Configuration

With a basic knowledge of Qt, you could change the configuration of the default Kvantum theme. That configuration can be easily copied by using **Kvantum Manager** (click *Save* button on its third page) or, manually and if the source of Kvantum is available, by following these steps:

(1) Create the folder *"~/.config/Kvantum/"* (~ is your home);

(2) Create the file *"kvantum.kvconfig"* in the above folder with this line in it:

theme=***DefaultCopy***

Here, ***DefaultCopy*** could be any name you choose for the new configuration;

(3) Create the folder *"~/.config/Kvantum/**DefaultCopy**/"* and the file *"**DefaultCopy**.kvconfig"* in it;

(4) Copy/paste the contents of *"style/themeconfig/default.kvconfig"* (from the source) to the file *"**DefaultCopy** .kvconfig"*.

Now, you can change the values of variables (keys). Please note that deleting a variable often means that its value will be taken from the default configuration, so that you could keep only those sections or variables you want to change. See below for a more accurate explanation.

There are many sections (groups) and variables (keys) in the config file. That is intentional: unlike most theme engines, Kvantum is supposed to be able to control virtually all aspects of widgets. Here are the meanings of various sections:

Sections Table

Section (Group)	Meaning
[%General]	General info on the theme and some general variables. ("%" *is required* because Qt uses [General] for another purpose.)
[GeneralColors]	The most important colors used by the theme.
[Hacks]	Hacks for specific apps or widgets.
[PanelButtonCommand]	Panel for a button used to initiate an action, for example, a push button.
[PanelButtonTool]	Panel for a tool button.

[Dock]	A dock widget.
[DockTitle]	The title of a dock widget.
[IndicatorSpinBox]	Indicators of a spin widget.
[RadioButton]	A radio button.
[CheckBox]	A check box.
[Focus]	Generic focus frame/rectangle, drawn on a widget that has keyboard focus.
[GenericFrame]	Generic frame. Mostly drawn around item/text views but some apps may draw it in other places too (see the key *remove_extra_frames* in *General* section).
[LineEdit]	A line edit (one-line text editor).
[DropDownButton]	Indicator for a drop down button, for example, a tool button that displays a menu.
[ToolboxTab]	Just text colors for tab labels of a toolbox.
[Tab]	The tab shape within a tab bar. Also the tear indicator of a tab bar and the close button of a tab.
[TabFrame]	The frame for tab widgets.
[TabBarFrame]	The frame that is drawn for a tab bar, ususally for a tab bar that isn't part of a tab widget.
[TreeExpander]	Indicators used to represent the branch of a tree in a tree view.
[HeaderSection]	A header section. Also its label and arrow.
[SizeGrip]	Window resize handle if it exists.
[Toolbar]	A toolbar. Also its handle and separator.
[ToolbarButton]	Panel for a tool button on a stylable toolbar. This section is optional and can be omitted. If present, it will take all of its variables from *PanelButtonTool*, except for its text colors, text shadow, and (indicator, frame and interior) elements.
[ToolbarComboBox]	Exactly like *ToolbarButton* but for combo boxes on stylable toolbars.
[ToolbarLineEdit]	Panel for a line-edit on a stylable toolbar. This section is optional and can be omitted. If present, it will take all of its variables from *LineEdit*, except for its frame and interior elements. Also, note that the text color of such a line-edit is always the text color of the stylable toolbar behind it. Therefore, its interior (if any) should have a good contrast with that color.

[Scrollbar]	Scrollbar increase/decrease indicators (arrows).
[ScrollbarGroove]	The groove of a scrollbar.
[ScrollbarSlider]	A scrollbar slider.
[ScrollbarTransientSlider]	An optional section for transient scrollbars (see *transient_scrollbar* below). It takes all of its values from *ScrollbarSlider*, except for its frame and interior elements. It is useful when the ordinary scrollbar slider elements are not good enough with transient scrollbars.
[Slider]	A slider (a classic widget for controlling a bounded value).
[SliderCursor]	The handle of a slider.
[Progressbar]	The groove and label of a progressbar.
[ProgressbarContents]	The progress indicator.
[ItemView]	An item in an item view.
[Splitter]	A splitter handle.
[Menu]	The panel and frame of a menu. Also its frame shadow.
[MenuItem]	A menu item in a menu. Also the tear-off section of a menu.
[MenuBar]	The empty area of a menu bar.
[MenuBarItem]	A menu bar item, like the buttons in a menubar.
[TitleBar]	A title bar, like those used in QMdiSubWindow.
[ComboBox]	A combo box and its label.
[GroupBox]	A group box and the frame around it.
[ToolTip]	The panel for a tooltip label.
[Window]	A window or dialog.
[WindowTranslucent]	This is used when a distinction is needed to be made between (backgrounds of) opaque and translucent windows. If it is omitted, the above section will be used for all windows.
[Dialog]	This is used when a distinction is needed to be made between (backgrounds of) dialogs and windows. If it is omitted, the section *Window* will be used for both windows and dialogs. Here, "dialog" means any window without menubar and toolbar but not necessarily a QDialog.
[DialogTranslucent]	Like *WindowTranslucent* but for dialogs.

