

Halo Event Connector

Integration with Sumo Logic

Prerequisites	1
How the Connector Works	1
A. Retrieve and Save Your CloudPassage API Key	3
B. Test the Connector Standalone	4
C. Set up and Activate the Connector in Sumo Logic	6
D. View Halo Events in Sumo Logic.....	9

This document describes the CloudPassage Halo Event Connector and explains how you can configure the connector to import Halo event data into Sumo Logic’s Cloud Service.

Prerequisites

To get started, you must have the following privileges and software resources:

- An active CloudPassage Halo subscription. If you don't have one, [Register for CloudPassage](#) to receive your credentials and further instructions by email.
- Access to your CloudPassage API key. Best practice is to create a new read-only key specifically for use with this script.
- Python 2.6 or later. You can download Python from [here](#).
- Access to Sumo Logic’s Cloud Service. You can sign up for Sumo Logic’s Cloud Service by going [here](#).
- Access to a computer running a Sumo collector sending data to your Sumo Logic Cloud Service account. See [here](#) for instructions on how to install and configure a Sumo Logic Collector.
- The Event Connector script (`haloEvents.py`) and its associated files.

Note: The Event Connector makes calls to the CloudPassage Events API, which is available to all Halo subscribers at all levels (including Basic). Many other parts of the CloudPassage API are available only to Halo users with a [NetSec](#) or [Professional](#) subscription; if you want to use those parts of the API, you can upgrade your subscription on the Manage Subscription page of the Halo Portal.

How the Connector Works

The purpose of the Halo Event Connector is to retrieve event data from a CloudPassage Halo account and import it into an external tool—such as Sumo Logic or Splunk Enterprise—for indexing or processing. The Connector is a Python script that is designed to execute repeatedly, keeping the external tool up-to-date with Halo events as time passes and new events occur:

- The first time the Connector runs, it by default retrieves all logged events from a single Halo account. Then the Connector creates a file, writes the timestamp of the last-retrieved event in it, and saves it in the current directory. However:
 - You can specify up to five Halo accounts to extract events from simultaneously.
 - Instead of retrieving all events when the script runs the first time, you can retrieve only events after a certain point by using the **--starting=datetime** command-line option.
 - You can store the timestamp in a directory of your choice by specifying it in the **--configdir=dirname** command-line option.
- Every subsequent time it runs, the Connector retrieves only those events that were created after the timestamp stored in the file. At the end of the run, the script updates the file with the timestamp of the last-retrieved event during that run.

Note: The **--starting=datetime** option applies only the first time that you run the script. Each subsequent execution starts from the timestamp of the last-retrieved event.

- During any script run, if no new events have occurred since the last run, no events are retrieved or imported into the external tool.

Output formats. The Event Connector receives event data from Halo in Halo's native JSON format. The Connector supports emitting the event data in several formats:

- **JSON** (the default): standard JSON format, with a line return after each event description.
- **syslog**: standard syslog format (*bitmask string*, where *bitmask* defines the event's facility (default = USER) and priority (default = INFO)), sent to the local syslog daemon.
- **key-value pairs**: Space-separated pairs in the form *attribute=value*, saved to a file or sent to standard output (terminal).

Command line arguments. In Python, you execute the Connector script with a command like this:

```
$ haloEvents.py arguments
```

To view the set of supported command-line arguments, launch the script with the argument `-?` or `-h` to view the usage page. These are the arguments:

<code>-?</code>	Print the usage page.
<code>--auth=filename</code>	Full pathname to the file that holds the Halo API key info.
<code>--starting=datetime</code>	Start retrieving events from this (ISO-8601) date-time. (Applies to initial script run only.)
<code>--configdir=dirname</code>	Full pathname to directory holding the timestamp of the last-retrieved event.
<code>--jsonfile=filename</code>	Save the output (in JSON format) to the path <i>filename</i> .
<code>--kvfile=filename</code>	Save the output (in key-value format) to the path <i>filename</i> .
<code>--kv</code>	Save the output (in key-value format) to standard output
<code>--txtsyslog</code>	Send the output (in format "value1 value2 ...valueN") to syslog.
<code>--kvsyslog</code>	Send the output (in key-value format) to syslog.

