

Custom Title Bar for VCL Forms

Overview

The `TForm.CustomTitleBar` property and new `TTitleBarPanel` control allow you to customize a VCL form's native title bar similar to Windows Explorer, Google Chrome, or other applications. You can place VCL controls on the title bar, and custom paint over the title bar, as well as control the default painting of elements such as the window icon and caption.

Contents

Overview

Basic Customization

VCL Control Location

Settings and Partial Custom Painting

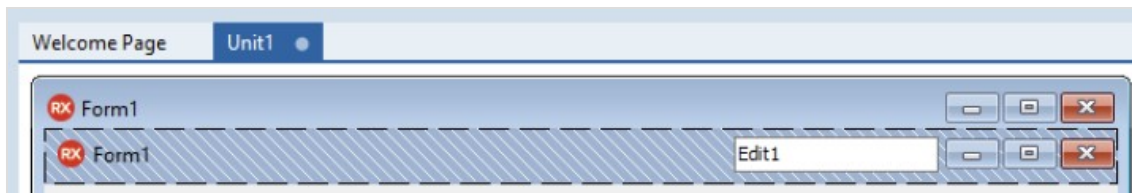
Full Custom Painting

Caption Buttons

Windows 7 and 10 support

Custom Controls

See Also



Note: *The custom title bar is supported on Windows 7 and later.*

The custom title bar support uses the genuine native Windows title bar, and relies on the DWM (Desktop Window Manager) being enabled. Turning off Aero on Windows 7, or other situations where the DWM is not used to render windows, results in title bar customization not being supported. This is gracefully handled by the control, but to check for this situation in your code check the value of `TForm.CustomTitleBar.Supported`. This is `True` when you can use the custom title bar.

To utilize the full power of the custom title bar, you must use a combination of the `TForm.CustomTitleBar` property and a `TTitleBarPanel` placed on the form. The panel is used to host VCL controls, but it is also required to enable advanced features such as customizing painting, even if you have no custom controls on the title bar. Basically, the `TForm.CustomTitleBar` property represents settings for the form title bar, and the `TTitleBarPanel` is a VCL control used to represent and access advanced features in the title bar itself.

Basic Customization

The `VCL TForm` class has a new property, `CustomTitleBar`. This allows you to change basic properties, such as displaying the caption or icon, as well as controlling the more advanced features including placing controls.

To enable title bar customization, set `TForm.CustomTitleBar.Enabled` to `True`. Notice that the top of your VCL form in the designer will now have a shaded blue area. This represents the area of the form which will be used for the title bar.

VCL Control Location

The title bar is implemented through `DwmExtendFrameIntoClientArea` function (<https://docs.microsoft.com/en-us/windows/win32/api/dwmapi/nf-dwmapi-dwmextendframeintoclientarea%7C>), meaning that there is no non-client area to your form, and the title bar will be rendered using some of the client area, i.e. the space in which VCL controls are able to be placed.

All controls parented to the form with an `Align` property, such as `alTop`, will adjust to the title bar area correctly. However, any control which is placed via `Left` and `Top` properties, i.e. absolute positioning, will remain where it is and so may overlap the title bar, and need to have its position adjusted when the custom title bar is enabled.

We recommend you place controls on your form on a client-aligned `TPanel`. Note that you may also want to make your form's `Height` larger to account for the space previously used by the title bar.

Settings and Partial Custom Painting

The following settings control the behavior of the title bar, as well as controlling some common painting scenarios (such as not painting the title bar caption), which lets you address these scenarios without implementing full custom painting.

- **CaptionAlignment:** Use this to specify where the caption is drawn. Windows 10 draws on the left by default; you can draw the caption centered like Windows 8 or right-aligned.
- **Control:** Set this to a `TTitleBarPanel` that is placed on the form. See the below section about custom controls on a title bar.
- **Enabled:** Activates or inactivates the use of the custom title bar.
- **Height:** Height of the title bar in pixels. To change the height of the title bar, turn off **SystemHeight**, which constrains the title bar to the default system height. You can then set the `Height` property to any value.
- **ShowCaption:** Controls if the the caption is drawn.
- **ShowIcon:** Controls if the form's icon is drawn on the left edge of the title bar.
- **SystemHeight** When set to `True`, the title bar height will always be the value used by other windows on the current screen. To customize the title bar height, disable **SystemHeight** and then enter a value in the `Height` property.
- **SystemColors:** Controls if the title bar caption and buttons use the system colors, or custom colors when active or inactive. This is useful when you want to draw your title bar using different background and foreground colors than the system uses, without requiring full custom painting.

When `True`, the background, caption, and system buttons paint using the system default colors like other applications. When `False`, the colors you provide in the color properties (see below) are used.

