snippy Documentation

Release 0.10.0

Heikki J. Laaksonen

May 01, 2019

Contents:

CHAPTER 1

Introduction

Manage command snippets and solution examples from command line or through a RESTish API that follows subset of JSON API specification v1.0.

The Snippy tool is intended to support software development and troubleshooting workflows by collecting command examples and troubleshooting solutions into one manager. The tool may be helpful for example when working with different open source components that all have different configuration settings and troubleshooting methods. You can share the best examples through REST API server or by exporting and importing the data with common data serialization languages YAML or JSON.

CHAPTER 2

Installation

To install, run:

```
pip install snippy --user
```

To remove, run:

```
pip uninstall --yes snippy
```

To install from Docker Hub, run:

```
docker pull heilaaks/snippy
```

To install from Github, run:

```
git clone https://github.com/heilaaks/snippy.git
cd snippy
make install
```

To try for the very first time, run:

CHAPTER 3

Features

3.1 Terminal

Content managed by Snippy is divided into two categories called snippets and solutions. Snippets are short command examples and solutions are longer solution descriptions. You can add metadata like links and tags to help to search content.

You can operate snippet or solution content with six basic operations: create, search, update, delete, import and export. These operations manage the content in persistent file storage installed into the same place as the tool.

3.2 Server

You can run the Snippy as a server. The server can create, search, update and delete snippets and solutions. The server operates through RESTish API that follows a subset of JSON API v1.0 specification.

The server does not bind to any address by default and the server-host option must be always defined. The server-host option supports format <ip>:<port>. You can also define the log format between string and JSON and verbosity level of logs.

The API is experimental and changes can be expected. The API is documented in Swagger Hub OpenAPI definitions.

The JSON REST API server is available only when the tool is installed from Docker Hub or directly from the source code.

```
# Start server by sharing host network and enable JSON logs with limited
# message length. Always remove previosly started container before running
# container with new options set.
docker rm -f snippy
sudo docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.0.
↪1:8080 --log-json -vv
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?limit=2" -H "accept:␣
↪application/vnd.api+json" | python -m json.tool
curl -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&limit=2" -H
↪"accept: application/vnd.api+json" | python -m json.tool
```

(continues on next page)

(continued from previous page)

```
# Start the server and define the port and IP address when the network is
# shared between the container and host. Generate full length logs with
# the --debug option.
docker rm -f snippy
sudo docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.0.
↪1:8080 --log-json --debug
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&limit=2" -
↪H "accept: application/vnd.api+json" | python -m json.tool

# Run the server with string logs.
docker rm -f snippy
sudo docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.0.
↪1:8080 -vv
```

You can query the server logs with the Docker log command.

```
docker logs snippy
```

You can remove the server with the rm command.

```
docker rm -f snippy
```

Note that Docker container is immutable and it does not share volume from the host. If you want to run a server that allows content modification, you must install the server from code repository.

```
git clone https://github.com/heilaaks/snippy.git
cd snippy
make server
```

With a local server, you can change to location of the storage from the default. If the default content is needed, you need to import it into the new location before starting the server.

```
snippy import --defaults --storage-path ${HOME}/devel/temp
snippy import --defaults --solution --storage-path ${HOME}/devel/temp
snippy --server-host 127.0.0.1:8080 --storage-path ${HOME}/devel/temp -vv
```

CHAPTER 4

Usage

4.1 Terminal

Snippy command line commands always must include content operation and category. Content operation is one of the six basic operations and the category can be snippet or solution. The content category is snippet by default. Metadata attached to the content allows adding brief description of the content, single group to which the content belongs, list of tags to assist search operations and a list of links for more information about the content.

Snippy tool outputs always OK after successful operation and NOK with a failure string in case of failure. You can use debug option with the command to investigate possible problems. For more detailed troubleshooting instructions, please refer to the development section.

The workflow chapter contains use cases and examples.

Note: The tool is used by the author in Linux environment. There is an edit functionality with editor that always assumes vi editor. This limitation can be circumvented by using text based templates to import content or command line options in case of snippets.

Note: The default content is provided "as is" basis without warranties of any kind.

4.2 Server

Snippy can be run as a server. In this case you must access the content through a REST API with same principles as from command line.

CHAPTER 5

Workflows

5.1 Getting help

Use help option with keyword examples to read about basic usage. Read the documentation from Read the Docs or dive into the code in Github.

```
snippy --help
snippy --help examples
```

5.2 Creating content

5.2.1 Creating snippet with vi editor

Following command uses vi editor to create new content. The command opens an input template where you can define the mandatory snippet and optional brief description, group, tags and links related to the snippet.

```
snippy create --snippets --editor
```

5.2.2 Creating snippet from text template

Following commands allows using a text template to import new snippet.

```
snippy export --snippets --template
snippy import --snippets -f snippet-template.txt
```

5.2.3 Creating snippet from command line

You can add new snippet directly from command line. How ever, easiest way to create new content is to use editor.

```
snippy create --content 'docker rm $(docker ps -a -q)' --brief 'Remove all docker␣
↪containers' --tags docker,image,cleanup
snippy create --content 'docker rm --volumes $(docker ps --all --quiet)' --brief
↪'Remove all docker containers with volumes' --group docker --tags docker-ce,docker,
↪moby,container,cleanup --links 'https://docs.docker.com/engine/reference/
↪commandline/rm/'
```

5.2.4 Creating solution from text template

```
snippy export --solutions --template
snippy import --snippets -f solution-template.txt
```

5.3 Searching content

5.3.1 Printing all examples on terminal

It is possible to list all snippets on screen by using dot as a search keyword.

```
snippy search --sall .
OK
```

5.3.2 Filtering with grep

With Linux grep it is possible to filter for example only the actual commands from the search query.

```
snippy search --sall . --no-ansi | grep '\$'
snippy search --sgrp docker --no-ansi | grep '\$'
```

Filtering out solution content to list only the metadata.

```
snippy search --solution --sall . | grep -Ev '[^\s]+:'
```

5.4 Updating content

5.4.1 Updating snippet with vi editor

Following command allows updating existing snippet with vi editor. The command will launch a vi editor which allows you to modify the content. The content is updated automatically after the file is saved and editor is exit.

```
snippy update --digest 54e41e9b52a02b63
```

5.4.2 Updating solution from text template

Following commands allow updating existing solution by exporting the content to text file and importing it again.

```
snippy export --digest 76a1a02951f6bcb4
snippy import --digest 76a1a02951f6bcb4 --file howto-debug-elastic-beats.txt
```

5.4.3 Updating duplicated content with message digest

There is an unique constraint defined for the content. This means that two examples with the same content cannot be stored. There are two supported work flows.

The tool will prompt failure log with a message digest for content that is already existing. User can change the create operation to update and define the message digest. This will launch a vi editor that contain the values that were previously stored. User may change the values in editor and save the content which will get then updated.

```
snippy create --content 'docker rm $(docker ps -a -q)' --brief 'Remove all docker␣
↪containers' --tags docker,image,cleanup
NOK: content already exist with digest f6062e09e2c11b47
snippy update --digest f6062e09e2c11b47
OK
```

5.4.4 Updating duplicated content by defining content

The tool will prompt failure log with a message digest for content that is already existing. User can change the create operation to uddate and use the same command. This will launch a vi editor with the content defined in command line. If some of the values are not defined in command line, they are shown as previously stored. User may change the values in editor and save the content which will get then updated.

```
snippy create --content 'docker rm $(docker ps -a -q)' --brief 'Remove all docker␣
↪containers' --tags docker,image,cleanup
NOK: content already exist with digest 5feded9ec5945d6a
snippy update --content 'docker rm $(docker ps -a -q)' --brief 'Remove all docker␣
↪containers' --tags docker,image,cleanup
OK
```

5.5 Deleting content

Delete snippet with index.

```
snippy delete --digest 96471dce19fe9c90
```

5.6 Migrating content

5.6.1 Exporting content

Following commands allow exporting all snippets and solutions to YAML file that you use to back-up your data. The commands below will create snippets.yaml and solutions.yaml files into same directory where the command was executed. You can define the file name and path with the -f|–file option.

```
snippy export --snippets
snippy export --snippets -f my-snippets.yaml
snippy export --solutions
snippy export --solutions -f my-solutions.yaml
```

5.6.2 Importing content

Following commands allow importing snippets and solutions from default YAML files named snippets.yaml and solutions.yaml that must be located in the same directory where the command is executed. You can define the file name and path with the -f|–file option.

```
snippy import --snippets
snippy import --solutions
```

CHAPTER 6

Api

POST /{category}
Add new resource in defined category.

