# eStreamer eNcore CLI Operations Guide v3.5

First Published: June 1, 2017
Last Updated: November 16, 2018

# Table of Contents

Table of Contents

## About This eStreamer eNcore CLI Operations Guide v3.5

| | |
|---|---|
| Author | Richard Clendenning (rclenden) |
| Change Authority | Cisco Systems Advanced Services, Security & Collaboration IDT, Implementation Americas |
| Content ID | |
| Project ID | 868408 |

## Revision History

| Revision | Date | Name or User ID | Comments |
|---|---|---|---|
| 0.1 | 11/16/2018 | Richard Clendenning | Initial Draft |
| | | | |
| | | | |

## Conventions

This document uses the following conventions.

| Convention | Indication |
|---|---|
| bold font | Commands and keywords and user-entered text appear in bold font. |
| *italic* font | Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font. |
| [ ] | Elements in square brackets are optional. |
| {x \| y \| z } | Required alternative keywords are grouped in braces and separated by vertical bars. |
| [ x \| y \| z ] | Optional alternative keywords are grouped in brackets and separated by vertical bars. |
| String | A non-quoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks. |
| `courier font` | Terminal sessions and information the system displays appear in `courier` font. |
| < > | Nonprinting characters such as passwords are in angle brackets. |
| [ ] | Default responses to system prompts are in square brackets. |
| !, # | An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line. |

## Conventions

Note: Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Caution: Means *reader be careful.* In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning: IMPORTANT SAFETY INSTRUCTIONS

Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Regulatory: Provided for additional information and to comply with regulatory and customer requirements.

# 1 Introduction

## 1.1 Document Purpose

This document explains how to install, configure, and operate the eStreamer eNcore CLI client.

## 1.2 Background

The Cisco Event Streamer (eStreamer) allows users to stream Firepower intrusion, discovery, and connection data from a Firepower Management Center or managed device (i.e., the eStreamer server) to external client applications. The eStreamer server responds to client requests with terse, compact, binary encoded messages that facilitate high performance.

eStreamer eNcore CLI is a multi-platform, multi-process eStreamer client application written in Python that is compatible with FMC versions 6.0 and above.

## 1.3 Application Summary

eNcore is an all-purpose client, which requests all possible events from eStreamer, parses the binary content, and outputs events in various formats to support other Security Information and Event Management tools (SIEMs). eNcore was built from scratch in Python with a scalable and fast multi-process architecture. It supports version 6.0 of Firepower Management Center. It was built and tested on CentOS 7, but should work with any Linux distribution that supports the pre-requisites. The software will run on Windows but is not supported.

There are three packages associated with eStreamer eNcore:

- eNcore CLI
- eNcore Add-on for Splunk
- eNcore Dashboard for Splunk

This Guide only addresses the eNcore CLI package. For more information on the Splunk packages, please see the eStreamer eNcore for Splunk Operations Guide.

eNcore CLI provides a command line interface for eStreamer eNcore. It runs as a standalone application that requests events from the FMC eStreamer server and outputs these events in one of these formats:

— key-value pairs designed to maintain compatibility with previous Splunk collectors

— JSON

— CEF for Arcsight which maintains backwards compatibility with the previous cef-agent.

The output can be streamed to files, a TCP or UDP network port, or stdout.

# 2 Prerequisites

eNcore CLI works with any Linux distribution that supports the prerequisites. It will run on Windows although it has not been made production-ready.

There are two prerequisites for the platform on which eNcore will be installed:

- Python 2.7
- pyOpenSSL

## 2.1 Python 2.7

To check whether Python2.7 is present, use following command:

```
which python
```

If python has been installed, the **`which python`** command provides the path to the installation directory. For example, if the output of the command is:

```
/usr/bin/python
```

then python has been installed. To determine whether the installed python is v2.7, list the contents of the parent of the installation directory (in the above example, the **`/usr/bin`** directory). For example, suppose the listing shows an entry such as the following:

```
lrwxrwxrwx 1 root root       9 Dec  9  2015 python -> python2.7*
```

This entry shows that **`python`** is a link to the **`python2.7`** directory, where Python v2.7 is installed. Another command, **`whereis python`**, can also be used to show if a python2.7 directory exists.

Note: If you are installing the CLI version on a device running Splunk, then it is worth noting that Splunk has its own version of Python. The Splunk Python has been compiled differently from the normal distribution – specifically, it is built with PyUnicodeUCS2. The encore.sh script will detect this and warn you. If you encounter this problem, then you will need to create a new user and run eStreamer-eNcore as that user. You should consider running the Splunk add-on instead.

If Python 2.7 has not been installed, on CentOS it can be installed with this command:

```
sudo yum install python
```

## 2.2 pyOpenSSL

The second eNcore CLI prerequisite is pyOpenSSL. pyOpenSSL may have been installed as part of the Python 2.7 installation. To check whether it is installed, use the following command:

```
pip list | grep -i pyOpenSSL
```

If pip is not installed, on CentOS it can be installed with this command:

```
sudo python get-pip.py
```

If pyOpenSSL is not installed, on CentOS it can be installed with these commands:

```
sudo yum install python-pip python-devel openssl-devel gcc

sudo pip install pyopenssl
```

## 2.3 EPEL Repo Dependency for RHEL

If you are having problems installing these packages, then you may need to enable the EPEL repository. It is recommended to search the Internet for instructions on installing the EPEL Repo.

## 2.4 Windows

Note: Windows is not yet supported for production execution. If, however, you wish to attempt an install for the CLI version, then you will need to run the following commands to install pyOpenSSL and win-inet-pton:

```
pip install pyOpenSSL
pip install win-inet-pton
```

# 3 Installation

There are 3 steps for installing eNcore CLI:

1. Download the eNcore CLI installation file, copy it to the target server, and unzip it.
2. Create the PKCS12 file on the eStreamer server and install it on the eNcore server.
3. Run the encore.sh script in test mode to complete the setup configuration.

These steps are described in detail in the sections below.

## 3.1 Installing the eNcore CLI Application

eNcore CLI is downloadable as a compressed file in the format `eStreamer-eNcore-cli-X.Y.Z.tar.gz`, where X, Y, and Z are major and minor version numbers. Download the file and copy it to the target server. One way to copy it is to use the Linux **`scp`** command from the computer where the file resides:

```
scp /path/to/eStreamer-eNcore-cli-X.Y.Z.tar.gz user@host:/path/on/installation/server
```

Unzip the compressed file on the installation server with the tar command:

```
sudo tar -xf eStreamer-eNcore-cli-X.Y.Z.tar.gz
```

## 3.2 Creating and Installing the PKCS12 File

The eStreamer server must be able to authenticate and authorize client connections. This requires a PKCS12 file on the eStreamer server that identifies the eStreamer client, and this file must be copied to the eNcore server.

Appendix A contains the instructions for creating the PKCS12 file on an FMC, downloading it to a local machine, and copying it to the eNcore server.

## 3.3 Complete the Setup Configuration

Change the working directory to eStreamer-eNcore. You may wish to verify that the client.pkcs12 file and the encore.sh script are in this directory.