The following table shows the variables (keys) you could change to configure the current theme – without necessarily making a new one – with the default values of some of them. These are the rules for the value inheritance:

(1) If a section (group) is not present in your configuration, its variables and their values will be taken from the default config file.

(2) If a variable is not present in a section of your configuration:
   (2a) First the *"inherits"* section will be searched for it and then, if nothing is found,
   (2b) its value will be taken from the same section of the default config file.

There are three exceptions to these rules:

Exception No.1: The *"inherits"* variable will not be taken from the default config file if it is not present in a section.

Exception No.2: If colors are omitted or not valid or if a section they could belong to is not present, they will be taken from the currently used color palette. Also, font boldness/italicity will be ignored if omitted.

Exception No.3: Any variable related to compositing or hacking and also *frame.expandedElement* will be ignored if omitted.

Variables Table

Variable (Key)	Value	Meaning
The %General Section		
author	String	Obvious.
comment	String	Obvious.
respect_DE	true/false	Should some settings of the current DE be respected? True by default. **KDE**, **Unity**, **Gnome** and **Pantheon** are supported. Under KDE, the keys *small_icon_size* and *large_icon_size* are overridden by KDE settings. Under Unity, Gnome and Pantheon, *iconless_pushbutton* and *iconless_menu* are set to *true*, *composite* is set to *false*, and *x11drag* is set to *menubar_and_primary_toolbar*.
x11drag	true/false or String	Drag windows from anywhere possible? True by default. Its values are *none* (or *false*), *menubar*,

		menubar_and_primary_toolbar, and *all* (or *true*). *x11drag* is disabled on wayland.
alt_mnemonic	true/false	Show underlines when Alt is pressed? True by default.
click_behavior	Integer	How to activate view items? They are activated according to the current DE's setting when the value of this key is 0, which is the default. 1 or 2 mean single and double click respectively. Any other value means 0.
inline_spin_indicators	true/false	Draw spin indicators inside the spin line-edit, without drawing any spin button? By default, they are drawn on adjacent buttons.
inline_spin_separator	true/false	If the spin box has separator SVG elements, they will be drawn between its line-edit and horizontal buttons (See *Theme-Making.pdf*). However, if this key is true, the separator will also be drawn when there is no button, i.e. when *inline_spin_indicators* is true. Usually, *inline_spin_indicators* is set to true to integrate the indicators into the line-edit and so, the default value of this key is false.
vertical_spin_indicators	true/false	Draw spin indicators vertically and inside the spin line-edit? By default, they are drawn on adjacent buttons. **Note:** If this key is set to true, the keys *inline_spin_indicators* and *inline_spin_separator* will have no effect.
spin_button_width	Integer	The width of horizontal spin buttons. It is 16px by default and is always between 16 and 32px.
combo_as_lineedit	true/false	Draw an editable combo box as a line-edit with an arrow (and an icon if any)? By default, editable combo boxes consist of combo, line-edit and drop down elements.
combo_menu	true/false	Should the popup of combo boxes be styled as per popup menus (i.e. like Gtk)? By default, they are shown below the combo box.

