
Documentation libranet_{logging}

Release 1.4.dev0

owner

Apr 18, 2023

Contents

1	Installation	2
2	Why use logging?	2
3	Goal of libranet-logging	3
4	Features	3
5	Usage	4
5.1	In your deployment	4
5.2	In your code	4
6	Unittesting	5
6.1	Introduction	5
6.2	Re-usable test-fixtures	5
6.3	Run tests	6
6.4	Run tests with code-coverage	7
7	Changelog	7
7.1	1.4 (unreleased)	7
7.2	1.3 (2023-01-24)	7
7.3	1.2 (2021-06-06)	7
7.4	1.1 (2020-02-13)	7
7.5	1.0 (2020-02-12)	8
7.6	0.5 (2019-08-19)	8
7.7	0.4 (2019-07-31)	8
7.8	0.3 (2019-05-28)	8
7.9	0.2 (2019-03-28)	8
7.10	0.1 (2019-03-28)	8
8	Security Policy	10
8.1	Supported Versions	10
8.2	Reporting a Vulnerability	10
9	MIT License	10
10	Contributors	11
11	How to contribute	11
11.1	Development environment setup	11
11.2	Issues and feature requests	11

12 Contributor Covenant Code of Conduct	12
12.1 Our Pledge	12
12.2 Our Standards	12
12.3 Enforcement Responsibilities	13
12.4 Scope	13
12.5 Enforcement	13
12.6 Enforcement Guidelines	13
12.7 Attribution	14
13 libranet_logging	14
13.1 Submodules	14
13.2 Package Contents	19
14 Indices and tables	21
Python Module Index	22
Index	23

Testing Linting Read the Docs Codecov PyPi Package MIT License

1 Installation

Install via pip:

```
> bin/pip install libranet-logging
```

Or add to your poetry-based project:

```
> poetry add libranet-logging
```

2 Why use logging?

Logfiles are your best-friend

- during development, where debugmode is developmentmode
- more important: while running in PRD,
 - it shows how the application is being used, by whom, and if it's successful
 - allows to become pro-active. There is no need to wait for bugreports from users.
- most important: during urgent troubleshooting on PRD (AKA panic-mode)
 - heisenbugs, difficult to reproduce.

3 Goal of libranet-logging

Make it as easy as possible to enable and properly use the full power of the python logging-framework

python logging-module contains:

- loggers, hierarchical
- handlers
 - formatters
 - filters

Think of logger=message-channel, handler=subscriber to a channel

Minimize the need to make changes in code

Move all config out of code and into a config-file “logging.yml”

- logging to a file should be as simple as:

```
```python
>>> import logging
>>> logging.getLogger('panicmode')
```
```

4 Features

- load logging-configuration from a yaml-config
- validate yaml-file for missing keys, invalid values
- configurable via env-variables
 - sane defaults if env-var is not set
- when logging to console, have colorized logging,
 - but nowhere else
 - configurable colors (avoid blue on black)
- integrate python-warnings
- add sample email-logger
- add sample syslog-logger
- avoid empty files that will remain empty
 - cleanup dedicated file-handlers based on root-loglevel
- future ideas:
 - integrate with kibana
 - log as json, structlog
 - * <https://logmatic.io/blog/python-logging-with-json-steroids/>
 - * <https://medium.com/@sanchitsokhey/centralised-logging-for-django-gunicorn-and-celery-using-elk-stack-76b13c54414c>
 - * <https://logmatic.io/blog/beyond-application-monitoring-discover-logging-best-practices/>
- in code throw out all
 - formatting,

- handler-config,
- setting loglevel
- debug-flags like::

```
>>> if DEBUG:
>>>     log.debug(...)
```

5 Usage

5.1 In your deployment

You can use following env-variables:

- PYTHON_LOG_CONFIG, path to logging.yml, e.g /opt/miniconda/envs/libranet/etc/logging.yml
- PYTHON_LOG_DIR, path to log-directory where logfiles will be created, /var/tmp/python
- PYTHON_ENABLE_LOGGING_TREE 1|0

optional env-vars:

- LOGLEVEL_ROOT
- LOGLEVEL_libranet_logging
- LOG_HANDLERS="console|debug_file|info_file|warning_file|error_file"

If missing, these default to DEBUG

5.2 In your code

To initialize the logging-infrastructure and set-up all configured loggers, handlers, formatters and filters:

```
> import libranet_logging
> libranet_logging.initialize()
2018/06/01 10:00:00 - root - DEBUG    - Logging configured from <path-to>/logging.yml
```

You do this once in your application, in the function that starts execution, not at the top of your module.

```
# Calling getLogger without arguments returns the root-logger,
# of which all other loggers descend.
# Normally you do NOT need this logger.
> import logging
> root_log = logging.getLogger()

# You normally use the module-logger
> log = logging.getLogger(__name__)

# and starting using it
> log.debug('This is debugging-information.')
> log.info('This is useful information.')
> log.warning('This is a warning.')
> log.error('This is a warning.')

# You can log a full-traceback by providing the exception to log.exception().
> try:
```

(continues on next page)

(continued from previous page)

```
> import foo
> except ImportError as e:
>     log.exception(e)
```

6 Unittesting

6.1 Introduction

Our python-package `libranet_logging` comes with a series of tests to check the validity of our code. Some are unittests, others are full integration-tests. The distinction between the several types of tests is of lesser importance.

However we try to minimize the amount of *mocking* and/or *patching*. The test-framework we use is `pytest`.

- <https://docs.pytest.org/en/latest/>
- <https://docs.pytest.org/en/latest/goodpractices.html#test-discovery>
- <https://pytest-cov.readthedocs.io/en/latest/>

Pytest is an extensible framework, and has a whole series of addons and plugins available on [PyPi](#).

You just can pip-install these packages:

```
<env-dir>/bin/pip install pytest pytest-cov pytest-flask pytest-mccabe pytest-mock
```

Note: Since the tests are importing the code of `libranet_logging`, this package needs be installed in a virtualenv, together with all its dependencies. See installation.

6.2 Re-usable test-fixtures

For our package, we have several application-specific test-fixtures in `tests/conftest.py`:

```
# pylint: disable=missing-function-docstring
"""conftest.py - custom pytest-plugins.

For more information about conftest.py, please see:

- https://docs.pytest.org/en/latest/writing_plugins.html
- https://pytest-flask.readthedocs.io/en/latest/tutorial.html

"""
import os
import pathlib

import pytest

@pytest.fixture(scope="session")
def tests_dir():
    tests_dir_ = os.path.dirname(os.path.realpath(__file__))
    return pathlib.Path(tests_dir_)

# @pytest.fixture(scope="session")
```

(continues on next page)

(continued from previous page)

```
# def pkg_dir():
#     pkg_dir_ = pkg_resources.resource_filename("libranet_logging", "")

#     return pl.Path(pkg_resources.files("libranet_logging") / "etc/logging.yml")
#     return pathlib.Path(pkg_dir_)

@pytest.fixture(scope="function")
def env(tmpdir):
    var_dir = tmpdir.mkdir("var")
    var_log_dir = var_dir.mkdir("log")
    # var_run_dir = var_dir.mkdir('run')
    # var_tmp_dir = var_dir.mkdir('tmp')

    # override any existing env-variable
    os.environ["PYTHON_LOG_DIR"] = str(var_log_dir)
    return tmpdir
```

