# Contents

1 Getting started ................................. 1
   1.1 Installing Qtile ........................................ 1
   1.2 Configuration ........................................... 4

2 Commands and scripting ....................... 21
   2.1 Commands API .......................................... 21
   2.2 Scripting ............................................... 24
   2.3 qshell .................................................. 24
   2.4 iqshell ................................................ 26
   2.5 qtile-top .............................................. 27
   2.6 qtile-run ............................................ 27
   2.7 qtile-cmd ............................................ 27
   2.8 dqtile-cmd .......................................... 30

3 Getting involved ................................ 33
   3.1 Contributing .......................................... 33
   3.2 Hacking on Qtile ..................................... 35

4 Miscellaneous .................................... 39
   4.1 Reference ........................................... 39
   4.2 Frequently Asked Questions ....................... 98
   4.3 License .............................................. 99
1.1 Installing Qtile

1.1.1 Distro Guides

Below are the preferred installation methods for specific distros. If you are running something else, please see Installing From Source.

Installing on Arch Linux

Stable versions of Qtile are currently packaged for Arch Linux. To install this package, run:

```
pacman -S qtile
```

Please see the ArchWiki for more information on Qtile.

Installing on Fedora

Stable versions of Qtile are currently packaged for current versions of Fedora. To install this package, run:

```
dnf -y install qtile
```

Installing on Funtoo

Latest versions of Qtile are available on Funtoo with Python 2.7, 3.4, and 3.5 implementations. To install it, run:

```
emerge -av x11-wm/qtile
```

You can also install the development version from GitHub:
echo "x11-wm/qtile-9999 **" >> /etc/portage/package.accept_keywords
emerge -av qtile

Customize

You can customize your installation with the following useflags:

- dbus
- widget-khal-calendar
- widget-imap
- widget-keyboardkbdd
- widget-launchbar
- widget-mpd
- widget-mpris
- widget-wlan

The dbus useflag is enabled by default. Disable it only if you know what it is and know you don’t use/need it.

All widget-* useflags are disabled by default because these widgets require additional dependencies while not everyone will use them. Enable only widgets you need to avoid extra dependencies thanks to these useflags.

Visit Funtoo Qtile documentation for more details on Qtile installation on Funtoo.

Installing on Debian or Ubuntu

On recent Ubuntu (17.04 Zesty+) and Debian unstable (Sid) versions, there are Qtile packages available via:

```
sudo apt-get install qtile
```

On older versions of Ubuntu (Wily+) and Debian testing (Stretch), the dependencies are available via:

```
sudo apt-get install python3-xcffib python3-cairocffi
```

Installing on Slackware

Qtile is available on the SlackBuilds.org as:

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qtile</td>
<td>stable branch (release)</td>
</tr>
</tbody>
</table>

Using slpkg (third party package manager)

The easy way to install Qtile is with slpkg. For example:

```
slpkg -s sbo qtile
```
Manual installation

Download dependencies first and install them. The order in which you need to install is:

- pycparser
- cffi
- six
- futures
- python-xcffib
- trollius
- cairocffi
- qtile

Please see the HOWTO for more information on SlackBuild Usage HOWTO.

1.1.2 Installing From Source

First, you need to install all of Qtile’s dependencies (although some are optional/not needed depending on your Python version, as noted below).

Note that Python 3 versions 3.4 and newer are currently supported and tested.

xcffib

Qtile uses xcffib as an XCB binding, which has its own instructions for building from source. However, if you’d like to skip building it, you can install its dependencies, you will need libxcb and libffi with the associated headers (libxcb-render0-dev and libffi-dev on Ubuntu), and install it via PyPI:

```
pip install xcffib
```

cairocffi

Qtile uses cairocffi with XCB support via xcffib. You’ll need libcairo2, the underlying library used by the binding. You should be sure before you install cairocffi that xcffib has been installed, otherwise the needed cairo-xcb bindings will not be built. Once you’ve got the dependencies installed, you can use the latest version on PyPI:

```
pip install cairocffi
```

pangocairo

You’ll also need libpangocairo, which on Ubuntu can be installed via sudo apt-get install libpangocairo-1.0-0. Qtile uses this to provide text rendering (and binds directly to it via cffi with a small in-tree binding).
asncio/trollius

Qtile uses the asyncio module as introduced in PEP 3156 for its event loop. Based on your Python version, there are different ways to install this:

- Python >=3.4: The asyncio module comes as part of the standard library, so there is nothing more to install.
- Python 3.3: This has all the infrastructure needed to implement PEP 3156, but the asyncio module must be installed from the Tulip project. This is done by calling:

  ```sh
  pip install asyncio
  ```

  Alternatively, you can install trollius (see next point). Note, however, that trollius is deprecated, and it is recommended that you use tulip, as trollius will likely be dropped if (and when) Python 2 support is dropped.
- Python 2 (and 3.3 without asyncio): You will need to install trollius, which backports the asyncio module functionality to work without the infrastructure introduced in PEP 3156. You can install this from PyPI:

  ```sh
  pip install trollius
  ```

dbus/gobject

Until someone comes along and writes an asyncio-based dbus library, qtile will depend on python-dbus to interact with dbus. This means that if you want to use things like notification daemon or mpris widgets, you’ll need to install python-gobject and python-dbus. Qtile will run fine without these, although it will emit a warning that some things won’t work.

Qtile

With the dependencies in place, you can now install qtile:

```sh
git clone git://github.com/qtile/qtile.git
cd qtile
sudo python setup.py install
```

Stable versions of Qtile can be installed from PyPI:

```sh
pip install qtile
```

As long as the necessary libraries are in place, this can be done at any point, however, it is recommended that you first install xcfilib to ensure the cairo-xcb bindings are built (see above).

1.2 Configuration

Qtile is configured in Python. A script (~/.config/qtile/config.py by default) is evaluated, and a small set of configuration variables are pulled from its global namespace.

1.2.1 Configuration lookup order

Qtile looks in the following places for a configuration file, in order:

- The location specified by the --config argument.
• $XDG_CONFIG_HOME/qtile/config.py, if it is set
• ~/.config/qtile/config.py
• It reads the module libqtile.resources.default_config, included by default with every Qtile installation.

1.2.2 Default Configuration

The default configuration is invoked when qtile cannot find a configuration file. In addition, if qtile is restarted via qshell, qtile will load the default configuration if the config file it finds has some kind of error in it. The documentation below describes the configuration lookup process, as well as what the key bindings are in the default config.

The default config is not intended to be suitable for all users; it’s mostly just there so qtile does /something/ when fired up, and so that it doesn’t crash and cause you to lose all your work if you reload a bad config.

Key Bindings

The mod key for the default config is mod4, which is typically bound to the “Super” keys, which are things like the windows key and the mac command key. The basic operation is:

• mod + k or mod + j: switch windows on the current stack
• mod + <space>: put focus on the other pane of the stack (when in stack layout)
• mod + <tab>: switch layouts
• mod + w: close window
• mod + <ctrl> + r: restart qtile with new config
• mod + <group name>: switch to that group
• mod + <shift> + <group name>: send a window to that group
• mod + <enter>: start xterm
• mod + r: start a little prompt in the bar so users can run arbitrary commands

The default config defines one screen and 8 groups, one for each letter in asdfuiop. It has a basic bottom bar that includes a group box, the current window name, a little text reminder that you’re using the default config, a system tray, and a clock.

The default configuration has several more advanced key combinations, but the above should be enough for basic usage of qtile.

Mouse Bindings

By default, holding your mod key and clicking (and holding) a window will allow you to drag it around as a floating window.

1.2.3 Configuration variables

A Qtile configuration consists of a file with a bunch of variables in it, which qtile imports and then runs as a python file to derive its final configuration. The documentation below describes the most common configuration variables; more advanced configuration can be found in the qtile-examples repository, which includes a number of real-world configurations that demonstrate how you can tune Qtile to your liking. (Feel free to issue a pull request to add your own configuration to the mix!)
Lazy objects

The `command.lazy` object is a special helper object to specify a command for later execution. This object acts like the root of the object graph, which means that we can specify a key binding command with the same syntax used to call the command through a script or through `qshell`.

Example

```python
from libqtile.config import Key
from libqtile.command import lazy

keys = [
    Key(
        ["mod1"], "k",
        lazy.layout.down()
    ),
    Key(
        ["mod1"], "j",
        lazy.layout.up()
    )
]
```

Lazy functions

This is overview of the commonly used functions for the key bindings. These functions can be called from commands on the `Qtile` object or on another object in the command tree.

Some examples are given below.

General functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lazy.spawn(&quot;app&quot;)</td>
<td>Run the application</td>
</tr>
<tr>
<td>lazy.spawncmd()</td>
<td>Open command prompt on the bar. See prompt widget.</td>
</tr>
<tr>
<td>lazy.restart()</td>
<td>Restart Qtile and reload its config. It won’t close your windows</td>
</tr>
<tr>
<td>lazy.shutdown()</td>
<td>Close the whole Qtile</td>
</tr>
</tbody>
</table>
Group functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lazy.next_layout()</td>
<td>Use next layout on the actual group</td>
</tr>
<tr>
<td>lazy.prev_layout()</td>
<td>Use previous layout on the actual group</td>
</tr>
<tr>
<td>lazy.screen.next_group()</td>
<td>Move to the group on the right</td>
</tr>
<tr>
<td>lazy.screen.prev_group()</td>
<td>Move to the group on the left</td>
</tr>
<tr>
<td>lazy.screen.toggle_group()</td>
<td>Move to the last visited group</td>
</tr>
<tr>
<td>lazy.group[&quot;group_name&quot;].toscreen()</td>
<td>Move to the group called group_name</td>
</tr>
<tr>
<td>lazy.layout.increase_ratio()</td>
<td>Increase the space for master window at the expense of slave windows</td>
</tr>
<tr>
<td>lazy.layout.decrease_ratio()</td>
<td>Decrease the space for master window in the advantage of slave windows</td>
</tr>
</tbody>
</table>

Window functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lazy.window.kill()</td>
<td>Close the focused window</td>
</tr>
<tr>
<td>lazy.layout.next()</td>
<td>Switch window focus to other pane(s) of stack</td>
</tr>
<tr>
<td>lazy.window.togroup(&quot;group_name&quot;)</td>
<td>Move focused window to the group called group_name</td>
</tr>
<tr>
<td>lazy.window.toggle_floating()</td>
<td>Put the focused window to/from floating mode</td>
</tr>
<tr>
<td>lazy.window.toggle_fullscreen()</td>
<td>Put the focused window to/from fullscreen mode</td>
</tr>
</tbody>
</table>

ScratchPad DropDown functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lazy.group[&quot;group_name&quot;].dropdown_toggle(&quot;name&quot;)</td>
<td>Toggles the visibility of the specified DropDown window. On first use, the configured process is spawned.</td>
</tr>
</tbody>
</table>

Groups

A group is a container for a bunch of windows, analogous to workspaces in other window managers. Each client window managed by the window manager belongs to exactly one group. The groups config file variable should be initialized to a list of DGroup objects.
DGroup objects provide several options for group configuration. Groups can be configured to show and hide themselves when they’re not empty, spawn applications for them when they start, automatically acquire certain groups, and various other options.

Example

```python
from libqtile.config import Group, Match
groups = [
    Group("a"),
    Group("b"),
    Group("c", matches=[Match(wm_class=['Firefox'])]),
]

# allow mod3+1 through mod3+0 to bind to groups; if you bind your groups
# by hand in your config, you don't need to do this.
from libqtile.dgroups import simple_key_binder
dgroups_key_binder = simple_key_binder("mod3")
```

Reference

Group

```python
class libqtile.config.Group(name, matches=None, exclusive=False, spawn=None, layout=None, layouts=None, persist=True, init=True, layout_opts=None, screen_affinity=None, position=9223372036854775807, label=None)
```

Represents a “dynamic” group

These groups can spawn apps, only allow certain Matched windows to be on them, hide when they’re not in use, etc. Groups are identified by their name.

Parameters

- **name** [string] the name of this group
- **matches** [default None] list of Match objects whose windows will be assigned to this group
- **exclusive** [boolean] when other apps are started in this group, should we allow them here or not?
- **spawn** [string or list of strings] this will be exec() d when the group is created, you can pass either a program name or a list of programs to exec()
- **layout** [string] the name of default layout for this group (e.g. ‘max’ or ‘stack’). This is the name specified for a particular layout in config.py or if not defined it defaults in general the class name in all lower case.
- **layouts** [list] the group layouts list overriding global layouts. Use this to define a separate list of layouts for this particular group.
- **persist** [boolean] should this group stay alive with no member windows?
- **init** [boolean] is this group alive when qtile starts?
- **position** [int] group position
- **label** [string] the display name of the group. Use this to define a display name other than name of the group. If set to None, the display name is set to the name.
libqtile.dgroups.simple_key_binder($mod$, keynames=None)

Bind keys to mod+group position or to the keys specified as second argument

Group Matching

Match

class libqtile.config.Match($title=None$, $wm_class=None$, $role=None$, $wm_type=None$,
                        $wm_instance_class=None$, $net_wm_pid=None$)

Match for dynamic groups

It can match by title, class or role.

Match supports both regular expression objects (i.e. the result of `re.compile()`) or strings (match as a “include” match). If a window matches any of the things in any of the lists, it is considered a match.

Parameters

- **title**: things to match against the title (WM_NAME)
- **wm_class**: things to match against the second string in WM_CLASS atom
- **role**: things to match against the WM_ROLE atom
- **wm_type**: things to match against the WM_TYPE atom
- **wm_instance_class**: things to match against the first string in WM_CLASS atom
- **net_wm_pid**: things to match against the _NET_WM_PID atom (only int allowed in this rule)

Rule

class libqtile.config.Rule($match$, $group=None$, $float=False$, $intrusive=False$,
                          $break_on_match=True$)

How to act on a Match

A Rule contains a Match object, and a specification about what to do when that object is matched.

Parameters

- **match**: Match object associated with this Rule
- **float**: auto float this window?
- **intrusive**: override the group’s exclusive setting?
- **break_on_match**: Should we stop applying rules if this rule is matched?

ScratchPad and DropDown

*ScratchPad* is a special - by default invisible - group which acts as a container for *DropDown* configurations. A *DropDown* can be configured to spawn a defined process and bind that process’ window to it. The associated window can then be shown and hidden by the lazy command `dropdown_toggle()` (see *Lazy objects*) from the ScratchPad group. Thus - for example - your favorite terminal emulator turns into a quake-like terminal by the control of qtile.

If the DropDown window turns visible it is placed as a floating window on top of the current group. If the DropDown is hidden, it is simply switched back to the ScratchPad group.
Example

```python
from libqtile.config import Group, ScratchPad, DropDown, Key
from libqtile.command import lazy

groups = [
    ScratchPad("scratchpad", [
        # define a drop down terminal.
        # it is placed in the upper third of screen by default.
        DropDown("term", "urxvt", opacity=0.8),

        # define another terminal exclusively for qshell at different position
        DropDown("qshell", "urxvt -hold -e qshell",
            x=0.05, y=0.4, width=0.9, height=0.6, opacity=0.9,
            on_focus_lost_hide=True)
    ]),
    Group("a"),
]

keys = [
    # toggle visibility of above defined DropDown named "term"
    Key([], 'F11', lazy.group['scratchpad'].dropdown_toggle('term')),
    Key([], 'F12', lazy.group['scratchpad'].dropdown_toggle('qshell'))
]
```

There is only one DropDown visible in current group at a time. If a further DropDown is set visible the currently shown DropDown turns invisible immediately.

Note that if the window is set to not floating, it is detached from DropDown and ScratchPad, and a new proces is spawned next time the DropDown is set visible.

Reference

**ScratchPad**

```python
class libqtile.config.ScratchPad(name, dropdowns=None, position=9223372036854775807, label="")
```

Represents a “ScratchPad” group

ScratchPad adds a (by default) invisible group to qtile. That group is used as a place for currently not visible windows spawned by a DropDown configuration.

Parameters

- `name` [string] the name of this group
- `dropdowns` [default None] list of DropDown objects
- `position` [int] group position
- `label` [string] The display name of the ScratchPad group. Defaults to the empty string such that the group is hidden in GroupList widget.

**DropDown**

```python
class libqtile.config.DropDown(name, cmd, **config)
```

Configure a specified command and its associated window for the ScratchPad. That window can be shown and hidden using a configurable keystroke or any other scripted trigger.
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>0.35</td>
<td>Height of window as fraction of current screen.</td>
</tr>
<tr>
<td>on_focus_lost</td>
<td>True</td>
<td>Shall the window be hidden if focus is lost? If so, the DropDown is hidden if window focus or the group is changed.</td>
</tr>
<tr>
<td>opacity</td>
<td>0.9</td>
<td>Opacity of window as fraction. Zero is opaque.</td>
</tr>
<tr>
<td>warp_pointer</td>
<td>True</td>
<td>Shall pointer warp to center of window on activation? This has only effect if any of the on_focus_lost_xxx configurations is True</td>
</tr>
<tr>
<td>width</td>
<td>0.8</td>
<td>Width of window as fraction of current screen width</td>
</tr>
<tr>
<td>x</td>
<td>0.1</td>
<td>X position of window as fraction of current screen width. 0 is the left most position.</td>
</tr>
<tr>
<td>y</td>
<td>0.0</td>
<td>Y position of window as fraction of current screen height. 0 is the top most position. To show the window at bottom, you have to configure a value &lt; 1 and an appropriate height.</td>
</tr>
</tbody>
</table>

### Keys

The `keys` variable defines Qtile’s key bindings. Individual key bindings are defined with `libqtile.config.Key` as demonstrated in the following example. Note that you may specify more than one callback functions.

```python
from libqtile.config import Key

keys = [
    # Pressing "Meta + Shift + a".
    Key(["mod4", "shift"], "a", callback, ...),

    # Pressing "Control + p".
    Key(["control"], "p", callback, ...),

    # Pressing "Meta + Tab".
    Key(["mod4", "mod1"], "Tab", callback, ...),
]
```

The above may also be written more concisely with the help of the `libqtile.config.EzKey` helper class. The following example is functionally equivalent to the above:

```python
from libqtile.config import EzKey as Key

keys = [
    # Pressing "Meta + Shift + a".
    Key("M-S-a", callback, ...),
    Key("C-p", callback, ...),
    Key("M-A-%Tab>", callback, ...),
]
```

The `EzKey` modifier keys (i.e. MASC) can be overwritten through the `EzKey.modifier_keys` dictionary. The defaults are:

```python
modifier_keys = {
    'M': 'mod4',
    'A': 'mod1',
    'S': 'shift',
    'C': 'control',
}
```
Modifiers

On most systems mod1 is the Alt key - you can see which modifiers, which are enclosed in a list, map to which keys on your system by running the xmodmap command. This example binds Alt-k to the “down” command on the current layout. This command is standard on all the included layouts, and switches to the next window (where “next” is defined differently in different layouts). The matching “up” command switches to the previous window.