Then, run the **`encore.sh`** shell script with **`test`** as an argument:

```
./encore.sh test
```

The script will check for the following:

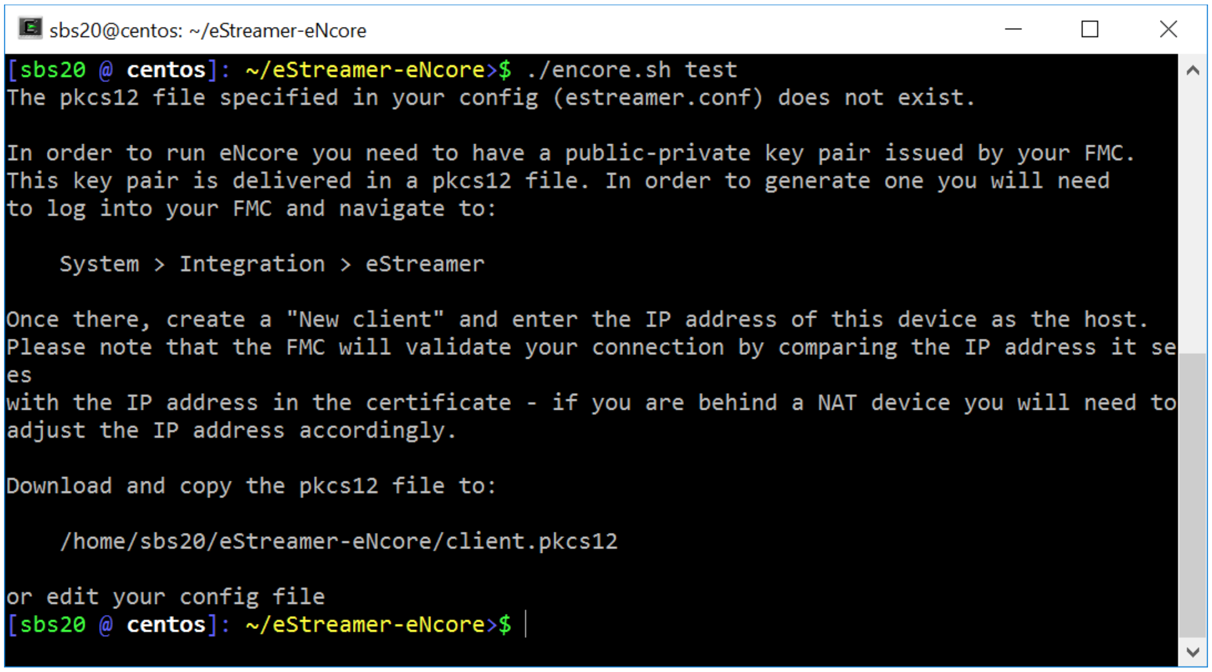
— Python 2.7

— pyOpenSSL

— the client.pkcs12 file in the eStreamer-eNcore directory

This verification is not noticeable if the results of the checks are positive, but the script will alert you if there is an issue. The script reads the estreamer.conf file to determine the name and location of the PKCS12 file, and by default the configuration file contains the name **`client.pcks12`** for the file, in the eStreamer-eNcore directory. If the file is not found with this name at this location, you will see the following error and instructions:

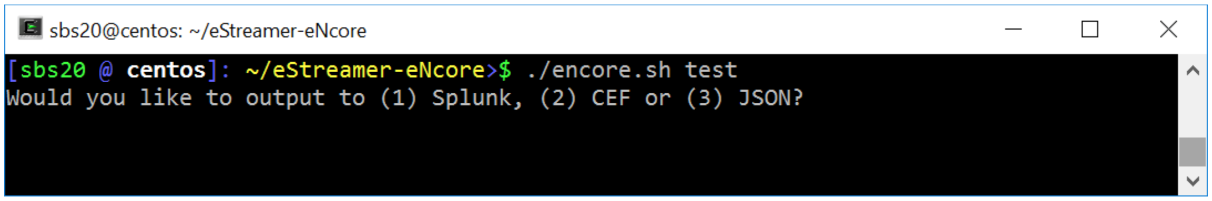
Figure 2: Missing pkcs12 File



```
sbs20@centos: ~/eStreamer-eNcore
[sbs20 @ centos]: ~/eStreamer-eNcore>$ ./encore.sh test
The pkcs12 file specified in your config (estreamer.conf) does not exist.

In order to run eNcore you need to have a public-private key pair issued by your FMC.
This key pair is delivered in a pkcs12 file. In order to generate one you will need
to log into your FMC and navigate to:

    System > Integration > eStreamer

Once there, create a "New client" and enter the IP address of this device as the host.
Please note that the FMC will validate your connection by comparing the IP address it se
es
with the IP address in the certificate - if you are behind a NAT device you will need to
adjust the IP address accordingly.

Download and copy the pkcs12 file to:

    /home/sbs20/eStreamer-eNcore/client.pkcs12

or edit your config file
[sbs20 @ centos]: ~/eStreamer-eNcore>$
```

If all is well, the script will prompt you to choose whether to output data for Splunk, CEF or JSON, as shown in the screen shot below.

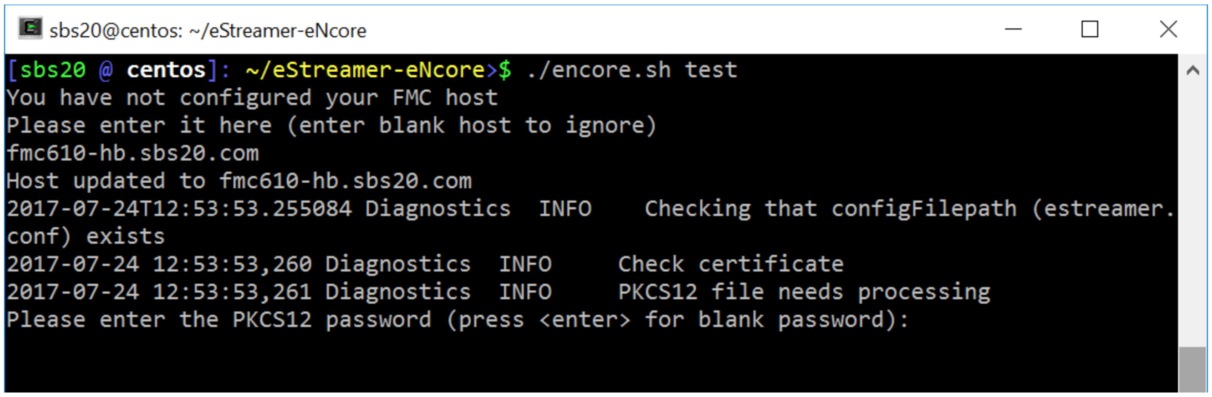
Figure 1. Choosing your output



```
sbs20@centos: ~/eStreamer-eNcore
[sbs20 @ centos]: ~/eStreamer-eNcore>$ ./encore.sh test
Would you like to output to (1) Splunk, (2) CEF or (3) JSON?
```

You will then be prompted to enter the IP or hostname of the FMC, and after that the PKCS12 file password, as shown in the screen shot below.

