i3pystatus Documentation

Release

Author

Dec 26, 2022

Contents

Contents:

CHAPTER 1

Installation

Supported Python Versions

i3pystatus requires Python 3.6 or newer and is not compatible with Python 2. Some modules require additional dependencies documented in the docs.

1.1 Into Virtualenv

i3pystatus no longer uses numbered releases. Therefore, the recommended method is to install from git via `pip`, and into a virtualenv to avoid polluting your site-packages directory.

First, create a virtualenv:

```
$ python3 -mvenv /path/to/virtualenv
```

Next, activate into the virtualenv and use pip to install i3pystatus into it:

```
$ source /path/to/virtualenv/bin/activate
$ pip install git+https://github.com/enkore/i3pystatus.git
```

If you are installing for development, use `pip install --editable` instead:

```
$ source /path/to/virtualenv/bin/activate
$ pip install --editable /path/to/clone/of/i3pystatus
```

NOTE: If you need to install any additional dependencies required by the i3pystatus modules you are using, you will also need to install them into this virtualenv.

1.2 Invoking From Virtualenv

To invoke i3pystatus from your virtualenv, use the `python` symlink from the virtualenv to run your i3pystatus config script. See the following example `bar` section from `~/.config/i3/config`:

```
bar {
    colors {
         statusline #949494
         separator #4e4e4e
    }
    separator_symbol "|"
```

```
    position top
    status_command /path/to/virtualenv/bin/python /home/username/.config/i3/status.py
}
```

CHAPTER 2

Configuration

The configuration file is a normal Python script. The status bar is controlled by a central *Status* object, which individual *modules* like a *clock* or a *battery* monitor are added to with the `register` method.

A typical configuration file could look like this (note the additional dependencies from *network* and *pulseaudio* in this example):

```
from i3pystatus import Status

status = Status()

# Displays clock like this:
# Tue 30 Jul 11:59:46 PM KW31
#                          ^-- calendar week
status.register("clock",
    format="%a %-d %b %X KW%V",)

# Shows the average load of the last minute and the last 5 minutes
# (the default value for format is used)
status.register("load")

# Shows your CPU temperature, if you have a Intel CPU
status.register("temp",
    format="{temp:.0f}°C",)

# The battery monitor has many formatting options, see README for details

# This would look like this, when discharging (or charging)
# ↓14.22W 56.15% [77.81%] 2h:41m
# And like this if full:
# =14.22W 100.0% [91.21%]
#
# This would also display a desktop notification (via D-Bus) if the percentage
# goes below 5 percent while discharging. The block will also color RED.
# If you don't have a desktop notification demon yet, take a look at dunst:
#   http://www.knopwob.org/dunst/
status.register("battery",
    format="{status}/{consumption:.2f}W {percentage:.2f}% [{percentage_design:.2f}%]
↪{remaining:%E%hh:%Mm}",
    alert=True,
    alert_percentage=5,
    status={
        "DIS": "↓",
        "CHR": "↑",
```

```
        "FULL": "=",
    },)

# This would look like this:
# Discharging 6h:51m
status.register("battery",
    format="{status} {remaining:%E%hh:%Mm}",
    alert=True,
    alert_percentage=5,
    status={
        "DIS":  "Discharging",
        "CHR":  "Charging",
        "FULL": "Bat full",
    },)

# Displays whether a DHCP client is running
status.register("runwatch",
    name="DHCP",
    path="/var/run/dhclient*.pid",)

# Shows the address and up/down state of eth0. If it is up the address is shown in
# green (the default value of color_up) and the CIDR-address is shown
# (i.e. 10.10.10.42/24).
# If it's down just the interface name (eth0) will be displayed in red
# (defaults of format_down and color_down)
#
# Note: the network module requires PyPI package netifaces
status.register("network",
    interface="eth0",
    format_up="{v4cidr}",)

# Note: requires both netifaces and basiciw (for essid and quality)
status.register("network",
    interface="wlan0",
    format_up="{essid} {quality:03.0f}%",)

# Shows disk usage of /
# Format:
# 42/128G [86G]
status.register("disk",
    path="/",
    format="{used}/{total}G [{avail}G]",)

# Shows pulseaudio default sink volume
#
# Note: requires libpulseaudio from PyPI
status.register("pulseaudio",
    format="♪{volume}",)

# Shows mpd status
# Format:
# Cloud connectedReroute to Remain
status.register("mpd",
    format="{title}{status}{album}",
    status={
        "pause": "",
        "play": "",
        "stop": "",
```

```
    },)

status.run()
```

Also change your i3wm config to the following:

```
# i3bar
bar {
    status_command    python ~/.path/to/your/config/file.py
    position          top
    workspace_buttons yes
}
```

Note: Don't name your config file `i3pystatus.py`, as it would make `i3pystatus` un-importable and lead to errors.

Another way to launch your configuration file is to use `i3pystatus` script from installation:

```
i3pystatus -c ~/.path/to/your/config/file.py
```

If no arguments were provided, `i3pystatus` script works as an example of `Clock` module.

2.1 Formatting

All modules let you specify the exact output formatting using a format string, which gives you a great deal of flexibility.

If a module gives you a float, it probably has a ton of uninteresting decimal places. Use `{somefloat:.0f}` to get the integer value, `{somefloat:0.2f}` gives you two decimal places after the decimal dot

2.1.1 formatp

Some modules use an extended format string syntax (the *mpd* and *weather* modules, for example). Given the format string below the output adapts itself to the available data.

```
[{artist}/{album}/]{title}{status}
```

Only if both the artist and album is known they're displayed. If only one or none of them is known the entire group between the brackets is excluded.

"is known" is here defined as "value evaluating to True in Python", i.e. an empty string or 0 (or 0.0) counts as "not known".

Inside a group always all format specifiers must evaluate to true (logical and).

You can nest groups. The inner group will only become part of the output if both the outer group and the inner group are eligible for output.

2.1.2 TimeWrapper

Some modules that output times use *TimeWrapper* to format these. TimeWrapper is a mere extension of the standard formatting method.

The time format that should be used is specified using the format specifier, i.e. with some_time being 3951 seconds a format string like `{some_time:%h:%m:%s}` would produce `1:5:51`.

- `%h`, `%m` and `%s` are the hours, minutes and seconds without leading zeros (i.e. 0 to 59 for minutes and seconds)
- `%H`, `%M` and `%S` are padded with a leading zero to two digits, i.e. 00 to 59
- `%l` and `%L` produce hours non-padded and padded but only if hours is not zero. If the hours are zero it produces an empty string.
- `%%` produces a literal %
- `%E` (only valid on beginning of the string) if the time is null, don't format anything but rather produce an empty string. If the time is non-null it is removed from the string.
- When the module in question also uses formatp, 0 seconds counts as "not known".
- The formatted time is stripped, i.e. spaces on both ends of the result are removed.

2.2 Logging

Errors do happen and to ease debugging i3pystatus includes a logging facility. By default i3pystatus will log exceptions raised by modules to files in your home directory named `.i3pystatus-<pid-of-thread>`. Some modules might log additional information.

2.2.1 Setting a specific logfile

When instantiating your `Status` object, the path to a log file can be specified (it accepts environment variables). If this is done, then log messages will be sent to that file and not to an `.i3pystatus-<pid-of-thread>` file in your home directory. This is useful in that it helps keep your home directory from becoming cluttered with files containing errors.

```
from i3pystatus import Status

status = Status(logfile='$HOME/var/i3pystatus.log')
```

2.2.2 Changing log format

New in version 3.35.

The `logformat` option can be useed to change the format of the log files, using LogRecord attributes.

```
from i3pystatus import Status

status = Status(
    logfile='/home/username/var/i3pystatus.log',
    logformat='%(asctime)s %(levelname)s:',
)
```

2.2.3 Log level

Every module has a `log_level` option which sets the *minimum* severity required for an event to be logged.

The numeric values of logging levels are given in the following table.

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

Exceptions raised by modules are of severity `ERROR` by default. The default `log_level` in i3pystatus (some modules might redefine the default, see the reference of the module in question) is 30 (`WARNING`).

2.3 Callbacks

Callbacks are used for click-events (merged into i3bar since i3 4.6, mouse wheel events are merged since 4.8), that is, you click (or scroll) on the output of a module in your i3bar and something happens. What happens is defined by these settings for each module individually:

- `on_leftclick`
- `on_doubleleftclick`
- `on_rightclick`
- `on_doublerightclick`
- `on_upscroll`
- `on_downscroll`

The global default action for all settings is `None` (do nothing), but many modules define other defaults, which are documented in the module reference.

Note: Each of these callbacks, when triggered, will call the module's `run()` function (typically only called each time the module's interval is reached). If there are things in the `run()` function of your module which you do not want to be executed every time a mouse event is triggered, then consider using threading to perform the module update, and manually sleep for the module's interval between updates. You can start the update thread in the module's `init()` function. The `run()` function can then either just update the module's displayed text, or simply do nothing (if your update thread also handles updating the display text). See the weather module for an example of this method.

The values you can assign to these four settings can be divided to following three categories:

Member callbacks

These callbacks are part of the module itself and usually do some simple module related tasks (like changing volume when scrolling, etc.). All available callbacks are (most likely not) documented in their respective module documentation.

For example the module *ALSA* has callbacks named `switch_mute`, `increase_volume` and `decrease volume`. They are already assigned by default but you can change them to your liking when registering the module.

```
status.register("alsa",
    on_leftclick = ["switch_mute"],
    # or as a strings without the list
    on_upscroll = "decrease_volume",
    on_downscroll = "increase_volume",
```

```
    # this will refresh any module by clicking on it
    on_rightclick = "run",
    )
```

Some callbacks also have additional parameters. Both `increase_volume` and `decrease_volume` have an optional parameter `delta` which determines the amount of percent to add/subtract from the current volume.

```
status.register("alsa",
    # all additional items in the list are sent to the callback as arguments
    on_upscroll = ["decrease_volume", 2],
    on_downscroll = ["increase_volume", 2],
    )
```

Python callbacks

These refer to to any callable Python object (most likely a function). To external Python callbacks that are not part of the module the `self` parameter is not passed by default. This allows to use many library functions with no additional wrapper.

If `self` is needed to access the calling module, the *`get_module()`* decorator can be used on the callback:

```
from i3pystatus import get_module

# Note that the 'self' parameter is required and gives access to all
# variables of the module.
@get_module
def change_text(self):
    self.output["full_text"] = "Clicked"

status.register("text",
    text = "Initial text",
    on_leftclick = [change_text],
    # or
    on_rightclick = change_text,
    )
```

If the module your attaching the callback too is not a subclass of *`IntervalModule`* you will need to invoke `init()`. using *`Uname`* as an example, the following code would suffice.

```
from i3pystatus import get_module

@get_module
def sys_info(self):
    if self.format == "{nodename}":
            self.format = "{sysname} {release} on {machine}"
        else:
            self.format = "{nodename}"
        self.init()

status.register("uname", format="{nodename}", on_rightclick=sys_info)
```

You can also create callbacks with parameters.

```
from i3pystatus import get_module

@get_module
```

```
def change_text(self, text="Hello world!", color="#ffffff"):
    self.output["full_text"] = text
    self.output["color"] = color

status.register("text",
    text = "Initial text",
    color = "#00ff00",
    on_leftclick = [change_text, "Clicked LMB", "#ff0000"],
    on_rightclick = [change_text, "Clicked RMB"],
    on_upscroll = change_text,
    )
```

External program callbacks

You can also use callbacks to execute external programs. Any string that does not match any *member callback* is treated as an external command. If you want to do anything more complex than executing a program with a few arguments, consider creating an *python callback* or execute a script instead.

```
status.register("text",
    text = "Launcher?",
    # open terminal window running htop
    on_leftclick = "i3-sensible-terminal -e htop",
    # open i3pystatus github page in firefox
    on_rightclick = "firefox --new-window https://github.com/enkore/i3pystatus",
    )
```

Most modules provide all the formatter data to program callbacks. The snippet below demonstrates how this could be used, in this case XMessage will display a dialog box showing verbose information about the network interface:

```
status.register("network",
    interface="eth0",
    on_leftclick="ip addr show dev {interface} | xmessage -file -"
    )
```

2.4 Hints

Hints are additional parameters used to customize output of a module. They give you access to all attributes supported by i3bar protocol.

Hints are available as the `hints` setting in all modules and its value should be a dictionary or `None`. An attribute defined in `hints` will be applied only if the module output does not contain attribute with the same name already.

Some possible uses for these attributes are:

- *min_width* and *align* can be used to set minimal width of output and align the text if its width is shorter than *minimal_width*.
- *separator* and *separator_block_width* can be used to remove the vertical bar that is separating modules.
- *background* can be used to set an alternative background color for the module. supports RGBA if your i3bar version does.
- *markup* can be set to *"none"* or *"pango"*. Pango markup provides additional formatting options for drawing rainbows and other fancy stuff.

Note: Pango markup requires that i3bar is configured to use Pango, too. It can't work with X core fonts.

Here is an example with the *network* module. Pango markup is used to keep the ESSID green at all times while the recieved/sent part is changing color depending on the amount of traffic.

```
status.register("network",
    interface = "wlp2s0",
    hints = {"markup": "pango"},
    format_up = "<span color=\"#00FF00\">{essid}</span> {bytes_recv:6.1f}KiB
↪{bytes_sent:5.1f}KiB",
    format_down = "",
    dynamic_color = True,
    start_color = "#00FF00",
    end_color = "#FF0000",
    color_down = "#FF0000",
    upper_limit = 800.0,
    )
```

Or you can use pango to customize the color of `status` setting in *now_playing* and *mpd* modules.

```
...
hints = {"markup": "pango"},
status = {
    "play": "",
    "pause": "<span color=\"orange\"></span>",
    "stop": "<span color=\"red\"></span>",
},
...
```

Or make two modules look like one.

```
status.register("text",
    text = "shmentarianism is a pretty long word.")
status.register("text",
    hints = {"separator": False, "separator_block_width": 0},
    text = "Antidisestabli",
    color="#FF0000")
```

Note: To prevent pango rendering errors, ampersands in the formatted text will be replaced with the HTML escape code `&`. Any ampersands that are themselves part of an HTML escape (e.g. `<`, `>`, etc.) will not be replaced.

2.5 Refreshing the bar

The whole bar can be refreshed by sending SIGUSR1 signal to i3pystatus process. This feature is not available in chained mode (*Status* was created with `standalone=False` parameter and gets it's input from `i3status` or a similar program).

To find the PID of the i3pystatus process look for the `status_command` you use in your i3 config file. If your *bar* section of i3 config looks like this

```
bar {
    status_command python ~/.config/i3/pystatus.py
}
```

then you can refresh the bar by using the following command:

```
pkill -SIGUSR1 -f "python /home/user/.config/i3/pystatus.py"
```

Note that the path must be expanded if using '~'.

2.6 Internet Connectivity

Module methods that `@require(internet)` won't be run unless a test TCP connection is successful. By default, this is made to Google's DNS server, but you can customize the host and port. See *internet*.

If you are behind a gateway that redirects web traffic to an authorization page and blocks other traffic, the DNS check will return a false positive. This is often encountered in open WiFi networks. In these cases it is helpful to try a service that is not traditionally required for web browsing:

```
from i3pystatus import Status

status = Status(check_internet=("whois.arin.net", 43))
```

```
from i3pystatus import Status

status = Status(check_internet=("github.com", 22))
```

2.7 Credentials

For modules which require credentials, i3pystatus supports credential management using the keyring module from PyPI.

Important: Many distributions have keyring pre-packaged, available as `python-keyring`. Unless you have KWallet or SecretService available, you will also most likely need to install keyrings.alt, which contains additional keyring backends for use by the keyring module.

Both i3pystatus and `i3pystatus-setting-util` will abort with a RuntimeError if keyring isinstalled but a usable keyring backend is not present, so it is a good idea to install both if you plan to use a module which supports credential handling.

To store credentials in a keyring, use the `i3pystatus-setting-util` script installed along i3pystatus.

Note: `i3pystatus-setting-util` will store credentials using the default keyring backend. The method for determining which backend is the default can be found *below*. If, for some reason, it is necessary to use a keyring other than the default, then you will need to override the default in your keyringrc.cfg for `i3pystatus-setting-util` to successfully use it.

Once you have successfully set up credentials, you can add the module to your config file without specifying the credentials in the registration, e.g.:

```
# Use the default keyring to retrieve credentials
status.register('github')
```

i3pystatus will locate and set the credentials during the module loading process. Currently supported credentials are `password`, `email` and `username`.

Note: To determine which backend is the default on your system, run the following:

```
python -c 'import keyring; print(keyring.get_keyring())'
```

If this command returns a `keyring.backends.fail.Keyring` object, none of the keyrings supported out-of-the box by the keyring module are available, and you will need to install the keyrings.alt Python module. keyrings.alt provides an encrypted keyring which will be seen as the default if both keyrings.alt and keyring are installed, and none of the keyrings supported by keyring are present:

```
$ python -c 'import keyring; print(keyring.get_keyring())'
<EncryptedKeyring at /home/username/.local/share/python_keyring/crypted_pass.cfg>
```

If the keyring backend you used to store credentials using `i3pystatus-setting-util` is not the default, then you can change which keyring backend i3pystatus will use in one of two ways:

1. Override the default in your keyringrc.cfg
2. Import and instantiate a keyring backend class, and pass it as the `keyring_backend` parameter when registering the module:

   ```
   # Requires the keyrings.alt package
   from keyrings.alt.file import PlaintextKeyring
   status.register('github', keyring_backend=PlaintextKeyring())
   ```

CHAPTER 3

Module reference

Module overview:

System *clock* - *cpu_freq* - *cpu_usage* - *disk* - *keyboard_locks* - *load* - *mem* - *swap* - *uname* - *uptime* - *xkblayout*

Audio *alsa* - *pulseaudio*

Hardware *backlight* - *battery* - *temp*

Network *net_speed* - *network* - *online* - *openstack_vms* - *openvpn*

Music *cmus* - *moc* - *mpd* - *now_playing* - *pianobar* - *spotify*

Websites *bitcoin* - *dota2wins* - *github* - *modsde* - *parcel* - *reddit* - *weather* - *whosonlocation*

Other *anybar* - *mail* - *pomodoro* - *pyload* - *text* - *updates*

Advanced *file* - *regex* - *makewatch* - *runwatch* - *shell*

Module list:

- *cpu_usage*
- *cpu_usage_bar*
- *cpu_usage_graph*
- *deluge*
- *disk*
- *dota2wins*
- *dpms*
- *exmo*
- *external_ip*
- *file*
- *github*
- *gpu_mem*
- *gpu_temp*
- *gpu_usage*
- *group*
- *hassio*
- *iinet*
- *keyboard_locks*
- *lastfm*
- *load*
- *mail*
- *makewatch*
- *mem*
- *mem_bar*
- *moc*
- *modsde*
- *moon*
- *mpd*
- *net_speed*
- *network*
- *now_playing*
- *online*
- *openfiles*
- *openstack_vms*
- *openvpn*
- *pagerduty*

- *parcel*
- *pianobar*
- *ping*
- *plexstatus*
- *pomodoro*
- *pulseaudio*
- *pyload*
- *random_password*
- *reddit*
- *redshift*
- *regex*
- *runwatch*
- *sabnzbd*
- *scores*
- *scratchpad*
- *sensu*
- *sge*
- *shell*
- *solaar*
- *sonos*
- *spaceapi*
- *spotify*
- *swap*
- *syncthing*
- *taskwarrior*
- *temp*
- *teslacharge*
- *text*
- *ticker*
- *timer*
- *timewarrior*
- *tlp*
- *travisci*
- *uname*
- *updates*
- *uptime*

- *vk*
- *weather*
- *weekcal*
- *whosonlocation*
- *window_title*
- *wireguard*
- *xkblayout*
- *yubikey*
- *zabbix*

class i3pystatus.abc_radio.**ABCRadio**

Streams ABC Australia radio - https://radio.abc.net.au/. Currently uses VLC to do the actual streaming.

Requires the PyPI packages *python-vlc*, *python-dateutil* and *requests*. Also requires VLC - https://www.videolan.org/vlc/index.html

Available formatters

- *{station}* — Current station
- *{title}* — Title of current show
- *{url}* — Show's URL
- *{remaining}* — Time left for current show
- *{player_state}* — Unicode icons representing play, pause and stop

Settings

- **format** (default: `{station} {title} {player_state}`) – format string for when the player is inactive
- **format_playing** (default: `{station} {title} {remaining} {player_state}`) – format string for when the player is playing
- **target_stations** (default: `[]`) – list of station ids to select from. Station ids can be obtained from the following XML - http://www.abc.net.au/radio/data/stations_apps_v3.xml. If the list is empty, all stations will be accessible.
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `toggle_play`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['cycle_stations',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['cycle_stations',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: `display_notification`) – Callback called on double left click (see *Callbacks*)

- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.alsa.`**ALSA**

Shows volume of ALSA mixer. You can also use this for inputs, btw.

Requires pyalsaaudio

Available formatters

- *{volume}* — the current volume in percent
- *{muted}* — the value of one of the *muted* or *unmuted* settings
- *{card}* — the associated soundcard
- *{mixer}* — the associated ALSA mixer

Settings

- **format** (default: ♪: `{volume}`)
- **format_muted** (default: *empty*) – optional format string to use when muted
- **mixer** (default: `Master`) – ALSA mixer
- **mixer_id** (default: `0`) – ALSA mixer id
- **card** (default: `-1`) – ALSA sound card
- **increment** (default: `5`) – integer percentage of max volume to in/decrement volume on mousewheel
- **muted** (default: `M`)
- **unmuted** (default: *empty*)
- **color_muted** (default: `#AAAAAA`)
- **color** (default: `#FFFFFF`)
- **channel** (default: `0`)
- **map_volume** (default: `False`) – volume display/setting as in AlsaMixer. increment option is ignored then.
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `switch_mute`) – Callback called on left click (see *Callbacks*)

- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `switch_mute`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `increase_volume`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `decrease_volume`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.amdgpu.`**Amdgpu**

Shows information about gpu's using the amdgpu driver

Available formatters

- *{temp}*
- *{sclk}* - Gpu clock speed
- *{mclk}* - Memory clock speed
- *{fan_speed}* - Fan speed
- *{gpu_usage}* - Gpu Usage percent

Settings

- **format** (default: `{temp} {mclk} {sclk}`)
- **color** (default: *empty*)
- **card** (default: `0`) – [1, 2, ...] card to read (options are in /sys/class/drm/)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.anybar.`**AnyBar**

This module shows dot with given color in your panel. What color means is up to you. When to change color is also up to you. It's a port of https://github.com/tonsky/AnyBar to i3pystatus. Color can be changed by sending text to UDP port. Check the original repo how to do it.

Settings

- **port** (default: `1738`) – UDP port to listen
- **color** (default: `#444444`) – initial color
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

main_loop()
Mainloop blocks so we thread it.

class i3pystatus.backlight.**Backlight**
Screen backlight info

•(Optional) requires *xbacklight* to change the backlight brightness with the scollwheel.

Available formatters

•*{brightness}* — current brightness relative to max_brightness

•*{max_brightness}* — maximum brightness value

•*{percentage}* — current brightness in percent

Settings

•**format** (default: `{brightness}/{max_brightness}`) – format string, formatters: brightness, max_brightness, percentage

•**format_no_backlight** (default: `No backlight`) – format string when no backlight file available

•**backlight** (default: `*`) – backlight, see */sys/class/backlight/*. Supports glob expansion, i.e. * matches anything. If it matches more than one filename, selects the first one in alphabetical order

•**color** (default: `#FFFFFF`)

•**components** (default: `{'brightness': (<class 'int'>,'brightness'),'max_brightness': (<class 'int'>,'max_brightness')}`)

•**transforms** (default: `{'percentage': <function Backlight.<lambda> at 0x7fe1f13597a0>}`)

•**base_path** (default: `/sys/class/backlight/{backlight}/`)

•**interval** (default: `5`)

•**on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: `lighter`) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: `darker`) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.battery.`**`BatteryChecker`**

This class uses the /sys/class/power_supply/.../uevent interface to check for the battery status.

Setting `battery_ident` to `ALL` will summarise all available batteries and aggregate the % as well as the time remaining on the charge. This is helpful when the machine has more than one battery available.

Available formatters

- *{remaining}* — remaining time for charging or discharging, uses TimeWrapper formatting, default format is *%E%h:%M*
- *{percentage}* — battery percentage relative to the last full value
- *{percentage_design}* — absolute battery charge percentage
- *{consumption (Watts)}* — current power flowing into/out of the battery
- *{status}*
- *{no_of_batteries}* — The number of batteries included
- *{battery_ident}* — the same as the setting
- *{bar}* —bar displaying the relative percentage graphically
- *{bar_design}* —bar displaying the absolute percentage graphically
- *{glyph}* — A single character or string (selected from 'glyphs') representing the current battery percentage

This module supports the *formatp* extended string format syntax. By setting the `FULL` status to an empty string, and including brackets around the `{status}` formatter, the text within the brackets will be hidden when the battery is full, as can be seen in the below example:

```
from i3pystatus import Status

status = Status()

status.register(
    'battery',
    interval=5,
    format='{battery_ident}: [{status} ]{percentage_design:.2f}%',
    alert=True,
    alert_percentage=15,
    status={
        'DPL': 'DPL',
        'CHR': 'CHR',
        'DIS': 'DIS',
        'FULL': '',
    }
)

# status.register(
#     'battery',
#     format='{status} {percentage:.0f}%',
#     levels={
```

```
#          25: "<=25",
#          50: "<=50",
#          75: "<=75",
#      },
# )

status.run()
```

Settings

•**battery_ident** (default: `ALL`) – The name of your battery, usually BAT0 or BAT1

•**format** (default: `{status} {remaining}`)

•**not_present_text** (default: `Battery {battery_ident} not present`) – Text displayed if the battery is not present. No formatters are available

•**alert** (default: `False`) – Display a libnotify-notification on low battery

•**critical_level_command** (default: *empty*) – Runs a shell command in the case of a critical power state

•**critical_level_percentage** (default: `1`)

•**alert_percentage** (default: `10`)

•**alert_timeout** (default: `-1`)

•**alert_format_title** (default: `Low battery`) – The title of the notification, all formatters can be used

•**alert_format_body** (default: `Battery {battery_ident} has only {percentage:.2f}% ({remaining:%E%hh:%Mm}) remaining!`) – The body text of the notification, all formatters can be used

•**path** (default: *empty*) – Override the default-generated path and specify the full path for a single battery

•**base_path** (default: `/sys/class/power_supply`) – Override the default base path for searching for batteries

•**battery_prefix** (default: `BAT`) – Override the default battery prefix

•**status** (default: `{'DPL': 'DPL','CHR': 'CHR','DIS': 'DIS','FULL': 'FULL'}`) – A dictionary mapping ('DPL', 'DIS', 'CHR', 'FULL') to alternative names

•**levels** (default: *empty*) – A dictionary mapping percentages of charge levels to corresponding names.

•**color** (default: `#ffffff`) – The text color

•**full_color** (default: `#00ff00`) – The full color

•**charging_color** (default: `#00ff00`) – The charging color

•**critical_color** (default: `#ff0000`) – The critical color

•**not_present_color** (default: `#ffffff`) – The not present color.

•**no_text_full** (default: `False`) – Don't display text when battery is full - 100%

•**glyphs** (default:) – Arbitrarily long string of characters (or array of strings) to represent battery charge percentage

•**use_design_percentage** (default: `False`) – Use design percentage rather then absolute percentage for alerts

•**interval** (default: `5`) – interval in seconds between module updates

- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.bitcoin.`**`Bitcoin`**

This module fetches and displays current Bitcoin market prices and optionally monitors transactions to and from a list of user-specified wallet addresses. Market data is pulled from the Bitaps Market API <https://bitaps.com> and it is possible to specify the exchange to be monitored. Transaction data is pulled from blockchain.info <https://blockchain.info/api/blockchain_api>.

Available formatters

- {last_price}
- {ask_price}
- {bid_price}
- {open_price}
- {volume}
- {volume_thousand}
- {volume_percent}
- {age}
- {status}
- {last_tx_type}
- {last_tx_addr}
- {last_tx_value}
- {balance_btc}
- {balance_fiat}

- {symbol}

Settings

- **format** (default: `{symbol} {status}{last_price}`) – Format string used for output.
- **currency** (default: `USD`) – Base fiat currency used for pricing.
- **wallet_addresses** (default: *empty*) – List of wallet address(es) to monitor.
- **color** (default: `#FFFFFF`) – Standard color
- **exchange** (default: `bitstamp`) – Get ticker from a custom exchange instead
- **colorize** (default: `False`) – Enable color change on price increase/decrease
- **color_up** (default: `#00FF00`) – Color for price increases
- **color_down** (default: `#FF0000`) – Color for price decreases
- **interval** (default: `600`) – Update interval.
- **symbol** (default:) – Symbol for bitcoin sign
- **status** (default: `{'price_up': '','price_down': ''}`)
- **on_leftclick** (default: `electrum`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `['open_something','https://bitaps.com/']`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

open_something(*url_or_command*)
Wrapper function, to pass the arguments to user_open

class i3pystatus.bluetooth.**Bluetooth**
Shows currently connected bluetooth devices.

•Requires `python-dbus` from your distro package manager, or `dbus-python` from PyPI.

Left click on the module to cycle forwards through devices, and right click to cycle backwards.

Available formatters (uses formatp)

•*{name}* — (the name of the device)

•*{dev_addr}* — (the bluetooth device address)

Available callbacks

•`next_device` — iterate forward through devices

•`prev_device` — iterate backwards through devices

Example module registration with callbacks:

::

status.register("now_playing", on_leftclick="next_device", on_rightclick="prev_device", on_upscroll="next_device", on_downscroll="prev_device")

Settings

•**format** (default: `{name}: {dev_addr}`) – formatp string

•**color** (default: `#ffffff`) – Text color

•**connected_color** (default: `#00ff00`) – Connected device color

•**show_disconnected** (default: `True`) – Show disconnected but paired devices

•**interval** (default: `1`) – interval in seconds between module updates

•**on_leftclick** (default: `next_device`) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: `prev_device`) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: `next_device`) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: `prev_device`) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

•**multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

•**hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.calendar.`**`Calendar`**

Generic calendar module. Requires the PyPI package `colour`.

Available formatters

•{title} - the title or summary of the event

•{remaining_time} - how long until this event is due

•{humanize_remaining} - how long until this event is due in human readable format

Additional formatters may be provided by the backend, consult their documentation for details.

Note: Optionally requires *humanize* to display time in human readable format.

Settings

•**format** (default: `{title} -{remaining}`) – Format string to display in the bar

•**backend** (required) – Backend to use for collecting calendar events

•**skip_recurring** (default: `False`) – Whether or not to skip recurring events

•**skip_all_day** (default: `False`) – Whether or not to skip all day events

•**skip_regex** (default: *empty*) – Skip events with titles that match this regex

•**update_interval** (default: `600`) – How often in seconds to call the backend's update method

•**urgent_seconds** (default: `300`) – When within this many seconds of the event, set the urgent flag

•**urgent_blink** (default: `False`) – Whether or not to blink when within urgent_seconds of the event

•**dynamic_color** (default: `True`) – Whether or not to change color as the event approaches

•**color** (default: *empty*)

•**interval** (default: `1`) – interval in seconds between module updates

•**on_leftclick** (default: `acknowledge`) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: `handle_click`) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

is_urgent()
Determine whether or not to set the urgent flag. If urgent_blink is set, toggles urgent flag on and off every second.

class i3pystatus.circleci.**CircleCI**
Get current status of circleci builds Requires *circleci dateutil.parser*

Formatters:

- *{repo_slug}* - repository owner/repository name
- *{repo_status}* - repository status
- *{repo_name}* - repository name
- *{repo_owner}* - repository owner
- *{last_build_started}* - date of the last finished started
- *{last_build_duration}* - duration of the last build, not populated with workflows(yet)

Examples

```
status_color_map = {
    'passed': '#00FF00',
    'failed': '#FF0000',
    'errored': '#FFAA00',
    'cancelled': '#EEEEEE',
    'started': '#0000AA',

}
```

```
repo_status_map={
    'success': '<span color="#00af00">success</span>',
    'running': '<span color="#0000af">running</span>',
    'failed': '<span color="#af0000">failed</span>',
}
```

Settings

- **format** (default: `{repo_owner}/{repo_name}-{repo_status} [({last_build_started}({last_build_duration}))]`)
- **circleci_token** (required) – circleci access token
- **repo_slug** (required) – repository identifier eg. “enkore/i3pystatus”
- **time_format** (default: `%m/%d`) – passed directly to .strftime() for *last_build_started*
- **repo_status_map** (default: *empty*) – map representing how to display status

- **duration_format** (default: `%m:%S`) – *last_build_duration* format string
- **status_color_map** (default: *empty*) – color for all text based on status
- **color** (default: `#DDDDDD`) – color for all text not otherwise colored
- **workflow_name** (default: *empty*) – [WORKFLOWS_ONLY] if specified, monitor this workflows status. if not specified this module will default to reporting the status of your last build
- **workflow_branch** (default: *empty*) – [WORKFLOWS_ONLY] if specified, monitor the workflows in this branch
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `open_build_webpage`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.clock.`**`Clock`**

This class shows a clock.

Note: Optionally requires *pytz* for time zone data when using time zones other than local time.

Format can be passed in four different ways:

- single string, no timezone, just the strftime-format
- one two-tuple, first is the format, second the timezone
- list of strings - no timezones
- list of two tuples, first is the format, second is timezone

Use mousewheel to cycle between formats.

For complete time format specification see:

```
man strftime
```

All available timezones are located in directory:

```
/usr/share/zoneinfo/
```

Format examples

```
# one format, local timezone
format = '%a %b %-d %b %X'
# multiple formats, local timezone
format = [ '%a %b %-d %b %X', '%X' ]
# one format, specified timezone
format = ('%a %b %-d %b %X', 'Europe/Bratislava')
# multiple formats, specified timezones
format = [ ('%a %b %-d %b %X', 'America/New_York'), ('%X', 'Etc/GMT+9') ]
```

Settings

- **format** (default: *empty*) – *None* means to use the default, locale-dependent format.
- **color** (default: `#ffffff`) – RGB hexadecimal code color specifier, default to #ffffff
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['scroll_format',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['scroll_format',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.cmus.`**Cmus**

Gets the status and current song info using cmus-remote

Available formatters

- *{status}* — current status icon (paused/playing/stopped)
- *{song_elapsed}* — song elapsed time (mm:ss format)
- *{song_length}* — total song duration (mm:ss format)
- *{artist}* — artist
- *{title}* — title
- *{album}* — album
- *{tracknumber}* — tracknumber
- *{file}* — file or url name
- *{stream}* — song name from stream
- *{bitrate}* — bitrate

Settings

- **format** (default: `{status} {song_elapsed}/{song_length} {artist} -{title}`) – formatp string
- **format_not_running** (default: `Not running`) – Text to show if cmus is not running
- **color** (default: `#ffffff`) – The color of the text
- **color_not_running** (default: `#ffffff`) – The color of the text, when cmus is not running
- **status** (default: `{'paused': '','playing': '','stopped': ''}`) – Dictionary mapping status to output
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `playpause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `next_song`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `previous_song`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.coin.`**Coin**

Fetches live data of all cryptocurrencies available at coinmarketcap <https://coinmarketcap.com/>. Coin setting should be equal to the 'id' field of your coin in <https://api.coinmarketcap.com/v1/ticker/>.

Example coin settings: bitcoin, bitcoin-cash, ethereum, litecoin, dash, lisk. Example currency settings: usd, eur, huf.

Available formatters

- {symbol}
- {price}
- {rank}
- {24h_volume}
- {market_cap}
- {available_supply}
- {total_supply}
- {max_supply}
- {percent_change_1h}
- {percent_change_24h}
- {percent_change_7d}
- {last_updated} - time of last update on the API's part
- {status}

Settings

- **format** (default: `{symbol} {price}{status}`) – format string used for output.
- **color** (default: *empty*)
- **coin** (default: `ethereum`) – cryptocurrency to fetch
- **decimal** (default: `2`) – round coin price down to this decimal place
- **currency** (default: `USD`) – fiat currency to show fiscal data
- **symbol** (default: ¤) – coin symbol
- **interval** (default: `600`) – update interval in seconds
- **status_interval** (default: `24h`) – percent change status in the last: '1h' / '24h' / '7d'
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.cpu_freq.`**CpuFreq**

class uses by default */proc/cpuinfo* to determine the current cpu frequency

Available formatters

- *{avg}* - mean from all cores in MHz *4.3f*
- *{avgg}* - mean from all cores in GHz *1.2f*
- *{coreX}* - frequency of core number *X* in MHz (format *4.3f*), where 0 <= *X* <= number of cores - 1
- *{coreXg}* - frequency of core number *X* in GHz (fromat *1.2f*), where 0 <= *X* <= number of cores - 1

Settings

- **format** (default: `{avgg}`)
- **color** (default: `#FFFFFF`) – The text color
- **file** (default: `/proc/cpuinfo`) – override default path
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

createvaluesdict()
function processes the /proc/cpuinfo file, use file=/sys to use kernel >=4.13 location :return: dictionary used as the full-text output for the module

class i3pystatus.cpu_usage.**CpuUsage**
Shows CPU usage. The first output will be inacurate.

Linux only Requires the PyPI package 'colour'.

Available formatters

- *{usage}* — usage average of all cores
- *{usage_cpu*}* — usage of one specific core. replace "*" by core number starting at 0
- *{usage_all}* — usage of all cores separate. usess natsort when available(relevant for more than 10 cores)

Settings

- **format** (default: `{usage:02}%`) – format string.
- **format_all** (default: `{core}:{usage:02}%`) – format string used for {usage_all} per core. Available formaters are {core} and {usage}.
- **exclude_average** (default: `False`) – If True usage average of all cores will not be in format_all.
- **color** (default: `#FFFFFF`) – HTML color code #RRGGBB
- **dynamic_color** (default: `False`) – Set color dynamically based on CPU usage. Note: this overrides color_up
- **start_color** (default: `#00FF00`) – Hex or English name for start of color range, eg '#00FF00' or 'green'
- **end_color** (default: `red`) – Hex or English name for end of color range, eg '#FF0000' or 'red'
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

calculate_usage(*cpu*, *total*, *busy*)
calculates usage

gen_format_all(*usage*)
generates string for format all

get_cpu_timings()
reads and parses /proc/stat returns dictionary with all available cores including global average

get_usage()
parses /proc/stat and calcualtes total and busy time (more specific USER_HZ see man 5 proc for further informations)

class i3pystatus.cpu_usage_bar.**CpuUsageBar**
Shows CPU usage as a bar (made with unicode box characters). The first output will be inacurate.

Linux only

Requires the PyPI package *colour*.

Available formatters

- *{usage_bar}* — usage average of all cores
- *{usage_bar_cpu*}* — usage of one specific core. replace "*" by core number starting at 0

Settings

- **format** (default: `{usage_bar}`) – format string
- **bar_type** (default: `horizontal`) – whether the bar should be vertical or horizontal. Allowed values: *vertical* or *horizontal*
- **cpu** (default: `usage_cpu`) – cpu to base the colors on. Choices are 'usage_cpu' for all or 'usage_cpu*'. Replace '*' by core number starting at 0.
- **start_color** (default: `#00FF00`) – Hex or English name for start of color range, eg '#00FF00' or 'green'
- **end_color** (default: `red`) – Hex or English name for end of color range, eg '#FF0000' or 'red'
- **dynamic_color** (default: `False`) – Use dynamic color
- **format_all** (default: `{core}:{usage:02}%`) – format string used for {usage_all} per core. Available formaters are {core} and {usage}.
- **exclude_average** (default: `False`) – If True usage average of all cores will not be in format_all.

- **color** (default: `#FFFFFF`) – HTML color code #RRGGBB
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.cpu_usage_graph.`**`CpuUsageGraph`**

Shows CPU usage as a Unicode graph. The first output will be inacurate.

Depends on the PyPI colour module - https://pypi.python.org/pypi/colour/0.0.5

Linux only

Available formatters

- *{cpu_graph}* — graph of cpu usage.
- *{usage}* — usage average of all cores
- *{usage_cpu*}* — usage of one specific core. replace "*" by core number starting at 0
- *{usage_all}* — usage of all cores separate. usess natsort when available(relevant for more than 10 cores)

Settings

- **cpu** (default: `usage_cpu`) – cpu to monitor, choices are 'usage_cpu' for all or 'usage_cpu*'. Replace '*' by core number starting at 0.
- **start_color** (default: `#00FF00`) – Hex or English name for start of color range, eg '#00FF00' or 'green'
- **end_color** (default: `red`) – Hex or English name for end of color range, eg '#FF0000' or 'red'
- **graph_width** (default: `15`) – Width of the cpu usage graph
- **graph_style** (default: `blocks`) – Graph style ('blocks', 'braille-fill', 'braille-peak', or 'braille-snake')

- **direction** (default: `left-to-right`) – Graph running direction ('left-to-right', 'right-to-left')
- **format** (default: `{cpu_graph}`) – format string.
- **format_all** (default: `{core}:{usage:02}%`) – format string used for {usage_all} per core. Available formaters are {core} and {usage}.
- **exclude_average** (default: `False`) – If True usage average of all cores will not be in format_all.
- **color** (default: `#FFFFFF`) – HTML color code #RRGGBB
- **dynamic_color** (default: `False`) – Set color dynamically based on CPU usage. Note: this overrides color_up
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.deluge.`**Deluge**

Deluge torrent module Requires *deluge-client*

Formatters:

- *{num_torrents}* - number of torrents in deluge
- *{free_space_bytes}* - bytes free in path
- *{used_space_bytes}* - bytes used in path
- *{upload_rate}* - bytes sent per second
- *{download_rate}* - bytes received per second
- *{total_uploaded}* - bytes sent total
- *{total_downloaded}* - bytes received total

Settings

- **format** (default: `{num_torrents} {free_space_bytes}`)
- **color** (default: *empty*)
- **rounding** (default: `2`) – number of decimal places to round numbers too
- **host** (default: `127.0.0.1`) – address of deluge server (default: 127.0.0.1)
- **port** (default: `58846`) – port of deluge server (default: 58846)
- **username** (required) – username to authenticate with deluge
- **password** (required) – password to authenticate to deluge
- **path** (default: *empty*) – override "download path" server-side when checking space used/free
- **offline_string** (default: `offline`) – string to output while unable to connect to deluge daemon
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

get_free_space(*path=None*)
get free space of path in bytes (default: download location)

get_path_size(*path=None*)
get used space of path in bytes (default: download location)

class i3pystatus.disk.**Disk**
Gets `{used}`, `{free}`, `{avail}` and `{total}` amount of bytes on the given mounted filesystem.

These values can also be expressed as percentages with the `{percentage_used}`, `{percentage_free}` and `{percentage_avail}` formats.

Settings

•**format** (default: `{free}/{avail}`)

•**path** (required)

•**divisor** (default: `1073741824`) – divide all byte values by this value, default is 1024**3 (gigabyte)

•**display_limit** (default: `inf`) – if more space is available than this limit the module is hidden

•**critical_limit** (default: `0`) – critical space limit (see critical_color)

•**critical_color** (default: `#FF0000`) – the critical color

•**color** (default: `#FFFFFF`) – the common color

•**round_size** (default: `2`) – precision, None for INT

•**mounted_only** (default: `False`) – display only if path is a valid mountpoint

•**format_not_mounted** (default: *empty*)

•**color_not_mounted** (default: `#FFFFFF`)

•**interval** (default: `5`) – interval in seconds between module updates

•**on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

•**multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

•**hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.dota2wins.`**Dota2wins**

Displays the win/loss ratio of a given Dota account. Requires: dota2py

Settings

•**matches** (default: `25`) – Number of recent matches to calculate

•**steamid** (required) – Steam ID or username to track

- **steam_api_key** (required) – Steam API key (http://steamcommunity.com/dev/apikey)
- **good_threshold** (default: `50`) – Win percentage (or higher) which you are happy with
- **bad_threshold** (default: `45`) – Win percentage you want to be alerted (difference between good_threshold and bad_threshold is cautious_threshold)
- **interval** (default: `1800`) – Update interval (games usually last at least 20 min).
- **good_color** (default: `#00FF00`) – Color of text while win percentage is above good_threshold
- **bad_color** (default: `#FF0000`) – Color of text while win percentage is below bad_threshold
- **caution_color** (default: `#FFFF00`) – Color of text while win precentage is between good and bad thresholds
- **screenname** (default: `retrieve`) – If set to 'retrieve', requests for the users's screenname via API calls. Else, use the supplied string as the user's screename
- **format** (default: `{screenname} {wins}W:{losses}L {win_percent:.2f}%`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.dpms.`**DPMS**

Shows and toggles status of DPMS which prevents screen from blanking.

Available formatters

- *{status}* — the current status of DPMS

@author Georg Sieber <g.sieber AT gmail.com>

Settings

- **format** (default: `DPMS: {status}`)
- **format_disabled** (default: `DPMS: {status}`)
- **color** (default: `#FFFFFF`)
- **color_disabled** (default: `#AAAAAA`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `toggle_dpms`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.exmo.`**Exmo**

This module fetching and displays exchange rates with EXMO. Using API <https://exmo.me/en/api>.

Available formatters

- {buy_price}
- {status}
- {pair}

Settings

- **format** (default: `{buy_price}[ {status}] {pair}`) – Format string used for output
- **pair** (default: `BTC_USD`) – Currency pair for display on output
- **color** (default: `#FFFFFF`) – Standard color
- **colorize** (default: `False`) – Enable color change on price increase/decrease

- **color_up** (default: `#00FF00`) – Color for price increases
- **color_down** (default: `#FF0000`) – Color for price decreases
- **interval** (default: `60`) – Update interval.
- **status** (default: `{'price_up': '','price_down': ''}`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

open_something(*url_or_command*)
Wrapper function, to pass the arguments to user_open

class i3pystatus.external_ip.**ExternalIP**
Shows the external IP with the country code/name.

Requires the PyPI package *GeoIP*.

Available formatters

- {country_name} the full name of the country from the IP (eg. 'United States')
- {country_code} the country code of the country from the IP (eg. 'US')
- {ip} the ip

Settings

- **format** (default: `{country_name} {country_code} {ip}`)
- **color** (default: `#FFFFFF`)
- **color_down** (default: `#FF0000`) – color when the http request failed
- **color_hide** (default: `#FFFF00`) – color when the user has decide to switch to the hide format

- **format_down** (default: `Timeout`) – format when the http request failed
- **format_hide** (default: `{country_code}`) – format when the user has decide to switch to the hide format
- **ip_website** (default: `https://api.ipify.org`) – http website where the IP is directly available as raw
- **timeout** (default: `5`) – timeout in seconds when the http request is taking too much time
- **interval** (default: `15`) – interval in seconds between module updates
- **on_leftclick** (default: `switch_hide`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `run`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.file.`**File**

Rip information from text files

components is a dict of pairs of the form:

```
name => (callable, file)
```

- Where *name* is a valid identifier, which is used in the format string to access the value of that component.
- *callable* is some callable to convert the contents of *file*. A common choice is float or int.
- *file* names a file, relative to *base_path*.

transforms is a optional dict of callables taking a single argument (a dictionary containing the values of all components). The return value is bound to the key.

Settings

- **format** (required)
- **components** (required)
- **transforms** (default: `{}`)
- **base_path** (default: `/`)
- **color** (default: `#FFFFFF`)
- **interval** (default: `5`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.github.`**`Github`**

This module checks the GitHub system status, and optionally the number of unread notifications.

Changed in version 3.36: Module now checks system status in addition to unread notifications.

Note: For notification checking, the following is required:

- The requests module must be installed.
- Either `access_token` (recommended) or `username` and `password` must be used to authenticate to GitHub.

Using an access token is the recommended authentication method. Click here to generate a new access token. Fill in the **Token description** box, and enable the **notifications** scope by checking the appropriate checkbox. Then, click the **Generate token** button.

Important: An access token is the only supported means of authentication for this module, if 2-factor authentication is enabled.

See here for more information on GitHub's authentication API.

If you would rather use a username and password pair, you can either pass them as arguments when registering the module, or use i3pystatus' *credential management* support to store them in a keyring. Keep in mind that if you do not pass a `username` or `password` parameter when registering the module, i3pystatus will still attempt to retrieve these values from a keyring if the keyring Python module is installed. This could result in i3pystatus aborting during startup if it cannot find a usable keyring backend. If you do not plan to use credential management at all in i3pystatus, then you should either ensure that A) keyring is not installed, or B) both keyring and keyrings.alt are installed, to avoid this error.

Available formatters

- *{status}* — Current GitHub status. This formatter can be different depending on the current outage status (`none`, `minor`, `major`, or `critical`). The content displayed for each of these statuses is defined in the **status** config option.
- *{unread}* — When there are unread notifications, this formatter will contain the value of the **unread_marker** marker config option. there are no unread notifications, it formatter will be an empty string.
- *{unread_count}* — The number of unread notifications notifications, it will be an empty string.
- *{update_error}* — When an error is encountered updating this module, this formatter will be set to the value of the **update_error** config option.