Modifiers include: “shift”, “lock”, “control”, “mod1”, “mod2”, “mod3”, “mod4”, and “mod5”. They can be used in combination by appending more than one modifier to the list:

```
Key(
    ["mod1", "control"], "k",
    lazy.layout.shuffle_down()
)
```

Special keys

These are most commonly used special keys. For complete list please see the code. You can create bindings on them just like for the regular keys. For example `Key(["mod1"], "F4", lazy.window.kill())`.

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
</tr>
<tr>
<td>BackSpace</td>
</tr>
<tr>
<td>Tab</td>
</tr>
<tr>
<td>space</td>
</tr>
<tr>
<td>Home, End</td>
</tr>
<tr>
<td>Left, Up, Right, Down</td>
</tr>
<tr>
<td>F1, F2, F3, ...</td>
</tr>
<tr>
<td>XF86AudioRaiseVolume</td>
</tr>
<tr>
<td>XF86AudioLowerVolume</td>
</tr>
<tr>
<td>XF86AudioMute</td>
</tr>
<tr>
<td>XF86AudioNext</td>
</tr>
<tr>
<td>XF86AudioPrev</td>
</tr>
<tr>
<td>XF86MonBrightnessUp</td>
</tr>
<tr>
<td>XF86MonBrightnessDown</td>
</tr>
</tbody>
</table>

Reference

Key

```python
class libqtile.config.Key(modifiers, key, *commands, **kwds)
```

Defines a keybinding.

Parameters

- **modifiers**: A list of modifier specifications. Modifier specifications are one of: “shift”, “lock”, “control”, “mod1”, “mod2”, “mod3”, “mod4”, “mod5”.
- **key**: A key specification, e.g. “a”, “Tab”, “Return”, “space”.
- **commands**: A list of lazy command objects generated with the command.lazy helper. If multiple Call objects are specified, they are run in sequence.
kwds: A dictionary containing “desc”, allowing a description to be added

**EzConfig**

```python
class libqtile.config.EzConfig
    Helper class for defining key and button bindings in an emacs-like format. Inspired by Xmonad’s XMonad.Util.EZConfig.
```

**Layouts**

A layout is an algorithm for laying out windows in a group on your screen. Since Qtile is a tiling window manager, this usually means that we try to use space as efficiently as possible, and give the user ample commands that can be bound to keys to interact with layouts.

The `layouts` variable defines the list of layouts you will use with Qtile. The first layout in the list is the default. If you define more than one layout, you will probably also want to define key bindings to let you switch to the next and previous layouts.

See *Built-in Layouts* for a listing of available layouts.

**Example**

```python
from libqtile import layout
layouts = [
    layout.Max(),
    layout.Stack(stacks=2)
]
```

**Mouse**

The `mouse` config file variable defines a set of global mouse actions, and is a list of `Click` and `Drag` objects, which define what to do when a window is clicked or dragged.

**Example**

```python
from libqtile.config import Click, Drag
mouse = [
    Drag([mod], "Button1", lazy.window.set_position_floating(),
         start=lazy.window.get_position()),
    Drag([mod], "Button3", lazy.window.set_size_floating(),
         start=lazy.window.get_size()),
    Click([mod], "Button2", lazy.window.bring_to_front())
]
```

The above example can also be written more concisely with the help of the EzClick and EzDrag helpers:

```python
from libqtile.config import EzClick as Click, EzDrag as Drag
mouse = [
    Drag("M-1", lazy.window.set_position_floating(),
         start=lazy.window.get_position()),
]
```
Reference

Click

class libqtile.config.Click(modifiers, button, *commands, **kwargs)
   Defines binding of a mouse click
   It focuses clicked window by default. If you want to prevent it, pass focus=None as an argument

Drag

class libqtile.config.Drag(modifiers, button, *commands, **kwargs)
   Defines binding of a mouse to some dragging action
   On each motion event command is executed with two extra parameters added x and y offset from previous move
   It focuses clicked window by default. If you want to prevent it pass, focus=None as an argument

Screens

The screens configuration variable is where the physical screens, their associated bars, and the widgets contained within the bars are defined.

See Built-in Widgets for a listing of available widgets.

Example

Tying together screens, bars and widgets, we get something like this:

```python
from libqtile.config import Screen
from libqtile import bar, widget

screens = [
    Screen(
        bottom=bar.Bar([
            widget.GroupBox(),
            widget.WindowName()
        ], 30),
    ),
    Screen(
        bottom=bar.Bar([
            widget.GroupBox(),
            widget.WindowName()
        ], 30),
    )
]
```
Bars support both solid background colors and gradients by supplying a list of colors that make up a linear gradient. For example, `bar.Bar(..., background='#000000')` will give you a black background (the default), while `bar.Bar(..., background=['#000000', '#FFFFFF'])` will give you a background that fades from black to white.

**Third-party bars**

There might be some reasons to use third-party bars. For instance you can come from another window manager and you have already configured dzen2, xmobar, or something else. They definitely can be used with Qtile too. In fact, any additional configurations aren’t needed. Just run the bar and qtile will adapt.

**Reference**

**Screen**

```python
class libqtile.config.Screen(top=None, bottom=None, left=None, right=None, x=None, y=None, width=None, height=None)
```

A physical screen, and its associated paraphernalia.

Define a screen with a given set of Bars of a specific geometry. Note that bar.Bar objects can only be placed at the top or the bottom of the screen (bar.Gap objects can be placed anywhere). Also, `x`, `y`, `width`, and `height` aren’t specified usually unless you are using ‘fake screens’.

**Parameters**

- `top`: List of Gap/Bar objects, or None.
- `bottom`: List of Gap/Bar objects, or None.
- `left`: List of Gap/Bar objects, or None.
- `right`: List of Gap/Bar objects, or None.
- `x`: [int or None]
- `y`: [int or None]
- `width`: [int or None]
- `height`: [int or None]

**Bar**

```python
class libqtile.bar.Bar(widgets, size, **config)
```

A bar, which can contain widgets

**Parameters**

- `widgets`: A list of widget objects.
- `size`: The “thickness” of the bar, i.e. the height of a horizontal bar, or the width of a vertical bar.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>'#000000'</td>
<td>Background colour.</td>
</tr>
<tr>
<td>opacity</td>
<td>1</td>
<td>Bar window opacity.</td>
</tr>
</tbody>
</table>
Gap

class libqtile.bar.Gap(size)
A gap placed along one of the edges of the screen

If a gap has been defined, Qtile will avoid covering it with windows. The most probable reason for configuring
a gap is to make space for a third-party bar or other static window.

Parameters

size: The “thickness” of the gap, i.e. the height of a horizontal gap, or the width of a vertical
gap.

Hooks

Qtile provides a mechanism for subscribing to certain events in libqtile.hook. To subscribe to a hook in your
configuration, simply decorate a function with the hook you wish to subscribe to.

See Built-in Hooks for a listing of available hooks.

Examples

Automatic floating dialogs

Let’s say we wanted to automatically float all dialog windows (this code is not actually necessary; Qtile floats all
dialogs by default). We would subscribe to the client_new hook to tell us when a new window has opened and, if
the type is “dialog”, as can set the window to float. In our configuration file it would look something like this:

```python
from libqtile import hook

@hook.subscribe.client_new
def floating_dialogs(window):
    dialog = window.window.get_wm_type() == 'dialog'
    transient = window.window.get_wm_transient_for()
    if dialog or transient:
        window.floating = True
```

A list of available hooks can be found in the Built-in Hooks reference.

Autostart

If you want to run commands or spawn some applications when Qtile starts, you’ll want to look at the startup and
startup_once hooks. startup is emitted every time Qtile starts (including restarts), whereas startup_once
is only emitted on the very first startup.

Let’s create a file ~/.config/qtile/autostart.sh that will set our desktop wallpaper and start a few pro-
grams when Qtile first runs.

```bash
#!/bin/sh
feh --bg-scale ~/images/wallpaper.jpg &
pidgin &
dropbox start &
```

We can then subscribe to startup_once to run this script:
```python
import os
import subprocess

@hook.subscribe.startup_once
def autostart():
    home = os.path.expanduser('~/config/qtile/autostart.sh')
    subprocess.call([home])
```

In addition to the above variables, there are several other boolean configuration variables that control specific aspects of Qtile’s behavior:

<table>
<thead>
<tr>
<th>variable</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_fullscreen</td>
<td>True</td>
<td>If a window requests to be fullscreen, it is automatically fullscreened. Set this to false if you only want windows to be fullscreen if you ask them to be.</td>
</tr>
<tr>
<td>bring_front_click</td>
<td>False</td>
<td>When clicked, should the window be brought to the front or not. (This sets the X Stack Mode to Above.)</td>
</tr>
<tr>
<td>cursor_warp</td>
<td>False</td>
<td>If true, the cursor follows the focus as directed by the keyboard, warping to the center of the focused window.</td>
</tr>
<tr>
<td>dgroups_key_binder</td>
<td>None</td>
<td>TODO</td>
</tr>
<tr>
<td>dgroups_app_rules</td>
<td>[]</td>
<td>TODO</td>
</tr>
<tr>
<td>extension_defaults</td>
<td>None</td>
<td>TODO</td>
</tr>
<tr>
<td>floating_layout</td>
<td>layout.Floating(float_rules=[...])</td>
<td>See the configuration file for the default float_rules.</td>
</tr>
</tbody>
</table>
| focus_on_window_activation | smart | Behavior of the _NET_ACTIVATE_WINDOW message sent by applications
  • urgent: urgent flag is set for the window
  • focus: automatically focus the window
  • smart: automatically focus if the window is in the current group |
| follow_focus          | None    | Controls whether or not focus follows the mouse around as it moves across windows in a layout. |
| main                 | None    | TODO                                                                       |
| widget_defaults      | dict(font='sans', fontsize=12, padding=3) | Default settings for bar widgets.                                        |
| wmname               | "LG3D"  | TODO                                                                       |

### 1.2.4 Testing your configuration

The best way to test changes to your configuration is with the provided Xephyr script. This will run Qtile with your config.py inside a nested X server and prevent your running instance of Qtile from crashing if something goes wrong.

See *Hacking Qtile* for more information on using Xephyr.

### 1.2.5 Starting Qtile

There are several ways to start Qtile. The most common way is via an entry in your X session manager’s menu. The default Qtile behavior can be invoked by creating a qtile.desktop file in /usr/share/xsessions.
A second way to start Qtile is a custom X session. This way allows you to invoke Qtile with custom arguments, and also allows you to do any setup you want (e.g. special keyboard bindings like mapping caps lock to control, setting your desktop background, etc.) before Qtile starts. If you’re using an X session manager, you still may need to create a custom.desktop file similar to the qtile.desktop file above, but with Exec=/etc/X11/xsession. Then, create your own ~/.xsession. There are several examples of user defined xsession in the qtile-examples repository.

Finally, if you’re a gnome user, you can start integrate Qtile into Gnome’s session manager and use gnome as usual:

**Running Inside Gnome**

Add the following snippet to your Qtile configuration. As per this page, it registers Qtile with gnome-session. Without it, a “Something has gone wrong!” message shows up a short while after logging in. dbus-send must be on your $PATH.

```python
import subprocess
import os
from libqtile import hook

@hook.subscribe.startup
def dbus_register():
    id = os.environ.get('DESKTOP_AUTOSTART_ID')
    if not id:
        return
    subprocess.Popen(['dbus-send',
                       '--session',
                       '--print-reply',
                       '--dest=org.gnome.SessionManager',
                       '/org/gnome/SessionManager',
                       'org.gnome.SessionManager.RegisterClient',
                       'string:qtile',
                       'string: ' + id])
```

This adds a new entry “Qtile GNOME” to GDM’s login screen.

```
$ cat /usr/share/xsessions/qtile_gnome.desktop
[Desktop Entry]
Name=Qtile GNOME
Comment=Tiling window manager
TryExec=/usr/bin/gnome-session
Exec=gnome-session --session=qtile
Type=XSession
```

The custom session for gnome-session.

```
$ cat /usr/share/gnome-session/sessions/qtile.session
[GNOME Session]
Name=Qtile session
RequiredComponents=qtile;gnome-settings-daemon;
```

So that Qtile starts automatically on login.

```
$ cat /usr/share/applications/qtile.desktop
[Desktop Entry]
Type=Application
Encoding=UTF-8
```

(continues on next page)
The above does not start gnome-panel. Getting gnome-panel to work requires some extra Qtile configuration, mainly making the top and bottom panels static on panel startup and leaving a gap at the top (and bottom) for the panel window.

You might want to add keybindings to log out of the GNOME session.

```python
Key([mod, 'control'], 'l', lazy.spawn('gnome-screensaver-command -l')),#
Key([mod, 'control'], 'q', lazy.spawn('gnome-session-quit --logout --no-prompt')),#
Key([mod, 'shift', 'control'], 'q', lazy.spawn('gnome-session-quit --power-off')),#
```

The above apps need to be in your path (though they are typically installed in `/usr/bin`, so they probably are if they’re installed at all).
2.1 Commands API

Qtile’s command API is based on a graph of objects, where each object has a set of associated commands. The graph and object commands are used in a number of different places:

- Commands can be bound to keys in the Qtile configuration file.
- Commands can be called through qshell, the Qtile shell.
- The qsh can also be hooked into a Jupyter kernel called iqshell.
- Commands can be called from a script to interact with Qtile from Python.

If the explanation below seems a bit complex, please take a moment to explore the API using the qshell command shell. Command lists and detailed documentation can be accessed from its built-in help command.

2.1.1 Object Graph

The objects in Qtile’s object graph come in seven flavours, matching the seven basic components of the window manager: layouts, windows, groups, bars, widgets, screens, and a special root node. Objects are addressed by a path specification that starts at the root, and follows the edges of the graph. This is what the graph looks like:
Each arrow can be read as “holds a reference to”. So, we can see that a widget object holds a reference to objects of type bar, screen and group. Let’s start with some simple examples of how the addressing works. Which particular objects we hold reference to depends on the context - for instance, widgets hold a reference to the screen that they appear on, and the bar they are attached to.

Let’s look at an example, starting at the root node. The following script runs the status command on the root node, which, in this case, is represented by the Client object:

```python
from libqtile.command import Client
c = Client()
print c.status()
```

From the graph, we can see that the root node holds a reference to group nodes. We can access the “info” command on the current group like so:

```python
c.group.info()
```

To access a specific group, regardless of whether or not it is current, we use the Python containment syntax. This
command sends group “b” to screen 1 (by the `libqtile.config.Group.to_screen()` method):

```python
c.group["b"].to_screen(1)
```

The current `group`, `layout`, `screen` and `window` can be accessed by simply leaving the key specifier out. The key specifier is mandatory for `widget` and `bar` nodes.

We can now drill down deeper in the graph. To access the screen currently displaying group “b”, we can do this:

```python
c.group["b"].screen.info()
```

Be aware, however, that group “b” might not currently be displayed. In that case, it has no associated screen, the path resolves to a non-existent node, and we get an exception:

```
libqtile.command.CommandError: No object screen in path 'group['b'].screen'
```

The graph is not a tree, since it can contain cycles. This path (redundantly) specifies the group belonging to the screen that belongs to group “b”:

```python
c.group["b"].screen.group
```

### 2.1.2 Keys

The key specifier for the various object types are as follows:

<table>
<thead>
<tr>
<th>Object</th>
<th>Key</th>
<th>Optional?</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar</td>
<td>“top”, “bottom”</td>
<td>No</td>
<td><code>c.screen.bar[&quot;bottom&quot;]</code></td>
</tr>
<tr>
<td>group</td>
<td>Name string</td>
<td>Yes</td>
<td><code>c.group[&quot;one&quot;]</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>c.group</code></td>
</tr>
<tr>
<td>layout</td>
<td>Integer index</td>
<td>Yes</td>
<td><code>c.layout[2]</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>c.layout</code></td>
</tr>
<tr>
<td>screen</td>
<td>Integer index</td>
<td>Yes</td>
<td><code>c.screen[1]</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>c.screen</code></td>
</tr>
<tr>
<td>widget</td>
<td>Widget name</td>
<td>No</td>
<td><code>c.widget[&quot;textbox&quot;]</code></td>
</tr>
<tr>
<td>window</td>
<td>Integer window ID</td>
<td>Yes</td>
<td><code>c.window[123456]</code></td>
</tr>
</tbody>
</table>
2.2 Scripting

2.2.1 Client-Server Scripting Model

Qtile has a client-server control model - the main Qtile instance listens on a named pipe, over which marshalled command calls and response data is passed. This allows Qtile to be controlled fully from external scripts. Remote interaction occurs through an instance of the `libqtile.command.Client` class. This class establishes a connection to the currently running instance of Qtile, and sources the user’s configuration file to figure out which commands should be exposed. Commands then appear as methods with the appropriate signature on the `Client` object. The object hierarchy is described in the `Commands API` section of this manual. Full command documentation is available through the *Qtile Shell*.

2.2.2 Example

Below is a very minimal example script that inspects the current qtile instance, and returns the integer offset of the current screen.

```python
from libqtile.command import Client
import os

client = Client()
print client.screen.info()['index']
```

2.2.3 Reference

**Client**

```python
class libqtile.command.Client (fname=None, is_json=False)

    Exposes a command tree used to communicate with a running instance of Qtile
```

2.3 qshell

The Qtile command shell is a command-line shell interface that provides access to the full complement of Qtile command functions. The shell features command name completion, and full command documentation can be accessed from the shell itself. The shell uses GNU Readline when it’s available, so the interface can be configured to, for example, obey VI keybindings with an appropriate `.inputrc` file. See the GNU Readline documentation for more information.

2.3.1 Navigating the Object Graph

The shell presents a filesystem-like interface to the object graph - the built-in “cd” and “ls” commands act like their familiar shell counterparts:

```bash
> ls
layout/ widget/ screen/ bar/ window/ group/

> cd bar

bar> ls
bottom/
```

(continues on next page)
bar> cd bottom

bar['bottom']> ls
screen/

bar['bottom']> cd ../..

