



## **RichViewActions**

RichViewActions © trichview.com

# Table of Contents

<b>Part I RichViewActions</b>	<b>16</b>
<b>1..Overview</b>	<b>16</b>
Using RichViewActions	16
Style templates	20
History	21
<b>2..Components</b>	<b>32</b>
<b>TCustomRVFontComboBox</b>	<b>33</b>
Properties	34
TCustomRVFontComboBox.ActionFont	34
TCustomRVFontComboBox.Editor	35
<b>TCustomRVFontListBox</b>	<b>35</b>
Properties	35
TCustomRVFontListBox.ActionFont	36
TCustomRVFontListBox.AllowFocusEditor	36
TCustomRVFontListBox.Editor	37
<b>TRulerItemSelector</b>	<b>37</b>
Properties	37
TCustomRulerItemSelector.Ruler	38
<b>TRVAControlPanel</b>	<b>38</b>
Properties	40
TRVAControlPanel.ActionsEnabled	41
TRVAControlPanel.AddColorNameToHints	41
TRVAControlPanel.AutoDeleteUnusedStyles	42
TRVAControlPanel.ColorDialog	42
TRVAControlPanel.ColorDialogInterface	42
TRVAControlPanel.DefaultColor	42
TRVAControlPanel.DefaultControl	43
TRVAControlPanel.DefaultCustomFilterIndex	43
TRVAControlPanel.DefaultDocParameters	43
TRVAControlPanel.DefaultExt	44
TRVAControlPanel.DefaultFileFormat	44
TRVAControlPanel.DefaultFileName	44
TRVAControlPanel.DefaultMargin	45
TRVAControlPanel.DialogFontName	45
TRVAControlPanel.DialogFontSize	45
TRVAControlPanel.DialogPosition	46
TRVAControlPanel.DialogZoomPercent	46
TRVAControlPanel.DownloadInterface	46
TRVAControlPanel.FirstPageNumber	47
TRVAControlPanel.Header, Footer	47
TRVAControlPanel.HelpType	47
TRVAControlPanel.HTMLComponent (deprecated)	51
TRVAControlPanel.Language	51
TRVAControlPanel.MetafileCompatibility	51
TRVAControlPanel.PixelBorders	52
TRVAControlPanel.RVFFilter	52
TRVAControlPanel.RVFLocalizable	52
TRVAControlPanel.RVFormatTitle	53
TRVAControlPanel.RVPrint	53
TRVAControlPanel.RVStylesFilter, RVStylesExt	53

TRVAControlPanel.SearchScope.....	53
TRVAControlPanel.ShowSoftPageBreaks .....	54
TRVAControlPanel.SpellInterface.....	54
TRVAControlPanel.TableGridStyle.....	54
TRVAControlPanel.UnitsDisplay .....	55
TRVAControlPanel.UnitsProgram.....	55
TRVAControlPanel.UseDefaultHelpFile.....	56
TRVAControlPanel.UseHelpFiles .....	56
TRVAControlPanel.UserInterface.....	56
TRVAControlPanel.UseTextCodePageDialog .....	57
TRVAControlPanel.UseXPThemes .....	57
TRVAControlPanel.XMLComponent.....	58
TRVAControlPanel.XMLFilter.....	58
TRVAControlPanel.XMLLocalizable.....	58
Methods .....	59
TRVAControlPanel.Activate.....	59
TRVAControlPanel.GetRealDialogFontName.....	59
TRVAControlPanel.InitImportPictures, DoneImportPictures, DoImportPicture.....	59
Events .....	60
TRVAControlPanel.OnAddStyle.....	60
TRVAControlPanel.OnBackgroundChange.....	61
TRVAControlPanel.OnChoosePicture.....	61
TRVAControlPanel.OnCreateForm.....	62
TRVAControlPanel.OnCustomFileOperation.....	62
TRVAControlPanel.OnDownload.....	64
TRVAControlPanel.OnGetActionControlCoords .....	64
TRVAControlPanel.OnGetHeaderFooterCode.....	65
TRVAControlPanel.OnMarginsChanged.....	65
TRVAControlPanel.OnStyleNeeded.....	66
TRVAControlPanel.OnViewChanged.....	66
<b>TRVAPopupMenu .....</b>	<b>66</b>
Properties .....	68
TRVAPopupMenu.ActionList.....	69
TRVAPopupMenu.MaxSuggestionsCount.....	69
Events .....	69
TRVAPopupMenu.OnLiveSpellAdd.....	69
TRVAPopupMenu.OnLiveSpellGetSuggestions.....	70
TRVAPopupMenu.OnLiveSpellIgnoreAll.....	70
TRVAPopupMenu.OnLiveSpellWordReplace.....	71
<b>TRVFontCharsetComboBox .....</b>	<b>71</b>
Properties .....	72
TRVFontCharsetComboBox.AddDefaultCharset.....	73
TRVFontCharsetComboBox.DefaultCharsetCaption .....	73
TRVFontCharsetComboBox.FontName.....	73
<b>TRVFontComboBox .....</b>	<b>73</b>
Properties .....	74
TRVFontComboBox.AutoCharset.....	75
TRVFontComboBox.DropDownWidth.....	75
TRVFontComboBox.ItemHeight.....	75
TRVFontComboBox.Preview.....	76
TRVFontComboBox.SymbolPreviewString.....	76
<b>TRVFontListBox .....</b>	<b>76</b>
Properties .....	77
TRVFontListBox.AutoCharset.....	78
TRVFontListBox.ItemHeight.....	78
TRVFontListBox.Preview .....	78
TRVFontListBox.SymbolPreviewString.....	78
<b>TRVFontSizeComboBox .....</b>	<b>79</b>

Properties .....	79
TRVFontSizeComboBox.FontName .....	80
<b>TRVRuler .....</b>	<b>80</b>
Properties .....	81
TCustomRuler.BiDiModeRuler .....	82
TCustomRuler.Flat .....	83
TCustomRuler.IndentColor, MarginColor, RulerColor, TickColor, RulerColorPageEnd .....	83
TCustomRuler.IndentSaturation, MarginSaturation, RulerSaturation .....	84
TCustomRuler.RulerType .....	84
TCustomRuler.ScreenRes .....	84
TCustomRuler.SkinType .....	85
TCustomRuler.UnitsDisplay, UnitsProgram .....	85
TCustomRuler.UnitsPixelsPerInch .....	85
TCustomRuler.Zoom .....	86
TRVRuler.RichViewEdit .....	86
Methods .....	86
TRVRuler.UpdateRulerMargins .....	87
<b>TRVStyleTemplateComboBox, TRVStyleTemplateListBox .....</b>	<b>87</b>
Properties .....	88
TRVStyleTemplateComboBox.*ImageIndex .....	89
TRVStyleTemplateComboBox.ControlPanel .....	89
TRVStyleTemplateComboBox.Editor .....	89
TRVStyleTemplateComboBox.Images .....	90
TRVStyleTemplateComboBox.ShowAllStyles .....	90
TRVStyleTemplateComboBox.ShowClearFormat .....	90
TRVStyleTemplateComboBox.SmartHeadings .....	90
Methods .....	91
TRVStyleTemplateComboBox.Localize .....	91
Events .....	91
TRVStyleTemplateComboBox.OnClickAllStyles .....	92
<b>3.Actions .....</b>	<b>92</b>
<b>File .....</b>	<b>96</b>
TrvActionCustomFileIO .....	96
Properties .....	97
TrvActionCustomFileIO.CustomFilter .....	98
TrvActionCustomIO .....	98
Properties .....	99
TrvActionCustomIO.AutoUpdateFileName .....	99
TrvActionCustomIO.DialogTitle .....	100
TrvActionCustomIO.FileName .....	100
TrvActionCustomIO.InitialDir .....	100
TrvActionCustomNew .....	101
Events .....	101
TrvActionCustomNew.OnNew .....	101
Properties .....	102
TrvActionCustomNew.ActionSave .....	102
TrvActionNew .....	103
Properties .....	104
TrvActionNew.StyleTemplates, ApplyStyleTemplates .....	104
Methods .....	105
TrvActionNew.Reset .....	105
TrvActionOpen .....	106
Properties .....	107
TrvActionOpen.ActionNew .....	108
TrvActionOpen.CustomFilter .....	108
TrvActionOpen.DialogTitle .....	108
TrvActionOpen.Filter .....	108



TrvActionOpen.InitialDir.....	109
TrvActionOpen.LastFilterIndex.....	109
Methods.....	109
TrvActionOpen.LoadFile.....	109
Events.....	110
TrvActionOpen.OnOpenFile.....	110
TrvActionExport.....	110
Properties.....	111
TrvActionExport.CreateDirectoryForHTMLImages.....	113
TrvActionExport.Filter.....	113
Methods.....	113
TrvActionExport.ExportToFile.....	114
TrvActionPageSetup.....	114
Properties.....	115
TrvActionPageSetup.MarginsUnits.....	115
TrvActionPageSetup.UnitsPixelsPerInch.....	116
Events.....	116
TrvActionPageSetup.OnChange.....	116
TrvActionPrint.....	116
Properties.....	117
TrvActionPrint.Title.....	118
TrvActionPrintPreview.....	118
Properties.....	119
TrvActionPrintPreview.ActionPageSetup.....	119
TrvActionPrintPreview.ActionPrint.....	119
TrvActionPrintPreview.Maximized.....	120
Events.....	120
TrvActionPrintPreview.OnGetPreviewFormClass.....	120
TrvActionQuickPrint.....	120
Properties.....	121
TrvActionQuickPrint.Title.....	122
TrvActionSave.....	122
Properties.....	123
TrvActionSave.ActionSaveAs.....	123
TrvActionSave.CreateDirectoryForHTMLImages.....	124
TrvActionSave.DisableWhenUnmodified.....	124
TrvActionSave.Documents.....	124
TrvActionSave.LostFormatWarning.....	125
TrvActionSave.SuppressNextErrorMessage.....	125
Methods.....	125
TrvActionSave.CanCloseDoc.....	125
TrvActionSave.FindDoc.....	126
Events.....	126
TrvActionSave.OnDocumentFileChange.....	126
TrvActionSave.OnSaving, OnSave.....	126
TrvActionSaveAs.....	127
Properties.....	128
TrvActionSaveAs.ActionSave.....	129
TrvActionSaveAs.Filter.....	129
TrvCustomPrintAction.....	129
<b>Edit.....</b>	<b>130</b>
TrvActionCharCase.....	130
TrvActionCopy.....	131
TrvActionCut.....	131
TrvActionFind.....	132
Properties.....	133
TrvActionFind.ActionReplace.....	133
TrvActionFind.FindText.....	134

Methods .....	134
TrvActionFind.CloseDialog .....	134
TrvActionFindNext .....	134
Properties .....	135
TrvActionFindNext.ActionFind .....	135
TrvActionPaste .....	135
TrvActionPasteAsText .....	136
TrvActionPasteSpecial .....	137
Properties .....	138
TrvActionPasteSpecial.StoreFileNameInItemName .....	138
Methods .....	139
TrvActionPasteSpecial.AddFormat .....	139
Events .....	139
TrvActionPasteSpecial.OnCanPaste .....	139
TrvActionPasteSpecial.OnCustomPaste .....	140
TrvActionPasteSpecial.OnShowing .....	140
TrvActionRedo .....	140
TrvActionReplace .....	141
Properties .....	141
TrvActionReplace.ActionFind .....	142
TrvActionReplace.FindText, ReplaceText .....	142
TrvActionReplace.ShowReplaceAllSummary .....	143
Methods .....	143
TrvActionReplace.CloseDialog .....	143
Events .....	143
TrvActionReplace.OnReplaceAllEnd .....	143
TrvActionReplace.OnReplaceAllStart .....	143
TrvActionReplace.OnReplacing .....	144
TrvActionSelectAll .....	144
TrvActionUndo .....	144
TrvCustomEditAction .....	145
<b>Font .....</b>	<b>146</b>
TrvActionFontAllCaps .....	146
TrvActionFontBackColor .....	147
TrvActionFontBold .....	147
TrvActionFontColor .....	148
TrvActionFontCustomColor .....	149
TrvActionFontEx .....	149
Properties .....	150
TrvActionFontEx.AutoApplySymbolCharset .....	151
TrvActionFontEx.BackColor .....	152
TrvActionFontEx.CharScale .....	152
TrvActionFontEx.CharSpacing .....	152
TrvActionFontEx.FontSizeDouble .....	153
TrvActionFontEx.FontStyleEx .....	153
TrvActionFontEx.PreviewInList .....	153
TrvActionFontEx.SubSuperScriptType .....	153
TrvActionFontEx.UnderlineColor .....	154
TrvActionFontEx.UnderlineType .....	154
TrvActionFontEx.ValidProperties .....	154
TrvActionFontEx.VShift .....	155
Events .....	155
TrvActionFontEx.OnShowingDialog .....	155
TrvActionFontGrow .....	155
Properties .....	156
TrvActionFontGrow.MaxSize .....	157
TrvActionFontGrowOnePoint .....	157
Properties .....	157

TrvActionFontGrowOnePoint.MaxSize.....	158
TrvActionFontItalic.....	158
TrvActionFontOverline.....	159
TrvActionFonts.....	159
Properties.....	160
TrvActionFonts.Font.....	161
TrvActionFonts.UserInterface.....	161
TrvActionFontShrink.....	161
Properties.....	162
TrvActionFontShrink.MinSize.....	163
TrvActionFontShrinkGrow.....	163
Properties.....	163
TrvActionFontShrinkGrow.Percent.....	164
TrvActionFontShrinkOnePoint.....	164
Properties.....	165
TrvActionFontShrinkOnePoint.MinSize.....	165
TrvActionFontStrikeout.....	166
TrvActionFontStyle.....	166
TrvActionFontStyleEx.....	167
TrvActionFontUnderline.....	168
TrvActionSSScript.....	168
TrvActionSubscript.....	169
TrvActionSuperscript.....	170
TrvActionTextBiDi.....	170
TrvActionTextLTR.....	171
TrvActionTextRTL.....	172
TrvActionTextStyles.....	173
<b>Paragraph.....</b>	<b>173</b>
TrvActionAlignCenter.....	174
TrvActionAlignCustomJustify.....	175
Properties.....	175
TrvActionAlignCustomJustify.LastLineAlignment, UseLastLineAlignment.....	176
TrvActionAlignDistribute.....	176
TrvActionAlignJustify.....	177
TrvActionAlignLeft.....	178
TrvActionAlignment.....	178
TrvActionAlignRight.....	179
TrvActionClearBoth.....	180
TrvActionClearLeft.....	180
TrvActionClearNone.....	181
TrvActionClearRight.....	181
TrvActionClearTextFlow.....	182
TrvActionCustomParaListSwitcher.....	182
Properties.....	183
TrvActionCustomParaListSwitcher.IndentStep.....	184
TrvActionCustomParaListSwitcher.ListLevels.....	184
TrvActionIndent.....	184
Properties.....	185
TrvActionIndent.IndentStep.....	186
TrvActionIndentDec.....	186
TrvActionIndentInc.....	187
Properties.....	187
TrvActionIndentInc.IndentMax.....	188
TrvActionLineSpacing.....	188
TrvActionLineSpacing100.....	189
TrvActionLineSpacing150.....	189
TrvActionLineSpacing200.....	190
TrvActionParaBiDi.....	191

TrvActionParaBorder.....	191
Properties .....	192
TrvActionParaBorder.Background.....	193
TrvActionParaBorder.Border.....	193
TrvActionParaBorder.UserInterface.....	194
TrvActionParaBorder.ValidProperties.....	194
Events .....	194
TrvActionParaBorder.OnShowingDialog.....	194
TrvActionParaBullets.....	195
TrvActionParaColor.....	195
TrvActionParaColorAndPadding.....	196
Properties .....	197
TrvActionParaColorAndPadding.Padding.....	197
TrvActionParaColorAndPadding.UsePadding.....	198
TrvActionParaCustomColor.....	198
TrvActionParagraph.....	198
Properties .....	199
TrvActionParagraph.Alignment.....	200
TrvActionParagraph.DeleteAllTabs.....	200
TrvActionParagraph.FirstIndent.....	201
TrvActionParagraph.KeepLinesTogether.....	201
TrvActionParagraph.KeepWithNext.....	201
TrvActionParagraph.LastLineAlignment.....	202
TrvActionParagraph.LeftIndent.....	202
TrvActionParagraph.LineSpacing.....	202
TrvActionParagraph.LineSpacingType.....	203
TrvActionParagraph.OutlineLevel.....	203
TrvActionParagraph.RightIndent.....	203
TrvActionParagraph.SpaceAfter.....	203
TrvActionParagraph.SpaceBefore.....	204
TrvActionParagraph.Tabs.....	204
TrvActionParagraph.TabsToDelete.....	204
TrvActionParagraph.UserInterface.....	205
TrvActionParagraph.ValidProperties.....	205
Events .....	205
TrvActionParagraph.OnShowingDialog.....	205
TrvActionParaList.....	206
Properties .....	206
TrvActionParaList.IndentStep.....	207
TrvActionParaList.UpdateAllActionsOnForm.....	207
TrvActionParaList.ActionParaNumbering.....	208
TrvActionParaList.ActionParaBullets.....	208
TrvActionParaLTR.....	208
TrvActionParaNumbering.....	209
TrvActionParaRTL.....	210
TrvActionParaStyles.....	210
TrvActionWordWrap.....	211
<b>Styles .....</b>	<b>212</b>
TrvActionStyleTemplates.....	212
Properties .....	213
TrvActionStyleTemplates.Images.....	214
TrvActionStyleTemplates.StandardStyleTemplates.....	214
TrvActionStyleTemplates.*StyleImageIndex.....	215
TrvActionAddStyleTemplate.....	215
Properties .....	216
TrvActionAddStyleTemplate.ActionStyleTemplates.....	216
TrvActionStyleInspector.....	216
Properties .....	217

TrvActionStyleInspector.Images.....	218
TrvActionStyleInspector.FontImageIndex, ParaImageIndex.....	218
Methods.....	218
TrvActionStyleInspector.UpdateInfo.....	218
Events.....	218
TrvActionCustomInfoWindow.OnShowing.....	219
TrvActionClearFormat.....	219
TrvActionClearTextFormat.....	219
<b>Insert .....</b>	<b>220</b>
TrvActionBookmarks.....	221
Properties.....	221
TrvActionBookmarks.AllowedCharacters.....	222
TrvActionBookmarks.ScrollToCenter.....	222
TrvActionInsertCaption.....	223
Properties.....	224
TrvActionInsertCaption.ExcludeLabel.....	224
TrvActionInsertCaption.InsertAbove.....	225
TrvActionInsertCustomPageNumber.....	225
Properties.....	225
TrvActionInsertCustomPageNumber.NumberType.....	226
TrvActionInsertEquation.....	226
Properties.....	227
TrvActionInsertEquation.UserInterface.....	228
TrvActionInsertEquation.Expression.....	228
TrvActionInsertFile.....	228
Properties.....	229
TrvActionInsertFile.Filter.....	230
Methods.....	230
TrvActionInsertFile.InsertFile.....	230
TrvActionInsertHLine.....	231
Properties.....	231
TrvActionInsertHLine.Color.....	232
TrvActionInsertHLine.Style.....	232
TrvActionInsertHLine.Width.....	232
TrvActionInsertHyperlink.....	232
Properties.....	234
TrvActionInsertHyperlink.ActionStyleTemplates.....	235
TrvActionInsertHyperlink.AutoAddHyperlinkStyleTemplate.....	235
TrvActionInsertHyperlink.BackColor.....	235
TrvActionInsertHyperlink.Color.....	236
TrvActionInsertHyperlink.Cursor.....	236
TrvActionInsertHyperlink.HoverBackColor.....	236
TrvActionInsertHyperlink.HoverColor.....	237
TrvActionInsertHyperlink.HoverEffects.....	237
TrvActionInsertHyperlink.HoverUnderlineColor.....	237
TrvActionInsertHyperlink.ScrollToCenter.....	238
TrvActionInsertHyperlink.SetToPictures.....	238
TrvActionInsertHyperlink.SpaceFiller.....	238
TrvActionInsertHyperlink.Style.....	238
TrvActionInsertHyperlink.StyleTemplateName.....	239
TrvActionInsertHyperlink.UnderlineColor.....	239
TrvActionInsertHyperlink.ValidProperties.....	239
Methods.....	240
TrvActionInsertHyperlink.DetectURL.....	240
TrvActionInsertHyperlink.EncodeTarget.....	241
TrvActionInsertHyperlink.GetHyperlinkStyleNo.....	241
TrvActionInsertHyperlink.GetNormalStyleNo.....	242
TrvActionInsertHyperlink.GoToLink.....	242

TrvActionInsertHyperlink.SetTags .....	243
TrvActionInsertHyperlink.TerminateHyperlink .....	243
Events .....	244
TrvActionInsertHyperlink.OnApplyHyperlinkToItem .....	244
TrvActionInsertHyperlink.OnGetHyperlinkTargetFromItem .....	244
TrvActionInsertHyperlink.OnHyperlinkForm .....	244
TrvActionInsertNumber .....	245
TrvActionInsertNumSequence .....	246
Properties .....	246
TrvActionInsertNumSequence.NumberType .....	247
TrvActionInsertNumSequence.SeqName .....	247
TrvActionInsertNumSequence.UseDefaults .....	247
TrvActionInsertPageBreak .....	248
TrvActionInsertPageCount .....	248
TrvActionInsertPageNumber .....	249
TrvActionInsertPicture .....	249
Properties .....	250
TrvActionInsertPicture.BackgroundColor .....	251
TrvActionInsertPicture.BorderWidth, BorderColor .....	251
TrvActionInsertPicture.DefaultExt .....	252
TrvActionInsertPicture.Filter .....	252
TrvActionInsertPicture.MaxImageSize .....	252
TrvActionInsertPicture.OuterHSpacing, OuterVSpacing .....	253
TrvActionInsertPicture.Spacing .....	253
TrvActionInsertPicture.StoreFileNameInItemName .....	253
TrvActionInsertPicture.VAlign .....	253
Events .....	254
TrvActionInsertPicture.OnInserting .....	254
TrvActionInsertSymbol .....	254
Properties .....	255
TrvActionInsertSymbol.CharCode .....	255
TrvActionInsertSymbol.FontName .....	256
TrvActionInsertSymbol.Protection .....	256
TrvActionInsertSymbol.UseCurrentFont .....	256
TrvActionInsertText .....	256
Events .....	257
TrvActionInsertText.OnInsertText .....	257
<b>Notes and text boxes .....</b>	<b>257</b>
TrvActionCustomInsertSidenote .....	258
Properties .....	258
TrvActionCustomInsertSidenote.BoxPosition .....	259
TrvActionCustomInsertSidenote.BoxProperties .....	260
TrvActionCustomInsertSidenote.StyleTemplateName .....	260
TrvActionEditNote .....	260
Properties .....	262
TrvActionEditNote.JumpToNextNote .....	262
Events .....	263
TrvActionEditNote.OnStartEditNote .....	263
TrvActionInsertEndnote .....	263
TrvActionInsertFootnote .....	264
TrvActionInsertNote .....	265
Properties .....	265
TrvActionInsertNote.ActionEditNote .....	266
TrvActionInsertNote.Font .....	266
TrvActionInsertSidenote .....	267
Properties .....	268
TrvActionInsertSidenote.RefStyleTemplateName .....	269
TrvActionInsertTextBox .....	269

<b>Tables .....</b>	<b>270</b>
TrvActionInsertTable.....	271
Properties .....	272
TrvActionInsertTable.BackgroundImage.....	273
TrvActionInsertTable.BackgroundStyle.....	273
TrvActionInsertTable.BestWidth.....	274
TrvActionInsertTable.BorderColor.....	274
TrvActionInsertTable.BorderHSpacing.....	274
TrvActionInsertTable.BorderLightColor.....	274
TrvActionInsertTable.BorderStyle.....	275
TrvActionInsertTable.BorderVSpacing.....	275
TrvActionInsertTable.BorderWidth.....	275
TrvActionInsertTable.CellBorderColor.....	275
TrvActionInsertTable.CellBorderLightColor.....	276
TrvActionInsertTable.CellBorderStyle.....	276
TrvActionInsertTable.CellBorderWidth.....	276
TrvActionInsertTable.CellHPadding,CellVPadding.....	276
TrvActionInsertTable.CellHSpacing.....	277
TrvActionInsertTable.CellVSpacing.....	277
TrvActionInsertTable.ColBandSize, RowBandSize.....	277
TrvActionInsertTable.ColCount.....	278
TrvActionInsertTable.Color.....	278
TrvActionInsertTable.HeadingRowCount.....	278
TrvActionInsertTable.HOutermostRule.....	278
TrvActionInsertTable.HRuleColor.....	278
TrvActionInsertTable.HRuleWidth.....	279
TrvActionInsertTable.ItemText.....	279
TrvActionInsertTable.RowCount.....	279
TrvActionInsertTable.RowsKeepTogether.....	279
TrvActionInsertTable.RowsVAlign.....	280
TrvActionInsertTable.TableOptions .....	280
TrvActionInsertTable.TablePrintOptions .....	280
TrvActionInsertTable.VisibleBorders.....	280
TrvActionInsertTable.VOutermostRule.....	280
TrvActionInsertTable.VRuleColor.....	281
TrvActionInsertTable.VRuleWidth.....	281
TrvActionInsertTable's row and column colors.....	281
Methods .....	281
TrvActionInsertTable.ShowTableSizeDialog.....	282
Events .....	282
TrvActionInsertTable.OnCloseTableSizeDialog.....	282
TrvActionInsertTable.OnInserting.....	282
TrvActionTableCell.....	282
TrvActionTableCellAllBorders.....	283
TrvActionTableCellBorder.....	284
TrvActionTableCellBottomBorder.....	284
TrvActionTableCellLeftBorder.....	285
TrvActionTableCellNoBorders.....	286
TrvActionTableCellRightBorder.....	287
TrvActionTableCellRotation.....	287
TrvActionTableCellRotationNone.....	288
TrvActionTableCellRotation90.....	289
TrvActionTableCellRotation180.....	289
TrvActionTableCellRotation270.....	290
TrvActionTableCellTopBorder.....	291
TrvActionTableCellVAlign.....	291
TrvActionTableCellVAlignBottom.....	292
TrvActionTableCellVAlignDefault.....	293

TrvActionTableCellVAlignMiddle.....	293
TrvActionTableCellVAlignTop.....	294
TrvActionTableDeleteCols.....	295
TrvActionTableDeleteRows.....	295
TrvActionTableDeleteTable.....	296
TrvActionTableGrid.....	296
TrvActionTableInsertColLeft.....	297
TrvActionTableInsertColRight.....	297
TrvActionTableInsertRowsAbove.....	298
TrvActionTableInsertRowsBelow.....	299
Properties.....	299
TrvActionTableInsertRowsBelow.AllowMultiple.....	300
TrvActionTableMergeCells.....	300
TrvActionTableMultiCellAttributes.....	301
TrvActionTableProperties.....	301
Properties.....	302
TrvActionTableProperties.ActionInsertTable, UpdateAllInsertTableActions.....	303
TrvActionTableProperties.BackgroundGraphicFilter.....	303
TrvActionTableProperties.DefaultChecked, DefaultPersistent.....	303
TrvActionTableRCBase.....	304
TrvActionTableSelectCell.....	305
TrvActionTableSelectCols.....	305
TrvActionTableSelectRows.....	306
TrvActionTableSelectTable.....	306
TrvActionTableSort.....	307
TrvActionTableSplit.....	308
TrvActionTableSplitCells.....	308
TrvActionTableToText.....	309
<b>Spelling check and thesaurus.....</b>	<b>310</b>
TrvActionSpellingCheck.....	310
TrvActionThesaurus.....	310
<b>Other actions.....</b>	<b>311</b>
TrvAction.....	311
Properties.....	312
TrvAction.Control.....	312
TrvActionBackground.....	313
Properties.....	314
TrvActionBackground.CanChangeMargins.....	314
TrvActionBackground.ImageFileName.....	315
Events.....	315
TrvActionBackground.OnChange.....	315
TrvActionColor.....	315
TrvActionUserDefinedColor.....	316
Properties.....	316
TrvActionUserDefinedColor.UseOpacity.....	317
Events.....	317
TrvActionUserDefinedColor.OnColorSelected.....	317
TrvActionUserDefinedColor.OnGetColor.....	318
TrvActionCustomColor.....	318
Properties.....	318
TrvActionCustomColor.CallerControl.....	319
TrvActionCustomColor.Color.....	319
TrvActionCustomColor.Opacity.....	320
TrvActionCustomColor.UserInterface.....	320
Events.....	321
TrvActionCustomColor.OnShowColorPicker.....	321
TrvActionCustomColor.OnHideColorPicker.....	321
TrvActionEvent.....	321



Events .....	322
TrvActionEvent.OnExecute.....	322
TrvActionEvent.OnUpdate.....	322
TrvActionFillColor.....	322
Properties .....	323
TrvActionFillColor.AllowApplyingTo.....	324
TrvActionHide.....	324
TrvActionItemProperties.....	324
Properties .....	325
TrvActionItemProperties.ActionInsertHLine.....	326
TrvActionItemProperties.ActionInsertTable, UpdateAllInsertTableActions.....	326
TrvActionItemProperties.BackgroundGraphicFilter.....	327
TrvActionItemProperties.DefaultChecked, DefaultPersistent.....	327
TrvActionItemProperties.GraphicFilter .....	328
TrvActionItemProperties.StoreImageFileName.....	328
Events .....	328
TrvActionItemProperties.OnCanApply.....	328
TrvActionItemProperties.OnCustomItemPropertiesDialog.....	329
TrvActionRemoveHyperlinks .....	329
Properties .....	330
TrvActionRemoveHyperlinks.ActionInsertHyperlink.....	330
TrvActionRemovePageBreak.....	330
TrvActionShowSpecialCharacters.....	331
Properties .....	332
TrvActionShowSpecialCharacters.ShowCheckpoints.....	332
TrvActionVAlign.....	333
TrvCustomAction .....	333
Properties .....	334
TrvCustomAction.Caption.....	334
TrvCustomAction.ControlPanel.....	335
TrvCustomAction.Disabled.....	335
TrvCustomAction.Hint.....	335
Methods .....	335
TrvCustomAction.GetControlPanel.....	336
TrvCustomEditorAction .....	336
Properties .....	336
TrvCustomEditorAction.Form.....	337
TrvCustomEditorAction.SubDocEditor .....	337
Events .....	337
TrvCustomEditorAction.OnFormCreate.....	338
TrvCustomEditorAction.OnShowing, OnHide.....	338
<b>4. Localization of RichViewActions .....</b>	<b>340</b>
<b>RVALocalize unit .....</b>	<b>341</b>
Procedures and functions .....	342
Functions for progress messages.....	343
RVA_EnumLanguages procedure.....	344
RVA_FillLanguageList procedure.....	344
RVA_GetCharset function.....	345
RVA_GetHelpFile function.....	345
RVA_GetLanguageName function .....	346
RVA_GetS function .....	346
RVA_SetHelpFile procedure.....	347
RVA_SwitchLanguage procedure.....	347
RVA_TranslateUnits procedure.....	347
Types .....	348
TRVALanguageName .....	348
TRVALocString, TRVALocStrings, TRVALocStringList.....	348

<b>RichViewActions unit</b> .....	<b>349</b>
Procedures and functions .....	349
RVA_ChoseLanguage function .....	349
RVA_GetColorName function .....	350
RVA_LocalizeForm procedure .....	350
<b>Other units</b> .....	<b>351</b>
GetAddictSpellLanguage, GetAddictThesLanguage functions .....	351
RVLocalizeRuler procedure .....	351
RVGetHelpKeyword and RVSetHelpKeyword .....	352
<b>5..Interfaces for third-party components</b> .....	<b>352</b>
<b>Download interfaces</b> .....	<b>353</b>
TRVACIDownloadInterface .....	353
TRVACustomDownloadInterface .....	354
TRVAIndyDownloadInterface .....	354
<b>Spelling check interfaces</b> .....	<b>355</b>
TRVSpellInterface .....	356
TRVAddictSpellInterface .....	356
TRVAASpellInterface .....	357
TRVACustomSpellInterface .....	357
TRVADXSpellInterface .....	358
TRVAHunSpellInterface .....	359
TRVAPolarSpellInterface .....	359
<b>Color dialog interfaces</b> .....	<b>360</b>
<b>6..Third-party and additional tools</b> .....	<b>360</b>
<b>Components</b> .....	<b>361</b>
Spelling checkers .....	362
Addict 3 and 4 .....	362
ASpell .....	363
ExpressSpellChecker .....	364
HunSpell .....	365
Polar SpellChecker ActiveX .....	366
Downloaders .....	367
CleverComponents .....	367
Indy .....	368
File readers and writers .....	368
RichViewXML .....	368
RvHtmlImporter (deprecated) .....	368
RvHtmlViewImporter (deprecated) .....	369
User interface .....	370
SpTBXLib .....	370
TNT Controls .....	370
TBX .....	371
Toolbar2000 .....	371
<b>Toolbar Images</b> .....	<b>371</b>
TRichView Icons .....	372
GlyFX toolbar images .....	374
Glyfz toolbar images .....	375
Fugue Icons toolbar images .....	375
FamFamFam Silk Icons toolbar images .....	376
<b>7..Procedures and Functions</b> .....	<b>376</b>
<b>RichViewActions unit</b> .....	<b>377</b>
GoToCheckpoint function .....	377
PasteHTML procedure (deprecated) .....	378
RVA_ChoseStyle function .....	378
RVA_EditorControlFunctionDef function .....	378
RVA_ConvertTo* procedures .....	378
RvHtmlHelp function .....	379

<b>RVARibbonUtils unit</b> .....	<b>379</b>
FillActionClientContents procedure.....	379
RemoveEllipsis, RemoveEllipsisFromRibbon procedures.....	379
LocalizeRibbonPage procedure.....	380
SetRulerColors procedure.....	380
<b>Other units</b> .....	<b>380</b>
LoadCSV function.....	380
MarkSubString functions.....	381
NormalizeRichView procedure.....	382
RVChangeCharCase, RVGetCharCase procedures .....	383
<b>8..Types</b> .....	<b>383</b>
TRulerUnits .....	384
TRVAEditEvent, TRVAEditEvent2 .....	384
TRVAHFInfo .....	384
TRVASpellString, TRVASpellStrings, TRVASpellStringList .....	385
TrvFileExportFilter, TrvFileSaveFilter .....	386
TRVFileFormatComponent .....	387
TrvFileImportFilter, TrvFileOpenFilter .....	387
TRVShowFormEvent .....	388
<b>9..Global variables</b> .....	<b>388</b>
MainRVAControlPanel .....	388
RVA_EditForceDefControl .....	389
RVA_EditorControlFunction .....	389
ShowUntranslatedControls .....	390
<b>10..How to</b> .....	<b>390</b>
How to change RVF and XML format name .....	390
 <b>Index</b>	 <b>391</b>

# 1 RichViewActions

RichViewActions is a set of actions and components for Delphi, C++Builder and Lazarus (for Windows and Linux) allowing you to create a user interface for the TRichView and ScaleRichView editors. Actions can be assigned to command buttons, menu items, toolbar buttons. You can use the standard Delphi/Lazarus components or your favorite third-party components if they support actions.

The actions require no programming, just add a new action in the action manager, assign it to a component, and it will do all the work automatically.

RichViewActions homepage: [www.trichview.com/resources/actions/](http://www.trichview.com/resources/actions/).

Minimal supported version of Delphi: 5

Minimal supported version of C++Builder: 6.

This help file includes the descriptions of all actions, main components, and utility functions. It does not describe visual controls installed with the package (the most of them are self-explanatory).

## See also:

- Overview <sup>(16)</sup>
- What's new? <sup>(21)</sup>

## 1.1 Overview

- Using RichViewActions <sup>(16)</sup>
- Style Templates <sup>(20)</sup>
- What's new <sup>(21)</sup>

### 1.1.1 Using RichViewActions

#### Overview

---

All actions have Control <sup>(312)</sup> property of TCustomRVControl type. If this property is not assigned, actions work with the focused control (or GetControlPanel <sup>(336)</sup>.DefaultControl <sup>(42)</sup> component).

All actions have Disabled <sup>(335)</sup> property (Boolean). If it is *True*, all actions are disabled. If it is *False* (default), actions update their Enabled property automatically.

TRVAControlPanel <sup>(38)</sup> component has a set of properties affecting all (or many) actions.

TRVAPopupMenu <sup>(66)</sup> is a popup menu for TCustomRichViewEdit that maintains items itself. Set its ActionList <sup>(69)</sup> property to the action list containing RichViewActions. Assign this component to the editor's PopupMenu. Do not add items in this menu manually – they all will be cleared (you can add them on each call of OnPopup, though). The menu supports live spelling, see below.

#### Preparing for the actions

---

Right-click RichViewEdit in Delphi/C++Builder/Lazarus, choose "Settings" in the context menu, select "Allow adding styles dynamically", check options for saving and loading background and layout.

Include *rvoCtrlJumps* in RichViewEdit.EditorOptions.

## How to use

Put an ActionList component on the form. Link it with some image list. Double click the ActionList. Click "New Standard Actions" on the component editor toolbar. Select the action and click OK. If the linked image-list is 16x16, an action image will be added automatically. However, we recommend to use high-quality images <sup>(372)</sup> instead of default images.

Alternatively, you can use existing datamodules:

- **dmActions.pas** contains an ActionList and an ImageList, with simple 16-color images;
- **dmActionImages1.pas** contains ImageLists with high-quality toolbar images, set #1, both normal and disabled versions (for Delphi/C++Builder 2009 and newer);
- **dmActionImages2.pas** the same, set #2;
- **dmactionimageslaz1.pas** the same, set #1 (for Lazarus)
- **dmactionimageslaz2.pas** the same, set #2 (for Lazarus)

For newer versions of Delphi and Lazarus, there are datamodules containing multi-resolution ImageLists:

- **dmActionsVirtualImageLists.pas** contains a virtual ImageList (for Delphi/C++Builder 10.3+), the actual images are in dmActionsImageCollection1.pas (set #1) and dmActionsImageCollection2.pas (set #2)
- **dmactionsimagesmultireslaz1.pas** the same, set #1 (for Lazarus 2+)
- **dmactionsimagesmultireslaz2.pas** the same, set #2 (for Lazarus 2+)

If you will not modify these units, you can include them in your project directly. But it's highly recommended to create a copy of them (under new names) and use the copies instead. Important: if you access actions in OnCreate event of some form, this form must be created after the datamodule (the form creation order may be specified in the project options).

## Localization

Several languages are available for translations of user interface. Some operations must be performed even if you use only one language. See Localization of RichViewActions <sup>(340)</sup>.

## Ruler

RichViewActions include TRuler component by Pieter Zijlstra and TRVRuler <sup>(80)</sup> component inherited from it.

Create a TRVRuler <sup>(80)</sup> component, place on the form above the editor, assign RVRuler.RichViewEdit <sup>(86)</sup> to this editor.

In TRVAControlPanel.OnMarginsChanged <sup>(65)</sup> call RVRuler.UpdateRulerMargins <sup>(87)</sup>.

The ruler supports units <sup>(85)</sup>: inches, centimeters, millimeters, picas, pixels and points.

If you want to remove support for tab stops, set ruler's MaxTabs property to 0.

## Actions requiring a special attention

It's recommended to assign Control <sup>(312)</sup> property for the following actions:

- TrvActionStyleInspector <sup>(216)</sup>;

- TrvActionEditNote<sup>(260)</sup>.

Otherwise, it will be assigned automatically when the action is executed for the first time.

Some additional code may be required to implement UI for a note editor of TrvActionEditNote<sup>(260)</sup>.

## Files

---

File-related actions (TrvActionNew<sup>(101)</sup>, TrvActionOpen<sup>(106)</sup>, TrvActionSave<sup>(122)</sup>, TrvActionSaveAs<sup>(127)</sup>) are linked together. The most of them cannot work without others. These actions keep track of all opened files and can be used even for multiple RichViewEdits (for example, in MDI application) without any additional effort from the programmer. These actions cannot be used for TDBRichViewEdit.

TrvActionExport<sup>(110)</sup>, TrvActionInsertFile<sup>(228)</sup> are independent – they are not linked with other actions, they just allow to export/insert a file.

## Printing

---

TRVAControlPanel<sup>(38)</sup> has RVPrint<sup>(53)</sup>: TRVPrint property. All printing actions (TrvActionPrintPreview<sup>(118)</sup>, TrvActionQuickPrint<sup>(120)</sup>, TrvActionPrint<sup>(116)</sup>) use it.

It is recommended to assign it to some TRVPrint component.

If this property is not assigned, these actions search TRVPrint component on the same form as the target editor. If it is not found, they create a temporal TRVPrint component.

Besides, these actions use another property of TRVAControlPanel<sup>(38)</sup> – ShowSoftPageBreaks<sup>(54)</sup>: Boolean. If True (default), soft page breaks are shown when possible (after print preview or printing – until the next change in the document).

TrvActionPageSetup<sup>(114)</sup> is a special printing action. It does not require RichViewEdit, but requires RVPrint<sup>(53)</sup> assigned to TRVAControlPanel<sup>(38)</sup>.

## Color

---

All the main coloring actions (TrvActionFontColor<sup>(148)</sup>, TrvActionFontBackColor<sup>(147)</sup>, TrvActionParaColor<sup>(195)</sup>, TrvActionColor<sup>(315)</sup>) can work in three modes, depending on UserInterface<sup>(320)</sup> property:

- *rvacAdvanced* (default) – on execution, a special non-modal color dialog pops up; it pops up at the position calculated by the coordinates of control that caused this action to execute (ActionComponent – button, for example) or by mouse coordinates;
- *rvacColorDialog* – on execution, a color is chosen with standard color dialog;
- *rvacNone* – no user interface, action just applies Color property.

This property gives you an ability to implement a color combo (you need two actions – one for applying the last chosen color (UserInterface<sup>(320)</sup> = *rvacNone*) and one for displaying a color dialog). Or you can implement your own interface for choosing colors.

All coloring actions have OnShowColorPicker<sup>(321)</sup> and OnHideColorPicker<sup>(321)</sup> events helping to implement a custom color combo.

## Formatting

TrvActionFontGrow<sup>(155)</sup> and TrvActionFontShrink<sup>(161)</sup> change the size of selected font by the given number of percents (Percent<sup>(164)</sup> property). If you change its value, do not forget to update hints for the actions.

TrvActionFontGrow<sup>(155)</sup>, TrvActionFontGrowOnePoint<sup>(157)</sup> have MaxSize property, TrvActionFontShrink<sup>(161)</sup>, TrvActionFontShrinkOnePoint<sup>(164)</sup> have MinSize property.

TrvActionParagraph<sup>(198)</sup> can work in two modes, depending on UserInterface<sup>(205)</sup>: Boolean property:

- *True* (default) – the action gets values of its properties from RichViewEdit, displays a dialog for modifying them, then apply them to the selection;
- *False* – the action applies its properties to the selection, without displaying a dialog.

TrvActionFontEx<sup>(149)</sup> is similar to TrvActionParagraph<sup>(198)</sup>, it can also work in two modes.

## Live spelling check

TRVAPopupMenu supports live spelling.

You have the following options:

- assign a spelling checker interface component<sup>(355)</sup> to TRVAControlPanel<sup>(38)</sup>.SpellInterface<sup>(54)</sup> property, and all commands (suggestions, "add to dictionary" and "ignore all", etc.) will be added in the menu automatically;
- use the menu events: OnLiveSpellGetSuggestions<sup>(70)</sup>, OnLiveSpellIgnoreAll<sup>(70)</sup>, OnLiveSpellAdd<sup>(69)</sup>; if your spellchecker can store a user choice to generate better suggestions in future, you can also process OnLiveSpellWordReplace<sup>(71)</sup>.

RichViewActions do not process OnSpellingCheck event, you still need to do it yourself, see the commented code in Unit3.pas.

## Bidirectional text

TrvActionParaLTR<sup>(208)</sup>, TrvActionParaRTL<sup>(210)</sup> – set default bidi mode for the selected paragraphs.

TrvActionTextLTR<sup>(171)</sup>, TrvActionTextRTL<sup>(172)</sup> – set bidi mode for the selected text.

Before using these actions set RichViewEdit.BiDiMode either to *rvbdLeftToRight* or *rvbdRightToLeft* (do not use for BiDiMode=*rvbdUnspecified*!)

## Providing help files

RichViewActions do not include help files for users. However, if you create help files for RichViewActions dialogs, RichViewActions can use them.

To enable using help files, assign RVAControlPanel.UseHelpFiles<sup>(56)</sup> = *True*.

Each UI language can have its own help file, you can specify in in RVA\_SetHelpFile<sup>(347)</sup> procedure.

If these help files are not defined, RichViewActions may use the application default help file (Application.HelpFile), if RVAControlPanel.UseDefaultHelpFile<sup>(56)</sup> = *True*.

You can use either HelpContext or HelpKeyword of RichViewActions forms, see RVAControlPanel.HelpType<sup>(47)</sup>.

The list of HelpContexts and HelpKeywords is in the topic about RVAControlPanel.HelpType<sup>(47)</sup>. The default HelpKeywords are in English. If you want to modify them, use RVSetHelpKeyword procedure.

### 1.1.2 Style templates

Several actions work differently depending on the value of UseStyleTemplates property of the target editor.

#### TrvActionNew<sup>(101)</sup>, TrvActionOpen<sup>(106)</sup>

---

If Editor.UseStyleTemplates=*True*, TrvActionNew resets the Editor.Style's TextStyles and ParaStyles according to StyleTemplates<sup>(104)</sup>, and clears its ListStyles.

If Editor.UseStyleTemplates=*False*, TrvActionNew does not reset styles; it may only delete unused styles (if GetControlPanel<sup>(336)</sup>.AutoDeleteUnusedStyles<sup>(42)</sup>=*True*). If you want to reset styles, use OnNew<sup>(101)</sup> event.

TrvActionOpen uses its linked<sup>(108)</sup> TrvActionNew to perform the same operations before loading.

#### TrvActionPasteSpecial<sup>(137)</sup>

---

If Editor.UseStyleTemplates=*True*, the action displays additional options for pasting RTF and RVF documents:

- "apply styles of the target document"
- "keep styles and appearance"
- "keep appearance, ignore styles".

The action temporarily changes Editor.StyleTemplateInsertMode according to the chosen option.

#### TrvActionInsertHyperlink<sup>(232)</sup>

---

If Editor.UseStyleTemplates=*True*, the action applies StyleTemplateName<sup>(239)</sup> to hyperlinks (or clears a style template when it converts hyperlinks to a plain text). The dialog window allows choosing a style template to apply to a hyperlink.

If Editor.UseStyleTemplates=*False*, the action applies its properties listed in ValidProperties<sup>(239)</sup> (or applies properties of Editor.Style.TextStyles[0] when it converts hyperlinks to a plain text). The dialog window allows choosing colors and effects to apply to a hyperlink.

TrvActionRemoveHyperlinks<sup>(329)</sup> is affected as well.

#### TrvActionClearFormat<sup>(219)</sup>, TrvActionClearTextFormat<sup>(219)</sup>

---

If Editor.UseStyleTemplates=*True*, the actions clear formatting according to style templates (applying "Normal" to paragraphs, clearing styles from normal text, applying "Hyperlink" to hyperlinks, resetting additional attributes).

If Editor.UseStyleTemplates=*False*, the actions apply Editor.Style.TextStyles[0] and Editor.Style.ParaStyles[0].



## TrvActionStyleTemplates<sup>(212)</sup>, TrvActionAddStyleTemplate<sup>(215)</sup>

These actions are enabled only if `Editor.UseStyleTemplates=True`.

## TrvActionInsertFootnote<sup>(264)</sup>, TrvActionInsertEndnote<sup>(263)</sup>, TrvActionInsertSidenote<sup>(267)</sup>

If `Editor.UseStyleTemplates=True`, `TrvActionInsertFootnote`<sup>(264)</sup> inserts footnote characters using "footnote reference" style template, and formats footnote text using "footnote text" style template. `TrvActionInsertEndnote`<sup>(263)</sup> uses "endnote reference" and "endnote text" style templates, `TrvActionInsertSidenote`<sup>(267)</sup> uses `RefStyleTemplateName`<sup>(269)</sup> ("Sidenote Reference" by default) and `StyleTemplateName`<sup>(260)</sup> ("Sidenote Text" by default) style templates.

"footnote reference", "footnote text", "endnote reference", "endnote text" are included in `TrvActionNew.StyleTemplates`<sup>(104)</sup> by default. "Sidenote Reference" and "Sidenote Text" are not included.

 **ScaleRichView:** `TsrvActionInsertFootnote`, `TsrvActionInsertEndnote`, `TsrvActionInsertSidenote` use the same style templates.

## TrvActionInsertCaption<sup>(223)</sup>

`TrvActionInsertCaption` uses "caption" style template. It is included in `TrvActionNew.StyleTemplates`<sup>(104)</sup> by default.

### 1.1.3 History

#### ▼ Changes after version 13 (after TRichView 23)

##### Compatibility issues

- Changes in packages for Lazarus: `rvrichviewactionslaz_dsgn.lpk` is removed. Install `rvrichviewactionslaz.lpk` instead (this package is "runtime + designtime")
- `TrvActionParaBullets` use Unicode characters in bullets in list templates (instead of characters of "Symbol" and "Wingdings" fonts)

##### Lazarus for Linux

`RichViewActions` can be used in Lazarus for Linux.

##### New component



`TRVFontListBox` (analog of `TRVFontComboBox`<sup>(73)</sup>). Internally, it is used in the font dialog<sup>(149)</sup> for Lazarus for Linux.

##### Changes

- dialogs are redesigned to look better both for Windows and Linux
- widths of all office radio buttons are calculated optimally (depending on the image and text widths), if they are arranged in a single row.

- HTML-style bullet and numbering dialog is obsolete and removed.
- TRVAControlPanel<sup>(38)</sup>.DialogFontName<sup>(45)</sup> now contains a list of font names in order of preference. This property is used only in the Windows version of RichViewActions. The default value is changed.
- new property TRVAControlPanel<sup>(38)</sup>.DialogFontNameLin<sup>(45)</sup> is used in the Linux version of RichViewActions.
- new value for TRVAControlPanel<sup>(38)</sup>.DialogPosition<sup>(46)</sup>: *rvafpEditorFormCenter*.

### ▼ Changes in version 13 (between TRichView 22 and 23)



New TRVSpellInterface<sup>(356)</sup> component allows using TRVSpellChecker (included in TRichView) Support of "Windows 64-bit (Modern)" platform in C++Builder 12+.

### ▼ Changes in version 12 (between TRichView 21 and 22)

**Delphi and C++Builder 12** Athens are supported.

TrvActionInsertPicture<sup>(249)</sup> allows inserting multiple images.

Support of decimal tab alignment (in TrvActionParagraph<sup>(198)</sup>, TRVRuler<sup>(80)</sup> and TRVRulerItemSelector<sup>(37)</sup>).

Option to turn on/off smooth image resizing in TrvActionItemProperties<sup>(324)</sup>.

New properties of TRVAControlPanel<sup>(38)</sup>:

- TRVAControlPanel<sup>(38)</sup>.DialogPosition<sup>(46)</sup> – position of dialog windows
- TRVAControlPanel<sup>(38)</sup>.DialogZoomPercent<sup>(46)</sup> allows scaling all dialog windows

New action: TrvActionUserDefinedColor<sup>(316)</sup>.

New events: TrvActionFontEx<sup>(149)</sup>.OnShowingDialog<sup>(155)</sup>, TrvActionParagraph<sup>(198)</sup>.OnShowingDialog<sup>(205)</sup>, TrvActionParaBorder<sup>(191)</sup>.OnShowingDialog<sup>(194)</sup>. They allow initializing dialogs with predefined values, instead of using attributes of the selected fragment.

New event: TrvActionSave<sup>(122)</sup>.OnSave<sup>(126)</sup>, it occurs on successful saving.

### ▼ Changes in version 11 (between TRichView 20 and 21)

#### Compatibility issues

ImagePrefix, FileTitle, SaveOptions properties of TrvActionSave<sup>(122)</sup> and TrvActionExport<sup>(110)</sup> are removed. Use HTMLSaveProperties and DocParameters.Title properties of the target editor.

TRVAControlPanel.HTMLComponent<sup>(51)</sup> is not removed but deprecated. It's highly recommended to remove rvHtmlImporter<sup>(368)</sup> and rvHtmlViewImporter<sup>(369)</sup> components from your application, because HTML loading methods of TRichView provide better results.

TRVAControlPanel.InitImportPictures, DoneImportPictures, DoImportPicture are removed. They are replaced by InitImportPicturesAndFiles and DoneImportPicturesAndFiles<sup>(59)</sup>.

---

## HTML

Since this version RichViewActions use TRichView methods for loading HTML, and new methods for saving HTML.

TRVAControlPanel.HTMLComponent<sup>(51)</sup> property still can be used, but it is deprecated, because TRichView methods provide much better results than imported components. We plan to remove this property in future updates of RichViewActions.

Some properties for HTML export are removed from actions (see the compatibility issues above).

A component assigned to TRVAControlPanel.DownloadInterface<sup>(46)</sup> can be used to download not only external pictures, but also external CSS files.

## Other changes

New properties controlling a state of the "Default" checkbox in property dialogs:

- TrvActionItemProperties.DefaultChecked, DefaultPersistent<sup>(327)</sup>
- TrvActionTableProperties.DefaultChecked, DefaultPersistent<sup>(303)</sup>

You can implement your own way for choosing pictures using the new TRVAControlPanel<sup>(38)</sup>.OnChoosePicture<sup>(61)</sup> event.

Table properties dialog allows choosing cell height mode: "exactly" or "at least".

---

## ▼ Changes in version 10 (between TRichView 19 and 20)

---

### RAD Studio 11 Alexandria

RichViewActions can be used in Delphi and C++Builder 11 Alexandria.

### Help files

RVA\_SetHelpFile<sup>(347)</sup> allows defining the help file that will be used in RichViewActions dialogs for the specified UI language.

TRVAControlPanel<sup>(38)</sup>.UseDefaultHelpFile<sup>(56)</sup> allows using Application.HelpFile.

TRVAControlPanel<sup>(38)</sup>.HelpType<sup>(47)</sup> allows choosing between using HelpContext and HelpKeyword properties.

RVSetHelpKeyword<sup>(352)</sup> allows changing HelpKeyword for the given HelpContext.

### Markdown

Markdown is supported not only as export format, but also as import format (TrvActionInsertFile<sup>(228)</sup>) and document format (TrvActionOpen<sup>(106)</sup> and TrvActionSaveAs<sup>(127)</sup>)

---

## ▼ Changes in version 9 (between TRichView 18 and 19)

---

### RAD Studio 10.4 Sydney

RichViewActions can be used in Delphi and C++Builder 10.4 Sydney. Per-control VCL styling is supported; in dialogs, previews use the style of the target editor.

## Localization

---

New UI translation: Slovenian

## Files: DocX and Markdown

---

TrvActionOpen<sup>(106)</sup> and TrvActionInsertFile<sup>(228)</sup> support DocX files.

TrvActionExport<sup>(110)</sup> support **Markdown** files.

## Find and replace

---

New properties:

- TrvActionFind<sup>(132)</sup>.FindText<sup>(134)</sup>
- TrvActionReplace<sup>(141)</sup>.FindText, ReplaceText<sup>(142)</sup>

The actions do not display "not found" dialog after the user answered "No" to "continue search from the beginning/end?" anymore.

## Other

---

Smooth scrolling to checkpoint (in TrvActionBookmarks<sup>(221)</sup> and TrvActionInsertHyperlink<sup>(232)</sup>.GoToLink<sup>(242)</sup>).

## ▼ Changes in version 8 (between TRichView 17 and 18)

---

### Compatibility issues:

---

- RVA\_ConvertToPixels and RVA\_ConvertToTwips<sup>(378)</sup> procedures have a new ARVStyle parameter.
- The following properties of TrvActionInsertSymbol<sup>(254)</sup> are removed: AlwaysInsertUnicode, DisplayUnicodeBlocks, SymbolType
- String parameters are changed to TRVUnicodeString in the events:
  - TRVAControlPanel.OnCustomFileOperation<sup>(62)</sup>, OnDownload<sup>(64)</sup>, OnGetHeaderFooterCode<sup>(65)</sup>.
  - TRVAPopupMenu.OnLiveSpellAdd<sup>(69)</sup>, OnLiveSpellGetSuggestions<sup>(70)</sup>, OnLiveSpellIgnoreAll<sup>(70)</sup>, OnLiveSpellWordReplace<sup>(71)</sup>
  - TrvActionOpen.OnOpenFile<sup>(110)</sup>
  - TrvActionSave.OnSaving<sup>(126)</sup>, OnDocumentFileChange<sup>(126)</sup>
  - TrvActionReplace.OnReplacing<sup>(144)</sup>
  - TrvActionInsertHyperlink.OnApplyHyperlinkToItem<sup>(244)</sup>, OnGetHyperlinkTargetFromItem<sup>(244)</sup>, OnHyperlinkForm<sup>(244)</sup>
  - TrvActionInsertPicture.OnInserting<sup>(254)</sup>
  - TrvActionInsertText.OnInsertText<sup>(257)</sup>
- Support for KSDev ThemeEngine is discontinued
- RVAFormat function is moved from RichViewActions to RVAFuncs unit (you can simply use Format instead).

## RAD Studio 10.3 Rio

---

RichViewActions can be used in Delphi and C++Builder 10.3 Rio.

---

## Lazarus

RichViewActions can be used in Lazarus (for Windows 32-bit and 64-bit).

---

### High-DPI display modes

All controls and dialogs in RichViewActions support high-dpi display modes.

"Per monitor" and "per monitor v2" modes are supported, if they are supported by the application (Delphi 10.1+ is required for "per monitor", Delphi 10.3+ for "per monitor v2").

---

### Virtual image lists

New data modules are added for RAD Studio 10.3. They contain 16x16, 32x32, and selected 64x64 toolbar images in TImageCollection and TVirtualImageList components. Pascal and C++ versions of these data modules are available. Details are explained in the topic about TRichView icons <sup>(372)</sup>.

New demo projects are included:

- DelphiUnicode\ActionTest\_MultiRes\
- CBuilderUnicode\ActionTest\_MultiRes\

These projects use virtual image lists and support "per monitor v2".

---

### Changes in "Insert Symbol" dialog

Since this version "Insert Symbol" allows inserting only Unicode characters. Now it supports all UTF-32 characters.

- TrvActionInsertSymbol <sup>(254)</sup> has new properties: CharCode <sup>(255)</sup>, FontName <sup>(256)</sup>, Protection <sup>(256)</sup>.

---

### Other changes

- new TrvActionEditNote <sup>(260)</sup>.OnStartEditNote <sup>(263)</sup> event
- new TrvActionInsertHyperlink <sup>(232)</sup>.ScrollToCenter <sup>(238)</sup> property
- new TRVFontComboBox <sup>(73)</sup>.AutoCharset <sup>(75)</sup> property
- ruler is updated to support high DPI screen modes

---

## ▼ Changes in version 7 (between TRichView 16 and 17)

---

### Compatibility issues:

- Ruler.pas is renamed to RVRulerBase.pas
- Color property is removed from Rules.Tabs[]
- new parameters in functions from MarkSearch <sup>(381)</sup> unit
- No more compiler \$defines for using Addict <sup>(362)</sup>, you cannot add RVAddictSpell3 and RVAddictThesaurus3 properties to TRVAControlPanel <sup>(38)</sup>. Use SpellInterface <sup>(54)</sup> property instead. TrvActionAddictSpell3, TrvActionAddictThesaurus3 are removed, they are superseded by TrvActionSpellingCheck <sup>(310)</sup> and TrvActionThesaurus <sup>(310)</sup>. RVA\_Addict3AutoCorrect function is removed, use TRVAAddictSpellInterface <sup>(356)</sup>'s auto-correct methods

- No more compiler \$defines for using TIdHttp (Indy<sup>368</sup>) and TClHttp (CleverComponents<sup>367</sup>), you cannot add IdHttp and ClHttp properties to TRVAControlPanel<sup>38</sup>. Use DownloadInterface<sup>46</sup> property instead.
- No more compiler \$defines for using RichViewXML<sup>368</sup>, RvHtmlImporter<sup>368</sup> and RvHtmlViewImporter<sup>369</sup>, you cannot add RVXML, RVHTMLImporter, RVHTMLViewImporter properties to TRVAControlPanel. Use XMLComponent<sup>58</sup> and HTMLComponent<sup>51</sup> properties instead.
- The compiler \$define for using TNT Controls<sup>370</sup> is moved from RichViewActions.inc to RV\_Defs.inc.
- TrvActionShowSpecialCharacters<sup>331</sup> shows/hides checkpoints by default.
- The following properties are removed: TrvActionPasteSpecial<sup>137</sup>.StoreFileName, TrvActionInsertPicture<sup>249</sup>.StoreFileName, TrvActionTableProperties<sup>301</sup>.StoreImageFileName, TrvActionItemProperties<sup>324</sup>.StoreImageFileName. Instead of these properties, the actions check *rvoAssignImageFileNames* in the Options property of the target editor.

### New "interface" components

Interface components<sup>352</sup> provide an intermediate layer between RichViewActions and third-party components, so RichViewActions can use third-party components without compiler \$defines.


New properties are added to TRVAControlPanel<sup>38</sup>: ColorDialogInterface<sup>42</sup>, DownloadInterface<sup>46</sup>, SpellInterface<sup>54</sup>.

Interface components for using TdxColorDialog, Indy<sup>368</sup>, CleverComponents<sup>367</sup>, Addict<sup>362</sup>, ASpell<sup>363</sup>, ExpressSpellChecker<sup>364</sup>, HunSpell<sup>365</sup>, Polar SpellChecker<sup>366</sup> are added.

### File loading and saving components

Previously, programmers needed to add \$defines in RichViewActions.inc to allow using RichViewXML<sup>368</sup>, RvHtmlImporter<sup>368</sup> and RvHtmlViewImporter<sup>369</sup> in RichViewActions. Now, you can use XMLComponent<sup>58</sup> and HTMLComponent<sup>51</sup> properties of TRVAControlPanel<sup>38</sup>.

### Checkpoints (Bookmarks)

New action  TrvActionBookmarks<sup>221</sup> allows adding and managing checkpoints.

In TrvActionInsertHyperlink<sup>232</sup>, the editor of the link target is now a combo-box containing a list of checkpoint names.

TrvActionShowSpecialCharacters<sup>331</sup> can show/hide checkpoints.

### Mathematical formulas (equations)

New action:  TrvActionInsertEquation<sup>226</sup>. TrvActionItemProperties<sup>324</sup> can edit properties of equation items.

### Font preview

TRVFontComboBox<sup>73</sup> can display preview of fonts. It has new properties: Preview<sup>76</sup>, DropDownWidth<sup>75</sup>, SymbolPreviewString<sup>76</sup>.

The dialog of TrvActionFontEx<sup>149</sup> can display a preview of font names in the list as well. It has a new property: PreviewInList<sup>153</sup>.

---

## Changes related to tables

- TrvActionInsertTable<sup>(271)</sup> has new properties to assign to table (HeadingRowColor and other colors<sup>(281)</sup>, ColBandSize, RowBandSize<sup>(277)</sup>) and to table rows (RowsVAlign<sup>(280)</sup>, RowsKeepTogether<sup>(279)</sup>).
- The table properties dialog (displayed by TrvActionItemProperties<sup>(324)</sup> and TrvActionTableProperties<sup>(301)</sup>) allows defining colors of rows and columns.
- new properties TrvActionItemProperties.UpdateAllInsertTableActions<sup>(326)</sup> and TrvActionTableProperties.UpdateAllInsertTableActions<sup>(303)</sup> allows applying default properties to all "insert table" actions on the same form/datamodule.
- The "Default" check box in the table properties affects not only the page "Table", but also pages "Rows" and "Columns".

---

## Changes related to printing

- TrvActionInsertPageBreak<sup>(248)</sup> and TrvActionRemovePageBreak<sup>(330)</sup>, when called from a table cell, add and remove page breaks before the current table row.
- PageNumberType<sup>(226)</sup> property is added to TrvActionInsertPageCount<sup>(248)</sup>. For inserted "page count" fields, a number type can be changed by TrvActionItemProperties<sup>(324)</sup>.
- New TRVAControlPanel<sup>(38)</sup>.OnGetHeaderFooterCode<sup>(65)</sup> event.

---

## Other changes

- OnInserting<sup>(254)</sup> event is added to TrvActionInsertPicture<sup>(249)</sup>.
- new property TRuler.TickColor for drawing ticks and tab stops.
- New untranslated features can be hidden using ShowUntranslatedControls<sup>(390)</sup> variable.
- MarkSearch<sup>(381)</sup> functions are improved
- Packages were separated into runtime and designtime packages. RVARibbonUtils.pas (unit providing TRibbon support) is moved to a separate runtime package.

---

## ▼ Changes in version 6 (between TRichView 15 and 16)

---

### Compatibility issues:

TrvActionItemProperties<sup>(324)</sup> and TrvActionTableProperties<sup>(301)</sup> update linked TrvActionInsertTable<sup>(271)</sup> and TrvActionInsertHLine<sup>(231)</sup> actions when the user checks "Default" checkbox. Previously, they were always updated when assigned explicitly.

---

### Installation and directory structure


Starting from this update, RichViewActions are installed automatically in Delphi and C++Builder IDE together with TRichView.


The new installer installs the components in Delphi and C++Builder, both for 32-bit and 64-bit platforms (if available). The installer adds all necessary paths to RAD Studio library.

Source code is moved to "Source" folder, inc-files are moved to "Source\Include" folder, demo projects are moved to "Demos" folder.

This help file is integrated in RAD Studio IDE (for XE8+)

### New actions

 TrvActionAlignDistribute<sup>(176)</sup> aligns the selected paragraph to the both left and right sides by adding space between all characters.

 TrvActionInsertPageCount<sup>(248)</sup> inserts a page count field.

### Changes related to paragraph alignments

TrvActionAlignJustify<sup>(177)</sup> (as well as TrvActionAlignDistribute<sup>(176)</sup>) has new properties: LastLineAlignment and UseLastLineAlignment<sup>(176)</sup>.

TrvActionParagraph<sup>(198)</sup> has a new property: LastLineAlignment<sup>(202)</sup>.

### New properties:

- TRVAControlPanel<sup>(38)</sup>.MetafileCompatibility<sup>(51)</sup> affects printing of TSRichViewEdit components;
- TrvActionSave<sup>(122)</sup>.DisableWhenUnmodified<sup>(124)</sup> allows to disable the action for unmodified documents.

TrvActionRemovePageBreak<sup>(330)</sup> removes a page break from the current paragraph (not the current item as before).


TrvActionItemProperties<sup>(324)</sup> and TrvActionTableProperties<sup>(301)</sup> update linked TrvActionInsertTable<sup>(271)</sup> and TrvActionInsertHLine<sup>(231)</sup> actions when the user checks "Default" checkbox.


## ▼ Changes in version 5 (between TRichView 14 and 15)


### Compatibility issues:


- TrvActionFontEx<sup>(149)</sup>.Font<sup>(161)</sup>.Size is no longer used. Existing code must be changed to use TrvActionFontEx<sup>(149)</sup>.FontSizeDouble<sup>(153)</sup> instead.
- 'Francais' cannot be used as a language name; use 'French' or 'Français'.


### New actions


 TrvActionInsertTextBox<sup>(269)</sup> inserts a floating box


 TrvActionInsertFootnote<sup>(264)</sup> inserts a footnote


 TrvActionInsertEndnote<sup>(263)</sup> inserts an endnote

 TrvActionInsertSidenote<sup>(267)</sup> inserts a note in a floating box

 TrvActionEditNote<sup>(260)</sup> displays a window for editing a note or a floating box

 TrvActionInsertNumber<sup>(245)</sup> inserts a "numbered sequence" item

 TrvActionInsertCaption<sup>(223)</sup> inserts a caption for an image or a table

 TrvActionInsertPageNumber<sup>(249)</sup> inserts a "page number" field

### Changes

TrvActionExport<sup>(110)</sup> can export DocX files (without office converters). Make sure that *ffeDocX* is included in Filter<sup>(113)</sup>.

Changes made by TrvActionColor<sup>(315)</sup>, TrvActionBackground<sup>(313)</sup> can be undone.



New property: TrvActionFontEx<sup>(149)</sup>.FontSizeDouble<sup>(153)</sup>.

TRVFontSizeComboBox<sup>(79)</sup> supports fractional font sizes.

Changes in TrvActionItemProperties<sup>(324)</sup>:

- properties for sidenotes and text boxes
- properties for "numbered sequence" items
- properties for "page number" item
- redesigned picture properties pages
- redesigned table properties pages
- changes can be applied to ActionInsertTable<sup>(326)</sup> and ActionInsertHLine<sup>(326)</sup> (in addition to the current item)

Changes in TrvActionNew<sup>(103)</sup>:

- Reset<sup>(105)</sup> is made public.
- new item is included in StyleTemplates<sup>(104)</sup>: 'caption'.

New TrvActionInsertPicture<sup>(249)</sup> properties to assign to inserted pictures: BackgroundColor<sup>(251)</sup>, BorderWidth, BorderColor<sup>(251)</sup>, OuterHSpacing, OuterVSpacing<sup>(253)</sup>.

New TrvActionInsertTable<sup>(271)</sup> properties to assign to inserted tables: CellHPadding and CellVPadding<sup>(276)</sup> (they replace CellPadding), VisibleBorders<sup>(280)</sup>, BackgroundStyle<sup>(273)</sup>, BackgroundPicture<sup>(273)</sup>.

TRVAControlPanel<sup>(38)</sup>.DialogFontName<sup>(45)</sup>'s default value is changed to 'Tahoma'.

New RemoveEllipsisFromRibbon<sup>(379)</sup> procedure replaces RemoveEllipsis.

Languages have two names specified: a name in English and a native name.

## ▼ Changes added between TRichView v13 and v14

### Compatibility issues:






TRVAControlPanel<sup>(38)</sup>.TableGridStyle property is removed.

### Redesigned control panel



TRVAControlPanel<sup>(38)</sup> is redesigned. It does not assign global variables any more. You can have more than one control panel in a single application. New properties of TRVAControlPanel: Header, Footer<sup>(47)</sup>.

### New actions

#### Styles:<sup>(20)</sup>

-  TrvActionStyleTemplates<sup>(212)</sup>
-  TrvActionAddStyleTemplate<sup>(215)</sup>
-  TrvActionClearFormat<sup>(219)</sup>
-  TrvActionClearTextFormat<sup>(219)</sup>
-  TrvActionStyleInspector<sup>(216)</sup>

#### Cell rotation:

-  TrvActionTableCellRotationNone<sup>(288)</sup>
-  TrvActionTableCellRotation90<sup>(289)</sup>

 TrvActionTableCellRotation180 <sup>(289)</sup>

 TrvActionTableCellRotation270 <sup>(290)</sup>

### Table operations:

 TrvActionTableSplit <sup>(308)</sup>

 TrvActionTableSort <sup>(307)</sup>

 TrvActionTableToText <sup>(309)</sup>


### New and improved components

New components for applying style templates <sup>(20)</sup>:

 TRVStyleTemplateComboBox <sup>(87)</sup>

 TRVStyleTemplateListBox <sup>(87)</sup>

Improved components (can be linked to an editor, do not require additional code any more):

 TRVFontComboBox <sup>(73)</sup>

 TRVFontSizeComboBox <sup>(79)</sup>

 TRVFontCharsetComboBox <sup>(71)</sup>

### ScaleRichView support

You can assign not only TRichViewEdit, but also TSRichViewEdit to TRVAControlPanel.DefaultControl <sup>(43)</sup>, or Control <sup>(312)</sup> properties of actions.

Simplification: since this version, when using ScaleRichView, you do not need to assign RVA\_GetRichViewEditFromPopupComponent and RVA\_GetRichViewEdit variables, you do not need to call SRichViewEdit.SetRVMargins in TRVAControlPanel.OnMarginsChanged <sup>(65)</sup>.

### Visual changes

Delphi XE2+'s visual styles are supported in components and dialogs. The function RVA\_ChooseStyle <sup>(378)</sup> allows choosing and applying one of available styles.

Inverted (a light text on a dark background) color schemes are supported.

### Text files

The file actions allow choosing a code page for text files (including UTF-8). This feature can be disabled by assigning TRVAControlPanel.UseTextCodePageDialog <sup>(57)</sup> = False.

### New features, properties and events

A new event TRVAControlPanel.OnGetActionControlCoords <sup>(64)</sup> allows to specify coordinates for color-picker windows when actions are linked to non-visual components (for example, when using ExpressBars by Developer Express).

TrvActionNew <sup>(103)</sup> resets styles according to StyleTemplates <sup>(104)</sup>, if style templates are used.

TrvActionOpen <sup>(106)</sup> can use a linked <sup>(108)</sup> TrvActionNew to reset document before loading.

TrvActionPasteSpecial <sup>(137)</sup>:

- allows pasting URLs;
- respects RichViewEdit.AcceptPasteFormats property;
- if style templates <sup>(20)</sup> are used, displays style modes for pasting RTF and RVF.

TrvActionFontEx <sup>(149)</sup> can change a text background color (new BackColor <sup>(152)</sup> property).

TrvActionInsertHyperlink<sup>(232)</sup> can apply HoverUnderlineColor and HoverEffects properties (if style templates are not used). If style templates are used, it applies the chosen StyleTemplateName<sup>(239)</sup>.

TrvActionTableGrid<sup>(296)</sup> shows/hides grid only in the target editor.

In the print preview form (TrvActionPrintPreview<sup>(118)</sup>), a mouse wheel changes pages, **Ctrl** + mouse wheel zooms in/out

New optional parameters in NormalizeRichView<sup>(382)</sup>.

### Localization

New localization functions: RVA\_GetProgressMessage, RVA\_GetPrintingMessage<sup>(343)</sup>.

New languages:

- Portuguese (European)
- Catalan
- Hindi (in Delphi 4-2007, it is available only if TNT Controls<sup>(370)</sup> are used);
- Thai

## ▼ Changes added between TRichView v12 and v13

### New units of measurement

All dialogs display values measured according to new TRVAControlPanel<sup>(38)</sup>.UnitsDisplay<sup>(55)</sup> and TRVAControlPanel.PixelBorders<sup>(52)</sup> properties.

Properties of several actions are measured in TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

### New images

A new set of toolbar images<sup>(372)</sup> is available, created specially for RichViewActions. This set of images is used in this manual.

New component images are used in the Delphi/C++Builder's Component Palette for the components included in RichViewActions.

### New properties of TRVAControlPanel

TRVAControlPanel.DefaultDocParameters<sup>(43)</sup> define a page layout for new documents (especially useful for ScaleRichView)


New TRVAControlPanel<sup>(38)</sup>.DialogFontSize<sup>(45)</sup> property.

### Changes in menus

New TRVAPopupActionBar<sup>(66)</sup> component (for Delphi 2006 or newer).

The type of ActionList<sup>(69)</sup> property for menus<sup>(66)</sup> was changed from TActionList to TCustomActionList, to allow using TActionManager.

### Changes in actions

New action:  TrvActionHide<sup>(324)</sup>. TrvActionShowSpecialCharacters<sup>(331)</sup> shows/hides a hidden text as well as special characters.

New properties:

- TrvActionFontEx<sup>(149)</sup>.AutoApplySymbolCharset<sup>(151)</sup>
- TrvActionInsertHLine<sup>(231)</sup>.Style<sup>(232)</sup> defines the style for inserted horizontal lines
- TrvActionBackground<sup>(313)</sup>.CanChangeMargins<sup>(314)</sup> allows/disallows changing margins
- TrvActionParaBullets<sup>(195)</sup>'s and TrvActionParaNumbering<sup>(209)</sup>'s IndentStep<sup>(184)</sup> defines default indents for list levels.

TrvActionParaList<sup>(206)</sup>.IndentStep<sup>(207)</sup> now affects indents of list style templates in non-HTML dialog too.








### Other changes

New RVARibbonUtils<sup>(379)</sup> unit.

Ability to use CleverComponents<sup>(367)</sup> for downloading images.

## ▼ Changes added between TRichView v11 and v12

### New actions:

-  TrvActionPasteAsText<sup>(136)</sup>
-  TrvActionVAlign<sup>(333)</sup>
-  TrvActionClearLeft<sup>(180)</sup>
-  TrvActionClearRight<sup>(181)</sup>
-  TrvActionClearBoth<sup>(180)</sup>
-  TrvActionClearNone<sup>(181)</sup>
-  TrvActionRemoveHyperlinks<sup>(329)</sup>

### New methods, properties and events

- TrvActionInsertFile.InsertFile<sup>(230)</sup> method
- TrvActionInsertPicture.Spacing<sup>(253)</sup> property
- TrvActionParagraph.OutlineLevel<sup>(203)</sup> property (paragraph dialog has a new combobox for specifying a paragraph outline level)
- TrvActionItemProperties.OnCanApply<sup>(328)</sup> event

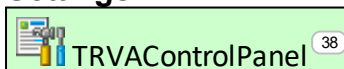
### New features:

- New types of vertical alignment for pictures in TrvActionItemProperties<sup>(324)</sup>.
- Properties dialog for breaks (TrvActionItemProperties<sup>(324)</sup>) has a new combobox for specifying a break style (line/rectangle/3d/dotted/dashed).
- Pasting HTML from the Clipboard using rvHtmlViewImporter<sup>(369)</sup>.
- New datamodules with alternative toolbar images (GlyFX<sup>(374)</sup> and Fugue Icons<sup>(375)</sup>)

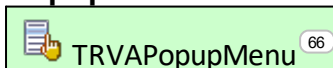
## 1.2 Components

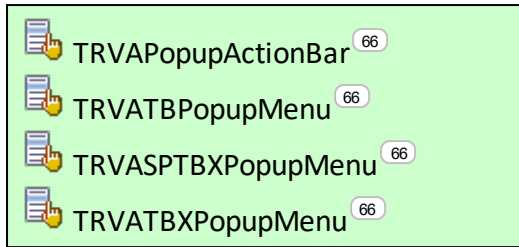
### Components

#### ▣ Settings

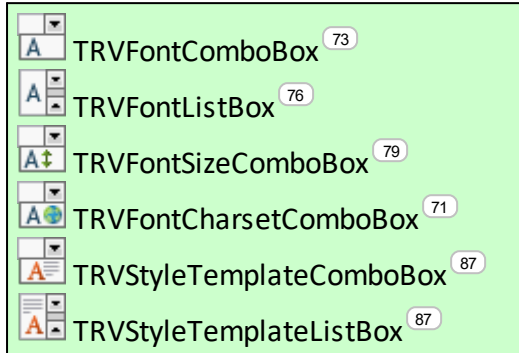


#### ▣ Popup Menus





## Combo- and List-Boxes



### 1.2.1 TCustomRVFontComboBox

TCustomRVFontComboBox is the base class for font combo-boxes.

**Unit** RVFontCombos;

**Syntax, if TNT Controls<sup>(370)</sup> are not used:**

```
TCustomRVFontComboBox = class(TComboBox)
```

**Syntax, if TNT Controls<sup>(370)</sup> are used:**

```
TCustomRVFontComboBox = class(TTntComboBox)
```

#### Hierarchy

(Hierarchy of TCustomRVFontComboBox, if TNT Controls<sup>(370)</sup> are not used)

```

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomComboBox
TComboBox

```

#### Description

This component is not used directly.

The following components are inherited from TCustomRVFontComboBox:

- TRVFontComboBox<sup>(73)</sup>
- TRVFontSizeComboBox<sup>(79)</sup>
- TRVFontCharsetComboBox<sup>(71)</sup>

### 1.2.1.1 Properties

#### In TCustomRVFontComboBox

---

- ActionFont<sup>(34)</sup>
- Editor<sup>(35)</sup>

#### Derived from TComboBox

---

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

#### 1.2.1.1.1 TCustomRVFontComboBox.ActionFont

Links this component to TrvActionFontEx<sup>(149)</sup> action.

```
property ActionFont: TrvActionFontEx(149);
```

This action is required to apply the selection to Editor<sup>(35)</sup>.

### 1.2.1.1.2 TCustomRVFontComboBox.Editor

Links this combo-box with an editor.

**property** Editor: TCustomRVControl;

Controls of the following types can be assigned to this property:

- TRichViewEdit
- TDBRichViewEdit
- TSRichViewEdit (ScaleRichView)
- TDBSRichViewEdit (ScaleRichView)

When the combo-box is linked with **Editor**:

- the combo-box is updated automatically according to changes in **Editor**
- the combo-box applies the user choice to the selected fragment in **Editor** (ActionFont<sup>34</sup> must be defined as well)

## 1.2.2 TCustomRVFontListBox

TCustomRVFontListBox is the base class for font list-boxes.

**Unit** RVFontCombos;

**Syntax, if TNT Controls<sup>370</sup> are not used:**

```
TCustomRVFontListBox = class(TListBox)
```

**Syntax, if TNT Controls<sup>370</sup> are used:**

```
TCustomRVFontListBox = class(TTntListBox)
```

### Hierarchy

(Hierarchy of TCustomRVFontListBox, if TNT Controls<sup>370</sup> are not used)

```

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomListBox
TListBox

```

### Description

This component is not used directly.

The following components are inherited from TCustomRVFontListBox:

- TRVFontListBox<sup>76</sup>

### 1.2.2.1 Properties

#### In TCustomRVFontComboBox

- ActionFont<sup>36</sup>
- AllowFocusEditor<sup>36</sup>
- Editor<sup>37</sup>

## Derived from TComboBox

---

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

### 1.2.2.1.1 TCustomRVFontListBox.ActionFont

Links this component to TrvActionFontEx<sup>149</sup> action.

**property** ActionFont: TrvActionFontEx<sup>149</sup>;

This action is required to apply the selection to Editor<sup>37</sup>.

### 1.2.2.1.2 TCustomRVFontListBox.AllowFocusEditor

Allows/disallows moving the input focus to Editor<sup>37</sup> when this list-box is clicked.

**property** AllowFocusEditor: Boolean;

#### Default value

*False*



### 1.2.2.1.3 TCustomRVFontListBox.Editor

Links this list-box with an editor.

**property** Editor: TCustomRVControl;

Controls of the following types can be assigned to this property:

- TRichViewEdit
- TDBRichViewEdit
- TSRichViewEdit (ScaleRichView)
- TDBSRichViewEdit (ScaleRichView)

When the list-box is linked with **Editor**:

- the list-box is updated automatically according to changes in **Editor**
- the list-box applies the user choice to the selected fragment in **Editor** (ActionFont<sup>36</sup> must be defined as well)

## 1.2.3 TRulerItemSelector

**TRVRulerItemSelector** defines the current tabulation type in TRVRuler<sup>80</sup> component.

**Unit** RVRuler;

TRVRulerItemSelector = **class** (TRulerItemSelector)

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TControl*  
*TWinControl*  
*TCustomControl*  
*TCustomPanel*  
*TCustomRulerItemSelector*  
*TRulerItemSelector*

### Description

This component allows switching a type of current tab stops in Ruler<sup>38</sup>: left-, right-, center-, decimal-aligned tab stops.

Tab stops of this type are added when the user clicks on the ruler.

### Acknowledgement

Thanks to Pieter Zijlstra who created this component.

## 1.2.3.1 Properties

### Derived from TCustomRulerSelector

- AvailableItems
- BevelHighLightColor
- BevelShadowColor

## ■ Ruler <sup>38</sup>

### Derived from TCustomPanel

- Align
- Anchors
- BevelEdges
- BevelInner
- BevelKind
- BevelOuter
- BevelWidth
- BorderWidth
- BorderStyle
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- Enabled
- FullRepaint
- Locked
- Padding
- ParentBackground
- ParentColor
- ParentCtl3D
- ParentShowHint
- PopupMenu
- ShowHint
- TabOrder
- TabStop
- Visible

#### 1.2.3.1.1 TCustomRulerItemSelector.Ruler

Specifies the ruler.

```
property Ruler: TCustomRuler 80;
```

This component changes the default type of tab stop for the ruler specified in this property.

## 1.2.4 TRVAControlPanel

TRVAControlPanel is the component allowing to change settings for RichViewActions

**Unit** RichViewActions;

### Syntax

```
TRVAControlPanel = class(TComponent)
```

### Hierarchy

*TObject*

*TPersistent*  
*TComponent*

## Description

This component contains properties affecting many RichViewActions.