Click events

This module responds to 4 different click events:

- **Left-click** — Forces an update of the module.
- **Right-click** — Triggers a desktop notification showing the most recent update to the GitHub status. This is useful when the status changes when you are away from your computer, so that the updated status can be seen without visiting the GitHub Status Dashboard. This click event requires **notify_status** to be set to `True`.
- **Double left-click** — Opens the GitHub notifications page in your web browser.
- **Double right-click** — Opens the GitHub Status Dashboard in your web browser.

Desktop notifications

New in version 3.36.

If **notify_status** is set to `True`, a notification will be displayed when the status reported by the GitHub Status API changes.

If **notify_unread** is set to `True`, a notification will be displayed when new unread notifications are found. Double-clicking the module will launch the GitHub notifications dashboard in your browser.

Note: A notification will be displayed if there was a problem querying the GitHub Status API, irrespective of whether or not **notify_status** or **notify_unread** is set to `True`.

Example configuration

The below example enables desktop notifications, enables Pango hinting for differently-colored **update_error** and **refresh_icon** text, and alters the both the status text and the colors used to visually denote the current status level. It also sets the log level to debug, for troubleshooting purposes.

```
status.register(
    'github',
    log_level=logging.DEBUG,
    notify_status=True,
    notify_unread=True,
    access_token='0123456789abcdef0123456789abcdef01234567',
    hints={'markup': 'pango'},
    update_error='<span color="#af0000">!</span>',
    refresh_icon='<span color="#ff5f00"></span>',
    status={
        'none': '✓',
        'minor': '!',
        'major': '!!',
        'critical': '!!!',
    },
    colors={
        'none': '#008700',
        'minor': '#d7ff00',
        'major': '#af0000',
        'critical': '#640000',
    },
)
```

Note: Setting debug logging and authenticating with an access token will include the access token in the log file, as the notification URL is logged at this level.

Extended string formatting

New in version 3.36.

This module supports the *formatp* extended string format syntax. This allows for values to be hidden when they evaluate as False. The default `format` string value for this module makes use of this syntax to conditionally show the value of the `update_error` config value when the backend encounters an error during an update, but this can also be used to only show the number of unread notifications when that number is not **0**. The below example would show the unread count as **(3)** when there are 3 unread notifications, but would show nothing when there are no unread notifications.

```
status.register(
    'github',
    notify_status=True,
    notify_unread=True,
    access_token='0123456789abcdef0123456789abcdef01234567',
    format='{status}[ ({unread_count})][ {update_error}]'
)
```

Settings

•**format** (default: `{status}[ {unread}][ {update_error}]`) – format string

•**status** (default: `{}`) – Dictionary mapping statuses to the text which represents that status type. This defaults to `GitHub` for all status types.

•**colors** (default: `{'none': '#28a745','maintenance': '#4f8cc9','minor': '#dbab09','major': '#e36209','critical': '#dc3545'}`) – Dictionary mapping statuses to the color used to display the status text

•**refresh_icon** (default:) – Text to display (in addition to any text currently shown by the module) when refreshing the GitHub status. **NOTE:** Depending on how quickly the update is performed, the icon may not be displayed.

•**update_error** (default: `!`) – Value for the `{update_error}` formatter when an error is encountered while checking GitHub status

•**keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials

•**username** (default: *empty*)

•**password** (default: *empty*)

•**access_token** (default: *empty*)

•**unread_marker** (default: `•`) – Defines the string that the `{unread}` formatter shows when there are pending notifications

•**notify_status** (default: `False`) – Set to `True` to display a desktop notification on status changes

•**notify_unread** (default: `False`) – Set to `True` to display a desktop notification when new notifications are detected

•**unread_notification_template** (default: `You have %d new notification(s)`) – String with no more than one `%d`, which will be replaced by the number of new unread notifications. Useful for those with non-English locales who would like the notification to be in their native language. The `%d` can be omitted if desired.

•**api_methods_url** (default: `https://www.githubstatus.com/api/v2/summary.json`) – URL from which to retrieve the API endpoint URL which this module will use to check the GitHub Status

•**status_url** (default: `https://www.githubstatus.com`) – The URL to the status page (opened when the module is double-clicked with the right mouse button

•**notifications_url** (default: `https://github.com/notifications`) – The URL to the GitHub notifications page (opened when the module is double-clicked with the left mouse button

•**interval** (default: `600`) – interval in seconds between module updates

•**on_leftclick** (default: `['perform_update']`) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: `['show_status_notification']`) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: `['launch_notifications_url']`) – Callback called on double left click (see *Callbacks*)

- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: `['launch_status_url']`) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.gpu_mem.`**GPUMemory**

Shows GPU memory load

Currently Nvidia only and nvidia-smi required

Available formatters

- {avail_mem}
- {percent_used_mem}
- {used_mem}
- {total_mem}

Settings

- **format** (default: `{avail_mem} MiB`) – format string used for output.
- **divisor** (default: `1`) – divide all megabyte values by this value, default is 1 (megabytes)
- **warn_percentage** (default: `50`) – minimal percentage for warn state
- **alert_percentage** (default: `80`) – minimal percentage for alert state
- **color** (default: `#00FF00`) – standard color
- **warn_color** (default: `#FFFF00`) – defines the color used when warn percentage is exceeded
- **alert_color** (default: `#FF0000`) – defines the color used when alert percentage is exceeded
- **round_size** (default: `1`) – defines number of digits in round
- **gpu_number** (default: `0`) – set the gpu number when you have several GPU
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.gpu_temp.`**GPUTemperature**

Shows GPU temperature

Currently Nvidia only and nvidia-smi required

Available formatters

- *{temp}* — the temperature in integer degrees celsius

Settings

- **format** (default: `{temp} °C`) – format string used for output. {temp} is the temperature in integer degrees celsius
- **display_if** (default: `True`) – snippet that gets evaluated. if true, displays the module output
- **gpu_number** (default: `0`) – set the gpu number when you have several GPU
- **color** (default: `#FFFFFF`)
- **alert_temp** (default: `90`)
- **alert_color** (default: `#FF0000`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.gpu_usage.`**GPUUsage**

Shows GPU load in percent

Currently Nvidia only and nvidia-smi required

Available formatters

- {usage}

Settings

- **format** (default: `{usage}%`) – format string used for output.
- **warn_percentage** (default: `50`) – minimal percentage for warn state
- **alert_percentage** (default: `80`) – minimal percentage for alert state
- **color** (default: `#00FF00`) – standard color
- **warn_color** (default: `#FFFF00`) – defines the color used when warn percentage is exceeded
- **alert_color** (default: `#FF0000`) – defines the color used when alert percentage is exceeded
- **gpu_number** (default: `0`) – set the gpu number when you have several GPU
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.group.`**Group**

Module for grouping modules together Cycles trough groups by means of scrolling

```
group = Group()
group.register("network",
    interface="eth0",
    divisor=1024,
    start_color='white',
    format_up="{bytes_recv}K / {bytes_sent}K"
)
group.register("network",
    interface="eth0",
    color_up='#FFFFFF',
    format_up="{v4}"
)
status.register(group)
```

Settings

- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['cycle_module',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['cycle_module',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

on_click(*button*, ***kwargs*)
Capture scrollup and scorlldown to move in groups Pass everthing else to the module itself

class i3pystatus.hassio.**Hassio**
Displays the state of a Homeassistant.io entity Requires the PyPI package *requests*

Settings

- **entity_id** (required) – Entity ID to track.
- **hassio_url** (required) – URL to your hassio install. (default: https://localhost:8123)
- **hassio_token** (required) – HomeAssistant API token (https://developers.home-assistant.io/docs/auth_api/#long-lived-access-token)
- **interval** (default: `15`) – Update interval.
- **desired_state** (default: `on`) – The desired or "good" state of the entity.
- **good_color** (default: `#00FF00`) – Color of text while entity is in desired state
- **bad_color** (default: `#FF0000`) – Color of text while entity is not in desired state
- **format** (default: `{friendly_name}: {state}`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.iinet.**IINet**
Check IINet Internet usage. Requires *requests* and *colour*

Formatters:

- *{percentage_used}* — percentage of your quota that is used
- *{percentage_available}* — percentage of your quota that is available
- *{used}* - GB of your quota used

Settings

- **format** (default: `{percent_used}`)
- **username** (default: *empty*) – Username for IINet
- **password** (default: *empty*) – Password for IINet
- **start_color** (default: `#00FF00`) – Beginning color for color range
- **end_color** (default: `#FF0000`) – End color for color range
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.keyboard_locks.`**Keyboard_locks**

Shows the status of CAPS LOCK, NUM LOCK and SCROLL LOCK

Available formatters

- *{caps}* — the current status of CAPS LOCK
- *{num}* — the current status of NUM LOCK
- *{scroll}* — the current status of SCROLL LOCK

Settings

- **format** (default: `{caps} {num} {scroll}`) – Format string
- **caps_on** (default: `CAP`) – String to show in {caps} when CAPS LOCK is on
- **caps_off** (default: `____`) – String to show in {caps} when CAPS LOCK is off

- **num_on** (default: `NUM`) – String to show in {num} when NUM LOCK is on
- **num_off** (default: `____`) – String to show in {num} when NUM LOCK is off
- **scroll_on** (default: `SCR`) – String to show in {scroll} when SCROLL LOCK is on
- **scroll_off** (default: `____`) – String to show in {scroll} when SCROLL LOCK is off
- **color** (default: `#FFFFFF`)
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.lastfm.`**`LastFM`**

Displays currently playing song as reported by last.fm. Get your API key from http://www.last.fm/api.

Settings

- **apikey** (required) – API key used to make calls to last.fm.
- **user** (required) – Name of last.fm user to track.
- **playing_format** (default: `{artist} -{track}`) – Output format when a song is playing
- **stopped_format** (default: *empty*) – Output format when nothing is playing
- **playing_color** (default: `FFFFFF`)
- **stopped_color** (default: `000000`)
- **interval** (default: `5`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)

- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.load.`**Load**

Shows system load

Available formatters

- *{avg1}* — the load average of the last minute
- *{avg5}* — the load average of the last five minutes
- *{avg15}* — the load average of the last fifteen minutes
- *{tasks}* — the number of tasks (e.g. 1/285, which indiciates that one out of 285 total tasks is runnable)

Settings

- **format** (default: `{avg1} {avg5}`)
- **color** (default: `#ffffff`) – The text color
- **critical_limit** (default: `2`) – Limit above which the load is considered critical, defaults to amount of cores.
- **critical_color** (default: `#ff0000`) – The critical color
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mail.`**Mail**

Generic mail checker

The *backends* setting determines the backends to use. For available backends see *Mail Backends*.

Settings

- **backends** (required) – List of backends (instances of `i3pystatus.mail.xxx.zzz`, e.g. *imap.IMAP*)
- **color** (default: `#ffffff`)
- **color_unread** (default: `#ff0000`)
- **format** (default: `{unread} new email`)
- **format_plural** (default: `{account} : {current_unread}/{unread} new emails`)
- **hide_if_null** (default: `True`) – Don't output anything if there are no new mails
- **email_client** (default: *empty*) – The command to run on left click. For example, to launch Thunderbird set `email_client` to ``thunderbird``. Alternatively, to bring Thunderbird into focus, set `email_client` to `i3-msg -q [class="^Thunderbird$"] focus`. Hint: To discover the X window class of your email client run 'xprop | grep -i class' and click on it's window
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `open_client`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['scroll_backend',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['scroll_backend',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

run()
Returns the sum of unread messages across all registered backends

class i3pystatus.makewatch.**MakeWatch**
Watches for make jobs and notifies when they are completed. requires: psutil

Settings

- **name** (default: `make`) – Listen for a job other than 'make' jobs
- **running_color** (default: `#FF0000`) – Text color while the job is running
- **idle_color** (default: `#00FF00`) – Text color while the job is not running
- **format** (default: `{name}: {status}`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.mem.**Mem**
Shows memory load

Available formatters

- {avail_mem}
- {percent_used_mem}
- {used_mem}
- {total_mem}

Requires psutil (from PyPI)

Settings

- **format** (default: `{avail_mem} MiB`) – format string used for output.
- **divisor** (default: `1048576`) – divide all byte values by this value, default is 1024**2 (megabytes)
- **warn_percentage** (default: `50`) – minimal percentage for warn state
- **alert_percentage** (default: `80`) – minimal percentage for alert state
- **color** (default: `#00FF00`) – standard color
- **warn_color** (default: `#FFFF00`) – defines the color used when warn percentage is exceeded
- **alert_color** (default: `#FF0000`) – defines the color used when alert percentage is exceeded
- **round_size** (default: `1`) – defines number of digits in round
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mem_bar.`**MemBar**

Shows memory load as a bar.

Available formatters

- {used_mem_bar}

Requires psutil and colour (from PyPI)

Settings

- **format** (default: `{used_mem_bar}`) – format string used for output.
- **warn_percentage** (default: `50`) – minimal percentage for warn state
- **alert_percentage** (default: `80`) – minimal percentage for alert state
- **color** (default: `#00FF00`) – standard color
- **warn_color** (default: `#FFFF00`) – defines the color used when warn percentage is exceeded
- **alert_color** (default: `#FF0000`) – defines the color used when alert percentage is exceeded
- **multi_colors** (default: `False`) – whether to use range of colors from 'color' to 'alert_color' based on memory usage.
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.moc.`**Moc**

Display various information from MOC (music on console)

Available formatters

- *{status}* — current status icon (paused/playing/stopped)
- *{song_elapsed}* — song elapsed time (mm:ss format)
- *{song_length}* — total song duration (mm:ss format)
- *{artist}* — artist
- *{title}* — title
- *{album}* — album
- *{tracknumber}* — tracknumber
- *{file}* — file or url name

Settings

- **format** (default: `{status} {song_elapsed}/{song_length} {artist} -{title}`) – formatp string
- **format_not_running** (default: `Not running`) – Text to show if MOC is not running
- **color** (default: `#ffffff`) – The color of the text
- **color_not_running** (default: `#ffffff`) – The color of the text, when MOC is not running
- **status** (default: `{'pause': '','play': '','stop': ''}`) – Dictionary mapping status to output
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `toggle_pause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `next_song`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `previous_song`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.modsde.**ModsDeChecker**
This class returns i3status parsable output of the number of unread posts in any bookmark in the mods.de forums.

Settings

- **format** (default: {unread} new posts in bookmarks) – Use {unread} as the formatter for number of unread posts
- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **offset** (default: 0) – subtract number of posts before output
- **color** (default: #7181fe)
- **username** (required)
- **password** (required)
- **interval** (default: 5) – interval in seconds between module updates
- **on_leftclick** (default: open_browser) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: 0.25) – Time (in seconds) before a single click is executed.
- **hints** (default: {'markup': 'none'}) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.moon.**MoonPhase**
Available Formatters

status: Allows for mapping of current moon phase - New Moon: - Waxing Crescent: - First Quarter: - Waxing Gibbous: - Full Moon: - Waning Gibbous: - Last Quarter: - Waning Crescent:

Settings

- **format** (default: {illum} {status} {moonicon})

- **status** (default: `{'New Moon': 'NM','Waxing Crescent': 'WaxCres','First Quarter': 'FQ','Waxing Gibbous': 'WaxGib','Full Moon': 'FM','Waning Gibbous': 'WanGib','Last Quarter': 'LQ','Waning Crescent': 'WanCres'}`) – Current moon phase
- **illum** (default: `<function MoonPhase.illum at 0x7fe1f12ce8c0>`) – Percentage that is illuminated
- **color** (default: `{'New Moon': '#00BDE5','Waxing Crescent': '#138DD8','First Quarter': '#265ECC','Waxing Gibbous': '#392FBF','Full Moon': '#4C00B3','Waning Gibbous': '#871181','Last Quarter': '#C32250','Waning Crescent': '#FF341F'}`) – Set color
- **moonicon** (default: `{'New Moon': '','Waxing Crescent': '','First Quarter': '','Waxing Gibbous': '','Full Moon': '','Waning Gibbous': '','Last Quarter': '','Waning Crescent': ''}`) – Set icon
- **interval** (default: `7200`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mpd.`**MPD**

Displays various information from MPD (the music player daemon)

Available formatters (uses formatp)

- *{title}* — (the title of the current song)
- *{album}* — (the album of the current song, can be an empty string (e.g. for online streams))
- *{artist}* — (can be empty, too)
- *{album_artist}* — (can be empty)
- *{filename}* — (file name with out extension and path; empty unless title is empty)

- *{song_elapsed}* — (Position in the currently playing song, uses *TimeWrapper*, default is *%m:%S*)
- *{song_length}* — (Length of the current song, same as song_elapsed)
- *{pos}* — (Position of current song in playlist, one-based)
- *{len}* — (Songs in playlist)
- *{status}* — (play, pause, stop mapped through the *status* dictionary)
- *{bitrate}* — (Current bitrate in kilobit/s)
- *{volume}* — (Volume set in MPD)

Available callbacks

- `switch_playpause` — Plays if paused or stopped, otherwise pauses. Emulates `mpc toggle`.
- `stop` — Stops playback. Emulates `mpc stop`.
- `next_song` — Goes to next track in the playlist. Emulates `mpc next`.
- `previous_song` — Goes to previous track in the playlist. Emulates `mpc prev`.
- `mpd_command` — Send a command directly to MPD's socket. The command is the second element of the list. Documentation for available commands can be found at https://www.musicpd.org/doc/protocol/command_reference.html

Example module registration with callbacks:

```
status.register("mpd",
    on_leftclick="switch_playpause",
    on_rightclick=["mpd_command", "stop"],
    on_middleclick=["mpd_command", "shuffle"],
    on_upscroll=["mpd_command", "seekcur -10"],
    on_downscroll=["mpd_command", "seekcur +10"])
```

Note that `next_song` and `previous_song`, and their `mpd_command` equivalents, are ignored while mpd is stopped.

Settings

- **host** (default: `localhost`)
- **port** (default: `6600`) – MPD port. If set to 0, host will we interpreted as a Unix socket.
- **format** (default: `{title} {status}`) – formatp string
- **status** (default: `{'pause': '','play': '','stop': ''}`) – Dictionary mapping pause, play and stop to output
- **color** (default: `#FFFFFF`) – The color of the text
- **color_map** (default: `{}`) – The mapping from state to color of the text
- **max_field_len** (default: `25`) – Defines max length for in truncate_fields defined fields, if truncated, ellipsis are appended as indicator. It's applied *before* max_len. Value of 0 disables this.
- **max_len** (default: `100`) – Defines max length for the hole string, if exceeding fields specefied in truncate_fields are truncated equaly. If truncated, ellipsis are appended as indicator. It's applied *after* max_field_len. Value of 0 disables this.

- **time_format** (default: `%m:%S`) – format string for 'pos' and 'len' fields
- **truncate_fields** (default: `('title','album','artist','album_artist')`) – fields that will be truncated if exceeding max_field_len or max_len.
- **hide_inactive** (default: `False`) – Hides status information when MPD is not running
- **password** (default: *empty*) – A password for access to MPD. (This is sent in cleartext to the server.)
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `switch_playpause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `next_song`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `previous_song`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.net_speed.`**`NetSpeed`**

Attempts to provide an estimation of internet speeds. Requires: speedtest-cli/modularize-2 speedtest-cli/modularize-2 can be installed using pip: *pip install git+https://github.com/sivel/speedtest-cli.git@modularize-2*

Settings

- **units** (default: `bits`) – Valid values are B, b, bytes, or bits
- **format** (default: `↓{speed_down:.1f}{down_units} ↑{speed_up:.1f}{up_units} ({hosting_provider})`)
- **color** (default: `#FFFFFF`)
- **interval** (default: `300`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

form_b(*n: float*) → tuple
formats a bps as bps/kbps/mbps/gbps etc handles whether its meant to be in bytes :param n: input float :rtype tuple: :return: tuple of float-number of mbps etc, str-units

class `i3pystatus.network.`**Network**
Displays network information for an interface. formatp support if u wanna display recv/send speed separate in dynamic color mode, please enable pango hint.

Requires the PyPI packages *colour*, *netifaces*, *psutil* (optional, see below) and *basiciw* (optional, see below).

Available formatters

Network Information Formatters:

- *{interface}* — same as setting
- *{v4}* — IPv4 address
- *{v4mask}* — subnet mask
- *{v4cidr}* — IPv4 address in cidr notation (i.e. 192.168.2.204/24)
- *{v6}* — IPv6 address
- *{v6mask}* — subnet mask
- *{v6cidr}* — IPv6 address in cidr notation
- *{mac}* — MAC of interface

Wireless Information Formatters (requires PyPI package *basiciw*):

- *{essid}* — ESSID of currently connected wifi
- *{freq}* — Current frequency
- *{freq_divisor}* — Frequency divisor
- *{quality}* — Link quality in percent
- *{quality_bar}* —Bar graphically representing link quality

Network Traffic Formatters (requires PyPI package *psutil*):

- *{interface}* — the configured network interface
- *{network_graph_recv}* – Unicode graph representing incoming network traffic
- *{network_graph_sent}* – Unicode graph representing outgoing network traffic
- *{bytes_sent}* — bytes sent per second (divided by divisor | auto calculated if auto_units == True)
- *{bytes_recv}* — bytes received per second (divided by divisor | auto calculated if auto_units == True)
- *{packets_sent}* — packets sent per second
- *{packets_recv}* — packets received per second
- *{rx_tot_Mbytes}* — total Mbytes received
- *{tx_tot_Mbytes}* — total Mbytes sent
- *{rx_tot}* — total traffic recieved (rounded to nearest unit: KB, MB, GB)
- *{tx_tot}* — total traffic sent (rounded to nearest unit: KB, MB, GB)

Settings

- **format_up** (default: `{interface} {network_graph_recv}{bytes_recv}KB/s`) – format string
- **format_active_up** (default: `{}`) – Dictionary containing format strings for auto-detected interfaces. Each key can be either a full interface name, or a pattern matching a interface, eg 'e*' for ethernet interfaces. Fallback to format_up if no pattern could be matched.
- **format_down** (default: `{interface}: DOWN`) – format string
- **color_up** (default: `#00FF00`)
- **color_down** (default: `#FF0000`)
- **interface** (default: `eth0`) – Interface to watch, eg 'eth0'
- **dynamic_color** (default: `True`) – Set color dynamically based on network traffic. Note: this overrides color_up
- **start_color** (default: `#00FF00`) – Hex or English name for start of color range, eg '#00FF00' or 'green'
- **end_color** (default: `red`) – Hex or English name for end of color range, eg '#FF0000' or 'red'
- **graph_width** (default: `15`) – Width of the network traffic graph
- **graph_style** (default: `blocks`) – Graph style ('blocks', 'braille-fill', 'braille-peak', or 'braille-snake')
- **graph_direction** (default: `left-to-right`) – left-to-right/right-to-left
- **separate_color** (default: `False`) – display recv/send color separate in dynamic color mode.Note: only network speed formatters will display with range color
- **coloring_type** (default: `recv`) – Whether to use the sent or received kb/s for dynamic coloring with non-separate colors. Allowed values 'recv' or 'sent'
- **divisor** (default: `1024`) – divide all byte values by this value
- **recv_limit** (default: `2048`) – Expected max KiB/s. This value controls the drawing color of receive speed
- **sent_limit** (default: `1024`) – Expected max KiB/s. similar with receive_limit
- **freq_divisor** (default: `1`) – divide Wifi frequency by this value

- **ignore_interfaces** (default: `['lo']`) – Array of interfaces to ignore when cycling through on click, eg, ['lo']
- **round_size** (default: `0`) – defines number of digits in round
- **detached_down** (default: `True`) – If the interface doesn't exist, display it as if it were down
- **unknown_up** (default: `False`) – If the interface is in unknown state, display it as if it were up
- **next_if_down** (default: `False`) – Change to next interface if current one is down
- **detect_active** (default: `False`) – Attempt to detect the active interface
- **auto_units** (default: `False`) – if true, unit of measurement is switched automatically (KB/MB/GB/...)
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `nm-connection-editor`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `cycle_interface`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['cycle_interface',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['cycle_interface',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

cycle_interface(*increment=1*)
Cycle through available interfaces in *increment* steps. Sign indicates direction.

class i3pystatus.now_playing.**NowPlaying**
Shows currently playing track information. Supports media players that conform to the Media Player Remote Interfacing Specification.

- Requires `python-dbus` from your distro package manager, or `dbus-python` from PyPI.

Left click on the module to play/pause, and right click to go to the next track.

