

---

# SeqLog Documentation

*Release 0.3.7*

**Adam Friedman**

**Jan 06, 2018**



---

# Contents

---

<b>1</b>	<b>SeqLog</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Stable release . . . . .	5
2.2	From sources . . . . .	5
<b>3</b>	<b>Usage</b>	<b>7</b>
3.1	Configure logging programmatically . . . . .	7
3.2	Configure logging from a file . . . . .	7
3.3	Batching and auto-flush . . . . .	9
3.4	Overriding the root logger . . . . .	9
3.5	Additional LogHandlers . . . . .	9
3.6	Global log properties . . . . .	9
<b>4</b>	<b>seqlog</b>	<b>11</b>
4.1	seqlog package . . . . .	11
<b>5</b>	<b>Contributing</b>	<b>15</b>
5.1	Types of Contributions . . . . .	15
5.2	Get Started! . . . . .	16
5.3	Pull Request Guidelines . . . . .	17
5.4	Tips . . . . .	17
<b>6</b>	<b>Credits</b>	<b>19</b>
6.1	Development Lead . . . . .	19
6.2	Contributors . . . . .	19
<b>7</b>	<b>History</b>	<b>21</b>
7.1	0.3.7 (2018-01-05) . . . . .	21
7.2	0.3.4 (2017-11-27) . . . . .	21
7.3	0.3.3 (2016-11-18) . . . . .	21
7.4	0.3.2 (2016-11-18) . . . . .	21
7.5	0.3.1 (2016-11-18) . . . . .	21
7.6	0.3.0 (2016-11-16) . . . . .	22
7.7	0.2.0 (2016-07-09) . . . . .	22
7.8	0.0.1 (2016-07-07) . . . . .	22
7.9	0.0.7 (2016-07-09) . . . . .	22

7.10 0.1.0 (2016-07-09) . . . . .	22
<b>8 Indices and tables</b>	<b>23</b>
<b>Python Module Index</b>	<b>25</b>

SeqLog is a plugin for Python logging that sends log messages to Seq.

It also adds support for logging with named format arguments (via keyword arguments) in the same way `"{arg1}"`. `format(arg1="foo")` does.

Contents:



# CHAPTER 1

---

## SeqLog

---

SeqLog enables logging from Python to Seq.

It also adds support for logging with named format arguments (via keyword arguments) in the same way `"{arg1}"`. `format(arg1="foo")` does.

- Free software: MIT license
- Documentation: <https://seqlog.readthedocs.io>.





## 2.1 Stable release

To install SeqLog, run this command in your terminal:

```
$ pip install seqlog
```

This is the preferred method to install SeqLog, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

## 2.2 From sources

The sources for SeqLog can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/tintoy/seqlog
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/tintoy/seqlog/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```



### 3.1 Configure logging programmatically

```
import seqlog

seqlog.log_to_seq(
    server_url="http://my-seq-server:5431/",
    api_key="My API Key",
    level=logging.INFO,
    batch_size=10,
    auto_flush_timeout=10000, # milliseconds
    override_root_logger=True
)
```

For the best experience, use {x}-style named format arguments (passing those format arguments as keyword arguments to the log functions `info`, `warning`, `error`, `critical`, etc). Using unnamed “holes” (i.e. `{}`) is not currently supported.

For example:

```
logging.info("Hello, {name}!", name="World")
```

If you specify ordinal arguments, the log message is interpreted as a “%s”-style format string. The ordinal format arguments are stored in the log entry properties using the 0-based ordinal index as the property name.

```
logging.info("Hello, %s!", "World")
```

Note that mixing named and ordinal arguments is not currently supported.

### 3.2 Configure logging from a file

Seqlog can also use a YAML-format file to describe the desired logging configuration.