combo_focus_rect	true/false	Should combo boxes have focus rectangles? The default is *false* because combo boxes are drawn by the "pressed" SVG elements when they have the keyboard focus but, if set to *true*, this key draws them with the "normal" SVG elements and adds a focus rectangle to them in that state. (The focus rectangle is defined under the *Focus* section.)
square_combo_button	true/false	Should the combo arrow buttons be square as far as possible? The default is false but if this key has a true value, editable combo boxes will be drawn as line-edits attached to square arrow buttons. Note that this key will have no effect if *combo_as_lineedit* is true because, in that case, the whole editable combo box will be drawn as a line-edit.
left_tabs	true/false	Align tabs to the left edge? Tabs are centered by default.
center_doc_tabs	true/false	Always center tabs if the tab widget is in the document mode and even when *left_tabs* is true? False by default, which means that tabs are aligned in the document mode as in the usual mode.
attach_active_tab	true/false	Attach the active tab to the tab widget or the tab-bar base? It is detached by default.
embedded_tabs	true/false	If *attach_active_tab* is false, should tabs be half embedded in their widgets (provided that it is not in the document mode)? This is false by default.
joined_inactive_tabs	true/false	Join inactive tabs together? They are joined by default. Also, see *Theme-Making.pdf* for tab separators.
mirror_doc_tabs	true/false	By default, bottom and right tab shapes are mirror images of top and left ones, respectively. Setting this key to false will change that behavior if the tab widget is in the document mode or if the active tab is detached (i.e. the value of the key *attach_active_tab* is false), so that the top/left and bottom/ right tab shapes will be identical. The default value is true.

no_active_tab_separator	true/false	Do not draw tab separator for the active tab? False by default, which means that, if tab separators exist in the SVG image, they will be drawn for all tabs. See *Theme-Making.pdf* for an explanation of tab separators.
active_tab_overlap	Integer or DECIMALfont	Number of pixels inactive tabs overlap the active one. It is zero by default. The active tab is drawn in front of its adjacent inactive tabs when the value of this key is positive. It is safe to set a great value here. The string "font" can be appended to the value (without quotes and space), in which case, the value will be multiplied by the height of the application font (in px) and could also be a decimal (as in ***active_tab_overlap=2.5font***).
no_inactive_tab_expansion	true/false	If tabs have frame expansion, setting this key to true will disable frame expansion for inactive tabs.
tab_button_extra_margin	Integer	The extra margin between the tab frame and tab buttons (the close button, for example). It is zero by default and its maximum is the font height (because some apps may not respect a greater value). Note that there is a default minimum margin without this key. The string "font" can be appended to the value (without quotes and space), in which case, the value will be multiplied by the height of the application font (in px) and could also be a decimal (as in ***tab_button_extra_margin=0.25font***).
bold_active_tab	true/false	Should the active tab text be bold? By default, the active tab text is like the inactive one.
group_toolbar_buttons	true/false	Raise and group neighbor toolbar buttons? By default, they are not raised.

toolbar_item_spacing	Integer	The space between toolbar buttons. Zero by default. It is also zero if the key *group_toolbar_buttons* is set to true.
toolbar_interior_spacing	Integer	The space around the interior of toolbars. Zero by default.
toolbar_separator_thickness	Integer	The thickness of toolbar separators. It is never less than the toolbar indicator size or 4, whichever is greater.
center_toolbar_handle	true/false	If *true*, the SVG element for the toolbar handle will be centered and its size will be that of the toolbar indicator. Otherwise, it will be scaled vertically with an 8-px width. False by default.
slim_toolbars	true/false	When true, the size of toolbar icons will be 16px if it is not set in the app. If false, the size will be determined by the DE or the app. False by default. **Note:** *toolbar_icon_size* will take priority over this key if existing.
toolbutton_style	Integer	Sets the toolbutton style when it is not set by the app. 0: follow, 1: icon only, 2: text only, 3: text beside icon, and 4: text under icon. The toolbutton style is 0 by default.
dialog_button_layout	Integer	The layout of dialog buttons. 0: follow, 1: KDE layout, 2: Gnome layout, 3: Mac layout, 4: Windows layout, and 5: Android layout. The default is 0.
spread_progressbar	true/false	Spread the progressbar's indicator across its whole groove and not just its interior? By default, the indicator is drawn inside progressbar's frame.
progressbar_thickness	Integer or DECIMALfont	If positive, it sets the (maximum) progressbar thickness as far as possible. It is zero by default, which means there is no limit to progressbar thickness. **Note 1:** If it is positive but less than the height of the progressbar text, the text will be put above the bar or, when there is not enough space above the bar, in front of it.

