Query records to find missing web resources

Published: 2022-01-05

When customers visit your website, a link that results in a "HTTP 404 - File not found" error message can be frustrating and might cause customers to leave your site without finding what they were searching for.

This walkthrough takes you through the steps of sending HTTP records to your recordstore, then drilling down on HTTP transaction metrics to discover the source of 404 errors and identify any missing resources on your web server.

Prerequisites

- Familiarize yourself with the concepts in this walkthrough by reading the Records topic.
- You must have access to an ExtraHop system that is connected to a recordstore.
- Your user account must have full write privileges to create a trigger.
- Your ExtraHop system must have network data with web server traffic and HTTP records that are being written to the recordstore. If you do not have access to web server data, you can perform this walkthrough in the ExtraHop demo.

Write a trigger to generate HTTP records

Before you can query for records, you must write a trigger to generate a record every time an HTTP response occurs on specified devices or networks.

Note: If you are performing this walkthrough in the ExtraHop demo, the trigger has already been created, and you can proceed to the Start a new query section.

1. Log in to the ExtraHop system through `https://<extrahop-hostname-or-IP-address>`.
2. Click the System Settings icon and then click **Triggers**.
3. On the Triggers page, click **New**.
4. Type a name for the trigger in the Name field. For this walkthrough, type `HTTP response`.
5. Select the **Enable debug log** checkbox to help you validate that the script is running correctly.
6. Click in the Events field and select **HTTP_RESPONSE**.
7. In the Assignments field, type the name or IP address of one of your HTTP servers and select the server from the search results.
8. Click the Editor tab.
9. In the Trigger Script editor, type the following code:

```
HTTP.commitRecord()
debug ("committing HTTP record")
```

`HTTP.commitRecord()` is the method of generating the HTTP records, and `"committing HTTP record"` is the text string that is written in the debug log when the trigger successfully commits the record.

10. Click **Save and Close**.

Start a new query

Now, you will create a new query to view all of the HTTP data received in the last 24 hours.

1. Click on the Global Time Selector, select **Last 1 day** and then click **Save**.

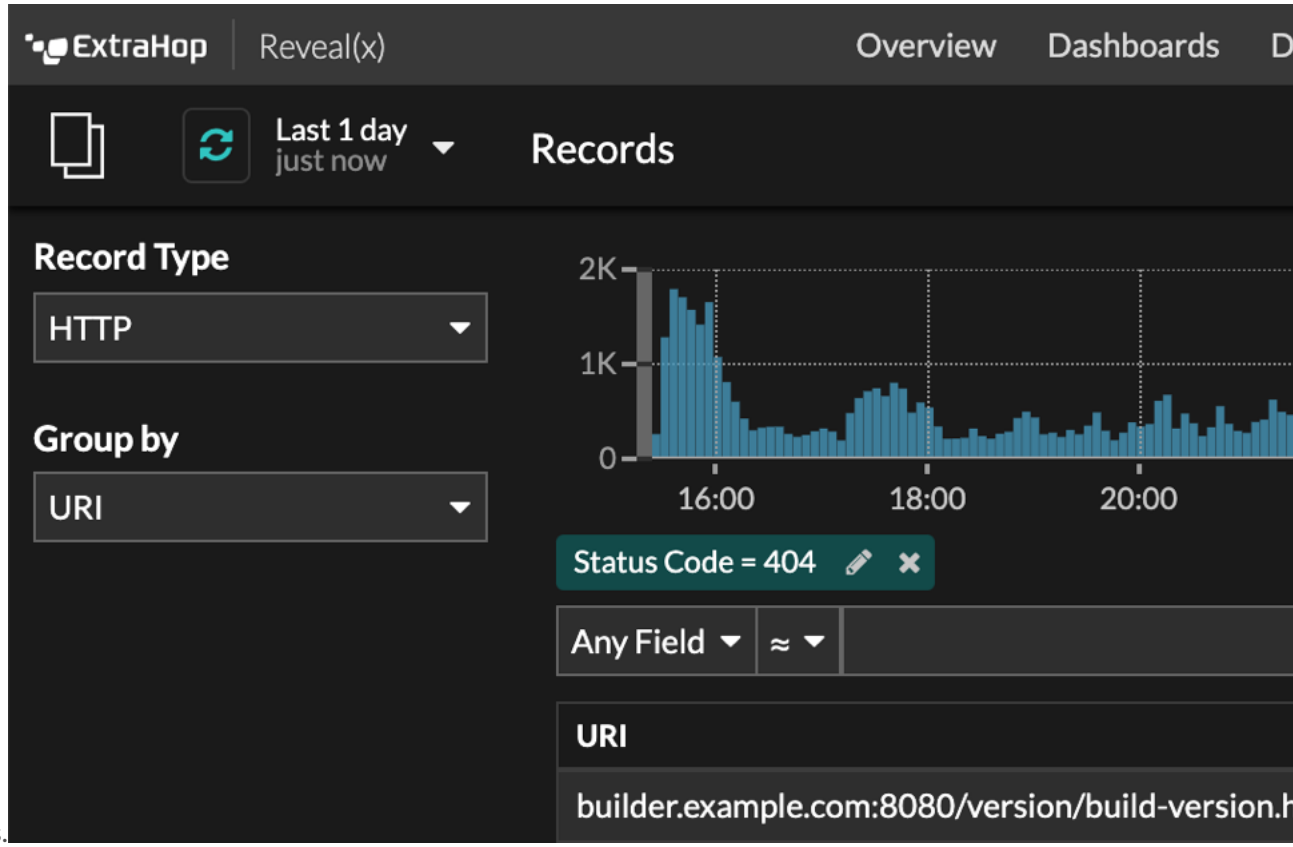
2. Click **Records**.
3. From the **Any Record Type** drop-down list, select **HTTP** and then click **Save** to filter the results of your query to only display the metrics related to HTTP records.
4. Click **View Records**.