## Main control panel

If the application does not create TRVAControlPanel explicitly, a default control panel is used (accessible as MainRVAControlPanel<sup>(388)</sup>). If the application contains a single TRVControlPanel, it becomes the main control panel automatically (assigned to MainRVAControlPanel). Another control panel can be made the main one if you call its Activate<sup>(59)</sup> method.

## Linking

All actions and many components in RichViewActions have links to a control panel, including:

- TrvCustomAction<sup>(333)</sup>.ControlPanel<sup>(335)</sup>
- TRVStyleTemplateCombo/ListBox<sup>(87)</sup>.ControlPanel<sup>(89)</sup>

If this link is not assigned (*nil*), the action/component uses the main control panel (see above).

## Properties

The component has the following group of properties:

- links to components providing additional functionality:
  - ColorDialog<sup>(42)</sup> and ColorDialogInterface<sup>(42)</sup>
  - DownloadInterface<sup>(46)</sup>
  - SpellInterface<sup>(54)</sup>
  - RVPrint<sup>(53)</sup>
  - XMLComponent<sup>(58)</sup>
- properties for customizing user interface:
  - Language<sup>(51)</sup>
  - AddColorNameToHints<sup>(41)</sup>
  - UserInterface<sup>(56)</sup>
  - UseXPThemes<sup>(57)</sup>
- properties for dialog windows:
  - DialogFontName<sup>(45)</sup> and DialogFontNameLin<sup>(45)</sup>
  - DialogFontSize<sup>(45)</sup>
  - DialogPosition<sup>(46)</sup>
  - DialogZoomPercent<sup>(46)</sup>
- properties related to end-user help files
  - UseHelpFiles<sup>(56)</sup>
  - UseDefaultHelpFile<sup>(56)</sup>
  - HelpType<sup>(47)</sup>
- properties of new documents:
  - DefaultColor<sup>(42)</sup>
  - DefaultCustomFilterIndex<sup>(43)</sup>
  - DefaultExt<sup>(44)</sup>

- DefaultFileFormat<sup>44</sup>
- DefaultFileName<sup>44</sup>
- DefaultMargin<sup>45</sup>
- DefaultDocParameters<sup>43</sup>
- properties controlling all actions:
  - DefaultControl<sup>43</sup>
  - ActionsEnabled<sup>41</sup>
- properties for changing names of file formats:
  - RVFFilter<sup>52</sup>
  - RVFLocalizable<sup>52</sup>
  - RVFormatTitle<sup>53</sup>
  - RVStylesFilter<sup>53</sup>
  - RVStylesExt<sup>53</sup>
  - XMLFilter<sup>58</sup>
  - XMLLocalizable<sup>58</sup>
- other properties:
  - AutoDeleteUnusedStyles<sup>42</sup>
  - FirstPageNumber<sup>47</sup>
  - SearchScope<sup>53</sup>
  - ShowSoftPageBreaks<sup>54</sup>
  - UseTextCodePageDialog<sup>57</sup>
  - MetafileCompatibility<sup>51</sup>

## Events ---

If you use TRVRuler<sup>80</sup>, process OnMarginsChanged<sup>65</sup>.

If you want to implement opening, saving, exporting and inserting files in additional formats, use OnCustomFileOperation<sup>62</sup>.

If you want to implement your own dialog for opening pictures, use OnChoosePicture<sup>61</sup> event.

### 1.2.4.1 Properties

#### In TRVAControlPanel ---

- ActionsEnabled<sup>41</sup>
- AddColorNameToHints<sup>41</sup>
- AutoDeleteUnusedStyles<sup>42</sup>
- ColorDialog<sup>42</sup>
- ColorDialogInterface<sup>42</sup>
- DefaultColor<sup>42</sup>
- DefaultControl<sup>43</sup>
- DefaultCustomFilterIndex<sup>43</sup>
- DefaultDocParameters<sup>43</sup>
- DefaultExt<sup>44</sup>
- DefaultFileFormat<sup>44</sup>
- DefaultFileName<sup>44</sup>
- DefaultMargin<sup>45</sup>
- DialogFontName<sup>45</sup>

- DialogFontNameLin <sup>(45)</sup>
- DialogFontSize <sup>(45)</sup>
- DialogPosition <sup>(46)</sup>
- DialogZoomPercent <sup>(46)</sup>
- DownloadInterface <sup>(46)</sup>
- FirstPageNumber <sup>(47)</sup>
- HelpType <sup>(47)</sup> (D6+)
- Language <sup>(51)</sup>
- MetafileCompatibility <sup>(51)</sup>
- PixelBorders <sup>(52)</sup>
- RVFFilter <sup>(52)</sup>
- RVFLocalizable <sup>(52)</sup>
- RVFormatTitle <sup>(53)</sup>
- HTMLComponent <sup>(51)</sup> (deprecated)
- RVPrint <sup>(53)</sup>
- RVStylesExt <sup>(53)</sup>
- RVStylesFilter <sup>(53)</sup>
- SearchScope <sup>(53)</sup>
- ShowSoftPageBreaks <sup>(54)</sup>
- SpellInterface <sup>(54)</sup>
- UnitsDisplay <sup>(55)</sup>
- UnitsProgram <sup>(55)</sup>
- UseDefaultHelpFile <sup>(56)</sup>
- UseHelpFiles <sup>(56)</sup>
- UserInterface <sup>(56)</sup>
- UseTextCodePageDialog <sup>(57)</sup>
- UseXPThemes <sup>(57)</sup>
- XMLComponent <sup>(58)</sup>
- XMLFilter <sup>(58)</sup>
- XMLLocalizable <sup>(58)</sup>

#### 1.2.4.1.1 TRVAControlPanel.ActionsEnabled

Allows to disable all actions.

**property** ActionsEnabled: Boolean;

Set to *False* to disable all actions inherited from TrvAction <sup>(311)</sup>, linked to this control panel.

**Default value:**

*True*

#### 1.2.4.1.2 TRVAControlPanel.AddColorNameToHints

Allowing adding color names to hints of actions inherited from TrvActionCustomColor <sup>(318)</sup>.

**property** AddColorNameToHints: Boolean;

A color name can be added to the action's hint, if its UserInterface <sup>(320)</sup> = *rvacNone* (i.e. the action applies a specific color without displaying a dialog).

**Default value:**

*True*

#### 1.2.4.1.3 TRVAControlPanel.AutoDeleteUnusedStyles

Instructs to delete unused styles from documents when it's possible.

**property** `AutoDeleteUnusedStyles: Boolean;`

If *True*, all unused text, paragraph and list styles are deleted after executing `TrvActionNew`<sup>(101)</sup> and `TrvActionOpen`<sup>(106)</sup>.

It's highly recommended to leave it equal to *True*, otherwise a number of unused styles will be constantly increased (as well as sizes of documents).

If style templates are used<sup>(20)</sup>, this property is ignored: instead of deleting unused styles, "New" and "Open" actions resets styles according to `StyleTemplates`<sup>(104)</sup>.

**Default value:**

*True*

#### 1.2.4.1.4 TRVAControlPanel.ColorDialog

Specifies a color dialog for using in RichViewActions.

**property** `ColorDialog: TColorDialog;`

This is a link to `TColorDialog` component. If it is not defined, actions create and use temporal color dialogs.

If the link is defined, the user can define a set of custom colors and use them in all actions. Otherwise, this link is optional.

This property can be overridden by `ColorDialogInterface`<sup>(42)</sup> property.

#### 1.2.4.1.5 TRVAControlPanel.ColorDialogInterface

Specifies a custom color dialog for using in RichViewActions.

**property** `ColorDialogInterface: TRVACustomColorDialogInterface`<sup>(360)</sup>;

This property allows using a custom (third-party) color dialog instead of the standard `TColorDialog`.

**ColorDialogInterface** has a higher priority than `ColorDialog`<sup>(42)</sup>: if both **ColorDialogInterface** and `ColorDialog`<sup>(42)</sup> are specified, **ColorDialogInterface** is used, `ColorDialog`<sup>(42)</sup> is ignored.

#### 1.2.4.1.6 TRVAControlPanel.DefaultColor

Specifies the default background color.

**property** `DefaultColor: TColor;`

This property is assigned to `TCustomRichViewEdit.Color` property when executing `TrvActionNew`<sup>(101)</sup> or `TrvActionOpen`<sup>(106)</sup> (before loading).

**See also:**

- `DefaultMargin`<sup>(45)</sup>

- DefaultDocParameters<sup>(43)</sup>
- TrvActionNew<sup>(103)</sup>.StyleTemplates<sup>(104)</sup>

#### 1.2.4.1.7 TRVAControlPanel.DefaultControl

Specifies the editor component for using in RichViewActions.

**property** DefaultControl: TCustomRVControl;

If this link is defined, all actions inherited from TrvAction<sup>(311)</sup> (and linked with this control panel) work with this editor. This property can be overridden by Control<sup>(312)</sup> property of the specific action.

Controls of the following types can be assigned to this property:

- TRichViewEdit
- TDBRichViewEdit
- TSRichViewEdit (ScaleRichView)
- TDBSRichViewEdit (ScaleRichView)

If this link (and action.Control<sup>(312)</sup> property) is not assigned, the action works with the editor component having the input focus, or, if an editor component is not focused, with the first found editor component.

#### 1.2.4.1.8 TRVAControlPanel.DefaultCustomFilterIndex

Specifies the default index of a custom format for new documents.

**property** DefaultCustomFilterIndex: Integer;

This property is used if DefaultFileFormat<sup>(44)</sup> = *ffeCustom*. This is an index of custom format (from 1); custom formats are listed in TrvActionSaveAs.CustomFilter<sup>(98)</sup> property.

This is a temporal format assigned to new documents after executing TrvActionNew<sup>(101)</sup>. This format will be offered as a default format on the first execution of TrvActionSave<sup>(122)</sup> or TrvActionSaveAs<sup>(127)</sup> for new documents.

OnCustomFileOperation<sup>(62)</sup> occurs when files of this format need to be saved.

**Default value:**

1

#### 1.2.4.1.9 TRVAControlPanel.DefaultDocParameters

Specifies default page parameters for new documents (page size, margins, etc.)


**property** DefaultDocParameters: TRVDocParameters;

If *rvfoSaveDocProperties* is included in RVFOptions properties of TRichViewEdit control, this property is assigned to its DocParameters property in the following cases:

- when TrvActionNew<sup>(101)</sup> is executed;
- before TrvActionOpen<sup>(106)</sup> loads a file.

ZoomPercent and ZoomMode sub-properties are not assigned.

Units sub-property is not assigned too. All sizes are converted to RichViewEdit.DocParameters.Units.

 **ScaleRichView note:** this property is especially important if RichViewActions are used with TSRichViewEdit control (ScaleRichView). DefaultDocParameters define a page layout for new documents, and DefaultMargin<sup>(45)</sup> property is ignored.

**See also:**

- DefaultColor<sup>(42)</sup>
- TrvActionNew<sup>(103)</sup>.StyleTemplates<sup>(104)</sup>

#### 1.2.4.1.10 TRVAControlPanel.DefaultExt

Specifies a default file extension for dialogs in TrvActionOpen<sup>(106)</sup>, TrvActionSaveAs<sup>(127)</sup>, and TrvActionExport<sup>(110)</sup> actions.

**property** DefaultExt: TRVALocString<sup>(348)</sup>;

This value is assigned to DefaultExt property of TOpenDialog and TSaveDialog components used in these actions.

**Default value:**

'rvf'

#### 1.2.4.1.11 TRVAControlPanel.DefaultFileFormat

Specifies the default format for new documents.

**property** DefaultFileFormat: TrvFileSaveFilter<sup>(386)</sup>;

If it equals to *ffeCustom*, DefaultCustomFilterIndex<sup>(43)</sup> property defines a subformat.

This is a temporal format assigned to new documents after executing TrvActionNew<sup>(101)</sup>. This format will be offered as a default format on the first execution of TrvActionSave<sup>(122)</sup> or TrvActionSaveAs<sup>(127)</sup> for new documents.

**Default value:**

*ffeRVF*

**See also:**

- DefaultExt<sup>(44)</sup>

#### 1.2.4.1.12 TRVAControlPanel.DefaultFileName

Specifies a default file name for new documents.

**property** DefaultFileName: TRVALocString<sup>(348)</sup>;

This is a temporal file name assigned to new documents after executing TrvActionNew<sup>(101)</sup>. This file name will be offered as a default file name on the first execution of TrvActionSave<sup>(122)</sup> or TrvActionSaveAs<sup>(127)</sup> for new documents.

**Default value:**

'Untitled.rvf'

**See also:**

- DefaultExt<sup>(44)</sup>




#### 1.2.4.1.13 TRVAControlPanel.DefaultMargin

Specifies default margins for new documents.

**property** DefaultMargin: TRVPixelLength;

This property is assigned to LeftMargin, TopMargin, RightMargin, and BottomMargin properties of TCustomRichViewEdit component in the following cases:

- when TrvActionNew<sup>(101)</sup> is executed;
- before TrvActionOpen<sup>(106)</sup> loads a file.

 **ScaleRichView note:** for TSRichViewEdit, this property is ignored, see DefaultDocParameters<sup>(43)</sup>.

**Default value:**

5

**See also:**

- DefaultColor<sup>(42)</sup>
- DefaultDocParameters<sup>(43)</sup>
- TrvActionNew<sup>(103)</sup>.StyleTemplates<sup>(104)</sup>

#### 1.2.4.1.14 TRVAControlPanel.DialogFontName

The properties define the default font name for all dialog windows displayed by RichViewActions.

**property** DialogFontName: String;

**property** DialogFontNameLin: String;

**DialogFontName** is used on Windows. **DialogFontNameLin** is used on Linux.

Each property contains a semicolon-delimited list of font names, in order of preference. The first font existed in the system is used.

'default' is a special font name, valid only for Lazarus. It means that the actual font name is determined by the application.

The actual font name used by RichViewActions is returned by GetRealDialogFontName<sup>(59)</sup> method.

**Default value:**

- DialogFontName: 'Segoe UI;Tahoma;Microsoft Sans Serif;MS Sans Serif;Arial'
- DialogFontNameLin: 'default'

**See also:**

- DialogFontSize<sup>(45)</sup>

#### 1.2.4.1.15 TRVAControlPanel.DialogFontSize

Defines the default font size for all dialog windows displayed by RichViewActions.

**property** DialogFontSize: Integer;

If want to use a large font size, you should increase sizes of dialogs, see DialogZoomPercent<sup>(46)</sup> property.

**Default value:**

8

**See also:**

- DialogFontName <sup>(45)</sup>

**1.2.4.1.16 TRVAControlPanel.DialogPosition**

Defines the position of dialog windows displayed by RichViewActions.

**type**

```
TRVAFormPosition =  
(rvafpMainFormCenter, rvafpScreenCenter,  
rvafpEditorFormCenter);
```

**property** DialogPosition: TRVAFormPosition;

Value	Meaning
<i>rvafpMainFormCenter</i>	Dialog windows are positioned in the center of the application's main form
<i>rvafpScreenCenter</i>	Dialog windows are positioned in the center of the screen
<i>rvafpEditorFormCenter</i>	Dialog windows are positioned in the center of the application's target editor's form

**Default value:**

*rvafpMainFormCenter*

**1.2.4.1.17 TRVAControlPanel.DialogZoomPercent**

Allows scaling all dialog windows displayed by RichViewActions.

**property** DialogZoomPercent: Integer;

The value of this property defines sizes of RichViewActions' dialogs, %.

If you assign a large value to DialogFontSize <sup>(45)</sup>, consider increasing sizes of dialogs using this property.

**Default value:**

100

**1.2.4.1.18 TRVAControlPanel.DownloadInterface**

Allows using a third-party components to download images and external files (such as CSS files) from HTTP and HTTPS links.

**property** DownloadInterface: TRVACustomDownloadInterface <sup>(354)</sup>;

The actions may need to download images when loading/inserting RTF, DocX or HTML files, if these files contain external images or CSS files.

You can assign one of the following components to this property:

- TRVAIndyDownloadInterface <sup>(354)</sup> (to use Indy <sup>(368)</sup>)

- TRVACIDownloadInterface<sup>(353)</sup> (to use CleverComponents<sup>(367)</sup>)

#### See also

- OnDownload<sup>(64)</sup> event
- InitImportPicturesAndFiles, DoneImportPicturesAndFiles<sup>(59)</sup> methods

#### 1.2.4.1.19 TRVAControlPanel.FirstPageNumber

Defines the page number for the first page.

**property** FirstPageNumber: Integer;

If page numbers are printed, this value specifies the number printed on the first page.

#### Default value:

1

#### See also properties:

- Header, Footer<sup>(47)</sup>

#### 1.2.4.1.20 TRVAControlPanel.Header, Footer

The properties define the page headers and footers.

**property** Header: TRVAHFInfo<sup>(384)</sup>;


**property** Footer: TRVAHFInfo<sup>(384)</sup>;

You can change properties of a header or a footer by assigning subproperties.

TrvActionPageSetup<sup>(114)</sup> changes these properties as well.

The following actions print headers and footers:

- TrvActionPrint<sup>(116)</sup>
- TrvActionQuickPrint<sup>(120)</sup>
- TrvActionPrintPreview<sup>(118)</sup>

 **ScaleRichView note:** these properties are ignored when printing TSRichViewEdit or TDBSRichViewEdit controls.

#### Default value:

Header: defaults for TRVAHFInfo, but Text := '- &p -';

Footer: defaults for TRVAHFInfo

#### See also properties:

- FirstPageNumber<sup>(47)</sup>

#### 1.2.4.1.21 TRVAControlPanel.HelpType

Specifies whether the RichViewActions forms' context-sensitive Help topics are identified by a context ID or by a keyword.

**property** HelpType: THelpType;

This property affects all RichViewActions forms.

The list of values of HelpContext and HelpKeyword properties are shown below.

Values of HelpContext are hard-coded. Values of HelpKeyword can be modified by RVSetHelpKeyword<sup>(352)</sup>.

## Core RichViewActions

HelpContext	HelpKeyword	Dialog is displayed by
90000	Document background dialog	TrvActionBackground <sup>(313)</sup>
90100	Fill color dialog	TrvActionFillColor <sup>(322)</sup>
90200	Font dialog	TrvActionFontEx <sup>(149)</sup>
90300	Paragraph padding dialog	TrvActionParaBorder <sup>(191)</sup> TrvActionFillColor <sup>(322)</sup>
90301	Paragraph text margins dialog	TrvActionParaBorder <sup>(191)</sup>
90302	Document padding dialog	TrvActionBackground <sup>(313)</sup>
90400	Hyperlink dialog	TrvActionInsertHyperlink <sup>(232)</sup>
90450	Hyperlink attributes dialog	TrvActionInsertHyperlink <sup>(232)</sup> (if StyleTemplates are not used)
90460	Active hyperlink attributes dialog	TrvActionStyleTemplates <sup>(212)</sup> TrvActionAddStyleTemplate <sup>(215)</sup> (when editing properties of hypertext style)
90500	Symbol insertion dialog	TrvActionInsertSymbol <sup>(254)</sup>
90600	Table insertion dialog	TrvActionInsertTable <sup>(271)</sup>
90700	Row count dialog	TrvActionTableInsertRowsBelow <sup>(299)</sup>
90800	Picture properties dialog	TrvActionItemProperties <sup>(324)</sup>
90900	Horizontal line properties dialog	TrvActionItemProperties <sup>(324)</sup>
91000	Table properties dialog	TrvActionTableProperties <sup>(301)</sup> TrvActionItemProperties <sup>(324)</sup>
90910	Sidenote properties dialog	TrvActionItemProperties <sup>(324)</sup>
90920	Number properties dialog	TrvActionItemProperties <sup>(324)</sup>
90930	Page number and page count properties dialog	TrvActionItemProperties <sup>(324)</sup>

90940	Equation dialog	TrvActionInsertEquation <sup>(226)</sup> TrvActionItemProperties <sup>(324)</sup>
91100	Bullets and numbering dialog	TrvActionParaList <sup>(206)</sup>
91200	HTML bullets and numbering dialog	TrvActionParaList <sup>(206)</sup> (if RVAControlPanel.UserInterface <sup>(56)</sup> = <i>rvauHTML</i> )
91300	Paste special dialog	TrvActionPasteSpecial <sup>(137)</sup>
91301	Language dialog	RVA_ChooseLanguage <sup>(349)</sup> function
91400	Page setup dialog	TrvActionPageSetup <sup>(114)</sup>
91500	Paragraph border and background dialog	TrvActionParaBorder <sup>(191)</sup>
91600	Bullets and numbering customization dialog	TrvActionParaList <sup>(206)</sup>
91700	Paragraph dialog	TrvActionParagraph <sup>(198)</sup>
91800	Print preview dialog	TrvActionPrintPreview <sup>(118)</sup>
91900	Cell spacing dialog	TrvActionTableProperties <sup>(301)</sup> TrvActionItemProperties <sup>(324)</sup>
92000	Cell splitting dialog	TrvActionTableSplitCells <sup>(308)</sup>
92100	Table background dialog	TrvActionTableProperties <sup>(301)</sup> TrvActionItemProperties <sup>(324)</sup>
92200	Table border dialog	TrvActionTableProperties <sup>(301)</sup> TrvActionItemProperties <sup>(324)</sup>
92300	Text and paragraph styles dialog	TrvActionStyleTemplates <sup>(212)</sup> TrvActionAddStyleTemplate <sup>(215)</sup>
92400	Text and paragraph styles import dialog	TrvActionStyleTemplates <sup>(212)</sup> TrvActionAddStyleTemplate <sup>(215)</sup> (when importing styles from a file)
92500	Number insertion dialog	TrvActionInsertNumber <sup>(245)</sup>

92600	Visible sides dialog	TrvActionTableProperties <sup>(301)</sup> TrvActionItemProperties <sup>(324)</sup>
92700	Caption insertion dialog	TrvActionInsertCaption <sup>(223)</sup>
92800	Object alignment dialog	TrvActionVAlign <sup>(333)</sup>
92900	Table sort dialog	TrvActionTableSort <sup>(307)</sup>
93000	Bookmark dialog	TrvActionBookmarks <sup>(221)</sup>
93100	Visual styles dialog	RVA_ChooseStyle <sup>(378)</sup> function
93110	File encoding dialog	File saving and loading actions (when choosing encoding of ANSI text files)
93120	Standard text and paragraph styles dialog	TrvActionStyleTemplates <sup>(212)</sup> TrvActionAddStyleTemplate <sup>(215)</sup> (when adding a standard style)
93130	Table delimiters dialog	TrvActionTableToText <sup>(309)</sup>

## ScaleRichView

HelpContext	HelpKeyword	Dialog is displayed by
91400	Page setup dialog	TsrvActionPageSetup

ScaleRichView introduces an alternative action for displaying a page setup dialog: `TsrvActionPageSetup`. When using `TSRichViewEdit`, `TsrvActionPageSetup` must be used instead of `TrvActionPageSetup`<sup>(114)</sup>. It's assumed that only one of these actions are used in an application, so these dialogs have the same `HelpContext` and `HelpKeyword`.

## ReportWorkshop

HelpContext	HelpKeyword	Dialog is displayed by
95000	Report table cell dialog	TrvrActionCellProperties
95100	Background diagram dialog	TrvrActionCellProperties
95200	Color scale dialog	TrvrActionCellProperties
95300	Row generation rules dialog	TrvrActionRowGenerationRules
95400	Cross tabulation dialog	TrvrActionCrossTab
95500	Report properties dialog	TrvrActionDocProperties

95600	Background diagram type dialog	TrvrActionCellProperties
95700	Shape properties dialog	TrvActionItemProperties <sup>(324)</sup>
95800	Report table insertion dialog	TrvrActionInsertTable
95900	Report Wizard	TrvrActionReportWizard
96000	Master-detail link dialog	TrvrActionReportWizard

**Default value:***htContext***See also:**

- UseHelpFiles <sup>(56)</sup>
- UseDefaultHelpFile <sup>(56)</sup>

**1.2.4.1.22 TRVAControlPanel.HTMLComponent (deprecated)**

Deprecated. Do not assign this property: TRichView HTML methods provide better results without it.

Defines a link to a component allowing to import and paste HTML files.

**property** HTMLComponent: TRVHTMLComponent <sup>(387)</sup>;

You can assign either TRvHtmlImporter <sup>(368)</sup> or TrvHtmlViewImporter <sup>(369)</sup> component to this property.

If this link is defined, HTML format can be used in TrvActionInsertFile <sup>(228)</sup>, TrvActionPaste <sup>(135)</sup>, and TrvActionPasteSpecial <sup>(137)</sup> actions.

**1.2.4.1.23 TRVAControlPanel.Language**

Defines the language for user interface.

**property** Language: TRVALanguageName <sup>(348)</sup>;

This property returns a language name in English.

You can assign either English or native language name to this property.

See Localization of RichViewActions <sup>(340)</sup>.

**1.2.4.1.24 TRVAControlPanel.MetafileCompatibility**

Specifies whether the printing output is compatible with metafiles.

**property** MetafileCompatibility: Boolean;

This property is ignored when printing TRichViewEdit, because TrvActionPrint <sup>(116)</sup>, TrvActionQuickPrint <sup>(120)</sup>, and TrvActionPrintPreview <sup>(118)</sup> use RVPrint <sup>(53)</sup>. MetafileCompatibility property instead.

This property is used when printing ScaleRichView (see TsrvActionPrint and TsrvActionQuickPrint actions in the ScaleRichView help file).

**Default value:***False***1.2.4.1.25 TRVAControlPanel.PixelBorders**

Specifies pixels as the units of measurement for widths of lines in RichViewActions dialogs.

**property** PixelBorders: Boolean;

If this property is *True*, widths of lines in RichViewActions dialogs are measured in pixels, even if UnitsDisplay<sup>(55)</sup> <>rvuPixels.

This property is used by the following actions:

- TrvActionParaBorder<sup>(191)</sup> for border widths, including an internal width;
- TrvActionTableProperties<sup>(301)</sup> for widths of a table and cells borders;
- TrvActionItemProperties<sup>(324)</sup> for widths of a table and cells borders; for width of horizontal lines.

**Default value:***False***1.2.4.1.26 TRVAControlPanel.RVFFilter**

Defines the file filter for RVF format.

**property** RVFFilter: TRVALocString<sup>(348)</sup>;

This value is used only if RVFLocalizable<sup>(52)</sup> = *False*. This string will appear in file filters in dialogs used in TrvActionOpen<sup>(106)</sup>, TrvActionSaveAs<sup>(127)</sup>, TrvActionExport<sup>(110)</sup>, TrvActionInsertFile<sup>(228)</sup>, TrvActionStyleTemplates<sup>(212)</sup> (style import) actions.

If RVFLocalizable<sup>(52)</sup> = *True*, this property is ignored (RVF file filter name is assigned automatically when a new language is applied).

**Default value:**

'RichView Files (\*.rvf)|\*.rvf'

**See also properties:**

- RVStylesFilter<sup>(53)</sup>
- DefaultExt<sup>(44)</sup>

**1.2.4.1.27 TRVAControlPanel.RVFLocalizable**

Specifies whether UI text related to RVF format is maintained automatically by the localization procedures.

**property** RVFLocalizable: Boolean;

If *True*, all text related to RVF format is maintained automatically (i.e. localized text is used when UI language is changed).

If *False*, RVFFilter<sup>(52)</sup>, RVFormatTitle<sup>(53)</sup>, RVStylesFilter, RVStylesExt<sup>(53)</sup> properties are used.

**Default value:***True***See also:**

- XMLLocalizable<sup>(58)</sup>



- DefaultExt<sup>(44)</sup>

#### 1.2.4.1.28 TRVAControlPanel.RVFormatTitle

Specifies how RVF (RichView Format) appears in the "Paste Special" dialog.

**property** RVFormatTitle: TRVALocString<sup>(348)</sup>;

This value is used only if RVFLocalizable<sup>(52)</sup>=*False*. This string will appear in the dialog used in TrvActionPasteSpecial<sup>(137)</sup> action and in RVA\_GetProgressMessage<sup>(343)</sup> function.

If RVFLocalizable<sup>(52)</sup>=*True*, this property is ignored (RVF title is assigned automatically when a new language is applied).

**Default value:**

'RichView Format'

#### 1.2.4.1.29 TRVAControlPanel.RVPrint

Contains a link to TRVPrint component.

**property** RVPrint: TRVPrint;

This link (if defined) is used by TrvActionPrint<sup>(116)</sup>, TrvActionQuickPrint<sup>(120)</sup>, TrvActionPrintPreview<sup>(118)</sup>, TrvActionPageSetup<sup>(114)</sup> actions.

For TrvActionPageSetup<sup>(114)</sup>, this link *must* be defined, otherwise the action will be disabled (because this action reads/writes margins from/to properties of this component).

#### 1.2.4.1.30 TRVAControlPanel.RVStylesFilter, RVStylesExt

RVStylesFilter defines the file filter for files containing style templates.

RVStylesExt defines the file extension for files containing style templates.

**property** RVStylesFilter: TRVALocString<sup>(348)</sup>;

**property** RVStylesExt: TRVALocString<sup>(348)</sup>;

These values are used only if RVFLocalizable<sup>(52)</sup>=*False*. They will appear in file filters in the style import and export dialogs used in TrvActionStyleTemplates<sup>(212)</sup>, TrvActionAddStyleTemplate<sup>(215)</sup> actions.

If RVFLocalizable<sup>(52)</sup>=*True*, these properties are ignored (a file filter and extension are assigned automatically when a new language is applied).

**Default values:**

- RVStylesFilter: 'RichView Styles (\*.rvst)|\*.rvst'
- sRVStylesExtension: 'rvst'

**See also properties:**

- RVFFilter<sup>(52)</sup>

#### 1.2.4.1.31 TRVAControlPanel.SearchScope

Defines a search scope for TrvActionFind<sup>(132)</sup>, TrvActionFindNext<sup>(134)</sup>, and TrvActionReplace<sup>(141)</sup> actions.

**type**

```
TRVASearchScope = (rvssFromCursor, rvssAskUser, rvssGlobal);
```

**property** SearchScope: TRVASearchScope;

Value	Meaning
<i>rvssFromCursor</i>	Searching from the caret position to the end/beginning of the document.
<i>rvssAskUser</i>	Searching from the caret position to the end/beginning of the document, then asking the user whether to continue from the beginning/end, then (if "yes") searching from the beginning/end.
<i>rvssGlobal</i>	Searching from the caret position to the end/beginning of the document, then searching from the beginning/end.

**Default value:**

*rvssAskUser*

#### 1.2.4.1.32 TRVAControlPanel.ShowSoftPageBreaks

Specifies whether page breaks should be displayed in TCustomRichViewEdit component when possible.

**property** ShowSoftPageBreaks: Boolean;

Soft page breaks can be displayed after the document is formatted for printing (after executing TrvActionPrint<sup>(116)</sup>, TrvActionQuickPrint<sup>(120)</sup>, or TrvActionPrintPreview<sup>(118)</sup> actions) until it is changed.

**Default value:**

*True*

#### 1.2.4.1.33 TRVAControlPanel.SpellInterface

Allows using a third-party components to correct spelling errors and to find synonyms.

**property** SpellInterface: TRVACustomSpellInterface<sup>(357)</sup>;

You can assign one of the following components to this property:

- TRVAAddictSpellInterface<sup>(356)</sup> (to use Addict<sup>(362)</sup>)
- TRVAASpellInterface<sup>(357)</sup> (to use ASpell<sup>(363)</sup>)
- TRVADXSpellInterface<sup>(358)</sup> (to use ExpressSpellChecker<sup>(364)</sup>)
- TRVAHunSpellInterface<sup>(359)</sup> (to use HunSpell<sup>(365)</sup>)
- TRVAPolarSpellInterface<sup>(359)</sup> (to use Polar SpellChecker<sup>(366)</sup>)

#### 1.2.4.1.34 TRVAControlPanel.TableGridStyle

Defines the pen style for drawing grid lines in table.

**property** TableGridStyle: TPenStyle;

This property is used in TrvActionTableGrid<sup>(296)</sup> action.

**Default value:**

*psDot*

#### 1.2.4.1.35 TRVAControlPanel.UnitsDisplay

Specifies units of measurement for using in RichViewActions dialogs.

**property** UnitsDisplay: TRVUnits;

These units are used for entering lengths in dialog windows. For pixels, only integer values are allowed. Values measured in other units can be fractional.

Small lengths, for example paragraph spacing, are measured either in pixels (if **UnitsDisplay**=*rvuPixels*) or in points (if **UnitsDisplay** has another value).

Line widths can be measured in pixels even if **UnitsDisplay**<>*rvuPixels*, if PixelBorders<sup>(52)</sup>=*True*.

Units of rulers must be assigned separately, see TRVRuler<sup>(80)</sup>.UnitsDisplay<sup>(85)</sup>.

**Default value:**

*rvuPixels*

#### 1.2.4.1.36 TRVAControlPanel.UnitsProgram

Specifies units of measurement for properties of TRVStyleLength type of some actions.

**property** UnitsProgram: TRVStyleUnits;

These units are used for properties of the type TRVStyleLength of the following actions:

- TrvActionNew<sup>(103)</sup>,
- TrvActionIndentInc<sup>(187)</sup>,
- TrvActionIndentDec<sup>(186)</sup>,
- TrvActionParaList<sup>(206)</sup>,
- TrvActionParaBullets<sup>(195)</sup>,
- TrvActionParaNumbering<sup>(209)</sup>,
- TrvActionStyleTemplates<sup>(212)</sup>,
- TrvActionInsertPicture<sup>(249)</sup>,
- TrvActionInsertHLine<sup>(231)</sup>,
- TrvActionInsertTable<sup>(271)</sup>.

Not all actions' properties of the type TRVStyleLength are measured in these units. Generally, if the action takes values from the editor, modifies them and applies back to the editor, properties of this action are measured in Editor.Style.Units (where Editor is the target TRichViewEdit control).

However, if the action applies some predefined values to the editor, these values are measured using this property.

At design time, values of all action's properties measured in these units are displayed with units ('px' or 'tw') in the Object Inspector (in Delphi 5 or newer). Values of properties measured in Editor.Style.Units are not displayed with units.

Note: assigning a new value to this property does not convert values of actions' properties. To convert properties to new units of measurement, RVA\_ConvertToPixels/RVA\_ConvertToTwips<sup>(378)</sup> must be called before changing value of this property.

**Default value:**

*rvstuPixels*

#### 1.2.4.1.37 TRVAControlPanel.UseDefaultHelpFile

Specifies whether Application.HelpFile can be used.

**property** UseDefaultHelpFile: Boolean;

This property is ignored if UseHelpFiles<sup>(56)</sup> = *False*.

Application.HelpFile is used only if other help file is not defined<sup>(347)</sup> for the current UI Language<sup>(51)</sup>.

**Default value:**

*False*

**See also:**

- HelpType<sup>(47)</sup>

#### 1.2.4.1.38 TRVAControlPanel.UseHelpFiles

Specifies whether help files should be used if available.

**property** UseHelpFiles: Boolean;

Help files can be:

- provided for RichViewActions; each language<sup>(51)</sup> may have its own help file (see RVA\_SetHelpFile<sup>(347)</sup> procedure).
- if UseDefaultHelpFile<sup>(56)</sup> = *True*, and the current language does not have a help file specified, RichViewActions uses Application.HelpFile.

If a help file is available, additional "Help" button is shown in all RichViewActions dialogs.

**Default value:**

*True*

**See also:**

- HelpType<sup>(47)</sup>
- UseDefaultHelpFile<sup>(56)</sup>

#### 1.2.4.1.39 TRVAControlPanel.UserInterface

Allows hiding UI options incompatible with some formats.

**type**

```
TRVAUserInterface = (rvaiFull, rvaiHTML, rvaiRTF, rvaiText);
```

**property** UserInterface: TRVAUserInterface;

Value	Meaning
<i>rvaiFull</i>	All user interface controls are available. Optimal for RVF documents.
<i>rvaiHTML</i>	All user interface controls providing options incompatible with HTML are hidden. The most noticeable changes: <ul style="list-style-type: none"> <li>• an alternative HTML-style dialog in TrvActionParaList<sup>(206)</sup> action;</li> </ul>

	<ul style="list-style-type: none"> <li>• a hidden "Tabs" page in the TrvActionParagraph<sup>(198)</sup> action's dialog.</li> </ul>
<i>rvauirTF</i>	<p>All user interface controls providing options incompatible with RTF are hidden. The most noticeable changes:</p> <ul style="list-style-type: none"> <li>• TrvActionColor<sup>(315)</sup>, TrvActionVAlign<sup>(333)</sup>, TrvActionBackground<sup>(313)</sup>, TrvActionTableCellRotation180<sup>(289)</sup> are disabled;</li> <li>• in the table properties dialog, table background is disabled, some options for table borders are not available;</li> <li>• in the image properties dialog, image alignment is not available;</li> <li>• in the paragraph borders dialog, some options are disabled</li> </ul>
<i>rvauirText</i>	<p>All user interface controls providing options incompatible with a plain text are hidden or disabled.</p>

**Default value:***rvauirFull***1.2.4.1.40 TRVAControlPanel.UseTextCodePageDialog**

Specifies whether the file-related actions can display a dialog for choosing a text file codepage.

**property** UseTextCodePageDialog: Boolean;

This property affects the following actions:

- TrvActionOpen<sup>(106)</sup>
- TrvActionExport<sup>(110)</sup>
- TrvActionSave<sup>(122)</sup>
- TrvActionInsertFile<sup>(228)</sup>

It affects loading and saving text files in the editor containing Unicode text.

If *False*, non-Unicode text files are saved using the system-default code page; a code page for insertion is taken from the Charset of the current text.

If *True*, a dialog for choosing a code page is displayed (the saving and exporting actions display this dialog only if the document is completely in Unicode). In addition to ANSI code pages, it allows choosing UTF-8 and UTF-16.

**Default value:***True***1.2.4.1.41 TRVAControlPanel.UseXPThemes**

Specifies whether controls on RichViewActions dialog should use Windows themes (visual styles) if available.

**property** UseXPThemes: Boolean;

This property affects the following controls placed on dialogs:

- TRichView

- TRVPrintPreview
- TRVOfficeRadioButton
- TRVOfficeRadioGroup
- TRVColorCombo
- TRVColorGrid
- color picker window used in TrvActionCustomColor<sup>(318)</sup>

Other controls use themes if it is supported by Delphi (and a manifest is included).

**Default value:**

*True*

#### 1.2.4.1.42 TRVAControlPanel.XMLComponent

Defines a link to a component allowing to load and save XML files.

**property** XMLComponent: TRVXMLComponent<sup>(387)</sup>;

You can assign TRichViewXML<sup>(368)</sup> component to this property.

If this link is defined, XML format can be used in TrvActionOpen<sup>(106)</sup>, TrvActionSaveAs<sup>(127)</sup>, TrvActionExport<sup>(110)</sup> and TrvActionInsertFile<sup>(228)</sup> actions.

**See also:**

- XMLFilter<sup>(58)</sup>
- XMLLocalizable<sup>(58)</sup>

#### 1.2.4.1.43 TRVAControlPanel.XMLFilter

Defines the file filter for XML format.

**property** XMLFilter: TRVALocString<sup>(348)</sup>;

This property is used if XMLComponent<sup>(58)</sup> is assigned.

This value is used only if XMLLocalizable<sup>(58)</sup> = *False*. This string will appear in file filters in dialogs used in TrvActionOpen<sup>(106)</sup>, TrvActionSaveAs<sup>(127)</sup>, TrvActionExport<sup>(110)</sup>, TrvActionInsertFile<sup>(228)</sup> actions.

If XMLLocalizable<sup>(58)</sup> = *True*, this property is ignored (XML file filter name is assigned automatically when a new language is applied).

**Default value:**

'XML Files (\*.xml)|\*.xml'

#### 1.2.4.1.44 TRVAControlPanel.XMLLocalizable

Specifies whether UI text related to XML format is maintained automatically by the localization procedures.

**property** RVFLocalizable: Boolean;

This property is used if XMLComponent<sup>(58)</sup> is assigned.

If *True*, all text related to XML format is maintained automatically (i.e. localized text is used when UI language is changed).

If *False*, XMLFilter<sup>(58)</sup> property is used.

**Default value:***True***See also:**

- RVFLocalizable <sup>(52)</sup>

## 1.2.4.2 Methods

### In TRVAControlPanel

Activate <sup>(59)</sup>  
 DoneImportPicturesAndFiles <sup>(59)</sup>  
 InitImportPicturesAndFiles <sup>(59)</sup>

#### 1.2.4.2.1 TRVAControlPanel.Activate

Makes this control panel the main control panel in the application.

**procedure** Activate;

This method is useful if you have more than one control panel in an application. The first created control panel becomes the main control panel automatically.

The main control panel is used for actions and components that do not have a link to the specific control panel.

The main control panel is returned in MainRVAControlPanel <sup>(388)</sup> variable.

**See also:**

- TrvCustomAction.ControlPanel <sup>(335)</sup>
- TRVStyleTemplateCombo/ListBox.ControlPanel <sup>(89)</sup>

#### 1.2.4.2.2 TRVAControlPanel.GetRealDialogFontName

Returns the font name used in all dialog windows of RichViewActions.

**function** GetRealDialogFontName: TFontName;

The function uses DialogFontName/DialogFontNameLin <sup>(45)</sup> properties.

#### 1.2.4.2.3 TRVAControlPanel.InitImportPictures, DoneImportPictures, DoImportPicture

Allows using DownloadInterface <sup>(46)</sup> component for a loading operation that were not initialized by RichViewActions.

```
procedure InitImportPicturesAndFiles(Sender: TrvAction (311);  
    Editor: TCustomRichViewEdit);  
procedure DoneImportPicturesAndFiles(  
    Editor: TCustomRichViewEdit);
```

Normally, when loading is initiated by an action, DownloadInterface <sup>(46)</sup> is used automatically. The following actions initializes loading:

- TrvActionOpen <sup>(106)</sup>
- TrvActionInsertFile <sup>(228)</sup>

- TrvActionPaste<sup>(135)</sup>
- TrvActionPasteSpecial<sup>(137)</sup>

However, when loading is not initiated by an action, DownloadInterface<sup>(46)</sup> is not used. To solve this problem, call **InitImportPicturesAndFiles** before loading, and **DoneImportPicturesAndFiles** after.

#### Parameters:

**Sender** will be used as a parameter of OnDownload<sup>(64)</sup> event

**Editor** is a target editor.

#### Example:

This code allows downloading external pictures referred by files that are inserted in the editor as a result of drag-and-drop operation.

```
procedure TMyForm.MyRichViewEditBeforeOleDrop(Sender: TObject);
begin
    MyRVAControlPanel.InitImportPicturesAndFiles(nil,
        (Sender as TCustomRichViewEdit));
end;
procedure TForm3.RichViewEdit1AfterOleDrop(Sender: TObject);
begin
    MyRVAControlPanel.DoneImportPicturesAndFiles(
        (Sender as TCustomRichViewEdit));
end;
```

### 1.2.4.3 Events

#### In TRVAControlPanel

- OnAddStyle<sup>(60)</sup>
- OnBackgroundChange<sup>(61)</sup>
- OnChoosePicture<sup>(61)</sup>
- OnCreateForm<sup>(62)</sup>
- OnCustomFileOperation<sup>(62)</sup>
- OnDownload<sup>(64)</sup>
- OnGetActionControlCoords<sup>(64)</sup>
- OnGetHeaderFooterCode<sup>(65)</sup>
- OnMarginsChanged<sup>(65)</sup>
- OnStyleNeeded<sup>(66)</sup>
- OnViewChanged<sup>(66)</sup>

#### 1.2.4.3.1 TRVAControlPanel.OnAddStyle

Occurs after some action added a style as a result of an editing operation.

##### type

```
TRVAddStyleEvent = procedure (Sender: TrvAction(311);
    StyleInfo: TCustomRVInfo) of object;
```



**property** OnAddStyle: TRVAddStyleEvent;

Actions may add text, paragraph, and list styles as a result of editing operations.

Depending on the style type, **StyleInfo** can be of TFontInfo, TParaInfo, or TRVListInfo class.

If you need to perform additional operations on new styles, use this event.

**See also:**

- OnStyleNeeded<sup>(66)</sup>
- TCustomRichView.OnAddStyle

#### 1.2.4.3.2 TRVAControlPanel.OnBackgroundChange

Occurs after TrvActionBackground<sup>(313)</sup> or TrvActionColor<sup>(315)</sup> actions change background properties.

**property** OnBackgroundChange: TRVAEditEvent<sup>(384)</sup>;

#### 1.2.4.3.3 TRVAControlPanel.OnChoosePicture

Using this event you can provide your own user interface for opening a picture.

**type**

```
TRVChoosePictureEvent = procedure (Sender: TObject;
  Editor: TCustomRichViewEdit;
  OwnerObject: TObject; const DialogTitle: TRVALocString(348);
  out FileName: TRVUnicodeString;
  out Graphic: TGraphic; var DoDefault: Boolean) of object;
```

**property** OnChoosePicture: TRVChoosePictureEvent;

This event is called in the following cases:

Picture for...	Called by the action	OwnerObject parameter
New picture item	TrvActionInsertPicture <sup>(249)</sup>	<i>nil</i>
Existing picture item	TrvActionItemProperties <sup>(324)</sup>	Picture item (TrvGraphicItemInfo or inherited)
Document background	TrvActionBackground <sup>(313)</sup>	Editor (inherited from TCustomRichViewEdit)
Table background	TrvActionItemProperties <sup>(324)</sup> TrvActionTableProperties <sup>(301)</sup>	Table (TRVTableItemInfo or inherited)
Table cell background	TrvActionItemProperties <sup>(324)</sup> TrvActionTableProperties <sup>(301)</sup>	Table (TRVTableItemInfo or inherited)
List marker picture	TrvActionParaList <sup>(206)</sup>	List level (TRVListLevel)

**Input parameters:**

**Sender** – the action that called the event

**Editor** – the target editor

**OwnerObject** depends on the action, see the table above.

**DialogTitle** – a suggested dialog title (if it is an empty string, a default dialog title must be used)

**DoDefault** = *True*

**Input parameters:**

**Graphic** – chosen picture; if the user canceled the selection, it must be *nil*.

**FileName** – file name (or another picture identifier). Depending on settings, it may be stored in the item.

**DoDefault:**

- if *True*, the action must perform the default way of picture choosing (using TOpenPicture dialog); in this case, **Graphic** must be *nil*;

- if *False*, and **Graphic** is not *nil*, the action must assign **Graphic** to the item. If **Graphic** is *nil*, the action should do nothing.

#### 1.2.4.3.4 TRVAControlPanel.OnCreateForm

Occurs before any RichViewActions' dialog window is displayed.

**type**

```
TRVCreateFormEvent = procedure (Form: TForm) of object;
```

**property** OnCreateForm: TRVCreateFormEvent;

The event allows setting properties to the form before it is displayed.

#### 1.2.4.3.5 TRVAControlPanel.OnCustomFileOperation

Occurs when the application needs to save or load a file in a custom format.

**type**

```
TRVAFileOperation = (rvafoOpen, rvafoSave, rvafoExport, rvafoInsert);
TRVCustomFileOperationEvent = procedure (Sender: TrvAction311;
  Edit: TCustomRichViewEdit; const FileName: TRVUnicodeString;
  Operation: TRVAFileOperation; var SaveFormat: TrvFileSaveFilter386;
  var CustomFilterIndex: Integer; var Success: Boolean) of object;
```

**property** OnCustomFileOperation: TRVCustomFileOperationEvent;

This event occurs when the following actions are executed:

- TrvActionOpen<sup>106</sup> (**Operation**=*rvafoOpen*)
- TrvActionSave<sup>122</sup> (**Operation**=*rvafoSave*)
- TrvActionExport<sup>110</sup> (**Operation**=*rvafoExport*)
- TrvActionInsertFile<sup>228</sup> (**Operation**=*rvafoInsertFile*)

**FileName** is a file name (full path) for opening/saving.

**CustomFilterIndex** is an index of custom format (in the CustomFilter property of the corresponding action), from 1.

For an opening operation, you can define a saving format in **SaveFormat** parameter (this format will be used for subsequent saving of document loaded from this file); you can also change **CustomFilterIndex** (on input, it corresponds to TrvActionOpen.CustomFilter<sup>(108)</sup>; on output, it must correspond to TrvActionSaveAs.CustomFilter<sup>(98)</sup>, if **SaveFormat=ffeCustom**). For all other operations, **SaveFormat** and output value of **CustomFilterIndex** are ignored.

Set **Success** to *True* if you loaded/saved file successfully.

**See also:**

- TrvActionSave.SuppressNextErrorMessage<sup>(125)</sup>

You can use LoadCSV<sup>(380)</sup> function to implement an insertion of CSV files.

## Example

Let you implemented saving and loading TRichView in your format. Let its extension is 'myf'.

Let you implemented the functions:

```
function SaveToMyFormat(Edit: TCustomRichViewEdit;
  const FileName: TRVUnicodeString): Boolean;
function LoadFromMyFormat(Edit: TCustomRichViewEdit;
  const FileName: TRVUnicodeString): Boolean;
```

These function must return *True* on success.

First, assign CustomFilter property of TrvActionOpen<sup>(106)</sup> and TrvActionSaveAs<sup>(127)</sup>. You can do it at design time in the Object Inspector, or in code:

```
rvActionsResource.rvActionOpen1.CustomFilter(108) :=
  'My Files (*.myf)|*.myf';
rvActionsResource.rvActionSaveAs1.CustomFilter(98) :=
  'My Files (*.myf)|*.myf';
```

Make sure that *ffiCustom* is in TrvActionOpen.Filter<sup>(108)</sup>, and *ffeCustom* is in TrvActionSaveAs.Filter<sup>(129)</sup>.

Next, process RVAContolPanel.OnCustomFileOperation event:

```
procedure TForm3.RVAControlPanel1CustomFileOperation(Sender: TrvAction(311);
  Edit: TCustomRichViewEdit; const FileName: TRVUnicodeString;
  Operation: TRVAFileOperation; var SaveFormat: TrvFileSaveFilter(386);
  var CustomFilterIndex: Integer; var Success: Boolean);
begin
  case Operation of
    rvafoOpen:
      begin
        try
          // This example assumes that there is only one custom open
          // format, so it does not check CustomFilterIndex
          // (it must be 1)
          Success := LoadFromMyFormat(Edit, FileName);
        end;
      rvafoSave:
```

```

begin
    // This example assumes that there is only one custom save
    // format, so it does not check CustomFilterIndex
    // (it must be 1)
    Success := SaveToMyFormat(Edit, FileName);
end;
end;
end;

```

#### 1.2.4.3.6 TRVAControlPanel.OnDownload

Occurs when DownloadInterface<sup>(46)</sup> downloads an image.

##### type

```

TRVDownloadEvent = procedure (Sender: TrvAction;
    const Source: TRVUnicodeString) of object;

```

**property** OnDownload: TRVDownloadEvent;

For each image, this event occurs twice:

- before downloading, **Source** is the image file name (HTTP location)
- after downloading, **Source** is an empty string.

##### See also:

- how to display a localized text message about downloading<sup>(343)</sup>

#### 1.2.4.3.7 TRVAControlPanel.OnGetActionControlCoords

The event requests coordinates of the control which called the given action.

##### type

```

TRVGetActionControlCoordsEvent = procedure (Sender: TrvAction;
    var R: TRect) of object;

```

**property** OnGetActionControlCoords: TRVGetActionControlCoordsEvent;

This event may occur when one of the following actions needs to position a color-picker window relative to the control which called this action:

- TrvActionFontColor<sup>(148)</sup>
- TrvActionFontBackColor<sup>(147)</sup>
- TrvActionParaColor<sup>(195)</sup>
- TrvActionColor<sup>(315)</sup>

(if UserInterface<sup>(320)</sup>=rvacAdvanced)

This event occurs only when the component linked to the action is not a TControl (if a control executes the action, coordinates are calculated automatically).

##### Input parameters:

**Sender** – the action that is being executed.

**R** = Rect(0,0,0,0).

##### Output parameter:

**R** – screen coordinates of the control (if non-empty).

If this action is not processed, or if empty **R** is returned, a color-picker window will be displayed at the mouse pointer position.

#### Example:

This example shows how to work with TdxBarManager by [Developer Express](#):

```
procedure TMyForm.RVAControlPanel1GetActionControlCoords (
  Sender: TrvAction; var R: TRect);
begin
  if (Sender.ActionComponent<>nil) and
    (Sender.ActionComponent is TdxBarItem) then
    with TdxBarItem(Sender.ActionComponent).ClickItemLink do begin
      R := BarControl.GetItemRect (Control);
      R.TopLeft := BarControl.ClientToScreen (R.TopLeft);
      R.BottomRight := BarControl.ClientToScreen (R.BottomRight);
    end;
end;
```

#### 1.2.4.3.8 TRVAControlPanel.OnGetHeaderFooterCode

This event allows requesting a value for the code in the header/footer string.

```
type
  TRVHeaderFooterCodeEvent = procedure (Sender: TObject;
    Edit: TCustomRichViewEdit; const Code: TRVUnicodeString;
    var Value: TRVUnicodeString) of object;
```

```
property OnGetHeaderFooterCode: TRVHeaderFooterCodeEvent;
```

The string is stored in Header<sup>(47)</sup>.Text<sup>(384)</sup> and Footer<sup>(47)</sup>.Text<sup>(384)</sup>. It can be edited in the dialog displayed by TrvActionPageSetup<sup>(114)</sup>.

#### Input parameters:

**Sender** – TRVPrint component

**Editor** – the editor for printing.

**Code** – the one-letter code (English character). The following codes are reserved: 'P', 'p', 'd', 't'.

**Value** is empty on input.

Output parameter:

**Value** – text to insert in the place of this code.

#### 1.2.4.3.9 TRVAControlPanel.OnMarginsChanged

Occurs after some action changes TCustomRichViewEdit control's margins (LeftMargin, TopMargin, RightMargin, and BottomMargin properties).

```
property OnMarginsChanged: TRVAEditEvent(384);
```

Margins can be changed by TrvActionOpen<sup>(106)</sup> and TrvActionBackground<sup>(313)</sup> actions.

If you use TRVRuler<sup>(80)</sup>, you must update it in this event:

```

procedure TForm1.RVAControlPanel1MarginsChanged(Sender: TrvAction311;
  Edit: TCustomRichViewEdit);
begin
  RVRule1.UpdateRulerMargins87;
end;

```

#### 1.2.4.3.10 TRVAControlPanel.OnStyleNeeded

Occurs when some action needs a style for an editing operation.

##### type

```

TRVStyleNeededEvent = procedure (Sender: TrvAction311;
  RVStyle: TRVStyle; StyleInfo: TCustomRVInfo;
  var StyleNo: Integer) of object;

```

**property** OnStyleNeeded: TRVStyleNeededEvent;

Actions may need text, paragraph and list styles.

**StyleInfo** is a sample of the required style. **Sender** needs a style with properties like in **StyleInfo**.

Depending on the style type, **StyleInfo** can be of TFontInfo, TParaInfo, or TRVListInfo class.

You need to return **StyleNo**, the index of style in the proper collection (TextStyles, ParaStyles, ListStyles) of **RVStyle** component. If you return -1, the action will perform a default processing (reusing existing styles if possible, adding new styles otherwise).

This event is rarely used. You can use it, for example, to disallow adding new styles (always return an index of an existing style the most similar to **StyleInfo**). Or you can prevent some style attributes from appearing in document as a result of editing operation (by changing **StyleInfo** properties and returning -1 in **StyleNo**).

#### 1.2.4.3.11 TRVAControlPanel.OnViewChanged

Occurs when some action changes view properties of the target editor.

**property** OnViewChanged: TRVAEditEvent2<sup>384</sup>;

This event is called by TrvActionNew<sup>103</sup> and TrvActionOpen<sup>106</sup> actions. In ScaleRichView, this event is also called by the actions that change view layouts and zooming in the target editor.

Usually, it makes sense to process this event only in ScaleRichView, if TSVRichViewEdit is used as a target editor. In this event you can update user interface components that display the current zooming percent.

### 1.2.5 TRVAPopupMenu

TRVAPopupMenu encapsulates the properties, methods, and events of a pop-up menu for TCustomRichViewEdit.

**Unit** RichViewActions, RVAPopupActionBar;

**Syntax, if TNT Controls<sup>370</sup> are not used:**

```

TRVAPopupMenu = class (TPopupMenu, IRVAPopupMenu)

```

**Syntax, if TNT Controls<sup>370</sup> are used:**

```

TRVAPopupMenu = class (TTntPopupMenu, IRVAPopupMenu)

```

**Syntax:**

```

TRVATBPopupMenu = class (TTBPopupMenu, IRVAPopupMenu)
TRVATBXPopupMenu = class (TTBXPopupMenu, IRVAPopupMenu)
TRVASPTBXPopupMenu = class (TSPTBXPopupMenu, IRVAPopupMenu)
TRVAPopupActionBar = class (TPopupActionBar, IRVAPopupMenu)

```

**Hierarchy**

(Hierarchy of TRVAPopupMenu, if TNT Controls<sup>(370)</sup> are not used)

*TObject*

*TPersistent*

*TComponent*

*TMenu*

*TPopupMenu*

**Description****Menus**

There are several similar components representing pop-up menu:

- TRVAPopupMenu is inherited from the standard TPopupMenu, or, if TNT Controls<sup>(370)</sup> are used, from TTntPopupMenu.
- TRVAPopupActionBar is inherited from TPopupMenuActionBar, available for Delphi 2006 or newer, defined in RVAPopupActionBar unit.
- TRVATBPopupMenu is available if Toolbar2000<sup>(371)</sup> is used.
- TRVATBXPopupMenu is available if TBX<sup>(371)</sup> is used.
- TRVASPTBXPopupMenu is available if SpTBXLib<sup>(370)</sup> is used.

**Standard actions**

This menu is automatically filled with RichViewActions from ActionList<sup>(69)</sup>.

The following actions are always added (if available):

- TrvActionCut<sup>(131)</sup>
- TrvActionCopy<sup>(131)</sup>
- TrvActionPaste<sup>(135)</sup>
- TrvActionFontEx<sup>(149)</sup>
- TrvActionParagraph<sup>(198)</sup>
- TrvActionParaList<sup>(206)</sup>

The following action is added if an image or a *break* is at the position of the caret:

- TrvActionItemProperties<sup>(324)</sup>

The following actions are added if there is a table with multicell selection:

- TrvActionTableInsertRowsAbove<sup>(298)</sup>
- TrvActionTableInsertRowsBelow<sup>(299)</sup>
- TrvActionTableInsertColLeft<sup>(297)</sup>
- TrvActionTableInsertColRight<sup>(297)</sup>

The following actions are added if the table row(s)/column(s) are selected completely:

- TrvActionTableDeleteRows<sup>(295)</sup>
- TrvActionTableDeleteCols<sup>(295)</sup>

The following action is added if there is a table without edited cell:

- TrvActionFillColor<sup>(322)</sup>

The following action is added if there is a table with selection, and the selected cells can be merged:

- TrvActionTableMergeCells<sup>(300)</sup>

Otherwise, the following action is added:

- TrvActionTableSplitCells<sup>(308)</sup>

The following action is added if the current item is a table:

- TrvActionTableProperties<sup>(301)</sup>

## Live spelling

This menu supports live spelling. If Addict 3 or 4<sup>(362)</sup> is used, the live spelling commands appear automatically (suggestions for the current misspelled word, "Add to Dictionary" and "Ignore" commands). For other spellcheckers, you need to process the events:

- OnLiveSpellAdd<sup>(69)</sup>
- OnLiveSpellGetSuggestions<sup>(70)</sup>
- OnLiveSpellIgnoreAll<sup>(70)</sup>

## Adding custom commands

This menu is recreated each time before it is displayed, before OnPopup event occur. All menu items are deleted and new items are added.

If you need adding another commands (or delete some existing commands), do it in OnPopup event.

### 1.2.5.1 Properties

#### In TRVAPopupMenu

---

- ActionList<sup>(69)</sup>
- MaxSuggestionsCount<sup>(69)</sup>

#### Derived from TPopupMenu

---

- Alignment
- AutoHotkeys
- AutoLineReduction
- AutoPopup
- BiDiMode
- HelpContext
- Images
- MenuAnimation
- OwnerDraw
- ParentBiDiMode
- PopupComponent
- PopupPoint
- TrackButton



### 1.2.5.1.1 TRVAPopupMenu.ActionList

Defines a link to TActionList or TActionManager component.

**property** ActionList: TCustomActionList;

The menu can be filled with actions only if this link is defined.

### 1.2.5.1.2 TRVAPopupMenu.MaxSuggestionsCount

Specifies the maximal possible count of suggestions displayed for changing the current misspelled word.

**property** MaxSuggestionsCount: Integer;

If this property is set to positive value, it limits the count of suggestions displayed in the menu.

Other suggestions returned by Addict 3<sup>(362)</sup> or OnLiveSpellGetSuggestions<sup>(70)</sup> event are ignored.

**Default value:**

0 (all suggestions are displayed)

## 1.2.5.2 Events

### In TRVAPopupMenu

- OnLiveSpellAdd<sup>(69)</sup>
- OnLiveSpellGetSuggestions<sup>(70)</sup>
- OnLiveSpellIgnoreAll<sup>(70)</sup>
- OnLiveSpellWordReplace<sup>(71)</sup>

### Derived from TPopupMenu

OnChange  
OnPopup

### 1.2.5.2.1 TRVAPopupMenu.OnLiveSpellAdd

Occurs when the user clicks "Add to Dictionary" command in the menu.

**type**

```
TRVALiveSpellAddEvent = procedure (Sender: TRVAPopupMenu(66);
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer) of object;
TRVA2LiveSpellAddEvent = procedure (Sender: TObject;
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer) of object;
```

**For TRVAPopupMenu:**

**property** OnLiveSpellAdd: TRVALiveSpellAddEvent;

**For TRVATBPopupMenu, TRVATBXPopupMenu, TRVASPTBXPopupMenu:**

**property** OnLiveSpellAdd: TRVA2LiveSpellAddEvent;

This event is used for live spelling check.

**Word** is a word for adding to the custom dictionary.

**StyleNo** is an index of text style for this word in **Edit.Style.TextStyles**.

If `SpellInterface`<sup>(54)</sup> is used, this command is processed automatically.

Otherwise, use this event for adding word to the user dictionary (this command appear in the menu only if this event is assigned).

#### 1.2.5.2.2 TRVAPopupMenu.OnLiveSpellGetSuggestions

Occurs when the menu needs to add items containing suggestions for the current misspelled word.

##### type

```
TRVALiveSpellGetSuggestionsEvent = procedure (Sender: TRVAPopupMenu(66);
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer; Suggestions: TRVASpellStrings(385)) of object;
TRVA2LiveSpellGetSuggestionsEvent = procedure (Sender: TObject;
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer; Suggestions: TRVASpellStrings(385)) of object;
```

##### For TRVAPopupMenu:

**property** OnLiveSpellGetSuggestions: TRVALiveSpellGetSuggestionsEvent

##### For TRVATBPopupMenu, TRVATBXPopupMenu, TRVASPTBXPopupMenu:

**property** OnLiveSpellGetSuggestions: TRVA2LiveSpellGetSuggestionsEvent

This event is used for live spelling check.

**Word** is a misspelled word.

**StyleNo** is an index of text style for this word in `Edit.Style.TextStyles`.

Add suggestions to **Suggestions**.

If `SpellInterface`<sup>(54)</sup> is used, this command is processed automatically.

Otherwise, use this event for generating suggestions.

##### See also:

- `MaxSuggestionsCount`<sup>(69)</sup>

#### 1.2.5.2.3 TRVAPopupMenu.OnLiveSpellIgnoreAll

Occurs when the user clicks "Ignore All" command in the menu.

##### type

```
TRVALiveSpellIgnoreAllEvent = procedure (Sender: TRVAPopupMenu(66);
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer) of object;
TRVA2LiveSpellIgnoreAllEvent = procedure (Sender: TObject;
  Edit: TCustomRichViewEdit; const Word: TRVUnicodeString;
  StyleNo: Integer) of object;
```

##### For TRVAPopupMenu:

**property** OnLiveSpellIgnoreAll: TRVALiveSpellIgnoreAllEvent;

##### For TRVATBPopupMenu, TRVATBXPopupMenu, TRVASPTBXPopupMenu:

**property** OnLiveSpellIgnoreAll: TRVA2LiveSpellIgnoreAllEvent;

This event is used for live spelling check.

**Word** is a word for adding to the ignore list.

**StyleNo** is an index of text style for this word in **Edit.Style.TextStyles**.

If SpellInterface<sup>(54)</sup> is used, this command is processed automatically.

Otherwise, use this event for adding word to the ignore list (this command appear in the menu only if this event is assigned).

#### 1.2.5.2.4 TRVAPopupMenu.OnLiveSpellWordReplace

Occurs before **Word** is changed to **Correction** in **Edit**.

##### type

```
TRVALiveSpellReplaceEvent = procedure (Sender: TRVAPopupMenu;
    Edit: TCustomRichViewEdit; const Word, Correction: TRVUnicodeString;
    StyleNo: Integer) of object;
TRVA2LiveSpellReplaceEvent = procedure (Sender: TObject;
    Edit: TCustomRichViewEdit; const Word, Correction: TRVUnicodeString;
    StyleNo: Integer) of object;
```

##### For TRVAPopupMenu:

**property** OnLiveSpellWordReplace: TRVALiveSpellReplaceEvent;

##### For TRVATBPopupMenu, TRVATBXPopupMenu, TRVASPTBXPopupMenu:

**property** OnLiveSpellWordReplace: TRVA2LiveSpellReplaceEvent;

This event is used for live spelling check. It occurs when the user clicks on one of suggestions for the misspelled word in the menu.

**StyleNo** is an index of text style for this word in **Edit.Style.TextStyles**.

Processing this event is not necessary. It may be used if a spellchecker can use a history of replacements to generate better suggestions in future.

### 1.2.6 TRVFontCharsetComboBox

TRVFontCharsetComboBox is a combo-box for choosing a font character set.

##### Unit RVFontCombos;

```
TRVFontCharsetComboBox = class (TCustomRVFontComboBox(33))
```

##### Hierarchy

(Hierarchy of TRVFontCharsetComboBox, if TNT Controls<sup>(370)</sup> are not used)

*TObject*

*TPersistent*

*TComponent*

*TControl*

*TWinControl*

*TCustomComboBox*

*TComboBox*

*TCustomRVFontComboBox*<sup>(33)</sup>

##### Description

This combo-box allows changing a font Charset for a selected text in Editor<sup>(35)</sup>.

This component works automatically: place it on a form, assign Editor<sup>(35)</sup> and ActionFont<sup>(34)</sup> properties; no additional code is required.

**Unicode note:** for Unicode text, Charset does not affect appearance (except for SYMBOL\_CHARSET).

In addition to character sets available for the current font, the combo-box may contain DEFAULT\_CHARSET item, if AddDefaultCharset<sup>(73)</sup>=*True*.

**See also:**

- TRVFontComboBox<sup>(73)</sup>
- TRVFontSizeComboBox<sup>(79)</sup>

### 1.2.6.1 Properties

#### In TRVFontCharsetComboBox

---

- AddDefaultCharset<sup>(73)</sup>
- DefaultCharsetCaption<sup>(73)</sup>  
FontName<sup>(73)</sup>

#### Derived from TCustomRVFontComboBox<sup>(33)</sup>

---

- ActionFont<sup>(34)</sup>
- Editor<sup>(35)</sup>

#### Derived from TComboBox

---

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint

- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

#### 1.2.6.1.1 TRVFontCharsetComboBox.AddDefaultCharset

Specifies whether the combo-box must contain DEFAULT\_CHARSET item.

**property** AddDefaultCharset: Boolean;

The caption for this item is defined in DefaultCharsetCaption<sup>(73)</sup> property.

**Default value:**

*False*

#### 1.2.6.1.2 TRVFontCharsetComboBox.DefaultCharsetCaption

Specifies the caption for DEFAULT\_CHARSET item.

**property** DefaultCharsetCaption: TRVLocString<sup>(348)</sup>;

This item is added if AddDefaultCharset<sup>(73)</sup>=*True*.

The recommended value for this property: RVA\_GetS<sup>(346)</sup>(rvam\_font\_DefaultCharset).

**Default value:**

'(Default)'

#### 1.2.6.1.3 TRVFontCharsetComboBox.FontName

Contains a font name.

**property** FontName: TRVLocString<sup>(348)</sup>;

This property may be used when the combo-box is not linked with Editor<sup>(35)</sup>. If linked, the combo-box items are maintained automatically.

When this property is assigned, the combo-box displays character sets available for this font.

### 1.2.7 TRVFontComboBox

TRVFontComboBox is a combo-box for choosing a font name.

**Unit** RVFontCombos;

TRVFontComboBox = **class** (TCustomRVFontComboBox<sup>(33)</sup>)

**Hierarchy**

(Hierarchy of TRVFontComboBox, if TNT Controls<sup>(370)</sup> are not used)

*TObject*

*TPersistent*

*TComponent*

*TControl*

*TWinControl*

*TCustomComboBox*

*TComboBox*

*TCustomRVFontComboBox* <sup>(33)</sup>

## Description

This combo-box allows changing a font name for a selected text in Editor <sup>(35)</sup>.

This component works automatically: place it on a form, assign Editor <sup>(35)</sup> and ActionFont <sup>(34)</sup> properties; no additional code is required.

The component displays preview of fonts in items. To disable this feature, assign Preview <sup>(76)</sup> = *False*.

### See also:

- TRVFontListBox <sup>(76)</sup>
- TRVFontSizeComboBox <sup>(79)</sup>
- TRVFontCharsetComboBox <sup>(71)</sup>

### 1.2.7.1 Properties

#### In TRVFontComboBox

- AutoCharset <sup>(75)</sup>
- DropDownWidth <sup>(75)</sup>
- ItemHeight <sup>(75)</sup>
- Preview <sup>(76)</sup>
- SymbolPreviewString <sup>(76)</sup>

#### Derived from TCustomRVFontComboBox <sup>(33)</sup>

- ActionFont <sup>(34)</sup>
- Editor <sup>(35)</sup>

#### Derived from TComboBox

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength

- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

#### 1.2.7.1.1 TRVFontComboBox.AutoCharset

Defines whether the combobox must change a text character set.

**property** AutoCharset: Boolean;

If *True*, the component changes not only font name, but also font' Charset. When applying a symbol font, the combo box applies SYMBOL\_CHARSET. Otherwise, it applies DEFAULT\_CHARSET.

**Default value:**

*True*

**See also:**

- TrvActionFontEx<sup>(149)</sup>.AutoApplySymbolCharset<sup>(151)</sup>

#### 1.2.7.1.2 TRVFontComboBox.DropDownWidth

Gets or sets the width of the of the drop-down portion of a combo box.

**property** DropDownWidth: TRVPixel96Length;

This property is used if Preview<sup>(76)</sup> = *True*.

This value is measured in pixels at 96 pixels per inch; the actual size scaled according to the current screen pixel density.

**Default value:**

300

#### 1.2.7.1.3 TRVFontComboBox.ItemHeight

Gets or sets the height of items in the drop-down portion of a combo box.

**property** ItemHeight: TRVPixel96Length;

This property is used if Preview<sup>(76)</sup> = *True*.

This value is measured in pixels at 96 pixels per inch; the actual size scaled according to the current screen pixel density.

**Default value:**

22

#### 1.2.7.1.4 TRVFontComboBox.Preview

Specifies whether the combo-box draws items using corresponding fonts.

**property** Preview: Boolean;

**Default value:**

*True*

**See also:**

- DropDownWidth<sup>(75)</sup>
- ItemHeight<sup>(75)</sup>
- SymbolPreviewString<sup>(76)</sup>

#### 1.2.7.1.5 TRVFontComboBox.SymbolPreviewString

Specifies a string to display after the font name for symbol fonts.

**property** SymbolPreviewString: TRVALocString<sup>(348)</sup>;

This property is used if Preview<sup>(76)</sup> = *True*.

For fonts of SYMBOL\_CHARSET, their font name is drawn using Font.Name, and a preview is drawn next to the font name. This property is used to draw this preview.

**Default value:**

'Abc'

### 1.2.8 TRVFontListBox

TRVFontListBox is a list-box for choosing a font name.

**Unit** RVFontCombos;

TRVFontListBox = **class** (TCustomRVFontListBox<sup>(35)</sup>)

**Hierarchy**

(Hierarchy of TRVFontListBox, if TNT Controls<sup>(370)</sup> are not used)

*TObject*

*TPersistent*

*TComponent*

*TControl*

*TWinControl*

*TCustomListBox*

*TListBox*

*TCustomRVFontListBox*<sup>(35)</sup>

#### Description

This list-box allows changing a font name for a selected text in Editor<sup>(37)</sup>.

This component works automatically: place it on a form, assign Editor<sup>(37)</sup> and ActionFont<sup>(36)</sup> properties; no additional code is required.

The component displays preview of fonts in items. To disable this feature, assign Preview<sup>(78)</sup> = *False*.



**See also:**

- TRVFontComboBox <sup>73</sup>

## 1.2.8.1 Properties

### In TRVFontComboBox

---

- AutoCharSet <sup>78</sup>
- ItemHeight <sup>78</sup>
- Preview <sup>78</sup>
- SymbolPreviewString <sup>78</sup>

### Derived from TCustomRVFontListBox <sup>35</sup>

---

- ActionFont <sup>36</sup>
- Editor <sup>37</sup>

### Derived from TComboBox

---

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

#### 1.2.8.1.1 TRVFontListBox.AutoCharset

Defines whether the listbox must change a text character set.

**property** AutoCharset: Boolean;

If *True*, the component changes not only font name, but also font' Charset. When applying a symbol font, the combo box applies SYMBOL\_CHARSET. Otherwise, it applies DEFAULT\_CHARSET.

**Default value:**

*True*

**See also:**

- TrvActionFontEx<sup>(149)</sup>.AutoApplySymbolCharset<sup>(151)</sup>

#### 1.2.8.1.2 TRVFontListBox.ItemHeight

Gets or sets the height of items of the list box.

**property** ItemHeight: TRVPixel96Length;

This property is used if Preview<sup>(78)</sup> = *True*.

This value is measured in pixels at 96 pixels per inch; the actual size scaled according to the current screen pixel density.

**Default value:**

22

#### 1.2.8.1.3 TRVFontListBox.Preview

Specifies whether the list-box draws items using corresponding fonts.

**property** Preview: Boolean;

**Default value:**

*True*

**See also:**

- ItemHeight<sup>(78)</sup>
- SymbolPreviewString<sup>(78)</sup>

#### 1.2.8.1.4 TRVFontListBox.SymbolPreviewString

Specifies a string to display after the font name for symbol fonts.

**property** SymbolPreviewString: TRVLocString<sup>(348)</sup>;

This property is used if Preview<sup>(78)</sup> = *True*.

For fonts of SYMBOL\_CHARSET, their font name is drawn using Font.Name, and a preview is drawn next to the font name. This property is used to draw this preview.

**Default value:**

'Abc'

## 1.2.9 TRVFontSizeComboBox

TRVFontSizeComboBox is a combo-box for choosing a font size (with the precision up to 0.5 point).

**Unit** RVFontCombos;

```
TRVFontSizeComboBox = class (TCustomRVFontComboBox33)
```

### Hierarchy

(Hierarchy of TRVFontSizeComboBox, if TNT Controls<sup>370</sup> are not used)

*TObject*

*TPersistent*

*TComponent*

*TControl*

*TWinControl*

*TCustomComboBox*

*TComboBox*

*TCustomRVFontComboBox*<sup>33</sup>

### Description

This combo-box allows changing a font size for a selected text in Editor<sup>35</sup>.

This component works automatically: place it on a form, assign Editor<sup>35</sup> and ActionFont<sup>34</sup> properties; no additional code is required.

#### See also:

- TRVFontComboBox<sup>73</sup>
- TRVFontCharsetComboBox<sup>71</sup>

## 1.2.9.1 Properties

### In TRVFontSizeComboBox

FontName<sup>80</sup>

### Derived from TCustomRVFontComboBox<sup>33</sup>

- ActionFont<sup>34</sup>
- Editor<sup>35</sup>

### Derived from TComboBox

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled

- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- MaxLength
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted
- Style
- TabOrder
- TabStop
- Text
- Visible

#### 1.2.9.1.1 TRVFontSizeComboBox.FontName

Contains a font name.

```
property FontName: TFontName;
```

This property may be used when the combo-box is not linked with Editor<sup>35</sup>. If linked, the combo-box items are maintained automatically.

When this property is assigned, the combo-box displays font sizes available for this font. For True Type and Open Type fonts, it displays a predefined set of font sizes (8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 36, 48, 72).

### 1.2.10 TRVRuler

**TRVRuler** is a ruler that works with TRichViewEdit component.

**Unit** RVRuler;

```
TRVRuler = class (TRuler)
```

#### Hierarchy

```
TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRuler
TRuler
```

## Description

---

This component has many properties customizing its appearance and behavior. In this manual, we describe only the most important of them.

To associate the ruler with the editor, assign `TRichViewEdit` or `TDBRichViewEdit` control to `RichViewEdit`<sup>(86)</sup> property. The ruler can change the following parts in the editor:

- `LeftMargin` and `RightMargin`;
- indents of paragraphs;
- levels of paragraph lists;
- widths of table columns;
- tab stops in paragraphs (see `TRVRulerItemSelector`<sup>(37)</sup> component).

If `rvoClientTextWidth` is not included in `RichViewEdit`<sup>(86)</sup>.Options, the ruler assigns `RichViewEdit`<sup>(86)</sup>.`MinTextWidth` and `MaxTextWidth` according to its `PageWidth` property.

To set the ruler margins and page sizes according to `RichViewEdit`<sup>(86)</sup> settings, call `UpdateRulerMargins`<sup>(87)</sup>. If you use `RichViewActions`, it's enough to call it in `TRVAControlPanel`<sup>(38)</sup>.`OnMarginsChanged`<sup>(65)</sup> event.

To change the ruler's hints according to `TRVAControlPanel`<sup>(38)</sup>.`Language`<sup>(51)</sup>, call `RVALocalizeRuler`<sup>(351)</sup>.

## Acknowledgement

---

Thanks to Pieter Zijlstra, who created the first version of `TRVRuler`.

Thanks to Rivarez, who implemented skin modes<sup>(85)</sup> for the rulers.

### 1.2.10.1 Properties

#### In `TRVRuler`

---

- `LineStyle`
- `RichViewEdit`<sup>(86)</sup>
- `RVRulerOptions`

#### Derived from `TCustomRuler`

---

- `BiDiModeRuler`<sup>(82)</sup>
- `BottomMargin`
- `DefaultTabWidth`
- `FirstIndent`
- `Flat`<sup>(83)</sup>
- `IndentColor`<sup>(83)</sup>
- `IndentSaturation`<sup>(84)</sup>
- `IndentSettings`
- `Inset`
- `LeftIndent`
- `LeftMargin`
- `MarginColor`<sup>(83)</sup>
- `MarginSaturation`<sup>(84)</sup>

- MarginSettings
- MaxTabs
- Options
- TickColor <sup>83</sup>
- SkinType <sup>85</sup>
- RulerSaturation <sup>84</sup>
- RightIndent
- RightMargin
- RulerColor <sup>83</sup>
- RulerColorPageEnd <sup>83</sup>
- RulerTexts
- RulerType <sup>84</sup>
- ScreenRes <sup>84</sup>
- Tabs
- TabSettings
- TopMargin
- UnitsDisplay <sup>85</sup>
- UnitsPixelsPerInch <sup>85</sup>
- UnitsProgram <sup>85</sup>
- Zoom <sup>86</sup>

## Derived from TCustomControl ---

- Align
- Anchors
- BiDiMode
- Color
- Constraints
- DragCursor
- DragKind
- DragMode
- Enabled
- Font
- Hint
- ParentBackground
- ParentBiDiMode
- ParentColor
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Visible

### 1.2.10.1.1 TCustomRuler.BiDiModeRuler

Allows forcing right-to-left or left-to-right ruler mode.

#### **type**

```
TBiDiModeRuler = (bmUseBiDiMode, bmLeftToRight, bmRightToLeft);
```

**property** BiDiModeRuler: TBiDiModeRuler;

Value	Meaning
<i>bmUseBiDiMode</i>	Left-to-right or right-to-left, depending in BiDiMode property
<i>bmLeftToRight</i>	Left-to-right
<i>bmRightToLeft</i>	Right-to-left

**Default value***bmUseBiDiMode***1.2.10.1.2 TCustomRuler.Flat**

Switches between flat and 3d ruler appearance.

**property** Flat: Boolean;

This property works only if SkinType<sup>(85)</sup> = *stNoSkin*.

If you assign *False* to **Flat**, SkinType<sup>(85)</sup> is changed to *stNoSkin*.

**Default value***False***1.2.10.1.3 TCustomRuler.IndentColor, MarginColor, RulerColor, TickColor, RulerColorPageEnd**

Colors of the ruler's visual elements.

**property** RulerColor: TColor;

**property** IndentColor: TColor;

**property** MarginColor: TColor;

**property** TickColor: TColor;

**property** RulerColorPageEnd: TColor;

**RulerColor** is used to paint an internal ruler area.

**IndentColor** is used to paint handles that change paragraph indents.

**MarginColor** is used to paint left and right margin areas (or top and bottom margin areas for a vertical ruler).

**TickColor** is used to draw ticks and tab stops.

**RulerColorPageEnd** is used to paint the area to the right of the editor width (or below the editor height for a vertical ruler); only if SkinType<sup>(85)</sup> = *stNoSkin*.

Color is used to paint outside areas.

**Default values:**

- RulerColor: *clWindow*;
- IndentColor: *clBtnFace*;
- MarginColor: *clInfoBk*;
- TickColor: *clWindowText*;
- RulerColorPageEnd: *clBtnFace*;

**See also:**

- IndentSaturation, MarginSaturation, RulerSaturation<sup>(84)</sup>

**1.2.10.1.4 TCustomRuler.IndentSaturation, MarginSaturation, RulerSaturation**

Saturation values of the ruler's visual elements in skin modes.

```
property RulerSaturation: Integer;
property IndentSaturation: Integer;
property MarginSaturation: Integer;
```

These properties are applied if SkinType<sup>(85)</sup> <> stNoSkin.

**RulerSaturation** and RulerColor<sup>(83)</sup> are applied to an internal ruler area.

**IndentSaturation** and IndentColor<sup>(83)</sup> are applied to handles that are used to change paragraph indents.

**MarginSaturation** and MarginColor<sup>(83)</sup> are applied to left and right margin areas (or top and bottom margin areas for a vertical ruler).

**Default value**

250

**1.2.10.1.5 TCustomRuler.RulerType**

Switches between horizontal and vertical ruler.

```
type
  TRulerType = (rtHorizontal, rtVertical);
property RulerType: TRulerType;
```

For TRichView, only a horizontal ruler makes sense.

For ScaleRichView, both horizontal and vertical rulers are useful.

**Default value**

rtHorizontal

**1.2.10.1.6 TCustomRuler.ScreenRes**

The number of screen pixels that make up a logical inch.

```
property ScreenRes: Integer;
```

By default, this property is equal to the screen DPI (or DPI of a monitor that contains a form containing this ruler, if the application supports it (in Delphi 10.1+)).

Moreover, the component resets this property to the screen DPI (as described above) after loading from a form and on resizing. So, if you want to change value of this property, you need to reassign it in these cases.

However, it's highly **not recommended to change value of this property**. Allow the ruler to maintain it automatically. Change Zoom<sup>(86)</sup> instead.

If you changed a value of the global RichViewPixelsPerInch variable (not recommended!), you need to update this property accordingly:

```
Ruler1.ScreenRes := RichViewPixelsPerInch;
```



This value affects not only sizes that this ruler measures, but also sizes of its visual elements (such as tab icons and thumbs), they are scaled from UnitsPixelsPerInch<sup>(85)</sup> to **ScreenRes**.




#### 1.2.10.1.7 TCustomRuler.SkinType

Changes the ruler appearance.