		Note 2: The string "font" can be appended to the value (without quotes and space), in which case, the value will be multiplied by the height of the application font (in px) and could also be a decimal (as in ***progressbar_thickness=2.5font***).
spread_header	true/false	Spread the header's interior so that it meets the frame of its view as far as possible (good for drawing simple headers). Its value is false by default but if you set it to true, do not remove top, right or left header frames because they may be needed when the header is RTL vertical or not stretched.
menubar_mouse_tracking	true/false	Enable mouse tracking in menubars? It is enabled by default.
merge_menubar_with_toolbar	true/false	Draw adjacent menu and tool bars as a whole? If *true*, the toolbar SVG interior and frame will be used for drawing them.
spread_menuitems	true/false	By default, menu-items are drawn inside the frames of their menu. This key spreads them horizontally, so that they cover the left and right frames of the menu but not its shadow. Also, see *Theme-Making.pdf → Translucency and Shadow for Menus and Tooltips*.
composite	true/false	Use compositing to have translucent menus or tooltips? It is automatically set to false if no compositing is available. Its absence also means false.
scrollable_menu	true/false	Should big menus be scrollable and have scroll arrows on their top and/or bottom? True by default. With a false value, all menu-items are shown.
menu_separator_height	Integer	The height of menu separators. 10 by default and always between 1 and 16 px.
submenu_overlap	Integer	The horizontal overlap between a submenu and its parent menu. The default is 0. It cannot be greater than 16px.
menu_shadow_depth	Integer	The depth of the shadow menus cast. A value of zero, its absence or a false value for *composite* means no shadow.

		Also, see *Theme-Making.pdf → Translucency and Shadow for Menus and Tooltips*.
tooltip_shadow_depth	Integer	The depth of the shadow tooltips cast. A value of zero, its absence or a false value for *composite* means no shadow.
translucent_windows	true/false	Translucent windows and dialogs? This requires a translucent SVG element for windows or a positive value for *reduce_window_opacity*. A false value, its absence or a false value for *composite* means no translucency. Window translucency is also disabled automatically when there is neither a *Window* nor a *WindowTranslucent* section or the value of *interior* is false for them and, at the same time, *reduce_window_opacity* is zero.
reduce_window_opacity	Integer	If *translucent_windows* is set to true, this key will reduce the window opacity by the percentage of its value regardless of whether a translucent element for window background exists in the SVG image or not. The default value is 0 and the maximum is 90.
reduce_menu_opacity	Integer	As above but for menus.
opaque	String list	A comma-separated list of executables, whose apps should not have window translucency. It has meaning only if *translucent_windows* is set to true.
blurring	true/false	Blur the screen area behind windows that are made translucent by Kvantum? This needs KDE blur effect and also a graphic card that supports it. It has no effect when *composite* or *translucent_windows* is false.
popup_blurring	true/false	Blur the regions behind menus and tooltips that are made translucent by Kvantum? This needs KDE blur effect and a graphic card that supports it. It will automatically be set to true if *blurring* is true.
shadowless_popup	true/false	Should the shadows that Kvantum adds to menus and tooltips with compositing be removed?

		May be useful under a DE that gives shadow to tooltips and menus.
contrast intensity saturation	Decimal	If not equal to 1, the values of these keys are used by the (KDE) *contrast effect* (when enabled) for translucent windows that support the blur effect, although the presence of the (KDE) blur effect is not needed. Their default value is 1 and they are always between 0 and 2.
animate_states	true/false	Animate the state change in some widgets, especially when they are under the mouse cursor? Widgets like buttons may have normal, focused (hover), pressed and toggled states. If this key is set to true, there will be a 200-ms fading animation on state change for some widgets. The default value is false. These widgets are supported: all kinds of buttons, combo boxes, line-edits and scroll views. For buttons, the animation happens only under the mouse cursor.
no_inactiveness	true/false	Ignore the inactive state? The purpose of this key is more flexibility with themes that distinguish between active and inactive states.
no_window_pattern	true/false	Do not draw window/dialog tiling patterns? The purpose of this key is more flexibility with themes that include tiling patterns for windows and/or dialogs.
splitter_width	Integer	The width of splitter handles. It cannot be greater than 32px and is 7px by default.
scroll_width	Integer	The thickness of scrollbars. It cannot be greater than 32px. The default value is 12px.
scroll_min_extent	Integer	The minimum height of a vertical scrollbar slider and the minimum width of a horizontal one. It cannot be greater that 100px or less than 16px. The default value is 36px.
center_scrollbar_indicator	true/false	By default, the decorative indicator of (vertical) *ScrollbarSlider* is scaled to the width of the slider interior and only its height is *indicator.size*. If *true*, this key tries to set both width and height to *indicator.size*.

