NAME

dunst − A customizable and lightweight notification−daemon

SYNOPSIS

dunst [−conf file] [−font font] [−geometry geom] [−format fmt] [−follow mode] [−monitor n] [−history_length n] ...

DESCRIPTION

Dunst is a highly configurable and lightweight notification daemon.

COMMAND LINE OPTIONS

−h/−−help
List all command line flags

−conf/−config file
Use alternative config file.

−v/−−version
Print version information.

−print
Print notifications to stdout. This might be useful for logging, setting up rules or using the output in other scripts.

CONFIGURATION

An example configuration file is included (usually /usr/share/doc/dunst/dunstrc.gz). To change the configuration, copy this file to ~/.config/dunst/dunstrc and edit it accordingly.

The configuration is divided into sections in an ini-like format. The 'global' section contains most general settings while the 'shortcuts' sections contains all keyboard configuration and the 'experimental' section all the features that have not yet been tested thoroughly.

Any section that is not one of the above is assumed to be a rule, see RULES for more details.

For backwards compatibility reasons the section name 'frame' is considered bound and can't be used as a rule.

Command line

Each configuration option in the global section can be overridden from the command line by adding a single dash in front of it's name. For example the font option can be overridden by running

```
$ dunst -font "LiberationSans Mono 4"
```

Configuration options that take boolean values can only currently be set to ''true'' through the command line via the same method. e.g.

```
$ dunst -shrink
```

This is a known limitation of the way command line parameters are parsed and will be changed in the future.

Available settings per section:

Global section

monitor (default: 0)
Specifies on which monitor the notifications should be displayed in, count starts at 0. See the **follow** setting.

follow (values: [none/mouse/keyboard] default: none)
Defines where the notifications should be placed in a multi-monitor setup. All values except *none* override the **monitor** setting.

none
The notifications will be placed on the monitor specified by the **monitor** setting.

mouse
The notifications will be placed on the monitor that the mouse is currently in.

keyboard
The notifications will be placed on the monitor that contains the window with keyboard focus.

geometry (format: [{width}][x{height}][+/−{x}[+/−{y}]], default: ‘‘0x0+0−0’’)
The geometry of the window the notifications will be displayed in.

width
The width of the notification window in pixels. A negative value sets the width to the screen width **minus the absolute value of the width**. If the width is omitted then the window expands to cover the whole screen. If it’s 0 the window expands to the width of the longest message being displayed.

height
The number of notifications that can appear at one time. When this limit is reached any additional notifications will be queued and displayed when the currently displayed ones either time out or are manually dismissed. If **indicate_hidden** is true, then the specified limit is reduced by 1 and the last notification is a message informing how many hidden notifications are waiting to be displayed. See the **indicate_hidden** entry for more information.

The physical(pixel) height of the notifications vary depending on the number of lines that need to be displayed.

See **notification_height** for changing the physical height.

x/y Respectively the horizontal and vertical offset in pixels from the corner of the screen that the notification should be drawn at. For the horizontal(x) offset, a positive value is measured from the left of the screen while a negative one from the right. For the vertical(y) offset, a positive value is measured from the top while a negative from the bottom.

It’s important to note that the positive and negative sign **DOES** affect the position even if the offset is 0. For example, a horizontal offset of +0 puts the notification on the left border of the screen while a horizontal offset of −0 at the right border. The same goes for the vertical offset.

indicate_hidden (values: [true/false], default: true)
If this is set to true, a notification indicating how many notifications are not being displayed due to the notification limit (see **geometry**) will be shown **in place of the last notification slot**.

Meaning that if this is enabled the number of visible notifications will be 1 less than what is specified in geometry, the last slot will be taken by the hidden count.

shrink (values: [true/false], default: false)
Shrink window if it’s smaller than the width. Will be ignored if width is 0.

This is used mainly in order to have the shrinking benefit of dynamic width (see geometry) while also having an upper bound on how long a notification can get before wrapping.

transparency (default: 0)
A 0−100 range on how transparent the notification window should be, with 0 being fully opaque and 100 invisible.