**type**

```
TSkinType = (stNoSkin, stSkin1, stSkin2);
```

**property** SkinType: TSkinType;

Value	Appearance
<i>stNoSkin</i>	
<i>stSkin1</i>	
<i>stSkin2</i>	

**Default value**

*stNoSkin*

**See also**

- RulerSaturation, IndentSaturation, MarginSaturation<sup>(84)</sup>
- Flat<sup>(83)</sup>

#### 1.2.10.1.8 TCustomRuler.UnitsDisplay, UnitsProgram

Measure units.

**property** UnitsDisplay: TRulerUnits<sup>(384)</sup>;

**property** UnitsProgram: TRulerUnits<sup>(384)</sup>;

**UnitsDisplay** defines units that the ruler displays. It's recommended to assign the value corresponding to TRVAControlPanel<sup>(38)</sup>.UnitsDisplay<sup>(55)</sup>.

**UnitsProgram** is used internally in the ruler (for properties of TRulerFloat type), it rarely need to be changed.

**Default value**

ruCentimeters

#### 1.2.10.1.9 TCustomRuler.UnitsPixelsPerInch

The number of pixels that make up a logical inch if ScreenRes<sup>(84)</sup> = 96 and Zoom<sup>(86)</sup> = 100%.

**property** UnitsPixelsPerInch: Integer;

This value must be equal to Style.UnitsPixelsPerInch of TRichViewEdit that works with this event.

It's highly **not recommended to change value of this property** (as well as RichViewEdit.Style.PixelsPerInch). Change Zoom<sup>(86)</sup> instead.

This value affects not only sizes that this ruler measures, but also sizes of its visual elements (such as tab icons and thumbs), they are scaled from **UnitsPixelsPerInch** to ScreenRes<sup>(84)</sup>.

#### Default value

96

#### 1.2.10.1.10 TCustomRuler.Zoom

This property must be equal to zooming in the editor that works with this ruler, percent.

##### type

```
TZoomRange = 1 .. 10000; // 1% - 10000%
```

**property** Zoom: TZoomRange;

By default, if you did not change any “pixels per inch” properties of TRichViewEdit and this ruler, then a document is not scaled, and **Zoom** = 100 (meaning 100%).

However, if you changed TRichViewEdit.DocumentPixelsPerInch, you must change **Zoom** accordingly:

```
RVRuler1.Zoom :=  
Round(100*RichViewEdit1.GetRealDocumentPixelsPerInch /  
RVRuler1.ScreenRes(84));
```

An alternative solution would be assigning RVRuler1.ScreenRes<sup>(84)</sup> = RichViewEdit1.GetRealDocumentPixelsPerInch. However, it is not recommended, because:

- it scales all the ruler visual elements too;
- ScreenRes is reset when the ruler is loaded from DFM and when it is resized.

#### Default value

100

#### 1.2.10.1.11 TRVRuler.RichViewEdit

Specifies the editor for this ruler.

**property** RichViewEdit: TCustomRichViewEdit;

The ruler works differently depending on RichViewEdit.Options.

If *rvoClientTextWidth* is included in **RichViewEdit.Options**, the ruler assumes that a document width is equal to **RichViewEdit.ClientWidth**.

If *rvoClientTextWidth* is excluded from **RichViewEdit.Options**, when the user resizes a margin, the ruler assigns RichViewEdit.MinTextWidth and MaxTextWidth according to its PageWidth property.

### 1.2.10.2 Methods

#### In TRVRuler

UpdateRulerMargins<sup>(87)</sup>

### 1.2.10.2.1 TRVRuler.UpdateRulerMargins

Change the ruler margins according to RichViewEdit<sup>(86)</sup>'s margins.

**procedure** UpdateRulerMargins;

If RulerType<sup>(84)</sup> = *rtHorizontal*, this procedure sets the ruler's LeftMargin, RightMargin, PageWidth according to RichViewEdit<sup>(86)</sup>.

If RulerType<sup>(84)</sup> = *rtVertical*, this procedure sets the ruler's TopMargin, BottomMargin, PageHeight according to RichViewEdit<sup>(86)</sup>.

If you use TRVRuler without RichViewActions, call this method:

- when the ruler is associated with the editor;
- any time when you call RichViewEdit<sup>(86)</sup>.Format or Reformat.

If you use TRVRuler with RichViewActions, call this method in TRVAControlPanel<sup>(38)</sup>.OnMarginsChanged<sup>(65)</sup> event.

If RichViewEdit<sup>(86)</sup> is TDBRichViewEdit, also call this method in DataSet.AfterScroll event

## 1.2.11 TRVStyleTemplateComboBox, TRVStyleTemplateListBox

TRVStyleTemplateComboBox is a combo-box for choosing a style template.

TRVStyleTemplateListBox is a list-box for choosing a style template.

**Unit** RVStyleFuncs;

**Syntax, if TNT Controls<sup>(370)</sup> are not used:**

TRVStyleTemplateComboBox = **class** (TCustomComboBox)

TRVStyleTemplateListBox = **class** (TCustomListBox)

**Syntax, if TNT Controls<sup>(370)</sup> are used:**

TRVStyleTemplateComboBox = **class** (TTntCustomComboBox)

TRVStyleTemplateListBox = **class** (TTntCustomListBox)

### Hierarchy

**(Hierarchy of TRVStyleTemplateComboBox/TRVStyleTemplateListBox, if TNT Controls<sup>(370)</sup> are not used:)**

*TObject*

*TPersistent*

*TComponent*

*TControl*

*TWinControl*

*TCustomComboBox*

*TCustomComboBox/TCustomListBox*

### Description

These component allow applying style templates to Editor<sup>(89)</sup>. The components require Editor<sup>(89)</sup>.UseStyleTemplates=*True*, otherwise their Items are empty.

The components work automatically: place them on a form, assign Editor<sup>(35)</sup> property; no additional code is required, unless you want to process "All Styles" command in OnClickAllStyles<sup>(92)</sup> event.

By default, the components show only style templates in use, "quick access" style templates, and heading<sup>(90)</sup> style templates. You can display all style templates by assigning ShowAllStyles<sup>(90)</sup>=True.

If ShowClearFormat<sup>(90)</sup>=True, the components display "Clear Format" command.

You can display icons in items by assigning Images<sup>(90)</sup> and image indices<sup>(89)</sup>.

When the application language is changed, call Localize<sup>(91)</sup>.

### 1.2.11.1 Properties

#### In TRVStyleTemplateCombo/ListBox

- AllStylesImageIndex<sup>(89)</sup>
- ClearFormatImageIndex<sup>(89)</sup>
- ControlPanel<sup>(89)</sup>
- Editor<sup>(89)</sup>
- Images<sup>(90)</sup>
- ParaStyleImageIndex<sup>(89)</sup>
- ParaTextStyleImageIndex<sup>(89)</sup>
- ShowAllStyles<sup>(90)</sup>
- ShowClearFormat<sup>(90)</sup>
- SmartHeadings<sup>(90)</sup>
- TextStyleImageIndex<sup>(89)</sup>

#### Derived from TCustomCombo/ListBox

- Anchors
- BiDiMode
- Color
- Constraints
- Ctl3D
- DragCursor
- DragKind
- DragMode
- DropDownCount
- Enabled
- Font
- ImeMode
- ImeName
- ItemHeight
- Items
- ParentBiDiMode
- ParentColor
- ParentCtl3D
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- Sorted

- TabOrder
- TabStop
- Visible

#### 1.2.11.1.1 TRVStyleTemplateComboBox.\*ImageIndex

The properties contain indexes in Images<sup>(90)</sup>.

```
property AllStylesImageIndex: Integer;
property ClearFormatImageIndex: Integer;
property TextStyleImageIndex: Integer;
property ParaStyleImageIndex: Integer;
property ParaTextStyleImageIndex: Integer;
```

**ClearFormatImageIndex** – index of an image for "Clear Format" item, shown if ShowClearFormat<sup>(90)</sup>=*True*.

**AllStylesImageIndex** – index of an image for "All Styles" item, shown if ShowAllStyles<sup>(90)</sup>=*False*, and not all styles are displayed.

**TextStyleImageIndex** – index of an image for style templates having Kind=*rvstkText*

**ParaStyleImageIndex** – index of an image for style templates having Kind=*rvstkPara*

**ParaTextStyleImageIndex** – index of an image for style templates having Kind=*rvstkParaText*

**Default value:**

-1

#### 1.2.11.1.2 TRVStyleTemplateComboBox.ControlPanel

Links this component with a control panel.

```
property ControlPanel: TComponent;
```

You can assign a TRVAControlPanel<sup>(38)</sup> component to this property. If this property is not assigned (*nil*), the component uses MainRVAControlPanel<sup>(388)</sup>.

**ControlPanel** is used to display localized captions according to its Language<sup>(51)</sup>.

#### 1.2.11.1.3 TRVStyleTemplateComboBox.Editor

Links this combo-box with an editor.

```
property Editor: TCustomRVControl;
```

Controls of the following types can be assigned to this property:

- TRichViewEdit
- TDBRichViewEdit
- TSRichViewEdit (ScaleRichView)
- TDBSRichViewEdit (ScaleRichView)

When the combo-box is linked with **Editor**:

- the combo-box is updated automatically according to changes in **Editor**
- the combo-box applies the user choice to the selected fragment in **Editor**

#### 1.2.11.1.4 TRVStyleTemplateComboBox.Images

A link to TImageList component containing images for items.

**property** Images: TCustomImageList;

**See also:**

- AllStylesImageIndex, ClearFormatImageIndex, ParaStyleImageIndex, ParaTextStyleImageIndex, TextStyleImageIndex<sup>(89)</sup>

#### 1.2.11.1.5 TRVStyleTemplateComboBox.ShowAllStyles

Specifies whether the component displays all available style templates.

**property** ShowAllStyles: Boolean;

If *True*, the component displays all style templates from Editor<sup>(89)</sup>.Style.StyleTemplates.

If *False*, the component displays:

- style templates in use (i.e. style templates referred by Editor<sup>(89)</sup>.Style.TextStyles and ParaStyles);
- style templates having QuickAccess=*True*
- "All styles" item (if not all style templates are displayed)

If SmartHeadings<sup>(90)</sup>=*True*, the component uses a special procedure to calculate a visibility of "heading 1".."heading 9" style templates.

When "All Styles" item is clicked:

- if OnClickAllStyles<sup>(92)</sup> is assigned, it is called
- otherwise, the combo-/list-box temporarily displays all style templates.

**Default value:**

*False*

**See also:**

- AllStylesImageIndex<sup>(89)</sup>

#### 1.2.11.1.6 TRVStyleTemplateComboBox.ShowClearFormat

Shows/hides "Clear Format" item.

**property** ShowClearFormat: Boolean;

If *True*, "Clear Format" is added as the first item.

**Default value:**

*True*

**See also:**

- ClearFormatImageIndex<sup>(89)</sup>

#### 1.2.11.1.7 TRVStyleTemplateComboBox.SmartHeadings

Specifies whether a special procedure should be used to calculate a visibility of heading style templates.

**property** SmartHeadings: Boolean;

If ShowAllStyles<sup>(90)</sup>=*False* and **SmartHeadings**=*True*, the component uses the following procedure to display heading style templates ("heading 1".."heading 9"):

- if headings are not used, or only "heading 1" is used, the component displays headings 1, 2, and 3
- otherwise, if headings up to level N are used (inclusive), the component displays headings from 1 to N+1

**Default value:**

*True*

### 1.2.11.2 Methods

#### In TRVStyleTemplateCombo/ListBox

---

- Localize<sup>91</sup>

#### Derived from TCustomCombo/ListBox

---

...

#### 1.2.11.2.1 TRVStyleTemplateComboBox.Localize

Rebuilds the content of the component.

**procedure** Localize;

Call this method when Language<sup>51</sup> of ControlPanel<sup>89</sup> is changed.

### 1.2.11.3 Events

#### In TRVStyleTemplateCombo/ListBox

---

- OnClickAllStyles<sup>92</sup>

#### Derived from TCustomCombo/ListBox

---

- OnChange
- OnClick
- OnContextPopup
- OnDbClick
- OnDragDrop
- OnDragOver
- OnDropDown (combo-box only)
- OnEndDock
- OnEndDrag
- OnEnter
- OnExit
- OnKeyDown
- OnKeyPress
- OnKeyUp
- OnStartDock
- OnStartDrag

### 1.2.11.3.1 TRVStyleTemplateComboBox.OnClickAllStyles

Occurs when "All Styles" item is clicked.

**property** OnClickAllStyles: TNotifyEvent;

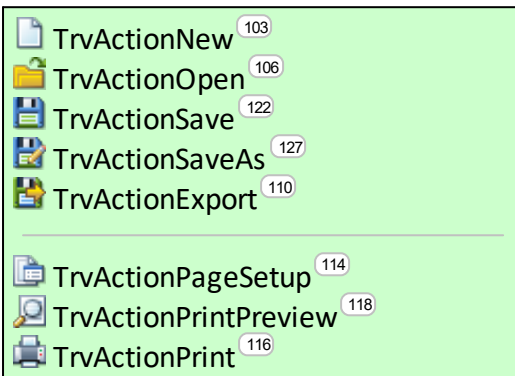
See ShowAllStyles<sup>(90)</sup> for details.

## 1.3 Actions

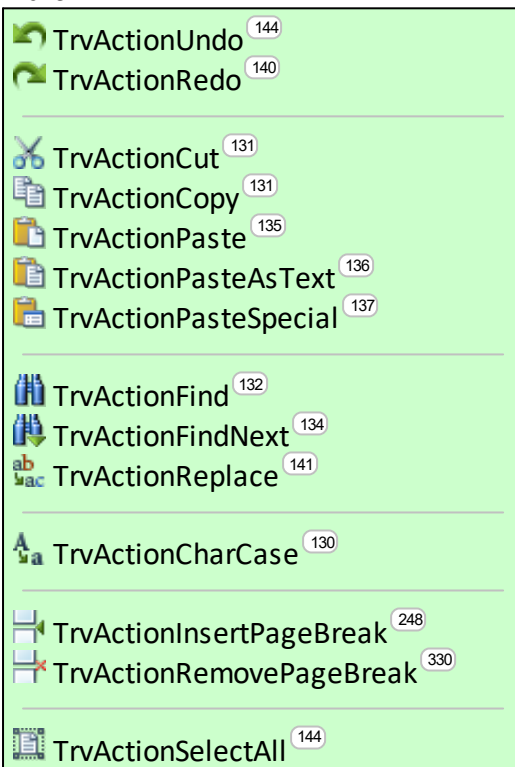
### Actions

The actions in the order as they appear in the menu of the ActionTest demo:

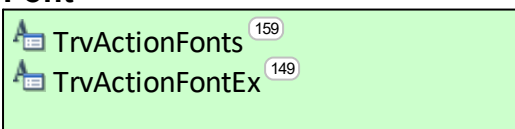
#### File



#### Edit



#### Font





**B** TrvActionFontBold <sup>147</sup>  
*I* TrvActionFontItalic <sup>158</sup>  
U TrvActionFontUnderline <sup>168</sup>  
~~S~~ TrvActionFontStrikeout <sup>166</sup>

**AB** TrvActionFontAllCaps <sup>146</sup>  
<sup>○</sup> TrvActionFontOverline <sup>159</sup>

<sub>f</sub> TrvActionSubscript <sup>169</sup>  
<sup>f</sup> TrvActionSuperscript <sup>170</sup>

↱ TrvActionFontShrink <sup>161</sup>  
↰ TrvActionFontGrow <sup>155</sup>

↱ TrvActionFontShrinkOnePoint <sup>164</sup>  
↰ TrvActionFontGrowOnePoint <sup>157</sup>

🎨 TrvActionFontColor <sup>148</sup>  
🖌 TrvActionFontBackColor <sup>147</sup>

## Paragraph

📄 TrvActionParagraph <sup>198</sup>  
📄 TrvActionParaBorder <sup>191</sup>

☰ TrvActionAlignLeft <sup>178</sup>  
☰ TrvActionAlignCenter <sup>174</sup>  
☰ TrvActionAlignRight <sup>179</sup>  
☰ TrvActionAlignJustify <sup>177</sup>  
☰ TrvActionAlignDistribute <sup>176</sup>



☰ TrvActionParaList <sup>206</sup>  
☰ TrvActionParaBullets <sup>195</sup>  
☰ TrvActionParaNumbering <sup>209</sup>

☰ TrvActionWordWrap <sup>211</sup>


☰ TrvActionIndentInc <sup>187</sup>  
☰ TrvActionIndentDec <sup>186</sup>

☰ TrvActionLineSpacing100 <sup>189</sup>  
☰ TrvActionLineSpacing150 <sup>189</sup>  
☰ TrvActionLineSpacing200 <sup>190</sup>






☰ TrvActionClearLeft <sup>180</sup>  
☰ TrvActionClearRight <sup>181</sup>

-  TrvActionClearBoth 180
-  TrvActionClearNone 181



---

-  TrvActionParaColor 195


## Format

-  TrvActionStyleTemplates 212
-  TrvActionAddStyleTemplate 215
-  TrvActionClearFormat 219
-  TrvActionClearTextFormat 219
-  TrvActionStyleInspector 216



---

-  TrvActionBackground 313
-  TrvActionColor 315




---

-  TrvActionFillColor 322












---

-  TrvActionHide 324
-  TrvActionRemoveHyperlinks 329





---

-  TrvActionItemProperties 324
-  TrvActionInsertCaption 223
-  TrvActionVAlign 333




## Insert































-  TrvActionInsertFile 228
-  TrvActionInsertPicture 249
-  TrvActionInsertHLine 231
-  TrvActionInsertHyperlink 232
-  TrvActionBookmarks 221
-  TrvActionInsertSymbol 254
-  TrvActionInsertEquation 226 \*
-  TrvActionInsertNumber 245
-  TrvActionInsertTextBox 269
-  TrvActionInsertPageNumber 249
-  TrvActionInsertPageCount 248

---



-  TrvActionInsertFootnote 264
-  TrvActionInsertEndnote 263
-  TrvActionInsertSidenote 267
-  TrvActionEditNote 260

## Table

-  TrvActionInsertTable 271
-  TrvActionTableInsertColLeft 297
-  TrvActionTableInsertColRight 297

	TrvActionTableInsertRowsAbove	298
	TrvActionTableInsertRowsBelow	299
<hr/>		
	TrvActionTableDeleteCols	295
	TrvActionTableDeleteRows	295
	TrvActionTableDeleteTable	296
	TrvActionTableToText	309
<hr/>		
	TrvActionTableSelectTable	306
	TrvActionTableSelectCols	305
	TrvActionTableSelectRows	306
	TrvActionTableSelectCell	305
<hr/>		
	TrvActionTableCellVAlignTop	294
	TrvActionTableCellVAlignMiddle	293
	TrvActionTableCellVAlignBottom	292
	TrvActionTableCellVAlignDefault	293
<hr/>		
	TrvActionTableCellRotationNone	288
	TrvActionTableCellRotation90	289
	TrvActionTableCellRotation180	289
	TrvActionTableCellRotation270	290
<hr/>		
	TrvActionTableCellLeftBorder	285
	TrvActionTableCellTopBorder	291
	TrvActionTableCellRightBorder	287
	TrvActionTableCellBottomBorder	284
	TrvActionTableCellAllBorders	283
	TrvActionTableCellNoBorders	286
<hr/>		
	TrvActionTableSplit	308
	TrvActionTableSplitCells	308
	TrvActionTableMergeCells	300
<hr/>		
	TrvActionTableGrid	296
<hr/>		
	TrvActionTableSort	307
	TrvActionTableProperties	301

### **Toolbar only**

	TrvActionQuickPrint	120
	TrvActionShowSpecialCharacters	331

### Not used in ActionTest

TrvActionTextLTR	171
TrvActionTextRTL	172
<hr/>	
TrvActionParaLTR	208
TrvActionParaRTL	210
<hr/>	
TrvActionSpellingCheck	310
TrvActionThesaurus	310

### Customized

TrvActionEvent	321
TrvActionInsertText	256
TrvActionUserDefinedColor	

\* the action is not included in RichViewActions directly, it is included in a separate package.

## 1.3.1 File

### File Actions

This group of actions creates a new document, saves, opens, prints it.

TrvActionNew	103
TrvActionOpen	106
TrvActionSave	122
TrvActionSaveAs	127
TrvActionExport	110
<hr/>	
TrvActionPageSetup	114
TrvActionPrintPreview	118
TrvActionPrint	116

#### 1.3.1.1 TrvActionCustomFileIO

TrvActionCustomFileIO is a base class for "Save As", "Export" and "Insert File" actions.

**Unit** RichViewActions<sup>92</sup>;

#### Syntax

```
TrvActionCustomFileIO = class (TrvActionCustomIO98)
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction

```

*TrvCustomAction* <sup>(333)</sup>

*TrvAction* <sup>(311)</sup>

*TrvActionCustomIO* <sup>(98)</sup>

## Description

This action is not used directly.

## Descendants:

- *TrvActionSaveAs* <sup>(127)</sup>
- *TrvActionExport* <sup>(110)</sup>
- *TrvActionInsertFile* <sup>(228)</sup>

## 1.3.1.1.1 Properties

### In *TrvActionCustomFileIO*

■ *CustomFilter* <sup>(98)</sup>

### Derived from *TrvActionCustomIO* <sup>(98)</sup>

■ *AutoUpdateFileName* <sup>(99)</sup>

■ *DialogTitle* <sup>(100)</sup>

■ *FileName* <sup>(100)</sup>

■ *InitialDir* <sup>(100)</sup>

### Derived from *TrvAction* <sup>(311)</sup>

■ *Control* <sup>(312)</sup>

### Derived from *TrvCustomAction* <sup>(333)</sup>

■ *Caption* <sup>(334)</sup>

■ *ControlPanel* <sup>(335)</sup>

■ *Disabled* <sup>(335)</sup>

■ *Hint* <sup>(335)</sup>

### Derived from *TAction*

■ *AutoCheck*

■ *Caption*

■ *Checked*

*DisableIfNoHandler*

■ *Enabled*

■ *GroupIndex*

■ *HelpContext*

■ *HelpKeyword*

■ *HelpType*

■ *Hint*

■ *ImageIndex*

■ *Name*

■ *SecondaryShortCuts*

■ *ShortCut*

■ Visible

#### 1.3.1.1.1 TrvActionCustomFileIO.CustomFilter

Specifies the file filter for custom file formats.

**property** CustomFilter: TRVALocString<sup>348</sup>;

This property allows opening, saving and inserting files in custom formats (in addition to the standard formats provided by RichViewActions).

Value of this property is appended to the Filter property of the file opening/saving dialog, if the action supports custom formats.

##### Default value:

" (empty string)

##### See also:

- TRVActionSaveAs.Filter<sup>129</sup>
- TRVActionExport.Filter<sup>113</sup>
- TRVActionInsertFile.Filter<sup>230</sup>
- TrvActionOpen.CustomFilter<sup>108</sup>
- TRVAControlPanel.OnCustomFileOperation<sup>62</sup>

#### 1.3.1.2 TrvActionCustomIO

TrvActionCustomIO is a base class for the actions displaying dialogs for opening or saving files.

**Unit** RichViewActions<sup>92</sup>;

##### Syntax

TrvActionCustomIO = **class** (TrvAction<sup>311</sup>)

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>

##### Description

This action is not used directly.

Direct descendants:

- TrvActionCustomFileIO<sup>96</sup>
- TrvActionInsertPicture<sup>249</sup>

### 1.3.1.2.1 Properties

#### In TrvActionCustomIO

- AutoUpdateFileName<sup>(99)</sup>
- DialogTitle<sup>(100)</sup>
- FileName<sup>(100)</sup>
- InitialDir<sup>(100)</sup>

#### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.2.1.1 TrvActionCustomIO.AutoUpdateFileName

Specifies whether FileName<sup>(100)</sup> is automatically updated by the action.

**property** AutoUpdateFileName: Boolean;

If this property is *True*, FileName<sup>(100)</sup> property is updated when the action is successfully executed (the user chose a new file from dialog).

This property is used and published in the following actions:

- TrvActionExport<sup>(110)</sup>
- TrvActionInsertFile<sup>(228)</sup>

**Default value:**

*True*

#### 1.3.1.2.1.2 TrvActionCustomIO.DialogTitle

Defines a custom title (caption) for the file dialog.

**property** DialogTitle: TRVLocString<sup>348</sup>;

If this property is empty, default (localized) dialog titles are used.

**Default value:**

" (empty string)

**See also:**

- TrvActionOpen.DialogTitle<sup>108</sup>

#### 1.3.1.2.1.3 TrvActionCustomIO.FileName

Specifies the default file name.

**property** FileName: TRVLocString<sup>348</sup>;

This property is assigned to FileName property of the file open/save dialog before showing it.

This property is used and public in the following actions:

- TrvActionExport<sup>110</sup>
- TrvActionInsertFile<sup>228</sup>
- TrvActionInsertPicture<sup>249</sup>

**Default value:**

" (empty string)

**See also properties:**

- InitialDir<sup>100</sup>
- AutoUpdateFileName<sup>99</sup>

#### 1.3.1.2.1.4 TrvActionCustomIO.InitialDir

Specifies the initial directory for the file dialog.

**property** InitialDir: TRVLocString<sup>348</sup>;

This property is used only if FileName<sup>100</sup> is empty.

It is assigned to InitialDir property of the file open/save dialog before showing it.

**Default value:**

" (empty string)



### 1.3.1.3 TrvActionCustomNew

TrvActionCustomNew is a base class for "New" and "Open" actions.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionCustomNew = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action is not used directly.

#### Descendants:

- TrvActionNew<sup>(103)</sup>
- TrvActionOpen<sup>(106)</sup>

### 1.3.1.3.1 Events

#### In TrvActionCustomNew

■ OnNew<sup>(101)</sup>

#### 1.3.1.3.1.1 TrvActionCustomNew.OnNew

Occurs when the action is executed.

**property** OnNew: TNotifyEvent;

This event occurs after the document is cleared.

This event is called by TrvActionNew<sup>(101)</sup>.

TrvActionOpen<sup>(106)</sup> has this event as well, it may be called before the loading occurs. However, it calls this event only if it is not linked (either explicitly<sup>(108)</sup> or implicitly) with TrvActionNew. If linked, TrvActionOpen calls the event of TrvActionNew instead.

If style templates are not used<sup>(20)</sup>, this is a good place to define default values for collections of styles. For example:

**procedure** TForm1.rvActionNew1New(Sender: TObject);

**begin**

RVStyle1.ParaStyles.Clear;

RVStyle1.ParaStyles.Add; // one default paragraph style

RVStyle1.ListStyles.Clear; // no default list styles

```
RVStyle1.TextStyles.Clear;
with RVStyle1.TextStyles.Add do begin // one default text style
    FontName := 'Times New Roman';
    Size      := 12;
end;
end;
```

If style templates are used, styles are reset automatically.

### 1.3.1.3.2 Properties

#### In TrvActionCustomNew

■ ActionSave <sup>(102)</sup>

#### Derived from TrvAction <sup>(311)</sup>

■ Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

#### Derived from TAction

■ AutoCheck

■ Caption

■ Checked

DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

#### 1.3.1.3.2.1 TrvActionCustomNew.ActionSave

Links this action to TrvActionSave <sup>(122)</sup> action.

**property** ActionSave: TrvActionSave <sup>(122)</sup>;

If this link is not defined, the first found TrvActionSave <sup>(122)</sup> action on the same form/datamodule is used. If no TrvActionSave <sup>(122)</sup> action found, an exception occurs on executing.

### 1.3.1.4 TrvActionNew

TrvActionNew is the action for "File | New" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionNew = class (TrvActionCustomNew(101))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomNew*<sup>(101)</sup>

#### Description

If you use this action, you must also use TrvActionSave<sup>(122)</sup> and TrvActionSaveAs<sup>(127)</sup> actions.

This action must be linked to TrvActionSave<sup>(122)</sup> action, because TrvActionSave<sup>(122)</sup> contains a list of all documents opened in TCustomRichViewEdit controls. This link may be defined explicitly (via ActionSave<sup>(102)</sup> property) or implicitly (this action finds and uses TrvActionSave<sup>(122)</sup> action placed on the same form/datamodule).

#### This action performs the following tasks, if style templates are not used<sup>(20)</sup>:

1. If the document was modified, asks for saving and saves it (using TrvActionSave<sup>(122)</sup>).
2. Clears the document (calls TCustomRichViewEdit.Clear); clears the background image; if *rvfoSaveDocProperties* is included in TCustomRichViewEdit.RVFOptions, clears its DocProperties and resets DocParameters properties.
3. Applies the default values specified in the properties of GetControlPanel<sup>(336)</sup>: DefaultColor<sup>(42)</sup>, DefaultMargin<sup>(45)</sup>, DefaultDocParameters<sup>(43)</sup>.
4. Calls OnNew<sup>(101)</sup> event.
5. Calls OnMarginsChanged<sup>(65)</sup> event of GetControlPanel<sup>(336)</sup> component.
6. Formats the document (calls TCustomRichViewEdit.Format).
7. If GetControlPanel<sup>(336)</sup>.AutoDeleteUnusedStyles<sup>(42)</sup> = *True*, deletes unused styles (calls TCustomRichViewEdit.DeleteUnusedStyles(True, True, True)).
8. Stores the following values for this TCustomRichViewEdit using the properties of GetControlPanel<sup>(336)</sup>: temporal file name = DefaultFileName<sup>(44)</sup>, file format = DefaultFileFormat<sup>(44)</sup>, custom filter index (used if file format is *ffeCustom*) = DefaultCustomFilterIndex<sup>(43)</sup>. These values are stored in the list of documents maintained by TrvActionSave<sup>(122)</sup>.
9. Calls OnDocumentFileChange<sup>(126)</sup> event of TrvActionSave<sup>(122)</sup> action.

#### If style templates are used<sup>(20)</sup>:

The sequence of operations is the same, but the step #7 is different. The action resets text and paragraph styles according to StyleTemplates<sup>(104)</sup>.

### 1.3.1.4.1 Properties

#### In TrvActionNew

---

- ApplyStyleTemplates <sup>(104)</sup>
- StyleTemplates <sup>(104)</sup>

#### Derived from TrvActionCustomNew <sup>(101)</sup>

---

- ActionSave <sup>(102)</sup>
- Derived from TrvAction <sup>(311)</sup>
- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.4.1.1 TrvActionNew.StyleTemplates, ApplyStyleTemplates

A collection of style templates for assigning to new documents.

```
property StyleTemplates: TRVStyleTemplateCollection;
property ApplyStyleTemplates: Boolean;
```

These properties are used:


- when TrvActionNew is executed
- when TrvActionOpen <sup>(106)</sup> (linked to this action) clears a target editor before loading a file.

Values in **StyleTemplates** are measured in GetControlPanel <sup>(336)</sup>.UnitsProgram <sup>(55)</sup>.

If style templates are used <sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=*True*), and **ApplyStyleTemplates** = *True*, the action:

1. deletes all items in TextStyles, ParaStyles, ListStyles;

2. adds one paragraph style; if StyleTemplates contain "Normal" style template, it is applied to this paragraph style;
3. adds one text style; if StyleTemplates contain "Normal" style template, this text style is formatted accordingly.

 **ScaleRichView note:** for TSRichViewEdit, this the action performs similar operations with its header and footer. However, instead of "Normal", the action uses "header" style template for RVHeader, and "footer" style template for RVFooter properties.

#### Default value:

By default, the property contains the following style templates: "Normal", "heading 1", "heading 2", "heading 3", "Hyperlink", "header", "footer", "footnote reference", "endnote reference", "footnote text", "endnote text", "caption".

"header" and "footer" are not used by TRichViewEdit (used only in ScaleRichView), so you can delete these items.

**See also properties of GetControlPanel** <sup>(336)</sup>:

- DefaultColor <sup>(42)</sup>
- DefaultMargin <sup>(45)</sup>
- DefaultDocParameters <sup>(43)</sup>

**See also:**

- TrvActionStyleTemplates <sup>(212)</sup>.StandardStyleTemplates <sup>(214)</sup>

### 1.3.1.4.2 Methods

#### In TrvActionNew

Reset <sup>(105)</sup>

**Inherited from TrvCustomAction** <sup>(333)</sup>

GetControlPanel <sup>(336)</sup>

#### 1.3.1.4.2.1 TrvActionNew.Reset

Resets properties of rve.

**procedure** Reset(rve: TCustomRichViewEdit);

This method performs almost the same operation as this action when it is executed: it assigns default values to properties of **rve**; if style templates are used, it applies StyleTemplates <sup>(104)</sup>.

This method:

- does not clear **rve**;
- does not delete unused styles (if style templates are not used, otherwise all old styles are deleted when StyleTemplates <sup>(104)</sup> are applied);
- (the most important) does not associate any file with **rve**.

Normally, you do not need to call this method, because all necessary operations are performed when the action is executed.

However, this method is useful if you work with TDBRichViewEdit or TDBSRichViewEdit. For db-controls, you cannot use file-based actions (TrvActionNew <sup>(103)</sup>, TrvActionSave <sup>(122)</sup>,

TrvActionSaveAs<sup>(127)</sup>, TrvActionOpen<sup>(106)</sup>) directly. However, you can place TrvActionNew of a form, clear its Shortcut (to prevent execution on Ctrl+N), and call rvActionNew1.Reset(DBRichViewEdit1) in DBRichViewEdit1.OnNewDocument event.

### 1.3.1.5 TrvActionOpen

TrvActionOpen is the action for "File | Open..." command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionOpen = class (TrvActionCustomNew(101))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomNew*<sup>(101)</sup>

#### Description

The action displays a file opening dialog and loads the chosen file in TCustomRichViewEdit component. After opening, this editor is associated with the chosen file name and format, so subsequent executions of TrvActionSave<sup>(122)</sup> action save content of this TCustomRichViewEdit component in this file in this format.

The following formats are supported:

- RichView Format (RVF)
- Microsoft Word Document (DocX)
- Rich Text Format (RTF)
- Text files (TXT, Unicode and ANSI)
- **Markdown** (MARKDOWN, MDOWN, MKDN, MD, MKD, MDWN, MDTXT, MDTEXT)
- HTML
- RichViewXML Format (XML, only if RichViewXML<sup>(368)</sup> is used)
- custom formats, supported in GetControlPanel<sup>(336)</sup>.OnCustomFileOperation<sup>(62)</sup> event.

If you use this action, you must also use TrvActionSave<sup>(122)</sup> and TrvActionSaveAs<sup>(127)</sup> actions.

This action must be linked to TrvActionSave<sup>(122)</sup> action, because TrvActionSave<sup>(122)</sup> contains a list of all documents opened in TCustomRichViewEdit controls. This link may be defined explicitly (via ActionSave<sup>(102)</sup> property) or implicitly (this action finds and uses TrvActionSave<sup>(122)</sup> action placed on the same form/datamodule).

This action may be linked to TrvActionNew<sup>(103)</sup> action. This link may be defined explicitly (via ActionNew<sup>(108)</sup> property) or implicitly (this action finds and uses TrvActionNew<sup>(103)</sup> action placed on

the same form/datamodule). If linked, TrvActionNew<sup>(103)</sup> is used to reset editor before loading (it is especially important, if style templates are used<sup>(20)</sup>).

**This action performs the following tasks:**

1. If the document was modified, asks for saving and saves it (using TrvActionSave<sup>(122)</sup>).
2. Shows a file opening dialog allowing the user to choose the file for opening.
3. If the user chose a file, calls LoadFile<sup>(109)</sup> method. The file format is determined not by the file extension but by the filter index chosen in the file opening dialog.

### 1.3.1.5.1 Properties

#### In TrvActionOpen

- ActionNew<sup>(108)</sup>
- CustomFilter<sup>(108)</sup>
- DialogTitle<sup>(108)</sup>
- Filter<sup>(108)</sup>
- InitialDir<sup>(109)</sup>
- LastFilterIndex<sup>(109)</sup>

#### Derived from TrvActionNew<sup>(101)</sup>

- ActionSave<sup>(102)</sup>

#### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.5.1.1 TrvActionOpen.ActionNew

Links this action to TrvActionNew<sup>(101)</sup> action.

**property** ActionNew: TrvActionNew<sup>(101)</sup>;

If this link is not defined, the first found TrvActionNew<sup>(101)</sup> action on the same form/datamodule is used.

#### 1.3.1.5.1.2 TrvActionOpen.CustomFilter

Specifies the file filter for custom file formats.

**property** CustomFilter: TRVALocString<sup>(348)</sup>;

This property allows opening files in custom formats (in addition to the standard formats listed in the Filter<sup>(108)</sup> property of this action).

Value of this property is appended to the Filter property of the file opening dialog, if the Filter<sup>(108)</sup> property includes *ffiCustom*.

**Default value:**

" (empty string)

**See also:**

- GetControlPanel<sup>(336)</sup>.OnCustomFileOperation<sup>(62)</sup>
- TrvActionCustomFileIO.CustomFilter<sup>(98)</sup>

#### 1.3.1.5.1.3 TrvActionOpen.DialogTitle

Defines a custom title (caption) for the file opening dialog.

**property** DialogTitle: TRVALocString<sup>(348)</sup>;

If this property is empty, default (localized) dialog title is used.

**Default value:**

" (empty string)

**See also:**

- TrvActionCustomIO.DialogTitle<sup>(100)</sup>

#### 1.3.1.5.1.4 TrvActionOpen.Filter

Defines a set of file formats appearing in the file opening dialog.

**property** Filter: TrvFileOpenFilterSet<sup>(387)</sup>;

If *ffiCustom* is included, formats listed in the CustomFilter<sup>(108)</sup> property are used.

Name and extension of RichView Format (RVF) may be customized, see GetControlPanel<sup>(336)</sup>.RVFFilter<sup>(52)</sup>.

**Default value:**

all formats



#### 1.3.1.5.1.5 TrvActionOpen.InitialDir

Specifies the initial directory for the file dialog.

**property** InitialDir: TRVALocString<sup>(348)</sup>;

Value of this property is assigned to InitialDir property of the file opening dialog before showing it.

**Default value:**

" (empty string)

#### 1.3.1.5.1.6 TrvActionOpen.LastFilterIndex

Defines file format selected in the file opening dialog by default.

**property** LastFilterIndex: Integer;

Value of this property is assigned to FilterIndex property of the file opening dialog before showing it.

**Default value:**

1

#### 1.3.1.5.2 Methods

##### In TrvActionOpen

---

LoadFile<sup>(109)</sup>

##### Inherited from TrvCustomAction<sup>(333)</sup>

---

GetControlPanel<sup>(336)</sup>

#### 1.3.1.5.2.1 TrvActionOpen.LoadFile

Loads the specified file in **rve**.

```
procedure LoadFile(rve: TCustomRichViewEdit;
  const FileName: TRVUnicodeString;
  FileFormat: TrvFileOpenFilter(387); CustomFilterIndex: Integer=0);
```

**Parameters:**

**rve** – editor where to load file.

**FileName** – name of the file to load.

**FileFormat** – format of this file.

**CustomFilterIndex** used only if **FileFormat** is *ffiCustom*.

If **FileFormat** = *ffiCustom*, **CustomFilterIndex** identifies a custom file format. Custom formats are numbered from 1 in the order they are listed in the CustomFilter<sup>(108)</sup> property.

**This procedure performs the following tasks:**

1. Clears the document (calls TCustomRichViewEdit.Clear); clears the background image; if *rvfoSaveDocProperties* is included in TCustomRichViewEdit.RVFOptions, clears its DocProperties and resets DocParameters properties\*

2. Applies the default values specified in the properties of GetControlPanel<sup>(336)</sup>: DefaultColor<sup>(42)</sup>, DefaultMargin\*<sup>(45)</sup>
3. Calls OnNew<sup>(101)</sup> event\*
4. If FileFormat = *ffiTextANSI*, and the linked<sup>(335)</sup> ControlPanel's UseTextCodePageDialog<sup>(57)</sup> = *True*, shows a dialog for choosing the text code page.
5. Loads the file in **rve**.
6. Associates **rve** with this file and file format (in the list of documents maintained by TrvActionSave<sup>(122)</sup>).
7. Calls OnDocumentFileChange<sup>(126)</sup> event of the linked (implicitly or explicitly<sup>(102)</sup>) TrvActionSave<sup>(122)</sup> action.
8. Calls OnMarginsChanged<sup>(65)</sup> event of GetControlPanel<sup>(336)</sup> component.
9. Formats the document (calls TCustomRichViewEdit.Format).
10. If GetControlPanel<sup>(336)</sup>.AutoDeleteUnusedStyles<sup>(42)</sup> = *True*, deletes unused styles (calls TCustomRichViewEdit.DeleteUnusedStyles(True, True, True)).
11. Calls OnOpenFile<sup>(110)</sup> event.

Subsequent executions of TrvActionSave<sup>(122)</sup> will save **rve** content in this file in this format.

\* If TrvActionNew<sup>(103)</sup> is linked (either implicitly or explicitly<sup>(108)</sup>), its functions for document clearing and resetting are used; they do the same, but also apply TrvActionNew.StyleTemplates<sup>(104)</sup>.

### 1.3.1.5.3 Events

#### In TrvActionOpen

■ OnOpenFile<sup>(110)</sup>

#### Derived from TrvActionNew<sup>(101)</sup>

■ OnNew<sup>(101)</sup>

#### 1.3.1.5.3.1 TrvActionOpen.OnOpenFile

Occurs when the action opened a file.

##### type

```
TRVOpenFileEvent = procedure (Sender: TObject;
  Editor: TCustomRichViewEdit;
  const FileName: TRVUnicodeString;
  FileFormat: TrvFileOpenFilter(387);
  CustomFilterIndex: Integer) of object;
```

**property** OnOpenFile: TRVOpenFileEvent;

Parameters are the same as in LoadFile<sup>(109)</sup> method.

### 1.3.1.6 TrvActionExport

TrvActionExport is the action for "File | Export..." command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionExport = class (TrvActionCustomFileIO96)
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>  
*TrvActionCustomIO*<sup>98</sup>  
*TrvActionCustomFileIO*<sup>96</sup>

## Description

This action displays a file saving dialog and saves the content of *TRichViewEdit* component in the chosen file.

In addition to formats supported by *TrvActionSaveAs*<sup>127</sup> action, this action supports:

- simplified HTML (without CSS)
- formats supported by Microsoft Office file export converters.

After the execution, this *TCustomRichViewEdit* component is still associated with the file name and format assigned by the last execution of *TrvActionNew*<sup>101</sup>, *TrvActionOpen*<sup>106</sup> or *TrvActionSaveAs*<sup>127</sup> action. *TrvActionExport* does not change this association.

### This action performs the following tasks:

1. Shows a file saving dialog allowing the user to choose the file name and format for exporting.
2. If the user chose a file, calls *ExportToFile*<sup>114</sup> method. The file format is determined not by the file extension but by the filter index chosen in the file saving dialog.

## 1.3.1.6.1 Properties

### In *TrvActionExport*

- *CreateDirectoryForHTMLImages*<sup>113</sup>
- *Filter*<sup>113</sup>

### Derived from *TrvActionCustomFileIO*<sup>96</sup>

- *CustomFilter*<sup>98</sup>

### Derived from *TrvActionCustomIO*<sup>98</sup>

- *AutoUpdateFileName*<sup>99</sup>
- *DialogTitle*<sup>100</sup>
- *FileName*<sup>100</sup>
- *InitialDir*<sup>100</sup>

### Derived from *TrvAction*<sup>311</sup>

- *Control*<sup>312</sup>

## Derived from TrvCustomAction <sup>333</sup>

---

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible CustomFilter <sup>98</sup>

## Derived from TrvActionCustomIO <sup>98</sup>

---

- AutoUpdateFileName <sup>99</sup>
- DialogTitle <sup>100</sup>
- FileName <sup>100</sup>
- InitialDir <sup>100</sup>

## Derived from TrvAction <sup>311</sup>

---

- Control <sup>312</sup>

## Derived from TrvCustomAction <sup>333</sup>

---

- Caption <sup>334</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword

- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.6.1.1 TrvActionExport.CreateDirectoryForHTMLImages

Specifies whether a subdirectory for saving images is created when exporting HTML or Markdown file.

**property** `CreateDirectoryForHTMLImages: Boolean;`

This property is used when exporting to HTML and Markdown formats (*ffeHTMLCSS*, *ffeHTML*, *ffeMarkdown*<sup>(386)</sup>).

If *True*, images for HTML/Markdown files are saved in a subdirectory having the same name as HTML/Markdown file but '.files' extension.

If *False*, all images are saved in the same directory as the HTML/Markdown file.

This option is used for images with undefined (empty) file names, or when *rvhtmlsioUseItemImageFileNames* is not included in `HTMLSaveProperties.ImageOptions` of the target editor.

##### Default value:

*True*

##### See also:

- `TrvActionSave.CreateDirectoryForHTMLImages`<sup>(124)</sup>

#### 1.3.1.6.1.2 TrvActionExport.Filter

Defines a set of file formats appearing in the file saving dialog.

**property** `Filter: TrvFileExportFilterSet`<sup>(386)</sup>;

If *ffeCustom* is included, formats listed in the `CustomFilter`<sup>(98)</sup> property are used.

Name and extension of RichView Format (RVF) may be customized, see `GetControlPanel`<sup>(336)</sup>.  
`.RVFilter`<sup>(52)</sup>.

##### Default value:

all formats

#### 1.3.1.6.2 Methods

##### In TrvActionExport

`ExportToFile`<sup>(114)</sup>

##### Inherited from TrvCustomAction<sup>(333)</sup>

`GetControlPanel`<sup>(336)</sup>

### 1.3.1.6.2.1 TrvActionExport.ExportToFile

Saves **Edit** in the specified file in the specified format.

```
function ExportToFile(Edit: TCustomRichViewEdit;
  const FileName: TRVUnicodeString;
  ExportFormat: TrvFileExportFilter386;
  ConverterOrCustomIndex: Integer;
  rvc: TRVOfficeConverter): Boolean;
```

#### Parameters:

**Edit** – editor for saving.

**FileName** – file name for saving.

**FileFormat** – file format for saving.

**CustomFilterIndex** used only if **FileFormat** is *ffeCustom* or *ffeOfficeConverters*.

If **FileFormat** = *ffeCustom*, **CustomFilterIndex** identifies a custom file format. Custom formats are numbered from 1 in the order they are listed in the CustomFilter<sup>98</sup> property.

If **FileFormat** = *ffeOfficeConverters*, **CustomFilterIndex** identifies a format supported by Microsoft Office file export converters. Formats are numbered from 0. A list of these formats and their order are different on different computers. This is the index in **rvc.ExportConverters** collection.

If saving is unsuccessful, an error message is displayed.

#### Return value:

*True* if the saving was successful.

### 1.3.1.7 TrvActionPageSetup

TrvActionPageSetup is the action for "File | Page Setup" command.

**Unit** RichViewActions<sup>92</sup>;

#### Syntax

```
TrvActionPageSetup = class (TrvCustomAction333)
```

#### Hierarchy

```
TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction333
```

#### Description

This action displays a page setup dialog.

This action does not work with the specific editor. It defines global page properties for the application.

This action is enabled in `GetControlPanel336.RVPrint53` is assigned, and there is at least one printer is installed in the system (`Printer.Printers.Count>0`).

The settings are stored:

- margins are stored in `GetControlPanel336.RVPrint53` properties;
- headers and footers are stored in `GetControlPanel336.Header and Footer47` properties;
- other properties (paper size, orientation, source) are stored in the application's printing settings (in the object returned by `Printer` function).

### 1.3.1.7.1 Properties

#### In TrvActionPageSetup

- `MarginsUnits115`
- `UnitsPixelsPerInch116`

#### Derived from TrvCustomAction<sup>333</sup>

- `Caption334`
- `ControlPanel335`
- `Disabled335`
- `Hint335`

#### Derived from TAction

- `AutoCheck`
- `Caption`
- `Checked`
- `DisableIfNoHandler`
- `Enabled`
- `GroupIndex`
- `HelpContext`
- `HelpKeyword`
- `HelpType`
- `Hint`
- `ImageIndex`
- `Name`
- `SecondaryShortCuts`
- `ShortCut`
- `Visible`

### 1.3.1.7.1.1 TrvActionPageSetup.MarginsUnits

Defines measuring units used for margins in the page setup dialog .

#### type

```
TrvPaperMarginsUnits = (rvpmuMillimeters, rvpmuInches);
```

**property** `MarginsUnits`: `TrvPaperMarginsUnits`;

**Default value:**

*rvpmuMillimeters*

### 1.3.1.7.1.2 TrvActionPageSetup.UnitsPixelsPerInch

Specify a count of pixels in one logical inch.

**property** UnitsPixelsPerInch: Integer;

This value is used when converting RVPrint<sup>(53)</sup>.Margins to MarginUnits<sup>(115)</sup> when displaying a page setup dialog, and back to RVPrint<sup>(53)</sup>.Units when applying changes.

This property must be equal to Style.UnitsPixelsPerInch property of target editors.

**Default value:**

96

### 1.3.1.7.2 Events

#### In TrvActionPageSetup

■ OnChange<sup>(116)</sup>

### 1.3.1.7.2.1 TrvActionPageSetup.OnChange

Occurs after the changes made in the page setup dialog are applied.

**property** OnChange: TNotifyEvent;

## 1.3.1.8 TrvActionPrint

TrvActionPrint is the action for "File | Print..." command.

**Unit** RichViewActions<sup>(92)</sup>;

### Syntax

TrvActionPrint = **class** (TrvCustomPrintAction<sup>(129)</sup>)

### Hierarchy

TObject  
 TPersistent  
 TComponent  
 TBasicAction  
 TContainedAction  
 TCustomAction  
 TAction  
 TrvCustomAction<sup>(333)</sup>  
 TrvAction<sup>(311)</sup>  
 TrvCustomPrintAction<sup>(129)</sup>

### Description

This action displays a printing dialog (TPrintDialog) and prints the document from TCustomRichViewEdit component.

This action uses settings defined in TrvActionPageSetup<sup>(114)</sup>. If GetControlPanel<sup>(336)</sup>.RVPrint<sup>(53)</sup> is not assigned, the action uses the first TRVPrint component found on the form owning the



TCustomRichViewEdit component. If not found, it creates a temporal TRVPrint with default property settings.

The action can print the whole document, the range of pages, or the selected fragment, depending on the user's choice.

This action is enabled only if at least one printer is installed in the system (Printer.Printers.Count>0).

**See also:**

- TrvActionQuickPrint<sup>(120)</sup>
- TrvActionPrintPreview<sup>(118)</sup>

### 1.3.1.8.1 Properties

#### In TrvActionPrint

---

- Title<sup>(118)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.1.8.1.1 TrvActionPrint.Title

Determines the text that is listed in the Print Manager.

**property** Title: TRVUnicodeString;

**See also:**

TrvActionQuickPrint.Title<sup>(122)</sup>

### 1.3.1.9 TrvActionPrintPreview

TrvActionPrintPreview is the action for "File | Print Preview" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionPrintPreview = **class** (TrvCustomPrintAction<sup>(129)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvCustomPrintAction*<sup>(129)</sup>

**Description**

This action displays a printing preview window. This window has "Page Setup" and "Print" buttons where the user can print the document. For the "Page Setup" button, TrvActionPageSetup<sup>(114)</sup> action is used. If it is not linked (via ActionPageSetup<sup>(119)</sup> property), a temporal TrvActionPageSetup<sup>(114)</sup> action is used. For the "Print" button, TrvActionPrint<sup>(116)</sup> action is used. If it is not linked (via ActionPrint<sup>(119)</sup> property), the first TrvActionPrint<sup>(116)</sup> action on the same form/datamodule is used; if not found, printing in this action is not possible.

This action uses settings defined in TrvActionPageSetup<sup>(114)</sup>. If GetControlPanel<sup>(336)</sup>.RVPrint<sup>(53)</sup> is not assigned, the action uses the first TRVPrint component found on the form owning the TCustomRichViewEdit component. If not found, it creates a temporal TRVPrint with default property settings.

This action is enabled only if at least one printer is installed in the system (Printer.Printers.Count>0).

**See also:**

- TrvActionQuickPrint<sup>(120)</sup>
- TrvActionPrint<sup>(116)</sup>

### 1.3.1.9.1 Properties

#### In TrvActionPrintPreview

- [ActionPageSetup](#) <sup>(119)</sup>
- [ActionPrint](#) <sup>(119)</sup>
- [Maximized](#) <sup>(120)</sup>

#### Derived from TrvAction <sup>(311)</sup>

- [Control](#) <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

- [Caption](#) <sup>(334)</sup>
- [Disabled](#) <sup>(335)</sup>
- [Hint](#) <sup>(335)</sup>

#### Derived from TAction

- [AutoCheck](#)
- [Caption](#)
- [Checked](#)
- [DisableIfNoHandler](#)
- [Enabled](#)
- [GroupIndex](#)
- [HelpContext](#)
- [HelpKeyword](#)
- [HelpType](#)
- [Hint](#)
- [ImageIndex](#)
- [Name](#)
- [SecondaryShortCuts](#)
- [ShortCut](#)
- [Visible](#)

#### 1.3.1.9.1.1 TrvActionPrintPreview.ActionPageSetup

A link to [TrvActionPageSetup](#) <sup>(114)</sup> action.

**property** `ActionPageSetup: TrvActionPageSetup (114);`

This link is used for the "Page Setup" button on the print preview dialog.

If this link is not assigned, a temporal action is used.

#### 1.3.1.9.1.2 TrvActionPrintPreview.ActionPrint

A link to [TrvActionPrint](#) <sup>(116)</sup> action.

**property** `ActionPrint: TrvActionPrint (116);`

This link is used for the "Print" button on the print preview dialog.

If this link is not assigned, the first found [TrvActionPrint](#) <sup>(116)</sup> action on the same form is used. If not found, printing is not possible.

### 1.3.1.9.1.3 TrvActionPrintPreview.Maximized

Defines the initial state of the print preview window.

**property** Maximized: Boolean;

Assign *True* to this property to display the print preview window maximized.

**Default value:**

*False*

### 1.3.1.9.2 Events

#### In TrvActionPrintPreview

■ OnGetPreviewFormClass <sup>(120)</sup>

#### 1.3.1.9.2.1 TrvActionPrintPreview.OnGetPreviewFormClass

Allows defining your own window for a print preview.

**type**

```
TRVGetFormClassEvent = procedure (Sender: TObject;  
    var FormClass: TFormClass) of object;
```

**property** OnGetPreviewFormClass: TRVGetFormClassEvent;

You can return your own form class for a print preview window. This class must be inherited from TfrmRVPreview, otherwise an exception occurs.

### 1.3.1.10 TrvActionQuickPrint

TrvActionQuickPrint is the action for the "quick print" command (usually assigned to a toolbar button).

**Unit** RichViewActions <sup>(92)</sup>;

**Syntax**

```
TrvActionQuickPrint = class (TrvCustomPrintAction (129))
```

**Hierarchy**

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction
```

*TAction*

*TrvCustomAction* <sup>(333)</sup>

*TrvAction* <sup>(311)</sup>

*TrvCustomPrintAction* <sup>(129)</sup>

## Description

This action prints the document from TCustomRichViewEdit component without displaying a printing dialog.

This action uses settings defined in TrvActionPageSetup <sup>(114)</sup> action. If GetControlPanel <sup>(336)</sup>.RVPrint <sup>(53)</sup> is not assigned, the action uses the first TRVPrint component found on the form owning the TCustomRichViewEdit component. If not found, it creates a temporal TRVPrint with default property settings.

This action is enabled only if at least one printer is installed in the system (Printer.Printers.Count>0).

## See also:

- TrvActionPrint <sup>(116)</sup>
- TrvActionPrintPreview <sup>(118)</sup>

## 1.3.1.10.1 Properties

### In TrvActionQuickPrint

■ Title <sup>(122)</sup>

### Derived from TrvAction <sup>(311)</sup>

■ Control <sup>(312)</sup>

### Derived from TrvCustomAction <sup>(333)</sup>

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

### Derived from TAction

■ AutoCheck

■ Caption

■ Checked

DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.10.1.1 TrvActionQuickPrint.Title

Determines the text that is listed in the Print Manager.

**property** Title: TRVUnicodeString;

**See also:**

TrvActionPrint.Title<sup>(118)</sup>

#### 1.3.1.11 TrvActionSave

TrvActionSave is the action for "File | Save" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

TrvActionSave = **class** (TrvAction<sup>(311)</sup>)

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

##### Description

This action saves the document from TCustomRichViewEdit to a file. In addition, this action maintains a list of all documents (TCustomRichViewEdits associated with their file names and formats).

If the document has a temporal name (assigned after executing TrvActionNew<sup>(101)</sup> action), this action executes TrvActionSaveAs<sup>(127)</sup> action. So, if you use this action, you must also use TrvActionSaveAs<sup>(127)</sup> action. The link between the actions may be defined explicitly (via ActionSaveAs<sup>(123)</sup> property) or implicitly (this action finds and uses TrvActionSaveAs<sup>(127)</sup> action placed on the same form/datamodule).

If the document has a permanent name and format (assigned after executing TrvActionOpen<sup>(106)</sup> or TrvActionSaveAs<sup>(127)</sup> action), the action just saves the file.

If the file cannot be saved, an error message is displayed.

CanCloseDoc<sup>(125)</sup> method should be used in Form.OnCloseQuery event.

### 1.3.1.11.1 Properties

#### In TrvActionSave

- ActionSaveAs <sup>(123)</sup>
- CreateDirectoryForHTMLImages <sup>(124)</sup>
- DisableWhenUnmodified <sup>(124)</sup>
- ▶ Documents <sup>(124)</sup>
- LostFormatWarning <sup>(125)</sup>
- SuppressNextErrorMessage <sup>(125)</sup>

#### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.11.1.1 TrvActionSave.ActionSaveAs

Links this action to TrvActionSaveAs <sup>(127)</sup> action.

```
property ActionSaveAs: TrvActionSaveAs (127);
```

TrvActionSaveAs <sup>(127)</sup> action is used when the document has a temporal name (after the execution of TrvActionNew <sup>(101)</sup> action). In this case, instead of saving, the action executes TrvActionSaveAs <sup>(127)</sup> action.

If this link is not defined, the first found TrvActionSaveAs <sup>(127)</sup> action on the same form/datamodule is used. If no TrvActionSaveAs <sup>(127)</sup> action found, an exception occurs on executing.

### 1.3.1.11.1.2 TrvActionSave.CreateDirectoryForHTMLImages

Specifies whether a subdirectory for saving images is created when saving HTML file.

**property** `CreateDirectoryForHTMLImages: Boolean;`

This property is used when saving to HTML format.

If *True*, images for HTML files are saved in subdirectory having the same name as HTML file but '.files' extension.

If *False*, all images are saved in the same directory as the HTML file.

This option is used for images with undefined (empty) file names, or when *rvhtmlsioUseItemImageFileNames* is not included in `HTMLSaveProperties.ImageOptions` of the target editor.

**Default value:**

*True*

**See also:**

`TrvActionExport.CreateDirectoryForHTMLImages` <sup>(113)</sup>

### 1.3.1.11.1.3 TrvActionSave.DisableWhenUnmodified

Allows disabling the action if the editor is unmodified.

**property** `DisableWhenUnmodified: Boolean;`

If *False*, the action is always enabled.

If *True*, the action is enabled only when a document in the target editor is changed.

**Default value**

*False*

### 1.3.1.11.1.4 TrvActionSave.Documents

A read-only list of opened documents.

**property** `Documents: TRVList;`

This is a list of `TrvaDocumentInfo` objects.

The main properties of `TrvaDocumentInfo` are:

- `rve`: `TCustomRichViewEdit` – editor;
  - `FileName`: `TRVUnicodeString` – file name (full path) for the document in `rve`.
  - `FileFormat`: `TrvFileSaveFilter` <sup>(386)</sup> – format for saving this file.
  - `CustomFilterIndex`: Integer, used only if `FileFormat=ffeCustom`. Custom formats are numbered from 1 in the order they are listed in the `CustomFilter` <sup>(98)</sup> property of the linked `TrvActionSaveAs` <sup>(127)</sup> action.
  - `Defined` – *False* if `FileName` and `FileFormat` contain temporal values assigned for new document.
- This list is maintained automatically.

**See also methods:**

- `FindDoc` <sup>(126)</sup>



### 1.3.1.11.1.5 TrvActionSave.LostFormatWarning

Lists "lossy" file formats.

**property** LostFormatWarning: TrvFileSaveFilterSet<sup>386</sup>;

When saving to file in one of these formats, a warning/confirmation is displayed:

*"<FileName> may contain features that are not compatible with the chosen saving format. Do you want to save the document in this format?".* This message is displayed only when saving the document in this format for the first time.

**Default value:**

all formats except for RVF (RichView Format)

### 1.3.1.11.1.6 TrvActionSave.SuppressNextErrorMessage

Turns off the default error message when saving to the custom format (*ffeCustom*<sup>386</sup>).

**property** SuppressNextErrorMessage: Boolean;

This property is set to *False* before calling GetControlPanel<sup>336</sup>.OnCustomFileOperation<sup>62</sup>.

If you assign *True* to this property in this event, an error message will not be displayed even if *False* is returned in the Success parameter of this event.

### 1.3.1.11.2 Methods

#### In TrvActionSave

CanCloseDoc<sup>125</sup>

FindDoc<sup>126</sup>

#### Inherited from TrvCustomAction<sup>333</sup>

GetControlPanel<sup>336</sup>

### 1.3.1.11.2.1 TrvActionSave.CanCloseDoc

Performs necessary operations when closing the form containing **rve**.

**function** CanCloseDoc(rve: TCustomRichViewEdit): Boolean;

If document in **rve** is not modified, the method returns *True*.

Otherwise, the method asks the user: *"Save changes to <FileName>?"* (Yes/No/Cancel). If "Yes", the method saves and returns *True*. If "No", the method does not save and returns *True*. If "Cancel", the method returns *False*.

This method should be used in OnCloseQuery method of the form containing **rve**.

**Example:**

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
    CanClose := rvActionSave1.CanCloseDoc(RichViewEdit1);
end;
```

### 1.3.1.11.2.2 TrvActionSave.FindDoc

Returns the index of item in Documents<sup>(124)</sup> for the given editor **rve**. Returns -1 if not found.

```
function FindDoc(rve: TCustomRichViewEdit): Integer;
```

### 1.3.1.11.3 Events

#### In TrvActionSave

- OnDocumentFileChange<sup>(126)</sup>
- OnSave<sup>(126)</sup>
- OnSaving<sup>(126)</sup>

#### 1.3.1.11.3.1 TrvActionSave.OnDocumentFileChange

Occurs when file for document in **Editor** is changed.

##### type

```
TRVFileChangeEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit; const FileName: TRVUnicodeString;  
    FileFormat: TrvFileSaveFilter(386); IsNew: Boolean) of object;
```

```
property OnDocumentFileChange: TRVFileChangeEvent;
```

This event occurs in the following cases:

- TrvActionNew<sup>(101)</sup> action is executed;
- TrvActionOpen<sup>(106)</sup> action is executed, or TrvActionOpen.LoadFile<sup>(109)</sup> is called;
- TrvActionSaveAs<sup>(127)</sup> action is executed.

**FileName** is a file name (full path) assigned to the document in **Editor**.

**FileFormat** is a file format assigned to the document in **Editor**.

**IsNew** is *True* after the execution of TrvActionNew<sup>(101)</sup> action.

If **FileFormat**=*ffeCustom*, this event does not allow to identify the specific custom format. You can get it using Documents<sup>(124)</sup> property.

#### 1.3.1.11.3.2 TrvActionSave.OnSaving, OnSave

The events occur before and after saving to a file.

##### type

```
TRVSaveFileEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit; const FileName: TRVUnicodeString;  
    FileFormat: TrvFileSaveFilter(386);  
    CustomFilterIndex: Integer) of object;
```

```
property OnSaving: TRVSaveFileEvent;
```

```
property OnSave: TRVSaveFileEvent;
```

**OnSaving** occurs before saving to a file. If saving shows a dialog window (for example, for choosing a text file code page), this event occurs before showing these dialogs.

**OnSave** occurs after successful saving. It does not occur if saving was canceled (for example, in a code page dialog), or if an error occurs when saving to a file.

**Parameters:**

**Editor** – editor for saving.

**FileName** – file name (full path) to save.

**FileFormat** – format to save.

**CustomFilterIndex** used only if **FileFormat** is *ffeCustom*.

If **FileFormat** = *ffeCustom*, **CustomFilterIndex** identifies a custom file format. Custom formats are numbered from 1 in the order they are listed in the CustomFilter<sup>(98)</sup> property of the linked TrvActionSaveAs<sup>(127)</sup> action.

**1.3.1.12 TrvActionSaveAs**

TrvActionSaveAs is the action for "File | Save As..." command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionSaveAs = class (TrvActionCustomFileIO(96))
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionCustomIO(98)
TrvActionCustomFileIO(96)

```

**Description**

This action displays a file saving dialog and saves the content of TRichViewEdit component in the chosen file. The editor remains associated with this file name and file format, so subsequent executions of TrvActionSave<sup>(122)</sup> action save its content in this file in this format.

The following formats are supported:

- RichView Format (RVF)
- Rich Text Format (RTF)
- Microsoft Word Document (DocX)
- Text files (TXT, Unicode and ANSI)
- **Markdown** (MARKDOWN, MDOWN, MKDN, MD, MKD, MDWN, MDTXT, MDTEXT)
- HTML
- RichViewXML Format (XML, only if RichViewXML<sup>(368)</sup> is used)
- custom formats, supported in GetControlPanel<sup>(336)</sup>.OnCustomFileOperation<sup>(62)</sup> event.

If you use this action, you must also use TrvActionSave<sup>(122)</sup> action.

This action must be linked to TrvActionSave<sup>(122)</sup> action, because TrvActionSave<sup>(122)</sup> contains a list of all documents opened in TCustomRichViewEdit controls. Besides, this action uses TrvActionSave<sup>(122)</sup> to save the file when file name and format are chosen. This link may be defined explicitly (via ActionSave<sup>(129)</sup> property) or implicitly (this action finds and uses TrvActionSave<sup>(122)</sup> action placed on the same form/datamodule).

#### This action performs the following tasks:

1. Shows a file saving dialog allowing the user to choose the file name and format for saving.
2. If the chosen format is in LostFormatWarning<sup>(125)</sup> of TrvActionSave<sup>(122)</sup>, displays a warning about lossy saving.
3. Saves the file, associates TCustomRichViewEdit control with this file name and file format.
4. Calls OnDocumentFileChange<sup>(126)</sup> event of TrvActionSave<sup>(122)</sup> action.

### 1.3.1.12.1 Properties

#### In TrvActionSaveAs

---

- ActionSave<sup>(129)</sup>
- Filter<sup>(129)</sup>

#### Derived from TrvActionCustomFileIO<sup>(96)</sup>

---

- CustomFilter<sup>(98)</sup>

#### Derived from TrvActionCustomIO<sup>(98)</sup>

---

- AutoUpdateFileName<sup>(99)</sup>
- DialogTitle<sup>(100)</sup>
- FileName<sup>(100)</sup>
- InitialDir<sup>(100)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType

- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.1.12.1.1 TrvActionSaveAs.ActionSave

Links this action to TrvActionSave<sup>(122)</sup> action.

**property** ActionSave: TrvActionSave<sup>(122)</sup>;

If this link is not defined, the first found TrvActionSave<sup>(122)</sup> action on the same form/datamodule is used. If no TrvActionSave<sup>(122)</sup> action found, an exception occurs on executing.

#### 1.3.1.12.1.2 TrvActionSaveAs.Filter

Defines a set of file formats appearing in the file saving dialog.

**property** Filter: TrvFileSaveFilterSet<sup>(386)</sup>;

If *ffeCustom* is included, formats listed in the CustomFilter<sup>(98)</sup> property are used.

Name and extension of RichView Format (RVF) may be customized, see GetControlPanel<sup>(336)</sup>.RVFFilter<sup>(52)</sup>.

**Default value:**

all formats

### 1.3.1.13 TrvCustomPrintAction

TrvCustomPrintAction is a base class for the printing actions.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvCustomPrintActions = **class** (TrvAction<sup>(311)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

**Description**

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This action is not used directly, this is the base class for the following actions:

- TrvActionPrintPreview<sup>(118)</sup>

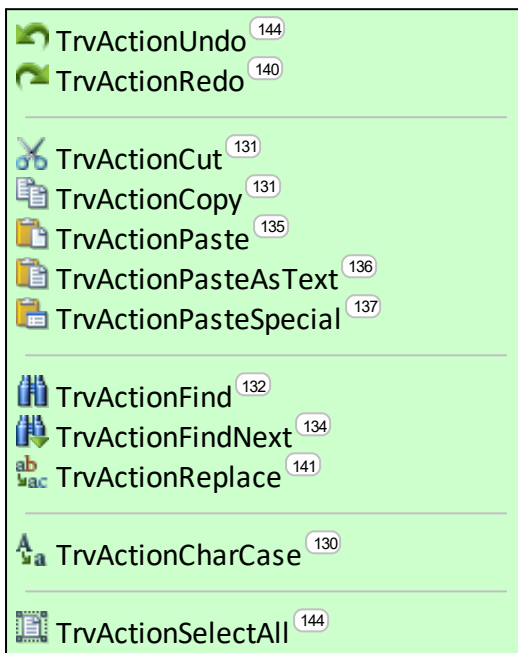
- TrvActionQuickPrint<sup>(120)</sup>
- TrvActionPrint<sup>(116)</sup>

All these actions are enabled only if at least one printer installed in the system (Printer.Printers.Count>0).

## 1.3.2 Edit

### Edit Actions

This group of actions includes undo/redo commands, Clipboard commands, find and replace commands, changing character case, page break commands.



#### 1.3.2.1 TrvActionCharCase

TrvActionCharCase is the action for "Character Case" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionCharCase = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

## Description

The action changes character case in the selected fragment: "all lower case" → "all upper case" → "first letter in each word upper case" → "all lower case" → ...

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

Limitation: this action cannot be applied to multicell selection in table.

## See also:

- RVChangeCharCase, RVGetCharCase <sup>(383)</sup> procedures.
- TrvActionFontAllCaps <sup>(146)</sup>

### 1.3.2.2 TrvActionCopy

TrvActionCopy is the action for "Edit | Copy" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionCopy = class (TrvCustomEditAction (145))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvCustomEditAction* <sup>(145)</sup>

## Description

This action copies the selection to the Clipboard (calls TCustomRichViewEdit.CopyDef).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action can work with several types of editors, see the comments in the TrvCustomEditAction <sup>(145)</sup> topic.

### 1.3.2.3 TrvActionCut

TrvActionCut is the action for "Edit | Cut" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionCut = class (TrvCustomEditAction (145))
```

## Hierarchy

*TObject*  
*TPersistent*

*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvCustomEditAction* <sup>(145)</sup>

## Description

This action cuts the selection to the Clipboard (calls *TCustomRichViewEdit.CutDef*).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action can work with several types of editors, see the comments in the *TrvCustomEditAction* <sup>(145)</sup> topic.

### 1.3.2.4 TrvActionFind

*TrvActionFind* is the action for "Edit | Find" command.

**Unit** *RichViewActions* <sup>(92)</sup>;

## Syntax

```
TrvActionFind = class (TrvAction (311))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

## Description

This action displays a *TFindDialog* for searching a substring in the *TCustomRichViewEdit* component. If at the moment of execution a single line of text is selected, this text is copied to the find dialog. If at the moment of execution a replace dialog is shown, it is closed.

The search starts from the caret position to the end or to the beginning of the document. If the text is found, it is selected. If not found, the action's behavior depends on *GetControlPanel* <sup>(336)</sup> *.SearchScope* <sup>(53)</sup> property:

- *rvssFromCursor*, the search stops;
- *rvssAskUser*, the action asks user whether to continue from the beginning/end;
- *rvssGlobal*, the search continues from the beginning/end.

If nothing found, a message is displayed.



In Delphi 2009+, or if TNT Controls <sup>(370)</sup> are used, the action searches for Unicode string. Otherwise, it searches for ANSI string.

**See also:**

TrvActionFindNext <sup>(134)</sup>  
TrvActionReplace <sup>(141)</sup>

### 1.3.2.4.1 Properties

#### In TrvActionFind

- ActionReplace <sup>(133)</sup>
- FindText <sup>(134)</sup>

#### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.2.4.1.1 TrvActionFind.ActionReplace

Links this action to TrvActionReplace <sup>(141)</sup> action.

**property** ActionReplace: TrvActionReplace <sup>(141)</sup>;

If this link is not defined, the first TrvActionReplace <sup>(141)</sup> action found in the same form/datamodule is used.

When this action is executed, it calls TrvActionReplace.CloseDialog <sup>(143)</sup>.

### 1.3.2.4.1.2 TrvActionFind.FindText

A string to initialize the find dialog.

**property** FindText: TRVUnicodeString;

If value of this property is not empty, it is assigned to the find dialog's FindText when a search is started. Otherwise, the dialog is initialized with text selected in the target editor.

#### Default value

'' (empty string)

### 1.3.2.4.2 Methods

#### In TrvActionFind

CloseDialog<sup>(134)</sup>

#### Inherited from TrvCustomAction<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

### 1.3.2.4.2.1 TrvActionFind.CloseDialog

Closes the find dialog (if it is shown).

**procedure** CloseDialog;

### 1.3.2.5 TrvActionFindNext

TrvActionFindNext is the action for "Edit | Find Next" action.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionFindNext = **class** (TrvAction<sup>(311)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action continues the search started by TrvActionFind<sup>(132)</sup> action. The link between the actions may be defined explicitly (via ActionFind<sup>(135)</sup> property) or implicitly (this action finds and uses TrvActionFind<sup>(132)</sup> action placed on the same form/datamodule).

### 1.3.2.5.1 Properties

#### In TrvActionFindNext

- ActionFind<sup>(135)</sup>

#### Derived from TrvAction<sup>(311)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.2.5.1.1 TrvActionFindNext.ActionFind

Links this action to TrvActionFind<sup>(132)</sup> action.

```
property ActionFind: TrvActionFind(132);
```

If this link is not defined, the first TrvActionFind<sup>(132)</sup> action found in the same form/datamodule is used. If it cannot be found, an exception occurs.

When executed, TrvActionFindNext action continues the search started by TrvActionFind<sup>(132)</sup> action. If search was not started, the action executes TrvActionFind<sup>(132)</sup> action instead.

### 1.3.2.6 TrvActionPaste

TrvActionPaste is the action for "Edit | Paste" command.

```
Unit RichViewActions(92);
```

#### Syntax

```
TrvActionPaste = class(TrvCustomEditAction(145))
```

#### Hierarchy

*TObject*

*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvCustomEditAction*<sup>(145)</sup>

### Description

This action pastes data from the Clipboard (calls *TCustomRichViewEdit.Paste*).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of *TrvAction*.

This action can work with several types of editors, see the comments in the *TrvCustomEditAction*<sup>(145)</sup> topic.

### 1.3.2.7 TrvActionPasteAsText

*TrvActionPasteAsText* is the action for "Edit | Paste as Text" command.

**Unit** *RichViewActions*<sup>(92)</sup>;

### Syntax

```
TrvActionPasteAsText = class (TrvCustomEditAction(145))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvCustomEditAction*<sup>(145)</sup>

### Description

This action pastes text from the Clipboard. If the current text in the editor is Unicode, it pastes Unicode text.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of *TrvAction*.

This action can work with several types of editors, see the comments in the *TrvCustomEditAction*<sup>(145)</sup> topic.

### 1.3.2.8 TrvActionPasteSpecial

TrvActionPasteSpecial is the action for "Edit | Paste Special" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionPasteSpecial = class (TrvActionPaste(135))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvCustomEditAction(145)
TrvActionPaste(135)

```

#### Description

This action displays a dialog containing a list of data formats available in the Clipboard. When the user chooses a format, data are pasted into TCustomRichViewEdit/TSRichViewEdit component in this format.

Supported formats:

- ANSI text (CF\_TEXT)
- Unicode text (CF\_UNICODETEXT)
- Windows bitmap (CF\_BITMAP)
- Windows metafile (CF\_METAFILEPICT)
- URL ("UniformResourceLocator")
- RTF ("Rich Text Format")
- RVF ("RichView Format")
- HTML ("HTML Format")
- Graphic files (CF\_HDROP)
- custom formats (see OnShowing<sup>(140)</sup> and OnCustomPaste<sup>(140)</sup> events).

The action displays only formats listed in in TCustomRichViewEdit.AcceptPasteFormats. **Note:** URL format is not included in AcceptPasteFormats by default.

If GetControlPanel<sup>(336)</sup>.RVFLocalizable<sup>(52)</sup>=True, GetControlPanel<sup>(336)</sup>.RVFFormatTitle<sup>(53)</sup> is used in the list for RVF Format. You can change it to something like "My Application's Format". But in this case you need to provide a localization of this property yourself.

When pasting graphic files and images from URL, the action assigns "file name" property to the inserted images, if *rvoAssignImageFileNames* is included in the Options property of the target editor (you can modify the assigned string in OnAssignImageFileName event of the target editor). If StoreFileNameInItemName<sup>(138)</sup> = True, it also assigns the item text.

If style templates are used (`RichViewEdit.UseStyleTemplates=True`), the action displays options for insertion of RTF and RVF.

Unlike all other actions inherited from `TrvCustomEditAction`<sup>(145)</sup>, this action cannot work with editor types other than `TCustomRichViewEdit` and `TSRichViewEdit`.

### 1.3.2.8.1 Properties

#### In `TrvActionPasteSpecial`

■ `StoreFileNameInItemName`<sup>(138)</sup>

#### Derived from `TrvAction`<sup>(311)</sup>

■ `Control`<sup>(312)</sup>

#### Derived from `TrvCustomAction`<sup>(333)</sup>

■ `Caption`<sup>(334)</sup>

■ `ControlPanel`<sup>(335)</sup>

■ `Disabled`<sup>(335)</sup>

■ `Hint`<sup>(335)</sup>

#### Derived from `TAction`

■ `AutoCheck`

■ `Caption`

■ `Checked`

`DisableIfNoHandler`

■ `Enabled`

■ `GroupIndex`

■ `HelpContext`

■ `HelpKeyword`

■ `HelpType`

■ `Hint`

■ `ImageIndex`

■ `Name`

■ `SecondaryShortCuts`

■ `ShortCut`

■ `Visible`

#### 1.3.2.8.1.1 `TrvActionPasteSpecial.StoreFileNameInItemName`

This property is used when pasting graphic files (CF\_HDROP).

**property** `StoreFileNameInItemName: Boolean;`

If this property is *True*, a path to the graphic file is stored in the document in the item name (item text).

It's not recommended to use this option. The recommended place for storing image file names is *rvesplImageFileName* extra item property. It is assigned if *rvoAssignImageFileNames* is included in the Options property of the target editor.

Paths cannot be stored for bitmaps and metafiles pasted in CF\_BITMAP and CF\_METAFILEPICT formats.

### Default value

*False*

### See also:

- TrvActionInsertPicture.StoreFileNameInItemName<sup>(253)</sup>
- TrvActionItemProperties<sup>(324)</sup> (does not support storing paths in item names)
- RTF reading (does not support storing paths in item names)

## 1.3.2.8.2 Methods

### In TrvActionPasteSpecial

AddFormat<sup>(139)</sup>

### Inherited from TrvCustomAction<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

#### 1.3.2.8.2.1 TrvActionPasteSpecial.AddFormat

Adds a new format in the paste-special dialog.

```
procedure AddFormat(const FormatName: TRVLocString(348); Format: Word);
```

This method must be called only from OnShowing<sup>(140)</sup> event.

#### Parameters:

**FormatName** – format name, how it appears in the dialog.

**Format** – Clipboard format. This parameter can be a registered format or any of the standard Clipboard formats.

Data in the format specified in this method must be available in the Clipboard. If the user will choose this format, OnCustomPaste<sup>(140)</sup> event will occur.

## 1.3.2.8.3 Events

### In TrvActionPasteSpecial

■ OnCanPaste<sup>(139)</sup>

■ OnCustomPaste<sup>(140)</sup>

■ OnShowing<sup>(140)</sup>

#### 1.3.2.8.3.1 TrvActionPasteSpecial.OnCanPaste

This event is used to update Enabled property of the action.

#### type

```
TRVACanPasteEvent = procedure (Sender: TrvActionPasteSpecial;  
    var CanPaste: Boolean) of object;
```

**property** OnCanPaste: TRVACanPasteEvent;

On input, **CanPaste** parameter is the value returned by `TCustomRichViewEdit.CanPaste`.

If you implement pasting in additional formats, assign *True* to **CanPaste** if data in these formats are available in the Clipboard.

#### See also events:

- `OnShowing` <sup>(140)</sup>
- `OnCustomPaste` <sup>(140)</sup>

#### 1.3.2.8.3.2 TrvActionPasteSpecial.OnCustomPaste

Occurs when the user chose a Clipboard format added with `AddFormat` <sup>(139)</sup> method.

##### type

```
TRVACustomPasteEvent = procedure (Sender: TrvActionPasteSpecial;  
    Editor: TCustomRichViewEdit; Format: Word) of object;
```

**property** `OnCustomPaste`: `TRVACustomPasteEvent`;

Paste data in the format **Format** into the editor **Editor**.

#### 1.3.2.8.3.3 TrvActionPasteSpecial.OnShowing

Occurs before the paste-special dialog is shown.

**property** `OnShowing`: `TNotifyEvent`;

In this event, use `AddFormat` <sup>(139)</sup> method to add additional formats in the dialog.

### 1.3.2.9 TrvActionRedo

`TrvActionRedo` is the action for "Edit | Redo" command.

**Unit** `RichViewActions` <sup>(92)</sup>;

#### Syntax

```
TrvActionRedo = class (TrvAction (311))
```

#### Hierarchy

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction  
TAction  
TrvCustomAction (333)  
TrvAction (311)
```

#### Description

This action redoes the last undone editing operation (calls `TCustomRichViewEdit.Redo`).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.



### 1.3.2.10 TrvActionReplace

TrvActionReplace is the action for "Edit | Replace" command

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionReplace = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action displays a TReplaceDialog for searching and replacing a substring in the TCustomRichViewEdit component. If at the moment of execution a single line of text is selected, this text is copied to the replace dialog (as a text to find). If at the moment of execution a find dialog is shown, it is closed.

The search starts from the caret position to the end or to the beginning of the document. If the text is found, it is replaced. If not found, the action's behavior depends on GetControlPanel<sup>(336)</sup>.SearchScope<sup>(53)</sup> property:

- *rvssFromCursor*, the search stops;
- *rvssAskUser*, the action asks user whether to continue from the beginning/end;
- *rvssGlobal*, the search continues from the beginning/end.

If nothing found, a message is displayed.

In Delphi 2009+, or if TNT Controls<sup>(370)</sup> are used, the action searches for Unicode string and replaces it with Unicode string. Otherwise, it works with ANSI strings.

#### See also:

TrvActionFindNext<sup>(134)</sup>

TrvActionFind<sup>(132)</sup>

#### 1.3.2.10.1 Properties

##### In TrvActionReplace

- ActionFind<sup>(142)</sup>
- FindText<sup>(142)</sup>
- ReplaceText<sup>(142)</sup>
- ShowReplaceAllSummary<sup>(143)</sup>

##### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

## Derived from TrvCustomAction <sup>333</sup>

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.2.10.1.1 TrvActionReplace.ActionFind

Links this action to TrvActionFind <sup>132</sup> action.

**property** ActionFind: TrvActionFind <sup>132</sup>;

If this link is not defined, the first TrvActionFind <sup>132</sup> action found in the same form/datamodule is used.

When this action is executed, it calls TrvActionFind.CloseDialog <sup>134</sup>.

### 1.3.2.10.1.2 TrvActionReplace.FindText, ReplaceText

Strings to initialize the replace dialog.

**property** FindText: TRVUnicodeString;

**property** ReplaceText: TRVUnicodeString;

If value of **FindText** is not empty, it is assigned to the replace dialog's FindText when a search is started. Otherwise, the dialog is initialized with text selected in the target editor.

If value of **ReplaceText** is not empty, it is assigned to the replace dialog's ReplaceText when a search is started. Otherwise, the dialog is initialized an empty string.

#### Default values

" (empty string)

### 1.3.2.10.1.3 TrvActionReplace.ShowReplaceAllSummary

Specifies whether the action should display a summary message after completing a "Replace All" command.

**property** ShowReplaceAllSummary: Boolean;

If *True*, a message "*N strings replaced.*" is shown.

**Default value:**

*True*

### 1.3.2.10.2 Methods

#### In TrvActionReplace

CloseDialog<sup>(143)</sup>

#### Inherited from TrvCustomAction<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

### 1.3.2.10.2.1 TrvActionReplace.CloseDialog

Closes the replace dialog (if it is shown).

**procedure** CloseDialog;

### 1.3.2.10.3 Events

#### In TrvActionReplace

■ OnReplaceAllEnd<sup>(143)</sup>

■ OnReplaceAllStart<sup>(143)</sup>

■ OnReplacing<sup>(144)</sup>

### 1.3.2.10.3.1 TrvActionReplace.OnReplaceAllEnd

Occurs when "replace all" operation finishes.

**property** OnReplaceAllEnd: TRVAEditEvent<sup>(384)</sup>;

If GetControlPanel<sup>(336)</sup>.SearchScope<sup>(53)</sup> property is equal to *rvssAskUser* or *rvssGlobal*, this event may occur twice: when the end/beginning of the document is reached, and when all replacements are completed.

The event occurs before showing a replace-all summary message.

### 1.3.2.10.3.2 TrvActionReplace.OnReplaceAllStart

Occurs when "replace all" operation starts.

**property** OnReplaceAllStart: TRVAEditEvent<sup>(384)</sup>;

If GetControlPanel<sup>(336)</sup>.SearchScope<sup>(53)</sup> property is equal to *rvssAskUser* or *rvssGlobal*, this event may occur twice: when the search starts, and when the search continues from the beginning/end of the document.

### 1.3.2.10.3.3 TrvActionReplace.OnReplacing

Occurs before the action replaces one occurrence of the found text with **NewText**.

#### type

```
TRVAReplacingEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit;  
    const NewText: TRVUnicodeString) of object;
```

**property** OnReplacing: TRVAReplacingEvent;

When this event occurs, the found text is selected in **Editor**.

### 1.3.2.11 TrvActionSelectAll

TrvActionSelectAll is the action for "Edit | Select All" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionSelectAll = class (TrvAction(311))
```

#### Hierarchy

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction  
TAction  
TrvCustomAction(333)  
TrvAction(311)
```

#### Description

This action selects the whole document (calls TCustomRichViewEdit.SelectAll).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

### 1.3.2.12 TrvActionUndo

TrvActionUndo is the action for "Edit | Undo" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionUndo = class (TrvCustomEditAction(145))
```

#### Hierarchy

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction
```

*TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>*TrvCustomEditAction* <sup>(145)</sup>**Description**

This action undoes the last editing operation (calls `TCustomRichViewEdit.Undo`).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

This action can work with several types of editors, see the comments in the `TrvCustomEditAction` <sup>(145)</sup> topic.

**1.3.2.13 TrvCustomEditAction**

`TrvCustomEditAction` is a base class for several editing actions.

**Unit** `RichViewActions` <sup>(92)</sup>;

**Syntax**

```
TrvCustomEditAction = class (TrvAction (311))
```

**Hierarchy***TObject**TPersistent**TComponent**TBasicAction**TContainedAction**TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>**Description**

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

This action is not used directly. The following actions are inherited from it:

- `TrvActionUndo` <sup>(144)</sup>
- `TrvActionCut` <sup>(131)</sup>
- `TrvActionCopy` <sup>(131)</sup>
- `TrvActionPaste` <sup>(135)</sup>

All these actions can work not only with `TCustomRichViewEdit` components, but also with another editors.

By default, the list of these editors includes:

- `TCustomEdit` (`TEdit`, `TMemo`, `TRichEdit` and others);
- `TComboBox` (having *csDropDown* or *csSimple* style).

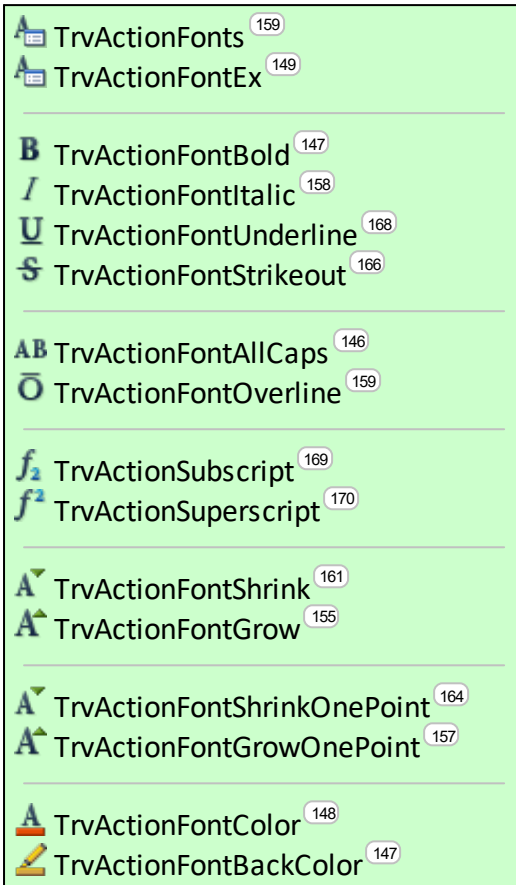
You can implement working with additional editor types by assigning your own function to `RVA_EditorControlFunction` <sup>(389)</sup> variable.

See also `RVA_EditForceDefControl` <sup>(389)</sup> global variable.

### 1.3.3 Font

#### Font Actions

This group of actions includes commands for changing font of selected text.



#### 1.3.3.1 TrvActionFontAllCaps

TrvActionFontAllCaps is the action for "All Capitals" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionFontAllCaps = class (TrvActionFontStyleEx(167))
```

##### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionTextStyles(173)
```

*TrvActionFontStyleEx* <sup>(167)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action converts the selected fragment to upper case characters (toggle). It adds/removes *rvfsAllCaps* in *StyleEx* property for styles of all selected text items, without changing the actual text.

This is a checkbox-like action, its *Checked* property is updated depending on the state of selection (or the current text style).

### See also:

- *TrvActionCharCase* <sup>(130)</sup>

## 1.3.3.2 TrvActionFontBackColor

*TrvActionFontBackColor* is the action for changing background color of the selected text.

**Unit** *RichViewActions* <sup>(92)</sup>;

### Syntax

```
TrvActionFontBackColor = class (TrvActionFontCustomColor (149))
```

### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionCustomColor (318)
TrvActionFontCustomColor (149)

```

### Description

This action does not introduce any new properties and events in addition to properties and events of *TrvActionCustomColor* <sup>(318)</sup>.

On execution, this action changes *BackColor* property of styles of the selected text (*RVStyle.TextStyles[].BackColor*).

See also the list of color changing actions in the topic about *TrvActionCustomColor* <sup>(318)</sup>.

## 1.3.3.3 TrvActionFontBold

*TrvActionFontBold* is the action for "Bold" command.

**Unit** *RichViewActions* <sup>(92)</sup>;

### Syntax

```
TrvActionFontBold = class (TrvActionFontStyle (166))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTextStyles* <sup>(173)</sup>  
*TrvActionFontStyle* <sup>(166)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action makes the selected text bold (or not bold if already bold). It adds/removes *fsBold* in *Style* property for styles of all selected text items.

This is a checkbox-like action, its *Checked* property is updated depending on the state of selection (or the current text style).

### 1.3.3.4 TrvActionFontColor

*TrvActionFontColor* is the action for changing color of the selected text.

**Unit** *RichViewActions* <sup>(92)</sup>;

## Syntax

```
TrvActionFontColor = class (TrvActionFontCustomColor (149))
```

## Hierarchy

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionCustomColor* <sup>(318)</sup>  
*TrvActionFontCustomColor* <sup>(149)</sup>

## Description

This action does not introduce any new properties and events in addition to properties and events of *TrvActionCustomColor* <sup>(318)</sup>. However, it changes the default value of *Color* <sup>(319)</sup> property to *clWindowText*.



On execution, this action changes Color property of styles of the selected text (RVStyle.TextStyles[].Color).

See also the list of color changing actions in the topic about TrvActionCustomColor<sup>(318)</sup>.

### 1.3.3.5 TrvActionFontCustomColor

TrvActionFontCustomColor is a base class for the actions changing color properties of the selected text.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFontCustomColor = class (TrvActionCustomColor(318))
```

#### Hierarchy

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>

#### Description

This action does not introduce any new properties and events in addition to properties and events of TrvActionCustomColor<sup>(318)</sup>.

This action is not used directly. The following actions are inherited from it:

- TrvActionFontColor<sup>(148)</sup>
- TrvActionFontBackColor<sup>(147)</sup>

### 1.3.3.6 TrvActionFontEx

TrvActionFontEx is the action for "Font" command. It uses an advanced font dialog.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFontEx = class (TrvActionFonts(159))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*

*TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>*TrvActionTextStyles* <sup>(173)</sup>*TrvActionFonts* <sup>(159)</sup>

## Description

If `UserInterface` <sup>(161)</sup> = *False*, the action applies its properties to the selected text. The list of properties to apply is specified in `ValidProperties` <sup>(154)</sup>.

If `UserInterface` <sup>(161)</sup> = *True*, the action uses a dialog window. The action performs the following steps:

1. assigns font attributes from the selected text to the action's properties
2. calls `OnShowingDialog` <sup>(155)</sup> event
3. displays a dialog window
4. applies changes made in the dialog window to the action's properties
5. applies the action's properties to the selected text.

For example, if you want to apply *{"Arial", 12, not bold, dotted underline}* to the selected text in `RichViewEdit1`:

```
with rvActionFontEx1 do
begin
  UserInterface (161) := False;
  ValidProperties (154) := [rvfimFontName, rvfimSize, rvfimBold,
    rvfimUnderline, rvfimUnderlineType];
  Font (161).Name := FontName;
  FontSizeDouble (153) := 12*2;
  Font (161).Style := [fsUnderline];
  UnderlineType (154) := rvutDotted;
  ExecuteTarget(RichViewEdit1);
  UserInterface (161) := True;
end;
```

### See also:

- `TrvActionFonts` <sup>(159)</sup>

### 1.3.3.6.1 Properties

#### In `TrvActionFontEx`

- `AutoApplySymbolCharset` <sup>(151)</sup>
  - `BackColor` <sup>(152)</sup>
  - `CharScale` <sup>(152)</sup>
  - `CharSpacing` <sup>(152)</sup>
  - `FontStyleEx` <sup>(153)</sup>
- `PreviewInList` <sup>(153)</sup>
  - `SubSuperScriptType` <sup>(153)</sup>
  - `UnderlineColor` <sup>(154)</sup>
  - `UnderlineType` <sup>(154)</sup>

ValidProperties<sup>(154)</sup>  
VShift<sup>(155)</sup>

### Derived from TrvActionFonts<sup>(159)</sup>

Font<sup>(161)</sup>  
■ UserInterface<sup>(161)</sup>

### Derived from TrvAction<sup>(311)</sup>

■ Control<sup>(312)</sup>

### Derived from TrvCustomAction<sup>(333)</sup>

■ Caption<sup>(334)</sup>  
■ ControlPanel<sup>(335)</sup>  
■ Disabled<sup>(335)</sup>  
■ Hint<sup>(335)</sup>

### Derived from TAction

■ AutoCheck  
■ Caption  
■ Checked  
  DisableIfNoHandler  
■ Enabled  
■ GroupIndex  
■ HelpContext  
■ HelpKeyword  
■ HelpType  
■ Hint  
■ ImageIndex  
■ Name  
■ SecondaryShortCuts  
■ ShortCut  
■ Visible

#### 1.3.3.6.1.1 TrvActionFontEx.AutoApplySymbolCharset

Allows to correct charset for symbol fonts.

**property** AutoApplySymbolCharset: Boolean;

This property works if UserInterface<sup>(161)</sup>=False.

If *True*, then when the action applies a font name without charset (i.e. *rvfimFontName* in ValidProperties<sup>(154)</sup>, *rvfimCharset* not in ValidProperties<sup>(154)</sup>), if this is a symbol font (i.e. if it supports only SYMBOL\_CHARSET), the action applies SYMBOL\_CHARSET to the selection; if this is not a symbol font, it applies DEFAULT\_CHARSET.

#### Default value:

*True*

#### See also:

- TRVFontComboBox<sup>(73)</sup>.AutoCharset<sup>(75)</sup>

#### 1.3.3.6.1.2 TrvActionFontEx.BackgroundColor

Specifies the background color for applying to the selected text.

**property** BackgroundColor: TColor;

If UserInterface<sup>(161)</sup>=False and *rvfimBackgroundColor* in ValidProperties<sup>(154)</sup>, the action applies this property to the selected text.

If UserInterface<sup>(161)</sup>=True, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, *rvfimBackgroundColor* is included in ValidProperties<sup>(154)</sup>, and the action applies this property to the selected text.

This property is applied to the text style's (RVStyle.TextStyles[]) property of the same name.

#### 1.3.3.6.1.3 TrvActionFontEx.CharScale

Specifies the character scale ratio for applying to the selected text.

**property** CharScale: Integer;

If UserInterface<sup>(161)</sup>=False and *rvfimCharScale* in ValidProperties<sup>(154)</sup>, the action applies this property to the selected text.

If UserInterface<sup>(161)</sup>=True, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, *rvfimCharScale* is included in ValidProperties<sup>(154)</sup>, and the action applies this property to the selected text.

This property is applied to the text style's (RVStyle.TextStyles[]) property of the same name.

#### 1.3.3.6.1.4 TrvActionFontEx.CharSpacing

Specifies the character spacing for applying to the selected text.

**property** CharSpacing: TRVStyleLength;

If UserInterface<sup>(161)</sup>=False and *rvfimCharSpacing* in ValidProperties<sup>(154)</sup>, the action applies this property to the selected text.

If UserInterface<sup>(161)</sup>=True, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, *rvfimCharSpacing* is included in ValidProperties<sup>(154)</sup>, and the action applies this property to the selected text.

This property is applied to the text style's (RVStyle.TextStyles[]) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in RVStyle.Units).

#### 1.3.3.6.1.5 TrvActionFontEx.FontSizeDouble

Specifies the font size (in half-points) for applying to the selected text.

**property** FontSizeDouble: Integer;

If `UserInterface(161)=False` and `rvfimSize` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161)=True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimSize` is included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) property `SizeDouble`.

When you assign this property, `Font(161).Size` is updated.

#### 1.3.3.6.1.6 TrvActionFontEx.FontStyleEx

Specifies the additional text styles (overline and/or all-capitals) for applying to the selected text.

**property** FontStyleEx: TRVFontStyles;

If `UserInterface(161)=False` and `rvfimOverline` or `rvfimAllCaps` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161)=True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimOverline` and/or `rvfimAllCaps` are included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) `StyleEx` property.

#### 1.3.3.6.1.7 TrvActionFontEx.PreviewInList

Specifies whether the font dialog displays a font preview in the list of font names.

**property** PreviewInList: Boolean;

**Default value**

*True*

#### 1.3.3.6.1.8 TrvActionFontEx.SubSuperScriptType

Specifies the sub/superscript mode for applying to the selected text.

**property** SubSuperScriptType: TRVSubSuperScriptType;

If `UserInterface(161)=False` and `rvfimSubSuperScriptType` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161)=True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimSubSuperScriptType` is included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) property of the same name.

### 1.3.3.6.1.9 TrvActionFontEx.UnderlineColor

Specifies the underline color for applying to the selected text.

**property** UnderlineColor: TColor;

If `UserInterface(161) = False` and `rvfimUnderlineColor` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161) = True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimUnderlineColor` is included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) property of the same name.

### 1.3.3.6.1.10 TrvActionFontEx.UnderlineType

Specifies the underline type for applying to the selected text.

**property** UnderlineType: TRVUnderlineType;

If `UserInterface(161) = False` and `rvfimUnderlineType` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161) = True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimUnderlineType` is included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) property of the same name.

### 1.3.3.6.1.11 TrvActionFontEx.ValidProperties

Defines a set of properties for applying to the selected text.

**type**

```
TRVFontInfoMainProperty = (rvfimFontName, rvfimSize, rvfimCharset,
    rvfimBold, rvfimItalic, rvfimUnderline, rvfimStrikeout,
    rvfimOverline, rvfimAllCaps, rvfimVShift, rvfimColor, rvfimBackColor,
    rvfimCharScale, rvfimCharSpacing, rvfimSubSuperScriptType,
    rvfimUnderlineType, rvfimUnderlineColor);
TRVFontInfoMainProperties = set of TRVFontInfoMainProperty;
```

**property** ValidProperties: TRVFontInfoMainProperties;

If `UserInterface(161) = False`, assign a subset of properties (for applying) to `ValidProperties`.

If `UserInterface(161) = True`, this property is maintained automatically. After displaying a font dialog, this property contains a set of properties corresponding to non-blank fields in the dialog.

For properties-sets (`Font(161).Style` and `FontStyleEx(153)`) each value in the set is represented by one value in this property. For example:

- if `rvfimOverline` in `ValidProperties`, `rvfsOverline` in `FontStyleEx(153)`, the selected text will be overlined;
- if `rvfimOverline` in `ValidProperties`, `rvfsOverline` not in `FontStyleEx(153)`, overlines will be removed from the selected text;
- if `rvfimOverline` not in `ValidProperties`, overlines in the selected text will not be affected.

#### 1.3.3.6.1.12 TrvActionFontEx.VShift

Specifies the vertical shift for applying to the selected text.

**property** VShift: Integer;

If `UserInterface(161)=False` and `rvfimVShift` in `ValidProperties(154)`, the action applies this property to the selected text.

If `UserInterface(161)=True`, the action displays a font dialog (initialized with font properties of the selected text); if the user changed value of this property in the dialog, `rvfimVShift` is included in `ValidProperties(154)`, and the action applies this property to the selected text.

This property is applied to the text style's (`RVStyle.TextStyles[]`) property of the same name.

#### 1.3.3.6.2 Events

##### In TrvActionFontEx

##### ■ OnShowingDialog<sup>(155)</sup>

#### 1.3.3.6.2.1 TrvActionFontEx.OnShowingDialog

Occurs before showing a dialog window.

**property** OnShowingDialog: TNotifyEvent;

This event occurs if `UserInterface(161)=True`.

It is called when properties of the action are already assigned from attributes of the selected text of the target editor, but the dialog is not initialized yet.

In this event, you can modify properties of the action, so modified properties will be applied to the dialog.

You can use this event to initialize a dialog with predefined values, instead of taking them from a selected text.

### 1.3.3.7 TrvActionFontGrow

TrvActionFontGrow is the action for "Grow Font" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionFontGrow = **class** (TrvActionFontShrinkGrow<sup>(163)</sup>)

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction

```

*TrvCustomAction*<sup>(333)</sup>*TrvAction*<sup>(311)</sup>*TrvActionTextStyles*<sup>(173)</sup>*TrvActionFontShrinkGrow*<sup>(163)</sup>

## Description

The action increases font size of the selected text by Percent<sup>(164)</sup> (until it reaches MaxSize<sup>(157)</sup>).

See the list of font size changing actions in the topic about TrvActionFontShrinkGrow<sup>(163)</sup>.

### 1.3.3.7.1 Properties

#### In TrvActionFontGrow

---

■ MaxSize<sup>(157)</sup>

#### Derived from TrvActionFontShrinkGrow<sup>(163)</sup>

---

■ Percent<sup>(164)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

■ Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

■ Caption<sup>(334)</sup>

■ ControlPanel<sup>(335)</sup>

■ Disabled<sup>(335)</sup>

■ Hint<sup>(335)</sup>

#### Derived from TAction

---

■ AutoCheck

■ Caption

■ Checked

    DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible



#### 1.3.3.7.1.1 TrvActionFontGrow.MaxSize

Defines maximal font size.

**property** MaxSize: Integer;

The action does not increase font size above this value.

**Default value:**

100

### 1.3.3.8 TrvActionFontGrowOnePoint

TrvActionFontGrowOnePoint is the action for "Grow Font by One Point" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionFontGrowOnePoint = **class** (TrvActionTextStyles<sup>(173)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTextStyles*<sup>(173)</sup>

**Description**

The action increases font size of the selected text by 1 (until it reaches MaxSize<sup>(158)</sup>).

See the list of font size changing actions in the topic about TrvActionFontShrinkGrow<sup>(163)</sup>.

#### 1.3.3.8.1 Properties

##### In TrvActionFontGrowOnePoint

■ MaxSize<sup>(158)</sup>

##### Derived from TrvAction<sup>(311)</sup>

■ Control<sup>(312)</sup>

##### Derived from TrvCustomAction<sup>(333)</sup>

■ Caption<sup>(334)</sup>

■ ControlPanel<sup>(335)</sup>

■ Disabled<sup>(335)</sup>

■ Hint<sup>(335)</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.3.8.1.1 TrvActionFontGrowOnePoint.MaxSize

Defines maximal font size.

**property** MaxSize: Integer;

The action does not increase font size above this value.

**Default value:**

100

### 1.3.3.9 TrvActionFontItalic

TrvActionFontItalic is the action for "Italic" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionFontItalic = **class** (TrvActionFontStyle<sup>(166)</sup>)

#### Hierarchy

TObject  
 TPersistent  
 TComponent  
 TBasicAction  
 TContainedAction  
 TCustomAction  
 TAction  
 TrvCustomAction<sup>(333)</sup>  
 TrvAction<sup>(311)</sup>  
 TrvActionTextStyles<sup>(173)</sup>  
 TrvActionFontStyle<sup>(166)</sup>

#### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action makes the selected text italic (or not italic if already italic). It adds/removes *fsItalic* in Style property for styles of all selected text items.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

### 1.3.3.10 TrvActionFontOverline

TrvActionFontOverline is the action for "Overline" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionFontOverline = class (TrvActionFontStyleEx (167))
```

#### Hierarchy

TObject  
 TPersistent  
 TComponent  
 TBasicAction  
 TContainedAction  
 TCustomAction  
 TAction  
 TrvCustomAction <sup>(333)</sup>  
 TrvAction <sup>(311)</sup>  
 TrvActionTextStyles <sup>(173)</sup>  
 TrvActionFontStyleEx <sup>(167)</sup>

#### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action overlines the selected text (or removes overline, if already overlined). It adds/removes *rvfsOverline* in StyleEx property for styles of all selected text items.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

### 1.3.3.11 TrvActionFonts

TrvActionFonts is the action for "Font" command. It uses a standard font dialog (TFontDialog).

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionFonts = class (TrvActionTextStyles (173))
```

#### Hierarchy

TObject  
 TPersistent  
 TComponent  
 TBasicAction

*TContainedAction**TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>*TrvActionTextStyles* <sup>(173)</sup>

## Description

If `UserInterface` <sup>(161)</sup> = *False*, the action applies `Font` <sup>(161)</sup> to the selected text.

If `UserInterface` <sup>(161)</sup> = *True*, the action displays a `TFontDialog` (initialized with font properties of the current text style), assigns the chosen font to `Font` <sup>(161)</sup>, and then applies `Font` <sup>(161)</sup> to the selected text.

It's recommended to use `TrvActionFontEx` <sup>(149)</sup> instead of this action because of two reasons:

- `TrvActionFonts` applies all font properties, it's not possible to leave some fields blank in `TFontDialog`. In `TrvActionFontEx` <sup>(149)</sup>, if you open a dialog and press OK without changing anything, nothing will be changed in the selected text; this is not so for this action;
- `TrvActionFontEx` <sup>(149)</sup> allows defining much more text properties.

### 1.3.3.11.1 Properties

#### In `TrvActionFonts`

---

`Font` <sup>(161)</sup>■ `UserInterface` <sup>(161)</sup>

#### Derived from `TrvAction` <sup>(311)</sup>

---

■ `Control` <sup>(312)</sup>

#### Derived from `TrvCustomAction` <sup>(333)</sup>

---

■ `Caption` <sup>(334)</sup>■ `ControlPanel` <sup>(335)</sup>■ `Disabled` <sup>(335)</sup>■ `Hint` <sup>(335)</sup>

#### Derived from `TAction`

---

■ `AutoCheck`■ `Caption`■ `Checked``DisableIfNoHandler`■ `Enabled`■ `GroupIndex`■ `HelpContext`■ `HelpKeyword`■ `HelpType`■ `Hint`■ `ImageIndex`■ `Name`■ `SecondaryShortCuts`

- ShortCut
- Visible

#### 1.3.3.11.1.1 TrvActionFonts.Font

Specifies the font for applying to the selected text.

##### type

```
TRVAFont = class(TFont);
```

**property** Font: TRVAFont;

If `UserInterface(161)=False`, the action applies this property to the selected text. For `TrvActionFontEx(149)`, only properties specified in `ValidProperties(154)` are applied (a subset of *rvfimFontName*, *rvfimSize*, *rvfimCharset*, *rvfimBold*, *rvfimItalic*, *rvfimUnderline*, *rvfimStrikeout*, *rvfimColor*).

If `UserInterface(161)=True`, the action displays a font dialog, assigns the chosen font to this property, and then applies this property to the selected text.

This property is applied to the following properties of text style (`RVStyle.TextStyles[]`):

- FontName
- Size (in `TrvActionFontEx(149)`, it is overridden by `FontSizeDouble(153)`; when you assign Size, the action's `FontSizeDouble(153)` is changed accordingly)
- Charset
- Style
- Color

#### 1.3.3.11.1.2 TrvActionFonts.UserInterface

Allows working with or without the user interaction.

**property** UserInterface: Boolean;

If `UserInterface=False`, the action applies its properties to the selected text.

If `UserInterface=True`, the action displays a font dialog, assigns changes to the action's properties, and then applies these properties to the selected text.

##### Default value:

*True*

### 1.3.3.12 TrvActionFontShrink

`TrvActionFontShrink` is the action for "Shrink Font" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionFontShrink = class(TrvActionFontShrinkGrow(163))
```

##### Hierarchy

```
TObject
TPersistent
TComponent
TBasicAction
```

*TContainedAction**TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>*TrvActionTextStyles* <sup>(173)</sup>*TrvActionFontShrinkGrow* <sup>(163)</sup>**Description**

The action decreases font size of the selected text by Percent <sup>(164)</sup> (until it reaches MinSize <sup>(163)</sup>).

See the list of font size changing actions in the topic about TrvActionFontShrinkGrow <sup>(163)</sup>.

**1.3.3.12.1 Properties****In TrvActionFontShrink**

---

- MinSize <sup>(163)</sup>

**Derived from TrvActionFontShrinkGrow** <sup>(163)</sup>

---

- Percent <sup>(164)</sup>

**Derived from TrvAction** <sup>(311)</sup>

---

- Control <sup>(312)</sup>

**Derived from TrvCustomAction** <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

**Derived from TAction**

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.3.12.1 TrvActionFontShrink.MinSize

Defines minimal font size.

**property** MinSize: Integer;

The action does not decrease font size below this value.

**Default value:**

1

#### 1.3.3.13 TrvActionFontShrinkGrow

TrvActionFontShrinkGrow is a base class for the actions changing font size of the selected text by the specified percent.

**Unit** RichViewActions <sup>(92)</sup>;

**Syntax**

TrvActionFontShrinkGrow = **class** (TrvActionTextStyles <sup>(173)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTextStyles* <sup>(173)</sup>

**Description**

The action changes Size property of text styles for the selected text items (RVStyle.TextStyles[].Size).

This action is not used directly. The following actions are inherited from it:

- TrvActionFontShrink <sup>(161)</sup>
- TrvActionFontGrow <sup>(155)</sup>

**See also:**

- TrvActionFontShrinkOnePoint <sup>(164)</sup>
- TrvActionFontGrowOnePoint <sup>(157)</sup>

#### 1.3.3.13.1 Properties

##### In TrvActionFontShrinkGrow

■ Percent <sup>(164)</sup>

##### Derived from TrvAction <sup>(311)</sup>

■ Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>333</sup>

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.3.13.1.1 TrvActionFontShrinkGrow.Percent

Defines the percent for increasing or decreasing font size.

**property** Percent: Integer;

**Default value:**

10

### 1.3.3.14 TrvActionFontShrinkOnePoint

TrvActionFontShrinkOnePoint is the action for "Shrink Font by One Point" command.

**Unit** RichViewActions <sup>92</sup>;

#### Syntax

TrvActionFontShrinkOnePoint = **class** (TrvActionTextStyles <sup>173</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>333</sup>



*TrvAction* <sup>(311)</sup>*TrvActionTextStyles* <sup>(173)</sup>**Description**

The action decreases font size of the selected text by 1 (until it reaches *MinSize* <sup>(165)</sup>).

See the list of font size changing actions in the topic about *TrvActionFontShrinkGrow* <sup>(163)</sup>.

**1.3.3.14.1 Properties****In TrvActionFontShrinkOnePoint**

- *MinSize* <sup>(165)</sup>

**Derived from TrvAction** <sup>(311)</sup>

- *Control* <sup>(312)</sup>

**Derived from TrvCustomAction** <sup>(333)</sup>

- *Caption* <sup>(334)</sup>
- *ControlPanel* <sup>(335)</sup>
- *Disabled* <sup>(335)</sup>
- *Hint* <sup>(335)</sup>

**Derived from TAction**

- *AutoCheck*
- *Caption*
- *Checked*
- *DisableIfNoHandler*
- *Enabled*
- *GroupIndex*
- *HelpContext*
- *HelpKeyword*
- *HelpType*
- *Hint*
- *ImageIndex*
- *Name*
- *SecondaryShortCuts*
- *ShortCut*
- *Visible*

**1.3.3.14.1.1 TrvActionFontShrinkOnePoint.MinSize**

Defines minimal font size.

**property** *MinSize*: Integer;

The action does not decrease font size below this value.

**Default value:**

1

### 1.3.3.15 TrvActionFontStrikeout

TrvActionFontStrikeout is the action for "Strikeout" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFontStrikeout = class (TrvActionFontStyle(166))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTextStyles*<sup>(173)</sup>  
*TrvActionFontStyle*<sup>(166)</sup>

#### Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This action strikes through the selected text (toggle). It adds/removes *fsStrikeOut* in Style property for styles of all selected text items.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

### 1.3.3.16 TrvActionFontStyle

TrvActionFontStyle is a base class for the actions changing Style property of text styles for the selected text (RVStyle.TextStyles[].Style)

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFontStyle = class (TrvActionTextStyles(173))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTextStyles*<sup>(173)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action is not used directly. The following actions are inherited from it:

- TrvActionFontBold <sup>(147)</sup>
- TrvActionFontItalic <sup>(158)</sup>
- TrvActionFontUnderline <sup>(168)</sup>
- TrvActionFontStrikeout <sup>(166)</sup>

These actions change one option in the Style property of styles of text in the selected fragment of TCustomRichViewEdit (for example, *fsBold*)

These actions are checked (Checked=*True*) in the following cases:

- if several table cells are selected, all text items in these cells have the required option included in Style (for example, all text is bold);
- otherwise, if the current text style (TCustomRichViewEdit.CurTextStyleNo) has this option included in Style (for example, it is bold),

### 1.3.3.17 TrvActionFontStyleEx

TrvActionFontStyle is a base class for the actions changing StyleEx property of text styles for the selected text (RVStyle.TextStyles[].StyleEx)

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionFontStyleEx = class (TrvActionTextStyles (173))
```

## Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionTextStyles (173)
```

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action is not used directly. The following actions are inherited from it:

- TrvActionFontAllCaps <sup>(146)</sup>
- TrvActionFontOverline <sup>(159)</sup>

These actions change one option in the `StyleEx` property of styles of text in the selected fragment of `TCustomRichViewEdit` (for example, *rvfsOverline*)

These actions are checked (`Checked=True`) in the following cases:

- if several table cells are selected, all text items in these cells have the required option included in `StyleEx` (for example, all text is overlined);
- otherwise, if the current text style (`TCustomRichViewEdit.CurTextStyleNo`) has this option included in `StyleEx` (for example, it is overlined),

### 1.3.3.18 TrvActionFontUnderline

`TrvActionFontUnderline` is the action for "Underline" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFontUnderline = class (TrvActionFontStyle(166))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionTextStyles(173)
TrvActionFontStyle(166)
```

#### Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of `TrvAction`.

This action underlines the selected text (or removes underline, if already underlined). It adds/removes *fsUnderline* in `Style` property for styles of all selected text items. This action changes `UnderlineType` property of text styles to *rvutNormal*. This action does not change underline color.

This is a checkbox-like action, its `Checked` property is updated depending on the state of selection (or the current text style).

### 1.3.3.19 TrvActionSSScript

`TrvActionSSScript` is a base class for the subscript and the superscript actions.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTextStyles = class (TrvActionTextStyles173)
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>  
*TrvActionTextStyles*<sup>173</sup>

### Description

This action does not introduce any new properties in addition to properties<sup>312</sup> of TrvAction.

This action changes SubSuperScriptType property for styles of the selected text (RVStyle.TextStyles[].SubSuperScriptType).

This action is not used directly. The following actions are inherited from it:

- TrvActionSubscript<sup>169</sup>
- TrvActionSuperscript<sup>170</sup>

## 1.3.3.20 TrvActionSubscript

TrvActionSubscript is the action for "Subscript" command.

**Unit** RichViewActions<sup>92</sup>;

### Syntax

```
TrvActionSubscript = class (TrvActionSSScript168)
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>  
*TrvActionTextStyles*<sup>173</sup>  
*TrvActionSSScript*<sup>168</sup>

### Description

This action does not introduce any new properties in addition to properties<sup>312</sup> of TrvAction.

The action makes the selected text subscript (or normal, if already subscript).

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

### 1.3.3.21 TrvActionSuperscript

TrvActionSuperscript is the action for "Superscript" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionSuperscript = class (TrvActionSSScript(168))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTextStyles*<sup>(173)</sup>  
*TrvActionSSScript*<sup>(168)</sup>

#### Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

The action makes the selected text superscript (or normal, if already superscript).

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

### 1.3.3.22 TrvActionTextBiDi

TrvActionTextBiDi is a base class for the actions changing the default text flow direction in the selected text.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTextBiDi = class (TrvActionTextStyles(173))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

*TrvActionTextStyles* <sup>(173)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action changes *BiDiMode* property for styles of the selected text (*RVStyle.TextStyles[]BiDiMode*).

This action is not used directly. The following actions are inherited from it:

- *TrvActionTextLTR* <sup>(171)</sup>
- *TrvActionTextRTL* <sup>(172)</sup>

If you use these actions, it's recommended to set *TCustomRichViewEdit.BiDiMode* to *rvbdLeftToRight* or *rvbdRightToLeft*.

Text flow direction specified in text styles has higher priority than text flow direction specified in paragraph styles.

### See also:

- *TrvActionParaBiDi* <sup>(191)</sup>
- *TrvActionParaLTR* <sup>(208)</sup>
- *TrvActionParaRTL* <sup>(210)</sup>

## 1.3.3.23 TrvActionTextLTR

*TrvActionTextLTR* is the action for "Left to Right Text" command.

**Unit** *RichViewActions* <sup>(92)</sup> ;

### Syntax

```
TrvActionTextLTR = class (TrvActionTextBiDi (170))
```

### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionTextStyles (173)
TrvActionTextBiDi (170)
```

### Description

This action changes the default text direction in the selected text to left-to-right. If you use this action, it's recommended to set *TCustomRichViewEdit.BiDiMode* to *rvbdLeftToRight* or *rvbdRightToLeft*.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action changes BiDiMode property for styles of the selected text (RVStyle.TextStyles[].BiDiMode) to *rvbdLeftToRight*.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

#### See also:

- TrvActionTextRTL <sup>172</sup>
- TrvActionParaLTR <sup>208</sup>
- TrvActionParaRTL <sup>210</sup>

### 1.3.3.24 TrvActionTextRTL

TrvActionTextRTL is the action for "Right to Left Text" command.

**Unit** RichViewActions <sup>92</sup>;

#### Syntax

```
TrvActionTextRTL = class (TrvActionTextBiDi 170)
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>333</sup>  
*TrvAction* <sup>311</sup>  
*TrvActionTextStyles* <sup>173</sup>  
*TrvActionTextBiDi* <sup>170</sup>

#### Description

This action changes the default text direction in the selected text to right-to-left. If you use this action, it's recommended to set TCustomRichViewEdit.BiDiMode to *rvbdLeftToRight* or *rvbdRightToLeft*.

This action does not introduce any new properties in addition to properties <sup>312</sup> of TrvAction.

This action changes BiDiMode property for styles of the selected text (RVStyle.TextStyles[].BiDiMode) to *rvbdRightToLeft*.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current text style).

#### See also:

- TrvActionTextLTR <sup>171</sup>
- TrvActionParaLTR <sup>208</sup>
- TrvActionParaRTL <sup>210</sup>



### 1.3.3.25 TrvActionTextStyles

TrvActionTextStyles is a base class for:

- the actions applying changes to text style of the selected fragment of TCustomRichViewEdit component,
- TrvActionInsertHyperlink<sup>(232)</sup>.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTextStyles = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*Action*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

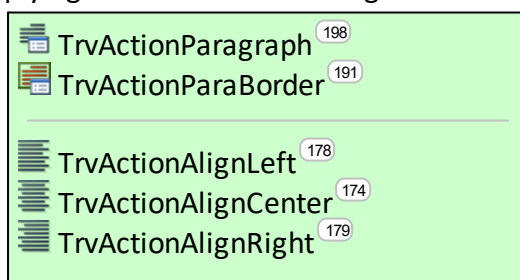
This action is not used directly. This action has the following direct descendants:

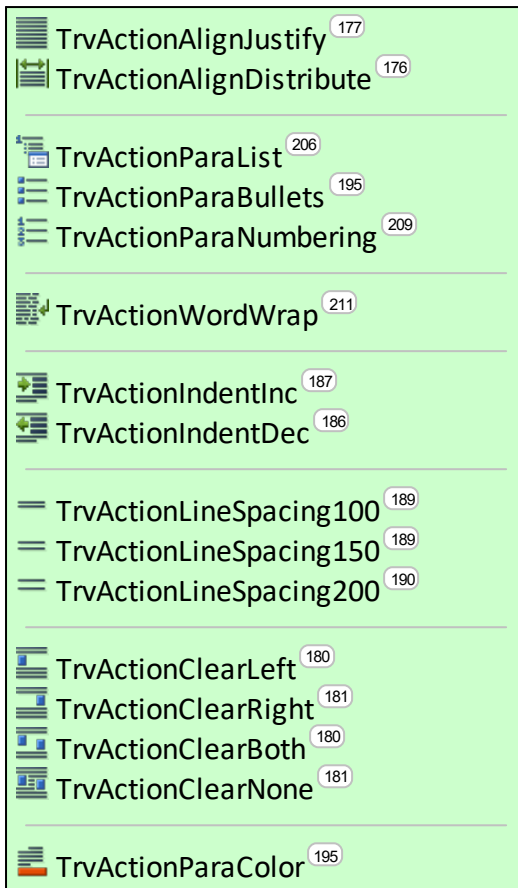
- TrvActionFonts<sup>(159)</sup>
- TrvActionFontStyle<sup>(166)</sup>
- TrvActionFontStyleEx<sup>(167)</sup>
- TrvActionSSScript<sup>(168)</sup>
- TrvActionTextBiDi<sup>(170)</sup>
- TrvActionFontShrinkGrow<sup>(163)</sup>
- TrvActionFontShrinkOnePoint<sup>(164)</sup>
- TrvActionFontGrowOnePoint<sup>(157)</sup>
- TrvActionInsertHyperlink<sup>(232)</sup>

## 1.3.4 Paragraph

### Paragraph Actions

This group of actions includes commands for changing attributes of selected paragraphs, and for applying bullets and numbering.





### 1.3.4.1 TrvActionAlignCenter

TrvActionAlignCenter is the action for "Align Center" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionAlignCenter = class (TrvActionAlignment (178))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionAlignment* <sup>(178)</sup>

#### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action aligns the selected paragraphs to the center.

This action changes Alignment property for styles of the selected paragraphs (RVStyle.ParaStyles[].Alignment) to *rvaCenter*.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style.

### 1.3.4.2 TrvActionAlignCustomJustify

TrvActionAlignCustomJustify is a base class for the actions changing alignment of selected paragraphs to the both left and right sides.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionAlignCustomJustify = class (TrvActionAlignment (178))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionAlignment* <sup>(178)</sup>

#### Description

##### Properties:

LastLineAlignment <sup>(176)</sup> is applied to the selected paragraphs, if UseLastLineAlignment <sup>(176)</sup> = *True*.

This action is not used directly. The following actions are inherited from it:

- TrvActionAlignLeft <sup>(178)</sup>
- TrvActionAlignRight <sup>(179)</sup>
- TrvActionAlignCenter <sup>(174)</sup>
- TrvActionAlignJustify <sup>(177)</sup>

#### 1.3.4.2.1 Properties

##### In TrvActionAlignCustomJustify

- LastLineAlignment <sup>(176)</sup>
- UseLastLineAlignment <sup>(176)</sup>

##### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>333</sup>

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.4.2.1.1 TrvActionAlignCustomJustify.LastLineAlignment, UseLastLineAlignment

Defines the alignment applied to the last line of paragraphs.

```
property LastLineAlignment: TRVLastLineAlignment;
property UseLastLineAlignment: Boolean;
```

If **UseLastLineAlignment**=*True*, **LastLineAlignment** is applied to the selected paragraphs.

**Default values:**

- **UseLastLineAlignment**:*True*
- **LastLineAlignment**:
  - in TrvActionAlignJustify <sup>177</sup>: *rvllaDefault*
  - in TrvActionAlignDistribute: *rvllaJustify*

### 1.3.4.3 TrvActionAlignDistribute

TrvActionAlignDistribute is the action for "Distribute" command.

**Unit** RichViewActions <sup>92</sup>;

**Syntax**

```
TrvActionAlignDistribute = class (TrvActionAlignCustomJustify 175)
```

**Hierarchy**

*TObject*  
*TPersistent*

*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionAlignment* <sup>(178)</sup>  
*TrvActionAlignCustomJustify* <sup>(175)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(175)</sup> of *TrvActionAlignCustomJustify*.

This action aligns the selected paragraphs both to the right and to the left sides by adding space between all characters.

This action changes Alignment property for styles of the selected paragraphs (*RVStyle.ParaStyles[]*.Alignment) to *rvaDistribute*. Additionally, it may change LastLineAlignment property to LastLineAlignment <sup>(176)</sup>.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style.

### 1.3.4.4 TrvActionAlignJustify

*TrvActionAlignJustify* is the action for "Justify" command.

**Unit** RichViewActions <sup>(92)</sup>;

### Syntax

```
TrvActionAlignJustify = class (TrvActionAlignCustomJustify (175))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionAlignment* <sup>(178)</sup>  
*TrvActionAlignCustomJustify* <sup>(175)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(175)</sup> of *TrvActionAlignCustomJustify*.

This action aligns the selected paragraphs both to the right and to the left sides by adding space between words.

This action changes Alignment property for styles of the selected paragraphs (RVStyle.ParaStyles[].Alignment) to *rvaJustify*. Additionally, it may change LastLineAlignment property to LastLineAlignment<sup>(176)</sup>.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style.

### 1.3.4.5 TrvActionAlignLeft

TrvActionAlignLeft is the action for "Align Left" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionAlignLeft = class (TrvActionAlignment(178))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionParaStyles(210)
TrvActionAlignment(178)

```

#### Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This action aligns the selected paragraphs to the left.

This action changes Alignment property for styles of the selected paragraphs (RVStyle.ParaStyles[].Alignment) to *rvaLeft*.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style.

### 1.3.4.6 TrvActionAlignment

TrvActionAlignment is a base class for the actions changing alignment of the selected paragraphs.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionAlignment = class (TrvActionParaStyles(210))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action is not used directly. The following actions are inherited from it:

- *TrvActionAlignLeft* <sup>(178)</sup>
- *TrvActionAlignRight* <sup>(179)</sup>
- *TrvActionAlignCenter* <sup>(174)</sup>
- *TrvActionAlignJustify* <sup>(177)</sup>
- *TrvActionAlignDistribute* <sup>(176)</sup>

### 1.3.4.7 TrvActionAlignRight

*TrvActionAlignRight* is the action for "Align Right" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionAlignRight = class (TrvActionAlignment (178))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionAlignment* <sup>(178)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action aligns the selected paragraphs to the right side.

This action changes Alignment property for styles of the selected paragraphs (*RVStyle.ParaStyles[]*.Alignment) to *rvaRight*.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style.

### 1.3.4.8 TrvActionClearBoth

TrvActionClearBoth is the action for "Clear Text Flow at Both Sides"

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionClearBoth = class (TrvActionClearTextFlow(182))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionClearTextFlow*<sup>(182)</sup>

#### Description

This action does not allow the selected paragraphs flowing around left- and right-aligned pictures. The selected paragraphs will be placed below side-aligned pictures.

This actions calls RichViewEdit.ClearTextFlow(*True*, *True*).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

### 1.3.4.9 TrvActionClearLeft

TrvActionClearLeft is the action for "Clear Text Flow at Left Side"

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionClearLeft = class (TrvActionClearTextFlow(182))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionClearTextFlow*<sup>(182)</sup>



## Description

This action does not allow the selected paragraphs flowing around left-aligned pictures. The selected paragraphs will be placed below left-aligned pictures. Text flow around right-aligned pictures is allowed.

This actions calls `RichViewEdit.ClearTextFlow(True, False)`.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

### 1.3.4.10 TrvActionClearNone

`TrvActionClearNone` is the action for "Normal Text Flow"

**Unit** `RichViewActions` <sup>(92)</sup>;

## Syntax

```
TrvActionClearNone = class (TrvActionClearTextFlow (182))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionClearTextFlow* <sup>(182)</sup>

## Description

This action allows the selected paragraphs flowing around left- and right-aligned pictures.

This actions calls `RichViewEdit.ClearTextFlow(False, False)`.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

### 1.3.4.11 TrvActionClearRight

`TrvActionClearRight` is the action for "Clear Text Flow at Right Side"

**Unit** `RichViewActions` <sup>(92)</sup>;

## Syntax

```
TrvActionClearRight = class (TrvActionClearTextFlow (182))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*

*TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>*TrvActionClearTextFlow* <sup>(182)</sup>

### Description

This action does not allow the selected paragraphs flowing around right-aligned pictures. The selected paragraphs will be placed below right-aligned pictures. Text flow around left-aligned pictures is allowed.

This actions calls `RichViewEdit.ClearTextFlow(False, True)`.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

### 1.3.4.12 TrvActionClearTextFlow

`TrvActionClearTextFlow` is a base class for the actions clearing a text flow around left- and right-aligned pictures.

**Unit** `RichViewActions` <sup>(92)</sup>;

### Syntax

```
TrvActionClearTextFlow = class (TrvAction (311))
```

### Hierarchy

*TObject**TPersistent**TComponent**TBasicAction**TContainedAction**TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

This action is not used directly. The following actions are inherited from it:

- `TrvActionClearLeft` <sup>(180)</sup>
- `TrvActionClearRight` <sup>(181)</sup>
- `TrvActionClearBoth` <sup>(180)</sup>
- `TrvActionClearNone` <sup>(181)</sup>

### 1.3.4.13 TrvActionCustomParaListSwitcher

`TrvActionCustomParaListSwitcher` is a base class for the actions toggling paragraphs' bullets or numbering on/off.

**Unit** `RichViewActions` <sup>(92)</sup>;

## Syntax

```
TrvActionCustomParaListSwitcher = class (TrvAction311)
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>

## Description

This action is checked when the current paragraph has a list marker with properties identical to the properties of ListLevels<sup>184</sup>. In this case, applying the action removes bullets and numbering in all selected paragraphs. Otherwise, applying this action applies a list style with levels specified in ListLevels<sup>184</sup> to all selected paragraphs. The 0th list level is applied to paragraphs that did not have list markers; for other paragraphs, level indices are not changed.

This action is not used directly. The following actions are inherited from it:

- TrvActionParaBullets<sup>195</sup>
- TrvActionParaNumbering<sup>209</sup>

### 1.3.4.13.1 Properties

#### In TrvActionCustomParaListSwitcher

- IndentStep<sup>184</sup>
- ListLevels<sup>184</sup>

#### Derived from TrvAction<sup>311</sup>

- Control<sup>312</sup>

#### Derived from TrvCustomAction<sup>333</sup>

- Caption<sup>334</sup>
- ControlPanel<sup>335</sup>
- Disabled<sup>335</sup>
- Hint<sup>335</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex

- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.4.13.1.1 TrvActionCustomParaListSwitcher.IndentStep

Defines the step for increasing LeftIndent and MarkerIndent of ListLevels <sup>(184)</sup>.

**property** IndentStep: TRVStyleLength;

This value is measured in GetControlPanel <sup>(336)</sup>.UnitsProgram <sup>(55)</sup>.

Assigning value to this property resets ListLevels <sup>(184)</sup> to default value.

**Default value:**

24

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

**See also:**

- TrvActionIndent <sup>(184)</sup>.IndentStep <sup>(186)</sup>
- TrvActionParaList <sup>(206)</sup>.IndentStep <sup>(207)</sup>

#### 1.3.4.13.1.2 TrvActionCustomParaListSwitcher.ListLevels

Specifies list levels for applying to the selected paragraphs.

**property** ListLevels: TRVListLevelCollection;

Subproperties of this property are measured in GetControlPanel <sup>(336)</sup>.UnitsProgram <sup>(55)</sup>.

**Default value for TrvActionParaBullets <sup>(195)</sup>:**

9 levels, all with ListType=*rvlstBullet*, FirstIndent=0, MarkerIndent is increased by IndentStep <sup>(184)</sup>\*2 starting from 0, LeftIndent is increased by IndentStep <sup>(184)</sup>\*2 starting from IndentStep <sup>(184)</sup>, Font.Size=12, disk/circle/square cycled 3 times.

**Default value for TrvActionParaNumbering <sup>(209)</sup>:**

9 levels, all with ListType=*rvlstDecimal*, FirstIndent=0, MarkerIndent is increased by IndentStep <sup>(184)</sup>\*2 starting from 0, LeftIndent is increased by IndentStep <sup>(184)</sup>\*2 starting from IndentStep <sup>(184)</sup>, Font.Size=10, format sting displays "N."

### 1.3.4.14 TrvActionIndent

TrvActionIndent is a base class for the actions changing indents in the selected paragraphs.

**Unit** RichViewActions <sup>(92)</sup>;

**Syntax**

TrvActionIndent = **class** (TrvActionParaStyles <sup>(210)</sup>)

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>

## Description

The actions inherited from this class perform the following operations:

- change LeftIndent property of styles for the selected paragraphs (RVStyle.ParaStyles[].LeftIndent); for paragraphs styles having BiDiMode=*rvbdRightToLeft*, RightIndent is changed instead;
- change list levels for all paragraphs with bullets&numbering (by calling TCustomRichViewEdit.ChangeListLevels).

The actions do not change indents of list styles for bulleted or numbered paragraphs, they promote/demote their levels instead.

**To do:** to implement changing of LeftIndent property of list style, if bullets or numbering has only one level.

This action is not used directly. The following actions are inherited from it:

- TrvActionIndentDec <sup>(186)</sup>
- TrvActionIndentInc <sup>(187)</sup>

### 1.3.4.14.1 Properties

#### In TrvActionIndent

- IndentStep <sup>(186)</sup>

#### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler

- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.4.14.1 TrvActionIndent.IndentStep

Defines the step for paragraph indenting or unindenting.

**property** IndentStep: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

**Default value:**

24

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

**See also:**

- TrvActionCustomParaListSwitcher<sup>(182)</sup>.IndentStep<sup>(184)</sup>
- TrvActionParaList<sup>(206)</sup>.IndentStep<sup>(207)</sup>

#### 1.3.4.15 TrvActionIndentDec

TrvActionIndentDec is the action for "Decrease Indent" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionIndentDec = **class** (TrvActionIndent<sup>(184)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionParaStyles*<sup>(210)</sup>  
*TrvActionIndent*<sup>(184)</sup>

**Description**

This action does not introduce any new properties in addition to properties <sup>(185)</sup> of TrvActionIndent.

The action does not allow changing the paragraph indent to a negative value. If TCustomRichViewEdit.LeftMargin+LeftIndent+FirstIndent becomes negative, the action changes FirstIndent to -TCustomRichViewEdit.LeftMargin-LeftIndent (for right-to-left paragraphs, these formulas use RightIndent instead).

### 1.3.4.16 TrvActionIndentInc

TrvActionIndentInc is the action for "Increase Indent" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionIndentInc = class (TrvActionIndent (184))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>  
*TrvActionIndent* <sup>(184)</sup>

#### Description

The action does not allow changing the paragraph indent above the MaxIndent <sup>(188)</sup>.

#### 1.3.4.16.1 Properties

##### In TrvActionIndentInc

■ IndentMax <sup>(188)</sup>

##### Derived from TrvActionIndent <sup>(184)</sup>

■ IndentStep <sup>(186)</sup>

##### Derived from TrvAction <sup>(311)</sup>

■ Control <sup>(312)</sup>

##### Derived from TrvCustomAction <sup>(333)</sup>

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.4.16.1.1 TrvActionIndentInc.IndentMax

Defines the maximal possible value for the indent.

**property** IndentMax: TRVStyleLength;

The action does not increase the paragraph's indent above this value.

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

**Default value:**

200

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

### 1.3.4.17 TrvActionLineSpacing

TrvActionLineSpacing is a base class for the actions changing line spacing of the selected paragraphs.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionLineSpacing = **class** (TrvActionParaStyles<sup>(210)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>



*TrvAction* <sup>(311)</sup>*TrvActionParaStyles* <sup>(210)</sup>**Description**

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action is not used directly. The following actions are inherited from it:

- *TrvActionLineSpacing100* <sup>(189)</sup>
- *TrvActionLineSpacing150* <sup>(189)</sup>
- *TrvActionLineSpacing200* <sup>(190)</sup>

These actions change the following properties of styles of the selected paragraphs (*RVStyle.ParaStyles[]*):

- *LineSpacingType* := *rvlsPercent*
- *LineSpacing*

**1.3.4.18 TrvActionLineSpacing100**

*TrvActionLineSpacing100* is the action for "Single Line Spacing" command.

**Unit** *RichViewActions* <sup>(92)</sup>;

**Syntax**

```
TrvActionLineSpacing100 = class (TrvActionLineSpacing (188))
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionParaStyles (210)
TrvActionLineSpacing (188)

```

**Description**

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This actions changes the following properties of styles of the selected paragraphs (*RVStyle.ParaStyles[]*):

- *LineSpacingType* := *rvlsPercent*
- *LineSpacing* := 100

**1.3.4.19 TrvActionLineSpacing150**

*TrvActionLineSpacing150* is the action for "1.5 Line Spacing" command.

**Unit** *RichViewActions* <sup>(92)</sup>;

**Syntax**

```
TrvActionLineSpacing150 = class (TrvActionLineSpacing188)
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction333
TrvAction311
TrvActionParaStyles210
TrvActionLineSpacing188

```

**Description**

This action does not introduce any new properties in addition to properties<sup>312</sup> of TrvAction.

This actions changes the following properties of styles of the selected paragraphs (RVStyle.ParaStyles[]):

- LineSpacingType := *rvlsPercent*
- LineSpacing := 150

**1.3.4.20 TrvActionLineSpacing200**

TrvActionLineSpacing200 is the action for "Double Line Spacing" command.

**Unit** RichViewActions<sup>92</sup>;

**Syntax**

```
TrvActionLineSpacing200 = class (TrvActionLineSpacing188)
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction333
TrvAction311
TrvActionParaStyles210
TrvActionLineSpacing188

```

**Description**

This action does not introduce any new properties in addition to properties<sup>312</sup> of TrvAction.

This actions changes the following properties of styles of the selected paragraphs (RVStyle.ParaStyles[]):

- LineSpacingType := *rvlsPercent*
- LineSpacing := 200

### 1.3.4.21 TrvActionParaBiDi

TrvActionParaBiDi is a base class for the actions changing the default text flow direction in the selected paragraphs.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionParaBiDi = class (TrvActionParaStyles (210))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>

#### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action changes BiDiMode property for styles of the selected paragraphs (RVStyle.ParaStyles[].BiDiMode).

This action is not used directly. The following actions are inherited from it:

- TrvActionParaLTR <sup>(208)</sup>
- TrvActionParaRTL <sup>(210)</sup>

If you use these actions, it's recommended to set TCustomRichViewEdit.BiDiMode to *rvbdLeftToRight* or *rvbdRightToLeft*.

Text flow direction specified in text styles has higher priority than text flow direction specified in paragraph styles.

#### See also:

- TrvActionTextBiDi <sup>(170)</sup>
- TrvActionTextLTR <sup>(171)</sup>
- TrvActionTextRTL <sup>(172)</sup>

### 1.3.4.22 TrvActionParaBorder

TrvActionParaBorder is the action for "Paragraph Border and Background" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionParaBorder = class (TrvActionParaStyles(210))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionParaStyles*<sup>(210)</sup>

## Description

If `UserInterface`<sup>(194)</sup> = *False*, the action applies its properties to the selected paragraphs. The list of properties to apply is specified in `ValidProperties`<sup>(194)</sup>.

If `UserInterface`<sup>(194)</sup> = *True*, the action uses a dialog window. The action performs the following steps:

1. assigns font attributes from the selected paragraphs to the action's properties
2. calls `OnShowingDialog`<sup>(194)</sup> event
3. displays a dialog window
4. applies changes made in the dialog window to the action's properties
5. applies the action's properties to the selected paragraphs.

This action allows changing attributes of paragraph border and background. For changing other paragraph attributes, see `TrvActionParagraph`<sup>(198)</sup>.

### 1.3.4.22.1 Properties

#### In TrvActionParaBorder

Background<sup>(193)</sup>  
 Border<sup>(193)</sup>  
 ■ `UserInterface`<sup>(194)</sup>  
 ValidProperties<sup>(194)</sup>

#### Derived from TrvAction<sup>(311)</sup>

■ `Control`<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

■ `Caption`<sup>(334)</sup>  
 ■ `ControlPanel`<sup>(335)</sup>  
 ■ `Disabled`<sup>(335)</sup>  
 ■ `Hint`<sup>(335)</sup>

#### Derived from TAction

■ `AutoCheck`

- Caption
- Checked
  - DisableViewNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.4.22.1.1 TrvActionParaBorder.Background

Specifies background properties for applying to the selected paragraphs.

**property** Background: TRVBackgroundRect;

If `UserInterface(194)=False` and `rvpibBackground_***` in ValidProperties<sup>(194)</sup>, the action applies selected subproperties of this property to the selected paragraphs.

If `UserInterface(194)=True`, the action displays a dialog (initialized with properties of the selected paragraphs); if the user changed values of subproperties of this property in the dialog, corresponding `rvpibBackground_***` values are included in ValidProperties<sup>(194)</sup>, and the action applies them to the selected paragraphs.

This property is applied to the paragraph style's (RVStyle.ParaStyles[]) property of the same name.

Subproperties of this property are measured in the same units (pixels or twips) as properties of the target editor (i.e. in RVStyle.Units).

#### 1.3.4.22.1.2 TrvActionParaBorder.Border

Specifies border properties for applying to the selected paragraphs.

**property** Border: TRVBorder;

If `UserInterface(194)=False` and `rvpibBorder_***` in ValidProperties<sup>(194)</sup>, the action applies selected subproperties of this property to the selected paragraphs.

If `UserInterface(194)=True`, the action displays a dialog (initialized with properties of the selected paragraphs); if the user changed values of subproperties of this property in the dialog, corresponding `rvpibBorder_***` values are included in ValidProperties<sup>(194)</sup>, and the action applies them to the selected paragraphs.

This property is applied to the paragraph style's (RVStyle.ParaStyles[]) property of the same name.

Subproperties of this property are measured in the same units (pixels or twips) as properties of the target editor (i.e. in RVStyle.Units).

### 1.3.4.22.1.3 TrvActionParaBorder.UserInterface

Allows working with or without user interaction.

**property** UserInterface: Boolean;

If UserInterface=*False*, the action applies its properties to the selected paragraphs.

If UserInterface=*True*, the action displays a dialog, assigns changes to the action's properties, and then applies these properties to the selected paragraphs.

**Default value:**

*True*

### 1.3.4.22.1.4 TrvActionParaBorder.ValidProperties

Defines a set of properties for applying to the selected paragraphs.

**type**

```
TRVParaInfoBorderProperty = (rvpibBackground_Color,
    rvpibBackground_BO_Left, rvpibBackground_BO_Top,
    rvpibBackground_BO_Right, rvpibBackground_BO_Bottom,
    rvpibBorder_Color, rvpibBorder_Style, rvpibBorder_Width,
    rvpibBorder_InternalWidth,
    rvpibBorder_BO_Left, rvpibBorder_BO_Top,
    rvpibBorder_BO_Right, rvpibBorder_BO_Bottom,
    rvpibBorder_Vis_Left, rvpibBorder_Vis_Top,
    rvpibBorder_Vis_Right, rvpibBorder_Vis_Bottom);
TRVParaInfoBorderProperties = set of TRVParaInfoBorderProperty;
```

**property** ValidProperties: TRVParaInfoBorderProperty;

If UserInterface<sup>(194)</sup>=*False*, assign a subset of properties (for applying) to ValidProperties .

If UserInterface<sup>(194)</sup>=*True*, this property is maintained automatically. After displaying a dialog, this property contains a set of properties corresponding to non-blank fields in the dialog.

## 1.3.4.22.2 Events

### In TrvActionParaBorder

■ OnShowingDialog<sup>(194)</sup>

#### 1.3.4.22.2.1 TrvActionParaBorder.OnShowingDialog

Occurs before showing a dialog window.

**property** OnShowingDialog: TNotifyEvent;

This event occurs if UserInterface<sup>(194)</sup>=*True*.

It is called when properties of the action are already assigned from attributes of the selected paragraphs of the target editor, but the dialog is not initialized yet.

In this event, you can modify properties of the action, so modified properties will be applied to the dialog.

You can use this event to initialize a dialog with predefined values, instead of taking them from selected paragraphs.

### 1.3.4.23 TrvActionParaBullets

TrvActionParaBullets is the action for "Bullets" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionParaBullets = class (TrvActionCustomParaListSwitcher(182))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomParaListSwitcher*<sup>(182)</sup>

#### Description

This action applies or removes bullets to the selected paragraphs.

This action does not introduce any new properties in addition to properties<sup>(183)</sup> of TrvActionCustomParaListSwitcher.

ListLevels<sup>(184)</sup> must not contains numbered levels.

When applying list, the action tries to reuse existing list styles with identical properties, if they exist.

### 1.3.4.24 TrvActionParaColor

TrvActionParaColor is the action for changing background color of the selected paragraphs.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionParaColor = class (TrvActionParaCustomColor(198))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*

*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>  
*TrvActionParaCustomColor*<sup>(198)</sup>

## Description

This action does not introduce any new properties and events in addition to properties and events of *TrvActionCustomColor*<sup>(318)</sup>. However, it published inherited *Opacity*<sup>(318)</sup> property.

On execution, this action changes *Background.Color* and *Opacity* properties of styles of the selected paragraphs (*RVStyle.ParaStyles[]*.*Background.Color* and *.Opacity*).

See also the list of color changing actions in the topic about *TrvActionCustomColor*<sup>(318)</sup>.

### 1.3.4.25 TrvActionParaColorAndPadding

*TrvActionParaColorAndPadding* is the action for changing background color and padding of the selected paragraphs.

**Unit** *RichViewActions*<sup>(92)</sup>;

## Syntax

```
TrvActionParaColorAndPadding = class (TrvActionParaColor(195))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>  
*TrvActionParaCustomColor*<sup>(198)</sup>  
*TrvActionParaColor*<sup>(195)</sup>

## Description

This action is reserved. It does not have *Caption* and *Hint* assigned by the localization procedure. If you use this action, assign its *Caption* and *Hint* yourself.

This action does not introduce any new events in addition to events of *TrvActionCustomColor*<sup>(318)</sup>.

On execution, this action changes *Background.Color* and *Background.BorderOffsets* properties of styles of the selected paragraphs (*RVStyle.ParaStyles[]*.*Background.Color* and *RVStyle.ParaStyles[]*.*Background.BorderOffsets*). The specific (left/top/right/bottom) padding value specified in *Padding*<sup>(197)</sup> property is assigned only if the corresponding property of *UsePadding*<sup>(198)</sup> is *True*. If the padding exceeds the paragraph's indents/spacing (*LeftIndent/SpaceBefore/RightIndent/SpaceAfter*), indents/spacing is increased.



Values of `Padding`<sup>(197)</sup> and `UsePadding`<sup>(198)</sup> properties must be assigned by the programmer.

#### 1.3.4.25.1 Properties

##### In `TrvActionParaColorAndPadding`

`Padding`<sup>(197)</sup>

`UsePadding`<sup>(198)</sup>

##### Derived from `TrvActionCustomColor`<sup>(318)</sup>

■ `CallerControl`<sup>(319)</sup>

■ `Color`<sup>(319)</sup>

■ `UserInterface`<sup>(320)</sup>

##### Derived from `TrvAction`<sup>(311)</sup>

■ `Control`<sup>(312)</sup>

##### Derived from `TrvCustomAction`<sup>(333)</sup>

■ `Caption`<sup>(334)</sup>

■ `Disabled`<sup>(335)</sup>

■ `Hint`<sup>(335)</sup>

##### Derived from `TAction`

■ `AutoCheck`

■ `Caption`

■ `Checked`  
`DisableIfNoHandler`

■ `Enabled`

■ `GroupIndex`

■ `HelpContext`

■ `HelpKeyword`

■ `HelpType`

■ `Hint`

■ `ImageIndex`

■ `Name`

■ `SecondaryShortCuts`

■ `ShortCut`

■ `Visible`

##### 1.3.4.25.1.1 `TrvActionParaColorAndPadding.Padding`

Specifies the padding for applying to the selected paragraphs.

**property** `Padding`: `TRVRect`;

This property is applied to `Background.BorderOffsets` property of paragraph styles (`RVStyle.Parastyles[]`.`Background.BorderOffsets`).

If `Padding.Left` exceeds `LeftIndent` property of the paragraph style, `LeftIndent` is increased.

If `Padding.Right` exceeds `RightIndent` property of the paragraph style, `RightIndent` is increased.

If `Padding.Top` exceeds `SpaceBefore` property of the paragraph style, `SpaceBefore` is increased.

If `Padding.Bottom` exceeds `SpaceAfter` property of the paragraph style, `SpaceAfter` is increased.

`UsePadding`<sup>(198)</sup> property specifies which sides of padding are applied.

#### 1.3.4.25.1.2 TrvActionParaColorAndPadding.UsePadding

Specifies sides of paragraphs for applying `Padding`<sup>(197)</sup>.

**property** `UsePadding`: `TRVBooleanRect`;

### 1.3.4.26 TrvActionParaCustomColor

`TrvActionParaCustomColor` is a base class for the actions changing color properties of the selected paragraphs.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

`TrvActionParaCustomColor` = **class** (`TrvActionCustomColor`<sup>(318)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>  
*TrvActionParaCustomColor*<sup>(198)</sup>

#### Description

This action does not introduce any new properties and events in addition to properties and events of `TrvActionCustomColor`<sup>(318)</sup>.

This action is not used directly. The following actions are inherited from it:

- `TrvActionParaColor`<sup>(195)</sup>
- `TrvActionParaColorAndPadding`<sup>(196)</sup>

### 1.3.4.27 TrvActionParagraph

`TrvActionParagraph` is the action for "Paragraph" command.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

`TrvActionParagraph` = **class** (`TrvActionParaStyles`<sup>(210)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionParaStyles* <sup>(210)</sup>

## Description

If `UserInterface` <sup>(205)</sup> = `False`, the action applies its properties to the selected paragraphs. The list of properties to apply is specified in `ValidProperties` <sup>(205)</sup>.

If `UserInterface` <sup>(205)</sup> = `True`, , the action uses a dialog window. The action performs the following steps:

1. assigns font attributes from the selected paragraphs to the action's properties
2. calls `OnShowingDialog` <sup>(205)</sup> event
3. displays a dialog window
4. applies changes made in the dialog window to the action's properties
5. applies the action's properties to the selected paragraphs.

This action allows changing almost all paragraph attributes, except for border and background (see `TrvActionParaBorder` <sup>(191)</sup>).

### 1.3.4.27.1 Properties

#### In `TrvActionParagraph`

`Alignment` <sup>(200)</sup>  
`DeleteAllTabs` <sup>(200)</sup>  
`FirstIndent` <sup>(201)</sup>  
`KeepLinesTogether` <sup>(201)</sup>  
`KeepWithNext` <sup>(201)</sup>  
`LastLineAlignment` <sup>(202)</sup>  
`LeftIndent` <sup>(202)</sup>  
`LineSpacing` <sup>(202)</sup>  
`LineSpacingType` <sup>(203)</sup>  
`OutlineLevel` <sup>(203)</sup>  
`RightIndent` <sup>(203)</sup>  
`SpaceAfter` <sup>(203)</sup>  
`SpaceBefore` <sup>(204)</sup>  
`Tabs` <sup>(204)</sup>  
`TabsToDelete` <sup>(204)</sup>  
■ `UserInterface` <sup>(205)</sup>  
`ValidProperties` <sup>(205)</sup>

## Derived from TrvAction <sup>(311)</sup>

---

- Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.4.27.1.1 TrvActionParagraph.Alignment

Specifies the alignment for applying to the selected paragraphs.

**property** Alignment: TRVAlignment;

If UserInterface <sup>(205)</sup> = *False* and *rvpimAlignment* in ValidProperties <sup>(205)</sup>, the action applies this property to the selected paragraphs.

If UserInterface <sup>(205)</sup> = *True*, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, *rvpimAlignment* is included in ValidProperties <sup>(205)</sup>, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (RVStyle.ParaStyles[]) property of the same name.

### 1.3.4.27.1.2 TrvActionParagraph.DeleteAllTabs

Instructs to delete all tab-stop positions from styles of the selected paragraphs.

**property** DeleteAllTabs: Boolean;

If *True*, all existing tab-stop positions are deleted from styles of the selected paragraphs.

If *False*, only tab-stops specified in TabsToDelete <sup>(204)</sup> are deleted.

After that, tab-stops from Tabs<sup>(204)</sup> are added.

All these properties are applied only if ValidProperties<sup>(205)</sup> contains *rvpimTabs*.

If UserInterface<sup>(205)</sup>=*True*, you should assign these properties yourself.

If UserInterface<sup>(205)</sup>=*False*, these properties are defined by the user in a dialog.

#### 1.3.4.27.1.3 TrvActionParagraph.FirstIndent

Specifies the first line indent for applying to the selected paragraphs.

**property** FirstIndent: TRVStyleLength;

If UserInterface<sup>(205)</sup>=*False* and *rvpimFirstIndent* in ValidProperties<sup>(205)</sup>, the action applies this property to the selected paragraphs.

If UserInterface<sup>(205)</sup>=*True*, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, *rvpimFirstIndent* is included in ValidProperties<sup>(205)</sup>, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (RVStyle.ParaStyles[]) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in RVStyle.Units).

#### 1.3.4.27.1.4 TrvActionParagraph.KeepLinesTogether

Specifies the keep-lines-together option for applying to the selected paragraphs.

**property** KeepLinesTogether: Boolean;

If UserInterface<sup>(205)</sup>=*False* and *rvpimKeepLinesTogether* in ValidProperties<sup>(205)</sup>, the action applies this property to the selected paragraphs.

If UserInterface<sup>(205)</sup>=*True*, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, *rvpimKeepLinesTogether* is included in ValidProperties<sup>(205)</sup>, and the action applies this property to the selected paragraphs.

This option adds/removes *rvpaoKeepLinesTogether* in the Options property of styles of the selected paragraphs (RVStyle.ParaStyles[]).Options).

#### 1.3.4.27.1.5 TrvActionParagraph.KeepWithNext

Specifies the keep-with-next option for applying to the selected paragraphs.

**property** KeepWithNext: Boolean;

If UserInterface<sup>(205)</sup>=*False* and *rvpimKeepWithNext* in ValidProperties<sup>(205)</sup>, the action applies this property to the selected paragraphs.

If UserInterface<sup>(205)</sup>=*True*, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, *rvpimKeepWithNext* is included in ValidProperties<sup>(205)</sup>, and the action applies this property to the selected paragraphs.

This option adds/removes *rvpaoKeepWithNext* in the Options property of styles of the selected paragraphs (RVStyle.ParaStyles[]).Options).

#### 1.3.4.27.1.6 TrvActionParagraph.LastLineAlignment

Specifies the alignment for applying to the last line of selected justified and distributed paragraphs.

**property** LastLineAlignment: TRVLastLineAlignment;

If `UserInterface(205) = False` and `rvpimLastLineAlignment` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205) = True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimLastLineAlignment` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

#### 1.3.4.27.1.7 TrvActionParagraph.LeftIndent

Specifies the left indent for applying to the selected paragraphs.

**property** LeftIndent: TRVStyleLength;

If `UserInterface(205) = False` and `rvpimLeftIndent` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205) = True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimLeftIndent` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.8 TrvActionParagraph.LineSpacing

Specifies the line spacing value for applying to the selected paragraphs.

**property** LineSpacing: TRVLineSpacingValue;

If `UserInterface(205) = False` and `rvpimLineSpacing` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205) = True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimLineSpacing` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

Meaning of this property depends on the value of `LineSpacingType(203)` property. These two properties are applied together.

If `LineSpacingType(203) <> rvsPercent`, this value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.9 TrvActionParagraph.LineSpacingType

Specifies the line spacing type for applying to the selected paragraphs.

**property** LineSpacingType: TRVLineSpacingType;

If `UserInterface(205)=False` and `rvpimLineSpacing` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205)=True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimLineSpacing` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

This property is applied together with `LineSpacing(202)` property.

#### 1.3.4.27.1.10 TrvActionParagraph.OutlineLevel

Specifies the line spacing type for applying to the selected paragraphs.

**property** OutlineLevel: Integer;

If `UserInterface(205)=False` and `rvpimOutlineLevel` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205)=True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimOutlineLevel` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

#### 1.3.4.27.1.11 TrvActionParagraph.RightIndent

Specifies the right indent for applying to the selected paragraphs.

**property** RightIndent: TRVStyleLength;

If `UserInterface(205)=False` and `rvpimRightIndent` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205)=True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimRightIndent` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.12 TrvActionParagraph.SpaceAfter

Specifies spacing for applying to the bottom side of the selected paragraphs.

**property** SpaceAfter: TRVStyleLength;

If `UserInterface(205)=False` and `rvpimSpaceAfter` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205) = True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimSpaceAfter` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.13 TrvActionParagraph.SpaceBefore

Specifies spacing for applying to the top side of the selected paragraphs.

**property** `SpaceBefore: TRVStyleLength;`

If `UserInterface(205) = False` and `rvpimSpaceBefore` in `ValidProperties(205)`, the action applies this property to the selected paragraphs.

If `UserInterface(205) = True`, the action displays a paragraph dialog (initialized with properties of the selected paragraphs); if the user changed value of this property in the dialog, `rvpimSpaceBefore` is included in `ValidProperties(205)`, and the action applies this property to the selected paragraphs.

This property is applied to the paragraph style's (`RVStyle.ParaStyles[]`) property of the same name.

This value is measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.14 TrvActionParagraph.Tabs

Contains a list of tab-stops for adding to styles of the selected paragraphs.

**property** `Tabs: TRVTabInfos;`

Before applying this property, some (`TabsToDelete(204)`) or all (`DeleteAllTabs(200)`) tab-stops are deleted.

All these properties are applied only if `ValidProperties(205)` contains `rvpimTabs`.

If `UserInterface(205) = True`, you should assign these properties yourself.

If `UserInterface(205) = False`, these properties are defined by the user in a dialog.

Position properties of tabs are measured in the same units (pixels or twips) as properties of the target editor (i.e. in `RVStyle.Units`).

#### 1.3.4.27.1.15 TrvActionParagraph.TabsToDelete

Contains a list of tab stop positions for deleting from styles of the selected paragraphs.

**property** `TabsToDelete: TRVIntegerList;`

If `DeleteAllTabs(200) = True`, all tab-stops are deleted instead.

After that, tab-stops from `Tabs(204)` are added.

All these properties are applied only if `ValidProperties(205)` contains `rvpimTabs`.

If `UserInterface(205) = True`, you should assign these properties yourself.

If `UserInterface(205) = False`, these properties are defined by the user in a dialog.



#### 1.3.4.27.1.16 TrvActionParagraph.UserInterface

Allows working with or without user interaction.

**property** UserInterface: Boolean;

If UserInterface=*False*, the action applies its properties to the selected paragraphs.

If UserInterface=*True*, the action displays a paragraph dialog, assigns changes to the action's properties, and then applies these properties to the selected paragraphs.

**Default value:**

*True*

#### 1.3.4.27.1.17 TrvActionParagraph.ValidProperties

Defines a set of properties for applying to the selected paragraphs.

**type**

```
TRVParaInfoMainProperty = (rvpimFirstIndent, rvpimLeftIndent,
    rvpimRightIndent, rvpimSpaceBefore, rvpimSpaceAfter,
    rvpimAlignment, rvpimLastLineAlignment,
    rvpimOutlineLevel, rvpimLineSpacing,
    rvpimKeepLinesTogether, rvpimKeepWithNext, rvpimTabs);
TRVParaInfoMainProperties = set of TRVParaInfoMainProperty;
```

**property** ValidProperties: TRVParaInfoMainProperties;

If UserInterface<sup>(205)</sup>=*False*, assign a subset of properties (for applying) to ValidProperties .

If UserInterface<sup>(205)</sup>=*True*, this property is maintained automatically. After displaying a paragraph dialog, this property contains a set of properties corresponding to non-blank fields in the dialog.

### 1.3.4.27.2 Events

#### In TrvActionParagraph

##### ■ OnShowingDialog<sup>(205)</sup>

#### 1.3.4.27.2.1 TrvActionParagraph.OnShowingDialog

Occurs before showing a dialog window.

**property** OnShowingDialog: TNotifyEvent;

This event occurs if UserInterface<sup>(205)</sup>=*True*.

It is called when properties of the action are already assigned from attributes of the selected paragraphs of the target editor, but the dialog is not initialized yet.

In this event, you can modify properties of the action, so modified properties will be applied to the dialog.

You can use this event to initialize a dialog with predefined values, instead of taking them from selected paragraphs.

### 1.3.4.28 TrvActionParaList

TrvActionParaList is the action for "Bullets and Numbering" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionParaList = class (TrvActionParaStyles(210))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionParaStyles*<sup>(210)</sup>

#### Description

The action displays a dialog window where the user can choose from the predefined list templates (may be after customizing them). If the caret is placed at the paragraph having bullets or numbering, the current list style takes the place of one of these templates, so the user can customize it. All templates have 9 list levels (maximal possible count that can be saved in RTF and DocX). For numbered styles, the user can choose to continue or to reset numbering, or to create a new list.

When applying bullets (without numbered levels), the action tries to reuse existing list styles with the same properties, if they exist. The action may apply properties to TrvActionParaBullets<sup>(195)</sup> actions.

When applying numbering (if at least one level is numbered), the action uses list styles around the selection (if they exist and have identical properties with the chosen properties). If it's not possible, the action tries to reuse unused list styles (if they exist and have identical properties with the chosen properties). Otherwise, a new list is created. If all levels are numbered, the action may apply properties to TrvActionParaNumbering<sup>(209)</sup> actions.

#### 1.3.4.28.1 Properties

##### In TrvActionParaList

ActionParaBullets<sup>(208)</sup>  
 ActionParaNumbering<sup>(208)</sup>  
 ■ IndentStep<sup>(207)</sup>  
 UpdateAllActionsOnForm<sup>(207)</sup>

##### Derived from TrvAction<sup>(311)</sup>

■ Control<sup>(312)</sup>

## Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.4.28.1.1 TrvActionParaList.IndentStep

Defines the step for increasing LeftIndent and MarkerIndent of list levels.

**property** IndentStep: TRVStyleLength;

This value is measured in GetControlPanel <sup>(336)</sup>.UnitsProgram <sup>(55)</sup>.

This property is used to adjust indents in the predefined set of lists (indents are adjusted when this action is executed for the first time).

#### Default value:

24

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

#### See also:

- TrvActionCustomParaListSwitcher <sup>(182)</sup>.IndentStep <sup>(184)</sup>
- TrvActionIndent <sup>(184)</sup>.IndentStep <sup>(186)</sup>

### 1.3.4.28.1.2 TrvActionParaList.UpdateAllActionsOnForm

Allows updating all TrvActionParaBullets <sup>(195)</sup> and TrvActionParaNumbering <sup>(209)</sup> actions on the same form as this action.

**property** UpdateAllActionsOnForm: Boolean;

If this property is *True*:

- when the user applies a list without numbered levels, the action assigns these levels to ListLevels<sup>(184)</sup> of all TrvActionParaBullets<sup>(195)</sup> actions on the same form/datamodule.
- when the user applies a list with all numbered levels, the action assigns these levels to ListLevels<sup>(184)</sup> of all TrvActionParaNumbering<sup>(209)</sup> actions on the same form/datamodule.

**Default value:**

*True*

**See also properties:**

- ActionParaNumbering<sup>(208)</sup>
- ActionParaBullets<sup>(208)</sup>

#### 1.3.4.28.1.3 TrvActionParaList.ActionParaNumbering

Defines a link to the "Numbering" action.

**property** ActionParaNumbering: TrvActionParaNumbering<sup>(209)</sup>;

When the user applies a list with all numbered levels, the action assigns these levels to ListLevels<sup>(184)</sup> properties to the action specified in this link. If UpdateAllActionsOnForm<sup>(207)</sup>=*True*, these levels are assigned to ListLevels<sup>(184)</sup> of all TrvActionParaNumbering<sup>(209)</sup> actions on the same form/datamodule.

#### 1.3.4.28.1.4 TrvActionParaList.ActionParaBullets

Defines a link to the "Bullets" action.

**property** ActionParaBullets: TrvActionParaBullets<sup>(195)</sup>;

When the user applies a list without numbered levels, the action assigns these levels to ListLevels<sup>(184)</sup> properties to the action specified in this link. If UpdateAllActionsOnForm<sup>(207)</sup>=*True*, these levels are assigned to ListLevels<sup>(184)</sup> of all TrvActionParaBullets<sup>(195)</sup> actions on the same form/datamodule.

### 1.3.4.29 TrvActionParaLTR

TrvActionParaLTR is the action for "Left to Right" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionParaLTR = **class** (TrvActionParaBiDi<sup>(191)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionParaStyles*<sup>(173)</sup>

*TrvActionParaBiDi* <sup>(191)</sup>

### Description

This action changes the default text direction in the selected paragraphs to left-to-right. If you use this action, it's recommended to set `TCustomRichViewEdit.BiDiMode` to *rvbdLeftToRight* or *rvbdRightToLeft*.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

This action changes `BiDiMode` property for styles of the selected paragraphs (`RVStyle.ParaStyles[].BiDiMode`) to *rvbdLeftToRight*.

This is a checkbox-like action, its `Checked` property is updated depending on the state of selection (or the current paragraph style).

### See also:

- `TrvActionTextLTR` <sup>(171)</sup>
- `TrvActionTextRTL` <sup>(172)</sup>
- `TrvActionParaRTL` <sup>(210)</sup>

## 1.3.4.30 TrvActionParaNumbering

`TrvActionParaNumbering` is the action for "Numbering" command.

**Unit** `RichViewActions` <sup>(92)</sup>;

### Syntax

```
TrvActionParaNumbering = class (TrvActionCustomParaListSwitcher (182))
```

### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionCustomParaListSwitcher (182)

```

### Description

This action applies or removes numbering to the selected paragraphs.

This action does not introduce any new properties in addition to properties <sup>(183)</sup> of `TrvActionCustomParaListSwitcher`.

`ListLevels` <sup>(184)</sup> should contain numbered levels.

When applying lists, the action does the following:

1. if the paragraph before the selection, the selection or the paragraph after the selection contains a list style with the same properties as `ListLevels` <sup>(184)</sup>, applies this list style;

2. otherwise, if RVStyle.ListStyles has unused list style with the same properties as ListLevels<sup>(184)</sup>, applies this list style;
3. otherwise, creates a new list style, assigns ListLevels<sup>(184)</sup> to it, applies this list style.