> ls
layout/ widget/ screen/ bar/ window/ group/

Note that the shell provides a “short-hand” for specifying node keys (as opposed to children). The following is a valid shell path:

> cd group/4/window/31457314

The command prompt will, however, always display the Python node path that should be used in scripts and key bindings:

group['4'].window[31457314]>

### 2.3.2 Live Documentation

The shell help command provides the canonical documentation for the Qtile API:

```shell
cd layout/1
layout[1]> help
help command  -- Help for a specific command.

Builtins
========
cd  exit  help  ls  q  quit

Commands for this object
========================
add  commands  current  delete  doc
down  get info  items  next  previous
rotate  shuffle_down  shuffle_up  toggle_split  up

layout[1]> help previous
previous()
Focus previous stack.
```

### 2.3.3 Reference

**Qsh**

```python
class libqtile.sh.QSh(client, completekey='tab')

Qtile shell instance

do_cd(arg)
    Change to another path.
```

### 2.3. qshell
Examples

```bash
cd layout/0

cd ../layout

do_exit (args)

Exit qshell

do_ls (arg)

List contained items on a node.
```

Examples

```bash
> ls > ls ../layout

do_pwd (arg)

Returns the current working location

This is the same information as presented in the qshell prompt, but is very useful when running iqshell.
```

Examples

```bash
> pwd / > cd bar/top bar['top'] > pwd bar['top']

do_help (arg)

Give help on commands and builtins

When invoked without arguments, provides an overview of all commands. When passed as an argument, also provides a detailed help on a specific command or builtin.
```

Examples

```bash
> help
> help command
```

2.4 iqshell

In addition to the standard qshell shell interface, we provide a kernel capable of running through Jupyter that hooks into the qshell client. The command structure and syntax is the same as qshell, so it is recommended you read that for more information about that.

2.4.1 Dependencies

In order to run iqshell, you must have ipykernel and jupyter_console. You can install the dependencies when you are installing qtile by running:

```bash
$ pip install qtile[ipython]
```

Otherwise, you can just install these two packages separately, either through PyPI or through your distribution package manager.
2.4.2 Installing and Running the Kernel

Once you have the required dependencies, you can run the kernel right away by running:

```bash
$ python -m libqtile.interactive.iqshell_kernel
```

However, this will merely spawn a kernel instance, you will have to run a separate frontend that connects to this kernel. A more convenient way to run the kernel is by registering the kernel with Jupyter. To register the kernel itself, run:

```bash
$ python -m libqtile.interactive.iqshell_install
```

If you run this as a non-root user, or pass the `--user` flag, this will install to the user Jupyter kernel directory. You can now invoke the kernel directly when starting a Jupyter frontend, for example:

```bash
$ jupyter console --kernel qshell
```

The `iqshell` script will launch a Jupyter terminal console with the `qshell` kernel.

2.4.3 iqshell vs qshell

One of the main drawbacks of running through a Jupyter kernel is the frontend has no way to query the current node of the kernel, and as such, there is no way to set a custom prompt. In order to query your current node, you can call `pwd`.

This, however, enables many of the benefits of running in a Jupyter frontend, including being able to save, run, and re-run code cells in frontends such as the Jupyter notebook.

The Jupyter kernel also enables more advanced help, text completion, and introspection capabilities (however, these are currently not implemented at a level much beyond what is available in the standard `qshell`).

2.5 qtile-top

Is a top like to measure memory usage of `qtile`’s internals.

2.6 qtile-run

Run a command applying rules to the new windows, i.e., you can start a window in a specific group, make it floating, intrusive, etc.

The Windows must have `NET_WM_PID`.

```bash
# run xterm floating on group "test-group"
qtile-run -g test-group -f xterm
```

2.7 qtile-cmd

This is a simple tool to expose `qtile.command` functionality to shell. This can be used standalone or in other shell scripts.
2.7.1 Examples:

Output of `qtile-cmd -h`

```bash
usage: qtile-cmd [-h] [--object OBJ_SPEC [OBJ_SPEC ...]]
    [--function FUNCTION] [--args ARGS [ARGS ...]] [--info]

Simple tool to expose `qtile.command` functionality to shell.

optional arguments:
- `h`, `--help` show this help message and exit
- `--object OBJ_SPEC [OBJ_SPEC ...], -o OBJ_SPEC [OBJ_SPEC ...]`
  Specify path to object (space separated). If no
  --function flag display available commands.
- `--function FUNCTION, -f FUNCTION`
  Select function to execute.
- `--args ARGS [ARGS ...], -a ARGS [ARGS ...]`
  Set arguments supplied to function.
- `--info, -i` With both --object and --function args prints
  documentation for function.

Examples:
qtile-cmd
qtile-cmd -o cmd
qtile-cmd -o cmd -f prev_layout -i
qtile-cmd -o cmd -f prev_layout -a 3 # prev_layout on group 3
qtile-cmd -o group 3 -f focus_back
```

Output of `qtile-cmd -o group 3`

```
- `o group 3 -f commands` Returns a list of possible commands for this object
- `o group 3 -f doc` Returns the documentation for a specified command.
- `--name`
- `o group 3 -f eval` Evaluates code in the same context as this function
- `o group 3 -f focus_back` Focus the window that had focus before the current
  one got it.
- `--name`
- `o group 3 -f focus_by_name` Focus the first window with the given name. Do
  nothing if the name is
- `--name`
- `o group 3 -f function` Call a function with current object as argument
- `o group 3 -f info` Returns a dictionary of info for this group
- `o group 3 -f info_by_name` Get the info for the first window with the given
  name without giving it
- `--name`
- `o group 3 -f items` Returns a list of contained items for the specified
  name
- `--name`
- `o group 3 -f next_window` Focus the next window in group.
- `o group 3 -f prev_window` Focus the previous window in group.
- `o group 3 -f set_label` Set the display name of current group to be used in
  GroupBox widget.
- `--GroupBox
- o group 3 -f setlayout` Switch position of current group with name
- `o group 3 -f toscreen` Pull a group to a specified screen.
- `o group 3 -f unminimize_all` Unminimise all windows in this group
```

Chapter 2. Commands and scripting
Output of `qtile-cmd -o cmd`

```
-o cmd -f add_rule * Add a dgroup rule, returns rule_id needed to remove it
-o cmd -f addgroup * Add a group with the given name
-o cmd -f commands Returns a list of possible commands for this object
-o cmd -f critical Set log level to CRITICAL
-o cmd -f debug Set log level to DEBUG
-o cmd -f delgroup * Delete a group with the given name
-o cmd -f display_kb * Display table of key bindings
-o cmd -f doc * Returns the documentation for a specified command name
-o cmd -f error Set log level to ERROR
-o cmd -f eval * Evaluates code in the same context as this function
-o cmd -f findwindow * Launch prompt widget to find a window of the given name
-o cmd -f focus_by_click * Bring a window to the front
-o cmd -f function * Call a function with current object as argument
-o cmd -f get_info Prints info for all groups
-o cmd -f get_state Get pickled state for restarting qtile
-o cmd -f get_test_data Returns any content arbitrarily set in the self. test_data attribute.
-o cmd -f groups Return a dictionary containing information for all groups
-o cmd -f hide_show_bar * Toggle visibility of a given bar
-o cmd -f info Set log level to INFO
-o cmd -f internal_windows Return info for each internal window (bars, for example)
-o cmd -f items * Returns a list of contained items for the specified name
-o cmd -f list_widgets List of all addressible widget names
-o cmd -f next_layout * Switch to the next layout.
-o cmd -f next_screen Move to next screen
-o cmd -f next_urgent Focus next window with urgent hint
-o cmd -f pause Drops into pdb
-o cmd -f prev_layout * Switch to the previous layout.
-o cmd -f prev_screen Move to the previous screen
-o cmd -f qtile_info Returns a dictionary of info on the Qtile instance
-o cmd -f qtilecmd * Execute a Qtile command using the client syntax
-o cmd -f remove_rule * Remove a dgroup rule by rule_id
-o cmd -f restart Restart Qtile
-o cmd -f run_extension * Run extensions
-o cmd -f run_extension * Deprecated alias for cmd_run_extension()
-o cmd -f run_external Run external Python script
-o cmd -f screens Return a list of dictionaries providing information on all screens
-o cmd -f shutdown Quit Qtile
-o cmd -f simulate_keypress * Simulates a keypress on the focused window.
-o cmd -f spawn * Run cmd in a shell.
-o cmd -f spawncmd * Spawn a command using a prompt widget, with tab-completion.
-o cmd -f status Return "OK" if Qtile is running
-o cmd -f switch_groups * Switch position of groupa to groupb
-o cmd -f switchgroup * Launch prompt widget to switch to a given group to the current screen
-o cmd -f sync Sync the X display. Should only be used for development
```

(continues on next page)
-o cmd -f to_layout_index * Switch to the layout with the given index in self.
-objlayouts.
-obj cmd -f to_screen * Warp focus to screen n, where n is a 0-based screen.
-obj number
-obj cmd -f togroup * Launch prompt widget to move current window to a
-obj given group
-obj cmd -f tracemalloc_dump Dump tracemalloc snapshot
-obj cmd -f tracemalloc_toggle Toggle tracemalloc status
-obj cmd -f warning Set log level to WARNING
-obj cmd -f windows Return info for each client window

2.8 dqtile-cmd

A Rofi/dmenu interface to qtile-cmd. Accepts all arguments of qtile-cmd.

2.8.1 Examples:

Output of dqtile-cmd --o cmd
dmenu:

Alt-l Prompt for args and show function help (if -f is present)
. Go back to menu.
C-u Clear input
Esc Exit
-o cmd -f add_rule * Add a dgroup rule, returns rule id needed to remove it
-o cmd -f addgroup * Add a group with the given name
-o cmd -f commands Returns a list of possible commands for this object
-o cmd -f critical Set log level to CRITICAL
-o cmd -f debug Set log level to DEBUG
-o cmd -f delgroup * Delete a group with the given name
-o cmd -f display_kb * Display table of key bindings
-o cmd -f doc * Returns the documentation for a specified command name
-o cmd -f error Set log level to ERROR
-o cmd -f eval * Evaluates code in the same context as this function
-o cmd -f findwindow * Launch prompt widget to find a window of the given name
-o cmd -f focus_by_click * Bring a window to the front
-o cmd -f function * Call a function with current object as argument
-o cmd -f get_info Prints info for all groups
-o cmd -f get_state Get pickled state for restarting qtile

Output of dqtile-cmd --h
dqtile-cmd

A Rofi/dmenu interface to qtile-cmd. Accepts all arguments of qtile-cmd
(see below).

usage: dqtile-cmd [-h] [--object OBJ_SPEC [OBJ_SPEC ...]]
                [--function FUNCTION] [--args ARGS [ARGS ...]] [--info]
Simple tool to expose qtile.command functionality to shell.

optional arguments:
  -h, --help  show this help message and exit
  --object OBJ_SPEC [OBJ_SPEC ...], -o OBJ_SPEC [OBJ_SPEC ...]
    Specify path to object (space separated). If no
    --function flag display available commands.
  --function FUNCTION, -f FUNCTION
    Select function to execute.
  --args ARGS [ARGS ...], -a ARGS [ARGS ...]
    Set arguments supplied to function.
  --info, -i  With both --object and --function args prints
    documentation for function.

Examples:
  dqtile-cmd
  dqtile-cmd -o cmd
  dqtile-cmd -o cmd -f prev_layout -i
  dqtile-cmd -o cmd -f prev_layout -a 3 # prev_layout on group 3
  dqtile-cmd -o group 3 -f focus_back

If both rofi and dmenu are present rofi will be selected as default, to change this
  --us --force-dmenu as the first argument.
3.1 Contributing

3.1.1 Reporting bugs

Perhaps the easiest way to contribute to Qtile is to report any bugs you run into on the github issue tracker.

Useful bug reports are ones that get bugs fixed. A useful bug report normally has two qualities:

1. **Reproducible.** If your bug is not reproducible it will never get fixed. You should clearly mention the steps to reproduce the bug. Do not assume or skip any reproducing step. Described the issue, step-by-step, so that it is easy to reproduce and fix.

2. **Specific.** Do not write a essay about the problem. Be Specific and to the point. Try to summarize the problem in minimum words yet in effective way. Do not combine multiple problems even they seem to be similar. Write different reports for each problem.

3.1.2 Writing code

To get started writing code for Qtile, check out our guide to *Hacking on Qtile*.

**Git workflow**

Our workflow is based on Vincent Driessen’s successful git branching model:

- The *master* branch is our current release
- The *develop* branch is what all pull requests should be based against
- Feature branches are where new features, both major and minor, should be developed.
git-flow is a git plugin that helps facilitate this branching strategy. It’s not required, but can help make things a bit easier to manage. There is also a good write up on using git-flow.

We also request that git commit messages follow the standard format.

**Submit a pull request**

You’ve done your hacking and are ready to submit your patch to Qtile. Great! Now it’s time to submit a pull request to our issue tracker on Github.

**Important:** Pull requests are not considered complete until they include all of the following:

- Code that conforms to PEP8.
• **Unit tests** that pass locally and in our CI environment.
• **Documentation** updates on an as needed basis.

Feel free to add your contribution (no matter how small) to the appropriate place in the CHANGELOG as well!

# 3.2 Hacking on Qtile

## 3.2.1 Requirements

Any reasonably recent version of these should work, so you can probably just install them from your package manager.

- pytest
- Xephyr
- xvfbwrapper
- `xrandr`, `xcalc`, `xeyes` and `xclock` *(x11-apps on Ubuntu)*

On Ubuntu, if testing on Python 3, this can be done with:

```
sudo apt-get install python3-pytest xserver-xephyr python3-xvfbwrapper x11-apps
```

Or, on Python 2:

```
sudo apt-get install python-pytest xserver-xephyr python-xvfbwrapper x11-apps
```

## 3.2.2 Building cffi module

Qtile ships with a small in-tree pangocairo binding built using cffi, `pangocffi.py`, and also binds to xcursor with cffi. The bindings are not built at run time and will have to be generated manually when the code is downloaded or when any changes are made to the cffi library. This can be done by calling:

```
python libqtile/ffi_build.py
```

## 3.2.3 Using Xephyr and the test suite

Qtile has a very extensive test suite, using the Xephyr nested X server. When tests are run, a nested X server with a nested instance of Qtile is fired up, and then tests interact with the Qtile instance through the client API. The fact that we can do this is a great demonstration of just how completely scriptable Qtile is. In fact, Qtile is designed expressly to be scriptable enough to allow unit testing in a nested environment.

The Qtile repo includes a tiny helper script to let you quickly pull up a nested instance of Qtile in Xephyr, using your current configuration. Run it from the top-level of the repository, like this:

```
./scripts/xephyr
```

In practice, the development cycle looks something like this:

1. make minor code change
2. run appropriate test: `pytest tests/test_module.py`
3. GOTO 1, until hackage is complete
4. run entire test suite: pytest
5. commit

3.2.4 Second X Session

Some users prefer to test Qtile in a second, completely separate X session: Just switch to a new tty and run startx normally to use the ~/.xinitrc startup script.

It’s likely though that you want to use a different, customized startup script for testing purposes, for example ~/.config/qtile/xinitrc. You can do so by launching X with:

```
startx ~/.config/qtile/xinitrc
```

startx deals with multiple X sessions automatically. If you want to use xinit instead, you need to first copy /etc/X11/xinit/xserverrc to ~/.xserverrc: when launching it, you have to specify a new session number:

```
xinit ~/.config/qtile/xinitrc -- :1
```

Examples of custom X startup scripts are available in qtile-examples.

3.2.5 Capturing an xtrace

Occasionally, a bug will be low level enough to require an xtrace of Qtile’s conversations with the X server. To capture one of these, create an xinitrc or similar file with:

```
exec xtrace qtile >> ~/qtile.log
```

This will put the xtrace output in Qtile’s logfile as well. You can then demonstrate the bug, and paste the contents of this file into the bug report.

3.2.6 Coding style

While not all of our code follows PEP8, we do try to adhere to it where possible. All new code should be PEP8 compliant.

The make lint command will run a linter with our configuration over libqtile to ensure your patch complies with reasonable formatting constraints. We also request that git commit messages follow the standard format.

3.2.7 Deprecation policy

When a widget API is changed, you should deprecate the change using libqtile.widget.base.deprecated to warn users, in addition to adding it to the appropriate place in the changelog. We will typically remove deprecated APIs one tag after they are deprecated.

3.2.8 Testing

Of course, your patches should also pass the unit tests as well (i.e. make check). These will be run by travis-ci on every pull request so you can see whether or not your contribution passes.
3.2.9 Resources

Here are a number of resources that may come in handy:

- Inter-Client Conventions Manual
- Extended Window Manager Hints
- A reasonable basic Xlib Manual
4.1 Reference

4.1.1 Scripting Commands

Here is documented some of the commands available on objects in the command tree when running qshell or scripting commands to qtile. Note that this is an incomplete list, some objects, such as *layouts* and *widgets*, may implement their own set of commands beyond those given here.

**Qtile**

```python
class libqtile.manager.Qtile(config, displayName=None, fname=None, no_spawn=False, state=None)
```

This object is the root of the command graph

```python
cmd_add_rule(match_args, rule_args, min_priority=False)
```

Add a dgroup rule, returns rule_id needed to remove it

**Parameters**

- `match_args`: config.Match arguments
- `rule_args`: config.Rule arguments
- `min_priority`: If the rule is added with minimum priority (last) (default: False)

```python
cmd_addgroup(group, label=None, layout=None, layouts=None)
```

Add a group with the given name

```python
cmd_commands()
```

Returns a list of possible commands for this object

*Used by __qsh__ for command completion and online help*

```python
cmd_critical()
```

Set log level to CRITICAL
**cmd_debug()**
Set log level to DEBUG

**cmd_delgroup**(group)
Delete a group with the given name

**cmd_display_kb**(args)
Display table of key bindings

**cmd_doc**(name)
Returns the documentation for a specified command name
Used by __qsh__ to provide online help.

**cmd_error()**
Set log level to ERROR

**cmd_eval**(code)
Evaluates code in the same context as this function
Return value is tuple (success, result), success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

**cmd_findwindow**(prompt='window', widget='prompt')
Launch prompt widget to find a window of the given name

**Parameters**

- **prompt** : Text with which to prompt user (default: “window”)
- **widget** : Name of the prompt widget (default: “prompt”)

**cmd_focus_by_click**(e)
Bring a window to the front

**Parameters**

- **e** [xcb event] Click event used to determine window to focus

**cmd_function**(function, *args, **kwargs)
Call a function with current object as argument

**cmd_get_info()**
Prints info for all groups

**cmd_get_state()**
Get pickled state for restarting qtile

**cmd_get_test_data()**
Returns any content arbitrarily set in the self.test_data attribute. Useful in tests.

**cmd_groups()**
Return a dictionary containing information for all groups

### Examples

groups()

**cmd_hide_show_bar**(position='all')
Toggle visibility of a given bar

**Parameters**

- **position** : one of: “top”, “bottom”, “left”, “right”, or “all” (default: “all”)
Qtile Documentation, Release 0.13.0

cmd_info()
    Set log level to INFO

cmd_internal_windows()  
    Return info for each internal window (bars, for example)

cmd_items(name)
    Returns a list of contained items for the specified name
    Used by __qsh__ to allow navigation of the object graph.

cmd_list_widgets()
    List of all addressible widget names

cmd_loglevel()

cmd_loglevelname()

cmd_next_layout (group=None)
    Switch to the next layout.

    Parameters
    
    group: Group name. If not specified, the current group is assumed

cmd_next_screen()
    Move to next screen

cmd_next_urgent()
    Focus next window with urgent hint

cmd_pause()
    Drops into pdb

cmd_prev_layout (group=None)
    Switch to the previous layout.

