NAME

sway - An i3-compatible Wayland compositor

SYNOPSIS

sway [options...] [command]

OPTIONS

-h, --help
Show help message and quit.

-c, --config <config>
Specifies a config file.

-C, --validate
Check the validity of the config file, then exit.

-d, --debug
Enables full logging, including debug information.

-v, --version
Show the version number and quit.

-V, --verbose
Enables more verbose logging.

--get-socketpath
Gets the IPC socket path and prints it, then exits.

DESCRIPTION

sway was created to fill the need of an i3-like window manager for Wayland. The upstream i3 developers have no intention of porting i3 to Wayland, and projects proposed by others ended up as vaporware. Many thanks to the i3 folks for providing such a great piece of software, so good that your users would rather write an entirely new window manager from scratch that behaved *exactly* like i3 rather than switch to something else.

You can run sway directly from a tty, or via a Wayland-compatible login manager.

CONFIGURATION

sway searches for a config file in the following locations, in this order:

1. ˜/.sway/config
2. $XDG_CONFIG_HOME/sway/config (suggested location)
3. ˜/.i3/config
4. $XDG_CONFIG_HOME/i3/config
5. /etc/sway/config
6. /etc/i3/config

If unset, $XDG_CONFIG_HOME defaults to **˜/.config**.

An error is raised when no config file is found. The recommended default configuration is usually installed to **/etc/sway/config**; you are encouraged to copy this to **˜/.config/sway/config** and edit it from there.

For information on the config file format, see **sway**(5).

IPC COMMANDS

Though **swaymsg**(1) is generally preferred, you may run **sway** *command* to send *command* to the running instance of sway. You can also issue commands with **i3-msg**(1) or even with **i3**(1).

ENVIRONMENT

The following environment variables have an effect on sway:

SWAYSOCK

Specifies the path to the sway IPC socket.

XKB_DEFAULT_RULES, *XKB_DEFAULT_MODEL*, *XKB_DEFAULT_LAYOUT*, *XKB_DEFAULT_VARIANT*, *XKB_DEFAULT_OPTIONS*
: Configures the xkb keyboard settings. See **xkeyboard-config**(7). The preferred way to configure the keyboard is via the configuration file, see **sway-input**(5).

The following environment variables are set by sway:

DISPLAY
: If compiled with Xwayland support and Xwayland is not disabled by the config, this will be set to the name of the X display used for Xwayland.

I3SOCK
: For compatibility with i3, specifies the path to the sway IPC socket.

SWAYSOCK
: Specifies the path to the sway IPC socket.

WAYLAND_DISPLAY
: Specifies the name of the Wayland display that sway is running on.

XCURSOR_SIZE
: Specifies the configured cursor size.

XCURSOR_THEME
: Specifies the configured cursor theme.

AUTHORS

Maintained by Simon Ser <contact@emersion.fr>, who is assisted by other open source contributors. For more information about sway development, see <https://github.com/swaywm/sway>.

SEE ALSO

sway(5) **swaymsg**(1) **sway-input**(5) **sway-output**(5) **sway-bar**(5) **sway-ipc**(7)

NAME

swaymsg - Send messages to a running instance of sway over the IPC socket.

SYNOPSIS

swaymsg [options...] [message]

OPTIONS

-h, --help
Show help message and quit.

-m, --monitor
Monitor for responses until killed instead of exiting after the first response. This can only be used with the IPC message type *subscribe*. If there is a malformed response or an invalid event type was requested, swaymsg will stop monitoring and exit.

-p, --pretty
Use pretty output even when not using a tty. Not available for all message types.

-q, --quiet
Sends the IPC message but does not print the response from sway.

-r, --raw
Use raw JSON output even if using a tty.

-s, --socket <path>
Use the specified socket path. Otherwise, swaymsg will ask sway where the socket is (which is the value of $SWAYSOCK, then of $I3SOCK).

-t, --type <type>
Specify the type of IPC message. See below.

-v, --version
Print the version (of swaymsg) and quit.

IPC MESSAGE TYPES

<command>
The message is a sway command (the same commands you can bind to keybindings in your sway config file). It will be executed immediately.

See **sway**(5) for a list of commands.

swaymsg can return pretty printed (standalone-default) or JSON-formatted (**--raw**) output. For detailed documentation on the returned JSON-data of each message type listed below, refer to **sway-ipc**(7). The JSON-format can contain more information than the pretty print.

Tips:

- Command expansion is performed twice: once by swaymsg, and again by sway. If you have quoted multi-word strings in your command, enclose the entire command in single-quotes. For example, use *swaymsg 'output "Foobar Display" enable'* instead of *swaymsg output "Foobar Display" enable*. Furthermore, note that comma separated options also count as multi-word strings, because commas can be used to execute commands on the same line.
- If you are providing a command that contains a leading hyphen (-), insert two hyphens (--) before the command to signal to swaymsg not to parse anything beyond that point as an option. For example, use *swaymsg -- mark --add test* instead of *swaymsg mark --add test*.

get_workspaces
Gets a list of workspaces and their status.

get_inputs
Gets a list of current inputs.

get_outputs

Gets a list of current outputs.

get_tree
Gets a JSON-encoded layout tree of all open windows, containers, outputs, workspaces, and so on.

get_seats
Gets a list of all seats, its properties and all assigned devices.

get_marks
Get a JSON-encoded list of marks.

get_bar_config
Get a JSON-encoded configuration for swaybar.

get_version
Get version information for the running instance of sway.

get_binding_modes
Gets a JSON-encoded list of currently configured binding modes.

get_binding_state
Gets JSON-encoded info about the current binding state.

get_config
Gets a copy of the current configuration. Doesn’t expand includes.

send_tick
Sends a tick event to all subscribed clients.

subscribe
Subscribe to a list of event types. The argument for this type should be provided in the form of a valid JSON array. If any of the types are invalid or if a valid JSON array is not provided, this will result in a failure.

RETURN CODES

0
Success

1
swaymsg errors such as invalid syntax, unable to connect to the ipc socket or unable to parse sway’s reply

2
Sway returned an error when processing the command (ex. invalid command, command failed, and invalid subscription request)

SEE ALSO

sway(5) **sway-bar**(5) **sway-input**(5) **sway-output**(5) **sway-ipc**(7)

NAME

swaynag - Show a warning or error message with buttons

SYNOPSIS

swaynag [options...]

OPTIONS

-b, --button <text> <action>
: Create a button with the text *text* that executes *action* when pressed. If the environment variable *TERMINAL* is set, *action* will be run inside the terminal. Otherwise, it will fallback to running directly. Multiple buttons can be defined by providing the flag multiple times.

-B, --button-no-terminal <text> <action>
: Create a button with the text *text* that executes *action* when pressed. *action* will be run directly instead of in a terminal. Multiple buttons can be defined by providing the flag multiple times.

-z, --button-dismiss <text> <action>
: Create a button with the text *text* that executes *action* when pressed, and dismisses swaynag. If the environment variable *TERMINAL* is set, *action* will be run inside the terminal. Otherwise, it will fallback to running directly. Multiple buttons can be defined by providing the flag multiple times.

-Z, --button-dismiss-no-terminal <text> <action>
: Create a button with the text *text* that executes *action* when pressed, and dismisses swaynag. *action* will be run directly instead of in a terminal. Multiple buttons can be defined by providing the flag multiple times.

-c, --config <path>
: The config file to use. By default, the following paths are checked: *$HOME/.swaynag/config*, *$XDG_CONFIG_HOME/swaynag/config*, and *SYSCONFDIR/swaynag/config*. All flags aside from this one and *debug* are valid options in the configuration file using the format *long-option=value*. All leading dashes should be omitted and the equals sign is required. See swaynag(5) for more information.

-d, --debug
: Enable debugging.

-e, --edge top|bottom
: Set the edge to use.

-y, --layer overlay|top|bottom|background
: Set the layer to use.

-f, --font <font>
: Set the font to use.

-h, --help
: Show help message and quit.

-l, --detailed-message
: Read a detailed message from stdin. A button to toggle details will be added. Details are shown in a scrollable multi-line text area.

-L, --detailed-button <text>
: Set the text for the button that toggles details. This has no effect if there is not a detailed message. The default is *Toggle details*.

-m, --message <msg>
: Set the message text.

-o, --output <output>
: Set the output to use. This should be the name of a *xdg_output*.

-s, --dismiss-button <text>
: Sets the text for the dismiss nagbar button. The default is *X*.

-t, --type <type>
Set the message type. Two types are created by default *error* and *warning*. Custom types can be defined in the config file. See *--config* and swaynag(5) for details. Both of the default types can be overridden in the config file as well.

-v, --version
Show the version number and quit.

APPEARANCE OPTIONS

--background <RRGGBB[AA]>
Set the color of the background.

--border <RRGGBB[AA]>
Set the color of the border.

--border-bottom <RRGGBB[AA]>
Set the color of the bottom border.

--button-background <RRGGBB[AA]>
Set the color for the background for buttons.

--text <RRGGBB[AA]>
Set the text color.

--button-text <RRGGBB[AA]>
Set the button text color.

--border-bottom-size <size>
Set the thickness of the bottom border.

--message-padding <padding>
Set the padding for the message.

--details-background <RRGGBB[AA]>
Set the color for the background for details.

--details-border-size <size>
Set the thickness for the details border.

--button-border-size <size>
Set the thickness for the button border.

--button-gap <gap>
Set the size of the gap between buttons.

--button-dismiss-gap <gap>
Set the size of the gap between the dismiss button and another button.

--button-margin-right <margin>
Set the margin from the right of the dismiss button to edge.

--button-padding <padding>
Set the padding for the button text.

SEE

swaynag(5)

NAME

sway - configuration file and commands

DESCRIPTION

A sway configuration file is a list of sway commands that are executed by sway on startup. These commands usually consist of setting your preferences and setting key bindings. An example config is likely present in /etc/sway/config for you to check out.

Lines in the configuration file might be extended through multiple lines by adding a '\' character at the end of line. e.g.:

```
bindsym Shift+XF86AudioRaiseVolume exec \
        pactl set-sink-volume @DEFAULT_SINK@ -1%
```

Commands can also be given as a block in the form **command { <subcommands...> }**. Anything before the opening **{** will be prepended to the lines inside the block. For example:

```
output eDP-1 {
        background ~/wallpaper.png fill
        resolution 1920x1080
}
```

is identical to

```
output eDP-1 background ~/wallpaper.png fill
output eDP-1 resolution 1920x1080
```

These commands can be executed in your config file, via **swaymsg**(1), or via the bindsym command.

COMMAND CONVENTIONS

Commands are split into several arguments using spaces. You can enclose arguments with quotation marks (**"..."** or **'...'**) to add spaces to a single argument. You may also run several commands in order by separating each with **,** or **;**. Criteria is retained across commands separated by **,**, but will be reset (and allow for new criteria, if desired) for commands separated by a **;**.

Throughout the documentation, **|** is used to distinguish between arguments for which you may only select one. **[...]** is used for optional arguments, and **<...>** for arguments where you are expected to supply some value.

COMMANDS

This section only lists general commands. For input and output commands, refer to **sway-input**(5) and **sway-output**(5).

The following commands may only be used in the configuration file.

bar [<bar-id>] <bar-subcommands...>
: For details on bar subcommands, see **sway-bar**(5).

default_orientation horizontal|vertical|auto
: Sets the default container layout for tiled containers.

include <path>
: Includes another file from *path*. *path* can be either a full path or a path relative to the parent config, and expands shell syntax (see **wordexp**(3) for details). The same include file can only be included once; subsequent attempts will be ignored.

swaybg_command <command>
: Executes custom background *command*. Default is *swaybg*. Refer to **sway-output**(5) for more information.

  It can be disabled by setting the command to a single dash: *swaybg_command -*

swaynag_command <command>
: Executes custom command for *swaynag*. Default is *swaynag*. Additional arguments may be appended to the end. This should only be used to either direct sway to call swaynag from a custom path or to

provide additional arguments. This should be placed at the top of the config for the best results.

It can be disabled by setting the command to a single dash: *swaynag_command -*

workspace_layout default|stacking|tabbed
Specifies the initial layout for new containers in an empty workspace.

xwayland enable|disable|force
Enables or disables Xwayland support, which allows X11 applications to be used. *enable* will lazily load Xwayland so Xwayland will not be launched until the first client attempts to connect. In some cases, such as slower machines, it may be desirable to have Xwayland started immediately by using *force* instead of *enable*.

The following commands cannot be used directly in the configuration file. They are expected to be used with **bindsym** or at runtime through **swaymsg**(1).

border none|normal|csd|pixel [<n>]
Set border style for focused window. *normal* includes a border of thickness *n* and a title bar. *pixel* is a border without title bar *n* pixels thick. Default is *normal* with border thickness 2. *csd* is short for client-side-decorations, which allows the client to draw its own decorations.

border toggle
Cycles through the available border styles.

exit
Exit sway and end your Wayland session.

floating enable|disable|toggle
Make focused view floating, non-floating, or the opposite of what it is now.