scroll_arrows	true/false	Draw scrollbar add-line and sub-line arrows? True by default. If set to false, it will remove scroll arrows as far as possible but some apps might still force scroll arrows.
scrollbar_in_view	true/false	Should scrollbars be inside the view frame? It is false (outside) by default.
transient_scrollbar	true/false	Should scrollbars appear only when needed and disappear when not needed? False by default. If this key is true, *scroll_arrows* and *scrollbar_in_view* are considered false, the SVG groove is not drawn, scrollbars fade out when not needed, and they usually take no extra space.
transient_groove	true/false	Should transient scrollbars have translucent backgrounds when needed? False by default but if *true*, the base color will be used for painting the background with 25% translucency.
tree_branch_line	true/false	Draw tree branch lines? False by default. If this key is true, the color of branch lines will be *"light.color"* for dark themes and *"dark.color"* for light ones.
groupbox_top_label	true/false	Draw the group-box label above the top frame? It is false by default, which means that the label is drawn *on* the top frame. If the value of this key is false, group boxes will not have interior or frame expansion.
button_contents_shift	true/false	Shift the contents of push-buttons when they are down? It is true by default.
slider_width	Integer	The width of sliders. It cannot be greater than 48px and is 8px by default.
slider_handle_width slider_handle_length	Integer	The width and the height of slider handles. They cannot be greater than 48px. The default values are 16px.
tickless_slider_handle_size	Integer	The size (width and height) of the special slider handle used for sliders without tick marks, if the interior element of such a handle exists in the SVG file (with "-tickless" appended to its element

		name; See *Theme-Making.pdf* → Indicators). Its value is never greater than *slider_handle_width*, which is also its default value.
check_size	Integer	The width and height of checkboxes and radio buttons. The default value is 13px.
tooltip_delay	Integer	The delay, in milliseconds, before a tooltip is shown. If set to zero, tooltips will be shown instantly. If missing or set to any negative integer, the default behavior of Qt is used for showing tooltips. (This key has *no* effect with Qt4!)
submenu_delay	Integer	The delay, in milliseconds, before opening a submenu. 250 by default. -1 means opening submens only by clicking, while 0 means showing them immediately.
layout_spacing	Integer	Obvious. Its value is between 2 and 16 (px), and will have no effect if set in the code of an app. It is 2 by default.
layout_margin	Integer	Obvious. Its value is between 2 and 16 (px), and will have no effect if set in the code of an app. It is 4 by default.
small_icon_size large_icon_size button_icon_size	Integer	These affect menu-items/headers, icon-views and buttons/tabbars/listviews respectively. KDE setting will have priority over these values if it exists.
toolbar_icon_size	Integer/String	The icons size of toolbar buttons (KDE setting will have priority if it exists). Its value can also be "font", in which case, the size of toolbar icons will be set to the application font height if possible. That may be useful for aligning textless toolbar buttons with others.
fill_rubberband	true/false	Always fill the rubber-band rectangle with the highlight color? By default, drop rectangles for movable toolbars and dock widgets are hollow.
remove_extra_frames	true/false	Some apps may draw extra generic frames around some contents. They will be removed if this key is set to true. Its default value is false.
dark_titlebar	true/false	Should a dark titlebar be requested under Gtk desktops like Gnome? Although the default is *false,* it is good to set this key to *true* for dark

		themes.
The *GeneralColors* Section		
window.color	String (#RRGGBBAA)	A general background color as #RRGGBB or #RRGGBBAA or with a valid name like white, black, red, etc. If it has translucency, the keys *composite* and *translucent_windows* should be set to true under the *General* section and compositing should be available for windows to really look translucent.
inactive.window.color	String (#RRGGBBAA)	Background color of inactive windows. If missing, it falls backs to *window.color*.
base.color	String (#RRGGBBAA)	Used mostly as the background color for text entry widgets. It should not have a high contrast with *window.color*.
inactive.base.color	String (#RRGGBBAA)	Background color of inactive text entry widgets. If missing, it falls backs to *base.color*.
alt.base.color	String (#RRGGBBAA)	Used as the alternate background color in views with alternating row colors. It should not have a high contrast with *base.color*. If it has translucency, it will be applied over *base.color*.
inactive.alt.base.color	String (#RRGGBBAA)	Like above but for inactive widgets.
button.color	String (#RRGGBBAA)	The general button background color (usually, the same as *window.color)*.
light.color	String (#RRGGBBAA)	Lighter than *button.color*. It is used in 3D bevels and also in drawing toolbox tabs.
mid.light.color	String (#RRGGBBAA)	Between *button.color* and *light.color*. It is the color of box frames.
dark.color	String (#RRGGBBAA)	Darker than *button.color* (used mostly for 3D bevels).
mid.color	String (#RRGGBBAA)	Between *button.color* and *dark.color*. It is the color of table grid lines but is also used in drawing toolbox tabs.
shadow.color	String (#RRGGBBAA)	A very dark color. By default, it is black (can be ignored).
highlight.color	String (#RRGGBBAA)	A color for text selection.