Available formatters (uses formatp)

- *{title}* — (the title of the current song)
- *{album}* — (the album of the current song, can be an empty string (e.g. for online streams))

- *{artist}* — (can be empty, too)
- *{filename}* — (file name with out extension and path; empty unless title is empty)
- *{song_elapsed}* — (position in the currently playing song, uses *TimeWrapper*, default is *%m:%S*)
- *{song_length}* — (length of the current song, same as song_elapsed)
- *{status}* — (play, pause, stop mapped through the *status* dictionary)
- *{volume}* — (volume)

Available callbacks

- `playpause` — Plays if paused or stopped, otherwise pauses.
- `next_song` — Goes to next track in the playlist.
- `player_command` — Invoke a command with the *MediaPlayer2.Player* interface. The method name and its arguments are appended as list elements.
- `player_prop` — Get or set a property of the *MediaPlayer2.Player* interface. Append the property name to get, or the name and a value to set.

MediaPlayer2.Player methods and properties are documented at https://specifications.freedesktop.org/mpris-spec/latest/Player_Interface.html

Your player may not support the full interface.

Example module registration with callbacks:

```
status.register("now_playing",
    on_leftclick=["player_command", "PlayPause"],
    on_rightclick=["player_command", "Stop"],
    on_middleclick=["player_prop", "Shuffle", True],
    on_upscroll=["player_command", "Seek", -10000000],
    on_downscroll=["player_command", "Seek", +10000000])
```

Settings

- **player** (default: *empty*) – Player name. If not set, compatible players will be detected automatically.
- **status** (default: `{'pause': '','play': '','stop': ''}`) – Dictionary mapping pause, play and stop to output text
- **format** (default: `{title} {status}`) – formatp string
- **color** (default: `#ffffff`) – Text color
- **format_no_player** (default: `No Player`) – Text to show if no player is detected
- **color_no_player** (default: `#ffffff`) – Text color when no player is detected
- **hide_no_player** (default: `True`) – Hide output if no player is detected
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `playpause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)

- **on_upscroll** (default: `volume_up`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `volume_down`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.online.`**Online**

Show internet connection status.

Settings

- **color** (default: `#ffffff`) – Text color when online
- **color_offline** (default: `#ff0000`) – Text color when offline
- **format_online** (default: `online`) – Status text when online
- **format_offline** (default: `offline`) – Status text when offline
- **interval** (default: `10`) – Update interval
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.openfiles.`**Openfiles**
Displays the current/max open files.

Settings

- **filenr_path** (default: `/proc/sys/fs/file-nr`) – Location to file-nr (usually /proc/sys/fs/file-nr
- **color** (default: `FFFFFF`)
- **format** (default: `open/max: {openfiles}/{maxfiles}`)
- **interval** (default: `30`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.openstack_vms.`**Openstack_vms**
Displays the number of VMs in an openstack cluster in ACTIVE and non-ACTIVE states. Requires: python-novaclient

Settings

- **auth_url** (required) – OpenStack cluster authentication URL (OS_AUTH_URL)
- **username** (required) – Username for OpenStack authentication (OS_USERNAME)
- **password** (required) – Password for Openstack authentication (OS_PASSWORD)
- **tenant_name** (required) – Tenant/Project name to view (OS_TENANT_NAME)
- **color** (default: `#00FF00`) – Display color when non-active VMs are =< *threshold*

- **crit_color** (default: `#FF0000`) – Display color when non-active VMs are => *threshold*
- **threshold** (default: `0`) – Set critical indicators when non-active VM pass this number
- **horizon_url** (default: *empty*) – When clicked, open this URL in a browser
- **format** (default: `{tenant_name}: {active_servers} up,{nonactive_servers} down`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `openurl`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.openvpn.`**`OpenVPN`**

Monitor OpenVPN connections.

Note: This module currently only supports systemd. Additionally, as of OpenVPN 2.4 the unit names have changed, as the OpenVPN server and client now have distinct unit files (`openvpn-server@.service` and `openvpn-client@.service`, respectively). Those who have updated to OpenVPN 2.4 will need to manually set the `status_command`, `vpn_up_command`, and `vpn_down_command`.

Formatters:

- {vpn_name} — Same as setting.
- {status} — Unicode up or down symbol.
- {output} — Output of status_command.
- {label} — Label for this connection, if defined.

Settings

•**format** (default: `{vpn_name} {status}`) – Format string

•**color_up** (default: `#00ff00`) – VPN is up

•**color_down** (default: `#FF0000`) – VPN is down

•**status_down** (default:) – Symbol to display when down

•**status_up** (default:) – Symbol to display when up

•**vpn_name** (default: *empty*) – Name of VPN

•**use_new_service_name** (default: `False`) – Use new openvpn service names (openvpn 2.4^)

•**vpn_up_command** (default: `sudo /bin/systemctl start openvpn@%(vpn_name)s.service`) – Command to bring up the VPN - default requires editing /etc/sudoers

•**vpn_down_command** (default: `sudo /bin/systemctl stop openvpn@%(vpn_name)s.service`) – Command to bring up the VPN - default requires editing /etc/sudoers

•**status_command** (default: `bash -c 'systemctl show openvpn@%(vpn_name)s | grep ActiveState=active'`) – command to find out if the VPN is active

•**interval** (default: `5`) – interval in seconds between module updates

•**on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

•**multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

•**hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.pagerduty.`**PagerDuty**

Module to get the current incidents in PD Requires *pypd*

Formatters:

•*{num_incidents}* - current number of incidents unresolved

•*{num_acknowledged_incidents}* - as it sounds

- *{num_triggered_incidents}* - number of unacknowledged incidents

Example:

```
status.register(
    'pagerduty',
    api_key='mah_api_key',
    user_id='LKJ19QW'
)
```

Settings

- **format** (default: `{num_triggered_incidents} triggered {num_acknowledged_incidents} acknowledged`)
- **api_key** (required) – pagerduty api key
- **color** (default: `#AA0000`) – module text color
- **interval** (default: `60`) – refresh interval
- **user_id** (default: *empty*) – your pagerduty user id, shows up in the url when viewing your profile *https://subdomain.pagerduty.com/users/<user_id>*
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.parcel.`**ParcelTracker**

Used to track parcel/shipments.

Supported carriers: DHL, UPS, DPD, Itella

- parcel.UPS("<id_code>")
- parcel.DHL("<id_code>")
- parcel.DPD("<id_code>")

•parcel.Itella("<id_code>"[, "en"|"fi"|"sv"]) Second parameter is language. Requires beautiful soup 4 (bs4) Requires lxml and cssselect.

Settings

- **instance** (required) – Tracker instance, for example `parcel.UPS('your_id_code')`
- **format** (default: `{name}:{progress}`)
- **name** (required)
- **interval** (default: `60`) – interval in seconds between module updates
- **on_leftclick** (default: `open_browser`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.pianobar.`**`Pianobar`**

Shows the title and artist name of the current music

In pianobar config file must be setted the fifo and event_command options (see man pianobar for more information)

For the event_cmd use: https://github.com/jlucchese/pianobar/blob/master/contrib/pianobar-song-i3.sh

Mouse events: - Left click play/pauses - Right click plays next song - Scroll up/down changes volume

Settings

- **format** (default: `{songtitle} -- {songartist}`)
- **songfile** (required) – File generated by pianobar eventcmd
- **ctlfile** (required) – Pianobar fifo file
- **color** (default: `#FFFFFF`) – The color of the text

- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `playpause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `increase_volume`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `decrease_volume`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.ping.`**`Ping`**

This module display the ping value between your computer and a host.

`switch_state` callback can disable the Ping when desired. `host` property can be changed for set a specific host.

Available formatters

- {ping} the ping value in milliseconds.

Settings

- **color** (default: `#FFFFFF`)
- **format** (default: `{ping} ms`)
- **color_disabled** (default: *empty*) – color when disabled
- **color_bad** (default: `#FFFF00`) – color when latency is above threshold
- **color_down** (default: `#FF0000`) – color when ping fail
- **format_disabled** (default: *empty*) – format string when disabled
- **format_down** (default: `down`) – format string when ping fail
- **latency_threshold** (default: `120`) – latency threshold in ms
- **host** (default: `8.8.8.8`) – host to ping
- **interval** (default: `5`) – interval in seconds between module updates

- **on_leftclick** (default: `switch_state`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.plexstatus.`**`Plexstatus`**

Displays what is currently being streamed from your Plex Media Server.

If you dont have an apikey you will need to follow this https://support.plex.tv/hc/en-us/articles/204059436-Finding-your-account-token-X-Plex-Token

Formatters

- *{title}* - title currently being streamed
- *{platform}* - plex recognised platform of the streamer
- *{product}* - plex product name on the streamer (Plex Web, Plex Media Player)
- *{address}* - address of the streamer
- *{streamer_os}* - operating system on the streaming device

Settings

- **format** (default: `{platform}: {title}`)
- **color** (default: `#00FF00`)
- **apikey** (required) – Your Plex API authentication key
- **address** (required) – Hostname or IP address of the Plex Media Server
- **port** (default: `32400`) – Port which Plex Media Server is running on
- **interval** (default: `120`) – Update interval
- **stream_divider** (default: `-`) – divider between stream info when multiple streams are active

- **format_no_streams** (default: *empty*) – String that is shown if nothing is being streamed
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.pomodoro.`**Pomodoro**

This plugin shows Pomodoro timer.

Left click starts/restarts timer. Right click stops it.

Example color settings.

```
color_map = {
    'stopped': '#2ECCFA',
    'running': '#FFFF00',
    'break': '#37FF00'
}
```

Settings

- **sound** (default: *empty*) – Path to sound file to play as alarm. Played by "aplay" utility
- **pomodoro_duration** (default: `1500`) – Working (pomodoro) interval duration in seconds
- **break_duration** (default: `300`) – Short break duration in seconds
- **long_break_duration** (default: `900`) – Long break duration in seconds
- **short_break_count** (default: `3`) – Short break count before first long break
- **format** (default: `{current_pomodoro}/{total_pomodoro} {time}`) – format string, available formatters: current_pomodoro, total_pomodoro, time
- **inactive_format** (default: `Start Pomodoro`) – format string to display when no timer is running
- **color** (default: *empty*) – dictionary containing a mapping of statuses to colours

- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `start`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `stop`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.pulseaudio.`**`PulseAudio`**

Shows volume of default PulseAudio sink (output).

- Requires amixer for toggling mute and incrementing/decrementing volume on scroll.
- Depends on the PyPI colour module - https://pypi.python.org/pypi/colour/0.0.5

Example configuration

The example configuration below uses only unicode to display the volume (tested with otf-font-awesome)

```
status.register(
    "pulseaudio",
    color_unmuted='#aa3300,
    color_muted='#aa0500',
    format_muted='',
    format='{volume_bar}',
    vertical_bar_width=1,
    vertical_bar_glyphs=['  ', ' ', '']
)
```

Available formatters

- *{volume}* — volume in percent (0...100)
- *{db}* — volume in decibels relative to 100 %, i.e. 100 % = 0 dB, 50 % = -18 dB, 0 % = -infinity dB (the literal value for -infinity is -∞)

•*{muted}* — the value of one of the *muted* or *unmuted* settings

•*{volume_bar}* — unicode bar showing volume

•*{selected}* — show the format_selected string if selected sink is the configured one

Settings

•**format** (default: ♪: `{volume}`)

•**format_muted** (default: *empty*) – optional format string to use when muted

•**format_selected** (default: ‘‘ ‘‘) – string used to mark this sink if selected

•**muted** (default: `M`)

•**unmuted** (default: *empty*)

•**color_muted** (default: `#FF0000`)

•**color_unmuted** (default: `#FFFFFF`)

•**step** (default: `5`) – percentage to increment volume on scroll

•**sink** (default: *empty*) – sink name to use, None means pulseaudio default

•**move_sink_inputs** (default: `True`) – Move all sink inputs when we change the default sink

•**bar_type** (default: `vertical`) – type of volume bar. Allowed values are ‘vertical’ or ‘horizontal’

•**multi_colors** (default: `False`) – whether or not to change the color from ‘color_muted’ to ‘color_unmuted’ based on volume percentage

•**vertical_bar_width** (default: `2`) – how many characters wide the vertical volume_bar should be

•**vertical_bar_glyphs** (default: *empty*) – custom array output as vertical bar instead of unicode bars

•**on_leftclick** (default: `pavucontrol`) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: `switch_mute`) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: `increase_volume`) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: `decrease_volume`) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: `change_sink`) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

•**multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

•**hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

context_notify_cb(*context*, *_*)
Checks wether the context is ready

-Queries server information (server_info_cb is called) -Subscribes to property changes on all sinks (update_cb is called)

init()
Creates context, when context is ready context_notify_cb is called

request_update(*context*)
Requests a sink info update (sink_info_cb is called)

server_info_cb(*context*, *server_info_p*, *userdata*)
Retrieves the default sink and calls request_update

sink_info_cb(*context*, *sink_info_p*, *eol*, *_*)
Updates self.output

update_cb(*context*, *t*, *idx*, *userdata*)
A sink property changed, calls request_update

class i3pystatus.pyload.**pyLoad**
Shows pyLoad status

Available formatters

- *{captcha}* — see captcha_true and captcha_false, which are the values filled in for this formatter
- *{progress}* — average over all running downloads
- *{progress_all}* — percentage of completed files/links in queue
- *{speed}* — kilobytes/s
- *{download}* — downloads enabled, also see download_true and download_false
- *{total}* — number of downloads
- *{free_space}* — free space in download directory in gigabytes

Settings

- **address** (default: `http://127.0.0.1:8000`) – Address of pyLoad webinterface
- **format** (default: `{captcha} {progress_all:.1f}% {speed:.1f} kb/s`)
- **captcha_true** (default: `Captcha waiting`)
- **captcha_false** (default: *empty*)
- **download_true** (default: `Downloads enabled`)
- **download_false** (default: `Downloads disabled`)
- **username** (required)
- **password** (required)
- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `open_webbrowser`) – Callback called on left click (see *Callbacks*)

- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.random_password.`**RandomPassword**

Generates a random password and copies it to the clipboard. Useful if you use any password manager and you want to generate a password in the moment and save it later in you manager's database.

Uses *SystemRandom* class as a cryptographically secure pseudo-number generator - <https://docs.python.org/3/library/random.html#random.SystemRandom>

- Requires *xsel* or *xclip* for copying to the clipboard.
- Generates a new password with a left click by default.
- Generates a password with a default length of 12 and with lowercase, uppercase, digits and special symbols.

Available formatters

- *{length}* — length of generated password

Settings

- **format** (default:) – Format string to be displayed in the status bar
- **length** (default: `12`) – Length of the generated password
- **charset** (default: `['lowercase','uppercase','digits','special']`) – Dictionary containing character types to be included in the password
- **cliptool** (default: *empty*) – Currently supports xsel and xclip
- **color** (default: *empty*) – HTML color hex code #RRGGBB
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: `generate_password`) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.reddit.`**Reddit**

This module fetches and displays posts and/or user mail/messages from reddit.com. Left-clicking on the display text opens the permalink/comments page using webbrowser.open() while right-clicking opens the URL of the submission directly. Depends on the Python Reddit API Wrapper (PRAW) <https://github.com/praw-dev/praw>.

PRAW must be configured for this module to work. https://praw.readthedocs.io/en/latest/

Available formatters

- {submission_title}
- {submission_author}
- {submission_points}
- {submission_comments}
- {submission_permalink}
- {submission_url}
- {submission_domain}
- {submission_subreddit}
- {message_unread}
- {message_author}
- {message_subject}
- {message_body}
- {link_karma}
- {comment_karma}

Settings

- **format** (default: `[{submission_subreddit}] {submission_title} ({submission_domain})`) – Format string used for output.
- **username** (default: *empty*) – Reddit username.
- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **subreddit** (default: *empty*) – Subreddit to monitor. Uses frontpage if unspecified.
- **sort_by** (default: `hot`) – 'hot', 'new', 'rising', 'controversial', or 'top'.
- **time_filter** (default: `all`) – 'all', 'day','hour', 'month', 'week', 'year'
- **color** (default: `#FFFFFF`) – Standard color.
- **colorize** (default: `True`) – Enable color change on new message.
- **color_orangered** (default: `#FF4500`) – Color for new messages.
- **mail_brackets** (default: `False`) – Display unread message count in square-brackets.
- **title_maxlen** (default: `80`) – Maximum number of characters to display in title.
- **interval** (default: `300`) – Update interval.
- **status** (default: `{'new_mail': '','no_mail': ''}`) – New message indicator.
- **on_leftclick** (default: `open_permalink`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.redshift.`**Redshift**

Show status and control redshift - http://jonls.dk/redshift/.

This module runs an instance of redshift by itself, since it needs to parse its output, so you should remove redshift/redshift-gtk from your i3 config before using this module.

Requires *redshift* installed.

Available formatters

- *{inhibit}* — show if redshift is currently On or Off (using *toggle_inhibit* callback)
- *{latitude}* — location latitude
- *{longitude}* — location longitude
- *{period}* — current period (Day or Night)
- *{temperature}* — current screen temperature in Kelvin scale (K)

Settings

- **color** (default: `#ffffff`) – Text color
- **error_color** (default: `#ff0000`) – Text color when an error occurs
- **format** (default: `{inhibit} {temperature}K`)
- **format_inhibit** (default: `['On','Off']`) – List of 2 strings for *{inhibit}*, the first is shown when Redshift is On and the second is shown when Off
- **redshift_parameters** (default: `[]`) – List of parameters to pass to redshift binary
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `toggle_inhibit`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

toggle_inhibit()
Enable/disable redshift

class i3pystatus.regex.**Regex**
Simple regex file watcher

The groups of the regex are passed to the format string as positional arguments.

Settings

- **format** (default: `{0}`) – format string used for output
- **regex** (required)
- **file** (required) – file to search for regex matches
- **flags** (default: `0`) – Python.re flags
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.runwatch.`**RunWatch**

Expands the given path using glob to a pidfile and checks if the process ID found inside is valid (that is, if the process is running). You can use this to check if a specific application, such as a VPN client or your DHCP client is running.

Available formatters

- {pid}
- {name}

Settings

- **format_up** (default: `{name}`)
- **format_down** (default: `{name}`)
- **color_up** (default: `#00FF00`)
- **color_down** (default: `#FF0000`)

- **path** (required)
- **name** (required)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.sabnzbd.`**sabnzbd**

Displays the current status of SABnzbd.

A leftclick pauses/resumes downloading. A rightclick opens SABnzbd inside a browser.

Available formatters

- All the first-level parameters from https://sabnzbd.org/wiki/advanced/api#queue (e.g. status, speed, timeleft, spaceleft, eta ...)

Settings

- **format** (default: `{speed} -{timeleft}`) – format string used for output
- **format_paused** (default: `{status}`) – format string used if SABnzbd is paused
- **host** (default: `127.0.0.1`) – address of the server running SABnzbd
- **port** (default: `8080`) – port that SABnzbd is running on
- **api_key** (default: *empty*) – api key of SABnzbd
- **color** (default: `#FFFFFF`) – default color
- **color_paused** (default: `#FF0000`) – color if SABnzbd is paused
- **color_downloading** (default: `#00FF00`) – color if downloading

- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `pause_resume`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `open_browser`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

init()
Initialize the URL used to connect to SABnzbd.

is_downloading()
Return True if downloads are running.

is_paused()
Return True if downloads are currently paused.

open_browser()
Open the URL of SABnzbd inside a browser.

pause_resume()
Toggle between pausing or resuming downloading.

run()
Connect to SABnzbd and get the data.

class i3pystatus.scores.**Scores**
This is a generic score checker, which must use at least one configured *score backend*.

Followed games can be scrolled through with the mouse/trackpad. Left-clicking on the module will refresh the scores, while right-clicking it will cycle through the configured backends. Double-clicking the module with the left button will launch the league-specific (MLB Gameday / NHL GameCenter / etc.) URL for the game. If there is not an active game, double-clicking will launch the league-specific scoreboard URL containing all games for the current day.

Double-clicking with the right button will reset the current backend to the first game in the scroll list. This is useful for quickly switching back to a followed team's game after looking at other game scores.

Scores for the previous day's games will be shown until 10am Eastern Time (US), after which time the current day's games will be shown.

Available formatters

Formatters are set in the backend instances, see the *Score Backends* for more information.

This module supports the *formatp* extended string format syntax. This allows for values to be hidden when they evaluate as False (e.g. when a formatter is blank (an empty string). The default values for the format strings set in the *score backends* (`format_pregame`, `format_in_progress`, etc.) make heavy use of formatp, hiding many formatters when they are blank.

Usage example

```
from i3pystatus import Status
from i3pystatus.scores import mlb, nhl, nba

status = Status()

status.register(
    'scores',
    hints={'markup': 'pango'},
    colorize_teams=True,
    favorite_icon='<span size="small" color="#F5FF00"></span>',
    team_format='abbreviation',
    backends=[
        mlb.MLB(
            teams=['CWS', 'SF'],
            team_format='name',
            format_no_games='No games today :(',
            inning_top='',
            inning_bottom='',
        ),
        nhl.NHL(teams=['CHI']),
        nba.NBA(
            teams=['GSW'],
            all_games=False,
        ),
    ],
)

status.run()
```

To enable colorized team name/city/abbbreviation, `colorize_teams` must be set to `True`. This also requires that i3bar is configured to use Pango, and that the *hints* param is set for the module and includes a `markup` key, as in the example above. To ensure that i3bar is configured to use Pango, the font param in your i3 config file must start with `pango:`. If a `teams` param is not specified for the backend, then all games for the current day will be tracked, and will be ordered by the start time of the game. Otherwise, only games from explicitly-followed teams will be tracked, and will be in the same order as listed. If `ALL` is part of the list, then games from followed teams will be first in the scroll list, followed by all remaining games in order of start time.

Therefore, in the above example, only White Sox and Giants games would be tracked, while in the below example all games would be tracked, with White Sox and Giants games appearing first in the scroll list and the remaining games appearing after them, in order of start time.

```
from i3pystatus import Status
from i3pystatus.scores import mlb

status = Status()
```

```
status.register(
    'scores',
    hints={'markup': 'pango'},
    colorize_teams=True,
    favorite_icon='<span size="small" color="#F5FF00"></span>',
    backends=[
        mlb.MLB(
            teams=['CWS', 'SF', 'ALL'],
            team_colors={
                'NYM': '#1D78CA',
            },
        ),
    ],
)

status.run()
```

Troubleshooting

If the module gets stuck during an update (i.e. the `refresh_icon` does not go away), then the update thread probably encountered a traceback. This traceback will (by default) be logged to `~/.i3pystatus-<pid>` where `<pid>` is the PID of the thread. However, it may be more convenient to manually set the logfile to make the location of the log data reliable and avoid clutter in your home directory. For example:

```
import logging
from i3pystatus import Status
from i3pystatus.scores import mlb, nhl

status = Status(
    logfile='/home/username/var/i3pystatus.log',
)

status.register(
    'scores',
    log_level=logging.DEBUG,
    backends=[
        mlb.MLB(
            teams=['CWS', 'SF'],
            log_level=logging.DEBUG,
        ),
        nhl.NHL(
            teams=['CHI'],
            log_level=logging.DEBUG,
        ),
        nba.NBA(
            teams=['CHI'],
            log_level=logging.DEBUG,
        ),
    ],
)

status.run()
```

Note: The `log_level` must be set separately in both the module and the backend instances (as shown above),

otherwise the backends will still use the default log level.

Settings

- **backends** (default: `[]`) – List of backend instances
- **interval** (default: `300`) – Update interval (in seconds)
- **favorite_icon** (default:) – Value for the `{away_favorite}` and `{home_favorite}` formatter when the displayed game is being played by a followed team
- **color** (default: *empty*) – Color to be used for non-colorized text (defaults to the i3bar color)
- **color_no_games** (default: *empty*) – Color to use when no games are scheduled for the currently-displayed backend (defaults to the i3bar color)
- **colorize_teams** (default: `False`) – Dislay team city, name, and abbreviation in the team's color (as defined in the *backend*'s `team_colors` attribute)
- **scroll_arrow** (default:) – Value used for the `{scroll}` formatter to indicate that more than one game is being tracked for the currently-displayed backend
- **refresh_icon** (default:) – Text to display (in addition to any text currently shown by the module) when refreshing scores. **NOTE:** Depending on how quickly the update is performed, the icon may not be displayed.
- **team_format** (default: `name`) – One of `name`, `abbreviation`, or `city`
- **on_leftclick** (default: `['check_scores','click event']`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `['cycle_backend',1]`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['scroll_game',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['scroll_game',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: `['launch_web']`) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: `['reset_backend']`) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.scratchpad.**Scratchpad**
Display the amount of windows and indicate urgency hints on scratchpad (async).

fork from scratchpad_async of py3status by cornerman

Requires the PyPI package *i3ipc*.