<criteria> **focus**
Moves focus to the container that matches the specified criteria.

focus up|right|down|left
Moves focus to the next container in the specified direction.

focus prev|next [sibling]
Moves focus to the previous or next container in the current layout. By default, the last active child of the newly focused container will be focused. The *sibling* option indicates not to immediately focus a child of the container.

focus child
Moves focus to the last-focused child of the focused container.

focus parent
Moves focus to the parent of the focused container.

focus output up|right|down|left
Moves focus to the next output in the specified direction.

focus output <name>
Moves focus to the named output.

focus tiling
Sets focus to the last focused tiling container.

focus floating
Sets focus to the last focused floating container.

focus mode_toggle
Moves focus between the floating and tiled layers.

fullscreen [enable|disable|toggle] [global]
Makes focused view fullscreen, non-fullscreen, or the opposite of what it is now. If no argument is given, it does the same as *toggle*. If *global* is specified, the view will be fullscreen across all outputs.

gaps inner|outer|horizontal|vertical|top|right|bottom|left all|current set|plus|minus|toggle <amount>
Changes the *inner* or *outer* gaps for either *all* workspaces or the *current* workspace. *outer* gaps can be altered per side with *top*, *right*, *bottom*, and *left* or per direction with *horizontal* and *vertical*.

inhibit_idle focus|fullscreen|open|none|visible
Set/unset an idle inhibitor for the view. *focus* will inhibit idle when the view is focused by any seat. *fullscreen* will inhibit idle when the view is fullscreen (or a descendant of a fullscreen container) and is visible. *open* will inhibit idle until the view is closed (or the inhibitor is unset/changed). *visible* will inhibit idle when the view is visible on any output. *none* will remove any existing idle inhibitor for the view.

This can also be used with criteria to set an idle inhibitor for any existing view or with *for_window* to set idle inhibitors for future views.

layout default|splith|splitv|stacking|tabbed
Sets the layout mode of the focused container.

When using the *stacking* layout, only the focused window in the container is displayed, with the opened windows' list on the top of the container.

The *tabbed* layout is similar to *stacking*, but the windows' list is vertically split.

layout toggle [split|all]
Cycles the layout mode of the focused container though a preset list of layouts. If no argument is given, then it cycles through stacking, tabbed and the last split layout. If *split* is given, then it cycles through splith and splitv. If *all* is given, then it cycles through every layout.

layout toggle [split|tabbed|stacking|splitv|splith] [split|tabbed|stacking|splitv|splith]...
Cycles the layout mode of the focused container through a list of layouts.

max_render_time off|<msec>
Controls when the relevant application is told to render this window, as a positive number of milliseconds before the next time sway composites the output. A smaller number leads to fresher rendered frames being composited by sway and lower perceived input latency, but if set too low, the application may not finish rendering before sway composites the output, leading to delayed frames.

When set to off, the relevant application is told to render this window immediately after display refresh. How much time is left for rendering before sway composites the output at that point depends on the output **max_render_time** setting.

To set this up for optimal latency:

1. Set up **output max_render_time** (see **sway-output**(5)).
2. Put the target application in *full-screen* and have it continuously render something.
3. Start by setting **max_render_time 1**. If the application drops frames, increment by **1**.

This setting only has an effect if a per-output **max_render_time** is in effect on the output the window is currently on. See **sway-output**(5) for further details.

move left|right|up|down [<px> px]
Moves the focused container in the direction specified. The optional *px* argument specifies how many pixels to move the container. If unspecified, the default is 10 pixels. Pixels are ignored when moving tiled containers.

move [absolute] position <pos_x> [px|ppt] <pos_y> [px|ppt]
Moves the focused container to the specified position in the workspace. The position can be specified in pixels or percentage points, omitting the unit defaults to pixels. If *absolute* is used, the position is relative to all outputs. *absolute* can not be used with percentage points.

move [absolute] position center
Moves the focused container to be centered on the workspace. If *absolute* is used, it is moved to the center of all outputs.

move position cursor|mouse|pointer
Moves the focused container to be centered on the cursor.

move [container|window] [to] mark <mark>
Moves the focused container to the specified mark.

move [--no-auto-back-and-forth] [container|window] [to] workspace [number] <name>
Moves the focused container to the specified workspace. The string *number* is optional and is used to match a workspace with the same number, even if it has a different name.

move [container|window] [to] workspace prev|next|current
Moves the focused container to the previous, next or current workspace on this output, or if no workspaces remain, the previous or next output.

move [container|window] [to] workspace prev_on_output|next_on_output
Moves the focused container to the previous or next workspace on this output, wrapping around if already at the first or last workspace.

move [container|window] [to] workspace back_and_forth
Moves the focused container to previously focused workspace.

move [container|window] [to] output <name-or-id>|current
Moves the focused container to the specified output.

move [container|window] [to] output up|right|down|left
Moves the focused container to next output in the specified direction.

move [container|window] [to] scratchpad
Moves the focused container to the scratchpad.

move workspace [to] output <name-or-id>|current
Moves the focused workspace to the specified output.

move workspace to [output] <name-or-id>|current
Moves the focused workspace to the specified output.

move workspace [to] output up|right|down|left
Moves the focused workspace to next output in the specified direction.

move workspace to [output] up|right|down|left
Moves the focused workspace to next output in the specified direction.

nop <comment>
A no operation command that can be used to override default behaviour. The optional comment argument is ignored, but logged for debugging purposes.

reload
Reloads the sway config file and applies any changes. The config file is located at path specified by the command line arguments when started, otherwise according to the priority stated in **sway**(1).

rename workspace [<old_name>] to <new_name>
Rename either <old_name> or the focused workspace to the <new_name>

resize shrink|grow width|height [<amount> [px|ppt]]
Resizes the currently focused container by *amount*, specified in pixels or percentage points. If the units are omitted, floating containers are resized in px and tiled containers by ppt. *amount* will default to 10 if omitted.

resize set height <height> [px|ppt]
Sets the height of the container to *height*, specified in pixels or percentage points. If the units are omitted, floating containers are resized in px and tiled containers by ppt. If *height* is 0, the container will not be resized.

resize set [width] <width> [px|ppt]
Sets the width of the container to *width*, specified in pixels or percentage points. If the units are

omitted, floating containers are resized in px and tiled containers by ppt. If *width* is 0, the container will not be resized.

resize set [width] <width> [px|ppt] [height] <height> [px|ppt]
Sets the width and height of the container to *width* and *height*, specified in pixels or percentage points. If the units are omitted, floating containers are resized in px and tiled containers by ppt. If *width* or *height* is 0, the container will not be resized on that axis.

scratchpad show
Shows a window from the scratchpad. Repeatedly using this command will cycle through the windows in the scratchpad.

shortcuts_inhibitor enable|disable
Enables or disables the ability of clients to inhibit keyboard shortcuts for a view. This is primarily useful for virtualization and remote desktop software. It affects either the currently focused view or a set of views selected by criteria. Subcommand *disable* additionally deactivates any active inhibitors for the given view(s). Criteria are particularly useful with the **for_window** command to configure a class of views differently from the per-seat defaults established by the **seat** subcommand of the same name. See **sway-input**(5) for more ways to affect inhibitors.

split vertical|v|horizontal|h|none|n|toggle|t
Splits the current container, vertically or horizontally. When *none* is specified, the effect of a previous split is undone if the current container is the only child of a split parent. When *toggle* is specified, the current container is split opposite to the parent container's layout.

splith
Equivalent to **split horizontal**

splitv
Equivalent to **split vertical**

splitt
Equivalent to **split toggle**

sticky enable|disable|toggle
"Sticks" a floating window to the current output so that it shows up on all workspaces.

swap container with id|con_id|mark <arg>
Swaps the position, geometry, and fullscreen status of two containers. The first container can be selected either by criteria or focus. The second container can be selected by *id*, *con_id*, or *mark*. *id* can only be used with xwayland views. If the first container has focus, it will retain focus unless it is moved to a different workspace or the second container becomes fullscreen on the same workspace as the first container. In either of those cases, the second container will gain focus.

title_format <format>
Sets the format of window titles. The following placeholders may be used:

```
%title - The title supplied by the window
                    %app_id - The wayland app ID (applicable to wayland windows only)
                    %class - The X11 classname (applicable to xwayland windows only)
                    %instance - The X11 instance (applicable to xwayland windows only)
                    %shell - The protocol the window is using (typically xwayland or
    xdg_shell)
```

This command is typically used with **for_window** criteria. For example:

```
for_window [title="."] title_format "<b>%title</b> (%app_id)"
```

Note that markup requires pango to be enabled via the **font** command.

The default format is "%title".

The following commands may be used either in the configuration file or at runtime.

assign <criteria> [→] [workspace] [number] <workspace>

Assigns views matching *criteria* (see **CRITERIA** for details) to *workspace*. The → (U+2192) is optional and cosmetic. This command is equivalent to:

for_window <criteria> move container to workspace <workspace>

assign <criteria> [→] output left|right|up|down|<name>
Assigns views matching *criteria* (see **CRITERIA** for details) to the specified output. The → (U+2192) is optional and cosmetic. This command is equivalent to:

for_window <criteria> move container to output <output>

bindsym [--whole-window] [--border] [--exclude-titlebar] [--release] [--locked] [--to-code] [--input-device=<device>] [--no-warn] [--no-repeat] [Group<1-4>+]<key combo> <command>
Binds *key combo* to execute the sway command *command* when pressed. You may use XKB key names here (**wev**(1) is a good tool for discovering these). With the flag *--release*, the command is executed when the key combo is released. If *input-device* is given, the binding will only be executed for that input device and will be executed instead of any binding that is generic to all devices. If a group number is given, then the binding will only be available for that group. By default, if you overwrite a binding, swaynag will give you a warning. To silence this, use the *--no-warn* flag.

Unless the flag *--locked* is set, the command will not be run when a screen locking program is active. If there is a matching binding with and without *--locked*, the one with will be preferred when locked and the one without will be preferred when unlocked. If there are matching bindings and one has both *--input-device* and *--locked* and the other has neither, the former will be preferred even when unlocked.

Unless the flag *--inhibited* is set, the command will not be run when a keyboard shortcuts inhibitor is active for the currently focused window. Such inhibitors are usually requested by remote desktop and virtualization software to enable the user to send keyboard shortcuts to the remote or virtual session. The *--inhibited* flag allows one to define bindings which will be exempt from pass-through to such software. The same preference logic as for *--locked* applies.

Unless the flag *--no-repeat* is set, the command will be run repeatedly when the key is held, according to the repeat settings specified in the input configuration.

Bindings to keysyms are layout-dependent. This can be changed with the *--to-code* flag. In this case, the keysyms will be translated into the corresponding keycodes in the first configured layout.

Mouse bindings operate on the container under the cursor instead of the container that has focus. Mouse buttons can either be specified in the form *button[1-9]* or by using the name of the event code (ex *BTN_LEFT* or *BTN_RIGHT*). For the former option, the buttons will be mapped to their values in X11 (1=left, 2=middle, 3=right, 4=scroll up, 5=scroll down, 6=scroll left, 7=scroll right, 8=back, 9=forward). For the latter option, you can find the event names using *libinput debug-events*.

The priority for matching bindings is as follows: input device, group, and locked state.

--whole-window, *--border*, and *--exclude-titlebar* are mouse-only options which affect the region in which the mouse bindings can be triggered. By default, mouse bindings are only triggered when over the title bar. With the *--border* option, the border of the window will be included in this region. With the *--whole-window* option, the cursor can be anywhere over a window including the title, border, and content. *--exclude-titlebar* can be used in conjunction with any other option to specify that the titlebar should be excluded from the region of consideration.

If *--whole-window* is given, the command can be triggered when the cursor is over an empty workspace. Using a mouse binding over a layer surface's exclusive region is not currently possible.

Example:

```
# Execute firefox when alt, shift, and f are pressed together
bindsym Mod1+Shift+f exec firefox
```

bindcode [--whole-window] [--border] [--exclude-titlebar] [--release] [--locked] [--input-device=<device>] [--no-warn] [Group<1-4>+]<code> <command> is also available for binding with key/button codes instead of key/button names.

bindswitch [--locked] [--no-warn] [--reload] <switch>:<state> <command>

Binds <switch> to execute the sway command *command* on state changes. Supported switches are *lid* (laptop lid) and *tablet* (tablet mode) switches. Valid values for *state* are *on*, *off* and *toggle*. These switches are on when the device lid is shut and when tablet mode is active respectively. *toggle* is also supported to run a command both when the switch is toggled on or off.

Unless the flag --*locked* is set, the command will not be run when a screen locking program is active. If there is a matching binding with and without --*locked*, the one with will be preferred when locked and the one without will be preferred when unlocked.

If the --*reload* flag is given, the binding will also be executed when the config is reloaded. *toggle* bindings will not be executed on reload. The --*locked* flag will operate as normal so if the config is reloaded while locked and --*locked* is not given, the binding will not be executed.

By default, if you overwrite a binding, swaynag will give you a warning. To silence this, use the --*no-warn* flag.

Example:

```
# Show the virtual keyboard when tablet mode is entered.
bindswitch tablet:on busctl call --user sm.puri.OSK0 /sm/puri/OSK0 sm.puri.OSK0 SetVisible b true

# Log a message when the laptop lid is opened or closed.
bindswitch lid:toggle exec echo "Lid moved"
```

bindgesture [--exact] [--input-device=<device>] [--no-warn] <gesture>[:<fingers>][:directions] <command>

Binds *gesture* to execute the sway command *command* when detected. Currently supports the *hold*, *pinch* or *swipe* gesture. Optionally can be limited to bind to a certain number of *fingers* or, for a *pinch* or *swipe* gesture, to certain *directions*.

type	**fingers**	**direction**
hold	1 - 5	none
swipe	3 - 5	up, down, left, right
pinch	2 - 5	all above + inward, outward, clockwise, counterclockwise

The *fingers* can be limited to any sensible number or left empty to accept any finger counts. Valid directions are *up*, *down*, *left* and *right*, as well as *inward*, *outward*, *clockwise*, *counterclockwise* for the *pinch* gesture. Multiple directions can be combined by a plus.