Parameters

- **category** (*string*) – # Brief

  Resource category.

Status Codes

- 201 Created – Resource created
- 400 Bad Request – Request not syntactically correct
- 403 Forbidden – Using client generated ID is not supported
- 500 Internal Server Error – Internal server error

GET /{category}
Search resource

Parameters

- **category** (*string*) – # Brief

  Resource category.

Query Parameters

- **sall** (*array*) – # Brief

  Search given keywords.

- **stag** (*array*) – # Brief

  Limit search to resources that have defined tags.

- **sgrp** (*array*) – # Brief

  Limit search to resources that are in defined groups.

- **`digest`** (*`array`*) – # Brief

  Limit search to resources that are have given digest.
- **`uuid`** (*`array`*) – # Brief

  Limit search to resources that are have given uuid.
- **`filter`** (*`string`*) – # Brief

  Filter search result with given regexp.
- **`limit`** (*`string`*) – # Brief

  Limit resources in search result.
- **`fields`** (*`array`*) – # Brief

  Limit fields that are returned in result list.
- **`sort`** (*`string`*) – # Brief

  Sort based on defined field.
- **`offset`** (*`string`*) – # Brief

  Offset from the beginning of the search results.

Status Codes

- 200 OK – OK
- 400 Bad Request – Request not syntactically correct
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

`POST /{category}/{id}`

Override POST with PUT, PATCH or DELETE

Parameters

- **`category`** (*`string`*) – # Brief

  Resource category.
- **`id`** (*`string`*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

Status Codes

- 200 OK – OK
- 204 No Content – Resource deleted
- 400 Bad Request – Request not syntactically correct
- 403 Forbidden – Using client generated ID is not supported
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

Request Headers

- *X-HTTP-Method-Override* – # Brief

  Override POST method with PUT, PATCH or DELETE.

`PUT /{category}/{id}`

Update defined resource based on given identity.

Parameters

- **`category`** (*`string`*) – # Brief

  Resource category.

- **`id`** (*`string`*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

Status Codes

- 200 OK – OK
- 400 Bad Request – Request not syntactically correct
- 403 Forbidden – Using client generated ID is not supported
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

`PATCH /{category}/{id}`

Update defined resource based on given identity.

Parameters

- **`category`** (*`string`*) – # Brief

  Resource category.

- **`id`** (*`string`*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

Status Codes

- 200 OK – OK
- 400 Bad Request – Request not syntactically correct
- 403 Forbidden – Using client generated ID is not supported
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

`GET /{category}/{id}`

Search defined resource based on given identity.

Parameters

- **`category`** (*`string`*) – # Brief

  Resource category.

- **`id`** (*string*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

Status Codes

- 200 OK – OK
- 400 Bad Request – Request not syntactically correct
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

`DELETE /{category}/{id}`

Delete defined resource based on given identity.

Parameters

- **`category`** (*string*) – # Brief

  Resource category.

- **`id`** (*string*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

Status Codes

- 204 No Content – Resource deleted
- 400 Bad Request – Request not syntactically correct
- 404 Not Found – Resource not found
- 409 Conflict – More than one resource found
- 500 Internal Server Error – Internal server error

`GET /{category}{id}{field}`

Get resource attribute.

Parameters

- **`category`** (*string*) – # Brief

  Resource category.

- **`id`** (*string*) – # Brief

  Resource identity that can be either digest or UUID. Partial identities are allowed but they may result multiple matching resources which is an error. The `id` path parameter must uniquely identify the requested resource.

- **`field`** (*string*) – # Brief

  Resource field.

Query Parameters

- **`sall`** (*array*) – # Brief

  Search given keywords.

- **`stag`** (*array*) – # Brief

  Limit search to resources that have defined tags.
- **`sgrp`** (*array*) – # Brief

  Limit search to resources that are in defined groups.
- **`digest`** (*array*) – # Brief

  Limit search to resources that are have given digest.
- **`uuid`** (*array*) – # Brief

  Limit search to resources that are have given uuid.
- **`filter`** (*string*) – # Brief

  Filter search result with given regexp.
- **`limit`** (*string*) – # Brief

  Limit resources in search result.
- **`fields`** (*array*) – # Brief

  Limit fields that are returned in result list.
- **`sort`** (*string*) – # Brief

  Sort based on defined field.
- **`offset`** (*string*) – # Brief

  Offset from the beginning of the search results.

Status Codes

- 200 OK – OK
- 400 Bad Request – Request not syntactically correct
- 404 Not Found – Resource not found
- 500 Internal Server Error – Internal server error

`GET /`

Server hello

Status Codes

- 200 OK – OK

CHAPTER 7

Development

7.1 Quick Start

For the development, you can clone the repository and run the setup for Python virtual environment like below:

```
git clone https://github.com/heilaaks/snippy.git
mkvirtualenv snippy
make install-devel
```

The basic commands to run and test are:

```
python3 runner create -c 'docker rm $(docker ps -a -q)' -b 'Remove all docker␣
↪containers' -t docker,container,cleanup
make test
make lint
make coverage
make docs
make clean
```

7.2 Python Virtual Environment

You can install the Python virtual environment wrapper like below:

```
mkdir -p ${HOME}/devel/python-virtualenvs
sudo pip3 install virtualenvwrapper
virtualenv --version
export WORKON_HOME=${HOME}/devel/python-virtualenvs # Add to ~/.bashrc
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3    # Add to ~/.bashrc
source /usr/bin/virtualenvwrapper.sh                # Add to ~/.bashrc
mkvirtualenv snippy
```

Example commands to operate the virtual environment are below. More information can be found from the Python virtualenvwrapper command reference documentation.

```
lssitepackages
lsvirtualenv
deactivate
workon snippy
rmvirtualenv snippy
```

7.3 Pylint

The Pylint rc file can be generated for the very first time like:

```
pylint --generate-rcfile > tests/pylint/pylint-snippy.rc
```

7.4 Apache Bench

```
# Install testing tools.
dnf install httpd-tools
go get -u github.com/rakyll/hey

# Generate TLS server certificates
openssl req -x509 -newkey rsa:4096 -nodes -keyout server.key -out server.crt -days␣
↪356 -subj "/C=US/O=Snippy/CN=127.0.0.1"

# Run HTTP server with sqlite backend with commit␣
↪f9f418256fccaf7f4c1ee3651b21044aba9a8948 (v0.10.0 + 20 commits)
docker run -d --net="host" --name snippy heilaaks/snippy:latest --server-host 127.0.0.
↪1:8080 --defaults
ab -n 10000 -c 1 -k http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        gunicorn/19.9.0
Server Hostname:        127.0.0.1
Server Port:            8080
```

(continues on next page)

(continued from previous page)

```
Document Path:          /api/snippy/rest/snippets?limit=20
Document Length:        31914 bytes

Concurrency Level:      1
Time taken for tests:   45.854 seconds
Complete requests:      10000
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      320920000 bytes
HTML transferred:       319140000 bytes
Requests per second:    218.08 [#/sec] (mean)
Time per request:       4.585 [ms] (mean)
Time per request:       4.585 [ms] (mean, across all concurrent requests)
Transfer rate:          6834.73 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     4    5   0.5      4      15
Waiting:        4    5   0.5      4      15
Total:          4    5   0.5      4      15
WARNING: The median and mean for the processing time are not within a normal deviation
        These results are probably not that reliable.
WARNING: The median and mean for the waiting time are not within a normal deviation
        These results are probably not that reliable.
WARNING: The median and mean for the total time are not within a normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      4
  75%      4
  80%      4
  90%      5
  95%      5
  98%      6
  99%      7
 100%     15 (longest request)

# Run HTTP server with sqlite backend with commit␣
↪f9f418256fccaf7f4c1ee3651b21044aba9a8948 (v0.10.0 + 20 commits)
docker run -d --net="host" --name snippy heilaaks/snippy:latest --server-host 127.0.0.
↪1:8080 --defaults
/root/go/bin/hey -n 10000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20

Summary:
  Total:        45.1121 secs
  Slowest:      0.0142 secs
  Fastest:      0.0044 secs
  Average:      0.0045 secs
  Requests/sec: 221.6700

  Total data:   319140000 bytes
  Size/request: 31914 bytes

Response time histogram:
  0.004 [1]     |
```