First, create your configuration file (e.g. `/foo/bar/my_config.yml`):

```
# This is the Python logging schema version (currently, only the value 1 is supported,
↪here).
version: 1

# Configure logging from scratch.
disable_existing_loggers: True

# Configure the root logger to use Seq
root:
  class: seqlog.structured_logging.StructuredRootLogger
  level: INFO
  handlers:
  - seq
  - console

# You can also configure non-root loggers.
loggers:
  another_logger:
    class: seqlog.structured_logging.StructuredLogger
    propagate: False
    level: INFO
    handlers:
    - seq
    - console

handlers:
# Log to STDOUT
console:
  class: seqlog.structured_logging.ConsoleStructuredLogHandler
  formatter: seq

# Log to Seq
seq:
  class: seqlog.structured_logging.SeqLogHandler
  formatter: seq

  # Seq-specific settings (add any others you need, they're just kwargs for
  ↪SeqLogHandler's constructor).
  server_url: 'http://localhost:5341'
  api_key: 'your_api_key_if_you_have_one'

formatters:
  seq:
    style: '{'
```

Then, call `seqlog.configure_from_file()`:

```
seqlog.configure_from_file('/foo/bar/my_config.yml')

# Use the root logger.
root_logger = logging.getLogger()
root_logger.info('This is the root logger.')

# Use another logger
another_logger = logging.getLogger('another_logger')
another_logger.info('This is another logger.')
```

### 3.3 Batching and auto-flush

By default SeqLog will wait until it has a batch of 10 messages before sending them to Seq. You can control the batch size by passing a value for `batch_size`.

If you also want it to publish the current batch of events when not enough of them have arrived within a certain period, you can pass `auto_flush_timeout` (a float representing the number of seconds before an incomplete batch is published).

### 3.4 Overriding the root logger

By default, SeqLog does not modify the root logger (and so calls to `logging.info()` and friends do not support named format arguments). To also override the root logger, pass `True` for `override_root_logger`.

### 3.5 Additional LogHandlers

By default, `log_to_seq` only configures a single `SeqLogHandler`.

To configure additional `LogHandlers`, pass them via `additional_handlers`.

### 3.6 Global log properties

SeqLog can also add static properties to each log entry that is sent to Seq. By default, the following properties are added:

- `MachineName` The local machine's fully-qualified host name.
- `ProcessId` The current process Id.

To configure global log properties, call `set_global_log_properties`, passing the properties as keyword arguments:

```
import seqlog

seqlog.set_global_log_properties(
    GlobalProperty1="foo",
    GlobalProperty2="bar"
    GlobalProperty3=26
)
```

Note that you can also clear the global log properties (so no properties are added) by calling `clear_global_log_properties`, and reset the global log properties to their defaults by calling `reset_global_log_properties`.



## 4.1 seqlog package

### 4.1.1 Module contents

`seqlog.clear_global_log_properties()`

Remove all global properties.

`seqlog.configure_from_dict(config, override_root_logger=True)`

Configure Seq logging using a dictionary.

Uses `logging.config.dictConfig()`.

#### Parameters

- **config** (*dict*) – A dict containing the configuration.
- **override\_root\_logger** (*bool*) – Override the root logger to use a Seq-specific implementation? (default: True)

`seqlog.configure_from_file(file_name, override_root_logger=True)`

Configure Seq logging using YAML-format configuration file.

Uses `logging.config.dictConfig()`.

#### Parameters

- **file\_name** (*str*) – The name of the configuration file to use.
- **override\_root\_logger** (*bool*) – Override the root logger to use a Seq-specific implementation? (default: True)

`seqlog.get_global_log_properties()`

Get the properties to be added to all structured log entries.

**Returns** A copy of the global log properties.

**Return type** dict

`seqlog.log_to_console` (*level=30, override\_root\_logger=False, \*\*kwargs*)

Configure the logging system to send log entries to the console.

Note that the root logger will not log to Seq by default.

**Parameters**

- **level** – The minimum level at which to log.
- **override\_root\_logger** – Override the root logger, too? Note - this might cause problems if third-party components try to be clever when using the logging.XXX functions.

`seqlog.log_to_seq` (*server\_url, api\_key=None, level=30, batch\_size=10, auto\_flush\_timeout=None, additional\_handlers=None, override\_root\_logger=False, \*\*kwargs*)

Configure the logging system to send log entries to Seq.

Note that the root logger will not log to Seq by default.

**Parameters**

- **server\_url** – The Seq server URL.
- **api\_key** – The Seq API key (optional).
- **level** – The minimum level at which to log.
- **batch\_size** – The number of log entries to collect before publishing to Seq.
- **auto\_flush\_timeout** – If specified, the time (in seconds) before the current batch is automatically flushed.
- **additional\_handlers** – Additional `LogHandler`'s (if any).
- **override\_root\_logger** – Override the root logger, too? Note - this might cause problems if third-party components try to be clever when using the logging.XXX functions.