If a *input-device* is given, the binding will only be executed for that input device and will be executed instead of any binding that is generic to all devices. By default, if you overwrite a binding, swaynag will give you a warning. To silence this, use the --*no-warn* flag.

The --*exact* flag can be used to ensure a binding only matches when exactly all specified directions are matched and nothing more. If there is matching binding with --*exact*, it will be preferred.

The priority for matching bindings is as follows: input device, then exact matches followed by matches with the highest number of matching directions.

Gestures executed while the pointer is above a bar are not handled by sway. See the respective documentation, e.g. **bindgesture** in **sway-bar**(5).

Example:

```
# Allow switching between workspaces with left and right swipes
bindgesture swipe:right workspace prev
bindgesture swipe:left workspace next

# Allow container movements by pinching them
bindgesture pinch:inward+up move up
bindgesture pinch:inward+down move down
bindgesture pinch:inward+left move left
bindgesture pinch:inward+right move right
```

client.background <color>
This command is ignored and is only present for i3 compatibility.

client.<class> <border> <background> <text> [<indicator> [<child_border>]]
Configures the color of window borders and title bars. The first three colors are required. When omitted *indicator* will use a sane default and *child_border* will use the color set for *background*. Colors may be specified in hex, either as *#RRGGBB* or *#RRGGBBAA*.

The available classes are:

client.focused
The window that has focus.

client.focused_inactive
The most recently focused view within a container which is not focused.

client.focused_tab_title
A view that has focused descendant container. Tab or stack container title that is the parent of the focused container but is not directly focused. Defaults to focused_inactive if not specified and does not use the indicator and child_border colors.

client.placeholder
Ignored (present for i3 compatibility).

client.unfocused
A view that does not have focus.

client.urgent
A view with an urgency hint. **Note**: Native Wayland windows do not support urgency. Urgency only works for Xwayland windows.

The meaning of each color is:

border
The border around the title bar.

background
The background of the title bar.

text
The text color of the title bar.

indicator
The color used to indicate where a new view will open. In a tiled container, this would paint the right border of the current view if a new view would be opened to the right.

child_border
The border around the view itself.

The default colors are:

class	*border*	*background*	*text*	*indicator*	*child_border*
background	n/a	#ffffff	n/a	n/a	n/a
focused	#4c7899	#285577	#ffffff	#2e9ef4	#285577
focused_inactive	#333333	#5f676a	#ffffff	#484e50	#5f676a
focused_tab_title	#333333	#5f676a	#ffffff	n/a	n/a
unfocused	#333333	#222222	#888888	#292d2e	#222222
urgent	#2f343a	#900000	#ffffff	#900000	#900000
placeholder	#000000	#0c0c0c	#ffffff	#000000	#0c0c0c

default_border normal|none|pixel [<n>]
Set default border style for new tiled windows.

default_floating_border normal|none|pixel [<n>]
Set default border style for new floating windows. This only applies to windows that are spawned in floating mode, not windows that become floating afterwards.

exec <shell command>
Executes *shell command* with sh.

exec_always <shell command>
Like **exec**, but the shell command will be executed *again* after **reload**.

floating_maximum_size <width> x <height>
Specifies the maximum size of floating windows. -1 x -1 removes the upper limit. The default is 0 x 0, which will use the width and height of the entire output layout as the maximums

floating_minimum_size <width> x <height>
Specifies the minimum size of floating windows. The default is 75 x 50.

floating_modifier <modifier> [normal|inverse]
When the *modifier* key is held down, you may hold left click to move windows, and right click to resize them. Setting *modifier* to *none* disables this feature. If *inverse* is specified, left click is used for resizing and right click for moving.

focus_follows_mouse yes|no|always
If set to *yes*, moving your mouse over a window will focus that window. If set to *always*, the window under the cursor will always be focused, even after switching between workspaces.

focus_on_window_activation smart|urgent|focus|none
This option determines what to do when a client requests window activation. If set to *urgent*, the urgent state will be set for that window. If set to *focus*, the window will become focused. If set to *smart*, the window will become focused only if it is already visible, otherwise the urgent state will be set. Default is *urgent*.

focus_wrapping yes|no|force|workspace
This option determines what to do when attempting to focus over the edge of a container. If set to *no*, the focused container will retain focus, if there are no other containers in the direction. If set to *yes*, focus will be wrapped to the opposite edge of the container, if there are no other containers in the direction. If set to *force*, focus will be wrapped to the opposite edge of the container, even if there are other containers in the direction. If set to *workspace*, focus will wrap like in the *yes* case and additionally wrap when moving outside of workspaces boundaries. Default is *yes*.

font [pango:]<font>
Sets font to use for the title bars. To enable support for pango markup, preface the font name with *pango:*. For example, *monospace 10* is the default font. To enable support for pango markup, *pango:monospace 10* should be used instead. Regardless of whether pango markup is enabled, *font* should be specified as a pango font description. For more information on pango font descriptions, see

https://docs.gtk.org/Pango/type_func.FontDescription.from_string.html#description

force_display_urgency_hint <timeout> [ms]
If an application on another workspace sets an urgency hint, switching to this workspace may lead to immediate focus of the application, which also means the window decoration color would be immediately reset to **client.focused**. This may make it unnecessarily hard to tell which window originally raised the event. This option allows one to set a *timeout* in ms to delay the urgency hint reset.

titlebar_border_thickness <thickness>
Thickness of the titlebar border in pixels

titlebar_padding <horizontal> [<vertical>]
Padding of the text in the titlebar. *horizontal* value affects horizontal padding of the text while *vertical* value affects vertical padding (space above and below text). Padding includes titlebar borders so their value should be greater than titlebar_border_thickness. If *vertical* value is not specified it is set to the *horizontal* value.

for_window <criteria> <command>
Whenever a window that matches *criteria* appears, run list of commands. See **CRITERIA** for more details.

gaps inner|outer|horizontal|vertical|top|right|bottom|left <amount>
Sets default *amount* pixels of *inner* or *outer* gap, where the inner affects spacing around each view and outer affects the spacing around each workspace. Outer gaps are in addition to inner gaps. To reduce or remove outer gaps, outer gaps can be set to a negative value. *outer* gaps can also be specified per side with *top*, *right*, *bottom*, and *left* or per direction with *horizontal* and *vertical*.

This affects new workspaces only, and is used when the workspace doesn't have its own gaps settings (see: workspace <ws> gaps ...).

hide_edge_borders [--i3] none|vertical|horizontal|both|smart|smart_no_gaps
Hides window borders adjacent to the screen edges. Default is *none*. The *--i3* option enables i3-compatible behavior to hide the title bar on tabbed and stacked containers with one child. The *smart*|*smart_no_gaps* options are equivalent to setting *smart_borders* smart|no_gaps and *hide_edge_borders* none.

input <input_device> <input-subcommands...>
For details on input subcommands, see **sway-input**(5).

* may be used in lieu of a specific device name to configure all input devices. A list of input device names may be obtained via **swaymsg -t get_inputs**.

seat <seat> <seat-subcommands...>
For details on seat subcommands, see **sway-input**(5).

kill
Kills (closes) the currently focused container and all of its children.

smart_borders on|no_gaps|off
If smart_borders are *on*, borders will only be enabled if the workspace has more than one visible child. If smart_borders is set to *no_gaps*, borders will only be enabled if the workspace has more than one visible child and gaps equal to zero.

smart_gaps on|off|toggle|inverse_outer
If smart_gaps are *on* gaps will only be enabled if a workspace has more than one child. If smart_gaps are *inverse_outer* outer gaps will only be enabled if a workspace has exactly one child.

mark --add|--replace [--toggle] <identifier>
Marks are arbitrary labels that can be used to identify certain windows and then jump to them at a later time. Each *identifier* can only be set on a single window at a time since they act as a unique identifier. By default, **mark** sets *identifier* as the only mark on a window. *--add* will instead add *identifier* to the list of current marks for that window. If *--toggle* is specified mark will remove *identifier* if it is already marked.

mode <mode>
Switches to the specified mode. The default mode is *default*.

mode [--pango_markup] <mode> <mode-subcommands...>
The only valid *mode-subcommands...* are **bindsym**, **bindcode**, **bindswitch**, and **set**. If *--pango_markup* is given, then *mode* will be interpreted as pango markup.

mouse_warping output|container|none
If *output* is specified, the mouse will be moved to new outputs as you move focus between them. If *container* is specified, the mouse will be moved to the middle of the container on switch. Default is *output*.

no_focus <criteria>
Prevents windows matching <criteria> from being focused automatically when they're created. This has no effect on the first window in a workspace.

output <output_name> <output-subcommands...>
For details on output subcommands, see **sway-output**(5).

* may be used in lieu of a specific output name to configure all outputs. A list of output names may be obtained via **swaymsg -t get_outputs**.

popup_during_fullscreen smart|ignore|leave_fullscreen
Determines what to do when a fullscreen view opens a dialog. If *smart* (the default), the dialog will be displayed. If *ignore*, the dialog will not be rendered. If *leave_fullscreen*, the view will exit fullscreen mode and the dialog will be rendered.

set $<name> <value>
Sets variable $*name* to *value*. You can use the new variable in the arguments of future commands. When the variable is used, it can be escaped with an additional $ (ie $$*name*) to have the replacement happen at run time instead of when reading the config. However, it does not always make sense for the variable to be replaced at run time since some arguments do need to be known at config time.

show_marks yes|no
If **show_marks** is yes, marks will be displayed in the window borders. Any mark that starts with an underscore will not be drawn even if **show_marks** is yes. The default is *yes*.

opacity [set|plus|minus] <value>
Adjusts the opacity of the window between 0 (completely transparent) and 1 (completely opaque). If the operation is omitted, *set* will be used.

tiling_drag enable|disable|toggle
Sets whether or not tiling containers can be dragged with the mouse. If *enabled* (default), the *floating_mod* can be used to drag tiling, as well as floating, containers. Using the left mouse button on title bars without the *floating_mod* will also allow the container to be dragged. *toggle* should not be used in the config file.

tiling_drag_threshold <threshold>
Sets the threshold that must be exceeded for a container to be dragged by its titlebar. This has no effect if *floating_mod* is used or if *tiling_drag* is set to *disable*. Once the threshold has been exceeded once, the drag starts and the cursor can come back inside the threshold without stopping the drag. *threshold* is multiplied by the scale of the output that the cursor on. The default is 9.

title_align left|center|right
Sets the title alignment. If *right* is selected and *show_marks* is set to *yes*, the marks will be shown on the *left* side instead of the *right* side.

unbindswitch <switch>:<state>
Removes a binding for when <switch> changes to <state>.

unbindgesture [--exact] [--input-device=<device>] <gesture>[:<fingers>][:directions]
Removes a binding for the specified *gesture*, *fingers* and *directions* combination.

unbindsym [--whole-window] [--border] [--exclude-titlebar] [--release] [--locked] [--to-code] [--input-device=<device>] <key combo>

Removes the binding for *key combo* that was previously bound with the given flags. If *input-device* is given, only the binding for that input device will be unbound.

unbindcode [--whole-window] [--border] [--exclude-titlebar] [--release] [--locked] [--input-device=<device>] <code> is also available for unbinding with key/button codes instead of key/button names.

unmark [<identifier>]

unmark will remove *identifier* from the list of current marks on a window. If *identifier* is omitted, all marks are removed.

urgent enable|disable|allow|deny

Using *enable* or *disable* manually sets or unsets the window's urgent state. Using *allow* or *deny* controls the window's ability to set itself as urgent. By default, windows are allowed to set their own urgency.

workspace [--no-auto-back-and-forth] [number] <[num:]name>

Switches to the specified workspace. The *num:* portion of the name is optional and will be used for ordering. If *num:* is not given and *name* is a number, then it will be also be used for ordering.

If the *no-auto-back-and-forth* option is given, then this command will not perform a back-and-forth operation when the workspace is already focused and *workspace_auto_back_and_forth* is enabled.

If the *number* keyword is specified and a workspace with the number already exists, then the workspace with the number will be used. If a workspace with the number does not exist, a new workspace will be created with the name *name*.

workspace prev|next

Switches to the next workspace on the current output or on the next output if currently on the last workspace.

workspace prev_on_output|next_on_output

Switches to the next workspace on the current output.

workspace back_and_forth

Switches to the previously focused workspace.

workspace <name> gaps inner|outer|horizontal|vertical|top|right|bottom|left <amount>

Specifies that workspace *name* should have the given gaps settings when it is created.

This command does not affect existing workspaces. To alter the gaps of an existing workspace, use the *gaps* command.

workspace <name> output <outputs...>

Specifies that workspace *name* should be shown on the specified *outputs*. Multiple outputs can be listed and the first available will be used. If the workspace gets placed on an output further down the list and an output that is higher on the list becomes available, the workspace will be moved to the higher priority output.