inactive.highlight.color	String (#RRGGBBAA)	Like *highlight.color* but when the text widget does not have focus.
tooltip.base.color	String (#RRGGBBAA)	Tooltip background color (used in "WhatsThis" tooltips).
text.color	String (#RRGGBBAA)	The foreground color used with *base.color*.
inactive.text.color	String (#RRGGBBAA)	The inactive foreground color used with *base.color*. It falls back to *text.color* if missing.
window.text.color	String (#RRGGBBAA)	A general foreground color.
inactive.window.text.color	String (#RRGGBBAA)	A general foreground color for inactive windows. It falls back to *window.text.color* if missing.
button.text.color	String (#RRGGBBAA)	Obvious.
disabled.text.color	String (#RRGGBBAA)	Obvious.
tooltip.text.color	String (#RRGGBBAA)	Obvious.
highlight.text.color	String (#RRGGBBAA)	The color of selected text.
inactive.highlight.text.color	String (#RRGGBBAA)	The color of inactive selected text. If missing, it falls back to *highlight.text.color*.
link.color	String (#RRGGBBAA)	Obvious.
link.visited.color	String (#RRGGBBAA)	Obvious.
progress.indicator.text.color	String (#RRGGBBAA)	The color of that part of the progress text, which is inside the progress indicator. Useful when the progress text does not have enough contrast with the progress indicator.
progress.inactive.indicator.text. color	String (#RRGGBBAA)	Like *progress.indicator.text.color* but for inactive progress bars.
The Hacks Section		
transparent_dolphin_view	true/false	No background or frame for Dolphin's view (Dolphin is the file manager of KDE)?
transparent_pcmanfm_sidepane	true/false	No background or frame for PCManFM-Qt's side-pane (PCManFM-Qt is the file manager of LXQt)?

transparent_pcmanfm_view	true/false	No background or frame for PCManFM-Qt's folder-view?
blur_translucent	true/false	Blur the regions behind windows that are made translucent by their apps, if possible?
transparent_ktitle_label	true/false	No background for the label of KtitleWidget (a KDE widget with a heading label)?
transparent_menutitle	true/false	No background for (KDE) menu titles?
kcapacitybar_as_progressbar	true/false	Draw KCapacityBar as progressbar? KCapacityBar has its hard-coded style but is drawn as a progressbar by default.
respect_darkness	true/false	Some apps don't respect dark themes. Fix that as far as possible?
force_size_grip	true/false	Show the size grips of dialogs and statusbars as far as possible?
tint_on_mouseover	Integer	Tint the label icons with the highlight color on mouseover by this percentage? This will not work if the containing widget is not styled by Kvantum.
no_selection_tint	true/false	Do not tint selected label icons with the highlight color! False by default.
disabled_icon_opacity	Integer	Set the opacity of disabled icons by this percentage? It is useful with monochrome icons and when the opacity of disabled icons is not reduced by other means.
normal_default_pushbutton	true/false	No bold font for default push buttons?
iconless_pushbutton	true/false	No icon for push buttons that have text?
transparent_arrow_button	true/false	Should tool buttons be transparent (without a panel) when they contain only an arrow?
iconless_menu	true/false	No icon for menus?
single_top_toolbar	true/false	Style only the top toolbar? If *true*, only the top horizontal toolbar, which is immediately below the menubar or at the top of the main window, will be styled. By default, all toolbars are styled, except for the vertical ones.
middle_click_scroll	true/false	Should the scroll slider jump to a position when the scrollbar is middle clicked? The default is false and the jump is done by a left click.

Other Sections		
inherits	String	The name of a section (in the same config file and without brackets) whose configuration is also used for this one.
frame	true/false	Draw a frame around the widget?
frame.top frame.bottom frame.left frame.right	Integer	The height or width of the corresponding frame part. These values are mostly respected but there are two exceptions: (1) Where there is not enough space (because of a bad GUI design); and (2) A few widgets may not accept the exact values – for example, toolbars use the *maximum* value in all directions.
frame.expanded.top frame.expanded.bottom frame.expanded.left frame.expanded.right	Integer	When a frame expansion is done with both *"expand-"* and *"border-"* SVG elements (see *Theme-Making.pdf → Frame Expansion and Border*), the drawn border serves as a kind of frame. Then, these keys determine the thicknesses of its corresponding parts. If they are set to zero, missing or greater than their corresponding ordinary frame sizes, the ordinary frame sizes will be used for the thickness of the expanded border.
frame.expansion	Integer or DECIMALfont	A positive value (in px) will expand the frames until the corner frames meet each other either vertically or horizontally, *provided that at least the height or the width of the widget is not greater than it*. With appropriate SVG images, this key can be used for making corners completely rounded. Its value is zero by default. Read the file *Theme-Making.pdf* for more explanation. The string "font" can be appended to the value (without quotes and space), in which case, the value will be multiplied by the height of the application font (in px) and could also be a decimal (as in ***frame.expansion=2.8font***).
frame.patternsize	Integer	The frame pattern size. Used for tiling the frame of a widget with a pattern. A value of zero means no

