

TRichView Components

TRichView © trichview.com

Table of Contents

Part I TRichView Reference 27

1..Version History 29

New after version 17 29
New in version 17 30
New in version 16 32
New in version 15 34
New in version 14 36
New in version 13 39
New in version 12 41
New in version 11 42
New in version 10 44
New in version 1.9 48
New in version 1.8 50
New in version 1.7 52
New in version 1.6 55
New in version 1.5 56
New in version 1.4 58
New in version 1.3 60
New in version 1.2 60
Compatibility with older versions 62

2..Overview 63

Parts of Documents 64
TRichView Item Types 64
"Checkpoints" 65
Items' "Tags" 68
Hypertext Overview 69
Paragraphs 72
Borders and Background of Paragraphs 73
Styles and StyleTemplates 74
Hidden Text 76
Basic Operations 77
Building a Document 77
Scrolling 80
Selecting 81
Clipboard 82
Searching and Replacing 83
Obtaining Items 84
Modifying Items 85
Editing 86
RichView Edit: Inserting Items in Position of Caret 86
Undo and Redo 88
Drag and Drop 89
Graphics 90
Animated Images 90
Smooth Image Scaling 91
Semitransparent Objects 91
Saving and Loading 92
Saving and Loading 92
RVF (RichView Format) 93
Export to HTML 96

Internationalization ... **99**
Unicode in TRichView ... 99
Bidirectional text ... 100
Spelling Check ... **101**
Live spelling check ... 101
Important Technical Information ... **102**
Viewer vs Editor ... 102
Controls - Documents - Items ... 104
Valid Documents ... 105
Units of Measurement ... 106
Custom drawing ... 108
Cursors ... 108
RVF Specification ... 109
3 Overview of Item Types ... **118**
Overview ... **118**
Text Items ... **120**
Tabulators ... **121**
Pictures ... **122**
Hot-Pictures ... **125**
Breaks ... **125**
Controls ... **126**
Bullets ... **128**
Hotspots ... **129**
Tables ... **130**
List Markers ... **131**
Labels ... **133**
Numbered Sequences ... **134**
Endnotes ... **135**
Footnotes ... **136**
Sidenotes ... **137**
Text Boxes ... **139**
References to Notes ... **140**
Page Numbers ... **141**
Page Counts ... **142**
Mathematical Expressions ... **143**
Custom Item Types ... **144**
4 Overview of Tables ... **144**
Tables in RichView ... **145**
Addressing Items in Table Cells ... **145**
Table Cells ... **146**
Colors and Layout of Tables ... **148**
Selection in Table ... **150**
Editing Cells ... **151**
Cell Merging ... **152**
Table Operations ... **153**
Undo/redo in Tables ... **155**
Export of Tables ... **157**
Table Resizing ... **158**
Example: table to text ... **159**
Example: dividing table ... **160**
5 Components ... **161**
Visual Components ... **162**
TRichView ... 162
TRichView : Component editor ... 168
Properties ... 169
TRichView .AllowSelection ... 169
TRichView .AnimationMode ... 169

TRichView.BackgroundBitmap ... 169
TRichView.BackgroundStyle ... 170
TRichView.BiDiMode ... 170
TRichView.BottomMargin ... 171
TRichView.ClearLeft ... 171
TRichView.ClearRight ... 172
TRichView.Color ... 173
TRichView.CPEventKind ... 173
TRichView.Cursor ... 174
TRichView.Delimiters ... 174
TRichView.DocObjects ... 174
TRichView.DocParameters ... 176
TRichView.DocProperties ... 176
TRichView.DocumentHeight ... 177
TRichView.DoInPaletteMode ... 177
TRichView.FirstItemVisible ... 179
TRichView.FirstJumpNo ... 179
TRichView.ItemCount ... 180
TRichView.LastItemVisible ... 180
TRichView.LeftMargin ... 180
TRichView.LineCount ... 181
TRichView.MaxLength ... 181
TRichView.MaxTextWidth ... 181
TRichView.MinTextWidth ... 181
TRichView.NoteText ... 182
TRichView.Options ... 182
TRichView.PageBreaksBeforeItems ... 184
TRichView.RightMargin ... 185
TRichView.RTFOptions ... 185
TRichView.RTFReadProperties ... 187
TRichView.RVData ... 187
TRichView.RVFOptions ... 188
TRichView.RVFParaStylesReadMode ... 190
TRichView.RVFTextStylesReadMode ... 191
TRichView.RVFWarnings ... 191
TRichView.SelectionHandlesVisible ... 192
TRichView.SingleClick ... 192
TRichView.Style ... 193
TRichView.StyleTemplateInsertMode ... 193
TRichView.TabNavigation ... 194
TRichView.TopMargin ... 194
TRichView.UseStyleTemplates ... 195
TRichView.UseVCLThemes ... 195
TRichView.VAlign ... 195
TRichView.VSmallStep ... 196
TRichView.WordWrap ... 196
Methods ... 196
TRichView.AddBreak ... 196
TRichView.AddBullet ... 197
TRichView.AddControl ... 198
TRichView.AddFmt ... 198
TRichView.AddHotPicture ... 199
TRichView.AddHotspot ... 200
TRichView.AddItem ... 201
TRichView.AddCheckpoint ... 202
TRichView.AddNL -A -W ... 203
TRichView.AddPicture ... 204
TRichView.AddTab ... 205

TRichView.AddTextNL -A -W ... 206
TRichView.AppendFrom ... 207
TRichView.AppendRVFFromStream ... 208
TRichView.AssignSoftPageBreaks ... 209
TRichView.BeginOleDrag ... 209
TRichView.Clear ... 210
TRichView.ClearLiveSpellingResults ... 210
TRichView.ClearSoftPageBreaks ... 211
TRichView.ClientToDocument, DocumentToClient ... 211
TRichView.ConvertDocTo ... 211
TRichView.Copy ... 212
TRichView.CopyDef ... 212
TRichView.CopyImage ... 212
TRichView.CopyRTF ... 213
TRichView.CopyRVF ... 213
TRichView.CopyText -W ... 213
TRichView.Create ... 214
TRichView.DeleteItems ... 214
TRichView.DeleteMarkedStyles ... 215
TRichView.DeleteParas ... 215
TRichView.DeleteSection ... 215
TRichView.DeleteUnusedStyles ... 216
TRichView.Deselect ... 216
TRichView.Destroy ... 217
TRichView.FindCheckpointByName ... 217
TRichView.FindCheckpointByTag ... 217
TRichView.FindControlItemNo ... 217
TRichView.Format ... 217
TRichView.FormatTail ... 218
TRichView.GetBreakInfo ... 218
TRichView.GetBulletInfo ... 219
TRichView.GetCheckpointByNo ... 220
TRichView.GetCheckpointInfo ... 220
TRichView.GetCheckpointItemNo ... 221
TRichView.GetCheckpointNo ... 221
TRichView.GetCheckpointXY ... 221
TRichView.GetCheckpointY ... 221
TRichView.GetCheckpointYEx ... 222
TRichView.GetControlInfo ... 222
TRichView.GetFirstCheckpoint ... 223
TRichView.GetFocusedItem ... 223
TRichView.GetHotspotInfo ... 224
TRichView.GetItem ... 225
TRichView.GetItemAt ... 225
TRichView.GetItemCheckpoint ... 226
TRichView.GetItemCoords ... 226
TRichView.GetItemCoordsEx ... 227
TRichView.GetItemExtraIntProperty -Ex ... 227
TRichView.GetItemExtraStrProperty -Ex ... 228
TRichView.GetItemNo ... 229
TRichView.GetItemPara ... 229
TRichView.GetItemStyle ... 229
TRichView.GetItemTag ... 230
TRichView.GetItemText -A -W ... 230
TRichView.GetItemVAlign ... 230
TRichView.GetJumpPointItemNo ... 231
TRichView.GetJumpPointLocation ... 231
TRichView.GetJumpPointY ... 232

TRichView .GetLastCheckpoint ... 232
TRichView .GetLineNo ... 233
TRichView .GetListMarkerInfo ... 233
TRichView .GetNextCheckpoint ... 234
TRichView .GetOffsAfterItem ... 235
TRichView .GetOffsBeforeItem ... 235
TRichView .GetPictureInfo ... 236
TRichView .GetPrevCheckpoint ... 237
TRichView .GetSelectedImage ... 237
TRichView .GetSelectionBounds ... 237
TRichView .GetSelText -A -W ... 238
TRichView .GetTextInfo ... 239
TRichView .GetWordAt -A -W ... 239
TRichView .InsertRVFFromStream ... 240
TRichView .IsFromNew Line ... 242
TRichView .IsParaStart ... 242
TRichView .LiveSpellingValidateWord ... 242
TRichView .LoadFromStream ... 243
TRichView .LoadRTF ... 243
TRichView .LoadRTFFromStream ... 244
TRichView .LoadRVF ... 245
TRichView .LoadRVFFromStream ... 245
TRichView .LoadText -W ... 246
TRichView .LoadTextFromStream -W ... 247
TRichView .MarkStylesInUse ... 247
TRichView .Reformat ... 248
TRichView .RefreshListMarkers ... 248
TRichView .RefreshListSequences ... 248
TRichView .RemoveCheckpoint ... 248
TRichView .ResetAnimation ... 249
TRichView .SaveDocX ... 249
TRichView .SaveDocXToStream ... 250
TRichView .SaveHTML ... 250
TRichView .SaveHTMLEx ... 251
TRichView .SaveHTMLToStream ... 252
TRichView .SaveHTMLToStreamEx ... 252
TRichView .SavePicture ... 253
TRichView .SaveRTF ... 253
TRichView .SaveRTFToStream ... 254
TRichView .SaveRVF ... 254
TRichView .SaveRVFToStream ... 254
TRichView .SaveText -W ... 255
TRichView .SaveTextToStream -W ... 255
TRichView .SearchText -A -W ... 256
TRichView .SelectAll ... 257
TRichView .SelectControl ... 257
TRichView .SelectionExists ... 258
TRichView .SelectWordAt ... 258
TRichView .SetAddParagraphMode ... 259
TRichView .SetBreakInfo ... 260
TRichView .SetBulletInfo ... 261
TRichView .SetCheckpointInfo ... 261
TRichView .SetControlInfo ... 262
TRichView .SetFooter ... 263
TRichView .SetHeader ... 264
TRichView .SetHotspotInfo ... 265
TRichView .SetItemExtraIntProperty -Ex ... 266
TRichView .SetItemExtraStrProperty -Ex ... 266

TRichView.SetItemTag ... 267
TRichView.SetItemText -A -W ... 267
TRichView.SetItemVAlign ... 268
TRichView.SetListMarkerInfo ... 269
TRichView.SetPictureInfo ... 269
TRichView.SetSelectionBounds ... 271
TRichView.StartAnimation ... 272
TRichView.StartLiveSpelling ... 273
TRichView.StopAnimation ... 273
TRichView.UpdatePaletteInfo ... 273
Events ... 274
TRichView.OnAddStyle ... 274
TRichView.OnAssignImageFileName ... 274
TRichView.OnCheckpointVisible ... 274
TRichView.OnCMHintShow ... 275
TRichView.OnControlAction ... 275
TRichView.OnCopy ... 276
TRichView.OnGetItemCursor ... 277
TRichView.OnHTMLSaveImage ... 277
TRichView.OnImportPicture ... 279
TRichView.OnItemAction ... 280
TRichView.OnItemHint ... 281
TRichView.OnJump ... 282
TRichView.OnPaint ... 283
TRichView.OnProgress ... 283
TRichView.OnReadHyperlink ... 284
TRichView.OnReadMergeField ... 286
TRichView.OnRVDblClick ... 287
TRichView.OnRVFControlNeeded ... 288
TRichView.OnRVFImageListNeeded ... 288
TRichView.OnRVFPictureNeeded ... 289
TRichView.OnRVMouseDown ... 289
TRichView.OnRVMouseMove ... 290
TRichView.OnRVMouseUp ... 291
TRichView.OnRVRightClick ... 291
TRichView.OnSaveComponentToFile ... 292
TRichView.OnSaveDocXExtra ... 293
TRichView.OnSaveHTMLExtra ... 295
TRichView.OnSaveImage2 ... 296
TRichView.OnSaveItemToFile ... 297
TRichView.OnSaveParaToHTML ... 298
TRichView.OnSaveRTFExtra ... 299
TRichView.OnSelect ... 301
TRichView.OnSpellingCheck ... 301
TRichView.OnStyleTemplatesChange ... 302
TRichView.OnWriteHyperlink ... 302
Classes of Properties ... 304
TRVRTFReaderProperties ... 304
Properties ... 304
TRVRTFReaderProperties.AutoHideTableGridLines ... 304
TRVRTFReaderProperties.BasePathLinks ... 305
TRVRTFReaderProperties.CharsetForUnicode ... 305
TRVRTFReaderProperties.ConvertHighlight ... 305
TRVRTFReaderProperties.ErrorCode ... 306
TRVRTFReaderProperties.ExtractMetafileBitmaps ... 306
TRVRTFReaderProperties.IgnoreBookmarks ... 306
TRVRTFReaderProperties.IgnoreFields ... 307
TRVRTFReaderProperties.IgnoreNotes ... 307

TRVRTFReaderProperties.IgnorePictures ... 307
TRVRTFReaderProperties.IgnoreSequences ... 307
TRVRTFReaderProperties.IgnoreTables ... 307
TRVRTFReaderProperties.LineBreaksAsParagraphs ... 307
TRVRTFReaderProperties.ParaStyleMode ... 308
TRVRTFReaderProperties.ParaStyleNo ... 308
TRVRTFReaderProperties.ReadDocParameters ... 308
TRVRTFReaderProperties.SkipHiddenText ... 309
TRVRTFReaderProperties.TextStyleMode ... 309
TRVRTFReaderProperties.TextStyleNo ... 309
TRVRTFReaderProperties.UnicodeMode ... 310
TRVRTFReaderProperties.UseCharsetForUnicode ... 310
TRVRTFReaderProperties.UseHypertextStyles ... 310
TRVRTFReaderProperties.UseSingleCellPadding ... 310
TRVDocParameters ... 310
Properties ... 311
TRVDocParameters.Author ... 311
TRVDocParameters.BottomMargin ... 311
TRVDocParameters.Comments ... 311
TRVDocParameters.FacingPages ... 312
TRVDocParameters.FooterY ... 312
TRVDocParameters.HeaderY ... 312
TRVDocParameters.LeftMargin ... 312
TRVDocParameters.MirrorMargins ... 313
TRVDocParameters.Orientation ... 313
TRVDocParameters.PageHeight ... 313
TRVDocParameters.PageWidth ... 313
TRVDocParameters.RightMargin ... 313
TRVDocParameters.Title ... 314
TRVDocParameters.TitlePage ... 314
TRVDocParameters.TopMargin ... 314
TRVDocParameters.Units ... 314
TRVDocParameters.ZoomMode ... 314
TRVDocParameters.ZoomPercent ... 315
Methods ... 315
TRVDocParameters.ConvertToUnits ... 315
TRVDocParameters.Reset ... 315
TRVDocParameters.ResetLayout ... 315
TRVDocParameters.UnitsPerInch ... 315
Examples ... 316
Loading UTF-8 files ... 316
TRichView Edit ... 316
Properties ... 321
TRichView Edit.AcceptDragDropFormats, AcceptPasteFormats ... 321
TRichView Edit.ActualCurTextStyleNo ... 322
TRichView Edit.CurItemNo ... 322
TRichView Edit.CurItemStyle ... 323
TRichView Edit.CurParaStyleNo ... 323
TRichView Edit.CurTextStyleNo ... 324
TRichView Edit.CustomCaretInterval ... 324
TRichView Edit.DefaultPictureVAlign ... 325
TRichView Edit.EditorOptions ... 325
TRichView Edit.ForceFieldHighlight ... 327
TRichView Edit.LiveSpellingMode ... 327
TRichView Edit.Modified ... 327
TRichView Edit.OffsetInCurItem ... 328
TRichView Edit.ReadOnly ... 328
TRichView Edit.SmartPopupProperties ... 329

TRichView Edit.SmartPopupVisible ... 329
TRichView Edit.TopLevelEditor ... 329
TRichView Edit.UndoLimit ... 329
Methods ... 330
TRichView Edit.AdjustControlPlacement ... 330
TRichView Edit.AdjustControlPlacement2 ... 330
TRichView Edit.ApplyListStyle ... 331
TRichView Edit.ApplyParaStyle ... 331
TRichView Edit.ApplyParaStyleConversion ... 332
TRichView Edit.ApplyParaStyleTemplate ... 332
TRichView Edit.ApplyStyleConversion ... 333
TRichView Edit.ApplyStyleTemplate ... 333
TRichView Edit.ApplyTextStyle ... 334
TRichView Edit.ApplyTextStyleTemplate ... 334
TRichView Edit.BeginItemModify ... 335
TRichView Edit.BeginUndoCustomGroup ... 336
TRichView Edit.BeginUndoGroup ... 336
TRichView Edit.BeginUpdate ... 336
TRichView Edit.CanChange ... 336
TRichView Edit.CanPaste ... 337
TRichView Edit.CanPasteRTF ... 337
TRichView Edit.CanPasteRVF ... 337
TRichView Edit.Change ... 337
TRichView Edit.ChangeListLevels ... 338
TRichView Edit.ChangeStyleTemplates ... 338
TRichView Edit.Clear ... 338
TRichView Edit.ClearTextFlow ... 339
TRichView Edit.ClearUndo ... 339
TRichView Edit.ConvertToHotPicture ... 339
TRichView Edit.ConvertToPicture ... 339
TRichView Edit.Create ... 340
TRichView Edit.CutDef ... 340
TRichView Edit.DeleteSelection ... 340
TRichView Edit.Destroy ... 340
TRichView Edit.EndItemModify ... 341
TRichView Edit.EndUpdate ... 341
TRichView Edit.GetCheckpointAtCaret ... 341
TRichView Edit.GetCurrentBreakInfo ... 341
TRichView Edit.GetCurrentBulletInfo ... 342
TRichView Edit.GetCurrentCheckpoint ... 343
TRichView Edit.GetCurrentControlInfo ... 343
TRichView Edit.GetCurrentHotspotInfo ... 344
TRichView Edit.GetCurrentItem ... 345
TRichView Edit.GetCurrentItemEx ... 345
TRichView Edit.GetCurrentItemExtraIntProperty -Ex ... 346
TRichView Edit.GetCurrentItemExtraStrProperty -Ex ... 346
TRichView Edit.GetCurrentItemText -A -W ... 347
TRichView Edit.GetCurrentItemVAlign ... 348
TRichView Edit.GetCurrentLineCol ... 348
TRichView Edit.GetCurrentMisspelling ... 348
TRichView Edit.GetCurrentPictureInfo ... 349
TRichView Edit.GetCurrentTag ... 349
TRichView Edit.GetCurrentTextInfo ... 350
TRichView Edit.InsertBreak ... 350
TRichView Edit.InsertBullet ... 351
TRichView Edit.InsertCheckpoint ... 351
TRichView Edit.InsertControl ... 352
TRichView Edit.InsertHotPicture ... 352

TRichView Edit.InsertHotspot ... 353
TRichView Edit.InsertItem ... 353
TRichView Edit.InsertOEMTextFromFile ... 354
TRichView Edit.InsertPageBreak ... 355
TRichView Edit.InsertPicture ... 355
TRichView Edit.InsertRTFFromFileEd ... 356
TRichView Edit.InsertRTFFromStreamEd ... 357
TRichView Edit.InsertRVFFromFileEd ... 357
TRichView Edit.InsertRVFFromStreamEd ... 358
TRichView Edit.InsertStringTag -A -W ... 359
TRichView Edit.InsertTab ... 359
TRichView Edit.InsertText -A -W ... 360
TRichView Edit.InsertTextFromFile -W ... 361
TRichView Edit.Paste ... 361
TRichView Edit.PasteBitmap ... 362
TRichView Edit.PasteGraphicFiles ... 362
TRichView Edit.PasteMetafile ... 363
TRichView Edit.PasteRTF ... 363
TRichView Edit.PasteRVF ... 364
TRichView Edit.PasteText -A -W ... 364
TRichView Edit.PasteURL ... 365
TRichView Edit.Redo ... 365
TRichView Edit.RedoAction ... 366
TRichView Edit.RedoName ... 366
TRichView Edit.RemoveCheckpointAtCaret ... 366
TRichView Edit.RemoveCheckpointEd ... 366
TRichView Edit.RemoveCurrentCheckpoint ... 367
TRichView Edit.RemoveCurrentPageBreak ... 367
TRichView Edit.RemoveLists ... 367
TRichView Edit.ResizeControl ... 368
TRichView Edit.ResizeCurrentControl ... 368
TRichView Edit.SearchText -W ... 369
TRichView Edit.SelectCurrentLine ... 369
TRichView Edit.SelectCurrentWord ... 370
TRichView Edit.Set*PropertyEd ... 370
TRichView Edit.SetBackgroundImageEd ... 371
TRichView Edit.SetBreakInfoEd ... 371
TRichView Edit.SetBulletInfoEd ... 372
TRichView Edit.SetCheckpointInfoEd ... 373
TRichView Edit.SetControlInfoEd ... 374
TRichView Edit.SetCurrentBreakInfo ... 375
TRichView Edit.SetCurrentBulletInfo ... 375
TRichView Edit.SetCurrentCheckpointInfo ... 376
TRichView Edit.SetCurrentControlInfo ... 377
TRichView Edit.SetCurrentHotspotInfo ... 377
TRichView Edit.SetCurrentItemExtraIntProperty -Ex ... 378
TRichView Edit.SetCurrentItemExtraStrProperty -Ex ... 379
TRichView Edit.SetCurrentItemText -A -W ... 380
TRichView Edit.SetCurrentItemVAlign ... 380
TRichView Edit.SetCurrentPictureInfo ... 381
TRichView Edit.SetCurrentTag ... 382
TRichView Edit.SetHotspotInfoEd ... 382
TRichView Edit.SetItemExtraIntProperty-Ed -ExEd ... 383
TRichView Edit.SetItemExtraStrProperty-Ed -ExEd ... 384
TRichView Edit.SetItemTagEd ... 384
TRichView Edit.SetItemTextEd -A -W ... 385
TRichView Edit.SetItemVAlignEd ... 386
TRichView Edit.SetPictureInfoEd ... 386

TRichView Edit.SetUndoGroupMode ... 387
TRichView Edit.Undo ... 387
TRichView Edit.UndoAction ... 388
TRichView Edit.UndoName ... 388
Events ... 388
TRichView Edit.OnBeforeOleDrop, OnAfterOleDrop ... 388
TRichView Edit.OnCaretGetOut ... 389
TRichView Edit.OnCaretMove ... 390
TRichView Edit.OnChange ... 390
TRichView Edit.OnChanging ... 390
TRichView Edit.OnCheckStickingItems ... 391
TRichView Edit.OnCurParaStyleChanged ... 391
TRichView Edit.OnCurTextStyleChanged ... 391
TRichView Edit.OnDraw CustomCaret ... 391
TRichView Edit.OnDropFile ... 392
TRichView Edit.OnDropFiles ... 393
TRichView Edit.OnItemResize ... 393
TRichView Edit.OnItemTextEdit ... 394
TRichView Edit.OnMeasureCustomCaret ... 394
TRichView Edit.OnOleDragEnter ... 395
TRichView Edit.OnOleDragLeave ... 395
TRichView Edit.OnOleDragOver ... 396
TRichView Edit.OnOleDrop ... 397
TRichView Edit.OnParaStyleConversion ... 397
TRichView Edit.OnPaste ... 399
TRichView Edit.OnSmartPopupClick ... 400
TRichView Edit.OnStyleConversion ... 400
Classes of Properties ... 402
TRVSmartPopupProperties ... 402
Properties ... 402
TRVSmartPopupProperties.ButtonType ... 402
TRVSmartPopupProperties.Color ... 403
TRVSmartPopupProperties.Hint ... 403
TRVSmartPopupProperties.HoverColor ... 403
TRVSmartPopupProperties.HoverLineColor ... 403
TRVSmartPopupProperties.ImageIndex ... 404
TRVSmartPopupProperties.ImageList ... 404
TRVSmartPopupProperties.LineColor ... 404
TRVSmartPopupProperties.Menu ... 404
TRVSmartPopupProperties.Position ... 404
TRVSmartPopupProperties.ShortCut ... 404
Methods ... 405
TRVSmartPopupProperties.SetButtonState ... 405
Examples ... 405
Moving caret to the beginning of paragraph ... 405
OnDropFile Example ... 405
TDBRichView ... 407
Properties ... 408
TDBRichView .AutoDisplay ... 408
TDBRichView .DataField ... 408
TDBRichView .DataSource ... 408
TDBRichView .Field ... 409
Methods ... 409
TDBRichView .Create ... 409
TDBRichView .Destroy ... 409
TDBRichView .LoadField ... 409
Events ... 410
TDBRichView .OnLoadCustomFormat ... 410