Note: The default event-output format is JSON to standard output (terminal).

Authentication to the Halo API. Halo requires the Connector to pass both the key ID and secret key values for a valid Halo API key in order to obtain the event data. You pass those values in a file named by default `haloEvents.auth`, located in the same directory as `haloEvents.py` and its associated script files. The format for the file is described in [Section A](#).

Alternatively, you can pass those values in a different file by specifying the full path to the file in the `--auth=filename` option.

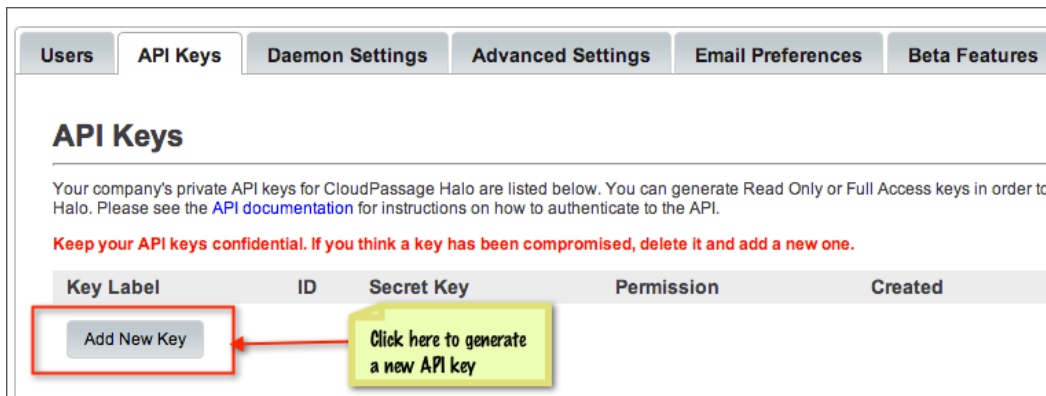
Output to a file. Whenever it writes event data to a disk file, the Connector appends the new data to the end of any existing data in the file.

Platform support. The Event Connector runs on both Linux and Windows operating systems.

A. Retrieve and Save your CloudPassage API Key

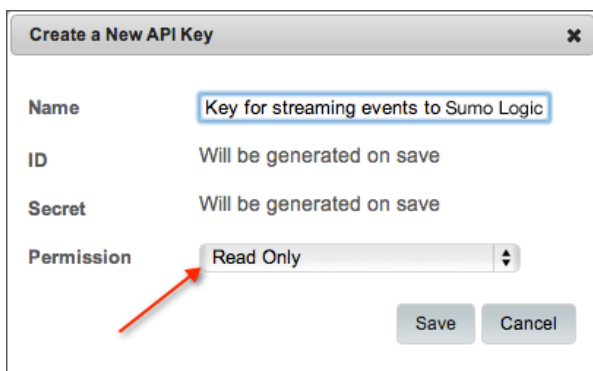
The Connector retrieves events from your CloudPassage Halo account by making calls to the CloudPassage API. The API requires the script to authenticate itself during every session; therefore, you need to make your CloudPassage API Key available to the script.

To retrieve your CloudPassage API key, log into the [CloudPassage Portal](#) and navigate to **Settings > Site Administration** and click the **API Keys** tab. (If you haven't generated an API key yet, do so by clicking **Add New Key**.)