6.3 Run tests

Run pytest via:

```
> cd <env-dir>
> bin/pytest src/libranet_logging/

> cd <env-dir>src/libranet_logging
> pytest

(libranet) wouter@midgard: <env-dir>src/libranet_logging > make py

platform linux -- Python 3.5.5, pytest-3.5.0, py-1.5.3, pluggy-0.6.0
rootdir: /opt/miniconda/envs/libranet/src/libranet_logging, inifile: setup.cfg
plugins: xdist-1.22.2, tap-2.2, forked-0.2, flask-0.10.0, flake8-1.0.0, cov-2.5.1,
celery-4.0.2
collected 15 items

===== test session starts =====
tests/test_cli.py .
tests/test_filters.py ....
tests/test_logconfig.py .....
tests/test_loglevel.py ..
===== 15 passed in 0.33 seconds =====
```

6.4 Run tests with code-coverage

```
> pytest --cov=libranet_logging --cov-report html .
```

7 Changelog

7.1 1.4 (unreleased)

- Remove compatibility with PyYaml < 5.1.
- Remove click as dependency.
- Remove libranet-print-logging-tree-executable.
- Remove dependency on distutils by copying over function `distutils.utils.str2bool`.
- Add `.readthedocs.yaml`.
- Modernize setup, switch to `pyproject.toml` and remove old-style `setup.py` and `setup.cfg`.
- Remove no longer needed `MANIFEST.in`.
- Remove `etc/pip-requirements`, replaced by `poetry.lock`.
- Move `.pylintrc` to `pyproject.toml`
- Update all dependencies.
- Add `pyroma` as dev-dependency.

7.2 1.3 (2023-01-24)

- No longer try to be smart about interactive mode or not. You can set the active logging-handlers by setting the env-var `LOG_HANDLERS``
- Remove `disable_console` as input-parameter to `libranet_logging.initialize()`.
- Change default separator from `;` to `|`. Use set via env-var `LOG_HANDLERS`.

7.3 1.2 (2021-06-06)

- Fix logo. [WVH]
- Add `.gitlab-ci.yml` [WVH]

7.4 1.1 (2020-02-13)

- Fix error `ModuleNotFoundError: No module named 'libranet_logging.version'`. [WVH]

7.5 1.0 (2020-02-12)

- Move `__version__`-attribute to `__init__`. [WVH]
- Package `libranet_logging` forked from WVH's unreleased package. [WVH]

7.6 0.5 (2019-08-19)

- Add docstrings and type-hinting.
- Fix a series of issues reported by pylint.
- Change function-signature of `libranet_logging.yaml.read_yaml`: change `vars` into `variables` to avoid shadowing the builtin `vars()`-function:

```
>>> config = read_yaml(path, vars=None)
>>> config = read_yaml(path, variables=None)
```

7.7 0.4 (2019-07-31)

- Add `version.py` with a `__version__`-attribute, rework version-management.
- In `setup.py` set minimum-requirements for `cerberus` `>=1.3.1`. [WVH]

7.8 0.3 (2019-05-28)

- In `sphinx-docs`, add link to coverage-report on https://example.com/docs/libranet_logging-coverage [WVH]
- In `Makefile` add `step`` copy-cov``` to copy coverage-report to apache-webdirectory. [WVH]
- In `libranet_logging.yaml.read_yaml` cdefault `vars` to empty dicts when not provided. [WVH]
- In `libranet_logging.logconfig.logging_schema` rename `valueschema` to `valuesrules` to avoid `DeprecationWarnings`. [WVH]

7.9 0.2 (2019-03-28)

- Make `libranet_logging.yaml.read_yaml` compatible with `PyYAML 5.1`, but keep backwards-compatibility with older versions. [WVH] Cfr:

```
- https://github.com/yaml/pyyaml/blob/master/CHANGES
- https://github.com/yaml/pyyaml/pull/257
- https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load\(input\)-Deprecation
```

7.10 0.1 (2019-03-28)