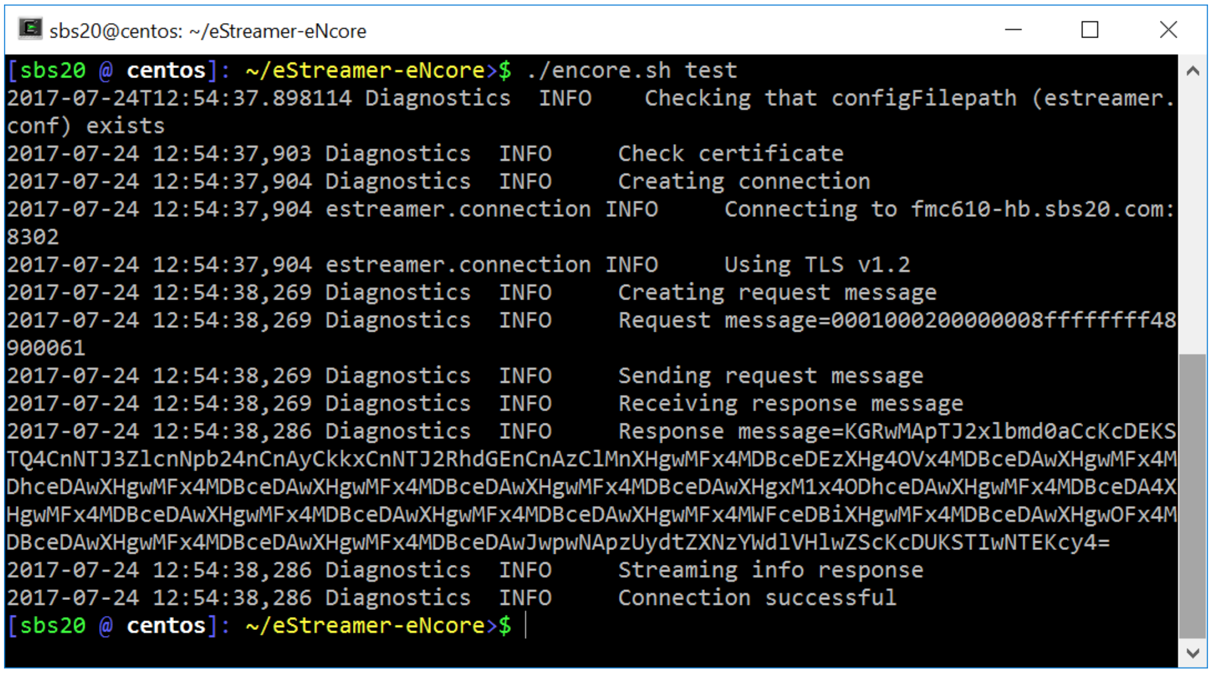
Figure 3: Enter Password



```
sbs20@centos: ~/eStreamer-eNcore
[sbs20 @ centos]: ~/eStreamer-eNcore>$ ./encore.sh test
You have not configured your FMC host
Please enter it here (enter blank host to ignore)
fmc610-hb.sbs20.com
Host updated to fmc610-hb.sbs20.com
2017-07-24T12:53:53.255084 Diagnostics  INFO    Checking that configFilepath (estreamer.
conf) exists
2017-07-24 12:53:53,260 Diagnostics  INFO     Check certificate
2017-07-24 12:53:53,261 Diagnostics  INFO     PKCS12 file needs processing
Please enter the PKCS12 password (press <enter> for blank password):
```

If the password is correct (it should match the password entered on the FMC when the client certificate was created), then the script will initiate a test connection to the FMC eStreamer server and report the results. An example of a successful connection is shown in the screen shot below.

Figure 4: Successful Test



```
sbs20@centos: ~/eStreamer-eNcore
[sbs20 @ centos]: ~/eStreamer-eNcore>$ ./encore.sh test
2017-07-24T12:54:37.898114 Diagnostics  INFO    Checking that configFilepath (estreamer.
conf) exists
2017-07-24 12:54:37,903 Diagnostics  INFO     Check certificate
2017-07-24 12:54:37,904 Diagnostics  INFO     Creating connection
2017-07-24 12:54:37,904 estreamer.connection INFO     Connecting to fmc610-hb.sbs20.com:
8302
2017-07-24 12:54:37,904 estreamer.connection INFO     Using TLS v1.2
2017-07-24 12:54:38,269 Diagnostics  INFO     Creating request message
2017-07-24 12:54:38,269 Diagnostics  INFO     Request message=0001000200000008ffffffff48
900061
2017-07-24 12:54:38,269 Diagnostics  INFO     Sending request message
2017-07-24 12:54:38,269 Diagnostics  INFO     Receiving response message
2017-07-24 12:54:38,286 Diagnostics  INFO     Response message=KGRwMApTJ2xlbmd0aCcKcDEKS
TQ4CnNTJ3ZlcnNpb24nCnAyCkkxCnNTJ2RhdGEnCnAzClMnXHgwMFx4MDBceDEzXHg4OVx4MDBceDAwXHgwMFx4M
DhceDAwXHgwMFx4MDBceDAwXHgwMFx4MDBceDAwXHgwMFx4MDBceDAwXHgxM1x4ODhceDAwXHgwMFx4MDBceDA4X
HgwMFx4MDBceDAwXHgwMFx4MDBceDAwXHgwMFx4MDBceDAwXHgwMFx4MWFceDBiXHgwMFx4MDBceDAwXHgwOFx4M
DBceDAwXHgwMFx4MDBceDAwXHgwMFx4MDBceDAwJwpwNApzUydtZXNzYWdlVHlwZScKcDUKSTIwNTEKcy4=
2017-07-24 12:54:38,286 Diagnostics  INFO     Streaming info response
2017-07-24 12:54:38,286 Diagnostics  INFO     Connection successful
[sbs20 @ centos]: ~/eStreamer-eNcore>$
```

If the test is successful, installation of eNcore CLI is complete. If the connection attempt is not successful, then the problem is likely either a network issue or a certificate issue. See the troubleshooting guidelines in Section 7 to resolve issues and create a successful connection.

# 4 Configuration

The eNcore CLI installation process covered in Section 3 requires configuration of basic items such as the FMC IP address in order to establish a connection to the FMC eStreamer server. This section covers general configuration of the application so that it will fulfill the solution requirements.

The configuration is stored in a file called estreamer.conf, in the eStreamer-eNcore directory, and initially it contains default settings which can be changed as needed. The file is in JSON format and contains keys that provide the configuration information. This section details the keys and sections that are most likely to be changed. The complete list of settings is provided in Appendix B.

## 4.1 Subscription

The subscription key contains two major subsections:

1. The records section allows the user to select which types of events eNcore will request upon connecting to the FMC eStreamer server.

2. The servers section contains the FMC host IP and associated information.

An example of this key and its value is shown below.

Figure 8: Subscription Section

```
"subscription": {
  "records": {
    "@comment": [
      "Just because we subscribe doesn't mean the server is sending. Nor does it mean",
      "we are writing the records either. See handler.records[]"
    ],
    "archiveTimestamps": true,
    "eventExtraData": true,
    "extended": true,
    "impactEventAlerts": true,
    "intrusion": true,
    "metadata": true,
    "packetData": true
  },
  "servers": [
    {
      "host": "1.2.3.4",
      "port": 8302,
      "pkcs12Filepath": "client.pkcs12",
      "@comment": "Valid values are 1.0 and 1.2",
      "tlsVersion": 1.2
    }
  ], ...
```

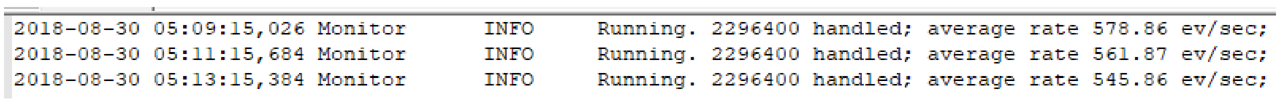
The encore.sh script, when run with test as an argument, prompts for the IP address of the FMC if it has not been configured in the estreamer.conf file, and writes the user-provided IP to the file. The port, pkcs12Filepath, and tlsVersion should generally not be changed. If you encounter TLS difficulties and are willing to downgrade the version, then you can change tlsVersion to 1.0.

Note: Note that downgrading the TLS version is useful for debugging and seeing the software work but it is not a recommended long-term strategy. It is recommended instead to fix the root cause.

## 4.2 The Monitor

The monitor is a separate thread that runs monitoring and maintenance tasks. By default, it runs every two minutes. It writes the number of events handled to the eNcore log and checks the status of subprocesses. If are any problems with subprocesses, the monitor places the client into an error state and the client shuts itself down.

An example of messages written to the log by the monitor thread is shown below.



```
2018-08-30 05:09:15,026 Monitor      INFO     Running. 2296400 handled; average rate 578.86 ev/sec;
2018-08-30 05:11:15,684 Monitor      INFO     Running. 2296400 handled; average rate 561.87 ev/sec;
2018-08-30 05:13:15,384 Monitor      INFO     Running. 2296400 handled; average rate 545.86 ev/sec;
```

Several aspects of the log messages can be configured in the monitor section of the estreamer.conf configuration file, such as:

- period: The interval in seconds at which the monitor performs its check of subprocesses and writes status messages to the log.
- bookmark: If true, the bookmark (the time of the latest event in Unix time format) is included in each monitor log message
- handled: If true, the number of events that eNcore has handled since being started.
- details: If true, then in addition to the brief status message that the monitor writes to the log, it will also write a detailed message containing many status items related to the operation of the eNcore client.