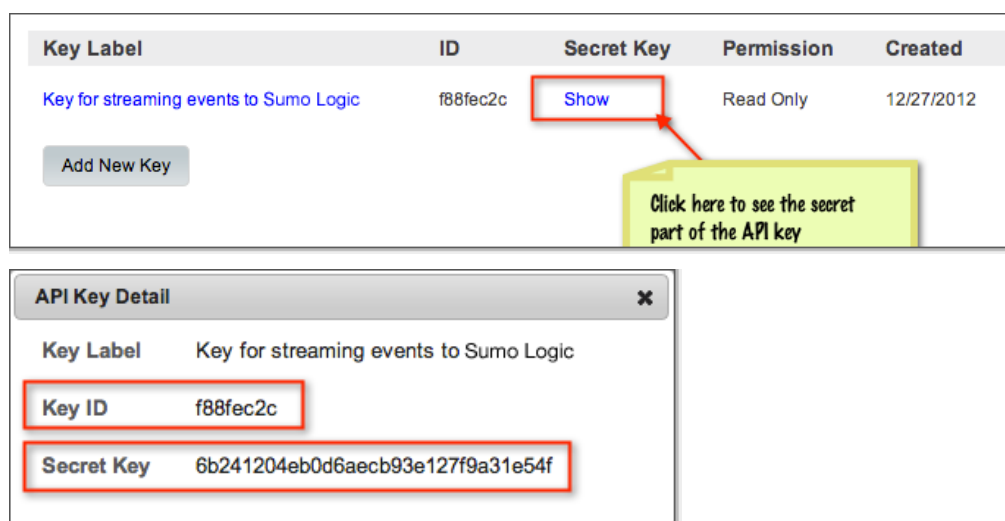
The screenshot shows the 'API Keys' section of the CloudPassage portal. At the top, there are navigation tabs: Users, API Keys (selected), Daemon Settings, Advanced Settings, Email Preferences, and Beta Features. Below the tabs, the title 'API Keys' is displayed. A paragraph explains that private API keys are listed below and that users can generate Read Only or Full Access keys. A red warning message states: 'Keep your API keys confidential. If you think a key has been compromised, delete it and add a new one.' Below this is a table with the following columns: Key Label, ID, Secret Key, Permission, and Created. A red box highlights the 'Add New Key' button in the 'Key Label' column. A yellow callout box with an arrow points to this button, containing the text 'Click here to generate a new API key'.

If you do create an API key, we recommend that, as a best practice, you create a read-only key. A read-only key is all that you need to be able to retrieve Halo event data.



The screenshot shows a 'Create a New API Key' dialog box. It has a title bar with a close button. The form contains the following fields: 'Name' with the value 'Key for streaming events to Sumo Logic'; 'ID' with the value 'Will be generated on save'; 'Secret' with the value 'Will be generated on save'; and 'Permission' with a dropdown menu set to 'Read Only'. A red arrow points to the 'Read Only' option in the dropdown. At the bottom right, there are 'Save' and 'Cancel' buttons.

You will need to retrieve both the **Key ID** and the **Secret Key** values for the API key. Click **Show** for your key on the **API Keys** tab to display both values.



Key Label	ID	Secret Key	Permission	Created
Key for streaming events to Sumo Logic	f88fec2c	Show	Read Only	12/27/2012

Add New Key

API Key Detail [X]

Key Label: Key for streaming events to Sumo Logic

Key ID: f88fec2c

Secret Key: 6b241204eb0d6aecb93e127f9a31e54f

Copy the ID and the secret into a text file so that it contains just one line, with the key ID and the secret separated by a vertical bar ("|"):

```
your_key_id|your_secret_key
```

Note: If you want to stream events from multiple Halo accounts, add one additional line to this file for each account, containing the account's key ID and secret key formatted as above.

Save the file as `haloEvents.auth` (or any other name, if you will be using the `--auth` command option). You will need this authentication file to run the Connector (in [Section B](#) and [Section C](#)).

B. Test the Connector Standalone

We recommend that you execute the Connector script standalone first, to get familiar with the different input switches and output formats it supports. Then you can choose the options that best suit your needs in Sumo Logic.