This command does not affect existing workspaces. To move an existing workspace, use the *move* command in combination with the *workspace* criteria (non-empty workspaces only) or *workspace* command (to switch to the workspace before moving).

workspace_auto_back_and_forth yes|no

When *yes*, repeating a workspace switch command will switch back to the prior workspace. For example, if you are currently on workspace 1, switch to workspace 2, then invoke the **workspace 2** command again, you will be returned to workspace 1. Default is *no*.

CRITERIA

A criteria is a string in the form of, for example:

```
[class="[Rr]egex.*" title="some title"]
```

The string contains one or more (space separated) attribute/value pairs. They are used by some commands to choose which views to execute actions on. All attributes must match for the criteria to match. Criteria is retained across commands separated by a **,**, but will be reset (and allow for new criteria, if desired) for commands separated by a **;**.

Criteria may be used with either the **for_window** or **assign** commands to specify operations to perform on new views. A criteria may also be used to perform specific commands (ones that normally act upon one window) on all views that match that criteria. For example:

Focus on a window with the mark "IRC":

```
[con_mark="IRC"] focus
```

Kill all windows with the title "Emacs":

```
[class="Emacs"] kill
```

You may like to use swaymsg -t get_tree for finding the values of these properties in practice for your applications.

The following attributes may be matched with:

app_id
: Compare value against the app id. Can be a regular expression. If value is __focused__, then the app id must be the same as that of the currently focused window. *app_id* are specific to Wayland applications.

class
: Compare value against the window class. Can be a regular expression. If value is __focused__, then the window class must be the same as that of the currently focused window. *class* are specific to X11 applications and require XWayland.

con_id
: Compare against the internal container ID, which you can find via IPC. If value is __focused__, then the id must be the same as that of the currently focused window.

con_mark
: Compare against the window marks. Can be a regular expression.

floating
: Matches floating windows.

id
: Compare value against the X11 window ID. Must be numeric. id is specific to X11 applications and requires XWayland.

instance
: Compare value against the window instance. Can be a regular expression. If value is __focused__, then the window instance must be the same as that of the currently focused window. instance is specific to X11 applications and requires XWayland.

pid
: Compare value against the window's process ID. Must be numeric.

shell
: Compare value against the window shell, such as "xdg_shell" or "xwayland". Can be a regular expression. If value is __focused__, then the shell must be the same as that of the currently focused window.

tiling
: Matches tiling windows.

title
: Compare against the window title. Can be a regular expression. If value is __focused__, then the window title must be the same as that of the currently focused window.

urgent
: Compares the urgent state of the window. Can be *first*, *last*, *latest*, *newest*, *oldest* or *recent*.

window_role
Compare against the window role (WM_WINDOW_ROLE). Can be a regular expression. If value is __focused__, then the window role must be the same as that of the currently focused window. window_role is specific to X11 applications and requires XWayland.

window_type
Compare against the window type (_NET_WM_WINDOW_TYPE). Possible values are normal, dialog, utility, toolbar, splash, menu, dropdown_menu, popup_menu, tooltip and notification. window_type is specific to X11 applications and requires XWayland.

workspace
Compare against the workspace name for this view. Can be a regular expression. If the value is __focused__, then all the views on the currently focused workspace matches.

SEE ALSO

sway(1) **sway-input**(5) **sway-output**(5) **sway-bar**(5) **sway-ipc**(7)

NAME

sway-bar - bar configuration file and commands

DESCRIPTION

Sway allows configuring swaybar in the sway configuration file.

COMMANDS

The following commands may only be used in the configuration file.

id <bar_id>
Sets the ID of the bar.

swaybar_command <command>
Executes custom bar command. Default is *swaybar*.

The following commands may be used either in the configuration file or at runtime.

bindcode [--release] <event-code> <command>
Executes *command* when the mouse button has been pressed (or if *released* is given, when the button has been released). The buttons can be given as an event code, which can be obtaining from **libinput debug-events**. To disable the default behavior for a button, use the command *nop*.

bindsym [--release] button[1-9]|<event-name> <command>
Executes *command* when the mouse button has been pressed (or if *released* is given, when the button has been released). The buttons can be given as a x11 button number or an event name, which can be obtained from **libinput debug-events**. To disable the default behavior for a button, use the command *nop*.

binding_mode_indicator yes|no
Enable or disable binding mode indicator. Default is *yes*.

font <font>
Specifies the font to be used in the bar. *font* should be specified as a pango font description. For more information on pango font descriptions, see https://docs.gtk.org/Pango/type_func.FontDescription.from_string.html#description

gaps <all> | <horizontal> <vertical> | <top> <right> <bottom> <left>
Sets the gaps from the edge of the screen for the bar. Gaps can either be set all at once, per direction, or per side. Note that only sides that touch an edge of the screen can have gaps. For the side that does not touch an edge of the screen, per-side outer gaps for workspaces may be of use.

height <height>
Sets the height of the bar. Default height (0) will match the font size.

hidden_state hide|show [<bar-id>]
Specifies the behaviour of the bar when it is in *hide* mode. When the hidden state is *hide*, then it is normally hidden, and only unhidden by pressing the modifier key or in case of urgency hints. When the hidden state is *show*, then it is permanently visible, drawn on top of the currently visible workspace. Default is *hide*.

For compatibility with i3, *bar hidden_state hide|show [<bar-id>]* is supported along with the sway only *bar <bar-id> hidden_state hide|show* syntax. When using the i3 syntax, if *bar-id* is omitted, the hidden_state will be changed for all bars. Attempting to use *bar <bar-id1> hidden_state hide|show <bar-id2>* will result in an error due to conflicting bar ids.

mode dock|hide|invisible|overlay [<bar-id>]
Specifies the visibility of the bar. In *dock* mode, it is permanently visible at one edge of the screen. In *hide* mode, it is hidden unless the modifier key is pressed, though this behaviour depends on the hidden state. In *invisible* mode, it is permanently hidden. In *overlay* mode, it is permanently visible on top of other windows. (In *overlay* mode the bar is transparent to input events.) Default is *dock*.

For compatibility with i3, *bar mode <mode> [<bar-id>]* syntax is supported along with the sway only *bar <bar-id> mode <mode>* syntax. When using the i3 syntax, if *bar-id* is omitted, the mode will be changed for all bars. Attempting to use *bar <bar-id1> mode <mode> <bar-id2>* will result in an error due to conflicting bar ids.

modifier <Modifier>|none
Specifies the modifier key that shows a hidden bar. Default is *Mod4*.

output <output>|*
Restrict the bar to a certain output, can be specified multiple times. If the output command is omitted, the bar will be displayed on all outputs. * can be given at any point to reset it back to all outputs.

pango_markup enabled|disabled
Enables or disables pango markup for status lines. This has no effect on status lines using the i3bar JSON protocol.

position top|bottom
Sets position of the bar. Default is *bottom*.

separator_symbol <symbol>
Specifies the separator symbol to separate blocks on the bar.

status_command <status command>
Executes the bar *status command* with *sh -c*. Each line of text printed to stdout from this command will be displayed in the status area of the bar. You may also use swaybar's JSON status line protocol. See **swaybar-protocol**(7) for more information on the protocol

If running this command via IPC, you can disable a running status command by setting the command to a single dash: *swaybar bar bar-0 status_command -*

status_edge_padding <padding>
Sets the padding that is used when the status line is at the right edge of the bar. This value will be multiplied by the output scale. The default is *3*.

status_padding <padding>
Sets the vertical padding that is used for the status line. The default is *1*. If *padding* is *0*, blocks will be able to take up the full height of the bar. This value will be multiplied by the output scale.

strip_workspace_name yes|no
If set to *yes*, then workspace names will be omitted from the workspace button and only the custom number will be shown. Default is *no*.

strip_workspace_numbers yes|no
If set to *yes*, then workspace numbers will be omitted from the workspace button and only the custom name will be shown. Default is *no*.

unbindcode [--release] <event-code>
Removes the binding with the given <event-code>.

unbindsym [--release] button[1-9]|<event-name>
Removes the binding with the given <button> or <event-name>.

wrap_scroll yes|no
Enables or disables wrapping when scrolling through workspaces with the scroll wheel. Default is *no*.

workspace_buttons yes|no
Enables or disables workspace buttons on the bar. Default is *yes*.

workspace_min_width <px> [px]
Specifies the minimum width for the workspace buttons on the bar. Default is *0*.

This setting also applies to the current binding mode indicator.

TRAY

Swaybar provides a system tray where third-party applications may place icons. The following commands configure the tray.

tray_bindcode <event-code>

ContextMenu|Activate|SecondaryActivate|ScrollDown|ScrollLeft|ScrollRight|ScrollUp|nop

Executes the action when the mouse button has been pressed. The buttons can be given as an event code, which can be obtained from **libinput debug-events**. To disable the default behavior for a button, use the command *nop*.

tray_bindsym button[1-9]|<event-name> ContextMenu|Activate|SecondaryActivate|ScrollDown|ScrollLeft|ScrollRight|ScrollUp|nop

Executes the action when the mouse button has been pressed. The buttons can be given as a x11 button number or an event name, which can be obtained from **libinput debug-events**. Use the command *nop* to disable the default action (Activate for button1, ContextMenu for button2 and SecondaryActivate for button3).

tray_padding <px> [px]

Sets the pixel padding of the system tray. This padding will surround the tray on all sides and between each item. The default value for *px* is 2.

tray_output none|<output>|*

Restrict the tray to a certain output, can be specified multiple times. If omitted, the tray will be displayed on all outputs. Unlike i3bar, swaybar can show icons on any number of bars and outputs without races. * can be given at any point to reset it to display on all outputs.

icon_theme <name>

Sets the icon theme that sway will look for item icons in. This option has no default value, because sway will always default to the fallback theme, hicolor.

COLORS

Colors are defined within a *colors { }* block inside a *bar { }* block. Colors must be defined in hex: *#RRGGBB* or *#RRGGBBAA*.

background <color>

Background color of the bar.

statusline <color>

Text color to be used for the statusline.

separator <color>

Text color to be used for the separator.

focused_background <color>

Background color of the bar on the currently focused monitor output. If not used, the color will be taken from *background*.

focused_statusline <color>

Text color to be used for the statusline on the currently focused monitor output. If not used, the color will be taken from *statusline*.

focused_separator <color>

Text color to be used for the separator on the currently focused monitor output. If not used, the color will be taken from *separator*.

focused_workspace <border> <background> <text>

Border, background and text color for a workspace button when the workspace has focus.

active_workspace <border> <background> <text>

Border, background and text color for a workspace button when the workspace is active (visible) on some output, but the focus is on another one. You can only tell this apart from the focused workspace when you are using multiple monitors.

inactive_workspace <border> <background> <text>

Border, background and text color for a workspace button when the workspace does not have focus and is not active (visible) on any output. This will be the case for most workspaces.

urgent_workspace <border> <background> <text>

Border, background and text color for a workspace button when the workspace contains a window with the urgency hint set.

binding_mode <border> <background> <text>
Border, background and text color for the binding mode indicator. If not used, the colors will be taken from *urgent_workspace*.

SEE ALSO

sway(5) **swaybar-protocol**(7)

NAME

sway-input - input configuration file and commands

DESCRIPTION

Sway allows for configuration of devices within the sway configuration file. To obtain a list of available device identifiers, run **swaymsg -t get_inputs**. Settings can also be applied to all input devices by using the wildcard, *, in place of *<identifier>* in the commands below. In addition, the settings can be applied to a type of device, by using *type:<input_type>* in place of *<identifier>*.

In the configuration file, settings with a more specific selector take precedence over more general ones: *<identifier>* > *type:<input_type>* > *. When executing input commands, however, the settings are applied to all matching input devices! This means that*type:<input_type>* can o verride previously set *<identifier>* settings, even though in a configuration file they would take precedence. Similarly * can override both *<identifier>* and *type:<input_type>* settings, if applied later.

Tip: If the configuration settings do not appear to be taking effect, you could try using * instead of *<identifier>*. If it works with the wildcard, try using a different identifier from **swaymsg -t get_inputs** until you find the correct input device.

Current available input types are:

- touchpad
- pointer
- keyboard
- touch
- tablet_tool
- tablet_pad
- switch

Note: The type configurations are applied as the devices appear and get applied on top of the existing device configurations.

INPUT COMMANDS

KEYBOARD CONFIGURATION

input <identifier> repeat_delay <milliseconds>
Sets the amount of time a key must be held before it starts repeating.

input <identifier> repeat_rate <characters per second>
Sets the frequency of key repeats once the repeat_delay has passed.

For more information on these xkb configuration options, see **xkeyboard-config**(7).

input <identifier> xkb_file <file_name>
Sets all xkb configurations from a complete .xkb file. This file can be dumped from *xkbcomp $DISPLAY keymap.xkb*. This setting overrides xkb_layout, xkb_model, xkb_options, xkb_rules, and xkb_variant settings.

input <identifier> xkb_layout <layout_name>
Sets the layout of the keyboard like *us* or *de*.

Multiple layouts can be specified by separating them with commas.

input <identifier> xkb_model <model_name>
Sets the model of the keyboard. This has an influence for some extra keys your keyboard might have.

input <identifier> xkb_options <options>
Sets extra xkb configuration options for the keyboard.

Multiple options can be specified by separating them with commas.

input <identifier> xkb_rules <rules>

Sets files of rules to be used for keyboard mapping composition.

input <identifier> xkb_switch_layout <index>|next|prev
Changes the active keyboard layout to <index> counting from zero or to next or previous layout on the list. If there is no next or previous layout, this command hops to the other end of the list.

This can be used when multiple layouts are configured with **xkb_layout**. A list of layouts you can switch between can be obtained with **swaymsg -t get_inputs**.

input <identifier> xkb_variant <variant>
Sets the variant of the keyboard like *dvorak* or *colemak*.

The following commands may only be used in the configuration file.

input <identifier> xkb_capslock enabled|disabled
Initially enables or disables CapsLock on startup, the default is disabled.

input <identifier> xkb_numlock enabled|disabled
Initially enables or disables NumLock on startup, the default is disabled.

TABLET CONFIGURATION

input <identifier> tool_mode <tool> <absolute|relative>
Sets whether movement of a tablet tool should be treated as absolute or relative; the default is absolute.

Valid values for *<tool>* are currently "pen", "eraser", "brush", "pencil", "airbrush", and the wildcard *, which matches all tools.

Mouse and lens tools ignore this setting and are always treated as relative.

MAPPING CONFIGURATION

input <identifier> map_to_output <identifier>
Maps inputs from this device to the specified output. Only meaningful if the device is a pointer, touch, or drawing tablet device.

The wildcard * can be used to map the input device to the whole desktop layout.

input <identifier> map_to_region <X> <Y> <width> <height>
Maps inputs from this device to the specified region of the global output layout. Only meaningful if the device is a pointer, touch, or drawing tablet device.

input <identifier> map_from_region <X1xY1> <X2xY2>
Ignores inputs from this device that do not occur within the specified region. Can be in millimeters (e.g. 10x20mm 20x40mm) or in terms of 0..1 (e.g. 0.5x0.5 0.7x0.7). Not all devices support millimeters. Only meaningful if the device is not a keyboard and provides events in absolute terms (such as a drawing tablet or touch screen - most pointers provide events relative to the previous frame).

LIBINPUT CONFIGURATION

input <identifier> accel_profile adaptive|flat
Sets the pointer acceleration profile for the specified input device.

input <identifier> calibration_matrix <6 space-separated floating point values>
Sets the calibration matrix.

input <identifier> click_method none|button_areas|clickfinger
Changes the click method for the specified device.

input <identifier> drag enabled|disabled
Enables or disables tap-and-drag for specified input device.

input <identifier> drag_lock enabled|disabled
Enables or disables drag lock for specified input device.

input <identifier> dwt enabled|disabled
Enables or disables disable-while-typing for the specified input device.

input <identifier> dwtp enabled|disabled

Enables or disables disable-while-trackpointing for the specified input device.

input <identifier> events enabled|disabled|disabled_on_external_mouse|toggle [<toggle-modes>]
Enables or disables send_events for specified input device. Disabling send_events disables the input device.

The *toggle* option cannot be used in the config. If no toggle modes are listed, all supported modes for the device will be toggled through in the order: enabled, disabled_on_external_mouse, disabled, (loop back). If toggle modes are listed, they will be cycled through, defaulting to the first mode listed if the current mode is not in the list. They will also not be checked to see if they are supported for the device and may fail.

input <identifier> left_handed enabled|disabled
Enables or disables left handed mode for specified input device.

input <identifier> middle_emulation enabled|disabled
Enables or disables middle click emulation.

input <identifier> natural_scroll enabled|disabled
Enables or disables natural (inverted) scrolling for the specified input device.

input <identifier> pointer_accel [<-1|1>]
Changes the pointer acceleration for the specified input device.

input <identifier> scroll_button disable|button[1-3,8,9]|<event-code-or-name>
Sets the button used for scroll_method on_button_down. The button can be given as an event name or code, which can be obtained from **libinput debug-events**, or as a x11 mouse button (button[1-3,8,9]). If set to *disable*, it disables the scroll_method on_button_down.

input <identifier> scroll_factor <floating point value>
Changes the scroll factor for the specified input device. Scroll speed will be scaled by the given value, which must be non-negative.

input <identifier> scroll_method none|two_finger|edge|on_button_down
Changes the scroll method for the specified input device.

input <identifier> tap enabled|disabled
Enables or disables tap for specified input device.

input <identifier> tap_button_map lrm|lmr
Specifies which button mapping to use for tapping. *lrm* treats 1 finger as left click, 2 fingers as right click, and 3 fingers as middle click. *lmr* treats 1 finger as left click, 2 fingers as middle click, and 3 fingers as right click.

SEAT CONFIGURATION

Configure options for multiseat mode.

A **seat** is a collection of input devices that act independently of each other. Seats are identified by name and the default seat is *seat0* if no seats are configured. While sway is running, - (hyphen) can be used as an alias for the current seat. Each seat has an independent keyboard focus and a separate cursor that is controlled by the pointer devices of the seat. This is useful for multiple people using the desktop at the same time with their own devices (each sitting in their own "seat"). The wildcard character, *, can also be used in place of *<identifier>* to change settings for all seats.

Tip: If the configuration settings do not appear to be taking effect, you could try using * instead of *<identifier>*. If it works with the wildcard, try using a different identifier from **swaymsg -t get_seats** until you find the correct seat.

seat <name> attach <input_identifier>
Attach an input device to this seat by its input identifier. A special value of "*" will attach all devices to the seat.

seat <seat> cursor move|set <x> <y>
Move specified seat's cursor relative to current position or wrap to absolute coordinates (with respect

to the global coordinate space). Specifying either value as 0 will not update that coordinate.

seat <seat> cursor press|release button[1-9]|<event-name-or-code>
Simulate pressing (or releasing) the specified mouse button on the specified seat. The button can either be provided as a button event name or event code, which can be obtained from **libinput debug-events**, or as an x11 mouse button (button[1-9]). If using button[4-7], which map to axes, an axis event will be simulated, however *press* and *release* will be ignored and both will occur.

seat <name> fallback true|false
Set this seat as the fallback seat. A fallback seat will attach any device not explicitly attached to another seat (similar to a "default" seat).

seat <name> hide_cursor <timeout>|when-typing [enable|disable]
Hides the cursor image after the specified event occurred.

If *timeout* is specified, then the cursor will be hidden after *timeout* (in milliseconds) has elapsed with no activity on the cursor. A timeout of 0 (default) disables hiding the cursor. The minimal timeout is 100 and any value less than that (aside from 0), will be increased to 100.

If *when-typing* is enabled, then the cursor will be hidden whenever a key is pressed.

seat <name> idle_inhibit <sources...>
Sets the set of input event sources which can prevent the seat from becoming idle, as a space separated list of source names. Valid names are "keyboard", "pointer", "touchpad", "touch", "tablet_pad", "tablet_tool", and "switch". The default behavior is to prevent idle on any event.

seat <name> idle_wake <sources...>
Sets the set of input event sources which can wake the seat from its idle state, as a space separated list of source names. Valid names are "keyboard", "pointer", "touchpad", "touch", "tablet_pad", "tablet_tool", and "switch". The default behavior is to wake from idle on any event.

seat <name> keyboard_grouping none|smart
Set how the keyboards in the seat are grouped together. Currently, there are two options. *none* will disable all keyboard grouping. This will make it so each keyboard device has its own isolated state. *smart* will group the keyboards in the seat by their keymap and repeat info. This is useful for when the keyboard appears as multiple separate input devices. In this mode, the effective layout is synced between the keyboards in the group. The default is *smart*. To restore the behavior of older versions of sway, use *none*.

seat <name> pointer_constraint enable|disable|escape
Enables or disables the ability for clients to capture the cursor (enabled by default) for the seat. This is primarily useful for video games. The "escape" command can be used at runtime to escape from a captured client.

seat <name> shortcuts_inhibitor enable|disable|activate|deactivate|toggle
Enables or disables the ability of clients to inhibit keyboard shortcuts for the seat. This is primarily useful for virtualization and remote desktop software. Subcommands *enable* and *disable* affect whether future inhibitors are honoured by default, i.e. activated automatically, the default being *enable*. When used at runtime, *disable* also disables any currently active inhibitors. *activate*, *deactivate* and *toggle* are only usable at runtime and change the state of a potentially existing inhibitor on the currently focused window. This can be used with the current seat alias (-) to affect only the currently focused window of the current seat. Subcommand *deactivate* is particularly useful in an *--inhibited* **bindsym** to escape a state where shortcuts are inhibited and the client becomes uncooperative. It is worth noting that whether disabled or deactivated inhibitors are removed is entirely up to the client. Depending on the client it may therefore be possible to (re-)activate them later. Any visual indication that an inhibitor is present is currently left to the client as well.

seat <name> xcursor_theme <theme> [<size>]
Override the system default XCursor theme. The default seat's (*seat0*) theme is also used as the default cursor theme in XWayland, and exported through the *XCURSOR_THEME* and *XCURSOR_SIZE* environment variables.

SEE ALSO

sway(5) **sway-output**(5) **xkeyboard-config**(7)

NAME

swaynag - swaynag configuration file

SYNOPSIS

$HOME/.swaynag/config, $XDG_CONFIG_HOME/swaynag/config, SYSCONFDIR/swaynag/config

CONFIG FILE

At the top of the config file, *swaynag* options can be set using the format *long-option=value*. These will be used as default values if *swaynag* is not given the option. This can be useful for setting a preferred font, output, and edge.

Below the options, custom types may be defined. To define a type, use the following format:

```
[name-of-type]
option=value
```

All colors may be given in the form *RRGGBB* or *RRGGBBAA*. The following colors can be set:

background=<color>
The background color for *swaynag*.

border=<color>
The color to use for borders of buttons.

border-bottom=<color>
The color of the border line at the bottom of *swaynag*.

button-background=<color>
The background color for the buttons.

text=<color>
The color of the text.

button-text=<color>
The color of the button text.

The following sizing options can also be set:

border-bottom-size=<size>
Set the thickness of the bottom border.

message-padding=<padding>
Set the padding for the message.

details-background=<color>
The background color for the details.

details-border-size=<size>
Set the thickness for the details border.

button-border-size=<size>
Set the thickness for the button border.

button-gap=<gap>
Set the size of the gap between buttons.

button-dismiss-gap=<gap>
Set the size of the gap between the dismiss button and another button.

button-margin-right=<margin>
Set the margin from the right of the dismiss button to edge.

button-padding=<padding>
Set the padding for the button text.

Additionally, the following options can be assigned a default per-type:

edge=top|bottom

Set the edge to use.

layer=overlay|top|bottom|background
Set the layer to use.

font=<font>
Set the font to use.

output=<output>
Set the output to use. This should be the name of a *xdg_output*.

EXAMPLE

```
font=Monospace 12
edge=bottom

[green]
edge=top
background=00AA00
border=006600
border-bottom=004400
text=FFFFFF
button-background=00CC00
message-padding=10
```

SEE

swaynag(1)

NAME

sway-output - output configuration commands for sway

DESCRIPTION

You may combine output commands into one, like so:

```
output HDMI-A-1 mode 1920x1080 pos 1920 0 bg ˜/wallpaper.png stretch
```

You can get a list of output names with **swaymsg -t get_outputs**. You may also match any output by using the output name "*". Additionally, "-" can be used to match the focused output by name and "--" can be used to match the focused output by its identifier.

Some outputs may have different names when disconnecting and reconnecting. To identify these, the name can be substituted for a string consisting of the make, model and serial which you can get from **swaymsg -t get_outputs**. Each value must be separated by one space. For example:

```
output "Some Company ABC123 0x00000000" pos 1920 0
```

COMMANDS

output <name> mode|resolution|res [--custom] <width>x<height>[@<rate>Hz]

Configures the specified output to use the given mode. Modes are a combination of width and height (in pixels) and a refresh rate that your display can be configured to use. For a list of available modes for each output, use **swaymsg -t get_outputs**.

To set a custom mode not listed in the list of available modes, use **--custom**. You should probably only use this if you know what you're doing.

Examples:

```
output HDMI-A-1 mode 1920x1080

output HDMI-A-1 mode 1920x1080@60Hz
```

output <name> modeline <clock> <hdisplay> <hsync_start> <hsync_end> <htotal> <vdisplay> <vsync_start> <vsync_end> <vtotal> <hsync> <vsync>

Configures the specified output to use the given modeline. It can be generated using **cvt**(1) and **gtf**(1) commands. See **xorg.conf**(5). Only supported on DRM backend.

Example:

```
output HDMI-A-1 modeline 173.00 1920 2048 2248 2576 1080 1083 1088 1120 -hsync +vsync
```

output <name> position|pos <X> <Y>

Places the specified output at the specific position in the global coordinate space. The cursor may only be moved between immediately adjacent outputs. If scaling is active, it has to be considered when positioning. For example, if the scaling factor for the left output is 2, the relative position for the right output has to be divided by 2. The reference point is the top left corner so if you want the bottoms aligned this has to be considered as well.

