
i3pystatus Documentation

Release

Author

August 31, 2016

1	Configuration	3
1.1	Formatting	5
1.1.1	formatp	5
1.1.2	TimeWrapper	5
1.2	Logging	6
1.2.1	Setting a specific logfile	6
1.2.2	Changing log format	6
1.2.3	Log level	6
1.3	Callbacks	7
1.4	Hints	9
1.5	Refreshing the bar	10
1.6	Internet Connectivity	10
1.7	Credentials	10
2	Module reference	13
2.1	Mail Backends	85
2.2	Score Backends	87
2.3	Update Backends	99
2.4	Weather Backends	100
3	Changelog	103
3.1	3.35 (2016-08-31)	103
3.2	3.34 (2016-02-14)	104
3.3	3.33 (2015-06-23)	105
3.4	3.32 (2014-12-14)	107
3.5	3.31 (2014-10-23)	107
3.6	3.30 (2014-08-04)	108
3.7	3.29 (2014-04-29)	109
3.8	3.28 (2014-04-12)	109
3.9	3.27 (2013-10-20)	109
3.10	3.26 (2013-10-03)	109
3.11	3.24 (2013-08-04)	110
4	Creating modules	111
4.1	Handling Dependencies	111
4.2	Testing changes	112
5	core Package	113

5.1	core Package	113
5.2	color Module	114
5.3	command Module	114
5.4	desktop Module	115
5.5	exceptions Module	115
5.6	imputil Module	116
5.7	io Module	116
5.8	modules Module	117
5.9	settings Module	119
5.10	threading Module	120
5.11	util Module	120
6	Indices and tables	125
	Python Module Index	127

Contents:

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%h",
    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.

1.1 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

1.1.1 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.

1.1.2 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.

1.2 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.

1.2.1 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')
```

1.2.2 Changing log format

New in version 3.35.

The `logformat` option can be used 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: ',
)
```

1.2.3 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`).

1.3 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_rightclick`
- `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.

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,
)
```

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 a *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 -"
)
```

1.4 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.
- `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")
```

1.5 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 its 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:

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

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

1.6 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))
```

1.7 Credentials

Settings that require credentials can utilize the keyring module to keep sensitive information out of config files. To take advantage of this feature, simply use the `i3pystatus-setting-util` script installed along i3pystatus to set the credentials for a module. Once this is done you can add the module to your config without specifying the credentials, e.g.:

```
# Use the default keyring to retrieve credentials.
# To determine which backend is the default on your system, run
# python -c 'import keyring; print(keyring.get_keyring()) '
status.register('github')
```

If you don't want to use the default you can set a specific keyring like so:

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

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

Note: Credential handling requires the PyPI package `keyring`. Many distributions have it pre-packaged available as `python-keyring`.

Module reference

Module overview:

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

Audio *alsa - pulseaudio*

Hardware *backlight - battery - temp*

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

Music *cmus - 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:

- *abc_radio*
- *alsa*
- *anybar*
- *backlight*
- *battery*
- *bitcoin*
- *clock*
- *cmus*
- *cpu_freq*
- *cpu_usage*
- *cpu_usage_bar*
- *cpu_usage_graph*
- *disk*
- *dota2wins*
- *dpms*

- *file*
- *github*
- *google_calendar*
- *gpu_mem*
- *gpu_temp*
- *inet*
- *keyboard_locks*
- *lastfm*
- *load*
- *mail*
- *makewatch*
- *mem*
- *mem_bar*
- *modsde*
- *moon*
- *mpd*
- *net_speed*
- *network*
- *now_playing*
- *online*
- *openfiles*
- *openstack_vms*
- *openvpn*
- *parcel*
- *pianobar*
- *ping*
- *plexstatus*
- *pomodoro*
- *pulseaudio*
- *pyload*
- *reddit*
- *regex*
- *runwatch*
- *scores*
- *scratchpad*
- *sge*

- *shell*
- *solaar*
- *spotify*
- *syncthing*
- *taskwarrior*
- *temp*
- *text*
- *timer*
- *uname*
- *updates*
- *uptime*
- *vk*
- *weather*
- *weekcal*
- *whosonlocation*
- *window_title*
- *xkbblayout*
- *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](#))
- **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: `0`) – 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](#))
- 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](#))
- **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
- **backlight** (default: *acpi_video0*) – backlight, see */sys/class/backlight/*
- **color** (default: #FFFFFF)
- **components** (default: {'brightness': (<class 'int'>, 'brightness'), 'max_brightness': (<class 'int'>, 'max_brightness')})
- **transforms** (default: {'percentage': <function Backlight.<lambda> at 0x7f4d36399b70>})
- **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](#))
- **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

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.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_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: {'DIS': 'DIS', 'CHR': 'CHR', 'DPL': 'DPL', 'FULL': 'FULL'}) – A dictionary mapping ('DPL', 'DIS', 'CHR', 'FULL') to alternative 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%
- **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](#))
- **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 BitcoinAverage Price Index API <<https://bitcoinaverage.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}
- {daily_average}
- {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: *empty*) – 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: `B`) – Symbol for bitcoin sign
- status** (default: `{ 'price_down' : ' ', 'price_up' : ' ' }`)
- 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://bitcoinaverage.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](#))
- 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.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](#))
- 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}`) – format 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': ' ', 'stopped': ' ', 'playing': ' ' }`) – 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](#))
- 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 $0 \leq X \leq \text{number of cores} - 1$
- `{coreXg}` - frequency of core number *X* in GHz (format *1.2f*), where $0 \leq X \leq \text{number of cores} - 1$

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](#))
- 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 :return: dictionary used as the full-text output for the module

class `i3pystatus.cpu_usage.CpuUsage`

Shows CPU usage. The first output will be inaccurate.

Linux only

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. uses 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 formatters are `{core}` and `{usage}`.
- exclude_average** (default: `False`) – If `True` usage average of all cores will not be in `format_all`.
- color** (default: `empty`) – 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](#))
- 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 calculates 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 inaccurate.

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’
- format_all** (default: `{core} : {usage:02}%`) – format string used for `{usage_all}` per core. Available formatters are `{core}` and `{usage}`.
- exclude_average** (default: `False`) – If True usage average of all cores will not be in `format_all`.
- color** (default: *empty*) – 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_doubleclick** (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*)
- 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 inaccurate.

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. uses 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’)
- format** (default: `{cpu_graph}`) – format string.
- format_all** (default: `{core} : {usage:02}%`) – format string used for `{usage_all}` per core. Available formatters are `{core}` and `{usage}`.
- exclude_average** (default: `False`) – If `True` usage average of all cores will not be in `format_all`.
- color** (default: `empty`) – 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](#))
- 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.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](#))
- 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 percentage 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 screenname
- **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](#))
- **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](#))
- 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 an 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*)
- 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`

Check GitHub for pending notifications. Requires *requests*

Available authentication methods:

- username + password
- access_token (manually generate a new token at <https://github.com/settings/tokens>)

See <https://developer.github.com/v3/#authentication> for more informations.

Formatters:

- {unread}* — contains the value of `unread_marker` when there are pending notifications
- {unread_count}* — number of unread notifications, empty if 0

Settings

- format** (default: `{unread}`) – format string
- keyring_backend** (default: *empty*) – alternative keyring backend for retrieving credentials

- **unread_marker** (default:) – sets the string that the “unread” formatter shows when there are pending notifications
- **username** (default: *empty*)
- **password** (default: *empty*)
- **access_token** (default: *empty*) – see <https://developer.github.com/v3/#authentication>
- **color** (default: #78EAF2)
- **interval** (default: 600) – interval in seconds between module updates
- **on_leftclick** (default: `open_github`) – 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*)
- **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.google_calendar.GoogleCalendar`

Simple module for displaying next Google Calendar event.

Requires the Google Calendar API package - <https://developers.google.com/google-apps/calendar/quickstart/python>. Additionally requires the *colour*, *httplib2*, *oauth2client*, *pytz*, *apiclient* and *dateutil* modules.