TDBRichView .OnLoadDocument ... 410
TDBRichView .OnNew Document ... 410
TDBRichView Edit ... 410
Properties ... 412
TDBRichView Edit.AutoDeleteUnusedStyles ... 412
TDBRichView Edit.AutoDisplay ... 412
TDBRichView Edit.DataField ... 413
TDBRichView Edit.DataSource ... 413
TDBRichView Edit.Field ... 413
TDBRichView Edit.FieldFormat ... 413
TDBRichView Edit.IgnoreEscape ... 414
TDBRichView Edit.ReadOnly ... 414
Methods ... 414
TDBRichView Edit.Change ... 414
TDBRichView Edit.Create ... 414
TDBRichView Edit.Destroy ... 415
TDBRichView Edit.LoadField ... 415
Events ... 415
TDBRichView Edit.OnLoadCustomFormat ... 415
TDBRichView Edit.OnLoadDocument ... 415
TDBRichView Edit.OnNew Document ... 416
TDBRichView Edit.OnSaveCustomFormat ... 416
Examples ... 416
Using viewer-style methods in DBRichView Edit ... 416
Inserting controls what can modify themselves in DBRichView Edit ... 417
TRVPrintPreview ... 417
Properties ... 418
TRVPrintPreview .RVPrint ... 418
Additional Components ... 418
TRVStyle ... 418
Properties ... 420
TRVStyle.CheckpointColor, CheckpointColor ... 420
TRVStyle.Color ... 421
TRVStyle.CurrentItemColor ... 421
TRVStyle.DefCodePage ... 421
TRVStyle.DefTabWidth ... 422
TRVStyle.DefUnicodeStyle ... 422
TRVStyle.DisabledFontColor ... 422
TRVStyle.EndnoteNumbering ... 422
TRVStyle.FieldHighlightColor ... 423
TRVStyle.FieldHighlightType ... 423
TRVStyle.FloatingLineColor ... 423
TRVStyle.FontQuality ... 424
TRVStyle.FootnotePageReset ... 425
TRVStyle.FullRedraw ... 425
TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle ... 425
TRVStyle.HoverColor ... 426
TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor ... 426
TRVStyle.InvalidPicture ... 427
TRVStyle.JumpCursor ... 427
TRVStyle.LineSelectCursor ... 427
TRVStyle.LineWrapMode ... 428
TRVStyle.ListStyles ... 429
TRVStyle.LiveSpellingColor ... 430
TRVStyle.MainRVStyle ... 430
TRVStyle.PageBreakColor ... 431
TRVStyle.ParaStyles ... 431
TRVStyle.SelectionHandleKind ... 431

TRVStyle.SelectionMode ... 432
TRVStyle.SelectionStyle ... 432
TRVStyle.SoftPageBreakColor ... 434
TRVStyle.SpacesInTab ... 435
TRVStyle.SpecialCharactersColor ... 435
TRVStyle.StyleTemplates ... 435
TRVStyle.TextBackgroundKind ... 435
TRVStyle.TextEngine ... 436
TRVStyle.TextStyles ... 437
TRVStyle.Units ... 437
TRVStyle.UseSound ... 437
Methods ... 438
TRVStyle.AddTextStyle ... 438
TRVStyle.Create ... 438
TRVStyle.ConvertTo* ... 438
TRVStyle.DeleteTextStyle ... 438
TRVStyle.Destroy ... 438
TRVStyle.GetAs* ... 439
TRVStyle.*ToUnits ... 439
TRVStyle.GetListStyleClass ... 439
TRVStyle.GetParaStyleClass ... 440
TRVStyle.GetTextStyleClass ... 440
TRVStyle.LoadINI ... 440
TRVStyle.FindTextStyle, FindParaStyle ... 440
TRVStyle.LoadReg ... 441
TRVStyle.SaveCSS ... 441
TRVStyle.SaveCSSToStream ... 442
TRVStyle.SaveINI ... 442
TRVStyle.SaveReg ... 442
Events ... 443
TRVStyle.OnAfterApplyStyle ... 443
TRVStyle.OnApplyStyle ... 443
TRVStyle.OnApplyStyleColor ... 444
TRVStyle.OnDrawCheckpoint ... 444
TRVStyle.OnDrawPageBreak ... 446
TRVStyle.OnDrawParaBack ... 447
TRVStyle.OnDrawStyleText ... 448
TRVStyle.OnDrawTextBack ... 449
TRVStyle.OnStyleHoverSensitive ... 450
Classes of Properties - Style Collections ... 451
TFontInfos ... 451
Properties ... 451
TFontInfos.InvalidItem ... 451
TFontInfos.Items ... 451
TFontInfos.PixelsPerInch ... 452
Methods ... 452
TFontInfos.Add ... 452
TFontInfos.AddFont ... 452
TFontInfos.AddFontEx ... 452
TFontInfos.AssignTo ... 453
TFontInfos.Create ... 453
TFontInfos.FindStyleWithCharset ... 453
TFontInfos.FindStyleWithColor ... 453
TFontInfos.FindStyleWithFont ... 454
TFontInfos.FindStyleWithFontName ... 454
TFontInfos.FindStyleWithFontSize ... 454
TFontInfos.FindStyleWithFontStyle ... 455
TFontInfos.FindSuchStyle, FindSuchStyleEx ... 455

TParaInfos ... 456
Properties ... 456
TParaInfos.InvalidItem ... 456
TParaInfos.Items ... 456
Methods ... 457
TParaInfos.Add ... 457
TParaInfos.AssignTo ... 457
TParaInfos.Create ... 457
TParaInfos.FindStyleWithAlignment ... 457
TParaInfos.FindSuchStyle, FindSuchStyleEx ... 458
TRVListInfos ... 458
Properties ... 459
TRVListInfos.Items ... 459
Methods ... 459
TRVListInfos.Add ... 459
TRVListInfos.AssignTo ... 459
TRVListInfos.FindStyleWithLevels ... 459
TRVListInfos.FindSuchStyle ... 459
TRVStyleTemplateCollection ... 460
Properties ... 460
TRVStyleTemplateCollection.DefStyleName ... 460
TRVStyleTemplateCollection.ForbiddenIds ... 461
TRVStyleTemplateCollection.Items ... 461
TRVStyleTemplateCollection.NormalStyleTemplate ... 461
Methods ... 461
TRVStyleTemplateCollection.AssignStyleTemplates ... 461
TRVStyleTemplateCollection.AssignToStrings ... 461
TRVStyleTemplateCollection.ClearTextFormat, ClearParaFormat ... 462
TRVStyleTemplateCollection.FindById, FindItemById ... 462
TRVStyleTemplateCollection.FindByName, FindItemByName ... 462
TRVStyleTemplateCollection.LoadFromRVST, SaveToRVST ... 462
TRVStyleTemplateCollection.InsertFromRVST ... 463
TRVStyleTemplateCollection.ResetNameCounter ... 463
Classes of Properties - Styles ... 463
TCustomRVInfo, TCustomRVTextOrParaInfo ... 463
Properties ... 464
TCustomRVInfo.Standard ... 464
TCustomRVInfo.StyleName ... 465
TCustomRVFontOrParaInfo.StyleTemplateId ... 465
TCustomRVFontInfo ... 466
Properties ... 467
TCustomRVFontInfo.BackColor ... 467
TCustomRVFontInfo.BiDiMode ... 467
TCustomRVFontInfo.CharScale ... 467
TCustomRVFontInfo.Charset ... 468
TCustomRVFontInfo.CharSpacing ... 468
TCustomRVFontInfo.Color ... 469
TCustomRVFontInfo.EmptyWidth ... 469
TCustomRVFontInfo.FontName ... 470
TCustomRVFontInfo.HoverBackColor ... 470
TCustomRVFontInfo.HoverColor ... 470
TCustomRVFontInfo.HoverEffects ... 471
TCustomRVFontInfo.HoverUnderlineColor ... 471
TCustomRVFontInfo.JumpCursor ... 472
TCustomRVFontInfo.Options ... 472
TCustomRVFontInfo.ParaStyleTemplateId ... 473
TCustomRVFontInfo.Protection ... 473
TCustomRVFontInfo.Size ... 475

TCustomRVFontInfo.Style ... 475
TCustomRVFontInfo.StyleEx ... 475
TCustomRVFontInfo.SubSuperScriptType ... 476
TCustomRVFontInfo.UnderlineColor ... 476
TCustomRVFontInfo.UnderlineType ... 476
TCustomRVFontInfo.VShift ... 477
Methods ... 477
TCustomRVFontInfo.Assign ... 477
TCustomRVFontInfo.AssignTo ... 477
TCustomRVFontInfo.Create ... 478
TCustomRVFontInfo.IsEqual ... 478
TFontInfo ... 478
Properties ... 479
TFontInfo.Jump ... 479
TFontInfo.ModifiedProperties ... 480
TFontInfo.NextStyleNo ... 480
TFontInfo.Unicode ... 480
Methods ... 480
TFontInfo.Assign ... 480
TFontInfo.Create ... 481
TFontInfo.Draw ... 481
TFontInfo.IsEqual ... 482
TCustomRVParaInfo ... 482
Properties ... 483
TCustomRVParaInfo.Alignment, LastLineAlignment ... 483
TCustomRVParaInfo.Background ... 484
TCustomRVParaInfo.BiDiMode ... 484
TCustomRVParaInfo.Border ... 485
TCustomRVParaInfo.FirstIndent ... 485
TCustomRVParaInfo.LeftIndent ... 485
TCustomRVParaInfo.LineSpacing ... 486
TCustomRVParaInfo.LineSpacingType ... 486
TCustomRVParaInfo.Options ... 487
TCustomRVParaInfo.OutlineLevel ... 488
TCustomRVParaInfo.RightIndent ... 488
TCustomRVParaInfo.SpaceAfter ... 489
TCustomRVParaInfo.SpaceBefore ... 489
TCustomRVParaInfo.Tabs ... 489
Methods ... 489
TCustomRVParaInfo.Assign ... 489
TCustomRVParaInfo.Create ... 489
TCustomRVParaInfo.IsEqual ... 490
TParaInfo ... 490
Properties ... 491
TParaInfo.DefStyleNo ... 491
TParaInfo.ModifiedProperties ... 491
TParaInfo.NextParaNo ... 492
Methods ... 492
TParaInfo.Assign ... 492
TParaInfo.Create ... 492
TParaInfo.IsEqual ... 492
TRVListInfo ... 493
Properties ... 493
TRVListInfo.Levels ... 493
TRVListInfo.OneLevelPreview ... 493
Methods ... 494
TRVListInfo.AllNumbered ... 494
TRVListInfo.HasNumbering ... 494

TRVStyleTemplate ... 494
Properties ... 495
TRVStyleTemplate.Id ... 495
TRVStyleTemplate.Kind ... 495
TRVStyleTemplate.Name ... 496
TRVStyleTemplate.NextId, Next ... 496
TRVStyleTemplate.ParaStyle ... 497
TRVStyleTemplate.ParentId, Parent ... 497
TRVStyleTemplate.QuickAccess ... 497
TRVStyleTemplate.TextStyle ... 497
TRVStyleTemplate.ValidParaProperties ... 498
TRVStyleTemplate.ValidTextProperties ... 498
Methods ... 498
TRVStyleTemplate.ApplyToParaStyle ... 498
TRVStyleTemplate.ApplyToTextStyle ... 499
TRVStyleTemplate.UpdateModifiedTextStyleProperties, UpdateModifiedParaStyleProperties ... 500
Classes of Properties of Paragraph Style ... 500
TRVBackgroundRect ... 500
Properties ... 501
TRVBackgroundRect.BorderOffsets ... 501
TRVBackgroundRect.Color ... 501
TRVBackgroundRect.Opacity ... 501
TRVBorder ... 501
Properties ... 502
TRVBorder.BorderOffsets ... 502
TRVBorder.Color ... 502
TRVBorder.InternalWidth ... 502
TRVBorder.Style ... 502
TRVBorder.VisibleBorders ... 502
TRVBorder.Width ... 503
TRVTabInfos ... 503
Properties ... 503
TRVTabInfos.Items ... 503
Methods ... 503
TRVTabInfos.AddFrom ... 503
TRVTabInfos.DeleteList ... 504
TRVTabInfos.Find ... 504
TRVTabInfos.Intersect ... 504
TRVTabInfo ... 504
Properties ... 504
TRVTabInfo.Align ... 504
TRVTabInfo.Leader ... 505
TRVTabInfo.Position ... 505
Classes of Properties of List Style ... 506
TRVListLevelCollection ... 506
Properties ... 506
TRVListLevelCollection.Items ... 506
Methods ... 506
TRVListLevelCollection.Add ... 506
TRVListLevel ... 506
Properties ... 507
TRVListLevel.FirstIndent ... 507
TRVListLevel.Font ... 507
TRVListLevel.FormatString ... 508
TRVListLevel.FormatStringW ... 509
TRVListLevel.ImageIndex ... 509
TRVListLevel.ImageList ... 509
TRVListLevel.LeftIndent ... 510

TRVListLevel.ListType ... 510
TRVListLevel.MarkerAlignment ... 511
TRVListLevel.MarkerIndent ... 512
TRVListLevel.Options ... 512
TRVListLevel.Picture ... 513
TRVListLevel.StartFrom ... 513
Methods ... 513
TRVListLevel.HasNumbering ... 513
TRVPrint ... 513
Properties ... 518
TRVPrint.ClipMargins ... 518
TRVPrint.FacingPages ... 518
TRVPrint.FixMarginsMode ... 518
TRVPrint.FooterY ... 519
TRVPrint.HeaderY ... 519
TRVPrint.Margins ... 520
TRVPrint.MirrorMargins ... 521
TRVPrint.Preview100PercentHeight ... 521
TRVPrint.Preview100PercentWidth ... 521
TRVPrint.TitlePage ... 521
TRVPrint.Units ... 521
TRVPrint.VirtualPrinter ... 522
Methods ... 522
TRVPrint.AssignDocParameters ... 522
TRVPrint.AssignSource ... 522
TRVPrint.ContinuousPrint ... 523
TRVPrint.ConvertToUnits ... 523
TRVPrint.Create ... 524
TRVPrint.DrawPreview ... 524
TRVPrint.DrawPage, DrawPageAt ... 524
TRVPrint.FormatPages ... 525
TRVPrint.GetFooterRect ... 525
TRVPrint.GetHeaderRect ... 525
TRVPrint.MakePreview ... 526
TRVPrint.MakeScaledPreview ... 526
TRVPrint.Print ... 526
TRVPrint.PrintPages ... 527
TRVPrint.SetFooter ... 528
TRVPrint.SetHeader ... 529
Events ... 529
TRVPrint.OnFormatting ... 529
TRVPrint.OnPagePostpaint ... 530
TRVPrint.OnPagePrepaint ... 530
TRVPrint.OnSendingToPrinter ... 531
Classes of Properties ... 532
TRVVirtualPrinterProperties ... 532
Properties ... 532
TRVVirtualPrinterProperties.Active ... 532
TRVVirtualPrinterProperties.PixelsPerInch ... 532
TRVVirtualPrinterProperties.PageWidth, PageHeight ... 532
TRVOfficeConverter ... 532
Properties ... 533
TRVOfficeConverter.ErrorCode ... 533
TRVOfficeConverter.ExcludeDocX-Converter ... 534
TRVOfficeConverter.ExcludeHTML-Converter ... 534
TRVOfficeConverter.ExportConverters ... 535
TRVOfficeConverter.ExtensionsInFilter ... 535
TRVOfficeConverter.ImportConverters ... 535

TRVOfficeConverter.PreviewMode ... 535
TRVOfficeConverter.Stream ... 536
Methods ... 536
TRVOfficeConverter.Create ... 536
TRVOfficeConverter.Destroy ... 536
TRVOfficeConverter.ExportRTF ... 536
TRVOfficeConverter.ExportRV ... 537
TRVOfficeConverter.GetExportFilter ... 537
TRVOfficeConverter.GetImportFilter ... 537
TRVOfficeConverter.ImportRTF ... 538
TRVOfficeConverter.ImportRV ... 538
TRVOfficeConverter.IsValidImporter ... 539
Events ... 539
TRVOfficeConverter.OnConverting ... 539
Classes of Properties ... 539
TRVOfficeCnvList ... 539
TRVOfficeConverterInfo ... 539
TRVReportHelper ... 540
Properties ... 541
TRVReportHelper.AllowTransparentDrawing ... 541
TRVReportHelper.RichView ... 541
Methods ... 541
TRVReportHelper.DrawPage ... 541
TRVReportHelper.DrawPageAt ... 542
TRVReportHelper.Finished ... 542
TRVReportHelper.FormatNextPage ... 542
TRVReportHelper.GetLastPageHeight ... 543
TRVReportHelper.Init ... 543
TRVReportHelper.UpdatePageNumbers ... 543
Events ... 543
TRVReportHelper.OnGetPageNumber ... 543
Components Ancestor Classes ... 544
TRVScroller ... 544
Properties ... 545
TRVScroller.BorderStyle ... 545
TRVScroller.HScrollMax ... 545
TRVScroller.HScrollPos ... 545
TRVScroller.HScrollVisible ... 545
TRVScroller.InplaceEditor ... 546
TRVScroller.NoVScroll ... 546
TRVScroller.Tracking ... 546
TRVScroller.UseXPThemes ... 546
TRVScroller.VScrollMax ... 547
TRVScroller.VScrollPos ... 547
TRVScroller.VScrollVisible ... 547
TRVScroller.WheelStep ... 548
Methods ... 548
TRVScroller.Create ... 548
TRVScroller.ScrollTo ... 548
Events ... 548
TRVScroller.OnHScrolled ... 548
TRVScroller.OnVScrolled ... 549
TCustomRVPrint ... 549
Properties ... 549
TCustomRVPrint.ColorMode ... 549
TCustomRVPrint.EndAt ... 549
TCustomRVPrint.FirstPageNo ... 550
TCustomRVPrint.IgnorePageBreaks ... 550

TCustomRVPrint.LastPageNo ... 550
TCustomRVPrint.MaxPrintedItemNo ... 550
TCustomRVPrint.MetafileCompatibility ... 551
TCustomRVPrint.MinPrintedItemNo ... 551
TCustomRVPrint.NoMetafiles ... 551
TCustomRVPrint.MetafilePixelsPerInch ... 551
TCustomRVPrint.PageNoFromNumber ... 552
TCustomRVPrint.PagesCount ... 552
TCustomRVPrint.Preview Correction ... 552
TCustomRVPrint.StartAt ... 552
TCustomRVPrint.TransparentBackground ... 553
Methods ... 553
TCustomRVPrint.Clear ... 553
TCustomRVPrint.GetFirstItemOnPage ... 553
TCustomRVPrint.GetPageNo ... 553
TCustomRVPrint.UpdatePaletteInfo ... 554
Events ... 554
TCustomRVPrint.OnPrintComponent ... 554
TCustomRVPrint.OnDraw Hyperlink ... 556
TCustomRVPrint.OnDraw Checkpoint ... 556
TCustomRVPrintPreview ... 556
Properties ... 558
TCustomRVPrintPreview .ClickMode ... 558
TCustomRVPrintPreview .Color ... 558
TCustomRVPrintPreview .MarginsPen ... 559
TCustomRVPrintPreview .PageBorderColor ... 559
TCustomRVPrintPreview .PageBorderWidth ... 559
TCustomRVPrintPreview .PageNo ... 559
TCustomRVPrintPreview .PrintableAreaPen ... 560
TCustomRVPrintPreview .Shadow Color ... 560
TCustomRVPrintPreview .Shadow Width ... 560
TCustomRVPrintPreview .ZoomInCursor ... 560
TCustomRVPrintPreview .ZoomMode ... 561
TCustomRVPrintPreview .ZoomOutCursor ... 561
TCustomRVPrintPreview .ZoomPercent ... 562
Methods ... 562
TCustomRVPrintPreview .Create ... 562
TCustomRVPrintPreview .First ... 562
TCustomRVPrintPreview .Last ... 562
TCustomRVPrintPreview .Next ... 563
TCustomRVPrintPreview .Prev ... 563
TCustomRVPrintPreview .SetZoom ... 563
Events ... 563
TCustomRVPrintPreview .OnZoomChanged ... 563
.6..Item Types ... 563
Table - TRVTableItemInfo ... 565
Properties ... 567
TRVTableItemInfo.BackgroundImage ... 567
TRVTableItemInfo.BackgroundImageFileName ... 568
TRVTableItemInfo.BackgroundStyle ... 568
TRVTableItemInfo.BestWidth ... 568
TRVTableItemInfo.BorderColor ... 569
TRVTableItemInfo.BorderHSpacing ... 569
TRVTableItemInfo.BorderLightColor ... 569
TRVTableItemInfo.BorderStyle ... 570
TRVTableItemInfo.BorderVSpacing ... 570
TRVTableItemInfo.BorderWidth ... 571

TRVTableItemInfo.CellBorderColor ... 571
TRVTableItemInfo.CellBorderLightColor ... 571
TRVTableItemInfo.CellBorderStyle ... 572
TRVTableItemInfo.CellBorderWidth ... 572
TRVTableItemInfo.CellHPadding ... 573
TRVTableItemInfo.CellHSpacing ... 573
TRVTableItemInfo.CellOverrideColor ... 573
TRVTableItemInfo.CellPadding ... 574
TRVTableItemInfo.Cells ... 574
TRVTableItemInfo.CellVPadding ... 574
TRVTableItemInfo.CellVSpacing ... 574
TRVTableItemInfo.ColBandSize, RowBandSize ... 575
TRVTableItemInfo.ColCount ... 575
TRVTableItemInfo.Color ... 575
TRVTableItemInfo.HeadingRowCount ... 576
TRVTableItemInfo.HOutermostRule ... 576
TRVTableItemInfo.HRuleColor ... 576
TRVTableItemInfo.HRuleWidth ... 577
TRVTableItemInfo.Opacity ... 577
TRVTableItemInfo.Options ... 577
TRVTableItemInfo.PrintOptions ... 579
TRVTableItemInfo.RowCount ... 580
TRVTableItemInfo.Rows ... 580
TRVTableItemInfo.TextColSeparator ... 581
TRVTableItemInfo.TextRowSeparator ... 581
TRVTableItemInfo.VisibleBorders ... 581
TRVTableItemInfo.VOutermostRule ... 582
TRVTableItemInfo.VRuleColor ... 582
TRVTableItemInfo.VRuleWidth ... 582
TRVTableItemInfo's row and column colors ... 583
Methods ... 584
TRVTableItemInfo.AssignProperties ... 584
TRVTableItemInfo.CanMergeCells ... 584
TRVTableItemInfo.CanMergeSelectedCells ... 585
TRVTableItemInfo.Changed ... 585
TRVTableItemInfo.Create ... 585
TRVTableItemInfo.CreateEx ... 586
TRVTableItemInfo.DeleteCols ... 586
TRVTableItemInfo.DeleteEmptyCols ... 587
TRVTableItemInfo.DeleteEmptyRows ... 587
TRVTableItemInfo.DeleteRows ... 587
TRVTableItemInfo.DeleteSelectedCols ... 588
TRVTableItemInfo.DeleteSelectedRows ... 588
TRVTableItemInfo.Deselect ... 588
TRVTableItemInfo.Destroy ... 589
TRVTableItemInfo.EditCell ... 589
TRVTableItemInfo.GetCellAt ... 589
TRVTableItemInfo.GetEditedCell ... 590
TRVTableItemInfo.GetNormalizedSelectionBounds ... 590
TRVTableItemInfo.GetSelectionBounds ... 591
TRVTableItemInfo.InsertCols ... 591
TRVTableItemInfo.InsertColsLeft ... 592
TRVTableItemInfo.InsertColsRight ... 592
TRVTableItemInfo.InsertRows ... 592
TRVTableItemInfo.InsertRowsAbove ... 593
TRVTableItemInfo.InsertRowsBelow ... 593
TRVTableItemInfo.IsCellSelected ... 593
TRVTableItemInfo.LoadFromStream ... 593

