

Halo Event Connector

Integration with Splunk Enterprise

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 Splunk	6
D. View Halo Events in Splunk.....	9

This document describes the CloudPassage Halo Event Connector and explains how you can configure the connector to import Halo event data into Splunk Enterprise.

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](#).
- Splunk Enterprise Server 4.x or later or 5.x or later. You can download Splunk Enterprise Server from [here](#).
- 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 Splunk Enterprise or Sumo Logic—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:

```
$ python 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:

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

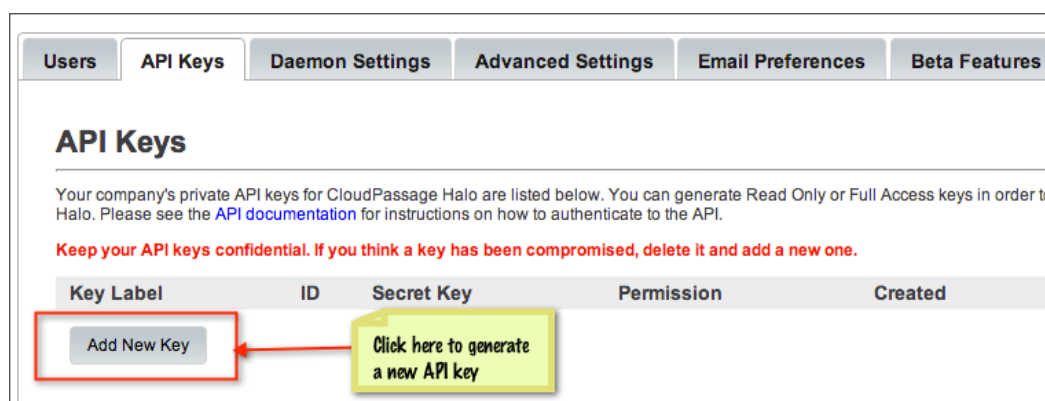
Output to a file. Whenever it writes event data to a disk file, the Connector appends the new data to the end of any existing data in the file.

Platform support. The Event Connector runs on both Linux and Windows operating systems, and it integrates with Splunk for both Linux and Windows.

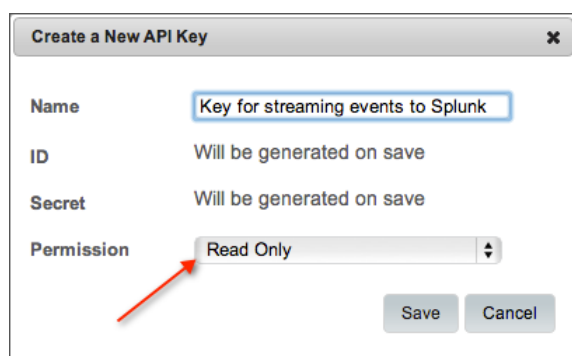
A. Retrieve and Save your CloudPassage API Key

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.

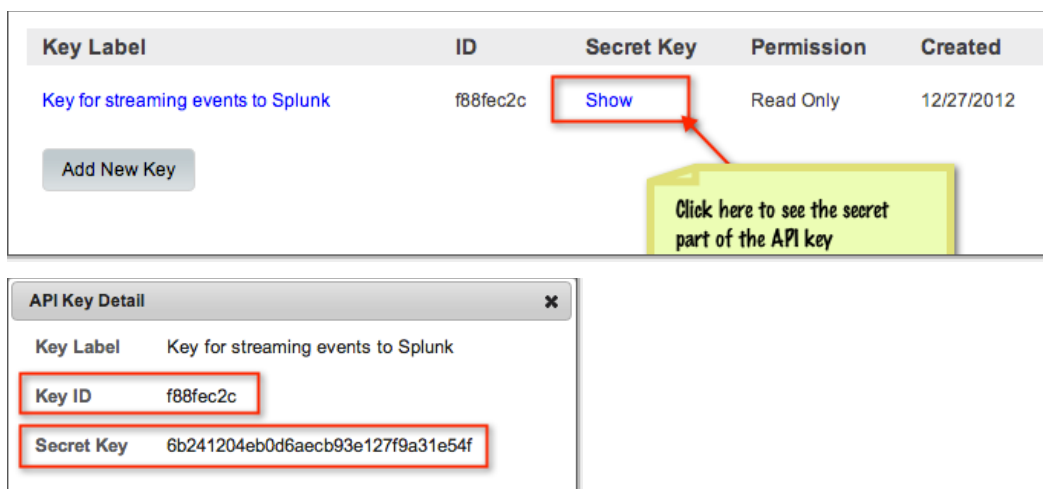
To retrieve your CloudPassage API key, log into the [CloudPassage Portal](#) and navigate to **Settings > Site Administration** and click the **API Keys** tab. (If you haven't generated an API key yet, do so by clicking **Add New Key**.)



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.



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 Splunk	f88fec2c	Show	Read Only	12/27/2012

Add New Key

Click here to see the secret part of the API key

API Key Detail

Key Label: Key for streaming events to Splunk

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 Splunk.

- 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:

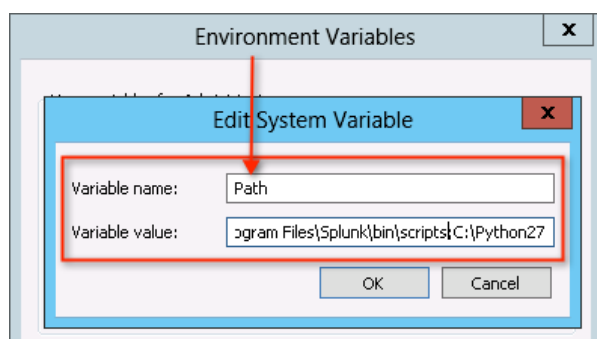
On Linux:

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

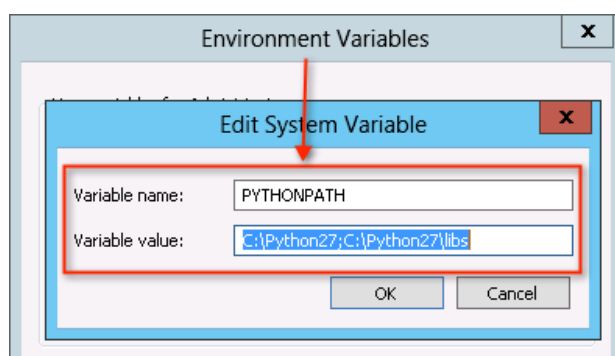
```
[root@ip-10-253-21-19 ~]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/home/ec2/bin:/usr/local/sbin:/usr/local/bin:/sbin:
[root@ip-10-253-21-19 ~]# which python
/usr/local/bin/python
```

On Windows:

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



- 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:

```
$ python 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 Splunk

The main purpose of the Halo Event Connector when integrated into a tool such as Splunk Enterprise is to feed event data to that tool. In the case of Splunk Enterprise, once it receives the data, Splunk transforms it into a series of indexed Splunk events, each consisting of searchable fields. There are many things you can do in Splunk to massage the data both before and after Splunk indexes it, but you don't usually need to. In most cases, Splunk can determine what type of data you are feeding it and then handle it appropriately.

5. Splunk expects the scripts it runs to collect output to be in a specific directory. Place all of these files:
 - `haloEvents.py`
 - `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 following location in your Splunk installation:

`$SPLUNK_HOME/bin/scripts/`

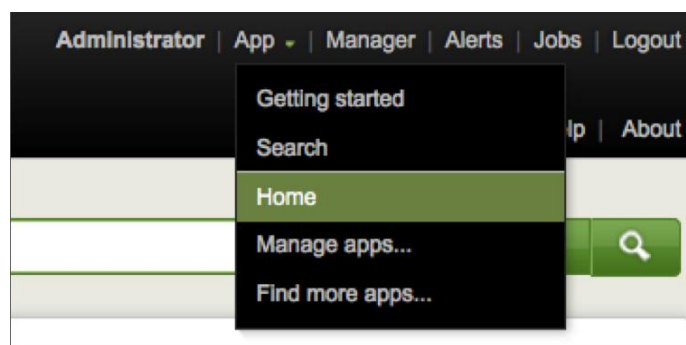
6. Make sure your `PATH` environment variable (and `PYTHONPATH` on Windows) is appropriately set, as shown in [Section B](#).
7. On either Linux or Windows, the Connector will emit the default output format (JSON) for Splunk. You need to specify how Splunk should interpret the JSON and extract the timestamp for each event. To do that, add the following lines to your Splunk `props.conf` file, in the directory `$SPLUNK_HOME/etc/local/default`.

[cp-halo] ← This defines a new source type in Splunk; use any name you wish

```
NO_BINARY_CHECK = 1
SHOULD_LINEMERGE = false
TIME_FORMAT = %Y-%m-%dT%H:%M%S.%6N
TIME_PREFIX = "\"created_at\"":\"s?\"
pulldown_type = 1
KV_MODE = json
```

Note: You will need to restart the Splunk server for it to recognize the newly created source type.

8. Now log into your Splunk installation. If Splunk Home does not appear immediately, choose **Home** from the **App** menu.



9. In Splunk Home, click **Add data**.
The **Add Data to Splunk** dialog box opens.

Add Data to Splunk

Choose a Data Type

A file or directory of files	Unix/Linux logs and metrics	IIS logs
Syslog	File integrity monitoring	Apache logs
Windows event logs	Configuration files	WebSphere logs, metrics and other data
Windows Registry	OPSEC LEA	Any other data...
Windows performance metrics	Cisco device logs	

Or Choose a Data Source

From files and directories From a TCP port From a UDP port	Run and collect the output of a script
--	--

You add new types of data to Splunk by telling it about them. There are a number of ways you can specify a data input, either in terms of its type or by its source. The Connector is a source that collects data for Splunk by connecting to an external source using an API.

- Under **Or Choose a Data Source**, click **Run and collect the output of a script**. The **Add new** dialog box opens.

Add new

Source

Command *

`$SPLUNK_HOME\bin\scripts\haloEvents.py`

On Unix: /opt/splunk/bin/scripts/getData.sh foo "bar baz"
On Windows: c:\program files\splunk\bin\scripts\getData.bat "foo bar" baz

Interval *

120

Number of seconds to wait before running the command again, or a valid cron schedule.

Source name override

If set, overrides the default source value for your script entry (script:path_to_script).

Source type

Set sourcetype field for all events from this source.

Set sourcetype

From list

Select source type from list

cp-halo

- Fill in these fields:

- Command** field—Enter the full path to `haloEvents.py`. Also, add any command-line parameters that you want to use when executing the script. For example:

```
$SPLUNK_HOME\bin\scripts\haloEvents.py --starting=2012-12-12 --kv
```

The above command will extract Halo events logged Dec 12, 2012 onwards and emit them to standard output in a key-value pair format.

- **Interval** field—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 Splunk.
- **Set sourcetype** field—Choose **From list**.
- **Select source type from list** field—Select the source type value that you specified in the Splunk `props.conf` file (see Step 7).

12. Click **Save**.

When it has finished adding the new data source, Splunk displays a success message.



You're done! The Halo Event Connector is now automatically providing events to Splunk for indexing.

D. View Halo Events in Splunk

Once the Connector runs successfully and is incorporating event data into Splunk, you will see Halo events such as the following appear in your Splunk searches:

