Window Rules

> ⚠ Warning
> Rules are evaluated top to bottom, so the order they're written in does matter! More info in Notes

Window Rules

You can set window rules to achieve different window behaviors based on their properties.

Syntax

Basic named rule syntax:

```
windowrule {
  name = apply-something
  match:class = my-window

  border_size = 10
}
```

Basic anonymous rule syntax:

```
windowrule = match:class my-window, border_size 10
```

Rules are split into two categories of parameters: *props* and *effects*. Props are the `match:` parts, which are used to determine if a window should get the rule. Effects are what is applied.

All props must match for a rule to be applied.

You can have as many props and effects per rule as you please, in any order as you please, as long as:

- there is only one of one type (e.g. specifying `match:class` twice is invalid)
- there is at least one *prop*

Props

The supported fields for props are:

Field	Argument	Description
match:class	[RegEx]	Windows with `class` matching `RegEx`.
match:title	[RegEx]	Windows with `title` matching `RegEx`.
match:initial_class	[RegEx]	Windows with `initialClass` matching `RegEx`.
match:initial_title	[RegEx]	Windows with `initialTitle` matching `RegEx`.
match:tag	[name]	Windows with matching `tag`.
match:xwayland	[bool]	Xwayland windows.
match:float	[bool]	Floating windows.
match:fullscreen	[bool]	Fullscreen windows.
match:pin	[bool]	Pinned windows.
match:focus	[bool]	Currently focused window.
match:group	[bool]	Grouped windows.
match:modal	[bool]	Modal windows (e.g. "Are you sure" popups)
match:fullscreen_state_client	[client]	Windows with matching `fullscreenstate.client` can be `0` - none, `1` - maximize, `2` - fullscreen, `3` - maximize and fullscreen.

Field	Argument	Description
match:fullscreen_state_internal	[internal]	Windows with matching `fullscreenstate.internal` can be `0` - none, `1` - maximize, `2` - fullscreen, `3` - maximize and fullscreen.
match:workspace	[workspace]	Windows on matching workspace. `w` can be `id`, `name:string` or `workspace selector`.
match:content	[int]	Windows with specified content type (none = 0, photo = 1, video = 2, game = 3)
match:xdg_tag	[RegEx]	Match a window by its xdgTag (see `hyprctl clients` to check if it has one)

Keep in mind that you *have* to declare at least one field, but not all.

> Note
> To get more information about a window's class, title, XWayland status or its size, you can use `hyprctl clients`.

> Note
> In the output of the `hyprctl clients` command: `fullscreen` refers to `fullscreen_state_internal` and `fullscreenClient` refers to `fullscreen_state_client`

RegEx writing

Please note Hyprland uses Google's RE2 for parsing RegEx. This means that all operations requiring polynomial time to compute will not work. See the RE2 wiki for supported extensions.

If you want to *negate* a RegEx, as in pass only when the RegEx *fails*, you can prefix it with `negative:`, e.g.: `negative:kitty`.

Effects

Static effects

Static effects are evaluated once when the window is opened and never again. This essentially means that it is always the `initialTitle` and `initialClass` which will be found when matching on `title` and `class`, respectively.

> Warning
> It is not possible to `float` (or any other static rule) a window based on a change in the `title` after the window has been created. This applies to all static effects listed here.

Effect	argument	Description
float	[on]	Floats a window.
tile	[on]	Tiles a window.
fullscreen	[on]	Fullscreens a window.
maximize	[on]	Maximizes a window.
fullscreen_state	[internal] [client]	Sets the focused window's fullscreen mode and the one sent to the client, where internal and client can be `0` - none, `1` - maximize, `2` - fullscreen, `3` - maximize and fullscreen.
move	[expr] [expr]	Moves a floating window to a given coordinate, monitor-local. Two expressions are space-separated.
size	[expr] [expr]	Resizes a floating window to a given size. Two expressions are space-separated.
center	[on]	If the window is floating, will center it on the monitor.
pseudo	[on]	Pseudotiles a window.
monitor	[id]	Sets the monitor on which a window should open. `id` can be either the id number or the name (e.g. `1` or `DP-1`).
workspace	[w]	Sets the workspace on which a window should open (for workspace syntax, see dispatchers->workspaces). You can also set [w] to `unset`. This will unset all previous workspace rules applied to this window. Additionally you can add `silent` after the workspace to make the window open silently.
no_initial_focus	[on]	Disables the initial focus to the window

Effect	argument	Description
pin	[on]	Pins the window (i.e. show it on all workspaces). *Note: floating only*.
group	[options]	Sets window group properties. See the note below.
suppress_event	[types...]	Ignores specific events from the window. Events are space separated, and can be: `fullscreen`, `maximize`, `activate`, `activatefocus`, `fullscreenoutput`.
content	[none|photo|video|game]	Sets content type.
no_close_for	[ms]	Makes the window uncloseable with the `killactive` dispatcher for a given amount of ms on open.

Expressions

Expressions are space-separated, so your math cannot have spaces. They are regular, math expressions, with a few variables exposed, names of which are self-explanatory. All position variables are monitor-local.

- `monitor_w` and `monitor_h` for monitor size
- `window_x` and `window_y` for window position
- `window_w` and `window_h` for window size
- `cursor_x` and `cursor_y` for cursor position

Example expressions:

- `window_w*0.5`
- `(monitor_w/2)+17`

It's probably a good idea to surround your expressions with parentheses for clarity, with space-separation:

- `(monitor_w*0.5) (monitor_h*0.5)`
- `((monitor_w*0.5)+17) (monitor_h*0.2)`

Dynamic effects

Dynamic effects are re-evaluated every time a property changes.

Effect	argument	Description
persistent_size	[on]	Allows size persistence between application launches for floating windows.
no_max_size	[on]	Removes max size limitations. Especially useful with windows that report invalid max sizes (e.g. winecfg).
stay_focused	[on]	Forces focus on the window as long as it's visible.
animation	[style] ([opt])	Forces an animation onto a window, with a selected opt. Opt is optional.
border_color	[c]	Force the border color of the window. Options for c: `color` / `color ... color angle` -> sets the active border color/gradient OR `color color` / `color ... color angle color ... color [angle]` -> sets the active and inactive border color/gradient of the window. See variables->colors for color definition.
idle_inhibit	[mode]	Sets an idle inhibit rule for the window. If active, apps like `hypridle` will not fire. Modes: `none`, `always`, `focus`, `fullscreen`.
opacity	[a]	Additional opacity multiplier. Options for a: `float` -> sets an overall opacity, `float float` -> sets active_opacity and inactive_opacity respectively, `float float float` -> sets active_opacity, inactive_opacity and fullscreen_opacity respectively.
tag	[name]	Applies the tag `name` to the window, use prefix `+` / `-` to set/unset flag, or no prefix to toggle the flag.
max_size	[w] [h]	Sets the maximum size (x,y -> int). Applies to floating windows. (use `misc:size_limits_tiled` to include tiled windows.)
min_size	[w] [h]	Sets the minimum size (x,y -> int). Applies to floating windows. (use `misc:size_limits_tiled` to include tiled windows.)
border_size	[int]	Sets the border size.
rounding	[int]	Forces the application to have X pixels of rounding, ignoring the set default (in `decoration:rounding`). Has to be an int.

Effect	argument	Description
rounding_power	[float]	Overrides the rounding power for the window (see `decoration:rounding_power`).
allows_input	[on]	Forces an XWayland window to receive input, even if it requests not to do so. (Might fix issues like Game Launchers not receiving focus for some reason)
dim_around	[on]	Dims everything around the window. Please note that this rule is meant for floating windows and using it on tiled ones may result in strange behavior.
decorate	[on]	(default is *true*) Whether to draw window decorations or not
focus_on_activate	[on]	Whether Hyprland should focus an app that requests to be focused (an `activate` request).
keep_aspect_ratio	[on]	Forces aspect ratio when resizing window with the mouse.
nearest_neighbor	[on]	Forces the window to use nearest neighbor filtering.
no_anim	[on]	Disables the animations for the window.
no_blur	[on]	Disables blur for the window.
no_dim	[on]	Disables window dimming for the window.
no_focus	[on]	Disables focus to the window.
no_follow_mouse	[on]	Prevents the window from being focused when the mouse moves over it when `input:follow_mouse=1` is set.
no_shadow	[on]	Disables shadows for the window.
no_shortcuts_inhibit	[on]	Disallows the app from inhibiting your shortcuts.
no_screen_share	[on]	Hides the window and its popups from screen sharing by drawing black rectangles in their place. The rectangles are drawn even if other windows are above.
no_vrr	[on]	Disables VRR for the window. Only works when `misc:vrr` is set to `2` or `3`.
opaque	[on]	Forces the window to be opaque.
force_rgbx	[on]	Forces Hyprland to ignore the alpha channel on the whole window's surfaces, effectively making it *actually, fully 100% opaque*.
sync_fullscreen	[on]	Whether the fullscreen mode should always be the same as the one sent to the window (will only take effect on the next fullscreen mode change).
immediate	[on]	Forces the window to allow tearing. See the Tearing page.
xray	[on]	Sets blur xray mode for the window.
render_unfocused	[on]	Forces the window to think it's being rendered when it's not visible. See also Variables - Misc for setting `render_unfocused_fps`.
scroll_mouse	[float]	Forces the window to override the variable `input:scroll_factor`.
scroll_touchpad	[float]	Forces the window to override the variable `input:touchpad:scroll_factor`.