(continues on next page)

(continued from previous page)

```
  0.005 [9974]  |
  0.006 [6]     |
  0.007 [6]     |
  0.008 [3]     |
  0.009 [3]     |
  0.010 [4]     |
  0.011 [2]     |
  0.012 [0]     |
  0.013 [0]     |
  0.014 [1]     |


Latency distribution:
  10% in 0.0045 secs
  25% in 0.0045 secs
  50% in 0.0045 secs
  75% in 0.0045 secs
  90% in 0.0046 secs
  95% in 0.0046 secs
  99% in 0.0048 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0044 secs, 0.0142 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0044 secs, 0.0043 secs, 0.0140 secs
  resp read:    0.0000 secs, 0.0000 secs, 0.0004 secs

Status code distribution:
  [200] 10000 responses

# Run HTTPS server with sqlite backend with commit␣
↪f9f418256fccaf7f4c1ee3651b21044aba9a8948 (v0.10.0 + 20 commits)
python runner --server-host 127.0.0.1:8080 --server-ssl-cert ./server.crt --server-
↪ssl-key ./server.key --defaults
/root/go/bin/hey -n 10000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?
↪limit=20

Summary:
  Total:        90.7888 secs
  Slowest:      0.0161 secs
  Fastest:      0.0088 secs
  Average:      0.0091 secs
  Requests/sec: 110.1457

  Total data:   319140000 bytes
  Size/request: 31914 bytes

Response time histogram:
  0.009 [1]     |
  0.010 [9856]  |
  0.010 [107]   |
  0.011 [9]     |
  0.012 [5]     |
  0.012 [5]     |
  0.013 [3]     |
  0.014 [1]     |
```

(continues on next page)

(continued from previous page)

```
  0.015 [8]     |
  0.015 [1]     |
  0.016 [4]     |


Latency distribution:
  10% in 0.0090 secs
  25% in 0.0090 secs
  50% in 0.0090 secs
  75% in 0.0091 secs
  90% in 0.0092 secs
  95% in 0.0093 secs
  99% in 0.0097 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0052 secs, 0.0088 secs, 0.0161 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0038 secs, 0.0037 secs, 0.0106 secs
  resp read:    0.0001 secs, 0.0001 secs, 0.0005 secs

Status code distribution:
  [200] 10000 responses


# Run HTTP server with PostgreSQL backend with commit␣
↪f9f418256fccaf7f4c1ee3651b21044aba9a8948 (v0.10.0 + 20 commits)
docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.0.1:8080␣
↪--storage-type postgresql --storage-host localhost:5432 --storage-database postgres␣
↪--storage-user postgres --storage-password postgres --defaults
ab -n 10000 -c 1 -k http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        gunicorn/19.9.0
Server Hostname:        127.0.0.1
Server Port:            8080

Document Path:          /api/snippy/rest/snippets?limit=20
Document Length:        31914 bytes

Concurrency Level:      1
```

(continues on next page)

(continued from previous page)

```
Time taken for tests:   52.412 seconds
Complete requests:      10000
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      320920000 bytes
HTML transferred:       319140000 bytes
Requests per second:    190.80 [#/sec] (mean)
Time per request:       5.241 [ms] (mean)
Time per request:       5.241 [ms] (mean, across all concurrent requests)
Transfer rate:          5979.51 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     5    5   0.4      5      21
Waiting:        5    5   0.4      5      21
Total:          5    5   0.4      5      21

Percentage of the requests served within a certain time (ms)
  50%      5
  66%      5
  75%      5
  80%      5
  90%      5
  95%      5
  98%      6
  99%      7
 100%     21 (longest request)

# Run HTTP server with PostgreSQL backend with commit␣
↪f9f418256fccaf7f4c1ee3651b21044aba9a8948 (v0.10.0 + 20 commits)
docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.0.1:8080␣
↪--storage-type postgresql --storage-host localhost:5432 --storage-database postgres␣
↪--storage-user postgres --storage-password postgres --defaults
/root/go/bin/hey -n 10000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20

Summary:
  Total:        52.7001 secs
  Slowest:      0.0211 secs
  Fastest:      0.0050 secs
  Average:      0.0053 secs
  Requests/sec: 189.7530

  Total data:   319140000 bytes
  Size/request: 31914 bytes

Response time histogram:
  0.005 [1]     |
  0.007 [9968]  |
  0.008 [9]     |
  0.010 [6]     |
  0.011 [8]     |
  0.013 [0]     |
  0.015 [1]     |
  0.016 [1]     |
  0.018 [1]     |
  0.020 [3]     |
```

(continues on next page)

(continued from previous page)

```
  0.021 [2]     |


Latency distribution:
  10% in 0.0051 secs
  25% in 0.0052 secs
  50% in 0.0053 secs
  75% in 0.0053 secs
  90% in 0.0054 secs
  95% in 0.0054 secs
  99% in 0.0058 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0050 secs, 0.0211 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0051 secs, 0.0048 secs, 0.0209 secs
  resp read:    0.0000 secs, 0.0000 secs, 0.0003 secs

Status code distribution:
  [200] 10000 responses

# HTTP server with PyPy and Sqlite as storage backed (comment psycopg2 out from setup)
sudo pypy -m pip install --editable .[devel]
pypy runner --server-host 127.0.0.1:8080 --defaults
/root/go/bin/hey -n 1000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 10000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 10000 -c 1 http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20

Summary:
  Total:        21.4936 secs
  Slowest:      0.0139 secs
  Fastest:      0.0017 secs
  Average:      0.0021 secs
  Requests/sec: 465.2553

  Total data:   319140000 bytes
  Size/request: 31914 bytes

Response time histogram:
  0.002 [1]     |
  0.003 [9489]  |
  0.004 [204]   |
  0.005 [77]    |
  0.007 [1]     |
  0.008 [146]   |
  0.009 [77]    |
  0.010 [2]     |
  0.011 [2]     |
  0.013 [0]     |
  0.014 [1]     |
```

(continues on next page)

(continued from previous page)

```
Latency distribution:
  10% in 0.0018 secs
  25% in 0.0019 secs
  50% in 0.0020 secs
  75% in 0.0020 secs
  90% in 0.0021 secs
  95% in 0.0029 secs
  99% in 0.0071 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0017 secs, 0.0139 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0020 secs, 0.0016 secs, 0.0127 secs
  resp read:    0.0000 secs, 0.0000 secs, 0.0004 secs

Status code distribution:
  [200] 10000 responses

# HTTPS server with PyPy and Sqlite as storage backed (comment psycopg2 out from␣
↪setup)
pypy runner --server-host 127.0.0.1:8080 --server-ssl-cert ./server.crt --server-ssl-
↪key ./server.key --defaults
/root/go/bin/hey -n 1000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 1000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?limit=20
/root/go/bin/hey -n 10000 -c 1 https://127.0.0.1:8080/api/snippy/rest/snippets?
↪limit=20

Summary:
  Total:        108.0445 secs
  Slowest:      0.0409 secs
  Fastest:      0.0075 secs
  Average:      0.0108 secs
  Requests/sec: 92.5545

  Total data:   319140000 bytes
  Size/request: 31914 bytes

Response time histogram:
  0.008 [1]     |
  0.011 [7368]  |
  0.014 [513]   |
  0.018 [721]   |
  0.021 [8]     |
  0.024 [1377]  |
  0.028 [9]     |
  0.031 [1]     |
  0.034 [0]     |
  0.038 [1]     |
  0.041 [1]     |


Latency distribution:
  10% in 0.0078 secs
```