An example of the configuration of these parameters in the estreamer.conf file is shown here:

```
"monitor": {
  "period": 120,
  "bookmark": false,
  "handled": true,
  "details": true
},
```

## 4.3 Start Time

The eStreamer server expects the client request to contain the start time, which specifies that the FMC should only send events that occurred after that time. There are three options:

0: Send all events from the earliest point available on the FMC.

1: Send all events that occur after receiving this client request.

2: Send all events that occurred after the bookmark time that is sent in the request.

When eNcore CLI is running, it frequently writes a bookmark containing the time of the latest received event (in UNIX time format) to a bookmark file. The "2" option is useful when eNcore resumes operation after a period of non-operation, to request that the FMC server start sending events from the point at which communication was interrupted. In this way, a SIEM can receive all events only once from eNcore CLI even when eNcore CLI has stopped for a period and then is restarted.

An example of the start configuration in the estreamer.conf file is shown here:

```
"@startComment": "0 for genesis, 1 for now, 2 for bookmark",
"start": 2,
```

## 4.4 Handler

The handler section represents configuration items for which records will be handled, and how they will be handled.

### 4.4.1 Outputters

The outputters section specifies how eNcore will write the events to output. eNcore CLI can provide output in one of these formats:

— Splunk

— JSON

— CEF for Arcsight

The output can be sent to a SIEM or other collector over a network connection, or written to a file.

The examples below show an ArcSight CEF outputter that is configured to send output to an ArcSight connector over udp, and a second ArcSight CEF outputter that will write the same events to a local file. In the second outputter, the {0} notation in the uri specifies that a UNIX timestamp should be placed in the filename.

```
"outputters": [
      {
        "adapter": "cef",
        "enabled": true,
        "name": "CEFarcsight",
        "stream": {
          "uri": "udp://192.168.1.160:514"
        }
      },
      {
        "adapter": "cef",
        "enabled": true,
        "name": "CEFfile",
        "stream": {
          "options": {
            "rotate": false
          },
          "uri": "file:///:data/data.{0}.cef"
        }
      }
    ],
```

### 4.4.2 Records

The records section specifies which records eNcore will process. There are two modes in which events are identified for handling (or for exclusion from handling):

1. The user can specify that a class of events, such as connections, should be processed by setting that class's value to true. An example of this is the key-value pair "connections": true. Conversely, the user can also specify that a class of events should not be processed by setting that class's value to false.
2. The user can specify exceptions to the handling of classes of events on a per-record-type basis by writing the record type as a value for the include or exclude keys. Multiple values should be comma-separated in the JSON array. As an example, to exclude record types 98 and 170, the exclude key-value pair would read:

   "exclude": [98, 170],

An example of the records key-value pair is shown below. Note that for classes of records to be handled, they must first be selected in the FMC eStreamer configuration. They must also be configured for subscription in the records portion of the subscription section of the eNcore configuration.

```
"records": {
      "connections": true,
      "core": true,
      "excl@comment": [
         "These records will be excluded regardless of above (overrides 'include')",
         "e.g. to exclude flow and IPS events use [ 71, 400 ]"
      ],
      "exclude": [],
      "inc@comment": "These records will be included regardless of above",
      "include": [],
      "intrusion": true,
      "metadata": false,
      "packets": true,
      "rna": true,
      "rua": true
   }
```

## 4.5 Performance Tuning

The performance of eNcore CLI has been improved in version 3.5 with the addition of multi-processing. By default, four worker processes operate on the incoming messages to achieve higher throughput. While multiple processes can provide significant performance gains, these gains are highly dependent on the platform because for each platform, the processing bottlenecks may be different. Multiple processes also require additional overhead for managing task distribution, so that increasing the number of processes could actually decrease the performance on platforms with a low number of CPU cores.

The number of worker processes is configurable through the workerProcesses parameter in the estreamer.conf configuration file. The number can be set from 1 to 12. Generally, the more capable the platform (i.e., more CPU cores, better I/O, etc.), more throughput is achieved through a higher number of worker processes. However, the only reliable approach is to test performance with various settings such as 1, 2, 4, 8, and 12, and in many cases the best performance may be gained with just one worker process because no process marshalling is required.

One scenario for testing is to:

1. Disable any outputters that write to a production SIEM, because the same events will be requested multiple times during the testing.
2. Configure a set number of workerProcesses (such as 8) and then start eNcore with a start parameter of 0 (for genesis) or at least an old start time.
3. Request connection events from the FMC (or in some other way request the FMC to send millions of backlogged events).
4. Observe the event rate reported by the monitor process in the estreamer.log file.
5. Repeat the test with a different number of workerProcesses.
6. When the optimal number has been determined, set the workerProcesses to that number. If any outputters were disabled in Step 1, they should now be re-enabled to resume production operations.

An example of the workerProcesses configuration in the estreamer.conf file is shown here:

```
"workerProcesses": 12
```

## 4.6 Batch Size

eNcore CLI also attempts to improve performance by batching received events and only writing them to output when the threshold for the batch has been reached. The default batch size is 100 events.

If the event rate is very low, then a batch size of 100 events could cause an unwanted delay in the appearance of events in a SIEM. For example, if intrusion events are the only events that are handled and the intrusion event rate averages 100 events per hour, then the first event in a batch could be delayed an hour or more until the batch completes and is written to disk. To reduce such delays the batchSize can be set to a lower value, or to eliminate delays entirely, the batchSize can be set to 1.

The disadvantage of setting batchSize to 1 is that in high-throughput environments the performance will be lower.

An example of the batchSize configuration in the estreamer.conf file is shown here:

```
"batchSize": 50
```

## 4.7 Enabled

The value of the enabled key must be set to true for eNcore to request events from the FMC and to begin streaming operations. An example of this key is:

```
"enabled": true,
```

## 4.8 Logging

The logging key specifies how the application will log information related to its own health and the processing of events. The subkeys specify:

- filepath – the path and name of the log file.
- format – the information that will be included in log messages, and the format of the messages

- level – the logging level. By default, the level is INFO, but for troubleshooting purposes it can be increased to DEBUG, VERBOSE, or TRACE. In a production environment the DEBUG level should only be used if the event rate is low (e.g., less than 500 events/second, although this depends heavily on the platform). VERBOSE and TRACE should never be used in production. These levels write heavily to the log file and should only be used in a test environment with an extremely low event rate.
- stdout – whether the log output should be written to the console.

An example of the logging key is shown below:

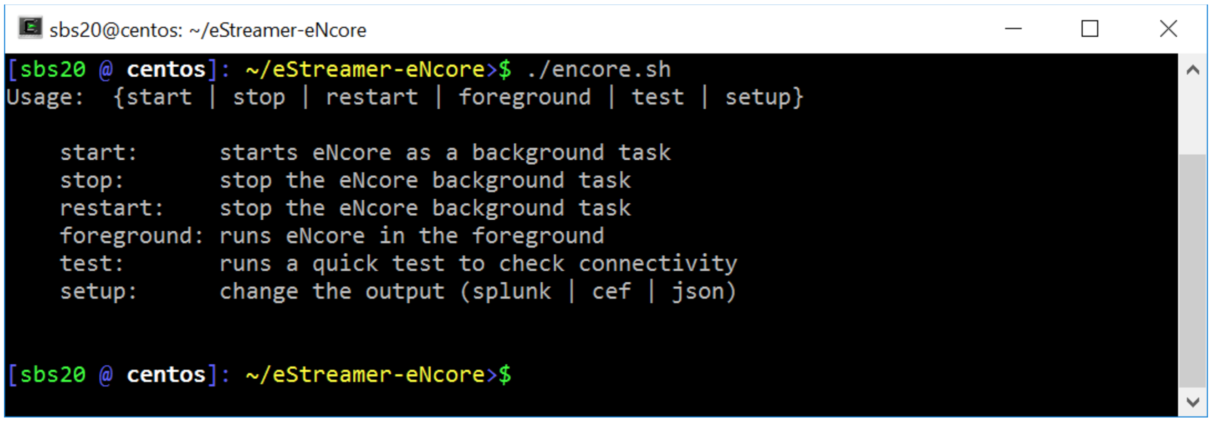
```
"logging": {
   "filepath": "estreamer.log",
   "format": "%(asctime)s %(name)-12s %(levelname)-8s %(message)s",
   "lev@comment": "Levels include FATAL, ERROR, WARNING, INFO, DEBUG, VERBOSE and TRACE",
   "level": "INFO",
   "stdOut": true
},
```

# 5 Operation

Once you have fully configured all items as described in section 4, eNcore CLI can be executed to begin streaming and writing events.