    Parameters
    
    group: Group name. If not specified, the current group is assumed

cmd_prev_screen()
    Move to the previous screen

cmd_qtile_info()
    Returns a dictionary of info on the Qtile instance

cmd_qtilecmd(prompt=’command’, widget=’prompt’, messenger=’xmessage’)
    Execute a Qtile command using the client syntax
    Tab completion aids navigation of the command tree

    Parameters
    
    prompt: Text to display at the prompt (default: “command: “)
    widget: Name of the prompt widget (default: “prompt”)
    messenger: Command to display output, set this to None to disable (default: “xmessage”)

cmd_remove_rule (rule_id)
    Remove a dgroup rule by rule_id

cmd_restart()
    Restart qtile
cmd_run_extension (extension)
Run extensions

cmd_run_extension (cls)
Deprecated alias for cmd_run_extension()
Note that it was accepting an extension class, not an instance.

cmd_run_external (full_path)
Run external Python script

cmd_screens ()
Return a list of dictionaries providing information on all screens

cmd_shutdown ()
Quit Qtile

cmd_simulate_keypress (modifiers, key)
Simulates a keypress on the focused window.

Parameters
 modifiers : A list of modifier specification strings. Modifiers can be one of “shift”, “lock”, “control” and “mod1” - “mod5”.
 key : Key specification.

Examples
 simulate_keypress([“control”, “mod2”], “k”)

cmd_spawn (cmd)
Run cmd in a shell.
 cmd may be a string, which is parsed by shlex.split, or a list (similar to subprocess.Popen).

Examples
 spawn(“firefox”)
 spawn([“xterm”, “-T”, “Temporary terminal”])

cmd_spawncmd (prompt=’spawn’, widget=’prompt’, command=’%s’, complete=’cmd’)
Spawn a command using a prompt widget, with tab-completion.

Parameters
 prompt : Text with which to prompt user (default: “spawn: “).
 widget : Name of the prompt widget (default: “prompt”).
 command : command template (default: “%s”).
 complete : Tab completion function (default: “cmd”)

cmd_status ()
Return “OK” if Qtile is running

cmd_switch_groups (groupa, groupb)
Switch position of groupa to groupb
cmd_switchgroup \( (prompt='\text{group}', \ widget='\text{prompt}') \)
Launch prompt widget to switch to a given group to the current screen

**Parameters**

- **prompt**: Text with which to prompt user (default: “group”)
- **widget**: Name of the prompt widget (default: “prompt”)

cmd_sync()
Sync the X display. Should only be used for development

cmd_to_layout_index \( (index, \ group=None) \)
Switch to the layout with the given index in self.layouts.

**Parameters**

- **index**: Index of the layout in the list of layouts.
- **group**: Group name. If not specified, the current group is assumed.

cmd_to_screen \( (n) \)
Warp focus to screen n, where n is a 0-based screen number

**Examples**

to_screen(0)


cmd_togroup \( (prompt='\text{group}', \ widget='\text{prompt}') \)
Launch prompt widget to move current window to a given group

**Parameters**

- **prompt**: Text with which to prompt user (default: “group”)
- **widget**: Name of the prompt widget (default: “prompt”)

cmd_tracemalloc_dump()
Dump tracemalloc snapshot

cmd_tracemalloc_toggle()
Toggle tracemalloc status
Running tracemalloc is required for qtile-top

cmd_warning()
Set log level to WARNING

cmd_windows()
Return info for each client window

**Bar**

class libqtile.bar.Bar \( (widgets, size, **config) \)
A bar, which can contain widgets

**Parameters**

- **widgets**: A list of widget objects.
- **size**: The “thickness” of the bar, i.e. the height of a horizontal bar, or the width of a vertical bar.
### Key and Default Values

<table>
<thead>
<tr>
<th>Key</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>'#000000'</td>
<td>Background colour.</td>
</tr>
<tr>
<td>opacity</td>
<td>1</td>
<td>Bar window opacity.</td>
</tr>
</tbody>
</table>

### cmd_commands()

Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

### cmd_doc(name)

Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

### cmd_eval(code)

Evaluates code in the same context as this function

Return value is tuple (success, result), success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

### cmd_fake_button_press(screen, position, x, y, button=1)

Fake a mouse-button-press on the bar. Co-ordinates are relative to the top-left corner of the bar.

:screen The integer screen offset :position One of “top”, “bottom”, “left”, or “right”

### cmd_function(function, *args, **kwargs)

Call a function with current object as argument

### cmd_info()

Info for this object.

### cmd_items(name)

Returns a list of contained items for the specified name

Used by __qsh__ to allow navigation of the object graph.

### Group

**class libqtile.config.Group**

(name, matches=None, exclusive=False, spawn=None, layout=None, layouts=None, persist=True, init=True, layout_opts=None, screen_affinity=None, position=9223372036854775807, label=None)

Represents a “dynamic” group

These groups can spawn apps, only allow certain Matched windows to be on them, hide when they’re not in use, etc. Groups are identified by their name.

#### Parameters

- **name** [string] the name of this group
- **matches** [default None] list of Match objects whose windows will be assigned to this group
- **exclusive** [boolean] when other apps are started in this group, should we allow them here or not?
- **spawn** [string or list of strings] this will be `exec()` d when the group is created, you can pass either a program name or a list of programs to `exec()`
- **layout** [string] the name of default layout for this group (e.g. ‘max’ or ‘stack’). This is the name specified for a particular layout in config.py or if not defined it defaults in general the class name in all lower case.
layouts [list] the group layouts list overriding global layouts. Use this to define a separate list of layouts for this particular group.

persist [boolean] should this group stay alive with no member windows?

init [boolean] is this group alive when qtile starts?

position [int] group position

label [string] the display name of the group. Use this to define a display name other than name of the group. If set to None, the display name is set to the name.

Screen

class libqtile.config.Screen(top=None, bottom=None, left=None, right=None, x=None, y=None, width=None, height=None)

A physical screen, and its associated paraphernalia.

Define a screen with a given set of Bars of a specific geometry. Note that bar.Bar objects can only be placed at the top or the bottom of the screen (bar.Gap objects can be placed anywhere). Also, x, y, width, and height aren’t specified usually unless you are using ‘fake screens’.

Parameters

top: List of Gap/Bar objects, or None.

bottom: List of Gap/Bar objects, or None.

left: List of Gap/Bar objects, or None.

right: List of Gap/Bar objects, or None.

x [int or None]

y [int or None]

width [int or None]

height [int or None]

cmd_commands ()

Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

cmd_doc (name)

Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

cmd_eval (code)

Evaluates code in the same context as this function

Return value is tuple (success, result), success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_function (function, *args, **kwargs)

Call a function with current object as argument

cmd_info ()

Returns a dictionary of info for this screen.

cmd_items (name)

Returns a list of contained items for the specified name
Used by __qsh__ to allow navigation of the object graph.

```python
cmd_next_group(skip_empty=False, skip_managed=False)
    Switch to the next group

cmd_prev_group(skip_empty=False, skip_managed=False)
    Switch to the previous group

cmd_resize(x=None, y=None, w=None, h=None)
    Resize the screen

cmd_toggle_group(group_name=None)
    Switch to the selected group or to the previously active one

cmd_togglegroup(groupName=None)
    Switch to the selected group or to the previously active one
    Deprecated: use toggle_group()
```

**Window**

```python
class libqtile.window.Window(window, qtile)
```

```python
cmd_bring_to_front()

cmd_commands()
    Returns a list of possible commands for this object
    Used by __qsh__ for command completion and online help

cmd_disable_floating()

cmd_disable_fullscreen()

cmd_disable_maximize()

cmd_disable_minimize()

cmd_doc(name)
    Returns the documentation for a specified command name
    Used by __qsh__ to provide online help.

cmd_down_opacity()

cmd_enable_floating()

cmd_enable_fullscreen()

cmd_enable_maximize()

cmd_enable_minimize()

cmd_eval(code)
    Evaluates code in the same context as this function
    Return value is tuple (success, result), success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_focus(warp=None)
    Focuses the window.

cmd_function(function, *args, **kwargs)
    Call a function with current object as argument
cmd_get_position()

cmd_get_size()

cmd_info()
    Returns a dictionary of info for this object

cmd_inspect()
    Tells you more than you ever wanted to know about a window

cmd_items(name)
    Returns a list of contained items for the specified name
    Used by __qsh__ to allow navigation of the object graph.

cmd_kill()
    Kill this window
    Try to do this politely if the client support this, otherwise be brutal.

cmd_match(*args, **kwargs)

cmd_move_floating(dx, dy, curx, cury)
    Move window by dx and dy

cmd_opacity(opacity)

cmd_resize_floating(dw, dh, curx, cury)
    Add dw and dh to size of window

cmd_set_position(dx, dy, curx, cury)

cmd_set_position_floating(x, y, curx, cury)
    Move window to x and y

cmd_set_size_floating(w, h, curx, cury)
    Set window dimensions to w and h

cmd_static(screen, x, y, width, height)

cmd_toggle_floating()

cmd_toggle_fullscreen()

cmd_toggle_maximize()

cmd_toggle_minimize()

cmd_togroup(groupName=None)
    Move window to a specified group.
    If groupName is not specified, we assume the current group

Examples

Move window to current group:

togroup()

Move window to group “a”:

togroup("a")
**cmd_toscreen** *(index=\text{None})*
Move window to a specified screen.
If index is not specified, we assume the current screen.

**Examples**

Move window to current screen:

```python
toscreen()
```

Move window to screen 0:

```python
toscreen(0)
```

**cmd_up_opacity**

### 4.1.2 Built-in Hooks

**classmethod subscribe.addgroup** *(func)*
Called when group is added

**Arguments**
- \text{qtile} manager instance
- name of new group

**classmethod subscribe.changegroup** *(func)*
Called whenever a group change occurs

**Arguments**
- None

**classmethod subscribe.client_focus** *(func)*
Called whenever focus changes

**Arguments**
- \text{window.Window} object of the new focus.

**classmethod subscribe.client_killed** *(func)*
Called after a client has been unmanaged

**Arguments**
- \text{window.Window} object of the killed window.

**classmethod subscribe.client_managed** *(func)*
Called after Qtile starts managing a new client

Called after a window is assigned to a group, or when a window is made static. This hook is not called for internal windows.

**Arguments**
- \text{window.Window} object of the managed window

---

**Chapter 4. Miscellaneous**
classmethod subscribe.client_mouse_enter(func)
    Called when the mouse enters a client

Arguments
    • window.Window of window entered

classmethod subscribe.client_name_updated(func)
    Called when the client name changes

Arguments
    • window.Window of client with updated name

classmethod subscribe.client_new(func)
    Called before Qtile starts managing a new client

Use this hook to declare windows static, or add them to a group on startup. This hook is not called for internal windows.

Arguments
    • window.Window object

Examples

```python
@libqtile.hook.subscribe.client_new
def func(c):
    if c.name == "xterm":
        c.togroup("a")
    elif c.name == "dzen":
        c.static(0)
```

classmethod subscribe.client_state_changed(func)
    Called whenever client state changes

Never fires

classmethod subscribe.client_type_changed(func)
    Called whenever window type changes

Never fires

classmethod subscribe.client_urgent_hint_changed(func)
    Called when the client urgent hint changes

Arguments
    • window.Window of client with hint change

classmethod subscribe.current_screen_change(func)
    Called when the current screen (i.e. the screen with focus) changes

Arguments
    None

classmethod subscribe.delgroup(func)
    Called when group is deleted

Arguments
    • qtile manager instance
• name of deleted group

classmethod subscribe.float_change(func)
Called when a change in float state is made

Arguments
None

classmethod subscribe.focus_change(func)
Called when focus is changed

Arguments
None

classmethod subscribe.group_window_add(func)
Called when a new window is added to a group

Arguments
None

classmethod subscribe.layout_change(func)
Called on layout change

Arguments
• layout object for new layout
• group object on which layout is changed

classmethod subscribe.net_wm_icon_change(func)
Called on _NET_WM_ICON change

Arguments
• window.Window of client with changed icon

classmethod subscribe.screen_change(func)
Called when a screen is added or screen configuration is changed (via xrandr)

Common usage is simply to call qtile.cmd_restart() on each event (to restart qtile when there is a new monitor):

Arguments
• qtile manager instance
• xproto.randr.ScreenChangeNotify event

Examples

```python
@libqtile.hook.subscribe.screen_change
def restart_on_randr(qtile, ev):
    qtile.cmd_restart()
```

classmethod subscribe.selection_change(func)
Called on selection change

Arguments
• name of the selection
• dictionary describing selection, containing owner and selection as keys
**classmethod** `subscribe.selection_notify(func)`
Called on selection notify

**Arguments**
- name of the selection
- dictionary describing selection, containing `owner` and `selection` as keys

**classmethod** `subscribe.setgroup(func)`
Called when group is changed

**Arguments**
None

**classmethod** `subscribe.startup(func)`
Called when Qtile is started

**Arguments**
None

**classmethod** `subscribe.startup_complete(func)`
Called when Qtile is started after all resources initialized

**Arguments**
None

**classmethod** `subscribe.startup_once(func)`
Called when Qtile has started on first start

This hook is called exactly once per session (i.e. not on each `lazy.restart()`).

**Arguments**
None

**classmethod** `subscribe.window_name_change(func)`
Called whenever a windows name changes

Deprecated: use `client_name_updated`

**Arguments**
None

### 4.1.3 Built-in Layouts

**Floating**

```python
class libqtile.layout.floating.Floating(float_rules=None, no_reposition_match=None, **config)
```

Floating layout, which does nothing with windows but handles focus order
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_float_types</td>
<td>'toolbar', 'notification', 'dialog', 'utility', 'splash'</td>
<td>default wm types to automatically float</td>
</tr>
<tr>
<td>border_focus</td>
<td>'#0000ff'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>fullscreen_border_width</td>
<td>0</td>
<td>Border width for fullscreen.</td>
</tr>
<tr>
<td>max_border_width</td>
<td>0</td>
<td>Border width for maximize.</td>
</tr>
<tr>
<td>name</td>
<td>'floating'</td>
<td>Name of this layout.</td>
</tr>
</tbody>
</table>

Bsp

class libqtile.layout.bsp.Bsp(**config)

This layout is inspired by bspwm, but it does not try to copy its features.

The first client occupies the entire screen space. When a new client is created, the selected space is partitioned in 2 and the new client occupies one of those subspaces, leaving the old client with the other.

The partition can be either horizontal or vertical according to the dimensions of the current space: if its width/height ratio is above a pre-configured value, the subspaces are created side-by-side, otherwise, they are created on top of each other. The partition direction can be freely toggled. All subspaces can be resized and clients can be shuffled around.

An example key configuration is:

```python
Key({mod}, "j", lazy.layout.down()),
Key({mod}, "k", lazy.layout.up()),
Key({mod}, "h", lazy.layout.left()),
Key({mod}, "l", lazy.layout.right()),
Key({mod, "shift"}, "j", lazy.layout.shuffle_down()),
Key({mod, "shift"}, "k", lazy.layout.shuffle_up()),
Key({mod, "shift"}, "h", lazy.layout.shuffle_left()),
Key({mod, "shift"}, "l", lazy.layout.shuffle_right()),
Key({mod, "mod1"}, "j", lazy.layout.flip_down()),
Key({mod, "mod1"}, "k", lazy.layout.flip_up()),
Key({mod, "mod1"}, "h", lazy.layout.flip_left()),
Key({mod, "mod1"}, "l", lazy.layout.flip_right()),
Key({mod, "control"}, "j", lazy.layout.grow_down()),
Key({mod, "control"}, "k", lazy.layout.grow_up()),
Key({mod, "control"}, "h", lazy.layout.grow_left()),
Key({mod, "control"}, "l", lazy.layout.grow_right()),
Key({mod, "shift"}, "n", lazy.layout.normalize()),
Key({mod}, "Return", lazy.layout.toggle_split()),
```
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#881111'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#220000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Border width.</td>
</tr>
<tr>
<td>fair</td>
<td>True</td>
<td>New clients are inserted in the shortest branch.</td>
</tr>
<tr>
<td>grow_amount</td>
<td>10</td>
<td>Amount by which to grow a window/column.</td>
</tr>
<tr>
<td>lower_right</td>
<td>True</td>
<td>New client occupies lower or right subspace.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout.</td>
</tr>
<tr>
<td>name</td>
<td>'bsp'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>ratio</td>
<td>1.6</td>
<td>Width/height ratio that defines the partition direction.</td>
</tr>
</tbody>
</table>

### Columns

**class** `libqtile.layout.columns.Columns(**config)**

Extension of the Stack layout.

The screen is split into columns, which can be dynamically added or removed. Each column can present windows in 2 modes: split or stacked. In split mode, all windows are presented simultaneously, splitting the column space. In stacked mode, only a single window is presented from the stack of windows. Columns and windows can be resized and windows can be shuffled around.

This layout can also emulate “Wmii”, “Vertical”, and “Max”, depending on the default parameters.

An example key configuration is:

```python
Key([mod], "j", lazy.layout.down()),
Key([mod], "k", lazy.layout.up()),
Key([mod], "h", lazy.layout.left()),
Key([mod], "l", lazy.layout.right()),
Key([mod, "shift"], "j", lazy.layout.shuffle_down()),
Key([mod, "shift"], "k", lazy.layout.shuffle_up()),
Key([mod, "shift"], "h", lazy.layout.shuffle_left()),
Key([mod, "shift"], "l", lazy.layout.shuffle_right()),
Key([mod, "control"], "j", lazy.layout.grow_down()),
Key([mod, "control"], "k", lazy.layout.grow_up()),
Key([mod, "control"], "h", lazy.layout.grow_left()),
Key([mod, "control"], "l", lazy.layout.grow_right()),
Key([mod], "Return", lazy.layout.toggle_split()),
Key([mod], "n", lazy.layout.normalize()),
```
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#881111'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_focus_stack</td>
<td>'#881111'</td>
<td>Border colour for the focused window in stacked columns.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#220000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_normal_stack</td>
<td>'#220000'</td>
<td>Border colour for un-focused windows in stacked columns.</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Border width.</td>
</tr>
<tr>
<td>fair</td>
<td>False</td>
<td>Add new windows to the column with least windows.</td>
</tr>
<tr>
<td>grow_amount</td>
<td>10</td>
<td>Amount by which to grow a window/column.</td>
</tr>
<tr>
<td>insert_position</td>
<td>0</td>
<td>Position relative to the current window where new ones are inserted (0 means right above the current window, 1 means right after).</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout.</td>
</tr>
<tr>
<td>name</td>
<td>'columns'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>num_columns</td>
<td>2</td>
<td>Preferred number of columns.</td>
</tr>
<tr>
<td>split</td>
<td>True</td>
<td>New columns presentation mode.</td>
</tr>
</tbody>
</table>

**Matrix**

**class** `libqtile.layout.matrix.Matrix(columns=2, **config)`

This layout divides the screen into a matrix of equally sized cells and places one window in each cell. The number of columns is configurable and can also be changed interactively.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#0000ff'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout.</td>
</tr>
<tr>
<td>name</td>
<td>'matrix'</td>
<td>Name of this layout.</td>
</tr>
</tbody>
</table>

**Max**

**class** `libqtile.layout.max.Max(**config)`

Maximized layout

A simple layout that only displays one window at a time, filling the screen. This is suitable for use on laptops and other devices with small screens. Conceptually, the windows are managed as a stack, with commands to switch to next and previous windows in the stack.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>'max'</td>
<td>Name of this layout.</td>
</tr>
</tbody>
</table>

**MonadTall**

**class** `libqtile.layout.xmonadMonadTall(**config)`

Emulate the behavior of XMonad’s default tiling scheme.

Main-Pane:

A main pane that contains a single window takes up a vertical portion of the screen based on the ratio setting. This ratio can be adjusted with the `cmd_grow_main` and `cmd_shrink_main` or, while the main pane is in focus, `cmd_grow` and `cmd_shrink`. 
Using the `cmd_flip` method will switch which horizontal side the main pane will occupy. The main pane is considered the “top” of the stack.

Secondary-panes:

Occupying the rest of the screen are one or more secondary panes. The secondary panes will share the vertical space of the screen however they can be resized at will with the `cmd_grow` and `cmd_shrink` methods. The other secondary panes will adjust their sizes to smoothly fill all of the space.

Panes can be moved with the `cmd_shuffle_up` and `cmd_shuffle_down` methods. As mentioned the main pane is considered the top of the stack; moving up is counter-clockwise and moving down is clockwise. The opposite is true if the layout is “flipped”.

Normalizing:

To restore all client windows to their default size ratios simply use the `cmd_normalize` method.

Maximizing:

To toggle a client window between its minimum and maximum sizes simply use the `cmd_maximize` on a focused client.

Suggested Bindings:
Key([modkey], "h", lazy.layout.left()),
Key([modkey], "l", lazy.layout.right()),
Key([modkey], "j", lazy.layout.down()),
Key([modkey], "k", lazy.layout.up()),
Key([modkey, "shift"], "h", lazy.layout.swap_left()),
Key([modkey, "shift"], "l", lazy.layout.swap_right()),
Key([modkey, "shift"], "j", lazy.layout.shuffle_down()),
Key([modkey, "shift"], "k", lazy.layout.shuffle_up()),
Key([modkey], "i", lazy.layout.grow()),
Key([modkey], "m", lazy.layout.shrink()),
Key([modkey], "n", lazy.layout.normalize()),
Key([modkey], "o", lazy.layout.maximize()),
Key([modkey, "shift"], "space", lazy.layout.flip()),

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>0</td>
<td>Which side master plane will be placed (one of MonadTall._left or MonadTall._right)</td>
</tr>
<tr>
<td>border_focus</td>
<td>'#ff0000'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Border width.</td>
</tr>
<tr>
<td>change_ratio</td>
<td>0.05</td>
<td>Resize ratio</td>
</tr>
<tr>
<td>change_size</td>
<td>20</td>
<td>Resize change in pixels</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>max_ratio</td>
<td>0.75</td>
<td>The percent of the screen-space the master pane should occupy at maximum.</td>
</tr>
<tr>
<td>min_ratio</td>
<td>0.25</td>
<td>The percent of the screen-space the master pane should occupy at minimum.</td>
</tr>
<tr>
<td>min_secondary_size</td>
<td>85</td>
<td>minimum size in pixel for a secondary pane window</td>
</tr>
<tr>
<td>name</td>
<td>'xmonadtall'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>new_at_current</td>
<td>False</td>
<td>Place new windows at the position of the active window.</td>
</tr>
<tr>
<td>ratio</td>
<td>0.5</td>
<td>The percent of the screen-space the master pane should occupy by default.</td>
</tr>
<tr>
<td>single_border_width</td>
<td>None</td>
<td>Border width for single window</td>
</tr>
</tbody>
</table>

**MonadWide**

```python
class libqtile.layout.xmonad.MonadWide(**config)
```

Emulate the behavior of XMonad's horizontal tiling scheme.

This layout attempts to emulate the behavior of XMonad wide tiling scheme.

Main-Pane:

A main pane that contains a single window takes up a horizontal portion of the screen based on the ratio setting. This ratio can be adjusted with the cmd_grow_main and cmd_shrink_main or, while the main pane is in focus, cmd_grow and cmd_shrink.
Using the `cmd_flip` method will switch which vertical side the main pane will occupy. The main pane is considered the "top" of the stack.

Secondary-panes:

Occupying the rest of the screen are one or more secondary panes. The secondary panes will share the horizontal space of the screen however they can be resized at will with the `cmd_grow` and `cmd_shrink` methods. The other secondary panes will adjust their sizes to smoothly fill all of the space.

Panes can be moved with the `cmd_shuffle_up` and `cmd_shuffle_down` methods. As mentioned the main pane is considered the top of the stack; moving up is counter-clockwise and moving down is clockwise. The opposite is true if the layout is "flipped".

Normalizing:

To restore all client windows to their default size ratios simply use the `cmd_normalize` method.

Maximizing:

To toggle a client window between its minimum and maximum sizes simply use the `cmd_maximize` on a focused client.

Suggested Bindings:

```python
Key([modkey], "h", lazy.layout.left()),
Key([modkey], "l", lazy.layout.right()),
Key([modkey], "j", lazy.layout.down()),
```

(continues on next page)
Key([modkey], "k", lazy.layout.up()),
Key([modkey, "shift"], "h", lazy.layout.swap_left()),
Key([modkey, "shift"], "l", lazy.layout.swap_right()),
Key([modkey, "shift"], "j", lazy.layout.shuffle_down()),
Key([modkey, "shift"], "k", lazy.layout.shuffle_up()),
Key([modkey], "i", lazy.layout.grow()),
Key([modkey, "shift"], "space", lazy.layout.flip()),

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>0</td>
<td>Which side master plane will be placed (one of MonadTall._left or MonadTall._right)</td>
</tr>
<tr>
<td>border_focus</td>
<td>'#ff0000'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Border width.</td>
</tr>
<tr>
<td>change_ratio</td>
<td>0.05</td>
<td>Resize ratio</td>
</tr>
<tr>
<td>change_size</td>
<td>20</td>
<td>Resize change in pixels</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>max_ratio</td>
<td>0.75</td>
<td>The percent of the screen-space the master pane should occupy at maximum.</td>
</tr>
<tr>
<td>min_ratio</td>
<td>0.25</td>
<td>The percent of the screen-space the master pane should occupy at minimum.</td>
</tr>
<tr>
<td>min_secondary_size</td>
<td>85</td>
<td>minimum size in pixel for a secondary pane window</td>
</tr>
<tr>
<td>name</td>
<td>'xmonadtall'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>new_at_current</td>
<td>False</td>
<td>Place new windows at the position of the active window.</td>
</tr>
<tr>
<td>ratio</td>
<td>0.5</td>
<td>The percent of the screen-space the master pane should occupy by default.</td>
</tr>
<tr>
<td>single_border_width</td>
<td></td>
<td>Border width for single window</td>
</tr>
</tbody>
</table>

**RatioTile**

```python
class libqtile.layout.ratiotile.RatioTile(**config)
```

Tries to tile all windows in the width/height ratio passed in

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#0000ff'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>fancy</td>
<td>False</td>
<td>Use a different method to calculate window sizes.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>name</td>
<td>'ratiotile'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>ratio</td>
<td>1.618</td>
<td>Ratio of the tiles</td>
</tr>
<tr>
<td>ratio_increment</td>
<td>0.1</td>
<td>Amount to increment per ratio increment</td>
</tr>
</tbody>
</table>

**Slice**

```python
class libqtile.layout.slice.Slice(**config)
```

Slice layout
This layout cuts piece of screen and places a single window on that piece, and delegates other window placement to other layout

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fallback</td>
<td><code>&lt;libqtile.layout.max.Max object at 0x7f3ca434ef28&gt;</code></td>
<td>Fallback layout</td>
</tr>
<tr>
<td>name</td>
<td>'max'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>role</td>
<td>None</td>
<td>WM_WINDOW_ROLE to match</td>
</tr>
<tr>
<td>side</td>
<td>'left'</td>
<td>Side of the slice (left, right, top, bottom)</td>
</tr>
<tr>
<td>width</td>
<td>256</td>
<td>Slice width</td>
</tr>
<tr>
<td>wmclass</td>
<td>None</td>
<td>WM_CLASS to match</td>
</tr>
<tr>
<td>wname</td>
<td>None</td>
<td>WM_NAME to match</td>
</tr>
</tbody>
</table>

**Stack**

class `libqtile.layout.stack.Stack(**config)`

A layout composed of stacks of windows

The stack layout divides the screen horizontally into a set of stacks. Commands allow you to switch between stacks, to next and previous windows within a stack, and to split a stack to show all windows in the stack, or unsplit it to show only the current window.

Unlike the columns layout the number of stacks is fixed.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autosplit</td>
<td>False</td>
<td>Auto split all new stacks.</td>
</tr>
<tr>
<td>border_focus</td>
<td>'#0000ff'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>fair</td>
<td>False</td>
<td>Add new windows to the stacks in a round robin way.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>name</td>
<td>'stack'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>num_stacks</td>
<td>2</td>
<td>Number of stacks.</td>
</tr>
</tbody>
</table>

**Tile**

class `libqtile.layout.tile.Tile(ratio=0.618, masterWindows=1, expand=True, ratio_increment=0.05, add_on_top=True, add_after_last=False, shift_windows=False, master_match=None, **config)`

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#0000ff'</td>
<td>Border colour for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#000000'</td>
<td>Border colour for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>name</td>
<td>'tile'</td>
<td>Name of this layout.</td>
</tr>
</tbody>
</table>
TreeTab

class libqtile.layout.tree.TreeTab(**config)

Tree Tab Layout

This layout works just like Max but displays tree of the windows at the left border of the screen, which allows you to overview all opened windows. It's designed to work with uzbl-browser but works with other windows too.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_bg</td>
<td>'000080'</td>
<td>Background color of active tab</td>
</tr>
<tr>
<td>active_fg</td>
<td>‘ffffff’</td>
<td>Foreground color of active tab</td>
</tr>
<tr>
<td>bg_color</td>
<td>'000000'</td>
<td>Background color of tabs</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Width of the border</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>14</td>
<td>Font pixel size.</td>
</tr>
<tr>
<td>inactive_bg</td>
<td>'606060'</td>
<td>Background color of inactive tab</td>
</tr>
<tr>
<td>inactive_fg</td>
<td>‘ffffff’</td>
<td>Foreground color of inactive tab</td>
</tr>
<tr>
<td>level_shift</td>
<td>8</td>
<td>Shift for children tabs</td>
</tr>
<tr>
<td>margin_left</td>
<td>6</td>
<td>Left margin of tab panel</td>
</tr>
<tr>
<td>margin_y</td>
<td>6</td>
<td>Vertical margin of tab panel</td>
</tr>
<tr>
<td>name</td>
<td>'treetab'</td>
<td>Name of this layout.</td>
</tr>
<tr>
<td>padding_left</td>
<td>6</td>
<td>Left padding for tabs</td>
</tr>
<tr>
<td>padding_x</td>
<td>6</td>
<td>Left padding for tab label</td>
</tr>
<tr>
<td>padding_y</td>
<td>2</td>
<td>Top padding for tab label</td>
</tr>
<tr>
<td>panel_width</td>
<td>150</td>
<td>Width of the left panel</td>
</tr>
<tr>
<td>previous_on_rm</td>
<td>False</td>
<td>Focus previous window on close instead of first.</td>
</tr>
<tr>
<td>section_bottom</td>
<td>6</td>
<td>Bottom margin of section</td>
</tr>
<tr>
<td>section_fg</td>
<td>‘ffffff’</td>
<td>Color of section label</td>
</tr>
<tr>
<td>section_fonts</td>
<td>'z8l'</td>
<td>Font pixel size of section label</td>
</tr>
<tr>
<td>section_left</td>
<td>4</td>
<td>Left margin of section label</td>
</tr>
<tr>
<td>section_padding</td>
<td>4</td>
<td>Bottom of margin section label</td>
</tr>
<tr>
<td>section_top</td>
<td>4</td>
<td>Top margin of section label</td>
</tr>
<tr>
<td>sections</td>
<td>['Default']</td>
<td>Foreground color of inactive tab</td>
</tr>
<tr>
<td>vspace</td>
<td>2</td>
<td>Space between tabs</td>
</tr>
</tbody>
</table>

VerticalTile

class libqtile.layout.verticaltile.VerticalTile(**config)

Tiling layout that works nice on vertically mounted monitors

The available height gets divided by the number of panes, if no pane is maximized. If one pane has been maximized, the available height gets split in master- and secondary area. The maximized pane (master pane) gets the full height of the master area and the other panes (secondary panes) share the remaining space. The master area (at default 75%) can grow and shrink via keybindings.

```
----------------- ----------------- ---
| | | | |
| 1 | <-- Panes | | |
|-----------------| | | |
```
Normal behavior. No One maximized pane in the master area maximized pane. No and two secondary panes in the specific areas. secondary area.

In some cases VerticalTile can be useful on horizontal mounted monitors two. For example if you want to have a webbrowser and a shell below it.

Suggested keybindings:

```python
Key([modkey], 'j', lazy.layout.down()),
Key([modkey], 'k', lazy.layout.up()),
Key([modkey], 'Tab', lazy.layout.next()),
Key([modkey, 'shift'], 'Tab', lazy.layout.next()),
Key([modkey, 'shift'], 'j', lazy.layout.shuffle_down()),
Key([modkey, 'shift'], 'k', lazy.layout.shuffle_up()),
Key([modkey], 'm', lazy.layout.maximize()),
Key([modkey], 'n', lazy.layout.normalize()),
```

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_focus</td>
<td>'#FF0000'</td>
<td>Border color for the focused window.</td>
</tr>
<tr>
<td>border_normal</td>
<td>'#FFFFFF'</td>
<td>Border color for un-focused windows.</td>
</tr>
<tr>
<td>border_width</td>
<td>1</td>
<td>Border width.</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Border margin.</td>
</tr>
<tr>
<td>name</td>
<td>'verticaltile'</td>
<td>Name of this layout.</td>
</tr>
</tbody>
</table>

**Wmii**

```python
class libqtile.layout.wmii.Wmii(**config)
```

This layout is deprecated in favor of *Columns*.

The only difference between the two are the default parameters.
### Zoomy

**class** `libqtile.layout.zoomy.Zoomy(**config)**

A layout with single active windows, and few other previews at the right

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnwidth</td>
<td>150</td>
<td>Width of the right column</td>
</tr>
<tr>
<td>margin</td>
<td>0</td>
<td>Margin of the layout</td>
</tr>
<tr>
<td>name</td>
<td>'zoomy'</td>
<td>Name of this layout</td>
</tr>
<tr>
<td>property_big</td>
<td>'1.0'</td>
<td>Property value to set on normal window</td>
</tr>
<tr>
<td>property_name</td>
<td>'ZOOM'</td>
<td>Property to set on zoomed window</td>
</tr>
<tr>
<td>property_small</td>
<td>'0.1'</td>
<td>Property value to set on zoomed window</td>
</tr>
</tbody>
</table>

### 4.1.4 Built-in Widgets

**AGroupBox**

**class** `libqtile.widget.AGroupBox(**config)**

A widget that graphically displays the current group

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>border</td>
<td>'000000'</td>
<td>group box border color</td>
</tr>
<tr>
<td>borderwidth</td>
<td>3</td>
<td>Current group border width</td>
</tr>
<tr>
<td>center_aligned</td>
<td>False</td>
<td>center-aligned group box</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>True</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>False</td>
<td>Padding. Calculated if None.</td>
</tr>
</tbody>
</table>
Backlight

class libqtile.widget.Backlight(**config)
A simple widget to show the current brightness of a monitor

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>backlight_name</td>
<td>'acpi_video0'</td>
<td>ACPI name of a backlight device</td>
</tr>
<tr>
<td>brightness_file</td>
<td>'brightness'</td>
<td>Name of file with the current brightness in /sys/class/backlight/backlight_name</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'{percent: 2.0%}'</td>
<td>Display format</td>
</tr>
<tr>
<td>markup</td>
<td>True</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>max_brightness_file</td>
<td>'max_brightness'</td>
<td>Name of file with the maximum brightness in /sys/class/backlight/backlight_name</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>step</td>
<td>10</td>
<td>Percent of backlight every scroll changed</td>
</tr>
<tr>
<td>update_interval</td>
<td>0.2</td>
<td>The delay in seconds between updates</td>
</tr>
</tbody>
</table>

BitcoinTicker

class libqtile.widget.BitcoinTicker(**config)
A bitcoin ticker widget, data provided by the coinbase.com API. Defaults to displaying currency in whatever the current locale is. Examples:

```python
# display the average price of bitcoin in local currency
widget.BitcoinTicker()