Example:

```
output HDMI1 scale 2

output HDMI1 pos 0 1020 res 3200x1800

output eDP1 pos 1600 0 res 1920x1080
```

Note that the left x-pos of eDP1 is 1600 = 3200/2 and the bottom y-pos is 1020 + (1800 / 2) = 1920 = 0 + 1920

output <name> scale <factor>

Scales the specified output by the specified scale *factor*. An integer is recommended, but fractional values are also supported. If a fractional value are specified, be warned that it is not possible to faithfully represent the contents of your windows - they will be rendered at the next highest integer scale factor and downscaled. You may be better served by setting an integer scale factor and adjusting the font size of your applications to taste. HiDPI isn't supported with Xwayland clients (windows will blur).

output <name> scale_filter linear|nearest|smart
Indicates how to scale application buffers that are rendered at a scale lower than the output's configured scale, such as lo-dpi applications on hi-dpi screens. Linear is smoother and blurrier, nearest (also known as nearest neighbor) is sharper and blockier. Setting "smart" will apply nearest scaling when the output has an integer scale factor, otherwise linear. The default is "smart".

output <name> subpixel rgb|bgr|vrgb|vbgr|none
Manually sets the subpixel hinting for the specified output. This value is usually auto-detected, but some displays may misreport their subpixel geometry. Using the correct subpixel hinting allows for sharper text. Incorrect values will result in blurrier text. When changing this via **swaymsg**, some applications may need to be restarted to use the new value.

output <name> background|bg <file> <mode> [<fallback_color>]
Sets the wallpaper for the given output to the specified file, using the given scaling mode (one of "stretch", "fill", "fit", "center", "tile"). If the specified file cannot be accessed or if the image does not fill the entire output, a fallback color may be provided to cover the rest of the output. *fallback_color* should be specified as *#RRGGBB*. Alpha is not supported.

output <name> background|bg <color> solid_color
Sets the background of the given output to the specified color. *color* should be specified as *#RRGGBB*. Alpha is not supported.

output <name> transform <transform> [clockwise|anticlockwise]
Sets the background transform to the given value. Can be one of "90", "180", "270" for rotation; or "flipped", "flipped-90", "flipped-180", "flipped-270" to apply a rotation and flip, or "normal" to apply no transform. The rotation is performed clockwise. If a single output is chosen and a rotation direction is specified (*clockwise* or *anticlockwise*) then the transform is added or subtracted from the current transform (this cannot be used directly in the configuration file).

output <name> disable|enable
Enables or disables the specified output (all outputs are enabled by default).

As opposed to the *power* command, the output will lose its current workspace and windows.

output <name> toggle
Toggle the specified output.

output <name> power on|off|toggle
Turns on or off the specified output.

As opposed to the *enable* and *disable* commands, the output keeps its current workspaces and windows.

output <name> dpms on|off|toggle
Deprecated. Alias for *power*.

output <name> max_render_time off|<msec>
Controls when sway composites the output, as a positive number of milliseconds before the next display refresh. A smaller number leads to fresher composited frames and lower perceived input latency, but if set too low, sway may not finish compositing in time for display refresh, leading to delayed frames.

When set to off, sway composites immediately after display refresh, maximizing time available for compositing.

To adjust when applications are instructed to render, see **max_render_time** in **sway**(5).

To set this up for optimal latency:

1. Launch some *full-screen* application that renders continuously, like **glxgears**.
2. Start with **max_render_time 1**. Increment by **1** if you see frame drops.

This setting only has an effect on Wayland and DRM backends, as support for presentation timestamps and predicted output refresh rate is required.

output <name> adaptive_sync on|off
Enables or disables adaptive synchronization (often referred to as Variable Refresh Rate, or by the vendor-specific names FreeSync/G-Sync).

Adaptive sync allows clients to submit frames a little too late without having to wait a whole refresh period to display it on screen. Enabling adaptive sync can improve latency, but can cause flickering on some hardware.

output <name> render_bit_depth 8|10
Controls the color channel bit depth at which frames are rendered; the default is currently 8 bits per channel.

Setting higher values will not have an effect if hardware and software lack support for such bit depths. Successfully increasing the render bit depth will not necessarily increase the bit depth of the frames sent to a display. An increased render bit depth may provide smoother rendering of gradients, and screenshots which can more precisely store the colors of programs which display high bit depth colors.

Warnings: this can break screenshot/screencast programs which have not been updated to work with different bit depths. This command is experimental, and may be removed or changed in the future.

SEE ALSO

sway(5) **sway-input**(5)

NAME

swaybar-protocol - JSON status line protocol for swaybar

SUMMARY

swaybar defines the following JSON protocol to specify what information is displayed in the status line on the right side of swaybar. The protocol comprises a header, that is a JSON object, followed by a newline (**0x0A**), followed by an infinite JSON array that represents the information to display. All communication is done by writing the status line to standard output and reading events from standard input.

HEADER

The header is a JSON object with support for the following properties (only *version* is required):

PROPERTY	DATA TYPE	DEFAULT	DESCRIPTION
version	integer	-	The protocol version to use. Currently, this must be **1**
click_events	boolean	false	Whether to receive click event information to standard input
cont_signal	integer	SIGCONT	The signal that sway-bar should send to con-tinue processing
stop_signal	integer	SIGSTOP	The signal that sway-bar should send to stop processing

MINIMAL HEADER

```
{
        "version": 1
}
```

FULL HEADER

```
{
        "version": 1,
        "click_events": true,
        "cont_signal": 18,
        "stop_signal": 19
}
```

BODY

The body is an infinite array, where each element of the array is a representation of the status line at the time that the element was written. Each element of the array is itself an array of JSON objects, where each object represents a block in the status line. Each block can have the following properties (only *full_text* is required):

Other properties may be given and swaybar will ignore any property that it does not know. It is recommended to prefix any custom properties with an underscore to make it easier to distinguish them from protocol properties.

BODY EXAMPLE

```
[
    [
        {
            "full_text": "25%",
            "min_width": "100%",
            "urgent": false
        },
        {
            "full_text": "Thu 30 May 2019 02:15:15"
        }
    ],
    [
        {
            "full_text": "20%",
            "min_width": "100%",
            "urgent": false
        },
        {
            "full_text": "Thu 30 May 2019 02:20:52"
        }
    ],
    [
        {
            "full_text": "15%",
            "min_width": "100%",
            "urgent": true
        },
        {
            "full_text": "Thu 30 May 2019 02:25:41"
        }
    ],
    ...
```

FULL BLOCK EXAMPLE

```
{
    "full_text": "Thu 30 May 2019 02:09:15",
    "short_text": "02:09",
    "color": "#ccccccff",
    "background": "#111111ff",
    "border": "#222222ff",
    "border_top": 1,
    "border_bottom": 1,
    "border_left": 1,
    "border_right": 1,
    "min_width": 100,
    "align": "center",
    "name": "clock",
    "instance": "edt",
    "urgent": false,
    "separator": true,
    "separator_block_width": 5,
        "markup": "none"
}
```

CLICK EVENTS

If requested in the header, swaybar will write a JSON object, that can be read from standard in, when the user clicks on a block. The event object will have the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
name	string	The name of the block, if set
instance	string	The instance of the block, if set
x	integer	The x location that the click occurred at
y	integer	The y location that the click occurred at
button	integer	The x11 button number for the click. If the button does not have an x11 button mapping, this will be *0*.
event	integer	The event code that corresponds to the button for the click
relative_x	integer	The x location of the click relative to the top-left of the block
relative_y	integer	The y location of the click relative to the top-left of the block
width	integer	The width of the block in pixels
height	integer	The height of the block in pixels

Differences from i3bar's protocol:

- *button* may be *0* for buttons that do not have x11 button mappings
- *event* has been introduced to allow for non-x11 buttons to be represented
- The *modifiers* property is not currently added by swaybar

EXAMPLE

```
{
        "name": "clock",
        "instance": "edt",
        "x": 1900,
        "y": 10,
        "button": 1,
        "event": 274,
        "relative_x": 100,
        "relative_y": 8,
        "width": 120,
        "height": 18
}
```

SEE ALSO

sway-bar(5)

NAME

sway-ipc - IPC protocol for sway

DESCRIPTION

This details the interprocess communication (IPC) protocol for **sway**(1). This IPC protocol can be used to control or obtain information from a sway process.

The IPC protocol uses a UNIX socket as the method of communication. The path to the socket is stored in the environment variable *SWAYSOCK* and, for backwards compatibility with i3, *I3SOCK*. You can also retrieve the socket path by calling *sway --get-socketpath*.

MESSAGE AND REPLY FORMAT

The format for messages and replies is:

```
<magic-string> <payload-length> <payload-type> <payload>
```

Where

<magic-string> is *i3-ipc*, for compatibility with i3
<payload-length> is a 32-bit integer in native byte order
<payload-type> is a 32-bit integer in native byte order

For example, sending the *exit* command would look like the following hexdump:

```
00000000 | 69 33 2d 69 70 63 04 00 00 00 00 00 00 00 65 78 |i3-ipc........ex|
00000010 | 69 74                                           |it              |
```

The payload for replies will be a valid serialized JSON data structure.

MESSAGES AND REPLIES

The following message types and their corresponding reply types are currently supported. **For all replies, any properties not listed are subject to removal.**

TYPE NUMBER	MESSAGE NAME	PURPOSE
0	RUN_COMMAND	Runs the payload as sway commands
1	GET_WORKSPACES	Get the list of current workspaces
2	SUBSCRIBE	Subscribe the IPC connection to the events listed in the payload
3	GET_OUTPUTS	Get the list of current outputs
4	GET_TREE	Get the node layout tree
5	GET_MARKS	Get the names of all the marks currently set
6	GET_BAR_CONFIG	Get the specified bar config or a list of bar config names
7	GET_VERSION	Get the version of sway that owns the IPC socket
8	GET_BINDING_MODES	Get the list of binding mode names
9	GET_CONFIG	Returns the config that was last loaded
10	SEND_TICK	Sends a tick event with the specified payload
11	SYNC	Replies failure object for i3 compatibility
12	GET_BINDING_STATE	Request the current binding state, e.g. the currently active binding mode name.
100	GET_INPUTS	Get the list of input devices
101	GET_SEATS	Get the list of seats

0. RUN_COMMAND

MESSAGE

Parses and runs the payload as sway commands

REPLY

An array of objects corresponding to each command that was parsed. Each object has the property *success*, which is a boolean indicating whether the command was successful. The object may also contain the properties *error* and *parse_error*. The *error* property is a human readable error message while *parse_error* is a boolean indicating whether the reason the command failed was because the command was unknown or not able to be parsed.

Example Reply:

```
[
        {
                "success": true
        },
        {
                "success": false,
                "parse_error": true,
                "error": "Invalid/unknown command"
        }
]
```

1. GET_WORKSPACES

MESSAGE

Retrieves the list of workspaces.

REPLY

The reply is an array of objects corresponding to each workspace. Each object has the following properties:

PROPERTY	**DATA TYPE**	**DESCRIPTION**
num	integer	The workspace number or -1 for workspaces that do not start with a number
name	string	The name of the workspace
visible	boolean	Whether the workspace is currently visible on any output
focused	boolean	Whether the workspace is currently focused by the default seat (*seat0*)
urgent	boolean	Whether a view on the workspace has the urgent flag set
rect	object	The bounds of the workspace. It consists of *x*, *y*, *width*, and *height*
output	string	The name of the output that the workspace is on

Example Reply:

```
[
        {
                "num": 1,
                "name": "1",
                "visible": true,
                "focused": true,
                "rect": {
                        "x": 0,
                        "y": 23,
                        "width": 1920,
                        "height": 1057
                },
                "output": "eDP-1"
        },
]
```

2. SUBSCRIBE

MESSAGE

Subscribe this IPC connection to the event types specified in the message payload. The payload should be a valid JSON array of events. See the *EVENTS* section for the list of supported events.

REPLY

A single object that contains the property *success*, which is a boolean value indicating whether the subscription was successful or not.