TRVTableItemInfo.MergeCells ... 594
TRVTableItemInfo.MergeSelectedCells ... 594
TRVTableItemInfo.SaveRowsToStream ... 595
TRVTableItemInfo.SaveToStream ... 595
TRVTableItemInfo.Select ... 595
TRVTableItemInfo.SelectCols ... 595
TRVTableItemInfo.SelectRows ... 596
TRVTableItemInfo.SetCellBackgroundImage ... 596
TRVTableItemInfo.SetCellBackgroundImageFileName ... 596
TRVTableItemInfo.SetCellBackgroundStyle ... 597
TRVTableItemInfo.SetCellBestHeight ... 597
TRVTableItemInfo.SetCellBestWidth ... 597
TRVTableItemInfo.SetCellBorderColor ... 597
TRVTableItemInfo.SetCellBorderLightColor ... 598
TRVTableItemInfo.SetCellColor ... 598
TRVTableItemInfo.SetCellHint ... 598
TRVTableItemInfo.SetCellOpacity ... 598
TRVTableItemInfo.SetCellOptions ... 599
TRVTableItemInfo.SetCellRotation ... 599
TRVTableItemInfo.SetCellTag ... 599
TRVTableItemInfo.SetCellVAlign ... 599
TRVTableItemInfo.SetCellVisibleBorders ... 600
TRVTableItemInfo.SetRow* ... 600
TRVTableItemInfo.SetTableVisibleBorders ... 600
TRVTableItemInfo.SplitSelectedCellsHorizontally ... 600
TRVTableItemInfo.SplitSelectedCellsVertically ... 601
TRVTableItemInfo.UnmergeCells ... 601
TRVTableItemInfo.UnmergeSelectedCells ... 601
Events ... 602
TRVTableItemInfo.OnCellEditing ... 602
TRVTableItemInfo.OnDrawBackground ... 603
TRVTableItemInfo.OnDrawBorder ... 603
Classes of Properties ... 605
TRVTableCellData ... 605
Properties ... 605
TRVTableCellData.BackgroundImage ... 605
TRVTableCellData.BackgroundImageFileName ... 606
TRVTableCellData.BackgroundStyle ... 606
TRVTableCellData.BestHeight ... 606
TRVTableCellData.BestWidth ... 607
TRVTableCellData.BorderColor ... 607
TRVTableCellData.BorderLightColor ... 608
TRVTableCellData.Color ... 608
TRVTableCellData.ColSpan ... 608
TRVTableCellData.Hint ... 609
TRVTableCellData.Opacity ... 609
TRVTableCellData.Options ... 609
TRVTableCellData.Rotation ... 611
TRVTableCellData.RowSpan ... 612
TRVTableCellData.Tag ... 612
TRVTableCellData.VAlign ... 613
TRVTableCellData.VisibleBorders ... 613
Methods ... 613
TRVTableCellData.GetRVData ... 613
Additional properties of TRVTableCellData ... 614
TRVTableRow ... 615
Properties ... 615
TRVTableRow.Items ... 615

TRVTableRow.KeepTogether ... 615
TRVTableRow.PageBreakBefore ... 616
TRVTableRow.VAlign ... 616
TRVTableRows ... 616
Properties ... 617
TRVTableRows.Items ... 617
Methods ... 617
TRVTableRows.GetMainCell ... 617
Label - TRVLabelItemInfo ... 617
Properties ... 618
TRVLabelItemInfo.Alignment ... 618
TRVLabelItemInfo.Cursor ... 618
TRVLabelItemInfo.MinWidth ... 618
TRVLabelItemInfo.ProtectTextStyleNo ... 619
TRVLabelItemInfo.RemoveInternalLeading, RemoveSpaceBelow ... 619
TRVLabelItemInfo.Text ... 619
TRVLabelItemInfo.TextStyleNo ... 620
Methods ... 620
TRVLabelItemInfo.Create ... 620
TRVLabelItemInfo.CreateEx ... 620
Math Expression - TRVMathItemInfo ... 621
Properties ... 621
TRVMathItemInfo.DisplayInline ... 621
TRVMathItemInfo.FontName ... 622
TRVMathItemInfo.FontSizeDouble ... 622
TRVMathItemInfo.Text ... 622
TRVMathItemInfo.TextColor ... 623
Methods ... 623
TRVMathItemInfo.Create ... 623
Numbered Sequence - TRVSeqItemInfo ... 624
Properties ... 625
TRVSeqItemInfo.FormatString ... 625
TRVSeqItemInfo.NumberType ... 625
TRVSeqItemInfo.Reset ... 626
TRVSeqItemInfo.SeqName ... 626
TRVSeqItemInfo.StartFrom ... 626
Methods ... 626
TRVSeqItemInfo.Create ... 626
TRVSeqItemInfo.CreateEx ... 627
Ancestor for Notes - TCustomRVNoteItemInfo ... 627
Properties ... 628
TCustomRVNoteItemInfo.Document ... 628
TCustomRVNoteItemInfo.NoteText ... 628
Methods ... 628
TCustomRVNoteItemInfo.CreateEx ... 628
TCustomRVNoteItemInfo.ReplaceDocumentEd ... 628
Endnote - TRVEndnoteItemInfo ... 629
Methods ... 630
TRVEndnoteItemInfo.Create ... 630
TRVEndnoteItemInfo.CreateEx ... 630
Footnote - TRVFootnoteItemInfo ... 630
Methods ... 632
TRVFootnoteItemInfo.Create ... 632
TRVFootnoteItemInfo.CreateEx ... 632
Sidenote - TRVSidenoteItemInfo ... 632
Properties ... 634
TRVSidenoteItemInfo.BoxProperties ... 634
TRVSidenoteItemInfo.BoxPosition ... 634

Methods ... 634
TRVSidenoteItemInfo.Create ... 634
TRVSidenoteItemInfo.CreateEx ... 634
Classes of Properties ... 635
TRVBoxProperties ... 635
Properties ... 635
TRVBoxProperties.Background ... 635
TRVBoxProperties.Border ... 635
TRVBoxProperties.Height, HeightType ... 636
TRVBoxProperties.VAlign ... 636
TRVBoxProperties.Width, WidthType ... 636
TRVBoxPosition ... 637
Properties ... 638
TRVBoxPosition.HorizontalAnchor ... 638
TRVBoxPosition.HorizontalXXX ... 639
TRVBoxPosition.PositionInText ... 640
TRVBoxPosition.RelativeToCell ... 640
TRVBoxPosition.VerticalAnchor ... 641
TRVBoxPosition.VerticalXXX ... 642
TextBox - TRVTextBoxItemInfo ... 643
Methods ... 643
TRVTextBoxItemInfo.Create ... 643
TRVTextBoxItemInfo.CreateEx ... 643
Reference to the Parent Note - TRVNoteReferenceItemInfo ... 644
Methods ... 645
TRVNoteReferenceItemInfo.Create ... 645
TRVNoteReferenceItemInfo.CreateEx ... 646
Ancestor for Text Fields - TCustomRVFieldItemInfo ... 646
Properties ... 646
TCustomRVPageNumberItemInfo.NumberType ... 646
Page Number - TRVPageNumberItemInfo ... 647
Methods ... 647
TRVPageNumberItemInfo.Create ... 647
TRVPageNumberItemInfo.CreateEx ... 648
Page Count - TRVPageCountItemInfo ... 648
Methods ... 648
TRVPageCountItemInfo.Create ... 648
TRVPageCountItemInfo.CreateEx ... 649
.7 Other Classes ... 649
ERichViewError ... 649
TCustomRVData and Others (TRichView Documents) ... 650
TCustomRVData.Edit ... 651
TCustomRVData.GetRVData ... 652
TCustomRVData.GetSourceRVData ... 652
TCustomRVFormattedData.GetOriginEx ... 653
TPrintableRV ... 653
TRVBooleanRect ... 653
TRVDeleteUnusedStylesData ... 653
TRVGraphicHandler ... 654
TRVIntegerList ... 656
TRVMathDocObject ... 656
TRVRect ... 657
TRVReportHelperWithHeaderFooters ... 657
Properties ... 657
TRVReportHelperWithHeaderFooters.MainDoc ... 657
TRVReportHelperWithHeaderFooters.UseStyleTemplates ... 658
Methods ... 658

TRVReportHelperWithHeaderFooters.AssignToRVPrint ... 658
TRVReportHelperWithHeaderFooters.Create ... 658
TRVReportHelperWithHeaderFooters.GetFooter ... 658
TRVReportHelperWithHeaderFooters.GetHeader ... 659
TRVReportHelperWithHeaderFooters.Reset ... 659
TRVThumbnailMaker ... 659
TRVUnitsRect ... 660
.8..Procedures and Functions ... 660
DrawControl and Others ... 660
GetRVESearchOptions ... 660
GetRVSearchOptions ... 661
RV_GetPrinterDC ... 661
RVGetFirstEndnote and Others ... 662
RVGetNoteTextStyleNo ... 663
RVIsURL, RVIsEmail ... 663
Unicode Conversion ... 664
Functions from RVGetText Unit ... 665
Functions from RVLinear Unit ... 666
Functions from RVMathItem Unit ... 667
Functions for TRVUnits conversion ... 667
.9..Types ... 667
TCheckpointData ... 667
TRVLoadFormat ... 668
TRVAnsiString ... 668
TRVBiDiMode ... 668
TRVBorderStyle ... 669
TRVBreakStyle ... 669
TRVCellVAlign ... 670
TRVCodePage ... 671
TRVColorMode ... 671
TRVDisplayOptions ... 671
TRVDocRotation ... 672
TRVESearchOptions ... 672
TRVExtraItemProperty ... 673
TRVExtraItemStrProperty ... 676
TRVFloatPosition ... 677
TRVFloatPositionKind ... 678
TRVFloatProperty ... 678
TRVFloatSize ... 678
TRVFontInfoProperties ... 679
TRVFReaderStyleMode ... 679
TRVHFType ... 680
TRVHTMLLength ... 680
TRVIntProperty ... 681
TRVItemBackgroundStyle ... 682
TRVLength ... 682
TRVOleDropEffects ... 682
TRVOpacity ... 683
TRVPageNumberType ... 683
TRVPageSet ... 683
TRVParaInfoProperties ... 684
TRVPrintingStep ... 684
TRVRawByteString ... 685
TRVReaderStyleMode ... 685
TRVSaveFormat ... 685
TRVSaveOptions ... 686
TRVSearchOptions ... 689

TRVSeqType ... 689
TRVStrProperty ... 690
TRVStyleLength ... 690
TRVStyleTemplateId ... 690
TRVStyleTemplateName ... 691
TRVStyleUnits ... 691
TRVTableBorderStyle ... 691
TRVTag ... 692
TRVTextDrawStates ... 692
TRVUndoType ... 693
TRVUnicodeString ... 694
TRVUnits ... 694
TRVVAlign ... 695
10 Global Constants and Variables ... 696
Clipboard Related Constants and Variables ... 698
Constants Related to HTML and Text ... 698
Constants Related to the Caret ... 699
RichViewAllowCopyTableCells ... 700
RichViewAlternativePicPrint ... 700
RichViewApostropheInWord ... 700
RichViewAutoAssignNormalStyleTemplate ... 700
RichViewCompareStyleNames ... 701
RichViewDoNotCheckRVFStyleRefs ... 701
RichViewDoNotMergeNumbering ... 701
RichViewEditDefaultProportionalResize ... 702
RichViewEditEnterAllowsEmptyMarkeredLines ... 702
RichViewJustifyBeforeLineBreak ... 702
RichViewMaxPictureCount ... 702
RichViewMaxThumbnailCount ... 703
RichViewPixelsPerInch ... 703
RichViewResetStandardFlag ... 703
RichViewSaveHyperlinksInDocXAsFields ... 704
RichViewShowGhostSpaces ... 704
RichViewStringFormatMethod ... 704
RichViewTableAutoAddRow ... 705
RichViewTableDefaultRTFAutofit ... 705
RichViewUnicodeInput ... 705
rveipcXXX constants ... 706
rvespcXXX constants ... 709
RVGraphicHandler ... 710
RVIsCustomURL ... 710
RVParagraphMarks ... 711
rvsXXX constants ... 711
RVThumbnailMaker ... 713
RVVisibleSpecialCharacters ... 714
RVEMPTYTAG ... 714
11 How To... ... 714
change page size and orientation ... 715
draw page numbers, headers and footers ... 715
implement commands like "make bold", "apply font", etc. ... 716
implement Find and Replace dialogs ... 716
switch insert/overtype mode ... 717
move the caret to the beginning or to the end of document in editor ... 717
make Unicode editor ... 718
make a plain text editor ... 718
implement smart indenting ... 720
combine several TRichView documents in one ... 720

use third-party graphic classes with RichView 720
create a chat window 722
remove text formatting 722

Index 725

1 TRichView Reference

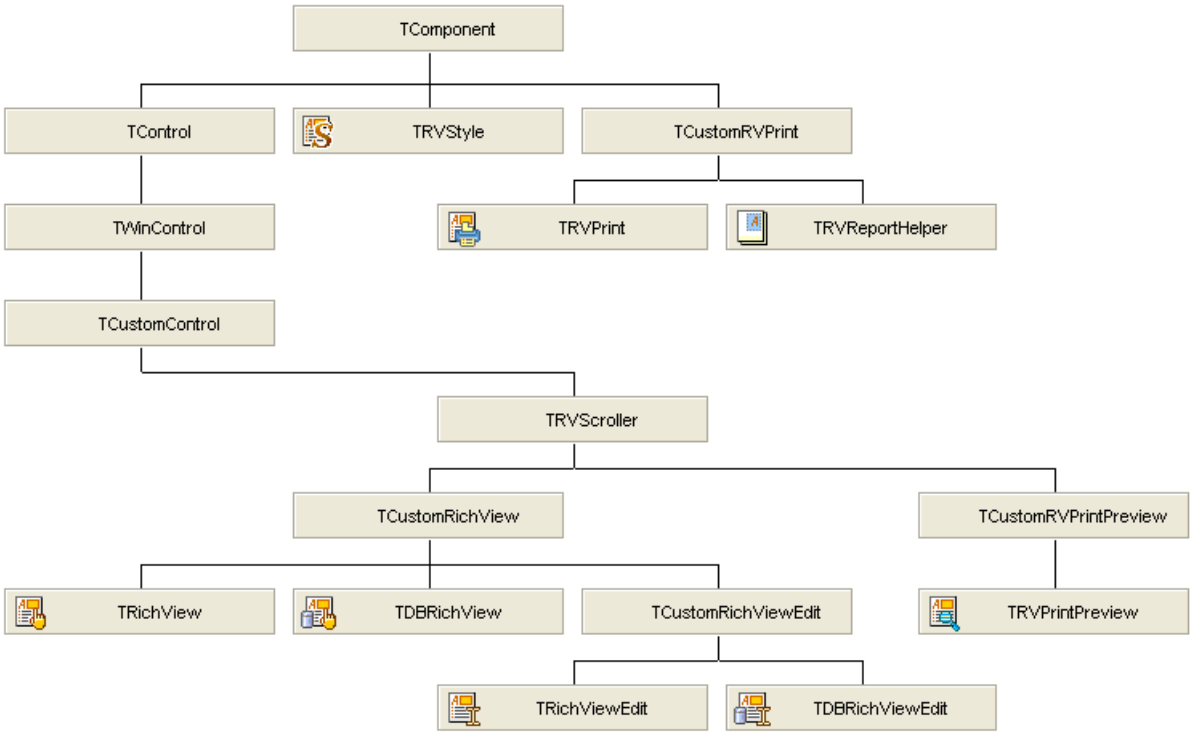
Copyright © Sergey Tkachenko (*www.trichview.com*)
(*svt@trichview.com*)

Version 17 (what's new? 30)

Components

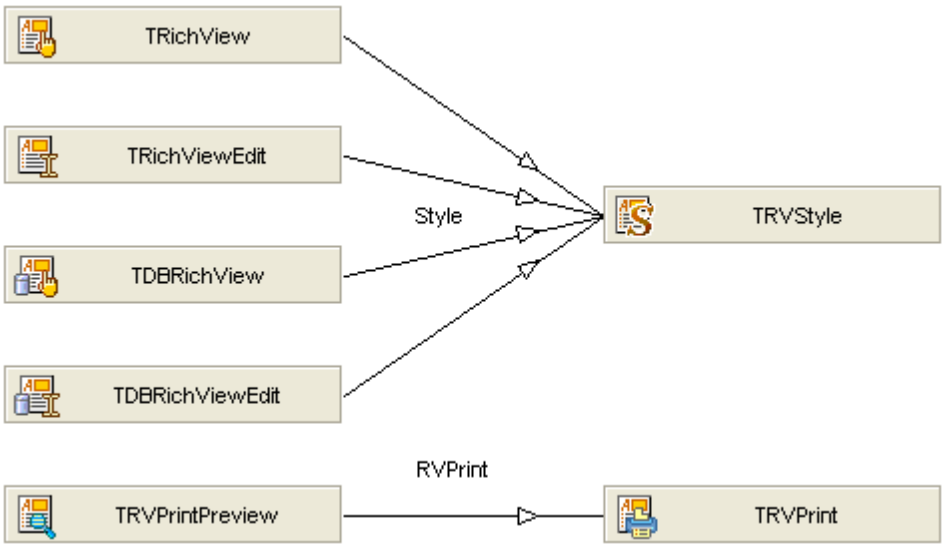
Icon	Class Name	Description
	TRVStyle 418	Component for defining visual appearance of RichView, RichViewEdit, DBRichView, DBRichViewEdit. It contains some properties which can be used by several RichViews.
	TRichView 162	Component for displaying hypertext documents (containing pictures, controls, tables)
	TRichViewEdit 316	Component for editing hypertext documents (containing pictures, controls, tables)
	TDBRichView 407	Data-aware version of RichView
	TDBRichViewEdit 410	Data-aware version of RichViewEdit
	TRVPrint 513	Component for printing documents contained in the components above
	TRVPrintPreview 417	Component allowing to show document as it will be printed by RVPrint
	TRVReportHelper 540	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter 532	Component allowing to use Microsoft Office text converters (import and export in different file formats)

Class Hierarchy



Hierarchy of TRichView classes

Diagram of Using



How TRichView classes use each other

Related topics

- version history and compatibility 29
- overview of features 63
- overview of item types 118 (documents consist of items)
- overview of tables 144

- how to... 714

1.1 Version History

Version History and Compatibility Issues

- New after version 17 29
- New in version 17 30
- New in version 16 32
- New in version 15 34
- New in version 14 36
- New in version 13 39
- New in version 12 41
- New in version 11 42
- New in version 10 44
- New in version 1.9 48
- New in version 1.8 50
- New in version 1.7 52
- New in version 1.6 55
- New in version 1.5 56
- New in version 1.4 58
- New in version 1.3 60
- New in version 1.2 60
- Compatibility with older (freeware) version 62

1.1.1 New after version 17

This topic includes changes of v17.2.

Compatibility issues

- Methods for compatibility with ancient versions of TRichView are removed.
- Redundant TRichView.Add*** methods are marked as deprecated. Only a single method for each item type is recommended. These methods have optional parameters. For example, use AddBreak 196 instead of AddBreakEx, AddBreakTag, AddBreakExTag.

All String, TRVRawByteString, TRVAnsiString properties and parameters of methods and events are changed to TRVUnicodeString 694, where it is possible.

Changed events:

- In TRichView: OnAssignImageFileName 274, OnImportPicture 279, OnItemAction 280, OnItemHint 281, OnReadHyperlink 284, OnRVDblClick 287, OnRVRightClick 291, OnRVFControlNeeded 288, OnRVFPictureNeeded 289, OnSaveComponentToFile 292, OnSaveDocXExtra 293, OnSaveHTMLExtra 295, OnSaveImage2 296, OnSaveItemToFile 297 (additionally, Unicode: Boolean parameter is removed), OnSaveParaToHTML 298, OnSaveRTFExtra 299, OnSpellingCheck 301, OnWriteHyperlink 302
- In TRichViewEdit: OnDropFile 392, OnItemTextEdit 394
- In TRVStyle: OnDrawStyleText 448

If you used these events in your application, you need to change types of parameters manually. Since TRVUnicodeString is defined in RVTypes unit, make sure that this unit is listed in "uses" of your forms.

Unicode

Since this version, all text is stored in our components as Unicode, for all versions of Delphi. All string properties and parameters are made Unicode, when possible.

Because of this change, some properties became obsolete. They are not removed, but do nothing:

- TFontInfo.Unicode 480

- TRVStyle.DefUnicodeStyle[422]
- TRVRTFReaderProperties.UnicodeMode[310]

New events

TRichView.OnReadMergeField[286] allows reading merge fields from RTF.

New features

- support for Unicode characters having codes greater than $FFFF
- support for pasting and drag&dropping of Unicode URLs
- GetCurrentCharacter[665] may return more than one characters, if they assemble a single glyph;
- entering a character by pressing and holding Alt and typing decimal character code on the numeric keypad, starting from '0' (Num Lock must be on)
- entering a character by typing its hexadecimal code (directly in the editor) and them pressing Alt + X. Pressing Alt + X again converts it back to a hexadecimal number.
- loading graphic from RVF even if the specified graphic class is not available (or not registered): the proper graphic class is chosen by the graphic content. New option for RichView.RVFOptions[188]: *rvfoIgnoreGraphicClasses*, it allows ignoring names of graphic classes specified in RVF.

1.1.2 New in version 17

This topic includes changes of v17.2.

Compatibility issues

- New parameter in TRVPageCountItemInfo[648].CreateEx[649]
- Improvements in the line breaking algorithm (in some cases, lines may wrap differently)
- When choosing format while inserting data as a result of OLE drag&drop[89], the editor gives a higher priority to URL than it was before
- *rvddHTML* is included in the default value of TRichViewEdit[316].AcceptPasteFormats[321] property.
- Because of "deactivation" of images (see below), you cannot store a pointer to images (TGraphic objects), because the image can be "deactivated" and TGraphic object can be destroyed
- Because of "deactivation" of images, all TGraphic objects used in TRichView must implement SaveToStream and LoadToStream methods (there are some third-party graphic classes that can load themselves but cannot save).
- TRichView.RTFReadProperties[187].StoreImagesFileNames property is removed.
- Previously, the functions for enumerations of notes and text boxes[662] returned hidden[76] notes/boxes if *rvoShowHiddenText* was included in Options[182], even if AllowHidden parameter was *False*. Now, AllowHidden parameter is processed strictly.
- TCustomRichView.OnURLNeeded event is removed (use OnWriteHyperlink[302])
- TRichView.RTFReadProperties.SetHeader and SetFooter are deprecated. Use TRichView.SetHeader[264] and SetFooter[263] instead. Note: they affect not only RTF and DocX, but RVF as well.
- type of the Sender parameter in TRVReportHelper.OnDrawHyperlink[556] and OnDrawCheckpoint[556] is changed from TRVReportHelper to TCustomRVPrint.

