Query records to find missing web resources

Published: 2018-04-18

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.

If you just deployed your ExtraHop Explore appliance, you are not going to have HTTP records to query. This walkthrough takes you through the steps of sending HTTP records to your Explore appliance, 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 concepts topic.
- You must have access to an ExtraHop Discover appliance that is connected to an Explore appliance or cluster.
- Your user account must have full write privileges to create a trigger.
- Your ExtraHop appliance must have network data with web server traffic and HTTP records that are being written to the Explore appliance. 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. In the Web UI, click the System Settings icon and then click **Triggers**.
2. On the Triggers page, click **New**.
3. Type a name for the trigger in the Name field. For this walkthrough, type `HTTP response`.
4. Select the Enable Debugging checkbox to help you validate that the script is running correctly.
5. Click in the Events field and select **HTTP_RESPONSE**.
6. Click the Editor tab.
7. In the Trigger Script editor, type the following code:

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

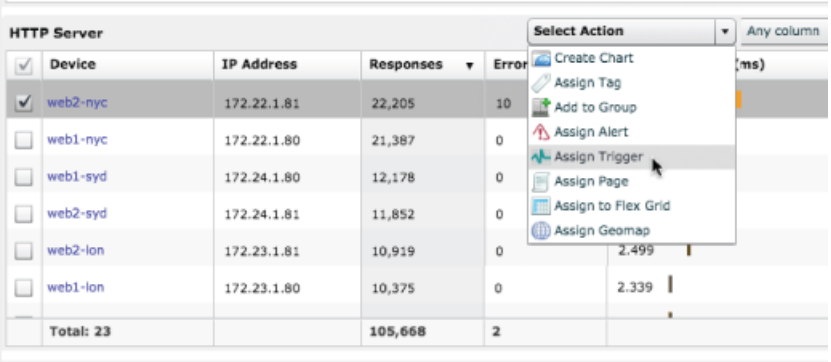
`HTTP.commitRecord()` is the method of generating the HTTP records, and `"committing HTTP record"` is the text string that is written in the debug log when the trigger successfully commits the record.

8. Click **Save and Close**.

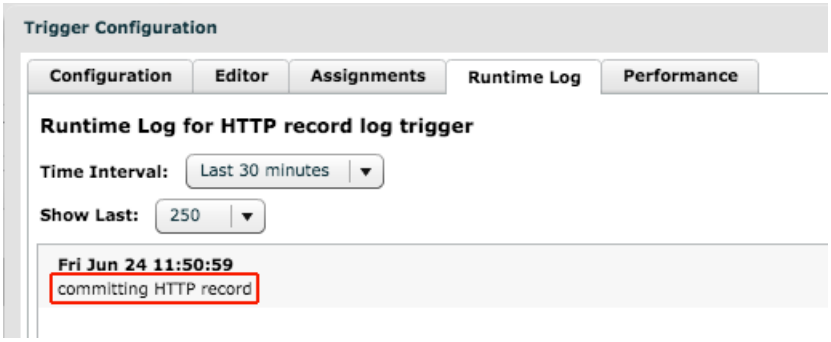
Assign the trigger to an HTTP server

Next, you will assign the trigger to a web server in an activity group on your network that you want to collect HTTP records for.

1. Click **Metrics**.
2. In the left pane, click **Activity Groups**.
3. In the content pane, click **HTTP Servers**.
4. Select one of your HTTP servers in the HTTP Server list.
5. In the Select Action drop-down menu, select **Assign Trigger**.