Example Reply:

```
{
        "success": true
}
```

3. GET_OUTPUTS

MESSAGE

Retrieve the list of outputs

REPLY

An array of objects corresponding to each output. Each object has the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
name	string	The name of the output. On DRM, this is the connector
make	string	The make of the output
model	string	The model of the output
serial	string	The output's serial number as a hexadecimal string
active	boolean	Whether this output is active/enabled
dpms	boolean	(Deprecated, use *power* instead) Whether this output is on/off (via DPMS)
power	boolean	Whether this output is on/off
primary	boolean	For i3 compatibility, this will be false. It does not make sense in Wayland
scale	float	The scale currently in use on the output or *-1* for disabled outputs
subpixel_hinting	string	The subpixel hinting current in use on the output. This can be *rgb*, *bgr*, *vrgb*, *vbgr*, or *none*
transform	string	The transform currently in use for the output. This can be *normal*, *90*, *180*, *270*, *flipped-90*, *flipped-180*, or *flipped-270*
current_workspace	string	The workspace currently visible on the output or *null* for disabled outputs
modes	array	An array of supported mode objects. Each object contains *width*, *height*, and *refresh*
current_mode	object	An object representing the current mode containing *width*, *height*, and *refresh*
rect	object	The bounds for the output consisting of *x*, *y*, *width*, and *height*

Example Reply:

```
[
        {
```

```
                        "name": "HDMI-A-2",
                        "make": "Unknown",
                        "model": "NS-19E310A13",
                        "serial": "0x00000001",
                        "active": true,
                        "primary": false,
                        "scale": 1.0,
                        "subpixel_hinting": "rgb",
                        "transform": "normal",
                        "current_workspace": "1",
                        "modes": [
                                {
                                        "width": 640,
                                        "height": 480,
                                        "refresh": 59940
                                },
                                {
                                        "width": 800,
                                        "height": 600,
                                        "refresh": 60317
                                },
                                {
                                        "width": 1024,
                                        "height": 768,
                                        "refresh": 60004
                                },
                                {
                                        "width": 1920,
                                        "height": 1080,
                                        "refresh": 60000
                                }
                        ],
                        "current_mode": {
                                "width": 1920,
                                "height": 1080,
                                "refresh": 60000
                        }
                }
        ]
```

4. GET_TREE

MESSAGE

Retrieve a JSON representation of the tree

REPLY

An array of objects that represent the current tree. Each object represents one node and will have the following properties:

Example Reply:

```
{
        "id": 1,
        "name": "root",
        "rect": {
                "x": 0,
                "y": 0,
                "width": 1920,
                "height": 1080
        },
        "focused": false,
        "focus": [
                3
        ],
        "border": "none",
        "current_border_width": 0,
        "layout": "splith",
        "orientation": "horizontal",
        "percent": null,
        "window_rect": {
                "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
        },
        "deco_rect": {
                "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
        },
        "geometry": {
                "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
        },
        "window": null,
        "urgent": false,
        "floating_nodes": [
        ],
        "sticky": false,
        "type": "root",
        "nodes": [
                {
                        "id": 2147483647,
                        "name": "__i3",
                        "rect": {
                                "x": 0,
                                "y": 0,
                                "width": 1920,
                                "height": 1080
                        },
                        "focused": false,
"focus": [
        2147483646
],
"border": "none",
"current_border_width": 0,
"layout": "output",
"orientation": "horizontal",
"percent": null,
"window_rect": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
},
"deco_rect": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
},
"geometry": {
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
},
"window": null,
"urgent": false,
"floating_nodes": [
],
"sticky": false,
"type": "output",
"nodes": [
        {
                "id": 2147483646,
                "name": "__i3_scratch",
                "rect": {
                        "x": 0,
                        "y": 0,
                        "width": 1920,
                        "height": 1080
                },
                "focused": false,
                "focus": [
                ],
                "border": "none",
                "current_border_width": 0,
                "layout": "splith",
                "orientation": "horizontal",
                "percent": null,
                "window_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                                        "height": 0
                                },
                                "deco_rect": {
                                        "x": 0,
                                        "y": 0,
                                        "width": 0,
                                        "height": 0
                                },
                                "geometry": {
                                        "x": 0,
                                        "y": 0,
                                        "width": 0,
                                        "height": 0
                                },
                                "window": null,
                                "urgent": false,
                                "floating_nodes": [
                                ],
                                "sticky": false,
                                "type": "workspace"
                        }
                ]
        },
        {
                "id": 3,
                "name": "eDP-1",
                "rect": {
                        "x": 0,
                        "y": 0,
                        "width": 1920,
                        "height": 1080
                },
                "focused": false,
                "focus": [
                        4
                ],
                "border": "none",
                "current_border_width": 0,
                "layout": "output",
                "orientation": "none",
                "percent": 1.0,
                "window_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "deco_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "geometry": {
```

```
        "x": 0,
        "y": 0,
        "width": 0,
        "height": 0
},
"window": null,
"urgent": false,
"floating_nodes": [
],
"sticky": false,
"type": "output",
"active": true,
"primary": false,
"make": "Unknown",
"model": "0x38ED",
"serial": "0x00000000",
"scale": 1.0,
"transform": "normal",
"current_workspace": "1",
"modes": [
        {
                "width": 1920,
                "height": 1080,
                "refresh": 60052
        }
],
"current_mode": {
        "width": 1920,
        "height": 1080,
        "refresh": 60052
},
"nodes": [
        {
                "id": 4,
                "name": "1",
                "rect": {
                        "x": 0,
                        "y": 23,
                        "width": 1920,
                        "height": 1057
                },
                "focused": false,
                "focus": [
                        6,
                        5
                ],
                "border": "none",
                "current_border_width": 0,
                "layout": "splith",
                "orientation": "horizontal",
                "percent": null,
                "window_rect": {
                        "x": 0,
                        "y": 0,
                "width": 0,
                "height": 0
        },
        "deco_rect": {
                "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
        },
        "geometry": {
                "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
        },
        "window": null,
        "urgent": false,
        "floating_nodes": [
        ],
        "sticky": false,
        "num": 1,
        "output": "eDP-1",
        "type": "workspace",
        "representation": "H[URxvt termite]",
        "nodes": [
                {
                        "id": 5,
                        "name": "urxvt",
                        "rect": {
                                "x": 0,
                                "y": 23,
                                "width": 960,
                                "height": 1057
                        },
                        "focused": false,
                        "focus": [
                        ],
                        "border": "normal",
                        "current_border_width": 2,
                        "layout": "none",
                        "orientation": "none",
                        "percent": 0.5,
                        "window_rect": {
                                "x": 2,
                                "y": 0,
                                "width": 956,
                                "height": 1030
                        },
                        "deco_rect": {
                                "x": 0,
                                "y": 0,
                                "width": 960,
                                "height": 25
                        },
                                        "geometry": {
                                                "x": 0,
                                                "y": 0,
                                                "width": 1124,
                                                "height": 422
                                        },
                                        "window": 4194313,
                                        "urgent": false,
                                        "floating_nodes": [
                                        ],
                                        "sticky": false,
                                        "type": "con",
                                        "fullscreen_mode": 0,
                                        "pid": 23959,
                                        "app_id": null,
                                        "visible": true,
                                        "shell": "xwayland",
                                        "inhibit_idle": true,
                                        "idle_inhibitors": {
                                                "application": "none",
                                                "user": "visible",
                                        },
                                        "window_properties": {
                                                "class": "URxvt",
                                                "instance": "urxvt",
                                                "title": "urxvt",
                                                "transient_for": null
                                        },
                                        "nodes": [
                                        ]
                                },
                                {
                                        "id": 6,
                                        "name": "",
                                        "rect": {
                                                "x": 960,
                                                "y": 23,
                                                "width": 960,
                                                "height": 1057
                                        },
                                        "focused": true,
                                        "focus": [
                                        ],
                                        "border": "normal",
                                        "current_border_width": 2,
                                        "layout": "none",
                                        "orientation": "none",
                                        "percent": 0.5,
                                        "window_rect": {
                                                "x": 2,
                                                "y": 0,
                                                "width": 956,
                                                "height": 1030
                                        },
                                                  "deco_rect": {
                                                       "x": 0,
                                                       "y": 0,
                                                       "width": 960,
                                                       "height": 25
                                                  },
                                                  "geometry": {
                                                       "x": 0,
                                                       "y": 0,
                                                       "width": 817,
                                                       "height": 458
                                                  },
                                                  "window": null,
                                                  "urgent": false,
                                                  "floating_nodes": [
                                                  ],
                                                  "sticky": false,
                                                  "type": "con",
                                                  "fullscreen_mode": 0,
                                                  "pid": 25370,
                                                  "app_id": "termite",
                                                  "visible": true,
                                                  "shell": "xdg_shell",
                                                  "inhibit_idle": false,
                                                  "idle_inhibitors": {
                                                       "application": "none",
                                                       "user": "fullscreen",
                                                  },
                                                  "nodes": [
                                                  ]
                                             }
                                        ]
                                   }
                              ]
                         }
                    ]
               }
```

5. GET_MARKS

MESSAGE

Retrieve the currently set marks

REPLY

An array of marks current set. Since each mark can only be set for one container, this is a set so each value is unique and the order is undefined.

Example Reply:

```
[
     "one",
     "test"
]
```

6. GET_BAR_CONFIG (WITHOUT A PAYLOAD)

MESSAGE

When sending without a payload, this retrieves the list of configured bar IDs

REPLY

An array of bar IDs, which are strings

Example Reply:

```
[
        "bar-0",
        "bar-1"
]
```

6. GET_BAR_CONFIG (WITH A PAYLOAD)

MESSAGE

When sent with a bar ID as the payload, this retrieves the config associated with the specified by the bar ID in the payload. This is used by swaybar, but could also be used for third party bars

REPLY

An object that represents the configuration for the bar with the bar ID sent as the payload. The following properties exists and more information about what their value mean can be found in **sway-bar**(5):

PROPERTY	DATA TYPE	DESCRIPTION
id	string	The bar ID
mode	string	The mode for the bar. It can be *dock*, *hide*, or *invisible*
position	string	The bar's position. It can currently either be *bottom* or *top*
status_command	string	The command which should be run to generate the status line
font	string	The font to use for the text on the bar
workspace_buttons	boolean	Whether to display the workspace buttons on the bar
workspace_min_width	integer	Minimum width in px for the workspace buttons on the bar
binding_mode_indicator	boolean	Whether to display the current binding mode on the bar
verbose	boolean	For i3 compatibility, this will be the boolean value *false*.
colors	object	An object containing the *#RRGGBBAA* colors to use for the bar. See below for more information
gaps	object	An object representing the gaps for the bar consisting of *top*, *right*, *bottom*, and *left*.
bar_height	integer	The absolute height to use for the bar or *0* to automatically size based on the font
status_padding	integer	The vertical padding to use for the status line
status_edge_padding	integer	The horizontal padding to use for the status line when at the end of an output

The colors object contains the following properties, which are all strings containing the *#RRGGBBAA* representation of the color:

Example Reply:

```
{
        "id": "bar-0",
        "mode": "dock",
        "position": "top",
        "status_command": "while date +'%Y-%m-%d %l:%M:%S %p'; do sleep 1; done",
        "font": "monospace 10",
        "gaps": {
                "top": 0,
                "right": 0,
                "bottom": 0,
                "left": 0
        },
        "bar_height": 0,
        "status_padding": 1,
        "status_edge_padding": 3,
        "workspace_buttons": true,
        "workspace_min_width": 0,
        "binding_mode_indicator": true,
        "verbose": false,
        "pango_markup": false,
        "colors": {
                "background": "#323232ff",
                "statusline": "#ffffffff",
                "separator": "#666666ff",
                "focused_background": "#323232ff",
                "focused_statusline": "#ffffffff",
                "focused_separator": "#666666ff",
                "focused_workspace_border": "#4c7899ff",
                "focused_workspace_bg": "#285577ff",
                "focused_workspace_text": "#ffffffff",
                "inactive_workspace_border": "#32323200",
                "inactive_workspace_bg": "#32323200",
                "inactive_workspace_text": "#5c5c5cff",
                "active_workspace_border": "#333333ff",
                "active_workspace_bg": "#5f676aff",
                "active_workspace_text": "#ffffffff",
                "urgent_workspace_border": "#2f343aff",
                "urgent_workspace_bg": "#900000ff",
                "urgent_workspace_text": "#ffffffff",
                "binding_mode_border": "#2f343aff",
                "binding_mode_bg": "#900000ff",
                "binding_mode_text": "#ffffffff"
        },
}
```

7. GET_VERSION

MESSAGE

Retrieve version information about the sway process

REPLY

An object containing the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
major	integer	The major version of the sway process
minor	integer	The minor version of the sway process
patch	integer	The patch version of the sway process
human_readable	string	A human readable version string that will likely contain more useful information such as the git commit short hash and git branch
loaded_config_file_name	string	The path to the loaded config file

Example Reply:

```
{
        "human_readable": "1.0-rc1-117-g2f7247e0 (Feb 24 2019, branch ’master’)",
        "major": 1,
        "minor": 0,
        "patch": 0,
        "loaded_config_file_name": "/home/redsoxfan/.config/sway/config"
}
```

8. GET_BINDING_MODES

MESSAGE

Retrieve the list of binding modes that currently configured

REPLY

An array of strings, with each string being the name of a binding mode. This will always contain at least one mode (currently *default*), which is the default binding mode

Example Reply:

```
[
        "default",
        "resize",
]
```

9. GET_CONFIG

MESSAGE

Retrieve the contents of the config that was last loaded

REPLY

An object with a single string property containing the contents of the config

Example Reply:

```
{
        "config": "set $mod Mod4nbindsym $mod+q exitn"
}
```

10. SEND_TICK

MESSAGE

Issues a *TICK* event to all clients subscribing to the event to ensure that all events prior to the tick were received. If a payload is given, it will be included in the *TICK* event

REPLY

A single object contains the property *success*, which is a boolean value indicating whether the *TICK* event was sent.

Example Reply:

```
{
        "success": true
}
```

11. SYNC

MESSAGE

For i3 compatibility, this command will just return a failure object since it does not make sense to implement in sway due to the X11 nature of the command. If you are curious about what this IPC command does in i3, refer to the i3 documentation.

REPLY

A single object that contains the property *success*, which is set to the boolean value *false*.

Exact Reply:

```
{
        "success": false
}
```

12. GET_BINDING_STATE

MESSAGE

Returns the currently active binding mode.

REPLY

A single object that contains the property *name*, which is set to the currently active binding mode as a string.

