

What's New in Raize Components 5

Raize Components is a *user interface design system* for Delphi and C++Builder. At its center is a collection of **more than 125** general-purpose native VCL controls. Built on a foundation of technology first created more than ten years ago, these high-quality components give developers unsurpassed power and flexibility without sacrificing ease-of-use. In addition to the core set of controls, Raize Components includes **more than 100** component designers focused on simplifying user interface development. Now more than ever, developers use Raize Components to build sophisticated user interfaces in less time with less effort.

Raize Components 5 focuses on the features and enhancements most requested by our users. For example, there are many developers interested in developing their user interfaces using **Unicode** in **CodeGear RAD Studio 2009** (including **Delphi 2009** and **C++Builder 2009**). We couldn't agree more, and Raize Components 5 provides full support for Unicode when using RAD Studio 2009.

Another very popular request has been for a mechanism to persist property values between program runs. Raize Components 5 now includes the new **TRzPropertyStore** component, which makes it very easy to do exactly this. Connect the **TRzPropertyStore** to a **TRzRegIniFile** and then select the components and properties to be persisted using a collection editor--you don't even need to type in property names as the built-in property editors take care listing all of the properties available in the selected component. Call the Save and Load methods in your code, and you're done.

Of course, we receive the most feedback for our more popular controls. TRzPageControl and TRzTabControl are certainly no exception, and Raize Components 5 includes many enhancements to these two powerful controls. For example, ever since Internet Explorer 7 started placing a close button just on the active tab, developers have requested the same functionality in Raize Components. Well, in Raize Components 5, all you need to do is to set the ShowCloseButtonOnActiveTab property in either control to True, and you get the enhanced functionality.

These are just a few of the new features and enhancements included in Raize Components 5. The following sections provide a more detailed look at what's new in **Raize Components 5**:

- Component Enhancements
- New Components
- Custom Framing Enhancements
- Design-Time Support Enhancements
- Package Changes
- Migration Issues

Component Enhancements

Unicode Support (in CodeGear RAD Studio 2009)

- Raize Components 5 provides complete support for Unicode when used with CodeGear RAD Studio 2009 (Delphi 2009 or C++Builder 2009).

TRzBackground

- Fixed problem where the **TRzBackground** component displaying a gradient would get into a repaint loop if the control was placed on a container that forced the component to occupy a space smaller than 5 pixels horizontally or vertically.

TRzButton, TRzBitBtn, TRzMenuButton

- Fixed issue in **TRzButton** and descendants (include **TRadioButton** and **TRzCheckBox**) where the text of the control would be unreadable if the color of the control was close to the font color used when the control was disabled.
- Modified the header file (RzButton.hpp) that gets generated by the Delphi compiler when compiling components for use in C++Builder. The new modifications allow C++Builder developers to create custom controls that descend from the **TRzCustomButton**, **TRzButton**, and **TRzShapeButton** classes (or from other classes that descend from these) without resulting in linker errors because of differences in how the **HDC** type is defined in Delphi and C++.

NOTE:

When using C++Builder 2009 or later, the above modifications are not necessary because of changes made to the Delphi compiler. However, the changes are dependent on the **STRICT** conditional symbol being defined. However, C++Builder projects define the **NO_STRICT** symbol by default. Therefore, in order to compile and link descendant controls in C++Builder 2009 or later, the **NO_STRICT** symbol must be replaced with the **STRICT** symbol.

- Registered custom property editor for the **Caption** property of **TRzButton**, **TRzBitBtn**, **TRzMenuButton**, and **TRzToolButton**. The editor allows the developer to display a dialog box that can be used to enter multi-line captions at design-time.
- Updated image position in **TRzBitBtn** and **TRzMenuButton** when running under right-to-left (RTL) systems. If the **Layout** property is set to **biGlyphLeft** (the default) and **BiDiMode** is set to **bdRightToLeft**, the image is positioned on the right side as if layout was **biGlyphRight**.

TRzButtonEdit & TRzDBButtonEdit

- Fixed problem with hot track custom framing not getting updated correctly in **TRzButtonEdit**.

TRzCheckBox, TRzDBCheckBox & TRzRadioButton

- Added new **AutoSize** property to **TRzRadioButton** and **TRzCheckBox**. When this property is True (the default), the bounds of the control are automatically adjusted so that the entire caption is visible. This also allows the focus rectangle to surround just the caption of the control rather than just the entire client area.
- Added new **CustomGlyphImages** property to **TRzRadioButton** and **TRzCheckBox**. This property is used to reference an **ImageList** that contains the glyphs to be used for the various states of the control. This new property should be used instead of the deprecated **CustomGlyphs** property, which is still available strictly for backward compatibility. By referencing an **ImageList** that holds the custom glyphs rather than an embedded bitmap, the actual glyph images are stored only once in the

application instead of inside each instance of the control. When populating a TImageList for use with **CustomGlyphImages**, each index in the ImageList represents a different state. The following tables describe the mapping:

TRzRadioButton CustomGlyphImages Index Mapping

Index	State	
0	Unchecked	
1	Checked	
2	Unchecked	Pressed
3	Checked	Pressed
4	Unchecked	Disabled
5	Checked	Disabled
6	Unchecked	Hot (Optional)
7	Checked	Hot (Optional)

TRzCheckBox CustomGlyphImages Index Mapping

Index	State	
0	Unchecked	
1	Checked	
2	Grayed	
3	Unchecked	Pressed
4	Checked	Pressed
5	Grayed	Pressed
6	Unchecked	Disabled
7	Checked	Disabled
8	Grayed	Disabled
9	Unchecked	Hot (Optional)
10	Checked	Hot (Optional)
11	Grayed	Hot (Optional)

- As noted in the above table, with the new **CustomGlyphImages** property, it is now possible to specify "hot" glyphs. That is, separate images to be displayed for a given state when the mouse is positioned over the control.
- Fixed issue where focus rectangle would not completely surround the caption of a **TRzRadioButton** or **TRzCheckBox** that contained tab characters.
- Modified the header file (RzRadChk.hpp) that gets generated by the Delphi compiler when compiling components for use in C++Builder. The new modifications allow C++Builder developers to create custom controls that descend from the **TRzCustomGlyphButton** class (or from other classes that descend from this class such as TRzCheckBox or TRzRadioButton) without resulting in linker errors because of differences in how the HDC type is defined in Delphi and C++.

NOTE:

When using C++Builder 2009 or later, the above modifications are not necessary because of changes made to the Delphi compiler. However, the changes are dependent on the STRICT conditional symbol being defined. However C++Builder projects define the **NO_STRICT** symbol by default. Therefore, in

What's New in Raize Components 5

order to compile and link descendant controls in C++Builder 2009 or later, the **NO_STRICT** symbol must be replaced with the **STRICT** symbol.

- Registered custom property editor for the **Caption** property of **TRzCheckBox** and **TRzRadioButton**. The editor allows the developer to display a dialog box that can be used to enter multi-line captions at design-time.
- Added new **ThemeAware** property to **TRzCheckBox** and **TRzRadioButton**. When this property is set to True (the default) the control will use XP/Vista themes for display (if available). When set to False, the control will be drawn using the visual property settings of the controls even if themes are available.

TRzCheckList

- The **TRzCheckList** component has been enhanced so that an image can be associated with each item in the list. The new **Images** property is used to reference the desired **ImageList** that contains the images to be displayed. The **ItemImageIndex[Index]** and **ItemDisabledIndex[Index]** array properties are used to specify the image to display when the item is enabled and disabled, respectively. The image is display between the check box glyph and the item's text.
- Added new **CustomGlyphImages** property to **TRzCheckList**. This property is used to reference an **ImageList** that contains the glyphs to be used for the various states of the control. This new property should be used instead of the deprecated **CustomGlyphs** property, which is still available strictly for backward compatibility. By referencing an **ImageList** that holds the custom glyphs rather than an embedded bitmap, the actual glyph images are stored only once in the application instead of inside each instance of the control. When populating a **TImageList** for use with **CustomGlyphImages**, each index in the **ImageList** represents a different state. The following tables describe the mapping. The mapping is the same as that of **TRzCheckBox**, which is not true for the old **CustomGlyphs** mapping.

TRzCheckGroup CustomGlyphImages Index Mapping

Index	State	
0	Unchecked	
1	Checked	
2	Grayed	
3	Unchecked	Pressed (Unused by TRzCheckList)
4	Checked	Pressed (Unused by TRzCheckList)
5	Grayed	Pressed (Unused by TRzCheckList)
6	Unchecked	Disabled
7	Checked	Disabled
8	Grayed	Disabled
9	Unchecked	Hot (Unused by TRzCheckList)
10	Checked	Hot (Unused by TRzCheckList)
11	Grayed	Hot (Unused by TRzCheckList)

- Added the **WinMaskColor** for use with **CustomGlyphs** and **CustomGlyphImages**. This property provides the same behavior as the **WinMaskColor** property that is available in **TRzCheckBox**. That is, any pixels of the specified color in the custom glyphs are replaced with the current **clWindow** color.
- Added new overloaded methods to **TRzCheckList** for **SaveToFile** and **LoadFromFile** that accept a **TEncoding** parameter in order to handle saving and loading items that contain Unicode characters. These methods are only available in RAD Studio 2009 or later.

TRzColorEdit

What's New in Raize Components 5

- Fixed problem in **TRzColorEdit** where changing the **DropButtonVisible** property to False, and then toggling the **ReadOnly** property would cause the drop button to reappear.