(continues on next page)

(continued from previous page)

```
  25% in 0.0079 secs
  50% in 0.0081 secs
  75% in 0.0138 secs
  90% in 0.0215 secs
  95% in 0.0217 secs
  99% in 0.0226 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0067 secs, 0.0075 secs, 0.0409 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0039 secs, 0.0021 secs, 0.0180 secs
  resp read:    0.0001 secs, 0.0001 secs, 0.0007 secs

Status code distribution:
  [200] 10000 responses
```

```
# Bench POST with ab.
{"data":[{"type":"snippet","attributes":{"data":["docker rm $(docker ps --all -q -f␣
↪status=exited)"],"brief":"testing performance","name":"testing performance","groups
↪":["default"],"tags":["test","performance"],"links":["https://jsonlint.com/"],
↪"versions":["ab==1.0"],"filename":"ab.txt"}}]}
ab -p snippet.txt -T application/vnd.api+json -c 1 -n 1000 http://127.0.0.1:8080/api/
↪snippy/rest/snippets

# Bench POST with hey.
/root/go/bin/hey -m POST -T application/vnd.api+json -D snippet.txt -n 1000 -c 1␣
↪http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20

Summary:
  Total:        2.8403 secs
  Slowest:      0.0255 secs
  Fastest:      0.0027 secs
  Average:      0.0028 secs
  Requests/sec: 352.0781

  Total data:   494000 bytes
  Size/request: 494 bytes

Response time histogram:
  0.003 [1]     |
  0.005 [994]   |
  0.007 [3]     |
  0.010 [0]     |
  0.012 [0]     |
  0.014 [0]     |
  0.016 [0]     |
  0.019 [0]     |
  0.021 [1]     |
  0.023 [0]     |
  0.025 [1]     |


Latency distribution:
  10% in 0.0027 secs
  25% in 0.0027 secs
```

(continues on next page)

(continued from previous page)

```
  50% in 0.0028 secs
  75% in 0.0028 secs
  90% in 0.0029 secs
  95% in 0.0030 secs
  99% in 0.0035 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0027 secs, 0.0255 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0002 secs
  resp wait:    0.0027 secs, 0.0026 secs, 0.0246 secs
  resp read:    0.0000 secs, 0.0000 secs, 0.0003 secs

Status code distribution:
  [409] 1000 responses

/root/go/bin/hey -m POST -T application/vnd.api+json -D snippet.txt -n 1000 -c 1␣
↪http://127.0.0.1:8080/api/snippy/rest/snippets?limit=20

Summary:
  Total:        2.8316 secs
  Slowest:      0.0184 secs
  Fastest:      0.0027 secs
  Average:      0.0028 secs
  Requests/sec: 353.1552

  Total data:   494000 bytes
  Size/request: 494 bytes

Response time histogram:
  0.003 [1]     |
  0.004 [987]   |
  0.006 [9]     |
  0.007 [0]     |
  0.009 [0]     |
  0.011 [2]     |
  0.012 [0]     |
  0.014 [0]     |
  0.015 [0]     |
  0.017 [0]     |
  0.018 [1]     |


Latency distribution:
  10% in 0.0027 secs
  25% in 0.0027 secs
  50% in 0.0028 secs
  75% in 0.0028 secs
  90% in 0.0029 secs
  95% in 0.0030 secs
  99% in 0.0045 secs

Details (average, fastest, slowest):
  DNS+dialup:   0.0001 secs, 0.0027 secs, 0.0184 secs
  DNS-lookup:   0.0000 secs, 0.0000 secs, 0.0000 secs
  req write:    0.0000 secs, 0.0000 secs, 0.0003 secs
  resp wait:    0.0027 secs, 0.0025 secs, 0.0167 secs
```

(continues on next page)

(continued from previous page)

```
  resp read:    0.0000 secs, 0.0000 secs, 0.0003 secs

Status code distribution:
  [409] 1000 responses
```

7.5 Releasing

The are two make targets for release. The `prepare-release` runs all tests and compiles the release packages. The `release-upload` will upload the new release.

The release steps that are automated in Makefile are documented here.

This is a semi automated release process that is not completed. Some of the steps must be executed manually as instructed below.

7.5.1 Preparations

```
# Update PyPy dependencies
sudo dnf install pypy3 -y
sudo dnf install pypy3-devel -y
sudo dnf install postgresql-devel -y
sudo dnf update pypy3 -y
sudo dnf update pypy3-devel -y
sudo dnf update postgresql-devel -y

pypy3 -m ensurepip
pypy3 -m pip install --upgrade pip setuptools wheel
pypy3 -m pip install .[tests]

# Manual: Start PostgreSQL.
sudo docker stop postgres
sudo docker rm postgres
sudo docker run -d --name postgres -e POSTGRES_PASSWORD=postgres -p␣
↪5432:5432 -d postgres

# Manual: Remove runnning Snippy containers
sudo docker stop snippy
sudo docker rm snippy

# Manual: Start virtual environment.
workon snippy

# Manual: Set the current development version and the new tagged
#         versions in Makefile.
DEV_VERSION := 0.10a0
TAG_VERSION := 0.10.0

# Run release preparations.
make prepare-release -s

    # Update Python setuptools, wheels and Twine.
    make upgrade-wheel -s
```

(continues on next page)

(continued from previous page)

```
    # Update version numbers in project. This target fails if
    # there are development versions found.
    make upgrade-tool-version -s

    # Rune automated tests and checks. The server tests are run
    # for each storage backend because the server uses the same
    # storage as rest of the tests.
    make test-release
```

7.5.2 Run tests with PyPy

```
# Example installation for Fedora 28.
make clean
make clean-db
dnf install pypy3
dnf install pypy3-devel
dnf install postgresql-devel
make upgrade-wheel PYTHON=pypy3
make install-devel PYTHON=pypy3
pypy3 -m ensurepip
pypy3 -m pip install --upgrade pip setuptools wheel
pypy3 -m pip install --editable .[devel]
pypy3 -m pytest -x ./tests/test_*.py --cov snippy -m "server"
pypy3 runner --help
pypy3 runner import --defaults --all
pypy3 runner --server-host 127.0.0.1:8080 -vv
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?limit=4" -H
↪"accept: application/vnd.api+json"
```

7.5.3 Run tests with PostgreSQL

```
# The test-all target runs test with Sqlite and PostgreSQL.
docker run -d --name postgres -e POSTGRES_PASSWORD=postgres -p 5432:5432 -d␣
↪postgres
make clean
make clean-db
make test-all
make test-postgresql
```

7.5.4 Run tests with HTTP server

```
# Generate TLS sertificates for server.
openssl req -x509 -newkey rsa:4096 -nodes -keyout server.key -out server.crt␣
↪-days 356 -subj "/C=US/O=Snippy/CN=127.0.0.1"
python runner --server-host 127.0.0.1:8080 -vv --server-ssl-cert ./server.
↪crt --server-ssl-key ./server.key
curl -k -s -X GET "https://127.0.0.1:8080/api/snippy/rest/snippets?
↪sall=docker&limit=2" -H "accept: application/vnd.api+json"
```

7.5.5 Test local installation

```
make clean
make clean-db
pip uninstall snippy -y
pip install .
snippy --help
snippy search --sall .
snippy import --defaults
snippy import --defaults --solutions
snippy import --defaults --references
snippy search --sall docker
rm -f ${HOME}/devel/temp/snippy.db
snippy import --defaults --storage-path ${HOME}/devel/temp
snippy import --defaults --solutions --storage-path ${HOME}/devel/temp
snippy import --defaults --references --storage-path ${HOME}/devel/temp
snippy --server-host 127.0.0.1:8080 --storage-path ${HOME}/devel/temp &
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?limit=4" -H
↪"accept: application/vnd.api+json"
pkill snippy
```