All dynamic effects can be set with `setprop`, see `setprop`.

`group` window rule options

- `set [always]` - Open window as a group.
- `new` - Shorthand for `barred set`.
- `lock [always]` - Lock the group that added this window. Use with `set` or `new` (e.g. `new lock`) to create a new locked group.
- `barred` - Do not automatically group the window into the focused unlocked group.
- `deny` - Do not allow the window to be toggled as or added to group (see `denywindowfromgroup` dispatcher).
- `invade` - Force open window in the locked group.
- `override` [other options] - Override other `group` rules, e.g. You can make all windows in a particular workspace open as a group, and use `group override barred` to make windows with specific titles open as normal windows.
- `unset` - Clear all `group` rules.

> Note
>
> The `group` rule without options is a shorthand for `group set`.
>
> By default, `set` and `lock` only affect new windows once. The `always` qualifier makes them always effective.

Tags

Window may have several tags, either static or dynamic. Dynamic tags will have a suffix of `*`. You may check window tags with `hyprctl clients`.

Use the `tagwindow` dispatcher to add a static tag to a window:

```
hyprctl dispatch tagwindow +code     # Add tag to current window.
hyprctl dispatch tagwindow -- -code  # Remove tag from current window (use `--` to protect the leading `-`).
hyprctl dispatch tagwindow code      # Toggle the tag of current window.

# Or you can tag windows matched with a window RegEx:
hyprctl dispatch tagwindow +music deadbeef
hyprctl dispatch tagwindow +media title:Celluloid
```

Use the `tag` rule to add a dynamic tag to a window:

```
windowrule = tag +term, match:class footclient  # Add dynamic tag `term*` to window footclient.
windowrule = tag term, match:class footclient   # Toggle dynamic tag `term*` for window footclient.
windowrule = tag +code, match:tag cpp           # Add dynamic tag `code*` to window with tag `cpp`.

windowrule = opacity 0.8, match:tag code        # Set opacity for window with tag `code` or `code*`.
windowrule = opacity 0.7, match:tag cpp         # Window with tag `cpp` will match both `code` and `cpp`, the last one will override prior m
windowrule = opacity 0.6, match:tag term*       # Set opacity for window with tag `term*` only, `term` will not be matched.

windowrule = tag -code, match:tag  term          # Remove dynamic tag `code*` for window with tag `term` or `term*`.
```

Or with a keybind for convenience:

```
bind = $mod Ctrl, 2, tagwindow, alpha_0.2
bind = $mod Ctrl, 4, tagwindow, alpha_0.4

windowrule = opacity 0.2 override, match:tag alpha_0.2
windowrule = opacity 0.4 override, match:tag alpha_0.4
```

The `tag` rule can only manipulate dynamic tags, and the `tagwindow` dispatcher only works with static tags (i.e. once the dispatcher is called, dynamic tags will be cleared).