TRzComboBox & TRzDBComboBox

- The **DropDownWidth** property of **TRzComboBox** (and descendants) has been enhanced such that when the value of the property is 0, the width of the drop down list is automatically adjusted so that all items are completely in view. In previous versions the width of the drop down list would always match the width of the combo box unless a non-zero value was specified for **DropDownWidth**.
- Added new **TextHint** property to **TRzComboBox**. The **TextHint** property allows a developer to specify a prompt that appears inside the edit area of the control, when the control is empty. The text prompt appears grayed.

NOTES:

- TextHint is only applicable under WinXP or Vista, and only when XP/Vista themes are in use.
- TextHint is also only applicable when the Style of the combo box is csDropDown (i.e. the user can type into the edit area).
- Unlike the Windows Edit Control, the Combo Box does not support the ability to keep the prompt visible while the control has the focus. Therefore, there is no TextHintVisibleOnFocus property like there is in TRzEdit.
- RAD Studio 2009 supports TextHint for TComboBox. TRzComboBox utilizes the inherited TextHint property where possible.
- Added **ClearItems** method to **TRzImageComboBox** that handles freeing the associated objects that define the image index and indent level when deleting each item.
- Modified the header file (RzCmboBx.hpp) that gets generated by the Delphi compiler when compiling components for use in C++Builder. The new modifications allow C++Builder developers to create custom controls that descend from the TRzCustomComboBox class (or from other classes that descend from this class) without resulting in linker errors because of differences in how the HWnd type is defined in Delphi and C++.

NOTE:

When using C++Builder 2009 or later, the above modifications are not necessary because of changes made to the Delphi compiler. However, the changes are dependent on the **STRICT** conditional symbol being defined. However, C++Builder projects define the **NO_STRICT** symbol by default. Therefore, in order to compile and link descendant controls in C++Builder 2009 or later, the **NO_STRICT** symbol must be replaced with the STRICT symbol.

TRzDateTimeEdit & TRzDBDateTimeEdit

- Fixed problem in **TRzDateTimeEdit** where changing the **DropButtonVisible** property to False, and then toggling the **ReadOnly** property would cause the drop button to reappear.
- The drop down calendar and time picker displayed by the **TRzDateTimeEdit** control now uses the same font size as the edit control itself. Therefore, if a larger font is used for the edit field text, the calendar and time picker will also be displayed using the larger font.

TRzDBGrid

- Fixed problem where setting **TRzDBGrid.AltRowShading** to True at design-time would not get picked up at runtime.
- If the total number of records in the dataset can be displayed completely in a **TRzDBGrid**, then the vertical scroll bar is not displayed.

What's New in Raize Components 5

- If a vertical scroll bar is displayed in a **TRzDBGrid**, the **UpArrow** and **DownArrow** buttons on the scroll bar move the current record up or down by 1 rather than jumping to the opposite side (top or bottom) of the grid as is done in the base **TDBGrid**. The result is a much more natural scrolling experience.

TRzDBLookupDialog

- Fixed problem where setting the **KeyField** or **SearchField** properties in **TRzDBLookupDialog** caused the IDE to immediately exit when setting at design-time.

TRzDialogButtons

- Optimized streaming code for several properties of **TRzDialogButtons**.

TRzEdit & TRzDBEdit

- Added new **TextHint** and **TextHintVisibleOnFocus** properties to **TRzEdit** and descendants (e.g. **TRzNumericEdit**, **TRzDateTimeEdit**). The **TextHint** property allows a developer to specify a prompt that appears inside the edit area of the control, when the control is empty. The text prompt appears grayed. By default, the prompt is automatically hidden when the control receives focus. To keep the prompt visible until the user enters some text, set the **TextHintVisibleOnFocus** property to True.

NOTES:

- **TextHint** is only applicable under WinXP or Vista, and only when XP/Vista themes are in use.
- The **TextHintVisibleOnFocus** is only applicable under Vista.
- RAD Studio 2009 supports **TextHint** for **TEdit**, but not the **TextHintVisibleOnFocus** property. The **TRzEdit** and descendants utilize the inherited **TextHint** property where possible, and add the **TextHintVisibleOnFocus** property where necessary.

TRzEditListBox

- Added **EditColor** and **EditFontColor** properties to **TRzEditListBox**. These properties control the appearance of the popup edit box that is displayed when editing an item in the list.
- The **TRzEditListBox.PopupEdit** property also has a new **Color** sub-property (a **Font** sub-property already existed), which can be used to dynamically change the popup edit's appearance in the **TRzEditListBox.OnShowingEditor** event.

TRzExpandEdit & TRzDBExpandEdit

- The **TRzExpandEdit** now expands to the left on right-to-left systems.