All top level keys returned by the Google Calendar API can be used as formatters. Some examples include:

Available formatters

- `{kind}` — type of event
- `{status}` — eg, confirmed
- `{summary}` — essentially the title
- `{remaining_time}` - how long remaining until the event
- `{start_time}` - when this event starts
- `{htmlLink}` — link to the calendar event

Settings

- **format** (default: `{summary} ({remaining_time})`) – format string
- **credential_path** (required) – Path to credentials
- **skip_recurring** (default: `True`) – Skip recurring events.
- **days** (default: `1`) – Only show events between now and this many days in the future
- **urgent_seconds** (default: `300`) – Add urgent hint when this many seconds until event startTime
- **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: `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](#))
- **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
- **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](#))
- **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
- **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](#))
- **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

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](#))
- 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](#))
- **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: FFFFFFFF)
- **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](#))
- **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 indicates that one out of 285 total tasks is runnable)

Settings

- format** (default: `{avg1} {avg5}`)
- color** (default: `#ffffff`) – The text color
- critical_limit** (default: 4) – 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](#))
- 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](#))
- **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](#))
- **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](#))
- 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](#))
- **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](#))
- **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})
- **status** (default: {'Waning Gibbous': 'WanGib', 'New Moon': 'NM', 'Waxing Gibbous': 'WaxGib', 'Full Moon': 'FM', 'Last Quarter': 'LQ', 'First Quarter': 'FQ', 'Waning Crescent': 'WanCres', 'Waxing Crescent': 'WaxCres'}) – Current moon phase
- **illum** (default: <function MoonPhase.illum at 0x7f4d34c1d1e0>) – Percentage that is illuminated
- **color** (default: {'Waning Gibbous': '#871181', 'New Moon': '#00BDE5', 'Waxing Gibbous': '#392FBF', 'Full Moon': '#4C00B3', 'Last Quarter': '#C32250', 'First Quarter': '#265ECC', 'Waning Crescent': '#FF341F', 'Waxing Crescent': '#138DD8'}) – Set color
- **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](#))
- **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)
- `{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`.

Settings

- host** (default: `localhost`)
- port** (default: `6600`) – MPD port. If set to 0, host will be interpreted as a Unix socket.
- format** (default: `{title} {status}`) – `formatp` string
- status** (default: `{ 'stop': '', 'pause': '', 'play': '' }`) – Dictionary mapping pause, play and stop to output
- color** (default: `#FFFFFF`) – The 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 whole string, if exceeding fields specified in `truncate_fields` are truncated equally. If truncated, ellipsis are appended as indicator. It's applied *after* `max_field_len`. Value of 0 disables this.
- truncate_fields** (default: `('title', '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](#))
- 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`

Settings

- url** (default: *empty*) – Target URL to download a file from. Uses `speedtest_cli` to find the ‘best’ server if none is supplied.
- units** (default: *bits*) – Valid values are B, b, bytes, or bits
- format** (default: {speed} ({hosting_provider}))
- 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](#))
- **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.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
- **{quality}** — Link quality in percent
- **{quality_bar}** — Bar graphically representing link quality

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

- **{interface}** — the configured network interface
- **{kbs}** – Float representing KiBs corresponds to graph type
- **{network_graph}** – Unicode graph representing network usage
- **{bytes_sent}** — bytes sent per second (divided by divisor)
- **{bytes_recv}** — bytes received per second (divided by divisor)
- **{packets_sent}** — bytes sent per second (divided by divisor)
- **{packets_recv}** — bytes received per second (divided by divisor)
- **{rx_tot_Mbytes}** — total Mbytes received
- **{tx_tot_Mbytes}** — total Mbytes sent

Settings

- format_up** (default: {interface} {network_graph}{kbs}KB/s) – format string
- 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’)
- 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
- separate_color** (default: False) – display recv/send color separate in dynamic color mode. Note: only network speed formatters will display with range color
- graph_type** (default: input) – Whether to draw the network traffic graph for input or output. Allowed values ‘input’ or ‘output’
- divisor** (default: 1024) – divide all byte values by this value
- ignore_interfaces** (default: [‘lo’]) – Array of interfaces to ignore when cycling through on click, eg, [‘lo’]
- round_size** (default: empty) – 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
- 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](#))
- **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 most media players

- Requires python-dbus available from every distros' package manager.

Left click on the module play/pauses, right click goes 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)

Settings

- **player** (default: *empty*) – Player name. If not set, compatible players will be detected automatically.
- **status** (default: {'stop': '', 'pause': '', 'play': ''}) – 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: *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](#))
- **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.Offline`
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](#))
- **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](#))
- **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 \leq *threshold*
- **crit_color** (default: `#FF0000`) – Display color when non-active VMs are \geq *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](#))
- **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. Currently only supports systems that use Systemd.

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 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](#))
- 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, Itella

- `parcel.UPS("<id_code>")`
- `parcel.DHL("<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*)
- **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*)
- **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` propertie 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_down** (default: `#FF0000`) – color when ping fail
- format_disabled** (default: *empty*) – format string when disabled
- format_down** (default: `down`) – format string when ping fail
- 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](#))
- 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.

Settings

- **apikey** (required) – Your Plex API authentication key (<https://support.plex.tv/hc/en-us/articles/204059436-Finding-your-account-token-X-Plex-Token>) .
- **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 (in seconds).
- **format_no_streams** (default: *empty*) – String that is shown if nothing is being streamed.
- **format** (default: {platform}: {title})
- **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](#))
- **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.

Settings

- **sound** (required) – 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`
- **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](#))
- **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>

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 $-\infty$)
- `{muted}` — the value of one of the *muted* or *unmuted* settings
- `{volume_bar}` — unicode bar showing volume

Settings

- **format** (default: `♪: {volume}`)
- **format_muted** (default: `empty`) – optional format string to use when muted
- **muted** (default: `M`)
- **unmuted** (default: `empty`)
- **color_muted** (default: `#FF0000`)
- **color_unmuted** (default: `#FFFFFF`)
- **step** (default: `5`) – percentage to increment volume on scroll
- **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
- **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](#))
- **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 whether 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*, *_*, *_*)

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](#))
- 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>>.

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.
- password** (default: *empty*) – Reddit password.
- 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’.
- 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: `{ 'no_mail' : ' ', 'new_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](#))
- **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.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](#))
- **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](#))
- 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.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

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'],
            format_no_games='No games today :(',
            inning_top='',
            inning_bottom='',
        ),
        nhl.NHL(teams=['CHI']),
        nba.NBA(
            teams=['GSW'],
            all_games=False,
        ),
        epl.EPL(),
    ],
)

status.run()
```

To enable colorized team name/city/abbreviation, *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,
        ),
    ],
)
```

```

    1,
)

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
- **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-colored 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`) – Display 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.
- **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*)
- **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 formatters

- `{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](#))
- 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](#))
- 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
- 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](#))
- 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 bluetooth-device

Available formatters

- {output}* — percentage of battery and status

Settings

- nameOfDevice** (required) – name of the bluetooth-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](#))

- **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**

Gets Spotify info using playerctl

Available formatters

- *{status}* — current status icon (paused/playing)
- *{length}* — total song duration (mm:ss format)
- *{artist}* — artist
- *{title}* — title
- *{album}* — album

Settings

- **format** (default: {status} {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': ''}) – 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](#))
- **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_info (*player*)
gets spotify track info from playerctl

next_song ()
skips to the next song

playpause ()
Pauses and plays spotify

previous_song ()
Plays the previous song

run ()
Main statement, executes all code every interval

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](#))
- **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*

Formatters:

- *{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

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: `empty`) – 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](#))
- 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

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
- 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](#))
- 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](#))
- **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', '#00ff00')]`)
- 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](#))
- 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.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](#))
- **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](#))
- 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](#))
- **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 access 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](#))
- **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](#).

Left clicking on the module will launch the forecast page for the location being checked.

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

This module supports the *formatp* extended string format syntax. This allows for values to be hidden when they evaluate as False. This comes in handy for the *weathercom* backend, which at a certain point in the afternoon will have a blank *{high_temp}* value. Using the following snippet in your format string will only display the high temperature information if it is not blank:

```
{current_temp}{temp_unit}[ Hi: {high_temp}{{temp_unit}}] Lo: {low_temp}{temp_unit}
```

Brackets are evaluated from the outside-in, so the fact that the only formatter in the outer block (*{high_temp}*) is empty would keep the inner block from being evaluated at all, and entire block would not be displayed.

See the following links for usage examples for the available weather backends:

- [Weather.com](#)
- [Weather Underground](#)

Settings

- colorize** (default: False) – Vary the color depending on the current conditions.
- color_icons** (default: `{'Fog': ('', '#949494'), 'Thunderstorm': ('', '#cbd2c0'), 'Snow': ('', '#ffffff'), 'Fair': ('', '#ffcc00'), 'Sunny': ('', '#ffff00'), 'default': ('', None), 'Rainy': ('', '#cbd2c0'), 'Cloudy': ('', '#f8f8ff'), 'Partly Cloudy': ('', '#f8f8ff')}`) – 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
- interval** (default: 1800)
- format** (default: `{current_temp}{temp_unit}`)
- on_leftclick** (default: `open_forecast_url`) – 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](#))
- **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_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](#))
- **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*)
- **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 formatters

- *{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](#))
- **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"])`

Settings

- **format** (default: `{name}`) – 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`) – 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](#))
- 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
- 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](#))
- **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.

2.1 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, )
```

- *imap*
- *maildir*
- *mbx*
- *notmuchmail*
- *thunderbird*

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.

2.2 Score Backends

- `epl`
- `mlb`
- `nba`
- `nhl`

class `i3pystatus.scores.epl.EPL`

Backend to retrieve scores from the English Premier League. For usage examples, see [here](#).

Promotion / Relegation

Due to promotion/relegation, the `team_colors` configuration will eventually become out of date. When this happens, it will be necessary to manually set the colors for the newly-promoted teams until the source for this module is updated. An example of setting colors for newly promoted teams can be seen below:

```
from i3pystatus import Status
from i3pystatus.scores import epl

status = Status()

status.register(
    'scores',
    hints={'markup': 'pango'},
    colorize_teams=True,
    backends=[
        epl.EPL(
            teams=['LIV'],
            team_colors={
                'ABC': '#1D78CA',
                'DEF': '#8AFEC3',
                'GHI': '#33FA6D',
            },
        ),
    ],
    1,
)

status.run()
```

Available formatters

- `{home_name}` — Name of home team (e.g. **Tottenham Hotspur**)
- `{home_name_short}` — Shortened team name (e.g. **Spurs**)
- `{home_abbrev}` — 2 or 3-letter abbreviation for home team's city (e.g. **TOT**)
- `{home_score}` — Home team's current score
- `{home_wins}` — Home team's number of wins
- `{home_losses}` — Home team's number of losses
- `{home_draws}` — Home team's number of draws
- `{home_points}` — Home team's number of standings points

- {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_name}* — Name of away team (e.g. **Manchester United**)
- {away_name_short}* — Name of away team's city (e.g. **Man Utd**)
- {away_abbrev}* — **2 or 3-letter abbreviation for away team's name (e.g. MUN)**
- {away_score}* — Away team's current score
- {away_wins}* — Away team's number of wins
- {away_losses}* — Away team's number of losses
- {away_draws}* — Away team's number of draws
- {away_points}* — Away team's number of standings points
- {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.
- {minute}* — Current minute of game when in progress
- {start_time}* — Start time of game in system's localtime (supports strftime formatting, e.g. *{start_time:%I:%M %p}*)

Team abbreviations

- ARS** — Arsenal
- AVL** — Aston Villa
- BOU** — Bournemouth
- CHE** — Chelsea
- CRY** — Crystal Palace
- EVE** — Everton
- LEI** — Leicester City
- LIV** — Liverpool
- MCI** — Manchester City
- MUN** — Manchester United
- NEW** — Newcastle United
- NOR** — Norwich City
- SOU** — Southampton
- STK** — Stoke City
- SUN** — Sunderland Association
- SWA** — Swansea City
- TOT** — Tottenham Hotspur
- WAT** — Watford
- WBA** — West Bromwich Albion
- WHU** — West Ham United

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']`) – 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: `EPL: No games`) – Format used when no tracked games are scheduled for the current day (does not support formatter placeholders)
- **format_pregame** (default: `[{scroll}] EPL: [{away_favorite}] {away_abbrev} {away_points}, {away_wins}-{away_losses}-{away_draws}) at [{home_favorite}] {home_abbrev} ({home_points}, {home_wins}-{home_losses}-{home_draws}) {start_time:%H:%M %Z})` – Format used when the game has not yet started
- **format_in_progress** (default: `[{scroll}] EPL: [{away_favorite}] {away_abbrev} {away_score} [({away_power_play})], [{home_favorite}] {home_abbrev} {home_score} [({home_power_play})] ({minute})`) – Format used when the game is in progress
- **format_final** (default: `[{scroll}] EPL: [{away_favorite}] {away_abbrev} {away_score} ({away_points}, {away_wins}-{away_losses}-{away_draws}) at [{home_favorite}] {home_abbrev} {home_score} ({home_points}, {home_wins}-{home_losses}-{home_draws}) (Final)`) – Format used when the game is complete
- **team_colors** (default: `{'WHU': '#9DE4FA', 'AVL': '#94BEE5', 'MUN': '#DD1921', 'CRY': '#195FAF', 'MCI': '#74B2E0', 'WAT': '#E4D500', 'LEI': '#304FB6', 'CHE': '#195FAF', 'BOU': '#CB0B0F', 'NOR': '#00A651', 'LIV': '#D72129', 'TOT': '#DADADA', 'WBA': '#B43C51', 'ARS': '#ED1B22', 'STK': '#D81732', 'SOU': '#DB1C26', 'SWA': '#B28250', 'NEW': '#06B3EB', 'SUN': '#BC0007', 'EVE': '#004F9E'}`) – 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.
- **date** (default: *empty*) – Date for which to display game scores, in **YYYY-MM-DD** format. If unspecified, the date will be determined by the return value of an API call to the **context_url**. Due to API limitations, the date can presently only be overridden to another date in the current week. This option exists primarily for troubleshooting purposes.
- **live_url** (default: `http://live.premierleague.com/#/gameweek/%s/matchday/%s/match/%s`) – URL string to launch EPL Live Match Centre. This value should not need to be changed.
- **scoreboard_url** (default: `http://live.premierleague.com/`) – Link to the EPL scoreboard page. Like **live_url**, this value should not need to be changed.
- **api_url** (default: `http://live.premierleague.com/syndicationdata/competitionId=%s/seasonId`) – Alternate URL string from which to retrieve score data. Like **live_url**, this value should not need to be changed.
- **stats_url** (default: `http://live.premierleague.com/syndicationdata/competitionId=%s/seasonId`) – Alternate URL string from which to retrieve team statistics. Like **live_url**, this value should not need to be changed.

- **match_details_url** (default: `http://live.premierleague.com/syndicationdata/competitionId=%s/`)
– Alternate URL string from which to retrieve match details. 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.mlb.MLB`

Backend to retrieve MLB scores. For usage examples, see [here](#).

Available formatters

- `{home_name}` — Name of home team
- `{home_city}` — Name of home team's city
- `{home_abbrev}` — 2 or 3-letter abbreviation for home team's 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_name}` — Name of away team
- `{away_city}` — Name of away team's city
- `{away_abbrev}` — 2 or 3-letter abbreviation for away team's 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.
- `{extra_innings}` — When a game lasts longer than 9 innings, this formatter will show that number of innings. Otherwise, it will be blank.

