


# Configuring eStreamer

After you create a client application, you can connect it to the eStreamer server, start the eStreamer service, and begin exchanging data.

**Note** An *eStreamer server* is a Management Center or managed device (version 4.9 or higher) where the eStreamer service is running.

Perform the following tasks to manage eStreamer and client interaction:

1. Enable eStreamer on the eStreamer server.

   See Configuring eStreamer on the eStreamer Server, page 6-1 for information about allowing access to the eStreamer server, adding clients, and generating authentication credentials to establish an authenticated connection.

2. If required, manually run the eStreamer service (eStreamer). You can stop, start, and view the status of the service, and use command line options to debug client-server communication.

   See Managing the eStreamer Service, page 6-4 for more information.

3. Optionally, to use the eStreamer reference client to troubleshoot a connection or data stream, set up the reference client on the computer where you plan to run your client.

   See Configuring the eStreamer Reference Clients, page 6-6.

# Configuring eStreamer on the eStreamer Server

**License:** Any

Before the Management Center or managed device you want to use as an eStreamer server can begin streaming events to a client application, you must configure the eStreamer server to send events to clients, provide information about the client, and generate a set of authentication credentials to use when establishing communication. You can perform all of these tasks from the Management Center or managed device user interface.

See the following sections for more information:

- Configuring eStreamer Event Types, page 6-2
- Adding Authentication for eStreamer Clients, page 6-3

# Configuring eStreamer Event Types

**License:** Any

You can control which types of events the eStreamer server is able to transmit to client applications that request them.

Available event types on a managed device or a Management Center include:

- Intrusion events
- Intrusion event packet data
- Intrusion event extra data

Available event types on a Management Center include:

- Discovery events (this also enables connection events)
- Correlation and allow list events
- Impact flag alerts
- User activity events
- Malware events
- File events

Note that the primary and secondary in a stacked 3D9900 pair report intrusion events to the Management Center as if they were separate managed devices. If you configure communication with an eStreamer client on the primary in a 3D9900 stack, you also need to configure the client on the secondary; the client configuration is not replicated. Similarly, when you delete the client, delete it in both places. If you configure an eStreamer client for a Management Center managing 3D9900s in a stack configuration, note that the Management Center reports all events received from both managed devices, even if the same event is reported by both.

If you configure an eStreamer client on a Management Center in a high availability configuration, the client configuration is not replicated from the primary Management Center to the secondary Management Center.

**To configure the types of events captured by eStreamer:**

**Access:** Admin

**Step 1** Select **System > Integration > eStreamer**.

**Step 2** Click **eStreamer**.

The eStreamer page appears with the **eStreamer Event Configuration** menu.

**Step 3** Select the check boxes next to the types of events you want eStreamer to capture and forward to requesting clients. Note that if a check box is currently unchecked, that data is not being captured. Unchecking a check box does not delete data that has already been captured.

You can select any or all of the following on a Management Center or managed device:

- **Intrusion Events** to transmit intrusion events generated by managed devices.
- **Intrusion Event Packet Data** to transmit packets associated with intrusion events.
- **Intrusion Event Extra Data** to transmit additional data associated with intrusion events, such as the URI associated with the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

You can also select any or all of the following on a Management Center:

- **Discovery Events** to transmit host discovery events
- **Correlation Events** to transmit correlation and allow list events.
- **Impact Flag Alerts** to transmit impact alerts generated by the Management Center.
- **User Activity Events** to transmit user events.
- **Intrusion Event Extra Data** to transmit additional data for intrusion events, such as the URI associated with the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

**Note** Note that this controls which events the eStreamer server can transmit. Your client application must still specifically request the types of events you want it to receive. For more information, see Request Flags, page 2-13.

**Step 4** Click **Save**.

Your settings are saved and the events you selected will be forwarded to eStreamer clients when requested.

# Adding Authentication for eStreamer Clients

**License:** Any

Before eStreamer can send events to a client, you must add the client to the eStreamer server's peers database. You must also copy the authentication certificate generated by the eStreamer server to the client.

**To add an eStreamer client:**

**Access:** Admin

**Step 1** Select **System > Integration > eStreamer**.

The **eStreamer** page appears.

**Step 2** Click **Create Client**.

The Create Client page appears.

**Step 3** In the **Hostname** field, enter the host name or IP address of the host running the eStreamer client.

**Note** If you use a host name, the host input server **must** be able to resolve the host to an IP address. If you have not configured DNS resolution, you should configure it first or use an IP address.