7.5.6 Test docker installation

```
# Compile docker image.
su
make clean
make clean-db
docker rmi --force $(docker images --filter=reference="*/snippy*:*" -q)
docker rm $(docker ps --all -q -f status=exited)
docker images -q --filter dangling=true | xargs docker rmi
docker images
make docker

# Run CLI commands with docker image.
docker run --rm --env SNIPPY_LOG_JSON=0 heilaaks/snippy --help
docker run --rm --env SNIPPY_LOG_JSON=0 heilaaks/snippy search --sall docker

# Run server with Sqlite database.
docker run -d --publish=127.0.0.1:8080:32768/tcp --name snippy heilaaks/
↪snippy --defaults -vv
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&
↪limit=2" -H "accept: application/vnd.api+json"
docker logs snippy
docker stop snippy
docker rm snippy
docker run --env SNIPPY_SERVER_HOST=127.0.0.1:8080 --net=host --name snippy -
↪-detach heilaaks/snippy --debug
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&
↪limit=2" -H "accept: application/vnd.api+json"
docker logs snippy
docker stop snippy
docker rm snippy

# Login into Docker image (requires change to Dockerfile).
docker exec -it heilaaks/snippy /bin/sh
```

(continues on next page)

(continued from previous page)

```
cd /
du -ah | sort -n -r | head -n 50
find / -name '*pycache*'

# Run server with PostgreSQL database.
docker run -d --net="host" --name snippy heilaaks/snippy --server-host 127.0.
↪0.1:8080 --storage-type postgresql --storage-host localhost:5432 --storage-
↪database postgres --storage-user postgres --storage-password postgres --
↪defaults --log-json -vv
#docker run -d --publish=8080:8080 --name snippy heilaaks/snippy --storage-
↪type postgresql --storage-host postgres:5432 --storage-database postgres --
↪storage-user postgres --storage-password postgres --defaults --log-json -vv
curl -s -X POST "http://127.0.0.1:8080/api/snippy/rest/snippets" -H "accept:␣
↪application/vnd.api+json; charset=UTF-8" -H "Content-Type: application/vnd.
↪api+json; charset=UTF-8" -d '{"data":[{"type": "snippet", "attributes": {
↪"data": ["docker ps"]}}]}'
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&
↪limit=2" -H "accept: application/vnd.api+json"
docker logs snippy
docker stop snippy
docker rm snippy

# Login to container to see security hardening and size.
find / -perm +6000 -type f -exec ls -ld {} \;
find / -perm +6000 -type f -exec chmod a-s {} \; || true # Check defang ->␣
↪Should return zero files.
du -a -h / | sort -n -r | head -n 20
```

7.5.7 Create new asciinema

```
# pip uninstall snippy --yes
deactivate
pip uninstall snippy --yes
make clean-all
pip install . --user

# Clear existing resources.
cd ~/snippy
cp ~/devel/snippy/docs/release/record-asciinema.sh ../
chmod 755 ../record-asciinema.sh
rm -f ../snippy.cast
sudo docker stop snippy
sudo docker rm snippy
rm ./*
clear

# Disable and enable terminal linewrap
printf '\033[?7l'
clear
#printf '\033[?7h'

# Start recording.
asciinema rec ../snippy.cast -c ../record-asciinema.sh

# Play recording.
```

(continues on next page)

(continued from previous page)

```
asciinema play ../snippy.cast

# Upload recording
asciinema upload ../snippy.cast

# Change the README file to link to new asciinema cast.
```

7.5.8 Test PyPI installation

```
# Test PyPI installation before official release into PyPI.
> https://testpypi.python.org/pypi
make clean-all
python setup.py sdist bdist_wheel
twine upload --repository-url https://test.pypi.org/legacy/ dist/*
pip uninstall snippy -y
pip3 uninstall snippy -y
pip install --index-url https://test.pypi.org/simple/ snippy
snippy --help
snippy import --defaults --all
snippy search --sall docker
pip uninstall snippy -y
pip3 install --index-url https://test.pypi.org/simple/ snippy
snippy --help
snippy import --defaults --all
snippy search --sall docker
pip3 uninstall snippy -y
pip3 install --user --index-url https://test.pypi.org/simple/ snippy
pip uninstall snippy -y
pip install --user --index-url https://test.pypi.org/simple/ snippy
which snippy
snippy --help
snippy import --defaults --all
snippy search --sall docker
pip3 uninstall snippy -y
pip uninstall snippy -y
```

7.5.9 Pre-release

1. Verify data in CHANGELOG.rst
    1. Update the CHANGELOG.rst release date if needed.
    2. Push changes to master.

7.5.10 Release

1. Make tag

```
git tag -a v0.10.0 -m "Add new release 0.1.0"
git push -u origin v0.10.0
```

2. Release in PyPI

```
make cleana-all
python setup.py sdist bdist_wheel
twine upload dist/*
```

3. Test PyPI release

```
sudo pip uninstall snippy -y
pip install snippy --user
snippy --help
snippy import --defaults
snippy import --defaults --solutions
snippy search --sall docker
```

4. Release in Docker Hub

```
su
docker rmi --force $(docker images --filter=reference="*/snippy*:*" -q)
docker rm $(docker ps --all -q -f status=exited)
docker images -q --filter dangling=true | xargs docker rmi
docker images
make docker
docker login docker.io
docker tag 86961c480391 docker.io/heilaaks/snippy:v0.10.0
docker tag 86961c480391 docker.io/heilaaks/snippy:latest
docker images
docker push docker.io/heilaaks/snippy:v0.10.0
docker push docker.io/heilaaks/snippy:latest
```

5. Test Docker release

```
su
docker rmi --force $(docker images --filter=reference="*/snippy*:*" -q)
docker rm $(docker ps --all -q -f status=exited)
docker images -q --filter dangling=true | xargs docker rmi
docker images
docker pull heilaaks/snippy
docker run heilaaks/snippy:latest --help
docker run heilaaks/snippy:latest search --sall docker
docker run -d --publish=127.0.0.1:8080:32768/tcp --name snippy heilaaks/snippy -vv
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&limit=2
↪" -H "accept: application/vnd.api+json"
docker stop snippy
docker rm snippy
docker run --env SNIPPY_SERVER_HOST=127.0.0.1:8080 --net=host --name snippy --
↪detach heilaaks/snippy --debug
curl -s -X GET "http://127.0.0.1:8080/api/snippy/rest/snippets?sall=docker&limit=2
↪" -H "accept: application/vnd.api+json"
docker stop snippy
docker rm snippy
```

6. Release news

   1. Make new release in Github.

7.6 Modules

7.6.1 snippy.logger

Description

Logger class offers logger for each caller based on the given module name. The configuration is controlled by global settings that are inherited by every logger.

The effective log level for all the loggers created under the 'snippy' logger namespace is inherited from the root logger which controls the log level. This relies on that the module level logger does not set the level and it remains as `NOTSET`. This causes module level logger to propagate the log record to parent where it eventually reaches the `snippy` top level namespace that is just below the `root` logger.

Design

Note: This chapter describes the Snippy logging design and rules, not the Logger class behaviour.

Note: The are the logging rules that must be followed.

1. Only OK or NOK with cause text must be printed with default settings.
2. There must be no logs printed to user.
3. There must be no exceptions printed to user.
4. Exceptions logs are printed as INFO and all other logs as DEBUG.
5. Variables printed in logs must be separated with colon.
6. All other than error logs must be printed as lower case string.
7. The –debug option must print logs without filters in full-length.
8. The -vv option must print logs in lower case and one log per line.
9. All external libraries must follow the same log format.
10. All logs must be printed to stdout.

Overview

There are two levels of logging verbosity. All logs are printed in full length without modifications with the `--debug` option unless the maximum log message length for safety and security reason is exceeded. The very verbose option `-vv` prints limited length log messages with all lower case letters.

There are two formats for logs: text (default) and JSON. JSON logs can be enabled with the `--log-json` option. A JSON log has more information fields than the text formatted log. When the `-vv` option is used with JSON logs, it truncates log message in the same way as with the text logs.

All logs including Gunicorn server logs, are formatted to match format defined in this logger.