This setting will only work if a compositor is running.

notification_height (default: 0)
The minimum height of the notification window in pixels. If the text and padding cannot fit in within the height specified by this value, the height will be increased as needed.

separator_height (default: 2)
The height in pixels of the separator between notifications, if set to 0 there will be no separating line between notifications.

padding (default: 0)
The distance in pixels from the content to the separator/border of the window in the vertical axis

horizontal_padding (default: 0)
The distance in pixels from the content to the border of the window in the horizontal axis

frame_width (default: 0)
Defines width in pixels of frame around the notification window. Set to 0 to disable.

frame_color color (default: #888888)
Defines color of the frame around the notification window. See COLORS.

separator_color (values: [auto/foreground/frame/#RRGGBB] default: auto)
Sets the color of the separator line between two notifications.

auto
Dunst tries to find a color that fits the rest of the notification color scheme automatically.

foreground
The color will be set to the same as the foreground color of the topmost notification that's being separated.

frame
The color will be set to the frame color of the notification with the highest urgency between the 2 notifications that are being separated.

anything else
Any other value is interpreted as a color, see COLORS

sort (values: [true/false], default: true)
If set to true, display notifications with higher urgency above the others.

idle_threshold (default: 0)
Don't timeout notifications if user is idle longer than this time. See TIME FORMAT for valid times.

Set to 0 to disable.

A client can mark a notification as transient to bypass this setting and timeout anyway. Use a rule with 'set_transient = no' to disable this behavior.

font (default: ''Monospace 8'')
Defines the font or font set used. Optionally set the size as a decimal number after the font name and space. Multiple font options can be separated with commas.

This options is parsed as a Pango font description.

line_height (default: 0)
The amount of extra spacing between text lines in pixels. Set to 0 to disable.

markup (values: [full/strip/no], default: no)
Defines how markup in notifications is handled.

It's important to note that markup in the format option will be parsed regardless of what this is set to.

Possible values:

full Allow a small subset of html markup in notifications

```
<b>bold</b>
<i>italic</i>
<s>strikethrough</s>
<u>underline</u>
```

For a complete reference see <http://developer.gnome.org/pango/stable/PangoMarkupFormat.html>

strip
This setting is provided for compatibility with some broken clients that send markup even though it's not enabled on the server.

Dunst will try to strip the markup but the parsing is simplistic so using this option outside of matching rules for specific applications **IS GREATLY DISCOURAGED**.

See RULES

no Disable markup parsing, incoming notifications will be treated as plain text. Dunst will not advertise that it can parse markup if this is set as a global setting.

format (default: "%s %b")
Specifies how the various attributes of the notification should be formatted on the notification window.

Regardless of the status of the **markup** setting, any markup tags that are present in the format will be parsed. Note that because of that, if a literal ampersand (&) is needed it needs to be escaped as '&'

If '\n' is present anywhere in the format, it will be replaced with a literal newline.

If any of the following strings are present, they will be replaced with the equivalent notification attribute.

%a appname
%s summary
%b body
%i iconname (including its path)
%I iconname (without its path)
%p progress value ([0%] to [100%])
%n progress value without any extra characters
%% Literal %

If any of these exists in the format but hasn't been specified in the notification (e.g. no icon has been set), the placeholders will simply be removed from the format.

alignment (values: [left/center/right], default: left)
Defines how the text should be aligned within the notification.

show_age_threshold (default: −1)
Show age of message if message is older than this time. See TIME FORMAT for valid times.

Set to −1 to disable.

word_wrap (values: [true/false], default: false)
Specifies how very long lines should be handled

If it's set to false, long lines will be truncated and ellipsized.

If it's set to true, long lines will be broken into multiple lines expanding the notification window height as necessary for them to fit.

ellipsize (values: [start/middle/end], default: middle)
If word_wrap is set to false, specifies where truncated lines should be ellipsized.

ignore_newline (values: [true/false], default: false)
If set to true, replace newline characters in notifications with whitespace.

stack_duplicates (values: [true/false], default: true)
If set to true, duplicate notifications will be stacked together instead of being displayed separately.