**Step 4** If you want to encrypt the certificate file, enter a password in the **Password** field.

**Step 5** Click **Save**.

The eStreamer server allows the client computer to access port 8302 on the Management Center and creates an authentication certificate to use during client-server authentication. The eStreamer Client page re-appears, with the new client listed under **eStreamer Clients**.

**Step 6** Click the download icon ( ) next to the certificate file.

**Step 7** Save the certificate file to the directory used by your client computer for SSL authentication.

The client can now connect to the Management Center.

**Tip** To revoke access for a client, click the delete icon ( ) next to the host you want to remove. Note that you do not need to restart the host input service on the Management Center; access is revoked immediately.

# Managing the eStreamer Service

**License:** Any

You can manage the eStreamer service from the user interface. However, you can also use the command line to start and stop the service. The following sections describe eStreamer command line options:

- Starting and Stopping the eStreamer Service, page 6-4 describes how to start and stop the eStreamer service.
- eStreamer Service Options, page 6-4 describes the command line options available for the eStreamer service and how to use them.

## Starting and Stopping the eStreamer Service

**License:** Any

You can manage the eStreamer service using the `manage_estreamer.pl` script, which allows you to start, stop, reload, and restart the service.

**Tip** You can also add command line options to the eStreamer initialization script. See eStreamer Service Options, page 6-4 for more information.

The following table describes the options in the `manage_estreamer.pl` script you can use on the Management Center or managed device.

*Table 6-1 eStreamer Management Options*

| Option | Description | Select option Number.... |
|---|---|---|
| enable | Starts the service. | `3` |
| disable | Stops the service. | `2` |
| restart | Restarts the service. | `4` |
| status | Indicates whether the service is running. | `1` |

## eStreamer Service Options

**License:** Any

eStreamer provides many service options that allow you to troubleshoot the service. You can use the options described in the following table with the eStreamer service.

*Table 6-2 eStreamer Service Options*

| Option | Description |
|---|---|
| `--debug` | Runs eStreamer with debug-level logging. Errors are saved in the syslog and (when used in conjunction with `--nodaemon`) appear on screen. |
| `--nodaemon` | Runs eStreamer as a foreground process. Errors appear on-screen. |
| `--nohostcheck` | Runs eStreamer with host name checking disabled. That is, if the client host name does not match the host name contained in the subjectAltName:dNSName entry in the client certificate, access is still allowed. The `nohostcheck` option is useful in cases where the network DNS and/or NAT configuration prevent the host name check from succeeding. Note that all other security checks are performed.<br>**Caution** Enabling this option can negatively affect the security of your system. |

Use the above options by first stopping the eStreamer service, then running it with the options you want, and finally restarting the service. For example, you can follow the instructions provided in Running the eStreamer Service in Debug Mode, page 6-5 to debug eStreamer functionality.

## Running the eStreamer Service in Debug Mode

**License:** Any

You can run the eStreamer service in debug mode to view each status message the service generates on your terminal screen. Use the following procedure to do debugging.

**To run the eStreamer service in debug mode:**

**Access:** Admin

**Step 1** Log into the Management Center or managed device using SSH.

**Step 2** Use `manage_estreamer.pl` and select option `2` to stop the eStreamer service.

**Step 3** Use `./usr/local/sf/bin/sfestreamer --nodaemon --debug` to restart the eStreamer service in debug mode.

Status messages for the service appear on the terminal screen.

**Step 4** When you are finished debugging, restart the service in normal mode using `manage_estreamer.pl` and selecting option `4`.

# Configuring the eStreamer Reference Clients

The *reference clients* provided with the eStreamer SDK are a set of sample client scripts and Perl modules, as well as Python scripts, included to illustrate how the eStreamer API can be used. You can run them to familiarize yourself with eStreamer output, or you can use them to debug problems with installations of your custom-built client.

For more information on setting up the reference client, see the following sections:


- Setting Up the eStreamer Reference Clients, page 6-6
- Running the eStreamer Perl Reference Client, page 6-11
- Running the eStreamer Python Reference Client, page 6-13


## Setting Up the eStreamer Reference Clients

To use the eStreamer reference clients, you must first configure the sample scripts to fit your environment and requirements.