Exact Reply:

```
{
        "name": "default"
}
```

100. GET_INPUTS

MESSAGE

Retrieve a list of the input devices currently available

REPLY

An array of objects corresponding to each input device. Each object has the following properties:

PROPERTY	**DATA TYPE**	**DESCRIPTION**
identifier	string	The identifier for the input device
name	string	The human readable name for the device
vendor	integer	The vendor code for the input device
product	integer	The product code for the input device
type	string	The device type. Currently this can be *keyboard*, *pointer*, *touch*, *tablet_tool*, *tablet_pad*, or *switch*
xkb_active_layout_name	string	(Only keyboards) The name of the active keyboard layout in use
xkb_layout_names	array	(Only keyboards) A list a layout names configured for the keyboard
xkb_active_layout_index	integer	(Only keyboards) The index of the active keyboard layout in use
scroll_factor	floating	(Only pointers) Multiplier applied on scroll event values.
libinput	object	(Only libinput devices) An object describing the current device settings. See below for more information

The *libinput* object describes the device configuration for libinput devices. Only properties that are supported for the device will be added to the object. In addition to the possible options listed, all string properties may also be *unknown*, in the case that a new option is added to libinput. See **sway-input**(5) for information on the meaning of the possible values. The following properties will be included for devices that support them:

Example Reply:

```
[
        {
                "identifier": "1:1:AT_Translated_Set_2_keyboard",
                "name": "AT Translated Set 2 keyboard",
                "vendor": 1,
                "product": 1,
                "type": "keyboard",
                "xkb_active_layout_name": "English (US)",
                "libinput": {
                        "send_events": "enabled"
                }
        },
        {
                "identifier": "1267:5:Elan_Touchpad",
                "name": "Elan Touchpad",
                "vendor": 1267,
                "product": 5,
                "type": "pointer",
                "libinput": {
                        "send_events": "enabled",
                        "tap": "enabled",
                        "tap_button_map": "lmr",
                        "tap_drag": "enabled",
                        "tap_drag_lock": "disabled",
                        "accel_speed": 0.0,
                        "accel_profile": "none",
                        "natural_scroll", "disabled",
                        "left_handed": "disabled",
                        "click_method": "button_areas",
                        "middle_emulation": "disabled",
                        "scroll_method": "edge",
                        "dwt": "enabled",
                        "dwtp": "enabled"
                }
        },
        {
                "identifier": "3034:22494:USB2.0_VGA_UVC_WebCam:_USB2.0_V",
                "name": "USB2.0 VGA UVC WebCam: USB2.0 V",
                "vendor": 3034,
                "product": 22494,
                "type": "keyboard",
                "xkb_active_layout_name": "English (US)",
                "libinput": {
                        "send_events": "enabled"
                }
        },
        {
                "identifier": "0:3:Sleep_Button",
                "name": "Sleep Button",
                "vendor": 0,
                "product": 3,
                "type": "keyboard",
                "xkb_active_layout_name": "English (US)",
                "libinput": {
                    "send_events": "enabled"
                }
            },
            {
                "identifier": "0:5:Lid_Switch",
                "name": "Lid Switch",
                "vendor": 0,
                "product": 5,
                "type": "switch",
                "libinput": {
                    "send_events": "enabled"
                }
            },
            {
                "identifier": "0:6:Video_Bus",
                "name": "Video Bus",
                "vendor": 0,
                "product": 6,
                "type": "keyboard",
                "xkb_active_layout_name": "English (US)",
                "libinput": {
                    "send_events": "enabled"
                }
            },
            {
                "identifier": "0:1:Power_Button",
                "name": "Power Button",
                "vendor": 0,
                "product": 1,
                "type": "keyboard",
                "xkb_active_layout_name": "English (US)",
                "libinput": {
                    "send_events": "enabled"
                }
            }
        ]
```

101. GET_SEATS

MESSAGE

Retrieve a list of the seats currently configured

REPLY

An array of objects corresponding to each seat. There will always be at least one seat. Each object has the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
name	string	The unique name for the seat
capabilities	integer	The number of capabilities that the seat has
focus	integer	The id of the node currently focused by the seat or *0* when the seat is not currently focused by a node (i.e. a surface layer or xwayland unmanaged has focus)
devices	array	An array of input devices that are attached to the seat. Currently, this is an array of objects that are identical to those returned by *GET_INPUTS*

Example Reply:

```
[
        {
                "name": "seat0",
                "capabilities": 3,
                "focus": 7,
                "devices": [
                        {
                                "identifier": "1:1:AT_Translated_Set_2_keyboard",
                                "name": "AT Translated Set 2 keyboard",
                                "vendor": 1,
                                "product": 1,
                                "type": "keyboard",
                                "xkb_active_layout_name": "English (US)",
                                "libinput": {
                                        "send_events": "enabled"
                                }
                        },
                        {
                                "identifier": "1267:5:Elan_Touchpad",
                                "name": "Elan Touchpad",
                                "vendor": 1267,
                                "product": 5,
                                "type": "pointer",
                                "libinput": {
                                        "send_events": "enabled",
                                        "tap": "enabled",
                                        "tap_button_map": "lmr",
                                        "tap_drag": "enabled",
                                        "tap_drag_lock": "disabled",
                                        "accel_speed": 0.0,
                                        "accel_profile": "none",
                                        "natural_scroll", "disabled",
                                        "left_handed": "disabled",
                                        "click_method": "button_areas",
                                        "middle_emulation": "disabled",
                "scroll_method": "edge",
                "dwt": "enabled",
                "dwtp": "enabled"
            }
        },
        {
            "identifier": "3034:22494:USB2.0_VGA_UVC_WebCam:_USB2.0_V",
            "name": "USB2.0 VGA UVC WebCam: USB2.0 V",
            "vendor": 3034,
            "product": 22494,
            "type": "keyboard",
            "xkb_active_layout_name": "English (US)",
            "libinput": {
                "send_events": "enabled"
            }
        },
        {
            "identifier": "0:3:Sleep_Button",
            "name": "Sleep Button",
            "vendor": 0,
            "product": 3,
            "type": "keyboard",
            "xkb_active_layout_name": "English (US)",
            "libinput": {
                "send_events": "enabled"
            }
        },
        {
            "identifier": "0:5:Lid_Switch",
            "name": "Lid Switch",
            "vendor": 0,
            "product": 5,
            "type": "switch",
            "libinput": {
                "send_events": "enabled"
            }
        },
        {
            "identifier": "0:6:Video_Bus",
            "name": "Video Bus",
            "vendor": 0,
            "product": 6,
            "type": "keyboard",
            "xkb_active_layout_name": "English (US)",
            "libinput": {
                "send_events": "enabled"
            }
        },
        {
            "identifier": "0:1:Power_Button",
            "name": "Power Button",
            "vendor": 0,
            "product": 1,
            "type": "keyboard",
                        "xkb_active_layout_name": "English (US)",
                        "libinput": {
                                "send_events": "enabled"
                        }
                }
            ]
        }
    ]
```

EVENTS

Events are a way for client to get notified of changes to sway. A client can subscribe to any events it wants to be notified of changes for. The event is sent in the same format as a reply. The following events are currently available:

EVENT TYPE	NAME	DESCRIPTION
0x80000000	workspace	Sent whenever an event involving a workspace occurs such as initialization of a new workspace or a different workspace gains focus
0x80000002	mode	Sent whenever the binding mode changes
0x80000003	window	Sent whenever an event involving a view occurs such as being reparented, focused, or closed
0x80000004	barconfig_update	Sent whenever a bar config changes
0x80000005	binding	Sent when a configured binding is executed
0x80000006	shutdown	Sent when the ipc shuts down because sway is exiting
0x80000007	tick	Sent when an ipc client sends a *SEND_TICK* message
0x80000014	bar_state_update	Send when the visibility of a bar should change due to a modifier
0x80000015	input	Sent when something related to input devices changes

0x80000000. WORKSPACE

Sent whenever a change involving a workspace occurs. The event consists of a single object with the following properties:

PROPERTY	**DATA TYPE**	**DESCRIPTION**
change	string	The type of change that occurred. See below for more information
current	object	An object representing the workspace effected or *null* for *reload* changes
old	object	For a *focus* change, this is will be an object representing the workspace being switched from. Otherwise, it is *null*

The following change types are currently available:

TYPE	**DESCRIPTION**
init	The workspace was created
empty	The workspace is empty and is being destroyed since it is not visible
focus	The workspace was focused. See the *old* property for the previous focus
move	The workspace was moved to a different output
rename	The workspace was renamed
urgent	A view on the workspace has had their urgency hint set or all urgency hints for views on the workspace have been cleared
reload	The configuration file has been reloaded

Example Event:

```
{
        "change": "init",
        "old": null,
        "current": {
                "id": 10,
                "name": "2",
                "rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "focused": false,
                "focus": [
                ],
                "border": "none",
                "current_border_width": 0,
                "layout": "splith",
                "percent": null,
                "window_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "deco_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "geometry": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "window": null,
                "urgent": false,
                "floating_nodes": [
                ],
                "num": 2,
                "output": "eDP-1",
                "type": "workspace",
                "representation": null,
                "nodes": [
                ]
        }
}
```

0x80000002. MODE

Sent whenever the binding mode changes. The event consists of a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
change	string	The binding mode that became active
pango_markup	boolean	Whether the mode should be parsed as pango markup

Example Event:

```
{
        "change": "default",
        "pango_markup": false
}
```

0x80000003. WINDOW

Sent whenever a change involving a view occurs. The event consists of a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
change	string	The type of change that occurred. See below for more information
container	object	An object representing the view effected

The following change types are currently available:

TYPE	DESCRIPTION
new	The view was created
close	The view was closed
focus	The view was focused
title	The view's title has changed
fullscreen_mode	The view's fullscreen mode has changed
move	The view has been reparented in the tree
floating	The view has become floating or is no longer floating
urgent	The view's urgency hint has changed status
mark	A mark has been added or removed from the view

Example Event:

```
{
        "change": "new",
        "container": {
                "id": 12,
                "name": null,
                "rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "focused": false,
                "focus": [
                ],
                "border": "none",
                "current_border_width": 0,
                "layout": "none",
                "percent": 0.0,
                "window_rect": {
                        "x": 0,
                        "y": 0,
                        "width": 0,
                        "height": 0
                },
                "deco_rect": {
                        "x": 0,
                "y": 0,
                "width": 0,
                "height": 0
            },
            "geometry": {
                "x": 0,
                "y": 0,
                "width": 1124,
                "height": 422
            },
            "window": 4194313,
            "urgent": false,
            "floating_nodes": [
            ],
            "type": "con",
            "pid": 19787,
            "app_id": null,
            "window_properties": {
                "class": "URxvt",
                "instance": "urxvt",
                "transient_for": null
            },
            "nodes": [
            ]
        }
    }
```

0x80000004. BARCONFIG_UPDATE

Sent whenever a config for a bar changes. The event is identical to that of *GET_BAR_CONFIG* when a bar ID is given as a payload. See *6. GET_BAR_CONFIG (WITH A PAYLOAD)* above for more information.

0x80000005. BINDING

Sent whenever a binding is executed. The event is a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
change	string	The change that occurred for the binding. Currently this will only be *run*
command	string	The command associated with the binding
event_state_mask	array	An array of strings that correspond to each modifier key for the binding
input_code	integer	For keyboard bindcodes, this is the key code for the binding. For mouse bindings, this is the X11 button number, if there is an equivalent. In all other cases, this will be *0*.
symbol	string	For keyboard bindsyms, this is the bindsym for the binding. Otherwise, this will be *null*
input_type	string	The input type that triggered the binding. This is either *keyboard* or *mouse*

Example Event:

```
{
        "change": "run",
        "binding": {
                "command": "workspace 2",
                "event_state_mask": [
                        "Mod4"
                ],
                "input_code": 0,
                "symbol": "2",
                "input_type": "keyboard"
        }
}
```

0x80000006. SHUTDOWN

Sent whenever the IPC is shutting down. The event is a single object with the property *change*, which is a string containing the reason for the shutdown. Currently, the only value for *change* is *exit*, which is issued when sway is exiting.

Example Event:

```
{
        "change": "exit"
}
```

0x80000007. TICK

Sent when first subscribing to tick events or by a *SEND_TICK* message. The event is a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
first	boolean	Whether this event was triggered by subscribing to the tick events
payload	string	The payload given with a *SEND_TICK* message, if any. Otherwise, an empty string

Example Event:

```
{
        "first": true
        "payload": ""
}
```

0x80000014. BAR_STATE_UPDATE

Sent when the visibility of a bar changes due to a modifier being pressed. The event is a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
id	string	The bar ID effected
visible_by_modifier	boolean	Whether the bar should be made visible due to a modifier being pressed

Example Event:

```
{
        "id": "bar-0",
        "visible_by_modifier": true
}
```

0x80000015. INPUT

Sent when something related to the input devices changes. The event is a single object with the following properties:

PROPERTY	DATA TYPE	DESCRIPTION
change	string	What has changed
input	object	An object representing the input that is identical the ones GET_INPUTS gives

The following change types are currently available:

TYPE	DESCRIPTION
added	The input device became available
removed	The input device is no longer available
xkb_keymap	(Keyboards only) The keymap for the keyboard has changed
xkb_layout	(Keyboards only) The effective layout in the keymap has changed
libinput_config	(libinput device only) A libinput config option for the device changed

Example Event:

```
{
        "change": "xkb_layout",
        "input": {
                "identifier": "1:1:AT_Translated_Set_2_keyboard",
                "name": "AT Translated Set 2 keyboard",
                "vendor": 1,
                "product": 1,
                "type": "keyboard",
                "xkb_layout_names": [
                        "English (US)",
                        "English (Dvorak)"
                ],
                "xkb_active_layout_index": 1,
                "xkb_active_layout_name": "English (Dvorak)",
                "libinput": {
                        "send_events": "enabled"
                }
        }
}
```

SEE ALSO

sway(1) **sway**(5) **sway-bar**(5) **swaymsg**(1) **sway-input**(5) **sway-output**(5)