**See also:**

- TrvActionParaBullets<sup>(195)</sup>
- TrvActionParaList<sup>(206)</sup>

### 1.3.4.31 TrvActionParaRTL

TrvActionParaRTL is the action for "Right to Left" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionParaRTL = class (TrvActionParaBiDi(191))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionParaStyles*<sup>(173)</sup>  
*TrvActionParaBiDi*<sup>(191)</sup>

**Description**

This action changes the default text direction in the selected paragraphs to right-to-left. If you use this action, it's recommended to set TCustomRichViewEdit.BiDiMode to *rvbdLeftToRight* or *rvbdRightToLeft*.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This action changes BiDiMode property for styles of the selected paragraphs (RVStyle.ParaStyles[].BiDiMode) to *rvbdRightToLeft*.

This is a checkbox-like action, its Checked property is updated depending on the state of selection (or the current paragraph style).

**See also:**

- TrvActionTextLTR<sup>(171)</sup>
- TrvActionTextRTL<sup>(172)</sup>
- TrvActionParaLTR<sup>(208)</sup>

### 1.3.4.32 TrvActionParaStyles

TrvActionParaStyles is a base class for the actions applying changes to the selected paragraphs.

**Unit** RichViewActions<sup>(92)</sup>;

## Syntax

```
TrvActionParaStyles = class (TrvAction311)
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>

## Description

This action does not introduce any new properties in addition to properties<sup>312</sup> of *TrvAction*.

This action is not used directly. This action has the following direct descendants:

- *TrvActionParagraph*<sup>198</sup>
- *TrvActionAlignment*<sup>178</sup>
- *TrvActionParaBiDi*<sup>191</sup>
- *TrvActionIndent*<sup>184</sup>
- *TrvActionWordWrap*<sup>211</sup>
- *TrvActionLineSpacing*<sup>188</sup>
- *TrvActionParaBorder*<sup>191</sup>

### 1.3.4.33 TrvActionWordWrap

*TrvActionWordWrap* is the action for "Word Wrap" command.

**Unit** RichViewActions<sup>92</sup>;

## Syntax

```
TrvActionWordWrap = class (TrvActionParaStyles210)
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>  
*TrvActionParaStyles*<sup>173</sup>

## Description

This action turns word wrapping on/off for the selected paragraphs. It excludes/includes *rvpaoNoWrap* from the Options property of styles of the selected paragraphs (RVStyle.ParaStyles[.Options).

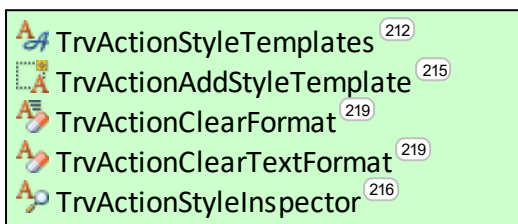
This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This is a checkbox-like action, its Checked property is updated depending on the current paragraph style (or selection).

### 1.3.5 Styles

#### Styles Actions

This group of actions includes commands for managing and applying named styles of text and paragraphs.



#### 1.3.5.1 TrvActionStyleTemplates

TrvActionStyleTemplates is the action for "Format | Styles" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionStyleTemplates = class (TrvAction(311))
```

##### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)

```

##### Description

This action is enabled only if Editor.UseStyleTemplates=True.

The action shows a dialog window allowing to manage style templates: to add, to delete, to modify properties, to import (from the special format, DocX, RTF, HTML, and RVF files), to export (to the special format).

When shown initially, the dialog window displays a style template of the current text in the editor.



You can define initial properties of standard style templates using `StandardStyleTemplates`<sup>(214)</sup> property.

You can customize the appearance of the dialog window using `Images`<sup>(214)</sup>, `TextStyleImageIndex`, `ParaStyleImageIndex`, `ParaTextStyleImageIndex`<sup>(215)</sup> properties.

By default, the style import and export dialogs use localized names of RVF files (\*.rvf) and of files containing style templates (\*.rvst). If you assign `GetControlPanel`<sup>(336)</sup>.`RVFLocalizable`<sup>(52)</sup>, you can change them in the control panel properties: `RVFFilter`<sup>(52)</sup>, `RVStylesFilter`, `RVStylesExt`<sup>(53)</sup>.

This action is used by other actions:

- `TrvActionAddStyleTemplate`<sup>(215)</sup> (to open a dialog window)
- `TrvActionInsertHyperlink`<sup>(232)</sup> (to get default properties of "Hyperlink" style template)

### 1.3.5.1.1 Properties

#### In `TrvActionStyleTemplates`

- `Images`<sup>(214)</sup>
- `ParaStyleImageIndex`<sup>(215)</sup>
- `ParaTextStyleImageIndex`<sup>(215)</sup>
- `StandardStyleTemplates`<sup>(214)</sup>
- `TextStyleImageIndex`<sup>(215)</sup>

#### Derived from `TrvAction`<sup>(311)</sup>

- `Control`<sup>(312)</sup>

#### Derived from `TrvCustomAction`<sup>(333)</sup>

- `Caption`<sup>(334)</sup>
- `ControlPanel`<sup>(335)</sup>
- `Disabled`<sup>(335)</sup>
- `Hint`<sup>(335)</sup>

#### Derived from `TAction`

- `AutoCheck`
- `Caption`
- `Checked`
- `DisableIfNoHandler`
- `Enabled`
- `GroupIndex`
- `HelpContext`
- `HelpKeyword`
- `HelpType`
- `Hint`
- `ImageIndex`
- `Name`
- `SecondaryShortCuts`
- `ShortCut`
- `Visible`

### 1.3.5.1.1.1 TrvActionStyleTemplates.Images

Images for icons in the styles dialog window.

**property** Images: TCustomImageList;

If **Images** is not assigned:

- the tree-view containing a list of styles uses standard 16-color icons;
- a description of styles is displayed without icons.

If **Images** is assigned:

- if at least one of TextStyleImageIndex, ParaStyleImageIndex, ParaTextStyleImageIndex<sup>(215)</sup> >=0, **Images** and these image indices are used in the tree-view; ImageIndex of this action is used for the root tree node;
- the component searches for the instances of TrvActionFontEx<sup>(149)</sup>, TrvActionParagraph<sup>(198)</sup>, TrvActionParaBorder<sup>(191)</sup>, TrvActionInsertHyperlink<sup>(232)</sup> actions on the same form; if found, **Images** and the image indices of these actions are used in a style description.

**Images** are used in the style import dialog window as well.

Generally, you should assign the same TImageList component as used for TActionList/TActionManager containing this action.

### 1.3.5.1.1.2 TrvActionStyleTemplates.StandardStyleTemplates

Contains a list of style templates used as samples of standard style templates.

**property** StandardStyleTemplates: TRVStyleTemplateCollection;

When an user adds a standard style template in the dialog, the action searches in **StandardStyleTemplates** for a style template having this name. If found, the action uses properties of the **StandardStyleTemplates**' item as initial values for the added style template. If not found, the action uses hard-coded values as initial properties.

The following names are used for standard styles: 'Normal', 'Normal Indent', 'No Spacing', 'heading 1'...'heading 9', 'List Paragraph', 'Hyperlink', 'Title', 'Subtitle', 'Emphasis', 'Subtle Emphasis', 'Intense Emphasis', 'Strong', 'Quote', 'Intense Quote', 'Subtle Reference', 'Intense Reference', 'Block Text', 'HTML Variable', 'HTML Code', 'HTML Acronym', 'HTML Definition', 'HTML Keyboard', 'HTML Sample', 'HTML Typewriter', 'HTML Preformatted', 'HTML Cite', 'header', 'footer', 'page number', 'endnote reference', 'footnote reference', 'endnote text', 'footnote text'. Names are case sensitive. These names are not shown to users, their localized versions are shown instead.

Values in this collection are measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property may also be used by TrvActionInsertHyperlink, when it adds "Hyperlink" style template.

**See also:**

TrvActionNew<sup>(103)</sup>.StyleTemplates<sup>(104)</sup>

TrvActionInsertHyperlink<sup>(232)</sup>.AutoAddHyperlinkStyleTemplate<sup>(235)</sup>.

### 1.3.5.1.1.3 TrvActionStyleTemplates.\*StyleImageIndex

The properties define image indices for icons in the tree-view containing a list of styles.

**property** TextStyleImageIndex: Integer;

**property** ParaStyleImageIndex: Integer;

**property** ParaTextStyleImageIndex: Integer;

If Images<sup>(214)</sup> is assigned, and at least one of these properties  $\geq 0$ , the tree-view in the style management dialog window uses these images:

- **TextStyleImageIndex** for style templates having Kind=*rvstkText* (and a tree node used as a root of for such style templates);
- **ParaStyleImageIndex** for style templates having Kind=*rvstkPara*;
- **ParaTextStyleImageIndex** for style templates having Kind=*rvstkParaText*.

Additionally, ImageIndex of this action is used for the root tree node.

**Default value:**

-1

## 1.3.5.2 TrvActionAddStyleTemplate

TrvActionAddStyleTemplate is the action for "Format | Add Style..." command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionAddStyleTemplate = **class** (TrvAction<sup>(311)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

**Description**

This action is enabled only if Editor.UseStyleTemplates=*True*.

The action shows a dialog window allowing to manage style templates. When shown initially, the dialog window has a new style template added; its properties are based on the text and paragraph attributes at the position of the caret.

This action can work only if it is linked to TrvActionStyleTemplates<sup>(212)</sup> action. It can be linked using ActionStyleTemplates<sup>(216)</sup> property; if this property is not assigned, the action searches for TrvActionStyleTemplates<sup>(212)</sup> action on the same form.

### 1.3.5.2.1 Properties

#### In TrvActionAddStyleTemplate

---

- ActionStyleTemplates <sup>(216)</sup>

#### Derived from TrvAction <sup>(311)</sup>

---

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.5.2.1.1 TrvActionAddStyleTemplate.ActionStyleTemplates

Links this action to TrvActionStyleTemplates <sup>(212)</sup> action.

**property** ActionStyleTemplates: TrvActionStyleTemplates <sup>(212)</sup>;

If this property is not assigned, the action searches for TrvActionStyleTemplates <sup>(212)</sup> action on the same form. If not found, the action does nothing.

TrvActionStyleTemplates <sup>(212)</sup> defines properties for the style management dialog window.

### 1.3.5.3 TrvActionStyleInspector

TrvActionStyleInspector is the action for "Format | Style Inspector" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionCustomInfoWindow = class (TrvAction (311)) ;
TrvActionStyleInspector = class (TrvActionCustomInfoWindow) ;
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

## Description

This action shows/hides a non-modal window containing information about text and paragraph styles and attributes at the position of the caret.

Assign Control <sup>(312)</sup> property of this action, and the info window will be updated automatically according to changes in Control <sup>(312)</sup>. When the action is executed for some editor, the action automatically assign this editor to Control <sup>(312)</sup>.

You can customize the appearance of the info window using Images <sup>(218)</sup>, FontImageIndex and ParaImageIndex <sup>(218)</sup> properties.

### 1.3.5.3.1 Properties

#### In TrvActionStyleInspector

---

- FontImageIndex <sup>(218)</sup>
- Images <sup>(218)</sup>
- ParaImageIndex <sup>(218)</sup>

#### Derived from TrvAction <sup>(311)</sup>

---

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType

- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.5.3.1.1 TrvActionStyleInspector.Images

Images for icons used in a description of text and paragraph attributes.

**property** Images: TCustomImageList;

This image list must contain two images: text and paragraph icons. Their indices should be specified in FontImageIndex and ParaImageIndex<sup>(218)</sup> properties.

#### 1.3.5.3.1.2 TrvActionStyleInspector.FontImageIndex, ParaImageIndex

Indices in Images<sup>(218)</sup>.

**property** FontImageIndex: Integer;

**property** ParaImageIndex: Integer;

FontImageIndex specifies the index of "font" icon.

ParaImageIndex specifies the index of "paragraph" icon.

**Default value:**

-1

#### 1.3.5.3.2 Methods

##### In TrvActionStyleInspector

UpdateInfo<sup>(218)</sup>

**Inherited from TrvCustomAction**<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

#### 1.3.5.3.2.1 TrvActionStyleInspector.UpdateInfo

Refresh a content of the info window.

**procedure** UpdateInfo(OnlyIfVisible: Boolean = True);

Normally, the action updates the window when necessary.

Call this method when GetControlPanel<sup>(336)</sup>.Language<sup>(51)</sup> is changed.

#### 1.3.5.3.3 Events

##### Inherited from TrvActionCustomInfoWindow

- OnShowing<sup>(219)</sup>

#### 1.3.5.3.3.1 TrvActionCustomInfoWindow.OnShowing

Occurs before showing the info window.

**property** OnShowing: TRVShowFormEvent<sup>(388)</sup>;

**Example** (embedding the info window in the main window TfrmMain, at the right side):

```
procedure TfrmMain.rvActionStyleInspector1Showing(Sender: TrvAction(311);
    Form: TForm);
begin
    Form.Align := alRight;
    Form.Parent := Self;
end;
```

#### 1.3.5.4 TrvActionClearFormat

TrvActionClearFormat is the action for "Format | Clear Format" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionClearFormat = class (TrvAction(311))
```

##### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
```

##### Description

This action resets text and/or paragraph attributes in the selected fragment by calling Editor.ApplyStyleTemplate(-1).

The results depend on the selection. The action clears the paragraph format in the following cases:

- the selection is empty
- the selection includes the whole paragraph completely
- the selection includes several paragraphs.

If the selection is not empty, the action clears its text format.

This action can be used even if Editor.UseStyleTemplates=*False*. In this case, it simply applies the 0th text and the 0th paragraph styles.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### 1.3.5.5 TrvActionClearTextFormat

TrvActionClearFormat is the action for "Format | Clear Text Format" command.

**Unit** RichViewActions<sup>(92)</sup>;

## Syntax

```
TrvActionClearTextFormat = class (TrvAction311)
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>333</sup>  
*TrvAction*<sup>311</sup>

## Description

This action resets text attributes in the selected fragment by calling `Editor.ApplyTextStyleTemplate(-1, True)`.

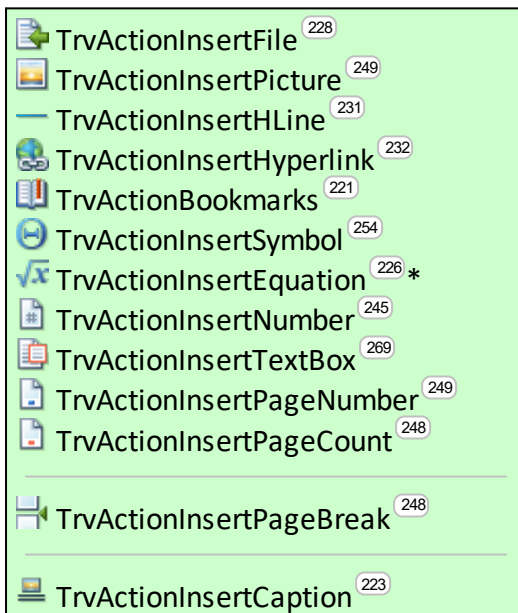
This action can be used even if `Editor.UseStyleTemplates=False`. In this case, it simply applies the 0th text style.

This action does not introduce any new properties in addition to properties<sup>312</sup> of `TrvAction`.

## 1.3.6 Insert

### Insertion Actions

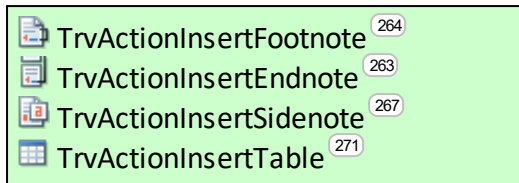
This group of actions includes commands to insert objects in the caret position, bookmark insertion and management.



\* the action is not included in RichViewActions directly, it is included in a separate package.



## See also



### 1.3.6.1 TrvActionBookmarks

TrvActionBookmarks is the action for "Insert Bookmark..." command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionBookmarks = class (TrvAction (311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

#### Description

This action displays a dialog window allowing to:

- insert a new checkpoint (bookmark) at the current position in the target editor
- delete existing checkpoints
- move the caret to an existing checkpoint

This action allows in bookmark names:

- alphanumeric characters;
- ' \_ ';
- characters included in AllowedCharacters <sup>(222)</sup>.

If you use this action with Report Workshop, assign '{}' to AllowedCharacters <sup>(222)</sup>.

#### 1.3.6.1.1 Properties

##### In TrvActionBookmarks

- AllowedCharacters <sup>(222)</sup>
- ScrollToCenter <sup>(222)</sup>

##### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>333</sup>

---

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.6.1.1.1 TrvActionBookmarks.AllowedCharacters

Specifies additional characters allowed in names of bookmarks.

**property** AllowedCharacters: TRVUnicodeString;

The action allows alphanumeric characters and underscores in bookmark names. To allow additional characters, include them in this property.

For example, if you use this action with Report Workshop, assign '{}' to allow field codes in bookmark names.

**Initial value:**

'' (empty string)

### 1.3.6.1.1.2 TrvActionBookmarks.ScrollToCenter

Specifies how the editor window is scrolled to show a checkpoint.

**property** ScrollToCenter: Boolean;

This property affects the behavior of "Go to" button in the bookmark dialog.

If *True*, the action scrolls the editor window to show the bookmarked item in the middle.

If *False*, the action scrolls the editor window to show the bookmarked item at the top.

**Default value:**

*True*

**See also:**

- `TrvActionInsertHyperlink(232).ScrollToCenter(222)`

### 1.3.6.2 TrvActionInsertCaption

`TrvActionInsertCaption` is the action for "Insert Object Caption..." command.

**Unit** `RichViewActions(92)`;

#### Syntax

```
TrvActionInsertCaption = class (TrvActionInsertNumSequence(246))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionInsertNumSequence(246)

```

#### Description

By default, this action allows inserting captions for pictures, "hot pictures" and tables. You can allow captions for additional items using `OnCanApply` event.

This action displays a dialog window allowing to enter caption for the object at the caret position.

The caption consists of the following parts:

- label
- "numbered sequence" item (name of its sequence=label)
- ' ' string
- used-defined text

**Example:** *Figure 1. Population dynamics of green zebras*

If `UseStyleTemplates=True` for the target editor, this caption is formatted using 'caption' style template. Otherwise, the current text style is used.

The following caption properties are supported:

- label (equal to a sequence name);
- numbering type (decimal, lower Roman, etc.);
- exclude label or not?
- caption text
- insert above or below?

If `UseDefaults(247)=True`, the dialog is initialized with a label 'Figure' (localized string), a decimal numbering type, `ExcludeLabel(224)` and `InsertAbove(225)` properties.

When the user clicks the "Insert" button, a caption is inserted above or below the selected object, `UseDefaults(247)` is reset to `False`, and chosen values are stored in `SeqName(247)`, `NumberType(247)`,

ExcludeLabel<sup>(224)</sup> and InsertAbove<sup>(225)</sup> properties (they are used to initialize a dialog when it will be displayed for the next time).

### 1.3.6.2.1 Properties

#### In TrvActionInsertCaption

---

ExcludeLabel<sup>(224)</sup>

InsertAbove<sup>(225)</sup>

#### Derived from TrvActionInsertNumSequence<sup>(246)</sup>

---

NumberType<sup>(247)</sup>

SeqName<sup>(247)</sup>

UseDefaults<sup>(247)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

■ Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

■ Caption<sup>(334)</sup>

■ ControlPanel<sup>(335)</sup>

■ Disabled<sup>(335)</sup>

■ Hint<sup>(335)</sup>

#### Derived from TAction

---

■ AutoCheck

■ Caption

■ Checked

DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

#### 1.3.6.2.1.1 TrvActionInsertCaption.ExcludeLabel

Specifies whether a label is included in a caption text.

**property** ExcludeLabel: Boolean;

This property is used to initialize a dialog. When a dialog is applied, this property is updated automatically.

Example of caption if `ExcludeLabel=False`:

*Table 1. Aging Statistics*

Example of caption if `ExcludeLabel=True`:

*1. Aging Statistics*

**Initial value:**

*False*

#### 1.3.6.2.1.2 TrvActionInsertCaption.InsertAbove

Specifies where a caption is inserted: above or below the object.

**property** `InsertAbove`: Boolean;

This property is used to initialize a dialog. When a dialog is applied, this property is updated automatically.

**Initial value:**

*False*

### 1.3.6.3 TrvActionInsertCustomPageNumber

`TrvActionInsertCustomPageNumber` is a base class for actions inserting page numbers and page counts.

**Unit** `RichViewActions` <sup>(92)</sup>;

**Syntax**

`TrvActionInsertCustomPageNumber = class (TrvAction (311))`

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

**Description**

This action is not used directly. It is a parent for the following actions:

- `TrvActionInsertPageNumber` <sup>(249)</sup>
- `TrvActionInsertPageCount` <sup>(248)</sup>

#### 1.3.6.3.1 Properties

##### In `TrvActionInsertCustomPageNumber`

`NumberType` <sup>(226)</sup>

## Derived from TrvAction <sup>(311)</sup>

---

■ Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>(333)</sup>

---

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

## Derived from TAction

---

■ AutoCheck

■ Caption

■ Checked

DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

### 1.3.6.3.1.1 TrvActionInsertCustomPageNumber.NumberType

A type of numbering for the inserted item (decimal, lower Roman, etc.)

**property** NumberType: TRVPageNumberType;

**Default value**

*rvpntDecimal*

### 1.3.6.4 TrvActionInsertEquation

TrvActionInsertEquation is the action for "Insert Equation..." command.

**Unit** RVMathActions;

**Syntax**

TrvActionInsertEquation = **class** (TrvAction <sup>(311)</sup>)

**Hierarchy**

*TObject*

*TPersistent*

*TComponent*

*TBasicAction*

*TContainedAction*

*TCustomAction*

*TAction*

*TrvCustomAction* <sup>(333)</sup>

*TrvAction* <sup>(311)</sup>

## Description

This action is not included in RichViewActions directly, it is included in a separate package.

This item uses Adit Math Engine by Denis Sletkov (Adit Software): ***aditsoftware.com***

Units of Adit Math Engine are included in TRichView installation (in Math folder). They are covered by **MPL 2.0** with the following addition restrictions:

*Adit Math Engine cannot be used in any E-learning/Assessment/Testing/Math software (Freeware or Shareware) or outside TRichView engine without our [Adit Software] written permission.*

This action inserts a mathematical equation (TRVMathItemInfo object) in the caret position. The user can edit equation before insertion, if *UserInterface* <sup>(228)</sup> = *True*.

The dialog window allows editing not only property of the new equation, but default properties (font name, size and color) of all equations in the current document.

### 1.3.6.4.1 Properties

#### In TrvActionInsertEquation

- Expression <sup>(228)</sup>
- UserInterface <sup>(228)</sup>

#### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType

- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.6.4.1.1 TrvActionInsertEquation.UserInterface

Specifies whether the action shows a dialog window for editing equation before the insertion.

**property** `UserInterface: Boolean;`

If **UserInterface** = *False*, the action inserts a new equation, assigning Expression <sup>(228)</sup> to its text.

If **UserInterface** = *False*, the action displays a dialog for editing properties of the new equation, initializing it with Expression <sup>(228)</sup>.

**Initial value:**

*True*

#### 1.3.6.4.1.2 TrvActionInsertEquation.Expression

Mathematical expression to insert.

**property** `Expression: TRVUnicodeString;`

This is a mathematical expression in LaTeX-like format.

**Initial value:**

'a+b=c'

**See also**

- UserInterface <sup>(228)</sup>

### 1.3.6.5 TrvActionInsertFile

TrvActionInsertFile is the action for "Insert | File" command.

**Unit** RichViewActions <sup>(92)</sup>;

**Syntax**

`TrvActionInsertFile = class (TrvActionCustomFileIO (96))`

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionCustomIO* <sup>(98)</sup>



*TrvActionCustomFileIO* <sup>(96)</sup>

### Description

This action displays a file opening dialog and inserts the chosen file at the caret position.

In addition to formats supported by *TrvActionOpen* <sup>(106)</sup> action, this action supports formats supported by Microsoft Office file import converters.

## 1.3.6.5.1 Properties

### In *TrvActionInsertFile*

■ Filter <sup>(230)</sup>

### Derived from *TrvActionCustomFileIO* <sup>(96)</sup>

■ CustomFilter <sup>(98)</sup>

### Derived from *TrvActionCustomIO* <sup>(98)</sup>

■ AutoUpdateFileName <sup>(99)</sup>

■ DialogTitle <sup>(100)</sup>

    FileName <sup>(100)</sup>

■ InitialDir <sup>(100)</sup>

### Derived from *TrvAction* <sup>(311)</sup>

■ Control <sup>(312)</sup>

### Derived from *TrvCustomAction* <sup>(333)</sup>

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

### Derived from *TAction*

■ AutoCheck

■ Caption

■ Checked

    DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

### 1.3.6.5.1.1 TrvActionInsertFile.Filter

Defines a set of file formats appearing in the file opening dialog.

**property** Filter: TrvFileImportFilterSet<sup>(387)</sup>;

If *ffiCustom* is included, formats listed in the CustomFilter<sup>(98)</sup> property are used.

Name and extension of RichView Format (RVF) may be customized, see GetControlPanel<sup>(336)</sup>.RVFFilter<sup>(52)</sup>.

**Default value:**

all formats

### 1.3.6.5.2 Methods

#### In TrvActionInsertFile

InsertFile<sup>(230)</sup>

#### Inherited from TrvCustomAction<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

### 1.3.6.5.2.1 TrvActionInsertFile.InsertFile

Inserts the specified file.

```
function InsertFile(rve: TCustomRichViewEdit;
  const FileName: TRVUnicodeString;
  FileFormat: TrvFileImportFilter(387);
  ConverterOrCustomIndex: Integer=0;
  rvc: TRVOfficeConverter=nil;
  Silent: Boolean = False): Boolean;
```

**Parameters:**

**rve** – editor where to insert file.

**FileName** – name of the file to insert.

**FileFormat** – format of this file.

**ConverterCustomFilterIndex** used only if **FileFormat** is *ffiCustom* or *ffiOfficeConverters*.

If **FileFormat** = *ffiCustom*, **ConverterCustomFilterIndex** identifies a custom file format. Custom formats are numbered from 1 in the order they are listed in the CustomFilter<sup>(98)</sup> property.

If **FileFormat** = *ffiOfficeConverters*, **ConverterCustomFilterIndex** identifies an import format, as an index in **rvc.ImportConverters**.

**rvc** is used if **FileFormat** = *ffiOfficeConverters*.

**Silent** – if **False**, the action displays an error message if the insertion was not successful.

**Return value:**

*True* if the insertion was successful.

If the insertion failed, this method displays an error message.

This method does not change values of the action's properties.

### 1.3.6.6 TrvActionInsertHLine

TrvActionInsertHLine is the action for "Insert | Horizontal Line" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertHLine = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action inserts a horizontal line ("break" item type) in the caret position in TCustomRichViewEdit component.

The inserted line has the specified Color<sup>(232)</sup> and Width<sup>(232)</sup>, and Style<sup>(232)</sup>.

#### 1.3.6.6.1 Properties

##### In TrvActionInsertHLine

- Color<sup>(232)</sup>
- Style<sup>(232)</sup>
- Width<sup>(232)</sup>

##### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

##### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

##### Derived from TAction

- AutoCheck
- Caption

- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.6.6.1.1 TrvActionInsertHLine.Color

Specifies the color for horizontal line.

**property** Color: TColor;

**Default value:**

*clWindowText*

#### 1.3.6.6.1.2 TrvActionInsertHLine.Style

Specifies the style for horizontal line.

**property** Style: TRVBreakStyle;

**Default value:**

*rvbsLine*

#### 1.3.6.6.1.3 TrvActionInsertHLine.Width

Specifies the width for horizontal line.

**property** Width: TRVStyleLenth;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

**Default value:**

1

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

### 1.3.6.7 TrvActionInsertHyperlink

TrvActionInsertHyperlink is the action for "Insert | Hyperlink" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionInsertHyperlink = **class** (TrvActionTextStyles<sup>(173)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTextStyles* <sup>(173)</sup>

## Description

This action can perform the following tasks:

- inserting a new text hyperlink, with the specified target and text;
- converting the selected text and (optionally) pictures to hyperlinks (with the specified target);
- removing hyperlink from the selected fragment.

When this action is executed, it displays a dialog for defining hyperlink attributes. But before the dialog is shown, the action adjusts the selection in *TCustomRichViewEdit*: if the selection includes a hyperlink partially (or if there is no selection but the caret is inside a link), this hyperlink is selected completely. If the selection is still empty, the action will insert a new hyperlink. Otherwise it will convert the selected fragment to hyperlinks (or will change targets for existing hyperlinks in the selection). If the user will enter an empty target, all links will be removed instead.

By default, the action stores targets of hyperlinks in items' tags. You can override this behavior using *OnApplyHyperlinkToItem* <sup>(244)</sup> and *OnGetHyperlinkTargetFromItem* <sup>(244)</sup> events.

## Using style templates

Visual appearance of a hyperlink depends on whether style templates are used <sup>(20)</sup>.

**If style templates are not used:** the action allows defining text and background colors for hyperlinks (including "hover" colors). The chosen colors will be applied to all new links. When converting a text to a hypertext, properties listed in *ValidProperties* <sup>(239)</sup> are applied to its style (and the original style is assigned to its *NextStyleNo* property).

When converting a hypertext to a text, the same set of properties are changed. But instead of properties of this action, properties of *RVStyle.TextStyles[0]* are applied (and -1 is assigned to *NextStyleNo* property).

The dialog window has a button invoking a new dialog for changing the link color.

**If style templates are used:** the action applies *StyleTemplateName* <sup>(239)</sup> to hyperlinks. When converting a hypertext to a text, the action clears a link to style template.

The dialog window has a combo-box allowing to choose a style template. Only style templates having *Kind=rvstkText* are listed.

## Methods

In addition to the operations performed on the execution, the action has several useful methods:

- *GetHyperlinkStyleNo* <sup>(241)</sup> returns the index of hypertext style for the given text style;
- *GetNormalStyleNo* <sup>(242)</sup> returns the index of non-hypertext style for the given text style;

- DetectURL<sup>(240)</sup> analyses text to the left of the caret; if it is URL, converts it to hyperlink;
- TerminateHyperlink<sup>(243)</sup> converts the current text style to non-hypertext, if the caret is at the end of hyperlink;
- GoToLink<sup>(242)</sup> opens the hyperlink target in the external browser; for local links, it moves the caret to the target *checkpoint*.

### 1.3.6.7.1 Properties

#### In TrvActionInsertHyperlink

---

- ActionStyleTemplates<sup>(235)</sup>
- AutoAddHyperlinkStyleTemplate<sup>(235)</sup>
- BackColor<sup>(235)</sup>
- Color<sup>(236)</sup>
- Cursor<sup>(236)</sup>
- HoverBackColor<sup>(236)</sup>
- HoverColor<sup>(237)</sup>
- HoverEffects<sup>(237)</sup>
- HoverUnderlineColor<sup>(237)</sup>
- ScrollToCenter<sup>(238)</sup>
- SetToPictures<sup>(238)</sup>
- SpaceFiller<sup>(238)</sup>
- Style<sup>(238)</sup>
- StyleTemplateName<sup>(239)</sup>
- UnderlineColor<sup>(239)</sup>
- ValidProperties<sup>(239)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType

- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.6.7.1.1 TrvActionInsertHyperlink.ActionStyleTemplates

Links this action to TrvActionStyleTemplates<sup>(212)</sup> action.

**property** ActionStyleTemplates: TrvActionStyleTemplates<sup>(212)</sup>;

This property is used only if style templates are used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=True).

If this link is not defined, the first found TrvActionStyleTemplates<sup>(212)</sup> action on the same form/datamodule is used.

See StyleTemplateName<sup>(239)</sup> for details.

#### 1.3.6.7.1.2 TrvActionInsertHyperlink.AutoAddHyperlinkStyleTemplate

Defines whether the action may add "Hyperlink" style template.

**property** AutoAddHyperlinkStyleTemplate: Boolean;

This property is used only if style templates are used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=True).

See StyleTemplateName<sup>(239)</sup> for details.

##### Default value:

True

#### 1.3.6.7.1.3 TrvActionInsertHyperlink.BackColor

Defines the background color for hyperlinks.

**property** BackColor: TColor;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=False). If style templates are used, the action applies StyleTemplateName<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if *rvhlBackColor* is included in ValidProperties<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

##### Default value:

*c/None*

#### 1.3.6.7.1.4 TrvActionInsertHyperlink.Color

Defines the text color for hyperlinks.

**property** Color: TColor;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=False). If style templates are used, the action applies StyleTemplateName<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if *rvhlColor* is included in ValidProperties<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

*clBlue*

#### 1.3.6.7.1.5 TrvActionInsertHyperlink.Cursor

Defines the hypertext cursor for the links.

**property** Cursor: TCursor;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=False). If style templates are used, the action applies StyleTemplateName<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if *rvhlCursor* is included in ValidProperties<sup>(239)</sup>.

**Default value:**

*crJump*

#### 1.3.6.7.1.6 TrvActionInsertHyperlink HoverBackColor

Defines the background color for hyperlinks under the mouse pointer.

**property** HoverBackColor: TColor;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=False). If style templates are used, the action applies StyleTemplateName<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if *rvhlHoverBackColor* is included in ValidProperties<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

*clNone* (transparent)



#### 1.3.6.7.1.7 TrvActionInsertHyperlink HoverColor

Defines the text color for hyperlinks under the mouse pointer.

**property** `HoverColor: TColor;`

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's `UseStyleTemplates=False`). If style templates are used, the action applies `StyleTemplateName`<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if `rvhlHoverColor` is included in `ValidProperties`<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

`c/Blue`

#### 1.3.6.7.1.8 TrvActionInsertHyperlink HoverEffects

Defines the effects (such as underlining) for hyperlinks under the mouse pointer.

**property** `HoverEffects: TRVHoverEffects;`

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's `UseStyleTemplates=False`). If style templates are used, the action applies `StyleTemplateName`<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if `rvhlHoverEffects` is included in `ValidProperties`<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

`[]`

#### 1.3.6.7.1.9 TrvActionInsertHyperlink HoverUnderlineColor

Defines the underline color for hyperlinks under the mouse pointer.

**property** `HoverUnderlineColor: TColor;`

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's `UseStyleTemplates=False`). If style templates are used, the action applies `StyleTemplateName`<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if `rvhlHoverUnderlineColor` is included in `ValidProperties`<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

`c/None` (instructs using `UnderlineColor`<sup>(239)</sup>)

### 1.3.6.7.1.10 TrvActionInsertHyperlink.ScrollToCenter

Specifies how the editor window is scrolled to show a checkpoint.

**property** `ScrollToCenter`: `Boolean`;

This property affect the behavior of `GoToLink`<sup>(242)</sup> method.

If *True*, the action scrolls the editor window to show the bookmarked item in the middle.

If *False*, the action scrolls the editor window to show the bookmarked item at the top.

**Default value:**

*True*

**See also:**

- `TrvActionBookmarks`<sup>(221)</sup>.`ScrollToCenter`<sup>(222)</sup>

### 1.3.6.7.1.11 TrvActionInsertHyperlink.SetToPictures

Specifies whether this action should convert pictures to hot-pictures and vice versa.

**property** `SetToPictures`: `Boolean`;

If this property is *False*, the action does nothing with the selected pictures.

**Default value:**

*True*

### 1.3.6.7.1.12 TrvActionInsertHyperlink.SpaceFiller

Specifies the string used for replacing space characters in hyperlink targets (by `EncodeTarget`<sup>(241)</sup>).

**property** `SpaceFiller`: `TRVUnicodeString`;

**Default value:**

' ' (space character, so nothing is replaced)

### 1.3.6.7.1.13 TrvActionInsertHyperlink.Style

Defines the font style for hyperlinks.

**property** `Style`: `TFontStyles`;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's `UseStyleTemplates=False`). If style templates are used, the action applies `StyleTemplateName`<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if *rvhlBold*, *rvhlItalic*, *rvhlUnderline*, or *rvhlStrikeout* are included in `ValidProperties`<sup>(239)</sup>.

For example:

- if *rvhlBold* is included in `ValidProperties`<sup>(239)</sup> and *fsBold* is included in `Style`, hyperlinks will be bold;
- if *rvhlBold* is included in `ValidProperties`<sup>(239)</sup> and *fsBold* is not included in `Style`, hyperlinks will not be bold;
- if *rvhlBold* is not included in `ValidProperties`<sup>(239)</sup> (default), boldness is not changed.

**Default value:**

*[fsUnderline]*

#### 1.3.6.7.1.14 TrvActionInsertHyperlink.StyleTemplateName

Defines the name of a style template for applying to hyperlinks.

**property** StyleTemplateName: TRVUnicodeString;

This property is used only if style templates are used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=True).

This property may be changed by an user in the dialog window.

This style template must be a text style template (Kind=rvskText), otherwise it will not be listed in a dialog window.

To get a style template for a hyperlink, the action performs the following steps:

1. The action searches in the target editor's Style.StyleTemplates for the style template named **StyleTemplateName**. If found, it is used. Otherwise,
2. The action searches for "Hyperlink" style template. If found, it is used. Otherwise,
3. If AutoAddHyperlinkStyleTemplate<sup>(235)</sup>=True, the action adds "Hyperlink" style template to the target editor's Style.StyleTemplates. Properties of this new style templates may be taken from the linked<sup>(235)</sup> TrvActionStyleTemplates<sup>(212)</sup> action, if it has "Hyperlink" in its StandardStyleTemplates<sup>(214)</sup>; otherwise, the action uses hard-coded default values.

**Default value:**

'Hyperlink'

#### 1.3.6.7.1.15 TrvActionInsertHyperlink.UnderlineColor

Defines the underline color for hyperlinks.

**property** UnderlineColor: TColor;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's UseStyleTemplates=False). If style templates are used, the action applies StyleTemplateName<sup>(239)</sup> instead.

This property is used when converting text to hypertext, if rvhlHoverUnderlineColor is included in ValidProperties<sup>(239)</sup>.

A value of this property can be changed by the user in the hyperlink properties dialog.

**Default value:**

c/None (instructs using Color<sup>(236)</sup>)

#### 1.3.6.7.1.16 TrvActionInsertHyperlink.ValidProperties

Defines properties distinguishing hyperlinks from normal text, if style templates are not used.

**type**

```
TRVHyperlinkProperty = (rvhlBold, rvhlItalic, rvhlUnderline,
    rvhlUnderlineColor, rvhlStrikeout, rvhlColor, rvhlBackColor,
    rvhlHoverColor, rvhlHoverBackColor, rvhlHoverUnderlineColor,
    rvhlHoverUnderline, rvhlCursor, rvhlJump);
TRVHyperlinkProperties = set of TRVHyperlinkProperty;
```

**property** ValidProperties: TRVHyperlinkProperties;

This property is used only if style templates are not used<sup>(20)</sup> (i.e. the target editor's `UseStyleTemplates=False`). If style templates are used, the action applies `StyleTemplateName`<sup>(239)</sup> instead.

When this action converts text to hypertext, properties of this action listed in this property are applied:

- `BackColor`<sup>(235)</sup>
- `Color`<sup>(236)</sup>
- `HoverBackColor`<sup>(236)</sup>
- `HoverColor`<sup>(237)</sup>
- `HoverEffects`<sup>(237)</sup>
- `HoverUnderlineColor`<sup>(237)</sup>
- `UnderlineColor`<sup>(239)</sup>
- `Style`<sup>(238)</sup>

When this action converts text to non-hypertext, properties of `RVStyle.TextStyles[0]` listed in this property are applied.

#### Default value:

`[rvhlUnderline, rvhlColor, rvhlBackColor, rvhlHoverColor, rvhlHoverBackColor, rvhlCursor, rvhlJump]`  
(hyperlink with the standard hyperlink cursor, blue and underlined)

### 1.3.6.7.2 Methods

#### In TrvActionInsertHyperlink

`DetectURL`<sup>(240)</sup>  
`EncodeTarget`<sup>(241)</sup>  
`GetHyperlinkStyleNo`<sup>(241)</sup>  
`GetNormalStyleNo`<sup>(242)</sup>  
`GoToLink`<sup>(242)</sup>  
`SetTags`<sup>(243)</sup>  
`TerminateHyperlink`<sup>(243)</sup>

#### Inherited from TrvCustomAction<sup>(333)</sup>

`GetControlPanel`<sup>(336)</sup>

##### 1.3.6.7.2.1 TrvActionInsertHyperlink.DetectURL

Helps to detect URLs on typing in **rve**.

```
procedure DetectURL(rve: TCustomRichViewEdit);
```

This action checks the text to the left of the caret position, and if its a URL, converts it to a hyperlink.

The function does nothing if the text is already a hyperlink, or if the selection in **rve** is not empty.

The method uses `GetHyperlinkStyleNo`<sup>(241)</sup> for this conversion.

#### Example [VCL]:

```
procedure TForm1.RichViewEdit1KeyPress(Sender: TObject; var Key: Char);  
begin
```

```

if Key in [#9, ' ', ',', ';'] then begin
    rvActionInsertHyperlink1.DetectURL(RichViewEdit1);
    rvActionInsertHyperlink1.TerminateHyperlink(243)(RichViewEdit1);
end;
end;

```

You should call this method:

- in OnKeyPress event (in OnUTF8KeyPress for Lazarus) when the user types tab, space, comma or semicolon characters
- in OnKeyDown event when the user pressed ENTER

#### 1.3.6.7.2.2 TrvActionInsertHyperlink.EncodeTarget

Encodes hypertext target before assigning it to the item's tag.

```
function EncodeTarget(const Target: TRVUnicodeString): TRVUnicodeString;
```

The function returns the processed **Target** parameter.

It removes the line break characters (#13 and #10) and replaces the space characters with SpaceFiller<sup>(238)</sup>.

Normally, the returned value is the same as **Target** (because the action does not allow entering multiline targets, and SpaceFiller<sup>(238)</sup>=' ').

#### 1.3.6.7.2.3 TrvActionInsertHyperlink.GetHyperlinkStyleNo

Returns the index of hypertext style for the given style.

```
function GetHyperlinkStyleNo(rve: TCustomRichViewEdit;
    StyleNo: Integer=-1): Integer;
```

If **StyleNo**=-1, the action returns the index of hypertext style for **rve.CurTextStyleNo**, otherwise for **StyleNo**.

The appearance of the returned style depends on **rve.UseStyleTemplates**.

If style templates are not used<sup>(20)</sup>, the method returns the index of style (in **rve.Style.TextStyles**) having all properties of the specified style, except for the properties defined in ValidProperties<sup>(239)</sup>. For these properties, values specified in properties of this action are used.

If style templates are used<sup>(20)</sup>, the method returns the index of style (in **rve.Style.TextStyles**) having all properties of the specified style, by with the style template StyleTemplateName<sup>(239)</sup> applied.

If a style with the required properties already exists in **rve.Style.TextStyles**, its index is returned. Otherwise, a new style is created and its index is returned.

This method is used in DetectURL<sup>(240)</sup> method, or when inserting/applying hyperlink. You can use it in other cases when you need a hypertext style index.

For example, in TCustomRichViewEdit.OnReadHyperlink event:

```

procedure TForm1.RichViewEdit1ReadHyperlink(Sender: TCustomRichView;
    const Target, Extras: TRVUnicodeString; DocFormat: TRVLoadFormat;
    var StyleNo: Integer; var ItemTag: TRVTag;
    var ItemName: TRVUnicodeString);
begin
    if DocFormat=rvlfURL then

```

```

    StyleNo := rvActionInsertHyperlink1.GetHyperlinkStyleNo(
        RichViewEdit1);
    ItemTag := rvActionInsertHyperlink1.EncodeTarget(241)(Target);
end;

```

#### See also methods:

- GetNormalStyleNo<sup>(242)</sup>

#### 1.3.6.7.2.4 TrvActionInsertHyperlink.GetNormalStyleNo

Returns the index of non-hypertext style for the given style.

```

function GetNormalStyleNo(rve: TCustomRichViewEdit;
    StyleNo: Integer=-1): Integer;

```

If **StyleNo**=-1, the action returns the index of non-hypertext style for **rve.CurTextStyleNo**, otherwise for **StyleNo**.

The appearance of the returned style depends on **rve.UseStyleTemplates**.

If style templates are not used<sup>(20)</sup>, the method returns the index of style (in **rve.Style.TextStyles**) having all properties of the specified style, except for the properties defined in **ValidProperties**<sup>(239)</sup>. For these properties, values of properties of **rve.Style.TextStyles[0]** are used.

If style templates are used<sup>(20)</sup>, the method returns the index of style (in **rve.Style.TextStyles**) having all properties of the specified style, by with link to a style template cleared.

If a style with the required properties already exists in **rve.Style.TextStyles**, its index is returned. Otherwise, a new style is created and its index is returned.

This method is used in **TerminateHyperlink**<sup>(243)</sup> method, or when user entered an empty target in the dialog.

#### See also methods:

- GetHyperlinkStyleNo<sup>(241)</sup>

#### 1.3.6.7.2.5 TrvActionInsertHyperlink.GoToLink

These methods execute the given hyperlink.

```

procedure GoToLink(rve: TCustomRichViewEdit; id: Integer);
procedure GoToLink(rve: TCustomRichViewEdit;
    const Target: TRVUnicodeString);

```

The first version of the method should be called in **rve.OnJump** event:

```

procedure TForm3.RichViewEdit1Jump(Sender: TObject; id: Integer);
begin
    rvActionInsertHyperlink1.GoToLink(RichViewEdit1, id);
end;

```

The second version of this method can be called from **rve.OnJump** or from any place of code.

If the link's target (stored in its tag) is started from '#', the action moves the caret to the *checkpoint* of the same name (but without '#'). If ScrollToCenter<sup>(238)</sup> = True, the bookmarked item is centered in the editor.

Otherwise, the action opens the link's target in external application (using ShellExecute function).

On error (no such checkpoint, or ShellExecute returned an error code), an error message is displayed.

**See also:**

- GoToCheckpoint<sup>(377)</sup> function

#### 1.3.6.7.2.6 TrvActionInsertHyperlink.SetTags

Applies the action to the selection.

```
procedure SetTags(rve: TCustomRichViewEdit;  
  const Target: TRVUnicodeString);
```

Normally, calling this method is not necessary. It is called automatically when the action applies **Target** to the selection.

If **Target** is not empty, the action converts the selection to hyperlink(s).

If **Target** is empty, the action removes hyperlinks from the selection.

#### 1.3.6.7.2.7 TrvActionInsertHyperlink.TerminateHyperlink

Helps to terminate hyperlinks when typing in **rve**.

```
procedure TerminateHyperlink(rve: TCustomRichViewEdit);
```

If the selection is empty, the current text item is a hyperlink, and the caret is at the end of this item, this function assigns the index non-hypertext style to the current text style (**rve.CurTextStyleNo**). GetNormalStyleNo<sup>(242)</sup> is used.

**Example [VCL]:**

```
procedure TForm1.RichViewEdit1KeyPress(Sender: TObject; var Key: Char);  
begin  
  if Key in [#9, ' ', ',', ';'] then begin  
    rvActionInsertHyperlink1.DetectURL(240)(RichViewEdit1);  
    rvActionInsertHyperlink1.TerminateHyperlink(RichViewEdit1);  
  end;  
end;
```

When the user types something at the end of a hyperlink, new characters are added to the hyperlink's text. But when he/she types a space character (or tab, or comma, or Enter, or semicolon), the current text style becomes a non-hypertext, so this and subsequent characters will not be added to the hyperlink's text.

You should call this method:

- in OnKeyPress event (in OnUTF8KeyPress for Lazarus) when the user types tab, space, comma or semicolon characters
- in OnKeyDown event when the user pressed ENTER.

### 1.3.6.7.3 Events

#### In TrvActionInsertHyperlink

- OnApplyHyperlinkToItem<sup>(244)</sup>
- OnGetHyperlinkTargetFromItem<sup>(244)</sup>
- OnHyperlinkForm<sup>(244)</sup>

##### 1.3.6.7.3.1 TrvActionInsertHyperlink.OnApplyHyperlinkToItem

Occurs when converting item of unknown type to/from hyperlink

###### type

```
TRVApplyHyperlinkToItemEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit; RVData: TCustomRVData; ItemNo: Integer;  
    const Target: TRVUnicodeString) of object;
```

**property** OnApplyHyperlinkToItem: TRVApplyHyperlinkToItemEvent;

For text and pictures, the action applies hypertext (or removes hypertext) automatically.

For other types of items, this event occurs.

The item is defined by **Editor** and **ItemNo** (it is the **ItemNo**-th item in **Editor**, where **Editor** can be a main editor or cell inplace editor). **RVData** is **Editor.RVData**.

When converting to hypertext, **Target** is not empty.

When converting from hypertext, **Target** is empty.

In this event, you can perform an editing operation on this item.

##### 1.3.6.7.3.2 TrvActionInsertHyperlink.OnGetHyperlinkTargetFromItem

Occurs when the action needs to get a hyperlink target from item of unknown type.

###### type

```
TRVGetHyperlinkTargetFromItem = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit; RVData: TCustomRVData; ItemNo: Integer;  
    var Target: TRVUnicodeString) of object;
```

**property** OnGetHyperlinkTargetFromItem: TRVGetHyperlinkTargetFromItem;

For text and pictures, the action gets targets automatically. For items of other types, this event occurs.

If you return empty **Target**, the action will treat this item as non-hypertext.

The item is defined by **Editor** and **ItemNo** (it is the **ItemNo**-th item in **Editor**, where **Editor** can be a main editor or cell inplace editor). **RVData** is **Editor.RVData**.

##### 1.3.6.7.3.3 TrvActionInsertHyperlink.OnHyperlinkForm

Allows displaying your own hypertext dialog.

###### type

```
TRVHyperlinkFormEvent = procedure (Sender: TObject; InsertNew: Boolean;  
    var Text, Target: TRVUnicodeString; var Proceed: Boolean) of object;
```



**property** OnHyperlinkForm: TRVHyperlinkFormEvent;

Use this event if you want to replace the default dialog used in this action.

**Input parameters:**

If **InsertNew**=*True*, this event is called for inserting a new hyperlink.

**Text** – the selected text, valid only if **InsertNew**=*False*.

**Target** – hypertext target of selected text, use only if **InsertNew**=*False*. It is not empty only in case when the selection contains only hyperlinks with equal targets.

**Proceed** = *True*.

**Output parameters:**

**Text** – text for new hyperlink, used only if **InsertNew**=*True*.

**Target** – hyperlink target. You can return empty string only if **InsertNew**=*False* (to remove hyperlinks from the selection)

**Proceed** must be *True* if the user pressed "OK", and *False* if the user pressed "Cancel".

### 1.3.6.8 TrvActionInsertNumber

TrvActionInsertNumber is the action for "Insert Number..." command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionInsertNumber = **class** (TrvActionInsertNumSequence<sup>(246)</sup>)

**Hierarchy**

TObject  
 TPersistent  
 TComponent  
 TBasicAction  
 TContainedAction  
 TCustomAction  
 TAction  
 TrvCustomAction<sup>(333)</sup>  
 TrvAction<sup>(311)</sup>  
 TrvActionInsertNumSequence<sup>(246)</sup>

**Description**

This action displays a dialog window allowing to enter properties of an item to insert.

The following properties are supported:

- sequence name (identifier);
- numbering type (decimal, lower Roman, etc.);
- continue numbering / start from the specified value.

If UseDefaults<sup>(247)</sup>=*True*, the dialog is initialized with a sequence name 'Numbering' (localized string) and a decimal numbering type.

When the user clicks the "Insert" button, a "numbered sequence" item is inserted in the caret position, `UseDefaults`<sup>(247)</sup> is reset to *False*, and chosen values are stored in `SeqName`<sup>(247)</sup> and `NumberType`<sup>(247)</sup> properties (they are used to initialize a dialog when it will be displayed for the next time).

### 1.3.6.9 TrvActionInsertNumSequence

`TrvActionInsertNumSequence` is a base class for the actions inserting a "numbered sequence" item.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertNumSequence = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action is not used directly.

The following actions are inherited from it:

- `TrvActionInsertNumber`
- `TrvActionInsertCaption`

#### 1.3.6.9.1 Properties

##### In `TrvActionInsertNumSequence`

`NumberType`<sup>(247)</sup>  
`SeqName`<sup>(247)</sup>  
`UseDefaults`<sup>(247)</sup>

##### Derived from `TrvAction`<sup>(311)</sup>

■ `Control`<sup>(312)</sup>

##### Derived from `TrvCustomAction`<sup>(333)</sup>

■ `Caption`<sup>(334)</sup>  
 ■ `ControlPanel`<sup>(335)</sup>  
 ■ `Disabled`<sup>(335)</sup>  
 ■ `Hint`<sup>(335)</sup>

---

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.6.9.1.1 TrvActionInsertNumSequence.NumberType

A type of numbering for the inserted item (decimal, lower Roman, etc.)

**property** `NumberType: TRVSeqType;`

If `UseDefaults`<sup>(247)</sup> = *False*, this property is used to initialize a numbering type field in a dialog.

This property is updated automatically to the value entered in this field, when this dialog is applied.

### 1.3.6.9.1.2 TrvActionInsertNumSequence.SeqName

A name (identifier) of the numbered sequence. The inserted item will continue numbering started by items having the same sequence name.

**property** `SeqName: TRVUnicodeString;`

If `UseDefaults`<sup>(247)</sup> = *False*, this property is used to initialize a sequence name field in a dialog.

This property is updated automatically to the value entered in this field, when this dialog is applied.

The following sequence names are not allowed (the "Insert" button is disabled if they are entered):

- empty strings
- strings started from '@'

### 1.3.6.9.1.3 TrvActionInsertNumSequence.UseDefaults

Specifies whether values of `SeqName`<sup>(247)</sup> and `NumberType`<sup>(247)</sup> properties are used.

**property** `UseDefaults: Boolean`

When this property is *True*, the following values are used to initialize a dialog:

- sequence name = 'Numbering' (localized) for `TrvActionInsertNumber` or 'Figure' (localized) for `TrvActionInsertCaption`;
- numbering type = decimal.

When this property is *False*, values of `SeqName`<sup>(247)</sup> and `NumberType`<sup>(247)</sup> properties are used instead.

After an insertion, this property is reset to *False* automatically.

**Initial value:**

*True*

### 1.3.6.10 TrvActionInsertPageBreak

TrvActionInsertPageBreak is the action for "Insert Page Break" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionInsertPageBreak = class (TrvAction(311))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

**Description**

This action inserts a page break in the caret position (or, when called from a table cell, before the current row).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

**See also:**

- TrvActionRemovePageBreak<sup>(330)</sup>

### 1.3.6.11 TrvActionInsertPageCount

TrvActionInsertPageCount is the action for "Insert | Page Count" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionInsertPageCount = class (TrvActionInsertCustomPageNumber(225))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>

*TrvAction* <sup>(311)</sup>*TrvActionInsertCustomPageNumber* <sup>(225)</sup>**Description**

This action inserts "page count" field in the position of the caret. The number is formatted according to *NumberType* <sup>(226)</sup> property.

**See also**

- *TrvActionInsertPageNumber* <sup>(249)</sup>

**1.3.6.12 TrvActionInsertPageNumber**

*TrvActionInsertPageNumber* is the action for "Insert | Page Number" command.

**Unit** *RichViewActions* <sup>(92)</sup> ;

**Syntax**

```
TrvActionInsertPageNumber = class (TrvActionInsertCustomPageNumber (225))
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionInsertCustomPageNumber (225)

```

**Description**

This action inserts "page number" field in the position of the caret. The number is formatted according to *NumberType* <sup>(226)</sup> property.

**See also**

- *TrvActionInsertPageCount* <sup>(248)</sup>

**1.3.6.13 TrvActionInsertPicture**

*TrvActionInsertPicture* is the action for "Insert | Picture" command.

**Unit** *RichViewActions* <sup>(92)</sup> ;

**Syntax**

```
TrvActionInsertPicture = class (TrvActionCustomIO (98))
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction

```

*TContainedAction*

*TCustomAction*

*TAction*

*TrvCustomAction* <sup>(333)</sup>

*TrvAction* <sup>(311)</sup>

*TrvActionCustomIO* <sup>(98)</sup>

## Description

This action inserts pictures from files at the caret position. Unlike all other file opening actions, this action detects file format by the file extension, not by the file filter index.

The action calls `GetControlPanel` <sup>(336)</sup>.`OnChoosePicture` <sup>(61)</sup> event (if assigned). If the event provides an image, the action inserts it.

Otherwise, the action displays `TOpenPictureDialog` to choose pictures. The action allows users to select more than one file in the dialog. Then it inserts chosen files one by one.

The following additional properties can be applied to inserted pictures:

- `VAlign` <sup>(253)</sup> – the image alignment relative to text;
- `Spacing` <sup>(253)</sup> – padding (spacing between the image itself and its border);
- `OuterHSpacing`, `OuterVSpacing` <sup>(253)</sup> – horizontal and vertical spacing around the image border;
- `BorderWidth`, `BorderColor` <sup>(251)</sup> – border width and color;
- `BackgroundColor` <sup>(251)</sup> – background color;
- `MaxImageSize` <sup>(252)</sup> – maximal allowed size (the image will be stretched if it exceeds this size);
- `StoreFileNameInItemName` <sup>(253)</sup> allows storing file name with the image (It's not recommended to use this option. The recommended place for storing image file names is *rvesplImageFileName* extra item property. It is assigned if *rvoAssignImageFileNames* is included in the Options property of the target editor).

You can modify a picture before insertion in `OnInserting` <sup>(254)</sup> event.

### 1.3.6.13.1 Properties

#### In TrvActionInsertPicture

- `BackgroundColor` <sup>(251)</sup>
- `BorderColor` <sup>(251)</sup>
- `BorderWidth` <sup>(251)</sup>
- `DefaultExt` <sup>(252)</sup>
- `Filter` <sup>(252)</sup>
- `MaxImageSize` <sup>(252)</sup>
- `OuterHSpacing` <sup>(253)</sup>
- `OuterVSpacing` <sup>(253)</sup>
- `Spacing` <sup>(253)</sup>
- `StoreFileNameInItemName` <sup>(253)</sup>

- VAlign<sup>(253)</sup>

### Derived from TrvActionCustomIO<sup>(98)</sup>

- AutoUpdateFileName<sup>(99)</sup>
- DialogTitle<sup>(100)</sup>
- FileName<sup>(100)</sup>
- InitialDir<sup>(100)</sup>

### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.6.13.1.1 TrvActionInsertPicture.BackgroundColor

The property specify a background color for inserted images.

**property** BackgroundColor: TColor;

#### Default value

clNone (no background color)

#### 1.3.6.13.1.2 TrvActionInsertPicture.BorderWidth, BorderColor

The property specify a border width and color for inserted images.

**property** BorderWidth: TRVStyleLength;

**property** BorderColor: TColor;

Assign a positive value to BorderWidth to add border to inserted pictures.

BorderWidth is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

**Default value**

- BorderWidth: 0
- BorderColor: clBlack

#### 1.3.6.13.1.3 TrvActionInsertPicture.DefaultExt

Specifies the default file extension.

**property** DefaultExt: TRVALocString;

DefaultExt specifies a file extension that is appended automatically to the selected file name, unless the selected file name already includes a registered extension. If the user selects a file name with an extension that is unregistered, DefaultExt is appended to the unregistered extension.

This property is assigned to DefaultExt property of the file opening dialog (TOpenPictureDialog).

**Default value:**

'bmp'

#### 1.3.6.13.1.4 TrvActionInsertPicture.Filter

Determines the file masks (filters) available in the file opening dialog.

**property** Filter: TRVALocString;

If this property is not empty, it is assigned to the Filter property of the file opening dialog (TOpenPictureDialog). Otherwise, a filter containing all registered graphic formats is created.

**Default value:**

'' (empty string)

**See also:**

- TrvActionItemProperties.GraphicFilter<sup>(328)</sup>
- TrvActionItemProperties.BackgroundGraphicFilter<sup>(327)</sup>
- TrvActionTableProperties.BackgroundGraphicFilter<sup>(303)</sup>

#### 1.3.6.13.1.5 TrvActionInsertPicture.MaxImageSize

Defines the maximal image size for inserting without scaling.

**property** MaxImageSize: TRVStyleLength;

A positive value limits sizes of inserted images. If width or height of the inserted picture exceeds this value, the image is scaled down proportionally. The actual image is not changed, it's just inserted scaled (its *rveplImageHeight* and *rveplImageWidth* properties are assigned, see the help file on TRVExtralItemProperty).

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

**Default value:**

0 (no scaling)



#### 1.3.6.13.1.6 TrvActionInsertPicture.OuterHSpacing, OuterVSpacing

Specifies outer spacing for inserted images (spacing around the image border)

**property** OuterHSpacing: TRVStyleLength;

**property** OuterVSpacing: TRVStyleLength;

OuterHSpacing defines a horizontal spacing, OuterVSpacing defines a vertical spacing,

Assign a positive values to these property to add spacing around inserted pictures.

These values are measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

##### Default value

0

#### 1.3.6.13.1.7 TrvActionInsertPicture.Spacing

Specifies padding for inserted images (spacing between the image and its border)

**property** Spacing: TRVStyleLength;

Assign a positive value to this property to add padding for inserted pictures.

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

##### Default value

0

#### 1.3.6.13.1.8 TrvActionInsertPicture.StoreFileNameInItemName

Specifies whether file names (full paths) are stored in item names of inserted pictures.

**property** StoreFileNameInItemName: Boolean;

If this property is *True*, path to the graphic file is stored in the document in the item name (item text).

It's not recommended to use this option. The recommended place for storing image file names is *rvsplImageFileName* extra item property. It is assigned if *rvoAssignImageFileNames* is included in the Options property of the target editor.

##### Default value

*False*

##### See also:

- TrvActionPasteSpecial.StoreFileNameInItemName<sup>(138)</sup>
- TrvActionItemProperties<sup>(324)</sup> (does not support storing paths in item names)
- RTF reading (does not support storing paths in item names)

#### 1.3.6.13.1.9 TrvActionInsertPicture.VAlign

Specifies the vertical alignment for inserted pictures.

**property** VAlign: TRVVAlign;

##### Default value:

*rvvaBaseline*

### 1.3.6.13.2 Events

#### In TrvActionInsertPicture

##### ■ OnInserting<sup>(254)</sup>

##### 1.3.6.13.2.1 TrvActionInsertPicture.OnInserting

Occurs before inserting image in the target editor.

##### type

```
TRVInsertPictureEvent = procedure (Sender: TObject;
  Editor: TCustomRichViewEdit; var Graphic: TGraphic;
  const FileName: TRVUnicodeString)
of object;
```

**property** OnInserting: TRVInsertPictureEvent;

##### Parameters

**Graphic** – picture that will be inserted in **Editor**.

**FileName** – name of file, from where **Graphic** was read.

You can:

- modify the content of **Graphic**, or
- free **Graphic** and assign a new graphic object to it, or
- free **Graphic** and assign *nil* to it (to prevent insertion)

### 1.3.6.14 TrvActionInsertSymbol

TrvActionInsertSymbol is the action for "Insert | Symbol" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionInsertSymbol = class (TrvAction(311))
```

##### Hierarchy

```
TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
```

##### Description

This action inserts a character in the caret position of the target editor.

### 1.3.6.14.1 Properties

#### In TrvActionInsertSymbol

---

CharCode <sup>(255)</sup>

FontName <sup>(256)</sup>

■ Protection <sup>(256)</sup>

■ UseCurrentFont <sup>(256)</sup>

#### Derived from TrvAction <sup>(311)</sup>

---

■ Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

---

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

#### Derived from TAction

---

■ AutoCheck

■ Caption

■ Checked  
DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

#### 1.3.6.14.1.1 TrvActionInsertSymbol.CharCode

Last inserted character (UTF-32)

**property** CharCode: Cardinal;

When the dialog will be shown next time, it will show this character. When the user inserts a character, it is assigned to this property.

**Default value:**

\$F0B7 (bullet of "Symbol" font)

### 1.3.6.14.1.2 TrvActionInsertSymbol.FontName

Last used font name.

**property** FontName: TRVALocString<sup>(348)</sup>;

When the dialog will be shown next time, it will show this font name. When the user inserts a character, its font is assigned to this property.

This property is used only if UseCurrentFont<sup>(256)</sup> = *False*.

**Default value:**

'Symbol'

### 1.3.6.14.1.3 TrvActionInsertSymbol.Protection

Defines protection for inserted characters;

**property** Protection: TRVProtectOptions;

Protection options specified in this property is added to the protection options of text styles of inserted characters.

**Default value:**

[rvprDoNotAutoSwitch]

### 1.3.6.14.1.4 TrvActionInsertSymbol.UseCurrentFont

Defines how the initial value for a character's font is chosen.

**property** UseCurrentFont: Boolean;

If *False*: "Symbol" font is used when the dialog is displayed for the first time. The font of the last chosen character is used for subsequent calls.

If *True*: the font of the current text style is used.

If you assign *True* to this property, it makes sense to assign a different initial value to CharCode<sup>(255)</sup>, because the default value is for "Symbol" font.

**Default value:**

*False*

## 1.3.6.15 TrvActionInsertText

TrvActionInsertText is the action for text insertion.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionInsertText = **class** (TrvAction<sup>(311)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*

*TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>

## Description

This action calls `OnInsertText` <sup>(257)</sup> event and inserts the returned text in the caret position. This action does not have predefined Caption and Hint. Assign these properties yourself.

For example, this action may be used to insert the current date or time.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of `TrvAction`.

## See also:

- `TrvActionEvent` <sup>(321)</sup>

### 1.3.6.15.1 Events

#### In `TrvActionInsertText`

■ `OnInsertText` <sup>(257)</sup>

##### 1.3.6.15.1.1 `TrvActionInsertText.OnInsertText`

Requests text for insertion.

#### type

```
TRVInsertTextEvent = procedure (Sender: TObject;
  Editor: TCustomRichViewEdit;
  var Text: TRVUnicodeString) of object;
```

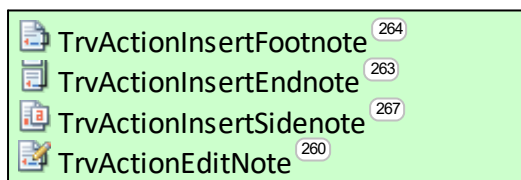
**property** `OnInsertText`: `TRVInsertTextEvent`;


The string assigned to **Text** will be inserted in **Editor**.

## 1.3.7 Notes and text boxes

### Actions Related to Notes and Text Boxes

This group of actions includes commands to insert notes and text boxes, and to edit them (in a separate editor).



 **ScaleRichView note:** it's not recommended to use these actions for `TSRichViewEdit`. `TSRichViewEdit` does not need a separate editor to edit notes and text boxes. `ScaleRichView` includes its own set of actions for notes and text boxes.

### 1.3.7.1 TrvActionCustomInsertSidenote

TrvActionCustomInsertSidenote is a base class for the actions inserting sidenotes and text box items.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionCustomInsertSidenote = class (TrvActionInsertNote (265))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionInsertNote* <sup>(265)</sup>

#### Description

This action introduces properties that are assigned to properties of an inserted sidenote (or a text box item):

- BoxProperties <sup>(260)</sup>
- BoxPosition <sup>(259)</sup>

Text in the note's Document is formatted either according to StyleTemplateName <sup>(260)</sup> (if style templates are used) or according to Font <sup>(266)</sup> (otherwise).

This class is not used directly. The following actions are inherited from TrvActionCustomInsertSidenote:

- TrvActionInsertSidenote <sup>(267)</sup>
- TrvActionInsertTextBox <sup>(269)</sup>

#### 1.3.7.1.1 Properties

##### In TrvActionCustomInsertSidenote

- BoxPosition <sup>(259)</sup>
- BoxProperties <sup>(260)</sup>
- StyleTemplateName <sup>(260)</sup>

##### Derived from TRVActionInsertNote <sup>(265)</sup>

- ActionEditNote <sup>(266)</sup>
- Font <sup>(266)</sup>

##### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>333</sup>

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.7.1.1.1 TrvActionCustomInsertSidenote.BoxPosition

Specifies position properties for the floating box.

#### type

```
TRVABoxPosition = class (TRVBoxPosition);
```

**property** BoxPosition: TRVABoxPosition;

This property is assigned to BoxPosition property of the inserted sidenote / text box item.

Default values are different in inherited actions.

#### TrvActionInsertSidenote <sup>267</sup>:

- HorizontalAnchor=*rvhanMainTextArea*;
- HorizontalPositionKind=*rvfpkAlignment*;
- HorizontalAlignment=*rvfphalRight*;
- VerticalAnchor=*rvvanParagraph*;
- VerticalPositionKind=*rvfpkAbsPosition*;
- VerticalOffset=0.

#### TrvActionInsertTextBox <sup>269</sup>:

- HorizontalAnchor=*rvhanMainTextArea*;
- HorizontalPositionKind=*rvfpkAlignment*;
- HorizontalAlignment=*rvfphalCenter*;
- VerticalAnchor=*rvvanLine*;
- VerticalPositionKind=*rvfpkAbsPosition*;
- VerticalOffset=20.

### 1.3.7.1.1.2 TrvActionCustomInsertSidenote.BoxProperties

Specifies size, background, border, vertical alignment properties for the floating box.

#### type

```
TRVABoxProperties = class (TRVBoxProperties);
```

**property** BoxProperties: TRVABoxProperties;

This property is assigned to BoxProperties property of the inserted sidenote / text box item.

#### Default values

- WidthType=rvbwtRelPage;
- Width=20000;

### 1.3.7.1.1.3 TrvActionCustomInsertSidenote.StyleTemplateName

Specifies the name of a style template that will be applied to a paragraph of a sidenote's Document.

**property** StyleTemplateName: TRVStyleTemplateName;

This property is used only if the target editor's UseStyleTemplates=*True*. Otherwise, the 0-th paragraph style and Font<sup>(266)</sup> are used.

Default values are different in inherited actions:

**TrvActionInsertSidenote**<sup>(267)</sup>:

'Sidenote Text' (note: this style template is standard for RichViewActions, but it is not standard for Microsoft Word)

**TrvActionInsertTextBox**<sup>(269)</sup>:

'Normal'

## 1.3.7.2 TrvActionEditNote

TrvActionEditNote allows editing a footnote, an endnote, a sidenote, or a text box in a non-modal dialog window.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertNote = class (TrvCustomEditorAction(336))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvCustomEditorAction(336)
```

#### Description





**ScaleRichView note:** it's not recommended to use this action for TSRichViewEdit. Use TrsvActionEditNote instead.

This action can be executed directly, or it is called by TrvActionInsertFootnote<sup>(264)</sup>, TrvActionInsertEndnote<sup>(263)</sup>, TrvActionInsertSidenote<sup>(267)</sup>, TrvActionInsertTextBox<sup>(269)</sup>.

It's recommended to assign Control<sup>(312)</sup> property for this action; otherwise, this property is assigned automatically when the action is executed for the first time.

When executed, the action shows a non-modal dialog window containing TRichViewEdit for editing subdocuments of items of the following types:

- footnote;
- endnote;
- sidenote;
- text box item.

If there is no item of these types in the caret position of the target editor, the action can move the caret to the next such item, if JumpToNextNote<sup>(262)</sup>=True. If there are no such items, a note editor is shown disabled.

If a note editor window is already visible when this action is executed, the action activates its window.

After the execution, the window is still open and the action reacts on the caret movement in the target editor (more exactly, in Control<sup>(312)</sup>): the window always shows a document for the current note. The user can edit the note document. It is saved to the note (as an undoable operation) automatically in the following cases:

- a caret movement in Control<sup>(312)</sup>;
- saving Control<sup>(312)</sup> to a file or a stream;
- printing or displaying a print preview of Control<sup>(312)</sup>;
- showing or hiding the note editor window;
- assigning a different value to Control<sup>(312)</sup>.

If you do not want to save recent changes, undo them in the note editor (there is no "Cancel Changes" command).

### Editing in the note editor

By default, a note editor is shown in a floating window, without any additional controls in this window. If you attempt to use toolbar buttons located on another form, you will see the problem: clicking the button activates another form, and the command will be executed for an editor in this form.

So you have two main options to implement UI for a note editor:

- use OnFormCreate<sup>(338)</sup> event to insert additional control in the note editor window (such as menu or toolbar)
- insert the note window in the main form, and switch RVAControlPanel.DefaultControl<sup>(42)</sup> when the note editor acquires and releases the input focus (see the example in the topic about OnShowing and OnHide<sup>(338)</sup>)

The latter is implemented in the ActionTest demos.

### 1.3.7.2.1 Properties

#### In TrvActionEditNote

---

- JumpToNextNote <sup>(262)</sup>

#### Derived from TRVCustomEditorAction <sup>(336)</sup>

---

- ▶ Form <sup>(337)</sup>
- ▶ SubDocEditor <sup>(337)</sup>

#### Derived from TrvAction <sup>(311)</sup>

---

- Control <sup>(312)</sup>

#### Derived from TrvCustomAction <sup>(333)</sup>

---

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.7.2.1.1 TrvActionEditNote.JumpToNextNote

Allows moving the caret to the next note in the target editor, when this action is executed.

**property** JumpToNextNote: Boolean;

The action always shows a note editor window on execution.

If **JumpToNextNote**=*True*, and the current item in the target editor is not a note when the action is executed, the action finds the next note in the target editor, moves the caret to it, and then shows a note editor window.

**Default value:**

*True*

### 1.3.7.2.2 Events

#### In TrvActionEditNote

- OnStartEditNote <sup>(263)</sup>

#### Inherited from TRVCustomEditorAction <sup>(336)</sup>

- OnFormCreate <sup>(338)</sup>
- OnHide <sup>(338)</sup>
- OnShowing <sup>(338)</sup>

#### 1.3.7.2.2.1 TrvActionEditNote.OnStartEditNote

Occurs when the action starts editing a new note.

**property** OnStartEditNote: TNotifyEvent;

This event occurs:

- when the action starts editing a note
- when the action ends editing a note

You can store a reference to the form containing the note editor in OnShowing <sup>(338)</sup> event, and use this form's caption in this event.

### 1.3.7.3 TrvActionInsertEndnote

TrvActionInsertEndnote inserts a new endnote in the document.

**Unit** RichViewActions <sup>(92)</sup>;


#### Syntax

TrvActionInsertEndnote = **class** (TrvActionInsertNote <sup>(265)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionInsertNote* <sup>(265)</sup>

#### Description

 **ScaleRichView note:** it's not recommended to use this action for TScaleRichViewEdit. Use TsrvActionInsertEndnote instead.

TrvActionInsertEndnote inserts a new endnote in the document, then executes ActionEditNote <sup>(266)</sup> to open an editor for this note.

If `TrvActionEditNote`<sup>(260)</sup>'s editor is active when this action is executed, this action inserts a reference to the parent endnote (`TRVNoteReferenceItemInfo` item).

When inserting an endnote, this action adds in its Document the following items:

- reference to this endnote;
- space character.

Attributes of inserted items depend on the target editor's `UseStyleTemplates` property.

If style templates are used, the endnote and the endnote reference characters are formatted using "endnote reference" style template. Otherwise, or if this style template does not exist, they are formatted as superscript text. In the both cases, `rvprDoNotAutoSwitch` is included in its Protection.

If style templates are used, endnote paragraph and text are formatted using "endnote text" style template. Otherwise, or if this style template does not exist, endnote text is formatted using `Font`<sup>(266)</sup> property.

**Note:** "endnote reference" and "endnote text" style templates are added by default to `TrvActionNew.StyleTemplates`, so they are included in new documents.

### 1.3.7.4 TrvActionInsertFootnote

`TrvActionInsertFootnote` inserts a new footnote in the document.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertFootnote = class(TrvActionInsertNote(265))
```


#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionInsertNote(265)

```

#### Description

 **ScaleRichView note:** it's not recommended to use this action for `TSRichViewEdit`. Use `TsrvActionInsertFootnote` instead.

`TrvActionInsertFootnote` inserts a new footnote in the document, then executes `ActionEditNote`<sup>(266)</sup> to open an editor for this note.

If `TrvActionEditNote`<sup>(260)</sup>'s editor is active when this action is executed, this action inserts a reference to the parent footnote (`TRVNoteReferenceItemInfo` item).

When inserting a footnote, this action adds in its Document the following items:

- reference to this footnote;

- space character.

Attributes of inserted items depend on the target editor's `UseStyleTemplates` property.

If style templates are used, the footnote and the footnote reference characters are formatted using "footnote reference" style template. Otherwise, or if this style template does not exist, they are formatted as superscript text. In the both cases, `rvprDoNotAutoSwitch` is included in its `Protection`.

If style templates are used, footnote paragraph and text are formatted using "footnote text" style template. Otherwise, or if this style template does not exist, footnote text is formatted using `Font`<sup>(266)</sup> property.

**Note:** "footnote reference" and "footnote text" style templates are added by default to `TrvActionNew.StyleTemplates`, so they are included in new documents.

### 1.3.7.5 TrvActionInsertNote

`TrvActionInsertNote` is a base class for the actions inserting footnotes, endnotes, sidenotes and text boxes.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertNote = class (TrvAction(311))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
```

#### Description

This class is not used directly.

The following actions are inherited from `TrvActionInsertNote`:

- `TrvActionInsertFootnote`<sup>(264)</sup>
- `TrvActionInsertEndnote`<sup>(263)</sup>
- `TrvActionInsertSidenote`<sup>(267)</sup>
- `TrvActionInsertTextBox`<sup>(269)</sup>

#### 1.3.7.5.1 Properties

##### In `TRVActionInsertNote`

- `ActionEditNote`<sup>(266)</sup>
- `Font`<sup>(266)</sup>

## Derived from TrvAction <sup>(311)</sup>

---

■ Control <sup>(312)</sup>

## Derived from TrvCustomAction <sup>(333)</sup>

---

■ Caption <sup>(334)</sup>

■ ControlPanel <sup>(335)</sup>

■ Disabled <sup>(335)</sup>

■ Hint <sup>(335)</sup>

## Derived from TAction

---

■ AutoCheck

■ Caption

■ Checked

DisableIfNoHandler

■ Enabled

■ GroupIndex

■ HelpContext

■ HelpKeyword

■ HelpType

■ Hint

■ ImageIndex

■ Name

■ SecondaryShortCuts

■ ShortCut

■ Visible

### 1.3.7.5.1.1 TrvActionInsertNote.ActionEditNote

Links this action to TrvActionEditNote <sup>(260)</sup> action.

```
property ActionEditNote: TrvActionEditNote (260);
```

The action executes ActionEditNote after inserting a note.

If this link is not defined, the first found TrvActionEditNote <sup>(260)</sup> action on the same form/datamodule is used. If not found, a window for editing the inserted note is not shown.

### 1.3.7.5.1.2 TrvActionInsertNote.Font

Default font for text in a note's document.

**type**

```
TRVAFont = TFont;
```

```
property Font: TRVAFont;
```

This value is used if UseStyleTemplates=*False* for the target editor, or the required style template does not exist in Style.StyleTemplates of the target editor. Otherwise, note text is formatted according to a style template.

Style template names:

- "footnote text" for footnotes,
- "endnote text" for endnotes,
- `StyleTemplateName` <sup>(260)</sup> for sidenotes and text boxes (default values: "Sidenote Text" for sidenotes, "Normal" for text boxes)

**Default value:**

'Arial', 10

### 1.3.7.6 TrvActionInsertSidenote

`TrvActionInsertSidenote` inserts a new sidenote (a note in a floating box) in the document.