For more information, see the following sections:


- Downloading the eStreamer Reference Clients, page 6-6
- Configuring Communications for the eStreamer Reference Clients, page 6-7
- Loading General Prerequisites for the Perl Reference Client, page 6-8
- Loading Prerequisites for the Perl SNMP Reference Client, page 6-8
- Understanding the Data Requested by a Perl Test Script, page 6-9
- Modifying the Type of Data Requested by a Perl Test Script, page 6-10
- Creating a Certificate for the Reference Clients, page 6-7


### Downloading the eStreamer Reference Clients

You can download the `eStreamerSDK.zip` package, which contains the eStreamer reference client files, from the Cisco support site. The following files are included in the `eStreamerSDK.zip` package:

- `SF_CUSTOM_ALERT.MIB`

  This MIB file is used by the `snmp.pm` file to set up traps for SNMP.

- `SFRecords.pm`

  This Perl module contains definitions of discovery message record blocks.

- `SFStreamer.pm`

  This Perl module contains the functions called by the Perl clients.

- `SFPkcs12.pm`

  This Perl module parses the client certificate and allows the client to connect to the eStreamer server.

- `SFRNABlocks.pm`

  This Perl module contains definitions of discovery data blocks.

- `ssl_test.pl`

  You can use this Perl script to test an intrusion event request over an SSL connection.

- `OutputPlugins/csv.pm`

  This Perl module prints intrusion events to a comma-separated value (CSV) format.

- `OutputPlugins/print.pm`

  This Perl module prints events to a human-readable format.

- `OutputPlugins/snmp.pm`

  This Perl module sends events to the specified SNMP server.

- `OutputPlugins/pcap.pm`

  This Perl module stores packet captures as a pcap file.

- `python_client/estreamer_client.py`

  You can use this Python script to test an intrusion event request over an SSL connection.

- `python_client/estreamer_connection.py`

  This Python script connects to the eStreamer server. It is necessary for the `estreamer_client.py`.

## Configuring Communications for the eStreamer Reference Clients

The reference client uses the Secure Sockets Layer (SSL) for data communication. You must install OpenSSL on the computer you plan to use as a client and configure it appropriately for your environment.

**Note** For initial installations on Linux operating systems, you must install the `libssl-dev` component as part of this download.

**To set up SSL on your client:**

**Step 1** Download OpenSSL from http://openssl.org/source/.

**Step 2** Unpack the source to `/usr/local/src`.

**Step 3** Configure the source by running the Configure script.

**Step 4** Make and install the compiled source.

## Creating a Certificate for the Reference Clients

**License:** Any

Before you can use the reference clients, you need to create a certificate on the Management Center or managed device for the computer where you want to run the client. You then download the certificate file to the client computer and use it to create a certificate (`server.crt`) and RSA key file (`server.key`).

**To create a certificate for the Reference Client:**

**Access:** Admin

**Step 1** Select **System > Integration > eStreamer**.

The eStreamer page appears.

**Step 2** Click **Create Client**.

The Create Client page appears.

**Step 3** In the **Hostname** field, enter the host name or IP address of the host running the eStreamer client.

**Note** If you use a host name, the host input server **must** be able to resolve the host to an IP address. If you have not configured DNS resolution, you should configure it first or use an IP address.

**Step 4** If you want to encrypt the certificate file, enter a password in the **Password** field.

**Step 5** Click **Save**.

The eStreamer server allows the client computer to access port 8302 on the Management Center and creates an authentication certificate to use during client-server authentication. The eStreamer Client page re-appears, with the new client listed under **eStreamer Clients**.

**Step 6** Click the download icon ( ) next to the certificate file.

**Step 7** Save the certificate file to the directory used by your client computer for SSL authentication.

The client can now connect to the Management Center.

**Tip** To revoke access for a client, click the delete icon ( ) next to the host you want to remove. Note that you do not need to restart the host input service on the Management Center; access is revoked immediately.

## Loading General Prerequisites for the Python Reference Client

Before you can run the eStreamer Python reference client, you must???

## Loading General Prerequisites for the Perl Reference Client

Before you can run the eStreamer Perl reference client, you must install the `IO::Socket::SSL` Perl module on the client computer. You can install the module manually or use `cpan` to do so.

**Note** If the `Net::SSLeay` module is not installed on the client computer, install that module as well. `Net::SSLeay` is required for communication with OpenSSL.

You also need to install and configure OpenSSL to support an SSL connection to the eStreamer server. For more information, see Configuring Communications for the eStreamer Reference Clients, page 6-7.

## Loading Prerequisites for the Perl SNMP Reference Client

Before you can run the eStreamer SNMP module of the Perl reference client, you must install the latest `net-snmp` Perl modules available for the client operating system on the client computer.

### Downloading and Unpacking the Reference Clients

You can download the `EventStreamerSDK.zip` file that contains the eStreamer reference clients fromthe Cisco support site.