- Add support for simple string-formatting in the `login.yml`. [WVH]
- Use `isort` to manage the imports. Add `isort-config` to `setup.cfg`. [WVH]
- Introduce environment-variable `PYTHON_CONSOLE_FORMATTER` to select which console-formatter to use. [WVH]
- Rename `colored-console-formatter` into `console_color` and add `console_bw`-formatter for simple black & white logging in the console. [WVH]
- Add `flask_wtf`-handler in default `logging.yml`. [WVH]
- If the `log-directory` does not yet exist, we now create it. [WVH]



- We should have a user-specific default location to avoid interference between users. The log-directory will be first taken from the direct function-parameters, then from the `logging.yml` if present. If not present, from the env-var `PYTHON_LOG_CONFIG`, and in case of no env-var we default to `$HOME/logs` instead of `var/tmp/python`. [WVH]
 - Support setting the log-directory via the `initialize`-function. [WVH]
 - Fix failing test `test_initialize_without_logging_tree`. It was failing when the env-var `PYTHON_ENABLE_LOGGING_TREE` was not set. [WVH]
 - Fix failing test of the click-command `cli.print_logging_tree`. [WVH]
 - Add new testing-dependency `pytest-click`. [WVH]
 - Convert `cli.print_logging_tree` to a click-command, accepting an optional path-argument. If the environment-variable `PYTHON_LOG_CONFIG` is set, we use that value as the path-default. [WVH]
 - Add `click` as a new dependency. [WVH]
 - Add documentation about unittesting. [WVH]
 - We now support arrays in environment-variables. Environment-variables containing a `;` are now converted to a list similar to the default value if that env-variable was not set. [WVH]
 - Fix `filters.RegexFilter` to use `search()` instead of `match()`. Cfr. <https://docs.python.org/3/library/re.html#search-vs-match> [WVH]
 - Add passing unit-tests. [WVH]
 - In `initialize()` allow Path-parameters as input instead of only string-paths. [WVH]
 - Generally make the code robust in case of loading a `logging.yml` with schema-errors. [WVH]
 - Add console-entrypoint `libranet-logging-print-logging-tree` to initialize the logging and print the logging-tree to the standard output. Add corresponding function in new `libranet_logging.cli`-module. [WVH]
 - Add function-parameter `use_print=False` to `logconfig.show_logging_tree` to enable printing to standard output instead of logging to the configured loggers. [WVH]
 - If we call `initialize()` without providing a path of setting the environment-variable `PYTHON_LOG_CONFIG`, we now use the default `logging.yml` shipped with this `libranet_logging`-package. [WVH]
 - Add logger `libranet_logging` to our default `logging.yml`. [WVH]
 - Instantiate the correct logger using `__name__` instead of logging to the root-logger. [WVH]
 - Add `recommonmark` and update `docs.conf.py` to allow markdown in docs. Cfr. <https://recommonmark.readthedocs.io/en/latest/> [WVH]
 - In `setup.py` and `docs/pip-requirements` add sphinx-related dependencies. [WVH]
 - Simplify public api:
 - Rename function `loglevel.create_loglevel` into `loglevel.create`.
 - Rename function `logconfig.initialize_logging` into `logconfig.initialize`.
- [WVH]
- Run `Black` on the code. `Black` is a code-formatter for Python. Cfr. <https://github.com/ambv/black> [WVH]
 - Add some basic Sphinx-based documentation. [WVH]
 - Factor out creating new loglevels into its own `loglevel`-module. [WVH]
 - Factor out logging-filters into its own `filters`-module. [WVH]
 - Add third-party dependency `colorlog`. This is not a code-dependency but rather a dependency of `logging.yml`. [WVH]
 - Add third-party dependencies `cerberus`, `logging_tree` and `PyYAML`. [WVH]

- Move logging-related code from `libdl.utils` into its own `libranet_logging`-package. [WVH]
- Package created via `cookiecutter templates/cookiecutter-libranet-python-package`. [Wouter Vanden Hove <wouter@wvhconsulting.org>]

8 Security Policy

8.1 Supported Versions

Use this section to tell people about which versions of your project are currently being supported with security updates.

| Version | Supported ||  |  || 0.x | :white_check_mark: || 1.0.x | :white_check_mark: |

8.2 Reporting a Vulnerability

This project follows a 90 day disclosure timeline.