TRzFontComboBox

- Fixed issue in **TRzFontComboBox** with **ShowStyle** set to **ssFontPreview** that resulted in the Preview panel being displayed on a different monitor when the combo box was positioned close to the right edge of a monitor.

TRzFontListBox

- Fixed issue in **TRzFontListBox** with **ShowStyle** set to **ssFontPreview** that resulted in the Preview panel being displayed on a different monitor when the list box was positioned at the right edge of a monitor.

TRzGroupBar

- Added **HideAccelerators** property to **TRzGroup** and **TRzGroupController**. Normally, the captions of group items are stripped of their Ampersand (&) characters before display to handle situations where actions are assigned to group items, and the actions' captions have accelerators in them. Set this property to False to display the & characters in the group item captions.
- Added new **SelectionFontColor** property to **TRzGroup** and **TRzGroupController**. This property is used to specify the text color of a selected item in a group. Like this **SelectionColor** and **SelectionFrameColor** properties, the **SelectionFontColor** property only takes affect when the **TRzGroupBar.GradientColorStyle** is set to **gcsCustom**.
- Moved the **TRzGroupBar.IsScrollBarVisible** method to the public section.

TRzGroupBox

- Added new **BannerHeight** property to **TRzGroupBox**, which is used to control the height of the banner area when the **gsBanner GroupStyle** is used. By default, this property is 0, which instructs the control to determine the height of the Banner based on the font size. If **BannerHeight** is set to a non-zero value, the height of the banner is sized accordingly.

TRzLabel

- Fixed problem where the **TRzLabel** would not get sized correctly when **AutoSize** was set to True and **TextMargin** was greater than zero.

TRzLine

- Added new **Caption** property to **TRzLine**. The **Caption** is displayed over the center of the line.

TRzListBox

- Fixed problem where the horizontal scroll bar would appear even when all of the text was visible in the **TRzListBox** (and descendants). Updated the calculation used by the **AdjustHorzExtent** methods.
- Modified **TRzTabbedListBox.DrawGroup** method such that when XP/Vista themes are used, the group's color and font color are drawn using appropriate colors regardless of the XP/Vista color scheme currently being used.

TRzNumericEdit & TRzDBNumericEdit

- The **TRzNumericEdit** control now has a **CalculatorVisible** property. When this property is set to True, a drop down button becomes visible, and when clicked, a popup **Calculator** (an instance of the new **TRzCalculator**) is displayed. The calculator allows the user to perform simple numeric calculations and then move the result into the Value property of the **TRzNumericEdit**. The calculator supports both Integer and floating point operations.

NOTE:

Please note that if the **TRzNumericEdit** specifies a **DisplayFormat** that does not specify a decimal placeholder and the calculator produces a floating-point result, the result will be rounded according to **DisplayFormat** property.

- Fixed problem in **TRzNumericEdit** where changing the **DropButtonVisible** property to False, and then toggling the **ReadOnly** property would cause the drop button to reappear.

TRzPageControl & TRzTabControl

What's New in Raize Components 5

- Added the new **ShowCloseButtonOnActiveTab** property to **TRzPageControl** and **TRzTabControl**. When this property is set to True, then a close button appears on the active tab. If the user clicks the close button, the **OnClose** event for the control fires. To actually close the tab, write an event handler for the **OnClose** event and set the **AllowClose** parameter to True. This is the same procedure that is used when setting **ShowCloseButton** to True and having a close button appear at the corner of the control.
- Added a new **TabStyle** called **tsSquareCorners** to **TRzPageControl** and **TRzTabControl**. This style is similar to **tsRoundCorners** in that the active tab is slightly larger than the other tabs, but the corners of the tabs are squared off. This style is used frequently in Windows Vista.
- Modified the hot tracking styles used by the **TRzPageControl** and **TRzTabControl**. Specifically, the default setting for **HotTrackStyle (htsTab)** now causes the entire background of the tab to be highlighted as the mouse is moved over the tab. The previous style of having a small bar appear at the edge of the tab is available in the new **htsTabBar** setting.
- Added new **HotTrackColorSource** property, which is used to determine how the **TRzPageControl** and **TRzTabControl** determine which color to use when highlighting a tab during hot-tracking. When set to **htcsTabColor**, the color of the tab itself is used as a basis for the hot tracking color. When set to **htcsHotTrackColorProp**, the color value specified in the **HotTrackColor** property is used.
- The **HotTrackColorType** property is now used in all hot tracking styles, and not just in the **htsTabBar** style. That is, if the **HotTrackColorSource** property is set to **htcsHotTrackColorProp**, then the actual color used for hot tracking will be the color specified in **HotTrackColor** if **HotTrackColorType** is set to **htctActual** (the default), or the complementary color of **HotTrackColor** if **HotTrackColorType** is set to **htctComplement**.
- The control buttons (Scroller, Menu, and Close) now hot track when the mouse is positioned over the control to provide better feedback to the user.
- Fixed issue where a **TRzTabControl** with **TabOrientation** set to **toBottom** or **toRight**, **ShowCard** set to **False**, and **TabStyle** set to **tsRoundCorners** would result in a 1 pixel gap between the frame line and the edge of the control.
- Fixed problem where assigning the **TRzTabControl.TabIndex** within a **BeginUpdate..EndUpdate** method pair would not actually change the active tab index when **EndUpdate** returns.
- Fixed issue where the buttons next to the tabs of a **TRzPageControl** or **TRzTabControl** (for scrolling, navigating, and closing) would get clipped if the height of the tabs was made smaller than the default size.
- Removed the extra space that was displayed on both sides of a tab's caption when using the **tsRoundCorners TabStyle**.
- Fixed issue where setting **AlignTabs** to True in **TRzTabControl** and the last tab had its **Visible** property set to False, would not result in the tabs occupying the entire width of the tab control.
- When **ShowMenuButton** is True, the drop down menu that is displayed is now sorted alphabetically to make it easier to locate the desired tab.