For custom colors to be used, you must place a `TTitleBarPanel` on the form and set the `CustomTitleBar.Control` property to it, exactly as though you were planning to place controls on the title bar, even if the title bar panel is empty and contains no controls.

- **BackgroundColor**, **ForegroundColor**, **InactiveBackgroundColor**, and **InactiveForegroundColor** are used to specify the colors the title bar should use when painting. Set the **SystemColors** to `True` to ignore these values and paint using the current system defaults.
- **ButtonBackgroundColor**, **ButtonForegroundColor**, **ButtonHoverBackgroundColor**, **ButtonHoverForegroundColor**, **ButtonInactiveBackgroundColor**, **ButtonInactiveForegroundColor**, **ButtonPressedBackgroundColor**, and **ButtonPressedForegroundColor** all specify the colors to be used by system buttons when there are custom buttons or when the title bar is custom drawn. Set **SystemColors** to `True` to ignore these values and paint using the current system defaults.

Full Custom Painting

You can fully customize how the title bar is rendered by painting on the title bar and drawing anything on it. To do this, you must place a `TTitleBarPanel` on the form and set the `CustomTitleBar.Control` property to it, exactly as you were planning to place controls on the title bar, even if the title bar panel is empty and contains no controls. You can then create an event handler for the `TTitleBarPanel.OnPaint` event.

The `TTitleBarPanel.OnPaint` event is called as the very last stage of painting. Painting will be over anything else already painted on the title bar, such as the caption or the system buttons. You can also use the `Rect` properties to customize your drawing. The `IconRect`, `FrameRect`, `ClientRect`, and `CaptionButtonsRect` properties help you to know where the system buttons, caption, etc are located.

The paint is a 32bit surface. It has the same painting requirements as painting on glass in Windows 7.

Caption Buttons

You can add custom caption buttons to the title bar. This is used in the IDE itself to add Help and Desktop buttons to the left of the Minimize / Maximize / Close buttons you normally see on a title bar.

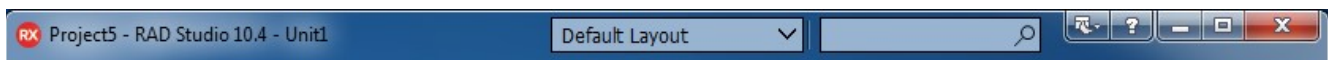
To do so, place a `TTitleBarPanel` on the form and set it to the form's `CustomTitleBar.Control` property as detailed above. You can then add custom buttons via the panel's `CustomButtons` property. This is a collection of `TCaptionButtonItem`. Buttons are automatically laid out in order right to left starting from the left side of the Minimize button. The 0th button in the collection will be rightmost, and the last button will be leftmost. On Windows 7, there is a small space between the minimize button and the first custom button.

The `TCaptionButtonItem` component represents a button on the title bar. It has the following properties:

- **ButtonType:** allows you to use a system button, such as Close, Minimize, or Restore, add a space, or have a fully custom button. Usually you will use the custom button.
- **Enabled:** if it is turned off, the button will be drawn and will not respond to mouse movements or clicks.
- **Hint:** allows you to specify a tooltip to be shown when the mouse hovers over the button.
- **Visible:** controls if the button is drawn on the title bar.
- **Width:** controls the width of the button.
- **OnClick:** an event handler for when the button is clicked.
- **OnPaint:** an event handler to draw on the button, such as a button glyph. Buttons do not connect to an image list.

Windows 7 and 10 support

The custom title bar has full support for Windows 7 and Windows 10, including for caption buttons:



The RAD Studio title bar on Windows 7.

Note:

Windows Vista or earlier are not supported platforms.

The custom title bar is supported on Windows 8 but there may be some visual differences compared to other applications, such as how system buttons are painted if customizing their colors (they will render like Windows 10 buttons).

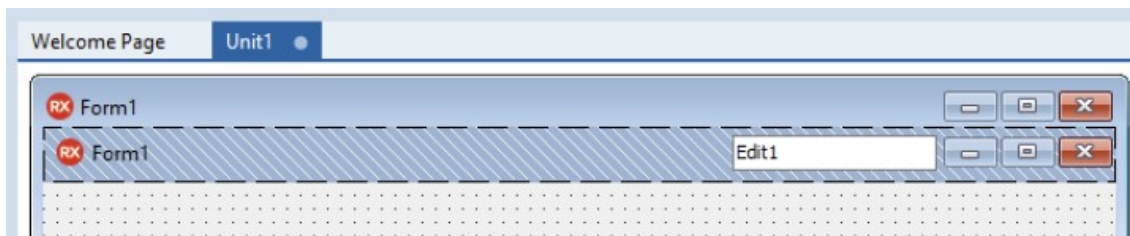
Custom Controls

The `TTitleBarPanel` control is used to place controls in the title bar area of the VCL form. It is a container that provides design time support for the controls. It also maintains a relationship between the form and the customized title bar area.