Two notifications are considered duplicate if the name of the program that sent it, summary, body, icon and urgency are all identical.

hide_duplicates_count (values: [true/false], default: false)
Hide the count of stacked duplicate notifications.

show_indicators (values: [true/false], default: true)
Show an indicator if a notification contains actions and/or open-able URLs. See ACTIONS below for further details.

icon_position (values: [left/right/off], default: off)
Defines the position of the icon in the notification window. Setting it to off disables icons.

max_icon_size (default: 0)
Defines the maximum size in pixels for the icons. If the icon is smaller than the specified value it won't be affected. If it's larger then it will be scaled down so that the larger axis is equivalent to the specified size.

Set to 0 to disable icon scaling. (default)

If **icon_position** is set to off, this setting is ignored.

icon_path (default: ''/usr/share/icons/gnome/16x16/status/:/usr/share/icons/gnome/16x16/devices/'')
Can be set to a colon-separated list of paths to search for icons to use with notifications.

Dunst doesn't currently do any type of icon lookup outside of these directories.

sticky_history (values: [true/false], default: true)
If set to true, notifications that have been recalled from history will not time out automatically.

history_length (default: 20)
Maximum number of notifications that will be kept in history. After that limit is reached, older notifications will be deleted once a new one arrives. See HISTORY.

dmenu (default: ''/usr/bin/dmenu'')
The command that will be run when opening the context menu. Should be either a dmenu command or a dmenu-compatible menu.

browser (default: ''/usr/bin/firefox'')
The command that will be run when opening a URL. The URL to be opened will be appended to the end of the value of this setting.

always_run_script (values: [true/false] default: true]
Always run rule-defined scripts, even if the notification is suppressed with format = "". See SCRIPTING.

title (default: ''Dunst'')
Defines the title of notification windows spawned by dunst. (_NET_WM_NAME property). There should be no need to modify this setting for regular use.

class (default: ''Dunst'')
Defines the class of notification windows spawned by dunst. (First part of WM_CLASS). There should be no need to modify this setting for regular use.

startup_notification (values: [true/false], default: false)
Display a notification on startup. This is usually used for debugging and there shouldn't be any need to use this option.

verbosity (values: 'crit', 'warn', 'mesg', 'info', 'debug' default 'mesg')
Do not display log messages, which have lower precedence than specified verbosity. This won't affect printing notifications on the terminal. Use the '−print' option for this.

force_xinerama (values: [true/false], default: false)
Use the Xinerama extension instead of RandR for multi-monitor support. This setting is provided for compatibility with older nVidia drivers that do not support RandR and using it on systems that support RandR is highly discouraged.

By enabling this setting dunst will not be able to detect when a monitor is connected or disconnected

which might break follow mode if the screen layout changes.

corner_radius (default: 0)
Define the corner radius in pixels. A corner radius of 0 will result in rectangular shaped notifications.

By enabling this setting the outer border and the frame will be shaped. If you have multiple notifications, the whole window is shaped, not every single notification.

To avoid the corners clipping the icon or text the corner radius will be automatically lowered to half of the notification height if it exceeds it.

mouse_left/middle/right_click (values: [none/do_action/close_current/close_all])
Defines action of mouse click.

none
Don't do anything.

do_action (default for mouse_middle_click)
If the notification has exactly one action, or one is marked as default, invoke it. If there are multiple and no default, open the context menu.

close_current (default for mouse_left_click)
Close current notification.

close_all (default for mouse_right_click)
Close all notifications.

Shortcut section
Keyboard shortcuts are defined in the following format: "Modifier+key" where the modifier is one of ctrl,mod1,mod2,mod3,mod4 and key is any keyboard key.

close
command line flag: –key <key>

Specifies the keyboard shortcut for closing a notification.

close_all
command line flag: –all_key <key>

Specifies the keyboard shortcut for closing all currently displayed notifications.

history
command line flag: –history_key <key>

Specifies the keyboard shortcut for recalling a single notification from history.

context
command line flag: –context_key <key>

Specifies the keyboard shortcut that opens the context menu.

Urgency sections
The urgency sections work in a similar way to rules and can be used to specify attributes for the different urgency levels of notifications (low, normal, critical). Currently only the background, foreground, timeout, frame_color and icon attributes can be modified.

The urgency sections are urgency_low, urgency_normal, urgency_critical for low, normal and critical urgency respectively.

See the example configuration file for examples.

Additionally, you can override these settings via the following command line flags:

Please note these flags may be removed in the future. See issue #328 in the bug tracker for discussions (See REPORTING BUGS).