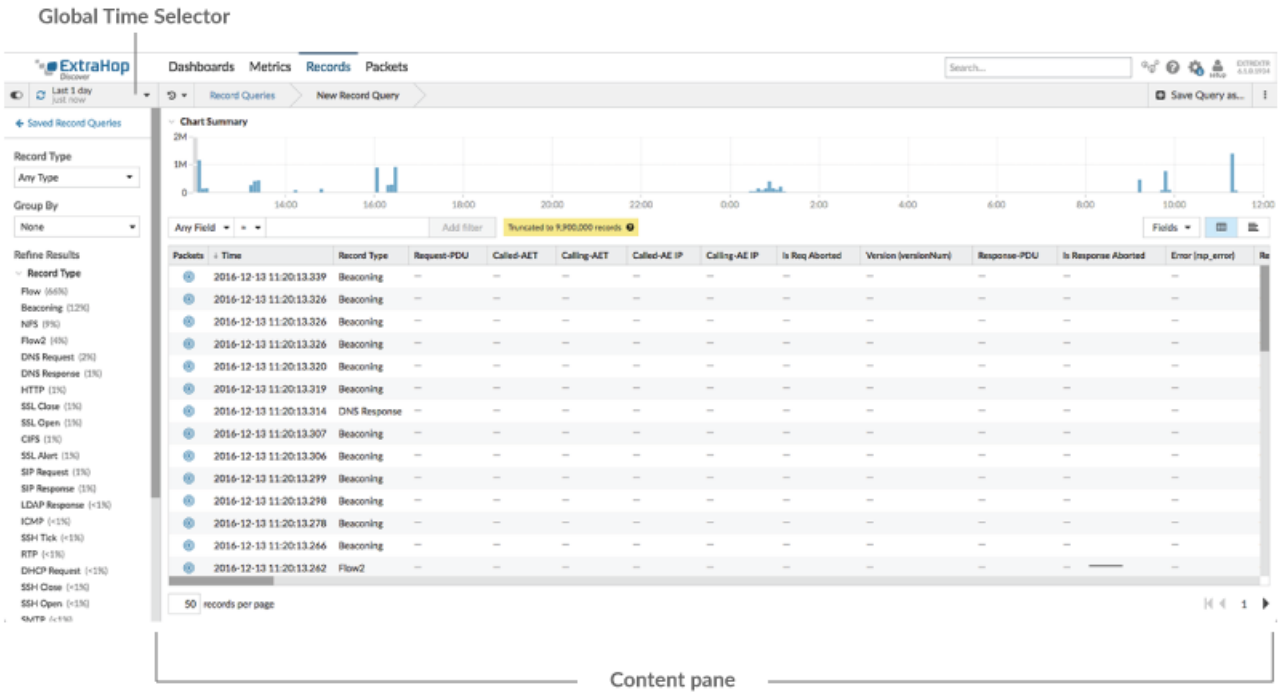
6. In the Assign Triggers dialog box, select the checkbox next to the trigger you created and then click **OK**.
7. Verify that the trigger is assigned to the web server by returning to the Triggers page in System Settings, clicking your trigger, and then clicking the Assignments tab. The web server should be listed in the Assignments section.
8. Next, verify that your trigger is generating HTTP records by clicking the Runtime Log tab. If the trigger is working correctly, you should see a `committing HTTP record` entry similar to the following:



Start a new query

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

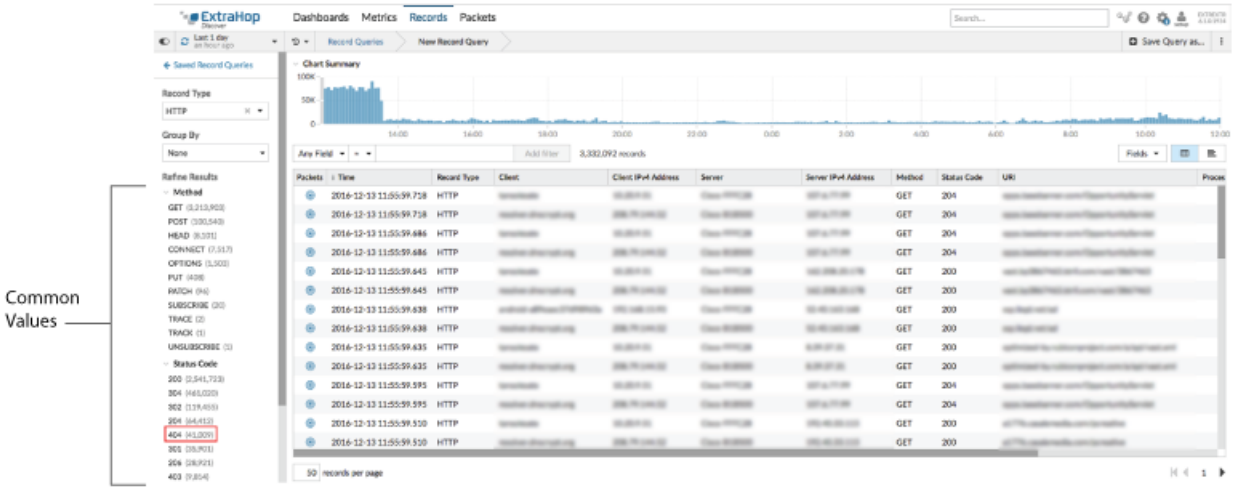
1. Click on the Global Time Selector, select **Last day** and then click **Save**.
2. Click **Records**. The query results for all records appear in the content pane.