To report a security issue, please an email security@libranet.eu with

- a description of the issue
- the steps you took to create the issue,
- affected versions
- and if known, mitigations for the issue

Our team will acknowledge receiving your email within 3 working days.

9 MIT License

Copyright (c) 2023 [Libranet.eu](https://libranet.eu).

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

10 Contributors

Special thanks for all the people who have helped on this project so far.

Append your name if you have contributed to this package. We use anti-chronological ordering (oldest on top).

- [Wouter Vanden Hove](#) <wouter@libranet.eu>

11 How to contribute

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change.

Please note we have a *code of conduct*, please follow it in all your interactions with the project.

11.1 Development environment setup

Proceed to describe how to setup local development environment. e.g:

To set up a development environment, please follow these steps:

1. Clone the repo

```
git clone https://github.com/libranet/httpclient-logging
```

2. Run make install

```
make install
```

11.2 Issues and feature requests

You've found a bug in the source code, a mistake in the documentation or maybe you'd like a new feature? Take a look at [GitHub Discussions](#) to see if it's already being discussed.

You can help us by [submitting an issue on GitHub](#). Before you create an issue, make sure to search the issue archive – your issue may have already been addressed.

Please try to create bug reports that are:

- *Reproducible*. Include steps to reproduce the problem.
- *Specific*. Include as much detail as possible: which version, what environment, etc.
- *Unique*. Do not duplicate existing opened issues.
- *Scoped to a Single Bug*. One bug per report.

Even better: Submit a pull request with a fix or new feature!

How to submit a Pull Request

1. Search our repository for open or closed [Pull Requests](#) that relate to your submission. You don't want to duplicate effort.
2. Fork the project
3. Create your feature branch (`git checkout -b feat/amazing_feature`)
4. Commit your changes (`git commit -m 'feat: add amazing_feature'`) `httpclient`-logging uses [conventional commits](#), so please follow the specification in your commit messages.
5. Push to the branch (`git push origin feat/amazing_feature`)
6. [Open a Pull Request](#)

12 Contributor Covenant Code of Conduct

12.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

12.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

12.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

12.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

12.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at security@libranet.eu. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

12.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

12.7 Attribution

This Code of Conduct is adapted from the [Contributor Covenant](https://www.contributor-covenant.org/version/2/0/code_of_conduct.html), version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by [Mozilla's code of conduct enforcement ladder](#).

For answers to common questions about this code of conduct, see the FAQ at <https://www.contributor-covenant.org/faq>. Translations are available at <https://www.contributor-covenant.org/translations>.

13 libranet_logging

libranet_logging.__init__

13.1 Submodules

libranet_logging.cli

libranet_logging.cli

Module Contents

Functions

| | |
|------------------------------|----------|
| <code>print_tree()</code> | Returns: |
| <code>print_loggers()</code> | Returns: |

libranet_logging.cli.**print_tree()**

Returns:

Return type

None

libranet_logging.cli.**print_loggers()**

Returns:

Return type

None

libranet_logging.filters

libranet_logging.filters.

Module Contents

Classes

| | |
|---------------------------|--|
| <i>SimpleStringFilter</i> | SimpleStringFilter is a logging-filter based in simple string occurence in the logmessage. |
| <i>RegexFilter</i> | RegexFilter is a logging-filter based in regular expressions. |

class libranet_logging.filters.**SimpleStringFilter**(*name=""*, *params=None*)

Bases: logging.Filter

SimpleStringFilter is a logging-filter based in simple string occurence in the logmessage.

filter(*record*)

Determine if the specified record is to be logged.