**Unit** `RichViewActions` <sup>(92)</sup>;

**Syntax**

```
TrvActionCustomInsertSidenote = class (TrvActionCustomInsertSidenote (258))
```


**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction (333)
TrvAction (311)
TrvActionInsertNote (265)
TrvActionCustomInsertSidenote (258)

```

**Description**

 **ScaleRichView note:** it's not recommended to use this action for `TSRichViewEdit`. Use `TsrvActionInsertSidenote` instead.

`TrvActionInsertSidenote` inserts a new sidenote in the document, then executes `ActionEditNote` <sup>(266)</sup> to open an editor for this note.

If `TrvActionEditNote` <sup>(260)</sup>'s editor is active when this action is executed, this action inserts a reference to the parent sidenote (`TRVNoteReferenceItemInfo` item).

When inserting a sidenote, this action adds in its Document the following items:

- reference to this sidenote;
- space character.

Attributes of inserted items depend on the target editor's `UseStyleTemplates` property.

If style templates are used, the sidenote and the sidenote reference characters are formatted using `RefStyleTemplateName` <sup>(269)</sup> style template. Otherwise, or if this style template does not exist, they are formatted as superscript text. In the both cases, `rvprDoNotAutoSwitch` is included in its Protection.

If style templates are used, sidenote paragraph and text are formatted using `StyleTemplateName` <sup>(260)</sup> style template. Otherwise, or if this style template does not exist, sidenote text is formatted using `Font` <sup>(266)</sup> property.

The following properties are assigned to the inserted sidenote:

- `BoxProperties` <sup>(260)</sup> (default: width: 20% of page width, height: auto)
- `BoxPosition` <sup>(259)</sup> (default: horizontal - right aligned to the main text area, vertical - aligned to the top of the paragraph)

### 1.3.7.6.1 Properties

#### In `TrvActionInsertSidenote`

---

- `RefStyleTemplateName` <sup>(269)</sup>

#### Derived from `TrvActionCustomInsertSidenote` <sup>(258)</sup>

---

- `BoxPosition` <sup>(259)</sup>
- `BoxProperties` <sup>(260)</sup>
- `StyleTemplateName` <sup>(260)</sup>

#### Derived from `TRVActionInsertNote` <sup>(265)</sup>

---

- `ActionEditNote` <sup>(266)</sup>
- `Font` <sup>(266)</sup>

#### Derived from `TrvAction` <sup>(311)</sup>

---

- `Control` <sup>(312)</sup>

#### Derived from `TrvCustomAction` <sup>(333)</sup>

---

- `Caption` <sup>(334)</sup>
- `ControlPanel` <sup>(335)</sup>
- `Disabled` <sup>(335)</sup>
- `Hint` <sup>(335)</sup>

#### Derived from `TAction`

---

- `AutoCheck`
- `Caption`
- `Checked`
- `DisableIfNoHandler`
- `Enabled`
- `GroupIndex`
- `HelpContext`
- `HelpKeyword`
- `HelpType`
- `Hint`
- `ImageIndex`
- `Name`
- `SecondaryShortCuts`
- `ShortCut`



■ Visible

#### 1.3.7.6.1.1 TrvActionInsertSidenote.RefStyleTemplateName

Specifies the name of a style template that will be applied to the sidenote character and to the reference of to the parent sidenote (inserted in the sidenote.Document)

**property** RefStyleTemplateName: TRVStyleTemplateName;

This property is used only if the target editor's UseStyleTemplates=*True*. Otherwise, these characters are formatted as a superscript.

##### Default value

'Sidenote Reference' (note: this style template is standard for RichViewActions, but it is not standard for Microsoft Word)

### 1.3.7.7 TrvActionInsertTextBox

TrvActionInsertTextBox inserts a new text box item in the document.

**Unit** RichViewActions<sup>(92)</sup>;


##### Syntax

TrvActionInsertTextBox = **class** (TrvActionCustomInsertSidenote<sup>(258)</sup>)

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionInsertNote*<sup>(265)</sup>  
*TrvActionCustomInsertSidenote*<sup>(258)</sup>

##### Description

 **ScaleRichView note:** it's not recommended to use this action for TScaleRichViewEdit. Use TsvActionInsertTextBox instead.

TrvActionInsertTextBox inserts a new text box item in the document, then executes ActionEditNote<sup>(266)</sup> to open an editor for its document.

When inserting a text box, this action adds a single empty text item in its Document .

Attributes of this text item depend on the target editor's UseStyleTemplates property.

If style templates are used, this text item's paragraph and text are formatted using StyleTemplateName<sup>(260)</sup> style template. Otherwise, or if this style template does not exist, this text is formatted using Font<sup>(266)</sup> property.

The following properties are assigned to the inserted text box:




























- BoxProperties<sup>(260)</sup> (default: width: 20% of page width, height: auto)

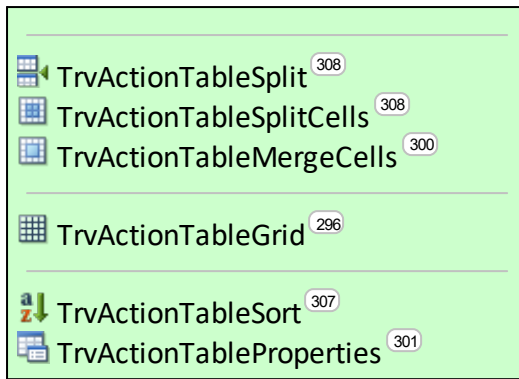
- `BoxPosition` <sup>259</sup> (default: horizontal - centered in the main text area, vertical - 20 pixels below the top of the line)

## 1.3.8 Tables

### Table Actions

This group of actions includes a command to insert table, and various operations on the selected table.

	<code>TrvActionInsertTable</code> <sup>271</sup>
	<code>TrvActionTableInsertColLeft</code> <sup>297</sup>
	<code>TrvActionTableInsertColRight</code> <sup>297</sup>
	<code>TrvActionTableInsertRowsAbove</code> <sup>298</sup>
	<code>TrvActionTableInsertRowsBelow</code> <sup>299</sup>
	<code>TrvActionTableDeleteCols</code> <sup>295</sup>
	<code>TrvActionTableDeleteRows</code> <sup>295</sup>
	<code>TrvActionTableDeleteTable</code> <sup>296</sup>
	<code>TrvActionTableToText</code> <sup>309</sup>
	<code>TrvActionTableSelectTable</code> <sup>306</sup>
	<code>TrvActionTableSelectCols</code> <sup>305</sup>
	<code>TrvActionTableSelectRows</code> <sup>306</sup>
	<code>TrvActionTableSelectCell</code> <sup>305</sup>
	<code>TrvActionTableCellVAlignTop</code> <sup>294</sup>
	<code>TrvActionTableCellVAlignMiddle</code> <sup>293</sup>
	<code>TrvActionTableCellVAlignBottom</code> <sup>292</sup>
	<code>TrvActionTableCellVAlignDefault</code> <sup>293</sup>
	<code>TrvActionTableCellRotationNone</code> <sup>288</sup>
	<code>TrvActionTableCellRotation90</code> <sup>289</sup>
	<code>TrvActionTableCellRotation180</code> <sup>289</sup>
	<code>TrvActionTableCellRotation270</code> <sup>290</sup>
	<code>TrvActionTableCellLeftBorder</code> <sup>285</sup>
	<code>TrvActionTableCellTopBorder</code> <sup>291</sup>
	<code>TrvActionTableCellRightBorder</code> <sup>287</sup>
	<code>TrvActionTableCellBottomBorder</code> <sup>284</sup>
	<code>TrvActionTableCellAllBorders</code> <sup>283</sup>
	<code>TrvActionTableCellNoBorders</code> <sup>286</sup>



### 1.3.8.1 TrvActionInsertTable

TrvActionInsertTable is the action for "Table | Insert Table" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionInsertTable = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

The action can insert a table in two ways:

- using a table dialog (when executed)
- using a special table size window (when ShowTableSizeDialog<sup>(282)</sup> is called).

In the table dialog, the user can specify a count of rows and columns and table width modes:

- autosize (table.BestWidth=0; widths of table cells are specified in pixels (or twips) so that the table has width not exceeding the current document width)
- fit window (table.BestWidth=-100 (i.e. 100%); widths of table cells are not specified)
- custom size (table.BestWidth is defined in the dialog, in pixels (or twips) or percent; widths of table cells are not specified)

When inserting using a table size window, table.BestWidth is specified in BestWidth<sup>(274)</sup>. If it is zero, cells widths are defined like in "autosize" mode.

If the user checked "remember dimensions for new tables" in the dialog, RowCount<sup>(279)</sup>, ColCount<sup>(278)</sup> and BestWidth<sup>(274)</sup> properties are set to the values entered in the dialog.

Before the insertion:

- the properties of this action are assigned to the table properties of the same names;

- RowsKeepTogether<sup>(279)</sup> and RowsVAlign<sup>(280)</sup> are assigned to properties of all rows of the table,
- OnInserting<sup>(282)</sup> event occurs.

The user can assign properties of this action by checking "Default" check box in the table properties dialog displayed by TrvActionTableProperties<sup>(301)</sup> or TrvActionItemProperties<sup>(324)</sup>.

### 1.3.8.1.1 Properties

#### In TrvActionInsertTable

- BackgroundPicture<sup>(273)</sup>
- BackgroundStyle<sup>(273)</sup>
- BestWidth<sup>(274)</sup>
- BorderColor<sup>(274)</sup>
- BorderHSpacing<sup>(274)</sup>
- BorderLightColor<sup>(274)</sup>
- BorderStyle<sup>(275)</sup>
- BorderVSpacing<sup>(275)</sup>
- BorderWidth<sup>(275)</sup>
- CellBorderColor<sup>(275)</sup>
- CellBorderLightColor<sup>(276)</sup>
- CellBorderStyle<sup>(276)</sup>
- CellBorderWidth<sup>(276)</sup>
- CellHPadding<sup>(276)</sup>
- CellHSpacing<sup>(277)</sup>
- CellPadding<sup>(276)</sup>
- CellVPadding<sup>(276)</sup>
- CellVSpacing<sup>(277)</sup>
- ColBandSize<sup>(277)</sup>
- ColCount<sup>(278)</sup>
- Color<sup>(278)</sup>
- EvenColumnsColor<sup>(281)</sup>
- EvenRowsColor<sup>(281)</sup>
- FirstColumnColor<sup>(281)</sup>
- HeadingRowColor<sup>(281)</sup>
- HeadingRowCount<sup>(278)</sup>
- HOutermostRule<sup>(278)</sup>
- HRuleColor<sup>(278)</sup>
- HRuleWidth<sup>(279)</sup>
- ItemText<sup>(279)</sup>
- LastColumnColor<sup>(281)</sup>
- LastRowColor<sup>(281)</sup>
- OddColumnsColor<sup>(281)</sup>
- OddRowsColor<sup>(281)</sup>
- RowBandSize<sup>(277)</sup>
- RowCount<sup>(279)</sup>
- RowsKeepTogether<sup>(279)</sup>
- RowsVAlign<sup>(280)</sup>
- TableOptions<sup>(280)</sup>

- TablePrintOptions <sup>(280)</sup>
- VisibleBorders <sup>(280)</sup>
- VOutermostRule <sup>(280)</sup>
- VRuleColor <sup>(281)</sup>
- VRuleWidth <sup>(281)</sup>

### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.8.1.1.1 TrvActionInsertTable.BackgroundImage

Defines the table background image.

**property** BackgroundPicture: TPicture;

BackgroundPicture.Graphic is assigned to the table's BackgroundImage

#### 1.3.8.1.1.2 TrvActionInsertTable.BackgroundStyle

Defines the table background style.

**property** BackgroundStyle: TRVItemBackgroundStyle;

This property is assigned to the table property of the same name.

**Default value:**

*rvbsColor*

#### 1.3.8.1.1.3 TrvActionInsertTable.BestWidth

Defines the table width.

**property** BestWidth: TRVHTMLLength;

This value is used:

- as an initial value for initializing a table dialog (on the action execution),
- directly (when ShowTableSizeDialog<sup>(282)</sup> is called).

If the user chose to remember dimensions for new tables, value from the dialog is assigned to this property.

This property (or the value from the table dialog) is assigned to table.BestWidth property. 0 means a default width, negative values mean percent, positive values mean GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

If it is zero (i.e. a default table width), the action sets BestWidth properties of all table cells to positive values so that the table width does not exceed the current document width. Otherwise, BestWidth properties of all cells are 0.

**Default value:**

0

#### 1.3.8.1.1.4 TrvActionInsertTable.BorderColor

Defines the table border color (or color of dark sides of 3D table border)

**property** BorderColor: TColor;

This property is assigned to the table property of the same name.

**Default value:**

*clWindowText*

#### 1.3.8.1.1.5 TrvActionInsertTable.BorderHSpacing

Defines the spacing between the left table border and the cells, and between the right table border and the cells.

**property** BorderHSpacing: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

**Default value:**

2

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

#### 1.3.8.1.1.6 TrvActionInsertTable.BorderLightColor

Defines the color of light sides of 3D table border

**property** BorderLightColor: TColor;

This property is assigned to the table property of the same name.

**Default value:***clBtnHighlight***1.3.8.1.1.7 TrvActionInsertTable.BorderStyle**

Defines the style of table border.

**property** `BorderStyle: TRVTableBorderStyle;`

This property is assigned to the table property of the same name.

**Default value:***rvtbColor***1.3.8.1.1.8 TrvActionInsertTable.BorderVSpacing**

Defines the spacing between the top table border and the cells, and between the bottom table border and the cells.

**property** `BorderVSpacing: TRVStyleLength;`

This value is measured in `GetControlPanel(336).UnitsProgram(55)`.

This property is assigned to the table property of the same name (converted to Units of the target editor).

**Default value:**

2

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

**1.3.8.1.1.9 TrvActionInsertTable.BorderWidth**

Defines the width of table border.

**property** `BorderWidth: TRVStyleLength;`

This value is measured in `GetControlPanel(336).UnitsProgram(55)`.

This property is assigned to the table property of the same name (converted to Units of the target editor).

0 hides the border.

**Default value:**

1

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

**1.3.8.1.1.10 TrvActionInsertTable.CellBorderColor**

Defines the default color of cell borders (or of dark sides of 3D cell borders).

**property** `CellBorderColor: TColor;`

This property is assigned to the table property of the same name.

**Default value:**

*clWindowText*

#### 1.3.8.1.1.11 TrvActionInsertTable.CellBorderLightColor

Defines the default color of light sides of 3D cell borders.

**property** CellBorderLightColor: TColor;

This property is assigned to the table property of the same name.

**Default value:**

*clBtnHighlight*

#### 1.3.8.1.1.12 TrvActionInsertTable.CellBorderStyle

Defines the style of cells borders.

**property** CellBorderStyle: TRVTableBorderStyle;

This property is assigned to the table property of the same name.

**Default value:**

*rvtbColor*

#### 1.3.8.1.1.13 TrvActionInsertTable.CellBorderWidth

Defines the width of cells borders.

**property** CellBorderWidth: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

0 hides the border.

**Default value:**

1

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

#### 1.3.8.1.1.14 TrvActionInsertTable.CellHPadding,CellVPadding

The properties define spacing between border and content in cells.

**property** CellHPadding: TRVStyleLength;

**property** CellVPadding: TRVStyleLength;

**property** CellPadding: TRVStyleLength;

These values are measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

CellHPadding defines a horizontal spacing, CellVPadding defines a vertical spacing.

These properties are assigned to the table properties of the same names (converted to Units of the target editor).



CellPadding is used to simplify access to these properties. Assigning value to CellPadding assigns it to CellHPadding and CellVPadding. CellPadding returns (CellHPadding+CellVPadding)/2.

**Default value:**

1

This default value assumes that these properties are measured in pixels. For twips, this value may be too small.

#### 1.3.8.1.1.15 TrvActionInsertTable.CellHSpacing

Defines the horizontal spacing between cells (i.e. spacing between table columns).

**property** CellHSpacing: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

**Default value:**

2

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

#### 1.3.8.1.1.16 TrvActionInsertTable.CellVSpacing

Defines the vertical spacing between cells (i.e. spacing between table rows).

**property** CellVSpacing: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

**Default value:**

2

This default value assumes that this property is measured in pixels. For twips, this value may be too small.

#### 1.3.8.1.1.17 TrvActionInsertTable.ColBandSize, RowBandSize

The properties defining count of columns/rows in alternating column/row bands.

**property** ColBandSize: Integer;

**property** RowBandSize: Integer;

RowBandSize specify the count of rows in row bands.

ColBandSize specify the count of columns in column bands.

These properties are assigned to the table properties of the same names.

**Default value:**

1

**See also**

- HeadingRowColor and other colors <sup>(281)</sup>

#### 1.3.8.1.18 TrvActionInsertTable.ColCount

Defines the count of table columns.

**property** ColCount: Integer;

This property is used as an initial value for initializing a table dialog. If the user chose to remember dimensions for new tables, value from the dialog is assigned to this property.

This property is ignored in ShowTableSizeDialog <sup>(282)</sup> method.

**Default value:**

2

#### 1.3.8.1.19 TrvActionInsertTable.Color

Defines the background table color.

**property** Color: TColor;

This property is assigned to the table property of the same name.

**Default value:**

*clNone*

#### 1.3.8.1.20 TrvActionInsertTable.HeadingRowCount

Defines the count of heading rows (rows repeated on each page containing this table).

**property** HeadingRowCount: Integer;

This property is assigned to the table property of the same name.

**Default value:**

0

#### 1.3.8.1.21 TrvActionInsertTable.HOutermostRule

Allows showing horizontal lines between the top and the bottom table borders and cells.

**property** HOutermostRule: Boolean;

This property is assigned to the table property of the same name.

Horizontal rules are drawn only if HRuleWidth <sup>(279)</sup> > 0.

**Default value:**

*False*

#### 1.3.8.1.22 TrvActionInsertTable.HRuleColor

Defines the color of horizontal rules.

**property** HRuleColor: TColor;

This property is assigned to the table property of the same name.

Horizontal rules are drawn only if HRuleWidth <sup>(279)</sup> > 0.

**Default value:***c/WindowText***1.3.8.1.1.23 TrvActionInsertTable.HRuleWidth**

Defines the width of horizontal rules.

**property** HRuleWidth: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

0 hides rules.

**Default value:**

0

**1.3.8.1.1.24 TrvActionInsertTable.ItemText**

Defines the item text (item name) for the table.

**property** ItemText: TRVUnicodeString;

This property is used as a parameter of InsertItem method.

**Default value:**

" (empty string)

**1.3.8.1.1.25 TrvActionInsertTable.RowCount**

Defines the count of table rows.

**property** RowCount: Integer;

This property is used as an initial value for initializing a table dialog. If the user chose to remember dimensions for new tables, value from the dialog is assigned to this property.

This property is ignored in ShowTableSizeDialog<sup>(282)</sup> method.

**Default value:**

2


**1.3.8.1.1.26 TrvActionInsertTable.RowsKeepTogether**

Defines the "keep together" property for all rows of inserted tables.

**property** RowsKeepTogether: Boolean;

This property is assigned to KeepTogether properties of all rows of inserted tables.

If *True*, the component avoids breaking pages inside rows (pagination adds page breaks between rows, when possible)

 **ScaleRichView note:** TScaleRichViewEdit does not support page breaks inside table cells yet.

**Default value:**

*False*

#### 1.3.8.1.1.27 TrvActionInsertTable.RowsVAlign

Defines the default vertical alignment of content of cells of all rows of inserted tables.

**property** RowsVAlign: TRVCellVAlign;

This property is assigned to VAlign properties of all rows of inserted tables.

**Default value:**

*rvcTop*

#### 1.3.8.1.1.28 TrvActionInsertTable.TableOptions

Defines the table options.

**property** TableOptions: TRVTableOptions;

This property is assigned to the table property of the same name.

**Default value:**

*[rvtoEditing, rvtoRowSizing, rvtoColSizing, rvtoRowSelect, rvtoColSelect]*

#### 1.3.8.1.1.29 TrvActionInsertTable.TablePrintOptions

Defines the table printing options.

**property** TablePrintOptions: TRVTablePrintOptions;

This property is assigned to the table's PrintOptions property.

**Default value:**

*[rvtoHalftoneBorders,rvtoRowsSplit]*

#### 1.3.8.1.1.30 TrvActionInsertTable.VisibleBorders

Defines visible sides of a table border.

**property** VisibleBorders: TRVBooleanRect;

This property is assigned to the table property of the same name.

**Default value:**

*(True, True, True, True)*

#### 1.3.8.1.1.31 TrvActionInsertTable.VOutermostRule

Allows showing vertical lines between the left and the right table borders and cells.

**property** VOutermostRule: Boolean;

This property is assigned to the table property of the same name.

Vertical rules are drawn only if VRuleWidth<sup>(281)</sup>>0.

**Default value:**

*False*

#### 1.3.8.1.1.32 TrvActionInsertTable.VRuleColor

Defines the color of vertical rules.

**property** VRuleColor: TColor;

This property is assigned to the table property of the same name.

Vertical rules are drawn only if VRuleWidth<sup>(281)</sup>>0.

**Default value:**

*c/WindowText*

#### 1.3.8.1.1.33 TrvActionInsertTable.VRuleWidth

Defines the width of vertical rules.

**property** VRuleWidth: TRVStyleLength;

This value is measured in GetControlPanel<sup>(336)</sup>.UnitsProgram<sup>(55)</sup>.

This property is assigned to the table property of the same name (converted to Units of the target editor).

0 hides rules.

**Default value:**

0

#### 1.3.8.1.1.34 TrvActionInsertTable's row and column colors

A set of properties defining default colors of cells in specific rows and columns.

**property** HeadingRowColor: TColor;

**property** LastRowColor: TColor;

**property** OddRowsColor: TColor;

**property** EvenRowsColor: TColor;

**property** FirstColumnColor: TColor;

**property** LastColumnColor: TColor;

**property** OddColumnsColor: TColor;

**property** EvenColumnsColor: TColor;

These properties are assigned to the table properties of the same names.

**Default values**

*c/None*

**See also**

- ColBandSize, RowBandSize

#### 1.3.8.1.2 Methods

##### In TrvActionInsertTable

ShowTableSizeDialog<sup>(282)</sup>

##### Inherited from TrvCustomAction<sup>(333)</sup>

GetControlPanel<sup>(336)</sup>

### 1.3.8.1.2.1 TrvActionInsertTable.ShowTableSizeDialog

These methods display a table sizing window. When this window is closed, a table with the specified count of rows and columns is inserted in **Target**.

```
procedure ShowTableSizeDialog(Target: TCustomRichViewEdit;  
    Button: TControl); overload;  
procedure ShowTableSizeDialog(Target: TCustomRichViewEdit;  
    const ButtonRect: TRect); overload;
```

The first version of this method displays this window relative to the position of **Button**.

In the second version of this method, this position is specified in **ButtonRect** (some third-party toolbar component do not use controls for buttons, so the first version of this method cannot be used).

If the user presses **Escape** while this window is open, the window is closed and insertion is canceled.

### 1.3.8.1.3 Events

#### In TrvActionInsertTable

- OnCloseTableSizeDialog<sup>(282)</sup>
- OnInserting<sup>(282)</sup>

### 1.3.8.1.3.1 TrvActionInsertTable.OnCloseTableSizeDialog

Occurs when a table sizing window (shown by ShowTableSizeDialog<sup>(282)</sup> method) is closed.

```
property OnCloseTableSizeDialog: TNotifyEvent;
```

This event can be used to uncheck a button pressed before calling ShowTableSizeDialog<sup>(282)</sup> method (if you have some button that must be pressed while this window is displayed).

### 1.3.8.1.3.2 TrvActionInsertTable.OnInserting

Occurs before **table** is inserted.

**type**

```
TRVInsertTableEvent = procedure (Sender: TrvActionInsertTable;  
    table: TRVTableItemInfo) of object;
```

```
property OnInserting: TRVInsertTableEvent;
```

This event occurs after all the action's properties are applied to **table**. You can make additional changes in this event.

## 1.3.8.2 TrvActionTableCell

TrvActionTableCell is a base class for actions performing operations on table and table cells.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionTableCell = class (TrvAction(311))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action is not used directly. This action has the following direct descendants:

- *TrvActionTableRCBase* <sup>(304)</sup>
- *TrvActionTableDeleteTable* <sup>(296)</sup>
- *TrvActionTableMergeCells* <sup>(300)</sup>
- *TrvActionTableSplitCells* <sup>(308)</sup>
- *TrvActionTableSelectTable* <sup>(306)</sup>
- *TrvActionTableSelectCell* <sup>(305)</sup>
- *TrvActionTableMultiCellAttributes* <sup>(301)</sup>
- *TrvActionTableProperties* <sup>(301)</sup>

### 1.3.8.3 TrvActionTableCellAllBorders

*TrvActionTableCellAllBorders* is the action for "Table | Cell Borders | All Borders" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableCellAllBorders = class (TrvActionTableCellBorder (284))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableMultiCellAttributes* <sup>(301)</sup>  
*TrvActionTableCellBorder* <sup>(284)</sup>

## Description

The action makes all sides of borders of the selected table cells (or the edited cell) visible/invisible.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

**See also:**

- TrvActionTableCellLeftBorder<sup>(285)</sup>
- TrvActionTableCellRightBorder<sup>(287)</sup>
- TrvActionTableCellTopBorder<sup>(291)</sup>
- TrvActionTableCellBottomBorder<sup>(284)</sup>
- TrvActionTableCellNoBorders<sup>(286)</sup>

### 1.3.8.4 TrvActionTableCellBorder

TrvActionTableCellBorder is a base class for the actions changing cell borders.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionTableCellBorder = class(TrvActionTableMultiCellAttributes(301))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableMultiCellAttributes*<sup>(301)</sup>

**Description**

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

This action is not used directly. The following actions are inherited from it:

- TrvActionTableCellLeftBorder<sup>(285)</sup>
- TrvActionTableCellRightBorder<sup>(287)</sup>
- TrvActionTableCellTopBorder<sup>(291)</sup>
- TrvActionTableCellBottomBorder<sup>(284)</sup>
- TrvActionTableCellAllBorders<sup>(283)</sup>
- TrvActionTableCellNoBorders<sup>(286)</sup>

These actions change table.Cells[].VisibleBorders properties for the selected cells (or the edited cell). All these actions (except for TrvActionTableCellNoBorders<sup>(286)</sup>) are checkbox-like buttons: they are checked if the corresponding sides of cells borders are visible.

### 1.3.8.5 TrvActionTableCellBottomBorder

TrvActionTableCellBottomBorder is the action for "Table | Cell Borders | Bottom Border" command.

**Unit** RichViewActions<sup>(92)</sup>;



**Syntax**

```
TrvActionTableCellBottomBorder = class (TrvActionTableCellBorder284)
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction333
TrvAction311
TrvActionTableCell282
TrvActionTableMultiCellAttributes301
TrvActionTableCellBorder284

```

**Description**

The action makes the bottom borders of the selected table cells (or the edited cell) visible/invisible.

This action does not introduce any new properties in addition to properties<sup>312</sup> of TrvAction.

**See also:**

- TrvActionTableCellLeftBorder<sup>285</sup>
- TrvActionTableCellRightBorder<sup>287</sup>
- TrvActionTableCellTopBorder<sup>291</sup>
- TrvActionTableCellAllBorders<sup>283</sup>
- TrvActionTableCellNoBorders<sup>286</sup>

**1.3.8.6 TrvActionTableCellLeftBorder**

TrvActionTableCellLeftBorder is the action for "Table | Cell Borders | Left Border" command.

**Unit** RichViewActions<sup>92</sup> ;

**Syntax**

```
TrvActionTableCellLeftBorder = class (TrvActionTableCellBorder284)
```

**Hierarchy**

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction333
TrvAction311
TrvActionTableCell282
TrvActionTableMultiCellAttributes301

```

*TrvActionTableCellBorder* <sup>(284)</sup>

### Description

The action makes the left borders of the selected table cells (or the edited cell) visible/invisible.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

### See also:

- TrvActionTableCellRightBorder <sup>(287)</sup>
- TrvActionTableCellTopBorder <sup>(291)</sup>
- TrvActionTableCellBottomBorder <sup>(284)</sup>
- TrvActionTableCellAllBorders <sup>(283)</sup>
- TrvActionTableCellNoBorders <sup>(286)</sup>

### 1.3.8.7 TrvActionTableCellNoBorders

TrvActionTableCellNoBorders is the action for "Table | Cell Borders | No Borders" command.

**Unit** RichViewActions <sup>(92)</sup>;

### Syntax

```
TrvActionTableCellNoBorders = class (TrvActionTableCellBorder (284))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableMultiCellAttributes* <sup>(301)</sup>  
*TrvActionTableCellBorder* <sup>(284)</sup>

### Description

The action makes all sides of border of the selected table cells (or the edited cell) invisible.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This is a redundant action, because it only provides a subset of functionality of TrvActionTableCellAllBorders <sup>(283)</sup>.

### See also:

- TrvActionTableCellLeftBorder <sup>(285)</sup>
- TrvActionTableCellRightBorder <sup>(287)</sup>
- TrvActionTableCellTopBorder <sup>(291)</sup>
- TrvActionTableCellBottomBorder <sup>(284)</sup>
- TrvActionTableCellAllBorders <sup>(283)</sup>

### 1.3.8.8 TrvActionTableCellRightBorder

TrvActionTableCellRightBorder is the action for "Table | Cell Borders | Right Border" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellRightBorder = class (TrvActionTableCellBorder (284))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableMultiCellAttributes* <sup>(301)</sup>  
*TrvActionTableCellBorder* <sup>(284)</sup>

#### Description

The action makes the right borders of the selected table cells (or the edited cell) visible/invisible.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableCellLeftBorder <sup>(285)</sup>
- TrvActionTableCellTopBorder <sup>(291)</sup>
- TrvActionTableCellBottomBorder <sup>(284)</sup>
- TrvActionTableCellAllBorders <sup>(283)</sup>
- TrvActionTableCellNoBorders <sup>(286)</sup>

### 1.3.8.9 TrvActionTableCellRotation

TrvActionTableCellRotation is a base class for the actions changing a rotation in table cells.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellRotation = class (TrvActionTableMultiCellAttributes (301))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*

*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableMultiCellAttributes*<sup>(301)</sup>

## Description

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of *TrvAction*.

This action is not used directly. The following actions are inherited from it:

- *TrvActionTableCellRotationNone*<sup>(288)</sup>
- *TrvActionTableCellRotation90*<sup>(289)</sup>
- *TrvActionTableCellRotation180*<sup>(289)</sup>
- *TrvActionTableCellRotation270*<sup>(290)</sup>

### 1.3.8.10 TrvActionTableCellRotationNone

*TrvActionTableCellRotationNone* is the action for "Table | No Cell Rotation" command.

**Unit** *RichViewActions*<sup>(92)</sup>;

## Syntax

```
TrvActionTableCellRotationNone = class (TrvActionTableCellRotation(287))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableMultiCellAttributes*<sup>(301)</sup>  
*TrvActionTableCellRotation*<sup>(287)</sup>

## Description

The action resets the rotation of the selected table cells (or the edited cell) to 0°. It changes `table.Cells[].Rotation` to *rvrotNone*. If the cell was rotated vertically (90° or 270°), it also exchanges `table.Cells[].BestWidth` and `BestHeight`.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of *TrvAction*.

## See also:

- *TrvActionTableCellRotation90*<sup>(289)</sup>
- *TrvActionTableCellRotation180*<sup>(289)</sup>
- *TrvActionTableCellRotation270*<sup>(290)</sup>

### 1.3.8.11 TrvActionTableCellRotation90

TrvActionTableCellRotation90 is the action for "Table | Rotate Cell by 90°" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellRotation90 = class (TrvActionTableCellRotation(287))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionTableCell(282)
TrvActionTableMultiCellAttributes(301)
TrvActionTableCellRotation(287)

```

#### Description

The action sets the rotation of the selected table cells (or the edited cell) to 90°. It changes table.Cells[].Rotation to *rvrot90*. If the cell was rotated horizontally (0° or 180°), it also exchanges table.Cells[].BestWidth and BestHeight.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableCellRotationNone<sup>(288)</sup>
- TrvActionTableCellRotation180<sup>(289)</sup>
- TrvActionTableCellRotation270<sup>(290)</sup>

### 1.3.8.12 TrvActionTableCellRotation180

TrvActionTableCellRotation180 is the action for "Table | Rotate Cell by 180°" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellRotation180 = class (TrvActionTableCellRotation(287))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction

```

[TrvCustomAction](#)<sup>(333)</sup>  
[TrvAction](#)<sup>(311)</sup>  
[TrvActionTableCell](#)<sup>(282)</sup>  
[TrvActionTableMultiCellAttributes](#)<sup>(301)</sup>  
[TrvActionTableCellRotation](#)<sup>(287)</sup>

## Description

The action sets the rotation of the selected table cells (or the edited cell) to 180°. It changes `table.Cells[].Rotation` to `rvrot180`. If the cell was rotated vertically (90° or 270°), it also exchanges `table.Cells[].BestWidth` and `BestHeight`.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of `TrvAction`.

## See also:

- [TrvActionTableCellRotationNone](#)<sup>(288)</sup>
- [TrvActionTableCellRotation90](#)<sup>(289)</sup>
- [TrvActionTableCellRotation270](#)<sup>(290)</sup>

### 1.3.8.13 TrvActionTableCellRotation270

`TrvActionTableCellRotation270` is the action for "Table | Rotate Cell by 270°" command.

**Unit** `RichViewActions`<sup>(92)</sup>;

## Syntax

```
TrvActionTableCellRotation270 = class (TrvActionTableCellRotation(287))
```

## Hierarchy

[TObject](#)  
[TPersistent](#)  
[TComponent](#)  
[TBasicAction](#)  
[TContainedAction](#)  
[TCustomAction](#)  
[TAction](#)  
[TrvCustomAction](#)<sup>(333)</sup>  
[TrvAction](#)<sup>(311)</sup>  
[TrvActionTableCell](#)<sup>(282)</sup>  
[TrvActionTableMultiCellAttributes](#)<sup>(301)</sup>  
[TrvActionTableCellRotation](#)<sup>(287)</sup>

## Description

The action sets the rotation of the selected table cells (or the edited cell) to 270°. It changes `table.Cells[].Rotation` to `rvrot270`. If the cell was rotated horizontally (0° or 180°), it also exchanges `table.Cells[].BestWidth` and `BestHeight`.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of `TrvAction`.

## See also:

- [TrvActionTableCellRotationNone](#)<sup>(288)</sup>
- [TrvActionTableCellRotation90](#)<sup>(289)</sup>

- TrvActionTableCellRotation180<sup>(289)</sup>

### 1.3.8.14 TrvActionTableCellTopBorder

TrvActionTableCellTopBorder is the action for "Table | Cell Borders | Top Border" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellTopBorder = class (TrvActionTableCellBorder(284))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionTableCell(282)
TrvActionTableMultiCellAttributes(301)
TrvActionTableCellBorder(284)

```

#### Description

The action makes the top borders of the selected table cells (or the edited cell) visible/invisible.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableCellLeftBorder<sup>(285)</sup>
- TrvActionTableCellRightBorder<sup>(287)</sup>
- TrvActionTableCellBottomBorder<sup>(284)</sup>
- TrvActionTableCellAllBorders<sup>(283)</sup>
- TrvActionTableCellNoBorders<sup>(286)</sup>

### 1.3.8.15 TrvActionTableCellVAlign

TrvActionTableCellVAlign is a base class for the actions changing a vertical alignment in table cells.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellVAlign = class (TrvActionTableMultiCellAttributes(301))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction

```

*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableMultiCellAttributes* <sup>(301)</sup>

## Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

This action is not used directly. The following actions are inherited from it:

- *TrvActionTableCellVAlignTop* <sup>(294)</sup>
- *TrvActionTableCellVAlignMiddle* <sup>(293)</sup>
- *TrvActionTableCellVAlignBottom* <sup>(292)</sup>
- *TrvActionTableCellVAlignDefault* <sup>(293)</sup>

### 1.3.8.16 TrvActionTableCellVAlignBottom

*TrvActionTableCellVAlignBottom* is the action for "Table | Align Cell to the Bottom" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableCellVAlignBottom = class (TrvActionTableCellVAlign (291))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableMultiCellAttributes* <sup>(301)</sup>  
*TrvActionTableCellVAlign* <sup>(291)</sup>

## Description

The action aligns contents of the selected table cells (or the edited cell) to bottom. It changes `table.Cells[].VAlign` to `rvcBottom`.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

## See also:

- *TrvActionTableCellVAlignTop* <sup>(294)</sup>
- *TrvActionTableCellVAlignMiddle* <sup>(293)</sup>
- *TrvActionTableCellVAlignDefault* <sup>(293)</sup>



### 1.3.8.17 TrvActionTableCellVAlignDefault

TrvActionTableCellVAlignDefault is the action for "Table | Default Cell Vertical Alignment" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellVAlignDefault = class(TrvActionTableCellVAlign(291))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)
TrvActionTableCell(282)
TrvActionTableMultiCellAttributes(301)
TrvActionTableCellVAlign(291)

```

#### Description

The action sets the default alignment for the selected table cells (or the edited cell). It changes table.Cells[].VAlign to rvcVDefault. In this mode, cells use table.Rows[].VAlign.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableCellVAlignTop<sup>(294)</sup>
- TrvActionTableCellVAlignMiddle<sup>(293)</sup>
- TrvActionTableCellVAlignBottom<sup>(292)</sup>

### 1.3.8.18 TrvActionTableCellVAlignMiddle

TrvActionTableCellVAlignMiddle is the action for "Table | Align Cell to the Middle" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableCellVAlignMiddle = class(TrvActionTableCellVAlign(291))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction

```

[TrvCustomAction](#)<sup>(333)</sup>  
[TrvAction](#)<sup>(311)</sup>  
[TrvActionTableCell](#)<sup>(282)</sup>  
[TrvActionTableMultiCellAttributes](#)<sup>(301)</sup>  
[TrvActionTableCellVAlign](#)<sup>(291)</sup>

## Description

The action aligns contents of the selected table cells (or the edited cell) to the middle. It changes `table.Cells[].VAlign` to `rvcMiddle`.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of `TrvAction`.

## See also:

- [TrvActionTableCellVAlignTop](#)<sup>(294)</sup>
- [TrvActionTableCellVAlignBottom](#)<sup>(292)</sup>
- [TrvActionTableCellVAlignDefault](#)<sup>(293)</sup>

### 1.3.8.19 TrvActionTableCellVAlignTop

`TrvActionTableCellVAlignTop` is the action for "Table | Align Cell to the Top" command.

**Unit** `RichViewActions`<sup>(92)</sup>;

## Syntax

```
TrvActionTableCellVAlignTop = class (TrvActionTableCellVAlign(291))
```

## Hierarchy

[TObject](#)  
[TPersistent](#)  
[TComponent](#)  
[TBasicAction](#)  
[TContainedAction](#)  
[TCustomAction](#)  
[TAction](#)  
[TrvCustomAction](#)<sup>(333)</sup>  
[TrvAction](#)<sup>(311)</sup>  
[TrvActionTableCell](#)<sup>(282)</sup>  
[TrvActionTableMultiCellAttributes](#)<sup>(301)</sup>  
[TrvActionTableCellVAlign](#)<sup>(291)</sup>

## Description

The action aligns contents of the selected table cells (or the edited cell) to top. It changes `table.Cells[].VAlign` to `rvcTop`.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of `TrvAction`.

## See also:

- [TrvActionTableCellVAlignMiddle](#)<sup>(293)</sup>
- [TrvActionTableCellVAlignBottom](#)<sup>(292)</sup>
- [TrvActionTableCellVAlignDefault](#)<sup>(293)</sup>

### 1.3.8.20 TrvActionTableDeleteCols

TrvActionTableDeleteCols is the action for "Table | Delete Columns" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableDeleteColumns = class (TrvActionTableRCBase(304))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableRCBase*<sup>(304)</sup>

#### Description

This action deletes all table columns containing selected cells (or the edited cell).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableDeleteRows<sup>(295)</sup>

### 1.3.8.21 TrvActionTableDeleteRows

TrvActionTableDeleteRows is the action for "Table | Delete Rows" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableDeleteRows = class (TrvActionTableRCBase(304))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableRCBase*<sup>(304)</sup>

#### Description

This action deletes all table rows containing selected cells (or the edited cell).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

**See also:**

- TrvActionTableDeleteCols<sup>(295)</sup>

### 1.3.8.22 TrvActionTableDeleteTable

TrvActionTableDeleteTable is the action for "Table | Delete Table" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionTableDeleteTable = class (TrvActionTableCell(282))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>

**Description**

The action deletes the table at the caret position.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

### 1.3.8.23 TrvActionTableGrid

TrvActionTableGrid is the action for "Table | Show Grid Lines" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionTableGrid = class (TrvAction(311))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

## Description

The action shows/hides grid lines in tables. Grid lines are shown in places of invisible borders (zero border width, or sides hidden by VisibleBorders properties).

The action includes/excludes rvoShowGridLines in Options of the target editor.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

### 1.3.8.24 TrvActionTableInsertColLeft

TrvActionTableInsertColLeft is the action for "Table | Insert Column Left" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableInsertColLeft = class (TrvActionTableRCBase (304))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableRCBase* <sup>(304)</sup>

## Description

This action inserts one new column to the left of the selected cells (or of the edited cell).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

## See also:

- TrvActionTableInsertRowsAbove <sup>(298)</sup>
- TrvActionTableInsertRowsBelow <sup>(299)</sup>
- TrvActionTableInsertColRight <sup>(297)</sup>

### 1.3.8.25 TrvActionTableInsertColRight

TrvActionTableInsertColRight is the action for "Table | Insert Column Right" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableInsertColRight = class (TrvActionTableRCBase (304))
```

## Hierarchy

*TObject*  
*TPersistent*

*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableRCBase* <sup>(304)</sup>

## Description

This action inserts one new column to the right of the selected cells (or of the edited cell).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

## See also:

- *TrvActionTableInsertRowsAbove* <sup>(298)</sup>
- *TrvActionTableInsertRowsBelow* <sup>(299)</sup>
- *TrvActionTableInsertColLeft* <sup>(297)</sup>

### 1.3.8.26 TrvActionTableInsertRowsAbove

*TrvActionTableInsertRowsAbove* is the action for "Table | Insert Row Above" command.

**Unit** *RichViewActions* <sup>(92)</sup>;

## Syntax

```
TrvActionTableInsertRowsAbove = class (TrvActionTableRCBase (304))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableRCBase* <sup>(304)</sup>

## Description

This action inserts one new row above the selected cells (or above the edited cell).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

## See also:

- *TrvActionTableInsertRowsBelow* <sup>(299)</sup>
- *TrvActionTableInsertColLeft* <sup>(297)</sup>
- *TrvActionTableInsertColRight* <sup>(297)</sup>

### 1.3.8.27 TrvActionTableInsertRowsBelow

TrvActionTableInsertRowsBelow is the action for "Table | Insert Row Below" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableInsertRowsBelow = class (TrvActionTableRCBase(304))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableRCBase*<sup>(304)</sup>

#### Description

This action inserts new rows below the selected cells (or below the edited cell).

When inserting rows to the end of the table, if AllowMultiple<sup>(300)</sup> = *True*, the action requests a number of rows to insert and inserts this count of rows.

#### See also:

- TrvActionTableInsertRowsAbove<sup>(298)</sup>
- TrvActionTableInsertColLeft<sup>(297)</sup>
- TrvActionTableInsertColRight<sup>(297)</sup>

#### 1.3.8.27.1 Properties

##### In TrvActionTableInsertRowsBelow

■ AllowMultiple<sup>(300)</sup>

##### Derived from TrvAction<sup>(311)</sup>

■ Control<sup>(312)</sup>

##### Derived from TrvCustomAction<sup>(333)</sup>

■ Caption<sup>(334)</sup>

■ ControlPanel<sup>(335)</sup>

■ Disabled<sup>(335)</sup>

■ Hint<sup>(335)</sup>

##### Derived from TAction

■ AutoCheck

■ Caption

- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.8.27.1.1 TrvActionTableInsertRowsBelow.AllowMultiple

Allowing inserting several rows at the end of table at once.

**property** AllowMultiple: Boolean;

If *True*, when inserting rows to the end of the table, the number of new rows is requested from the user.

**Default value:**

*True*

### 1.3.8.28 TrvActionTableMergeCells

TrvActionTableMergeCells is the action for "Table | Merge Cells" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

TrvActionTableMergeCells = **class** (TrvActionTableCell<sup>(282)</sup>)

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>

**Description**

The action merges the selected table cells in one.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.



### 1.3.8.29 TrvActionTableMultiCellAttributes

TrvActionTableMultiCellAttributes is the base class for actions changing properties of table cells.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionTableMultiCellAttributes = class (TrvActionTableCell (282))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

#### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action is not used directly. This action has the following direct descendants:

- TrvActionTableCellVAlign <sup>(291)</sup>
- TrvActionTableCellBorder <sup>(284)</sup>

### 1.3.8.30 TrvActionTableProperties

TrvActionTableProperties is the action for "Table | Table Properties" command.

**Unit** RichViewActions <sup>(92)</sup>;

#### Syntax

```
TrvActionTableProperties = class (TrvActionTableCell (282))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

#### Description

The action displays a dialog for editing table properties. If the table has no selection (i.e. the caret is placed to the right or to the left of the table), this dialog has one page ("Table"). If there is a multicell selection or an edited cell, the dialog has additional pages ("Rows", "Columns", "Cells").

The same dialog is shown by TrvActionItemProperties<sup>(324)</sup> action. But TrvActionItemProperties<sup>(324)</sup> edits properties of an item at the position of caret. For example, if a picture inside a table is selected, TrvActionItemProperties<sup>(324)</sup> edits this picture, while TrvActionTableProperties edits this table.

The user can check "Default" check box in this dialog to assign properties of new tables.

When the action assigns background images of tables and cells, the action assigns their BackgroundImageFileName properties, if *rvoAssignImageFileNames* is included in the Options property of the target editor.

### 1.3.8.30.1 Properties

#### In TrvActionTableProperties

- ActionInsertTable<sup>(303)</sup>
- BackgroundGraphicFilter<sup>(303)</sup>
- DefaultChecked<sup>(303)</sup>
- DefaultPersistent<sup>(303)</sup>
- UpdateAllInsertTableActions<sup>(303)</sup>

#### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut

## ■ Visible

### 1.3.8.30.1.1 TrvActionTableProperties.ActionInsertTable, UpdateAllInsertTableActions

ActionInsertTable specifies the "insert table" action for applying default properties to.

UpdateAllInsertTableActions allows applying to all actions "insert table" actions on the same form/datamodule.

```
property ActionInsertTable: TrvActionInsertTable(271);
property UpdateAllInsertTableActions: Boolean;
```

If "Default" checkbox is checked, when properties are applied, they are assigned to "insert table" action(s) as well, so they become default properties for new tables.

If **ActionInsertTable** is defined, properties are applied only to this action. If this link is not defined, they are applied to the first found TrvActionInsertTable<sup>(271)</sup> action on the same form/datamodule (if **UpdateAllInsertTableActions=False**), or to all TrvActionInsertTable<sup>(271)</sup> actions on the same form/datamodule (if **UpdateAllInsertTableActions=True**)

**Note:** Report Workshop includes an additional action for inserting a report table:

TrvrActionInsertTable, inherited from TrvActionInsertTable<sup>(271)</sup>. If you want to apply changes both to instances of TrvrActionInsertTable and TrvActionInsertTable<sup>(271)</sup>, **UpdateAllInsertTableActions** must be True.

#### Default value

**UpdateAllInsertTableActions** = *True*

#### See also

- DefaultChecked, DefaultPersistent<sup>(303)</sup>
- TRVActionItemProperties<sup>(324)</sup>.ActionInsertTable<sup>(326)</sup>

### 1.3.8.30.1.2 TrvActionTableProperties.BackgroundGraphicFilter

Determines the file masks (filters) available in the file opening dialog displayed to open a picture to assign to a background picture of table or selected cells.

```
property BackgroundGraphicFilter: TRVALocString(348);
```

If this property is not empty, it is assigned to the Filter property of the file opening dialog (TOpenPictureDialog). Otherwise, a filter containing all registered graphic formats is created.

#### Default value:

" (empty string)

#### See also:

- TrvActionInsertPicture.Filter<sup>(252)</sup>
- TrvActionItemProperties.GraphicFilter<sup>(328)</sup>
- TrvActionItemProperties.BackgroundGraphicFilter<sup>(327)</sup>

### 1.3.8.30.1.3 TrvActionTableProperties.DefaultChecked, DefaultPersistent

These properties define the state of "Default" checkbox in the dialog.

```
property DefaultChecked: Boolean;
property DefaultPersistent: Boolean
```

If "Default" checkbox is checked, when properties are applied, they are assigned to "insert table" action(s) as well, so they become default properties for new tables.

**DefaultChecked** specifies the initial value of this checkbox (when the form is shown).

if **DefaultPersistent** = *True*, when the user pressed "OK" in the dialog, **DefaultChecked** property is updated according to the checkbox state (so it will have the same state when the dialog will be shown next time).

### Default value

*False*

### See also

- ActionInsertTable, UpdateAllInsertTableActions <sup>(303)</sup>
- TRVActionItemProperties <sup>(324)</sup>.DefaultChecked, DefaultPersistent <sup>(327)</sup>

## 1.3.8.31 TrvActionTableRCBase

TrvActionTableRCBase is a base class for the actions for insertion and deletion of table rows and columns.

**Unit** RichViewActions <sup>(92)</sup>;

### Syntax

```
TrvActionTableRCBase = class (TrvActionTableCell (282))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

### Description

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

This action is not used directly. The following actions are inherited from it:

- TrvActionTableInsertRowsAbove <sup>(298)</sup>
- TrvActionTableInsertRowsBelow <sup>(299)</sup>
- TrvActionTableInsertColLeft <sup>(297)</sup>
- TrvActionTableInsertColRight <sup>(297)</sup>
- TrvActionTableDeleteRows <sup>(295)</sup>
- TrvActionTableDeleteCols <sup>(295)</sup>
- TrvActionTableSelectCols <sup>(305)</sup>
- TrvActionTableSelectRows <sup>(306)</sup>

### 1.3.8.32 TrvActionTableSelectCell

TrvActionTableSelectCell is the action for "Table | Select Cell" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableSelectCell = class (TrvActionTableCell(282))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>

#### Description

The action selects the edited table cell.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionTableSelectCols<sup>(305)</sup>
- TrvActionTableSelectRows<sup>(306)</sup>
- TrvActionTableSelectTable<sup>(306)</sup>

### 1.3.8.33 TrvActionTableSelectCols

TrvActionTableSelectCols is the action for "Table | Select Columns" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionTableSelectCols = class (TrvActionTableRCBase(304))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionTableCell*<sup>(282)</sup>  
*TrvActionTableRCBase*<sup>(304)</sup>

## Description

The action selects table columns containing selected cells (or the edited cell).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

## See also:

- TrvActionTableSelectRows <sup>(306)</sup>
- TrvActionTableSelectCell <sup>(305)</sup>
- TrvActionTableSelectTable <sup>(306)</sup>

### 1.3.8.34 TrvActionTableSelectRows

TrvActionTableSelectRows is the action for "Table | Select Rows" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableSelectRows = class(TrvActionTableRCBase (304))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>  
*TrvActionTableRCBase* <sup>(304)</sup>

## Description

The action selects table rows containing selected cells (or the edited cell).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

## See also:

- TrvActionTableSelectCols <sup>(305)</sup>
- TrvActionTableSelectCell <sup>(305)</sup>
- TrvActionTableSelectTable <sup>(306)</sup>

### 1.3.8.35 TrvActionTableSelectTable

TrvActionTableSelectTable is the action for "Table | Select Table" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableSelectTable = class(TrvActionTableCell (282))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

## Description

The action selects all cells in the table at the caret position.

Note: a selection containing a table, and a selection containing all selected table cells are different selections, although they look identically.

When all cells are selected (for example, after execution of this action), the user can perform table operations (for example, inserting rows or merging cells). When the user presses **Delete** key, all cells are cleared. TCustomRichViewEdit can copy such selection to the Clipboard in RVF and text format, but cannot copy in RTF format yet.

When the table is selected, the user cannot perform table operations. When the user presses **Delete** key, the table is deleted. TCustomRichViewEdit can copy such selection to the Clipboard in all supported formats.

## See also:

- TrvActionTableSelectCols <sup>(305)</sup>
- TrvActionTableSelectRows <sup>(306)</sup>
- TrvActionTableSelectCell <sup>(305)</sup>

## 1.3.8.36 TrvActionTableSort

TrvActionTableSort is the action for "Table | Sort" command.

**Unit** RichViewActions <sup>(92)</sup> ;

## Syntax

```
TrvActionTableSort = class (TrvActionTableCell (282))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

## Description

The action sorts the table rows. If the action is called when more than one row are selected, it sorts the selected rows; otherwise, it sorts all rows.

Table heading rows are never sorted. Additionally, you can exclude the top [selected] row from sorting.

The action displays a dialog window allowing to specify parameters for sorting:

- column to sort by;
- ascending/descending order;
- sort as numbers or as text; if as text, case sensitivity

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

### 1.3.8.37 TrvActionTableSplit

TrvActionTableSplit is the action for "Table | Split Table" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableSplit = class (TrvActionTableCell (282))
```

## Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

## Description

This action splits the current table into two tables. The second table starts from the first selected row.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of TrvAction.

### 1.3.8.38 TrvActionTableSplitCells

TrvActionTableSplitCells is the action for "Table | Split Cells" command.

**Unit** RichViewActions <sup>(92)</sup>;

## Syntax

```
TrvActionTableSplitCells = class (TrvActionTableCell (282))
```

## Hierarchy

*TObject*



*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

### Description

The action displays a dialog where the user can choose:

- to unmerge the selected cells (if the selection contains cells having values of ColSpan or RowSpan properties greater than 1);
- to split the selected cells into the specified number of rows and columns (optionally with preliminary merging).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

### 1.3.8.39 TrvActionTableToText

*TrvActionTableToText* is the action for "Table | Convert to Text" command.

**Unit** RichViewActions <sup>(92)</sup>;

### Syntax

```
TrvActionTableToText = class (TrvActionTableCell (282))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>  
*TrvActionTableCell* <sup>(282)</sup>

### Description

This action converts the current table to text (in other words, it removes the table without removing its content).

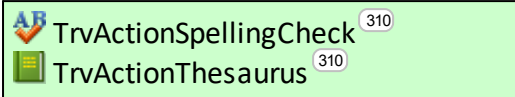
The action allows choosing a delimiter to insert between text from cells: line break, tab, semicolon, or comma. Delimiters are inserted using the current text style.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

## 1.3.9 Spelling check and thesaurus

### Spelling Check and Thesaurus Actions

This group of actions allows using third-party spelling checkers and thesauri



#### 1.3.9.1 TrvActionSpellingCheck

TrvActionSpellingCheck is the action for "Spell Check" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionSpellingCheck = class (TrvAction(311))
```

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

##### Description

This action is enabled if a spelling check interface component is assigned to ControlPanel<sup>(335)</sup>.SpellInterface<sup>(54)</sup>.

The action starts a spelling checking process; when a misspelled word is found, a dialog is displayed to suggests a replacement.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### 1.3.9.2 TrvActionThesaurus

TrvActionThesaurus is the action for "Thesaurus" command.

**Unit** RichViewActions<sup>(92)</sup>;

##### Syntax

```
TrvActionThesaurus = class (TrvAction(311))
```

##### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*

*TBasicAction**TContainedAction**TCustomAction**TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>

## Description

This action is enabled a spelling check interface component is assigned to *ControlPanel* <sup>(335)</sup>. *SpellInterface* <sup>(54)</sup>, and this component supports a thesaurus.

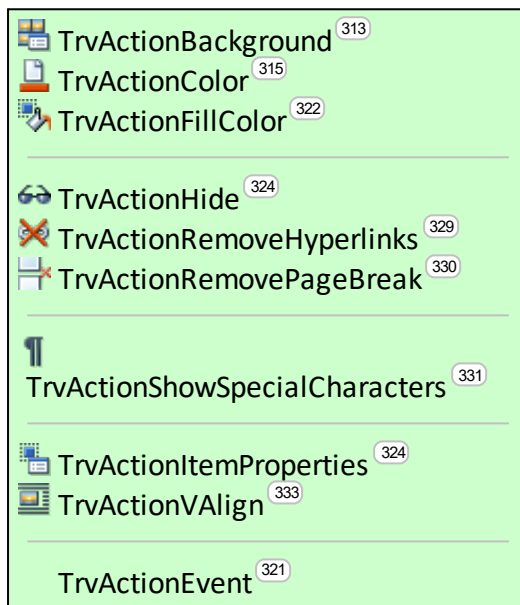
In this version, a thesaurus is provided only by *TRVAAddictSpellInterface* <sup>(356)</sup> component.

The action offers synonyms for the current word.

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

## 1.3.10 Other actions

### Other Actions



### 1.3.10.1 TrvAction

*TrvAction* is a base class for all *RichViewActions* working with *TCustomRichViewEdit*.

**Unit** *RichViewActions* <sup>(92)</sup>;

#### Syntax

```
TrvAction = class (TrvCustomAction (333))
```

#### Hierarchy

*TObject**TPersistent**TComponent*

*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>333</sup>

## Description

This action is not used directly, but all RichViewActions working with TCustomRichViewEdit are inherited from it.

### 1.3.10.1.1 Properties

#### In TrvAction

■ Control <sup>312</sup>

#### Derived from TrvCustomAction <sup>333</sup>

■ Caption <sup>334</sup>  
 ■ ControlPanel <sup>335</sup>  
 ■ Disabled <sup>335</sup>  
 ■ Hint <sup>335</sup>

#### Derived from TAction

■ AutoCheck  
 ■ Caption  
 ■ Checked  
     DisableIfNoHandler  
 ■ Enabled  
 ■ GroupIndex  
 ■ HelpContext  
 ■ HelpKeyword  
 ■ HelpType  
 ■ Hint  
 ■ ImageIndex  
 ■ Name  
 ■ SecondaryShortCuts  
 ■ ShortCut  
 ■ Visible

#### 1.3.10.1.1.1 TrvAction.Control

Defines the editor for this action.

**property** Control: TCustomRVControl;

If an editor is assigned to this property, the action works with this component.

Controls of the following types can be assigned to this property:

- TRichViewEdit
- TDBRichViewEdit

- TSRichViewEdit (ScaleRichView)
- TDBSRichViewEdit (ScaleRichView)

Otherwise, if `GetControlPanel`<sup>(336)</sup>.`DefaultControl`<sup>(43)</sup> is assigned, the action works with it.

Otherwise, if an editor component is focused, the action works with it.

Otherwise, it works with the first found editor component.

### Special actions

The following actions automatically assign this property, if it was not assigned before the first execution:

- TrvActionStyleInspector<sup>(216)</sup>;
- TrvActionEditNote<sup>(260)</sup>.

They need this property assigned, because they display information for the current item in the target editor, even when the caret is moved after the execution of the action.

### 1.3.10.2 TrvActionBackground

TrvActionBackground is the action for "Background" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionBackground = class (TrvAction(311))
```

#### Hierarchy

```

TObject
TPersistent
TComponent
TBasicAction
TContainedAction
TCustomAction
TAction
TrvCustomAction(333)
TrvAction(311)

```

#### Description

This action changes background (Color, BackgroundBitmap, BackgroundStyle) and "padding" (LeftMargin, TopMargin, RightMargin, BottomMargin) properties of the target editor. "Padding" can be changed only if `CanChangeMargins`<sup>(314)</sup> = `True`.

if `rvfoSaveBack` and/or `rvfoSaveLayout` are included in the target editor's `RVFOptions` property, the action changes background and/or margin properties as an editing operation (and these changes can be undone by the user).

The action calls the following events:

- TCustomRichViewEdit.OnChange (called by TCustomRichViewEdit.Change, see above)
- GetControlPanel<sup>(336)</sup>.OnMarginsChanged<sup>(65)</sup> (only if the action changed values of LeftMargin, TopMargin, RightMargin, BottomMargin properties of TCustomRichViewEdit)

- GetControlPanel<sup>(336)</sup>.OnBackgroundChange<sup>(61)</sup>

### 1.3.10.2.1 Properties

#### In TrvActionBackground

---

- CanChangeMargins<sup>(314)</sup>
- ▶ ImageFileName<sup>(315)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.10.2.1.1 TrvActionBackground.CanChangeMargins

Specifies whether the action can change margin properties of the target TRichViewEdit control

**property** CanChangeMargins: Boolean;

If *True*, the "Padding" button is added to the action's dialog, allowing to define values for LeftMargin, RightMargin, TopMargin, BottomMargin properties of the target TRichViewEdit control.

It's recommended to hide this button when working with TScaleRichViewEdit controls from ScaleRichView.

#### Default value

*True*

### 1.3.10.2.1.2 TrvActionBackground.ImageFileName

Returns the file name (full path) of image file assigned to background bitmap.

**property** ImageFileName: TRVUnicodeString;

You can use this property to store this path somewhere.

The following events can be used for this:

- OnChange<sup>(315)</sup>
- GetControlPanel<sup>(336)</sup>.OnBackgroundChange<sup>(61)</sup>

### 1.3.10.2.2 Events

#### In TrvActionBackground

OnChange<sup>(315)</sup>

### 1.3.10.2.2.1 TrvActionBackground.OnChange

Occurs after the background is changed by this action.

**property** OnChange: TRVAEditEvent<sup>(384)</sup>;

You can use either this event or GetControlPanel<sup>(336)</sup>.OnBackgroundChange<sup>(61)</sup>.

## 1.3.10.3 TrvActionColor

TrvActionColor is the action for "Background Color" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionCustomColor = **class** (TrvActionCustomColor<sup>(318)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>

#### Description

The action changes the target editor's Color property.

if *rvfoSaveBack* is included in the target editor's RVFOptions property, this change is made as an editing operation (and can be undone by the user)

The action calls GetControlPanel<sup>(336)</sup>.OnBackgroundChange<sup>(61)</sup> event.

This action does not introduce any new properties and events in addition to properties and events of `TrvActionCustomColor`<sup>(318)</sup>.

### 1.3.10.4 TrvActionUserDefinedColor

`TrvActionUserDefinedColor` allows changing a user-defined color.

**Unit** `RichViewActions`<sup>(92)</sup>;

#### Syntax

```
TrvActionUserDefinedColor = class (TrvActionCustomColor(318))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>  
*TrvActionCustomColor*<sup>(318)</sup>

#### Description

The action allows changing some color and (optionally) its opacity. You can provide the current color value in `OnGetColor`<sup>(318)</sup> event, and apply a new color in `OnColorSelected`<sup>(317)</sup> event.

Assign `UseOpacity`<sup>(317)</sup> = True if you want to change a color opacity.

#### 1.3.10.4.1 Properties

##### TrvActionUserDefinedColor

■ `UseOpacity`<sup>(317)</sup>

##### Derived from TrvActionCustomColor<sup>(318)</sup>

■ `CallerControl`<sup>(319)</sup>  
 ■ `Color`<sup>(319)</sup>  
 ■ `Opacity`<sup>(320)</sup>  
 ■ `UserInterface`<sup>(320)</sup>

##### Derived from TrvAction<sup>(311)</sup>

■ `Control`<sup>(312)</sup>

##### Derived from TrvCustomAction<sup>(333)</sup>

■ `Caption`<sup>(334)</sup>  
 ■ `ControlPanel`<sup>(335)</sup>  
 ■ `Disabled`<sup>(335)</sup>  
 ■ `Hint`<sup>(335)</sup>



---

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.10.4.1.1 TrvActionUserDefinedColor.UseOpacity

Specifies whether the action uses Opacity<sup>(320)</sup>.

**property** UseOpacity: Boolean;

**Default value:**

*False*

### 1.3.10.4.2 Events

#### In TrvActionUserDefinedColor

---

- OnColorSelected<sup>(317)</sup>
- OnGetColor<sup>(318)</sup>

#### Derived from TrvActionCustomColor<sup>(318)</sup>

---

- OnHideColorPicker<sup>(321)</sup>
- OnShowColorPicker<sup>(321)</sup>

### 1.3.10.4.2.1 TrvActionUserDefinedColor.OnColorSelected

Occurs when a color and an opacity have been changed by the user.

**property** OnColorSelected: TRVAEditEvent<sup>(384)</sup>;

This even occurs when the user changed values of a color and an opacity via user interface (an opacity can be changed only if UseOpacity<sup>(317)</sup> = *True*).

These values are already assigned to Color<sup>(319)</sup> and Opacity<sup>(320)</sup> properties.

#### 1.3.10.4.2.2 TrvActionUserDefinedColor.OnGetColor

Occurs when the action needs to know the current values of a color and an opacity.

**property** OnGetColor: TRVAEditEvent<sup>(384)</sup>;

This even occurs when the action needs to initialize user interface controls with the proper values of a color and an opacity (an opacity is used only if UseOpacity<sup>(317)</sup> = *True*).

If you process this event, assign the proper values to Color<sup>(319)</sup> and Opacity<sup>(320)</sup> properties.

### 1.3.10.5 TrvActionCustomColor

TrvActionCustomColor is a base class of the actions for color changing.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionCustomColor = **class** (TrvAction<sup>(311)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action is not used directly. The following actions are inherited from it:

- TrvActionFontCustomColor<sup>(149)</sup>
- TrvActionFontColor<sup>(148)</sup>
- TrvActionFontBackColor<sup>(147)</sup>
- TrvActionParaCustomColor<sup>(198)</sup>
- TrvActionParaColor<sup>(195)</sup>
- TrvActionColor<sup>(315)</sup>

#### 1.3.10.5.1 Properties

##### In TrvActionCustomColor

- CallerControl<sup>(319)</sup>
- Color<sup>(319)</sup>
- Opacity<sup>(320)</sup>
- UserInterface<sup>(320)</sup>

##### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

---

## Derived from TrvCustomAction <sup>333</sup>

---

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.10.5.1.1 TrvActionCustomColor.CallerControl

Specifies the control (for example a toolbar button) which executed this action.

**property** CallerControl: TControl;

Assign this property if you use Delphi 5. For the newer versions of Delphi, the caller control is determined automatically (using ActionComponent property).

A color-picker window is positioned relative to this control. If this control is undefined, a color-picker window is displayed at the mouse pointer position.

### 1.3.10.5.1.2 TrvActionCustomColor.Color

Specifies the color to apply.

**property** Color: TColor;

See the topic about UserInterface <sup>320</sup> property for details.

**Default value for TrvActionCustomColor:**

*c/None*

**Default value for TrvActionFontColor <sup>148</sup>:**

*c/WindowText*

### 1.3.10.5.1.3 TrvActionCustomColor.Opacity

Specifies the color opacity to apply.

**property** Opacity: TRVOpacity;

See the topic about UserInterface<sup>(320)</sup> property for details.

Not all actions inherited from TrvActionCustomColor use opacity.

**Default value:**

100000 (100% opaque)

### 1.3.10.5.1.4 TrvActionCustomColor.UserInterface

Defines the way how the user chooses a color.

**type**

```
TrvaColorInterface =  
    (rvacNone, rvacColorDialog, rvacAdvanced);
```

**property** UserInterface: TrvaColorInterface;

Value	Meaning
<i>rvacNone</i>	Color is not chosen by the user. When the action is executed, values specified in Color <sup>(319)</sup> and Opacity <sup>(320)</sup> * properties are applied.
<i>rvacColorDialog</i>	Color is chosen using TColorDialog (or GetControlPanel <sup>(336)</sup> .ColorDialogInterface <sup>(42)</sup> ). Values of Color <sup>(319)</sup> and Opacity <sup>(320)</sup> * properties before the execution is ignored (they are taken from the target editor). When the color and opacity are chosen, they are assigned to Color <sup>(319)</sup> and Opacity <sup>(320)</sup> * properties. Note: TColorDialog cannot edit opacity. It can be edited only if GetControlPanel <sup>(336)</sup> .ColorDialogInterface <sup>(42)</sup> supports it.
<i>rvacAdvanced</i>	Color is chosen using a special popup color-picker window. Values of Color <sup>(319)</sup> and Opacity <sup>(320)</sup> * properties before the execution is ignored (they are taken from the target editor). When the color and opacity are chosen, they are assigned to Color <sup>(319)</sup> and Opacity <sup>(320)</sup> * properties.

\* only for actions that use opacity.

This action uses `GetControlPanel(336).ColorDialog(42)`, if assigned. Otherwise it creates a temporal color dialog.

In *rvacAdvanced* mode, `GetControlPanel(336).OnGetActionControlCoords(64)` may be used to position a color-picker window.

#### Default value:

*rvacAdvanced*

### 1.3.10.5.2 Events

#### In TrvActionCustomColor

- OnHideColorPicker<sup>(321)</sup>
- OnShowColorPicker<sup>(321)</sup>

##### 1.3.10.5.2.1 TrvActionCustomColor.OnShowColorPicker

Occurs before the color-picker window is displayed.

**property** OnShowColorPicker: TNotifyEvent;

This event occurs if `UserInterface(320) = rvacAdvanced`.

##### 1.3.10.5.2.2 TrvActionCustomColor.OnHideColorPicker

Occurs when the color-picker window is destroyed.

**property** OnHideColorPicker: TNotifyEvent;

This event occurs if `UserInterface(320) = rvacAdvanced`.

### 1.3.10.6 TrvActionEvent

TrvActionEvent is the action to perform custom operations.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionEvent = **class** (TrvAction<sup>(311)</sup>)

#### Hierarchy

TObject  
 TPersistent  
 TComponent  
 TBasicAction  
 TContainedAction  
 TCustomAction  
 TAction  
 TrvCustomAction<sup>(333)</sup>  
 TrvAction<sup>(311)</sup>

#### Description

This action calls `OnExecute(322)` and `OnUpdate(322)` events allowing to implement custom commands.

This action does not have predefined Caption and Hint. Assign these properties yourself.

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also:

- TrvActionInsertText<sup>(256)</sup>

### 1.3.10.6.1 Events

#### In TrvActionEvent

- OnExecute<sup>(322)</sup>
- OnUpdate<sup>(322)</sup>

##### 1.3.10.6.1.1 TrvActionEvent.OnExecute

Occurs when the action is executed.

#### type

```
TRVAEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit) of object;
```

**property** OnExecute: TRVAEvent;

##### 1.3.10.6.1.2 TrvActionEvent.OnUpdate

Occurs when the application is idle or when the action list updates.

#### type

```
TRVAEvent = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit) of object;
```

**property** OnUpdate: TRVAEvent;

Update values of Enabled and Checked properties (and may be values of other action's properties) in this event.

This event is not called if Disabled<sup>(335)</sup>=True, or GetControlPanel<sup>(336)</sup>.ActionsEnabled<sup>(41)</sup>=False.

### 1.3.10.7 TrvActionFillColor

TrvActionFillColor is the action for "Fill Color" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionFillColor = class (TrvAction(311))
```

#### Hierarchy

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction
```

*TAction**TrvCustomAction* <sup>(333)</sup>*TrvAction* <sup>(311)</sup>**Description**

This action can change a background color of:

- text (RVStyle.TextStyles[].BackColor),
- paragraph (RVStyle.ParaStyles[].Background.Color),
- table cell (table.Cells[].Color)
- table (table.Color).

For paragraphs, it can also change padding (RVStyle.ParaStyles[].Background.BorderOffsets).

The action displays a dialog where the user can define a color and an object for application.

This is a redundant action, because these properties can be changed by other actions. But it may be convenient to have an action allowing to change all these properties.

**1.3.10.7.1 Properties****In TrvActionFillColor**

- AllowApplyingTo <sup>(324)</sup>

**Derived from TrvAction** <sup>(311)</sup>

- Control <sup>(312)</sup>

**Derived from TrvCustomAction** <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>
- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

**Derived from TAction**

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.10.7.1.1 TrvActionFillColor.AllowApplyingTo

Contains objects that can be changed by this action.

#### type

```
TRVAFillColorApplyToElement = (rvafcaText, rvafcaParagraph,
    rvafcaCell, rvafcaTable);
TRVAFillColorApplyToSet = set of TRVAFillColorApplyToElement;
```

**property** AllowApplyingTo: TRVAFillColorApplyToSet;

The action can change a background color only for objects listed in this property.

#### Default value:

[rvafcaText..rvafcaTable]

## 1.3.10.8 TrvActionHide

TrvActionHide switches visibility of the selected fragment.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionHide = class (TrvAction(311))
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

This action makes the selected text and objects hidden (or shows them if they are already hidden).

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

#### See also

- TrvActionShowSpecialCharacters<sup>(331)</sup>

## 1.3.10.9 TrvActionItemProperties

TrvActionItemProperties is the action for "Object Properties" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

```
TrvActionItemProperties = class (TrvAction(311))
```

#### Hierarchy

*TObject*



*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

## Description

This action displays a dialog for changing properties of items of the following types:

- pictures and "hot-pictures"
- "breaks" (horizontal lines)
- tables
- sidenotes and text box items
- numbered sequences
- page numbers and page counts
- other (using OnCustomItemPropertiesDialog <sup>(329)</sup> event)

The user can check "Default" check box in this dialog to assign properties of new tables and horizontal lines.

If *rvoAssignImageFileNames* is included in the Options property of the target editor, path to the graphic file is stored in the document:

- in the *rvesplImageFileName* item property (see the TRichView help file, TRVExtraItemStrProperty type) for tables, pictures and "hot-pictures";
- in BackgroundImageFileName cell property for table cells.

## See also:

- TrvActionTableProperties <sup>(301)</sup>

## 1.3.10.9.1 Properties

### In TrvActionItemProperties

- ActionInsertHLine <sup>(326)</sup>
- ActionInsertTable <sup>(326)</sup>
- BackgroundGraphicFilter <sup>(327)</sup>
- DefaultChecked <sup>(327)</sup>
- DefaultPersistent <sup>(327)</sup>
- GraphicFilter <sup>(328)</sup>
- UpdateAllInsertTableActions <sup>(326)</sup>

### Derived from TrvAction <sup>(311)</sup>

- Control <sup>(312)</sup>

### Derived from TrvCustomAction <sup>(333)</sup>

- Caption <sup>(334)</sup>
- ControlPanel <sup>(335)</sup>

- Disabled <sup>(335)</sup>
- Hint <sup>(335)</sup>

## Derived from TAction

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.10.9.1.1 TrvActionItemProperties.ActionInsertHLine

Specifies the "insert horizontal line" action for applying default properties to.

**property** ActionInsertHLine: TrvActionInsertHLine <sup>(231)</sup>;

If this link is not defined, the first found TrvActionInsertHLine <sup>(231)</sup> action on the same form/datamodule is used.

If "Default" checkbox is checked, when properties are applied, they are assigned to "insert table" and "insert horizontal line" action as well, so they become default properties for new tables / lines.

**See also:**

- ActionInsertTable, UpdateAllInsertTableActions <sup>(326)</sup>
- DefaultChecked, DefaultPersistent <sup>(327)</sup>

### 1.3.10.9.1.2 TrvActionItemProperties.ActionInsertTable, UpdateAllInsertTableActions

ActionInsertTable specifies the "insert table" action for applying default properties to.

UpdateAllInsertTableActions allows applying to all actions "insert table" actions on the same form/datamodule.

**property** ActionInsertTable: TrvActionInsertTable <sup>(271)</sup>;

**property** UpdateAllInsertTableActions: Boolean;

If "Default" checkbox is checked, when properties are applied, they are assigned to "insert table" and "insert horizontal line" action as well, so they become default properties for new tables / lines.

If **ActionInsertTable** is defined, properties are applied only to this action. If this link is not defined, they are applied to the first found TrvActionInsertTable <sup>(271)</sup> action on the same form/datamodule (if **UpdateAllInsertTableActions=False**), or to all TrvActionInsertTable <sup>(271)</sup> actions on the same form/datamodule (if **UpdateAllInsertTableActions=True**).

**Note:** Report Workshop includes an additional action for inserting a report table:

TrvrActionInsertTable, inherited from TrvActionInsertTable<sup>(271)</sup>. If you want to apply changes both to instances of TrvrActionInsertTable and TrvActionInsertTable<sup>(271)</sup>, **UpdateAllInsertTableActions** must be True.

#### Default value

**UpdateAllInsertTableActions** = *True*

#### See also:

- ActionInsertHLine<sup>(326)</sup>
- DefaultChecked, DefaultPersistent<sup>(327)</sup>
- TrvActionTableProperties<sup>(301)</sup>.ActionInsertTable, UpdateAllInsertTableActions<sup>(303)</sup>

#### 1.3.10.9.1.3 TrvActionItemProperties.BackgroundGraphicFilter

Determines the file masks (filters) available in the file opening dialog displayed to open a picture to assign to a background picture of table or selected cells.

**property** BackgroundGraphicFilter: TRVLocString<sup>(348)</sup>;

If this property is not empty, it is assigned to the Filter property of the file opening dialog (TOpenPictureDialog). Otherwise, a filter containing all registered graphic formats is created.

#### Default value:

" (empty string)

#### See also:

- TrvActionInsertPicture.Filter<sup>(252)</sup>
- GraphicFilter<sup>(328)</sup>
- TrvActionTableProperties.BackgroundGraphicFilter<sup>(303)</sup>

#### 1.3.10.9.1.4 TrvActionItemProperties.DefaultChecked, DefaultPersistent

These properties define the state of "Default" checkbox in the dialog.

**property** DefaultChecked: Boolean;

**property** DefaultPersistent: Boolean

If "Default" checkbox is checked, when properties are applied, they are assigned to "insert table" and "insert horizontal line" action as well, so they become default properties for new tables / lines.

**DefaultChecked** specifies the initial value of this checkbox (when the form is shown).

if **DefaultPersistent** = *True*, when the user pressed "OK" in the dialog, **DefaultChecked** property is updated according to the checkbox state (so it will have the same state when the dialog will be shown next time).

#### Default value

*False*

#### See also

- ActionInsertTable, UpdateAllInsertTableActions<sup>(326)</sup>
- ActionInsertHLine<sup>(326)</sup>
- TrvActionTableProperties<sup>(301)</sup>.DefaultChecked, DefaultPersistent<sup>(303)</sup>

### 1.3.10.9.1.5 TrvActionItemProperties.GraphicFilter

Determines the file masks (filters) available in the file opening dialog for pictures and "hot-pictures".

**property** GraphicFilter: TRVALocString<sup>348</sup>;

If this property is not empty, it is assigned to the Filter property of the file opening dialog (TOpenPictureDialog). Otherwise, a filter containing all registered graphic formats is created.

**Default value:**

" (empty string)

**See also:**

- TrvActionInsertPicture.Filter<sup>252</sup>
- TrvActionItemProperties.BackgroundGraphicFilter<sup>327</sup>
- TrvActionTableProperties.BackgroundGraphicFilter<sup>303</sup>

### 1.3.10.9.1.6 TrvActionItemProperties.StoreImageFileName

Specifies whether file names (full paths) are stored for images assigned by this action.

**property** StoreImageFileName: Boolean;

If this property is *True*, path to the graphic file is stored in the document:

- in the *rvesplImageFileName* item property (see the RichView help file, TRVExtraltemStrProperty type) for tables, pictures and "hot-pictures";
- in BackgroundImageFileName cell property for table cells.

This is a recommended way to store path to graphic files. You can use these file names when saving HTML files (include *rvhtmlsioUseItemImageFileNames* in HTMLSaveProperties.ImageOptions of the target editor).

**Default value**

*False*

**See also:**

- TCustomRichView.RTFReadProperties.StoreImagesFileNames
- HTML Import (for images inserted with HTML, paths to graphic files are always stored in *rvesplImageFileName*)

### 1.3.10.9.2 Events

#### In TrvActionItemProperties

- OnCanApply<sup>328</sup>
- OnCustomItemPropertiesDialog<sup>329</sup>

### 1.3.10.9.2.1 TrvActionItemProperties.OnCanApply

Requests whether a custom dialog can be displayed for the given item.

**type**

```
TRVCanApplyEvent = procedure (Sender: TObject;
    Editor: TCustomRichViewEdit;
    Item: TCustomRVItemInfo; var CanApply: Boolean) of object;
```

**property** OnCanApply: TRVCanApplyEvent;

This event occurs only if OnCustomItemPropertiesDialog<sup>(329)</sup> event is assigned.

A dialog will be displayed for **Item** in the **Editor** (for the selected item, if only one item is selected), if **CanApply** is set to *True*.

#### 1.3.10.9.2.2 TrvActionItemProperties.OnCustomItemPropertiesDialog

Allows displaying a dialog for editing properties of different item types, or replacing dialogs for items of the standard types, or forbidding displaying a standard dialog for items of standard types.

**type**

```
TRVCustomItemPropertiesDialog = procedure (Sender: TObject;  
    Editor: TCustomRichViewEdit; var DoDefault: Boolean) of object;
```

**property** OnCustomItemPropertiesDialog: TRVCustomItemPropertiesDialog;

A dialog must be displayed for the current item in the **Editor** (or for the selected item, if only one item is selected).

Assign *False* to **DoDefault** if you do not want to display a standard dialog after calling this event.

**See also:**

- OnCanApply<sup>(328)</sup>

### 1.3.10.10TrvActionRemoveHyperlinks

TrvActionRemoveHyperlinks is the action for "Remove Hyperlinks" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionRemoveHyperlinks = class (TrvAction(311))
```

**Hierarchy**

```
TObject  
TPersistent  
TComponent  
TBasicAction  
TContainedAction  
TCustomAction  
TAction  
TrvCustomAction(333)  
TrvAction(311)
```

**Description**

This action converts all hyperlinks in the selected fragment to a plain text. This action does the same work as TrvActionInsertHyperlink<sup>(232)</sup>, if the user entered an empty link target.

This action is a wrapper for TrvActionInsertHyperlink<sup>(232)</sup> action referenced in ActionInsertHyperlink property.

### 1.3.10.10.1 Properties

#### In TrvActionRemoveHyperlinks

---

- ActionInsertHyperlink<sup>(330)</sup>

#### Derived from TrvAction<sup>(311)</sup>

---

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

---

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

#### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

#### 1.3.10.10.1.1 TrvActionRemoveHyperlinks.ActionInsertHyperlink

Links this action to TrvActionInsertHyperlink<sup>(232)</sup> action.

**property** ActionInsertHyperlink: TrvActionInsertHyperlink<sup>(232)</sup>;

If this link is not defined, the first found TrvActionInsertHyperlink<sup>(232)</sup> action on the same form/datamodule is used. If no TrvActionInsertHyperlink<sup>(232)</sup> action found, this action is disabled.

### 1.3.10.11TrvActionRemovePageBreak

TrvActionRemovePageBreak is the action for "Remove Page Break" command.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvActionRemovePageBreak = **class** (TrvAction<sup>(311)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

### Description

This action removes a page break from the paragraph at the caret position (or, when called from a table cell, from the current table row).

This action does not introduce any new properties in addition to properties <sup>(312)</sup> of *TrvAction*.

### See also:

- *TrvActionInsertPageBreak* <sup>(248)</sup>

## 1.3.10.12TrvActionShowSpecialCharacters

*TrvActionShowSpecialCharacters* is the action for "Non-printing Characters" command.

**Unit** RichViewActions <sup>(92)</sup> ;

### Syntax

```
TrvActionShowSpecialCharacters = class (TrvAction (311))
```

### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction* <sup>(333)</sup>  
*TrvAction* <sup>(311)</sup>

### Description

The action shows/hides:

- marks for non-printing characters (spaces, non breaking spaces, tabs, soft hyphens, paragraphs separators, line separators) and placeholders of floating items;
- hidden items;
- (if ShowCheckpoints <sup>(332)</sup> = True) checkpoints.

The set of non-printing characters to display is defined in *RVVisibleSpecialCharacters* global variable (declared in *RVStyle.pas*).

The action includes/excludes *rvoShowSpecialCharacters*, *rvoShowHiddenText*, *rvoShowCheckpoints* from *TCustomRichViewEdit.Options* property and reformats *TCustomRichViewEdit*.

This action assumes that these options are included and excluded from Options together. Make sure that it is true for the initial state of editors in your application.

### See also

- TrvActionHide <sup>324</sup>

## 1.3.10.12.1 Properties

### In TrvActionShowSpecialCharacters

---

- ShowCheckpoints <sup>332</sup>

### Derived from TrvAction <sup>311</sup>

---

- Control <sup>312</sup>

### Derived from TrvCustomAction <sup>333</sup>

---

- Caption <sup>334</sup>
- ControlPanel <sup>335</sup>
- Disabled <sup>335</sup>
- Hint <sup>335</sup>

### Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

## 1.3.10.12.1.1 TrvActionShowSpecialCharacters.ShowCheckpoints

Defines whether the action hides/shows checkpoints in the target editor.

**property** ShowCheckpoints: Boolean;

If set, the action, in addition to switching other options, includes/excludes rvoShowCheckpoints in the Options property of the target editor.

Assign *False* to this property if you want checkpoints to be always shown/hidden, or if you want to implement their showing/hiding in other place of your application.



**Default value***True***1.3.10.13TrvActionVAlign**

TrvActionVAlign is the action for "Object Position" command.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax**

```
TrvActionVAlign = class (TrvAction(311))
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

**Description**

This action displays a dialog for changing VAlign property for the items of the following types:

- pictures
- *hot-pictures*
- *bullets* (image lists)
- *hotspots* (image lists)
- controls
- label items
- numbered sequences
- custom item types inherited from TRVRectItemInfo

This action does not introduce any new properties in addition to properties<sup>(312)</sup> of TrvAction.

**1.3.10.14TrvCustomAction**

TrvCustomAction is a base class for all RichViewActions.

**Unit** RichViewActions<sup>(92)</sup>;

**Syntax (normal)**

```
TrvCustomAction = class (TAction)
```

**Syntax (if TNT Controls<sup>(370)</sup> are used)**

```
TrvCustomAction = class (TAction, ITntAction)
```

**Hierarchy**

*TObject*  
*TPersistent*

*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*

## Description

This action is not used directly, but all RichViewActions are inherited from it.

## Main properties:

- ControlPanel<sup>(335)</sup> links this action to a control panel defining additional properties for the action.
- Disabled<sup>(335)</sup> allows to disable this action.

## Direct descendants:

- TrvAction<sup>(311)</sup>
- TrvActionPageSetup<sup>(114)</sup>

## 1.3.10.14.1 Properties

### In TrvCustomAction

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

### Derived from TAction

- AutoCheck
- Caption
- Checked
- DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

## 1.3.10.14.1.1 TrvCustomAction.Caption

Represents the caption of client controls and menu items.

If TNT Controls<sup>(370)</sup> are used, this property overrides the standard action's Caption property:

```
property Caption: WideString;
```

Caption holds the string that is the caption for the action when it is set. The value of Caption is propagated to client controls and menu items.

**See also properties:**

- Hint<sup>(335)</sup>

#### 1.3.10.14.1.2 TrvCustomAction.ControlPanel

Links this action with a control panel.

**property** ControlPanel: TRVAControlPanel<sup>(38)</sup>;

A control panel defines properties used for many actions.

If this property is not assigned (*nil*), the action uses MainRVAControlPanel<sup>(388)</sup>.

**See also:**

- GetControlPanel<sup>(336)</sup>.

#### 1.3.10.14.1.3 TrvCustomAction.Disabled

Allows to disable the action.

**property** Disabled: Boolean;

If *False*, the action is disabled and enabled automatically.

If *True*, the action is disabled.

**Default value:**

*False*

**See also properties:**

- Enabled

**See also:**

- GetControlPanel<sup>(336)</sup>.ActionsEnabled<sup>(41)</sup>

#### 1.3.10.14.1.4 TrvCustomAction.Hint

Indicates the Help hint for client controls and menu items.

If TNT Controls<sup>(370)</sup> are used, this property overrides the standard action's Hint property.

**property** Hint: WideString;

Hint holds the string that is the hint for that action when it is set. This value is propagated to client controls and menu items.

**See also properties:**

- Caption<sup>(334)</sup>

#### 1.3.10.14.2 Methods

##### In TrvCustomAction

GetControlPanel<sup>(336)</sup>

### 1.3.10.14.2.1 TrvCustomAction.GetControlPanel

Returns a control panel used by this action.

**function** GetControlPanel: TRVAControlPanel<sup>(38)</sup>;

If ControlPanel<sup>(335)</sup> is assigned, the method returns it. Otherwise it returns MainRVAControlPanel<sup>(388)</sup>.

### 1.3.10.15TrvCustomEditorAction

TrvCustomEditorAction is a base class for action displaying a non-modal window containing TRichViewEdit component.

**Unit** RichViewActions<sup>(92)</sup>;

#### Syntax

TrvCustomEditorAction = **class** (TrvAction<sup>(311)</sup>)

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TBasicAction*  
*TContainedAction*  
*TCustomAction*  
*TAction*  
*TrvCustomAction*<sup>(333)</sup>  
*TrvAction*<sup>(311)</sup>

#### Description

When executed, this action shows Form<sup>(337)</sup> containing SubDocEditor<sup>(337)</sup>.