Example Rules

```
# Move kitty to 100 100 and add an anim style
windowrule {
  name = move-kitty
  match:class = kitty

  move = 100 100
  animation = popin
}

windowrule = no_blur on, match:class firefox                                          # Disable blur for firefox
windowrule = move (cursor_x-(window_w*0.5)) (cursor_y-(window_h*0.5)), match:class kitty  # Move kitty to the center of the cursor
windowrule = border_color rgb(FF0000) rgb(880808), match:fullscreen 1                 # Set border color to red if window is fullscreen
windowrule = border_color rgb(FFFF00), match:title .*Hyprland.*                       # Set border color to yellow when title contains H
windowrule = opacity 1.0 override 0.5 override 0.8 override, match:class kitty        # Set opacity to 1.0 active, 0.5 inactive and 0.8
windowrule = match:class kitty, rounding 10                                           # Set rounding to 10 for kitty
windowrule = match:class (pinentry-)(.*), stay_focused on                              # Fix pinentry losing focus
```

Notes

Effects that are marked as *Dynamic* will be reevaluated if the matching property of the window changes.
For instance, if a rule is defined that changes the `border_color` of a window when it is floating, then the `border_color` will change to the requested color when it is set to floating, and revert to the default color when it is tiled again.

Effects will be processed from top to bottom, where the *last* match will take precedence. i.e.

```
windowrule = opacity 0.8 0.8, match:class kitty
windowrule = opacity 0.5 0.5, match:float yes
```

Here, all non-fullscreen kitty windows will have `opacity 0.8`, except if they are floating. Otherwise, they will have `opacity 0.5`. The rest of the non-fullscreen floating windows will have `opacity 0.5`.

```
windowrule = opacity 0.5 0.5, match:float true
windowrule = opacity 0.8 0.8, match:class kitty
```

Here, all kitty windows will have `opacity 0.8`, even if they are floating. The rest of the floating windows will have `opacity 0.5`.

> Important
> Named rules take precedence over anonymous ones. That is, rules are evaluated top to bottom, but all named rules get evaluated first, then all anonymous ones.

> Note
> Opacity is a PRODUCT of all opacities by default. For example, setting `active_opacity` to `0.5` and `opacity` to `0.5` will result in a total opacity of `0.25`.
> You are allowed to set opacities over `1.0`, but any opacity product over `1.0` will cause graphical glitches.
> For example, using `0.5 * 2 = 1` is fine, but `0.5 * 4 = 2` will cause graphical glitches.
> You can put `override` after an opacity value to override it to an exact value rather than a multiplier. For example, to set active and inactive opacity to 0.8, and make fullscreen windows fully opaque regardless of other opacity rules:

```
windowrule = match:class kitty, opacity 0.8 override 0.8 override 1.0 override
```

Dynamically enabling / disabling / changing rules

Only named rules can be dynamically changed, enabled or disabled. Anonymous rules cannot be.

For enabling or disabling, the keyword `enable` is exposed:

```
hyprctl keyword 'windowrule[my-rule-name]:enable false'
```

For changing properties, the same syntax can be used:

```
hyprctl keyword 'windowrule[my-rule-name]:match:class kitty'
```

the singlequotes are necessary, otherwise your shell might try to parse the string

Layer Rules

Some things in Wayland are not windows, but layers. That includes, for example: app launchers, status bars, or wallpapers.

Those have specific rules, separate from windows. Their syntax is the exact same, but they have different props and effects.

Props

Field	Argument	Description
match:namespace	[RegEx]	namespace of the layer, check `hyprctl layers`.

Effects

effect	argument	description
no_anim	[on]	Disables animations.
blur	[on]	Enables blur for the layer.
blur_popups	[on]	Enables blur for the popups.
ignore_alpha	[a]	Makes blur ignore pixels with opacity of `a` or lower. `a` is float value from `0` to `1`. `a = 0` if unspecified.
dim_around	[on]	Dims everything behind the layer.
xray	[on]	Sets the blur xray mode for a layer. `0` for off, `1` for on, `unset` for default.
animation	[style]	Allows you to set a specific animation style for this layer.

effect	argument	description
order	[n]	Sets the order relative to other layers. A higher `n` means closer to the edge of the monitor. Can be negative. `n = 0` if unspecified.
above_lock	[0/1/2]	If non-zero, renders the layer above the lockscreen when the session is locked. If set to `2`, you can interact with the layer on the lockscreen, otherwise it will only be rendered above it.
no_screen_share	[on]	Hides the layer from screen sharing by drawing a black rectangle over it.

Examples

```
layerrule = blur on, match:namespace waybar

layerrule {
  name = no_anim_for_selection
  no_anim = on
  match:namespace = selection
}
```

Last updated on January 12, 2026