Mathematical expressions[143]

TRichView includes a new item type displaying mathematical formulas (TRVMathItemInfo[621]). It requires Delphi XE4 or newer.

Tables[565]

New feature: colors of groups or rows and columns (alternate colors). New properties:

- HeadingRowColor[583], LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor;

- RowBandSize, ColBandSize 575.

A new optional parameter (EditCellAfter) is added in DeleteRows 587 and DeleteCols 586 methods.
New property: CellOverrideColor 573.
New event: OnDrawBackground 603.

RVF 93

New DocObjects 174 property allows saving custom objects in RVF.

Headers and footers

TRichView,SetHeader 264 and SetFooter 263 define documents as headers/footers. They can be used not only for RTF (as before), but also for RVF and HTML.
A new option 686 is available for HTML saving, *rvsoHeaderFooter*.

Printing

New property TRVPrint 513.VirtualPrinter 522 allows drawing pages on any canvas using the specified page size and DPI. New methods DrawPage and DrawPageAt 524.

New optional parameters in TRVPrint 513.Print 526 and PrintPages 527 allow printing all/odd/even pages, in normal/reverse order.

"Widow and orphan control" is implemented, see *rvpaoWidowOrphanControl* 487 option of paragraphs.

The following events and properties are moved from TRVReportHelper 540 to its parent class: OnDrawCheckpoint 556, OnDrawHyperlink 556, NoMetafiles 551. Now you can use them in TRVPrint 513 as well.

A new property is added in TRVPrint 513 and TRVReportHelper 540: MetafilePixelsPerInch 551.

TRVPageCountItemInfo 648 now has NumberType 646 property, like TRVPageNumberItemInfo 647,

OLE Drag&drop 89

New events of TRichViewEdit 316: OnBeforeOleDrop and OnAfterOleDrop 388.

New option *rvoDragDropPicturesFromLinks* for TRichViewEdit.EditorOptions 325. If included, and the dropped URL is a link to a picture, the editor attempts to insert it as a picture, not as a link.

Fields

New property is added to text styles: EmptyWidth 469. It defines width of empty text items of this style. Text items of this style are processed specially in editor. They allow implementing editable fields in documents. The demo "Assorted\Fields\FillInGaps" is changed to use EmptyWidth.

Image activation/deactivation

Starting from this version, TRichView deactivates rarely used images (i.e. stores them to memory streams and destroys their graphic objects). This feature allows saving GDI resources and memory.

New RichViewMaxPictureCount 702 global variable.

Loading external images

A new option is added to TRichView.Options 182: *rvoAssignImageFileNames*. If included, the component assigns "image file name" properties when reading external images. Previously, RTFReadProperties 187 .StoreImagesFileNames was used for this purpose for external images in RTF; this property is removed.

New event: OnAssignImageFileName 274 allows modifying file names before assigning to items/cells.

Item properties

rveipcPercentWidth additional integer property 706 is added to controls 126.

rvepSpacing, rvepBorderWidth, rvepBorderColor, rvepOuterHSpacing, rvepOuterVSpacing integer properties[673] work for labels[133].

rveipcRemoveSpaceBelow additional integer property[706] is added to labels[133].

Caret

The editor uses the system caret width (by default).

New global variables controlling the caret size and position[699] are added. The new options include:

- setting the caret width equal to the next character/item width;
- using OnMeasureCustomCaret[394] event for a standard caret (not only for a custom caret);
- options for aligning wide carets horizontally.

Other changes

New TRichView.UseVCLThemes[195] property allows using colors of active VCL theme (in Delphi XE2+)

New TRichViewEdit event: OnDropFile[392]. It's more easy to use then OnDropFiles[393], and occurs not only on drag&drop, but also on pasting.

New TRichView event: OnCMHintShow[275] allows displaying custom hints.

Text flow options (ClearLeft[171] and ClearRight[172]) are shown as vertical lines around paragraph/line-break marks, if *rvoShowSpecialCharacters* is included in TRichView.Options[182]. Text flow options can be exported to DocX.

New method TRVStyleTemplateCollection.InsertFromRVST[463], it allows applying styles from a file to existing documents.

Packages are separated into runtime and designtime packages.

1.1.3 New in version 16

Compatibility issues

- Since this version, text measuring and drawing is performed using ***Uniscribe***; so text size may be different (for example, Uniscribe always uses text kerning); usually, text formatting is slower with Uniscribe (if you do not need support of bidirected text); you can switch back to Windows API, see "Uniscribe" below
- Uniscribe text output (and sometimes Windows text output) is not compatible with metafiles; see "Uniscribe" below;
- ParaStyle.Alignment[483] = *rvaJustify* is implemented by modifying space character widths when drawing text, while older versions of the components drew each word separately. The new method complicates custom drawing of text[448]; you can return the old justification method by activating RVOLDJUSTIFY in RV_Defs.inc; however, this directive disables ParaStyle.Alignment[483] = *rvaDistribute* (it works like *rvaJustify*)
- TRVStyle.OnDrawStyleText[448] has new parameters.
- RichViewSafeFormatting constant is removed (replaced with RichViewStringFormatMethod[704])
- applications compiled with older versions of TRichView cannot read RVF files containing tables having changed values of Opacity (Opacity<>100000).
- starting from this version, all projects for C++Builder 6 that use TRichView must include $(BCB)\Lib\Psdk\msimg32.lib. Otherwise, linking ends with an error about unresolved external "AlphaBlend". No changes are required for Delphi and newer versions of C++Builder
- the algorithm for calculation of width of table columns is improved and may produce different results.

Uniscribe

Since this version, the components use ***Uniscribe*** for text drawing and measuring.

You can switch back to Windows API using TRVStyle.TextEngine[436] property.

By default, Uniscribe text output is not compatible with metafiles. You can turn on metafile compatibility for TRVReportHelper[540] and TRVPrint[513] components using MetafileCompatibility[551] property.

New paragraph properties

New value for Alignment[483] – *rvaDistribute*. It aligns paragraph content to the both left and right sides by adding space between all characters.

New property LastLineAlignment[483] defines alignment for the last line of justified and distributed paragraphs.

Improved attributes in bi-directed text

The following features were previously disabled for bi-directed text; they work now:

- TextStyle.CharSpacing[468]
- ParaStyle.Alignment[483] = *rvaJustify* (and *rvaDistribute*)
- TRVPrint.PreviewCorrection[552]
- custom drawing of selection (although, custom drawing is more complicated now)

All these functions are implemented both for Windows and Uniscribe API.

Vertical text

Special options[609] are implemented for vertical text in table cells.

In addition to rotation[611] by 90°, the component can automatically apply vertical versions of fonts (useful for Chinese, Japanese, Korean) and reverse line order (useful for traditional Mongolian)

Formatting

The components need less memory for documents containing justified text.

Improvements

- documents containing long strings in text items are formatted much faster than before; improved fast text formatting method now can be used not only in TRichViewEdit, but in TRVPrint and in ScaleRichView;
- justified text is formatted much faster than before.

Table sizing algorithm is improved, it processes wide spanned cells much better.

Styles

New RVFOption[188]: *rvfoSaveStyleTemplatesOnlyNames*. It allows creating multiple RVF documents using the same collection of StyleTemplates[435].

Transparency

Background of paragraphs, tables, table cells, sidenotes and text boxes can be semitransparent: a new Opacity property is added to these objects.

New property TRVReportHelper.AllowTransparentDrawing[541] allows using methods for semitransparent drawing. It is useful for drawing over other images.

See the topic about semitransparent objects[91].

Support of high DPI screen modes

Size of touch screen selection handles[431] is changed proportionally to the screen resolution. Sizes of list markers are stored as Size (instead of Height) property in DFM and RVF files, so they do not depend on screen resolution any more.

Installation and directory structure

Starting from this update, TRichView is installed automatically in Delphi and C++Builder IDE.

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-file is moved to "Source\Include" folder.

For RAD Studio XE8+, this help file is integrated in IDE (both F1 help, and menu "Help | Third-Party Help")

Other changes

TRichViewEdit.RemoveCurrentPageBreak[367] deletes a page break from the current paragraph (not from the current item as before)

New RichViewSaveHyperlinksInDocXAsFields[704] global variable.

Ctrl + Delete deletes a word to the right of the caret.

1.1.4 New in version 15

Compatibility issues

- The following graphic classes are registered automatically (Do not call RegisterClass or RegisterClassAlias for these classes):
  - (for Delphi 2009+) TPngImage. Also, TPngImage is automatically registered as the default PNG format and as HTML graphic format (see TRVGraphicHandler[654])
  - (for Delphi 2010+) TWicImage. Also, thumbnails are activated by default for TWicImage.
  - (for Delphi 2007+) standard TGifImage – if you include RVGifAnimate2007 unit in your project;
  - Anders Melander's TGifImage – if you include RVGifAnimate unit in your project;
  - TJvGifImage from JEDI's JVCL – if you include RVJvGifAnimate unit in your project.
- A parameter is added to TRVPrint.GetHeaderRect[525], GetFooterRect[525].
- New values are added to TRVUndoType[693].

New types of items

- sidenotes[137] and text box items[139] are printed in floating boxes.
- "page number" fields[141]
- "page count" fields[142]

DocX export

TRichView can save Microsoft Word 2007+ files (*.docx).

Minimal required versions for this feature: Delphi: 4; C++Builder: 2006.

New methods of TCustomRichView[162]: SaveDocX[249], SaveDocXToStream[250].

New OnSaveDocXExtra[293] event.

New options for TCustomRVFontInfo[466].Options[472]: *rvteoDocXCode* and *rvteoDocXInRunCode.*

New properties of TRVOfficeConverter[532]: ExcludeDocXImportConverter, ExcludeDocXExportConverter[534].

Touch screen support

Multi-touch screen operations are supported (for Delphi 2010+). The standard touch-related properties and events (Touch and OnGesture) are published.

All visual controls support a panning gesture. TRVPrintPreview[417] supports a two finger zooming as well.

Selection handles are implemented in TRichView[162] and TRichViewEdit[316]. They are shown for the selection if it was created by a double (word selection) or a triple (paragraph selection) tap. Alternatively, you can show and hide them using SelectionHandlesVisible[192] property.

New property: TRVStyle.SelectionHandleKind[431].

Printing and drawing

New properties defining sizes in TRVPrint[513]: Margins[520], HeaderY[519], FooterY[519], Units[521]. The new properties replace LeftMarginMM, TopMarginMM, RightMarginMM, BottomMarginMM, HeaderYMM, FooterYMM properties. The old properties still work. The main advantage of the new properties is a higher precision. Even if Units is millimeters, you can define properties more accurately, because sizes are floating point values.

Now you can define different headers and footers for the first page, and for odd/even pages. TRVPrint.SetHeader[529] and SetFooter[528] have additional optional parameter, specifying header/footer type. New properties: TitlePage[521], FacingPages[518].

New property: PageNoFromNumber[552] specifies the starting value for "page number" fields[141]. In TRVReportHelper[540], page numbers can be requested in OnGetPageNumber[543] event.

New TRVReportHelper[540].NoMetafiles property allows drawing metafiles as bitmaps.

Editing

New methods allowing to change properties of TRichViewEdit as an editing (undoable) operation: SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd[370], SetBackgroundImageEd[371]. They allow changing margins, DocParameters, background properties.

New methods allowing to change properties of items in TRichViewEdit as an editing (undoable) operation:

- SetCurrentItemExtraIntPropertyEx[378], SetItemExtraIntPropertyExEd[378];
- SetCurrentItemExtraStrPropertyEx[379], SetItemExtraStrPropertyExEd[379].

They support more properties than the old methods.

Text attributes

New property SizeDouble[475] allows defining font size with higher precision.

Visual changes

- New TRVStyle[418].FontQuality[424] property.
- Smooth image scaling[91].

Other changes

Packages for Delphi and C++Builder XE4, XE5, XE6, XE7 are added.

Footnotes[136]/endnotes[135] displaying a character (instead of a number) are supported.

All work with graphic classes is encapsulated in TRVGraphicHandler[654] class, available via RVGraphicHandler[710] variable.

RichViewJustifyBeforeLineBreak[702] variable allow to turn on justifying lines before line breaks.

Faster processing of hidden text.

New properties:

- TRVScroller[544].NoHScroll[546];
- TCustomRichViewEdit[316].ForceFieldHighlight[327].

New option for TCustomRichView[162].Options[182]: *rvoPercentEncodedURL*.

1.1.5 New in version 14

Compatibility issues

- Item tags[68] are strings, they are not integers any more. The default tag value is changed from 0 to ''. If you used tags in an integer mode, use StrToInt and IntToStr when calling TRichView methods returning and accepting tags. Not only methods having Tag parameters, but some events are affected (OnReadHyperlink[284], OnRVControlNeeded[288]). In existing projects, change the types of Tag parameters of these events handlers from Integer to TRVTag manually. RVOLDTAGS can be defined in RV_Defs.inc to make tags integers (not recommended).
- Parameters of OnRVFPictureNeeded[289] were changed. If you used this event, change events handlers manually (or delete the old event handler and create a new one).
- For rotated cells, coordinates stored in DrawItems (a low-level undocumented property) do not define the position of items on the screen directly.
- TRichView.LoadText[246], LoadTextFromStream[247], SaveText[255], SaveTextToStream[255], TRichViewEdit.InsertTextFromFile[361] have a new parameter. This parameter is optional, but in old versions of C++Builder, it must be added explicitly in the code (use CP_ACP for compatibility).
- A text search behavior can be modified with the new "smart start" option (see TRichView.SearchText[256] and TRichViewEdit.SearchText[369]). GetRVSearchOptions[661] and GetRVESearchOptions[660] include this option by default, so it may affect existing projects.
- Changes in Unicode line wrapping procedure: '.' , '!', ',', ';' characters are treated (almost) as alphabetic characters now (because they are often used in phrases like '.Net' or in URLs).
- Changes in DB components: If *rvfoLoadBack* is in RVFOptions[188], TDBRichView[407] clears BackgroundBitmap[169] before loading data from its field. If *rvfoLoadBack* is in RVFOptions[188] and FieldFormat[413]=*rvdbRVF*, TDBRichViewEdit[410] clears BackgroundBitmap[169] before loading data from its field.
- PNG images are saved to RTF as PNG images only if *rvrtfSavePngAsPng* is included in RTFOptions[185]. It is included by default. However, for existing projects, you may need to set it manually.
- The declaration of TRVLongOperation[283] type is changed.
- The following global variables were removed from RVTable.pas: RichViewTableGridStyle, RichViewTableGridStyle2, RichViewTableGridColor. They were replaced with properties of TRVStyle[425].
- A new option *rvoShowGridLines* in TRichView.Options[182] is set by default, but for existing projects you may need to set it manually, otherwise grid lines of tables will not be shown by default.

Style templates

StyleTemplates[435] represent "real" text and paragraph styles. When style templates are used, an appearance of text and paragraphs are still defined in TextStyles[437] and ParaStyles[431]. Items a document refer to items in TextStyles and ParaStyles, while items in TextStyles and ParaStyles refer to items in StyleTemplates. Style templates allows modifying TextStyles and ParaStyles.

By default, style templates are not used. To activate using of style templates, assign TRichView.UseStyleTemplates[195]=*True*.

TRichView.StyleTemplateInsertMode[193] specifies how style templates affect insertion of RVF and RTF data.

When the collection of style templates has been modified, OnStyleTemplatesChange[302] occurs.

In TRichViewEdit, you can apply the specified style template to the selection using the methods: ApplyStyleTemplate[333], ApplyTextStyleTemplate[334], ApplyParaStyleTemplate[332]. Style templates can be edited using ChangeStyleTemplates[338] method.

The function RVGetNoteTextStyleNo[663] has an additional optional parameter: a name of a style template for using for a note character.

TRVStyle.MainRVStyle[430] property allows using a single collection of style templates for several documents.

See the overview topic about style templates[74].

Rotation

Table cells can be rotated[611] by 90°, 180°, or 270°.

A new method returning item coordinates: GetItemCoordsEx[227]; it takes cell rotation into account.

String tags

Since version 13.2, item tags[68] are strings (Unicode for Delphi 2009 or newer, ANSI for older versions of Delphi). No more typecasting from integer to PChar!

New property: cell tags[612].

64-bit

The both 32-bit and 64-bit compilers are supported in RAD Studio XE2+.

Note: TRVOfficeConverter[532] can be compiled in 64-bit application, but lists of converters will be empty (because converters are 32-bit DLLs and cannot be used in 64-bit applications).

Skins (Delphi XE2+ visual UI styles)

The components support visual UI styles introduced in RAD Studio XE2:

- all visual components display scrollbars and borders according to styles;
- TRVPrintPreview[417] adjusts its colors (background, page border, shadow) according to the current style.

Text files

A new optional parameter for TRichView.LoadText[246], LoadTextFromStream[247], SaveText[255], SaveTextToStream[255], TRichViewEdit.InsertTextFromFile[361]: CodePage. It allows to specify a code page of the file (this parameter is used only for Unicode text in a document; for ANSI text, it is ignored).

Drag and drop: when accepting text files, TRichViewEdit detects their encoding (UTF16, UTF8 or ANSI).

When saving Unicode text, characters of "Symbol" and "Wingdings" fonts are converted to the most similar Unicode characters.

RVF

VAlign property of *bullets*[128] and *hotspots*[129] is saved to RVF.

The parameters of TRichView.OnRVFPictureNeeded[289] were changed. Now, this event is called not only for pictures[122] and hot-pictures[125], but also for tables[130] and table cells to read background images. This change solved the following problems:

- ability to request table and cell background images;
- access to additional item properties.

RTF

New option in TRichView.RTFOptions[185]: *rvrtfSavePngAsPng*. If included (default), PNG images (of the class type registered with RV_RegisterPngGraphic) are saved as PNG in RTF (as it was before this version). You can exclude this option to save them as a bitmap or a metafile.

Default processing if OnReadHyperlink[284]/OnWriteHyperlink[302] is not assigned.

HTML saving

TRichView.SaveHTMLEx[251] can save stretched background images (previously, they were saved as centered).

TRichView.SaveHTMLEx[251] saves list markers better (both in normal and rvsoMarkersAsText[686] modes).

New "hidden" parameters in TRichView.OnSaveImage2[296] event.

Default processing if OnWriteHyperlink[302] is not assigned.

SaveHTMLEx[251] saves characters of "Symbol" and "Wingdings" fonts as the most similar Unicode characters (it was implemented for "Symbol" before, now it is improved).

Visual changes

- Better displaying of text background, new TRVStyle.TextBackgroundKind[435] property.
- All adjacent hypertext items[69] having the same non-empty tags[68] are highlighted below the mouse pointer; so visually they represent a single hyperlink.
- New properties[673] for non-text items: background color, border width and color, spacing around the border.
- Footnote[136] and endnote[135] characters are kept on the same line as the previous item.
- Better calculation of superscript position.

Clipboard

New: TRichViewEdit can paste URL. A new method is available: PasteURL[365].

New property: AcceptPasteFormats[321] allows to restrict a list of formats that the editor can paste.

New property: DefaultPictureVAlign[325] defines an alignment for pasted and dropped images.

DB components

If *rvfoLoadBack* is in RVFOptions[188], TDBRichView[407] clears BackgroundBitmap[169] before loading data from its field.

If *rvfoLoadBack* is in RVFOptions[188] and FieldFormat[413]=*rvdbRVF*, TDBRichViewEdit[410] clears BackgroundBitmap[169] before loading data from its field.

Printing

Tables

New property for a table row: KeepTogether[615]. New table methods: SetRowPageBreakBefore, SetRowKeepTogether[600].

New option for table.PrintOptions[579]: *rvtoContinue*.

Other

New property: TCustomRVPrint.IgnorePageBreaks[550].

Tables

New properties of TRVStyle defining the appearance of grid lines: GridColor, GridStyle, GridReadOnlyColor, GridReadOnlyStyle[425]. They replace global variables used in older versions. New option *rvoShowGridLines* in TRichView.Options[182].

Other changes

Better processing of Up and Down keys when moving in and out of table cells.

New event of TRichViewEdit[316]: OnCheckStickingItems[391] event allows to stick items together.

New typed constant: RichViewDoNotMergeNumbering[701].

"Smart start" search option for TRichView.SearchText[256] and TRichViewEdit.SearchText[369].

TRichView.OnProgress[283] now occurs not only when reading an RTF, but also when saving an RTF, reading and writing an RVF, writing an HTML, reading and writing a text, calling TRVOfficeConverter.ImportRV[538] and ExportRV[537].

Better table layout algorithm.

New methods for TRVStyle: FindTextStyle, FindParaStyle[440], TextStyles.FindSuchStyleEx[455], ParaStyles.FindSuchStyleEx[458].

1.1.6 New in version 13

Compatibility issues

1. This version introduces a new line wrapping algorithm, so layouts of existing documents may become different. The old line wrapping mode can be returned using TRVStyle.LineWrapMode[428] property (*rvWrapSimple* value). But even with this value, a line breaking can be different for Unicode[99] text in paragraphs with Alignment[483]=*rvaJustify*.
   RichViewWrapAnywhere (global constant) was removed, superseded by TRVStyle.LineWrapMode[428] (*rvWrapAnywhere* value).
2. New parameter in ApplyStyleConversion[333] and ApplyParaStyleConversion[332]. It is optional in Delphi, but for existing C++Builder projects, *True* must be added as the last parameter.
3. This version can write twips instead of pixels in RVF files. Applications compiled with older versions of RichView interprets all sizes an pixels, so if they load such RVF files, all lengths will be too large.
4. This version defines widths of breaks as TRVStyleLength[690] instead of bytes. All applications that use GetBreakInfo[218] and GetCurrentBreakInfo[341] methods must correct the type of variable used read a line width.
5. Paragraph borders are drawn slightly different.

New line wrapping algorithm

New line wrapping algorithm takes connections between text items into account.

New property TRVStyle.LineWrapMode[428] allows changing a line wrapping mode.

Hidden items

Some items can be hidden[76].

Measuring in twips instead of pixels

Previously, all sizes in documents were defined in pixels. Since this version, you can define measuring units in TRVStyle[418].Units[437] property. Instead of pixels, you can use twips[106].

Advantages:

- you can define small widths with better precision than 1 screen pixel;
- you can create user interface allowing entering sizes in different units (fractional inches, cm) and convert them to/from twips without losing precision;
- values in twips are independent from the screen resolution, even if RichViewPixelsPerInch[703]=0.

Shared item content

A new item property is added: rvepShared[673]. It allows for several picture items[122] to share the same graphic object. For a control item[126], it allows to reuse its control after the item deletion.

Incremental printing

TRVPrint[513] and TRVReportHelper[540] components have new properties: MinPrintedItemNo[551] and MaxPrintedItemNo[550]. If these properties define a subrange of items, only this subrange (and only the pages containing it) is printed.

These properties allow implementing an incremental printing: when the next portion of the document is ready, it can be printed below the previously printed fragment. This feature can be used for printing accounting journals, logs, etc.

Editing

Triple click selects paragraph.

Horizontal coordinate is kept when moving the caret up or down with the keyboard.

When selecting from bottom to top, CurTextStyleNo[324] is defined as a style of the last selected text (to the right of the caret, not to the left as usual).