**Returns** The `SeqLogHandler` that sends events to Seq. Can be used to forcibly flush records to Seq.

**Return type** `SeqLogHandler`

`seqlog.reset_global_log_properties` ()

Initialize global log properties to their default values.

`seqlog.set_global_log_properties` (*\*\*properties*)

Configure the properties to be added to all structured log entries.

**Parameters** **properties** (*dict*) – Keyword arguments representing the properties.

## 4.1.2 Submodules

### 4.1.3 seqlog.structured\_logging module

**class** `seqlog.structured_logging.ConsoleStructuredLogHandler`

Bases: `logging.Handler`

**emit** (*record*)

**class** `seqlog.structured_logging.SeqLogHandler` (*server\_url, api\_key=None, batch\_size=10, auto\_flush\_timeout=None*)

Bases: `logging.Handler`

Log handler that posts to Seq.



**close()**

Close the log handler.

**emit(record)**

Emit a log record.

**Parameters record** – The LogRecord.

**flush()**

**publish\_log\_batch(batch)**

Publish a batch of log records.

**Parameters batch** – A list representing the batch.

**class** seqlog.structured\_logging.**StructuredLogRecord**(*name, level, pathname, lineno, msg, args, exc\_info, func=None, sinfo=None, log\_props=None, \*\*kwargs*)

Bases: logging.LogRecord

An extended LogRecord that with custom properties to be logged to Seq.

**getMessage()**

Get a formatted message representing the log record (with arguments replaced by values as appropriate).  
:return: The formatted message.

**class** seqlog.structured\_logging.**StructuredLogger**(*name, level=0*)

Bases: logging.Logger

Custom (dummy) logger that understands named log arguments.

**makeRecord**(*name, level, fn, lno, msg, args, exc\_info, func=None, extra=None, sinfo=None*)

Create a LogRecord.

#### Parameters

- **name** – The name of the logger that produced the log record.
- **level** – The logging level (severity) associated with the logging record.
- **fn** – The name of the file (if known) where the log entry was created.
- **lno** – The line number (if known) in the file where the log entry was created.
- **msg** – The log message (or message template).
- **args** – Ordinal message format arguments (if any).
- **exc\_info** – Exception information to be included in the log entry.
- **func** – The function (if known) where the log entry was created.
- **extra** – Extra information (if any) to add to the log record.
- **sinfo** – Stack trace information (if known) for the log entry.

**class** seqlog.structured\_logging.**StructuredRootLogger**(*level=0*)

Bases: logging.RootLogger

Custom root logger that understands named log arguments.

**makeRecord**(*name, level, fn, lno, msg, args, exc\_info, func=None, extra=None, sinfo=None*)

Create a LogRecord.

#### Parameters

- **name** – The name of the logger that produced the log record.

- **level** – The logging level (severity) associated with the logging record.
- **fn** – The name of the file (if known) where the log entry was created.
- **lno** – The line number (if known) in the file where the log entry was created.
- **msg** – The log message (or message template).
- **args** – Ordinal message format arguments (if any).
- **exc\_info** – Exception information to be included in the log entry.
- **func** – The function (if known) where the log entry was created.
- **extra** – Extra information (if any) to add to the log record.
- **sinfo** – Stack trace information (if known) for the log entry.

`seqlog.structured_logging.clear_global_log_properties()`  
Remove all global properties.

`seqlog.structured_logging.get_global_log_properties(logger_name=None)`  
Get the properties to be added to all structured log entries.

**Parameters** `logger_name` (*str*) – An optional logger name to be added to the log entry.

**Returns** A copy of the global log properties.

**Return type** dict

`seqlog.structured_logging.reset_global_log_properties()`  
Initialize global log properties to their default values.

`seqlog.structured_logging.set_global_log_properties(**properties)`  
Configure the properties to be added to all structured log entries.

**Parameters** `properties` (*dict*) – Keyword arguments representing the properties.

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 5.1 Types of Contributions

### 5.1.1 Report Bugs

Report bugs at <https://github.com/tintoy/seqlog/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 5.1.4 Write Documentation

SeqLog could always use more documentation, whether as part of the official SeqLog docs, in docstrings, or even on the web in blog posts, articles, and such.

## 5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/tintoy/seqlog/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 5.2 Get Started!

