How to use Helix test log search

This functionality is accessible through an endpoint in Helix Services at `/logs/search`

Input

All of the following are required parameters: - `repository`: The error string the program will search for - `searchString`: The public repository whose test logs will be parsed. The program will only search for non-internal test logs. - `startDate`: The start date of the date range in which the test logs will be searched. This argument must be in the format yyyy/MM/dd. For example, 2022-05-10. The date must be within the last 14 days. - `endDate`: The end date of the date range in which the test logs will be searched. This argument must be in the format yyyy/MM/dd. For example, 2022-05-10. The date must be within the last 14 days. - `responseType`: This argument is one of "Hits" or "HitsPerFile". The string must exactly match. This determines the type of response returned (sample responses can be found in the Outputs section below).

Note that the `repository` and `searchString` values should be URL encoded when they are passed in through the browser.

Example:

`http://localhost:8080/logs/search/dotnet%2Fruntime/Remote%20process%20failed%20with%20an%20u`

Output

Hits

Here is sample output for a `Hits` response. Each item in the hits array corresponds to a string match in a log file.


```
{
"filter": {
    "repository": "dotnet/runtime",
    "errorString": "Microsoft.DotNet.RemoteExecutor.RemoteExecutionException : Remote proces
    "startDate": "2022-07-15T07:00:00Z",
    "endDate": "2022-07-21T07:00:00Z",
    "responseType": "Hits"
},
"hits": [
    {
    "lineContent": "Microsoft.DotNet.RemoteExecutor.RemoteExecutionException : Remote proces
    "lineNumber": 29,
    "jobId": 20180433,
    "friendlyName": "System.Net.Security.Tests",
```

```
    "status": "Fail",
    "started": "2022-07-15T12:00:20.807Z",
    "finished": "2022-07-15T12:01:18.036Z",
    "consoleUri": "https://helixre107v0xdeko0k025g8.blob.core.windows.net/dotnet-runtime-ref
    "queueName": "osx.1015.amd64.open",
    "attempt": 1
    },
    ...
],
"occurrenceCount": 4,
"filesCount": 4
}
```


HitsPerFile

Here is sample output for a `HitsPerFile` response. Each item in the hits array corresponds to one log file.


```
{
"filter": {
    "repository": "dotnet/runtime",
    "errorString": "Microsoft.DotNet.RemoteExecutor.RemoteExecutionException : Remote proces
    "startDate": "2022-07-15T07:00:00Z",
    "endDate": "2022-07-21T07:00:00Z",
    "responseType": "HitsPerFile"
},
"hits": [
    {
    "occurrences": 1,
    "jobId": 20214225,
    "friendlyName": "System.Net.Security.Tests",
    "status": "Fail",
    "started": "2022-07-20T12:04:48.094Z",
    "finished": "2022-07-20T12:05:47.138Z",
    "consoleUri": "https://helixre107v0xdeko0k025g8.blob.core.windows.net/dotnet-runtime-ref
    "queueName": "osx.1015.amd64.open",
    "attempt": 1
    },
    ...
],
"occurrenceCount": 4,
"filesCount": 4
}
```

Other notes

- There is a timeout of 120 seconds/2 minutes on the whole program. If the query and parsing time out, the program throws an error and will not return any data.

Was this helpful?