The methods ApplyStyleConversion[333] and ApplyParaStyleConversion[332] have the new parameter (Recursive) allowing to prevent processing of table cells.

Drawing and formatting

Improved table layout algorithm.

New option in RVVisibleSpecialCharacters[714]: *rvscTab*. RVParagraphMarks[711] global variable allows defining how paragraph breaks and line breaks are displayed in "show special characters" mode.

New property TRVStyle.DisabledFontColor[422] allows change text color in disabled controls.

Calls of TRVStyle.OnDrawTextBack[449] are optimized; now it is called only once for one text item on the line (previously, for justified paragraphs, it was called separately for each word)

TCustomRVPrint.GetPageNo[553] returns the page index for the given position in the document.

Saving and loading

Improved saving of footnotes[136] and endnotes[135] in plain text files.

Smarter replacing space characters to   when saving HTML.

New text properties for TRichView.DocParameters[176]:

- Author[311],
- Title[314],
- Comments[311]

Hypertext

rvoNoReadOnlyJumps option from EditorOptions[325] disallows automatic switching to a hypertext mode when TRichViewEdit becomes read-only[328].

1.1.7 New in version 12

Compatibility issues

Assignment to PageBreaksBeforeItems 184 requires reformatting even if the given item starts a paragraph (because it clears text flow around side-aligned items).

OutlineLevel 488 property allows saving headings in HTML and RTF, so be careful if you assumed that saved HTML and RTF files do not have headings.

Some existing code may assume that the text searching functions can select a substring only in a single text item 120. While the new behavior (searching in multiple text items) requires specifying a new option in the parameter of SearchText methods explicitly, GetRVSearchOptions 661 and GetRVESearchOptions 660 functions always return this new option.

Left- and right-aligned objects

Two new values are available for TRVVAlign 695 type: *rvvaLeft* and *rvvaRight*. They can be used for all non-text items, except for tables 130, *breaks* 125, tabs 121, list markers 131, footnotes 136 and endnotes 135.

New methods are available for accessing vertical alignment:

- TRichView.GetItemVAlign 230,
- TRichView.SetItemVAlign 268,
- TRichViewEdit.GetCurrentItemVAlign 348,
- TRichViewEdit.SetItemVAlignEd 386,
- TRichViewEdit.SetCurrentItemVAlign 386.

Properties and methods for controlling text flow around side-aligned images:

- TRichView.ClearLeft 171,
- TRichView.ClearRight 172,
- TRichViewEdit.ClearTextFlow 339.

Showing invisible characters

New options for RVVisibleSpecialCharacters 714 allowing to display/hide lines for floating objects, paragraph directions, printing options and outline levels 488.

New colors can be defined in TRVStyle:

- FloatingLineColor 423,
- SpecialCharactersColor 435.

Headings

New property in paragraph style: OutlineLevel 488 defines headings.

Other

New option of text searching (*rvsroMultiItem* for TRichView.SearchText 256, *rvseoMultiItem* for TRichViewEdit.SearchText 369) allows the searched string matching substrings of several text items.

New event in TRVStyle: OnAfterApplyStyle 443.

New event in TRichView: OnGetItemCursor 277

New property in TRVLabelItemInfo: RemoveInternalLeading 619

New methods in TRVOfficeConverter: IsValidImporter 539 allows checking a file before importing.

New values in TRichView.RVFOptions 188 and RVFWarnings 191 (*rvfoIgnoreUnknownCtrlProperties*, *rvfwUnknownCtrlProperties*)

1.1.8 New in version 11

The main feature of this update is compatibility with Delphi and C++Builder 2009.

Update: since TRichView 18, most information here is obsolete. Text is no more stored as ANSI/"raw Unicode", it is stored as Unicode.

Compatibility issues

New string types

New string types are declared in RVTypes.pas: TRVAnsiString[668], TRVUnicodeString[694], TRVRawByteString[685]. These string types are used instead of String and WideString in many methods, properties and events.

TRVAnsiString is used for parameters and variables containing ANSI strings.

TRVUnicodeString is used for parameters and variables containing Unicode strings.

TRVRawByteString is used for parameters and variables containing either ANSI or “raw Unicode” strings.

The standard String type is used for parameters and variables containing ANSI strings in older versions of Delphi/C++Builder, and Unicode strings for Delphi/C++Builder 2009.

Changed parameters of events

In the following events, type of some parameters was changed from String to TRVRawByteString[685]: OnDrawStyleText[448], OnReadHyperlink[284], OnItemAction[280], OnSaveItemToFile[297], OnRVDblClick[287], OnRVRightClick[291], OnItemTextEdit[394]. In existing projects, change types of parameters in these events manually.

In OnSaveComponentToFile[292] event, it is assumed that OutStr parameter is ANSI string (for Delphi/C++Builder versions prior to 2009). When saving to UTF-8 HTML files, it is converted to UTF-8 automatically. In previous versions of TRichView, programmers need to return UTF-8 string in this case.

Removing (renaming) "not recommended for Unicode applications" methods

In older versions of TRichView, there were a group of methods marked as "not recommended for Unicode applications". In TRichView version 11, these all these methods were renamed, R is added to the end of names of these methods. For example, "GetItemText" was renamed to GetItemTextR. These methods have TRVRawByteString[685] parameters. These methods are not described in the help file any more (you will not see "not recommended for Unicode applications" remark in this help file any more). Instead of these methods, a new group of methods is introduced: methods working with String type (Unicode for Delphi/C++Builder 2009, ANSI for older versions).

For example, GetItemTextA always returns ANSI string, GetItemTextW always returns Unicode string, GetItemText returns ANSI/Unicode string depending on Delphi version. The methods without -A and -W postfixes can be used to create projects portable to all versions of Delphi.

The affected methods are:

- AddNLTag[203] (and its versions);
- GetWordAt[239];
- GetItemText[230];
- SetItemText[267];
- GetCurrentItemText[347];
- SetItemTextEd[385];
- SetCurrentItemText[380].

The same naming convention is applied to the methods that previously had two versions (ANSI and Unicode):

- CopyText[213];
- GetSelText[238];
- InsertStringTag[359];
- InsertText[360];
- PasteText[364];

- SearchText 256.

The following methods have only String (ANSI/Unicode string depending on Delphi version) version:

- AddFmt 198;
- AddItem 201;
- InsertItem 353;
- GetTextInfo 239;
- GetCurrentTextInfo 350.

The following method is removed (only -R version): AddTextBlockNL.

These changes break some existing low level code, especially the changes in GetItemText and SetItemText. Specifically, in the following examples GetItemText/SetItemText must be changed to GetItemTextR/SetItemTextR:

- conversion to/from Unicode (ConvertToUnicode) 99;
- removing text formatting 722.

Delphi and C++Builder 2009

In the new versions of Delphi/C++Builder, String type is Unicode by default.

Many properties and parameters in TRichView become Unicode in Delphi/C++Builder 2009:

- *tag* 68 strings 68 (in tags-are-PChars mode);
- extra item properties 676;
- names of *checkpoints* 65;
- visible text in labels 133, numbered sequences 134, endnotes 135, footnotes 136;
- text in list markers 131 (not only bullets, but all numbering);
- live spelling 101 interface;
- hypertext targets (OnReadHyperlink 284, OnWriteHyperlink 284);
- and others.

Default (initial) values of the following properties are changed in Delphi/C++Builder 2009:

- Unicode 480 property of text style (from False to True);
- TRichView.RTFReadProperties 187.UnicodeMode 310 (from *rvruNoUnicode* to *rvruOnlyUnicode*);
- TRichView.Options 182 (*rvoAutoCopyUnicodeText* is included, *rvoAutoCopyText* is excluded).

When saving text styles (in RVF files or Delphi forms) in older versions of Delphi/C++Builder, only non-default value (True) of Unicode 480 property of text style is saved. When saving text styles (in RVF files 93 or Delphi forms) in Delphi/C++Builder 2009, value of Unicode 480 property is always saved, default or not. The main consequence is the following: when loading forms/RVF files with styles saved by older versions of Delphi/C++Builder in Delphi/C++Builder 2009+, Unicode property of all text styles become True. For RVF files, all text in text items is converted to Unicode automatically.

Events parameters of TRVRawByteString type (OnDrawStyleText 448, OnReadHyperlink 284, OnItemAction 280, OnSaveItemToFile 297, OnRVDblClick 287, OnRVRightClick 291, OnItemTextEdit 394) must be converted to String before using. For example, for OnRVDblClick 287 event:

```
// Example for Delphi 2009 or newer!!!
procedure TForm1.RichView1RVDblClick(Sender: TCustomRichView 162;
  ClickedWord: TRVRawByteString 685; Style: Integer);
var s: String;
begin
  if (Style>=0) and Sender.Style 193.TextStyles 437[Style].Unicode 480 then
     s := RVU_RawUnicodeToWideString 664(ClickedWord) // converting from "raw Unicode" to U
  else
     s := RVU_RawUnicodeToWideString 664(  // converting from ANSI to UnicodeString
      RVU_AnsiToUnicode 664(Sender.RVData 187.GetStyleCodePage(Style), ClickedWord));
  Panel1.Caption := s;
end;
```

New methods, properties and events

New TCustomRichView.OnAddStyle[274] event.

New NoVScroll[546] property of TRichView.

New options for line spacing[486] ("at least" and "exactly" options).

New in saving and loading

RTF:

New option for saving formatted header and footer in RTF: include *rvrtfSaveHeaderFooter* in RTFOptions[185].

New properties for RTFReadProperties[187]: CharsetForUnicode[305] and UseCharsetForUnicode[310].

RVF[93]:

Applications compiled in Delphi/C++Builder 2009 save RVF version 1.3.1, applications compiled in the older versions of Delphi saves RVF version 1.3 (previous version of RVF). RVF version 1.3.1 stores many strings (checkpoints names, DocProperties, tags, and others) in UTF-8 encoding.

Other

Special support for non-breaking hyphens in Unicode text, better table layout algorithm, copying multicell selection as text (in addition to RVF), making multicell selection with keyboard.

1.1.9 New in version 10

Compatibility issues

The following changes may affect existing applications.

HTML

- HTML tag names, color constants, CSS names are saved in lower case;
- SaveHTMLEx[251] does not use the following attributes for cells and tables any more: bordercolor, bordercolorlight, bordercolordark, background, bgcolor. CSS style is used instead. These attributes are not standard.
- OnSaveItemToFile[297] is called with Unicode parameter = True when saving UTF-8 HTML files.

Tables

- Table layout algorithm: when there is not enough space for columns having widths specified in pixels, they are decreased proportionally (previously, only the rightmost columns were decreased)
- By default, table grid is not shown in read-only editors any more (it also affects TBDRichViewEdit when its dataset is not in the edit mode).
- In editor, Tab key in the last table cell adds a new row; set the global variable RichViewTableAutoAddRow[705] to *False* to disable.

Editing

- BeginUpdate[336] and EndUpdate[341] have new functionality.
- Simple version GetWordAt[239] always returns ANSI string.
- Paste[361] can paste graphic files in CF_HDROP format.

RTF

- bookmarks are read from RTF as checkpoints[65].
- Local links are converted automatically when reading and writing RTF.
- Sub/superscripts and VShift[477] are saved and loaded to RTF differently.

Other

- meaning of ItemNo parameter in OnControlAction[275] is changed.
- RVF insertion methods ignore background.

Main new features

- Animation of pictures 90;
- "Smart popups" 402;
- page breaks inside table cells.

New item types 64

- labels 133;
- sequences 134;
- footnotes 136;
- endnotes 135;
- references to parent notes 140.

New in saving and loading

HTML:

New options in Options parameter of SaveHTML 250 and SaveHTMLEx 251:

- rvsoXHTML 686 – if included, XHTML is saved instead of HTML;
- rvsoUTF8 686 – if included, HTML/XHTML is saved in UTF-8 encoding.

You can choose between using <p> and 698.

Font families are stored in CSS for the most frequently used fonts. For example, if font name is 'Arial', the string *'Arial', 'Helvetica', sans-serif* will be saved.

RTF:

You can specify class representing PNG images using the function RegisterPngGraphic. After that, TRichView will be able to write and read PNG images in RTF.

You can save page parameters using DocParameters 176 property.

Options 185 for saving smaller RTF files: *rvrtfSavePicturesBinary*, *rvrtfPNGInsteadOfBitmap*.

RTF bookmarks are read as *checkpoints* 65. When exporting, links started from '#' are converted to links to checkpoints. When importing, links to checkpoints are converted to links started from '#'.

New properties for TCustomRichView.RTFReadProperties 187:

- StoreImagesFileNames allows storing file names of external images (update: this property is removed in version 16);
- BasePathLinks 305 specifies whether to add path to file in hyperlinks;
- IgnoreBookmarks 306 allows ignoring bookmarks;
- IgnoreNotes 307 allows ignoring footnotes and endnotes;
- IgnoreSequences 307 allows ignoring numbered sequences;
- IgnoreTables 307 allows ignoring tables;
- ReadDocParameters 308 allows reading DocParameters;
- UseSingleCellPadding 310 sets mode for loading tables cell padding.

Text:

- You can allow saving explicit page breaks in text files/streams. Set RichViewSavePageBreaksInText 698 typed constant to *True*.

Other:

- LoadFromStream 243 loads document in RVF, RTF or text formats (autodetect).

New in DB controls

- TDBRichView 407 and TDBRichViewEdit 410 can be placed in TDBCtrlGrid.
- TDBRichView 407 and TDBRichViewEdit 410 can update standard DB actions.
- TDBRichView 407.OnLoadCustomFormat 410, TDBRichViewEdit 410.OnLoadCustomFormat 415 and OnSaveCustomFormat 416 events allow loading and saving data to fields in custom formats.
- Special support for Unicode memo fields.

New in drag&drop 89

Low level events:

- OnOleDragEnter 395;
- OnOleDragOver 396;
- OnOleDragLeave 395;
- OnOleDrop 397.

New in Unicode processing

- Support for the word joiner Unicode character (code $2060). This character is invisible and it disallows line breaking before and after it. A special support is required because almost no font has a glyph for this character (it works properly only if inserted in the middle of text item).
- WM_UNICHAR message is processed, if the current text is Unicode.
- TCustomRichView.SearchTextW 256 and TCustomRichViewEdit.SearchTextW 369 allows searching for WideString.

New in layout and formatting

- If you set RichViewShowGhostSpaces 704 typed constant to *True*, spaces hidden between lines will be shown if *rvoShowSpecialCharacters* is in TCustomRichView.Options 182.
- You can define which characters to show in ShowSpecialCharacters 182 mode. They are listed in RVVisibleSpecialCharacters 714 typed constant.
- New possible values for TCustomRichView.BackgroundStyle 170.
- New property TCustomRichView.MaxLength 181 – maximal count of characters per line.
- New property TCustomRichView.WordWrap 196 allows to disable word wrapping.
- If you set RichViewWrapAnywhere typed constant to *True*, text can be wrapped in any place (this constant was removed in version 13 39).

New text attributes

- New text Protection 473 options: *rvprStyleSplitProtect*, *rvprSticking2*, *rvprSticking3*.
- New property HoverEffects 471 allows hyperlinks underlining.
- New properties UnderlineType 476, UnderlineColor 476 and HoverUnderlineColor 471 allow to customize underlines.
- New property SubSuperScriptType 476 allows implementing subscripts and superscripts.

New in printing

- TRVReportHelper 540 and TRVPrint 513 do not require parent control any more. They can be placed on data module or created without owner.
- TRVPrintPreview 417 displays full paper size, including areas where printer cannot print (see WinAPI function GetDeviceCaps, PHYSICAL*** constants).
- TRVPrintPreview.PrintableAreaPen 560 property. If you set its Style<>*psClear*, RVPrintPreview shows borders of printable area (this rectangle was displayed as a full page in previous versions)
- TRVPrint.FixMarginsMode 518 property.

New in item properties

- *rvepVisible* item property 673 allows to hide controls 126;
- *rvepResizable* item property 673 works for pictures 122 too;
- *rvespAlt* item property 676 can be applied to *bullets* 128 and *hotspots* 129 as well as to pictures 122 and *hot-pictures* 122;
- new values for VAlign 695 property of items: *rvvaAbsTop*, *rvvaAbsBottom*, *rvvaAbsMiddle*.
- new styles 669 for *breaks* 125 (horizontal lines): dashed and dotted; all types of *breaks* are drawn without rounded corners.

New in *checkpoints* 65

New set of editing methods working with checkpoint exactly at the position of caret:

- InsertCheckpoint 351;
- GetCheckpointAtCaret 341;
- RemoveCheckpointAtCaret 366.

New in tables 130

- Table cells can be printed on several pages (previously, page breaks were possible only between rows).
- New option for table.Options 577: *rvtoNoCellSelect*. If included, multicell selection by mouse is not allowed.
- New property table.VisibleBorders 581 allows to hide some sides of table border. Direct assignment to this property cannot be undone/redone, use table.SetTableVisibleBorders 600 for that.
- Table grid is not shown in read-only editors any more. More exactly, it is shown using global variable RichViewTableGridStyle2 (default value: *psClear*). The same pen style is used for drawing table grid in viewer.
- Copying multicell selection as a table fragment (in RVF 93 format); special pasting is not implemented yet, pasted as a new table.
- Adding new table row when user presses Tab in the last cell (set the global variable RichViewTableAutoAddRow 705 to *False* to disable this feature).
- New properties RowCount 580, ColCount 575.
- New properties CellHPadding 573, CellVPadding 574.
- New property Cell 605.Hint 609.
- New property Row 615.PageBreakBefore 616.
- New methods SetCellBackgroundImageFileName 596 and SetCellHint 598.
- Improved table layout algorithm.
- better SplitSelectedCellsVertically 601 and SplitSelectedCellsHorizontally 600 implementations.

New in RVF 93

If image was not returned in OnRVFPictureNeeded 289 event, this event is called the second time with the value of *rvespImageFileName* 676 property in the Name parameter (if value of this property was stored for this item). (update: in TRichView 14, OnRVFPictureNeeded is reworked and it is not called twice any more)

Custom Caret

You can draw your own caret instead of the system one. It is activated CustomCaretInterval 324, measured in OnMeasureCustomCaret 394 event, drawn in OnDrawCustomCaret 391 event.

Other

- The following Windows messages are processed by TCustomRichView: EM_GETSEL, EM_SETSEL, EM_GETTEXTRANGE. If you do not need these messages, you can define RVDONOTUSELINEARPOSITIONS in RV_Defs.inc. It slightly reduces exe-file size.
- Pasting graphic files in CF_HDROP format, see PasteGraphicFile 362.
- New characters are included in the default value of TCustomRichView.Delimiters 174.
- meaning of TCustomRichViewEdit's BeginUpdate 336 and EndUpdate 341 was changed.
- New option for TCustomRichView.SearchText 256: *rvsroFromStart*.
- Changes in TCustomRichView.GetWordAt 239.
- Ability to delete unused styles in multiple documents, see TRVDeleteUnusedStylesData 653.
- Much faster reformatting, especially if *rvoFastFormatting* is included in TCustomRichView.Options 182 (default). Much faster loading RTF files. Much faster copying to the Clipboard.
- Function RVIsCustomURL 710.
- Ctrl+↑ and Ctrl+↓ scroll the document down/up instead of moving caret.
- (in Unicode mode in WinNT/2000/XP/Vista) special support for typing in Kazakh, Tatar, Mongolian, Azerbaijani (Cyrillic & Latin), Uzbek, Kyrgyz, Serbian.
- New "hidden" parameters in TCustomRichView.OnControlAction 275.

1.1.10 New in version 1.9

Compatibility issues

- TRVStyle.CurrentItemColor[421] is used for drawing frame around the current item instead of TRVStyle.HoverColor[426].
- ApplyTextStyle[334], ApplyParaStyle[331], ApplyStyleConversion[333], ApplyParaStyleConversion[332] are applied to cells of all selected tables.
- OnSaveImage2[296] event: ImageSaveNo is declared as a var-parameter (it does not make sense otherwise)
- RTF export: only Standard[464] styles can be exported as style sheet.
- Default value of RVStyle.SpacesInTab[435] is changed from 8 to 0.
- Default value of TRichViewEdit.EditorOptions[325] is changed from [rvoClearTagOnStyleApp] to [rvoWantTabs, rvoCtrlJumps].
- New characters are added to default value of Delimiters[174] property.

New: tab stops

- Tabs[489] – properties of paragraph style defining tab stops.
- TRVStyle.DefTabWidth[422] defines default distance between tab stops.
- TRVStyle.SpacesInTab[435]: if equals to 0, methods for adding/inserting multiline text, RTF, and Tab key inserts tabulator[121] (item of special type). If this property has positive value, all these methods work like before (inserting several spaces in place of tabs).
- TRichView.AddTab[205], TRichViewEdit.InsertTab[359] adds/inserts tabulator[121]

New: Live spelling check

See overview topic[101]

New in TRichView

- VAlign[195] property, ClientToDocument[211] method;
- *rvoShowSpecialCharacters* in Options[182];
- BeginOleDrag[209] – allows drag&drop of inserted controls;
- Reformat[248] method – formatting method that keeps selection;
- AddTextNLA[206] method;
- LoadTextFromStream, LoadTextFromStreamW[247];
- Events: OnItemHint[281], OnProgress[283].

New item properties

- additional string properties[676]: hint, alt text, image file name; a set of methods supporting them: TRichView's GetItemExtraStrProperty[228] and SetItemExtraStrProperty[266], TRichViewEdit's GetCurrentItemExtraStrProperty[346], SetCurrentItemExtraStrProperty[379] and SetItemExtraStrPropertyEd[384] (this set of methods is similar to methods working with additional integer properties[673]);
- *rveoNoHTMLImagesSize* additional integer property[673]

Printing

Added: *rvpaoKeepLinesTogether* and *rvpaoKeepWithNext* Options[487] for paragraph style.

Tables

- *rvtoIgnoreContentWidth* and *rvtoIgnoreContentHeight* options for TRVTableItemInfo[565].Options[577];
- AssignProperties[584] method;
- better table layout algorithm.

RTF

- TRichView.RTFReadProperties[304].ExtractMetafileBitmaps[306] – allows extracting bitmaps wrapped in metafiles.
- output RTF file size is greatly reduced;
- a number of small improvements in RTF import and export.

HTML export

- SaveHTMLEx[251] is modified to conform "HTML 4.01 Transitional" standard, see http://validator.w3.org/;
- improvements in CSS export;
- SaveHTMLEx[251] saves characters of "Symbol" font as HTML entities (like α);
- better saving of RTF-like tables (with CellHSpacing[573]=CellVSpacing[574]=-1 to simulate 1-pixel width border) with SaveHTMLEx[251], or SaveHTML[250] with *rvsoForceNonTextCSS* in the Options parameter;
- ImagesPrefix parameter of SaveHTML[250]/SaveHTMLEx[251] can contain a full path (determined by the presence of ':').

TRVOfficeConverter[532]

- added: PreviewMode[535] property.

TRVReportHelper[540]

- added: OnDrawCheckpoint[556] event

Database controls

- OnLoadDocument[415] event for TDBRichViewEdit.
- OnNewDocument[410] and OnLoadDocument[410] for TDBRichView.

Other improvements

- TRVStyle.CurrentItemColor[421].
- *rvoNoCaretHighlightJumps* in EditorOptions[325];
- faster redrawing on typing;
- better processing of bidirected text[100];
- ApplyTextStyle[334], ApplyParaStyle[331], ApplyStyleConversion[333], ApplyParaStyleConversion[332] are applied to cells of all selected tables;
- all TRichViewEdit methods for inserting one item[86] return boolean value: *True* if the item was successfully inserted (insertion can fail because of protection). You can use SetCurrent*** methods to set additional properties of this item after the insertion.
- support for soft (optional) hyphens (WideChar($AD)) and zero-width spaces (WideChar($200B)) in Unicode text. Normally soft hyphen is invisible. It is displayed as a hyphen only if line breaks is after it.If *rvoShowSpecialCharacters* is in Options.[182] it is shown as "not" sign (WideChar($AC)), like in MS Word.