TRzPathBar

- Added **ItemHotStyle** property to **TRzPathBar**. This property is used to change how an item appears when the mouse is positioned over the item. The default is **ihsBox**. The other options is **ihsUnderline**.
- When running under right-to-left locales and the default **Separator** is used (i.e. '>'), the character is automatically switched to the '<' symbol.

TRzRadioGroup, TRzDBRadioGroup & TRzCheckGroup

- Added new **CustomGlyphImages** property to **TRzRadioGroup** and **TRzCheckGroup**. This property is used to reference an **ImageList** that contains the glyphs to be used for the various states of the

control. This new property should be used instead of the deprecated **CustomGlyphs** property, which is still available strictly for backward compatibility. By referencing an `ImageList` that holds the custom glyphs rather than an embedded bitmap, the actual glyph images are stored only once in the application instead of inside each instance of the control. When populating a `TImageList` for use with **CustomGlyphImages**, each index in the `ImageList` represents a different state. The following tables describe the mapping:

TRzRadioGroup CustomGlyphImages Index Mapping

Index	State	
0	Unchecked	
1	Checked	
2	Unchecked	Pressed
3	Checked	Pressed
4	Unchecked	Disabled
5	Checked	Disabled
6	Unchecked	Hot (Optional)
7	Checked	Hot (Optional)

TRzCheckGroup CustomGlyphImages Index Mapping

Index	State	
0	Unchecked	
1	Checked	
2	Grayed	
3	Unchecked	Pressed
4	Checked	Pressed
5	Grayed	Pressed
6	Unchecked	Disabled
7	Checked	Disabled
8	Grayed	Disabled
9	Unchecked	Hot (Optional)
10	Checked	Hot (Optional)
11	Grayed	Hot (Optional)

- As noted in the above table, with the new **CustomGlyphImages** property, it is now possible to specify "hot" glyphs. That is, separate images to be displayed for a given state when the mouse is positioned over the control.
- Added **ReadOnly**, **ReadOnlyColor**, and **ReadOnlyColorOnFocus** properties to the **TRzCheckGroup**.
- Fixed issue where the widths of individual items (radio buttons in a **TRzRadioGroup**, or check boxes in a **TRzCheckGroup**) would not get calculated correctly if the captions contained tab characters.
- Added new **BannerHeight** property to **TRzRadioGroup**, which is used to control the height of the banner area when the **gsBanner GroupStyle** is used. By default, this property is 0, which instructs the control to determine the height of the Banner based on the font size. If **BannerHeight** is set to a non-zero value, the height of the banner is sized accordingly.

TRzRegIniFile

- Fixed issue with **TRzRegIniFile** component where an INI file with no name would get created if a **SpecialFolder** was specified and the **Path** property remained empty. The component now correctly creates an INI file using the name of the executable.

TRzShellList

- Fixed issue where the **TRzShellList** was unable to navigate to links in My Network Places.

TRzShellTree

- Added the **ItemHeight** property to **TRzShellTree**, which can be used to specify the number of pixels to be used to display each node in the tree. The default value is 0, which instructs the tree view to use the default height as determined by the selected font.

TRzSizePanel

- Fixed issue where a closed hot spot in a **TRzSizePanel** would not appear active the first time the user moved the mouse over the hot spot. This particular issue only occurred in BDS 2006 and RAD Studio 2007 because of the changes made to the VCL regarding **cm_MouseEnter** and **cm_MouseLeave**.

TRzSpinEdit & TRzDBSpinEdit

- Added new **TextHint** and **TextHintVisibleOnFocus** properties to **TRzSpinEdit**. The **TextHint** property allows a developer to specify a prompt that appears inside the edit area of the control, when the control is empty. The text prompt appears grayed. By default, the prompt is automatically hidden when the control receives focus. To keep the prompt visible until the user enters a value, set the **TextHintVisibleOnFocus** property to True.