Available formaters

- *{number}* — amount of windows on scratchpad

@author jok @license BSD

Settings

- **format** (default: `{number} `) – format string.
- **always_show** (default: `True`) – whether the indicator should be shown if there are no scratchpad windows
- **color_urgent** (default: `#900000`) – color of urgent
- **color** (default: `#FFFFFF`) – text color
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.sensu.**SensuCheck**
Pool sensu api events

Available formatters

- {status} OK if not events else numbers of events
- {last_event} Display the output of the most recent event (with priority on error event)

Settings

- **api_url** (required) – URL of Sensu API. e.g: http://localhost/sensu/
- **api_username** (default: *empty*)
- **api_password** (default: *empty*)
- **format** (default: `{status}`)
- **color_error** (default: `#ff0000`)
- **color_warn** (default: `#f9ba46`)
- **color_ok** (default: `#00ff00`)
- **last_event_label** (default: `Last:`) – Label to put before the last event output (default 'Last:')
- **max_event_field** (default: `50`) – Defines max length of the last_event message field (default: 50)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.sge.`**SGETracker**

Used to display status of Batch computing jobs on a cluster running Sun Grid Engine. The data is collected via ssh, so a valid ssh address must be specified.

Requires lxml.

Settings

- **ssh** (required) – The SSH connection address. Can be user@host or user:password@host or user@host -p PORT etc.
- **color** (default: `#ffffff`)
- **format** (default: `SGE qw: {queued} / r: {running} / Eqw: {error}`)
- **interval** (default: `60`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.shell.`**`Shell`**

Shows output of shell command

Available formatters

- *{output}* — just the striped command output without newlines

Settings

- **command** (required) – command to be executed
- **ignore_empty_stdout** (default: `False`) – Let the block be empty
- **color** (default: `#FFFFFF`) – standard color
- **error_color** (default: `#FF0000`) – color to use when non zero exit code is returned
- **format** (default: `{output}`)
- **interval** (default: `5`) – interval in seconds between module updates

- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.solaar.`**Solaar**

Shows status and load percentage of logitech's unifying device

Available formatters

- *{output}* — percentage of battery and status

Settings

- **nameOfDevice** (required) – name of the logitech's unifying device
- **color** (default: `#FFFFFF`) – standard color
- **error_color** (default: `#FF0000`) – color to use when non zero exit code is returned
- **interval** (default: `30`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.sonos.`**Sonos**

Controls and displays information from Sonos devices.

A devices is found by IP, by name, or automatically if no IP or name is supplied.

Available formatters

- *{player_name}* — player name
- *{volume}* — volume from 0 to 100
- *{muted}* — "M" if muted, else ""
- *{title}* — the title of the current song
- *{artist}* — the artist of the current song
- *{album}* — the album of the current song
- *{duration}* — duration of the current song (%M:%S)
- *{position}* — position in the current song (%M:%S)
- *{state}* — Playing, Paused, Stopped

Requires: soco (can be installed using pip)

Settings

- **ip** (default: *empty*) – Speaker IP address.
- **name** (default: *empty*) – Speaker name (used if no IP is given).
- **format** (default: `{state}: {artist} -{title} [{muted}{volume:.0f}%]`) – Format used when playing or paused.
- **color** (default: `#FFFFFF`) – Color used when playing or paused.
- **format_no_music** (default: `No music`) – Format used when stopped.
- **color_no_music** (default: `#888888`) – Color used when stopped.
- **format_no_connection** (default: `No connection`) – Format used if no player is connected.
- **color_no_connection** (default: `#888888`) – Color used if no player is connected.
- **hide_no_connection** (default: `False`) – Hide output if no player is connected.

- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `play_pause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: `toggle_mute`) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `incr_vol`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `decr_vol`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: `next_song`) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.spaceapi.`**SpaceAPI**

Show if a hackerspace is open

Available formatters

- {state}
- {message}
- {lastchange}

Settings

- **url** (required) – spaceapi endpoint
- **format** (default: `S: {state}`) – format string used for output.
- **color_open** (default: `#00FF00`) – color if hackerspace is opened
- **color_closed** (default: `#FF0000`) – color if hackerspace is closed
- **interval** (default: `10`) – update interval
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.spotify.`**Spotify**

Get Spotify info using dbus interface. Based on *now_playing* module.

Settings

- **player** (default: `spotify`) – Player name. If not set, compatible players will be detected automatically.
- **status** (default: `{'pause': '','play': '','stop': ''}`) – Dictionary mapping pause, play and stop to output text
- **format** (default: `{title} {status}`) – formatp string
- **color** (default: `#ffffff`) – Text color
- **format_no_player** (default: `No Player`) – Text to show if no player is detected
- **color_no_player** (default: `#ffffff`) – Text color when no player is detected
- **hide_no_player** (default: `True`) – Hide output if no player is detected
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `playpause`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `next_song`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `volume_up`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `volume_down`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.swap.`**Swap**

Shows swap load

Available formatters

- {free}
- {percent_used}
- {used}
- {total}

Requires psutil (from PyPI)

Settings

- **format** (default: `{free} MiB`) – format string used for output.
- **format_no_swap** (default: `No swap`) – format string used when no swap is enabled, set to None to use default format
- **hide_if_empty** (default: `False`) – hide swap block when swap is not used
- **divisor** (default: `1048576`) – divide all byte values by this value, default is 1024**2 (megabytes)
- **warn_percentage** (default: `50`) – minimal percentage for warn state
- **alert_percentage** (default: `80`) – minimal percentage for alert state
- **color** (default: `#00FF00`) – standard color
- **warn_color** (default: `#FFFF00`) – defines the color used when warn percentage is exceeded
- **alert_color** (default: `#FF0000`) – defines the color used when alert percentage is exceeded
- **color_no_swap** (default: `#FFFFFF`) – defines the color used when no swap is enabled, set to None to use default color
- **round_size** (default: `1`) – defines number of digits in round
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.syncthing.`**Syncthing**

Check Syncthing's online status and start/stop Syncthing via click events.

Requires *requests*.

Settings

- **format_up** (default: `ST up`) – Text to show when Syncthing is running
- **format_down** (default: `ST down`) – Text to show when Syncthing is not running
- **color_up** (default: `#00ff00`) – Color when Syncthing is running
- **color_down** (default: `#ff0000`) – Color when Syncthing is not running
- **configfile** (default: `~/.config/syncthing/config.xml`) – Path to Syncthing config
- **url** (default: `auto`) – Syncthing GUI URL; "auto" reads from local config
- **apikey** (default: `auto`) – Syncthing APIKEY; "auto" reads from local config
- **verify_ssl** (default: `True`) – Verify SSL certificate
- **interval** (default: `10`) – interval in seconds between module updates
- **on_leftclick** (default: `st_open`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `st_toggle_systemd`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

st_open()
Callback: Open Syncthing web UI

st_restart()
Callback: Restart Syncthing

st_restart_systemd()
Callback: systemctl –user restart syncthing.service

st_start_systemd()
Callback: systemctl –user start syncthing.service

st_stop()
Callback: Stop Syncthing

st_stop_systemd()
Callback: systemctl –user stop syncthing.service

st_toggle_systemd()
Callback: start Syncthing service if offline, or stop it when online

class i3pystatus.taskwarrior.**Taskwarrior**
Check Taskwarrior for pending tasks Requires *json*

Available formatters (uses formatp)

- *{ready}* — contains number of tasks returned by *ready_filter*
- *{urgent}* — contains number of tasks returned by *urgent_filter*
- *{next}* — contains the description of next task
- *{project}* — contains the projects the next task belongs to

Available callbacks

- `get_next_task` — Display the next most urgent task.
- `get_prev_task` — Display the previous most urgent task.
- `reset_next_task` — Display the most urgent task, resetting any switching by other callbacks.

Settings

- **format** (default: `Task: {next}`) – format string
- **ready_filter** (default: `+READY`) – Filters to get ready tasks example: *+READY*
- **urgent_filter** (default: `+TODAY`) – Filters to get urgent tasks example: *+TODAY*
- **enable_mark_done** (default: `False`) – Enable right click mark task as done

- **color_urgent** (default: `#FF0000`) – #FF0000
- **color_ready** (default: `#78EAF2`) – #78EAF2
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `reset_next_task`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `mark_task_as_done`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `get_prev_task`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `get_next_task`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.temp.`**Temperature**

Shows CPU temperature of Intel processors.

AMD is currently not supported as they can only report a relative temperature, which is pretty useless.

Requires *colour* module from PyPi

Modes of operation

If lm_sensors_enabled is set to False, the module operates in default mode. This means that:

- only the {temp} formatter is available
- alert_temp is honored

If lm_sensors_enabled is set to True, the module operates in lm_sensors mode. This means that:

- pysensors must be installed (https://github.com/bastienleonard/pysensors)
- CPU sensors are discovered dynamically (supporting a sensor per core and multiple CPUs)
- alert_temp is ignored. The warning or critical values reported by the sensor are used instead (see urgent_on)

lm_sensors installation

In order to take advantage of the lm_sensors library and tools, it must first be installed and configured.

On Arch this is as simple as:

```
pacman -S lm_sensors
```

On Ubuntu:

```
sudo apt-get update && sudo apt-get install lm-sensors libsensors4-dev
```

The Arch Wiki has a good page on the library - https://wiki.archlinux.org/index.php/lm_sensors

lm_sensors_mode formatters

When `lm_sensors_enabled` is True the formatters are created dynamically. In order to discover the formatters that are available, it is best to run the sensors command:

```
 sensors
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +48.0°C  (high = +80.0°C, crit = +99.0°C)
Core 0:         +48.0°C  (high = +80.0°C, crit = +99.0°C)
Core 1:         +46.0°C  (high = +80.0°C, crit = +99.0°C)
Core 2:         +43.0°C  (high = +80.0°C, crit = +99.0°C)
Core 3:         +47.0°C  (high = +80.0°C, crit = +99.0°C)
```

The module replaces spaces in sensor names with underscores, therefore from the above output we can identify the following sensor formatters:

- Physical_id_0
- Core_0
- Core_1
- Core_2
- Core_3

For each sensor a vertical bar is also generated. In this example we would also have the following bars:

- Physical_id_0_bar
- Core_0_bar
- Core_1_bar
- Core_2_bar
- Core_3_bar

Thus, this format string would be valid: "{Physical_id_0}°C {Core_0_bar}{Core_1_bar}{Core_2_bar}{Core_3_bar}"

Pango Markup and lm_sensors_mode

When Pango Markup is enabled and `dynamic_color` is True, each sensor's formatter color is displayed independently. The color is determined by the proximity of the sensors current value to it's critical value.

Example Configuration

Here is an example configuration based on the sensor values discovered above:

```
status.register("temp",
                format="{Physical_id_0}°C {Core_0_bar}{Core_1_bar}{Core_2_bar}
↪{Core_3_bar}",
                hints={"markup": "pango"},
                lm_sensors_enabled=True,
                dynamic_color=True)
```

Settings

- **format** (default: `{temp} °C`) – format string used for output. {temp} is the temperature in degrees celsius
- **display_if** (default: `True`) – snippet that gets evaluated. if true, displays the module output
- **lm_sensors_enabled** (default: `False`) – whether or not lm_sensors should be used for obtaining CPU temperature information
- **urgent_on** (default: `warning`) – whether to flag as urgent when temperature exceeds urgent value or critical value (requires lm_sensors_enabled)
- **dynamic_color** (default: `False`) – whether to set the color dynamically (overrides alert_color)
- **color** (default: `#FFFFFF`)
- **file** (default: `/sys/class/thermal/thermal_zone0/temp`)
- **alert_temp** (default: `90`)
- **alert_color** (default: `#FF0000`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

format_sensor(*sensor*)
Format a sensor value. If pango is enabled color is per sensor.

format_sensor_bar(*sensor*)
Build and format a sensor bar. If pango is enabled bar color is per sensor.

get_output_original()
Build the output the original way. Requires no third party libraries.

get_output_sensors()
Build the output using lm_sensors. Requires sensors Python module (see docs).

get_urgent(*sensors*)
Determine if any sensors should set the urgent flag.

class `i3pystatus.teslacharge.`**TeslaCharge**
Displays the current charge/range of your Tesla vehicle. There is a ton of data that could be displayed, so read this module to see the full list of datapoints that can be displayed. Requires: teslapy

Settings

•**email** (required) – Email address used to login to your Tesla account.

•**password** (required) – Password for accessing your Tesla account.

•**units** (default: `miles`) – Miles (default) or km

•**interval** (default: `900`) – Poll frequency. Polling too often can drain the battery

•**format_with_charge** (default: `{name}: {charge_state_icon}{charge_rate} {battery_level}%/{charge_limit_soc}% ({battery_range})`) – Display format while car is charging

•**format_without_charge** (default: `{name}: {battery_level}%/{charge_limit_soc}% ({battery_range})`) – Display format while car is not charging

•**on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)

•**on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)

•**on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)

•**on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)

•**on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

•**on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)

•**on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

•**on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)

•**on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)

•**on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

•**on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

•**on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)

•**on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)

•**multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.

- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.text.`**`Text`**
Display static, colored text.

Settings

- **text** (required)
- **color** (default: *empty*) – HTML color code #RRGGBB
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.ticker.`**`Ticker`**
Displays stock ticker information from yahoo finance Requires: yfinance

Settings

- **symbol** (required) – Stock symbol to track
- **interval** (default: `300`) – Update interval (in seconds)
- **good_color** (default: `#00FF00`) – Color of text while 'regularMarketPrice' is above good_threshold
- **bad_color** (default: `#FF0000`) – Color of text while 'regularMarketPrice' is below bad_threshold
- **caution_color** (default: `#FFFF00`) – Color of text while 'regularMarketPrice' is between good and bad thresholds
- **good_threshold** (default: `100`) – The target value for consindering the stock a good value
- **bad_threshold** (default: `50`) – The target value for consindering the stock a poor value

- **format** (default: `{symbol}: {regularMarketPrice} ({regularMarketDayHigh}/{regularMarketD`
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.timer.`**Timer**

Timer module to remind yourself that there probably is something else you should be doing right now.

Main features include:

- Set custom time interval with click events.
- Different output formats triggered when remaining time is less than *x* seconds.
- Execute custom python function or external command when timer overflows (or reaches zero depending on how you look at it).

Available formatters

Time formatters are available to show the remaining time. These include `%h`, `%m`, `%s`, `%H`, `%M`, `%S`. See *`TimeWrapper`* for detailed description.

The `format_custom` setting allows you to display different formats when certain amount of seconds is remaining. This setting accepts list of tuples which contain time in seconds, format string and color string each. See the default settings for an example:

- `(0,"+%M:%S","#ffffff")` - Use this format after overflow. White text with red background set by the urgent flag.
- `(60,"-%M:%S","#ffa500")` - Change color to orange in last minute.
- `(3600,"-%M:%S","#00ff00")` - Hide hour digits when remaining time is less than one hour.

Only first matching rule is applied (if any).

Callbacks

Module contains three mouse event callback methods:

- *start()* - Default: Left click starts (or adds) 5 minute countdown.
- *increase()* - Default: Upscroll/downscroll increase/decrease time by 1 minute.
- *reset()* - Default: Right click resets timer.

Two new event settings were added:

- on_overflow - Executed when remaining time reaches zero.
- on_reset - Executed when timer is reset but only if overflow occurred.

These settings accept either a python callable object or a string with shell command. Python callbacks should be non-blocking and without any arguments.

Here is an example that plays a short sound file in 'loop' every 60 seconds until timer is reset. (play is part of SoX - the Swiss Army knife of audio manipulation)

```
on_overflow = "play -q /path/to/sound.mp3 pad 0 60 repeat -"
on_reset = "pkill -SIGTERM -f 'play -q /path/to/sound.mp3 pad 0 60 repeat -'"
```

Settings

- **format** (default: `-%h:%M:%S`) – Default format that is showed if no `format_custom` rules are matched.
- **format_stopped** (default: `T`) – Format showed when timer is inactive.
- **color** (default: `#00ff00`)
- **color_stopped** (default: `#ffffff`)
- **format_custom** (default: `[(0,'+%M:%S','#ffffff'),(60,'-%M:%S','#ffa500'),(3600,'-%M:%S','#`
- **overflow_urgent** (default: `True`) – Set urgent flag on overflow.
- **on_overflow** (default: *empty*)
- **on_reset** (default: *empty*)
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `['start',300]`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `reset`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['increase',60]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['increase',-60]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

increase(*seconds*)
Change remaining time value.

Parameters **seconds** (*int*) – Seconds to add. Negative value subtracts from remaining time.

reset()
Stop timer and execute `on_reset` if overflow occurred.

start(*seconds=300*)
Starts timer. If timer is already running it will increase remaining time instead.

Parameters **seconds** (*int*) – Initial time.

class i3pystatus.timewarrior.**Timewarrior**
Show current Timewarrior tracking Requires *json dateutil*

Formaters:

- *{tags}* — contains tags of current track
- *{start}* - contains start of track
- *{duration}* — contains time of current track

Settings

- **format** (default: `{duration}`) – format string
- **duration_format** (default: `{years}y{months}m{days}d{hours}h{minutes}m{seconds}s`) – duration format string
- **enable_stop** (default: `True`) – Allow right click to stop tracking
- **enable_continue** (default: `True`) – ALlow right click to continue tracking
- **color_running** (default: `#00FF00`) – #00FF00
- **color_stopped** (default: `#F00000`) – #F00000
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `stop_or_continue`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)

- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.tlp.`**`Tlp`**

Shows the current mode of TLP (Linux power management tool), either battery, AC or unknown.

Available formatters

- *{output}* - one of the strings configured through the **_text* settings

Settings

- **last_pwr_file** (default: `/run/tlp/last_pwr`) – path to the TLP 'last pwr' file, default is */run/tlp/last_pwr*
- **bat_color** (default: `#00FF00`) – color of text when TLP is in battery mode
- **ac_color** (default: `#FFAA00`) – color of text when TLP is in AC mode
- **na_color** (default: `#FF0000`) – color of text when TLP is in unknown mode
- **bat_text** (default: `BAT`) – text to show when TLP is in battery mode
- **ac_text** (default: `AC`) – text to show when TLP is in AC mode
- **na_text** (default: `N/A`) – text to show when TLP is in unknown mode
- **format** (default: `{output}`)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)

- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.travisci.`**`TravisCI`**

Get current status of travis builds Requires *travispy dateutil.parser*

Formatters:

- *{repo_slug}* - repository owner/repository name
- *{repo_status}* - repository status
- *{repo_name}* - repository name
- *{repo_owner}* - repository owner
- *{last_build_finished}* - date of the last finished build
- *{last_build_duration}* - duration of the last build

Examples

```
status_color_map = {
    'passed': '#00FF00',
    'failed': '#FF0000',
    'errored': '#FFAA00',
    'cancelled': '#EEEEEE',
    'started': '#0000AA',

}
```

```
repo_status_map={
    'passed': '<span color="#00af00">passed</span>',
    'started': '<span color="#0000af">started</span>',
    'failed': '<span color="#af0000">failed</span>',
}
```

Settings

- **format** (default: `{repo_owner}/{repo_name}-{repo_status} [({last_build_finished}({last_build_duration}))]`)
- **github_token** (required) – github personal access token
- **repo_slug** (required) – repository identifier eg. “enkore/i3pystatus”
- **time_format** (default: `%m/%d`) – passed directly to .strftime() for *last_build_finished*
- **repo_status_map** (default: *empty*) – map representing how to display status
- **duration_format** (default: `%m:%S`) – *last_build_duration* format string
- **status_color_map** (default: *empty*) – color for all text based on status

- **color** (default: `#DDDDDD`) – color for all text not otherwise colored
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `open_build_webpage`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.uname.`**`Uname`**

uname(1) like module.

Available formatters

- *{sysname}* — operating system name
- *{nodename}* — name of machine on network (implementation-defined)
- *{release}* — operating system release
- *{version}* — operating system version
- *{machine}* — hardware identifier

Settings

- **format** (default: `{sysname} {release}`) – format string used for output
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)

- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.updates.`**Updates**

Generic update checker. To use select appropriate backend(s) for your system. For list of all available backends see *Update Backends*.

Left clicking on the module will refresh the count of upgradeable packages. This may be used to dismiss the notification after updating your system.

Right clicking shows a desktop notification with a summary count and a list of available updates.

Available formatters

- *{count}* — Sum of all available updates from all backends.
- For each backend registered there is one formatter named after the backend, multiple identical backends do not accumulate, but overwrite each other.
- For example, *{Cower}* (note capital C) is the number of updates reported by the cower backend, assuming it has been registered.

Usage example

```
from i3pystatus import Status
from i3pystatus.updates import pacman, cower

status = Status()

status.register("updates",
                format = "Updates: {count}",
                format_no_updates = "No updates",
                backends = [pacman.Pacman(), cower.Cower()])

status.run()
```

Settings

- **backends** (required) – Required list of backends used to check for updates.

- **format** (default: `Updates: {count}`) – Format used when updates are available. May contain formatters.
- **format_no_updates** (default: *empty*) – String that is shown if no updates are available. If not set the module will be hidden if no updates are available.
- **format_working** (default: *empty*) – Format used while update queries are run. By default the same as `format`.
- **format_summary** (default: *empty*) – Format for the summary line of notifications. By default the same as `format`.
- **notification_icon** (default: `software-update-available`) – Icon shown when reporting the list of updates. Default is `software-update-available`, and can be None for no icon.
- **color** (default: `#00DD00`)
- **color_no_updates** (default: *empty*)
- **color_working** (default: *empty*)
- **interval** (default: `3600`) – Default interval is set to one hour.
- **on_leftclick** (default: `run`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: `report`) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.uptime.`**Uptime**

Outputs Uptime

Available formatters

- *{days}* - uptime in days
- *{hours}* - rest of uptime in hours
- *{mins}* - rest of uptime in minutes

- *{secs}* - rest of uptime in seconds
- *{uptime}* - deprecated: equals '*{hours}:{mins}*'

Settings

- **format** (default: `up {hours}:{mins}`) – Format string
- **color** (default: `#ffffff`) – String color
- **alert** (default: `False`) – If you want the string to change color
- **seconds_alert** (default: `2592000`) – How many seconds necessary to start the alert
- **color_alert** (default: `#ff0000`) – Alert color
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.vk.`**Vk**

Display amount of unread messages in VK social network. Creating your own VK API app is highly recommended for your own privacy, though there is a default one provided. Reference vk.com/dev for instructions on creating VK API app. If access_token is not specified, the module will try to open a request page in browser. You will need to manually copy obtained acess token to your config file. Requires the PyPI package *vk*.

Settings

- **app_id** (default: `5160484`) – Id of your VK API app
- **access_token** (default: *empty*) – Your access token. You must have *messages* and *offline* access permissions

- **token_error** (default: `Vk: token error`) – Message to be shown if there's some problem with your token
- **color** (default: `#ffffff`) – General color of the output
- **color_bad** (default: `#ff0000`) – Color of the output in case of access token error
- **color_unread** (default: `#ffffff`) – Color of the output if there are unread messages
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.weather.`**Weather**

This is a generic weather-checker which must use a configured weather backend. For list of all available backends see *Weather Backends*.

Double-clicking on the module will launch the forecast page for the location being checked, and single-clicking will trigger an update.

Available formatters

- *{city}* — Location of weather observation
- *{condition}* — Current weather condition (Rain, Snow, Overcast, etc.)
- *{icon}* — Icon representing the current weather condition
- *{observation_time}* — Time of weather observation (supports strftime format flags)
- *{current_temp}* — Current temperature, excluding unit
- *{low_temp}* — Forecasted low temperature, excluding unit
- *{high_temp}* — Forecasted high temperature, excluding unit (may be empty in the late afternoon)
- *{temp_unit}* — Either `°C` or `°F`, depending on whether metric or

- *{feelslike}* — "Feels Like" temperature, excluding unit
- *{dewpoint}* — Dewpoint temperature, excluding unit imperial units are being used
- *{wind_speed}* — Wind speed, excluding unit
- *{wind_unit}* — Either `kph` or `mph`, depending on whether metric or imperial units are being used
- *{wind_direction}* — Wind direction
- *{wind_gust}* — Speed of wind gusts in mph/kph, excluding unit
- *{pressure}* — Barometric pressure, excluding unit
- *{pressure_unit}* — `mb` or `in`, depending on whether metric or imperial units are being used
- *{pressure_trend}* — + if rising, − if falling, or an empty string if the pressure is steady (neither rising nor falling)
- *{visibility}* — Visibility distance, excluding unit
- *{visibility_unit}* — Either `km` or `mi`, depending on whether metric or imperial units are being used
- *{humidity}* — Current humidity, excluding percentage symbol
- *{uv_index}* — UV Index
- *{update_error}* — When the configured weather backend encounters an error during an update, this formatter will be set to the value of the backend's **update_error** config value. Otherwise, this formatter will be an empty string.