Unpack the zip file to a computer running the Linux operating system, where you plan to run the client.

## Understanding the Data Requested by a Perl Test Script

By default, when you use the `ssl_test -o` setting in the reference client, you request data as indicated in the following table.

***Table 6-3 Default Requests Made by Output Plugins***

| This syntax... | Calls plugin... | And sends... | To request the following data... |
|---|---|---|---|
| `./ssl_test.pl` *`eStreamerServerName`* `-h HostIPAddresses` | N/A | Host request, message type 5, with bit 11 set to `1` | Host data (see Host Data and Multiple Host Data Message Format, page 2-32) |
| `./ssl_test.pl` *`eStreamerServerName`* `-d "Global \ domain \ subdomain"` | N/A | Event stream request for the specified domain or subdomain. | Streaming event information for the specified domain (see Domain Streaming Request Message Format, page 2-36) |
| `./ssl_test.pl` *`eStreamerServerName`* `-o print -f` *`TextFile`* | OutputPlugins/print.pm | Event stream request, message type 2, with bits 2 and 20-24 set to `1` | Event data (see Event Stream Request Message Format, page 2-12, Correlation Policy Record, page 3-25, Correlation Rule Record, page 3-26, Metadata for Discovery Events, page 4-7, Host Discovery Structures by Event Type, page 4-44, and User Data Structures by Event Type, page 4-61)<br>eStreamer transmits type 1 intrusion events because bit 2 is set on the event stream request. |
| `./ssl_test.pl` *`eStreamerServerName`* `-o pcap -f` *`TargetPCAPFile`* | OutputPlugins/pcap.pm | Event stream request, message type 2, with bits 0 and 23 set to `1` | Packet data (see Event Data Message Format, page 2-19 and Packet Record 4.8.0.2+, page 3-5)<br>eStreamer transmits only packet data because bit 0 is set on the event stream request. |
| `./ssl_test.pl` *`eStreamerServerName`* `-o csv -f` *`CSVFile`* | OutputPlugins/csv.pm | Event stream request, message type 2, with bits 2 and 23 set to `1` | Intrusion event data (see Event Data Message Format, page 2-19 and Intrusion Event Record 7.1+, page 3-7)<br>eStreamer transmits type 1 intrusion events because bit 2 is set on the event stream request. |
| `./ssl_test.pl` *`eStreamerServerName`* `-o snmp -f` *`SNMPServer`* | OutputPlugins/snmp.pm | Event stream request, message type 2, with bits 2, 20, and 23 set to `1` | Intrusion event data (see Event Data Message Format, page 2-19 and Intrusion Event Record 7.1+, page 3-7)<br>eStreamer transmits type 1 intrusion events because bit 2 is set on the event stream request. |

*Table 6-3 Default Requests Made by Output Plugins (continued)*

| This syntax... | Calls plugin... | And sends... | To request the following data... |
|---|---|---|---|
| `./ssl_test.pl` *`eStreamerServerName`* `-o syslog` | OutputPlugins/ syslog.pm | Event stream request, message type 2, with bits 2, 20, and 23 set to `1` | Intrusion event data (see Event Data Message Format, page 2-19 and Intrusion Event Record 7.1+, page 3-7)<br>eStreamer transmits type 1 intrusion events because bit 2 is set on the event stream request. |
| `./ssl_test.pl` *`eStreamerServerName`* `json=<filename>` | N/A | Event stream request, message type 2, with bit 23 set to `1` and all other bits set to `0`. Sends JSON file named <filename> | Intrusion, Connection, and File event data in JSON format as provided. |

## Modifying the Type of Data Requested by a Perl Test Script

The `SFStreamer.pm` Perl module defines several request flag variables that you can use in the sample scripts to request data. The following table indicates what request flag variable to call to set each request flag in an event stream request message. If you want to request different data using one of the output modules, you can edit the `$FLAG` settings in the module.

For more information on the request flags, the data they request, and the product versions corresponding to each flag, see Request Flags, page 2-13.