Team abbreviations

- ARI** — Arizona Diamondbacks
- ATL** — Atlanta Braves
- BAL** — Baltimore Orioles
- BOS** — Boston Red Sox
- CHC** — Chicago Cubs
- CIN** — Cincinnati Reds
- CLE** — Cleveland Indians
- 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
- NYN** — 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', 'postponed', 'pregame']`) – 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_pregame** (default: `[{scroll}] MLB: [{away_favorite}] {away_abbrev} ({away_wins}-{away_losses}) at [{home_favorite}] {home_abbrev} ({home_wins}-{home_losses}) {start_time:%H:%M %Z} [({delay} Delay)]`) – Format used when the game has not yet started
- **format_in_progress** (default: `[{scroll}] MLB: [{away_favorite}] {away_abbrev} {away_score}, [{home_favorite}] {home_abbrev} {home_score} ({top_bottom} {inning}, {outs} Out) [({delay} Delay)]`) – Format used when the game is in progress
- **format_final** (default: `[{scroll}] MLB: [{away_favorite}] {away_abbrev} {away_score} ({away_wins}-{away_losses}) at [{home_favorite}] {home_abbrev} {home_score} ({home_wins}-{home_losses}) (Final[/ {extra_innings}])`) – Format used when the game is complete
- **format_postponed** (default: `[{scroll}] MLB: [{away_favorite}] {away_abbrev} ({away_wins}-{away_losses}) at [{home_favorite}] {home_abbrev} ({home_wins}-{home_losses}) (PPD: {postponed})`) – Format used when the game has been postponed
- **format_suspended** (default: `[{scroll}] MLB: [{away_favorite}] {away_abbrev} {away_score} ({away_wins}-{away_losses}) at [{home_favorite}] {home_abbrev} {home_score} ({home_wins}-{home_losses}) (Suspended: {suspended})`) – Format used 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: `{'TOR': '#0046DD', 'OAK': '#006659', 'KC': '#0046DD', 'ATL': '#CE1141', 'LAA': '#BA0021', 'NYM': '#FF5910', 'TEX': '#C0111F', 'MIN': '#D31145', 'PIT': '#FFCC01', 'BOS': '#BD3039', 'BAL': '#DF4601', 'COL': '#5E5EB6', 'HOU': '#EB6E1F', 'SF': '#FD5A1E', 'PHI': '#E81828', 'STL': '#B53B30', 'CLE': '#E31937', 'WSH': '#C70003', 'CIN': '#C6011F', 'CHC': '#004EC1', 'MIA': '#F14634', 'CWS': '#DADADA', 'SD': '#285F9A', 'DET': '#FF6600', 'TB': '#8FBCE6', 'ARI': '#A71930', 'LAD': '#005A9C', 'NYY': '#0747CC', 'SEA': '#2E8B90', 'MIL': '#0747CC'}`) – 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.
- **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: `http://mlb.mlb.com/mlb/gameday/index.jsp?gid=%s`) – 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: `http://gd2.mlb.com/components/game/mlb/year_%04d/month_%02d/day_%02d/mi`) – 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_name}** — Name of home team
- {home_city}** — Name of home team's city
- {home_abbrev}** — 3-letter abbreviation for home team's 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_name}** — Name of away team
- {away_city}** — Name of away team's city
- {away_abbrev}** — 2 or 3-letter abbreviation for away team's 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
- {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, 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']`) – 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_pregame** (default: `[[{scroll}] NBA: [{away_favorite}] [{away_seed}] {away_abbrev} ({away_wins}-{away_losses}) at [{home_favorite}] [{home_seed}] {home_abbrev} ({home_wins}-{home_losses}) {start_time:%H:%M %Z}`) – Format used when the game has not yet started
- **format_in_progress** (default: `[[{scroll}] NBA: [{away_favorite}] [{away_abbrev}] {away_score} [({away_power_play})], [{home_favorite}] {home_abbrev} {home_score} [({home_power_play})] ({time_remaining} {quarter})`) – Format used when the game is in progress
- **format_final** (default: `[[{scroll}] NBA: [{away_favorite}] [{away_abbrev}] {away_score} ({away_wins}-{away_losses}) at [{home_favorite}] {home_abbrev} {home_score} ({home_wins}-{home_losses}) (Final [/{overtime}])`) – Format used when the game is complete
- **team_colors** (default: `{ 'TOR': '#CD112C', 'ORL': '#1980CB', 'CHA': '#00798D', 'DEN': '#5593C3', 'POR': '#B03037', 'ATL': '#E2383F', 'UTA': '#4B7059', 'PHX': '#E76120', 'DET': '#207EC0', 'MIL': '#4C7B4B', 'BOS': '#178D58', 'NYK': '#F68428', 'HOU': '#CD1042', 'NOP': '#A78F59', 'PHI': '#006BB7', 'MEM': '#628BBC', 'CLE': '#FDBA31', 'MIA': '#A72249', 'CHI': '#CD1041', 'IND': '#FFBB33', 'GSW': '#DEB934', 'SAC': '#7A58A1', 'DAL': '#006BB7', 'LAC': '#ED174C', 'BKN': '#DADADA', 'LAL': '#FDB827', 'OKC': '#F05033', 'WAS': '#E51735', 'SAS': '#DADADA', 'MIN': '#35749F' }`) – 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.
- **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: `http://www.nba.com/gametracker/#/%s/lp`) – URL string to launch NBA Game Tracker. This value should not need to be changed.
- **scoreboard_url** (default: `http://www.nba.com/scores`) – Link to the NBA.com scoreboard page. Like **live_url**, this value should not need to be changed.
- **api_url** (default: `http://data.nba.com/data/10s/json/cms/noseason/scoreboard/%04d%02d%02d/`) – Alternate URL string from which to retrieve score data. Like, **live_url**, this value should not need to be changed.
- **standings_url** (default: `http://data.nba.com/data/json/cms/%s/league/standings.json`) – Alternate URL string from which to retrieve team standings. 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_name}` — Name of home team
- `{home_city}` — Name of home team's city
- `{home_abbrev}` — 3-letter abbreviation for home team's 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_name}` — Name of away team
- `{away_city}` — Name of away team's city
- `{away_abbrev}` — 2 or 3-letter abbreviation for away team's 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_pregame = ' [{scroll} ]NHL: [{away_favorite} ]{away_abbrev} ({away_wins}) at [
        format_final = ' [{scroll} ]NHL: [{away_favorite} ]{away_abbrev} {away_score} ({away_
    ),
],
)

```

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
- SJS** — San Jose Sharks
- STL** — St. Louis Blues
- TBL** — Tampa Bay Lightning

- TOR** — Toronto Maple Leafs
- VAN** — Vancouver Canucks
- 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']`) – 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_pregame** (default: `[{scroll}] NHL: [{away_favorite}] [{away_abbrev}] ({away_wins}-{away_losses}-{away_otl}) at [{home_favorite}] [{home_abbrev}] ({home_wins}-{home_losses}-{home_otl}) {start_time:%H:%M %Z}`) – Format used when the game has not yet started

•**format_in_progress** (default: `[{scroll}] NHL: [{away_favorite}] [{away_abbrev}] {away_score} [({away_power_play})] [({away_empty_net})], [{home_favorite}] [{home_abbrev}] {home_score} [({home_power_play})] [({home_empty_net})] ({time_remaining} {period})`) – Format used when the game is in progress

•**format_final** (default: `[{scroll}] NHL: [{away_favorite}] [{away_abbrev}] {away_score} ({away_wins}-{away_losses}-{away_otl}) at [{home_favorite}] [{home_abbrev}] {home_score} ({home_wins}-{home_losses}-{home_otl}) (Final [/{overtime}])`) – Format used when the game is complete

•**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', 'WPG': '#1568C5', 'OTT': '#C50B2F', 'CGY': '#D23429', 'PIT': '#D9CBAE', 'TBL': '#296AD5', 'EDM': '#2F6093', 'FLA': '#E51837', 'TOR': '#296AD5', 'DET': '#E51937', 'SJS': '#007888', 'CAR': '#FA272E', 'MTL': '#C8011D', 'BOS': '#F6BD27', 'LAK': '#DADADA', 'NJD': '#CC0000', 'BUF': '#1568C5', 'NSH': '#FDB71A', 'CBJ': '#1568C5', 'PHI': '#FF690B', 'VAN': '#0454FA', 'COL': '#9F415B', 'CHI': '#CD0E24', 'NYI': '#F8630D', 'WSH': '#E51937', 'DAL': '#058158', 'NYR': '#1576CA', 'ARI': '#AC313A', 'STL': '#1764AD', 'MIN': '#176B49'}`) – 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.

•**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/%s`) – 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=%04d-%02d-%02d&`) – 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.

2.3 Update Backends

- *aptget*
- *cower*
- *dnf*
- *pacman*
- *yaourt*

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.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 with *dnf check-update*.

The notification body consists of the status line followed by the package name and version for each update.

https://dnf.readthedocs.org/en/latest/command_ref.html#check-update-command

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* 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()])
```