Refine results

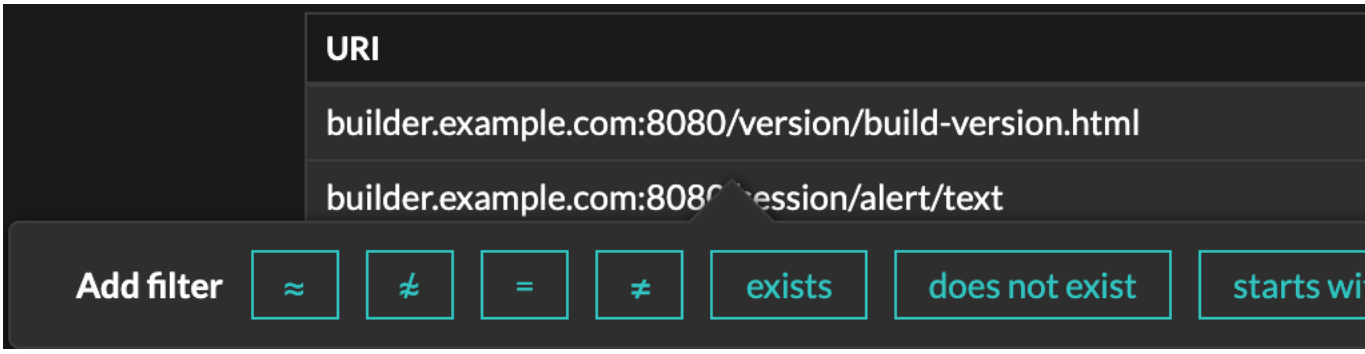
Refine the results further to get a clearer picture of which server is supposed to store the requested resource, the client that is requesting the resource, and finally the path to where the resource should be located.

1. Click **404** in the Status Code section in the left pane.
2. From the Group By drop-down list in the left pane, select **URI**.

   You now have a list of URIs that are returning 404 errors. In the figure below, the `builder.example.com:8080/version/build-version.html` URI appears to be problematic, recording over 45,000 errors.

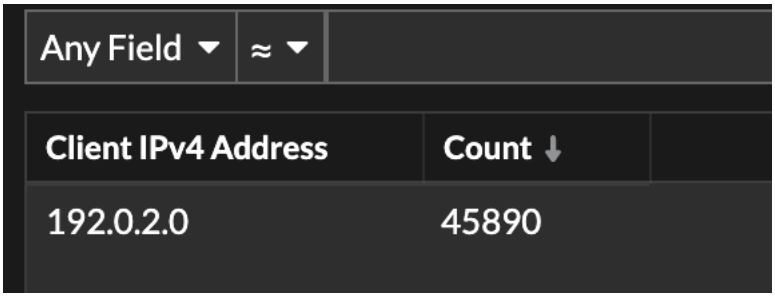


3. Click the URI with the highest count of 404 errors and then click the equals sign (**=**) to add the URI as a filter.



4. Find the client or clients that are making the request for that URI.

   From the Group By drop-down list, select **Client IPv4 Address**. From this result, you can see that only one client is requesting this URI that is returning a 404 status code.



Interpreting results

So, what do you know now? With a few simple clicks, you were able to drill down to find a specific client that was requesting a specific URI from a specific server. You now have the information to track the errors back to the source and resolve the 404 error.