		pattern (default). Mostly useful for making dotted frames.
focusRectElement	String	If present, to draw the generic focus rectangle of the current widget, this element will be used instead of the frame element of the *Focus* section. Some widgets may need separate elements for their generic focus rectangles because of their background colors. If so, this key can be used. It has effect for button like widgets, tabs, group boxes and sliders. All other parameters will be taken from the *Focus* section.
focusFrame	true/false	Draw a focus frame when the widget has the keyboard focus? This replaces the generic focus frame/rectangle (see *Focus* section) for button like widgets, tabs, group boxes and sliders. If it is true, the element name that is used for drawing the focus frame will be *frame.element* (see below) plus the string "-focus". All sizes will be those of the usual frame. This key will be ignored if there is no frame and is false by default. Also, note that a focus frame can be expanded like an ordinary frame when *frame.expansion* is positive.
interior	true/false	Draw an interior for the widget?
interior.x.patternsize interior.y.patternsize	Integer	The interior pattern sizes. Used for tiling the interior of a widget with a pattern. A value of zero means no tiling in the corresponding direction. Their absence also means no pattern. *Some widget types may never accept patterns.*
focusInterior	true/false	Draw a focus interior when the widget has the keyboard focus? If it is true, the element name that is used for drawing the focus interior will be *interior.element* (see below) plus the string "-focus". This key will be ignored if *focusFrame* is false or there is no interior or frame and is false by default.

indicator.size	Integer	Some widgets, like scrollbar arrows, have indicators. This is their size.
text.margin	true/false	Put a margin around the text?
text.margin.top text.margin.bottom text.margin.left text.margin.right	Integer	The sizes of the text margins if there is any. These values are mostly respected but not if there is a lack of space because of a bad GUI design. Also, in widgets like LineEdit, where texts are not drawn by Kvantum, the real left/right text margins may be a little greater than *text.margin.left/right*.
text.normal.color	String (#RRGGBBAA)	The color of the normal text as #RRGGBB or #RRGGBBAA or with a valid name like white, black, red, etc. It may override the text colors defined under the GeneralColors section. **Note 1:** *MenuBarItem* always gets its normal text color from the *MenuBar* section or, if menubar and toolbar are merged (see the key *merge_menubar_with_toolbar*), from the *Toolbar* section. **Note 2:** State-specific text colors do not have meaning for *Window*, *Dialog*, *Dock*, *LineEdit* and frame widgets (namely *GenericFrame*, *TabFrame* and *TabBarFrame*). In the case of *LineEdit*, the text color is *text.color*.
text.focus.color	String (#RRGGBBAA)	The color of the focused (hover) text. **Note:** For *MenuBarItem*, it serves as a fallback color in case the press or toggle text colors are not set.
text.press.color	String (#RRGGBBAA)	The color of the pressed text.
text.toggle.color	String (#RRGGBBAA)	The color of the toggled text.
text.normal.inactive.color text.focus.inactive.color text.press.inactive.color text.toggle.inactive.color	String (#RRGGBBAA)	These keys are for text colors of inactive widgets and fall back to their corresponding ordinary keys if missing. **Note:** Except for button like widgets and,