- Place all of the script-related files in the same directory. That is:
 - `haloEvents.py`
 - `cpapi.py` and `cputils.py`
 - `remote_syslog.py` (if you are running on Windows *and* you want to generate syslog output)
 - `haloEvents.auth` (unless you will use the `--auth` command option, in which case the authentication file can be anywhere.)
- Set environment variables as necessary:

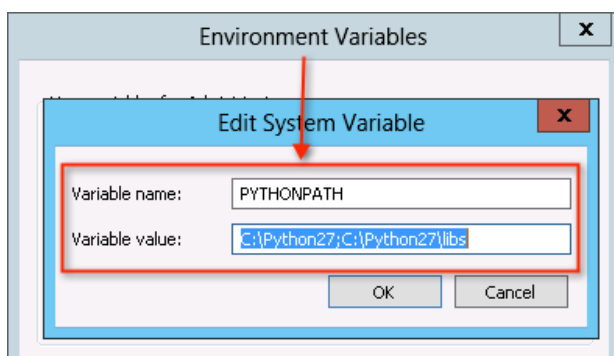
On Linux:

 - Include the full path to the Python interpreter in the PATH environment variable.

```
[root@ip-10-253-21-19 ~]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/home/ec2/bin:/usr/local/sbin:/usr/local/bin:/sbin:
[root@ip-10-253-21-19 ~]# which python
/usr/local/bin/python
[root@ip-10-253-21-19 ~]#
```

On Windows:

- o Set the variable PATH to include the location of `haloEvents.py` and the Python interpreter.
- o Set the variable PYTHONPATH to include the location of the Python libraries and the Python interpreter.



3. Launch the Connector from that directory, with a command like this:

```
$ haloEvents.py
```

Since the arguments are defaulted, you should soon see JSON-formatted event values streaming to output. You may want to abort execution if your Halo account has accumulated a large number of events.

4. Run the script a few more times, experimenting with arguments to save output to a file, or to produce other output formats. (A syslog daemon must be running if you want to output syslog format. On Linux systems, the syslog daemon typically stores the data at `/var/log/messages`.)

C. Set Up and Activate the Connector in Sumo Logic

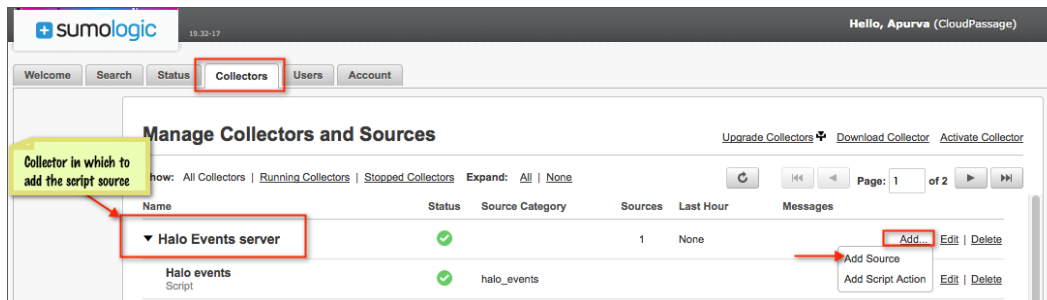
The main purpose of the Halo Event Connector when integrated into a tool such as Sumo Logic is to feed event data to that tool. In the case of Sumo Logic, once it receives the data, it transforms it into a series of indexed Sumo Logic events, each consisting of searchable fields.

5. Place all of these files:
 - o `haloEvents.py`
 - o `cpapi.py` and `cputils.py`
 - o `remote_syslog.py` (if you are running on Windows *and* you want to generate syslog output)
 - o `haloEvents.auth` (or its equivalent in any location, if you are using the `--auth` option)

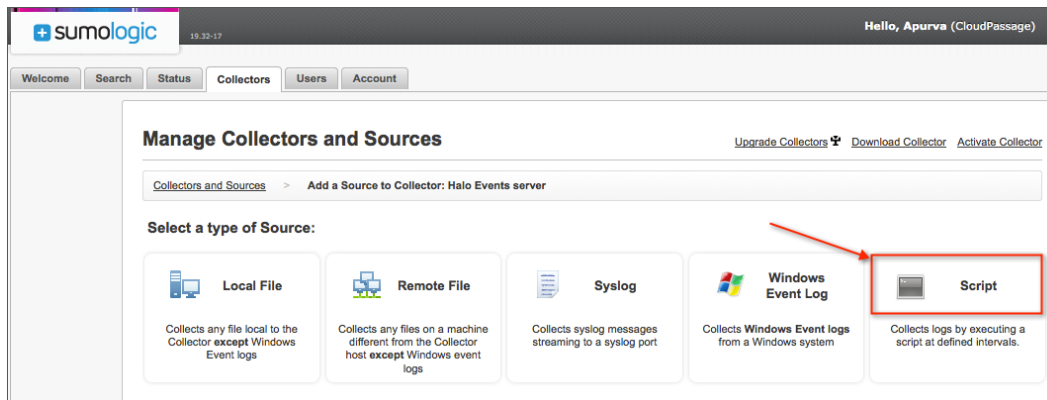
into the desired location on the server.
6. Make sure your PATH environment variable (and PYTHONPATH on Windows) is appropriately set, as shown in [Section B](#).

