App.

1. Click **Metrics** in the top menu, then click **Sources** > **Applications** in the left pane.
2. Select **Finance App** from the list of applications.
3. Click on a top-level metric in the left pane to see detailed metrics that apply specifically to this application in the content pane.

Create a custom page

You can display recorded metrics on custom pages with graphs and charts. After you create a new custom page, you must assign devices to it and add charts to it. The content displayed on these charts comes from your trigger. Newly created custom pages are blank until you add charts to them.

1. Click the **System Settings** icon, then click **Pages**.
2. Click **New**.
3. Provide the following information:
   - **Name**: Type a name for your custom page.
   - **Author**: Type an author name for your custom page. This field defaults to your username.
   - **Page Type**: Select the page type from the drop-down list. The default selection is Device.
   - **Description**: Type an optional description of your page.
4. Click **OK**.

Next steps
Assign the custom page to a device and add charts to it.

Assign a custom page to a device

After you create a new custom page, you must assign devices to it .

1. Click **Metrics** in the top menu, then click **Sources** > **Devices** in the left pane.
2. Select the devices you want to assign the custom page to.
3. In the **Select Action** drop-down menu, select **Assign Page**.
4. Select the custom page you want to assign the selected devices.
5. Click **OK**.

Add a chart with built-in metrics to a custom page

After you create a new custom page, you must assign devices to it and add charts to it. The content displayed on these charts comes from your trigger. Newly created custom pages are blank until you add charts to them.

Before you begin
Note the following about metrics and charts:

- Metrics are of a specific type. If your metric is specified as type Count using the `metricAddCount()` method, your metric is returned in a metric search only if the Metric Type is set to Count. If your metric is of type Dataset, Sampleset, etc., your metric will be returned only if the type selected in the Metric Type drop-down matches the type of your custom metric.
- The chart displays only metrics with at least one data point. If your metric is specified inside a trigger but has not yet received any data, it will not be displayed. However, you can manually enter your metric name in the Chart Configuration window and the metric will display on your chart with a value of 0. When the metric begins to collect data, the value automatically updates to reflect the new information.

- Custom metrics and custom detail metrics are added to devices using the `metricAdd()*` methods. For example:

```
Device.metricAddCount("http_error", 1);
 Device.metricAddDetailCount("http_error_detail", http_status.toString(),
```

1. Click **Metrics** in the top menu, then click **Sources** > **Devices** in the left pane.
2. Click a device that the custom page is assigned to.
3. Click the custom page name in the left pane.
4. On the custom page, click **Edit Page**.
5. Click **Add Chart**.
6. Provide the following information:
   - **Title**: Type a title for the chart.
   - **Metric Source**: From the Metric Source drop-down, select either Built-In or Trigger.
   - **Metric Type**: Select the type of metric from the Metric-Type drop-down.
   - **Chart Type**: Select the chart type from the Chart Type drop-down.
   - **Units Label**: Type a label for the units of measure applied in the chart. (Only applies to some metric and chart types.)
   - **Percentiles**: Select the percentiles from the drop-down. (Only applies to some metric and chart types.)
   - **Use logarithmic scale**: Select if the chart should apply a logarithmic scale. (Only applies to some metric and chart types.)
7. In the Metrics section, click **Find**.
8. Click the metric to include in the chart's top-level view from the list in the Available Metrics pane.
9. Optionally, click inside the **Detail Metric Name** field, then click the detail metric to include in the chart's detail view from the list in the Available Metrics pane.
10. Click **OK**.
11. Click **OK** again.

Troubleshoot triggers and custom pages

The following information might help you solve common problems with triggers and provide optimization tips.

I don't see a new button on my trigger page. How do I create one?

A **New** button is available only when you are logged in to the ExtraHop appliance with an account that has write permissions. You cannot create triggers without write permissions. You must either log in with another account that has write permissions or contact your ExtraHop administrator to enable write permissions for your account.

How can I tell if my trigger is being executed?

To see when the trigger was last executed:

1. With the **Enable Debugging** check box selected for the trigger, click the **System Settings** icon, and then click **Triggers**.
2. Click the trigger you want to view, then click the **Performance** tab. If your trigger is executing, you will see a non-zero display of the CPU cycles being used by the trigger. To ensure functionality with the **Enable Debugging** check box selected for the trigger, you can use the `debug()` function within the trigger to output debug statements to the Runtime Log, then verify those statements have been written to the Runtime Log.