*Table 6-4 Request Flag Variables Used in Sample Scripts*

| Variable | Sets Request Flag... | To request the following data... |
|---|---|---|
| `$FLAG_PKTS` | `0` | Packet data |
| `$FLAG_METADATA` | `1` | Version 1 metadata |
| `$FLAG_IDS` | `2` | Type 1 intrusion events |
| `$FLAG_RNA` | `3` | Version 1 discovery events |
| `$FLAG_POLICY_EVENTS` | `4` | Version 1 correlation events |
| `$FLAG_IMPACT_ALERTS` | `5` | Intrusion impact alerts |
| `$FLAG_IDS_IMPACT_FLAG` | `6` | Type 7 intrusion events |
| `$FLAG_RNA_EVENTS_2` | `7` | Version 2 discovery events |
| `$FLAG_RNA_FLOW` | `8` | Version 1 connection data |
| `$FLAG_POLICY_EVENTS_2` | `9` | Version 2 correlation events |
| `$FLAG_RNA_EVENTS_3` | `10` | Version 3 discovery events |
| `$FLAG_HOST_ONLY` | `11` | When sent in conjunction with `$FLAG_HOST_SINGLE` (for one host) or `$FLAG_HOST_MULTI` (for multiple hosts), only host data with no event data |
| `$FLAG_RNA_FLOW_3` | `12` | Version 3 connection data |
| `$FLAG_POLICY_EVENTS_3` | `13` | Version 3 correlation events |

*Table 6-4 Request Flag Variables Used in Sample Scripts (continued)*

| Variable | Sets Request Flag... | To request the following data... |
|---|---|---|
| `$FLAG_METADATA_2` | `14` | Version 2 metadata |
| `$FLAG_METADATA_3` | `15` | Version 3 metadata |
| `$FLAG_RNA_EVENTS_4` | `17` | Version 4 discovery events |
| `$FLAG_RNA_FLOW_4` | `18` | Version 4 connection data |
| `$FLAG_POLICY_EVENTS_4` | `19` | Version 4 correlation events |
| `$FLAG_METADATA_4` | `20` | Version 4 metadata |
| `$FLAG_RUA` | `21` | User activity events |
| `$FLAG_POLICY_EVENTS_5` | `22` | Version 5 correlation events |
| `$FLAGS_SEND_ARCHIVE_TIMESTAMP` | `23` | Extended event headers that include the timestamp applied when the event was archived for eStreamer server to process |
| `$FLAG_RNA_EVENTS_5` | `24` | Version 5 discovery events |
| `$FLAG_RNA_EVENTS_6` | `25` | Version 6 discovery events |
| `$FLAG_RNA_FLOW_5` | `26` | Version 5 connection data |
| `$FLAG_EXTRA_DATA` | `27` | Intrusion event extra data record |
| `$FLAG_RNA_EVENTS_7` | `28` | Version 7 discovery events |
| `$FLAG_POLICY_EVENTS_6` | `29` | Version 6 correlation events |
| `$FLAG_DETAIL_REQUEST` | `30` | Extended request to eStreamer |

**Caution** In all event types, prior to version 5.x, the reference client labels `detection engine ID` fields as `sensor ID`.

# Running the eStreamer Perl Reference Client

The eStreamer Perl reference client scripts are designed for use on a 64-bit operating system with the Linux kernel but should work on any POSIX-based 64-bit operating system, as long as the client machine meets the prerequisites defined in Setting Up the eStreamer Reference Clients, page 6-6.

For more information, see the following sections:


- Testing a Client Connection over SSL Using a Host Request, page 6-12
- Capturing a PCAP Using the Reference Client, page 6-12
- Capturing CSV Records Using the Reference Client, page 6-12
- Sending Records to an SNMP Server Using the Reference Client, page 6-12
- Logging Events to the Syslog Using the Reference Client, page 6-13
- Connecting to an IPv6 Address, page 6-13

## Testing a Client Connection over SSL Using a Host Request

You can use the `ssl_test.pl` script to test the connection between the eStreamer server and the eStreamer client. The `ssl_test.pl` script handles any record type and prints it to STDOUT or to an output plugin you specify. When you use the `-h` option without an output option, it streams host data for the specified hosts to your terminal.

**Note** You cannot use this script to stream packet data without directing it to an output plugin because printing raw packet data to STDOUT interferes with your terminal.

Use the following syntax to use the `ssl_test.pl` script to send host data to the standard output:

```
./ssl_test.pl eStreamerServerIPAddress -h HostIPAddresses
```

For example, to test receipt of host data for the hosts in the 10.0.0.0/8 subnet over a connection to an eStreamer server with an IP address of 10.10.0.4:

```
./ssl_test.pl 10.10.0.4 -h 10.0.0.0/8
```

## Capturing a PCAP Using the Reference Client

You can use the reference client to capture streamed packet data in a PCAP file to see the structure of the data the client receives. Note that you must use `-f` to specify a target file when you use the `-o pcap` output option.