Parameters

record –

Return type

bool

Returns:

class libranet_logging.filters.**RegexFilter**(*name=""*, *params=None*)

Bases: logging.Filter

RegexFilter is a logging-filter based in regular expressions.

filter(*record*)

Determine if the specified record is to be logged.

Parameters

record –

Return type

bool

Returns:

libranet_logging.logconfig

libranet_logging.logconfig.

Module Contents

Functions

| | |
|--|--|
| <code>is_interactive_shell()</code> | Decide if this process is run in an interactive shell or not. |
| <code>get_sorted_lognames()</code> | Returns: |
| <code>remove_console(config[, disable_console])</code> | param config |
| <code>ensure_dir(directory)</code> | param directory |
| <code>convert_filenames(config[, logdir])</code> | "Convert all relative filenames in the handlers to absolute paths. |
| <code>remove_lower_level_handlers(config)</code> | Remove lower-level handlers from dedicated-level loggers. |
| <code>validate_logging(log_config, path)</code> | Validate the syntax of a logging.yml-file. |
| <code>strtobool(val)</code> | Convert a string representation of truth to true (1) or false (0). |
| <code>output_logging_tree([use_print])</code> | param use_print |
| <code>get_default_logging_yaml()</code> | Returns the path to the default logging configuration file. |
| <code>initialize([path, logdir, capture_warnings, silent, ...])</code> | Initialize logging configuration with a yaml-file. |

Attributes

| |
|-----------------------------|
| <code>cerberus</code> |
| <code>log</code> |
| <code>logging_schema</code> |

`libranet_logging.logconfig.cerberus`

`libranet_logging.logconfig.log`

`libranet_logging.logconfig.logging_schema`

exception `libranet_logging.logconfig.CerberusValidationError`

Bases: Exception

CerberusValidationError-class.

`libranet_logging.logconfig.is_interactive_shell()`

Decide if this process is run in an interactive shell or not.

If environment-variable \$TERM is present, we are running this code in a interactive shell, else we are run from cron or called via nrpe as a nagios-check.

Returns: boolean

`libranet_logging.logconfig.get_sorted_lognames()`

Returns:

`libranet_logging.logconfig.remove_console(config, disable_console=False)`

Parameters

- **config** –
- **disable_console** –

Returns:

`libranet_logging.logconfig.ensure_dir(directory)`

Parameters

directory –

Returns:

`libranet_logging.logconfig.convert_filenames(config, logdir="")`

“Convert all relative filenames in the handlers to absolute paths.

Parameters

- **config** –
- **logdir** –

Returns:

`libranet_logging.logconfig.remove_lower_level_handlers(config)`

Remove lower-level handlers from dedicated-level loggers.

We have dedicated file-handlers for each logging-level

- `debug_file_handler`
- `info_file_handler`
- `warning_file_handler`
- `error_file_handler`

If the root-level is set higher, we remove the lower-level handlers This avoids creating logfiles that will always remain empty.

`libranet_logging.logconfig.validate_logging(log_config, path)`

Validate the syntax of a logging.yml-file.

`libranet_logging.logconfig.strptime(val)`

Convert a string representation of truth to true (1) or false (0).

True values are ‘y’, ‘yes’, ‘t’, ‘true’, ‘on’, and ‘1’; false values are ‘n’, ‘no’, ‘f’, ‘false’, ‘off’, and ‘0’. Raises `ValueError` if ‘val’ is anything else.

`libranet_logging.logconfig.output_logging_tree(use_print=False)`

Parameters

use_print –

Returns:

`libranet_logging.logconfig.get_default_logging_yaml()`

Returns the path to the default logging configuration file.

Returns

A *Path* object representing the path to the default logging configuration file.

Return type

`pathlib.Path`

```
libranet_logging.logconfig.initialize(path="", logdir="", capture_warnings=True, silent=False,
                                     use_print=False, variables=None)
```

Initialize logging configuration with a yaml-file.