On either Linux or Windows, the Connector will emit the default output format (JSON).

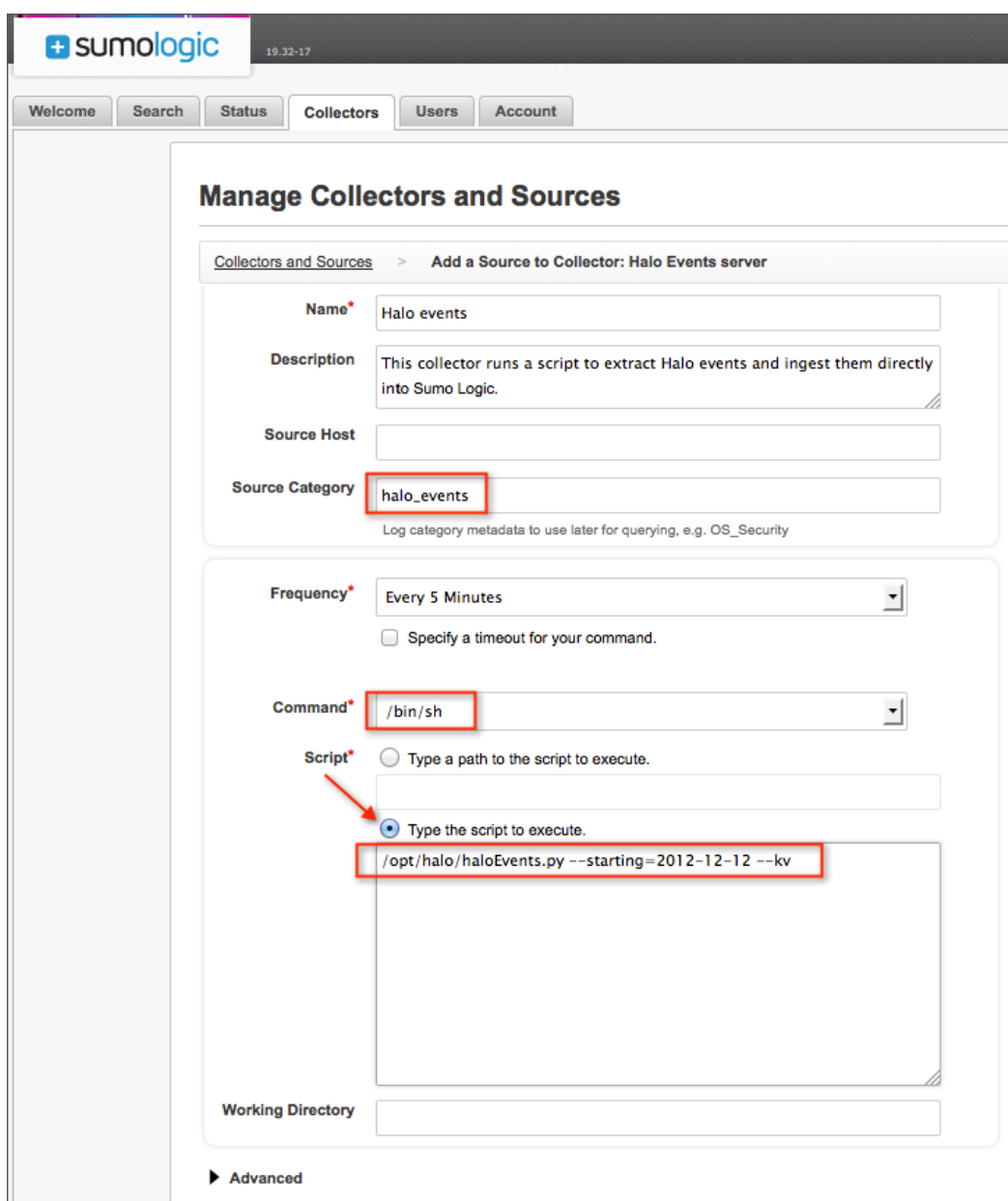
7. Now log into your [Sumo Logic Web Application](#).
8. In the Web Application, click the **Collectors** tab, and then click **Add** for the Collector you choose. Choose **Add Source** from the pop-up menu.