Use the following syntax to capture streamed packet data in a PCAP file using the `ssl_test.pl` script:

```
./ssl_test.pl eStreamerServerIPAddress -o pcap -f ResultingPCAPFile
```

For example, to create a PCAP file named `test.pcap` using events streamed from an eStreamer server with an IP address of 10.10.0.4:

```
./ssl_test.pl 10.10.0.4 -o pcap -f test.pcap
```

## Capturing CSV Records Using the Reference Client

You can also use the reference client to capture streamed intrusion event data in a CSV file to see the structure of the data the client receives.

Use the following syntax to run the `streamer_csv.pl` script:

```
./ssl_test.pl eStreamerServerIPAddress -o csv -f ResultingCSVFile
```

For example, to create a CSV file named `test.csv` using events streamed from an eStreamer server with an IP address of 10.10.0.4:

```
./ssl_test.pl 10.10.0.4 -o csv -f test.csv
```

## Sending Records to an SNMP Server Using the Reference Client

You can also use the reference client to stream intrusion event data to an SNMP server. Use the `-f` option to indicate the name of the SNMP trap server that should receive events. Note that this output method requires a binary named `snmptrapd` in the path and therefore only works on UNIX-like systems.

Use the following syntax to send intrusion events to an SNMP server:

```
./ssl_test.pl eStreamerServerIPAddress -o snmp
    -f SNMPServerName
```

For example, to send events to an SNMP server at 10.10.0.3 using events streamed from an eStreamer server with an IP address of 10.10.0.4:

```
./ssl_test.pl 10.10.0.4 -o snmp -f 10.10.0.3
```

### Logging Events to the Syslog Using the Reference Client

You can also use the reference client to stream intrusion events to the local syslog server on the client.

Use the following syntax to send events to the syslog:

```
./ssl_test.pl eStreamerServerIPAddress -o syslog
```

For example, to log events streamed from an eStreamer server with an IP address of 10.10.0.4:

```
./ssl_test.pl 10.10.0.4 -o syslog
```

### Connecting to an IPv6 Address

You can use the reference client to connect to a Management Center with an IPv6 address through the primary management interface. You must have the Socket6 and IO::Socket::INET6 Perl modules installed on the client machine and use the `-ipv6` option or the shortened form `-i`.

Use the following syntax to specify an IPv6 address using the ssl_test.pl script:

```
./ssl_test.pl -ipv6 eStreamerServerIPAddress
```

or

```
./ssl_test.pl -i eStreamerServerIPAddress
```

For example, to connect to a Management Center with the IPv6 address `2001:470:e09c:20:7c1e:5248:1bf7:2ea0` use the following:

```
./ssl_test.pl -ipv6 2001:470:e09c:20:7c1e:5248:1bf7:2ea0
```

## Running the eStreamer Python Reference Client

The eStreamer Python reference client script demonstrates a new and much simpler mechanism for getting event data from the Secure Firewall System Management Center eStreamer service. Instead of returning event information as binary data the events are returned as fully-qualified text in formats such as JSON or CSV.

This API only supports requesting information for three event types: connection events, intrusion events, and file events. For all other events you must use a separate client and the regular method documented in the eStreamer Integration Guide.

The Python code provides a simple example client which uses the new mechanism. The Perl sample client code has also been modified to optionally use this new mechanism (using the `json=<filename>` command line argument), but the Python example is much easier to follow since it only supports the new mechanism.

Sample usage:

```
./estreamer_client.py --server 192.168.1.1 --configfile json_request.json --pkcs12_file
192.168.1.2_8.pkcs12 --start all
```

*Table 6-5 Python Script Arguments*

| This argument... | Does the following... |
|---|---|
| `-h, --help` | shows this help message and exits. |
| `--server SERVER` | Specifies the IP address of eStreamer server. This IP address must be accessible from the machine running the client. |
| `--port` PORT | Specifies the port of eStreamer server. Default is `8302` |
| `--configfile` CONFIGFILE | Gives the JSON formatted configuration file. See Format of the JSON File, page 2-5 for more information. |
| `--pkcs12_file` PKCS12_FILE | Gives the Pkcs12 file for authentication to the eStreamer server. |
| `--pkcs12_password` PKCS12_PASSWORD | Gives the Pkcs12 password, if necessary. |
| `--debug` | Enables debugging mode. |
| `--start` {now,all,bookmark} | Starting time to stream events |
| `--outfile` OUTFILE | Output file to store events. Default is to print to stdout |