How can I debug my trigger?

With the **Enable Debugging** check box selected, use the debug() statement extensively when developing and testing your trigger to output debug statements to the Runtime Log. You can use these debug statement to ensure that your trigger runs and generates the data you expect. This debug output appears in the Runtime Log tab for the trigger. Each unique debug statement is printed only once.

After you verify that your trigger performs as expected, remove the debug() statements. Disable all debugging and optimize performance by clearing the **Enable Debugging** check box in the Trigger Configuration window.

Why don't my custom metrics appear on my custom page?

The custom page type must match the custom metric type for metrics to be displayed.

Metrics only display on a page when they contain at least one data point. A custom metric that has not received any data points will not appear on your page. However, you can manually enter the metric name so it appears on your custom page with a value of 0. When the ExtraHop system begins collecting samples for your metric, this number will update.

How can I improve the performance of my trigger?

ExtraHop recommends the following practices to improve trigger performance:

- Assign triggers only to devices that require the collection of custom metrics. Assigning triggers to all devices causes unnecessary trigger executions that may degrade system performance.
- Monitor how frequently your triggers execute by using the Performance tab in the Trigger Configuration window. ExtraHop recommends optimizing triggers that execute frequently.
- Remove all debug() statements after you test your trigger and know that your trigger works correctly. You can also disable all debugging for your trigger.
- Avoid unnecessarily complex code, such as loops with a potentially high number of iterations, nested while and for loops, and inefficient regular expressions.
- Use exclusion logic so your trigger does the minimum amount of processing and object creation. For example, you can optimize the following trigger code:

```
var example1 = HTTP.cookies;
var example2 = HTTP.method;
if (example2 === 'POST') {
       // process cookies
}
```

to

```
var example1;
var example2 = HTTP.method;
if (example2 === 'POST') {
example1 = HTTP.cookies;
   // process cookies
}
```

- Always use `var` to declare variables before using them. For example, use `var example1 = HTTP.uri;` instead of `uri = HTTP.uri;`
- Cache array lengths before iterating. For example, you can optimize the following code:

```
for (i = 0; i < list.length; i++) { do something; }
```

to evaluate the length of the list once, as opposed to evaluating the list length each time through the loop.

```
var len = list.lengthfor (i = 0; i < len; i++) { do something; }
```

- Use strict equality (triple equals) whenever possible. For example:

```
if (some_thing !== null) { do something; }
```

- Do not put large objects into `Flow.store`. For example, instead of storing all of `DNS.answers`, keep only the information that will be used for future events:

```
Flow.store.ttl = DNS.answers[i].ttl;
```

- Clear `Flow.store` values when you no longer need them by setting the property to null. For example:

```
if (event === 'DB_REQUEST') {
   Flow.store.stmt = DB.statement;
} else if (
event === '
   DB_RESPONSE') {
   Device.metricAddCount('count', 1);
   Device.metricAddDetailCount('count', Flow.store.stmt, 1);
   Flow.store.stmt = null;
}
```

  This also helps to prevent errors in which the trigger reads previously recorded data from `Flow.store`.

  Note: There are separate flow stores for every flow, so how much you can store on each flow store depends on the number and the size of the values being stored in each flow store.

- Conserve CPU cycles by caching created applications that are used multiple times:

```
var myapp = Application('example');
// now commit like
myapp.commit();
myapp.metricAddCount('custom', 1);
```

- Avoid property lookups whenever possible, especially when using a property lookup more than once. For example, you can optimize the following code:

```
if (HTTP.uri.indexOf('example1') > -1) // content
else if (HTTP.uri.indexOf('example2') > -1) // content
```

- Cache information once and save the property lookups:

```
var uri = HTTP.uri;
if (uri.indexOf('example1') > -1) // content
else if (uri.indexOf('example2') > -1) // content
```

- Use `return` instead of `exit()`. The `exit()` global function is supported but it might cause the system to run slowly. For example, use `(!HTTP.uri) return;` instead of `HTTP.uri || exit();`
- The following are no longer valid as of version 3.10+:
  - `Application[...]` syntax
  - `foreach` syntax
  - E4X syntax extensions. Existing uses of new `XML()` are still valid via a native XML class.
  - Internal objects that previously compared equal to each other no longer compare equal. For example, `Flow.client.device == new Device(id)` is not valid.