		probably, view items, the normal and inactive colors should not have high contrast with each other.
text.bold	true/false	Bold font for text? The font is not bold by default. **Note:** For technical reasons, texts of combo boxes, menu-items and view items are not made bold by Kvantum.
text.boldness	Integer	The weight of bold texts (if any). Its value is from 1 to 5, with 3 (normally bold) as default. Note that some fonts may have only one degree of boldness.
text.italic	true/false	Italic font for text? The font is not italic by default. **Note:** For technical reasons, texts of combo boxes, menu-items and view items are not made italic by Kvantum.
text.shadow	true/false	Draw a shadow for the text?
text.shadow.xshift text.shadow.yshift	Integer	The vertical/horizontal shifts of the text shadow if it exists.
text.shadow.color	String (#RRGGBBAA)	The color of the text shadow as #RRGGBB or #RRGGBBAA or with a valid name like white, black, red, etc. It should have enough contrast with the text color; otherwise, it won't be drawn; e.g., black texts can't have black shadows.
text.inactive.shadow.color	String (#RRGGBBAA)	Like *text.shadow.color* and falling back to it if missing but for inactive widgets.
text.shadow.alpha	Integer (0-255)	The opacity of the text shadow. 255 means completely opaque. If the alpha is set in #RRGGBBAA as the shadow color name, this key can be left to 255 (its default value).
text.shadow.depth	Integer	The text shadow depth. **Note:** For technical reasons, depths > 1 are considered as 1. It is kept until Qt supports a better way of adding text shadows by using its painter.

| min_width min_height | Integer or DECIMALfont or +DECIMALfont | Minimum width or height (in px). They have meaning only for menuitems, menubar items, push and tool buttons, tabs, combo boxes, line-edits, spin boxes (only for their height), check and radio buttons (only the height) and, sometimes, view items (only the height).

They can also be used under Window and Dialog sections (and their translucent counterparts), in which case, they mean the minimum width/height of the drawn background (interior) SVG element, so that if the widget's width/height is smaller, the background will be cut from right/bottom. This can be useful with gradients.

The string "font" can be appended to the value (without quotes and space), in which case, the value will be multiplied by the height of the application font (in px) and could also be a decimal (as in *min_height=2.3font*).

If "+" is prepended to the value (without quotes and space), the value will be added to the width/height of the widget, instead of being considered as the minimum. For example, *min_width=+0.5font* adds half the font height to the widget width. Please do not mistake "+" for a positive value here; the values are always considered to be nonnegative.

Note: For combo boxes and line-edits, when "+" is used with *min_width*, the horizontal text margins are also increased. |

If you want to make your own theme (see the file "Theme-Making"), you will also need to know the meanings of these variables:

Elements Table

Variable (Key)	Value	Meaning
interior.element	String	The SVG element to be used for drawing the interior of a widget.
frame.element	String	The SVG element to be used for drawing the frame of a widget.
frame.expandedElement	String	The optional SVG element to be used only for drawing the expanded frame. If not present, *frame.element* will be used for expanded frames too. See *frame.expansion*!
indicator.element	String	The SVG element to be used for drawing the indicator of a widget.

Application Themes

Specific Themes can be assigned to applications, so that those applications use their corresponding themes instead of the active theme. This can be done easily by using the last page of **Kvantum Manager** (*Application Themes*) or manually, by adding app lists (executable names) to '~/.config/Kvantum/kvantum.kvconfig' with the following format:

[General]
theme=ACTIVE_THEME

[Applications]
THEME1=app1, app2
THEME2=app3
…

Some Examples

If you don't want menus and tooltips to be translucent or cast shadow and want the current color palette to be used for all texts, you could use a blank configuration or a very basic one with just this in it:

[%General]

You could also be more explicit:

[%General]
composite=false

[PanelButtonCommand]
text.normal.color=none
text.focus.color=none
text.press.color=none
text.toggle.color=none

Here "none" is not a valid color, so text colors will be taken from the currently used color palette.

If you want to have bigger buttons without increasing your font sizes, you could use this:

[%General]
composite=true
menu_shadow_depth=6
tooltip_shadow_depth=6

[PanelButtonCommand]
text.normal.color=white
text.focus.color=#80C0FF
text.press.color=white
text.toggle.color=white
text.margin.top=4
text.margin.bottom=4
text.margin.left=5
text.margin.right=5

[PanelButtonTool]
inherits=PanelButtonCommand

To have black text shadows with light green focused text, use this (black text shadows are already defined but disabled in the default config file):

[%General]
composite=true
menu_shadow_depth=6
tooltip_shadow_depth=6

[PanelButtonCommand]
text.normal.color=white
text.focus.color=lightgreen
text.press.color=white
text.toggle.color=white

text.shadow=true

Note that, in the two examples above, the compositing values and normal/focused/pressed text colors are also added because otherwise, they would be disabled (see the exceptions above). In the previous example, customized text colors were disabled for all widgets other than push-buttons because there were no sections for them. If you want them back, you could add sections like these:

[PanelButtonTool]
inherits=PanelButtonCommand
[Tab]
inherits=PanelButtonCommand
[MenuItem]
inherits=PanelButtonCommand

And so on.