–li/ni/ci icon
Defines the icon for low, normal and critical notifications respectively.

Where *icon* is a path to an image file containing the icon.

–lf/nf/cf color
Defines the foreground color for low, normal and critical notifications respectively.

See COLORS for the value format.

–lb/nb/cb color
Defines the background color for low, normal and critical notifications respectively.

See COLORS for the value format.

–lfr/nfr/cfr color
Defines the frame color for low, normal and critical notifications respectively.

See COLORS for more information

–lto/nto/cto secs
Defines the timeout time for low, normal and critical notifications respectively. See TIME FORMAT for valid times.

HISTORY

Dunst saves a number of notifications (specified by **history_length**) in memory. These notifications can be recalled (i.e. redisplayed) by pressing the **history_key** (see the shortcuts section), whether these notifications will time out like if they have been just send depends on the value of the **sticky_history** setting.

Past notifications are redisplayed in a first-in-last-out order, meaning that pressing the history key once will bring up the most recent notification that had been closed/timed out.

RULES

Rules allow the conditional modification of notifications. They are defined by creating a section in the configuration file that has any name that is not already used internally (i.e. any name other than 'global', 'experimental', 'frame', 'shortcuts', 'urgency_low', 'urgency_normal' and 'urgency_critical').

There are 2 parts in configuring a rule: Defining the filters that control when a rule should apply and then the actions that should be taken when the rule is matched.

filtering
Notifications can be matched for any of the following attributes:

`appname` (discouraged, see desktop_entry)
The name of the application as reported by the client. Be aware that the name can often differ depending on the locale used.

`body`
The body of the notification

`category`
The category of the notification as defined by the notification spec. See https://developer.gnome.org/notification−spec/#categories

`desktop_entry`
GLib based applications export their desktop-entry name. In comparison to the appname, the desktop-entry won't get localized.

`icon`
The icon of the notification in the form of a file path. Can be empty if no icon is available or a raw icon is used instead.

`match_transient`
Match if the notification has been declared as transient by the client or by some other rule.

See `set_transient` for more details about this attribute.

`msg_urgency`
Matches the urgency of the notification as set by the client or by some other rule.

`stack_tag`
Matches the stack tag of the notification as set by the client or by some other rule.

See set_stack_tag for more information about stack tags.

`summary`
Matches the summary, 'title', of the notification.

`msg_urgency` is the urgency of the notification, it is named so to not conflict with trying to modify the urgency.

Instead of the appname filter, it's recommended to use the desktop_entry filter.

To define a matching rule simply assign the specified value to the value that should be matched, for example:

```
appname="notify-send"
```

Matches only messages that were send via notify-send. If multiple filter expressions are present, all of them have to match for the rule to be applied (logical AND).

Shell-like globing is supported.

modifying
The following attributes can be overridden:

`background`
The background color of the notification. See COLORS for possible values.

`foreground`
The background color of the notification. See COLORS for possible values.

`format`
Equivalent to the `format` setting.

`frame_color`
The frame color color of the notification. See COLORS for possible values.

`fullscreen`
One of show, delay, or pushback.

This attribute speicifies how notifications are handled if a fullscreen window is focused. By default it's set to show so notifications are being shown.

Other possible values are delay: Already shown notifications are continued to be displayed until they are dismissed or time out but new notifications will be held back and displayed when the focus to the fullscreen window is lost.

Or pushback which is equivalent to delay with the difference that already existing notifications are paused and hidden until the focus to the fullscreen window is lost.

`new_icon`
Updates the icon of the notification, it should be a path to a valid image.

`set_stack_tag`
Sets the stack tag for the notification, notifications with the same (non-empty) stack tag will replace each-other so only the newest one is visible. This can be useful for example in volume or brightness notifications where only want one of the same type visible.

The stack tag can be set by the client with the 'synchronous', 'private−synchronous' 'x−canonical−private−synchronous' or the 'x−dunst−stack−tag' hints.

`set_transient`
Sets whether the notification is considered transient. Transient notifications will bypass the idle_threshold setting.

By default notifications are _not_ considered transient but clients can set the value of this by specifying the 'transient' hint when sending notifications.

`timeout`
Equivalent to the `timeout` setting in the urgency sections.