Filter for HTTP traffic

Next, filter the results of your query to only display the metrics related to HTTP records.

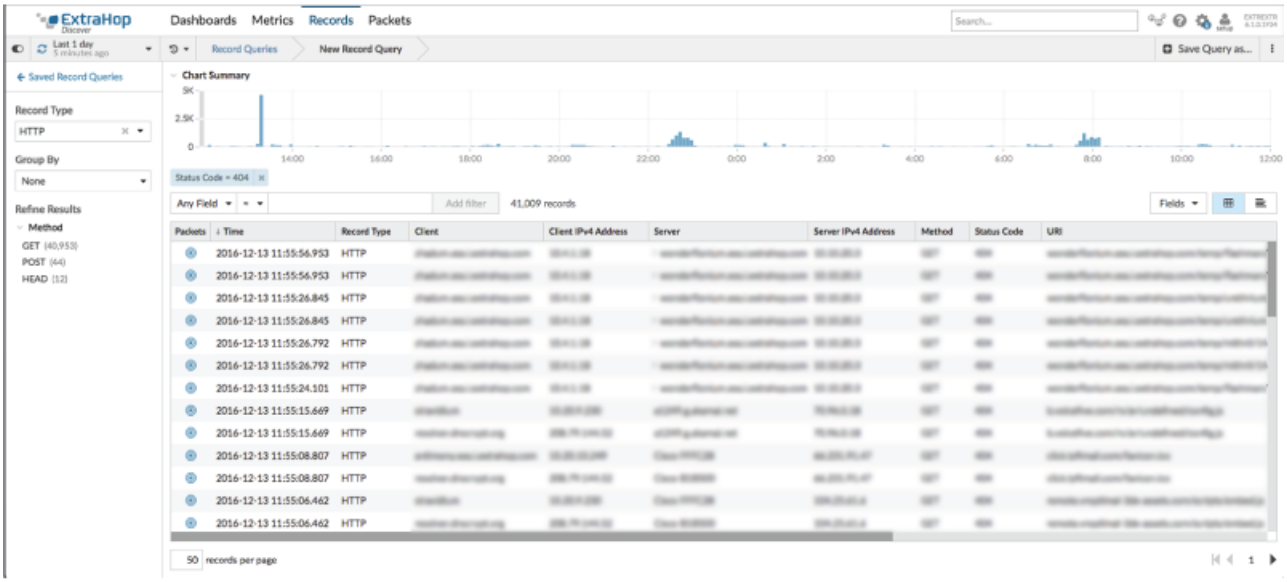
1. From the Record Type drop-down menu, select **HTTP** and then click out of the field. The content pane updates to display the HTTP transactions and in the left pane, the most common values for Method and Status Code appear.
2. The following figure shows the results of the query. In this example, there are 41,009 records with a 404 status code.