9. In the "Select a type of Source" page, select **Script** for the Source type.



The "Add a Source to Collector" dialog box opens.



The screenshot shows the Sumo Logic interface for adding a source. The form is titled "Manage Collectors and Sources" and is for "Add a Source to Collector: Halo Events server". The fields are as follows:

- Name:** Halo events
- Description:** This collector runs a script to extract Halo events and ingest them directly into Sumo Logic.
- Source Host:** (empty)
- Source Category:** halo_events
- Frequency:** Every 5 Minutes
- Command:** /bin/sh
- Script:** Type the script to execute. (Selected radio button)
- Working Directory:** (empty)

10. Fill in these fields

- **Name** — Enter a name to display for this Source in the Sumo Logic Web Application.
- **Frequency** —Enter the time in seconds between successive automatic executions of the Connector. In a production environment, a value for this field between 300 (5 minutes) and 86400 (1 day) might be reasonable, depending on the rate of event production from Halo and the desired immediacy of reporting in Sumo Logic.
- **Source Category** —Enter any information you'd like to include in the metadata. For example, entering say, "halo_events" will allow you to search for all indexed Halo events in Sumo Logic by typing `_sourceCategory = "halo_events"` in the search page.

- **Command**—Select `/bin/sh` for this. Note: Even though the Connector script is written in Python, don't select `/usr/bin/python` from the list.
- **Script**—Enter the full path to `haloEvents.py`. Also, add any command-line parameters that you want to use when executing the script. For example:

```
/opt/halo/haloEvents.py --starting=2012-12-12 -kv
```

Note: It is imperative that you select **Type the script to execute** and enter the information there.

11. Click **Save**.

When the Web Application has finished adding the new data source, it returns you to the **Collectors** tab, where the newly added Script source is listed.

The screenshot shows the Sumo Logic interface for managing collectors and sources. A table lists the following source:

Name	Status	Source Category	Sources	Last Hour	Messages	Actions
Halo Events server	✓		1	[Line graph]	357	Add... Edit Delete
Halo events Script	✓	halo_events				Edit Delete

You're done! The Halo Event Connector is now automatically providing events to Sumo Logic for indexing and searching.

D. View Halo Events in Sumo Logic

It may take up to 10 minutes or more to see your first results, but once the Connector has run successfully and has incorporated the data into Sumo Logic, you will see Halo events such as the following appear in your Sumo Logic searches:

The screenshot shows a search for Halo events. The search query is `_sourceCategory=halo_events`. The results show two messages:

#	Time	Message
1	03/08/2013 11:05:29.425	name="Halo logout" created_at="2013-03-08T08:05:29.425840Z" critical="False" actor_ip_address="..." actor_username="vendorIntegrations" message="Halo user vendorIntegrations logged out of the Halo Portal from IP address ... (USA)." actor_country="USA" Host: sumo-test Name: Halo events Category: halo_events
2	03/08/2013 10:49:13.669	name="Halo login success" created_at="2013-03-08T07:49:13.669495Z" critical="False" actor_ip_address="..." actor_username="vendorIntegrations" message="Halo user vendorIntegrations logged into the Halo Portal from IP address ... (USA)." actor_country="USA" Host: sumo-test Name: Halo events Category: halo_events