Controls that are placed on the title bar must support drawing on a 32bit surface, similar to glass in Windows 7. As a general guide, if a control is rendered correctly on glass in Windows 7, it will likely render correctly on the title bar in Windows 7 and 10. It is common to place a control and see it renders incorrectly, even in the designer. You may need to make use of properties like `DoubleBuffered` for normal VCL controls.

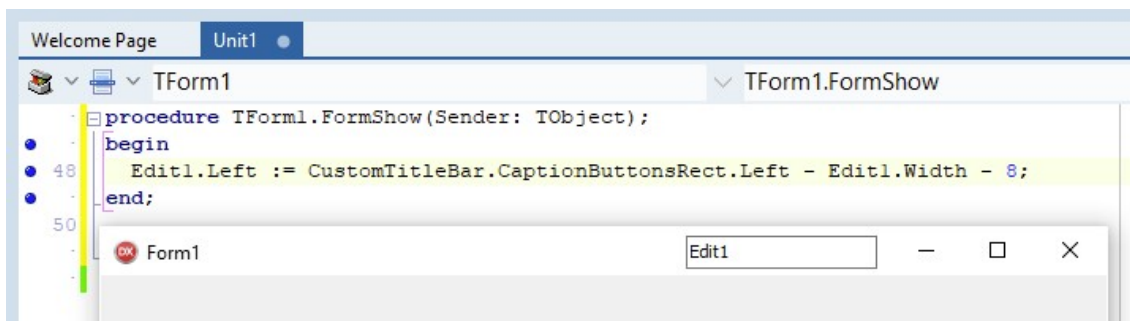
To place controls on the title bar:

- In the Palette, locate the `TTitleBarPanel` control and place one on the form. It will appear at the top of the form.
- Set your form's `CustomTitleBar.Control` property to the new title bar panel. You will see the panel has a blue background color, and displays system buttons on its right side.
- Optionally, change the title bar panel's `Height` property to match the `CustomTitleBar.Height` property. Only controls within the title bar's height will be rendered over the title bar itself.
- In the Palette, locate a control, such as `TEdit`, and place it on the `TTitleBarPanel` with this control:
 - In the Object Inspector, set your new control's `DoubleBuffered` property to `True`.
 - Since the `TTitleBarPanel` is a normal VCL control, you can set `Align`, `Anchor`s, and other positioning properties for controls placed on it. To have your edit control right-aligned next to the system buttons, as a search edit might be, drag it until it is in the right position, and then change the `Anchor`s property to remove `akLeft` and add `akRight`.



- Back in the form, add an OnShow event, or use OnResize if you want to manually position controls in the title bar instead of using alignment or anchors. (We recommend using alignment and anchors, since positioning via OnResize can visually lag behind the window size). You can refer to the CustomTitleBar's [IconRect](#), [FrameRect](#), [ClientRect](#), and [CaptionButtonsRect](#) properties to position your controls relative to known parts of the title bar. Here, set your edit control's Left property to CaptionButtonsRect.Left minus the width of your edit minus a small margin (the margin just for visual esthetics) by adding the following line of code to your event handler:

```
Edit1.Left := CustomTitleBar.CaptionButtonsRect.Left - Edit1.Width - 8;
```



See Also

- [CustomTitleBar](#)
- [TTitleBarPanel](#)
- [CaptionAlignment](#)
- [Control](#)
- [Enabled](#)
- [Height](#)
- [SystemHeight](#)
- [ShowCaption](#)
- [ShowIcon](#)
- [SystemColors](#)
- [BackgroundColor](#)
- [ForegroundColor](#)
- [InactiveBackgroundColor](#)
- [InactiveForegroundColor](#)
- [SystemColors](#)
- [ButtonBackgroundColor](#)
- [ButtonForegroundColor](#)
- [ButtonHoverBackgroundColor](#)
- [ButtonHoverForegroundColor](#)
- [ButtonInactiveBackgroundColor](#)
- [ButtonInactiveForegroundColor](#)
- [ButtonPressedBackgroundColor](#)
- [ButtonPressedForegroundColor](#)
- [TTitleBarPanel.OnPaint](#)

- IconRect
- FrameRect
- ClientRect
- CaptionButtonsRect

Retrieved from "http://docwiki.embarcadero.com/RADStudio/Sydney/e/index.php?title=Custom_Title_Bar_for_VCL_Forms&oldid=271435"

This page was last edited on 22 May 2020, at 12:19.