If you want to count both pacman and aur packages with this module you can set the variable `count_only_aur = False` like this:

```
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.

2.4 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 the location on weather.com, and the location code will be everything after the last slash (e.g. 94107:4:US).

Usage example

```
from i3pystatus import Status
from i3pystatus.weather import weathercom

status = Status()

status.register(
    'weather',
    format='{condition} {current_temp}{temp_unit}{icon}[ Hi: {high_temp}] Lo: {low_temp}',
    colorize=True,
    backend=weathercom.Weathercom(
        location_code='94107:4:US',
        units='imperial',
    ),
)

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

`weather_data()`

Fetches the current weather from `wxdata.weather.com` service.

`class i3pystatus.weather.wunderground.Wunderground`

This module retrieves weather data using the Weather Underground API.

Note: A Weather Underground API key is required to use this module, you can sign up for a developer API key free at <https://www.wunderground.com/weather/api/>

A developer API key is allowed 500 queries per day, and no more than 10 in a given minute. Therefore, it is recommended to be conservative when setting the update interval.

Valid values for `location_code` include:

- **State/City_Name** - CA/San_Francisco
- **Country/City** - France/Paris
- **Geolocation by IP** - autoip
- **Zip or Postal Code** - 60616
- **ICAO Airport Code** - icao:LAX
- **Latitude/Longitude** - 41.8301943,-87.6342619
- **Personal Weather Station (PWS)** - pws:KILCHICA30

When not using a `pws` or `icao` station ID, the location will be queried, and the closest station will be used. For a list of PWS station IDs, visit the following URL:

<http://www.wunderground.com/weatherstation/ListStations.asp>

Usage example

```
from i3pystatus import Status
from i3pystatus.weather import wunderground

status = Status()

status.register(
    'weather',
    format='{condition} {current_temp}{temp_unit}{icon} [ Hi: {high_temp}] Lo: {low_temp}',
    colorize=True,
    backend=wunderground.Wunderground(
        api_key='dbafe887d56ba4ad',
        location_code='pws:MAT645',
        units='imperial',
    ),
)
```

```
status.run()
```

See [here](#) for a list of formatters which can be used.

Settings

- **api_key** (required) – Weather Underground API key
- **location_code** (required) – Location code from wunderground.com
- **units** (default: `metric`) – ‘metric’ or ‘imperial’
- **use_pws** (default: `True`) – Set to `False` to use only airport stations
- **forecast** (default: `False`) – Set to `True` to check forecast (generates one additional API request per weather update). If set to `False`, then the `low_temp` and `high_temp` formatters will be set to empty strings.
- **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](#))
- **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.

api_request (*url*)

Execute an HTTP POST to the specified URL and return the content

geolookup ()

Use the `location_code` to perform a geolookup and find the closest station. If the location is a pws or icao station ID, no lookup will be performed.

get_forecast ()

If configured to do so, make an API request to retrieve the forecast data for the configured/queried weather station, and return the low and high temperatures. Otherwise, return two empty strings.

weather_data ()

Query the configured/queried station and return the weather data

Changelog

3.1 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 development branch)
- *clock*: timezone-related fixes with multiple clocks
- *dpms*: Added *format_disabled* option
- *github*: Added support for access 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* $\geq 0.10.5$ 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

3.2 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

3.3 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

3.4 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

3.5 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

3.6 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 “formatted_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

3.7 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

3.8 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.

3.9 3.27 (2013-10-20)

- Add *weather* module
- Add *text* module
- *pulseaudio*: Add muted/unmuted options

3.10 3.26 (2013-10-03)

- Add *mem* module

3.11 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

Creating modules

Creating new modules (“things that display something”) to contribute to `i3pystatus` is reasonably easy. If the module you want to write updates its 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.
- 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, its 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

4.1 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.

4.2 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.

core Package

5.1 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*, *logformat='%%(asctime)s [%(levelname)-8s][%(name)s %(lineno)d] %(message)s'*)

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*.

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.

5.2 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'

5.3 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 to *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.

5.4 desktop Module

```
class i3pystatus.core.desktop.BaseDesktopNotification(title, body, icon='dialog-
information', urgency=1, time-
out=0)
```

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

```
class i3pystatus.core.desktop.DesktopNotification(title, body, icon='dialog-information',
urgency=1, timeout=0)
```

Bases: *i3pystatus.core.desktop.BaseDesktopNotification*

5.5 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)
```

5.6 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)
    instantiate_class_from_module (module, *args, **kwargs)
    predicate_factory (module)
```

5.7 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, 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 mod-
        ules 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.
```

5.8 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_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 = None
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 *&amp;*;

`i3pystatus.core.modules.is_method_of (method, object)`

Decide whether *method* is contained within the MRO of *object*.

5.9 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 ()
```

5.10 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)

    start ()

    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 ()

    run ()

