
confiddler Documentation

Release 0.1

Stuart Berg

Jul 02, 2020

Contents

1	Install	3
2	Contents	5
2.1	Quickstart	5
2.2	API Reference	6
	Index	9

confiddler has tools to help you define and load YAML configuration files.

Here's the basic idea:

1. Define your config file structure with a JSON schema. (See json-schema.org or [jsonschema](https://github.com/epoberezkin/jsonschema)).
2. Help your users get started by showing them a **default config file** which is:
 - auto-generated from your defaults, and
 - auto-commented with your schema description.
3. Load a user's config file with `confiddler.load_config()`, which will:
 - validate it against your schema
 - auto-inject default values for any settings the user omitted

See the [Quickstart](#) for a short example.

CHAPTER 1

Install

Install from PyPI:

```
pip install confiddler
```

...or with conda:

```
conda install -c stuarteberg -c conda-forge confiddler
```


2.1 Quickstart

Define your schema

```
>>> from confidder import dump_default_config, load_config
>>> schema = {
    "description": "Settings for a robot vacuum cleaner",
    "properties": {
        "speed": {
            "description": "Speed of the robot (1-10)",
            "type": "number",
            "minValue": 1,
            "maxValue": 10,
            "default": 1
        },
        "movement": {
            "description": "Movement strategy: random, raster, or spiral",
            "type": "string",
            "enum": ["random", "raster", "spiral"],
            "default": "random"
        }
    }
}
```

Show your user the default config.

```
>>> dump_default_config(schema, sys.stdout, 'yaml')
speed: 1
movement: random

>>> dump_default_config(schema, sys.stdout, 'yaml-with-comments')
#
```

(continues on next page)

(continued from previous page)

```
# Settings for a robot vacuum cleaner
#
# Speed of the robot (1-10)
speed: 1
# Movement strategy: random, raster, or spiral
movement: random
```

Load your user's config.

```
# my-robot-config.yaml
speed: 2
```

```
>>> load_config('my-robot-config.yaml', schema, inject_defaults=True)
{'speed': 2, 'movement': 'random'}
```

2.2 API Reference

`confiddler.load_config(path_or_file, schema={}, inject_defaults=True)`

Convenience wrapper around `validate()`. (This function accepts a file).

Load the config data from the given file (or path to a file), and validate it against the given schema.

All missing values will be inserted from schema defaults. If a setting is missing and the schema contains no default value for it, a `ValidationError` is raised.

Note: If your config data is already loaded into a dict and you just want to validate it and/or inject defaults, see `validate()`.

Parameters

- **path_or_file** – The raw config data. Either a file object or a file path.
- **schema** – The config schema, already loaded into a Python dict.

Returns dict

`confiddler.dump_default_config(schema, f=None, format='yaml')`

Convenience wrapper around `emit_defaults()`. (This function writes to a file).

Dump the default config settings from the given schema. Settings without default values will use `"{{NO_DEFAULT}}"` as a placeholder.

Parameters

- **schema** – The config schema
- **f** – File object to which default config data will be dumped. If `None`, then the default config is returned as a string.

format: Either `"json"`, `"yaml"`, or `"yaml-with-comments"`. The `"yaml-with-comments"` format inserts comments above each setting, populated with the setting's `"description"` field from the schema.

Returns None, unless no file was provided, in which case the default config is returned as a string.

`confiddler.validate(instance, schema, base_cls=None, *args, inject_defaults=False, **kwargs)`

Drop-in replacement for `jsonschema.validate()`, with the following extended functionality:

- Specifically allow types from `ruamel.yaml.comments`
- If `inject_defaults` is True, this function *modifies* the instance IN-PLACE to fill missing properties with their schema-provided default values.

See the [jsonschema FAQ](#) for details and caveats.

`confiddler.emit_defaults(schema, include_yaml_comments=False, yaml_indent=2, base_cls=None, *args, **kwargs)`

Emit all default values for the given schema.

Similar to calling `validate({}, schema, inject_defaults=True)`, except:

1. Ignore schema validation errors and ‘required’ property errors
2. If no default is given for a property, inject "`{{NO_DEFAULT}}`", even if the property isn’t supposed to be a string.
3. If `include_yaml_comments` is True, insert `CommentedMap` objects instead of ordinary dicts, and insert a comment above each key, with the contents of the property "`description`" in the schema.

Parameters

- **schema** – The schema data to pull defaults from
- **include_yaml_comments** – Whether or not to return `ruamel.yaml` objects so that comments will be written when the data is dumped to YAML.
- **yaml_indent** – To ensure correctly indented comments, you must specify the indent step you plan to use when this data is eventually dumped as yaml.

Returns A copy of instance, with default values injected, and comments if specified.

`confiddler.flow_style(ob)`

This function can be used to fine-tune the format of exported YAML configs. (It is only needed rarely.)

By default, `dump_default_config()` uses ‘block style’:

```
>>> schema = {
    "properties": {
        "names": {
            "default": ['a', 'b', 'c']
        }
    }
}

>>> dump_default_config(schema, sys.stdout)
names:
- a
- b
- c
```

But if you’d prefer for a particular value to be written with ‘flow style’, wrap it with `flow_style()`:

```
>>> from confiddler import flow_style
>>> schema = {
    "properties": {
        "names": {
            "default": flow_style(['a', 'b', 'c'])
        }
    }
}

>>> dump_default_config(schema, sys.stdout)
names: [a, b, c]
```

D

`dump_default_config()` (*in module confiddler*), 6

E

`emit_defaults()` (*in module confiddler*), 7

F

`flow_style()` (*in module confiddler*), 7

L

`load_config()` (*in module confiddler*), 6

V

`validate()` (*in module confiddler*), 7