developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Sep 20, 2021

HOME

Documentation Status

This project allows standard logging configuration across all SKA projects. The format used is described in detail on the developer portal.

This library should be used to configure the application's logging system as early as possible at start up. Note that for Python TANGO devices that inherit from SKABaseDevice, this is already done in the base class, so it does not need to be done again.

CHAPTER

ONE

USAGE

For Python applications, this is as simple as:

```
import logging
from ska_ser_logging import configure_logging

def main():
    configure_logging()
    logger = logging.getLogger("ska.example")
    logger.info("Logging started for Example application")
```

The `configure_logging` function takes additional arguments, including one that allows the initial log level to be specified. It may be useful to link that to a command line option or environment variable.

```
import logging
from ska_ser_logging import configure_logging

def main():
    configure_logging(logging.DEBUG)
```

SKA's logging format allows for simple tags (key-value pairs) to be included in log messages. Application-specific behaviour can be provided via a filter, which will be added to all handlers in the configuration used by `configure_logging`. If this filter is provided, it must add a `tags` attribute to each record, as the log message formatting string will include a `%(tags)s` field.

Note that the logging format limits the characters allowed in tags. For example, `|` is not allowed. No validation is done by this library. If the filter is `None` (the default), then the field will be omitted.

```
import logging
from ska_ser_logging import configure_logging


class TangoDeviceTagsFilter(logging.Filter):
    def filter(self, record):
        record.tags = "tango-device:my/tango/device"
        return True

def main():
    configure_logging(level="DEBUG", tags_filter=TangoDeviceTagsFilter)
```

In the more general case, the configuration can be updated with an additional dict, matching the `logging.config.dictConfig` schema. This additional dict is recursively merged with the default configuration. While not recommended, keys in the default configuration can be removed by setting the corresponding override key's value to `None`. In

the example below, we add output to a file. Note that the "default" formatter and "console" handler are provided by the default configuration.

```
import logging.handlers
from ska_ser_logging import configure_logging


ADDITIONAL_LOGGING_CONFIG = {
    "handlers": {
        "file": {
            "()" : logging.handlers.RotatingFileHandler,
            "formatter": "default",
            "filename": "./ska.example.log",
            "maxBytes": 2048,
            "backupCount": 2,
        }
    },
    "root": {
        "handlers": ["console", "file"],
    }
}

def main():
    configure_logging(overrides=ADDITIONAL_LOGGING_CONFIG)
```

Custom handlers that use the standard logging format may be useful. In this case, the function get_default_formatter is available. The example below is contrived, but shows the approach. A more practical use case is adding and removing handlers at runtime.

```
import logging
import logging.handlers
from ska_ser_logging import configure_logging, get_default_formatter


def main():
    configure_logging()
    logger = logging.getLogger("ska.example")
    handler = logging.handlers.MemoryHandler(capacity=10)
    handler.setFormatter(get_default_formatter())
    logger.addHandler(handler)
    logger.info("Logging started for Example application")
```

By default, calls to configure_logging do not disable existing non-root loggers. This allows multiple calls to the function, although that will generally not be required. This behaviour can be overridden using the "disable_existing_loggers" key.

CHAPTER

TWO

REQUIREMENTS

The system used for development needs to have Python 3 and `pip` installed.

CHAPTER

THREE

INSTALL

Always use a virtual environment. Pipenv is now Python's officially recommended method and the one used by default in this repo.

Follow these steps at the project root:

First, ensure that `~/.local/bin` is in your `PATH` with:

```
> echo $PATH
```

In case `~/.local/bin` is not part of your `PATH` variable, under Linux add it with:

```
> export PATH=~/.local/bin:$PATH
```

or the equivalent in your particular OS.

Then proceed to install pipenv and the required environment packages:

```
> pip install pipenv # if you don't have pipenv already installed on your system
> pipenv install
> pipenv shell
```

You will now be inside a pipenv shell with your virtual environment ready.

Use `exit` to exit the pipenv environment.

CHAPTER

FOUR

TESTING

- Put tests into the `tests` folder
- Use PyTest as the testing framework
  - Reference: PyTest introduction
- Run tests with `python3 setup.py test` or just `make test`
  - Configure PyTest in `setup.py` and `setup.cfg`
- Running the test creates the `htmlcov` folder
  - Inside this folder a rundown of the issues found will be accessible using the `index.html` file
- All the tests should pass before merging the code

CHAPTER

FIVE

CODE ANALYSIS

- Use Pylint as the code analysis framework
- By default it uses the PEP8 style guide
- Code analysis should be run by calling `make lint`. All pertaining options reside under the `.pylintrc` file.
- Code analysis should only raise document related warnings (i.e. `#FIXME` comments) before merging the code

CHAPTER

SIX

WRITING DOCUMENTATION

- The documentation generator for this project is derived from SKA's SKA Developer Portal repository
- The documentation can be edited under `./docs/src`
- If you want to include only your README.md file, create a symbolic link inside the `./docs/src` directory if the existing one does not work:

```
$ cd docs/src
$ ln -s ../../README.md README.md
```

- In order to build the documentation for this specific project, execute the following under `./docs`:

```
$ make html
```

- The documentation can then be consulted by opening the file `./docs/build/html/index.html`

Todo:

- Insert todo's here

CHAPTER

SEVEN

API DOCUMENTATION

This section details the public API for configuring application logging across the SKA project.

7.1 Python

The API for the configuration using Python is shown below.

7.1.1 Public API Documentation

Functions

Module init code.

ska_ser_logging.**configure_logging**(*level=None*, *tags_filter=None*, *overrides=None*)

Configure Python logging for SKA applications.

This function should be used to configure the application's logging system as early as possible at start up.

Note: For Python TANGO devices that inherit from *lmcbaseclasses.SKABaseDevice* this is already done in that base class, so it does not need to be done again.

Example usage is shown in this repo's `README.md` file.

Parameters

- **level** (*str or int, optional*) – Set the logging level to this instead of the default. Use the string representations like ""INFO" and "DEBUG", or integer values like *logging.INFO* and *logging.DEBUG*.
- **tags_filter** (type derived from *logging.Filter*, optional) – If this type (not instance) is provided, the default formatter will include a "%(tags)s" specifier. The filter must inject the *tags* attribute in each record.
- **overrides** (*dict, optional*) – The default logging configuration can be modified by passing in this dict. It will be recursively merged with the default. This allows existing values to be modified, or even removed. It also allows additional loggers, handlers, filters and formatters to be added. See the *_override* function for more details on the merging process. The end result must be compatible with the *logging.config.dictConfig* schema. Note that the *level* and *tags_filter* parameter are applied after merging the overrides.

ska_ser_logging.**get_default_formatter**(*tags=False*)

Return a formatter configured with the standard logging format.

Parameters **tags** (*bool, optional*) – If true, then include the "tags" field in the format string. This requires a tags filter to be linked to the corresponding handler.

Returns A new default formatter.

Return type logging.Formatter

CHAPTER

EIGHT

SKA LOGGING CONFIGURATION DOCUMENTATION

These are all the packages, functions and scripts that form part of the project.

- *API documentation*

PYTHON MODULE INDEX

S

INDEX