If you run encore.sh without any parameters, you will be presented with brief instructions.

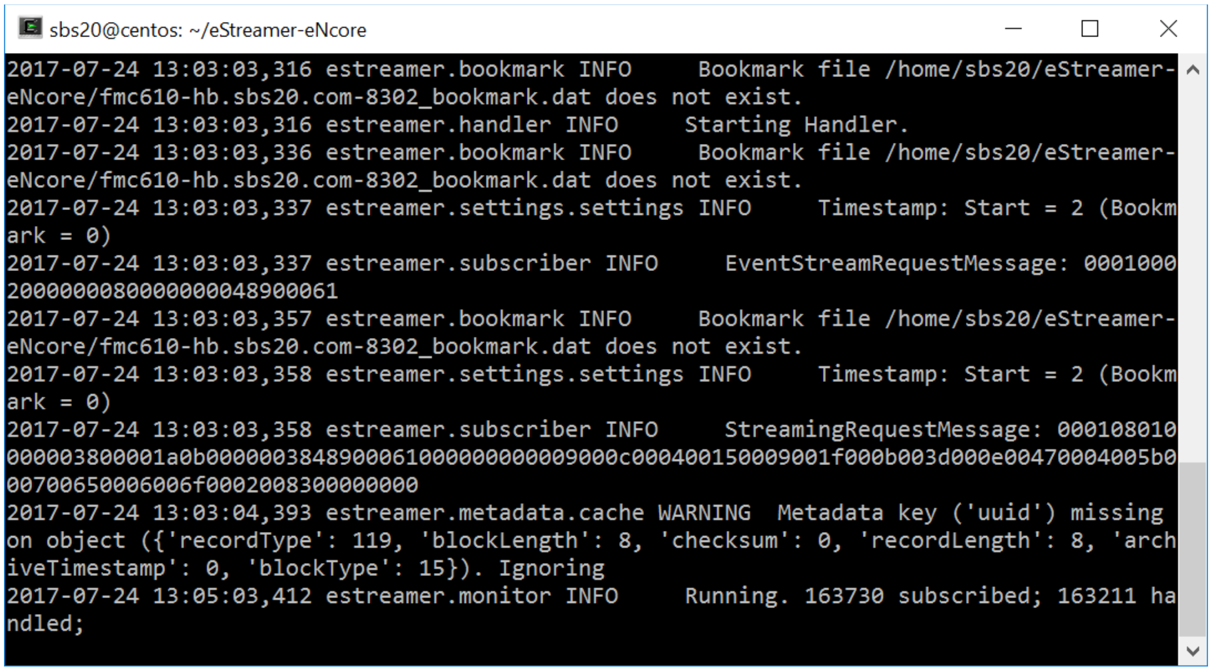
Figure 5: Help Screen



```
[sbs20 @ centos]: ~/eStreamer-eNcore>$ ./encore.sh
Usage:  {start | stop | restart | foreground | test | setup}

    start:      starts eNcore as a background task
    stop:       stop the eNcore background task
    restart:    stop the eNcore background task
    foreground: runs eNcore in the foreground
    test:       runs a quick test to check connectivity
    setup:      change the output (splunk | cef | json)


[sbs20 @ centos]: ~/eStreamer-eNcore>$
```

For the first run, it is recommended to run it in the foreground so you can see what is happening. Every two minutes, the screen will update with a note of how many records have been processed. If you wish to change the update frequency, then see the monitor.period configuration setting.

Figure 6: Running in the Foreground with Monitor Status



```
2017-07-24 13:03:03,316 estreamer.bookmark INFO     Bookmark file /home/sbs20/eStreamer-
eNcore/fmc610-hb.sbs20.com-8302_bookmark.dat does not exist.
2017-07-24 13:03:03,316 estreamer.handler INFO     Starting Handler.
2017-07-24 13:03:03,336 estreamer.bookmark INFO     Bookmark file /home/sbs20/eStreamer-
eNcore/fmc610-hb.sbs20.com-8302_bookmark.dat does not exist.
2017-07-24 13:03:03,337 estreamer.settings.settings INFO     Timestamp: Start = 2 (Bookm
ark = 0)
2017-07-24 13:03:03,337 estreamer.subscriber INFO     EventStreamRequestMessage: 0001000
20000000800000000048900061
2017-07-24 13:03:03,357 estreamer.bookmark INFO     Bookmark file /home/sbs20/eStreamer-
eNcore/fmc610-hb.sbs20.com-8302_bookmark.dat does not exist.
2017-07-24 13:03:03,358 estreamer.settings.settings INFO     Timestamp: Start = 2 (Bookm
ark = 0)
2017-07-24 13:03:03,358 estreamer.subscriber INFO     StreamingRequestMessage: 000108010
000003800001a0b000000384890006100000000000009000c000400150009001f000b003d000e00470004005b0
00700650006006f0002008300000000
2017-07-24 13:03:04,393 estreamer.metadata.cache WARNING  Metadata key ('uuid') missing
on object ({'recordType': 119, 'blockLength': 8, 'checksum': 0, 'recordLength': 8, 'arch
iveTimestamp': 0, 'blockType': 15}). Ignoring
2017-07-24 13:05:03,412 estreamer.monitor INFO     Running. 163730 subscribed; 163211 ha
ndled;
```

Note: To stop the foreground process, press ctrl-c.

To ensure that eNcore is writing the streamed events to the output as it has been configured to do in the outputters section of the estreamer.conf file, compare the events as shown in the FMC with the events in the SIEM or other collector (or in the local file if one has been configured).

