

---

# Logsna Documentation

*Release 1.0*

**Ruslan Spivak**

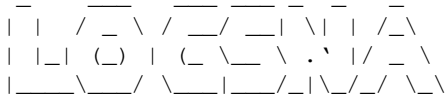
October 02, 2012



# CONTENTS

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>How to use it</b>	<b>5</b>
<b>3</b>	<b>The Log Format</b>	<b>7</b>
<b>4</b>	<b>The Log Format Goals</b>	<b>9</b>
<b>5</b>	<b>The Log Format Notes</b>	<b>11</b>
<b>6</b>	<b>Enhancing The Log Format Output</b>	<b>13</b>
<b>7</b>	<b>Acknowledgments</b>	<b>15</b>
<b>8</b>	<b>License</b>	<b>17</b>
<b>9</b>	<b>Indices and tables</b>	<b>19</b>





**logsna** is a small Python library that provides a sane log output format.

<http://logsna.readthedocs.org>



# INSTALLATION

```
$ [sudo] pip install logsna
```

Or the bleeding edge version from the git master branch:

```
$ [sudo] pip install git+https://github.com/rspivak/logsna.git#egg=logsna
```





# HOW TO USE IT

**logsna** provides a custom formatter class *logsna.Formatter* that can be used in a logging config file:

```
# sanefmt.py
import logging
import logging.config
from StringIO import StringIO

CONFIG = """\
[loggers]
keys=root

[handlers]
keys=console

[handler_console]
class=logging.StreamHandler
args=(sys.stderr,)
formatter=sane

[formatters]
keys=sane

[logger_root]
level=DEBUG
handlers=console

# Our custom formatter class
[formatter_sane]
class=logsna.Formatter
"""

config = StringIO(CONFIG)
logging.config.fileConfig(config)

log = logging.getLogger('mylogger.component1')

log.debug('debug message')
log.info('info message')
log.warning('warning message')
log.critical('critical message')
try:
    1 / 0
except:
    log.exception('Houston we have a problem')
```

This is how to use it in code directly:

```
import logging

import logsna

# create logger
log = logging.getLogger('mylogger.component1')
log.setLevel(logging.DEBUG)

# create console handler and set level to debug
ch = logging.StreamHandler()
ch.setLevel(logging.DEBUG)

# create an instance of the sane formatter
formatter = logsna.Formatter()

# add our formatter to the console handler
ch.setFormatter(formatter)

# add the console handler to the logger
log.addHandler(ch)

log.debug('debug message')
log.info('info message')
log.warning('warning message')
log.critical('critical message')
try:
    1 / 0
except:
    log.exception('Houston we have a problem')
```

# THE LOG FORMAT

Here is an output from the above program:

```
DEBUG      [2012-05-21 01:59:23,686] mylogger.component1: debug message
INFO       [2012-05-21 01:59:23,686] mylogger.component1: info message
WARNING    [2012-05-21 01:59:23,686] mylogger.component1: warning message
CRITICAL   [2012-05-21 01:59:23,686] mylogger.component1: critical message
ERROR      [2012-05-21 01:59:23,686] mylogger.component1: Houston we have a problem
! Traceback (most recent call last):
!   File "/home/alienoid/python/sanefmt.py", line 67, in <module>
!     1 / 0
! ZeroDivisionError: integer division or modulo by zero
```



# THE LOG FORMAT GOALS

1. To be human readable as much as possible
2. Make it easy to use with standard Unix utilities **tail** and **grep** to help quickly figure out why things are going south



# THE LOG FORMAT NOTES

1. All timestamps are in **ISO8601** and **UTC** format

2. To grep for messages of a specific level

```
$ tail -f sanefmt.log | grep '^INFO'
```

3. To grep for messages from a particular logger

```
$ tail -f sanefmt.log | grep 'component1:'
```

4. To pull out full exception tracebacks with a corresponding log message

```
$ tail -f sanefmt.log | grep -B 1 '^!'
```

The output of the above command will look like this

```
ERROR      [2012-05-21 01:59:23,686] mylogger.component1: Houston we have a problem
! Traceback (most recent call last):
!   File "fmttest.py", line 72, in <module>
!     1 / 0
! ZeroDivisionError: integer division or modulo by zero
```





# ENHANCING THE LOG FORMAT OUTPUT

Here is the format string used by *Logdna* formatter:

```
'%(levelname)-8s [% (asctime)s] %(name)s: %(message)s'
```

You can explicitly specify it in your configuration file using **format** directive

```
# Our custom formatter class
[formatter_sane]
format=%(levelname)-8s [% (asctime)s] %(name)s: %(message)s
class=logdna.Formatter
```

And you can also enhance the format string by adding your custom attributes to it if you need to. For a set of predefined log record attributes see [here](#)



# ACKNOWLEDGMENTS

- [Release It!](#)
- [Logula](#)



# LICENSE

Copyright (c) 2012 Ruslan Spivak

Published under The MIT License, see LICENSE



# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*