NOTES:

- Please note that for the **TextHint** to appear, the **AllowBlank** property has to be set to True *and* the Text property has to programmatically be cleared (e.g. `RzSpinEdit1.Text := ''`).
- Please see the comments for **TRzEdit** above for more details on **TextHint** and **TextHintVisibleOnFocus**.

TRzSplitter

- Fixed issue where the HotSpot area would not remain highlighted after dragging the splitter bar in a **TRzSplitter** or the sizing bar in a **TRzSizePanel** (and the mouse was still positioned over the HotSpot area).
- Fixed issue where splitter bar mask in **TRzSplitter** would not get drawn in the correct location if the splitter's parent was offset from the edge of the form *and* the user was running under Vista *and* using Delphi 7. Actual cause of problem was in the **TControl.ClientToParent** method.

TRzStatusPane & descendants

- Fixed problem where **TRzResourceStatus** would report 100% memory utilization on systems with 4GB memory. The problem was fixed by switching to use the **GlobalMemoryStatusEx** function instead **GlobalMemoryStatus**. MSDN reports that **GlobalMemoryStatus** returns incorrect results.
- Fixed problem where deleting the **TImageList** that was currently being referenced by the **TRzGlyphStatus** control would result in an exception in the Form Designer.

What's New in Raize Components 5

TRzTrackBar

- Added new **OnDrawTrack** event to **TRzTrackBar**. This event allows a developer to customize the appearance of the track used for the track bar. For example, a gradient fill could be used to display the track to indicate importance of the selected position. Event handlers are passed the **TCanvas** to use, and the bounds of the track.

TRzTrayIcon

- Fixed issue in **TRzTrayIcon** where the application's Task Bar entry would not get hidden when minimizing the application. This particular issue resulted from the addition of the **Application.MainFormOnTaskBar** property, which was added in Delphi 2007.
- Added new **HideOnStartup** property to **TRzTrayIcon**. When this property is set to True, the Application's MainForm is automatically hidden when the application is started so that only the tray icon is initially visible.
- Modified the header file (RzTray.hpp) that gets generated by the Delphi compiler when compiling components for use in C++Builder. The new modification adds the Shellapi:: namespace to the **_NOTIFYICONDATAA** data structure so that it can be correctly resolved during the linking stage.

TRzTreeView & TRzCheckTree

- Reimplemented **SaveToFile**, **SaveToStream**, **LoadFromFile**, and **LoadFromStream** methods for **TRzTreeView** and **TRzCheckTree**. The change was necessary in order to support using text encodings, which is not possible using the inherited methods from **TCustomTreeView**. Without **TEncoding** support, the files are streams saved with the base methods do not specify any BOM. The problem with this is that the tree view is unable to correctly load the same file/stream (without the BOM).
- Fixed problem where editing a node with a long caption would leave remnants of the selection rectangle (**HideSelection = False**) if the caption was made shorter.
- Added the **ItemHeight** property to **TRzTreeView** and **TRzCheckTree**, which can be used to specify the number of pixels to be used to display each node in the tree. The default value is 0, which instructs the tree view to use the default height as determined by the selected font.
- Modified the display of check boxes in **TRzCheckTree**. Specifically, a few more pixels of space have been added between the check box and the node's caption or image, whichever is adjacent to the check box.

RzCommon Unit

- Removed static link to **SHGetSpecialFolderPath** API function as some users needed to run programs on Windows NT, which does not have this API function in Shell32.dll.
- Renamed the **RemovePrefixChars** function to **RemoveAccelerators** to more accurately describe what the method does.
- Added new **DrawString** function and **DrawStringCentered** procedure. These methods are wrappers around the highly used **DrawText** Windows API function and primarily eliminate the need to case the desired string to a PChar.
- Added new **SendTextMessage** function. This is a wrapper around the **SendMessage** function and is designed specifically for situations where a string needs to be sent in a Window message. **NOTE:** RAD Studio 2009 defines its own **SendTextMessage**, so in RS 2009 and later, the built-in **SendTextMessage** will be used instead.

New Components

The following controls have been added in the Raize Components 5:

TRzPropertyStore

The new **TRzPropertyStore** component is designed to work with a **TRzRegIniFile** component to persist property settings in your applications. To use the **TRzPropertyStore**, simply drop one on the form containing the components whose properties you wish to persist and then connect the **TRzPropertyStore** to a **TRzRegIniFile** component. Next, double-click the component (or edit the Properties property) to display the collection editor for the **TRzPropertyStore.Properties** collection. Use the collection editor to add a new property item (i.e. Component, Property pair). Use the drop down list to select the desired component on the form, and then use the drop down list to select one of the properties of the selected component. Keep adding items to collection to store more properties.