This module supports the *formatp* extended string format syntax. This allows for values to be hidden when they evaluate as False. The default **format** string value for this module makes use of this syntax to conditionally show the value of the **update_error** config value when the backend encounters an error during an update.

See the following links for usage examples for the available weather backends:

- *Weather.com*
- *Weather Underground*

Troubleshooting

If an error is encountered while updating, the `{update_error}` formatter will be set, and (provided it is in your `format` string) will show up next to the forecast to alert you to the error. The error message will (by default be logged to `~/.i3pystatus-<pid>` where `<pid>` is the PID of the update thread. However, it may be more convenient to manually set the logfile to make the location of the log data predictable and avoid clutter in your home directory. Additionally, using the `DEBUG` log level can be helpful in revealing why the module is not working as expected. For example:

```
import logging
from i3pystatus import Status
from i3pystatus.weather import weathercom

status = Status(logfile='/home/username/var/i3pystatus.log')

status.register(
    'weather',
    format='{condition} {current_temp}{temp_unit}[ {icon}][ Hi: {high_temp}][ Lo:
↪{low_temp}][ {update_error}]',
    colorize=True,
    hints={'markup': 'pango'},
```

```
    update_error='<span color="#ff0000">!</span>',
    log_level=logging.DEBUG,
    backend=weathercom.Weathercom(
        location_code='94107:4:US',
        units='imperial',
        log_level=logging.DEBUG,
    ),
)
```

Note: The log level must be set separately in both the module and backend contexts.

Settings

- **colorize** (default: `False`) – Vary the color depending on the current conditions.
- **color_icons** (default: `{'Fair': ('','#ffcc00'),'Fog': ('','#949494'),'Cloudy': ('','#f8f8ff'),'Partly Cloudy': ('','#f8f8ff'),'Rainy': ('','#cbd2c0'),'Thunderstorm': ('','#cbd2c0'),'Sunny': ('','#ffff00'),'Snow': ('','#ffffff'),'default': ('',None)}`) – Dictionary mapping weather conditions to tuples containing a UTF-8 code for the icon, and the color to be used.
- **color** (default: *empty*) – Display color (or fallback color if `colorize` is True). If not specified, falls back to default i3bar color.
- **backend** (required) – Weather backend instance
- **refresh_icon** (default:) – Text to display (in addition to any text currently shown by the module) when refreshing weather data. **NOTE:** Depending on how quickly the update is performed, the icon may not be displayed.
- **online_interval** (default: *empty*) – seconds between updates when online (defaults to interval)
- **offline_interval** (default: `300`) – seconds between updates when offline (default: 300)
- **format** (default: `{current_temp}{temp_unit}[ {update_error}]`)
- **interval** (default: `1800`) – interval in seconds between module updates
- **on_leftclick** (default: `['check_weather']`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: `['launch_web']`) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)

- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

check_weather()
Check the weather using the configured backend

get_color_data(*condition*)
Disambiguate similarly-named weather conditions, and return the icon and color that match.

class i3pystatus.weekcal.**WeekCal**
Displays the days of the current week as they would be represented on a calendar sheet, with the current day highlighted. By default, the current day of week is displayed in the front, and the month and year are displayed in the back.

Example: `Sat 16 17 18 19 20[21]22 May 2016`

Settings

- **startofweek** (default: `0`) – First day of the week (0 = Monday, 6 = Sunday), defaults to 0.
- **prefixformat** (default: `%a`) – Prefix in strftime-format
- **suffixformat** (default: `%b %Y`) – Suffix in strftime-format
- **todayhighlight** (default: `('[',']')`) – Characters to highlight today's date
- **interval** (default: `30`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.whosonlocation.**WOL**
Change your whosonlocation.com status.

Requires the PyPi module *beautifulsoup4*

Settings

- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **email** (default: *empty*)
- **password** (default: *empty*)
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: `change_status`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.window_title.**WindowTitle**
Display the current window title with async update. Uses asynchronous update via i3 IPC events. Provides instant title update only when it required.

fork from window_tile_async of py3status by Anon1234 https://github.com/Anon1234

Requires the PyPI package *i3ipc*.

Available formaters

- *{title}* — title of current focused window
- *{class_name}* - name of application class

@author jok @license BSD

Settings

- **format** (default: `{title}`) – format string.
- **always_show** (default: `False`) – do not hide the title when it can be already visible
- **empty_title** (default: *empty*) – string that will be shown instead of the title when the title is hidden
- **max_width** (default: `79`) – maximum width of title
- **color** (default: `#FFFFFF`) – text color
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.wireguard.`**`Wireguard`**

Monitor Wireguard connections.

Note: You might want to add something like this to /etc/sudoers:

```
%wheel ALL = NOPASSWD: /bin/systemctl start wg-quick@wg0.service,/bin/systemctl
stop wg-quick@wg0.service
```

Formatters:

- {vpn_name} — Same as setting.
- {status} — Unicode up or down symbol.
- {output} — Output of status_command.
- {label} — Label for this connection, if defined.

Settings

- **format** (default: `{vpn_name} {status}`) – Format string
- **color_up** (default: `#00ff00`) – VPN is up
- **color_down** (default: `#FF0000`) – VPN is down
- **status_down** (default:) – Symbol to display when down
- **status_up** (default:) – Symbol to display when up
- **vpn_name** (default: *empty*) – Name of VPN
- **vpn_up_command** (default: `sudo /bin/systemctl start wg-quick@{vpn_name}.service`) – Command to bring up the VPN - default requires editing /etc/sudoers
- **vpn_down_command** (default: `sudo /bin/systemctl stop wg-quick@{vpn_name}.service`) – Command to bring up the VPN - default requires editing /etc/sudoers
- **status_command** (default: `systemctl is-active wg-quick@{vpn_name}`) – command to find out if the VPN is active
- **interval** (default: `5`) – interval in seconds between module updates
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.xkblayout.`**`Xkblayout`**

Displays and changes current keyboard layout.

`change_layout` callback finds the current layout in the `layouts` setting and enables the layout following it. If the current layout is not in the `layouts` setting the first layout is enabled.

`layouts` can be stated with or without variants, e.g.: `status.register("xkblayout",layouts=["de neo","de"])`

Requires xkbgroup (from PyPI)

Available formatters

- *{num}* — current group number
- *{name}* — current group name
- *{symbol}* — current group symbol
- *{variant}* — current group variant
- *{count}* — number of all groups
- *{names}* — names of all groups
- *{symbols}* — symbols of all groups
- *{variants}* — variants of all groups

Settings

- **color** (default: `#FFFFFF`) – RGB hexadecimal color code specifuer, defaults to #FFFFFF
- **format** (default: `{symbol}`) – Format string
- **layouts** (default: `[]`) – List of layouts
- **uppercase** (default: `True`) – Flag for uppercase output
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `['change_layout',1]`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: `['change_layout',1]`) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: `['change_layout',-1]`) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.yubikey.`**Yubikey**

This module allows you to lock and unlock your Yubikey in order to avoid the OTP to be triggered accidentally.

@author Daniel Theodoro <daniel.theodoro AT gmail.com>

Settings

- **format** (default: `Yubikey: `) – Format string
- **unlocked_format** (default: `Yubikey: `) – Format string when the key is unlocked
- **timeout** (default: `5`) – How long the Yubikey will be unlocked (default: 5)
- **color** (default: `#00FF00`) – Standard color
- **unlock_color** (default: `#FF0000`) – Set the color used when the Yubikey is unlocked
- **interval** (default: `1`) – interval in seconds between module updates
- **on_leftclick** (default: `['set_lock',True]`) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.zabbix.`**Zabbix**

Zabbix alerts watcher

Requires: pyzabbix

Available formatters

- {default} - Full output count alerts like total:a5/a4/a3/a2/a1/a0
- {total} - Total count of alerts
- {aX_count} - Count alerts of X severity
- {colorX} - Predicted color for X severity. It can be used with Pango markup hint for different colours at each severity with

Settings

- **zabbix_server** (required) – Zabbix Server URL
- **zabbix_user** (required) – Zabbix API User
- **zabbix_password** (required) – Zabbix users password
- **interval** (default: `60`) – Update interval
- **groups** (default: *empty*) – Provide groupids(e.g ['102', '10'])
- **filter** (default: *empty*) – Provide API-Filter(e.g {'status': '1'})
- **min_severity** (default: `2`) – Specify min severity (0-5)
- **format** (default: `{default}`)
- **on_leftclick** (default: *empty*) – Callback called on left click (see *Callbacks*)
- **on_middleclick** (default: *empty*) – Callback called on middle click (see *Callbacks*)
- **on_rightclick** (default: *empty*) – Callback called on right click (see *Callbacks*)
- **on_upscroll** (default: *empty*) – Callback called on scrolling up (see *Callbacks*)
- **on_downscroll** (default: *empty*) – Callback called on scrolling down (see *Callbacks*)
- **on_doubleleftclick** (default: *empty*) – Callback called on double left click (see *Callbacks*)
- **on_doublemiddleclick** (default: *empty*) – Callback called on double middle click (see *Callbacks*)
- **on_doublerightclick** (default: *empty*) – Callback called on double right click (see *Callbacks*)
- **on_doubleupscroll** (default: *empty*) – Callback called on double scroll up (see *Callbacks*)
- **on_doubledownscroll** (default: *empty*) – Callback called on double scroll down (see *Callbacks*)
- **on_otherclick** (default: *empty*) – Callback called on other click (see *Callbacks*)
- **on_doubleotherclick** (default: *empty*) – Callback called on double other click (see *Callbacks*)
- **on_change** (default: *empty*) – Callback called when output is changed (see *Callbacks*)
- **multi_click_timeout** (default: `0.25`) – Time (in seconds) before a single click is executed.
- **hints** (default: `{'markup': 'none'}`) – Additional output blocks for module output (see *Hints*)
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

3.1 Mail Backends

The generic mail module can be configured to use multiple mail backends. Here is an example configuration for the MaildirMail backend:

```
from i3pystatus.mail import maildir
status.register("mail",
                backends=[maildir.MaildirMail(
                        directory="/home/name/Mail/inbox")
                ],
                format="P {unread}",
                log_level=20,
                hide_if_null=False, )
```

- *ews*
- *imap*
- *maildir*
- *mbox*
- *notmuchmail*
- *thunderbird*

class i3pystatus.mail.ews.**ExchangeMailAccount**
Checks for mail on an Exchange account.

Requires the python exchangelib library - https://github.com/ecederstrand/exchangelib.

Settings

- **host** (default: *empty*) – The url to connect to. If unset, autodiscover is tried with the email address domain. If set, autodiscover is disabled.
- **username** (required)
- **password** (required)
- **email_address** (required)
- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **account** (default: *empty*) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.mail.imap.**IMAP**
Checks for mail on a IMAP server

Settings

- **host** (required)
- **port** (default: `993`)
- **username** (required)
- **password** (required)
- **keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials
- **ssl** (default: `True`)
- **mailbox** (default: `INBOX`)
- **account** (default: `Default account`) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

imap_class
alias of `IMAP4`

class i3pystatus.mail.maildir.**MaildirMail**
Checks for local mail in Maildir

Settings

- **directory** (default: *empty*)
- **account** (default: `Default account`) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mail.mbox.`**`MboxMail`**
Checks for local mail in mbox

Settings

- **account** (default: `Default account`) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mail.notmuchmail.`**`Notmuch`**
This class uses the notmuch python bindings to check for the number of messages in the notmuch database with the tags "inbox" and "unread"

Settings

- **db_path** (default: *empty*) – Path to the directory of your notmuch database
- **query** (default: `tag:unread and tag:inbox`) – Same query notmuch would accept, by default 'tag:unread and tag:inbox'
- **account** (default: `Default account`) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.mail.thunderbird.`**`Thunderbird`**
This class listens for dbus signals emitted by the dbus-sender extension for thunderbird.

Requires python-dbus

Settings

- **account** (default: `Default account`) – Account name
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

3.2 Score Backends

- *mlb*
- *nba*
- *nhl*

class `i3pystatus.scores.mlb.`**`MLB`**
Backend to retrieve MLB scores. For usage examples, see *here*.

Available formatters

- *{home_team}* — Depending on the value of the `team_format` option, will contain either the home team's name, abbreviation, or city
- *{home_score}* — Home team's current score
- *{home_wins}* — Home team's number of wins
- *{home_losses}* — Home team's number of losses
- *{home_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the home team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{away_team}* — Depending on the value of the `team_format` option, will contain either the away team's name, abbreviation, or city
- *{away_score}* — Away team's current score
- *{away_wins}* — Away team's number of wins
- *{away_losses}* — Away team's number of losses
- *{away_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the away team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{top_bottom}* — Displays the value of either `inning_top` or `inning_bottom` based on whether the game is in the top or bottom of an inning.
- *{inning}* — Current inning
- *{outs}* — Number of outs in current inning
- *{venue}* — Name of ballpark where game is being played
- *{start_time}* — Start time of game in system's localtime (supports strftime formatting, e.g. *{start_time:%I:%M %p}*)
- *{delay}* — Reason for delay, if game is currently delayed. Otherwise, this formatter will be blank.
- *{postponed}* — Reason for postponement, if game has been postponed. Otherwise, this formatter will be blank.
- *{suspended}* — Reason for suspension, if game has been suspended. Otherwise, this formatter will be blank.
- *{extra_innings}* — When a game lasts longer than 9 innings, this formatter will show that number of innings. Otherwise, it will blank.

Team abbreviations

- **ARI** — Arizona Diamondbacks
- **ATL** — Atlanta Braves
- **BAL** — Baltimore Orioles
- **BOS** — Boston Red Sox
- **CHC** — Chicago Cubs
- **CIN** — Cincinnati Reds
- **CLE** — Cleveland Guardians

- **COL** — Colorado Rockies
- **CWS** — Chicago White Sox
- **DET** — Detroit Tigers
- **HOU** — Houston Astros
- **KC** — Kansas City Royals
- **LAA** — Los Angeles Angels of Anaheim
- **LAD** — Los Angeles Dodgers
- **MIA** — Miami Marlins
- **MIL** — Milwaukee Brewers
- **MIN** — Minnesota Twins
- **NYY** — New York Yankees
- **NYM** — New York Mets
- **OAK** — Oakland Athletics
- **PHI** — Philadelphia Phillies
- **PIT** — Pittsburgh Pirates
- **SD** — San Diego Padres
- **SEA** — Seattle Mariners
- **SF** — San Francisco Giants
- **STL** — St. Louis Cardinals
- **TB** — Tampa Bay Rays
- **TEX** — Texas Rangers
- **TOR** — Toronto Blue Jays
- **WSH** — Washington Nationals

Settings

- **favorite_teams** (default: `[]`) – List of abbreviations of favorite teams. Games for these teams will appear first in the scroll list. A detailed description of how games are ordered can be found *here*.
- **all_games** (default: `True`) – If set to `True`, all games will be present in the scroll list. If set to `False`, then only games from **favorite_teams** will be present in the scroll list.
- **display_order** (default: `['in_progress','suspended','final','pregame','postponed']`) – When **all_games** is set to `True`, this option will dictate the order in which games from teams not in **favorite_teams** are displayed
- **format_no_games** (default: `MLB: No games`) – Format used when no tracked games are scheduled for the current day (does not support formatter placeholders)
- **format** (default: `[{scroll} ]MLB: [{away_favorite} ]{away_team} [{away_score} ]({away_wins}-{away_losses}) at [{home_favorite} ]{home_team} [{home_score} ]({home_wins}-{home_losses}) {game_status}`) – Format used to display game information

- **status_pregame** (default: `{start_time:%H:%M %Z}[ ({delay} Delay)]`) – Format string used for the `{game_status}` formatter when the game has not started
- **status_in_progress** (default: `({top_bottom} {inning},{outs} Out)[ ({delay} Delay)]`) – Format string used for the `{game_status}` formatter when the game is in progress
- **status_final** (default: `(Final[/{extra_innings}])`) – Format string used for the `{game_status}` formatter when the game has finished
- **status_postponed** (default: `(PPD: {postponed})`) – Format string used for the `{game_status}` formatter when the game has been postponed
- **status_suspended** (default: `(Suspended: {suspended})`) – Format string used for the `{game_status}` formatter when the game has been suspended
- **inning_top** (default: `Top`) – Value for the `{top_bottom}` formatter when game is in the top half of an inning
- **inning_bottom** (default: `Bot`) – Value for the `{top_bottom}` formatter when game is in the bottom half of an inning
- **team_colors** (default: `{'ARI': '#A71930','ATL': '#CE1141','BAL': '#DF4601','BOS': '#BD3039','CHC': '#004EC1','CIN': '#C6011F','CLE': '#E31937','COL': '#5E5EB6','CWS': '#DADADA','DET': '#FF6600','HOU': '#EB6E1F','KC': '#0046DD','LAA': '#BA0021','LAD': '#005A9C','MIA': '#00A3E0','MIL': '#0747CC','MIN': '#D31145','NYY': '#0747CC','NYM': '#FF5910','OAK': '#006659','PHI': '#E81828','PIT': '#FFCC01','SD': '#FFC425','SEA': '#2E8B90','SF': '#FD5A1E','STL': '#B53B30','TB': '#8FBCE6','TEX': '#C0111F','TOR': '#0046DD','WSH': '#C70003'}`) – Dictionary mapping team abbreviations to hex color codes. If overridden, the passed values will be merged with the defaults, so it is not necessary to define all teams if specifying this value.
- **team_format** (default: *empty*) – One of `name`, `abbreviation`, or `city`. If not specified, takes the value from the `scores` module.
- **date** (default: *empty*) – Date for which to display game scores, in **YYYY-MM-DD** format. If unspecified, the current day's games will be displayed starting at 10am Eastern time, with last evening's scores being shown before then. This option exists primarily for troubleshooting purposes.
- **live_url** (default: `https://www.mlb.com/gameday/{id}`) – Alternate URL string to launch MLB Gameday. This value should not need to be changed
- **scoreboard_url** (default: `http://m.mlb.com/scoreboard`) – Link to the MLB.com scoreboard page. Like **live_url**, this value should not need to be changed.
- **api_url** (default: `https://statsapi.mlb.com/api/v1/schedule?sportId=1,51&date={date:%Y-%m-`
  – Alternate URL string from which to retrieve score data. Like **live_url*** this value should not need to be changed.
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.scores.nba.`**NBA**

Backend to retrieve NBA scores. For usage examples, see *here*.

Available formatters

- *{home_team}* — Depending on the value of the `team_format` option, will contain either the home team's name, abbreviation, or city
- *{home_score}* — Home team's current score

- *{home_wins}* — Home team's number of wins
- *{home_losses}* — Home team's number of losses
- *{home_seed}* — During the playoffs, shows the home team's playoff seed. When not in the playoffs, this formatter will be blank.
- *{home_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the home team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{away_team}* — Depending on the value of the `team_format` option, will contain either the away team's name, abbreviation, or city
- *{away_score}* — Away team's current score
- *{away_wins}* — Away team's number of wins
- *{away_losses}* — Away team's number of losses
- *{away_seed}* — During the playoffs, shows the away team's playoff seed. When not in the playoffs, this formatter will be blank.
- *{away_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the away team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{time_remaining}* — Time remaining in the current quarter/OT period
- *{quarter}* — Number of the current quarter
- *{start_time}* — Start time of game in system's localtime (supports strftime formatting, e.g. *{start_time:%I:%M %p}*)
- *{overtime}* — If the game ended in overtime, this formatter will show `OT`. If the game ended in regulation, or has not yet completed, this formatter will be blank.

Team abbreviations

- **ATL** — Atlanta Hawks
- **BKN** — Brooklyn Nets
- **BOS** — Boston Celtics
- **CHA** — Charlotte Hornets
- **CHI** — Chicago Bulls
- **CLE** — Cleveland Cavaliers
- **DAL** — Dallas Mavericks
- **DEN** — Denver Nuggets
- **DET** — Detroit Pistons
- **GSW** — Golden State Warriors
- **HOU** — Houston Rockets
- **IND** — Indiana Pacers
- **MIA** — Miami Heat
- **MEM** — Memphis Grizzlies
- **MIL** — Milwaukee Bucks

- **LAC** — Los Angeles Clippers
- **LAL** — Los Angeles Lakers
- **MIN** — Minnesota Timberwolves
- **NOP** — New Orleans Pelicans
- **NYK** — New York Knicks
- **OKC** — Oklahoma City Thunder
- **ORL** — Orlando Magic
- **PHI** — Philadelphia 76ers
- **PHX** — Phoenix Suns
- **POR** — Portland Trailblazers
- **SAC** — Sacramento Kings
- **SAS** — San Antonio Spurs
- **TOR** — Toronto Raptors
- **UTA** — Utah Jazz
- **WAS** — Washington Wizards

Settings

- **favorite_teams** (default: `[]`) – List of abbreviations of favorite teams. Games for these teams will appear first in the scroll list. A detailed description of how games are ordered can be found *here*.
- **all_games** (default: `True`) – If set to `True`, all games will be present in the scroll list. If set to `False`, then only games from **favorite_teams** will be present in the scroll list.
- **display_order** (default: `['in_progress','final','pregame','postponed']`) – When **all_games** is set to `True`, this option will dictate the order in which games from teams not in **favorite_teams** are displayed
- **format_no_games** (default: `NBA: No games`) – Format used when no tracked games are scheduled for the current day (does not support formatter placeholders)
- **format** (default: `[{scroll} ]NBA: [{away_favorite} ][{away_seed} ]{away_team} [{away_score} ]({away_wins}-{away_losses}) at [{home_favorite} ][{home_seed} ]{home_team} [{home_score} ]({home_wins}-{home_losses}) {game_status}`) – Format used to display game information
- **status_pregame** (default: `{start_time:%H:%M %Z}`) – Format string used for the `{game_status}` formatter when the game has not started
- **status_in_progress** (default: `({time_remaining} {quarter})`) – Format string used for the `{game_status}` formatter when the game is in progress
- **status_final** (default: `(Final[/{overtime}])`) – Format string used for the `{game_status}` formatter when the game has finished
- **status_postponed** (default: `PPD`) – Format string used for the `{game_status}` formatter when the game has been postponed

- **team_colors** (default: `{'ATL': '#E2383F','BKN': '#DADADA','BOS': '#178D58','CHA': '#00798D','CHI': '#CD1041','CLE': '#FDBA31','DAL': '#006BB7','DEN': '#5593C3','DET': '#207EC0','GSW': '#DEB934','HOU': '#CD1042','IND': '#FFBB33','MIA': '#A72249','MEM': '#628BBC','MIL': '#4C7B4B','LAC': '#ED174C','LAL': '#FDB827','MIN': '#35749F','NOP': '#A78F59','NYK': '#F68428','OKC': '#F05033','ORL': '#1980CB','PHI': '#006BB7','PHX': '#E76120','POR': '#B03037','SAC': '#7A58A1','SAS': '#DADADA','TOR': '#CD112C','UTA': '#4B7059','WAS': '#E51735'}`) – Dictionary mapping team abbreviations to hex color codes. If overridden, the passed values will be merged with the defaults, so it is not necessary to define all teams if specifying this value.
- **team_format** (default: *empty*) – One of `name`, `abbreviation`, or `city`. If not specified, takes the value from the `scores` module.
- **date** (default: *empty*) – Date for which to display game scores, in **YYYY-MM-DD** format. If unspecified, the current day's games will be displayed starting at 10am Eastern time, with last evening's scores being shown before then. This option exists primarily for troubleshooting purposes.
- **live_url** (default: `https://www.nba.com/game/{id}`) – URL string to launch NBA Game Tracker. This value should not need to be changed.
- **api_url** (default: `https://cdn.nba.com/static/json/liveData/scoreboard/todaysScoreboard_00` – Alternate URL string from which to retrieve score data. Like, **live_url**, this value should not need to be changed.
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class `i3pystatus.scores.nhl.`**NHL**

Backend to retrieve NHL scores. For usage examples, see *here*.