Unlike spaces and paragraph marks, soft hyphens change their widths when switching normal/show-special-chars modes. So call Reformat[248] when switching these modes. Processing of these characters may slow down formatting of Unicode text. If you do not need them, activate RVDONOTUSESOFTHYPHENS in RV_Defs.inc Zero-width spaces are processed correctly even if this character is not present in the font.

- RVF (RichView Format) forward compatibility improved.

1.1.11 New in version 1.8

Compatibility issues

Changes that can affect existing code:

- OnHTMLSaveImage[277] is called for tables to save background image;
- OnHTMLSaveImage[277] is called to save cell background image; in this case, parameter ItemNo = -1; you can distinguish saving of document background and cell background by RVData parameter.
- MaxTextWidth[181] property does not include left and right margins any more, so now it works according to the help file[80];
- parameters are changed in OnChanging[390] event (should not be a problem because this event is rarely used);
- table.OnDrawBorder[603] event has two new parameters: Row and Col;
- changes in *rvteoHTMLCode* and *rvteoRTFCode* (Options[472] of text style);
- TRichView does not use font external leading any more (it may add extra spacing between lines), so its line spacing just like in TRichViewEdit now.
- default selection mode: select by words[432];
- RVF version number is changed from 1.2 to 1.3 (see RVF specification[109]); new version of RVF is not completely compatible with the old version (new version of TRichView can read old RVF; old versions of TRichView may not read new RVF);
- *rvprSticking*[473] protection option meaning was changed (split into 3 rvprStick*** options).

Drag&Drop

See overview[89].

Background images for tables and cells

- table.BackgroundImage[567], BackgroundStyle[568];
- cell.BackgroundImage[605], BackgroundStyle[606].

Resizing pictures and controls with mouse

By default, all pictures[122] and hot-pictures[125] can be resized with the mouse (except for TIcon-s).

By default, controls[126] cannot be resized. But you can explicitly allow resizing for the given control, if you set its *rvepResizable*[673] property to non-zero value. See also SelectControl[257] method.

You can disallow resizing in the given RichViewEdit by including *rvoNoImageResize* in its EditorOptions[325] property.

Extra items' integer properties[673]

- *rvepSpacing* – spacing around the item (image or control);
- *rvepResizable* allows resizing the given control with the mouse;
- *rvepDeleteProtect* protects the non-text item from deletion in editing operations.

Printing and preview

- ColorMode 549 property for TRVPrint 513 and TRVReportHelper 540: allows black and white or grayscale printing/previewing for color documents;
- much more accurate print preview 417;
- new properties of TRVPrintPreview: ClickMode 558, Color 558, PageBorderColor 559, PageBorderWidth 559, ShadowColor 560, ShadowWidth 560.

Selecting:

- new line selection mode, see TRVStyle.LineSelectCursor 427
- defining visual appearance of selection, see TRVStyle.SelectionStyle 432
- defining selection mode, see TRVStyle.SelectionMode 432

RVF 93

- tags can contain space characters;
- in text-mode RVF, TRichView saves text RVF files even if the document contains Unicode text (text files can be edited in text editors)
- ability to save arbitrary additional text strings (DocProperties 176).

HTML export

- by default, all images are saved in HTMLs as Jpegs. This is obviously undesirable if images already have formats recognizable by web browsers (such as Gif or Png). Now you can specify graphic formats that must not be converted to Jpeg on HTML export, using RV_RegisterHTMLGraphicFormat procedure.
- OnSaveHTMLExtra 295 event now allows to insert HTML code before the </body> tag;
- new OnSaveParaToHTML 298, OnSaveImage2 296, OnSaveItemToFile 297 events;
- new options 686 for HTML export;
- OnWriteHyperlink 302 event supersedes OnURLNeeded.

RTF export

- OnWriteHyperlink 302 event supersedes OnURLNeeded;
- new OnSaveItemToFile 297 event.

RTF import

- import of DDB (device dependent bitmaps);
- using some advanced RTF features when importing images from RTF saved by MS Word 2000+.
- more special RTF characters are recognized
- new event: OnImportPicture 279

DB controls

- new event: TDBRichViewEdit.OnNewDocument 416;
- new property: TDBRichViewEdit.IgnoreEscape 414.

Other improvements

- TRVReportHelper.OnDrawHyperlink 556 event;

- default text styles for paragraph styles (TParaInfo.DefStyleNo[491]);
- improvements in processing of bidirected text[100];
- new action in OnItemAction[280] event: *rviaInserted*;
- faster redrawing on editing;
- previously reserved break styles[669] are implemented;
- RefreshListMarkers[248] method.

1.1.12 New in version 1.7

Compatibility issues

- TCustomRichView.DeleteLines is renamed to DeleteItems[214].
- If you assign non-default values to extra item properties[673], saved RVF files will not be opened with applications that use older version of RichView.
- OnDrawPageBreak[446] event has additional parameter. If you already use this event, you must change declaration of event handler manually.
- All references to TImageList were replaced with TCustomImageList. Especially important: parameter of OnRVFImageListNeeded[288] was changed; if you already use this event, you must change declaration of event handler manually;
- FirstIndent[485] is not applied to line starts inside paragraphs (Shift+Enter).
- DeleteUnusedStyles[216] has a new parameter (deleting list styles).
- GetWordAt[239] parameters are changed.
- Accessing invalid item in RVStyle.TextStyles and ParaStyles does not cause "List index is out of bounds" error any more.
- OnHTMLSaveImage[277] is called for background bitmap[169]; this may affect old code that assumes ItemNo is always an index of item.
- Declarations of some types are moved in other files.
- Format of CSS was slightly changed to support list styles.

The main new feature: paragraph lists (bullets and numbering)

- TRVStyle.ListStyles[429]

Methods of TCustomRichView:
- SetListMarkerInfo[269],
- GetListMarkerInfo[233].

Methods of TCustomRichViewEdit:
- ApplyListStyle[331],
- RemoveLists[367],
- ChangeListLevels[338].

New options[686] for HTML export – *rvsoMarkersAsText*.

New parameter in DeleteUnusedStyles[216] (deleting list styles)

Extra integer properties for items (stretching, transparent color for bitmaps, etc)

Methods of TCustomRichView:
- SetItemExtraIntProperty[266] – changes value of property (without document reformatting);

- GetItemExtraIntProperty 227 – returns value of property.

Methods of TCustomRichViewEdit:

- SetItemExtraIntPropertyEd 383 – changes value of property, updates the document;
- SetCurrentItemExtraIntProperty 378 – the same for item at the position of caret;
- GetCurrentItemExtraIntProperty 346 – returns value of property of item at the position of caret.

Soft page breaks (preliminary implementation)

Methods of TCustomRichView:

- AssignSoftPageBreaks 209, ClearSoftPageBreaks 211 - displays/hides soft page breaks

Property of TRVStyle

- SoftPageBreakColor 434

Other improvements in TCustomRichView

New methods:

- DeleteParas 215 – deletes the specified paragraphs, updates the document;
- GetLineNo 233 – returns line for the specified position;
- GetWordAt 239 (parameters are changed);
- GetItemAt 225 – returns item in the given coordinates.

New events:

- OnItemAction 280 – occurs on operations with item;
- OnPaint 283.

New properties:

- RTFReadProperties.ConvertHighlight 305, LineBreaksAsParagraphs 307.

New Options 686 for HTML export: *rvsoUseCheckpointsNames.*

New Options 188 for RVF: *rvfoSaveLayout*, *rvfoLoadLayout*.

Events:

- OnSaveHTMLExtra 295, OnSaveRTFExtra 299 – allow saving additional information in HTML and RTF.

Other improvements in TCustomRichViewEdit

Methods:

- ApplyParaStyleConversion 332 (and OnParaStyleConversion 397 event) – implements commands like "change paragraph alignment", "increase indent", etc.
- InsertText, InsertTextW 360 have additional optional parameter;
- GetCurrentLineCol 348 – returns line and column at the caret position;
- ConvertToPicture 339, ConvertToHotPicture 339 – convert picture to *hot-picture* and vice versa

Properties:

- Modified 327: Boolean

New EditorOption 325 – *rvoDoNotWantShiftReturns*.

Other improvements in TRVStyle 418

- InvalidPicture 427 – picture that used instead of invalid or missed pictures;
- InvalidItem 451 for TextStyles, InvalidItem 456 for ParaStyles – are returned when accessing item which is out of index range.

New properties of text style (TFontInfo 478):

- Options 472 – affects HTML and RTF export

New properties of paragraph style (TParaInfo 482):

- new protection Option 487 – *rvpaoDoNotWantReturns*.

Improvements in TDBRichViewEdit 410

- AutoDeleteUnusedStyles 412

Improvements in TRVOfficeConverter 532

- ExcludeHTMLImportConverter, ExcludeHTMLExportConverter 534;
- ExtensionsInFilter 535;
- ErrorCode 533 – returns error code for the last import/export operation.

Improvements in TRVPrint 513

- OnPrintComponent 554 is optional since this version;
- ClipMargins 518 property – disallows printing on margins;
- headers and footers: TRVPrint.SetHeader 529, SetFooter 528, HeaderYMM, FooterYMM.
- reading headers and footers from RTF: TRichView.RTFReadProperties.SetHeader, SetFooter, HeaderYMM, FooterYMM (update: these properties and methods are deprecated now; use TRichView.SetHeader 264, SetFooter 263, DocParameters 176).
- OnPagePostpaint 530;
- ColorMode 549 – allows black&white and grayscale printing and preview;

Improvements in all visual controls of the package

- UseXPThemes 546 - allows using WindowsXP themes (visual styles)

Improvements in tables 145

Global variable RichViewTableGridStyle: TPenStyle – drawing invisible table borders.

Methods:

- SetCellVisibleBorders 600 – sets VisibleBorders property for the cell.

Properties:

- HeadingRowCount 576 – defines table header;
- new Option 577 – *rvtoRTFAllowAutofit*.

Events:

- OnCellEditing 602 – allows to prevent cell editing;
- OnDrawBorder 603 – custom drawing for cell borders.

New help topics

- Valid Documents 105
- Custom Drawing 108
- Examples for DBRichViewEdit 410
- New overview: Item Types 118
- More examples and cross references.

Miscellaneous

Better RTF import (including import of inserted symbols, Unicode paired quotes and dashes), better IME support, better Unicode text processing, faster formatting, and so on.

Packages for D7 and CB6.

1.1.13 New in version 1.6

Compatibility issues

- TRVPrint.OnPrintComponent[554] event has changed type of Sender parameter (from TRVPrint to TCustomRVPrint). If you use this event in your existing projects, change type of this parameter in your handlers for this event.
- Parameters added in TRichView.DeleteUnusedStyles[216].

Changes in TRVPrint

- new: TRVPrint.TransparentBackground[553]
- new: TRVPrint.GetFirstItemOnPage[553]
- new: TRVPrint.MirrorMargins[521]

Changes in tables

- Vertical alignment of cells (cells' VAlign[613] and rows' VAlign[616], SetCellVAlign[599] and SetRowVAlign[600] of table).
- BorderColor[607] and BorderLightColor[608] of cells allow to override default values (see also SetCellBorderColor[597] and SetCellBorderLightColor[598]).
- Ability to hide table grid in editor (new option for table Options[577] – *rvtoHideGridLines*).
- Ability to print table background in white (new option for table PrintOptions[579] – *rvtoWhiteBackground*).
- New table Options[577] – *rvtoCellBelowBorders* and *rvtoOverlappingCorners*.
- Tables with *rvtoEditing* excluded from Options now ignore commands for cells clearing and style applying.
- Hidden sides of borders can be shown as dotted gray line (if *rvtoHideGridLines* is not in table Options[577]). Previous versions show grid only for zero-width borders.

New components

 TRVOfficeConverter[532] – allows using Microsoft Office converters

 TRVReportHelper[540] – reserved for future use.

HTML export

New options[686] – *rvsoImageSizes*, *rvsoForceNonTextCSS*.

RTF export

You can save hyperlinks in RTF, using OnURLNeeded event (update: this event is removed in v17; use OnWriteHyperlink[302]).

New option[185] for saving RTF files – *rvrtfSaveEMFDefault*.

RTF import

- new: reading hyperlinks: OnReadHyperlink[284] event;
- reading tables (see also: RTFReadProperties.AutoHideTableGridLines[304]);
- reading external images (see important note for using third-party graphics formats[721]);
- new: ignoring hidden text (see RTFReadProperties.SkipHiddenText[309]).

Horizontal scrolling

new: HScrollPos[545], HScrollMax[545], OnHScrolled[548].

Visual effects and formatting

- new: TRVStyle.InactiveSelColor and InactiveSelTextColor[426];
- new: CharSpacing[468] property of text style.

DB Components

DB components can display RTF and plain text fields.

new: TDBRichViewEdit.FieldFormat[413] allows to edit RTF and text fields

Miscellaneous

- New component-editor (click RichView and choose "Settings..." in popup menu) allows to set properties related to updating styles.
- New methods of TRichViewEdit: InsertStringTag and InsertStringWTag[359].
- Parameters are added in TRichView.DeleteUnusedStyles[216].
- Essential styles can be marked with Standard[464] property.
- TRichView.SearchText[256] and TRichViewEdit.SearchText[369] now searches in table cells.
- Hypertext works automatically in read-only[328] editors (user does not need to press and hold Ctrl).
- Editor can hide caret in read-only mode (new option in EditorOptions[325] - *rvoHideReadOnlyCaret*).
- Methods for loading text files understand page-break characters (characters with code $0C).
- Methods for loading text files can be used for Unix text files (with #$0A characters separating lines).
- TRichViewEdit.SelectCurrentLine[369].

1.1.14 New in version 1.5

Compatibility issues

1. Changed types in events. Important (please read before installing the update!):

Types of parameters were changed in some events: "TRichView" was changed to "TCustomRichView", "TRichViewEdit" to "TCustomRichViewEdit".

After installing, open all forms containing RichView or RichViewEdit, and modify declarations of event handlers: change "TRichView" to "TCustomRichView", "TRichViewEdit" to "TCustomRichViewEdit". You must do it manually!

(This is one-time change that is required only for projects created with older version of the components)

For example, for TRichView.OnRVMouseDown

```
procedure TForm1.RichView1RVMouseDown(
                Sender: TRichView;
                Button: TMouseButton;
                Shift: TShiftState;
                ItemNo, X, Y: Integer);
begin
...
end;
```

must be changed to

```
procedure TForm1.RichView1RVMouseDown(
                Sender: TCustomRichView;
                Button: TMouseButton;
                Shift: TShiftState;
                ItemNo, X, Y: Integer);
begin
...
end;
```

Sorry for inconvenience.

2. Changed component hierarchy

TRichViewEdit, TDBRichView, TDBRichViewEdit are not descendants of TRichView any more. All of them (and TRichView itself) are descendants of TCustomRichView.

TDBRichViewEdit is not a descendant of TRichViewEdit any more. Both them are descendants of TCustomRichViewEdit.

See new scheme of hierarchy 27.

3. New parameter

New parameter of SaveCSS 441/SaveCSSToStream 442. Please use empty set ([]) for it.

New feature: RTF Import

- TRichView.LoadRTF 243;
- TRichView.LoadRTFFromStream 244;
- TRichViewEdit.InsertRTFFromFileEd 356;
- TRichViewEdit.InsertRTFFromStreamEd 357;
- TRichViewEdit.PasteRTF 363;
- TRichViewEdit.CanPasteRTF 337;
- and property TRichView.RTFReadProperties 187.

New feature: saving text and paragraph styles in RVF

- New *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions 188
- TRichView.RVFTextStylesReadMode 191;
- TRichView.RVFParaStylesReadMode 190;
- Related: TRichView.DeleteUnusedStyles 216.

New feature: *hot pictures*

Hot pictures are the same as normal pictures, but they are also hypertext jumps.

New methods: AddHotPicture 199, AddHotPictureTag 199, InsertHotPicture 352.

Old methods (GetPictureInfo 236, SetPictureInfo 269, GetCurrentPictureInfo 349, SetCurrentPictureInfo 381, SetPictureInfoEd 386) now work with both pictures and *hot pictures*.

New properties for text and paragraph formatting:

- TFontInfo.StyleEx 475 (new option – *rvfsAllCaps*);
- TFontInfo.CharScale 467;

- TParaInfo.LineSpacing[486];
- TParaInfo.LineSpacingType[486];
- TParaInfo.Options[487].

New feature: Tab navigation through links and inserted controls

TRichView.TabNavigation[194] property;

TRichView.GetFocusedItem[223] method.

New feature: Proportional resizing of controls in documents

RichView now can resize controls depending on document width.

Let ItemNo - index of control item (rvsComponent).

```
var Item: TRVControlItemInfo;
  ...
Item := RichView.GetItem(ItemNo) as TRVControlItemInfo;
Item.PercentWidth := 50; // 50% of document width (minus paragraph indents)
```

Effect will be after reformatting[217].

Known problem: such controls may not work properly in paragraphs with word-wrapping turned off.

Miscellaneous

- OnCaretMove[390]
- GetLastCheckpoint[232], GetPrevCheckpoint[237]
- two new possible values of Direction parameter of OnCaretGetOut[389]
- TRVStyle.ParaStyles.FindStyleWithAlignment[457] and FindSuchStyle[458]
- "whole word" option for TRichView.SearchText[256] and TRichViewEdit.SearchText[369] (thanks to Louis Kessler)

Improved:

- support of third party graphic classes[720]
- RTF export
- HTML export: SaveHTML[250] now retains right and justify alignments.

1.1.15 New in version 1.4

Compatibility issues

- TRichView.OnControlAction[275] declaration was changed. If you use this event, please add "var" to ctrl parameter of your event handler after rebuilding the package. Failing to do it causes memory access errors when running your applications.
- GetCurrent***Info/SetCurrent***Info methods of TRichViewEdit now differ from Get***Info(ItemNo,...)/Set***InfoEd(ItemNo,...). "Current" methods can be used to work with items in cell-inplace editor in the same way as with items in main editor (programmer does not need to know if the item is in main editor or in sub-editor, he/she can always work with item at the position of caret). "Not-current" methods cannot be used to work with items in inplace editor, because such items cannot be addressed using only their indices.

The main new feature: tables

There is a new set of topics about them 144.

New methods

TRichView:

- AddNLATag 203, AddItem 201, GetItem 225, GetItemNo 229,
- GetJumpPointLocation 231
- GetItemCoords 226, GetItemClientCoords 226

TRichViewEdit:

- InsertItem 353,
- GetCurrentItem 345, GetCurrentItemEx 345
- SetItemTextEd 385,SetCurrentItemText 380, GetCurrentItemText 347,
- CanChange 336
- BeginItemModify 335, EndItemModify 341
- ResizeCurrentControl 368, AdjustControlPlacement2 330

TFontInfos:

- FindStyleWithCharset 453

New properties

- TParaInfo.NextParaNo 492
- TRVStyle.DefCodePage 421
- TRichView.HScrollVisible 545
- TRichView.RVData 187
- TRichView.InplaceEditor 546
- New option (rvoAutoSwitchLang) in TRichViewEdit.EditorOptions 325 (thanks to Pavel Zhelty)
- New states in TRVTextDrawStates 692

New events:

- TRichView.OnCopy 276
- TRichView.OnHTMLSaveImage 277
- TRichViewEdit.OnCaretGetOut 389
- New "hidden" parameters in TRVStyle.OnDrawStyleText 448 and OnDrawTextBack 449

Cursors

Two new cursors. Ability to use system default hypertext cursor 108

Miscellaneous

Special support for Unicode Hindi keyboard input (note: Hindi uses complex glyphs composed from several Unicode characters; such glyphs are not fully supported yet)

Known problems

- TRVStyle does not work properly in frame (Delphi 5+ and C++Builder 5+)
- wrong displaying of underline for Unicode text in Windows95 (WinNT/2000 is OK)

1.1.16 New in version 1.3

The main additions in version 1.3

- Undo/Redo 88 (many topics, this is the link to the overview);
- Unicode support 99 (many topics, this is the link to the overview);
- Events for custom drawing (several topics; this is the link to the list)
- TRichView.RTFOptions 185, CopyRTF 213, SaveRTF 253, SaveRTFToStream 254
- TRVPrint.OnPagePrepaint event
- TFontInfo.Protection 473 property
- TRVStyle.UseSound 437 property
- TRichView.AddTextBlockNL, SetItemText 267, GetItemText 230, SaveTextToStream, SaveTextToStreamW 255 methods;
- TRichView.OnControlAction 275 event;
- new options in TRichView.Options 182, TRichViewEdit.EditorOptions 325;
- new parameter in TRichView.LoadText 246;
- TFontInfo.IsEqual 478, TFontInfos.FindStyleWithFontStyle 455, FindStyleWithFontSize 454, FindStyleWithColor 453, FindStyleWithFontName 454, FindSuchStyle 455, FindStyleWithFont 454
- new global variables 699
- HTML export improved.

Compatibility issues

- declaration of some types were moved (usually from RichView.pas to either RVScroll.pas or RVStyle.pas; in some cases you will need to correct uses clause of units in older projects, if Delphi will not find some types).
- TRichView.LoadText has a new parameter.
- TFontInfo.SingleSymbols property removed; see TFontInfo.Protection 473.
- Older version of RichView will not be able to read RVF 93 with Unicode.

1.1.17 New in version 1.2

The main additions in version 1.2

- Borders and Background of Paragraphs 73;
- spacing between paragraphs (TParaInfo.SpaceBefore 489,TParaInfo.SpaceAfter 489) and ability to move items to the new line inside paragraphs (Shift+Enter in editor, also see TRichView.IsFromNewLine 242, TRichView.IsParaStart 242, TRichView.SetAddParagraphMode 259);
- ability to specify explicit page breaks (TRVStyle.PageBreakColor 431, TRichView.PageBreaksBeforeItems 184, TRichView.Options 182, TRichViewEdit.InsertPageBreak 355, TRichViewEdit.RemoveCurrentPageBreak 367);
- ability to change background color of text under mouse (TFontInfo.HoverBackColor 470);
- ability to save and load styles to registry (TRVStyle.LoadReg 441, TRVStyle.SaveReg 442, Delphi 4+ is required);
- ability to search upwards in RichView (TRichView.SearchText 256);
- ability to save background in RVF file or stream (optionally) (TRichView.RVFOptions 188, RVF Overview 93).

Compatibility issues

- default search direction in TRichView.SearchText 256 was changed;
- some keys for saving styles in ini-files were changed (SingleSymbolsN -> FontSingleSymbolsN, VShiftN -> FontVShiftN, NextStyleNoN -> FontNextStyleNoN, AlignmentN -> ParaAlignmentN);
- components do not use text HoverColor for drawing selected hypertext links any more; you can return old behavior turning on RVUSETEXTHOVERCOLORWITHSELECTED define;
- removed method TRVStyle.SaveCSSFile (replaced with TRVStyle.SaveCSSToStream 442);
- RVF format was changed. RichView v1.2 can read RVF files saved by old versions of RichView, but older versions can not read new RVF files.