    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
```

5.11 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>\n) | # Escape sequence of two delimiters\n (?P<named>[a-zA-Z])`

`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.

Used server is determined by the `address` class variable which consists of server host name and port number.

Return type `bool`

See also:

`require()`

`address = ('google-public-dns-a.google.com', 53)`

`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_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*)

Draws a vertical bar made of unicode characters.

Parameters

- **value** – 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, othewise 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

Indices and tables

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

i

`i3pystatus.abc_radio`, 15
`i3pystatus.alsa`, 16
`i3pystatus.anybar`, 17
`i3pystatus.backlight`, 18
`i3pystatus.battery`, 19
`i3pystatus.bitcoin`, 21
`i3pystatus.clock`, 22
`i3pystatus.cmus`, 23
`i3pystatus.core`, 113
`i3pystatus.core.color`, 114
`i3pystatus.core.command`, 114
`i3pystatus.core.desktop`, 115
`i3pystatus.core.exceptions`, 115
`i3pystatus.core.imputil`, 116
`i3pystatus.core.io`, 116
`i3pystatus.core.modules`, 117
`i3pystatus.core.settings`, 119
`i3pystatus.core.threading`, 120
`i3pystatus.core.util`, 120
`i3pystatus.cpu_freq`, 25
`i3pystatus.cpu_usage`, 25
`i3pystatus.cpu_usage_bar`, 26
`i3pystatus.cpu_usage_graph`, 28
`i3pystatus.disk`, 29
`i3pystatus.dota2wins`, 29
`i3pystatus.dpms`, 30
`i3pystatus.file`, 31
`i3pystatus.github`, 32
`i3pystatus.google_calendar`, 33
`i3pystatus.gpu_mem`, 34
`i3pystatus.gpu_temp`, 35
`i3pystatus.iinet`, 36
`i3pystatus.keyboard_locks`, 37
`i3pystatus.lastfm`, 38
`i3pystatus.load`, 39
`i3pystatus.mail`, 39
`i3pystatus.mail.imap`, 85
`i3pystatus.mail.maildir`, 86
`i3pystatus.mail.mbox`, 86
`i3pystatus.mail.notmuchmail`, 86
`i3pystatus.mail.thunderbird`, 86
`i3pystatus.makewatch`, 40
`i3pystatus.mem`, 41
`i3pystatus.mem_bar`, 42
`i3pystatus.modsde`, 43
`i3pystatus.moon`, 44
`i3pystatus.mpd`, 44
`i3pystatus.net_speed`, 46
`i3pystatus.network`, 47
`i3pystatus.now_playing`, 49
`i3pystatus.online`, 50
`i3pystatus.openfiles`, 50
`i3pystatus.openstack_vms`, 51
`i3pystatus.openvpn`, 52
`i3pystatus.parcel`, 53
`i3pystatus.pianobar`, 54
`i3pystatus.ping`, 54
`i3pystatus.plexstatus`, 55
`i3pystatus.pomodoro`, 56
`i3pystatus.pulseaudio`, 57
`i3pystatus.pyload`, 58
`i3pystatus.reddit`, 59
`i3pystatus.regex`, 61
`i3pystatus.runwatch`, 61
`i3pystatus.scores`, 62
`i3pystatus.scores.epl`, 87
`i3pystatus.scores.mlb`, 90
`i3pystatus.scores.nba`, 93
`i3pystatus.scores.nhl`, 95
`i3pystatus.scratchpad`, 66
`i3pystatus.sge`, 66
`i3pystatus.shell`, 67
`i3pystatus.solaar`, 68
`i3pystatus.spotify`, 69
`i3pystatus.syncthing`, 70
`i3pystatus.taskwarrior`, 71
`i3pystatus.temp`, 72
`i3pystatus.text`, 72
`i3pystatus.timer`, 73
`i3pystatus.uname`, 75

- `i3pystatus.updates`, 76
- `i3pystatus.updates.aptget`, 99
- `i3pystatus.updates.cower`, 99
- `i3pystatus.updates.dnf`, 99
- `i3pystatus.updates.pacman`, 99
- `i3pystatus.updates.yaourt`, 100
- `i3pystatus.uptime`, 77
- `i3pystatus.vk`, 78
- `i3pystatus.weather`, 79
- `i3pystatus.weather.weathercom`, 100
- `i3pystatus.weather.wunderground`, 101
- `i3pystatus.weekcal`, 81
- `i3pystatus.whosonlocation`, 81
- `i3pystatus.window_title`, 82
- `i3pystatus.xkblayout`, 83
- `i3pystatus.zabbix`, 84

A

ABCRadio (class in `i3pystatus.abc_radio`), 15
 address (`i3pystatus.core.util.internet` attribute), 122
 ALSA (class in `i3pystatus.alsa`), 16
 AnyBar (class in `i3pystatus.anybar`), 17
 api_request() (`i3pystatus.weather.wunderground.Wunderground` method), 102
 append() (`i3pystatus.core.threading.Manager` method), 120
 append() (`i3pystatus.core.threading.Thread` method), 120
 append() (`i3pystatus.core.util.ModuleList` method), 121
 AptGet (class in `i3pystatus.updates.aptget`), 99
 async_refresh() (`i3pystatus.core.io.StandaloneIO` method), 117

B

Backlight (class in `i3pystatus.backlight`), 18
 BaseDesktopNotification (class in `i3pystatus.core.desktop`), 115
 BatteryChecker (class in `i3pystatus.battery`), 19
 Bitcoin (class in `i3pystatus.bitcoin`), 21
 branch() (`i3pystatus.core.threading.Thread` method), 120

C

calculate_usage() (`i3pystatus.cpu_usage.CpuUsage` method), 26
 check_double() (`i3pystatus.core.util.MultiClickHandler` method), 121
 ClassFinder (class in `i3pystatus.core.imputil`), 116
 clear_timer() (`i3pystatus.core.util.MultiClickHandler` method), 121
 Clock (class in `i3pystatus.clock`), 22
 Cmus (class in `i3pystatus.cmus`), 23
 ColorRangeModule (class in `i3pystatus.core.color`), 114
 CommandEndpoint (class in `i3pystatus.core`), 113
 CommandResult (in module `i3pystatus.core.command`), 114
 compute_treshold_interval() (`i3pystatus.core.io.StandaloneIO` method), 117

ConfigAmbigiousClassesError, 115

ConfigError, 115

ConfigInvalidModuleError, 115

ConfigKeyError, 116

ConfigMissingError, 116

context_notify_cb() (`i3pystatus.pulseaudio.PulseAudio` method), 58

convert_position() (in module `i3pystatus.core.util`), 121

Cower (class in `i3pystatus.updates.cower`), 99

CpuFreq (class in `i3pystatus.cpu_freq`), 25

CpuUsage (class in `i3pystatus.cpu_usage`), 25

CpuUsageBar (class in `i3pystatus.cpu_usage_bar`), 26

CpuUsageGraph (class in `i3pystatus.cpu_usage_graph`), 28

create_thread() (`i3pystatus.core.threading.Manager` method), 120

create_threads() (`i3pystatus.core.threading.Manager` method), 120

createvaluesdict() (`i3pystatus.cpu_freq.CpuFreq` method), 25

cycle_interface() (`i3pystatus.network.Network` method), 49

D

delimiter (`i3pystatus.core.util.TimeWrapper.TimeTemplate` attribute), 121

DesktopNotification (class in `i3pystatus.core.desktop`), 115

Disk (class in `i3pystatus.disk`), 29

display() (`i3pystatus.core.desktop.BaseDesktopNotification` method), 115

Dnf (class in `i3pystatus.updates.dnf`), 99

Dota2wins (class in `i3pystatus.dota2wins`), 29

DPMS (class in `i3pystatus.dpms`), 30

E

end_color (`i3pystatus.core.color.ColorRangeModule` attribute), 114

EPL (class in `i3pystatus.scores.epl`), 87

ExceptionWrapper (class in `i3pystatus.core.threading`), 120

execute() (in module i3pystatus.core.command), 114
 execute_workloads() (i3pystatus.core.threading.Thread method), 120

F

File (class in i3pystatus.file), 31
 flatten() (in module i3pystatus.core.util), 121
 flatten_settings() (i3pystatus.core.settings.SettingsBase static method), 119
 format() (i3pystatus.core.exceptions.ConfigAmbigiousClassesError method), 115
 format() (i3pystatus.core.exceptions.ConfigError method), 115
 format() (i3pystatus.core.exceptions.ConfigInvalidModuleError method), 116
 format() (i3pystatus.core.exceptions.ConfigKeyError method), 116
 format() (i3pystatus.core.exceptions.ConfigMissingError method), 116
 format_exception() (i3pystatus.core.threading.ExceptionWrapper method), 120
 formatp() (in module i3pystatus.core.util), 122

G

gen_format_all() (i3pystatus.cpu_usage.CpuUsage method), 26
 geolookup() (i3pystatus.weather.wunderground.Wunderground method), 102
 get() (i3pystatus.core.util.ModuleList method), 121
 get_class() (i3pystatus.core.imputil.ClassFinder method), 116