All logs are printed to stdout with the exception of command line parse failures that are printed to stdout.

Text logs are optimized for a local development done by for humans and JSON logs for automation and analytics.

There are no logs printed to users by default. This applies also to error logs.

Timestamps

Timestamps are in local time with text formatted logs. In case of JSON logs, the timestamp is in GMT time zone and it follows strictly the ISO8601 format. Text log timestamp is presented in millisecond granularity and JSON log in microsecond granularity.

Python 2 does not support timezone parsing. The `%z` directive is available only from Python 3.2 onwards. From Python 3.7 and onwards, the datetime `strptime` is able to parse timezone in format that includes colon delimiter in UTC offset.

```
>>> import datetime
>>>
>>> timestamp = '2018-02-02T02:02:02.000001+00:00'
>>>
>>> # Python 3.7 and later
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f%z')
>>>
>>> # Python 3 before 3.7
>>> timestamp = timestamp.replace('+00:00', '+0000')
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f%z')
>>>
>>> # Python 2.7
>>> timestamp = timestamp[:-6]  # Remove last '+00:00'.
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f')
```

Log levels

The log levels are are from Python logger but they follow severity level names from RFC 5424. There is a custom security level reserved only for security events.

Operation ID (OID)

All logs include operation ID that uniquely identifies all logs within specific operation. The operation ID must be refreshed by logger user after each operation is completed or the method must be wrapped with the `@Logger.timeit` decorator which takes care of the OID refreshing.

Security

There is a custom security level above critical level. This log level must be used only when there is a suspected security related event.

There is a hard maximum for log messages length for safety and security reasons. This tries to prevent extremely long log messages which may cause problems for the server.

Examples

```
# Variable printed at the end of log message is separated with colon.
2018-06-03 19:20:54.838 snippy[5756] [d] [b339bab5]: configured option server: true

# Variable printed in the middle of log message is separated colons and
# space from both sides. The purpose is to provide possibility to allow
# log message post processing and to parse variables from log messages.
2018-06-03 19:20:54.838 snippy[5756] [d] [b339bab5]: server ip: 127.0.0.1 :and port:␣
↪8080
```

logger: Logging services.

class snippy.logger.**Logger**
Global logging service.

classmethod get_logger(*name='snippy.logger'*)
Get logger.

A custom logger adapater is returned to support a custom log level and additional logging parameters.

Parameters **name** (*str*) – Name of the module that requests a Logger.

Returns CustomLoggerAdapter logger to be used by caller.

Return type obj

classmethod configure(*config*)
Set and update logger configuration.

The `debug` and `very_verbose` options have precedence over the `quiet` option. That is, either of the debug options are enabled, the quiet option does not have any effect.

Parameters **config** (*dict*) – Logger configuration dictionary.

Examples

```
>>> Logger.configure({'debug': True,
>>>                   'log_json': True,
>>>                   'log_msg_max': Logger.DEFAULT_LOG_MSG_MAX,
>>>                   'quiet': False,
>>>                   'very_verbose': False})
```

static reset()
Reset log level to default.

static remove()
Delete all logger handlers.

classmethod refresh_oid()
Refresh operation ID (OID).

The OID is used to separate logs within one operation. The helps post-processing of the logs by allowing for example querying all the logs in failing operation.

classmethod print_stdout(*message*)
Print output to stdout.

Take care of nasty details like broken pipe when printing to stdout.

Parameters **message** (*str*) – Text string to be printed to stdout.

classmethod print_status(*status*)
Print status information like exit cause or server running.

Print user formatted log messages unless the JSON log formating is enabled. The `debug` and `very_verbose` options have precedence over the quiet option.

If JSON logs are used, the format of the log must always be JSON. This is important for server installation where post processing of logs might be done elsewhere and incorrectly formatted logs may be discarded or cause errors.

In order to post process a JSON log, the dictionary structure must always follow the same format. Because of this, the log is pushed always as a debug level log regardless of the log level to get the formatting done.

Parameters **status** (*str*) – Status to be printed on stdout.

static timeit(*method=None*, *refresh_oid=False*)
Time method by measuring it latency.

The operation ID (OID) is refreshed at the end.

Parameters

- **method** (`str`) – Name of the method calling the timeit.
- **refresh_oid** (`bool`) – Define if operation ID is refreshed or not.

Returns Timeit wrapper function for decorators.

Return type obj

static remove_ansi(*message*)
Remove all ANSI escape codes from log message.

Parameters **message** (`str`) – Log message which ANSI escape codes are removed.

Returns Same log message but without ANSI escape codes.

Return type str

static debug()
Debug Logger by printing logging hierarchy.

class snippy.logger.**CustomLoggerAdapter**(*logger*, *extra*)
Custom logger adapter.

The logging.LoggerAdapter does not support custom log levels and therefore they need to be implemented here.

security(*msg*, **args*, ***kwargs*)
Customer log level for security events.

Parameters **msg** (`str`) – Log message as a string.

class snippy.logger.**CustomFormatter**(**args*, ***kwargs*)
Custom log formatting.

format(*record*)
Format log record.

Text logs are optimized for a local development done by for humans and JSON logs for automation and analytics. Text logs are printed by default unless the `log_json` option is activated.

The `debug` option prints logs "as is" in full length unless the log message security limit is reached. Text logs are pretty printed with the debug option.

The `very_verbose` option truncates log message to try to keep one log per line for easier reading. The very verbose option prints the whole log in all lower case letters. The very verbose option is made for a local development to provide faster overview of logs compared to debug option output.

There is a maximum limitation for log message for safety and security reasons. The security maximum is tested after the very verbose option because it already truncates the log.

Gunicorn logs have special conversion for info level logs. In order to follow the Snippy logging standard, which defines the usage of debug level, Gunicorn informative logs are converted to debug level logs. Warning and error levels do not get converted because in these cases the level is considered relevant for user.

Parameters **record** (`obj`) – Logging module LogRecord.

Returns Log string.

Return type str

formatTime (*record*, *datefmt=None*)

Format log timestamp.

JSON logs are printed in ISO8601 format with UTC timestamps. All other logs are printed in local time with space between date and time instead of 'T' because of better readability.

The ISO8601 formatted JSON timestamp is set in microseconds. It seems that the msecs field of the logging record contains mseconds as floating point number. It is assumed that the microseconds can be read by reading three significat digits after point.

Python 2 does not support timezone parsing. The `%z` directive is available only from Python 3.2 onwards. From Python 3.7 and onwards, the datetime `strptime` is able to parse timezone in format that includes colon delimiter in UTC offset.

Parameters

- **record** (*obj*) – Logging module LogRecord.
- **datefmt** (*str*) – Datetime format as accepted by time.strftime().

Returns Log timestamp in string format.

Return type str

Examples

```
>>> import datetime
>>>
>>> timestamp = '2018-02-02T02:02:02.000001+00:00'
>>>
>>> # Python 3.7 and later
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f%z')
>>>
>>> # Python 3 before 3.7
>>> timestamp = timestamp.replace('+00:00', '+0000')
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f%z')
>>>
>>> # Python 2.7
>>> timestamp = timestamp[:-6]  # Remove last '+00:00'.
>>> datetime.datetime.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%f')
```

class snippy.logger.**CustomFilter** (*name=''*)

Customer log filter.

filter (*record*)

Filtering with dynamic operation ID (OID) setting.

Parameters **record** (*obj*) – Logging module LogRecord.

class snippy.logger.**CustomGunicornLogger** (*cfg*)

Custom logger for Gunicorn HTTP server.

setup (*cfg*)

Custom setup.

Disable all handlers under the 'gunicorn' namespace and prevent log propagation to root logger. The loggers under the 'snippy' namespace will take care of the log writing for Gunicorn server.

Both Gunicor error and access log categories are printed from the same namespace. In case of 'snippy.server.gunicorn.error', informative logs in JSON format would have this in the class `name` attribute which is considered to be misleading for other than error logs.

Parameters **`cfg`** (*`obj`*) – The Gunicorn server class Config() object.

`snippy.logger.`**`getrandbits`**`(`*`k`*`)` → x. Generates a long int with k random bits.