New and updated topics (from version 1.1.3 to version 1.2)

New:

- TFontInfo.HoverBackColor 470
- TRVStyle.LoadReg 441
- TRVStyle.SaveReg 442

- TRVRect 657
- TRVBooleanRect 653
- TRVBorderStyle 669

- Borders and Background of Paragraphs 73

- TParaInfo.SpaceBefore 489
- TParaInfo.SpaceAfter 489
- TParaInfo.Border 485
- TParaInfo.Background 484

- TRVBorder 501
- TRVBorder Properties
- TRVBorder.BorderOffsets 502
- TRVBorder.Color 502
- TRVBorder.InternalWidth 502
- TRVBorder.Style 502
- TRVBorder.VisibleBorders 502
- TRVBorder.Width 503

- TRVBackgroundRect 500
- TRVBackgroundRect Properties
- TRVBackgroundRect.BorderOffsets 501
- TRVBackgroundRect.Color 501

- TRichView.IsFromNewLine 242
- TRichView.SetAddParagraphMode 259

- TRVStyle.PageBreakColor 431
- TRichView.PageBreaksBeforeItems 184
- TRichViewEdit.InsertPageBreak 355
- TRichViewEdit.RemoveCurrentPageBreak 367

- TRVStyle.SaveCSSToStream 442
- TRichView.SaveHTMLToStream 252
- TRichView.SaveHTMLToStreamEx 252

Updated (not including topics where only references were corrected):

- RichView Paragraphs 72, Building RichView Document 77
- Export of RichView contents to HTML 96, RichView Methods for Saving and Loading 92, Searching and Replacing 83,
- RVF Overview 93, RVF Specification 109
- TRVSearchOption 689
- TRichView.SearchText 256
- TRichViewEdit.SearchText 369
- TRichView.IsParaStart 242
- TRichView.Options 182
- TRichView.RVFOptions 188
- TRichView.SaveHTML 250
- TRichView.SaveHTMLEx 251
- TRVSaveOption 686
- TRichView.SavePicture 253

Removed:

- TRVStyle.SaveCSSFile

1.1.18 Compatibility with older versions

See compatibility issues for versions: 10 44, 1.9 48 1.8 50, 1.7 52 1.6 55 1.5 56, 1.4 58, 1.3 60, 1.2 60

For former users of freeware version

You can use this version of TRichView, TRVStyle, TRVPrint components in your projects started with older version.

But

- You lose all hypertext in text styles then you open old project with new TRVStyle; this is because there were only two hypertext styles in older versions (had indices *rvsJump1*, *rvsJump2*); now any text style can be hypertext, regardless of its index. In order to fix, set Jump 479=True, HoverColor 470 to TRVStyle.HoverColor 426 for hypertext items.
- There are some obsolete methods, but they still work. For example, all methods which in older versions added left-aligned items (AddFromNewLine), now add items with the 0th paragraph style; all methods which in older versions added center-aligned items (AddCenterLine) now add items with the 1st paragraph style. Initial values of paragraph styles 431: 0th paragraph is left aligned, 1th is centered by default.
- *Checkpoints* are not items any more, but some additional information associated with items. Add***Checkpoint methods still work: such method appends *checkpoint* that will be associated with the next appended item. Therefore, you now cannot add two or more *checkpoints* one by one without item between them. You can choose one of two options when compiling RichView source:
  - o RichView raises exception when you try to add *checkpoint* after *checkpoint* (default mode);
  - o RichView ignores the second *checkpoint*; Add***Checkpoint*** methods will return the index of previous added *checkpoint* in this case; this mode is turned on by defining RVALLOWCPBYCP conditional define.
- New version has no special design-time property editors for RVStyle.TextStyles
- For C++Builder users: some names of methods and properties are now written in another case. "CheckPoint" is "Checkpoint" now, "HotSpot" is "Hotspot" now.

1.2 Overview

Parts of Documents

- RichView Item Types Overview 64
- RichView Items' "Checkpoints" Overview 65
- RichView Items' "Tags" Overview 68
- RichView Hypertext Overview 69
- RichView Paragraphs 72
- Borders and Background of Paragraphs 73
- Styles and Style Templates 74
- Hidden Text 76

Basic Operations

- Building RichView Document 77
- Scrolling in RichView 80
- Selecting Part of RichView Document 81
- RichView Clipboard Functions Overview 82
- Searching and Replacing in RichView and RichViewEdit 83
- Obtaining Items of RichView 84
- Modifying RichView Items 85

Editing

- RichViewEdit: Inserting items in position of caret 86
- Undo/redo in RichViewEdit 88
- Drag&drop 89

Graphics

- Animated Images 90
- Smooth Image Scaling 91
- Semitransparent Objects 91

Saving and Loading

- RichView Methods for Saving and Loading 92
- RichView Format (RVF) Overview 93
- Export of RichView Contents to HTML 96

Internationalization

- Unicode in RichView 99
- Bi-directional text in RichView 100

Spelling Check

- Live spelling check 101

Important Technical Information

- Viewer vs Editor 102
- Controls, Documents, Items 104
- Valid Documents 105
- Units of Measurement 106
- Custom Drawing 108
- TRichView Cursors 108
- RichView Format (RVF) Specification 109

1.2.1 Parts of Documents

1.2.1.1 TRichView Item Types

Document in TRichView can contain items of the following types:

- **Text**[120] can have different fonts. Each text item is linked to one item in the collection of text attributes ("styles")[437]. A text style defines a font of the text item and some additional effects (such as background color, vertical offset (for subscripts/superscripts), overline), it has some properties affecting editing. A text style can define a hyperlink; for hyperlinks you can define additional properties such as mouse cursor and hover color. A text style also defines whether the text is in ANSI or Unicode[99] encoding.
- **Tabulators**[121] are used instead of TAB (#9) characters.
- **Pictures**[122]: you can insert any Delphi graphic type in RichView. You can use TBitmap, TIcon, TMetafile, TJpegImage or third-party graphic classes[720]. Gif images can be animated.
- **"Hot Pictures"**[125]: the same as pictures, but they are also hypertext links.
- **"Bullets"**[128] are pictures from ImageList. Bullets provide an easy and efficient way for adding many similar "standard" pictures in the document. You can use any number of ImageLists. These bullets have no relation to paragraph bullets and numbering; see "list markers" below.
- **"Hotspots"**[129] are bullets-"hypertext links". They can specify two pictures: normal and "hot" (for highlighting effect under the mouse pointer)
- **"Breaks"**[125] are horizontal lines.
- **Controls**[126]: You can use any Delphi/C++Builder control merged in text.
- **List Markers**[131] represent paragraph bullets & numbering.
- **Tables**[130] arrange subitems in rows and columns.
- **Labels**[133] are non-text items looking like a text.
- **Numbered sequences**[134] allow to insert numbered fields (like "Fig. 1", "Fig. 2", ...) in text.
- **Endnotes**[135].
- **Footnotes**[136].
- **Sidenotes**[137] are notes displayed in floating boxes.
- **Text boxes**[139]
- **References to parent note**[140]
- **"Page number" fields**[141]
- **"Page count" fields**[142]
- **Mathematical expressions**[143]

Non-text items have an associated string referred in this help as a "name of item". TRichView does not use these names itself, use them for your own purposes.

Items are organized in paragraphs. You can define your paragraph attributes ("styles")[431] with different alignment, indents, spacing, line height, background and other attributes.

All the information above can be saved and loaded to stream, file, database field or clipboard in a special format (RVF[93]). You can also export document as HTML, RTF, DocX file or print it (see export to html[96]).

Internally, all item types are implemented as classes. Standard item types are implemented in the unit RVItem.

A detailed description of item types is beyond the scope of this help file.

All item types are classes derived from one base class: TCustomRVItemInfo[563].

See also: detailed overview of item types 118

1.2.1.2 "Checkpoints"

Checkpoints are...

Each item can have an associated "checkpoint" – an invisible label that you can use for jumping to the specified part of document. Other word-processing tools call similar objects "bookmarks" or "anchors".

Checkpoint has:

- name;
- *tag* (just like any RichView items tags 68)
- *RaiseEvent* flag; if *RaiseEvent*=True then TRichView generates OnCheckpointVisible 274 event when this checkpoint appears in the field of view (usually as a result of scrolling); this feature only works for TRichView, and does not work for TRichViewEdit (in this version).

Limitations:

- checkpoint names and tags must not contain #0, #1 and #2 characters; failing to follow this rule will cause problems in saving documents in RichView Format 93;
- the following names are reserved: '_footnoteN', '_endnoteN', '_sidenoteN', where N is a number.

Drawing checkpoints

You can set the option to make checkpoints visible: include *rvoShowCheckpoints* in RichView.Options 182. By default, checkpoints are shown as dotted horizontal lines with a small circle in the top left corner of the associated item. Displaying checkpoints is useful in editing mode.

Checkpoints colors:

- RichView.Style.CheckpointColor 420 for "normal" checkpoints (*clGreen* by default)
- RichView.Style.CheckpointEvColor 420 for checkpoints with *RaiseEvent=True* (*clLime* by default)

You can create your own procedure for drawing checkpoints, see TRVStyle.OnDrawCheckpoint 444.

Using checkpoints

The usual using of checkpoint is obtaining its vertical coordinate (relative to the top of document area) and scrolling document so that the item associated with the checkpoint becomes visible. There are some more interesting applications for checkpoints.

Adding checkpoints in the document

You can add checkpoint at the end of the document using AddCheckpoint 202 method.

When you add a new item to the end of the document, the last added checkpoint becomes associated with this item. (So, TRichView can contain any number of checkpoints associated with items and only one checkpoint at the end of the document).

You can modify the checkpoint by SetCheckpointInfo 261 method, and remove it by RemoveCheckpoint 248.

The rest of methods can be separated into two groups:

1. Methods returning information about some checkpoint. These methods return a value of TCheckpointData class.
2. Methods extracting checkpoint properties from the TCheckpointData value.

Methods and events of the first group (obtaining TCheckpointData):

TCheckpointData is a pointer. If these methods return **nil**, it means that there is no such checkpoint, item has no associated checkpoint, etc.

- GetFirstCheckpoint 223, GetNextCheckpoint 234(CheckpointData), GetLastCheckpoint 232, GetPrevCheckpoint 237(CheckpointData) – allow you to obtain a list of all checkpoints in the document (except for checkpoints in table 145 cells, see below);

- GetItemCheckpoint[226](ItemNo) – returns the checkpoint associated with the specified item;
- GetCheckpointByNo[220](No) – returns the checkpoint with the specified index;
- FindCheckpointByName[217](Name) – returns the first checkpoint having the specified Name;
- FindCheckpointByTag[217](Tag) – returns the first checkpoint having the specified Tag;
- event OnCheckpointVisible[274] (Sender: TRichView; CheckpointData: TCheckpointData); this event is generated when:
  - RichView.CPEventKind[173] = *cpeWhenVisible* and some checkpoint (with RaiseEvent=*True*) becomes visible *(example of application: you can mark pictures in TRichView with checkpoints; when this picture becomes visible you can display information about it in a separate window)*
  - RichView.CPEventKind = *cpeAsSectionStart*; in this mode it's guaranteed that the last event is generated for visible checkpoint or for the nearest checkpoint above the visible area *(example of application: you can mark beginnings of chapters of your document with checkpoints, create a table of contents in ListBox (one listbox item = one chapter = one "RaiseEvent" checkpoint); if you select listbox item associated with the last raised checkpoint, the list box will always have name of the chapter in the field of view highlighted)*

Methods of the second group (extracting information from TCheckpointData):

- GetCheckpointInfo[220](CheckpointData, Tag, Name, RaiseEvent) – returns Tag, Name and "RaiseEvent" flag of checkpoint;
- GetCheckpointXY[221](CheckpointData, X,Y), GetCheckpointYEx[222](CheckpointData): Integer – obtaining information about coordinates of checkpoint (Y is the most useful; you can pass this Y to RichView.ScrollTo[548] method);
- GetCheckpointItemNo[221](CheckpointData) : Integer – returns the index of the associated item; it can return -1 for checkpoint in the end of document (no associated item);
- GetCheckpointNo[221](CheckpointData): Integer – returns the index of the checkpoint;

Methods belonging to the both groups:

- GetCheckpointY[221](no): Integer – equivalent to GetCheckpointYEx(GetCheckpointByNo(No));

Methods of editor

The first group of methods contains editing-style analogs of SetCheckpointInfo and RemoveCheckpoint:

- SetCheckpointInfoEd[373] – adds checkpoint for the specified item, as an editing operation;
- RemoveCheckpointEd[366] – deletes checkpoint for the specified item, as an editing operation.

The second group contains methods working with the item at the position of caret (if the caret is between items, they work with the item to the left):

- GetCurrentCheckpoint[343] – returns checkpoint associated with the item at the caret position;
- SetCurrentCheckpointInfo[376] – adds/modifies checkpoint for the item at the caret position, as an editing operation;
- RemoveCurrentCheckpoint[367] – deletes checkpoint for the item at the caret position, as an editing operation.

The third group contains methods working with the checkpoint at the position of caret:

- InsertCheckpoint[351] – inserts checkpoint in the caret position (adds/modifies checkpoint for the item to the right; if the caret was in the middle of text item, the method preliminarily splits this item at the caret position), as an editing operation;
- GetCheckpointAtCaret[341] – returns the checkpoint exactly at the caret position;
- RemoveCheckpointAtCaret[366] – deletes the checkpoint exactly at the caret position, as an editing operation.

RTF and HTML

In HTML[96], checkpoints are saved as anchors (<a name> tag). Either checkpoint name or checkpoint index can be used, see *rvsoUseCheckpointsNames* option[686] for SaveHTML[250] and SaveHTMLEx[251]. Links to checkpoints must start with '#' character.

In RTF, checkpoints are saved as bookmarks (checkpoints names are used). When importing RTF, bookmark start is read as checkpoint (bookmark name is written in checkpoint name). When reading RTF, links to

bookmarks are converted to links started from '#'. When writing RTF, links started from '#' are written as links to bookmarks.

Example 1

Enumerating all checkpoints in the document (not including checkpoints in table cells). This method is very fast even for large documents, because all checkpoints are organized in a special list.

```
var Tag: TRVTag[692];
    Name: TRVUnicodeString[694];
    RaiseEvent: Boolean;
    CheckpointData: TCheckpointData;
begin
  CheckpointData := MyRichView.GetFirstCheckPoint[223];
  while CheckpointData<>nil do begin
    MyRichView.GetCheckpointInfo[220](CheckpointData,Tag,Name,RaiseEvent);
      // processing this checkpoint
    ...
    CheckpointData := MyRichView.GetNextCheckpoint[234](CheckpointData);
  end;
end;
```

Example 2

Enumerating all checkpoints in the document, including checkpoints in table cells

```
procedure EnumCheckpoints(RVData: TCustomRVData[650]);
var i,r,c: Integer;
  table: TRVTableItemInfo[565];
  Tag: TRVTag[692];
  Name: TRVUnicodeString[694];
  RaiseEvent: Boolean;
  CheckpointData: TCheckpointData;
begin
  for i := 0 to RVData.ItemsCount-1 do begin
    CheckpointData := RVData.GetItemCheckpoint[226](i);
    if CheckpointData<>nil then begin
      RVData.GetCheckpointInfo[220](CheckpointData,Tag,Name,RaiseEvent);
      // processing this checkpoint
      ...
    end;
    if RVData.GetItem[225](i) is TRVTableItemInfo[565] then
    begin
      table := TRVTableItemInfo[565](RVData.GetItem[225](i));
      for r := 0 to table.RowCount[580]-1 do
        for c := 0 to table.ColCount[575]-1 do
          if table.Cells[574][r,c]<>nil then
            EnumCheckpoints(table.Cells[574][r,c].GetRVData[613]);
    end;
  end;
end;
```

Call:

```
EnumCheckpoints(MyRichView.RVData[187]);
```

In the second example, RVData.GetCheckpointXY[221] returns checkpoint coordinates relative to the top left corner of this RVData (it may be left top corner of the cell). In order to get absolute coordinates of this checkpoint in the document (for MyRichView.ScrollTo[548]), obtain the coordinates of top left corner of this

RVData (TCustomRVFormattedData(RVData).GetOriginEx(X,Y)), relative coordinates of checkpoint (TCustomRVFormattedData(RVData).GetCheckpointXY(X,Y)), and sum them up

See also...

See also properties and events of TRVStyle:

CheckpointColor, CheckpointEvColor 420, OnDrawCheckpoint 444

See also properties of TRichView:

Options 182, .CPEventKind 173

See also events of TRichView:

OnCheckpointVisible 274

See also methods of TRichView:

AddCheckpoint 202, SetCheckpointInfo 261, RemoveCheckpoint 248, GetFirstCheckpoint 223, GetNextCheckpoint 234, GetItemCheckpoint 226, GetCheckpointByNo 220, FindCheckpointByName 217, FindCheckpointByTag 217, GetCheckpointInfo 220, GetCheckpointXY 221, GetCheckpointYEx 222, GetCheckpointItemNo 221, GetCheckpointNo 221, GetCheckpointY 221

See also methods of TRichViewEdit:

GetCurrentCheckpoint 343, SetCheckpointInfoEd 373, RemoveCheckpointEd 366

1.2.1.3 Items' "Tags"

Each item of TRichView has an associated string value – *tag*. *Tags* are provided for the convenience of storing additional string value for special needs in your application. The main purpose of tags is organizing a work with a hypertext.

Do not confuse RichView *items tags* with "Tag" properties of TComponent descendants! (TRichView uses "Tag" properties of inserted controls for its own needs; you can set "Tag" properties of ImageList for organizing saving/loading "bullets" and "hotspots" from RVF; but this topic is about completely different tags!)

What you need to know about items tags:

- if you do not want to use them – just forget about them; they always will be equal to '';
- "checkpoints" 65 also have tags; all rules for items tags work for checkpoints tags;
- table cells also have tags 612; all rules for items tags work for checkpoints tags;
- editor can clear (set to '') tags when applying different style to text according to RichViewEdit.EditorOptions 325;
- If you want to save and load document in RVF you must provide that tag strings do not contain #1 and #2 characters;
- You can use AddXXX methods 77 which do not have Tag parameter. These functions create new items with Tag=''.

Unicode note: tags are Unicode strings.

Example:

```
var Tag: TRVTag 692;
    S: TRVUnicodeString 694;
// Converting the first text item and its tag string to upper case
  MyRichView.GetTextInfo 239(0, S, Tag);
  MyRichView.SetItemText 267(0, UpperCase(S));
  MyRichView.SetItemTag 267(0, UpperCase(Tag));
```

Tags in older version of TRichView

In TRichView versions prior to v13.2, tags were Integers. Depending on *rvoTagsArePChars* Option[182], they can be either plain integer values or pointers to strings.

This mode can be returned by defining RVOLDTAGS in RV_Defs.inc.

See also...

TRichView methods for adding items with tags to the end of document (a tag is an optional parameter):

AddNL[203], AddBreak[196], AddPicture[204], AddHotspot[200], AddBullet[197], AddControl[198], AddBreak[196]

TRichView methods for adding "checkpoints" with tags to the end of document (a tag is an optional parameter):

AddCheckpoint[202].

TRichView methods for obtaining information about items (including tags):

GetItemTag[230], GetBreakInfo[218], GetBulletInfo[219], GetHotspotInfo[224], GetPictureInfo[236], GetControlInfo[222], GetTextInfo[239].

TRichView methods for modifying items (including tags):

SetItemTag[267], SetBreakInfo[260], SetBulletInfo[261], SetHotspotInfo[265], SetPictureInfo[269], SetControlInfo[262].

TRichView methods allowing work with checkpoints tags:

FindCheckpointByTag[217], GetCheckpointInfo[220], SetCheckpointInfo[261].

TRichViewEdit methods for obtaining information about item in caret position:

GetCurrentTag[349], GetCurrentBreakInfo[341], GetCurrentBulletInfo[342], GetCurrentHotspotInfo[344], GetCurrentPictureInfo[349], GetCurrentControlInfo[343], GetCurrentTextInfo[350].

TRichViewEdit methods for inserting:

InsertStringTag, InsertStringATag, InsertStringWTag[359]

TRichViewEdit methods for modifying items (including tags):

SetItemTagEd[384], SetBreakInfoEd[371], SetBulletInfoEd[372], SetHotspotInfoEd[382], SetPictureInfoEd[386], SetControlInfoEd[374].

TRichViewEdit methods for modifying information about item in caret position:

SetCurrentTag[382], SetCurrentBreakInfo[375], SetCurrentBulletInfo[375], SetCurrentHotspotInfo[377], SetCurrentPictureInfo[381], SetCurrentControlInfo[377]

TRichViewEdit methods for working with checkpoints tags:

SetCheckpointInfoEd[373], SetCurrentCheckpointInfo[376].

TRichViewEdit options affecting tags:

EditorOptions[325].

1.2.1.4 Hypertext Overview

TRichView has no default actions on hypertext events. However, it provides a convenient way for you to create hypertext document and define any actions that you want in response on user click on documents. RichView provides visual effects for items-hypertext links (highlighting).

There are the following types of hypertext items:

- text of hypertext styles[479];

- tabs 121 and label items 133, linked to hypertext styles 479;
- *hot pictures* 125
- *hotspots* 129

RichView provides the following indication of hypertext items (when the mouse pointer moves above them):

- changing color of text items (TFontInfo.HoverColor 470, HoverBackColor 470);
- underlining text items (TFontInfo.HoverEffects 471);
- changing mouse cursor (any hypertext text style can have its own cursor, see TRVStyle.JumpCursor 427, TFontInfo.JumpCursor 472)
- for hotspots 129 it can change picture (there are two pictures for hotspot – "normal" and "hot"; they must be in the same imagelist but can have different indices);

By default, main events of hypertext (OnRVMouseMove 290, OnJump 282) and cursor indication do not work in editor (TRichViewEdit 316). Instead of highlighting items under mouse pointer editor highlights the item at the position of caret (item that is being edited).

But editor can be switched to hypertext mode when user presses and holds Ctrl key, if *rvoCtrlJumps* is in EditorOptions 325.

Since version 1.6, hypertext works in ReadOnly 328 editor just like in viewer (since version 13, this behavior can be turned off using *rvoNoReadOnlyJumps* option from EditorOptions 325)

Besides, you can provide some specific actions in response on user clicking any RichView item, using OnRVMouseDown 289 and OnRVMouseUp 291 events.

Hypertext

Any hypertext link (text, *hotspot* or *hot picture*) has its own **id**. This **id** is passed in hypertext events - OnRVMouseMove 290 and OnJump 282. The first hypertext item has **id** equal to RichView.FirstJumpNo 179, the next one is greater by one and so on. By default, RichView.FirstJumpNo=0 and hypertext items have **id** = 0, 1, 2, ...

Use OnRVMouseMove 290 to provide action when user moves cursor to hypertext link. For example, you can change text in the status bar.

Example

```
procedure TMyForm.MyRichViewRVMouseMove(Sender: TObject; id: Integer);
begin
  if id=-1 then
    MyStatusbar.SimpleText := ''
  else
    MyStatusbar.SimpleText := 'Click here';
end;
```

Special value (-1) is passed to this event as **id** when the user moves the mouse pointer outside any hypertext link. This event never occurs with the same **id** two times one after one.

The main hypertext event is OnJump 282.

Example

```
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
begin
  case id of
    0: Application.MessageBox('Ah!','',0);
    1: Application.MessageBox('Ugh!','',0);
    2: Application.MessageBox('Oops!','',0);
  end;
end;
```

The method shown in the example above (using **id** for defining specific actions) was the only available method in the older versions of RichView. Now you can use other ways, described below.

Sometimes information contained in the clicked item is enough to know what to do. The key method here is GetJumpPointLocation[231] This function allows you to obtain the index of the clicked item. For example, if you have only text links (no *hotspots* and *hot pictures*) and they are URLs, you can write:

Example

```
uses ..., CRVFData, ShellApi;
...
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var URL: String;
    RVData: TCustomRVFormattedData[650];
    ItemNo: Integer;
begin
  MyRichView.GetJumpPointLocation[231](id, RVData, ItemNo);
  URL := RVData.GetItemText[230](ItemNo);
  ShellExecute(Application.Handle, 'open', PChar(URL),
    nil, nil, SW_NORMAL);
end;
```

In the example above, we use GetJumpPointLocation[231] to get the document containing this hyperlink (RVData) and the index of the clicked hypertext item in this document (ItemNo).