 get_color_data() (i3pystatus.weather.Weather method), 81
 get_cpu_timings() (i3pystatus.cpu_usage.CpuUsage method), 26
 get_forecast() (i3pystatus.weather.wunderground.Wunderground method), 102
 get_gradient() (i3pystatus.core.color.ColorRangeModule method), 114
 get_hex_color_range() (i3pystatus.core.color.ColorRangeModule static method), 114
 get_info() (i3pystatus.spotify.Spotify method), 69
 get_matching_classes() (i3pystatus.core.imputil.ClassFinder method), 116
 get_merged_settings() (i3pystatus.core.settings.SettingsBase static method), 120
 get_module() (i3pystatus.core.imputil.ClassFinder method), 116
 get_module() (in module i3pystatus.core.util), 122
 get_protected_settings() (i3pystatus.core.settings.SettingsBase method), 119
 get_setting_from_keyring() (i3pystatus.core.settings.SettingsBase method), 119

get_usage() (i3pystatus.cpu_usage.CpuUsage method), 26
 Github (class in i3pystatus.github), 32
 GoogleCalendar (class in i3pystatus.google_calendar), 33
 GPUmemory (class in i3pystatus.gpu_mem), 34
 GPUTemperature (class in i3pystatus.gpu_temp), 35

H

hints (i3pystatus.core.modules.Module attribute), 117

I

i3pystatus.abc_radio (module), 15
 i3pystatus.alsa (module), 16
 i3pystatus.anybar (module), 17
 i3pystatus.backlight (module), 18
 i3pystatus.battery (module), 19
 i3pystatus.bitcoin (module), 21
 i3pystatus.clock (module), 22
 i3pystatus.cmus (module), 23
 i3pystatus.core (module), 113
 i3pystatus.core.color (module), 114
 i3pystatus.core.command (module), 114
 i3pystatus.core.desktop (module), 115
 i3pystatus.core.exceptions (module), 115
 i3pystatus.core.imputil (module), 116
 i3pystatus.core.io (module), 116
 i3pystatus.core.modules (module), 117
 i3pystatus.core.settings (module), 119
 i3pystatus.core.threading (module), 120
 i3pystatus.core.util (module), 120
 i3pystatus.cpu_freq (module), 25
 i3pystatus.cpu_usage (module), 25
 i3pystatus.cpu_usage_bar (module), 26
 i3pystatus.cpu_usage_graph (module), 28
 i3pystatus.disk (module), 29
 i3pystatus.dota2wins (module), 29
 i3pystatus.dpms (module), 30
 i3pystatus.file (module), 31
 i3pystatus.github (module), 32
 i3pystatus.google_calendar (module), 33
 i3pystatus.gpu_mem (module), 34
 i3pystatus.gpu_temp (module), 35
 i3pystatus.iinet (module), 36
 i3pystatus.keyboard_locks (module), 37
 i3pystatus.lastfm (module), 38
 i3pystatus.load (module), 39
 i3pystatus.mail (module), 39
 i3pystatus.mail.imap (module), 85
 i3pystatus.mail.maildir (module), 86
 i3pystatus.mail.mbox (module), 86
 i3pystatus.mail.notmuchmail (module), 86
 i3pystatus.mail.thunderbird (module), 86
 i3pystatus.makewatch (module), 40
 i3pystatus.mem (module), 41

i3pystatus.mem_bar (module), 42
 i3pystatus.modsde (module), 43
 i3pystatus.moon (module), 44
 i3pystatus.mpd (module), 44
 i3pystatus.net_speed (module), 46
 i3pystatus.network (module), 47
 i3pystatus.now_playing (module), 49
 i3pystatus.online (module), 50
 i3pystatus.openfiles (module), 50
 i3pystatus.openstack_vms (module), 51
 i3pystatus.openvpn (module), 52
 i3pystatus.parcel (module), 53
 i3pystatus.pianobar (module), 54
 i3pystatus.ping (module), 54
 i3pystatus.plexstatus (module), 55
 i3pystatus.pomodoro (module), 56
 i3pystatus.pulseaudio (module), 57
 i3pystatus.pyload (module), 58
 i3pystatus.reddit (module), 59
 i3pystatus.regex (module), 61
 i3pystatus.runwatch (module), 61
 i3pystatus.scores (module), 62
 i3pystatus.scores.epl (module), 87
 i3pystatus.scores.mlb (module), 90
 i3pystatus.scores.nba (module), 93
 i3pystatus.scores.nhl (module), 95
 i3pystatus.scratchpad (module), 66
 i3pystatus.sge (module), 66
 i3pystatus.shell (module), 67
 i3pystatus.solaar (module), 68
 i3pystatus.spotify (module), 69
 i3pystatus.syncthing (module), 70
 i3pystatus.taskwarrior (module), 71
 i3pystatus.temp (module), 72
 i3pystatus.text (module), 72
 i3pystatus.timer (module), 73
 i3pystatus.uname (module), 75
 i3pystatus.updates (module), 76
 i3pystatus.updates.aptget (module), 99
 i3pystatus.updates.cower (module), 99
 i3pystatus.updates.dnf (module), 99
 i3pystatus.updates.pacman (module), 99
 i3pystatus.updates.yaourt (module), 100
 i3pystatus.uptime (module), 77
 i3pystatus.vk (module), 78
 i3pystatus.weather (module), 79
 i3pystatus.weather.weathercom (module), 100
 i3pystatus.weather.wunderground (module), 101
 i3pystatus.weekcal (module), 81
 i3pystatus.whosonlocation (module), 81
 i3pystatus.window_title (module), 82
 i3pystatus.xkblayout (module), 83
 i3pystatus.zabbix (module), 84

idpattern (i3pystatus.core.util.TimeWrapper.TimeTemplate attribute), 121
 IINet (class in i3pystatus.iinet), 36
 IMAP (class in i3pystatus.mail.imap), 85
 imap_class (i3pystatus.mail.imap.IMAP attribute), 85
 increase() (i3pystatus.timer.Timer method), 75
 init() (i3pystatus.core.settings.SettingsBase method), 119
 init() (i3pystatus.pulseaudio.PulseAudio method), 58
 inject() (i3pystatus.core.modules.Module method), 117
 instantiate_class_from_module()
 (i3pystatus.core.imputil.ClassFinder method), 116
 internet (class in i3pystatus.core.util), 122
 interval (i3pystatus.core.modules.IntervalModule attribute), 117
 IntervalModule (class in i3pystatus.core.modules), 117
 IOHandler (class in i3pystatus.core.io), 116
 is_method_of() (in module i3pystatus.core.modules), 119

J

JSONIO (class in i3pystatus.core.io), 116

K

Keyboard_locks (class in i3pystatus.keyboard_locks), 37
 KeyConstraintDict (class in i3pystatus.core.util), 120
 KeyConstraintDict.MissingKeys, 121

L

LastFM (class in i3pystatus.lastfm), 38
 lchop() (in module i3pystatus.core.util), 122
 Load (class in i3pystatus.load), 39
 log_level (i3pystatus.core.settings.SettingsBase attribute), 119
 logger (i3pystatus.core.settings.SettingsBase attribute), 119

M

Mail (class in i3pystatus.mail), 39
 MaildirMail (class in i3pystatus.mail.maildir), 86
 main_loop() (i3pystatus.anybar.AnyBar method), 18
 make_bar() (in module i3pystatus.core.util), 122
 make_graph() (in module i3pystatus.core.util), 123
 make_vertical_bar() (in module i3pystatus.core.util), 123
 MakeWatch (class in i3pystatus.makewatch), 40
 Manager (class in i3pystatus.core.threading), 120
 managers (i3pystatus.core.modules.IntervalModule attribute), 117
 MboxMail (class in i3pystatus.mail.mbox), 86
 Mem (class in i3pystatus.mem), 41
 MemBar (class in i3pystatus.mem_bar), 42
 missing()
 (i3pystatus.core.util.KeyConstraintDict method), 121
 MLB (class in i3pystatus.scores.mlb), 90
 ModsDeChecker (class in i3pystatus.modsde), 43

Module (class in `i3pystatus.core.modules`), 117
 ModuleList (class in `i3pystatus.core.util`), 121
 MoonPhase (class in `i3pystatus.moon`), 44
 move() (`i3pystatus.core.modules.Module` method), 117
 MPD (class in `i3pystatus.mpd`), 44
 multi_click_timeout (`i3pystatus.core.modules.Module` attribute), 117
 MultiClickHandler (class in `i3pystatus.core.util`), 121

N

n (`i3pystatus.core.io.StandaloneIO` attribute), 117
 NBA (class in `i3pystatus.scores.nba`), 93
 NetSpeed (class in `i3pystatus.net_speed`), 46
 Network (class in `i3pystatus.network`), 47
 next_song() (`i3pystatus.spotify.Spotify` method), 70
 NHL (class in `i3pystatus.scores.nhl`), 95
 Notmuch (class in `i3pystatus.mail.notmuchmail`), 86
