# Export logs for Machine Learning Service API interactions

Published: 2023-07-26

You can configure sensors and consoles to export logs of API interactions with the ExtraHop Machine Learning Service. The ExtraHop system exports API logs through HTTPS POST requests. Any HTTP server can receive the logs, as long as the server is reachable by the ExtraHop system and the server has a TLS certificate installed.

## Configure an HTTP server to receive the logs

Before you configure the ExtraHop system to export API logs, you must configure an HTTP server to receive and record the logs. This topic demonstrates how to configure an example Go application that is available in the ExtraHop code-examples GitHub repository .

**Before you begin**

* You must install Go on your machine. For more information, see the Go documentation at https://go.dev/learn/ .

1.  Generate a server certificate for the HTTP server.

    > **Note:** If you already have a certificate, you can skip this step.

    a)  Run the following command to generate the certificate authority (CA) key:

    ```
    openssl genrsa –aes256 –out serverca.key 4096
    ```

    b)  Run the following command to generate the CA certificate:

    ```
    openssl req –new –key serverca.key –x509 –out serverca.crt –days 3650
    ```

    c)  Run the following command with your variables to generate a certificate signing request:

    ```
    openssl req -new \
        -nodes \
        -newkey rsa:4096 \
        -keyout server.key \
        -out server.req \
        -batch \
        -subj "/C=US/ST=WA/L=Seattle/O=ORGANIZATION_NAME/OU=router/
    CN=SERVER_URL" \
        -reqexts SAN \
        -config <(cat /etc/ssl/openssl.cnf <(printf
     "[SAN]\nsubjectAltName=DNS:SERVER_URL,IP:SERVER_IP_ADDRESS"))
    ```

    Replace the following variables in the command above:

    * **SERVER_IP_ADDRESS:** The IP address of your server.

    * **SERVER_URL:** The URL of your server.

    * **ORGANIZATION_NAME:** The name of your organization.

    d)  Run the following command with your variables to generate the server certificate:

    ```
    openssl x509 -req \
        -in server.req \
        -CA serverca.crt \
        -CAkey serverca.key \
    ```

```
      -CAcreateserial \
      -out server.crt \
      -days 3650 \
      -sha256 \
      -extfile <(printf
   "subjectAltName=DNS:SERVER_URL,IP:SERVER_IP_ADDRESS")
```

Replace the following variables in the command above:

- **SERVER_IP_ADDRESS:** The IP address of your server.

- **SERVER_URL:** The URL of your server.

e) Run the following command to create a certificate in a format that you can upload to the sensor or console ⌐.

```
cat server.key server.crt serverca.crt > trusted-server.pem
```

2. Go to the ExtraHop code-examples GitHub repository ⌐ and download the `ml_api_logger/ml_api_logger.go` file to your local machine.

3. Run the following command, replacing `CERT_PATH` with the path of the TLS certificate you generated for the server:

```
export LOGGER_CERT=CERT_PATH
```

4. Run the following command, replacing `KEY_PATH` with the path of the key you signed the certificate with:

```
export LOGGER_KEY=KEY_PATH
```

5. Run the following command, replacing PORT with the port that the server listens on:

```
export LOGGER_PORT=PORT
```

6. Restrict the logger to only receive packets on a specific server IP address.

By default, the Go application receives logs on all IP addresses configured for the server. To restrict the application to a single IP address, run the following command, replacing IP with the IP address:

```
export LOGGER_IP=IP
```

7. In the ml_api_logger directory, run the following command to compile the Go code:

```
go build ml_api_logger.go
```

8. Run the following command to start the server and save the API logs to `extrahop-ade-log.json`:

```
ml_api_logger > extrahop-ade-log.json
```

## Configure the sensor or console

You must configure the sensor or console to export logs to the server you configured.

- If the certificate for your server is not trusted by the built-in certificate on the sensor, you must add the certificate ⌐ to the sensor or console.