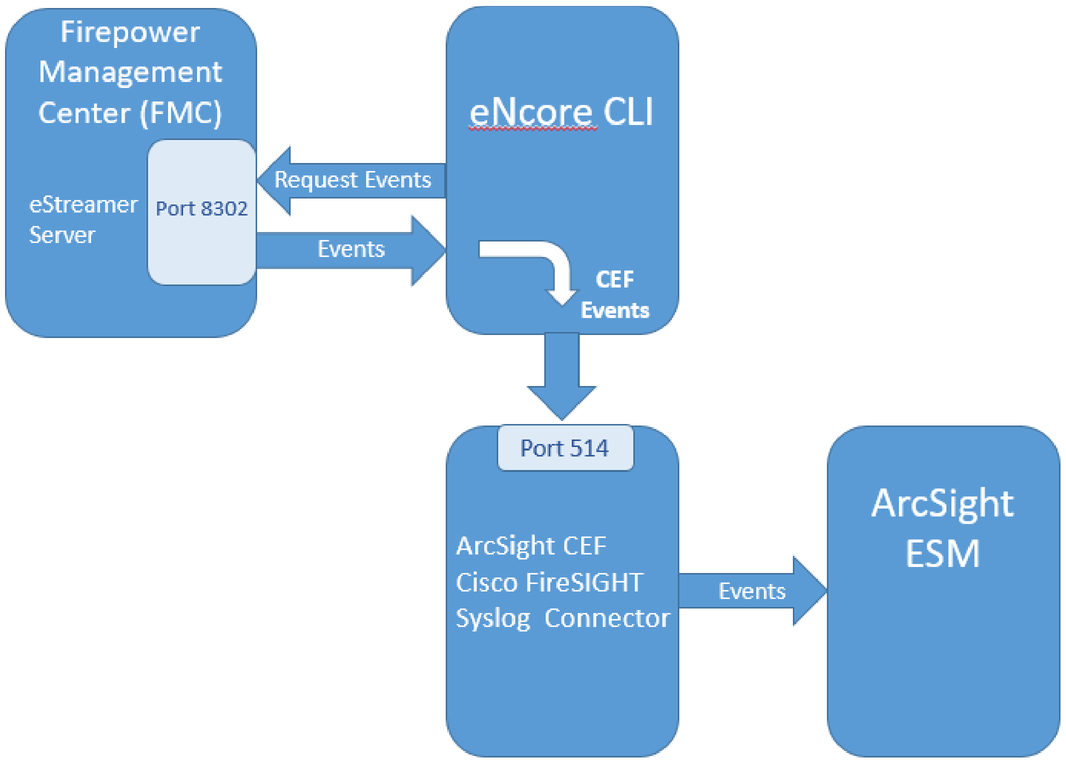
When it has been verified that the solution is working properly and it is ready to be put into production, eNcore CLI can be started as a background task by this command:

```
./encore start
```

# 6 Integrating eNcore with ArcSight ESM

The eNcore CLI CEF outputter is designed to provide Firepower events to the ArcSight CEF Cisco FireSIGHT Syslog Connector, so that the connector can in turn provide the events to ArcSight ESM. The solution is illustrated below.

Figure 7: Integrating eNcore with ArcSight ESM



The flow of events is as follows:

1. eNcore CLI establishes the connection to the FMC and requests events.
2. The eStreamer server on the FMC accepts the connection attempt, receives the request for events, and starts streaming events over an SSL-encrypted tunnel.
3. eNcore CLI receives the events in the eStreamer binary format, parses them, and writes them in CEF format to the ArcSight CEF Cisco FireSIGHT Syslog Connector.
4. The ArcSight CEF Cisco FireSIGHT Syslog Connector accepts the events and forwards them to ArcSight ESM.

The installation, configuration, and operation of the ArcSight Connector are provided in the "HPE Security ArcSight Connectors: SmartConnector for ArcSight CEF Cisco FireSIGHT Syslog Configuration Guide" published by Hewlett Packard Enterprise (HPE).

# 7 Troubleshooting and Questions

## 7.1 Error Messages

eNcore CLI is engineered to provide meaningful error messages. Below is an example error message.

> The eStreamer service has closed the connection. There are a number of possible causes which may show above in the error log.
>
> If you see no errors then this could be that
> * the server is shutting down
> * there has been a client authentication failure (please check that your outbound IP address matches that associated with your certificate - note that if your device is subject to NAT then the certificate IP must match the upstream NAT IP)
> * there is a problem with the server. If you are running FMC v6.0, you may need to install "Sourcefire 3D Defense Center S3 Hotfix AZ 6.1.0.3-1")

If you encounter errors that do not make sense or require further explanation, then please contact support so that we can fix the problem and improve the error messages.

## 7.2 Error Connecting to the FMC

If eNcore CLI does not get a response when attempting to establish a connection with the FMC, the log provides the following message:

```
TimeoutException: Could not connect to eStreamer Server at all.
```

The error message goes on to recommend checking the host and port. This means to check that the FMC host configured in estreamer.conf is correct, and that the port (which should be left at the default value of 8302) is correct. The error message goes on to suggest it could be a firewall issue. This is because tcp port 8302 may need to be opened explicitly between eNcore CLI and the FMC if there are one or more firewalls between them.

If the above checks do not help resolve the issue, then traditional network troubleshooting techniques should be employed, such as using the traceroute utility and/or network captures at critical points in the routing.

## 7.3 Error: PID File Already Exists

When eNcore starts, it writes a PID file that contains the PID of the eNcore main process. This file is named using the IP address of the eNcore server, as in this example:

```
192.168.1.20-8302_proc.pid
```

When eNcore exits gracefully, it deletes this file. But if eNcore terminates without deleting this file, then when eNcore is restarted and it finds this file, eNcore will terminate with this error:

```
ERROR    EncoreException: PID file already exists
```

The solution is to manually delete the PID file (it is in the eStreamer-eNcore directory), and then eNcore should start normally and write a new PID file.

## 7.4 Frequently Asked Questions

### Can I connect to more than one FMC?

Currently, not within a single instance.

### Can eNcore de-duplicate data to keep my SIEM costs lower?

Not today. It is on the roadmap.

### Can I run two instances of eNcore in a HA pair?

Yes and no. It is technically possible to run two side-by-side, but they will be completely ignorant of each other and output double the data. It may be preferable to run them in a hot-stand-by configuration where the primary client's state and configuration data is regularly copied to the secondary client. The state and configuration data in question are comprised of:

- estreamer.conf
- x.x.x.x-port_bookmark.dat
- x.x.x.x-port_cache.dat
- x.x.x.x-port_pkcs.cert
- x.x.x.x-port_pkcs.key
- x.x.x.x-port_status.dat

### Can I increase the logging granularity?

Yes, by changing logging.level in the estreamer.conf file. Please note that while it is possible to increase this level to VERBOSE, the performance impact will be crippling. DEBUG may be useful but slow. We strongly recommend not going above INFO for standard production execution.

# 8 Cisco Support

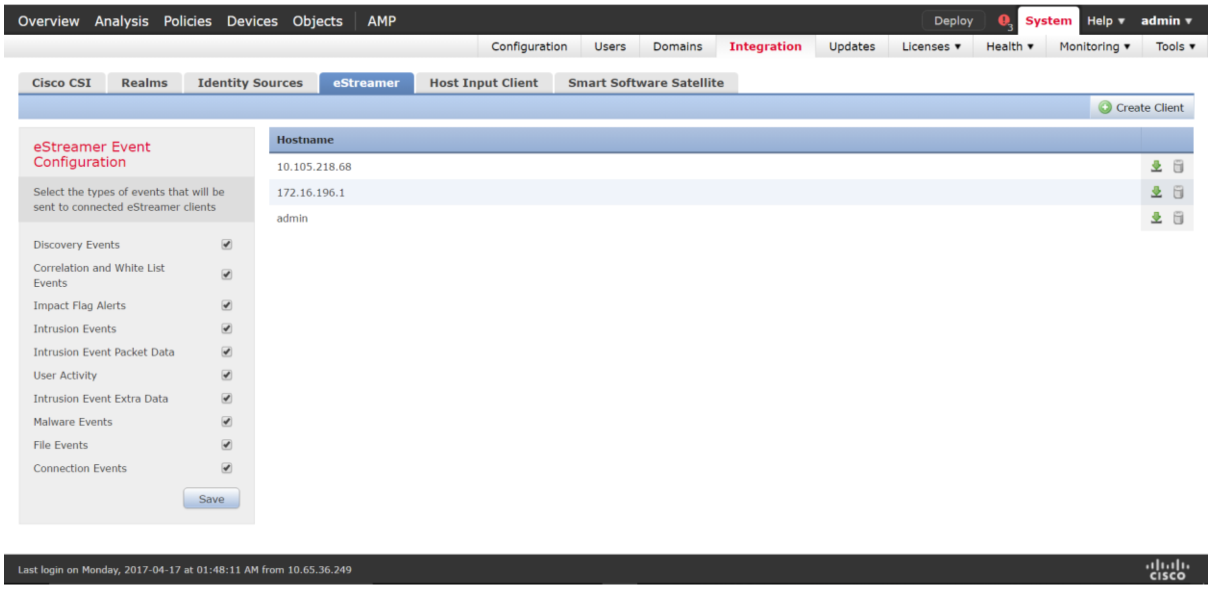
The application is free to use and is community supported. Questions can be emailed to encore-community@cisco.com. A paid and TAC supported version is available by ordering FP-CEF-SW-K9.