To save the current values of the selected properties call the **TRzPropertyStore.Save** method. To restore property settings persisted previously, call the **TRzPropertyStore.Load** method. The **TRzPropertyStore** component will persist the following property types: Integers, Floating-Point Numbers, Booleans, Enumerations, Sets, Classes, and Collections.

TRzCalculator

The **TRzCalculator** component is a **TRzCustomPanel** descendant provides an interface to perform simple math calculations. Integer and floating point numbers are supported. The calculator is designed to allow for continuous calculations. (For example, $10 + 20 + 30 - 25 =$). Button colors are customizable based on functionality through the **CalculatorColors** property. The control is also fully theme aware. The **TRzCalculator** is used in the **TRzNumericEdit** and **TRzDBNumericEdit** components when the **CalculatorVisible** property is set to True.

Design-Time Support Enhancements

Form Editor

- Added new menu item to create an instance of the new TRzPropertyStore component.
- Updated the positioning of non-visual components in the TRzFormEditor.

Caption Editor

- Registered with the **TRzCaptionProperty** property editor with all controls in Raize Components that support a multi-line Caption. For example, TRzButton.Caption.

TRzCheckList Editor

- The CheckList Editor has been redesigned to make it easier to manage the items and groups in the check list. The editor has also been enhanced to allow setting the ImageIndex and DisabledIndex values for each item, which are used to specify an image to be displayed next each item.
- The CheckList Editor is now resizable.
- When saving check list data in the CheckList Editor (in RS 2009 or later with Unicode support), and the check list contains non-ANSI characters, the file is saved as a UTF8 encoded text file. The CheckList Editor is able to load ANSI or UTF8 encoded text files.

ImageList Editor

- Redesigned the **TRzImageEditor** component editor such the built-in editor dialog for editing image lists is used instead of the previous custom editing dialog. This change will provide seamless support for future image formats supported by the VCL. The Select Image dialog with the stock images is still present.

TRzLabel Editor

- Enhanced the **TRzLabelEditor** so that the developer is able to enter a multi-line caption along with the other properties accessible in the component editor.

TRzPropertyStore Editors

- Added new component editor for TRzPropertyStore to provide quick access to Properties collection and to set the referenced TRzRegIniFile instance.
- Added new property editors for property items used in the TRzPropertyStore.Properties collection. The Component property editor provides a drop down list of all components on the active form, while the Property property editor provides a drop down of all of the published properties available in the selected component.

TRzPageControl Editor

- Removed the "Next Page" and "Previous Page" menu items from the TRzPageControl component editor, and replaced them with a new "Switch to Page" menu item. This new menu displays a cascading menu that lists all of the pages in the control and allows for quickly switching to the desired page.

Radio Button & Check Box Editors

What's New in Raize Components 5

- Updated TRzRadioButtonEditor and TRzCheckBoxEditor by adding AutoSize and OptimizeSize menu items to component editor menu.

TRzSplitter Editor

- Fixed problem where Splitter Editor would not correctly detect the value of the FixedPane property when the Orientation for the splitter was set to vertical.
- Created Sprig classes for the **TRzSplitter** and **TRzSplitterPane** classes to allow the Structure Pane in the IDE to show the true relationship between the **TRzSplitter** and the various controls that are parented by the two embedded panes of the **TRzSplitter**.

TRzTabControl Editor

- Removed the "Next Tab" and "Previous Tab" menu items from the **TRzTabControl** component editor, and replaced them with a new "Switch to Tab" menu item. This new menu displays a cascading menu that lists all of the tabs in the control and allows for quickly switching to the desired tab.

Package Changes

The packages distributed with Raize Components 5 are listed below. The naming convention used in the packages is such that in all future versions of Raize Components, the names of the package DCP files will no longer need to be changed. As always, there are two sets of packages associated with each version of the VCL, one runtime/design package pair for all non-data-aware components and another runtime/design package pair for all data-aware components.

When building an application that uses runtime packages, you may need to deploy the Raize Components runtime packages with your application. The design-time packages are **not** redistributable.

The following table describes the new package BPL files. The IDEs that support each version of the VCL are listed at the bottom of the page:

Package	Pkg Type	VCL	Contents
RaizeComponentsVcl70.bpl	Runtime	7.0	Non-Data-Aware components
RaizeComponentsVcl90.bpl	Runtime	9.0	Non-Data-Aware components
RaizeComponentsVcl100.bpl	Runtime	10.0	Non-Data-Aware components
RaizeComponentsVcl120.bpl	Runtime	12.0	Non-Data-Aware components
RaizeComponentsVclDb70.bpl	Runtime	7.0	Data-Aware components
RaizeComponentsVclDb90.bpl	Runtime	9.0	Data-Aware components
RaizeComponentsVclDb100.bpl	Runtime	10.0	Data-Aware components
RaizeComponentsVclDb120.bpl	Runtime	12.0	Data-Aware components
RaizeComponentsVcl_Design70.bpl	Design	7.0	Raize Components 5.x
RaizeComponentsVcl_Design90.bpl	Design	9.0	Raize Components 5.x
RaizeComponentsVcl_Design100.bpl	Design	10.0	Raize Components 5.x
RaizeComponentsVcl_Design120.bpl	Design	12.0	Raize Components 5.x
RaizeComponentsVclDb_Design70.bpl	Design	7.0	Raize Components 5.x (Data-Aware)
RaizeComponentsVclDb_Design90.bpl	Design	9.0	Raize Components 5.x (Data-Aware)
RaizeComponentsVclDb_Design100.bpl	Design	10.0	Raize Components 5.x (Data-Aware)
RaizeComponentsVclDb_Design120.bpl	Design	12.0	Raize Components 5.x (Data-Aware)

If you are building your own packages that are dependent on the Raize Components units, you will need to add the appropriate DCP package information file to the **requires** clause of your package. With the new naming convention, the name of the DCP file is same regardless of the version of the VCL.

The following table lists the new package DCP files.

DCP File	Contents
RaizeComponentsVcl.dcp	Non-Data-Aware package information
RaizeComponentsVclDb.dcp	Data-Aware package information

VCL Versions

The following table lists the IDEs that support each version of the VCL:

VCL	IDE
7.0	Delphi 7
9.0	Delphi 2005
10.0	Borland Developer Studio 2006, Delphi 2006, C++Builder 2006, Turbo Delphi 2006, Turbo C++Builder 2006, CodeGear RAD Studio 2007, Delphi 2007, C++Builder 2007
12.0	CodeGear RAD Studio 2009, Delphi 2009, C++Builder 2009

Migration Issues

Raize Components 5 is backward compatible with earlier versions of Raize Components. That is, if you created an application using an earlier version of Raize Components, you will be able to load your forms under version 5.

Package Changes

If you are migrating from Raize Components 4, there should be no changes necessary in your packages. Simply rebuild them with Raize Components 5 installed.

If you are using an older version of Raize Components, you will need to modify the requires clauses of your packages to reflect the new package names. (Please see the Package Changes section for more details). As a result, there are some potential migration issues for Delphi and C++Builder users.

Delphi Users

If you have an existing project that used an earlier version of Raize Components, the option file (*.dof) for that project will contain references to the Raize Components 3.0 or 2.x runtime packages. The DefProj.dof file in your Delphi Bin directory is automatically updated during the install program, but your existing DOF files are not modified.

IMPORTANT: The runtime package list in a project's DOF file is only used if the project is built with runtime packages. The following steps are not necessary if you are **not** using runtime packages for your project.

If you are using runtime packages, then you should open the DOF file and look for the Packages entry in the [Directories] section. There you will find the two runtime packages for Raize Components listed. The actual file names will depend on the version of Raize Components you had previous to 5.

***nn represents VCL version number (e.g. 70 for VCL 7.0)**

Old Version:	Old Runtime Package:	Change to:
2.5	Rz25NDnn	RaizeComponentsVcl
2.5	Rz25DBnn	RaizeComponentsVclDb
2.51	Rz251Nnn	RaizeComponentsVcl
2.51	Rz251Dnn	RaizeComponentsVclDb
2.52	Rz252Nnn	RaizeComponentsVcl
2.52	Rz252Dnn	RaizeComponentsVclDb
3.x	Rz30Ctlnnn	RaizeComponentsVcl
3.x	Rz30DBCtlnnn	RaizeComponentsVclDb

Simply change the runtime package names to the 5.0 versions also listed in the above table, and then save the project option file.

C++Builder Users

What's New in Raize Components 5

If you have an existing project that used an earlier version of Raize Components, the project (*.bpr) file for that project will contain references to the Raize Components 3.0 or 2.x package BPI files. The Default.bpr file in your Builder Bin directory is automatically updated during the install program, but your existing BPR files are not modified.

As a result, you should open the BPR file and look for the line starting with PACKAGES. There you will find the two BPI files for Raize Components. The actual file names will depend on the version of Raize Components you had previous to version 5.

***nn represents VCL version number (e.g. 70 for VCL 7.0)**

Old Version:	Old BPI Files:	Change to:
2.52	Rz252Nnn.bpi	RaizeComponentsVcl.bpi
2.52	Rz252Dnn.bpi	RaizeComponentsVclDb.bpi
3.0	Rz30CtIsnn.bpi	RaizeComponentsVcl.bpi
3.0	Rz30DBCtIsnn.bpi	RaizeComponentsVclDb.bpi

Simply change the BPI file names to the 5.0 versions also listed in the above table, and then save the project file.