# display it in Euros:
widget.BitcoinTicker(currency="EUR")
```

Supported bar orientations: horizontal only
### CPUGraph

**class** `libqtile.widget.CPUGraph(**config)`

Display CPU usage graph

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>border_color</td>
<td>'215578'</td>
<td>Widget border color</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Widget border width</td>
</tr>
<tr>
<td>core</td>
<td>'all'</td>
<td>Which core to show (all/0/1/2/...)</td>
</tr>
<tr>
<td>fill_color</td>
<td>'1667EB.3'</td>
<td>Fill color for linefill graph</td>
</tr>
<tr>
<td>frequency</td>
<td>1</td>
<td>Update frequency in seconds</td>
</tr>
<tr>
<td>graph_color</td>
<td>'18BAEB'</td>
<td>Graph color</td>
</tr>
<tr>
<td>line_width</td>
<td>3</td>
<td>Line width</td>
</tr>
<tr>
<td>margin_x</td>
<td>3</td>
<td>Margin X</td>
</tr>
<tr>
<td>margin_y</td>
<td>3</td>
<td>Margin Y</td>
</tr>
<tr>
<td>samples</td>
<td>100</td>
<td>Count of graph samples.</td>
</tr>
<tr>
<td>start_pos</td>
<td>'bottom'</td>
<td>Drawer starting position ('bottom'/'top')</td>
</tr>
<tr>
<td>type</td>
<td>'linefill'</td>
<td>'box', 'line', 'linefill'</td>
</tr>
</tbody>
</table>

### Canto

**class** `libqtile.widget.Canto(**config)`

Display RSS feeds updates using the canto console reader

Supported bar orientations: horizontal only
### CapsNumLockIndicator

**class** `libqtile.widget.CapsNumLockIndicator(**config)`

Really simple widget to show the current Caps/Num Lock state.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
</tbody>
</table>
| one_format   | '{name}:
{number}' | One feed display format                                                     |
| padding      | None            | Padding. Calculated if None.                                                |
| update_interval | 1600          | Update interval in seconds, if none, the widget updates whenever the event loop is idle. |

### CheckUpdates

**class** `libqtile.widget.CheckUpdates(**config)`

Shows number of pending updates in different unix systems

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>10.5</td>
<td>Update Time in seconds.</td>
</tr>
</tbody>
</table>
### Clipboard

**class** `libqtile.widget.Clipboard(width=CALCULATED, **config)`

Display current clipboard contents

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
<td></td>
</tr>
<tr>
<td>blacklist</td>
<td>['keepassx']</td>
<td>list with blacklisted wm_class, sadly not every clipboard window sets them, keepassx does.Clipboard contents from blacklisted wm_classes will be replaced by the value of blacklist_text.</td>
<td></td>
</tr>
<tr>
<td>blacklist_text</td>
<td>***********</td>
<td>text to display when the wm_class is blacklisted</td>
<td></td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
<td></td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
<td></td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
<td></td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
<td></td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
<td></td>
</tr>
<tr>
<td>max_width</td>
<td>10</td>
<td>maximum number of characters to display (None for all, useful when width is bar.STRETCH)</td>
<td></td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
<td></td>
</tr>
<tr>
<td>selection</td>
<td>'CLIPBOARD'</td>
<td>the selection to display(CLIPBOARD or PRIMARY)</td>
<td></td>
</tr>
<tr>
<td>timeout</td>
<td>10</td>
<td>Default timeout (seconds) for display text, None to keep forever</td>
<td></td>
</tr>
</tbody>
</table>

### Clock

**class** `libqtile.widget.Clock(**config)`

A simple but flexible text-based clock

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'%H:%M'</td>
<td>A Python datetime format string</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>timezone</td>
<td>None</td>
<td>The timezone to use for this clock, e.g. “US/Central” (or anything in /usr/share/zoneinfo). None means the default timezone.</td>
</tr>
<tr>
<td>update_interval</td>
<td>1.0</td>
<td>Update interval for the clock</td>
</tr>
</tbody>
</table>

**Cmus**

```python
class libqtile.widget.Cmus(**config)**
```

A simple Cmus widget.

Show the artist and album of now listening song and allow basic mouse control from the bar:

- toggle pause (or play if stopped) on left click;
- skip forward in playlist on scroll up;
- skip backward in playlist on scroll down.

Cmus (https://cmus.github.io) should be installed.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>max_chars</td>
<td>0</td>
<td>Maximum number of characters to display in widget.</td>
</tr>
<tr>
<td>noplay_color</td>
<td>'cecece'</td>
<td>Text colour when not playing.</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>play_color</td>
<td>'00ff00'</td>
<td>Text colour when playing.</td>
</tr>
<tr>
<td>update_interval</td>
<td>10.5</td>
<td>Update Time in seconds.</td>
</tr>
</tbody>
</table>

**Countdown**

```python
class libqtile.widget.Countdown(**config)**
```

A simple countdown timer text widget

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>date</td>
<td>datetime(2018, 12, 24, 3, 9, 58, 803063)</td>
<td>The datetime for the end of the countdown</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground color</td>
</tr>
<tr>
<td>format</td>
<td>'{D}d {H}h {M}m {S}s'</td>
<td>Format of the displayed text. Available variables:{D} == days, {H} == hours, {M} == minutes, {S} seconds.</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>1.0</td>
<td>Update interval in seconds for the clock</td>
</tr>
</tbody>
</table>

CurrentLayout

class libqtile.widget.CurrentLayout (width=CALCULATED, **config)

Display the name of the current layout of the current group of the screen, the bar containing the widget, is on.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground color</td>
</tr>
<tr>
<td>format</td>
<td>'{D}d {H}h {M}m {S}s'</td>
<td>Format of the displayed text. Available variables:{D} == days, {H} == hours, {M} == minutes, {S} seconds.</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
</tbody>
</table>

CurrentLayoutIcon

class libqtile.widget.CurrentLayoutIcon(**config)

Display the icon representing the current layout of the current group of the screen on which the bar containing the widget is.

If you are using custom layouts, a default icon with question mark will be displayed for them. If you want to use custom icon for your own layout, for example, FooGrid, then create a file named “layout-foogrid.png” and place it in ~/.icons directory. You can as well use other directories, but then you need to specify those directories in custom_icon_paths argument for this plugin.

The order of icon search is:

- dirs in custom_icon_paths config argument
- ~/.icons
- built-in qtile icons

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>custom_icon_paths</td>
<td></td>
<td>List of folders where to search icons before using built-in icons or icons in ~/.icons dir. This can also be used to provide missing icons for custom layouts. Defaults to empty list.</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use Pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>scale</td>
<td>1</td>
<td>Scale factor relative to the bar height. Defaults to 1</td>
</tr>
</tbody>
</table>

**CurrentScreen**

class libqtile.widget.CurrentScreen(*width=CALCULATED, **config*)

Indicates whether the screen this widget is on is currently active or not

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_color</td>
<td>'00ff00'</td>
<td>Color when screen is active</td>
</tr>
<tr>
<td>active_text</td>
<td>'A'</td>
<td>Text displayed when the screen is active</td>
</tr>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>inactive_color</td>
<td>'ff0000'</td>
<td>Color when screen is inactive</td>
</tr>
<tr>
<td>inactive_text</td>
<td>'I'</td>
<td>Text displayed when the screen is inactive</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use Pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
</tbody>
</table>

**DF**

class libqtile.widget.DF(**config**)

Disk Free Widget

By default the widget only displays if the space is less than warn_space.

Supported bar orientations: horizontal only
### Qtile Documentation, Release 0.13.0

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'{p}(uf</td>
<td>m){{r:.0f}}%'</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>measure</td>
<td>'G'</td>
<td>Measurement (G, M, B)</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>partition</td>
<td>'/'</td>
<td>the partition to check space</td>
</tr>
<tr>
<td>update_interval</td>
<td>60</td>
<td>The update interval</td>
</tr>
<tr>
<td>visible_on_warn</td>
<td>True</td>
<td>Only display if warning</td>
</tr>
<tr>
<td>warn_color</td>
<td>'ff0000'</td>
<td>Warning color</td>
</tr>
<tr>
<td>warn_space</td>
<td>2</td>
<td>Warning space in scale defined by the measure option.</td>
</tr>
</tbody>
</table>

---

### DebugInfo

**class** `libqtile.widget.DebugInfo(**config)`

Displays debugging infos about selected window

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
</tbody>
</table>

---

### GenPollText

**class** `libqtile.widget.GenPollText(**config)`

A generic text widget that polls using poll function to get the text

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>func</td>
<td>None</td>
<td>Poll Function</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>1600</td>
<td>Update interval in seconds, if none, the widget updates whenever the event loop is idle.</td>
</tr>
</tbody>
</table>

**GenPollUrl**

class libqtile.widget.GenPollUrl(**config**)
A generic text widget that polls an url and parses it using parse function

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>data</td>
<td>None</td>
<td>Post Data</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>headers</td>
<td>{}</td>
<td>Extra Headers</td>
</tr>
<tr>
<td>json</td>
<td>True</td>
<td>Is Json?</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>parse</td>
<td>None</td>
<td>Parse Function</td>
</tr>
<tr>
<td>update_interval</td>
<td>1600</td>
<td>Update interval in seconds, if none, the widget updates whenever the event loop is idle.</td>
</tr>
<tr>
<td>url</td>
<td>None</td>
<td>Url</td>
</tr>
<tr>
<td>user_agent</td>
<td>'Qtile'</td>
<td>Set the user agent</td>
</tr>
<tr>
<td>xml</td>
<td>False</td>
<td>Is XML?</td>
</tr>
</tbody>
</table>

**GmailChecker**

class libqtile.widget.GmailChecker(**config**)
A simple gmail checker

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>email_path</td>
<td>'INBOX'</td>
<td>email_path</td>
</tr>
<tr>
<td>fmt</td>
<td>'inbox[%s], unseen[%s]'</td>
<td>fmt</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>password</td>
<td>None</td>
<td>password</td>
</tr>
<tr>
<td>status_only_unseen</td>
<td>False</td>
<td>Only show unseen messages</td>
</tr>
<tr>
<td>update_interval</td>
<td>30</td>
<td>Update time in seconds.</td>
</tr>
<tr>
<td>username</td>
<td>None</td>
<td>username</td>
</tr>
</tbody>
</table>

**GroupBox**

```python
class libqtile.widget.GroupBox(**config)
```

A widget that graphically displays the current group. All groups are displayed by their label. If the label of a group is the empty string that group will not be displayed.

Supported bar orientations: horizontal only
### HDDBusyGraph

class libqtile.widget.HDBusyGraph(**config)
Display HDD busy time graph


Supported bar orientations: horizontal only
HDDGraph

class libqtile.widget.HDDGraph(**config)
Display HDD free or used space graph

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>border_color</td>
<td>'215578'</td>
<td>Widget border color</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Widget border width</td>
</tr>
<tr>
<td>device</td>
<td>'sda'</td>
<td>Block device to display info for</td>
</tr>
<tr>
<td>fill_color</td>
<td>'1667EB.3'</td>
<td>Fill color for linefill graph</td>
</tr>
<tr>
<td>frequency</td>
<td>1</td>
<td>Update frequency in seconds</td>
</tr>
<tr>
<td>graph_color</td>
<td>'18BAEB'</td>
<td>Graph color</td>
</tr>
<tr>
<td>line_width</td>
<td>3</td>
<td>Line width</td>
</tr>
<tr>
<td>margin_x</td>
<td>3</td>
<td>Margin X</td>
</tr>
<tr>
<td>margin_y</td>
<td>3</td>
<td>Margin Y</td>
</tr>
<tr>
<td>samples</td>
<td>100</td>
<td>Count of graph samples.</td>
</tr>
<tr>
<td>start_pos</td>
<td>'bottom'</td>
<td>Drawer starting position ('bottom'/'top')</td>
</tr>
<tr>
<td>type</td>
<td>'linefill'</td>
<td>'box', 'line', 'linefill'</td>
</tr>
</tbody>
</table>

IdleRPG

class libqtile.widget.IdleRPG(**config)
A widget for monitoring and displaying IdleRPG stats.

# display idlerpg stats for the player 'pants' on freenode's #idlerpg
widget.IdleRPG(url="http://xethron.lolhosting.net/xml.php?player=pants")

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>data</td>
<td>None</td>
<td>Post Data</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'IdleRPG: (online) TTL: {ttl}'</td>
<td>Display format</td>
</tr>
<tr>
<td>headers</td>
<td>{}</td>
<td>Extra Headers</td>
</tr>
<tr>
<td>json</td>
<td>False</td>
<td>Not json :</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>parse</td>
<td>None</td>
<td>Parse Function</td>
</tr>
<tr>
<td>update_interval</td>
<td>600</td>
<td>Update interval in seconds, if none, the widget updates whenever the event loop is idle.</td>
</tr>
<tr>
<td>url</td>
<td>None</td>
<td>Url</td>
</tr>
<tr>
<td>user_agent</td>
<td>'Qtile'</td>
<td>Set the user agent</td>
</tr>
<tr>
<td>xml</td>
<td>True</td>
<td>Is XML :)</td>
</tr>
</tbody>
</table>

**Image**

libqtile.widget.Image

alias of libqtile.widget.import_error.make_error.<locals>.ImportErrorWidget

**ImapWidget**

class libqtile.widget.ImapWidget(**config**)

Email IMAP widget

This widget will scan one of your imap email boxes and report the number of unseen messages present. I've configured it to only work with imap with ssl. Your password is obtained from the Gnome Keyring.

Writing your password to the keyring initially is as simple as (changing out <userid> and <password> for your userid and password):

1. create the file ~/.local/share/python_keyring/keyringrc.cfg with the following contents:

```
[backend]
default-keyring=keyring.backends.Gnome.Keyring
keyring-path=/home/<userid>/.local/share/keyring/
```

2. Execute the following python shell script once:

```
#!/usr/bin/env python3
import keyring
user = <userid>
password = <password>
keyring.set_password('imapwidget', user, password)
```

mbox names must include the path to the mbox (except for the default INBOX). So, for example if your mailroot is ~/Maildir, and you want to look at the mailbox at HomeMail/fred, the mbox setting would be: mbox="~/.Maildir/HomeMail/fred". Note the nested sets of quotes! Labels can be whatever you choose, of course.
Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>label</td>
<td>'INBOX'</td>
<td>label for display</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>mbox</td>
<td>'&quot;INBOX&quot;'</td>
<td>mailbox to fetch</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>server</td>
<td>None</td>
<td>email server name</td>
</tr>
<tr>
<td>update_interval</td>
<td>600</td>
<td>Update interval in seconds, if none, the widget updates whenever the event loop is idle.</td>
</tr>
<tr>
<td>user</td>
<td>None</td>
<td>email username</td>
</tr>
</tbody>
</table>

**KeyboardKbdd**

```python
class libqtile.widget.KeyboardKbdd(**config)
```

Widget for changing keyboard layouts per window, using kbdd

kbdd should be installed and running, you can get it from: https://github.com/qnikst/kbdd

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>colours</td>
<td>None</td>
<td>foreground colour for each layout  either None or a list of colours. example: ['ffffff', 'E6F0AF'].</td>
</tr>
<tr>
<td>configured_keyboards</td>
<td>['us', 'ir']</td>
<td>your predefined list of keyboard layouts.example: ['us', 'ir', 'es']</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None (no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>1</td>
<td>Update interval in seconds.</td>
</tr>
</tbody>
</table>

**KeyboardLayout**

```python
class libqtile.widget.KeyboardLayout(**config)
```

Widget for changing and displaying the current keyboard layout

It requires setxkbmap to be available in the system.

Supported bar orientations: horizontal only
KhalCalendar

class libqtile.widget.KhalCalendar(**config)

Khal calendar widget

This widget will display the next appointment on your Khal calendar in the qtile status bar. Appointments within the “reminder” time will be highlighted.

Supported bar orientations: horizontal only

LaunchBar

class libqtile.widget.LaunchBar(progs=None, width=CALCULATED, **config)

A widget that display icons to launch the associated command

Parameters

progs: a list of tuples (software_name, command_to_execute, comment), for example:

([('thunderbird', 'thunderbird -safe-mode', 'launch thunderbird in safe mode')], ('logout', 'qshell:self.qtile.cmd_shutdown()', 'logout from qtile'))

Supported bar orientations: horizontal only
Maildir

class libqtile.widget.Maildir(**config)
A simple widget showing the number of new mails in maildir mailboxes

Supported bar orientations: horizontal only

Memory

class libqtile.widget.Memory(**config)
Displays memory usage

Supported bar orientations: horizontal only
### MemoryGraph

**class** `libqtile.widget.MemoryGraph(**config)`

Displays a memory usage graph

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>fmt</td>
<td>'{MemUsed}M/(\text{MemTotal})M'</td>
<td>see /proc/meminfo for field names</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>False</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>1600</td>
<td>Update interval in seconds, if none, the widget updates whenever the event loop is idle.</td>
</tr>
</tbody>
</table>

### Moc

**class** `libqtile.widget.Moc(**config)`

A simple MOC widget.

Show the artist and album of now listening song and allow basic mouse control from the bar:

- toggle pause (or play if stopped) on left click;
- skip forward in playlist on scroll up;
- skip backward in playlist on scroll down.

MOC (http://moc.daper.net) should be installed.

Supported bar orientations: horizontal only
## Mpd

**class libqtile.widget.Mpd(**config)

A widget for the Music Player Daemon (MPD)

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>do_color_progress</td>
<td>True</td>
<td>Whether to indicate progress in song by altering message color</td>
</tr>
<tr>
<td>fmt_playing</td>
<td>'%a - %t [%v%%]'</td>
<td>Format string to display when playing/paused</td>
</tr>
<tr>
<td>fmt_stopped</td>
<td>'Stopped [%v%%]'</td>
<td>Format strings to display when stopped</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>foreground_progress</td>
<td>'ffffff'</td>
<td>Foreground progress colour</td>
</tr>
<tr>
<td>host</td>
<td>'localhost'</td>
<td>Host to connect to, can be either an IP address or a UNIX socket path</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>msg_nc</td>
<td>'Mpd off'</td>
<td>Which message to show when we’re not connected</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>password</td>
<td>None</td>
<td>Password to use</td>
</tr>
<tr>
<td>port</td>
<td>6600</td>
<td>Port to connect to</td>
</tr>
<tr>
<td>reconnect</td>
<td>False</td>
<td>Attempt to reconnect if initial connection failed</td>
</tr>
<tr>
<td>reconnect_interval</td>
<td></td>
<td>Time to delay between connection attempts.</td>
</tr>
<tr>
<td>update_interval</td>
<td>10.5</td>
<td>Update Time in seconds</td>
</tr>
</tbody>
</table>

- **background**: Widget background color
- **font**: Default font
- **fontshadow**: Font shadow color, default is None (no shadow)
- **fontsize**: Font size. Calculated if None.
- **foreground**: Foreground colour
- **markup**: Whether or not to use pango markup
- **max_chars**: Maximum number of characters to display in widget.
- **noplay_color**: Text colour when not playing.
- **padding**: Padding. Calculated if None.
- **play_color**: Text colour when playing.
- **update_interval**: Update Time in seconds.
**Mpd2**

```python
class libqtile.widget.Mpd2(status_format='{play_status}{artist}/[title] [[repeat][random][single][consume][updating_db]]', prepare_status={'consume': <function option._convert at 0x7f3ca4261bf8>, 'random': <function option._convert at 0x7f3ca4261ae8>, 'repeat': <function option._convert at 0x7f3ca4261a60>, 'single': <function option._convert at 0x7f3ca4261b70>, 'updating_db': <function option._convert at 0x7f3ca4261c80>}, **config)
```

A widget for Music Player Daemon (MPD) based on python-mpd2

This widget exists since python-mpd library is no more supported.

**Parameters**

- **status_format**: format string to display status

  Full list of values see in `status` and `currentsong` commands


  Default:

  ```
  {play_status}{artist}/[title] [[repeat][random][single][consume]
  %{updating_db}]
  ```

  play_status is string from `play_states` dict

  Note that `time` property of song renamed to `fulltime` to prevent conflicts with status information during formatting.

- **prepare_status**: dict of functions for replace values in status with custom

  ```
  f(status, key, space_element) => str
  ```

  Supported bar orientations: horizontal only
### Mpris