7.6.2 snippy.cause

Service

Cause class offers storage services for normal and error causes. The causes are stored in a list where user can get all the failues that happened for example during the operation.

All causes are operated with predefind constants for HTTP causes and short descriptions of the event.

`class` `snippy.cause.`**`Cause`**

Cause code services.

`classmethod reset``()`

Reset cause to initial value.

`classmethod push``(`*status*, *message*`)`

Append cause to list.

Message will always contain only the string till the first newline. The reason is that the message may be coming from an exception which message may contain multiple lines. In this case it is always assumed that the first line contains the actual exception message. The whole message is always printed into log.

Parameters

- **`status`** (*`str`*) – One of the predefined HTTP status codes.
- **`message`** (*`str`*) – Description of the cause.

Examples

```
>>> Cause.push(Cause.HTTP_CREATED, 'content created')
```

`classmethod insert``(`*status*, *message*`)`

Insert cause as a first cause.

Parameters

- **`status`** (*`str`*) – One of the predefined HTTP status codes.
- **`message`** (*`str`*) – Description of the cause.

Examples

```
>>> Cause.insert(Cause.HTTP_CREATED, 'content created')
```

`classmethod is_ok``()`

Test if errors were detected.

The status is considered ok in following cases:

1. There are no errors at all.

2. There are only accepted error codes.
3. Content has been created without internal errors.

The last case is a special case. The problem is that currently the case where multiple contents are imported when some of them fail due to data already existing is considered successful. That is, user should get OK when importing a list of data when some of them are already imported. For this reason, the Created is searched without internal error.

The UUID collision is considered internal error because that field is set by the application.

Returns Define if the cause list can be considered ok.

Return type bool

classmethod http_status()

Return the HTTP status.

classmethod json_message()

Return errors in JSON data structure.

classmethod get_message()

Return cause message.

Cause codes follow the same rules as the logs with the title or message. If there are variables within the message, the variables are separated with colon. The end user message is beautified so that if there is more than one colon, it indicates that variable is in the middle of the message. This is not considered good layout for command line interface messages.

How ever, if there is only one colon, it is used to sepatate the last part which is considered clear for user.

Because of these rules, the colon delimiters are removed only if there is more than one.

Examples

1. cannot use empty content uuid for: delete :operation
2. cannot find content with content uuid: 1234567

classmethod print_message()

Print cause message.

classmethod print_failure()

Print only failure message.

classmethod debug()

Debug Cause.

7.6.3 snippy.config

Service

Global configuration.

class snippy.config.config.**Config**

Global configuration object.

classmethod init(*args*)

Initialize global configuration.

classmethod load(*source*)
: Load dynamic configuration from source.

classmethod reset()
: Reset configuration.

classmethod get_collection(*update=None*)
: Get collection of resources.

  Read collection of resources from the used configuration source. If a resource update is provided on top of configured content, the update is merged or migrated on top of the configuration.

  Parameters **update** (*Resource()*) – Content updates on top of configured content.

  Returns Configured content in Collection object.

  Return type Collection()

classmethod get_resource(*update*)
: Get resource.

  Read a resource from the used configuration source. If an update is provided on top of configured content, the update is merged or migrated on top of configuration.

  Parameters **update** (*Resource()*) – Update to be used on top of configuration.

  Returns Updated resource.

  Return type Resource()

classmethod server_schema()
: Get server API validation schema.

  Returns Server API schema to validate incoming HTTP requests.

  Return type str

classmethod server_schema_base_uri()
: Get server API schema base URI.

  Returns Path where the API schema is stored in URI format.

  Return type str

classmethod get_operation_file(*collection=None*)
: Return file for operation.

  Use the resource filename field only in case of export operation when there is a single resource in collection and when user did not define target file from command line.

  If collection is provided with more than one resource, the operation file is still updated. The collection might be a search result from different category than originally defined.

  Parameters **collection** (*Collection*) – Resources in Collection container.

  Returns Operation filename.

  Return type string

classmethod is_supported_file_format()
: Test if file format is supported.

classmethod default_content_file(*category*)
: Return default content file.

  Parameters **category** (*str*) – User defined content category.

Returns Filename with absolute path.

Return type string

classmethod validate_search_context(*collection*, *operation*)
Validate content search context.

classmethod is_search_criteria()
Test if any of the search criterias were used.

static utcnow()
Get UTC time stamp in ISO8601 format.

classmethod debug()
Debug Config.

Do not print any configuration attrubutes from here. Use only the string presentation of the Config class to print attributes. This is because of security reasons.

7.6.4 snippy.config.source.cli

Service

Command line configuration source.

class `snippy.config.source.cli.`**Cli**(*args*)
Command line argument parser.

7.6.5 snippy.config.source.api

Service

REST API configuration source.

class `snippy.config.source.api.`**Api**(*category*, *operation*, *parameters*)
API parameter management.

7.6.6 snippy.config.source.base

Service

Configuration source base class.

class `snippy.config.source.base.`**ConfigSourceBase**(*derived*, *parameters=None*)
Base class for configuration sources.

init_conf(*parameters*)
Initialize configuration parameters.

Configuration can be read from command line interface or received from API query. It is also possible to configure for example server and storage parameters with environment variables. The precedence of configuration is:

1. Command line option.
2. Environment variable.
3. Hard coded default.

Parameters **parameters** (*dict*) – Parameters from configuration source.

`data`
Get content data.

`brief`
Get content brief.

`description`
Get content description.

`name`
Get content name.

`groups`
Get content groups.

`tags`
Get content tags.

`links`
Get content links.

`source`
Get content source.

`versions`
Get content versions.

`filename`
Get content filename.

`sall`
Get 'search all' keywords.

`scat`
Get 'search categories' keywords.

`stag`
Get 'search tag' keywords.

`sgrp`
Get 'search groups' keywords.

`search_filter`
Get search regexp filter.

`search_limit`
Get search result limit.

`search_offset`
Get search offset from start.

`sort_fields`
Get sorted fields.

`remove_fields`
Get removed fields.

`reset_fields`
Get reset fields.

`run_server`
Get bool value that tells if Snippy server is run.

`server_base_path_rest`
Get REST API base path.

`server_host`
Get server host IP and port

`identity`
Get content identity.

`classmethod read_env`(*option*, *default*)
Read parameter from optional environment variable.

Read parameter value from environment variable or return given default value. Environment variable names follow the same command line option naming convesion with modifications:

1. Leading hyphens are removed.
2. Option casing is converted to full upper case.
3. Hyphens are replaced with underscores.
4. `SNIPPY_` prefix is added,

For example corresponding environment variable for the `--server-host` command line option is `SNIPPY_SERVER_HOST`.

Parameters

- **`option`** (*`str`*) – Command line option.
- **`default`** – Default value.

Returns Same command line option name as received with value.

Return type tuple

`classmethod read_arg`(*option*, *default*, *args*)
Read command line argument directly from sys.argv.

This is intenden to be used only in special cases that are related to debug options. The debug options are required for example to print logs before parsing command line arguments.

This function supports only bool and integer values because there are currently no other use cases.

This follows the standard command option parsing precedence:

1. Command line option.
2. Environment variable.
3. Hard coded default.

Parameters

- **`option`** (*`string`*) – Command line option.
- **`default`** – Default value if option is not configured.
- **`args`** (*`list`*) – Argument list received from command line.

Returns Value for the command line option.

Return type int,bool

7.6.7 snippy.content.parser

Service

Parser class offers a parser to extract content fields from text source.

class `snippy.content.parser.`**`Parser`**(*filetype*, *timestamp*, *source*, *collection*)
Parse content attributes from text source.

`read`()
Read content attributes from text source.

Text source specific parser is run against the provided text string. The text source can be either the tool specific text or Markdown template.

7.6.8 snippy.content.parsers.base

Service

Content parser base class offers basic parsing methods.

class `snippy.content.parsers.base.`**`ContentParserBase`**
Base class for text content parser.

classmethod **`format_data`**(*category*, *value*)
Convert content data to utf-8 encoded tuple of lines.