Available formatters

- *{home_team}* — Depending on the value of the `team_format` option, will contain either the home team's name, abbreviation, or city
- *{home_score}* — Home team's current score
- *{home_wins}* — Home team's number of wins
- *{home_losses}* — Home team's number of losses
- *{home_otl}* — Home team's number of overtime losses
- *{home_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the home team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{home_empty_net}* — Shows the value from the `empty_net` parameter when the home team's net is empty.
- *{away_team}* — Depending on the value of the `team_format` option, will contain either the away team's name, abbreviation, or city
- *{away_score}* — Away team's current score
- *{away_wins}* — Away team's number of wins
- *{away_losses}* — Away team's number of losses
- *{away_otl}* — Away team's number of overtime losses

- *{away_favorite}* — Displays the value for the *scores* module's `favorite` attribute, if the away team is one of the teams being followed. Otherwise, this formatter will be blank.
- *{away_empty_net}* — Shows the value from the `empty_net` parameter when the away team's net is empty.
- *{period}* — Current period
- *{venue}* — Name of arena where game is being played
- *{start_time}* — Start time of game in system's localtime (supports strftime formatting, e.g. *{start_time:%I:%M %p}*)
- *{overtime}* — If the game ended in overtime or a shootout, this formatter will show `OT` kor `SO`. If the game ended in regulation, or has not yet completed, this formatter will be blank.

Playoffs

In the playoffs, losses are not important (as the losses will be equal to the other team's wins). Therefore, it is a good idea during the playoffs to manually set format strings to exclude information on team losses. For example:

```
from i3pystatus import Status
from i3pystatus.scores import nhl

status = Status()
status.register(
    'scores',
    hints={'markup': 'pango'},
    colorize_teams=True,
    favorite_icon='<span size="small" color="#F5FF00"></span>',
    backends=[
        nhl.NHL(
            favorite_teams=['CHI'],
            format='[{scroll} ]NHL: [{away_favorite} ]{away_team} ({away_wins})␣
↪at [{home_favorite} ]{home_team} ({home_wins}) {game_status}'
        ),
    ],
)
```

Team abbreviations

- **ANA** — Anaheim Ducks
- **ARI** — Arizona Coyotes
- **BOS** — Boston Bruins
- **BUF** — Buffalo Sabres
- **CAR** — Carolina Hurricanes
- **CBJ** — Columbus Blue Jackets
- **CGY** — Calgary Flames
- **CHI** — Chicago Blackhawks
- **COL** — Colorado Avalanche

- **DAL** — Dallas Stars
- **DET** — Detroit Red Wings
- **EDM** — Edmonton Oilers
- **FLA** — Florida Panthers
- **LAK** — Los Angeles Kings
- **MIN** — Minnesota Wild
- **MTL** — Montreal Canadiens
- **NJD** — New Jersey Devils
- **NSH** — Nashville Predators
- **NYI** — New York Islanders
- **NYR** — New York Rangers
- **OTT** — Ottawa Senators
- **PHI** — Philadelphia Flyers
- **PIT** — Pittsburgh Penguins
- **SEA** — Seattle Kraken
- **SJS** — San Jose Sharks
- **STL** — St. Louis Blues
- **TBL** — Tampa Bay Lightning
- **TOR** — Toronto Maple Leafs
- **VAN** — Vancouver Canucks
- **VGK** — Vegas Golden Knights
- **WPG** — Winnipeg Jets
- **WSH** — Washington Capitals

Settings

- **favorite_teams** (default: `[]`) – List of abbreviations of favorite teams. Games for these teams will appear first in the scroll list. A detailed description of how games are ordered can be found *here*.
- **all_games** (default: `True`) – If set to `True`, all games will be present in the scroll list. If set to `False`, then only games from **favorite_teams** will be present in the scroll list.
- **display_order** (default: `['in_progress','final','pregame','postponed']`) – When **all_games** is set to `True`, this option will dictate the order in which games from teams not in **favorite_teams** are displayed
- **format_no_games** (default: `NHL: No games`) – Format used when no tracked games are scheduled for the current day (does not support formatter placeholders)
- **format** (default: `[{scroll} ]NHL: [{away_favorite} ]{away_team} [{away_score} ]({away_wins}-{away_losses}-{away_otl}) at [{home_favorite} ]{home_team} [{home_score} ]({home_wins}-{home_losses}-{home_otl}) {game_status}`) – Format used to display game information

•**status_pregame** (default: `{start_time:%H:%M %Z}`) – Format string used for the `{game_status}` formatter when the game has not started

•**status_in_progress** (default: `({time_remaining} {period})`) – Format string used for the `{game_status}` formatter when the game is in progress

•**status_final** (default: `(Final[/{overtime}])`) – Format string used for the `{game_status}` formatter when the game has finished

•**status_postponed** (default: `PPD`) – Format string used for the `{game_status}` formatter when the game has been postponed

•**empty_net** (default: `EN`) – Value for the `{away_empty_net}` or `{home_empty_net}` formatter when the net is empty. When the net is not empty, these formatters will be empty strings.

•**team_colors** (default: `{'ANA': '#B4A277','ARI': '#AC313A','BOS': '#F6BD27','BUF': '#1568C5','CAR': '#FA272E','CBJ': '#1568C5','CGY': '#D23429','CHI': '#CD0E24','COL': '#9F415B','DAL': '#058158','DET': '#E51937','EDM': '#2F6093','FLA': '#E51837','LAK': '#DADADA','MIN': '#176B49','MTL': '#C8011D','NJD': '#CC0000','NSH': '#FDB71A','NYI': '#F8630D','NYR': '#1576CA','OTT': '#C50B2F','PHI': '#FF690B','PIT': '#FFB81C','SEA': '#96D8D8','SJS': '#007888','STL': '#1764AD','TBL': '#296AD5','TOR': '#296AD5','VAN': '#0454FA','VGK': '#B4975A','WPG': '#1568C5','WSH': '#E51937'}`) – Dictionary mapping team abbreviations to hex color codes. If overridden, the passed values will be merged with the defaults, so it is not necessary to define all teams if specifying this value.

•**team_format** (default: *empty*) – One of `name`, `abbreviation`, or `city`. If not specified, takes the value from the `scores` module.

•**date** (default: *empty*) – Date for which to display game scores, in **YYYY-MM-DD** format. If unspecified, the current day's games will be displayed starting at 10am Eastern time, with last evening's scores being shown before then. This option exists primarily for troubleshooting purposes.

•**live_url** (default: `https://www.nhl.com/gamecenter/{id}`) – URL string to launch NHL GameCenter. This value should not need to be changed.

•**scoreboard_url** (default: `https://www.nhl.com/scores`) – Link to the NHL.com scoreboard page. Like **live_url**, this value should not need to be changed.

•**api_url** (default: `https://statsapi.web.nhl.com/api/v1/schedule?startDate={date:%Y-%m-%d}&`) – Alternate URL string from which to retrieve score data. Like **live_url**, this value should not need to be changed.

•**log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

3.3 Update Backends

- *aptget*
- *auracle*
- *cower*
- *dnf*
- *packagekit*
- *pacman*
- *yaourt*

- *yay*

class i3pystatus.updates.aptget.**AptGet**
Gets update count for Debian based distributions.

This mimics the Arch Linux *checkupdates* script but with apt-get and written in python.

Settings

•**log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.auracle.**Auracle**
Checks for updates in Arch User Repositories using the *auracle* AUR helper.

Depends on auracle AUR agent - https://github.com/falconindy/auracle

Settings

•**log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.cower.**Cower**
Checks for updates in Arch User Repositories using the *cower* AUR helper.

Depends on cower AUR agent - https://github.com/falconindy/cower

Settings

•**log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.dnf.**Dnf**
Gets updates for RPM-based distributions using the DNF API

The notification body consists of the package name and version for each available update.

Note: Users running i3pystatus from a virtualenv may see the updates display as `?` due to an inability to import the `dnf` module. To ensure that i3pystatus can access the DNF Python bindings, the virtualenv should be created with `--system-site-packages`.

If using pyenv-virtualenv, the virtualenv must additionally be created to use the system Python binary:

```
$ pyenv virtualenv --system-site-packages --python=/usr/bin/python3 pyenv_name
```

To invoke i3pystatus with this virtualenv, your `bar` section in `~/.config/i3/config` would look like this:

```
bar {
    position top
    status_command PYENV_VERSION=pyenv_name python /path/to/i3pystatus/script.py
}
```

Settings

- **log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.packagekit.**PackageKit**
Gets update count for distributions using PackageKit.

At the moment, it works with english localization, only.

Settings

- **log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.pacman.**Pacman**
Checks for updates in Arch Linux repositories using the *checkupdates* script which is part of the *pacman-contrib* package.

Settings

- **log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.yaourt.**Yaourt**
This module counts the available updates using yaourt. By default it will only count aur packages. Thus it can be used with the pacman backend like this:

```
from i3pystatus.updates import pacman, yaourt
status.register("updates", backends = [pacman.Pacman(), yaourt.Yaourt()])
```

To count both pacman and aur packages, pass False in the constructor:

```
from i3pystatus.updates import yaourt
status.register("updates", backends = [yaourt.Yaourt(False)])
```

Settings

- **log_level** (default: 30) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.updates.yay.**Yay**
This module counts the available updates using yay. By default it will only count aur packages. Thus it can be used with the pacman backend like this:

```
from i3pystatus.updates import pacman, yay
status.register("updates", backends = [pacman.Pacman(), yay.Yay()])
```

To count both pacman and aur packages, pass False in the constructor:

```
from i3pystatus.updates import yay
status.register("updates", backends = [yay.Yay(False)])
```

Settings

- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

3.4 Weather Backends

- *weathercom*
- *wunderground*

class `i3pystatus.weather.weathercom.`**`Weathercom`**

This module gets the weather from weather.com. The `location_code` parameter should be set to the location code from weather.com. To obtain this code, search for your location on weather.com, and when you go to the forecast page, the code you need will be everything after the last slash in the URL (e.g. `94107:4:US`, or `8b7867695971473a260df2c5d49ff92dc9079dcb673c545f5f107f5c4ab30732`).

Usage example

```
from i3pystatus import Status
from i3pystatus.weather import weathercom

status = Status(logfile='/home/username/var/i3pystatus.log')

status.register(
    'weather',
    format='{condition} {current_temp}{temp_unit}[ {icon}][ Hi: {high_temp}][ Lo:
↪{low_temp}][ {update_error}]',
    interval=900,
    colorize=True,
    hints={'markup': 'pango'},
    backend=weathercom.Weathercom(
        location_code='94107:4:US',
        units='imperial',
        update_error='<span color="#ff0000">!</span>',
    ),
)

status.run()
```

See *here* for a list of formatters which can be used.

Settings

- **location_code** (required) – Location code from www.weather.com
- **units** (default: `metric`) – 'metric' or 'imperial'
- **update_error** (default: `!`) – Value for the `{update_error}` formatter when an error is encountered while checking weather data
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

`check_weather``()`

Fetches the current weather from wxdata.weather.com service.

class i3pystatus.weather.wunderground.**Wunderground**
This module retrieves weather data from Weather Underground.

Note: Previous versions of this module required an API key to work. Weather Underground has since discontinued their API, and this module has been rewritten to reflect that.

Finding your weather station

To use this module, you must provide a weather station code (as the `location_code` option). To find your weather station, first search for your city and click to view the current conditions. Below the city name you will see the station name, and to the right of that a `CHANGE` link. Clicking that link will display a map, where you can find the station closest to you. Clicking on that station will take you back to the current conditions page. The weather station code will now be the last part of the URL. For example:

```
https://www.wunderground.com/weather/us/ma/cambridge/KMACAMBR4
```

In this case, the weather station code would be `KMACAMBR4`.

Usage example

```
from i3pystatus import Status
from i3pystatus.weather import wunderground

status = Status(logfile='/home/username/var/i3pystatus.log')

status.register(
    'weather',
    format='{condition} {current_temp}{temp_unit}[ {icon}][ Hi: {high_temp}][ Lo:
↪{low_temp}][ {update_error}]',
    colorize=True,
    hints={'markup': 'pango'},
    backend=wunderground.Wunderground(
        location_code='KMACAMBR4',
        units='imperial',
        update_error='<span color="#ff0000">!</span>',
    ),
)

status.run()
```

See *here* for a list of formatters which can be used.

Settings

- **location_code** (required) – Location code from wunderground.com
- **units** (default: `metric`) – 'metric' or 'imperial'
- **update_error** (default: `!`) – Value for the `{update_error}` formatter when an error is encountered while checking weather data
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

check_weather()
Query the desired station and return the weather data

get_api_key()
Grab the API key out of the page source from the home page

3.5 Calendar Backends

Generic calendar interface. Requires the PyPI package `colour`.

Available formatters

- {title} - the title or summary of the event
- {remaining_time} - how long until this event is due

Additional formatters may be provided by the backend, consult their documentation for details.

Settings

- {update_interval} - how often (in seconds) the calendar backend should be called to update events
- {dynamic_color} - when set, the color shifts as the event approaches
- {urgent_blink} - when set, urgent is toggled every second when within urgent_seconds of the event
- {urgent_seconds} - how many seconds before the event to begin blinking
- {skip_recurring} - when set, recurring events are skipped

Here is an example of configuring the calendar module to use the `Lightning` backend:

```
status.register("calendar",
            format="{title} {remaining}",
            update_interval=10,
            urgent_blink=True,
            backend=Lightning(database_path=path, days=2))
```

- *google*
- *khal_calendar*
- *lightning*

class i3pystatus.calendar.google.**Google**
Calendar backend for interacting with Google Calendar.

Requires the Google Calendar API package - https://developers.google.com/google-apps/calendar/quickstart/python. Additionally requires the *colour*, *httplib2*, *oauth2client*, *pytz*, *google-api-python-client* and *dateutil* modules.

The first time this module is ran, you will need to specify the location of *credentials.json* (as credentials_json) acquired from: https://developers.google.com/google-apps/calendar/quickstart/python this will open a browser window for auth, and save a token to *credential_path*. you will need to reload i3poystatus afterwards

If you already have a token *credentials_json* is not required (though highly recommended incase your token gets broken)

Available formatters

- *{kind}* — type of event
- *{status}* — eg, confirmed
- *{htmlLink}* — link to the calendar event

Settings

- **credential_path** (required) – Path to save credentials to (auto generated the first time this module is ran)
- **credentials_json** (default: *empty*) – path to credentials.json (generated by google)
- **days** (default: `7`) – Only show events between now and this many days in the future
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

get_timerange_formatted(*now*)
Return two ISO8601 formatted date strings, one for timeMin, the other for timeMax (to be consumed by get_events)

refresh_events()
Retrieve the next N events from Google.

class i3pystatus.calendar.khal_calendar.**Khal**
Backend for Khal. Requires *khal* to be installed.

Available formatters * *{calendar}* — Calendar event is from.

Settings

- **config_path** (default: *empty*) – Path to your khal.conf
- **calendars** (default: *empty*) – Restrict to these calendars pass as a list)
- **days** (default: `7`) – Check for the next X days
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

class i3pystatus.calendar.lightning.**Lightning**
Backend for querying the Thunderbird's Lightning database. Requires *pytz* and *dateutil*.

Available formatters

- *{location}* — Where the event occurs

Settings

- **database_path** (required) – Path to local.sqlite.
- **days** (default: `7`) – Only show events between now and this many days in the future
- **log_level** (default: `30`) – Set to true to log error to .i3pystatus-<pid> file.

CHAPTER 4

Changelog

No further releases are planned. Install it from Git.

4.1 3.35 (2016-08-31)

- New modules
  - `google_calendar`: Displays next Google Calendar event
  - *`openfiles`*: Report open files count
  - *`ping`*: Display ping time to host
  - *`scores`*: Display sport scores
  - *`scratchpad`*: Display number of windows and urgency hints on i3 scratchpad
  - *`taskwarrior`*: Pending tasks in taskwarrior
  - *`wunderground`*: Similar to *`weather`*, but uses wunderground
  - *`zabbix`*: Zabbix alerts watcher
- `i3pystatus` binary now takes an optional path to a config file
  - (purely optional, doesn't change any existing configurations)
- Fixed a bug with required settings (did only occur in develoment branch)
- *`clock`*: timezone-related fixes with multiple clocks
- *`dpms`*: Added format_disabled option
- *`github`*: Added support for acccess tokens
- *`gpu_temp`*: Added display_if setting
- *`mail.imap`*: Add support for IDLE if imaplib2 is installed
- *`mpd`*: Bug fixes
- *`network`*: Bug fixes. Upgrading to `netifaces>=0.10.5` is recommended for avoiding IPv6-related bugs (disabling IPv6 is of course also a well-working solution)
- *`now_playing`*: Also check activatable D-Bus services, bug fixes
- *`openvpn`*: Added support for toggling connection on click
- *`pomodoro`*: Bug fixes

- *pulseaudio*: Display/control active sink, bug fixes
- *reddit*: Fixes for praw
- *temp*: Added display_if setting
- *updates*: Added dnf (rpm-based distros) backend
- updates: Added notification support with summary of all available updates
- *weather*: Added color_icons option, bug fixes
- *xkblayout*: Bug fixes

4.2 3.34 (2016-02-14)

- **New modules**
  - *moon*: Display moon phase
  - *online*: Display internet connectivity
  - *xkblayout*: View and change keyboard layout
  - *plexstatus*: View status of Plex Media Server
  - *iinet*: View iiNet internet usage
  - *gpu_mem*, *gpu_temp*: View memory and temperature stats of nVidia cards
  - *solaar*: Show battery status of Solaar / Logitech Unifying devices
  - *zabbix*: Alerts watcher for the Zabbix enterprise network monitor
  - *sge*: Sun Grid Engine (SGE) monitor
  - *timer*: Timer
  - *syncthing*: Syncthing monitor and control
  - *vk*: Displays number of messages in VKontakte
- Applications started from click events don't block other click events now
- Fixed crash with desktop notifications when python-gobject is installed, but no notification daemon is running
- Log file name is now an option (`logfile` of *Status*)
- Server used for checking internet connectivity is now an option (`internet_check` of *Status*)
- Added double click support for click events
- Formatter data is now available with most modules for program callbacks
- Changed default mode to standalone mode
- `self` is not passed anymore by default to external Python callbacks (see *get_module()*)
- *dota2wins*: Now accepts usernames in place of a Steam ID
- dota2wins: Changed win percentage to be a float
- *uptime*: Added days, hours, minutes, secs formatters
- *battery*: Added alert command feature (runs a shell command when the battery is discharged below a preset threshold)
- *spotify*: Added status, format_not_running and color_not_running settings, rewrite

- *cmus*: Added status, format_not_running and color_not_running settings
- *cmus*: Fixed bug that sometimes lead to empty output
- *shell*: Added formatting capability
- *cpu_usage*: Added color setting
- *mpd*: Added hide_inactive settings
- mpd: Fixed a bug where an active playlist would be assumed, leading to no output
- mpd: Added support for UNIX sockets
- *updates*: Added yaourt backend
- updates: Can display a working/busy message now
- updates: Additional formatters for every backend (to distinguish pacman vs. AUR updates, for example)
- *reddit*: Added link_karma and comment_karma formatters
- *openvpn*: Configurable up/down symbols
- openvpn: Rename colour_up/colour_down to color_up/color_down
- openvpn: NetworkManager compatibility
- *disk*: Improved handling of unmounted drives. Previously the free space of the underlying filesystem would be reported if the path provided was a directory but not a valid mountpoint. This adds a check to first confirm whether a directory is a mountpoint using os.path.ismount(), and if not, then runs an os.listdir() to count the files; empty directories are considered not mounted. This functionality allows for usage on setups with NFS and will not report free space of underlying filesystem in cases with local mountpoints as path.
- *battery*: Added `bar_design` formatter
- *alsa*: Implemented optional volume display/setting as in AlsaMixer
- *pulseaudio*: Fixed bug that created zombies on a click event
- *backlight*: Fixed bug preventing brightness increase

4.3 3.33 (2015-06-23)

- **Errors can now be logged to `~/.i3pystatus-<pid>`**
  - See *Logging*
- **Added new callback system**
  - See *Callbacks*
- **Added credentials storage**
  - See *Credentials*
- Added *Hints* to support special uses cases
- Added support for Pango markup
- **Sending SIGUSR1 to i3pystatus refreshes the bar**
  - See *Refreshing the bar*
- Modules are refreshed instantly after a callback was handled
- Fixed issue where i3bar would interpret plain-text with "HTML-look-alike" characters in them as HTML/Pango

- **New modules**
  - *github*: Check Github for pending notifications.
  - *whosonlocation*: Change your whosonlocation.com status.
  - *openvpn*: Monitor OpenVPN connections. Currently only supports systems that use Systemd.
  - *net_speed*: Attempts to provide an estimation of internet speeds.
  - *makewatch*: Watches for make jobs and notifies when they are completed.
  - *dota2wins*: Displays the win/loss ratio of a given Dota account.
  - *dpms*: Shows and toggles status of DPMS which prevents screen from blanking.
  - *cpu_freq*: uses by default /proc/cpuinfo to determine the current cpu frequency
  - *updates*: Generic update checker. Currently supports apt-get, pacman and cower
  - *openstack_vms*: Displays the number of VMs in an openstack cluster in ACTIVE and non-ACTIVE states.
- *backlight*: add xbacklight support for changing brightness with mouse wheel
- *battery*: added support for depleted batteries
- battery: added support for multiple batteries
- battery: added option to treat all batteries as one large battery (ALL)
- *cpu_usage*: removed hard coded interval setting
- *cpu_usage_bar*: fixed wrong default setting
- *clock*: removed optional pytz dependency
- *network*: cycle available interfaces on click
- **network: centralized network modules**
  - Removed `network_graph`
  - Removed `network_traffic`
  - Removed `wireless`
  - All the features of these three modules are now found in network
- network: added total traffic in Mbytes formatters
- network: `basiciw` is only required if it is used (wireless)
- network: `psutil` is only required if it is used (traffic)
- network: scrolling changes displayed interface
- network: fixed bug that prevented color_up being shown if the user is not using network_traffic
- network: various other enhancements
- `notmuch`: fixed sync issue with database
- *now_playing*: added custom format and color when no player is running
- now_playing: differentiates between D-Bus errors and no players running
- now_playing: fixed D-Bus compatibility with players
- *mail*: added capability to display unread messages per account individually

- *mpd*: various enhancements and fixes
- *pulseaudio*: detect default sink changes in pulseaudio
- *reddit*: can open users mailbox now
- *shell*: fixed module not stripping newlines
- *spotify*: check for metadata on start
- *temp*: alert temperatures
- *weather*: removed pywapi dependency
- weather: add min_temp and max_temp formatters for daily min/max temperature

4.4 3.32 (2014-12-14)

- Added *keyboard_locks* module
- Added *pianobar* module
- Added *uname* module
- *cmus*: enhanced artist/title detection from filenames
- cmus: fixed issue when cmus is not running
- *mpd*: added text_len and truncate_fields options to truncate long artist, album or song names
- network_traffic: added hide_down and format_down options
- *pomodoro*: added format option
- pomodoro: reset timer on left click
- *pulseaudio*: fix rounding error of percentage volume

4.5 3.31 (2014-10-23)

- Unexpected exceptions are now displayed in the status bar
- Core: added mouse wheel handling for upcoming i3 version
- Fixed issues with internet-related modules
- New module mixin: ip3ystatus.core.color.ColorRangeModule
- Added *cmus* module
- Added *cpu_usage_graph* module
- Added network_graph module
- Added network_traffic module
- Added *pomodoro* module
- Added *uptime* module
- *alsa*: mouse wheel changes volume
- *battery*: Added no_text_full option
- *cpu_usage*: Add multicore support

- *cpu_usage_bar*: Add multicore support
- *mail*: db_path option made optional
- *mpd*: Play song on left click even if stopped
- *network*: Add unknown_up setting
- *parcel*: Document lxml dependency
- *pulseaudio*: Added color_muted and color_unmuted options
- pulseaudio: Added step, bar_type, multi_colors, vertical_bar_width options
- pulseaudio: Scroll to change master volume, right click to (un)mute

4.6 3.30 (2014-08-04)

- Added *bitcoin* module
- Added *now_playing* module
- Added *reddit* module
- Added *shell* module
- Core: fixed custom statusline colors not working properly (see issue #74)
- *alsa* and *pulseaudio*: added optional "formated_muted" audio is muted.
- *battery*: add bar formatter, add not_present_text, full_color, charging_color, not_present_color settings
- *disk*: add color and round_size options
- *maildir*: use os.listdir instead of ls
- *mem*: add round_size option
- *mpd*: add color setting
- mpd: add filename formatter
- mpd: next song on right click
- *network* and wireless: support interfaces enslaved to a bonding master
- network: detached_down is now True by default
- network: fixed some issues with interface up/down detection
- *parcel*: added support for Itella (Finnish national postal service) setting. If provided, it will be used instead of "format" when the
- *temp*: add file setting
- temp: fixed issue with Linux kernels 3.15 and newer
- temp: removed color_critical and high_factor options
- *text*: add cmd_leftclick and cmd_rightclick options
- *weather*: add colorize option
- `wireless`: Add quality_bar formatter

4.7 3.29 (2014-04-29)

- *network*: prefer non link-local v6 addresses
- *mail*: Open email client and refresh email with mouse click
- *disk*: Add display and critical limit
- *battery*: fix errors if CURRENT_NOW is not present
- battery: add configurable colors
- *load*: add configurable colors and limit
- *parcel*: rewrote DHL tracker
- Add *spotify* module

4.8 3.28 (2014-04-12)

- **If you're currently using the i3pystatus command to run your i3bar:** Replace `i3pystatus` command in your i3 configuration with `python ~/path/to/your/config.py`
- Do not name your script i3pystatus.py or it will break imports.
- New options for *mem*
- Added *cpu_usage*
- Improved error handling
- Removed `i3pystatus` binary
- `pulseaudio:` changed context name to "i3pystatus_pulseaudio"
- Add maildir backend for mails
- Code changes
- Removed DHL tracker of parcel module, because it doesn't work anymore.