Parameters

- **path** –
- **logdir** –
- **capture_warnings** –
- **silent** –
- **use_print** –
- **variables** –

Returns:

libranet_logging.loglevel

libranet_logging.loglevel.

Module Contents

Functions

| | |
|---|---------------------------|
| <code>create_loglevel([level_name, level_num])</code> | Create a custom loglevel. |
|---|---------------------------|

Attributes

| |
|------------------------------------|
| <code>create_loglevel_trace</code> |
|------------------------------------|

`libranet_logging.loglevel.create_loglevel(level_name="", level_num=0)`

Create a custom loglevel.

Defining your own levels is possible, but should not be necessary, as the existing levels have been chosen on the basis of practical experience. However, if you are convinced that you need custom levels, great care should be exercised when doing this, and it is possibly a very bad idea to define custom levels if you are developing a library. That's because if multiple library authors all define their own custom levels, there is a chance that the logging output from such multiple libraries used together will be difficult for the using developer to control and/or interpret, because a given numeric value might mean different things for different libraries. Cfr. <https://docs.python.org/3/howto/logging.html#custom-levels>

Default levels:

0 NOTSET 10 DEBUG 20 INFO 30 WARNING 40 ERROR 50 CRITICAL

Parameters

- **level_name** – logger-name
- **level_num** – numeric level of the custom logger, positive integer

Returns

None

Side-effect:

adds attribute to Logger-class

`libranet_logging.loglevel.create_loglevel_trace`

libranet_logging.yaml

`libranet_logging.yaml`.

In pyyaml 5.1 some incompatibilities were introduced with regard to `yaml.load` to make it more safe by default.

please see:

- <https://github.com/yaml/pyyaml/blob/master/CHANGES>
- <https://github.com/yaml/pyyaml/pull/257>
- [https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load\(input\)-Deprecation](https://github.com/yaml/pyyaml/wiki/PyYAML-yaml.load(input)-Deprecation)

Module Contents**Functions**

| | |
|--|---|
| <code>constructor_env(loader, node)</code> | YAML-Constructor to load a value from a env-variable. |
| <code>add_constructor()</code> | |
| <code>read_yaml(path[, variables])</code> | Read the yaml-file. |

`libranet_logging.yaml.constructor_env(loader, node)`

YAML-Constructor to load a value from a env-variable.

Usage in yaml:

> !env ENVVAR_NAME, DEFAULTVALUE_IF_ENVVAR_NOT_SET

`libranet_logging.yaml.add_constructor()`

`libranet_logging.yaml.read_yaml(path, variables=None)`

Read the yaml-file.

Returns: dict

13.2 Package Contents**Functions**

| | |
|--|--|
| <code>print_loggers()</code> | Returns: |
| <code>print_tree()</code> | Returns: |
| <code>initialize([path, logdir, capture_warnings, silent, ...])</code> | Initialize logging configuration with a yaml-file. |
| <code>output_logging_tree([use_print])</code> | param use_print |
| <code>create_loglevel([level_name, level_num])</code> | Create a custom loglevel. |

Attributes

`__version__`

`create_loglevel_trace`

```
libranet_logging.__version__ = '1.4.dev0'
```

```
libranet_logging.print_loggers()
```

Returns:

Return type

None

```
libranet_logging.print_tree()
```

Returns:

Return type

None

```
libranet_logging.initialize(path="", logdir="", capture_warnings=True, silent=False, use_print=False,
                           variables=None)
```

Initialize logging configuration with a yaml-file.

Parameters

- **path** –
- **logdir** –
- **capture_warnings** –
- **silent** –
- **use_print** –
- **variables** –

Returns:

```
libranet_logging.output_logging_tree(use_print=False)
```

Parameters

use_print –

Returns:

```
libranet_logging.create_loglevel(level_name="", level_num=0)
```

Create a custom loglevel.