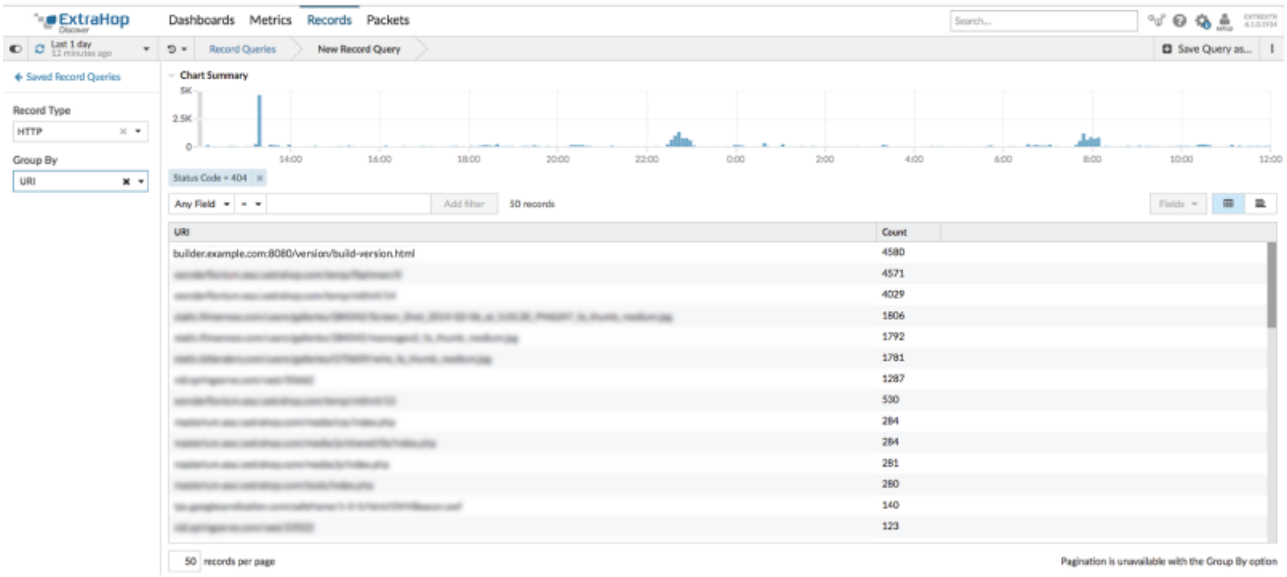
Refine results

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.

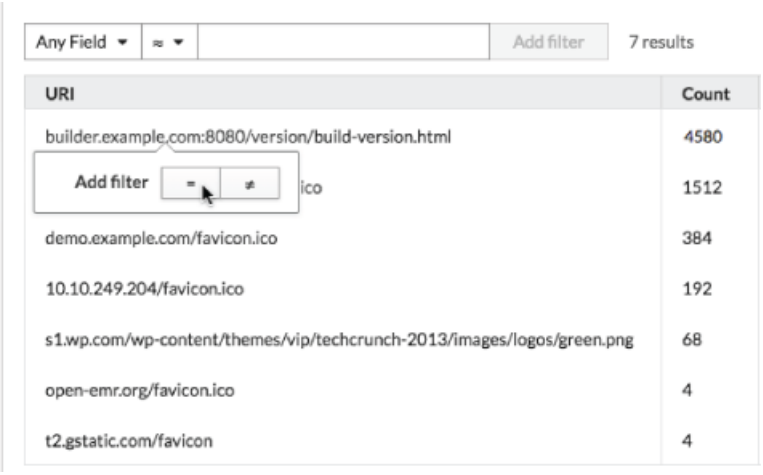
1. Click **404** in the Status Code section in the left pane. Results similar to the following appear in table view.



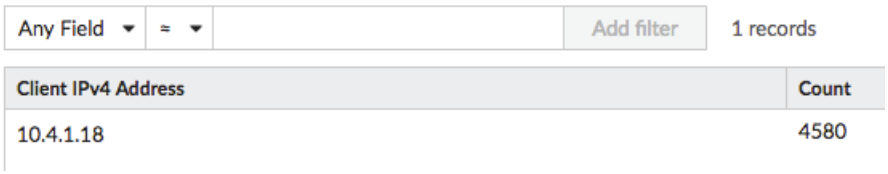
2. From the Group By drop-down list in the left pane, select **URI**. You now have a list of URIs that are returning 404 errors. In the figure below, the `builder.example.com:8080/version/build-version.htm` URI appears to be problematic, recording over 4,500 errors.



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.