```python
class libqtile.widget.Mpris(**config)
```

**MPRIS player widget**

A widget which displays the current track/artist of your favorite MPRIS player. It should work with all players which implement a reasonably correct version of MPRIS, though I have only tested it with clementine.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>command</td>
<td>None</td>
<td>Executable command by &quot;command&quot; shortcut</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>host</td>
<td>'localhost'</td>
<td>Host of mpd server</td>
</tr>
<tr>
<td>idletimeout</td>
<td>5</td>
<td>MPDClient idle command timeout</td>
</tr>
<tr>
<td>keys</td>
<td>('command': None, 'next': 5, 'previous': 4, 'stop': 3, 'toggle': 1)</td>
<td>Shortcut keys</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>no_connection</td>
<td>'No connection'</td>
<td>Text when mpd is disconnected</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>password</td>
<td>None</td>
<td>Password for auth on mpd server</td>
</tr>
<tr>
<td>play_states</td>
<td>('pause': '', 'play': '', 'stop': '')</td>
<td>Play state mapping</td>
</tr>
<tr>
<td>port</td>
<td>6600</td>
<td>Port of mpd server</td>
</tr>
<tr>
<td>space</td>
<td>'-'</td>
<td>Space keeper</td>
</tr>
<tr>
<td>timeout</td>
<td>30</td>
<td>MPDClient timeout</td>
</tr>
<tr>
<td>update_interval</td>
<td>1</td>
<td>Interval of update widget</td>
</tr>
</tbody>
</table>

**stop_pause_text**: 'Stopped' Optional text to display when in the stopped/paused state
Mpris2

class libqtile.widget.Mpris2(**config)

An MPRIS 2 widget

A widget which displays the current track/artist of your favorite MPRIS player. It should work with all MPRIS 2 compatible players which implement a reasonably correct version of MPRIS, though I have only tested it with audacious. This widget scrolls the text if necessary and information that is displayed is configurable.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>display_metadata</td>
<td>['xesam:title','xesam:album','xesam:artist']</td>
<td>Which metadata identifiers to display. See <a href="http://www.freedesktop.org/wiki/Specifications/mpris-spec/metadata/#index5h3">http://www.freedesktop.org/wiki/Specifications/mpris-spec/metadata/#index5h3</a> for available values</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'fffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>name</td>
<td>'audacious'</td>
<td>Name of the MPRIS widget.</td>
</tr>
<tr>
<td>objname</td>
<td>'org.mpris.MediaPlayer2.audacious'</td>
<td>DBUS MPRIS 2 compatible player identifier- Find it out with dbus-monitor - Also see: <a href="http://specifications.freedesktop.org/mpris-spec/latest/#Bus-Name-Policy">http://specifications.freedesktop.org/mpris-spec/latest/#Bus-Name-Policy</a></td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>scroll_chars</td>
<td>30</td>
<td>How many chars at once to display.</td>
</tr>
<tr>
<td>scroll_interval</td>
<td>0.5</td>
<td>Scroll delay interval.</td>
</tr>
<tr>
<td>scroll_wait_intervals</td>
<td>8</td>
<td>Wait x scroll_interval beforescrolling/removing text</td>
</tr>
<tr>
<td>stop_pause_text</td>
<td>None</td>
<td>Optional text to display when in the stopped/paused state</td>
</tr>
</tbody>
</table>

Net

class libqtile.widget.Net(**config)

Displays interface down and up speed

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'fffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>interface</td>
<td>'wlan0'</td>
<td>The interface to monitor</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td></td>
<td>The update interval.</td>
</tr>
</tbody>
</table>
NetGraph

class libqtile.widget.NetGraph(**config)

Display a network usage graph

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>bandwidth_type</td>
<td>'down'</td>
<td>down(load)/up(load)</td>
</tr>
<tr>
<td>border_color</td>
<td>'215578'</td>
<td>Widget border color</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Widget border width</td>
</tr>
<tr>
<td>fill_color</td>
<td>'1667EB.3'</td>
<td>Fill color for linefill graph</td>
</tr>
<tr>
<td>frequency</td>
<td>1</td>
<td>Update frequency in seconds</td>
</tr>
<tr>
<td>graph_color</td>
<td>'18BAEB'</td>
<td>Graph color</td>
</tr>
<tr>
<td>interface</td>
<td>'auto'</td>
<td>Interface to display info for ('auto' for detection)</td>
</tr>
<tr>
<td>line_width</td>
<td>3</td>
<td>Line width</td>
</tr>
<tr>
<td>margin_x</td>
<td>3</td>
<td>Margin X</td>
</tr>
<tr>
<td>margin_y</td>
<td>3</td>
<td>Margin Y</td>
</tr>
<tr>
<td>samples</td>
<td>100</td>
<td>Count of graph samples.</td>
</tr>
<tr>
<td>start_pos</td>
<td>'bottom'</td>
<td>Drawer starting position ('bottom'/top')</td>
</tr>
<tr>
<td>type</td>
<td>'linefill'</td>
<td>'box', 'line', 'linefill'</td>
</tr>
</tbody>
</table>

Notify

class libqtile.widget.Notify(width=CALCULATED, **config)

A notify widget

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audiofile</td>
<td>None</td>
<td>Audiofile played during notifications</td>
</tr>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>default_timeout</td>
<td>None</td>
<td>Default timeout (seconds) for notifications</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>foreground_low</td>
<td>'ddddd'</td>
<td>Foreground low priority colour</td>
</tr>
<tr>
<td>foreground_urgent</td>
<td>'fff0000'</td>
<td>Foreground urgent priority colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
</tbody>
</table>

Pacman

class libqtile.widget.Pacman(**config)

Shows number of available updates

Needs the pacman package manager installed. So will only work in Arch Linux installation.

Supported bar orientations: horizontal only
Pomodoro

```
class libqtile.widget.Pomodoro(**config)
```

Pomodoro technique widget

Supported bar orientations: horizontal only

```
key                  default     description
background           None        Widget background color
execute              None        Command to execute on click
font                 'sans'      Default font
fontshadow           None        Font shadow color, default is None(no shadow)
fontsize             None        Font size. Calculated if None.
foreground           'ffffff'    Foreground colour
markup               False       Whether or not to use pango markup
padding              None        Padding. Calculated if None.
unavailable          'ffffff'    Unavailable Color - no updates.
update_interval      160        The update interval.
length_pomodori      25         Length of one pomodori in minutes
length_short_break   5          Length of a short break in minutes
length_long_break    15         Length of a long break in minutes
prefix_active        ''         Prefix then app is active
prefix_break         'B '       Prefix during short break
prefix_inactive      'POMODORO'  Prefix when app is inactive
prefix_long_break    'LB '      Prefix during long break
prefix_paused        'PAUSE'    Prefix during pause
```

Prompt

```
class libqtile.widget.Prompt (name='prompt', **config)
```

A widget that prompts for user input

Input should be started using the .startInput() method on this class.

4.1. Reference
Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>bell_style</td>
<td>'audible'</td>
<td>Alert at the begin/end of the command history. Possible values: 'audible', 'visual' and None.</td>
</tr>
<tr>
<td>cursor</td>
<td>True</td>
<td>Show a cursor</td>
</tr>
<tr>
<td>cursor_color</td>
<td>'bef098'</td>
<td>Color for the cursor and text over it.</td>
</tr>
<tr>
<td>cursorblink</td>
<td>0.5</td>
<td>Cursor blink rate. 0 to disable.</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow).</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>ignore_dups_history</td>
<td>False</td>
<td>Don’t store duplicates in history</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>max_history</td>
<td>100</td>
<td>Commands to keep in history. 0 for no limit.</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>prompt</td>
<td>'{prompt}: '</td>
<td>Text displayed at the prompt</td>
</tr>
<tr>
<td>record_history</td>
<td>True</td>
<td>Keep a record of executed commands</td>
</tr>
<tr>
<td>visual_bell_color</td>
<td>'ff0000'</td>
<td>Color for the visual bell (changes prompt background).</td>
</tr>
<tr>
<td>visual_bell_time</td>
<td>0.2</td>
<td>Visual bell duration (in seconds).</td>
</tr>
</tbody>
</table>

**Sep**

```python
class libqtile.widget.Sep(height_percent=None, **config)
A visible widget separator
```

Supported bar orientations: horizontal and vertical

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>foreground</td>
<td>'888888'</td>
<td>Separator line colour.</td>
</tr>
<tr>
<td>linewidth</td>
<td>1</td>
<td>Width of separator line.</td>
</tr>
<tr>
<td>padding</td>
<td>2</td>
<td>Padding on either side of separator.</td>
</tr>
<tr>
<td>size_percent</td>
<td>80</td>
<td>Size as a percentage of bar size (0-100).</td>
</tr>
</tbody>
</table>

**She**

```python
class libqtile.widget.She(**config)
Widget to display the Super Hybrid Engine status
```

Can display either the mode or CPU speed on eeePC computers.

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>device</td>
<td>'/sys/</td>
<td>sys path to cpufv</td>
</tr>
<tr>
<td></td>
<td>devices/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>platform/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eepc/cpufv'</td>
<td></td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'speed'</td>
<td>Type of info to display “speed” or “name”</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td>10.5</td>
<td>Update Time in seconds.</td>
</tr>
</tbody>
</table>

**Spacer**

class libqtile.widget.Spacer(length=STRETCH, width=None)

Just an empty space on the bar

Often used with length equal to bar.STRETCH to push bar widgets to the right or bottom edge of the screen.

**Parameters**

- **length**: Length of the widget. Can be either bar.STRETCH or a length in pixels.
- **width**: DEPRECATED, same as length.

Supported bar orientations: horizontal and vertical

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
</tbody>
</table>

**StockTicker**

class libqtile.widget.StockTicker(**config**)

A stock ticker widget, based on the alphavantage API. Users must acquire an API key from https://www.alphavantage.co/support/#api-key

The widget defaults to the TIME_SERIES_INTRADAY API function (i.e. stock symbols), but arbitrary Alpha Vantage API queries can be made by passing extra arguments to the constructor.

```
# Display AMZN
widget.StockTicker(apikey=..., symbol="AMZN")

# Display BTC
widget.StockTicker(apikey=..., function="DIGITAL_CURRENCY_INTRADAY", symbol="BTC", market="USD")
```

Supported bar orientations: horizontal only
### SwapGraph

**class** `libqtile.widget.SwapGraph(**config)`

Display a swap info graph

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>border_color</td>
<td>'215578'</td>
<td>Widget border color</td>
</tr>
<tr>
<td>border_width</td>
<td>2</td>
<td>Widget border width</td>
</tr>
<tr>
<td>fill_color</td>
<td>'1667EB.3'</td>
<td>Fill color for linefill graph</td>
</tr>
<tr>
<td>frequency</td>
<td>1</td>
<td>Update frequency in seconds</td>
</tr>
<tr>
<td>graph_color</td>
<td>'18BAEB'</td>
<td>Graph color</td>
</tr>
<tr>
<td>line_width</td>
<td>3</td>
<td>Line width</td>
</tr>
<tr>
<td>margin_x</td>
<td>3</td>
<td>Margin X</td>
</tr>
<tr>
<td>margin_y</td>
<td>3</td>
<td>Margin Y</td>
</tr>
<tr>
<td>samples</td>
<td>100</td>
<td>Count of graph samples.</td>
</tr>
<tr>
<td>start_pos</td>
<td>'bottom'</td>
<td>Drawer starting position ('bottom'/'top')</td>
</tr>
<tr>
<td>type</td>
<td>'linefill'</td>
<td>'box', 'line', 'linefill'</td>
</tr>
</tbody>
</table>

### Systray

**class** `libqtile.widget.Systray(**config)`

A widget that manages system tray

Supported bar orientations: horizontal only
### TaskList

**class** `libqtile.widget.TaskList(**config)`

Displays the icon and name of each window in the current group

Contrary to WindowTabs this is an interactive widget. The window that currently has focus is highlighted.

Supported bar orientations: horizontal only
| key                | default       | description                                                      |
|--------------------|---------------|                                                                |
| background         | None          | Widget background color                                         |
| border             | '215578'      | Border colour                                                    |
| borderwidth        | 2             | Current group border width                                       |
| font               | 'sans'        | Default font                                                     |
| fontshadow         | None          | Font shadow color, default is None (no shadow)                  |
| fontsize           | None          | Font size. Calculated if None.                                   |
| foreground         | 'ffffff'      | Foreground colour                                                |
| highlight_method   | 'border'      | Method of highlighting (one of ‘border’ or ‘block’) Uses *_bor-
|                    |               | der color settings                                               |
| icon_size          | None          | Icon size. (Calculated if set to None. Icons are hidden if set to
|                    |               | 0.)                                                             |
| markup_floating    | None          | Text markup of the floating window state. Supports pangomarkup
|                    |               | with markup=True.e.g., "{}" or "<span underline="low">{}<\span>" |
| markup_focused     | None          | Text markup of the focused window state. Supports pangomarkup
|                    |               | with markup=True.e.g., "{}" or "<span underline="low">{}<\span>" |
| markup_maximized   | None          | Text markup of the maximized window state. Supports pangomarkup
|                    |               | with markup=True.e.g., "{}" or "<span underline="low">{}<\span>" |
| markup_minimized   | None          | Text markup of the minimized window state. Supports pangomarkup
|                    |               | with markup=True.e.g., "{}" or "<span underline="low">{}<\span>" |
| markup_normal      | None          | Text markup of the normal window state. Supports pangomarkup
|                    |               | with markup=True.e.g., "{}" or "<span underline="low">{}<\span>" |
| max_title_width    | None          | Max size in pixels of task title.(if set to None, as much as avai-
|                    |               | lable.)                                                         |
| rounded            | True          | To round or not to round borders                                 |
| spacing            | None          | Spacing between tasks.(if set to None, will be equal to mar-
|                    |               | gin_x)                                                          |
| title_width_method | None          | Method to compute the width of task title. (None, 'uni-
|                    |               | form'.)Defaults to None, the normal behaviour.                   |
| txt_floating       | 'V '          | Text representation of the floating window state. e.g., "V " or "  |
|                    |               | "                                                 |
| txt_maximized      | '[] '         | Text representation of the maximized window state. e.g., "[ ]" or "  |
|                    |               | "                                                 |
| txt_minimized      | '_ '          | Text representation of the minimized window state. e.g., "_ " or "  |
|                    |               | "                                                 |
| unfocused_border   | None          | Border color for unfocused windows. Affects only highlight_method
|                    |               | ‘border’ and ‘block’. Defaults to None, which means no special color. |
| urgent_alert_method| 'border'      | Method for alerting you of WM urgent hints (one of ‘border’ or
|                    |               | ‘text’)                                                          |
| urgent_border      | 'FF0000'      | Urgent border color                                              |

**TextBox**

```python
class libqtile.widget.TextBox(text=' ', width=CALCULATED, **config)
```

A flexible textbox that can be updated from bound keys, scripts, and qshell
Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Text font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font pixel size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'#ffffff'</td>
<td>Foreground colour.</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding left and right. Calculated if None.</td>
</tr>
</tbody>
</table>

**ThermalSensor**

`class libqtile.widget.ThermalSensor(**config)`

Widget to display temperature sensor information

For using the thermal sensor widget you need to have lm-sensors installed. You can get a list of the tag_sensors executing “sensors” in your terminal. Then you can choose which you want, otherwise it will display the first available.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'#ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>foreground_alert</td>
<td>'#ff0000'</td>
<td>Foreground colour alert</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>metric</td>
<td>True</td>
<td>True to use metric/C, False to use imperial/F</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>show_tag</td>
<td>False</td>
<td>Show tag sensor</td>
</tr>
<tr>
<td>tag_sensor</td>
<td>None</td>
<td>Tag of the temperature sensor. For example: “temp1” or “Core 0”</td>
</tr>
<tr>
<td>threshold</td>
<td>70</td>
<td>If the current temperature value is above, then change to foreground_alert colour</td>
</tr>
<tr>
<td>update_interval</td>
<td>12</td>
<td>Update interval in seconds</td>
</tr>
</tbody>
</table>

**Volume**

`class libqtile.widget.Volume(**config)`

Widget that display and change volume

If theme_path is set it draw widget as icons.

Supported bar orientations: horizontal only
### Wallpaper

**class** `libqtile.widget.Wallpaper(**config)`

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>directory</td>
<td>'~/Pictures/wallpapers/'</td>
<td>Wallpaper Directory</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>label</td>
<td>None</td>
<td>Use a fixed label instead of image name.</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>one_screen</td>
<td>False</td>
<td>Treat the whole X display as one screen when setting wallpapers (does not work if wallpaper_command is set).</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>random_selection</td>
<td>False</td>
<td>If set, use random initial wallpaper and randomly cycle through the wallpapers.</td>
</tr>
<tr>
<td>wallpaper</td>
<td>None</td>
<td>Wallpaper</td>
</tr>
<tr>
<td>wallpaper_command</td>
<td>None</td>
<td>Wallpaper command</td>
</tr>
</tbody>
</table>

### Table of Key Values

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>cardid</td>
<td>None</td>
<td>Card Id</td>
</tr>
<tr>
<td>channel</td>
<td>'Master'</td>
<td>Channel</td>
</tr>
<tr>
<td>device</td>
<td>'default'</td>
<td>Device Name</td>
</tr>
<tr>
<td>emoji</td>
<td>False</td>
<td>Use emoji to display volume states, only if theme_path is not set. The specified font needs to contain the correct unicode characters.</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>get_volume_command</td>
<td>None</td>
<td>Command to get the current volume</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>mute_command</td>
<td>None</td>
<td>Mute command</td>
</tr>
<tr>
<td>padding</td>
<td>3</td>
<td>Padding left and right. Calculated if None.</td>
</tr>
<tr>
<td>step</td>
<td>2</td>
<td>Volume change for up an down commands in percentage. Only used if volume_up_command and volume_down_command are not set.</td>
</tr>
<tr>
<td>theme_path</td>
<td>None</td>
<td>Path of the icons</td>
</tr>
<tr>
<td>update_interval</td>
<td>0.2</td>
<td>Update time in seconds.</td>
</tr>
<tr>
<td>volume_app</td>
<td>None</td>
<td>App to control volume</td>
</tr>
<tr>
<td>volume_down_command</td>
<td>None</td>
<td>Volume down command</td>
</tr>
<tr>
<td>volume_up_command</td>
<td>None</td>
<td>Volume up command</td>
</tr>
</tbody>
</table>

92 Chapter 4. Miscellaneous
WindowName

```python
class libqtile.widget.WindowName (width=STRETCH, **config)
```

Displays the name of the window that currently has focus

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>for_current_screen</td>
<td>False</td>
<td>instead of this bars screen use currently active screen</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>show_state</td>
<td>True</td>
<td>show window status before window name</td>
</tr>
</tbody>
</table>

WindowTabs

```python
class libqtile.widget.WindowTabs (**config)
```

Displays the name of each window in the current group. Contrary to TaskList this is not an interactive widget. The window that currently has focus is highlighted.

Supported bar orientations: horizontal only

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>selected</td>
<td>('&lt;', '&gt;')</td>
<td>Selected task indicator</td>
</tr>
<tr>
<td>separator</td>
<td>'</td>
<td>'</td>
</tr>
</tbody>
</table>

Wlan