Ready to contribute? Here's how to set up *seqlog* for local development.

1. Fork the *seqlog* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/seqlog.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv seqlog
$ cd seqlog/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 --ignore E501 seqlog tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check [https://travis-ci.org/tintoy/seqlog/pull\\_requests](https://travis-ci.org/tintoy/seqlog/pull_requests) and make sure that the tests pass for all supported Python versions.

## 5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_seqlog
```



### 6.1 Development Lead

- Adam Friedman <tintoy@tintoy.io>

### 6.2 Contributors

None yet. Why not be the first?





### **7.1 0.3.7 (2018-01-05)**

- Implement and document logging configuration from file (#3)

### **7.2 0.3.4 (2017-11-27)**

- Fix sample code (#2).

### **7.3 0.3.3 (2016-11-18)**

- Use streaming mode when posting to Seq (#1)

### **7.4 0.3.2 (2016-11-18)**

- Updated release notes

### **7.5 0.3.1 (2016-11-18)**

- Further work relating to intermittent “`RuntimeError: The content for this response was already consumed`” when publishing log entries (#1)

## 7.6 0.3.0 (2016-11-16)

- Fix for intermittent “RuntimeError: The content for this response was already consumed” when publishing log entries (#1)

## 7.7 0.2.0 (2016-07-09)

- Support for configuring additional log handlers when calling `log_to_seq`.
- Support for global log properties (statically-configured properties that are added to all outgoing log entries).

## 7.8 0.0.1 (2016-07-07)

- First release on PyPI.

## 7.9 0.0.7 (2016-07-09)

- `log_to_seq` now returns the `SeqLogHandler` to enable forced flushing of log records to Seq.
- Change `auto_flush_timeout` to a float representing seconds (instead of milliseconds).
- Update `testharness.py` to actually log to Seq. You can override the server URL and API key using the `SEQ_SERVER_URL` and `SEQ_API_KEY` environment variables.
- Update usage information in documentation.
- Python 3 only for now (sorry, but logging in Python 2 doesn’t have all the required extensibility points). If the need to support Python 2 becomes great enough then I’ll try to find a way.

## 7.10 0.1.0 (2016-07-09)

- Proper versioning starts today :)

## CHAPTER 8

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**S**

`seqlog`, [11](#)

`seqlog.structured_logging`, [12](#)



**C**

clear\_global\_log\_properties() (in module seqlog), 11  
 clear\_global\_log\_properties() (in module seqlog.structured\_logging), 14  
 close() (seqlog.structured\_logging.SeqLogHandler method), 12  
 configure\_from\_dict() (in module seqlog), 11  
 configure\_from\_file() (in module seqlog), 11  
 ConsoleStructuredLogHandler (class in seqlog.structured\_logging), 12

**E**

emit() (seqlog.structured\_logging.ConsoleStructuredLogHandler method), 12  
 emit() (seqlog.structured\_logging.SeqLogHandler method), 13

**F**

flush() (seqlog.structured\_logging.SeqLogHandler method), 13

**G**

get\_global\_log\_properties() (in module seqlog), 11  
 get\_global\_log\_properties() (in module seqlog.structured\_logging), 14  
 getMessage() (seqlog.structured\_logging.StructuredLogRecord method), 13

**L**

log\_to\_console() (in module seqlog), 11  
 log\_to\_seq() (in module seqlog), 12

**M**

makeRecord() (seqlog.structured\_logging.StructuredLogger method), 13  
 makeRecord() (seqlog.structured\_logging.StructuredRootLogger method), 13

**P**

publish\_log\_batch() (seqlog.structured\_logging.SeqLogHandler method), 13

**R**

reset\_global\_log\_properties() (in module seqlog), 12  
 reset\_global\_log\_properties() (in module seqlog.structured\_logging), 14

**S**

seqlog (module), 11  
 seqlog.structured\_logging (module), 12  
 SeqLogHandler (class in seqlog.structured\_logging), 12  
 set\_global\_log\_properties() (in module seqlog), 12  
 set\_global\_log\_properties() (in module seqlog.structured\_logging), 14  
 StructuredLogger (class in seqlog.structured\_logging), 13  
 StructuredLogRecord (class in seqlog.structured\_logging), 13  
 StructuredRootLogger (class in seqlog.structured\_logging), 13