# 9 Appendix A: eNcore Client Certificate Creation and Installation

The steps to generate an eStreamer client certificate are as follows:

1. Navigate to the web interface of the FMC at https://fmc-ip-address and log in with your FMC credentials.
2. In the FMC 6.x GUI, navigate to System > Integration > eStreamer.

FMC eStreamer Certificate Creation



eStreamer Event Configuration

Select the types of events that will be sent to connected eStreamer clients

| | |
|---|---|
| Discovery Events | ✓ |
| Correlation and White List Events | ✓ |
| Impact Flag Alerts | ✓ |
| Intrusion Events | ✓ |
| Intrusion Event Packet Data | ✓ |
| User Activity | ✓ |
| Intrusion Event Extra Data | ✓ |
| Malware Events | ✓ |
| File Events | ✓ |
| Connection Events | ✓ |

| Hostname |
|---|
| 10.105.218.68 |
| 172.16.196.1 |
| admin |

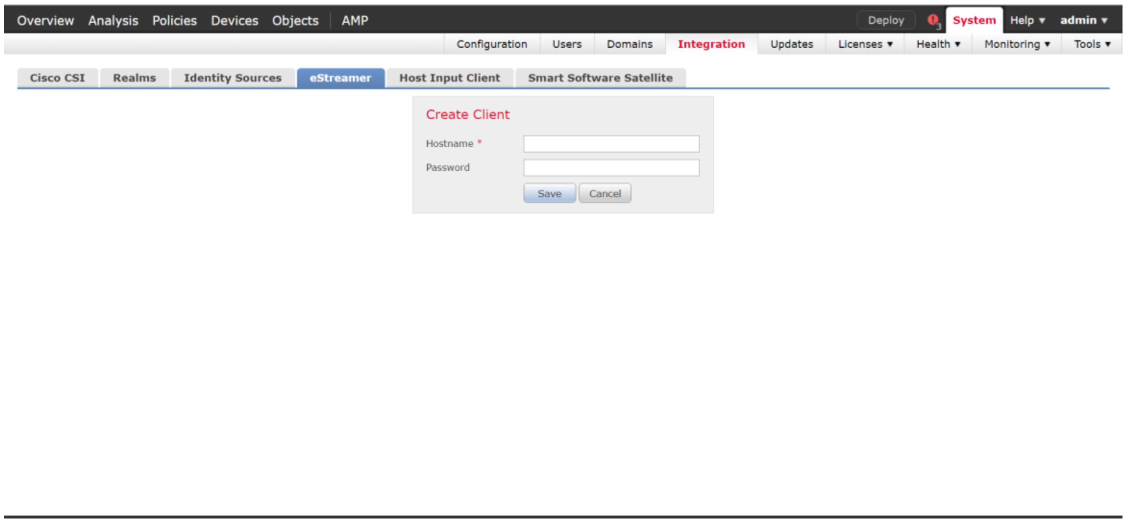
Last login on Monday, 2017-04-17 at 01:48:11 AM from 10.65.36.249

3. Click Create Client.
4. Provide the Hostname and password.

Note: This should be the IP of the client, which will be collecting the event data from the FMC. The password you enter here will be required when you first execute eStreamer eNcore.

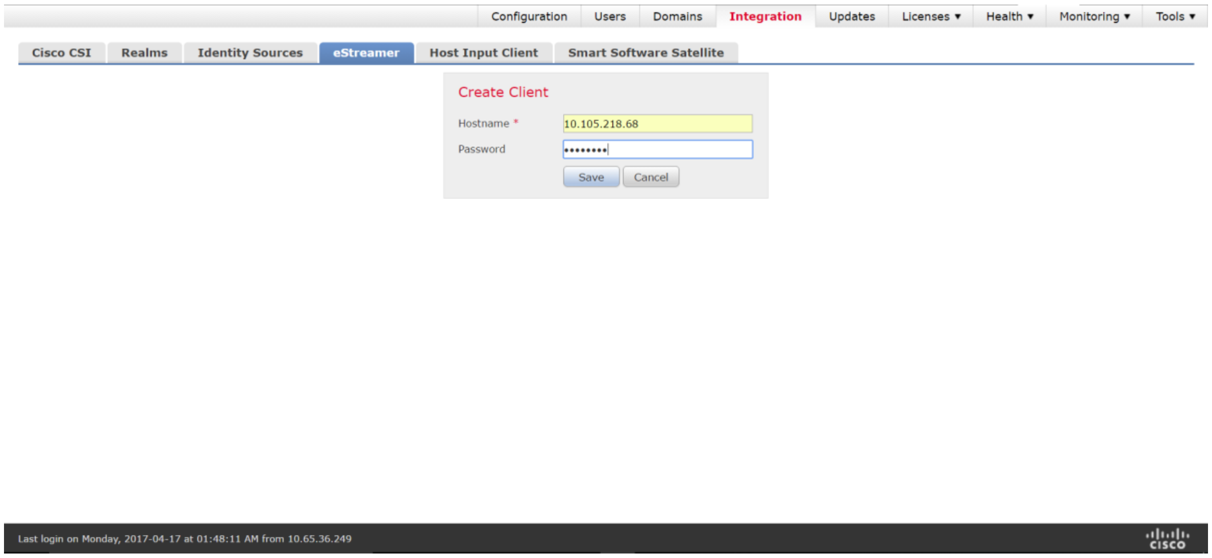
Please note that the IP address you enter here must be the IP address of the eStreamer-eNcore client *from the perspective of the FMC*. In other words, if the client is behind a NAT device, then the IP address must be that of the upstream NAT interface.