1. Log in to the Administration settings on the sensor or console through `https://<extrahop-hostname-or-IP-address>/admin`.

2. In the Appliance Settings section, click **Running Config**.

3. Click **Edit config**.

4.  In the `hop_cloud` section, add an entry where the key is `api_logging_target` and the value is an object with the following fields:

    **enabled:** *Boolean*

    Specifies whether API interaction logging is enabled. Specify `true`.

    **hostname:** *String*

    The hostname of the server you configured to receive API interaction logs.

    **port:** *Number*

    The port that the third-party server is listening on.

    The updated `hop_cloud` section should look similar to the following JSON:

    ```
    "hopcloud": {
        "api_logging_target": {
            "enabled": true
            "hostname": "example.extrahop.com"
            "port": 100
        }
        "analysis_settings": {}
    }
    ```

## API log format

The logs are exported in JSON format. Each log of an HTTPS request to the Machine Learning Service contains the following fields:

**sequence:** *Number*

A numerical ID that correlates requests and responses. For example, if a request has a sequence number of 1, the response log will also have a sequence number of 1.

**request:** *Object*

An object that contains details about the request. The object contains the following fields:

**Close:** *Boolean*

Indicates whether the Connection header is set to close.

**ContentLength:** *Number*

The value of the ContentLength header.

**Header:** *Object*

An object that contains the HTTPS headers.

**Host:** *String*

The hostname of the server.

**Method:** *String*

The method of the request.

**Proto:** *String*

The HTTP protocol the request was sent with.

**RemoteAddr:** *String*

The IP address of the server.

**RequestURI:** *String*

The URI of the request.

**TLSVersion:** *String*

The TLS version the request was encrypted with.

**Trailer:** *String*

> The value of the Trailer header.

**TransferEncoding:** *String*

> The value of the Transfer-Encoding header.

**request_body:** *Object*

> The JSON body of POST, PUT, and PATCH requests.

Each log for an HTTPS response from the Machine Learning Service contains the following fields:

**sequence:** *Number*

> A numerical ID that correlates requests and responses. For example, if a request has a sequence number of 1, the response log will also have a sequence number of 1.

**response_status_code:** *Number*

> The status code of the response.

**response_headers:** *Object*

> An object that contains the HTTPS headers

**response_body:** *Object*

> The JSON body of the response.

## Example request

> The following JSON object is an example of a log for an API request:

```
{
    "sequence": 302,
    "request": {
        "Method": "POST",
        "Host": "appliance.example.extrahop.com",
        "RemoteAddr": "127.0.0.1:1234",
        "RequestURI": "/api/v1/metrics",
        "TLSVersion": "TLS1.2",
        "Proto": "HTTP/1.1",
        "ContentLength": 149,
        "TransferEncoding": null,
        "Header": {
            "Accept": [
                "application/json"
            ],
            "Content-Length": [
                "149"
            ],
            "Content-Type": [
                "application/json"
            ]
        },
        "Close": false,
        "Trailer": null
    },
    "request_body": {
        "metric_category": "net",
        "from": -1,
        "object_type": "capture",
        "object_ids": [
            0
        ],
        "until": 0,
        "metric_specs": [
            {
```

```
            "name": "pkts"
        }
    ],
    "cycle": "30sec"
    }
}
```

## Example response

The following JSON object is an example of a log for an API response:

```json
{
    "sequence": 302,
    "response_status_code": "200",
    "response_headers": {
        "Content-Type": [
            "application/json; charset=utf-8"
        ],
        "Vary": [
            "Accept-Encoding"
        ]
    },
    "response_body": {
        "cycle": "30sec",
        "node_id": 0,
        "clock": 1678150650000,
        "from": 1678150649999,
        "until": 1678150650000,
        "stats": [
            {
                "oid": 0,
                "time": 1678150650000,
                "duration": 30000,
                "values": [
                    1260
                ]
            }
        ]
    }
}
```