 NowPlaying (class in `i3pystatus.now_playing`), 49

O

on_click() (`i3pystatus.core.modules.Module` method), 117
 on_doubledownscroll (`i3pystatus.core.modules.Module` attribute), 118
 on_doubleleftclick (`i3pystatus.core.modules.Module` attribute), 118
 on_doublemiddleclick (`i3pystatus.core.modules.Module` attribute), 118
 on_doubleotherclick (`i3pystatus.core.modules.Module` attribute), 118
 on_doublerightclick (`i3pystatus.core.modules.Module` attribute), 118
 on_doubleupscroll (`i3pystatus.core.modules.Module` attribute), 118
 on_downscroll (`i3pystatus.core.modules.Module` attribute), 118
 on_leftclick (`i3pystatus.core.modules.Module` attribute), 118
 on_middleclick (`i3pystatus.core.modules.Module` attribute), 118
 on_otherclick (`i3pystatus.core.modules.Module` attribute), 118
 on_rightclick (`i3pystatus.core.modules.Module` attribute), 118
 on_upscroll (`i3pystatus.core.modules.Module` attribute), 118
 Online (class in `i3pystatus.online`), 50
 open_something() (`i3pystatus.bitcoin.Bitcoin` method), 22
 Openfiles (class in `i3pystatus.openfiles`), 50
 Openstack_vms (class in `i3pystatus.openstack_vms`), 51
 OpenVPN (class in `i3pystatus.openvpn`), 52
 output (`i3pystatus.core.modules.Module` attribute), 118

P

Pacman (class in `i3pystatus.updates.pacman`), 99
 ParcelTracker (class in `i3pystatus.parcel`), 53
 parse_line() (`i3pystatus.core.io.JSONIO` method), 116
 partition() (in module `i3pystatus.core.util`), 123
 partition_workloads() (`i3pystatus.core.threading.Manager` method), 120
 pattern (`i3pystatus.core.util.TimeWrapper.TimeTemplate` attribute), 121
 percentage() (`i3pystatus.core.color.ColorRangeModule` static method), 114
 Pianobar (class in `i3pystatus.pianobar`), 54
 Ping (class in `i3pystatus.ping`), 54
 playpause() (`i3pystatus.spotify.Spotify` method), 70
 Plexstatus (class in `i3pystatus.plexstatus`), 55
 Pomodoro (class in `i3pystatus.pomodoro`), 56
 pop() (`i3pystatus.core.threading.Thread` method), 120
 popwhile() (in module `i3pystatus.core.util`), 123
 position (`i3pystatus.core.modules.Module` attribute), 118
 predicate_factory() (`i3pystatus.core.imputil.ClassFinder` method), 116
 previous_song() (`i3pystatus.spotify.Spotify` method), 70
 proto (`i3pystatus.core.io.StandaloneIO` attribute), 117
 PulseAudio (class in `i3pystatus.pulseaudio`), 57
 pyLoad (class in `i3pystatus.pyload`), 58

R

read() (`i3pystatus.core.io.IOHandler` method), 116
 read() (`i3pystatus.core.io.JSONIO` method), 116
 read() (`i3pystatus.core.io.StandaloneIO` method), 117
 read_line() (`i3pystatus.core.io.IOHandler` method), 116
 read_line() (`i3pystatus.core.io.StandaloneIO` method), 117
 Reddit (class in `i3pystatus.reddit`), 59
 refresh_signal_handler() (`i3pystatus.core.io.StandaloneIO` method), 117
 Regex (class in `i3pystatus.regex`), 61
 register() (`i3pystatus.core.Status` method), 113
 registered() (`i3pystatus.core.modules.IntervalModule` method), 117
 registered() (`i3pystatus.core.modules.Module` method), 118
 request_update() (`i3pystatus.pulseaudio.PulseAudio` method), 58
 require() (in module `i3pystatus.core.util`), 123
 required (`i3pystatus.core.modules.IntervalModule` attribute), 117
 required (`i3pystatus.core.modules.Module` attribute), 118
 required (`i3pystatus.core.settings.SettingsBase` attribute), 119
 reset() (`i3pystatus.timer.Timer` method), 75
 round_dict() (in module `i3pystatus.core.util`), 123
 run() (`i3pystatus.core.modules.IntervalModule` method), 117

run() (i3pystatus.core.modules.Module method), 119
 run() (i3pystatus.core.Status method), 114
 run() (i3pystatus.core.threading.Thread method), 120
 run() (i3pystatus.mail.Mail method), 40
 run() (i3pystatus.spotify.Spotify method), 70
 run_through_shell() (in module i3pystatus.core.command), 114
 RunWatch (class in i3pystatus.runwatch), 61

S

Scores (class in i3pystatus.scores), 62
 Scratchpad (class in i3pystatus.scratchpad), 66
 send_output() (i3pystatus.core.modules.Module method), 119
 server_info_cb() (i3pystatus.pulseaudio.PulseAudio method), 58
 set_timer() (i3pystatus.core.util.MultiClickHandler method), 121
 settings (i3pystatus.core.modules.IntervalModule attribute), 117
 settings (i3pystatus.core.modules.Module attribute), 119
 settings (i3pystatus.core.settings.SettingsBase attribute), 119
 SettingsBase (class in i3pystatus.core.settings), 119
 SettingsBaseMeta (class in i3pystatus.core.settings), 119
 SGETracker (class in i3pystatus.sge), 66
 Shell (class in i3pystatus.shell), 67
 sink_info_cb() (i3pystatus.pulseaudio.PulseAudio method), 58
 Solaar (class in i3pystatus.solaar), 68
 Spotify (class in i3pystatus.spotify), 69
 st_open() (i3pystatus.syncthing.Syncthing method), 71
 st_restart() (i3pystatus.syncthing.Syncthing method), 71
 st_restart_systemd() (i3pystatus.syncthing.Syncthing method), 71
 st_start_systemd() (i3pystatus.syncthing.Syncthing method), 71
 st_stop() (i3pystatus.syncthing.Syncthing method), 71
 st_stop_systemd() (i3pystatus.syncthing.Syncthing method), 71
 st_toggle_systemd() (i3pystatus.syncthing.Syncthing method), 71
 StandaloneIO (class in i3pystatus.core.io), 116
 start() (i3pystatus.core.CommandEndpoint method), 113
 start() (i3pystatus.core.threading.Manager method), 120
 start() (i3pystatus.timer.Timer method), 75
 start_color (i3pystatus.core.color.ColorRangeModule attribute), 114
 Status (class in i3pystatus.core), 113
 Syncthing (class in i3pystatus.syncthing), 70

T

Taskwarrior (class in i3pystatus.taskwarrior), 71
 Temperature (class in i3pystatus.temp), 72

Text (class in i3pystatus.text), 72
 text_to_pango() (i3pystatus.core.modules.Module method), 119
 Thread (class in i3pystatus.core.threading), 120
 Thunderbird (class in i3pystatus.mail.thunderbird), 86
 time (i3pystatus.core.threading.Thread attribute), 120
 time (i3pystatus.core.threading.WorkloadWrapper attribute), 120
 Timer (class in i3pystatus.timer), 73
 TimeWrapper (class in i3pystatus.core.util), 121
 TimeWrapper.TimeTemplate (class in i3pystatus.core.util), 121
 truncate_error() (i3pystatus.core.threading.ExceptionWrapper method), 120

U

Uname (class in i3pystatus.uname), 75
 update_cb() (i3pystatus.pulseaudio.PulseAudio method), 58
 Updates (class in i3pystatus.updates), 76
 Uptime (class in i3pystatus.uptime), 77
 user_open() (in module i3pystatus.core.util), 123

V

Vk (class in i3pystatus.vk), 78

W

wait_for_start_barrier() (i3pystatus.core.threading.Thread method), 120
 Weather (class in i3pystatus.weather), 79
 weather_data() (i3pystatus.weather.weathercom.Weathercom method), 101
 weather_data() (i3pystatus.weather.wunderground.Wunderground method), 102
 Weathercom (class in i3pystatus.weather.weathercom), 100
 WeekCal (class in i3pystatus.weekcal), 81
 WindowTitle (class in i3pystatus.window_title), 82
 WOL (class in i3pystatus.whosonlocation), 81
 WorkloadWrapper (class in i3pystatus.core.threading), 120
 wrap() (i3pystatus.core.threading.Manager method), 120
 Wrapper (class in i3pystatus.core.threading), 120
 write_line() (i3pystatus.core.io.IOHandler method), 116
 Wunderground (class in i3pystatus.weather.wunderground), 101

X

Xkblayout (class in i3pystatus.xkblayout), 83

Y

Yaourt (class in i3pystatus.updates.yaourt), 100

Z

Zabbix (class in `i3pystatus.zabbix`), [84](#)