This class is not used directly. The following actions are inherited from TrvCustomEditorAction:

- TrvActionEditNote<sup>(260)</sup>

### 1.3.10.15.1 Properties

#### In TRVCustomEditorAction

- ▶ Form<sup>(337)</sup>
- ▶ SubDocEditor<sup>(337)</sup>

#### Derived from TrvAction<sup>(311)</sup>

- Control<sup>(312)</sup>

#### Derived from TrvCustomAction<sup>(333)</sup>

- Caption<sup>(334)</sup>
- ControlPanel<sup>(335)</sup>
- Disabled<sup>(335)</sup>
- Hint<sup>(335)</sup>

---

## Derived from TAction

---

- AutoCheck
- Caption
- Checked
  - DisableIfNoHandler
- Enabled
- GroupIndex
- HelpContext
- HelpKeyword
- HelpType
- Hint
- ImageIndex
- Name
- SecondaryShortCuts
- ShortCut
- Visible

### 1.3.10.15.1.1 TrvCustomEditorAction.Form

Returns a form containing TRichViewEdit control.

**property** Form: TForm;

This form is created when the action is executed for the first time. This form is destroyed together with the action: on closing, it only hides itself.

#### See also

- OnFormCreate<sup>(338)</sup>
- OnShowing and OnHide<sup>(338)</sup>
- SubDocEditor<sup>(337)</sup>

### 1.3.10.15.1.2 TrvCustomEditorAction.SubDocEditor

Returns a TRichViewEdit control inserted in Form<sup>(337)</sup>.

**property** SubDocEditor: TRichViewEdit;

This control is created and destroyed together with Form<sup>(337)</sup>,

## 1.3.10.15.2 Events

### In TRVCustomEditorAction

---

- OnFormCreate<sup>(338)</sup>
- OnHide<sup>(338)</sup>
- OnShowing<sup>(338)</sup>

### 1.3.10.15.2.1 TrvCustomEditorAction.OnFormCreate

Occurs after the Form<sup>337</sup> is created.

**property** OnFormCreate: TRVShowFormEvent<sup>388</sup>;

**See also:**

- OnShowing, OnHide<sup>338</sup> (contains example)

### 1.3.10.15.2.2 TrvCustomEditorAction.OnShowing, OnHide

Occurs before Form<sup>337</sup> is shown, and when it is hidden.

**property** OnShowing: TRVShowFormEvent<sup>388</sup>;

**property** OnHide: TNotifyEvent;

#### Example

This example shows how to work with a form displayed by TrvActionEditNote<sup>260</sup>.

Let the main form TfrmMain have:

- RichViewEdit: TRichViewEdit – the main editor;
- RVAControlPanel1: TRVAControlPanel<sup>38</sup>;
- RVAPopupMenu1: TRVAPopupMenu<sup>66</sup>;
- rvActionEditNote1: TrvActionEditNote;
- rvActionStyleInspector1: TrvActionStyleInspector<sup>216</sup>;
- cmbFont: TRVFontComboBox<sup>73</sup>;
- cmbFontSize: TRVFontSizeComboBox<sup>79</sup>;
- cmbStyles: TRVStyleTemplateComboBox<sup>87</sup>.

We want to insert Form<sup>337</sup> at the bottom of TfrmMain.

Initial property settings:

- RVAControlPanel1.DefaultControl<sup>43</sup> := RichViewEdit1;
- rvActionStyleInspector1.Control<sup>312</sup> := RichViewEdit1;
- cmbFont.Editor<sup>35</sup> := RichViewEdit1;
- cmbFontSize.Editor<sup>35</sup> := RichViewEdit1;
- cmbStyles.Editor<sup>89</sup> := RichViewEdit1

// OnFormCreate<sup>338</sup> event

**procedure** TfrmMain.rvActionEditNoteFormCreate(Sender: TrvAction<sup>311</sup>;  
Form: TForm);

**begin**

```
// assigning properties of the note editor
rvActionEditNote1.SubDocEditor.OnEnter := NoteEditorEnter;
rvActionEditNote1.SubDocEditor.OnExit := NoteEditorExit;
rvActionEditNote1.SubDocEditor.PopupMenu := RVAPopupMenu1;
// moving the form to the bottom of frmMain
Form.Align := alBottom;
Form.Height := 100;
Form.Parent := Self;
```

**end;**

// OnShowing event

```
procedure TfrmMain.rvActionEditNoteShowing(Sender: TrvAction;  
    Form: TForm);  
begin  
    // leaving the current note in RichViewEdit1 highlighted  
    // after moving the input focus  
    RichViewEdit1.ForceFieldHighlight := True;  
end;
```

```
// OnHide event  
procedure TfrmMain.rvActionEditNoteHide(Sender: TObject);  
begin  
    // restoring old values of properties  
    RichViewEdit1.ForceFieldHighlight := False;  
    // focusing the main editor  
    if RichViewEdit1.CanFocus then  
        RichViewEdit1.SetFocus;  
end;
```

```
// OnEnter: occurs when the input focus is moved to the note editor  
procedure TfrmMain.NoteEditorEnter(Sender: TObject);  
begin  
    // making the note editor default  
    RVAControlPanell1.DefaultControl :=  
        rvActionsResource.rvActionEditNote1.SubDocEditor;  
    UpdateLinkedControls;  
end;
```

```
// OnExit: making the main editor default  
procedure TfrmMain.NoteEditorExit(Sender: TObject);  
begin  
    RVAControlPanell1.DefaultControl := RichViewEdit1;  
    UpdateLinkedControls;  
end;
```

```
procedure TfrmMain.UpdateLinkedControls;  
begin  
    rvActionStyleInspector1.Control := RVAControlPanell1.DefaultControl;  
    cmbFont.Editor := RVAControlPanell1.DefaultControl;  
    cmbFontSize.Editor := RVAControlPanell1.DefaultControl;  
    cmbStyles.Editor := RVAControlPanell1.DefaultControl;  
end;
```

Note: a note editor is special, it redirects many actions (for example, TrvActionNew<sup>(103)</sup>, TrvActionSave<sup>(122)</sup>, TrvActionPrint<sup>(116)</sup>) to the main editor.

## 1.4 Localization of RichViewActions

### Changing UI language

---

RichViewActions require localization, even if you use only one UI language.

Initially, the language specified in `TRVAControlPanel.Language`<sup>(51)</sup> is used. If `TRVAControlPanel` component is not used, 'English (US)' is used. This language will be used in dialogs displayed by RichViewActions, but Captions and Hints of the actions themselves are not localized yet. Call `RVA_LocalizeForm`<sup>(350)</sup> to localize them.

For changing UI language, use `RVA_ChooseLanguage`<sup>(349)</sup> function. The topic about this function contains an example.

Other localization functions are declared in `RVALocalize`<sup>(341)</sup> unit.

### Hiding untranslated controls

---

It is possible that some new features added to RichViewActions were not translated to all languages yet.

Such controls can be hidden by `ShowUntranslatedControls`<sup>(390)</sup> global variable.

### Excluding languages

---

You can reduce the size of your application by excluding some languages.

To exclude languages, define one more of the following compiler defines in the options of your project:

- `RVA_NO_ARMENIAN`
- `RVA_NO_BULGARIAN`
- `RVA_NO_BYELORUSSIAN`
- `RVA_NO_CATALAN`
- `RVA_NO_CHINESE_SIMPL`
- `RVA_NO_CHINESE_TRAD`
- `RVA_NO_CZECH`
- `RVA_NO_DANISH`
- `RVA_NO_DUTCH`
- `RVA_NO_ENGLISH_US`
- `RVA_NO_FARSI`
- `RVA_NO_FINNISH`
- `RVA_NO_FRENCH`
- `RVA_NO_GERMAN`
- `RVA_NO_HINDI`
- `RVA_NO_HUNGARIAN`
- `RVA_NO_ITALIAN`
- `RVA_NO_LITHUANIAN`
- `RVA_NO_MALAY`
- `RVA_NO_NORWEGIAN`
- `RVA_NO_POLISH`
- `RVA_NO_PORTUGUESE_BR`



- RVA\_NO\_PORTUGUESE\_PT
- RVA\_NO\_ROMANIAN
- RVA\_NO\_RUSSIAN
- RVA\_NO\_SLOVAK
- RVA\_NO\_SLOVENE
- RVA\_NO\_SPANISH
- RVA\_NO\_SWEDISH
- RVA\_NO\_THAI
- RVA\_NO\_TURKISH
- RVA\_NO\_UKRAINIAN

## Resource language

---

There is one special language: "Resource language". It contains the same string as in "English (US)", but they are stored in resources. You can use this language together with an external localization tool.

By default, this language is excluded. You can enable it by removing the following line from RichViewActions.inc:

```
{ $DEFINE RVA_NO_RESOURCELANGUAGE }
```

## Adding a new language

---

**If you want to add a translation to you language (for example, Greek):**

1. Open RVAL\_EngUS.pas, rename it to RVAL\_Greek.pas. Add RVAL\_Greek in "uses" of RVALocalize.pas. (If you use Delphi/C++Builder 2009, save this unit as UTF-8).
2. Translate all text in this unit (do not translate comments!)
3. Change the call of RVA\_RegisterLanguage<sup>347</sup> to RVA\_RegisterLanguage('Greek', Ελληνικά', GREEK\_CHARSET, @Messages).
4. Remove the call of RVA\_SwitchLanguage<sup>347</sup>.
5. Send the translated file to [svt@trichview.com](mailto:svt@trichview.com) for including in the main installation of RichViewActions. Your file will be maintained (it will be corrected when new messages will be added in the future versions of RichViewActions)

### 1.4.1 RVALocalize unit

This unit contains procedures and functions working with a language of user interface in RichViewActions<sup>16</sup>.

## Procedures and functions declared in RVALocalize

---

Procedures returning names of registered UI languages:

- RVA\_EnumLanguages<sup>344</sup>
- RVA\_FillLanguageList<sup>344</sup>

Functions returning properties of the current language:

- RVA\_GetLanguageName<sup>346</sup>
- RVA\_GetCharset<sup>345</sup>
- RVA\_GetHelpFile<sup>345</sup>

Functions for help files:

- RVA\_SetHelpFile <sup>(347)</sup>

Functions returning a translated string:

- RVA\_GetS, RVA\_GetSH, RVA\_GetPC <sup>(346)</sup>
- RVA\_TranslateUnits <sup>(347)</sup>
- RVA\_GetProgressMessage, RVA\_GetPrintingMessage <sup>(343)</sup>

Procedure for changing the current language:

- RVA\_SwitchLanguage <sup>(347)</sup>

## Other procedures

---

The following procedures and functions are declared in other units, but related to localization:

- RVA\_ChooseLanguage <sup>(349)</sup> (displays a dialog to choose a language name, changes the current language)
- RVA\_LocalizeForm <sup>(350)</sup> (localizes all actions on the specified form/datamodule)
- RVA\_LocalizeRuler <sup>(351)</sup> (localizes TRVRuler <sup>(80)</sup> component)
- RVA\_GetColorName <sup>(350)</sup> (returns the color name)

## Types declared in RVALocalize

---

- TRVALanguageName <sup>(348)</sup>
- TRVALocString, TRVALocStrings, TRVALocStringList <sup>(348)</sup>

## Additional information

---

See also:

- Localization of RichViewActions <sup>(340)</sup>

### 1.4.1.1 Procedures and functions

#### Procedures and functions declared in RVALocalize

---

Procedures returning names of registered UI languages:

- RVA\_EnumLanguages <sup>(344)</sup>
- RVA\_FillLanguageList <sup>(344)</sup>

Functions returning properties of the current language:

- RVA\_GetLanguageName <sup>(346)</sup>
- RVA\_GetCharset <sup>(345)</sup>
- RVA\_GetHelpFile <sup>(345)</sup>

Functions for help files:

- RVA\_SetHelpFile <sup>(347)</sup>

Functions returning a translated string:

- RVA\_GetS, RVA\_GetSH, RVA\_GetPC <sup>(346)</sup>
- RVA\_TranslateUnits <sup>(347)</sup>
- RVA\_GetProgressMessage, RVA\_GetPrintingMessage <sup>(343)</sup>

Procedure for changing the current language:

- RVA\_SwitchLanguage<sup>(347)</sup>

## Other procedures

The following procedures and functions are declared in other units, but related to localization:

- RVA\_ChooseLanguage<sup>(349)</sup> (displays a dialog to choose a language name, changes the current language)
- RVA\_LocalizeForm<sup>(350)</sup> (localizes all actions on the specified form/datamodule)
- RVA\_LocalizeRuler<sup>(351)</sup> (localizes TRVRuler<sup>(80)</sup> component)
- RVA\_GetColorName<sup>(350)</sup> (returns the color name)

### 1.4.1.1.1 Functions for progress messages

```
function RVA_GetProgressMessage(Operation: TRVLongOperation;
  ControlPanel: TComponent=nil): TRVALocString(348);
function RVA_GetPrintingMessage(PageCompleted: Integer;
  Step: TRVPrintingStep;
  ControlPanel: TComponent=nil): TRVALocString(348);
```

If **ControlPanel** is a TRVAControlPanel<sup>(38)</sup> component, the functions use it. Otherwise, they use MainRVAControlPanel<sup>(388)</sup>.

**RVA\_GetProgressMessage** returns a localized text message describing the specified long operation. It can be used in TRichViewEdit.OnProgress event.

**RVA\_GetPrintingMessage** returns a localized text message describing the current printing progress. It can be used in TRVPrint.OnSendingToPrinter event.

The name of the current language is returned as ControlPanel.Language<sup>(51)</sup>, or by RVA\_GetLanguageName<sup>(346)</sup> function.

The current language can be changed by assigning a new value to ControlPanel.Language<sup>(51)</sup>, or by calling RVA\_SwitchLanguage<sup>(347)</sup> procedure.

#### Example:

This example assumes that Application.Hint is displayed in a status bar.

```
// TRVAControlPanel.OnDownload(64)
procedure TForm3.RVAControlPanel1Download(Sender: TrvAction;
  const Source: TRVUnicodeString);
begin
  if Source='' then
    Application.Hint := ''
  else
    Application.Hint := Format(RVA_GetS(rvam_msg_Downloading),
      [Source]);
end;
{-----}
// TRVPrint.OnSendingToPrinter
procedure TForm3.RVPrint1SendingToPrinter(Sender: TCustomRichView;
  PageCompleted: Integer; Step: TRVPrintingStep);
begin
  Application.Hint := RVA_GetPrintingMessage(PageCompleted, Step);
end;
```

```

{-----}
// Reading or writing
procedure TForm3.RichViewEdit1Progress(Sender: TCustomRichView;
Operation: TRVLongOperation; Stage: TRVProgressStage;
PercentDone: Byte);
begin
  case Stage of
    rvpstgStarting:
      begin
        ProgressBar1.Position := 0;
        ProgressBar1.Visible := True;
        Application.Hint := RVA_GetProgressMessage(Operation);
      end;
    rvpstgRunning:
      begin
        ProgressBar1.Position := PercentDone;
        ProgressBar1.Update;
      end;
    rvpstgEnding:
      begin
        Application.Hint := '';
        ProgressBar1.Visible := False;
      end;
  end;
end;

```

#### 1.4.1.1.2 RVA\_EnumLanguages procedure

This procedure allows to enumerate all UI languages available for RichViewActions.

**Unit** RVALocalize<sup>(341)</sup>.

```

//type
// TGetStrProc = procedure (const S: string) of object;
procedure RVA_EnumLanguages(Proc: TGetStrProc);

```

This procedure calls **Proc** one time for each language name. Only English language names are passed to this procedure.

Names of languages can be assigned to TRVAControlPanel.Language<sup>(51)</sup> or used as a parameter for RVA\_SwitchLanguage<sup>(347)</sup> procedure.

**See also:**

- RVA\_FillLanguageList<sup>(344)</sup>

#### 1.4.1.1.3 RVA\_FillLanguageList procedure

This procedure adds all UI language names available for RichViewActions in **sl**.

**Unit** RVALocalize<sup>(341)</sup>.

```

procedure RVA_FillLanguageList(sl: TRVALocStrings(348);
English: Boolean = True;
Native: Boolean = False);

```

English	Native	Result
True	False	English language names
False	True	Native language names
True	True	Strings in format: "English language name - native language name" (or English language name, if they are equal)
False	False	English language names

Names of languages can be assigned to `TRVAControlPanel.Language`<sup>(51)</sup> or used as a parameter for `RVA_SwitchLanguage`<sup>(347)</sup> procedure.

#### 1.4.1.1.4 RVA\_GetCharset function

This method returns the charset for the current UI language.

**Unit** `RVALocalize`<sup>(341)</sup>.

**function** `RVA_GetCharset(ControlPanel: TComponent=nil): TFontCharset;`

If **ControlPanel** is a `TRVAControlPanel`<sup>(38)</sup> component, the function uses it. Otherwise, it uses `MainRVAControlPanel`<sup>(388)</sup>.

The name of the current language is returned as `ControlPanel.Language`<sup>(51)</sup>, or by `RVA_GetLanguageName`<sup>(346)</sup> function.

The current language can be changed by assigning a new value to `ControlPanel.Language`<sup>(51)</sup>, or by calling `RVA_SwitchLanguage`<sup>(347)</sup> procedure.

This charset is ignored in Delphi/C++Builder 2009 or newer, because Unicode strings do not need charsets. Use this function for Delphi/C++Builder 5-2007.

`DEFAULT_CHARSET` is used for languages that do not have ANSI charset (Armenian, Hindi). For these languages, localization strings have UTF-8 encoding, and can be used in old versions of Delphi only with TNT Controls<sup>(370)</sup>.

RichViewActions apply this charset to all fonts in all dialogs displayed by RichViewActions.

You should use this property to apply to your own forms (if they support a localization) and to `Screen.HintFont` and `Screen.MenuFont`:

```
Screen.HintFont.Charset := RVA_GetCharset; // Delphi 6-2007 is required
Screen.MenuFont.Charset := RVA_GetCharset; // Delphi 6-2007 is required
```

#### 1.4.1.1.5 RVA\_GetHelpFile function

This method returns the help file name for the current UI language.

**Unit** `RVALocalize`<sup>(341)</sup>.

**function** `RVA_GetHelpFile(ControlPanel: TComponent): String;`

If **ControlPanel** is a `TRVAControlPanel`<sup>(38)</sup> component.

The name of the current language is returned as **ControlPanel.Language**<sup>(51)</sup>, or by **RVA\_GetLanguageName**<sup>(346)</sup> function.

The current language can be changed by assigning a new value to **ControlPanel.Language**<sup>(51)</sup>, or by calling **RVA\_SwitchLanguage**<sup>(347)</sup> procedure.

#### See also:

- **TRVAControlPanel.UseHelpFiles**<sup>(56)</sup>

#### 1.4.1.1.6 RVA\_GetLanguageName function

This method returns the name of the current UI language.

**Unit** **RVALocalize**<sup>(341)</sup>.

```
function RVA_GetLanguageName (
    ControlPanel: TComponent=nil): TRVALanguageName(348);
```

If **ControlPanel** is a **TRVAControlPanel**<sup>(38)</sup> component, the function uses it. Otherwise, it uses **MainRVAControlPanel**<sup>(388)</sup>.

The same value is returned in **ControlPanel.Language**<sup>(51)</sup> property.

The current language can be changed by assigning a new value to **ControlPanel.Language**<sup>(51)</sup>, or by calling **RVA\_SwitchLanguage**<sup>(347)</sup> procedure.

#### 1.4.1.1.7 RVA\_GetS function

This method returns a localized string.

**Unit** **RVALocalize**<sup>(341)</sup>.

```
type
    TRVAMessageID = (rvam_Empty, rvam_menu_File, rvam_menu_Edit, ...);
function RVA_GetS(MsgID: TRVAMessageID;
    ControlPanel: TComponent=nil): TRVALocString(348);
function RVA_GetSH(MsgID: TRVAMessageID;
    ControlPanel: TComponent=nil): TRVALocString(348);
function RVA_GetPC(MsgID: TRVAMessageID;
    ControlPanel: TComponent=nil): PChar
// RVA_GetPC returns WideString for TNT Controls
```

If **ControlPanel** is a **TRVAControlPanel**<sup>(38)</sup> component, the functions use it. Otherwise, they use **MainRVAControlPanel**<sup>(388)</sup>.

These functions return strings corresponding to **MsgID**, translated to the current language. The name of the current language is returned as **ControlPanel.Language**<sup>(51)</sup>, or by **RVA\_GetLanguageName**<sup>(346)</sup> function.

The current language can be changed by assigning a new value to **ControlPanel.Language**<sup>(51)</sup>, or by calling **RVA\_SwitchLanguage**<sup>(347)</sup> procedure.

**RVA\_GetS** just returns a string.

**RVA\_GetSH(MsgId)** returns ' '+**RVA\_GetS(MsgId)**+' ' (used for assigning to captions of radio groups and group boxes).

**RVA\_GetPC** returns the same value as **RVA\_GetS**, but not as a **String** but as **PChar**.

**Note:** if RichViewActions are compiled with TNT Controls <sup>(370)</sup>, these functions return WideString.

#### 1.4.1.1.8 RVA\_SetHelpFile procedure

This method defines the help file that may be used in RichViewActions dialogs.

**Unit** RVALocalize <sup>(341)</sup>.

```
procedure RVA_SetHelpFile(const LanguageName: TRVALanguageName (348);  
    const HelpFile: String);
```

The specified **HelpFile** is used when **LanguageName** language is an active UI language.

**See also:**

- TRVAControlPanel.Language <sup>(51)</sup>
- TRVAControlPanel.UseHelpFiles <sup>(56)</sup>

#### 1.4.1.1.9 RVA\_SwitchLanguage procedure

Changes the current language.

```
procedure RVA_SwitchLanguage(const LanguageName: TRVALanguageName;  
    ControlPanel: TComponent=nil);
```

If **ControlPanel** is a TRVAControlPanel <sup>(38)</sup> component, the function uses it. Otherwise, it uses MainRVAControlPanel <sup>(388)</sup>.

Assigning a new value to ControlPanel.Language <sup>(51)</sup> has the same effect.

To get a list of valid languages, use RVA\_EnumLanguages <sup>(344)</sup> or RVA\_FillLanguageList <sup>(344)</sup>.

You can use either English or native language name as a parameter.

To complete a language changing, the following procedures must be called after RVA\_SwitchLanguage:

- RVA\_LocalizeForm <sup>(350)</sup> for localization of Captions and Hints of all actions on the given form/datamodule;
- RVALocalizeRuler <sup>(351)</sup> for localizations of hints in TRVRuler <sup>(80)</sup> component;
- RVA\_GetCharset <sup>(345)</sup> for changing Charsets of menus, hints and your forms;
- RVA\_GetS <sup>(346)</sup> for localizing standard menu items in your menu (such as "File", "Edit"...).

See also:

- RVA\_ChooseLanguage <sup>(349)</sup>

#### 1.4.1.1.10 RVA\_TranslateUnits procedure

Translates units of measurement to the current language.

```
procedure RVA_TranslateUnits(SL: TStrings; ControlPanel: TComponent=nil);
```

If **ControlPanel** is a TRVAControlPanel <sup>(38)</sup> component, the function uses it. Otherwise, it uses MainRVAControlPanel <sup>(388)</sup>.

**SL** must have 6 items corresponding to the following units, in the specified order:

- Inches,
- Centimeters,

- Millimeters,
- Picas,
- Pixels,
- Points.

The name of the current language is returned as `ControlPanel.Language`<sup>(51)</sup>, or by `RVA_GetLanguageName`<sup>(346)</sup> function.

The current language can be changed by assigning a new value to `ControlPanel.Language`<sup>(51)</sup>, or by calling `RVA_SwitchLanguage`<sup>(347)</sup> procedure.

### 1.4.1.2 Types

#### Types declared in RVALocalize

- `TRVALanguageName`<sup>(348)</sup>
- `TRVALocString`, `TRVALocStrings`, `TRVALocStringList`<sup>(348)</sup>

##### 1.4.1.2.1 TRVALanguageName

A type for UI language name.

**Unit** RVALocalize<sup>(341)</sup>.

**type**

```
TRVALanguageName = type String;
```

**See also:**

- `TRVAControlPanel.Language`<sup>(51)</sup>
- `RVA_GetLanguageName`<sup>(346)</sup>
- `RVA_SwitchLanguage`<sup>(347)</sup>

##### 1.4.1.2.2 TRVALocString, TRVALocStrings, TRVALocStringList

**unit** RVALocalize;

If TNT controls<sup>(370)</sup> are not used:

**type**

```
TRVALocString = String;
TRVALocStrings = TStrings;
TRVALocStringList = TStringList;
```

If TNT controls are used:

**type**

```
TRVALocString = WideString;
TRVALocStrings = TTntStrings;
TRVALocStringList = TTntStringList;
```

Strings of these types are used in user interface of RichViewActions.

If TNT controls are used, or in Delphi/C++Builder 2009 or newer, the strings are Unicode (UTF-16).

In Lazarus, the strings are Unicode (UTF-8).

In older versions of Delphi, the strings are ANSI strings. Charset of these strings for the active language is returned by `RVA_GetCharset`<sup>(345)</sup>.



## 1.4.2 RichViewActions unit

### Procedures and functions declared in RichViewActions

---

- RVA\_ChoseLanguage function<sup>(349)</sup> displays a dialog allowing to switch UI language;
- RVA\_GetColorName<sup>(350)</sup> returns a localized color name;
- RVA\_LocalizeForm<sup>(350)</sup> translates all actions on the specified form/datamodule.

### 1.4.2.1 Procedures and functions

#### Procedures and functions declared in RichViewActions

---

- RVA\_ChoseLanguage function<sup>(349)</sup> displays a dialog allowing to switch UI language;
- RVA\_GetColorName<sup>(350)</sup> returns a localized color name;
- RVA\_LocalizeForm<sup>(350)</sup> translates all actions on the specified form/datamodule.

#### 1.4.2.1.1 RVA\_ChoseLanguage function

Displays a dialog for choosing UI language and applies the chosen language.

**Unit** RichViewActions.

```
function RVA_ChoseLanguage (ControlPanel: TRVAControlPanel(38) = nil): Boolean;
```

This function displays a dialog containing a list with all available UI languages.

If the user pressed "OK", this function calls RVA\_SwitchLanguage<sup>(347)</sup> and returns *True*.

You may need to perform additional operations on the language changing, see RVA\_SwitchLanguage<sup>(347)</sup> for details.

In Delphi 4-2007, this function displays language names in English.

In new versions of Delphi (or if TNT Controls<sup>(370)</sup> are used), this function displays both English and native language names.

**Example (from the ActionTest demo):**

```
// button at the right side of status bar
procedure TForm3.Button1Click(Sender: TObject);
begin
    if RVA_ChoseLanguage then
        Localize;
end;

procedure TForm3.FormCreate(Sender: TObject);
begin
    ...
    Localize;
    ...
end;

procedure TForm3.Localize;
var Index: Integer;
begin
    // Fonts (for Delphi 5-2007)
```

```

Font.Charset          := RVA_GetCharset(345);
StatusBar1.Font.Charset := RVA_GetCharset(345);
// Fonts (for Delphi 6-2007)
Screen.HintFont.Charset := RVA_GetCharset(345);
Screen.MenuFont.Charset := RVA_GetCharset(345);
// Localizing all actions on rvActionsResource
RVA_LocalizeForm(350)(rvActionsResource);
// Localizing all actions on this form
RVA_LocalizeForm(350)(Self);
// Localizing ruler
RVLocalizeRuler(351)(RVRuler1);
// Localizing menus
mitFile.Caption := RVA_GetS(346)(rvam_menu_File);
mitEdit.Caption := RVA_GetS(346)(rvam_menu_Edit);
// Styles
rvActionsResource.rvActionStyleInspector1.UpdateInfo(218);
RVStyleTemplateComboBox1.Localize(91);
// Localizing combobox with units of measurement
Index := cmbUnits.ItemIndex;
RVA_TranslateUnits(347)(cmbUnits.Items);
cmbUnits.ItemIndex := Index;
end;

```

#### 1.4.2.1.2 RVA\_GetColorName function

Returns a localized color name.

**Unit** RichViewActions.

```

function RVA_GetColorName(Color: TColor;
  ControlPanel: TRVAControlPanel(38) = nil):TRVALocString(348);

```

If **ControlPanel** parameter is defined, its Language<sup>(51)</sup> property is used. Otherwise, the function uses MainRVAControlPanel<sup>(388)</sup>.Language.

For *clNone*, it returns 'Transparent' (translated to the current language).

For *clWindow*, *clBtnFace*, *clWindowText*, it returns 'Auto' (translated to the current language).

For 40 standard colors, it returns their localized names.

For other colors, it returns 'RGB(NN,NN,NN)' string.

**See also:**

- Localization of RichViewActions<sup>(340)</sup>.

#### 1.4.2.1.3 RVA\_LocalizeForm procedure

Applies the current language to all RichViewActions on **Form**.

**Unit** RichViewActions.

```

procedure RVA_LocalizeForm(Form: TComponent);

```

**Form** is a form or datamodule. This procedure changes Captions and Hints of all actions (inherited from TrvCustomAction<sup>(333)</sup>) owned by this form/datamodule.

The name of the current language is returned as `Action.GetControlPanel(336).Language(51)`.

The current language can be changed by assigning a new value to `TRVAControlPanel.Language(51)`, or by calling `RVA_SwitchLanguage(347)` procedure.

**See also:**

- `RVALocalize unit(341)`.

### 1.4.3 Other units

#### Procedures and functions declared in other units

`GetAddictSpellLanguage`, `GetAddictThesLanguage(351)`

`RVALocalizeRuler procedure(351)`

`RVGetHelpKeyword` and `RVSetHelpKeyword(352)`

#### 1.4.3.1 GetAddictSpellLanguage, GetAddictThesLanguage functions

The functions convert a language of RichViewActions to languages used by Addict 3<sup>(362)</sup> components.

**Unit** `RVAddictLocalize`.

**function** `GetAddictSpellLanguage(const s: String): TSpellLanguageType;`

**function** `GetAddictThesLanguage(const s: String): TThesaurusLanguageType;`

**Example:**

```
RVAddictSpell31.UILanguage :=
  GetAddictSpellLanguage(RVA_GetLanguageName(346));
RVThesaurus31.UILanguage :=
  GetAddictThesLanguage(RVA_GetLanguageName(346));
```

It's not necessary to use these functions: `TrvActionSpellingCheck(310)` and `TrvActionThesaurus(310)` displays dialogs using the proper UI language.

**Note:** `RVAddictLocalize` unit is included in `Addict3_RichView*` packages. These packages are installed only if Addict 3/4 is already installed.

#### 1.4.3.2 RVALocalizeRuler procedure

Localizes a `TRVRuler(80)` component.

**Unit** `RVALocRuler`.

**procedure** `RVALocalizeRuler(Ruler: TRVRuler(80);  
 ControlPanel: TRVAControlPanel(38) = nil);`

The method assigns localized strings to hints displayed by **Ruler**.

If **ControlPanel** parameter is defined, its `Language(51)` property is used. Otherwise, the function uses `MainRVAControlPanel(388).Language`.

**See also:**

- Localization of RichViewActions <sup>(340)</sup>
- RVALocalize unit <sup>(341)</sup>

### 1.4.3.3 RVGetHelpKeyword and RVSetHelpKeyword

**Unit** RVHelp.

These functions may be useful if you provide a help file and use HelpKeywords instead of HelpContexts.

```
function RVGetHelpKeyword(HelpContext: Integer): String;  
procedure RVSetHelpKeyword(HelpContext: Integer;  
    const HelpKeyword String);
```

HelpKeyword properties are used if the active TRVAControlPanel <sup>(38)</sup>'s HelpType <sup>(47)</sup> = *htKeyword*.

The list of HelpContext and HelpKeyword properties used by RichViewActions is provided in the topic about HelpType <sup>(47)</sup>.

By default, HelpContexts are used. You can switch to HelpKeywords. Default values of HelpKeywords are in English. Since keywords are visible to the user (in case of CHM help, they are in the "Index" page of the CHM viewer), it makes sense to translate them, if your help file is not in English. You can change a HelpKeyword corresponding to the given HelpContext using **RVSetHelpKeyword**. This change happens for all UI languages, so, if you provide multiple help files for different languages, you need to change keywords after a UI language change (or use language-independent HelpContexts).

**See also properties of TRVAControlPanel <sup>(38)</sup>:**

- UseHelpFiles <sup>(56)</sup>
- UseDefaultHelpFile <sup>(56)</sup>

## 1.5 Interfaces for third-party components

RichViewActions interface components is a group of components allowing RichViewActions to use third-party components.

A third-party component is assigned to a property of an interface component, and this interface component is assigned to a property of TRVAControlPanel <sup>(38)</sup>.

This indirect linking allows using RichViewActions and third-party components together, without losing a possibility using them separately, because:

- RichViewActions units do not refer to units of third-party components
- units of third-party components do not refer to RichViewActions units
- only units of interface components refer to the both of them.

RichViewActions support the following types of interface components:



- downloaders <sup>(353)</sup>
- spelling checkers <sup>(355)</sup>
- color dialogs <sup>(360)</sup>

### 1.5.1 Download interfaces

Download interfaces allow using third-party components to download external images and CSS files referred from RTF, DocX and HTML files.

To use a download interface component, assign it to DownloadInterface<sup>(46)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

The following download interface components are provided:

Component	Requires
 TRVAIndyDownloadInterface <sup>(354)</sup>	Indy: included in Delphi or <a href="http://www.indyproject.org">http://www.indyproject.org</a>
 TRVACIDownloadInterface <sup>(353)</sup>	Clever Internet Suite: <a href="http://www.clevercomponents.com">http://www.clevercomponents.com</a>

#### See also

- RichViewActions interfaces for third-party components<sup>(352)</sup>

#### 1.5.1.1 TRVACIDownloadInterface

Allows using TClHttp component (Clever Internet Suite) to download external images and CSS files when loading files.

**Unit** RVACIDownloadInterface;

```
TRVACIDownloadInterface = class (TRVACustomDownloadInterface(354));
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*  
*TRVACustomDownloadInterface*<sup>(354)</sup>

#### Description

To use this component, assign it to DownloadInterface<sup>(46)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

TRVACIDownloadInterface has property: **ClHttp**. Assign TClHttp component to this property, and actions will use it to download images and CSS files. If this property is not assigned, a temporary TClHttp component is created each time when an action needs to download an image or a CSS file.

The TRichView installer installs this component automatically if Clever Internet Suite is already installed.

Clever Internet Suite can be downloaded from <http://www.clevercomponents.com>.

#### See also:

- RichViewActions download interfaces<sup>(353)</sup>

### 1.5.1.2 TRVACustomDownloadInterface

A base class for RichViewActions download interface components.

**Unit** RVADownloadInterface;

```
TRVACustomDownloadInterface = class (TComponent);
```

#### Hierarchy

*TObject*

*TPersistent*

*TComponent*

#### Description

---

This component is not used directly. Instead, use one of the following components inherited from it:

- TRVAIndyDownloadInterface <sup>(354)</sup>
- TRVACIDownloadInterface <sup>(353)</sup>

To use a download interface component, assign it to DownloadInterface <sup>(46)</sup> property of TRVAControlPanel <sup>(38)</sup> component.

#### See also:

- RichViewActions download interfaces <sup>(353)</sup>

### 1.5.1.3 TRVAIndyDownloadInterface

Allows using TIdHttp component (Indy) to download external images and CSS files when loading files.

**Unit** RVAIndyDownloadInterface;

```
TRVAIndyDownloadInterface = class (TRVACustomDownloadInterface (354));
```

#### Hierarchy

*TObject*

*TPersistent*

*TComponent*

*TRVACustomDownloadInterface* <sup>(354)</sup>

#### Description

---

To use this component, assign it to DownloadInterface <sup>(46)</sup> property of TRVAControlPanel <sup>(38)</sup> component.

TRVAIndyDownloadInterface has property: **IdHttp**. Assign TIdHttp component to this property, and actions will use it to download images and CSS files. If this property is not assigned, a temporary TIdHttp component is created each time when an action needs to download an image or a CSS file.

The TRichView installer installs this component automatically if Indy is already installed.

Indy components are included in Delphi, or can be downloaded from <http://www.indyproject.org>.

#### See also:

- RichViewActions download interfaces <sup>(353)</sup>

## 1.5.2 Spelling check interfaces

Spelling check interfaces allow using third-party components to correct spelling errors in TRichViewEdit.



To use a spelling check interface component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Then you can:

- use TrvActionSpellingCheck<sup>(310)</sup> action
- use TrvActionThesaurus<sup>(310)</sup> action (if the chosen interface component supports a thesaurus)
- call its AutoCorrectInKeyDown in the editor's OnKeyDown event and AutoCorrectInKeyPress in the editor's OnKeyPress event to perform auto-correction on typing (if the chosen interface component supports an auto-correction)
- assign the inverted result of its IsWordValid method to the Misspelled parameter of the editor's OnSpellingCheck event to perform a live spelling check.

### Components

All spelling check interface components implement both live spelling check (spelling check in a background thread) and a traditional spelling check using dialogs

Component	Requires	Thesaurus	Auto-correcti on	Dialog Localizati on	Dialog Type	Win32/Wi n64	Parser is implemen ted by
 TRVSpellInterface <sup>(356)</sup>	—	—	Yes	Yes	Classic/ Word	Win32+Win 64	TRichView
 TRVAAddictSpellInt erface <sup>(356)</sup>	Addict 3 or 4 <sup>(362)</sup>	Yes	Yes	Yes <sup>1</sup>	Classic/ Word	Win32+Win 64	Spell checker
 TRVAASpellInterfac e <sup>(357)</sup>	ASpell <sup>(363)</sup>	—	—	Yes	Classic/ Word	Win32	TRichView
 TRVADXSpellInterfa ce <sup>(358)</sup>	ExpressSpellC hecker <sup>(364)</sup>	—	Yes	Yes <sup>2</sup>	Classic/ Word	Win32+Win 64	TRichView
 TRVAHunSpellInterf ace <sup>(359)</sup>	HunSpell <sup>(365)</sup>	—	—	Yes	Classic/ Word	Win32+Win 64	TRichView
 TRVAPolarSpellInte rface <sup>(359)</sup>	Polar SpellChecker <sup>(366)</sup>	—	Yes	Yes <sup>3</sup>	Classic	Win32	TRichView

Localization:

- 1: a spelling form is localized by the spelling checker, RichViewActions select the most appropriate language for TRVAControlPanel<sup>(38)</sup>.Language<sup>(51)</sup> automatically.
- 2: a spelling form is localized by the spelling checker.
- 3: a spelling form can be localized by the spelling checker, however, ready-to-use localized text is not supplied.

**See also**

- RichViewActions interfaces for third-party components<sup>(352)</sup>

### 1.5.2.1 TRVSpellInterface

Allows using TRVSpellChecker (included in TRichView components) to correct spelling errors.

**Unit** RVAAddictSpellInterface;

```
TRVSpellChecker = class (TRVACustomSpellInterface(357));
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TRVACustomSpellInterface*<sup>(357)</sup>

**Description**

To use this component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

**Properties:**

- **SpellChecker** – a link to TRVSpellChecker component. Assignment to this property is required, otherwise a spelling check will not work.

**See also:**

- RichViewActions spelling check interfaces<sup>(355)</sup>

### 1.5.2.2 TRVAAddictSpellInterface

Allows using Addict (by Addictive Software)<sup>(362)</sup> to correct spelling errors and to find synonyms.

**Unit** RVAAddictSpellInterface;

```
TRVAAddictSpellInterface = class (TRVACustomSpellInterface(357));
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TRVACustomSpellInterface*<sup>(357)</sup>

**Description**

To use this component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.



**Properties:**

- **AddictSpell** – a link to TRVAddictSpell3 component. Assignment to this property is required, otherwise a spelling check will not work.
- **Thesaurus** – a link to TRVThesaurus3 component. Assignment to this property is required, otherwise TrvActionThesaurus<sup>(310)</sup> will be disabled.

The TRichView installer installs this component automatically if Addict is already installed, and "build control" option is changed<sup>(362)</sup> for Addict packages.

Addict can be downloaded from <http://www.addictivesoftware.com>.

**See also:**

- RichViewActions spelling check interfaces<sup>(355)</sup>

**1.5.2.3 TRVAASpellInterface**

Allows using ASpell<sup>(363)</sup> to correct spelling errors.

**Unit** RVAASpellInterface;

```
TRVAASpellInterface = class (TRVACustomSpellInterface(357)) ;
```

**Hierarchy**

*TObject*  
*TPersistent*  
*TComponent*  
*TRVACustomSpellInterface*<sup>(357)</sup>

**Description**

To use this component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

**Properties:**

- **ASpell** – a link to TRVASpell component. Assignment to this property is required, otherwise spelling check will not work.

ASpell support does not require additional components, all necessary components are installed by the TRichView installer.

However, the components works only if ASpell is installed. The installer can be found on [aspell.net/win32](http://aspell.net/win32).

**See also:**

- RichViewActions spelling check interfaces<sup>(355)</sup>

**1.5.2.4 TRVACustomSpellInterface**

A base class for RichViewActions spelling check interface components.

**Unit** RVADownloadInterface;

```
TRVACustomDownloadInterface = class (TComponent) ;
```

**Hierarchy**

*TObject*  
*TPersistent*

*TComponent*

## Description

This component is not used directly. Instead, use one of the following components inherited from it:

- TRVAAddictSpellInterface<sup>(356)</sup> (to use Addict<sup>(362)</sup>)
- TRVAASpellInterface<sup>(357)</sup> (to use ASpell<sup>(363)</sup>)
- TRVADXSpellInterface<sup>(358)</sup> (to use ExpressSpellChecker<sup>(364)</sup>)
- TRVAHunSpellInterface<sup>(359)</sup> (to use HunSpell<sup>(365)</sup>)
- TRVAPolarSpellInterface<sup>(359)</sup> (to use Polar SpellChecker<sup>(366)</sup>)

To use a spelling check interface component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Then you can:

- use TrvActionSpellingCheck<sup>(310)</sup> action
- use TrvActionThesaurus<sup>(310)</sup> action (if the chosen interface component supports a thesaurus)
- call its AutoCorrectInKeyDown in the editor's OnKeyDown event and AutoCorrectInKeyPress in the editor's OnKeyPress (in OnUTF8KeyPress for Lazarus) event to perform auto-correction on typing (if the chosen interface component supports an auto-correction)
- assign the inverted result of its IsWordValid method to the Misspelled parameter of the editor's OnSpellingCheck event to perform a live spelling check.

## Methods [VCL]

```
function AutoCorrectInKeyDown(Editor: TCustomRichViewEdit;
  Key: Word): Boolean;
function AutoCorrectInKeyPress(Editor: TCustomRichViewEdit;
  Key: Char): Boolean;
function IsWordValid(const S: TRVUnicodeString;
  rv: TCustomRichView; StyleNo: Integer): Boolean;
virtual; abstract;
```

## Methods [FPC]

For Lazarus, the methods are the same, with the small difference:

```
function AutoCorrectInKeyPress(Editor: TCustomRichViewEdit;
  Key: TUTF8Char): Boolean;
```

## See also:

- RichViewActions spelling check interfaces<sup>(355)</sup>

### 1.5.2.5 TRVADXSpellInterface

Allows using ExpressSpellChecker<sup>(364)</sup> to correct spelling errors.

**Unit** RVADXSpellInterface;

```
TRVADXSpellInterface = class (TRVACustomSpellInterface(357));
```

## Hierarchy

*TObject*

*TPersistent*

*TComponent*

*TRVACustomSpellInterface* <sup>(357)</sup>

## Description

To use this component, assign it to SpellInterface <sup>(54)</sup> property of TRVAControlPanel <sup>(38)</sup> component.

### Properties:

- **SpellChecker** – a link to TRvDxSpellChecker component. Assignment to this property is required, otherwise a spelling check will not work.

### See also:

- RichViewActions spelling check interfaces <sup>(355)</sup>

### 1.5.2.6 TRVAHunSpellInterface

Allows using HunSpell <sup>(365)</sup> to correct spelling errors.

**Unit** RVAASpellInterface;

```
TRVAHunSpellInterface = class (TRVACustomSpellInterface (357));
```

### Hierarchy

*TObject*

*TPersistent*

*TComponent*

*TRVACustomSpellInterface* <sup>(357)</sup>

## Description

To use this component, assign it to SpellInterface <sup>(54)</sup> property of TRVAControlPanel <sup>(38)</sup> component.

### Properties:

- **HunSpell** – a link to TRVHunSpell component. Assignment to this property is required, otherwise a spelling check will not work.

HunSpell support does not require additional components, all necessary components are installed by the TRichView installer.

However, the components works only if HunSpell DLL and dictionaries are available for the application.

### See also:

- RichViewActions spelling check interfaces <sup>(355)</sup>

### 1.5.2.7 TRVAPolarSpellInterface

Allows using Polar SpellChecker <sup>(366)</sup> to correct spelling errors.

**Unit** RVAPolarSpellInterface;

```
TRVAPolarSpellInterface = class (TRVACustomSpellInterface (357));
```

### Hierarchy

*TObject*

*TPersistent*

*TComponent*

*TRVACustomSpellInterface* <sup>(357)</sup>

## Description

To use this component, assign it to *SpellInterface* <sup>(54)</sup> property of *TRVAControlPanel* <sup>(38)</sup> component.

### Properties:

- **PolarSpellChecker** – a link to *TRVPolarSpellChecker* component. Assignment to this property is required, otherwise spelling check will not work.

### See also:

- RichViewActions spelling check interfaces <sup>(355)</sup>


## 1.5.3 Color dialog interfaces

By default, RichViewActions use the standard *TColorDialog*. You can assign a specific *TColorDialog* component to *ColorDialog* <sup>(42)</sup> property of *TRVAControlPanel* <sup>(38)</sup>, and all RichViewActions will use this component.

Alternatively, you can use third-party color dialogs via a color interface component. All color dialog interface components are inherited *TRVCustomColorDialogInterface*.

To use a color dialog interface component, assign it to *ColorDialogInterface* <sup>(42)</sup> property of *TRVAControlPanel* <sup>(38)</sup> component.

The following download interface components are provided:

Component	Requires	Dialog Component
 <i>TRVADXColorDialogInterface</i>	DevExpress VCL Components	<i>TdxColorDialog</i>

### See also

- RichViewActions interfaces for third-party components <sup>(352)</sup>

## 1.6 Third-party and additional tools

### Components

The following additional components can be optionally used in RichViewActions.

Spell checkers:

- Addict 3 or 4 <sup>(362)</sup>
- ASpell <sup>(363)</sup>
- ExpressSpellChecker <sup>(364)</sup>
- HunSpell <sup>(365)</sup>
- Polar Spell Checker ActiveX

Downloaders:

- CleverComponents <sup>(367)</sup>

- Indy<sup>(368)</sup>

File readers and writers:

- RichViewXML<sup>(368)</sup> for opening and saving documents in XML format (freeware)

User interface:

- Toolbar 2000<sup>(371)</sup> (free for non-commercial use), TBX<sup>(371)</sup>, SpTBXLib<sup>(370)</sup> for providing alternative versions of TRVAPopupMenu<sup>(66)</sup>
- TNT Controls<sup>(370)</sup> for Unicode support in Delphi 5-2007.

## Toolbar Images

---

By default, RichViewActions use simple 16-color 16x16 images.

In Delphi 2009+ (or if you use a thirdparty image list component supporting semitransparent images), you can use TRichView icons<sup>(372)</sup>, or the following alternative toolbar images:

- GlyFX<sup>(374)</sup>
- Glyfz<sup>(375)</sup>
- Fugue Icons<sup>(375)</sup>
- FamFamFam Silk Icons<sup>(376)</sup>

### 1.6.1 Components

#### Components

---

The following additional components can be optionally used in RichViewActions.

Spell checkers:

- TRVSpellChecker (included in TRichView)
- Addict 3 or 4<sup>(362)</sup>
- ASpell<sup>(363)</sup>
- ExpressSpellChecker<sup>(364)</sup>
- HunSpell<sup>(365)</sup>
- Polar Spell Checker ActiveX

Downloaders:

- CleverComponents<sup>(367)</sup>
- Indy<sup>(368)</sup>

File readers and writers:

- RichViewXML<sup>(368)</sup> for opening and saving documents in XML format (freeware)

User interface:

- Toolbar 2000<sup>(371)</sup> (free for non-commercial use), TBX<sup>(371)</sup>, SpTBXLib<sup>(370)</sup> for providing alternative versions of TRVAPopupMenu<sup>(66)</sup>
- TNT Controls<sup>(370)</sup> for Unicode support in Delphi 5-2007.

### 1.6.1.1 Spelling checkers

#### Spelling Checking Components

---

The following additional components can be optionally used in RichViewActions:

- TRVSpellChecker (included in TRichView components)
- Addict 3 or 4 <sup>(362)</sup>
- ASpell <sup>(363)</sup>
- ExpressSpellChecker <sup>(364)</sup>
- HunSpell <sup>(365)</sup>
- Polar Spell Checker ActiveX

##### 1.6.1.1.1 Addict 3 and 4

Addict 3 and 4 is a spelling check and thesaurus component suite. Visit [www.addictive-software.com](http://www.addictive-software.com) for additional information.

(discontinued)

#### Installing

---

1. Addict 4 packages must be installed before installing TRichView.
2. Addict 4 packages must be changed before installing this package. Open Addict 4 packages, open the package dialog. Go to the page "Description". Change "Build control" = "Explicit rebuild". Recompile the packages. Make sure that you do not have duplicated BPL and DCP files for the Addict packages in different directories (they can be in <AddictDir>\BPL and in Delphi projects directories)
3. If Addict 4 is installed, the TRichView installer installs the component integrating Addict in TRichView and RichViewActions automatically.

If you installed Addict 4 after TRichView, you can run the TRichView installer again using the shortcut in the Windows Start menu.

#### Using

---

Create TRVAddictSpellInterface <sup>(356)</sup> component, assign it to SpellInterface <sup>(54)</sup> property of TRVAControlPanel <sup>(38)</sup> component.

Create TRVAddictSpell3 component, assign it to AddictSpell property of this TRVAddictSpellInterface <sup>(356)</sup> component.

Create TRVThesaurus3 component, assign it to Thesaurus property of this TRVAddictSpellInterface <sup>(356)</sup> component.

Assign TrvActionSpellingCheck <sup>(310)</sup> and TrvActionThesaurus <sup>(310)</sup> to menu items or toolbar buttons.

If you need an auto-correction feature, see the information in the topic for TRVACustomSpellInterface <sup>(357)</sup>.

#### Live spelling check

---

To enable a live spelling check, assign OnSpellingCheck event of TCustomRichViewEdit component:

```
procedure TForm1.RichViewEdit1SpellingCheck(Sender: TCustomRichView;
  const AWord: String; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not RVAAddictSpellInterface1.IsWordValid(AWord,
    RichViewEdit1, StyleNo);
end;
```

If you want to start a live spelling check when the user starts editing, assign *rvlspOnChange* to LiveSpellingMode property of TCustomRichViewEdit component.

If you want to start a live spelling check when the user loaded a document, use OnOpenFile<sup>(110)</sup> event of TrvActionOpen<sup>(106)</sup> action:

```
procedure TForm1.RvActionOpen1OpenFile(Sender: TObject;
  Editor: TCustomRichViewEdit; const FileName: String;
  FileFormat: TrvFileOpenFilter; CustomFilterIndex: Integer);
begin
  Editor.StartLiveSpelling;
end;
```

Assign TRVAPopupMenu<sup>(66)</sup> component to RichViewEdit1.PopupMenu.

#### 1.6.1.1.2 ASpell

GNU Aspell is a Free and Open Source spell checker.

Addict 3 and 4 is a spelling check and thesaurus component suite. Visit [aspell.net](http://aspell.net) for additional information.

### Installing

ASpell support does not require additional components, all necessary components are installed by the TRichView installer.

However, the components works only if ASpell is installed. The installer can be found on [aspell.net/win32](http://aspell.net/win32).

### Using

Create TRVAASpellInterface<sup>(357)</sup> component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Create TRVASpell component, assign it to ASpell property of this TRVAASpellInterface<sup>(357)</sup> component.

Assign TrvActionSpellingCheck<sup>(310)</sup> to a menu item or a toolbar button.

### Live spelling check

To enable a live spelling check, assign OnSpellingCheck event of TCustomRichViewEdit component:

```
procedure TForm1.RichViewEdit1SpellingCheck(Sender: TCustomRichView;
  const AWord: String; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not RVAASpellInterface1.IsWordValid(AWord,
    RichViewEdit1, StyleNo);
end;
```

If you want to start a live spelling check when the user starts editing, assign *rv/spOnChange* to LiveSpellingMode property of TCustomRichViewEdit component.

If you want to start a live spelling check when the user loaded a document, use OnOpenFile<sup>(110)</sup> event of TrvActionOpen<sup>(106)</sup> action:

```
procedure TForm1.RvActionOpen1OpenFile(Sender: TObject;
  Editor: TCustomRichViewEdit; const FileName: String;
  FileFormat: TrvFileOpenFilter; CustomFilterIndex: Integer);
begin
  Editor.StartLiveSpelling;
end;
```

Assign TRVAPopupMenu<sup>(66)</sup> component to RichViewEdit1.PopupMenu.

### 1.6.1.1.3 ExpressSpellChecker

ExpressSpellChecker is a VCL spell checker component.

Visit <https://www.devexpress.com> for additional information.

## Installing

If DevExpress VCL components are installed, the TRichView installer installs the component integrating ExpressSpellChecker in TRichView and RichViewActions automatically.

## Using

Create TRVADXSpellInterface<sup>(358)</sup> component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Create TRvDxSpellChecker component, assign it to SpellChecker property of this TRVADXSpellInterface<sup>(358)</sup> component. Add the code to initialize the spell checker.

Assign TrvActionSpellingCheck<sup>(310)</sup> to a menu item or a toolbar button.

If you need an auto-correction feature, see the information in the topic for TRVACustomSpellInterface<sup>(357)</sup>.

## Live spelling check in TcxTRichViewEdit

In applications using DxSpellChecker, it is assumed that live spelling is performed only in the focused component.

If you use TcxTRichViewEdit component, it will be checked automatically when focused, if RvDxSpellChecker.CheckAsYouTypeOptions.Active = True.

You can assign TRVAPopupMenu<sup>(66)</sup> component to RichViewEdit1.PopupMenu, however, TcxTRichViewEdit can display a popup menu with live spelling options even without it.

## Live spelling check in TRichViewEdit (and TSRichViewEdit)

In applications using DxSpellChecker, it is assumed that live spelling is performed only in the focused component.

If you use TRichViewEdit component:



TRichViewEdit, process OnEnter and OnExit:

**OnEnter:**

```
RvDxSpellChecker1.CheckAsYouTypeOptions.Active := False;
RvDxSpellChecker1.UpdateRules;
RichViewEdit1.StartLiveSpelling;
```

**OnExit:**

```
RichViewEdit1.ClearLiveSpellingResults;
RvDxSpellChecker1.CheckAsYouTypeOptions.Active := True;
```

Assign OnSpellingCheck event:

```
procedure TForm1.RichViewEdit1SpellingCheck(Sender: TCustomRichView;
  const AWord: String; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not RVADxSpellInterface1.IsWordValid(AWord,
    RichViewEdit1, StyleNo);
end;
```

#### 1.6.1.1.4 HunSpell

HunSpell is a spell checker.

Hunspell is free software, distributed under the terms of a GPL, LGPL and MPL tri-license.

Visit <http://hunspell.github.io/> for additional information.

### Installing

HunSpell support does not require additional components, all necessary components are installed by the TRichView installer.

However, the components works only if HunSpell DLLs and dictionaries are available.

For example, you can download DLLs from our web site:

- **32 bit**
- **64 bit**

Dictionaries (\*.aff and \*.dic files) can be downloaded, for example, here:

<https://cgkit.freedesktop.org/libreoffice/dictionaries/tree/>.

### Using

Create TRVAHunSpellInterface<sup>(359)</sup> component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Create TRVHunSpell component, assign it to HunSpell property of this TRVAHunSpellInterface<sup>(359)</sup> component. Assign its DictDir, DIIDir, DIIDName properties, if necessary. Load dictionaries (for example, by calling LoadAllDictionaries).

Assign TrvActionSpellingCheck<sup>(310)</sup> to a menu item or a toolbar button.

### Live spelling check

To enable a live spelling check, assign OnSpellingCheck event of TCustomRichViewEdit component:

```
procedure TForm1.RichViewEdit1SpellingCheck(Sender: TCustomRichView;
  const AWord: String; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not RVAHunSpellInterface1.IsWordValid(AWord,
    RichViewEdit1, StyleNo);
end;
```

If you want to start a live spelling check when the user starts editing, assign *rv/spOnChange* to LiveSpellingMode property of TCustomRichViewEdit component.

If you want to start a live spelling check when the user loaded a document, use OnOpenFile<sup>(110)</sup> event of TrvActionOpen<sup>(106)</sup> action:

```
procedure TForm1.RvActionOpen1OpenFile(Sender: TObject;
  Editor: TCustomRichViewEdit; const FileName: String;
  FileFormat: TrvFileOpenFilter; CustomFilterIndex: Integer);
begin
  Editor.StartLiveSpelling;
end;
```

Assign TRVAPopupMenu<sup>(66)</sup> component to RichViewEdit1.PopupMenu.

#### 1.6.1.1.5 Polar SpellChecker ActiveX

Polar SpellChecker component is a spell checker. It may be used as DLL or as ActiveX. TRichView uses it as ActiveX.

Last time Polar SpellChecker was updated in 2005.

Visit <http://www.polarsoftware.com/> for additional information.

### Installing

1. Install the Polar SpellChecker Components.
2. Import it in Delphi. Menu "Component | Import Component" (or "Component | Import ActiveX Control", depending on your Delphi version) import and install "Polar SpellChecker 5.0 Component", The component must be installed in the package PolarSpellCheckerPkg\*, where \* depends on Delphi version:
  - Delphi 5..7: D5..D7
  - Delphi 2005..2010: D2005..D2010
  - Delphi XE..XE8: DXE..DXE8
  - Delphi 10 Seattle: D10
  - Delphi 10.1 Berlin..10.3 Rio: D10\_1..D10\_3
3. TRichView includes packages for installing support for Polar SpellChecker (in ThirdParty\Polar\ folder), but the installer does not install them automatically. Open the package RVPolarPkg\* and compile. Open the package RVPolarPkg\*\_Dsgn and install.

### Using

Create TRVAPolarSpellInterface<sup>(359)</sup> component, assign it to SpellInterface<sup>(54)</sup> property of TRVAControlPanel<sup>(38)</sup> component.

Create TRVPolarSpellChecker component, assign it to PolarSpellChecker property of this TRVAPolarSpellInterface<sup>(359)</sup> component.

Assign TrvActionSpellingCheck<sup>(310)</sup> to a menu item or a toolbar button.

If you need an auto-correction feature, see the information in the topic for TRVACustomSpellInterface<sup>(357)</sup>.

## Live spelling check

To enable a live spelling check, assign OnSpellingCheck event of TCustomRichViewEdit component:

```
procedure TForm1.RichViewEdit1SpellingCheck(Sender: TCustomRichViewEdit;
  const AWord: String; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not RVAPolarSpellInterface1.IsWordValid(AWord,
    RichViewEdit1, StyleNo);
end;
```

If you want to start a live spelling check when the user starts editing, assign *rvlspOnChange* to LiveSpellingMode property of TCustomRichViewEdit component.

If you want to start a live spelling check when the user loaded a document, use OnOpenFile<sup>(110)</sup> event of TrvActionOpen<sup>(106)</sup> action:

```
procedure TForm1.RvActionOpen1OpenFile(Sender: TObject;
  Editor: TCustomRichViewEdit; const FileName: String;
  FileFormat: TrvFileOpenFilter; CustomFilterIndex: Integer);
begin
  Editor.StartLiveSpelling;
end;
```

Assign TRVAPopupMenu<sup>(66)</sup> component to RichViewEdit1.PopupMenu.

### 1.6.1.2 Downloaders

#### Downloading Components

The following additional components can be optionally used in RichViewActions:

- CleverComponents<sup>(367)</sup>
- Indy<sup>(368)</sup>

They allow downloading external images referred from imported RTF and HTML files.

##### 1.6.1.2.1 CleverComponents

HTML, Markdown, DocX and RTF files can contain links to images located on a web server. If you want to use them, it's required to download them inside OnImportPicture event of TCustomRichViewEdit component.

RichViewActions can do it for you using CleverComponents (TCIHTTP) component.

Visit <http://www.clevercomponents.com> for additional information.

CleverComponents can be used via TRVACIDownloadInterface<sup>(353)</sup> component.

#### 1.6.1.2.2 Indy

HTML, Markdown, DocX, and RTF files can contain links to images located on web server. If you want to use them, it's required to download them inside OnImportPicture event of TCustomRichViewEdit component.

RichViewActions can do it for you using Indy (TIdHTTP) component.

Indy is included in new versions of Delphi, or it can be [downloaded](#).

Indy can be used via TRVAIndyDownloadInterface<sup>(354)</sup> component.

### 1.6.1.3 File readers and writers

#### File Components

---

The following additional components can be optionally used in RichViewActions:

File readers and writers:

- RichViewXML<sup>(368)</sup> for opening and saving documents in XML format (freeware)

#### 1.6.1.3.1 RichViewXML

RichViewXML provides an alternative to RVF way for lossless storing of RichView documents.

If activated, RichViewActions support RichView XML files in the following actions:

- TrvActionOpen<sup>(106)</sup>
- TrvActionSaveAs<sup>(127)</sup>
- TrvActionExport<sup>(110)</sup>
- TrvActionInsertFile<sup>(228)</sup>

RichViewXML is installed by TRichView Setup.

#### How to use

---

Place TRVAControlPanel<sup>(38)</sup> component on a form. If you have more than one TRVAControlPanel component in your application, link<sup>(335)</sup> it to actions.

Assign TRichViewXML component to TRVAControlPanel.XMLComponent<sup>(58)</sup>.

Make sure that *ffiXML* is in Filter of TrvActionOpen<sup>(106)</sup>, TrvActionInsertFile<sup>(228)</sup>, *ffeXML* in Filter of TrvActionSaveAs<sup>(127)</sup>, TrvActionExport<sup>(110)</sup>.

You can change a file filter for XML files: see TRVAControlPanel.XMLFilter<sup>(58)</sup> property.

#### 1.6.1.3.2 RvHtmlImporter (deprecated)

Deprecated. TRichView HTML methods for HTML import provide better result.

---

RVHTMLImporter allows importing HTML files in TCustomRichView components.

RichViewAction support RVHTMLImporter in the following actions:

- TrvActionInsertFile<sup>(228)</sup> (RVHTMLImporter replaces the MS Office HTML import converter in the file insertion dialog)

- TrvActionPasteSpecial<sup>(137)</sup>
- TrvActionPaste<sup>(135)</sup> (pasted if not RVF or non-empty RTF available in the Clipboard)

RVHTMLImporter is installed by TRichView Setup

## How to use

Place TRVAControlPanel<sup>(38)</sup> component on a form. If you have more than one TRVAControlPanel component in your application, link<sup>(335)</sup> it to actions.

Assign TRvHtmlImporter component to TRVAControlPanel.HTMLComponent<sup>(51)</sup>.

Make sure that *ffiHTML* is in Filter<sup>(230)</sup> property of TrvActionInsertFile<sup>(228)</sup> action.

Include *rvddHTML* in AcceptPasteFormats property of the editor.

### See also:

- PasteHTML<sup>(378)</sup> procedure
- RvHtmlViewImporter<sup>(369)</sup>

### 1.6.1.3.3 RvHtmlViewImporter (deprecated)

Deprecated. TRichView HTML methods for HTML import provide better result.

RVHTMLViewImporter allows loading HTML files in RichView using HTML parser from THTMLViewer (freeware with source).

If activated, RichViewActions support HTML files in the following actions:

- TrvActionOpen<sup>(106)</sup>
- TrvActionSaveAs<sup>(127)</sup> (RVHTMLViewImporter is not used for saving, but HTML is added in the "Save As" action because it is added in the "Open" action)
- TrvActionInsertFile<sup>(228)</sup>
- TrvActionPasteSpecial<sup>(137)</sup>
- TrvActionPaste<sup>(135)</sup> (pasted if not RVF or non-empty RTF available in the Clipboard)

RVHTMLViewImporter is installed by TRichView Setup if THTMLViewer is already installed.

THTMLViewer can be downloaded from [github.com/BerndGabriel/HtmlViewer](https://github.com/BerndGabriel/HtmlViewer).

## How to use

Place TRVAControlPanel<sup>(38)</sup> component on a form. If you have more than one TRVAControlPanel component in your application, link<sup>(335)</sup> it to actions.

Assign TRvHtmlViewImporter component to TRVAControlPanel.HTMLComponent<sup>(51)</sup>.

Assign THTMLViewer component to HTMLViewer property of TRVHTMLViewImporter (make THTMLViewer hidden).

Make sure that *ffiHTML* is in Filter<sup>(230)</sup> property of TrvActionInsertFile<sup>(228)</sup> action.

Include *rvddHTML* in AcceptPasteFormats property of the editor.

### See also:

- PasteHTML<sup>(378)</sup> procedure

- RvHtmlImporter<sup>(368)</sup>

### 1.6.1.4 User interface

#### UI Components

---

The following additional components can be optionally used in RichViewActions:

- Toolbar 2000<sup>(371)</sup> (free for non-commercial use), TBX<sup>(371)</sup>, SpTBXLib<sup>(370)</sup> for providing alternative versions of TRVAPopupMenu<sup>(66)</sup>
- TNT Controls<sup>(370)</sup> for Unicode support in Delphi 5-2007.

##### 1.6.1.4.1 SpTBXLib

SpTBXLib is an expansion package for TB2K<sup>(371)</sup> components that adds Unicode support and skinning. You can download it from [www.silverpointdevelopment.com/sptbplib/](http://www.silverpointdevelopment.com/sptbplib/).

If activated, TRVASPTBXPopupMenu<sup>(66)</sup> component is available.

#### Installing

---

1. Install the SpTBXLib package.
2. Open **RichViewActions.inc**, remove the dot from the line `{$DEFINE USESPTBX}`.
3. Open the RichViewActions package. Add a link to the SpTBXLib package in "Requires" section. Install the package (or rebuild it, if already installed).

##### 1.6.1.4.2 TNT Controls

TMS Unicode Component Pack controls allow you to develop applications that take advantage of the Unicode capabilities of Windows NT/2000/XP/2003/Vista without abandoning Delphi, C++Builder or Windows 95/98/ME. You can download them from [www.tmssoftware.com](http://www.tmssoftware.com).

Since Delphi/C++Builder 2009 features a built-in support of Unicode, TMS Unicode Component Pack should be used only in older versions of Delphi.

If activated, the most of standard controls on RichViewActions dialogs are replaced with TNT controls. Caption<sup>(334)</sup> and Hints<sup>(335)</sup> properties of RichViewActions are changed to Unicode. TRVAPopupMenu<sup>(66)</sup>, find and replace dialogs support Unicode.

#### Installing

---

1. Open TMS Unicode Component Pack packages and set "Explicit rebuild" option. For Delphi 4-7, open the package, click "Options" in the package window, on the tab "Description" set "Build control"="Explicit rebuild". For newer version of Delphi, open the package, right click in the Project Manager and choose "Options", on the page "Description" set "Build control"="Explicit rebuild".
2. Install TMS Unicode Component Pack.
3. Open **RV\_Defs.inc**, remove the dot from the line `{$DEFINE USERVTNT}`.
4. Recompile TRichView and RichViewActions. You can use "Install in IDE" shortcut in the Start menu.

See ActionTest\_TNT.dpr demo.

#### 1.6.1.4.3 TBX

SpTBXLib is an expansion package for TB2K<sup>(371)</sup> components. You can download it from [github.com/plashenkov/TBX](https://github.com/plashenkov/TBX)

If activated, TRVATBXPopupMenu<sup>(66)</sup> component is available.

#### Installing

---

1. Install the TBX package.
2. Open **RichViewActions.inc**, remove the dot from the line `{$DEFINE USETBX}`.
3. Open the RichViewActions package. Add a link to the TBX package in "Requires" section. Install the package (or rebuild it, if already installed).

#### 1.6.1.4.4 Toolbar2000

Toolbar2000 is a set of components designed to mimic the look and behavior of Office 2000's menus and toolbars. You can download it from [www.jrsoftware.org](http://www.jrsoftware.org).

It has the following extensions:

- TBX<sup>(371)</sup>
- SpTBXLib<sup>(370)</sup>

If activated, TRVATBPopupMenu<sup>(66)</sup> component is available.

#### Installing

---

1. Install the Toolbar2000 package.
2. Open **RichViewActions.inc**, remove the dot from the line `{$DEFINE USETB2K}`.
3. Open the RichViewActions package. Add a link to the Toolbar2000 package in "Requires" section. Install the package (or rebuild it, if already installed).

### 1.6.2 Toolbar Images

#### Toolbar Images

---

By default, RichViewActions use simple 16-color 16x16 images.

In Delphi 2009+ (or if you use a thirdparty image list component supporting semitransparent images), you can use TRichView icons<sup>(372)</sup>, or the following alternative toolbar images:

- GlyFX<sup>(374)</sup>
- Glyfz<sup>(375)</sup>
- Fugue Icons<sup>(375)</sup>
- FamFamFam Silk Icons<sup>(376)</sup>

### 1.6.2.1 TRichView Icons

#### TRichView icons

---

There are two sets of toolbar images created for RichViewActions. They include 16x16 and 32x32 images for all actions, 64x64 images for selected actions, some special images for ScaleRichView.

These images can be used in image lists in Delphi 2009 and newer, because older versions of Delphi do not support semitransparent images in image lists.

These images can be downloaded separately, from TRichView web site. In RichViewActions, they are included in several data modules.

The images are free for registered TRichView users. You can use these images in your applications. Any other use requires a special permission. You cannot sell these images or any graphic work created basing on them.

#### Data modules for Delphi 2009+, standard toolbar

---

Content: 16x16 images, TImageList, normal and disabled images.

- Source\dmActionsImages1.pas - image set #1
- Source\dmActionsImages2.pas - image set #2

These data modules are used in the demos:

- DelphiUnicode\ActionTest\
- CBuilderUnicode\ActionTest\

These data modules do not include icons for ScaleRichView and Report Workshop actions. These components include their own analogs of these data modules.

#### Data modules for Delphi 2009+, TRibbon

---

Content: 16x16 for all actions, 32x32 for selected actions, TImageList, normal and disabled images.

- ActionTestRibbon\RVARibbonImages1.pas - image set #1
- ActionTestRibbon\RVARibbonImages2.pas - image set #2

These data modules are used in the demos:

- DelphiUnicode\ActionTestRibbon\

These data modules do not include icons for ScaleRichView and Report Workshop actions. These components include their own analogs of these data modules.

#### Data modules for Delphi 10.3+

---

Content: 16x16 and 32x32 images for all actions, 64x64 images for selected actions, TImageCollection, normal images. Some special images for ScaleRichView are included as well in two sizes.

- Source\dmActionsImageCollection1.pas - image set #1
- Source\dmActionsImageCollection1.pas - image set #2

These images are used in the following data module:

- Source\dmActionsVirtualImageLists.pas



This data module has two TVirtualImageList components, for normal and disabled images (disabled images are auto-generated). You can switch between image sets by calling SetImageCollection(1) or SetImageCollection(2).

If you use the data module from dmActionsVirtualImageLists.pas in your project, make sure that it is created after the data modules from dmActionsImageCollection1.pas and dmActionsImageCollection2.pas.

These data modules are used in the demo:

- DelphiUnicode\ActionTest\_MultiRes\

The image collections include all images, including images for ScaleRichView and Report Workshop. These components use dmActionsImageCollection1.pas and dmActionsImageCollection2.pas units, but include their own data modules with TVirtualImageList components.

---

## Data modules for C++Builder 10.3+

There are C++ analogs of the data modules described in the previous section:

- SourceCPP\dmActionsImageCollectionCPP1.cpp
- SourceCPP\dmActionsImageCollectionCPP2.cpp
- SourceCPP\dmActionsVirtualImageListsCPP.cpp

These data modules are used in the demo:

- CBuilderUnicode\ActionTest\_MultiRes\

---

## Data modules for Lazarus 2+

Content: 16x16 and 32x32 images for all actions, 64x64 images for selected actions, TImageList, normal and disabled images.

- SourceEx\dmactionsimagesmultireslaz1.pas - image set #1
- SourceEx\dmactionsimagesmultireslaz2.pas - image set #2

These data modules are used in the demo:

- Lazarus\ActionTest\

These data modules do not include icons for ScaleRichView and Report Workshop actions. These components include their own analogs of these data modules.

---

## Data modules for Lazarus

Content: 16x16 images, TImageList, normal and disabled images.

- SourceEx\dmactionsimageslaz1.pas - image set #1
- SourceEx\dmactionsimageslaz2.pas - image set #2

These images are included for older versions of Lazarus.

---

## Compatibility of image lists

Indexes of images in all image lists mentioned above are the same, except for TRibbon version.

---

## Additional information

Additional information is available on the support forum:

[\*image set #1\*](#)

[\*image set #2\*](#)

### 1.6.2.2 GlyFX toolbar images

#### GlyFX Images

---

*(deprecated, this datamodule the not maintained anymore)*

If you want to use modern looking images, consider [GlyFX](#) toolbar images.

There are 4 datamodules available.

Two datamodules based on GlyFX Windows XP icon sets:

- XP Common (free)
- XP Core II
- XP Word processing.

Two datamodules based on GlyFX Windows Vista icon sets:

- Vista Common (free)
- Vista Core II
- Vista Word processing.

The corresponding sets must be purchased in order to use these datamodules. After purchasing, request datamodules from GlyFX.

All registered users of TRichView get a 25% discount when registering all GlyFX products.

#### Datamodules:

- *dmActionsGlyFX.pas* uses XP-style images without alpha channel (based on bmp). Can be used in Delphi 4+
- *dmActionsGlyFXAlpha.pas* uses XP-style images with alpha channel (based on png). Can be used in Delphi 2009+
- *dmActionsGlyFXAero.pas* uses Vista-style images without alpha channel (based on bmp). Can be used in Delphi 4+
- *dmActionsGlyFXAeroAlpha.pas* uses Vista-style images with alpha channel (based on png). Can be used in Delphi 2009+

All datamodules include 6 image lists: normal, highlighted disabled; all in 16x16 and 24x24.

Additionally, projects with GlyFX and TRibbon are available for TRichViewEdit and TSRichViewEdit.

#### Installing

---

It's not necessary to install files to use these datamodules.

However, if you want to use one of these modules at design time instead of dmActions.pas:

1. Open **RichViewActions.inc** and remove the dot from one of the following lines:

```
{.$DEFINE USEGLYFX}      // to use dmActionsGlyFX      (XP-style, 1 bit transparency)
{$DEFINE USEGLYFXALPHA}  // to use dmActionsGlyFXAlpha  (XP-style, 1 byte transparency,
D2009+)
{$DEFINE USEGLYFXAERO}   // to use dmActionsGlyFXAero   (Vista-style, 1 bit transparency)
```

```
{.$DEFINE USEGLYFXAEROALPHA} // to use dmActionsGlyFXAeroAlpha (Vista-style, 1 byte transparency, D2009+)
```

2. Remove dmActions.pas from the RichViewActions package
3. Install (or recompile) the package.

## Additional information

---

[Additional information is available on the support forum.](#)

### 1.6.2.3 Glyfz toolbar images

#### Glyfz Images

---

*(deprecated, this datamodule the not maintained anymore)*

If you want to use toolbar images looking very close to MS Word 2007/2010 images, consider [Glyfz](#) toolbar images.

We have the following demos:

- analog of Ribbon\ActionTestRibbon demo for TRichViewEdit
- analog of ActionTestTabsRibbon demo for ScaleRichView.

These projects can be used only in Delphi 2009+, because they require TRibbon component, and because images are semitransparent.

They use the following sets of Glyfz images:

- Office 2007 Ribbon Bar Icons, Core Set 1
- Office 2007 Ribbon Bar Icons, Core Set 2
- Office 2007 Ribbon Bar Icons, Word Processing Set 1
- Office 2007 Ribbon Bar Icons, Word Processing Set 2

The corresponding sets must be purchased in order to use these projects. After purchasing, request this project from Glyfz.

The projects use 16x16 and 32x32 images (32x32 images are used only for several commands)

## Additional information

---

[Additional information is available on the support forum.](#)

### 1.6.2.4 Fugue Icons toolbar images

#### Fugue Icons

---

*(deprecated, this datamodule the not maintained anymore)*

If you want to use modern looking images, consider [Fugue Icons](#) toolbar images.

Fugue Icons – Copyright © 2009 Yusuke Kamiyamane.

The icons can be used free for any personal or commercial projects under a [Creative Commons Attribution 3.0 license](#).

If you can't or don't want to place link to the icons author website , please visit [p.yusukekamiyamane.com](http://p.yusukekamiyamane.com) and purchase a royalty-free license.

Download a datamodule with these icons from

<http://www.trichview.com/resources/actions/dmactionsfugue.zip>

For Delphi 2009+ only (because of semitransparent images)

The datamodule contains 2 identical sets of icons: with and without shadows.

## Installing

---

It's not necessary to install files to use this datamodule.

However, If you want to use the datamodule at design time instead of dmActions.pas:

1. Open **RichViewActions.inc** and remove the dot from one of the following line:  
{.\$DEFINE USEFUGUEICONS} // to use dmActionsFugue (1 byte transparency, D2009+)
2. Remove dmActions.pas from the RichViewActions package
3. Install (or recompile) the package.

## Additional information

---

[Additional information is available on the support forum.](#)

### 1.6.2.5 FamFamFam Silk Icons toolbar images

#### FamFamFam Silk Icons

---

*(deprecated, this datamodule the not maintained anymore)*

A FamFamFam Icons datamodule for RichViewActions is available.

You can download it from <http://www.trichview.com/resources/actions/dmactionssilk.zip>

For Delphi 2009+ only (because of semitransparent images)

Silk Icons - Copyright © Mark James, <http://www.famfamfam.com>.

The icons are licensed under a Creative Commons Attribution 2.5 or 3.0 License, details can be found on the author's website.

Unfortunately, this set contains not too many icons for table operations.

## Additional information

---

[Additional information is available on the support forum.](#)

## 1.7 Procedures and Functions

### Procedures and functions declared in RichViewActions

---

- GoToCheckpoint<sup>(377)</sup> moves the caret to the specified checkpoint;
- RVA\_ChooseStyle<sup>(378)</sup> allows choosing and applies Delphi VCL themes (requires Delphi XE2 or

newer);

- RVA\_EditorControlFunctionDef<sup>(378)</sup> allows using actions with other editor classes;
- RVA\_ConvertToPixels, RVA\_ConvertToTwips, RVA\_ConvertToEMU. RVA\_ConvertToUnits procedures<sup>(378)</sup>
- RvHtmlHelp function<sup>(379)</sup>

## See also

- Localization procedures and functions declared in RichViewActions unit<sup>(349)</sup>

## Procedures and functions declared in RVARibbonUtils

This unit contains procedures and functions for working with TRibbon control.

- FillActionClientContents<sup>(379)</sup>
- RemoveEllipsis<sup>(379)</sup>, RemoveEllipsisFromRibbon<sup>(379)</sup>
- LocalizeRibbonPage<sup>(380)</sup>
- SetRulerColors<sup>(380)</sup>

## Procedures and functions declared in other units

- LoadCSV<sup>(380)</sup> loads CSV ("comma-separated values") files as a table;
- MarkSubString<sup>(381)</sup> marks all occurrences of the specified string in a document;
- NormalizeRichView<sup>(382)</sup> fixes possible problems in a document;
- RVChangeCharCase, RVGetCharCase<sup>(383)</sup> set/get character case of the selected text.

### 1.7.1 RichViewActions unit

#### Procedures and functions declared in RichViewActions

- GoToCheckpoint<sup>(377)</sup> moves the caret to the specified checkpoint;
- RVA\_ChooseStyle<sup>(378)</sup> allows choosing and applies Delphi VCL themes (requires Delphi XE2 or newer);
- RVA\_EditorControlFunctionDef<sup>(378)</sup> allows using actions with other editor classes;
- RVA\_ConvertToPixels, RVA\_ConvertToTwips, RVA\_ConvertToEMU. RVA\_ConvertToUnits procedures<sup>(378)</sup>
- RvHtmlHelp function<sup>(379)</sup>

## See also

- Localization procedures and functions declared in RichViewActions unit<sup>(349)</sup>

#### 1.7.1.1 GoToCheckpoint function

Moves the caret in **rv** to the checkpoint having the specified name. Optionally, can scroll the editor to show the marked item in the center.

**Unit** RichViewActions;

```
function GoToCheckpoint(rv: TCustomRichViewEdit;  
    const CPName: TRVUnicodeString;  
    ScrollToCenter: Boolean = False): Boolean;
```

Returns *True* if the caret was successfully moved. Returns *False* if the checkpoint is not found.

This function is used in `TrvActionInsertHyperlink.GoToLink`<sup>(242)</sup> methods.

### 1.7.1.2 PasteHTML procedure (deprecated)

Deprecated. Use `TRichViewEdit.PasteHTML`.

Pastes HTML from the Clipboard to **rve** (if available)

**Unit** RichViewActions.

```
procedure PasteHTML(rve: TCustomRichViewEdit);
```

### 1.7.1.3 RVA\_ChooseStyle function

Displays a dialog for choosing a visual style (Delphi XE2+) and applies the chosen style.

**Unit** RichViewActions.

```
function RVA_ChooseStyle: Boolean;
```

This function requires Delphi XE2 or newer. When called from applications compiled in older version of Delphi, it displays an error message.

### 1.7.1.4 RVA\_EditorControlFunctionDef function

The function allows for actions inherited from `TrvCustomEditAction`<sup>(145)</sup> working with additional editor types.

**Unit** RichViewActions;

```
function RVA_EditorControlFunctionDef(Control: TControl;  
    Command: TRVAEditorControlCommand): Boolean;
```

This function is a default value of `RVA_EditorControlFunction`<sup>(389)</sup> variable.

It allows working with:

- `TCustomEdit` (`TEdit`, `TMemo`, `TRichEdit` and others);
- `TComboBox` (having `csDropDown` or `csSimple` style).

### 1.7.1.5 RVA\_ConvertTo\* procedures

The procedures convert all properties measured in `TRVAControlPanel`<sup>(38)</sup>.`UnitsProgram`<sup>(55)</sup> for all actions on the **Form**.

**Unit** RichViewActions.