4.9 3.27 (2013-10-20)

- Add *weather* module
- Add *text* module
- *pulseaudio*: Add muted/unmuted options

4.10 3.26 (2013-10-03)

- Add *mem* module

4.11 3.24 (2013-08-04)

This release introduced changes that may require manual changes to your configuration file

- Introduced TimeWrapper
- *battery*: removed remaining_* formatters in favor of TimeWrapper, as it can not only reproduce all the variants removed, but can do much more.
- *mpd*: Uses TimeWrapper for song_length, song_elapsed

CHAPTER 5

Creating modules

Creating new modules ("things that display something") to contribute to i3pystatus is reasonably easy. If the module you want to write updates it's info periodically, like checking for a network link or displaying the status of some service, then we have prepared common tools for this which make this even easier:

- Common base classes: *Module* for everything and *IntervalModule* specifically for the aforementioned usecase of updating stuff periodically.

  the *Module* class inherits a *logger* attribute and as such all logging should be implemented via *self.logger.<level>* rather then initializing a new logger in the module.

- Settings (already built into above classes) allow you to easily specify user-modifiable attributes of your class for configuration.

  See *SettingsBase* for details.

- For modules that require credentials, it is recommended to add a keyring_backend setting to allow users to specify their own backends for retrieving sensitive credentials.

  Required settings and default values are also handled.

Check out i3pystatus' source code for plenty of (simple) examples on how to build modules.

The settings system is built to ease documentation. If you specify two-tuples like `("setting","description")` then Sphinx will automatically generate a nice table listing each option, it's default value and description.

The docstring of your module class is automatically used as the reStructuredText description for your module in the README file.

See also:

SettingsBase for a detailed description of the settings system

5.1 Handling Dependencies

To make it as easy as possible to use i3pystatus we explicitly document all dependencies in the docstring of a module.

The wording usually used goes like this:

```
Requires the PyPI package `colour`
```

To allow automatic generation of the docs without having all requirements of every module installed mocks are used. To make this work simply add all modules of dependencies (so no standard library modules or modules provided by i3pystatus) you import to the `MOCK_MODULES` list in `docs/conf.py`. This needs to be the actual name

of the imported module, so for example if you have `from somepkg.mod import AClass`, you need to add `somepkg.mod` to the list.

5.2 Testing changes

i3pystatus uses continuous integration (CI) techniques, which means in our case that every patch and every pull request is tested automatically. While Travis is used for automatic building of GitHub pull requests, it is not the authoritative CI system (which is Der Golem) for the main repository.

The `ci-build.sh` script needs to run successfully for a patch to be accepted. It can be run on your machine, too, so you don't need to wait for the often slow Travis build to complete. It does not require any special privileges, except write access to the `ci-build` directory (a different build directory can be specified as the first parameter to `ci-build.sh`).

The script tests the following things:

1. PEP8 compliance of the entire codebase, *excluding* errors of too long lines (error code E501). Line lengths of about 120 characters are acceptable.
2. That `setup.py` installs i3pystatus and related binaries (into a location below the build directory)
3. Unit tests pass, they are tested against the installed version from 2.). A unit test log in JUnit format is generated in the build directory (`testlog.xml`).
4. Sphinx docs build without errors or warnings. The HTML docs are generated in the `docs` directory in the build directory.

CHAPTER 6

core Package

6.1 core Package

class i3pystatus.core.**CommandEndpoint**(*modules*, *io_handler_factory*, *io*)
Bases: object

Endpoint for i3bar click events: http://i3wm.org/docs/i3bar-protocol.html#_click_events

Parameters

- **modules** – dict-like object with item access semantics via .get()
- **io_handler_factory** – function creating a file-like object returning a JSON generator on .read()

start()
Starts the background thread

class i3pystatus.core.**Status**(*standalone=True*, *click_events=True*, *interval=1*, *input_stream=None*, *logfile=None*, *internet_check=None*, *keep_alive=False*, *logformat='%(asctime)s [%(levelname)-8s][%(name)s %(lineno)d] %(message)s'*, *default_hints=None*)
Bases: object

The main class used for registering modules and managing I/O

Parameters

- **standalone** (*bool*) – Whether i3pystatus should read i3status-compatible input from *input_stream*.
- **interval** (*int*) – Update interval in seconds.
- **input_stream** – A file-like object that provides the input stream, if *standalone* is False.
- **click_events** (*bool*) – Enable click events, if *standalone* is True.
- **logfile** (*str*) – Path to log file that will be used by i3pystatus.
- **internet_check** (*tuple*) – Address of server that will be used to check for internet connection by *internet*.
- **keep_alive** – If True, modules that define the keep_alive flag will not be put to sleep when the status bar is hidden.
- **default_hints** (*dictionary*) – Dictionary of default hints to apply to all modules. Can be overridden at a module level.

register(*module*, **args*, ***kwargs*)
Register a new module.

Parameters

- **module** – Either a string module name, or a module class, or a module instance (in which case args and kwargs are invalid).
- **kwargs** – Settings for the module.

Returns module instance

run()
Run main loop.

6.2 color Module

class i3pystatus.core.color.**ColorRangeModule**
Bases: object

Class to dynamically generate and select colors.

Requires the PyPI package *colour*

end_color = 'red'

get_gradient(*value*, *colors*, *upper_limit=100*)
Map a value to a color :param value: Some value :return: A Hex color code

static get_hex_color_range(*start_color*, *end_color*, *quantity*)
Generates a list of quantity Hex colors from start_color to end_color.

Parameters

- **start_color** – Hex or plain English color for start of range
- **end_color** – Hex or plain English color for end of range
- **quantity** – Number of colours to return

Returns A list of Hex color values

static percentage(*part*, *whole*)
Calculate percentage

start_color = '#00FF00'

6.3 command Module

i3pystatus.core.command.**CommandResult**
alias of Result

i3pystatus.core.command.**execute**(*command*, *detach=False*)
Runs a command in background. No output is retrieved. Useful for running GUI applications that would block click events.

Parameters

- **command** – A string or a list of strings containing the name and arguments of the program.

- **detach** – If set to *True* the program will be executed using the *i3-msg* command. As a result the program is executed independent of i3pystatus as a child of i3 process. Because of how i3-msg parses its arguments the type of *command* is limited to string in this mode.

i3pystatus.core.command.**run_through_shell**(*command*, *enable_shell=False*)

Retrieve output of a command. Returns a named tuple with three elements:

- `rc` (integer) Return code of command.
- `out` (string) Everything that was printed to stdout.
- `err` (string) Everything that was printed to stderr.

Don't use this function with programs that outputs lots of data since the output is saved in one variable.

Parameters

- **command** – A string or a list of strings containing the name and arguments of the program.
- **enable_shell** – If set ot *True* users default shell will be invoked and given `command` to execute. The `command` should obviously be a string since shell does all the parsing.

6.4 desktop Module

class i3pystatus.core.desktop.**BaseDesktopNotification**(*title*, *body*, *icon='dialog-information'*, *urgency=1*, *timeout=-1*, *log_level=30*)

Bases: `object`

Class to display a desktop notification

Parameters

- **title** – Title of the notification
- **body** – Body text of the notification, depending on the users system configuration HTML may be used, but is not recommended
- **icon** – A XDG icon name, see http://standards.freedesktop.org/icon-naming-spec/icon-naming-spec-latest.html
- **urgency** – A value between 1 and 3 with 1 meaning low urgency and 3 high urgency.
- **timeout** – Timeout in seconds for the notification. Zero means it needs to be dismissed by the user.

display()

Display this notification

Returns boolean indicating success

update(*title=None*, *body=None*, *icon=None*)

Update this notification.

Parameters

- **title** – Title of the notification
- **body** – Body text of the notification, depending on the users system configuration HTML may be used, but is not recommended
- **icon** – A XDG icon name, see http://standards.freedesktop.org/icon-naming-spec/icon-naming-spec-latest.html

:return boolean indicating success

class i3pystatus.core.desktop.**DesktopNotification**(***kwargs*)
Bases: *i3pystatus.core.desktop.DesktopNotification*

URGENCY_LUT = (<Mock name='mock.Notify.Urgency.LOW' id='140608391492560'>, <Mock name='mock.Notify.Urgen

display()

update(*title=None*, *body=None*, *icon=None*)

6.5 exceptions Module

exception i3pystatus.core.exceptions.**ConfigAmbigiousClassesError**(*module*, **args*, ***kwargs*)
Bases: *i3pystatus.core.exceptions.ConfigError*

format(*ambigious_classes*)

exception i3pystatus.core.exceptions.**ConfigError**(*module*, **args*, ***kwargs*)
Bases: Exception

ABC for configuration exceptions

format(**args*, ***kwargs*)

exception i3pystatus.core.exceptions.**ConfigInvalidModuleError**(*module*, **args*, ***kwargs*)
Bases: *i3pystatus.core.exceptions.ConfigError*

format()

exception i3pystatus.core.exceptions.**ConfigKeyError**(*module*, **args*, ***kwargs*)
Bases: *i3pystatus.core.exceptions.ConfigError*, KeyError

format(*key*)

exception i3pystatus.core.exceptions.**ConfigMissingError**(*module*, **args*, ***kwargs*)
Bases: *i3pystatus.core.exceptions.ConfigError*

format(*missing*)

6.6 imputil Module

class i3pystatus.core.imputil.**ClassFinder**(*baseclass*)
Bases: object

Support class to find classes of specific bases in a module

get_class(*module*)

get_matching_classes(*module*)

get_module(*module*)

instanciate_class_from_module(*module*, **args*, ***kwargs*)

predicate_factory(*module*)

6.7 io Module

class i3pystatus.core.io.**IOHandler**(*inp=<_io.TextIOWrapper name='<stdin>' mode='r' encoding='UTF-8'>, out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)

Bases: object

read()
Iterate over all input lines (Generator)

read_line()
Interrupted respecting reader for stdin.

Raises EOFError if the end of stream has been reached

write_line(*message*)
Unbuffered printing to stdout.

class i3pystatus.core.io.**JSONIO**(*io*, *skiplines=2*)

Bases: object

parse_line(*line*)
Parse a single line of JSON and write modified JSON back.

read()
Iterate over all JSON input (Generator)

class i3pystatus.core.io.**StandaloneIO**(*click_events*, *modules*, *keep_alive*, *interval=1*)

Bases: *i3pystatus.core.io.IOHandler*

I/O handler for standalone usage of i3pystatus (w/o i3status)

Writing works as usual, but reading will always return a empty JSON array, and the i3bar protocol header

async_refresh()
Calling this method will send the status line to i3bar immediately without waiting for timeout (1s by default).

compute_treshold_interval()
Current method is to compute average from all intervals.

n = -1

proto = [{'version': 1, 'click_events': True}, '[', '[]', ',[]']

read()

read_line()

refresh_signal_handler(*signo*, *frame*)
This callback is called when SIGUSR1 signal is received.

It updates outputs of all modules by calling their *run* method.

Interval modules are updated in separate threads if their interval is above a certain treshold value. This treshold is computed by *compute_treshold_interval()* class method. The reasoning is that modules with larger intervals also usually take longer to refresh their output and that their output is not required in 'real time'. This also prevents possible lag when updating all modules in a row.

suspend_signal_handler(*signo*, *frame*)
By default, i3bar sends SIGSTOP to all children when it is not visible (for example, the screen sleeps or you enter full screen mode). This stops the i3pystatus process and all threads within it. For some modules,

this is not desirable. Thankfully, the i3bar protocol supports setting the "stop_signal" and "cont_signal" key/value pairs in the header to allow sending a custom signal when these events occur.

Here we use SIGUSR2 for both "stop_signal" and "cont_signal" and maintain a toggle to determine whether we have just been stopped or continued. When we have been stopped, notify the IntervalModule managers that they should suspend any module that does not set the keep_alive flag to a truthy value, and when we have been continued, notify the IntervalModule managers that they can resume execution of all modules.

6.8 modules Module

class i3pystatus.core.modules.**IntervalModule**(**args*, ***kwargs*)
Bases: *i3pystatus.core.modules.Module*

interval = 5

managers = {}

registered(*status_handler*)

required = set()

run()
Called approximately every self.interval seconds

Do not rely on this being called from the same thread at all times. If you need to always have the same thread context, subclass AsyncModule.

settings = [('interval', 'interval in seconds between module updates'), ('on_leftclick', 'Callback called on left click (see

class i3pystatus.core.modules.**Module**(**args*, ***kwargs*)
Bases: *i3pystatus.core.settings.SettingsBase*

hints = {'markup': 'none'}

inject(*json*)

move(*position*)

multi_click_timeout = 0.25

on_change = None

on_click(*button*, ***kwargs*)
Maps a click event with its associated callback.

Currently implemented events are:

Event	Callback setting	Button ID
Left click	on_leftclick	1
Middle click	on_middleclick	2
Right click	on_rightclick	3
Scroll up	on_upscroll	4
Scroll down	on_downscroll	5
Others	on_otherclick	> 5

The action is determined by the nature (type and value) of the callback setting in the following order:

1.If null callback (`None`), no action is taken.

2.If it's a *python function*, call it and pass any additional arguments.

3.If it's name of a *member method* of current module (string), call it and pass any additional arguments.

4.If the name does not match with *member method* name execute program with such name.

See also:

Callbacks for more information about callback settings and examples.

Parameters

- **button** – The ID of button event received from i3bar.
- **kwargs** – Further information received from i3bar like the positions of the mouse where the click occurred.

Returns Returns `True` if a valid callback action was executed. `False` otherwise.

on_doubledownscroll = None

on_doubleleftclick = None

on_doublemiddleclick = None

on_doubleotherclick = None

on_doublerightclick = None

on_doubleupscroll = None

on_downscroll = None

on_leftclick = None

on_middleclick = None

on_otherclick = None

on_rightclick = None

on_upscroll = None

output

position = 0

registered(*status_handler*)
Called when this module is registered with a status handler

required = set()

run()

send_output()
Send a status update with the current module output

settings = [('on_leftclick', 'Callback called on left click (see :ref:'callbacks')'), ('on_middleclick', 'Callback called on m

text_to_pango()
Replaces all ampersands in *full_text* and *short_text* attributes of *self.output* with *&*.

It is called internally when pango markup is used.

Can be called multiple times (*&* won't change to *&*).

i3pystatus.core.modules.**is_method_of**(*method*, *object*)
Decide whether `method` is contained within the MRO of `object`.

6.9 settings Module

class i3pystatus.core.settings.**SettingsBase**(**args*, ***kwargs*)
Bases: object

Support class for providing a nice and flexible settings interface

Classes inherit from this class and define what settings they provide and which are required.

The constructor is either passed a dictionary containing these settings, or keyword arguments specifying the same.

Settings are stored as attributes of self.

static flatten_settings(*settings*)

get_protected_settings(*settings_source*)
Attempt to retrieve protected settings from keyring if they are not already set.

get_setting_from_keyring(*setting_identifier*, *keyring_backend=None*)
Retrieves a protected setting from keyring :param setting_identifier: must be in the format package.module.Class.setting

init()
Convenience method which is called after all settings are set

In case you don't want to type that super()... blabla :-)

log_level = 30

logger = None

required = set()
required can list settings which are required

settings = [('log_level', 'Set to true to log error to .i3pystatus-<pid> file.')]
settings should be tuple containing two types of elements:

- bare strings, which must be valid Python identifiers.
- two-tuples, the first element being a identifier (as above) and the second a docstring for the particular setting

class i3pystatus.core.settings.**SettingsBaseMeta**(*name*, *bases*, *namespace*)
Bases: type

Add interval setting to *settings* attribute if it does not exist.

static get_merged_settings()

6.10 threading Module

class i3pystatus.core.threading.**ExceptionWrapper**(*workload*)
Bases: *i3pystatus.core.threading.Wrapper*

format_exception()

truncate_error(*exception_message*)

class i3pystatus.core.threading.**Manager**(*target_interval*)
Bases: object

append(*workload*)

create_thread(*workloads*)

create_threads(*threads*)

partition_workloads(*workloads*)

resume()

start()

suspend()

wrap(*workload*)

class i3pystatus.core.threading.**Thread**(*target_interval*, *workloads=None*, *start_barrier=1*)
Bases: threading.Thread

append(*workload*)

branch(*vtime*, *bound*)

execute_workloads()

pop()

resume()

run()

should_execute(*workload*)
If we have been suspended by i3bar, only execute those modules that set the keep_alive flag to a truthy value. See the docs on the suspend_signal_handler method of the io module for more information.

suspend()

time

wait_for_start_barrier()

class i3pystatus.core.threading.**WorkloadWrapper**(*workload*)
Bases: *i3pystatus.core.threading.Wrapper*

time = 0.0

class i3pystatus.core.threading.**Wrapper**(*workload*)
Bases: object

i3pystatus.core.threading.**unwrap_workload**(*workload*)
Obtain the module from it's wrapper.

6.11 util Module

class i3pystatus.core.util.**KeyConstraintDict**(*valid_keys*, *required_keys*)
Bases: collections.UserDict

A dict implementation with sets of valid and required keys

Parameters

- **valid_keys** – Set of valid keys
- **required_keys** – Set of required keys, must be a subset of valid_keys

exception MissingKeys(*keys*)
Bases: Exception

KeyConstraintDict.**missing**()
Returns a set of keys that are required but not set

class i3pystatus.core.util.**ModuleList**(*status_handler*, *class_finder*)
Bases: collections.UserList

append(*module*, **args*, ***kwargs*)

get(*find_id*)

class i3pystatus.core.util.**MultiClickHandler**(*callback_handler*, *timeout*)
Bases: object

check_double(*button*)

clear_timer()

set_timer(*button*, *cb*, ***kwargs*)

class i3pystatus.core.util.**TimeWrapper**(*seconds*, *default_format='%m:%S'*)
Bases: object

A wrapper that implements __format__ and __bool__ for time differences and time spans.

Parameters

- **seconds** – seconds (numeric)
- **default_format** – the default format to be used if no explicit format_spec is passed to __format__

Format string syntax:

- %h, %m and %s are the hours, minutes and seconds without leading zeros (i.e. 0 to 59 for minutes and seconds)
- %H, %M and %S are padded with a leading zero to two digits, i.e. 00 to 59
- %l and %L produce hours non-padded and padded but only if hours is not zero. If the hours are zero it produces an empty string.
- %% produces a literal %
- %E (only valid on beginning of the string) if the time is null, don't format anything but rather produce an empty string. If the time is non-null it is removed from the string.

The formatted string is stripped, i.e. spaces on both ends of the result are removed

class TimeTemplate(*template*)
Bases: string.Template

delimiter = '%'

idpattern = '[a-zA-Z]'

pattern = re.compile('\n %(?:\n (?P<escaped>%) | # Escape sequence of two delimiters\n (?P<named>[a-zA-Z]) | #

i3pystatus.core.util.**bytes_info_dict**(*in_bytes*)

i3pystatus.core.util.**convert_position**(*pos*, *json*)

i3pystatus.core.util.**flatten**(*l*)
Flattens a hierarchy of nested lists into a single list containing all elements in order

Parameters **l** – list of arbitrary types and lists

Returns list of arbitrary types

i3pystatus.core.util.**formatp**(*string*, ***kwargs*)
Function for advanced format strings with partial formatting

This function consumes format strings with groups enclosed in brackets. A group enclosed in brackets will only become part of the result if all fields inside the group evaluate True in boolean contexts.

Groups can be nested. The fields in a nested group do not count as fields in the enclosing group, i.e. the enclosing group will evaluate to an empty string even if a nested group would be eligible for formatting. Nesting is thus equivalent to a logical or of all enclosing groups with the enclosed group.

Escaped brackets, i.e. \[and \] are copied verbatim to output.

Parameters

- **string** – Format string
- **kwargs** – keyword arguments providing data for the format string

Returns Formatted string

i3pystatus.core.util.**get_module**(*function*)
Function decorator for retrieving the `self` argument from the stack.

Intended for use with callbacks that need access to a modules variables, for example:

```
from i3pystatus import Status, get_module
from i3pystatus.core.command import execute
status = Status(...)
# other modules etc.
@get_module
def display_ip_verbose(module):
    execute('sh -c "ip addr show dev {dev} | xmessage -file -"'.format(dev=module.
↪interface))
status.register("network", interface="wlan1", on_leftclick=display_ip_verbose)
```

class i3pystatus.core.util.**internet**
Bases: `object`

Checks for internet connection by connecting to a server.

This class exposes two configuration variables:

- address - a tuple containing (host,port) of the server to connect to
- check_frequency - the frequency in seconds for checking the connection

Return type bool

See also:

require()

address = ('google.com', 80)

static check(*res*)

static check_connection()

check_frequency = 1

connected = False

dns_cache = []

last_checked = 3519.025350712

static resolve()

i3pystatus.core.util.**lchop**(*string*, *prefix*)
Removes a prefix from string

Parameters

- **string** – String, possibly prefixed with prefix
- **prefix** – Prefix to remove from string

Returns string without the prefix

i3pystatus.core.util.**make_bar**(*percentage*)
Draws a bar made of unicode box characters.

Parameters **percentage** – A value between 0 and 100

Returns Bar as a string

i3pystatus.core.util.**make_glyph**(*number*, *glyphs=' _'*, *lower_bound=0*, *upper_bound=100*, *enable_boundary_glyphs=False*)
Returns a single glyph from the list of glyphs provided relative to where the number is in the range (by default a percentage value is expected).

This can be used to create an icon based representation of a value with an arbitrary number of glyphs (e.g. 4 different battery status glyphs for battery percentage level).

Parameters

- **number** – The number being represented. By default a percentage value between 0 and 100 (but any range can be defined with lower_bound and upper_bound).
- **glyphs** – Either a string of glyphs, or an array of strings. Using an array of strings allows for additional pango formatting to be applied such that different colors could be shown for each glyph).
- **lower_bound** – A custom lower bound value for the range.
- **upper_bound** – A custom upper bound value for the range.
- **enable_boundary_glyphs** – Whether the first and last glyphs should be used for the special case of the number being <= lower_bound or >= upper_bound respectively.

Returns The glyph found to represent the number

i3pystatus.core.util.**make_graph**(*values*, *lower_limit=0.0*, *upper_limit=100.0*, *style='blocks'*)
Draws a graph made of unicode characters.

Parameters

- **values** – An array of values to graph.
- **lower_limit** – Minimum value for the y axis (or None for dynamic).
- **upper_limit** – Maximum value for the y axis (or None for dynamic).
- **style** – Drawing style ('blocks', 'braille-fill', 'braille-peak', or 'braille-snake').

Returns Bar as a string

i3pystatus.core.util.**make_vertical_bar**(*percentage*, *width=1*, *glyphs=None*)
Draws a vertical bar made of unicode characters.

Parameters

- **percentage** – A value between 0 and 100

- **width** – How many characters wide the bar should be.

Returns Bar as a String

i3pystatus.core.util.**partition**(*iterable*, *limit*, *key*=<*function* <*lambda*>>)

i3pystatus.core.util.**popwhile**(*predicate*, *iterable*)

Generator function yielding items of iterable while predicate holds for each item

Parameters

- **predicate** – function taking an item returning bool
- **iterable** – iterable

Returns iterable (generator function)

i3pystatus.core.util.**require**(*predicate*)

Decorator factory for methods requiring a predicate. If the predicate is not fulfilled during a method call, the method call is skipped and None is returned.

Parameters **predicate** – A callable returning a truth value

Returns Method decorator

See also:

internet

i3pystatus.core.util.**round_dict**(*dic*, *places*)

Rounds all values in a dict containing only numeric types to *places* decimal places. If places is None, round to INT.

i3pystatus.core.util.**user_open**(*url_or_command*)

Open the specified paramater in the web browser if a URL is detected, othewrise pass the paramater to the shell as a subprocess. This function is inteded to bu used in on_leftclick/on_rightclick callbacks.

Parameters **url_or_command** – String containing URL or command

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

i

A

B

C

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z