Defining your own levels is possible, but should not be necessary, as the existing levels have been chosen on the basis of practical experience. However, if you are convinced that you need custom levels, great care should be exercised when doing this, and it is possibly a very bad idea to define custom levels if you are developing a library. That's because if multiple library authors all define their own custom levels, there is a chance that the logging output from such multiple libraries used together will be difficult for the using developer to control and/or interpret, because a given numeric value might mean different things for different libraries. Cfr. <https://docs.python.org/3/howto/logging.html#custom-levels>

Default levels:

0 NOTSET 10 DEBUG 20 INFO 30 WARNING 40 ERROR 50 CRITICAL

Parameters

- **level_name** – logger-name

- **level_num** – numeric level of the custom logger, positive integer

Returns

None

Side-effect:

adds attribute to Logger-class

`libranet_logging.create_loglevel_trace`

14 Indices and tables

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

Python Module Index

|

`libranet_logging`, [14](#)
`libranet_logging.cli`, [14](#)
`libranet_logging.filters`, [15](#)
`libranet_logging.logconfig`, [15](#)
`libranet_logging.loglevel`, [18](#)
`libranet_logging.yaml`, [19](#)

Index

Symbols

`__version__` (in module `libranet_logging`), 20

A

`add_constructor()` (in module `libranet_logging.yaml`), 19

C

`cerberus` (in module `libranet_logging.logconfig`), 16

`CerberusValidationError`, 16

`constructor_env()` (in module `libranet_logging.yaml`), 19

`convert_filenames()` (in module `libranet_logging.logconfig`), 17

`create_loglevel()` (in module `libranet_logging`), 20

`create_loglevel()` (in module `libranet_logging.loglevel`), 18

`create_loglevel_trace` (in module `libranet_logging`), 21

`create_loglevel_trace` (in module `libranet_logging.loglevel`), 19

E

`ensure_dir()` (in module `libranet_logging.logconfig`), 17

F

`filter()` (`libranet_logging.filters.RegexFilter` method), 15

`filter()` (`libranet_logging.filters.SimpleStringFilter` method), 15

G

`get_default_logging_yaml()` (in module `libranet_logging.logconfig`), 17

`get_sorted_lognames()` (in module `libranet_logging.logconfig`), 16

I

`initialize()` (in module `libranet_logging`), 20

`initialize()` (in module `libranet_logging.logconfig`), 18

`is_interactive_shell()` (in module `libranet_logging.logconfig`), 16

L

`libranet_logging`
module, 14

`libranet_logging.cli`
module, 14

`libranet_logging.filters`
module, 15

`libranet_logging.logconfig`

module, 15

`libranet_logging.loglevel`
module, 18

`libranet_logging.yaml`
module, 19

`log` (in module `libranet_logging.logconfig`), 16

`logging_schema` (in module `libranet_logging.logconfig`), 16

M

module

`libranet_logging`, 14

`libranet_logging.cli`, 14

`libranet_logging.filters`, 15

`libranet_logging.logconfig`, 15

`libranet_logging.loglevel`, 18

`libranet_logging.yaml`, 19

O

`output_logging_tree()` (in module `libranet_logging`), 20

`output_logging_tree()` (in module `libranet_logging.logconfig`), 17

P

`print_loggers()` (in module `libranet_logging`), 20

`print_loggers()` (in module `libranet_logging.cli`), 14

`print_tree()` (in module `libranet_logging`), 20

`print_tree()` (in module `libranet_logging.cli`), 14

R

`read_yaml()` (in module `libranet_logging.yaml`), 19

`RegexFilter` (class in `libranet_logging.filters`), 15

`remove_console()` (in module `libranet_logging.logconfig`), 17

`remove_lower_level_handlers()` (in module `libranet_logging.logconfig`), 17

S

`SimpleStringFilter` (class in `libranet_logging.filters`), 15

`strtobool()` (in module `libranet_logging.logconfig`), 17

V

`validate_logging()` (in module `libranet_logging.logconfig`), 17