`urgency`
This sets the notification urgency.

IMPORTANT NOTE: This currently DOES NOT re-apply the attributes from the urgency_* sections. The changed urgency will only be visible in rules defined later. Use `msg_urgency` to match it.

`skip_display`
Setting this to true will prevent the notification from being displayed initially but will be saved in history for later viewing.

As with the filtering attributes, each one corresponds to the respective notification attribute to be modified.

As with filtering, to make a rule modify an attribute simply assign it in the rule definition.

If the format is set to an empty string, the notification will not be suppressed.

SCRIPTING

Within rules you can specify a script to be run every time the rule is matched by assigning the 'script' option to the name of the script to be run.

When the script is called details of the notification that triggered it will be passed via command line parameters in the following order: appname, summary, body, icon, urgency.

Where icon is the absolute path to the icon file if there is one and urgency is one of ''LOW'', ''NORMAL'' or ''CRITICAL''.

If the notification is suppressed, the script will not be run unless **always_run_scripts** is set to true.

If '~/' occurs at the beginning of the script parameter, it will get replaced by the users' home directory. If the value is not an absolute path, the directories in the PATH variable will be searched for an executable of the same name.

COLORS

Colors are interpreted as X11 color values. This includes both verbatim color names such as ''Yellow'', ''Blue'', ''White'', etc as well as #RGB and #RRGGBB values.

NOTE: '#' is interpreted as a comment, to use it the entire value needs to be in quotes like so: separator_color=''#123456''

NOTIFY-SEND

dunst is able to get different colors for a message via notify-send. In order to do that you have to add a hint via the −h option. The progress value can be set with a hint, too.

notify-send −h string:fgcolor:#ff4444
notify-send −h string:bgcolor:#4444ff −h string:fgcolor:#ff4444 −h string:frcolor:#44ff44
notify-send −h int:value:42 ''Working ...''

ACTIONS

Dunst allows notifiers (i.e.: programs that send the notifications) to specify actions. Dunst has support for both displaying indicators for these, and interacting with these actions.

If ‘‘show_indicators’’ is true and a notification has an action, an ‘‘(A)’’ will be prepended to the notification format. Likewise, an ‘‘(U)’’ is prepended to notifications with URLs. It is possible to interact with notifications that have actions regardless of this setting, though it may not be obvious which notifications HAVE actions.

The ‘‘context’’ keybinding is used to interact with these actions, by showing a menu of possible actions. This feature requires ‘‘dmenu’’ or a dmenu drop-in replacement present.

Alternatively, you can invoke an action with a middle click on the notification. If there is exactly one associated action, or one is marked as default, that one is invoked. If there are multiple, the context menu is shown. The same applies to URLs when there are no actions.

TIME FORMAT

A time can be any decimal integer value suffixed with a time unit. If no unit given, seconds (‘‘s’’) is taken as default.

Time units understood by dunst are ‘‘ms’’, ‘‘s’’, ‘‘m’’, ‘‘h’’ and ‘‘d’’.

Example time: ‘‘1000ms’’ ‘‘10m’’

MISCELLANEOUS

Dunst can be paused by sending a notification with a summary of ‘‘DUNST_COMMAND_PAUSE’’, resumed with a summary of ‘‘DUNST_COMMAND_RESUME’’ and toggled with a summary of ‘‘DUNST_COMMAND_TOGGLE’’. Alternatively you can send SIGUSR1 and SIGUSR2 to pause and unpause respectively. For Example:

killall −SIGUSR1 dunst # pause
killall −SIGUSR2 dunst # resume

When paused dunst will not display any notifications but keep all notifications in a queue. This can for example be wrapped around a screen locker (i3lock, slock) to prevent flickering of notifications through the lock and to read all missed notifications after returning to the computer.

FILES

`$XDG_CONFIG_HOME`/dunst/dunstrc

−or−

`$HOME`/.config/dunst/dunstrc

AUTHORS

Written by Sascha Kruse <knopwob@googlemail.com>

REPORTING BUGS

Bugs and suggestions should be reported on GitHub at https://github.com/dunst−project/dunst/issues

COPYRIGHT

SEE ALSO

dwm (1), **dmenu** (1), **twmn** (1), **notify−send** (1)