```
procedure RVA_ConvertToTwips(Form: TComponent; ARVStyle: TRVStyle);  
procedure RVA_ConvertToPixels(Form: TComponent; ARVStyle: TRVStyle);  
procedure RVA_ConvertToEMU(Form: TComponent; ARVStyle: TRVStyle);  
procedure RVA_ConvertToUnits(Form: TComponent; ARVStyle: TRVStyle;  
    NewUnits: TRVStyleUnits);
```

**Form** is a form or datamodule. This procedure changes values of properties of all actions (inherited from `TrvCustomAction`<sup>(333)</sup>) owned by this form/datamodule.

**RVA\_ConvertToPixels** must be called before assigning *rvstuPixels* to TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

**RVA\_ConvertToTwips** must be called before assigning *rvstuTwips* to TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

**RVA\_ConvertToEMU** must be called before assigning *rvstuEMU* to TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

**RVA\_ConvertToUnits** must be called before assigning **NewUntits** to TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

The property **ARVStyle**.UnitsPixelsPerInch is used when converting to/from pixels (this is a number of pixels in one logical inch).

These procedures do not change value of TRVAControlPanel<sup>(38)</sup>.UnitsProgram<sup>(55)</sup>.

If you have actions linked to different control panels on this form/datamodule, you must perform this conversion for all these control panels at the same time (first, call **RVA\_ConvertTo\***, next, change UnitsProgram<sup>(55)</sup> for all these control panels).

### 1.7.1.6 RvHtmlHelp function

Opens an HTML help file (\*.CHM)

**Unit** RichViewActions.

**function** RvHtmlHelp (HelpFileName: TRVUnicodeString): Boolean;

**HelpFileName** is a name of CHM file, for example 'RichViewActions.chm'. It may include a topic name, for example 'RichViewActions.chm::RvHtmlHelp.htm' displays this topic.

## 1.7.2 RVARibbonUtils unit

### Procedures and functions declared in RVARibbonUtils

This unit contains procedures and functions for working with TRibbon control.

- FillActionClientContents<sup>(379)</sup>
- RemoveEllipsis<sup>(379)</sup>, RemoveEllipsisFromRibbon<sup>(379)</sup>
- LocalizeRibbonPage<sup>(380)</sup>
- SetRulerColors<sup>(380)</sup>

#### 1.7.2.1 FillActionClientContents procedure

For all action items in Manager, copies Action.Hint to CommandProperties.Content;

**Unit** RVARibbonUtils<sup>(379)</sup>.

**procedure** FillActionClientContents (Manager: TActionManager);

#### 1.7.2.2 RemoveEllipsis, RemoveEllipsisFromRibbon procedures

**Unit** RVARibbonUtils<sup>(379)</sup>.

**procedure** RemoveEllipsis (Form: TComponent);

**procedure** RemoveEllipsisFromRibbon (Manager: TActionManager);

RemoveEllipsis removes ending '...' from captions of all actions on the form. Not recommended for using.

RemoveEllipsisFromRibbon removes ending '...' from captions of all `Manager.ActionBars[.].Items[.]`.

### 1.7.2.3 LocalizeRibbonPage procedure

Assigns Caption and KeyTip to the ribbon **Page**.

**Unit** RVARibbonUtils<sup>(379)</sup>.

**procedure** LocalizeRibbonPage (Page: TRibbonPage; CaptionID: TRVAMessageID);

This procedure assigns the message identified by **CaptionID** to **Page.Caption**, after removing '&' from it. If it finds '&', it assigns a hot key from this message to **Page.KeyTip**.

### 1.7.2.4 SetRulerColors procedure

Changes **Ruler**'s colors according to **ColorMap**.

**Unit** RVARibbonUtils<sup>(379)</sup>.

**procedure** SetRulerColors (Ruler: TRVRuler<sup>(80)</sup>;  
ColorMap: TCustomRibbonColorMap);

## 1.7.3 Other units

### Procedures and functions declared in other units

- LoadCSV<sup>(380)</sup> loads CSV ("comma-separated values") files as a table;
- MarkSubString<sup>(381)</sup> marks all occurrences of the specified string in a document;
- NormalizeRichView<sup>(382)</sup> fixes possible problems in a document;
- RVChangeCharCase, RVGetCharCase<sup>(383)</sup> set/get character case of the selected text.

### 1.7.3.1 LoadCSV function

The function loads CSV ("comma-separated values") files as a table. Only ANSI files can be loaded.

**Unit** rvcsv.

**function** LoadCSV (const FileName: String; rv: TCustomRichView;  
Separator: TRVAnsiChar; UseQuotes: Boolean;  
TextStyleNo, ParaNo: Integer): TRVTableItemInfo;

**Parameters:**

**FileName** – file to load.

**rv** – TCustomRichView component where you plan to insert the loaded table (it is not inserted by this function).

**Separator** – character used to separate values. The most common possible values:

- ',' (comma)
- ';' (semicolon)
- ' ' (space)
- #9 (tab)



**UseQuotes** – if *True*, double-quotes are used to allow separators inside text values (in the input file); double double-quotes are treated as double-quote character.

**TextStyleNo, ParaNo** – styles of text and paragraph used to add text

**Return value:**

the loaded table; *nil* in case of failure.

**Example:**

```
var table: TRVTableItemInfo;
...
table := LoadCSV(FileName, RichViewEdit1, ',', True, 0, 0);
if table<>nil then
    RichViewEdit1.InsertItem(',', table);
```

You can implement inserting CSV files using `TRVAControlPanel.OnCustomFileOperation` <sup>62</sup> event.

### 1.7.3.2 MarkSubString functions

The functions mark all occurrences of **s** in **rv**.

**Unit** MarkSearch.

**type**

```
TRVMarkStringOption = (
    rvmsIgnoreCase, rvmsWholeWords, rvmsNormalize,
    rvmsStripDiacritic, rvmsIgnorePunctuation);
TRVMarkStringOptions = set of TRVMarkStringOption;
```

```
function MarkSubStringA(const s: TRVAnsiString;
    Color, BackColor: TColor;
    Options: TRVMarkStringOptions; rv: TCustomRichView;
    RVData: TCustomRVData = nil): Integer;
function MarkSubStringW(const s: TRVUnicodeString;
    Color, BackColor: TColor;
    Options: TRVMarkStringOptions; rv: TCustomRichView;
    RVData: TCustomRVData = nil): Integer;
```

All occurrences of **s** are marked with the specified **Color** and **BackColor**.

**RVData** may specify the document to process. If it is *nil*, the whole document in **rv** is processed (i.e. **RVData** = *nil* works like **RVData** = **rv.RVData**).

**Options** for the both functions:

Option	Meaning
<i>rvmsIgnoreCase</i>	If set, character case is ignored when comparing
<i>rvmsWholeWords</i>	If set, only occurrences as whole words are marked. "Word" is surrounded by characters listed in <b>rv.Delimiters</b> property.

**Options** for the both `MarkSubStringW`:

Option	Meaning
<i>rvmsNormalize</i>	If set, text is normalized when comparing ( <b>Compatibility Decomposition</b> form). With this option, you can search for '1' and find '①' (and vice versa); or you can search for '卜' and find 'ㄣ'.
<i>rvmsStripDiacritic</i>	Works like <i>rvmsNormalize</i> , and also all diacritic characters are ignored. For example, you can search for 'Ä' and find 'A' (and vice versa).
<i>rvmsIgnorePunctuation</i>	If set, any sequence of punctuation characters (i.e. characters listed in <b>rv.Delimiters</b> property) is treated as a single space character. For example, you can search for 'Hello world.' and find 'Hello, world!!!' (and vice versa).

This operation is not an editing operation, it cannot be undone.

These functions are not used in RichViewActions. They are included in RichViewActions for a convenience.

Limitation: this function searches text in each text item separately.

#### Platform notes:

- *rvmsIgnoreCase* for Unicode text works only on WinNT-based systems (WinNT4, Win2000, WinXP and newer)
- *rvmsNormalize* and *rvmsStripDiacritic* work in Windows Vista and newer. For WinXP, it requires libraries from <https://www.microsoft.com/en-us/download/details.aspx?id=734>

### 1.7.3.3 NormalizeRichView procedure

Fixes possible problems in TCustomRichView component.

**Unit** RVNormalize.

```
procedure NormalizeRichViewNormalizeRichView(RVData: TCustomRVData;
  FirstItemNo: Integer=0; Recursive: Boolean=True);
```

This procedure does not update undo/redo buffer in editor, so it must be called before the first editing operations after clearing and creating/loading document.

This procedure:

- fixes incorrect "new line" flags
- fixes paragraphs consisting only of list markers without items
- fixes incorrect ParaNo assignment
- removes empty text items (from places where they should not be)
- combines adjacent text items of the same text style and tag

This procedure fixes issues 1-3 and 5-7 listed in the TRichView help topic "Valid Documents"

#### Example:

```
NormalizeRichView(rv.RVData);
rv.Format;
```

Optional parameters:

**FirstItemNo** defines the first item to check. The procedure checks items from FirstItemNo to RVData.ItemCount-1. By default, all items are checked.

If **Recursive**=*True*, the procedure checks subdocuments of RVData (table cells).

### 1.7.3.4 RVChangeCharCase, RVGetCharCase procedures

These procedures allow changing character case in the selected fragment of TCustomRichViewEdit component.

**Unit** RVCharCase;

**type**

```
TRVCharCase = (rvccLowerCase, rvccUpperCase, rvccTitleWord);
```

```
procedure RVChangeCharCase(rve: TCustomRichViewEdit;  
    CharCase: TRVCharCase);
```

```
procedure RVGetCharCase(rve: TCustomRichViewEdit;  
    var AllUpperCase, AllLowerCase: Boolean);
```

**RVChangeCharCase** changes character case in the selection of **rve** according to **CharCase** parameter.

Option	Meaning
<i>rvccLowerCase</i>	change all characters to lower case
<i>rvccUpperCase</i>	change all characters to upper case
<i>rvccTitleWord</i>	change the first character of each word to upper case, other characters to lower case.

**RVGetCharCase** returns *True* in **AllUpperCase** parameter if all characters in the selection are in upper case.

**RVGetCharCase** returns *True* in **AllLowerCase** parameter if all characters in the selection are in lower case.

**See also:**

- TrvActionCharCase<sup>(130)</sup>

## 1.8 Types

### Other Types Declared in RichViewActions

Classes of properties:

- TRVAHFInfo<sup>(384)</sup> – properties of a header/footer drawn by RichViewActions;
- TRVFileFormatComponent<sup>(387)</sup> – a component providing file saving or loading.

Strings:

- TRVASpellString, TRVASpellStrings, TRVASpellStringList<sup>(385)</sup> – string types for user interface.

Files:

- TrvFileExportFilter, TrvFileSaveFilter<sup>386</sup> – saving file format;
- TrvFileImportFilter, TrvFileOpenFilter<sup>387</sup> – loading file format.

Rulers:

- TRulerUnits<sup>384</sup>

Events:

- TRVAEditEvent<sup>384</sup>
- TRVShowFormEvent<sup>388</sup>

### 1.8.1 TRulerUnits

Measure units used in TRVRuler<sup>80</sup>.

Unit RVRulerBase;

**type**

```
TRulerUnits = (ruInches, ruCentimeters, ruMillimeters,
               ruPicas, ruPixels, ruPoints);
```

Value	Units
<i>ruInches</i>	Inches
<i>ruCentimeters</i>	Centimeters
<i>ruMillimeters</i>	Millimeters
<i>ruPicas</i>	Picas (1 pica = 1/6 of an inch = 12 points)
<i>ruPixels</i>	Pixels (1 pixel = 1/UnitsPixelsPerInch <sup>85</sup> of an inch)
<i>ruPoints</i>	Points (1 point = 1/72 of an inch)

### 1.8.2 TRVAEditEvent, TRVAEditEvent2

These types are used for several events.

**type**

```
TRVAEditEvent = procedure (Sender: TrvAction311;
                             Edit: TCustomRichViewEdit) of object;
TRVAEditEvent2 = procedure (Sender: TrvCustomAction333;
                              Edit: TCustomRichViewEdit) of object;
```

### 1.8.3 TRVAHFInfo

TRVAHFInfo is the class used for storing properties of a header and a footer in RichViewActions<sup>16</sup>.

Unit RichViewActions

**Syntax**

```
TRVAHFInfo = class (TPersistent);
```

**Hierarchy**

*TObject*

*TPersistent*

**Description**

This is the type of TRVAControlPanel.Header and Footer<sup>(47)</sup> properties.

This class has the following properties.

**Text:** String – text displayed in header/footer. This text may contain the following codes:

- &p – index of the current page (counted from TRVAControlPanel.FirstPageNumber<sup>(47)</sup>);
- &P – count of pages;
- &d – current date (DateToStr)
- &t – current time (DateTimeToString(..., ShortTimeFormat, Time))
- && – '&' character.

You can implement your own one-letter codes using OnGetHeaderFooterCode<sup>(65)</sup> event.

Default value: '' (empty string) for footer, '- &p -' for header.

**Alignment:** TAlignment – horizontal alignment of text. Default value: *taCenter*.

**PrintOnFirstPage:** Boolean – specifies whether the header/footer is printed on the first page.

Default value: *True*.

**Font:** TRVAFont – font for printing text. Default value: Arial, 10 (where TRVAFont = class(TFont)).

RichViewActions print header and footer by assigning a temporal handler of TRVPrint.OnPagePrepaint event. If TRVPrint component (specified in TRVAControlPanel.RVPrint<sup>(53)</sup>) already has this event assigned, the temporal event handler calls the old handler before drawing text.

## 1.8.4 TRVASpellString, TRVASpellStrings, TRVASpellStringList

**Unit** RVASpellInterface.

If TNT Controls are not used, these strings are Unicode in Delphi 2009 and newer, and ANSI in older versions of Delphi:

**type**

```
TRVASpellListString = String;  
TRVASpellStrings = TStrings;  
TRVASpellStringList = TStringList;
```

If TNT Controls are used<sup>(370)</sup>, these strings are always Unicode:

**type**

```
TRVASpellListString = WideString;  
TRVASpellStrings = TTntStrings;  
TRVASpellStringList = TTntStringList;
```

## 1.8.5 TrvFileExportFilter, TrvFileSaveFilter

Types for Filter property of "Save As" and "Export" actions.

**Unit** RichViewActions;

**type**

```
TrvFileExportFilter = (ffeRVF, ffeRTF, ffeXML, ffeTextANSI,
    ffeTextUnicode, ffeCustom, ffeDocX, ffeMarkdown, ffeHTMLCSS,
    ffeHTML, ffeOfficeConverters);
TrvFileSaveFilter = ffeRVF..ffeHTMLCSS;
TrvFileExportFilterSet = set of TrvFileExportFilter;
TrvFileSaveFilterSet   = set of TrvFileSaveFilter;
```

The declaration above includes the full set of values, but the actual declaration depends on the third-party tools used in the project.

Value	File Type
<i>ffeRVF</i>	RichView Format (RVF)
<i>ffeRTF</i>	Rich Text Format (RTF)
<i>ffeXML</i>	Format of RichViewXML <sup>368</sup> (XML)
<i>ffeTextANSI</i>	Text files (TXT). The action asks user to select encoding (from Windows code pages, UTF-8, UTF-16)
<i>ffeTextUnicode</i>	Unicode (UTF-16) text files (TXT)
<i>ffeCustom</i>	Custom file formats defined in CustomFilter property of the corresponding action
<i>ffeHTMLCSS</i>	HTML files (HTM, HTML) saved by TCustomRichViewEdit.SaveHTMLEx method
<i>ffeHTML</i>	Simplified HTML files (HTM, HTML) saved by TCustomRichViewEdit.SaveHTML method
<i>ffeMarkdown</i>	<b>Markdown</b> (MARKDOWN, MDOWN, MKDN, MD, MKD, MDWN, MDTXT, MDTEXT)
<i>ffeDocX</i>	Microsoft Word 2007+ Format (DocX)
<i>ffeOfficeConverters</i>	Formats supported by Microsoft Office text export converters

**See also:**

- TrvActionSaveAs.Filter<sup>129</sup>
- TrvActionExport.Filter<sup>113</sup>

### 1.8.6 TRVFileFormatComponent

TRVFileFormatComponent is a base class for components allowing to load and save TRichView documents in files of various formats.

Unit RVFile;

#### Syntax

```
TRVFileFormatComponent = class (TComponent);  
TRVHTMLComponent = class (TRVFileFormatComponent);  
TRVXMLComponent = class (TRVFileFormatComponent);
```

#### Hierarchy

*TObject*  
*TPersistent*  
*TComponent*

#### Description

Although RVFile unit is included in TRichView, not in RichViewActions, we describe it here, because it is used in RichViewActions.

This class is not used directly. Instead, the following inherited components are used:

- TRichViewXML<sup>(368)</sup>: allows loading, saving, and inserting XML files (an alternative to RVF);

These components are not included in RichViewActions. However, they are installed by TRichView Setup (if possible). RichViewActions can use them; but if you do not use them in your application, their code is not linked to your exe-file.

#### See also properties of TRVAControlPanel:

- XMLComponent<sup>(58)</sup>

### 1.8.7 TrvFileImportFilter, TrvFileOpenFilter

Types for Filter property of "Open" and "Insert File" actions.

Unit RichViewActions;

#### type

```
TrvFileImportFilter = (ffiRVF, ffiRTF, ffiXML, ffiTextANSI,  
    ffiTextUnicode, ffiTextAuto, ffiCustom, ffiDocX, ffiMarkdown,  
    ffiHTML, ffiOfficeConverters);  
TrvFileOpenFilter = ffiRVF..ffiHTML;  
TrvFileImportFilterSet = set of TrvFileImportFilter;  
TrvFileOpenFilterSet = set of TrvFileOpenFilter;
```

The declaration above includes the full set of values, but the actual declaration depends on the third-party tools used in the project.

*ffiXML* can be used only if RichViewXML<sup>(368)</sup> is used.

Value	File Type
<i>ffiRVF</i>	RichView Format (RVF)
<i>ffiRTF</i>	Rich Text Format (RTF)

<i>ffiXML</i>	Format of RichViewXML <sup>(368)</sup> (XML)
<i>ffiTextANSI</i>	Text files (TXT). The action asks user to select encoding (from Windows code pages, UTF-8, UTF-16)
<i>ffiTextUnicode</i>	Unicode (UTF-16) text files (TXT)
<i>ffiTextAuto</i>	ANSI or Unicode (UTF-16) text files (TXT), autodetected
<i>ffiCustom</i>	Custom file formats defined in CustomFilter property of the corresponding action
<i>ffiDocX</i>	Microsoft Word 2007+ Format (DocX)
<i>ffiMarkdown</i>	<b>Markdown</b> (MARKDOWN, MDOWN, MKDN, MD, MKD, MDWN, MDTXT, MDTEXT)
<i>ffiHTML</i>	HTML files (HTM, HTML)
<i>ffiOfficeConverters</i>	Formats supported by Microsoft Office text export converters

**See also:**

- TrvActionOpen.Filter<sup>(108)</sup>
- TrvActionInsertFile.Filter<sup>(230)</sup>

## 1.8.8 TRVShowFormEvent

This type is used in the following events:

- TrvActionCustomInfoWindow.OnShowing<sup>(219)</sup>
- TrvCustomEditorAction.OnShowing<sup>(338)</sup>
- TrvCustomEditorAction.OnFormCreate<sup>(338)</sup>

**type**

```
TRVShowFormEvent = procedure (Sender: TrvAction(311);
    Form: TForm) of object;
```

## 1.9 Global variables

### Global Variables Used in RichViewActions

- MainRVAControlPanel<sup>(388)</sup> – the default control panel component;
- RVA\_EditForceDefControl<sup>(389)</sup> specified priority of components when applying actions;
- RVA\_EditorControlFunction<sup>(389)</sup> – a function implementing operations on third-party editor control;
- ShowUntranslatedControls<sup>(390)</sup> – shows/hides untranslated UI messages.

### 1.9.1 MainRVAControlPanel

The main control panel for RichViewActions.

**Unit** RichViewActions;

```
var MainRVAControlPanel: TRVAControlPanel(38);
```



If no other control panels are created, this variable returns a default control panel object (not belonging to any form or datamodule).

If an application contains a single TRVControlPanel on a form, it becomes the main control panel automatically, and MainRVAControlPanel returns this control panel.

Another control panel can be made a main one if you call its Activate<sup>(59)</sup> method.

## 1.9.2 RVA\_EditForceDefControl

Specifies a priority of non-TCustomRichViewEdit editors for editing actions.

**var**

```
RVA_EditForceDefControl: Boolean = False;
```

By default, focused non-TCustomRichViewEdit controls have higher priority in editing actions<sup>(145)</sup> (such as Cut, Copy, Paste) than Control<sup>(312)</sup> or GetControlPanel<sup>(336)</sup>.DefaultControl<sup>(43)</sup> properties.

For example, in the ActionTest demo, if you select text in the font name combo box and press **Ctrl + X**, the selected text from this combo box will be cut to the Clipboard, the action ignores RVAControlPanel1.DefaultControl<sup>(43)</sup>=RichViewEdit1.

If you want to change this priority, assign RVA\_EditForceDefControl := True.

## 1.9.3 RVA\_EditorControlFunction

This variable refers to the function allowing for the actions inherited from TrvCustomEditAction<sup>(145)</sup> to work with additional editor types.

**Unit RichViewActions;**

**type**

```
TRVAEditorControlCommand = (rvaeccIsEditorControl, rvaeccHasSelection,
    rvaeccCanPaste, rvaeccCanPasteText,
    rvaeccCanUndo, rvaeccCopy, rvaeccCut,
    rvaeccPaste, rvaeccPasteText, rvaeccUndo);
TRVAEditorControlCommandFunction =
    function (Control: TControl;
        Command: TRVAEditorControlCommand): Boolean;
```

**var**

```
RVA_EditorControlFunction: TRVAEditorControlCommandFunction;
```

By default, this variable is initialized with RVA\_EditorControlFunctionDef<sup>(378)</sup> function.

Command	The function must...
<i>rvaeccIsEditorControl</i>	... return True if Control is a supported editor control.
<i>rvaeccHasSelection</i>	... return True if the selection in Control is not empty
<i>rvaeccCanPaste</i>	... return True if Control can paste data from the Clipboard.
<i>rvaeccCanPasteText</i>	... return True if Control can paste text from the Clipboard.
<i>rvaeccCanUndo</i>	... return True if Control can undo the last editing operation

<i>rvaeccCopy</i>	... copy the selection from Control to the Clipboard
<i>rvaeccCut</i>	... cut the selection from Control to the Clipboard
<i>rvaeccPaste</i>	... paste data from the Clipboard into Control
<i>rvaeccPasteText</i>	... paste text from the Clipboard into Control
<i>rvaeccUndo</i>	... undo the last editing operation in Control

### 1.9.4 ShowUntranslatedControls

This variable allows hiding controls in RichViewActions dialog that were not translated yet.

**Unit** RichViewActions;

**var**

```
ShowUntranslatedControls: Boolean = True;
```

## 1.10 How to

### How To...

- How to change RVF and XML format name <sup>(390)</sup>

#### 1.10.1 How to change RVF and XML format name

By default, RVF (RichView Format) is the main file format of RichViewActions.

If you want to change file extension and the format name, you can do it in the following way.

Place RVAControlPanel1: TRVAControlPanel <sup>(38)</sup> component on the form. Change the following properties (in the code or in the Object Inspector):

```
RVAControlPanel1.RVFLocalizable (52) := False;
RVAControlPanel1.RVFFilter (52) := 'SuperApp Files (*.saf)|*.saf';
RVAControlPanel1.DefaultExt (44) := 'saf';
RVAControlPanel1.RVFormatTitle (53) := 'SuperApp Format';
```

Additionally, you can change names of files used by TrvActionStyleTemplates <sup>(212)</sup> to save and load style templates:

```
RVAControlPanel1.RVStylesFilter (53) := 'SuperApp Styles (*.sas)|*.sas';
RVAControlPanel1.RVStylesExt (53) := 'sas';
```

Now, if you want to localize these properties <sup>(340)</sup>, you need to assign them yourself every time when the application language is changed.

For XML format, use XMLLocalizable <sup>(58)</sup> and XMLFilter <sup>(58)</sup> properties.

# Index

## - A -

- ActionEditNote 266
  - TrvActionInsertNote 266
- ActionFind 135, 142
  - TrvActionFindNext 135
  - TrvActionReplace 142
- ActionFont 34, 36
  - TCustomRVFontComboBox 34
  - TCustomRVFontListBox 36
- ActionInsertHLine 326
  - TrvActionItemProperties 326
- ActionInsertHyperlink 330
  - TrvActionRemoveHyperlinks 330
- ActionInsertTable 303, 326
  - TrvActionItemProperties 326
  - TrvActionTableProperties 303
- ActionList 69
  - TRVAPopupMenu 69
  - TRVASPTBXPopupMenu 69
  - TRVATBPopupMenu 69
  - TRVATBXPopupMenu 69
- ActionNew 108
  - TrvActionOpen 108
- ActionPageSetup 119
  - TrvActionPrintPreview 119
- ActionParaBullets 208
  - TrvActionParaList 208
- ActionParaNumbering 208
  - TrvActionParaList 208
- ActionPrint 119
  - TrvActionPrintPreview 119
- ActionReplace 133
  - TrvActionFind 133
- Actions
  - Edit 130
  - File 96
  - Font 146
  - Notes and text boxes 257
  - Other 311
  - Spelling check and thesaurus 310
  - Tables 270
- ActionSave 102, 129
  - TrvActionCustomNew 102
  - TrvActionSaveAs 129

- ActionSaveAs 123
  - TrvActionSave 123
- ActionsEnabled 41
  - TRVAControlPanel 41
- ActionStyleTemplates 216, 235
  - TrvActionAddStyleTemplate 216
  - TrvActionInsertHyperlink 235
- Activate 59
  - TRVAControlPanel 59
- AddColorNameToHints 41
  - TRVAControlPanel 41
- AddDefaultCharset 73
  - TRVFontCharsetComboBox 73
- AddFormat 139
  - TrvActionPasteSpecial 139
- Addict 3 362
- Alignment 200, 202
  - TrvActionParagraph 200, 202
- AllowApplyingTo 324
  - TrvActionFillColor 324
- AllowedCharacters 222
  - TrvActionBookmarks 222
- AllowFocusEditor 36
  - TCustomRVFontListBox 36
- AllowMultiple 300
  - TrvActionTableInsertRowsBelow 300
- AllStylesImageIndex 89
  - TRVStyleTemplateComboBox 89
  - TRVStyleTemplateListBox 89
- ASpell 363
- AutoAddHyperlinkStyleTemplate 235
  - TrvActionInsertHyperlink 235
- AutoApplySymbolCharset 151
  - TrvActionFontEx 151
- AutoCharset 75, 78
  - TRVFontComboBox 75
  - TRVFontListBox 78
- AutoDeleteUnusedStyles 42
  - TRVAControlPanel 42
- AutoUpdateFileName 99
  - TrvActionCustomIO 99

## - B -

- BackColor 152, 235
  - TrvActionFontEx 152
  - TrvActionInsertHyperlink 235
- Background 193
  - TrvActionParaBorder 193
- BackgroundColor 251

BackgroundColor 251  
     TrvActionInsertPicture 251  
 BackgroundGraphicFilter 303, 327  
     TrvActionItemProperties 327  
     TrvActionTableProperties 303  
 BackgroundPicture 273  
     TrvActionInsertTable 273  
 BackgroundStyle 273  
     TrvActionInsertTable 273  
 BestWidth 274  
     TrvActionInsertTable 274  
 BiDiModeRuler 82  
     TCustomRuler 82  
 Border 193  
     TrvActionParaBorder 193  
 BorderColor 251, 274  
     TrvActionInsertPicture 251  
     TrvActionInsertTable 274  
 BorderHSpacing 274  
     TrvActionInsertTable 274  
 BorderLightColor 274  
     TrvActionInsertTable 274  
 BorderStyle 275  
     TrvActionInsertTable 275  
 BorderVSpacing 275  
     TrvActionInsertTable 275  
 BorderWidth 251, 275  
     TrvActionInsertPicture 251  
     TrvActionInsertTable 275  
 BoxPosition 259  
     TrvActionCustomInsertSidenote 259  
 BoxProperties 260  
     TrvActionCustomInsertSidenote 260

## - C -

CallerControl 319  
     TrvActionCustomColor 319  
 CanChangeMargins 314  
     TrvActionBackground 314  
 CanCloseDoc 125  
     TrvActionSave 125  
 Caption 334  
     TrvCustomAction 334  
 CellBorderColor 275  
     TrvActionInsertTable 275  
 CellBorderLightColor 276  
     TrvActionInsertTable 276  
 CellBorderStyle 276  
     TrvActionInsertTable 276  
 CellBorderWidth 276  
     TrvActionInsertTable 276  
 CellHPadding 276  
     TrvActionInsertTable 276  
 CellHSpacing 277  
     TrvActionInsertTable 277  
 CellPadding 276  
     TrvActionInsertTable 276  
 CellVPadding 276  
     TrvActionInsertTable 276  
 CellVSpacing 277  
     TrvActionInsertTable 277  
 CharCode 255  
     TrvActionInsertSymbol 255  
 CharScale 152  
     TrvActionFontEx 152  
 CharSpacing 152  
     TrvActionFontEx 152  
 ClearFormatImageIndex 89  
     TRVStyleTemplateComboBox 89  
     TRVStyleTemplateListBox 89  
 CleverComponents 367  
 CloseDialog 134, 143  
     TrvActionFind 134  
     TrvActionReplace 143  
 ColBandSize 277  
     TrvActionInsertTable 277  
 ColCount 278  
     TrvActionInsertTable 278  
 Color 232, 236, 278, 319  
     TrvActionCustomColor 319  
     TrvActionInsertHLine 232  
     TrvActionInsertHyperlink 236  
     TrvActionInsertTable 278  
 ColorDialog 42  
     TRVAControlPanel 42  
 ColorDialogInterface 42  
     TRVAControlPanel 42  
 Control 312  
     TrvAction 312  
 ControlPanel 89, 335  
     TrvCustomAction 335  
     TRVStyleTemplateComboBox 89  
     TRVStyleTemplateListBox 89  
 CreateDirectoryForHTMLImages 113, 124  
     TrvActionExport 113  
     TrvActionSave 124  
 Cursor 236  
     TrvActionInsertHyperlink 236  
 CustomFilter 98, 108

CustomFilter 98, 108  
     TrvActionCustomFileIO 98  
     TrvActionOpen 108

## - D -

DefaultCharsetCaption 73  
     TRVFontCharsetComboBox 73  
 DefaultChecked 303, 327  
     TrvActionItemProperties 327  
     TrvActionTableProperties 303  
 DefaultColor 42  
     TRVAControlPanel 42  
 DefaultControl 43  
     TRVAControlPanel 43  
 DefaultCustomFilterIndex 43  
     TRVAControlPanel 43  
 DefaultDocParameters 43  
     TRVAControlPanel 43  
 DefaultExt 44, 252  
     TRVAControlPanel 44  
     TrvActionInsertPicture 252  
 DefaultFileFormat 44  
     TRVAControlPanel 44  
 DefaultFileName 44  
     TRVAControlPanel 44  
 DefaultMargin 45  
     TRVAControlPanel 45  
 DefaultPersistent 303, 327  
     TrvActionItemProperties 327  
     TrvActionTableProperties 303  
 DeleteAllTabs 200  
     TrvActionParagraph 200  
 DetectURL 240  
     TrvActionInsertHyperlink 240  
 DialogFontName 45  
     TRVAControlPanel 45  
 DialogFontNameLin 45  
     TRVAControlPanel 45  
 DialogFontSize 45  
     TRVAControlPanel 45  
 DialogPosition 46  
     TRVAControlPanel 46  
 DialogTitle 100, 108  
     TrvActionCustomIO 100  
     TrvActionOpen 108  
 DialogZoomPercent 46  
     TRVAControlPanel 46  
 Disabled 335  
     TrvCustomAction 335

DisableWhenUnmodified 124  
     TrvActionSave 124  
 Documents 124  
     TrvActionSave 124  
 DoImportPicturesAndFiles 59  
     TRVAControlPanel 59  
 DownloadInterface 21, 46  
     TRVAControlPanel 46  
 DropDownWidth 75  
     TRVFontComboBox 75

## - E -

Editor 35, 37, 89  
     TCustomRVFontComboBox 35  
     TCustomRVFontListBox 37  
     TRVStyleTemplateComboBox 89  
     TRVStyleTemplateListBox 89  
 EncodeTarget 241  
     TrvActionInsertHyperlink 241  
 EvenColumnsColor 281  
     TrvActionInsertTable 281  
 EvenRowsColor 281  
     TrvActionInsertTable 281  
 ExcludeLabel 224  
     TrvActionInsertCaption 224  
 ExportToFile 114  
     TrvActionExport 114  
 Expression 228  
     TrvActionInsertEquation 228  
 ExpressSpellChecker 364

## - F -

FamFamFam Silk Icons toolbar images 376  
 FileName 100  
     TrvActionCustomIO 100  
 FillActionClientContents 379  
 Filter 108, 113, 129, 230, 252  
     TrvActionExport 113  
     TrvActionInsertFile 230  
     TrvActionInsertPicture 252  
     TrvActionOpen 108  
     TrvActionSaveAs 129  
 FindDoc 126  
     TrvActionSave 126  
 FindText 134, 142  
     TrvActionFind 134  
     TrvActionReplace 142  
 FirstColumnColor 281

FirstColumnColor 281  
     TrvActionInsertTable 281  
 FirstIndent 201  
     TrvActionParagraph 201  
 FirstPageNumber 47  
     TRVAControlPanel 47  
 Flat 83  
     TCustomRuler 83  
 Font 161, 266  
     TrvActionFonts 161  
     TrvActionInsertNote 266  
 FontImageIndex 218  
     TrvActionStyleInspector 218  
 FontName 73, 80, 256  
     TrvActionInsertSymbol 256  
     TRVFontCharsetComboBox 73  
     TRVFontSizeComboBox 80  
 FontSizeDouble 153  
     TrvActionFontEx 153  
 FontStyleEx 153  
     TrvActionFontEx 153  
 Footer 47  
     TRVAControlPanel 47  
 Form 337  
     TrvCustomEditorAction 337  
 Fugue Icons toolbar images 375

## - G -

GetAddictSpellLanguage 351  
 GetAddictThesLanguage 351  
 GetControlPanel 336  
     TrvCustomAction 336  
 GetHyperlinkStyleNo 241  
     TrvActionInsertHyperlink 241  
 GetNormalStyleNo 242  
     TrvActionInsertHyperlink 242  
 GetRealDialogFontName 59  
     TRVAControlPanel 59  
 GlyFX toolbar images 374  
 Glyfz toolbar images 375  
 GoToCheckpoint 377  
 GoToLink 242  
     TrvActionInsertHyperlink 242  
 GraphicFilter 328  
     TrvActionItemProperties 328

## - H -

Header 47

TRVAControlPanel 47  
 HeadingRowColor 281  
     TrvActionInsertTable 281  
 HeadingRowCount 278  
     TrvActionInsertTable 278  
 HelpType 47  
     TRVAControlPanel 47  
 Hint 335  
     TrvCustomAction 335  
 HOutermostRule 278  
     TrvActionInsertTable 278  
 HoverBackColor 236  
     TrvActionInsertHyperlink 236  
 HoverColor 237  
     TrvActionInsertHyperlink 237  
 HoverEffects 237  
     TrvActionInsertHyperlink 237  
 HoverUnderlineColor 237  
     TrvActionInsertHyperlink 237  
 How to  
     change RVF format name 390  
 How to change RVF format name 390  
 HRuleColor 278  
     TrvActionInsertTable 278  
 HRuleWidth 279  
     TrvActionInsertTable 279  
 HTMLComponent 51  
     TRVAControlPanel 51  
 HunSpell 365

## - I -

If 176  
 ImageFileName 315  
     TrvActionBackground 315  
 Images 90, 214, 218  
     TrvActionStyleInspector 218  
     TrvActionStyleTemplates 214  
     TRVStyleTemplateComboBox 90  
     TRVStyleTemplateListBox 90  
 IndentColor 83  
     TCustomRuler 83  
 IndentMax 188  
     TrvActionIndentInc 188  
 IndentSaturation 84  
     TCustomRuler 84  
 IndentStep 184, 186, 207  
     TrvActionCustomParaListSwitcher 184  
     TrvActionIndent 186  
     TrvActionParaList 207

Indy 368  
 InitialDir 100, 109  
     TrvActionCustomIO 100  
     TrvActionOpen 109  
 InitImportPicturesAndFiles 59  
     TRVAControlPanel 59  
 InsertAbove 225  
     TrvActionInsertCaption 225  
 InsertFile 230  
     TrvActionInsertFile 230  
 Interfaces for downloader third-party components in RichViewActions 353  
 Interfaces for spell checker third-party components in RichViewActions 355  
 Interfaces for third-party color-dialog components in RichViewActions 360  
 Interfaces for third-party components in RichViewActions 352  
 ItemHeight 75, 78  
     TRVFontComboBox 75  
     TRVFontListBox 78  
 ItemText 279  
     TrvActionInsertTable 279

## - J -

JumpToNextNote 262  
     TrvActionEditNote 262

## - K -

KeepLinesTogether 201  
     TrvActionParagraph 201  
 KeepWithNext 201  
     TrvActionParagraph 201

## - L -

Language 51  
     TRVAControlPanel 51  
 LastColumnColor 281  
     TrvActionInsertTable 281  
 LastFilterIndex 109  
     TrvActionOpen 109  
 LastLineAlignment 176, 202  
     TrvActionAlignCustomJustify 176  
     TrvActionParagraph 202  
 LastRowColor 281  
     TrvActionInsertTable 281  
 LeftIndent 202  
     TrvActionParagraph 202

LineSpacing 202  
     TrvActionParagraph 202  
 LineSpacingType 203  
     TrvActionParagraph 203  
 ListLevels 184  
     TrvActionCustomParaListSwitcher 184  
 LoadCSV 380  
 LoadFile 109  
     TrvActionOpen 109  
 Localization of RichViewActions 340  
 Localize 91  
     TRVStyleTemplateComboBox 91  
     TRVStyleTemplateListBox 91  
 LocalizeRibbonPage 380  
 LostFormatWarning 125  
     TrvActionSave 125

## - M -

MainRVAControlPanel 388  
 MarginColor 83  
     TCustomRuler 83  
 MarginSaturation 84  
     TCustomRuler 84  
 MarginsUnits 115  
     TrvActionPageSetup 115  
 MarkSubStringA 381  
 MarkSubStringW 381  
 MaxImageSize 252  
     TrvActionInsertPicture 252  
 Maximized 120  
     TrvActionPrintPreview 120  
 MaxSize 157, 158  
     TrvActionFontGrow 157  
     TrvActionFontGrowOnePoint 158  
 MaxSuggestionsCount 69  
     TRVAPopupMenu 69  
     TRVASPTBXPopupMenu 69  
     TRVATBPopupMenu 69  
     TRVATBXPopupMenu 69  
 MetafileCompatibility 51  
     TRVAControlPanel 51  
 MinSize 163, 165  
     TrvActionFontShrink 163  
     TrvActionFontShrinkOnePoint 165

## - N -

NormalizeRichView 382  
 NumberType 226, 247

NumberType 226, 247  
 TrvActionInsertCustomPageNumber 226  
 TrvActionInsertNumSequence 247

## - O -

OddColumnsColor 281  
 TrvActionInsertTable 281

OddRowsColor 281  
 TrvActionInsertTable 281

OnAddStyle 60  
 TRVAControlPanel 60

OnApplyHyperlinkToItem 244  
 TrvActionInsertHyperlink 244

OnBackgroundChange 61  
 TRVAControlPanel 61

OnCanApply 328  
 TrvActionItemProperties 328

OnCanPaste 139  
 TrvActionPasteSpecial 139

OnChange 116, 315  
 TrvActionBackground 315  
 TrvActionPageSetup 116

OnChoosePicture 61  
 TRVAControlPanel 61

OnClickAllStyles 92  
 TRVStyleTemplateComboBox 92  
 TRVStyleTemplateListBox 92

OnCloseTableSizeDialog 282  
 TrvActionInsertTable 282

OnColorSelected 317  
 TrvActionUserDefinedColor 317

OnCreateForm 62  
 TRVAControlPanel 62

OnCustomFileOperation 62  
 TRVAControlPanel 62

OnCustomItemPropertiesDialog 329  
 TrvActionItemProperties 329

OnCustomPaste 140  
 TrvActionPasteSpecial 140

OnDocumentFileChange 126  
 TrvActionSave 126

OnDownload 64  
 TRVAControlPanel 64

OnExecute 322  
 TrvActionEvent 322

OnFormCreate 338  
 TrvCustomEditorAction 338

OnGetActionControlCoords 64  
 TRVAControlPanel 64

OnGetColor 318  
 TrvActionUserDefinedColor 318

OnGetHeaderFooterCode 65  
 TRVAControlPanel 65

OnGetHyperlinkTargetFromItem 244  
 TrvActionInsertHyperlink 244

OnGetPreviewFormClass 120  
 TrvActionPrintPreview 120

OnHide 338  
 TrvCustomEditorAction 338

OnHideColorPicker 321  
 TrvActionCustomColor 321

OnHyperlinkForm 244  
 TrvActionInsertHyperlink 244

OnInserting 254, 282  
 TrvActionInsertPicture 254  
 TrvActionInsertTable 282

OnInsertText 257  
 TrvActionInsertText 257

OnLiveSpellAdd 69  
 TRVAPopupMenu 69  
 TRVASPTBXPopupMenu 69  
 TRVATBPopupMenu 69  
 TRVATBXPopupMenu 69

OnLiveSpellGetSuggestions 70  
 TRVAPopupMenu 70  
 TRVASPTBXPopupMenu 70  
 TRVATBPopupMenu 70  
 TRVATBXPopupMenu 70

OnLiveSpellIgnoreAll 70  
 TRVAPopupMenu 70  
 TRVASPTBXPopupMenu 70  
 TRVATBPopupMenu 70  
 TRVATBXPopupMenu 70

OnLiveSpellWordReplace 71  
 TRVAPopupMenu 71  
 TRVASPTBXPopupMenu 71  
 TRVATBPopupMenu 71  
 TRVATBXPopupMenu 71

OnMarginsChanged 65  
 TRVAControlPanel 65

OnNew 101  
 TrvActionCustomNew 101

OnOpenFile 110  
 TrvActionOpen 110

OnReplaceAllEnd 143  
 TrvActionReplace 143

OnReplaceAllStart 143  
 TrvActionReplace 143

OnReplacing 144



- OnReplacing 144
  - TrvActionReplace 144
- OnSave 126
  - TrvActionSave 126
- OnSaving 126
  - TrvActionSave 126
- OnShowColorPicker 321
  - TrvActionCustomColor 321
- OnShowing 140, 219, 338
  - TrvActionCustomInfoWindow 219
  - TrvActionPasteSpecial 140
  - TrvCustomEditorAction 338
- OnShowingDialog 155, 194, 205
  - TrvActionFontEx 155
  - TrvActionParaBorder 194
  - TrvActionParagraph 205
- OnStartEditNote 263
  - TrvActionEditNote 263
- OnStyleNeeded 66
  - TRVAControlPanel 66
- OnUpdate 322
  - TrvActionEvent 322
- OnViewChanged 66
  - TRVAControlPanel 66
- Opacity 320
  - TrvActionCustomColor 320
- OuterHSpacing 253
  - TrvActionInsertPicture 253
- OuterVSpacing 253
  - TrvActionInsertPicture 253
- OutlineLevel 203
  - TrvActionParagraph 203

## - P -

- Padding 197
  - TrvActionParaColorAndPadding 197
- ParaImageIndex 218
  - TrvActionStyleInspector 218
- ParaStyleImageIndex 89, 215
  - TrvActionStyleTemplates 215
  - TRVStyleTemplateComboBox 89
  - TRVStyleTemplateListBox 89
- ParaTextStyleImageIndex 89, 215
  - TrvActionStyleTemplates 215
  - TRVStyleTemplateComboBox 89
  - TRVStyleTemplateListBox 89
- PasteHTML 378
- Percent 164
  - TrvActionFontShrinkGrow 164

- PixelBorders 52
  - TRVAControlPanel 52
- Polar SpellChecker ActiveX 366
- Preview 76, 78
  - TRVFontComboBox 76
  - TRVFontListBox 78
- PreviewInList 153
  - TrvActionFontEx 153
- Protection 256
  - TrvActionInsertSymbol 256

## - R -

- RefStyleTemplateName 269
  - TrvActionInsertSidenote 269
- RemoveEllipsis 379
- RemoveEllipsisFromRibbon 379
- ReplaceText 142
  - TrvActionReplace 142
- Reset 105
  - TrvActionNew 105
- RichViewActions
  - downloading images 367, 368
  - FamFamFam Silk Icons toolbar images 376
  - Fugue Icons toolbar images 375
  - GlyFX toolbar images 374
  - Glyfz toolbar images 375
  - history 21
  - inserting and pasting HTML 368
  - interfaces for downloader third-party components 353
  - interfaces for spell checker third-party components 355
  - interfaces for third-party color-dialog components 360
  - interfaces for third-party components 352
  - localizing 340
  - opening-inserting-saving HTML 369
  - opening-inserting-saving XML 368
  - Style Templates 20
  - Third-party tools 360
  - TNT Controls 370
  - TRichView toolbar images 372
  - Types 383
  - using 16
  - using Addict 3 components 362
  - using ASpell 363
  - using ExpressSpellChecker 364
  - using HunSpell 365
  - using Polar SpellChecker 366
  - using SpTBXLib 370

RichViewActions  
   using TBX 371  
   using Toolbar 2000 371  
 RichViewActions overview 16  
 RichViewActions unit 349  
 RichViewActions.Overview 16  
 RichViewEdit 86  
   TRVRuler 86  
 RichViewXML 368  
 RightIndent 203  
   TrvActionParagraph 203  
 RowBandSize 277  
   TrvActionInsertTable 277  
 RowCount 279  
   TrvActionInsertTable 279  
 RowsKeepTogether 279  
   TrvActionInsertTable 279  
 RowsVAlign 280  
   TrvActionInsertTable 280  
 Ruler 38  
   TCustomRulerItemSelector 38  
 RulerSaturation 83, 84  
   TCustomRuler 83, 84  
 RulerType 84  
   TCustomRuler 84  
 RVA\_ChooseLanguage 349  
 RVA\_ChooseStyle 378  
 RVA\_ConvertToEMU 378  
 RVA\_ConvertToPixels 378  
 RVA\_ConvertToTwips 378  
 RVA\_ConvertToUnits 378  
 RVA\_EditForceDefControl 389  
 RVA\_EditorControlFunction 389  
 RVA\_EditorControlFunctionDef 378  
 RVA\_EnumLanguages 344  
 RVA\_FillLanguageList 344  
 RVA\_GetCharset 345  
 RVA\_GetColorName 350  
 RVA\_GetHelpFile 345  
 RVA\_GetLanguageName 346  
 RVA\_GetPC 346  
 RVA\_GetPrintingMessage 343  
 RVA\_GetProgressMessage 343  
 RVA\_GetS 346  
 RVA\_GetSH 346  
 RVA\_LocalizeForm 350  
 RVA\_SetHelpFile 347  
 RVA\_SwitchLanguage 347  
 RVA\_TranslateUnits 347

RVAAddictLanguages unit 351  
 RVALocalize unit 341  
 RVALocalizeRuler 351  
 RVAlocRuler unit 351  
 RVARibbonUtils unit 376, 379  
 RVChangeCharCase 383  
 RVCharCase 383  
 rvcsv unit 380  
 RVFFilter 52  
   TRVAControlPanel 52  
 RVFLocalizable 52  
   TRVAControlPanel 52  
 RVFormatTitle 53  
   TRVAControlPanel 53  
 RVGetCharCase 383  
 RVGetHelpKeyword 352  
 RVHelp unit 352  
 RVHtmlHelp 379  
 RVHTMLImporter 368  
 RVHTMLViewImporter 369  
 RVNormalize 382  
 RVPrint 53  
   TRVAControlPanel 53  
 RVSetHelpKeyword 352  
 RVStylesExt 53  
   TRVAControlPanel 53  
 RVStylesFilter 53  
   TRVAControlPanel 53

## - S -

ScreenRes 84  
   TCustomRuler 84  
 ScrollToCenter 222, 238  
   TrvActionBookmarks 222  
   TrvActionInsertHyperlink 238  
 SearchScope 53  
   TRVAControlPanel 53  
 SeqName 247  
   TrvActionInsertNumSequence 247  
 SetRulerColors 380  
 SetTags 243  
   TrvActionInsertHyperlink 243  
 SetToPictures 238  
   TrvActionInsertHyperlink 238  
 ShowAllStyles 90  
   TRVStyleTemplateComboBox 90  
   TRVStyleTemplateListBox 90  
 ShowCheckpoints 332  
   TrvActionShowSpecialCharacters 332

ShowClearFormat 90  
     TRVStyleTemplateComboBox 90  
     TRVStyleTemplateListBox 90

ShowReplaceAllSummary 143  
     TrvActionReplace 143

ShowSoftPageBreaks 54  
     TRVAControlPanel 54

ShowTableSizeDialog 282  
     TrvActionInsertTable 282

ShowUntranslatedControls 390

SkinType 85  
     TCustomRuler 85

SmartHeadings 90  
     TRVStyleTemplateComboBox 90  
     TRVStyleTemplateListBox 90

SpaceAfter 203  
     TrvActionParagraph 203

SpaceBefore 204  
     TrvActionParagraph 204

SpaceFiller 238  
     TrvActionInsertHyperlink 238

Spacing 253  
     TrvActionInsertPicture 253

SpellInterface 54  
     TRVAControlPanel 54

SpTBXLib 370

StandardStyleTemplates 214  
     TrvActionStyleTemplates 214

StoreFileNameInItemName 138, 253  
     TrvActionInsertPicture 253  
     TrvActionPasteSpecial 138

StoreImageFileName 328  
     TrvActionItemProperties 328

Style 232, 238  
     TrvActionInsertHLine 232  
     TrvActionInsertHyperlink 238

Style Templates in RichViewActions 20

StyleTemplateName 239, 260  
     TrvActionCustomInsertSidenote 260  
     TrvActionInsertHyperlink 239

StyleTemplates 104  
     TrvActionNew 104

SubDocEditor 337  
     TrvCustomEditorAction 337

SubSuperScriptType 153  
     TrvActionFontEx 153

SuppressNextErrorMessage 125  
     TrvActionSave 125

SymbolPreviewString 76, 78  
     TRVFontComboBox 76

TRVFontListBox 78

## - T -

TableGridStyle 54  
     TRVAControlPanel 54

TableOptions 280  
     TrvActionInsertTable 280

TablePrintOptions 280  
     TrvActionInsertTable 280

Tabs 204  
     TrvActionParagraph 204

TabsToDelete 204  
     TrvActionParagraph 204

TBiDiModeRuler 82

TBX 371

TCustomRuler.BiDiModeRuler 82

TCustomRuler.Flat 83

TCustomRuler.IndentColor 83

TCustomRuler.IndentSaturation 84

TCustomRuler.MarginColor 83

TCustomRuler.MarginSaturation 84

TCustomRuler.RulerColor 83

TCustomRuler.RulerSaturation 84

TCustomRuler.RulerType 84

TCustomRuler.ScreenRes 84

TCustomRuler.SkinType 85

TCustomRuler.UnitsDisplay 85

TCustomRuler.UnitsPixelsPerInch 85

TCustomRuler.UnitsProgram 85

TCustomRuler.Zoom 86

TCustomRulerItemSelector.Ruler 38

TCustomRVFontComboBox 33

TCustomRVFontComboBox.ActionFont 34

TCustomRVFontComboBox.Editor 35

TCustomRVFontListBox 35

TCustomRVFontListBox.ActionFont 36

TCustomRVFontListBox.AllowFocusEditor 36

TCustomRVFontListBox.Editor 37

TerminateHyperlink 243  
     TrvActionInsertHyperlink 243

TextStyleImageIndex 89, 215  
     TrvActionStyleTemplates 215  
     TRVStyleTemplateComboBox 89  
     TRVStyleTemplateListBox 89

Title 118, 122  
     TrvActionPrint 118  
     TrvActionQuickPrint 122

TNT Controls 370

- Toolbar 2000 371
- TRichView toolbar images 372
- TRulerType 84
- TRulerUnits 384
- TRVA2LiveSpellAddEvent type 69
- TRVA2LiveSpellGetSuggestionsEvent type 70
- TRVA2LiveSpellIgnoreAllEvent 70
- TRVA2LiveSpellReplaceEvent type 71
- TRVAAddictSpellInterface 356
- TRVAASpellInterface 357
- TRVABoxPosition 259
- TRVABoxProperties 260
- TRVACanPasteEvent 139
- TRVACIDownloadInterface 353
- TrvaColorInterface 320
- TRVAControlPanel 38
- TRVAControlPanel.ActionsEnabled 41
- TRVAControlPanel.Activate 59
- TRVAControlPanel.AddColorNameToHints 41
- TRVAControlPanel.AutoDeleteUnusedStyles 42
- TRVAControlPanel.ColorDialog 42
- TRVAControlPanel.ColorDialogInterface 42
- TRVAControlPanel.DefaultColor 42
- TRVAControlPanel.DefaultControl 43
- TRVAControlPanel.DefaultCustomFilterIndex 43
- TRVAControlPanel.DefaultDocParameters 43
- TRVAControlPanel.DefaultExt 44
- TRVAControlPanel.DefaultFileFormat 44
- TRVAControlPanel.DefaultFileName 44
- TRVAControlPanel.DefaultMargin 45
- TRVAControlPanel.DialogFontName 45
- TRVAControlPanel.DialogFontNameLin 45
- TRVAControlPanel.DialogFontSize 45
- TRVAControlPanel.DialogPosition 46
- TRVAControlPanel.DialogZoomPercent 46
- TRVAControlPanel.DonImportPicturesAndFiles 59
- TRVAControlPanel.DownloadInterface 46
- TRVAControlPanel.FirstPageNumber 47
- TRVAControlPanel.Footer 47
- TRVAControlPanel.GetRealDialogFontName 59
- TRVAControlPanel.Header 47
- TRVAControlPanel.HelpType 47
- TRVAControlPanel.HTMLComponent 51
- TRVAControlPanel.InitImportPicturesAndFiles 59
- TRVAControlPanel.Language 51
- TRVAControlPanel.MetafileCompatibility 51
- TRVAControlPanel.OnAddStyle 60
- TRVAControlPanel.OnBackgroundChange 61
- TRVAControlPanel.OnChoosePicture 61
- TRVAControlPanel.OnCreateForm 62
- TRVAControlPanel.OnCustomFileOperation 62
- TRVAControlPanel.OnDownload 64
- TRVAControlPanel.OnGetActionControlCoords 64
- TRVAControlPanel.OnGetHeaderFooterCode 65
- TRVAControlPanel.OnMarginsChanged 65
- TRVAControlPanel.OnStyleNeeded 66
- TRVAControlPanel.OnViewChanged 66
- TRVAControlPanel.PixelBorders 52
- TRVAControlPanel.RVFFilter 52
- TRVAControlPanel.RVFLocalizable 52
- TRVAControlPanel.RVFormatTitle 53
- TRVAControlPanel.RVPrint 53
- TRVAControlPanel.RVStylesExt 53
- TRVAControlPanel.RVStylesFilter 53
- TRVAControlPanel.SearchScope 53
- TRVAControlPanel.ShowSoftPageBreaks 54
- TRVAControlPanel.SpellInterface 54
- TRVAControlPanel.TableGridStyle 54
- TRVAControlPanel.UnitsDisplay 55
- TRVAControlPanel.UnitsProgram 55
- TRVAControlPanel.UseDefaultHelpFile 56
- TRVAControlPanel.UseHelpFiles 56
- TRVAControlPanel.UserInterface 56
- TRVAControlPanel.UseTextCodePageDialog 57
- TRVAControlPanel.UseXPThemes 57
- TRVAControlPanel.XMLComponent 58
- TRVAControlPanel.XMLFilter 58
- TRVAControlPanel.XMLLocalizable 58
- TrvAction 311
- TrvAction.Control 312
- TrvActionAddStyleTemplate 215
- TrvActionAddStyleTemplate.ActionStyleTemplates 216
- TrvActionAlignCenter 174
- TrvActionAlignCustomJustify 175
- TrvActionAlignCustomJustify.LastLineAlignment 176
- TrvActionAlignCustomJustify.UseLastLineAlignment 176
- TrvActionAlignDistribute 176
- TrvActionAlignJustify 177
- TrvActionAlignLeft 178
- TrvActionAlignment 178
- TrvActionAlignRight 179
- TrvActionBackground 313
- TrvActionBackground.CanChangeMargins 314
- TrvActionBackground.ImageFileName 315
- TrvActionBackground.OnChange 315
- TrvActionBookmarks 221
- TrvActionBookmarks.AllowedCharacters 222
- TrvActionBookmarks.ScrollToCenter 222

- TrvActionCharCase 130
- TrvActionClearBoth 180
- TrvActionClearFormat 219
- TrvActionClearLeft 180
- TrvActionClearNone 181
- TrvActionClearRight 181
- TrvActionClearTextFlow 182
- TrvActionClearTextFormat 219
- TrvActionColor 315
- TrvActionCopy 131
- TrvActionCustomColor 318
- TrvActionCustomColor.CallerControl 319
- TrvActionCustomColor.Color 319
- TrvActionCustomColor.OnHideColorPicker 321
- TrvActionCustomColor.OnShowColorPicker 321
- TrvActionCustomColor.Opacity 320
- TrvActionCustomColor.UserInterface 320
- TrvActionCustomFileIO 96
- TrvActionCustomFileIO.CustomFilter 98
- TrvActionCustomInfoWindow 216
- TrvActionCustomInfoWindow.OnShowing 219
- TrvActionCustomInsertSidenote 258
- TrvActionCustomInsertSidenote.BoxPosition 259
- TrvActionCustomInsertSidenote.BoxProperties 260
- TrvActionCustomInsertSidenote.StyleTemplateName 260
- TrvActionCustomIO 98
- TrvActionCustomIO.AutoUpdateFileName 99
- TrvActionCustomIO.DialogTitle 100
- TrvActionCustomIO.FileName 100
- TrvActionCustomIO.InitialDir 100
- TrvActionCustomNew 101
- TrvActionCustomNew.ActionSave 102
- TrvActionCustomNew.OnNew 101
- TrvActionCustomParaListSwitcher 182
- TrvActionCustomParaListSwitcher.IndentStep 184
- TrvActionCustomParaListSwitcher.ListLevels 184
- TrvActionCut 131
- TrvActionEditNote 260
- TrvActionEditNote.JumpToNextNote 262
- TrvActionEditNote.OnStartEditNote 263
- TrvActionEvent 321
- TrvActionEvent.OnExecute 322
- TrvActionEvent.OnUpdate 322
- TrvActionExport 110
- TrvActionExport.CreateDirectoryForHTMLImages 113
- TrvActionExport.ExportToFile 114
- TrvActionExport.Filter 113
- TrvActionFillColor 322
- TrvActionFillColor.AllowApplyingTo 324
- TrvActionFind 132
- TrvActionFind.ActionReplace 133
- TrvActionFind.CloseDialog 134
- TrvActionFind.FindText 134
- TrvActionFindNext 134
- TrvActionFindNext.ActionFind 135
- TrvActionFontAllCaps 146
- TrvActionFontBackColor 147
- TrvActionFontBold 147
- TrvActionFontColor 148
- TrvActionFontCustomColor 149
- TrvActionFontEx 149
- TrvActionFontEx.AutoApplySymbolCharset 151
- TrvActionFontEx.BackColor 152
- TrvActionFontEx.CharScale 152
- TrvActionFontEx.CharSpacing 152
- TrvActionFontEx.FontSizeDouble 153
- TrvActionFontEx.FontStyleEx 153
- TrvActionFontEx.OnShowingDialog 155
- TrvActionFontEx.PreviewInList 153
- TrvActionFontEx.SubSuperScriptType 153
- TrvActionFontEx.UnderlineColor 154
- TrvActionFontEx.UnderlineType 154
- TrvActionFontEx.ValidProperties 154
- TrvActionFontEx.VShift 155
- TrvActionFontGrow 155
- TrvActionFontGrow.MaxSize 157
- TrvActionFontGrowOnePoint 157
- TrvActionFontGrowOnePoint.MaxSize 158
- TrvActionFontItalic 158
- TrvActionFontOverline 159
- TrvActionFonts 159
- TrvActionFonts.Font 161
- TrvActionFonts.UserInterface 161
- TrvActionFontShrink 161
- TrvActionFontShrink.MinSize 163
- TrvActionFontShrinkGrow 163
- TrvActionFontShrinkGrow.Percent 164
- TrvActionFontShrinkOnePoint 164
- TrvActionFontShrinkOnePoint.MinSize 165
- TrvActionFontStrikeout 166
- TrvActionFontStyle 166
- TrvActionFontStyleEx 167
- TrvActionFontUnderline 168
- TrvActionHide 324
- TrvActionIndent 184
- TrvActionIndent.IndentStep 186
- TrvActionIndentDec 186

TrvActionIndentInc	187	TrvActionInsertNote.ActionEditNote	266
TrvActionIndentInc.IndentMax	188	TrvActionInsertNote.Font	266
TrvActionInsertCaption	223	TrvActionInsertNumber	245
TrvActionInsertCaption.ExcludeLabel	224	TrvActionInsertNumSequence	246
TrvActionInsertCaption.InsertAbove	225	TrvActionInsertNumSequence.NumberType	247
TrvActionInsertCustomPageNumber	225	TrvActionInsertNumSequence.SeqName	247
TrvActionInsertCustomPageNumber.NumberType	226	TrvActionInsertNumSequence.UseDefaults	247
TrvActionInsertEndnote	263	TrvActionInsertPageBreak	248
TrvActionInsertEquation	226	TrvActionInsertPageCount	248
TrvActionInsertEquation.Expression	228	TrvActionInsertPageNumber	249
TrvActionInsertEquation.UserInterface	228	TrvActionInsertPicture	249
TrvActionInsertFile	228	TrvActionInsertPicture.BackgroundColor	251
TrvActionInsertFile.Filter	230	TrvActionInsertPicture.BorderColor	251
TrvActionInsertFile.InsertFile	230	TrvActionInsertPicture.BorderWidth	251
TrvActionInsertFootnote	264	TrvActionInsertPicture.DefaultExt	252
TrvActionInsertHLine	231	TrvActionInsertPicture.Filter	252
TrvActionInsertHLine.Color	232	TrvActionInsertPicture.MaxImageSize	252
TrvActionInsertHLine.Style	232	TrvActionInsertPicture.OnInserting	254
TrvActionInsertHLine.Width	232	TrvActionInsertPicture.OuterHSpacing	253
TrvActionInsertHyperlink	232	TrvActionInsertPicture.OuterVSpacing	253
TrvActionInsertHyperlink.ActionStyleTemplates	235	TrvActionInsertPicture.Spacing	253
TrvActionInsertHyperlink.AutoAddHyperlinkStyleTemplate	235	TrvActionInsertPicture.StoreFileNameInItemName	253
TrvActionInsertHyperlink.BackColor	235	TrvActionInsertPicture.VAlign	253
TrvActionInsertHyperlink.Color	236	TrvActionInsertSidenote	267
TrvActionInsertHyperlink.Cursor	236	TrvActionInsertSidenote.RefStyleTemplateName	269
TrvActionInsertHyperlink.DetectURL	240	TrvActionInsertSymbol	254
TrvActionInsertHyperlink.EncodeTarget	241	TrvActionInsertSymbol.CharCode	255
TrvActionInsertHyperlink.GetHyperlinkStyleNo	241	TrvActionInsertSymbol.FontName	256
TrvActionInsertHyperlink.GetNormalStyleNo	242	TrvActionInsertSymbol.Protection	256
TrvActionInsertHyperlink.GoToLink	242	TrvActionInsertSymbol.UseCurrentFont	256
TrvActionInsertHyperlink HoverBackColor	236	TrvActionInsertTable	271
TrvActionInsertHyperlink HoverColor	237	TrvActionInsertTable.BackgroundImage	273
TrvActionInsertHyperlink HoverEffects	237	TrvActionInsertTable.BackgroundStyle	273
TrvActionInsertHyperlink HoverUnderlineColor	237	TrvActionInsertTable.BestWidth	274
TrvActionInsertHyperlink.OnApplyHyperlinkToItem	244	TrvActionInsertTable.BorderColor	274
TrvActionInsertHyperlink.OnGetHyperlinkTargetFromItem	244	TrvActionInsertTable.BorderHSpacing	274
TrvActionInsertHyperlink.OnHyperlinkForm	244	TrvActionInsertTable.BorderLightColor	274
TrvActionInsertHyperlink.ScrollToCenter	238	TrvActionInsertTable.BorderStyle	275
TrvActionInsertHyperlink.SetTags	243	TrvActionInsertTable.BorderVSpacing	275
TrvActionInsertHyperlink.SetToPictures	238	TrvActionInsertTable.BorderWidth	275
TrvActionInsertHyperlink.SpaceFiller	238	TrvActionInsertTable.CellBorderColor	275
TrvActionInsertHyperlink.Style	238	TrvActionInsertTable.CellBorderLightColor	276
TrvActionInsertHyperlink.StyleTemplateName	239	TrvActionInsertTable.CellBorderStyle	276
TrvActionInsertHyperlink.TerminateHyperlink	243	TrvActionInsertTable.CellBorderWidth	276
TrvActionInsertHyperlink.UnderlineColor	239	TrvActionInsertTable.CellHPadding	276
TrvActionInsertHyperlink.ValidProperties	239	TrvActionInsertTable.CellHSpacing	277
TrvActionInsertNote	265	TrvActionInsertTable.CellPadding	276
		TrvActionInsertTable.CellVPadding	276
		TrvActionInsertTable.CellVSpacing	277

- TrvActionInsertTable.ColBandSize 277
- TrvActionInsertTable.ColCount 278
- TrvActionInsertTable.Color 278
- TrvActionInsertTable.EvenColumnsColor 281
- TrvActionInsertTable.EvenRowsColor 281
- TrvActionInsertTable.FirstColumnColor 281
- TrvActionInsertTable.HeadingRowColor 281
- TrvActionInsertTable.HeadingRowCount 278
- TrvActionInsertTable.HOutermostRule 278
- TrvActionInsertTable.HRuleColor 278
- TrvActionInsertTable.HRuleWidth 279
- TrvActionInsertTable.ItemText 279
- TrvActionInsertTable.LastColumnColor 281
- TrvActionInsertTable.LastRowColor 281
- TrvActionInsertTable.OddColumnsColor 281
- TrvActionInsertTable.OddRowsColor 281
- TrvActionInsertTable.OnCloseTableSizeDialog 282
- TrvActionInsertTable.OnInserting 282
- TrvActionInsertTable.RowBandSize 277
- TrvActionInsertTable.RowCount 279
- TrvActionInsertTable.RowsKeepTogether 279
- TrvActionInsertTable.RowsVAlign 280
- TrvActionInsertTable.ShowTableSizeDialog 282
- TrvActionInsertTable.TableOptions 280
- TrvActionInsertTable.TablePrintOptions 280
- TrvActionInsertTable.VisibleBorders 280
- TrvActionInsertTable.VOutermostRule 280
- TrvActionInsertTable.VRuleColor 281
- TrvActionInsertTable.VRuleWidth 281
- TrvActionInsertText 256
- TrvActionInsertText.OnInsertText 257
- TrvActionInsertTextBox 269
- TrvActionItemProperties 324
- TrvActionItemProperties.ActionInsertHLine 326
- TrvActionItemProperties.ActionInsertTable 326
- TrvActionItemProperties.BackgroundGraphicFilter 327
- TrvActionItemProperties.DefaultChecked 327
- TrvActionItemProperties.DefaultPersistent 327
- TrvActionItemProperties.GraphicFilter 328
- TrvActionItemProperties.OnCanApply 328
- TrvActionItemProperties.OnCustomItemPropertiesDialog 329
- TrvActionItemProperties.StoreImageFileName 328
- TrvActionItemProperties.UpdateAllInsertTableActions 326
- TrvActionLineSpacing 188
- TrvActionLineSpacing100 189
- TrvActionLineSpacing150 189
- TrvActionLineSpacing200 190
- TrvActionNew 103
- TrvActionNew.Reset 105
- TrvActionNew.StyleTemplates 104
- TrvActionOpen 106
- TrvActionOpen.ActionNew 108
- TrvActionOpen.CustomFilter 108
- TrvActionOpen.DialogTitle 108
- TrvActionOpen.Filter 108
- TrvActionOpen.InitialDir 109
- TrvActionOpen.LastFilterIndex 109
- TrvActionOpen.LoadFile 109
- TrvActionOpen.OnOpenFile 110
- TrvActionPageSetup 114
- TrvActionPageSetup.MarginsUnits 115
- TrvActionPageSetup.OnChange 116
- TrvActionPageSetup.UnitsPixelsPerInch 116
- TrvActionParaBiDi 191
- TrvActionParaBorder 191
- TrvActionParaBorder.Background 193
- TrvActionParaBorder.Border 193
- TrvActionParaBorder.OnShowingDialog 194
- TrvActionParaBorder.UserInterface 194
- TrvActionParaBorder.ValidProperties 194
- TrvActionParaBullets 195
- TrvActionParaColor 195
- TrvActionParaColorAndPadding 196
- TrvActionParaColorAndPadding.Padding 197
- TrvActionParaColorAndPadding.UsePadding 198
- TrvActionParaCustomColor 198
- TrvActionParagraph 198
- TrvActionParagraph.Alignment 200, 202
- TrvActionParagraph.DeleteAllTabs 200
- TrvActionParagraph.FirstIndent 201
- TrvActionParagraph.KeepLinesTogether 201
- TrvActionParagraph.KeepWithNext 201
- TrvActionParagraph.LastLineAlignment 202
- TrvActionParagraph.LeftIndent 202
- TrvActionParagraph.LineSpacing 202
- TrvActionParagraph.LineSpacingType 203
- TrvActionParagraph.OnShowingDialog 205
- TrvActionParagraph.OutlineLevel 203
- TrvActionParagraph.RightIndent 203
- TrvActionParagraph.SpaceAfter 203
- TrvActionParagraph.SpaceBefore 204
- TrvActionParagraph.Tabs 204
- TrvActionParagraph.TabsToDelete 204
- TrvActionParagraph.UserInterface 205
- TrvActionParagraph.ValidProperties 205
- TrvActionParaList 206

TrvActionParaList.ActionParaBullets	208	TrvActionSave.SuppressNextErrorMessage	125
TrvActionParaList.ActionParaNumbering	208	TrvActionSaveAs	127
TrvActionParaList.IndentStep	207	TrvActionSaveAs.ActionSave	129
TrvActionParaList.UpdateAllActionsOnForm	207	TrvActionSaveAs.Filter	129
TrvActionParaLTR	208	TrvActionSelectAll	144
TrvActionParaNumbering	209	TrvActionShowSpecialCharacters	331
TrvActionParaRTL	210	TrvActionShowSpecialCharacters.ShowCheckpoints	332
TrvActionParaStyles	210	TrvActionSpellingCheck	310
TrvActionPaste	135	TrvActionSSScript	168
TrvActionPasteAsText	136	TrvActionStyleInspector	216
TrvActionPasteSpecial	137	TrvActionStyleInspector.FontImageIndex	218
TrvActionPasteSpecial.AddFormat	139	TrvActionStyleInspector.Images	218
TrvActionPasteSpecial.OnCanPaste	139	TrvActionStyleInspector.ParImageIndex	218
TrvActionPasteSpecial.OnCustomPaste	140	TrvActionStyleInspector.UpdateInfo	218
TrvActionPasteSpecial.OnShowing	140	TrvActionStyleTemplates	212
TrvActionPasteSpecial.StoreFileNameInItemName	138	TrvActionStyleTemplates.Images	214
TrvActionPrint	116	TrvActionStyleTemplates.ParImageIndex	215
TrvActionPrint.Title	118	TrvActionStyleTemplates.ParTextStyleImageIndex	215
TrvActionPrintPreview	118	TrvActionStyleTemplates.StandardStyleTemplates	214
TrvActionPrintPreview.ActionPageSetup	119	TrvActionStyleTemplates.TextStyleImageIndex	215
TrvActionPrintPreview.ActionPrint	119	TrvActionSubscript	169
TrvActionPrintPreview.Maximized	120	TrvActionSuperscript	170
TrvActionPrintPreview.OnGetPreviewFormClass	120	TrvActionTableCell	282
TrvActionQuickPrint	120	TrvActionTableCell.AllBorders	283
TrvActionQuickPrint.Title	122	TrvActionTableCell.Border	284
TrvActionRedo	140	TrvActionTableCell.BottomBorder	284
TrvActionRemoveHyperlinks	329	TrvActionTableCell.LeftBorder	285
TrvActionRemoveHyperlinks.ActionInsertHyperlink	330	TrvActionTableCell.NoBorders	286
TrvActionRemovePageBreak	330	TrvActionTableCell.RightBorder	287
TrvActionReplace	141	TrvActionTableCell.Rotation	287
TrvActionReplace.ActionFind	142	TrvActionTableCell.Rotation180	289
TrvActionReplace.CloseDialog	143	TrvActionTableCell.Rotation270	290
TrvActionReplace.FindText	142	TrvActionTableCell.Rotation90	289
TrvActionReplace.OnReplaceAllEnd	143	TrvActionTableCell.RotationNone	288
TrvActionReplace.OnReplaceAllStart	143	TrvActionTableCell.TopBorder	291
TrvActionReplace.OnReplacing	144	TrvActionTableCell.VAlign	291
TrvActionReplace.ReplaceText	142	TrvActionTableCell.VAlignBottom	292
TrvActionReplace.ShowReplaceAllSummary	143	TrvActionTableCell.VAlignDefault	293
TrvActionSave	122	TrvActionTableCell.VAlignMiddle	293
TrvActionSave.ActionSaveAs	123	TrvActionTableCell.VAlignTop	294
TrvActionSave.CanCloseDoc	125	TrvActionTableDeleteCols	295
TrvActionSave.CreateDirectoryForHTMLImages	124	TrvActionTableDeleteRows	295
TrvActionSave.DisableWhenUnmodified	124	TrvActionTableDeleteTable	296
TrvActionSave.Documents	124	TrvActionTableGrid	296
TrvActionSave.FindDoc	126	TrvActionTableInsertColLeft	297
TrvActionSave.LostFormatWarning	125	TrvActionTableInsertColRight	297
TrvActionSave.OnDocumentFileChange	126	TrvActionTableInsertRowsAbove	298
TrvActionSave.OnSave	126	TrvActionTableInsertRowsBelow	299
TrvActionSave.OnSaving	126	TrvActionTableInsertRowsBelow.AllowMultiple	300



- TrvActionTableMergeCells 300
- TrvActionTableMultiCellAttributes 301
- TrvActionTableProperties 301
- TrvActionTableProperties.ActionInsertTable 303
- TrvActionTableProperties.BackgroundGraphicFilter 303
- TrvActionTableProperties.DefaultChecked 303
- TrvActionTableProperties.DefaultPersistent 303
- TrvActionTableProperties.UpdateAllInsertTableActions 303
- TrvActionTableRCBase 304
- TrvActionTableSelectCell 305
- TrvActionTableSelectCols 305
- TrvActionTableSelectRows 306
- TrvActionTableSelectTable 306
- TrvActionTableSort 307
- TrvActionTableSplit 308
- TrvActionTableSplitCells 308
- TrvActionTableToText 309
- TrvActionTextBiDi 170
- TrvActionTextLTR 171
- TrvActionTextRTL 172
- TrvActionTextStyles 173
- TrvActionThesaurus 310
- TrvActionUndo 144
- TrvActionUserDefinedColor 316
- TrvActionUserDefinedColor.OnColorSelected 317
- TrvActionUserDefinedColor.OnGetColor 318
- TrvActionUserDefinedColor.UseOpacity 317
- TrvActionVAlign 333
- TrvActionWordWrap 211
- TRVACustomDownloadInterface 354
- TRVACustomPasteEvent 140
- TRVACustomSpellInterface 357
- TRVAddStyleEvent 60
- TrvaDocumentInfo 124
- TRVADownloadEvent 64
- TRVADXSpellInterface 358
- TRVAEditEvent 384
- TRVAEditorControlCommand 389
- TRVAEditorControlCommandFunction 389
- TRVAEvent 322
- TRVAFileOperation 62
- TRVAFillColorApplyToElement 324
- TRVAFillColorApplyToSet 324
- TRVAFormPosition 46
- TRVAHFInfo 384
- TRVAHunSpellInterface 359
- TRVAIndyDownloadInterface 354
- TRVInsertPictureEvent 254
- TRVInsertTableEvent 282
- TRVInsertTextEvent 257
- TRVALanguageName 348
- TRVALiveSpellAddEvent type 69
- TRVALiveSpellGetSuggestionsEvent type 70
- TRVALiveSpellIgnoreAllEvent type 70
- TRVALiveSpellReplaceEvent type 71
- TRVALocString 348
- TRVAMessageID 346
- TRVAPolarSpellInterface 359
- TRVAPopupActionBar 66
- TRVAPopupMenu 66
- TRVAPopupMenu.ActionList 69
- TRVAPopupMenu.MaxSuggestionsCount 69
- TRVAPopupMenu.OnLiveSpellAdd 69
- TRVAPopupMenu.OnLiveSpellGetSuggestions 70
- TRVAPopupMenu.OnLiveSpellIgnoreAll 70
- TRVAPopupMenu.OnLiveSpellWordReplace 71
- TRVApplyHyperlinkToItemEvent 244
- TRVReplacingEvent 144
- TRVSearchScope 53
- TRVSpellString 385
- TRVSpellStringList 385
- TRVSpellStrings 385
- TRVASPTBXPopupMenu 66
- TRVASPTBXPopupMenu.ActionList 69
- TRVASPTBXPopupMenu.MaxSuggestionsCount 69
- TRVASPTBXPopupMenu.OnLiveSpellAdd 69
- TRVASPTBXPopupMenu.OnLiveSpellGetSuggestions 70
- TRVASPTBXPopupMenu.OnLiveSpellIgnoreAll 70
- TRVASPTBXPopupMenu.OnLiveSpellWordReplace 71
- TRVATBPopupMenu 66
- TRVATBPopupMenu.ActionList 69
- TRVATBPopupMenu.MaxSuggestionsCount 69
- TRVATBPopupMenu.OnLiveSpellAdd 69
- TRVATBPopupMenu.OnLiveSpellGetSuggestions 70
- TRVATBPopupMenu.OnLiveSpellIgnoreAll 70
- TRVATBPopupMenu.OnLiveSpellWordReplace 71
- TRVATBXPopupMenu 66
- TRVATBXPopupMenu.ActionList 69
- TRVATBXPopupMenu.MaxSuggestionsCount 69
- TRVATBXPopupMenu.OnLiveSpellAdd 69
- TRVATBXPopupMenu.OnLiveSpellGetSuggestions 70
- TRVATBXPopupMenu.OnLiveSpellIgnoreAll 70
- TRVATBXPopupMenu.OnLiveSpellWordReplace 71
- TRVAUserInterface 56
- TRVCanApplyEvent 328
- TRVCharCase 383
- TRVChoosePictureEvent 61

TRVCreateFormEvent	62	TRVHyperlinkFormEvent	244
TrvCustomAction	333	TRVHyperlinkProperties	239
TrvCustomAction.Caption	334	TRVHyperlinkProperty	239
TrvCustomAction.ControlPanel	335	TrvPaperMarginsUnits	115
TrvCustomAction.Disabled	335	TRVParaInfoBorderProperties	194
TrvCustomAction.GetControlPanel	336	TRVParaInfoBorderProperty	194
TrvCustomAction.Hint	335	TRVParaInfoMainProperties	205
TrvCustomEditAction	145	TRVParaInfoMainProperty	205
TrvCustomEditorAction	336	TRVRuler	80
TrvCustomEditorAction.Form	337	TRVRuler.RichViewEdit	86
TrvCustomEditorAction.OnFormCreate	338	TRVRuler.UpdateRulerMargins	87
TrvCustomEditorAction.OnHide	338	TRVRulerItemSelector	37
TrvCustomEditorAction.OnShowing	338	TRVSaveFileEvent	126
TrvCustomEditorAction.SubDocEditor	337	TRVShowFormEvent	219, 388
TRVCustomFileOperationEvent	62	TRVSpellInterface	356
TRVCustomItemPropertiesDialog	329	TRVStyleNeededEvent	66
TrvCustomPrintAction	129	TRVStyleTemplateComboBox	87
TrvFileExportFilter	386	TRVStyleTemplateComboBox.AllStylesImageIndex	89
TrvFileExportFilterSet	386	TRVStyleTemplateComboBox.ClearFormatImageIndex	89
TRVFileFormatComponent	387	TRVStyleTemplateComboBox.ControlPanel	89
TrvFileImportFilter	387	TRVStyleTemplateComboBox.Editor	89
TrvFileImportFilterSet	387	TRVStyleTemplateComboBox.Images	90
TrvFileOpenFilter	387	TRVStyleTemplateComboBox.Localize	91
TrvFileOpenFilterSet	387	TRVStyleTemplateComboBox.OnClickAllStyles	92
TrvFileSaveFilter	386	TRVStyleTemplateComboBox.ParaStyleImageIndex	89
TrvFileSaveFilterSet	386	TRVStyleTemplateComboBox.ParaTextStyleImageIndex	89
TRVFontCharsetComboBox	71	TRVStyleTemplateComboBox.ShowAllStyles	90
TRVFontCharsetComboBox.AddDefaultCharset	73	TRVStyleTemplateComboBox.ShowClearFormat	90
TRVFontCharsetComboBox.DefaultCharsetCaption	73	TRVStyleTemplateComboBox.SmartHeadings	90
TRVFontCharsetComboBox.FontName	73	TRVStyleTemplateComboBox.TextStyleImageIndex	89
TRVFontComboBox	73	TRVStyleTemplateListBox.AllStylesImageIndex	89
TRVFontComboBox.AutoCharset	75	TRVStyleTemplateListBox.ClearFormatImageIndex	89
TRVFontComboBox.DropDownWidth	75	TRVStyleTemplateListBox.ControlPanel	89
TRVFontComboBox.ItemHeight	75	TRVStyleTemplateListBox.Editor	89
TRVFontComboBox.Preview	76	TRVStyleTemplateListBox.Images	90
TRVFontComboBox.SymbolPreviewString	76	TRVStyleTemplateListBox.Localize	91
TRVFontInfoMainProperties	154	TRVStyleTemplateListBox.OnClickAllStyles	92
TRVFontInfoMainProperty	154	TRVStyleTemplateListBox.ParaStyleImageIndex	89
TRVFontListBox	76	TRVStyleTemplateListBox.ParaTextStyleImageIndex	89
TRVFontListBox.AutoCharset	78	TRVStyleTemplateListBox.ShowAllStyles	90
TRVFontListBox.ItemHeight	78	TRVStyleTemplateListBox.ShowClearFormat	90
TRVFontListBox.Preview	78	TRVStyleTemplateListBox.SmartHeadings	90
TRVFontListBox.SymbolPreviewString	78	TRVStyleTemplateListBox.TextStyleImageIndex	89
TRVFontSizeComboBox	79	TRVXMLComponent	387
TRVFontSizeComboBox.FontName	80	TSkinType	85
TRVGetFormClassEvent	120	TZoomRange	86
TRVGetHyperlinkTargetFromItem	244		
TRVHeaderFooterCodeEvent	65		
TRVHTMLComponent	387		

## - U -

- UnderlineColor 154, 239
  - TrvActionFontEx 154
  - TrvActionInsertHyperlink 239
- UnderlineType 154
  - TrvActionFontEx 154
- UnitsDisplay 55, 85
  - TCustomRuler 85
  - TRVAControlPanel 55
- UnitsPixelsPerInch 85, 116
  - TCustomRuler 85
  - TrvActionPageSetup 116
- UnitsProgram 55, 85
  - TCustomRuler 85
  - TRVAControlPanel 55
- UpdateAllActionsOnForm 207
  - TrvActionParaList 207
- UpdateAllInsertTableActions 303, 326
  - TrvActionTableProperties 303
- UpdateAllInsertTableActions.TrvActionItemProperties 326
- UpdateInfo 218
  - TrvActionStyleInspector 218
- UpdateRulerMargins 87
  - TRVRuler 87
- UseCurrentFont 256
  - TrvActionInsertSymbol 256
- UseDefaultHelpFile 56
  - TRVAControlPanel 56
- UseDefaults 247
  - TrvActionInsertNumSequence 247
- UseHelpFiles 56
  - TRVAControlPanel 56
- UseLastLineAlignment 176
  - TrvActionAlignCustomJustify 176
- UseOpacity 317
  - TrvActionUserDefinedColor 317
- UsePadding 198
  - TrvActionParaColorAndPadding 198
- UserInterface 56, 161, 194, 205, 228, 320
  - TRVAControlPanel 56
  - TrvActionCustomColor 320
  - TrvActionFonts 161
  - TrvActionInsertEquation 228
  - TrvActionParaBorder 194
  - TrvActionParagraph 205
- UseTextCodePageDialog 57
  - TRVAControlPanel 57

- UseXPThemes 57
  - TRVAControlPanel 57
- Using RichViewActions 16

## - V -

- ValidProperties 154, 194, 205, 239
  - TrvActionFontEx 154
  - TrvActionInsertHyperlink 239
  - TrvActionParaBorder 194
  - TrvActionParagraph 205
- VAlign 253
  - TrvActionInsertPicture 253
- VisibleBorders 280
  - TrvActionInsertTable 280
- VOutermostRule 280
  - TrvActionInsertTable 280
- VRuleColor 281
  - TrvActionInsertTable 281
- VRuleWidth 281
  - TrvActionInsertTable 281
- VShift 155
  - TrvActionFontEx 155

## - W -

- Width 232
  - TrvActionInsertHLine 232

## - X -

- XMLComponent 58
  - TRVAControlPanel 58
- XMLFilter 58
  - TRVAControlPanel 58
- XMLLocalizable 58
  - TRVAControlPanel 58

## - Z -

- Zoom 86
  - TCustomRuler 86