Content data is stored as a tuple with one line per element.

All but solution data is trimmed from right for every line. In case of solution data, it is considered that user wants to leave it as is. Solutions are trimmed only so that there will be only one newline at the end of the solution data.

Any value including empty string is considered as a valid data.

Parameters

- **`category`** (*`str`*) – Content category.
- **`value`** (*`str, list`*) – Content data in string or list.

Returns Tuple of utf-8 encoded unicode strings.

Return type tuple

classmethod **`format_string`**(*value*)
Convert content string value to utf-8 encoded string.

Parameters **`value`** (*`str, list, tuple`*) – Content field value in string, list or tuple.

Returns Utf-8 encoded unicode string.

Return type str

classmethod **`format_search_keywords`**(*value*)
Convert search keywords to utf-8 encoded tuple.

If the value is None it indicates that the search keywords were not given at all.

The keyword list may be empty or it can contain empty string. Both cases must be evaluated to ‘match any’.

Parameters **`value`** (*`str, list, tuple`*) – Search keywords in string, list or tuple.

Returns Tuple of utf-8 encoded keywords.

Return type tuple

classmethod format_list(*keywords*, *unique=True*, *sort_=True*)

Convert list of keywords to utf-8 encoded list of strings.

Parse user provided keyword list. The keywords are for example groups, tags search all keywords or versions. It is possible to use string or list context for the given keywords. In case of list context for the given keywords, each element in the list is split separately.

The keywords are split in word boundary.

The dot is a special case. It is allowed for the regexp to match and print all records.

Content versions field must support specific mathematical operators that do not split the keyword.

Parameters

- **keywords** (*str, list, tuple*) – Keywords in string, list or tuple.
- **unique** (*bool*) – Return unique keyword values.
- **sort** (*bool*) – Return sorted keywords.

Returns Tuple of utf-8 encoded keywords.

Return type tuple

classmethod format_links(*links*, *unique=True*)

Convert links to utf-8 encoded list of links.

Parse user provided link list. Because URL and keyword have different forbidden characters, the methods to parse keywords are similar but still they are separated. URLs can be separated only with space, bar or newline. Space and bar characters are defined 'unsafe characters' in URL character set [1]. The newline is always URL encoded so it does not appear as newline.

The newline is supported here because that is used to separate links in text input.

Links are not sorted. The reason is that the sort is done based on content category. The content category is not know for sure when command options are parsed in this class. For this reason, the sort is always made later in the Resource when content category is known for sure.

[1] https://perishablepress.com/stop-using-unsafe-characters-in-urls/

Parameters

- **links** (*str, list, tuple*) – Links in a string, list or tuple.
- **unique** (*bool*) – Return unique keyword values.

Returns Tuple of utf-8 encoded links.

Return type tuple

classmethod format_versions(*versions*)

Convert versions to utf-8 encoded list of version.

Only specific operators between key value versions are allowed.

Parameters versions (*str, list, tuple*) – Versions in a string, list or tuple.

Returns Tuple of utf-8 encoded versions.

Return type tuple

classmethod parse_groups(*category*, *regexp*, *text*)

Parse content groups from text string.

There is always a default group added into the content group field.

Parameters

- **category** (*str*) – Content category.
- **regexp** (*re*) – Compiled regexp to search groups.
- **text** (*str*) – Content text string.

Returns Tuple of utf-8 encoded groups.

Return type tuple

classmethod parse_links(*category*, *regexp*, *text*)

Parse content links from text string.

Parameters

- **category** (*str*) – Content category.
- **regexp** (*re*) – Compiled regexp to search links.
- **text** (*str*) – Content text string.

Returns Tuple of utf-8 encoded links.

Return type tuple

classmethod parse_versions(*category*, *regexp*, *text*)

Parse content versions from text string.

Version strings are validated. Only versions which pass the validation rules are stored. The rules allow only specific operators between key value pairs.

Parameters

- **category** (*str*) – Content category.
- **regexp** (*re*) – Compiled regexp to search versions.
- **text** (*str*) – Content text string.

Returns Tuple of utf-8 encoded versions.

Return type tuple

classmethod remove_template_fillers(*content*)

Remove tags and examples from content.

There are examples and tags in content templates that need to be removed before further processing the content. This method removes all the unnecessary tags and examples that are set to help user to fill a content template.

The received content can be text for Markdown based.

Parameters **content** (*str*) – Content text or Markdown string.

Returns String without content fillers.

Return type str

classmethod to_unicode(*value*, *strip_lines=True*)

Convert value to utf-8 coded unicode string.

If the value is already an unicode character, it is assumed that it is a valid utf-8 encoded unicode character.

The conversion quarantees one newline at the end of string.

Parameters

- **value** (*str, list, tuple*) – Value in a string, list or tuple.
- **strip_lines** (*bool*) – Defines if all lines are stripped.

Returns Utf-8 encoded unicode string.

Return type str

7.6.9 snippy.content.parsers.text

Service

Content parser for text content.

class snippy.content.parsers.text.**ContentParserText**(*timestamp*, *text*, *collection*)
Parse content from text template.

read_collection()
Read collection from the given text source.

7.6.10 snippy.content.parsers.mkdn

Service

Content parser for Markdown content.

class snippy.content.parsers.mkdn.**ContentParserMkdn**(*timestamp*, *text*, *collection*)
Parse content from Markdown template.

read_collection()
Read collection from the given Markdown source.

7.6.11 snippy.content.parsers.dict

Service

Content parser for YAML and JSON content.

class snippy.content.parsers.dict.**ContentParserDict**(*timestamp*, *dictionary*, *collection*)
Parse content from dictionary.

read_collection()
Read collection from the given dictionary source.

7.6.12 snippy.storage.storage

Service

Storage class offers database agnosting storage services. This abstracts the actual database solution from rest of the implementation.

class snippy.storage.storage.**Storage**
Storage management for content.

create(*collection*)
Create new content.

Parameters collection (*Collection*) – Content container to be stored into database.

search(*scat=()*, *sall=()*, *stag=()*, *sgrp=()*, *search_filter=None*, *uuid=None*, *digest=None*, *identity=None*, *data=None*)
Search content.

Parameters

- **scat** (*tuple*) – Search category keyword list.
- **sall** (*tuple*) – Search all keyword list.
- **stag** (*tuple*) – Search tag keyword list.
- **sgrp** (*tuple*) – Search group keyword list.
- **search_filter** (*str*) – Regexp filter to limit search results.
- **uuid** (*str*) – Search specific uuid or part of it.
- **digest** (*str*) – Search specific digest or part of it.
- **identity** (*str*) – Search specific digest or UUID or part of them.
- **data** (*str*) – Search specific content data or part of it.

Returns Search result in Collection of content.

Return type Collection

unique_values(*field*)
Get unique values for given field.

Parameters **field** (*str*) – Content field which unique values are read.

Returns List of unique values for give field.

Return type tuple

update(*digest*, *resource*)
Update resource specified by digest.

Parameters

- **digest** (*str*) – Content digest that is udpated.
- **resource** (*Resource*) – A single Resource() container that contains updates.

delete(*digest*)
Delete content.

Parameters **digest** (*str*) – Content digest that is deleted.

export_content(*scat=()*)
Export content.

Parameters **scat** (*tuple*) – Search category keyword list.

import_content(*collection*)
Import content.

Parameters **collection** (*Collection*) – Content container to be imported into database.

disconnect()
Disconnect storage.

debug()
Debug storage.

7.6.13 snippy.storage.database

Service

SqliteDb class offers database implementation for the Storage class.

`snippy.storage.`**`database`**

alias of *`snippy.storage.database`*

CHAPTER 8

License

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

```
  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
```

(continues on next page)

(continued from previous page)

```
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Remote Network Interaction; Use with the GNU General Public License.

  Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your version
supports such interaction) an opportunity to receive the Corresponding
Source of your version by providing access to the Corresponding Source
from a network server at no charge, through some standard or customary
```

(continues on next page)

(continued from previous page)

(continues on next page)

(continued from previous page)

HTTP Routing Table

Python Module Index

Index