```python
class libqtile.widget.Wlan (**config)
```

Displays Wifi ssid and quality

Supported bar orientations: horizontal only
<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>Widget background color</td>
</tr>
<tr>
<td>disconnected_message</td>
<td>'Disconnected'</td>
<td>String to show when the wlan is disconnected.</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>Default font</td>
</tr>
<tr>
<td>fontshadow</td>
<td>None</td>
<td>Font shadow color, default is None(no shadow)</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>Font size. Calculated if None.</td>
</tr>
<tr>
<td>foreground</td>
<td>'ffffff'</td>
<td>Foreground colour</td>
</tr>
<tr>
<td>format</td>
<td>'{essid} (quality)/70'</td>
<td>Display format. For percents you can use &quot;{essid} {percent:2.0%}&quot;</td>
</tr>
<tr>
<td>interface</td>
<td>'wlan0'</td>
<td>The interface to monitor</td>
</tr>
<tr>
<td>markup</td>
<td>False</td>
<td>Whether or not to use pango markup</td>
</tr>
<tr>
<td>padding</td>
<td>None</td>
<td>Padding. Calculated if None.</td>
</tr>
<tr>
<td>update_interval</td>
<td></td>
<td>The update interval.</td>
</tr>
</tbody>
</table>

**YahooWeather**

class libqtile.widget.YahooWeather(**config**)  
A weather widget, data provided by the Yahoo! Weather API.

Format options:

- astronomy_sunrise
- astronomy_sunset
- atmosphere_humidity
- atmosphere_visibility
- atmosphere_pressure
- atmosphere_rising
- condition_text
- condition_code
- condition_temp
- condition_date
- location_city
- location_region
- location_country
- units_temperature
- units_distance
- units_pressure
- units_speed
- wind_chill

Supported bar orientations: horizontal only
### 4.1.5 Built-in Extensions

**Dmenu**

```python
class libqtile.extension.Dmenu(**config)
```


<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>defines the normal background color</td>
</tr>
<tr>
<td>command</td>
<td>None</td>
<td>the command to be launched (string or list with arguments)</td>
</tr>
<tr>
<td>dmenu_bottom</td>
<td>False</td>
<td>dmenu appears at the bottom of the screen</td>
</tr>
<tr>
<td>dmenu_command</td>
<td>'dmenu'</td>
<td>the dmenu command to be launched</td>
</tr>
<tr>
<td>dmenu_font</td>
<td>None</td>
<td>override the default 'font' and 'fontsize' options for dmenu</td>
</tr>
<tr>
<td>dmenu_height</td>
<td>None</td>
<td>defines the height (only supported by some dmenu forks)</td>
</tr>
<tr>
<td>dmenu_ignorecase</td>
<td>False</td>
<td>dmenu matches menu items case insensitively</td>
</tr>
<tr>
<td>dmenu_lines</td>
<td>None</td>
<td>dmenu lists items vertically, with the given number of lines</td>
</tr>
<tr>
<td>dmenu_prompt</td>
<td>None</td>
<td>defines the prompt to be displayed to the left of the input field</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>defines the font name to be used</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>defines the font size to be used</td>
</tr>
<tr>
<td>foreground</td>
<td>None</td>
<td>defines the normal foreground color</td>
</tr>
<tr>
<td>selected_background</td>
<td>'default'</td>
<td>defines the selected background color</td>
</tr>
<tr>
<td>selected_foreground</td>
<td>'default'</td>
<td>defines the selected foreground color</td>
</tr>
</tbody>
</table>
DmenuRun

class libqtile.extension.DmenuRun(**config)

Special case to run applications.

config.py should have something like:

```python
from libqtile import extension
keys = [
    Key(['mod4'], 'r', lazy.run_extension(extension.DmenuRun(
        dmenu_prompt=">",
        dmenu_font="Andika-8",
        background="#15181a",
        foreground="#00ff00",
        selected_background="#079822",
        selected_foreground="#fff",
        dmenu_height=24,  # Only supported by some dmenu forks
    )),
]
```

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>defines the normal background color</td>
</tr>
<tr>
<td>command</td>
<td>None</td>
<td>the command to be launched (string or list with arguments)</td>
</tr>
<tr>
<td>dmenu_bottom</td>
<td>False</td>
<td>dmenu appears at the bottom of the screen</td>
</tr>
<tr>
<td>dmenu_command</td>
<td>'dmenu_run'</td>
<td>the dmenu command to be launched</td>
</tr>
<tr>
<td>dmenu_font</td>
<td>None</td>
<td>override the default ‘font’ and ‘fontsize’ options for dmenu</td>
</tr>
<tr>
<td>dmenu_height</td>
<td>None</td>
<td>defines the height (only supported by some dmenu forks)</td>
</tr>
<tr>
<td>dmenu_ignorecase</td>
<td>False</td>
<td>dmenu matches menu items case insensitively</td>
</tr>
<tr>
<td>dmenu_lines</td>
<td>None</td>
<td>dmenu lists items vertically, with the given number of lines</td>
</tr>
<tr>
<td>dmenu_prompt</td>
<td>None</td>
<td>defines the prompt to be displayed to the left of the input field</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>defines the font name to be used</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>defines the font size to be used</td>
</tr>
<tr>
<td>foreground</td>
<td>None</td>
<td>defines the normal foreground color</td>
</tr>
<tr>
<td>selected_background</td>
<td>'baned'</td>
<td>defines the selected background color</td>
</tr>
<tr>
<td>selected_foreground</td>
<td>'bane'</td>
<td>defines the selected foreground color</td>
</tr>
</tbody>
</table>

J4DmenuDesktop

class libqtile.extension.J4DmenuDesktop(**config)

Python wrapper for j4-dmenu-desktop https://github.com/enkore/j4-dmenu-desktop
### RunCommand

**class** `libqtile.extension.RunCommand(**config)`

Run an arbitrary command.

Mostly useful as a superclass for more specific extensions that need to interact with the qtile object.

Also consider simply using `lazy.spawn()` or writing a client.

<table>
<thead>
<tr>
<th>key</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>None</td>
<td>defines the normal background color</td>
</tr>
<tr>
<td>command</td>
<td>None</td>
<td>the command to be launched (string or list with arguments)</td>
</tr>
<tr>
<td>dmenu_bottom</td>
<td>False</td>
<td>dmenu appears at the bottom of the screen</td>
</tr>
<tr>
<td>dmenu_command</td>
<td>'dmenu'</td>
<td>the dmenu command to be launched</td>
</tr>
<tr>
<td>dmenu_font</td>
<td>None</td>
<td>override the default ‘font’ and ‘fontsize’ options for dmenu</td>
</tr>
<tr>
<td>dmenu_height</td>
<td>None</td>
<td>defines the height (only supported by some dmenu forks)</td>
</tr>
<tr>
<td>dmenu_ignorecase</td>
<td>False</td>
<td>dmenu matches menu items case insensitively</td>
</tr>
<tr>
<td>dmenu_lines</td>
<td>None</td>
<td>dmenu lists items vertically, with the given number of lines</td>
</tr>
<tr>
<td>dmenu_prompt</td>
<td>None</td>
<td>defines the prompt to be displayed to the left of the input field</td>
</tr>
<tr>
<td>font</td>
<td>'sans'</td>
<td>defines the font name to be used</td>
</tr>
<tr>
<td>fontsize</td>
<td>None</td>
<td>defines the font size to be used</td>
</tr>
<tr>
<td>foreground</td>
<td>None</td>
<td>defines the normal foreground color</td>
</tr>
<tr>
<td>j4dmenu_command</td>
<td>'j4-dmenu-desktop'</td>
<td>the dmenu command to be launched</td>
</tr>
<tr>
<td>j4dmenu_display_binary</td>
<td>True</td>
<td>display binary name after each entry</td>
</tr>
<tr>
<td>j4dmenu_generic</td>
<td>False</td>
<td>include the generic name of desktop entries</td>
</tr>
<tr>
<td>j4dmenu_terminal</td>
<td>None</td>
<td>terminal emulator used to start terminal apps</td>
</tr>
<tr>
<td>j4dmenu_usage_log</td>
<td>True</td>
<td>read $XDG_CURRENT_DESKTOP to determine the desktop environment</td>
</tr>
<tr>
<td>selected_background</td>
<td>None</td>
<td>defines the selected background color</td>
</tr>
<tr>
<td>selected_foreground</td>
<td>None</td>
<td>defines the selected foreground color</td>
</tr>
</tbody>
</table>

### WindowList

**class** `libqtile.extension.WindowList(**config)`

Give vertical list of all open windows in dmenu. Switch to selected.
### Frequently Asked Questions

#### 4.2.1 Why the name Qtile?

Users often wonder, why the Q? Does it have something to do with Qt? No. Below is an IRC excerpt where cortesi explains the great trial that ultimately brought Qtile into existence, thanks to the benevolence of the Open Source Gods. Praise be to the OSG!

ramnes: what does Qtile mean?
ramnes: what's the Q?
@tych0: ramnes: it doesn't :)
@tych0: cortesi was just looking for the first letter that wasn't registered in a domain name with "tile" as a suffix
@tych0: qtile it was :)
cortesi: tych0, dx: we really should have something more compelling to explain the name. one day i was swimming at manly beach in sydney, where i lived at the time. suddenly, i saw an enormous great white right beside me. it went for my leg with massive, gaping jaws, but quick as a flash, i thumb-punched it in both eyes. when it reared back in agony, i saw that it had a jagged, gnarly scar on its stomach... a scar shaped like the letter "Q".
cortesi: while it was distracted, i surfed a wave to shore. i knew that i had to dedicate my next open source project to the ocean gods, in thanks for my lucky escape. and thus, qtile got its name...

#### 4.2.2 When I first start xterm/urxvt/rxvt containing an instance of Vim, I see text and layout corruption. What gives?

Vim is not handling terminal resizes correctly. You can fix the problem by starting your xterm with the “-wf” option, like so:

```bash
vim -wf /path/to/vim
```
Alternatively, you can just cycle through your layouts a few times, which usually seems to fix it.

### 4.2.3 How do I know which modifier specification maps to which key?

To see a list of modifier names and their matching keys, use the `xmodmap` command. On my system, the output looks like this:

```
$ xmodmap
xmodmap: up to 3 keys per modifier, (keycodes in parentheses):

    shift        Shift_L (0x32), Shift_R (0x3e)
    lock         Caps_Lock (0x9)
    control      Control_L (0x25), Control_R (0x69)
    mod1         Alt_L (0x40), Alt_R (0x6c), Meta_L (0xcd)
    mod2         Num_Lock (0x4d)
    mod3         Super_L (0xce), Hyper_L (0xcf)
    mod4         ISO_Level3_Shift (0x5c), Mode_switch (0xcb)
```

### 4.2.4 My “pointer mouse cursor” isn’t the one I expect it to be!

Qtile should set the default cursor to left_ptr, you must install xcb-util-cursor if you want support for themed cursors.

### 4.2.5 LibreOffice menus don’t appear or don’t stay visible

A workaround for problem with the mouse in libreoffice is setting the environment variable `SAL_USE_VCLPLUGIN=gen`. It is dependet on your system configuration where to do this. e.g. Arch-Linux with libreoffice-fresh in `/etc/profile.d/libreoffice-fresh.sh`.

### 4.3 License

This project is distributed under the MIT license.

Copyright (c) 2008, Aldo Cortesi All rights reserved.

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

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

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

A
addgroup() (libqtile.hook.subscribe class method), 48
AGroupBox (class in libqtile.widget), 62

B
Backlight (class in libqtile.widget), 63
Bar (class in libqtile.bar), 15
BitcoinTicker (class in libqtile.widget), 63
Bsp (class in libqtile.layout.bsp), 52

C
Canto (class in libqtile.widget), 64
CapsNumLockIndicator (class in libqtile.widget), 65
changegroup() (libqtile.hook.subscribe class method), 48
CheckUpdates (class in libqtile.widget), 65
Click (class in libqtile.config), 14
Client (class in libqtile.command), 24
client_focus() (libqtile.hook.subscribe class method), 48
client_killed() (libqtile.hook.subscribe class method), 48
client_managed() (libqtile.hook.subscribe class method), 48
client_mouse_enter() (libqtile.hook.subscribe class method), 48
client_name_updated() (libqtile.hook.subscribe class method), 49
client_new() (libqtile.hook.subscribe class method), 49
client_state_changed() (libqtile.hook.subscribe class method), 49
client_type_changed() (libqtile.hook.subscribe class method), 49
client_urgent_hint_changed() (libqtile.hook.subscribe class method), 49
Clipboard (class in libqtile.widget), 66
Clock (class in libqtile.widget), 66
cmd_add_rule() (libqtile.manager.Qtile method), 39
cmd_addgroup() (libqtile.manager.Qtile method), 39
cmd_bring_to_front() (libqtile.window.Window method), 46
cmd_commands() (libqtile.config.Screen method), 45
cmd_commands() (libqtile.manager.Qtile method), 39
cmd_commands() (libqtile.window.Window method), 46
cmd_critical() (libqtile.manager.Qtile method), 39
cmd_debug() (libqtile.manager.Qtile method), 39
cmd_delgroup() (libqtile.manager.Qtile method), 40
cmd_disable_floating() (libqtile.window.Window method), 46
cmd_disable_fullscreen() (libqtile.window.Window method), 46
cmd_disable_maximize() (libqtile.window.Window method), 46
cmd_disable_minimize() (libqtile.window.Window method), 46
cmd_display_kb() (libqtile.manager.Qtile method), 40
cmd_doc() (libqtile.bar.Bar method), 44
cmd_doc() (libqtile.config.Screen method), 45
cmd_doc() (libqtile.manager.Qtile method), 40
cmd_doc() (libqtile.window.Window method), 46
cmd_down_opacity() (libqtile.window.Window method), 46
cmd_enable_floating() (libqtile.window.Window method), 46
cmd_enable_fullscreen() (libqtile.window.Window method), 46
cmd_enable_maximize() (libqtile.window.Window method), 46
cmd_enable_minimize() (libqtile.window.Window method), 46
cmd_error() (libqtile.manager.Qtile method), 40
cmd_eval() (libqtile.bar.Bar method), 44
cmd_eval() (libqtile.config.Screen method), 45
cmd_eval() (libqtile.manager.Qtile method), 40
cmd_eval() (libqtile.window.Window method), 46
cmd_fake_button_press() (libqtile.bar.Bar method), 44
cmd_findwindow() (libqtile.manager.Qtile method), 40
cmd_focus() (libqtile.window.Window method), 46
cmd_focus_by_click() (libqtile.manager.Qtile method), 40
cmd_function() (libqtile.bar.Bar method), 44
do_ls() (libqtile.sh.QSh method), 26
do_pwd() (libqtile.sh.QSh method), 26
Drag (class in libqtile.config), 14
DropDown (class in libqtile.config), 10
E
EzConfig (class in libqtile.config), 13
F
float_change() (libqtile.hook.subscribe class method), 50
Floating (class in libqtile.layout.floating), 51
focus_change() (libqtile.hook.subscribe class method), 50
G
Gap (class in libqtile.bar), 16
GenPollText (class in libqtile.widget), 70
GenPollUrl (class in libqtile.widget), 71
GmailChecker (class in libqtile.widget), 71
Group (class in libqtile.config), 8
group_window_add() (libqtile.hook.subscribe class method), 50
GroupBox (class in libqtile.widget), 72
H
HDDBusyGraph (class in libqtile.widget), 73
HDDGraph (class in libqtile.widget), 74
I
IdleRPG (class in libqtile.widget), 74
Image (in module libqtile.widget), 75
ImapWidget (class in libqtile.widget), 75
J
J4DmenuDesktop (class in libqtile.extension), 96
K
Key (class in libqtile.config), 12
KeyboardKbdd (class in libqtile.widget), 76
KeyboardLayout (class in libqtile.widget), 76
KhalCalendar (class in libqtile.widget), 77
L
LaunchBar (class in libqtile.widget), 77
layout_change() (libqtile.hook.subscribe class method), 50
M
Maildir (class in libqtile.widget), 78
Match (class in libqtile.config), 9
Matrix (class in libqtile.layout.matrix), 54
Max (class in libqtile.layout.max), 54
Memory (class in libqtile.widget), 78
MemoryGraph (class in libqtile.widget), 79
Moc (class in libqtile.widget), 79
MonadTall (class in libqtile.layout.xmonad), 54
MonadWide (class in libqtile.layout.xmonad), 56
Mpd (class in libqtile.widget), 80
Mpd2 (class in libqtile.widget), 81
Mpris (class in libqtile.widget), 82
Mpris2 (class in libqtile.widget), 83
N
Net (class in libqtile.widget), 83
net_wm_icon_change() (libqtile.hook.subscribe class method), 50
NetGraph (class in libqtile.widget), 84
Notify (class in libqtile.widget), 84
P
Pacman (class in libqtile.widget), 84
Pomodoro (class in libqtile.widget), 85
Prompt (class in libqtile.widget), 85
Q
QSh (class in libqtile.sh), 25
Qtile (class in libqtile.manager), 39
R
RatioTile (class in libqtile.layout.ratiotile), 58
Rule (class in libqtile.config), 9
RunCommand (class in libqtile.extension), 97
S
ScratchPad (class in libqtile.config), 10
Screen (class in libqtile.config), 15
screen_change() (libqtile.hook.subscribe class method), 50
selection_change() (libqtile.hook.subscribe class method), 50
selection_notify() (libqtile.hook.subscribe class method), 50
Sep (class in libqtile.widget), 86
setgroup() (libqtile.hook.subscribe class method), 51
She (class in libqtile.widget), 86
simple_key_binder() (in module libqtile.dgroups), 8
Slice (class in libqtile.layout.slice), 58
Spacer (class in libqtile.widget), 87
Stack (class in libqtile.layout.stack), 59
startup() (libqtile.hook.subscribe class method), 51
startup_complete() (libqtile.hook.subscribe class method), 51
startup_once() (libqtile.hook.subscribe class method), 51
StockTicker (class in libqtile.widget), 87
SwapGraph (class in libqtile.widget), 88
Systray (class in libqtile.widget), 88
TaskList (class in libqtile.widget), 89
TextBox (class in libqtile.widget), 90
ThermalSensor (class in libqtile.widget), 91
Tile (class in libqtile.layout.tile), 59
TreeTab (class in libqtile.layout.tree), 60
VerticalTile (class in libqtile.layout.verticaltile), 60
Volume (class in libqtile.widget), 91
Wallpaper (class in libqtile.widget), 92
Window (class in libqtile.window), 46
window_name_change() (libqtile.hook.subscribe class method), 51
WindowList (class in libqtile.extension), 97
WindowName (class in libqtile.widget), 93
WindowTabs (class in libqtile.widget), 93
Wlan (class in libqtile.widget), 93
Wmii (class in libqtile.layout.wmii), 61
YahooWeather (class in libqtile.widget), 94
Zoomy (class in libqtile.layout.zoomy), 62