

# Exportieren von Protokollen für Machine-Learning-Dienst-API-Interaktionen


Veröffentlicht: 2024-02-16


Sie können Sensoren und Konsolen so konfigurieren, dass sie Protokolle von API-Interaktionen mit dem ExtraHop Machine Learning Service exportieren. Das ExtraHop-System exportiert API-Protokolle über HTTPS-POST-Anfragen. Jeder HTTP-Server kann die Protokolle empfangen, sofern der Server für das ExtraHop-System erreichbar ist und auf dem Server ein TLS-Zertifikat installiert ist.

## Konfigurieren Sie einen HTTP-Server für den Empfang der Protokolle

Bevor Sie das ExtraHop-System für den Export von API-Protokollen konfigurieren, müssen Sie einen HTTP-Server für den Empfang und die Datensatz der Protokolle konfigurieren. In diesem Thema wird gezeigt, wie Sie eine Go-Beispielanwendung konfigurieren, die in der [GitHub-Repository mit ExtraHop-Codebeispielen](#) .

### Bevor Sie beginnen

- Sie müssen Go auf Ihrem Computer installieren. Weitere Informationen finden Sie in der Go-Dokumentation unter <https://go.dev/learn/> .
1. Generieren Sie ein Serverzertifikat für den HTTP-Server.

 **Hinweis** Wenn Sie bereits über ein Zertifikat verfügen, können Sie diesen Schritt überspringen.

- a) Führen Sie den folgenden Befehl aus, um den Zertifizierungsstellen-Schlüssel (CA) zu generieren:

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

- b) Führen Sie den folgenden Befehl aus, um das CA-Zertifikat zu generieren:

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

- c) Führen Sie den folgenden Befehl mit Ihren Variablen aus, um eine Zertifikatsignieranforderung zu generieren:

```
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"))
```

Ersetzen Sie die folgenden Variablen im obigen Befehl:

- **SERVER-IP-ADRESSE:** Die IP-Adresse Ihres Server.
  - **SERVER-URL:** Die URL Ihres Server.
  - **NAME DER ORGANISATION:** Der Name Ihrer Organisation.
- d) Führen Sie den folgenden Befehl mit Ihren Variablen aus, um das Serverzertifikat zu generieren:

```
openssl x509 -req \  
  -in server.req -inkey server.key -out server.crt -days 3650
```

```
-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")
```

Ersetzen Sie die folgenden Variablen im obigen Befehl:

- **SERVER-IP-ADRESSE:** Die IP-Adresse Ihres Server.
  - **SERVER-URL:** Die URL Ihres Server.
- e) Führen Sie den folgenden Befehl aus, um ein Zertifikat in einem Format zu erstellen, das Sie [auf den Sensor oder die Konsole hochladen](#).

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

2. Gehe zum [GitHub-Repository mit ExtraHop-Codebeispielen](#) und laden Sie die `ml_api_logger/ml_api_logger.go` Datei auf Ihrem lokalen Computer.
3. Führen Sie den folgenden Befehl aus und ersetzen Sie `CERT_PATH` mit dem Pfad des TLS-Zertifikats, das Sie für den Server generiert haben:

```
export LOGGER_CERT=CERT_PATH
```

4. Führen Sie den folgenden Befehl aus und ersetzen Sie `KEY_PATH` mit dem Pfad des Schlüssels, mit dem Sie das Zertifikat signiert haben:

```
export LOGGER_KEY=KEY_PATH
```

5. Führen Sie den folgenden Befehl aus und ersetzen Sie `PORT` durch den Port, den der Server überwacht :

```
export LOGGER_PORT=PORT
```

6. Beschränken Sie den Logger so, dass er nur Pakete auf einer bestimmten Server-IP-Adresse empfängt. Standardmäßig empfängt die Go-Anwendung Protokolle über alle für den Server konfigurierten IP-Adressen. Um die Anwendung auf eine einzige IP-Adresse zu beschränken, führen Sie den folgenden Befehl aus und ersetzen Sie `IP` durch die IP-Adresse:

```
export LOGGER_IP=IP
```

7. Führen Sie im Verzeichnis `ml_api_logger` den folgenden Befehl aus, um den Go-Code zu kompilieren:

```
go build ml_api_logger.go
```

8. Führen Sie den folgenden Befehl aus, um den Server zu starten und die API-Protokolle zu speichern `extrahop-ade-log.json`:

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

## Konfigurieren Sie den Sensor oder die Konsole

Sie müssen den Sensor oder die Konsole so konfigurieren, dass Protokolle auf den von Ihnen konfigurierten Server exportiert werden.

- Wenn das integrierte Zertifikat auf dem Sensor dem Zertifikat für Ihren Server nicht vertraut, müssen Sie [füge das Zertifikat hinzu](#) zum Sensor oder zur Konsole.
1. Melden Sie sich bei den Administrationseinstellungen auf dem Sensor oder der Konsole an über `https://<extrahop-hostname-or-IP-address>/admin`.
  2. Klicken Sie im Abschnitt Appliance-Einstellungen auf **Konfiguration ausführen**.
  3. klicken **Konfiguration bearbeiten**.
  4. In der `hop_cloud` Abschnitt, füge einen Eintrag hinzu, in dem sich der Schlüssel befindet `api_logging_target` und der Wert ist ein Objekt mit den folgenden Feldern:

**aktiviert: Boolescher Wert**

Gibt an, ob die API-Interaktionsprotokollierung aktiviert ist. Spezifizieren `true`.

**Hostname: Schnur**

Der Hostname des Server, den Sie für den Empfang von API-Interaktionsprotokollen konfiguriert haben.

**Hafen: Zahl**

Der Port, auf dem der Drittanbieter-Server lauscht.

Das aktualisierte `hop_cloud` Der Abschnitt sollte dem folgenden JSON ähneln:

```

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

```

## API-Protokollformat

Die Protokolle werden im JSON-Format exportiert. Jedes Protokoll einer HTTPS-Anfrage an den Machine Learning Service enthält die folgenden Felder:

**Reihenfolge: Zahl**

Eine numerische ID, die Anfragen und Antworten korreliert. Wenn eine Anfrage beispielsweise die Sequenznummer 1 hat, hat das Antwortprotokoll auch die Sequenznummer 1.

**Anfrage: Objekt**

Ein Objekt, das Details zur Anfrage enthält. Das Objekt enthält die folgenden Felder:

**Schliessen: Boolescher Wert**

Gibt an, ob der Connection-Header auf Schließen eingestellt ist.

**Länge des Inhalts: Zahl**

Der Wert des ContentLength-Headers.

**Kopfzeile: Objekt**

Ein Objekt, das die HTTPS-Header enthält.

**Gastgeber: Schnur**

Der Hostname des Server.

**Methode: Schnur**

Die Methode der Anfrage.

**Proto: Schnur**

Das HTTP-Protokoll, mit dem die Anfrage gesendet wurde.

**Remote-Addr: *Schnur***

Die IP-Adresse des Server.

**URI der Anfrage: *Schnur***

Die URI der Anfrage.

**TLS-Version: *Schnur***

Die TLS-Version, mit der die Anfrage verschlüsselt wurde.

**Trailer: *Schnur***

Der Wert des Trailer-Headers.

**Übertragungskodierung: *Schnur***

Der Wert des Transfer-Encoding-Headers.

**Hauptteil der Anfrage: *Objekt***

Der JSON-Hauptteil von POST-, PUT- und PATCH-Anfragen.

Jedes Protokoll für eine HTTPS-Antwort vom Machine Learning Service enthält die folgenden Felder:

**Reihenfolge: *Zahl***

Eine numerische ID, die Anfragen und Antworten korreliert. Wenn eine Anfrage beispielsweise die Sequenznummer 1 hat, hat das Antwortprotokoll auch die Sequenznummer 1.

**Statuscode der Antwort: *Zahl***

Der Statuscode der Antwort.

**response\_headers: *Objekt***

Ein Objekt, das die HTTPS-Header enthält

**Hauptteil der Antwort: *Objekt***

Der JSON-Hauptteil der Antwort.

## Beispiel für eine Anfrage

Das folgende JSON-Objekt ist ein Beispiel für ein Protokoll für eine API-Anfrage:

```
{
  "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"
  }
}

```

## Beispiel für eine Antwort

Das folgende JSON-Objekt ist ein Beispiel für ein Protokoll für eine API-Antwort:

```

{
  "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
        ]
      }
    ]
  }
}

```