### Create Client Hostname and Password Screen



5. Click Save.

### Create Client Save Screen



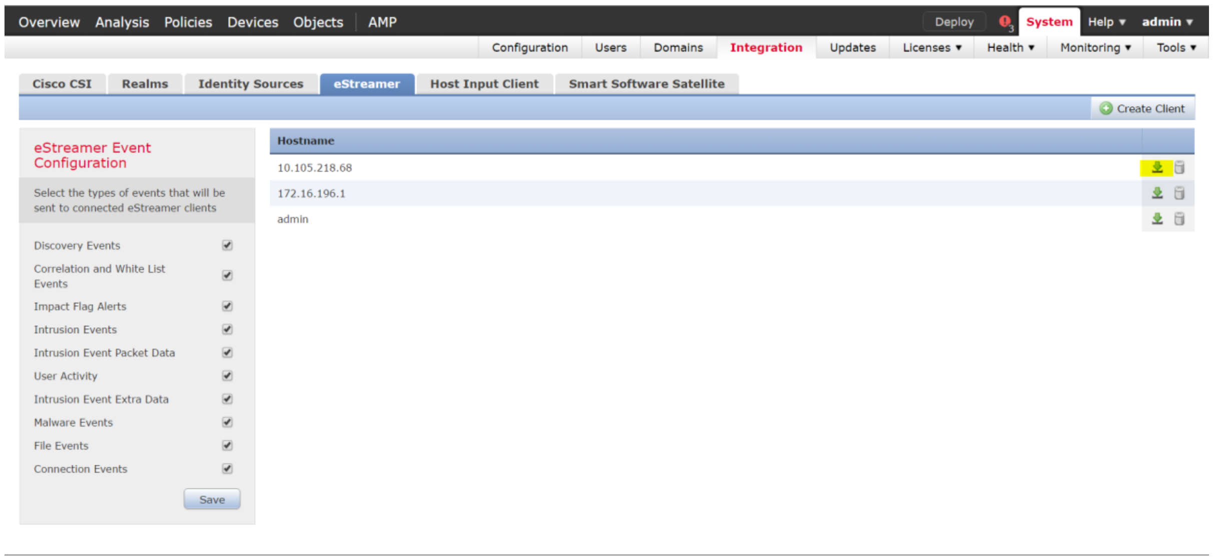
Hostname * 10.105.218.68

Password ••••••••

Last login on Monday, 2017-04-17 at 01:48:11 AM from 10.65.36.249

6. Download the PKCS12 file by clicking the Download icon at the right.

### Download Screen



eStreamer Event Configuration

Select the types of events that will be sent to connected eStreamer clients

| | |
|---|---|
| Discovery Events | ✓ |
| Correlation and White List Events | ✓ |
| Impact Flag Alerts | ✓ |
| Intrusion Events | ✓ |
| Intrusion Event Packet Data | ✓ |
| User Activity | ✓ |
| Intrusion Event Extra Data | ✓ |
| Malware Events | ✓ |
| File Events | ✓ |
| Connection Events | ✓ |

| Hostname |
|---|
| 10.105.218.68 |
| 172.16.196.1 |
| admin |

7. Copy the PKCS12 file to the desired location in the target device. By default, eNcore will look for the file named client.pkcs12 in the **`/path/eStreamer-eNcore`** directory, or **`/path/eStreamer-eNcore/client.pkcs12`**. If you wish to use a different filename, then you must edit the estreamer.conf file.

# 10 Appendix B: List of Configuration Options

The following table provides a comprehensive list of configuration options for eNcore CLI.

| Key | Definition |
|---|---|
| Enabled | true \| false. Controls whether eNcore will run. |
| connectTimeout | The duration in seconds the client will wait for a connection to establish before failing. |
| responseTimeout | The duration in seconds the client will wait for a response before timing out. |
| monitor.details | true \| false. Controls whether the monitor writes a detailed status message to the log. |
| monitor.period | The period in seconds between each execution of monitor tasks. Default is 120. Lower numbers are useful for debugging but will create more log traffic. |
| monitor.bookmark | true \| false. True will show the last bookmark timestamp. This is useful to see how far behind the eNcore client is. |
| monitor.handled | true \| false. True will report the total number of events written to output. |
| Start | 0 specifies oldest data available<br><br>1 specifies data as of now<br><br>2 specifies use of bookmark |
| logging.level | Levels include FATAL, ERROR, WARNING, INFO, DEBUG, VERBOSE, and TRACE. Select the level of logging as per your requirement. It is strongly recommended that you do not use anything above INFO for production environments. DEBUG will generate very large log files and TRACE will significantly affect performance. |
| logging.format | This describes the format of the log and how they are stored. Default configuration setting for message format is "{date-time}-{name of module}-{level of logging-message}". |
| logging.stdOut | true \| false. This determines whether log output is also shown in Standard Output. |

| Key | Definition |
|---|---|
| logging.filepath | This specifies the location of the application log. |
| subscription.servers[] | While this is an array, eNcore can only currently support one server. The array is to support the future ability to connect to multiple hosts. |
| server.host | The IP address of the FMC (eStreamer Server). Default configuration is 1.2.3.4. If you change the host entry after having run eNcore then new cache, bookmark and metadata files will be generated. |
| server.port | The server port to connect to. Default 8302. |
| server.pkcs12Filepath | The PKCS12 filepath location. If you change this having already run eNcore, then you must also delete the cached public and private key otherwise eNcore will continue to use those. They are called {host}-{port}_pkcs.cert and {host}-{port}_pkcs.key. |
| server.tlsVersion | Valid options are 1.0 and 1.2. |
| subscription.records | Do not change these values. |
| handler.records.metadata | true \| false. If you wish to exclude the output of metadata (since it has no timestamp information) then set this to false. |
| handler.records.flows | true \| false. If you wish to exclude connection flow records then set this to false. |
| handler.outputters[] | An array of outputter controllers which define the behavior and format of what gets written by eNcore. |
| outputter.name | This is a human readable name for your convenience. It is unused by the code. |
| outputter.adapter | Data is read from eStreamer and stored in a structured internal format. The adapter transforms the data to a desired format. Recognized values are: — cef — splunk — json |

| Key | Definition |
|---|---|
| outputter.enabled | true \| false. You can have more than one outputter specified at once. If you wish to disable a specific out-putter, set this flag to false. If all outputters are false (or there are no outputters) then it behaves as a sink. |
| outputter.passthru | true \| false. If true then data flowing through bypasses decoding and metadata processing. It is very fast but of limited use. Its primary purpose is for debugging. |
| outputter.stream.uri | Specify the location where the output will be stored. You can specify a file URI as normal (e.g., file:///absolute/path/to/file) or a relative filepath (rel-file:////relative/path/to/file).<br><br>Only file URLs are supported currently. |
| outputter.stream.options | File-based streams require additional options. |
| option.rotate | true \| false. Set if you want log rotation. Default con-figuration setting for this is true. Please note that eN-core will not delete any old files. If you wish to do that, you will need to script it separately and schedule it. |
| option.maxLogs | Specify the size of the log (number of lines). *Default configuration for this is 10,000. You can have fewer, larger files (e.g, 50,000).* |
| workerProcesses | Specify the number of worker processes, from 1 to 12, to optimize performance. See section 6.4 for guidance on setting this option. |
| batchSize | Specify the threshold for a batch. See section 6.5 for guidance on setting the batch size. |

# 11 Appendix C: Example Configuration File

Section 4 of this Guide described sections of the configuration file and how to configure many of the options. Appendix B provided a table listing all the options and how to set them. For reference purposes, an example configuration file is provided below.

## Example estreamer.conf Configuration File

```
{
  "connectTimeout":  10,
  "responseTimeout":  10,


  "@startComment": "0 for genesis, 1 for now, 2 for bookmark",
  "start": 2,


  "monitor": {
    "period": 120,
    "bookmark": false,
    "handled": true,
    "details": true
  },


  "logging": {
    "@comment": "Levels include FATAL, ERROR, WARNING, INFO, DEBUG, VERBOSE and TRACE",
    "level": "INFO",
    "format": "%(asctime)s %(name)-12s %(levelname)-8s %(message)s",
    "stdOut": true,
    "filepath": "estreamer.log"
  },

  "subscription": {
    "servers": [
      {
        "host": "1.2.3.4",
        "port": 8302,
        "pkcs12Filepath": "client.pkcs12",
        "@comment": "Valid values are 1.0 and 1.2",
        "tlsVersion": 1.2
      }
    ],


    "records": {
      "@comment": [
        "Just because we subscribe doesn't mean the server is sending. Nor does it mean",
        "we are writing the records either. See handler.records[]"
      ],
      "packetData": true,
      "extended": true,
      "metadata": true,
      "eventExtraData": true,
      "impactEventAlerts": true,
      "intrusion": true,
      "archiveTimestamps": true
    }
  },


  "handler": {
    "records": {
      "core": true,
      "metadata": true,
      "flows": true,
      "packets": true,
      "intrusion": true,
      "rua": true,
      "rna": true,


      "@includeComment": "These records will be included regardless of above",
      "include": [],


      "@excludeComment": [
        "These records will be excluded regardless of above (overrides 'include')",
        "e.g. to exclude flow and IPS events use []"
      ],
      "exclude": []
    },


    "@comment": "If you disable all outputters it behaves as a sink",
    "outputters": [
      {
        "adapter": "cef",
        "enabled": true,
        "name": "CEFarcsight",
        "stream": {
            "uri": "udp://192.168.1.160:514"
        }
      }
    ]
  }
}
```

## Trademarks and Disclaimers