Next, we get information about the clicked text item with GetItemText and use the received string to launch a browser.

If you have both text and *hotspot/hot pictures* hypertext items, you can use RVData.GetItemStyle[229] method to recognize the item type ("style"), and next use GetTextInfo[239] or GetHotspotInfo[224]/GetPictureInfo[236]. For *hotspots* you can use image indices and *hotspot* name to decide which action to do.

The method shown in the next example is universal. You can use *tags* to add your information to any RichView item. (See tags[68]).

Example

```
uses ..., CRVFData, ShellApi;
...
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var URL: String;
    RVData: TCustomRVFormattedData[650];
    ItemNo: Integer;
begin
  MyRichView.GetJumpPointLocation[231](id, RVData, ItemNo);
  URL := RVData.GetItemTag[230](ItemNo);
  ShellExecute(Application.Handle, 'open', PChar(URL),
    nil, nil, SW_NORMAL);
end;
```

Inserted controls cannot be hyperlinks. But, of course, you can set your own cursors and mouse events for any inserted control.

Hypertext and tables

Hypertext works not only in a the main document but also in table cells (subdocuments)

Hypertext links are numbered sequentially through the document, including links in cells. All links in the document have unique indices (**id**).

If you have a hypertext **id** you can obtain the hypertext item location with the method:

```
procedure TRichView.GetJumpPointLocation[231](id: Integer;
  var RVData: TCustomRVFormattedData; var ItemNo: Integer);
```

A hypertext item is the ItemNo-th item in RVData object.

RVData can be RichView[Edit].RVData[187], cell[605], or RVData[187] of inplace editor.

See also...

- OnReadHyperlink[284], OnWriteHyperlink[302];
- example how to create a list of hyperlinks and move the caret to the selected hyperlink: ***http://www.trichview.com/forums/viewtopic.php?t=397***.

1.2.1.5 Paragraphs

Paragraphs

TRichView documents are organized in paragraphs. Each paragraph has its attributes ("style") defined as an index in the collection of paragraph styles (RichView.Style[193].ParaStyles[431]).

For the list of properties available for paragraphs, see TParaInfo[482] class (TParaInfo is a class of item in the collection of paragraph styles).

Line breaks inside paragraphs

(Since version 1.2) You can move am item to the new line within the same paragraph (press Shift+Enter instead of Enter). Visually, such line breaks have the following differences from paragraph breaks:

- FirstIndent[485] is not applied to the first line after the break;
- no additional spacing between this line and the previous line (SpaceBefore[489] and SpaceAfter[489]) is applied;
- if the paragraph has border/background[73], this line will be inside the same border/background as the previous line.

Specifying paragraph attributes when generating document

All methods of TRichView for appending items have **ParaNo** parameter. It can be:

- *zero or positive value*: index in the collection of paragraph styles, if this item will start a new paragraph;
- *-1*, if this item will continue the paragraph (*please do not add items with ParaNo=-1 after breaks*[125] or tables[565]).

Breaks (horizontal lines) do not have paragraph style, so AddBreak method does not have ParaNo parameter. But SpaceBefore[489] and SpaceAfter[489] values of the 0-th paragraph style are added to spacing of all breaks.

These methods are:

```
procedure AddNL[203](s, StyleNo, ParaNo, Tag);
procedure AddFmt[198](FormatStr, Args, StyleNo, ParaNo);
procedure AddTextNL[206](s, StyleNo, FirstParaNo, OtherParaNo,
  DefAsSingleParagraph, Tag);
procedure AddPicture[204](Name, gr, ParaNo, VAlign, Tag);
procedure AddHotspot[200](Name, ImageIndex, HotImageIndex,
  ImageList, ParaNo, VAlign, Tag);
procedure AddBullet[197](Name, ImageIndex, ImageList, ParaNo, VAlign, Tag);
procedure AddControl[198](Name, ctrl, ParaNo, VAlign, Tag);
procedure Add[203](s, StyleNo); // equivalent to AddNL(s, StyleNo, -1).
```

See also: building a document[77], item types[64].

See also methods of TRichView:

```
function LoadText[246](FileName, StyleNo, ParaNo, AsSingleParagraph, CodePage):Boolean;
function LoadTextW[246](FileName, StyleNo, ParaNo, DefAsSingleParagraph):Boolean;
function AppendRVFFromStream[208](Stream, ParaNo):Boolean;
function IsParaStart[242](ItemNo): Boolean;
function IsFromNewLine[242](ItemNo): Boolean;
function GetItemPara[229](ItemNo): Integer;
```

Text flow around left- and right-aligned items

Some items (for example, pictures[122]) can be aligned to the left or the right side (see TRVVAlign[695]). For such items, an invisible placeholder is left in the place of their insertion; the item itself is placed at the specified side. If the item is at the beginning of the line, this item is placed on the same level; otherwise, it is placed below this line.

Paragraphs flow around side-aligned images. You can disallow flowing around left- and/or right-aligned items using ClearLeft[171] and/or ClearRight[172] properties.

Editing

RichViewEdit has a group of Insert*** methods inserting new item(s) in the position of caret, using the current paragraph style. Usually, if you implement RichViewEdit-based editor, you need to provide interface for displaying and changing the current paragraph style. You can do it using ComboBox with style names, toolbar buttons or/and keyboard shortcuts.

```
property CurParaStyleNo[323]: Integer;
```

– index of the current (in the position of caret) paragraph style;

When the user moves the caret to paragraph with different style, OnCurParaStyleChanged[391] event is generated, so you can select another item in ComboBox of styles or check toolbar buttons.

Assigning to CurParaStyleNo property is possible, but will not create a desired effect. To apply style to the paragraph at the position of the caret (and all selected paragraphs) use

```
procedure ApplyParaStyle[331](ParaStyleNo: Integer);
```

ApplyParaStyle applies only one paragraph style to all selected paragraphs. This method does not allow to implement more complex commands, such as "increase indents", "align to the right", etc. For implementing these commands, use

```
procedure ApplyParaStyleConversion[332](UserData: Integer);
```

See also...

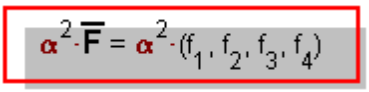
Borders and Background of Paragraphs[73]

1.2.1.6 Borders and Background of Paragraphs

Paragraphs can have borders and/or background color. These properties are defined in paragraph attributes ("styles") (see TParaInfo[482]). Border is defined in Border[485] property, background is in Background[484] property.

By default, there is no border around paragraph (TRVStyle.ParaStyles[i].Border.Style[502]=*rvbsNone*), and background of paragraph is transparent (TRVStyle.ParaStyles[i].Background.Color=*clNone*).

Border offsets and colored area offsets are defined separately from each other, so you can draw border inside colored area, border around colored area with gap, and even more interesting effects:



Example of paragraph border and background

Border and colored area both have BorderOffsets property, which defines:

- for borders: a gap between the border and the paragraph contents;
- for background: a colored area overhang (padding) from the paragraph content.

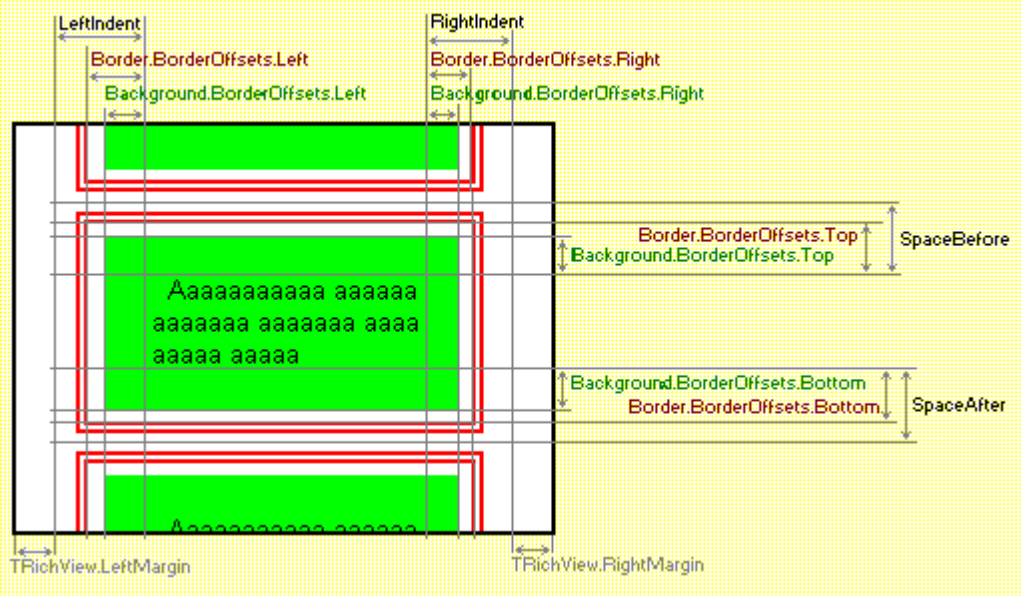
Top, right, bottom, left offsets (i.e. gap and padding) are defined by Top, Right, Bottom, Left properties of BorderOffsets.

Positive values increase gap (padding). Negative values put the border and edges of the colored area inside the paragraph content.

In the example above, Background.BorderOffsets = (Left: -7; Top: -7; Right: 7; Bottom: 7), and values are measured in pixels.

BorderOffsets.Top must not exceed SpaceBefore[489] property of the same paragraph.

BorderOffsets.Bottom must not exceed SpaceAfter[489] property of the same paragraph.



Scheme: properties of paragraph border, background and layout

You can hide some edges of border using Border.VisibleBorders[502] property.

Breaks[125] use SpaceBefore and SpaceAfter values of the 0-th paragraph style, but they never have a border and a colored background.

It's possible to create a line break without starting a new paragraph. In editor, press Shift+Enter instead of Enter. Items after the caret will be moved to the new line, but still they will be inside the same paragraph (inside the common border and the colored area, no SpaceBefore and SpaceAfter will be applied).

See also...

TParaInfo.Border[485]

TRVBorder[501]

1.2.1.7 Styles and StyleTemplates

As you could see, this manual sometimes calls items in TextStyles[437] and ParaStyles[431] "styles", and sometimes it calls them simply "text/paragraph attributes". Apart from historical reasons, there are other reasons for this dual naming, resulting from different cases of use.

Using a predefined set of styles

In this mode, you have some predefined set of items in TextStyles[437] and ParaStyles[431], and they are not changed while the application works.

In this mode, these items may be called "styles".

Example

In this example, RichView1: TRichView[162] is linked to RVStyle1: TRVStyle[418].

```
RVStyle1.TextStyles.Clear;
```

```
with RVStyle1.TextStyles.Add do begin
  StyleName[465] := 'Normal Text';
end;
with RVStyle1.TextStyles.Add do begin
  StyleName := 'Title Text';
  Size[475] := 14;
  Style[475] := [fsBold];
end;
RVStyle1.ParaStyles.Clear;
with RVStyle1.ParaStyle.Add do begin
  StyleName := 'Normal Paragraph';
end;
with RVStyle1.ParaStyle.Add do begin
  StyleName := 'Title Paragraph';
  Alignment[483] := rvaCenter;
  Options[487] := [rvpaoKeepWithNext];
end;
...
RichView1.Clear[210];
RichView1.AddNL[203]('Chapter 1', 1, 1);
RichView1.AddNL('Once upon a time...', 1, 1);
RichView1.Format[217];
```

You can change properties of styles to change the appearance of the created document, without recreating the document itself:

```
with RVStyle1.TextStyles[1] do begin
  Color[469] := clRed;
  FontName[470] := 'Verdana';
end;
with RVStyle1.ParaStyles[1] do begin
  Alignment := rvaLeft;
end;
RichView1.Reformat[248];
```

You can configure[168] RichView1 to allow insertion of RVF and RTF files/streams without adding new styles.

However, this case of usage is very limited. Normally, users want to apply new formatting to parts of a document: change colors, fonts, alignments, etc., so it is impossible to create all necessary styles initially.

Using a dynamic set of styles

In this mode, you can have some initial set of items in TextStyles[437] and ParaStyles[431]; but the most of items are added while the application works. For example, new items may be added when inserting RVF or RTF files/streams, or when applying changes to the selection (ApplyStyleConversion[333], ApplyParaStyleConversion[332]). RichViewActions add new items as well.

In this mode, items of TextStyles[437] and ParaStyles[431] work more like direct text/paragraph attributes rather than styles: the most of them do not have a special unique name or meaning.

Some items of TextStyles[437] and ParaStyles[431] may be marked as Standard[464] (usually, the items added initially). They may have a special meaning, they are not removed by DeleteUnusedStyles[216], they can be saved in RTF as a style sheet (if *rvrtfSaveStyleSheet* in RTFOptions[185]). However, this is not a complete solution for styles, because all other items are completely independent of them: if you change properties of standard styles, only text/paragraphs formatted using them are changed; all other parts of the document are unstyled. It makes standard styles almost useless, so usually an application should create only one standard text style and one standard paragraph style (defaults for new documents).

The complete solution for implementing real text and paragraph styles is "style templates".

Style templates (+dynamic set of styles)

StyleTemplates[435] is a collection in TRVStyle[418] component. Unlike TextStyles[437] and ParaStyles[431], style templates do not define text and paragraph properties of document items[118] and paragraphs directly. Instead, they provide a mechanism for modifying TextStyles[437] and ParaStyles[431].

Using style templates is optional; they are used only if TRichView.UseStyleTemplates[195]=*True*.

Document items do not have direct links to style templates, they are still linked with TextStyles[437] and ParaStyles[431]. But items in TextStyles[437] and ParaStyles[431] may be linked to style templates using StyleTemplateId[465] property.

Each item in a document may be (indirectly) linked to up to 2 style templates:

- via its paragraph style, or
- via its text style (if this is a text item[120], tab[121], label[133], sequence[134], footnote[136], endnote[135], or note reference[140]).

It's recommended (but not necessary) to link every paragraph style to some style template. The recommended name[496] for the default paragraph style template is "Normal".

As for text styles, it is recommended to link them to style templates only if it is necessary.

There are 3 kinds[495] of style templates: applicable to text, applicable to paragraphs, applicable to text and paragraphs.

Each style template may have properties of both text[497] and paragraph[497]. Unlike items of TextStyles or ParaStyles, a style template:

- may define only a subset of text and paragraph properties (listed in ValidTextProperties[498] and ValidParaProperties[498]);
- may inherit text and paragraph properties from a parent style template[497].

When some document item is (indirectly) linked to both a paragraph style template and a text style template, both these style templates are linked with its text style: a text style template as TextStyles[437][StyleNo].StyleTemplateId[465], a paragraph style template as TextStyles[437][StyleNo].ParaStyleTemplateId[473]. TRichView automatically provides for each document item that TextStyles[437][StyleNo].ParaStyleTemplateId[473] = ParaStyles[431][ParaNo].StyleTemplateId[465]. In such cases, a text style template has a priority in defining text attributes: a paragraph style template defines only text attributes not defined by a text style template.

Not all properties of items in TextStyles[437] and ParaStyles[431] are defined by their style template: they can have some properties changed. A list of changed properties is not stored, it is calculated when it is needed. These changes may be (optionally) kept when applying another style template to these items, and they are kept when StyleTemplates[435] are edited.

TRichViewEdit has the following methods for applying the specified style template to the selection: ApplyStyleTemplate[333], ApplyTextStyleTemplate[334], ApplyParaStyleTemplate[332]. The same methods may be used to clear a style template or additional formatting.

Style templates can be edited using ChangeStyleTemplates[338] method.

1.2.1.8 Hidden Text

Hidden items

Some items can be hidden. Hidden items are displayed only if *rvoShowHiddenText* is included in the Options[182] property of TRichView[162] control (they are displayed with a special dotted underline).

Text items[120] are hidden, if *rvteoHidden* is included in the Options[472] property of their style[478].

Non-text items[64] are hidden, if a non-zero value is assigned to *rvepHidden* extra item integer property[673].

Import and Export

RVF (RichView Format): hidden items are saved and loaded as hidden items.

RTF (Rich Text Format): hidden items are saved as hidden text and objects. Loading depends on the value of RTFReadProperties[187].SkipHiddenText[309] property of TRichView[162] control. By default (*False*), hidden text and objects are not loaded from RTF (skipped).

HTML: The CSS version of HTML export (SaveHTMLEx[251]) saves hidden items as hidden text and objects. The simple version (SaveHTML[250]) saves hidden items like normal items.

Text: the text saving methods save a hidden text like a normal text. However, when copying selection to the Clipboard as a text, hidden text is not included.

Performance

A text formatting procedure is somewhat faster, if the option for displaying hidden items (*rvoShowHiddenText* in Options[182]) is set.

1.2.2 Basic Operations

1.2.2.1 Building a Document

Creating a New Document

You can build document by:

1. clearing TRichView,
2. appending items,
3. reformatting and redrawing RichView.

Therefore, a general structure of document generation procedure is:

```
MyRichView.Clear[210];
<Methods for adding contents>
MyRichView.Format[217];
```

Note: all methods introduced in RichView (except for the formatting methods) do not reformat document, do not redraw RichView and do not call OnChange[390] event (RichView does not have OnChange event, it is introduced in TRichViewEdit) (viewer-style methods)

A group of methods for adding items is the largest group of methods (and contains a large number of methods for backward compatibility).

Appending Text Items[120]

Important: Do not pass string with cr, lf or cr+lf, tab and page break characters in the functions below, except for AddText*** methods!

Some text items can be hypertext. Hypertext items are added by the same methods as regular text items.

Methods working with String parameter

These methods add text specified in a String parameter. This parameter is Unicode for Delphi/C++Builder 2009, and ANSI for the older versions of Delphi/C++Builder. Since TRichView stores text as Unicode, when using ANSI parameter, the methods perform conversion to Unicode.

- AddNL(S, StyleNo, ParaNo, Tag)[203] – adds one item with text *S*; *StyleNo* is an index in the collection of text styles (TRichView.Style.TextStyles[437]); *ParaNo* is an index in the collection of paragraph styles (RichView.Style.ParaStyles[431]) (if this item will start a new paragraph) or -1 (if this item will continue the paragraph). Optionally, you can specify Tag[68].
- Add(S, StyleNo)[203] – equivalent to AddNL(S, StyleNo, -1).

- AddTextNL(s, StyleNo, FirstParaNo, OtherParaNo, Tag)[206] – adds one or more text items; the first item is added with FirstParaNo paragraph style, and other items are added as new paragraphs with OtherParaNo paragraph style; items are separated by cr, lf or cr+lf characters in s; s can contain page break and tab characters. Optionally, you can specify Tag[68].
- AddFmt(FormatStr, Args, StyleNo, ParaNo)[198] adds Format(FormatStr, Args) string, where Format is a function from SysUtils unit.

Methods working with ANSI/Unicode explicitly

- AddNLA, AddNLW[203], AddTextNLA, AddTextNLW[206]

Appending Images[122]

You can add picture of any Delphi format – TBitmap, TIcon, TMetafile, TJpegImage or third party graphic format[720].

You should create a picture and TRichView will destroy it itself when necessary (for example, when you call RichView.Clear[210]).

AddPicture(Name, gr, ParaNo, VAlign, Tag)[204], where:

- Name: TRVUnicodeString[694] – name of picture; this string is not used by TRichView itself; this string should not contain cr and lf characters;
- gr: TGraphic - picture;
- ParaNo:Integer is an index in the collection of paragraph styles (TRichView.Style.ParaStyles) (if this picture will start the paragraph) or -1 (if this picture will continue paragraph);
- VAlign: TRVVAlign – vertical alignment of a picture;
- optionally, you can specify Tag[68].

Appending *Hot Images*[125]

- AddHotPicture[199]

Appending Delphi Controls[126]

You can add any Delphi control in TRichView, and it will work as usual. You should create the control and TRichView will destroy it itself when necessary (for example, when you call RichView.Clear). The method for adding controls assigns TRichView to control.Parent.

AddControl(Name, ctrl,ParaNo, VAlign, Tag)[198] adds a control, just like AddPicture adds a picture.

This version of TRichView, like old ones, uses Tag properties of inserted components for internal needs (do not confuse with TRichView items tags).

Appending *Bullets*[128] – Images from ImageLists

AddBullet (Name, ImageIndex, ImageList, ParaNo, VAlign, Tag)[197], where:

- Name: TRVUnicodeString[694] – name of the bullet; this string is not used by TRichView itself; this string should not contain cr and lf characters;
- ImageIndex: Integer – index of the picture in image list;
- ImageList: TCustomImageList – image list; this will be just a reference to imagelist; TRichView does not hold copy of image list, and does not free it;
- ParaNo:Integer is an index in the collection of paragraph styles (TRichView.Style.ParaStyles) (if this bullet will start a paragraph) or -1 (if this bullet will continue paragraph)).
- VAlign: TRVVAlign – vertical alignment of a bullet;
- optionally, you can specify Tag[68].

Appending *Hotspots*[129] – Hypertext Images from ImageLists

Hotspots are just like *bullets*, but can have two images ("normal" and "hot") and can be used as hypertext link.

AddHotspot(Name, ImageIndex, HotImageIndex, ImageList, ParaNo, VAlign, Tag)[200] is similar to AddBullet. It has one additional parameter (HotImageIndex) defining the image index used when this item is below the mouse pointer.

Appending *Breaks*[125] – Horizontal Lines

- AddBreak.[196]

Adding Paragraphs' Bullets and Numbering[131] (list markers)

List markers can be added using SetListMarkerInfo[269] method. Unlike the methods above, this method can insert list markers not only to the end of document, but set them for existing paragraphs.

Adding Other Items

Tables[130], labels[133], numbered sequences[134], endnotes[135], footnotes[136], references to parent footnotes and endnotes[140], page numbers[141], page counts[142], equations[143], custom item types[144] do not have special methods for adding them to documents. All of them are added by AddItem[201] method.

Appending *Checkpoints*

Checkpoints are not items of TRichView, but the method for adding *checkpoints* is similar to methods for adding items: AddCheckpoint[202]. See "checkpoints"[65].

Adding Page Breaks

Page break is an item property ("this item starts a new page" flag). Only items that start a new paragraph can start a new page.

Example:

```
var ItemNo: Integer;

MyRichView.AddNL[203]('First page', 0, 0);
ItemNo := MyRichView.ItemCount[180];
MyRichView.AddNL[203]('Second page', 0, 0);
MyRichView.PageBreaksBeforeItems[184][ItemNo] := True;
```

Appending and Inserting RVF[93]

You can append RVF data by the methods:

- function InsertRVFFromStream[240](Stream: TStream; Index: Integer):Boolean;

This method can insert RVF. The first item of RVF will have the specified Index. For example, if you want to append items from RVF, write:

```
MyRichView.InsertRVFFromStream(MyStream, MyRichView.ItemCount);
```

InsertRVFFromStream is the only method of RichView (not RichViewEdit) for inserting items.

- function AppendRVFFromStream[208](Stream: TStream; ParaNo: Integer):Boolean;

This method is similar to InsertRVFFromStream (if it used for appending RVF), but it ignores paragraph style of the first item in RVF and uses ParaNo instead of it. ParaNo is an index in the collection of paragraph styles (RichView.Style.ParaStyles[431]) (if item will start a paragraph) or -1 (if the item will continue paragraph).

Appending Content of Another TRichView

- procedure AppendFrom[207](Source: TCustomRichView);

This is the most efficient method to copy contents of one RichView to another one. But items of some types (including Inserted controls and tables) are not copied with this method (just ignored).

Loading

Methods for loading [92] (**Load*****) belong to this group of methods as well.

Additional Notes

There is an important change in version 1.2: you can add items from new line without starting a new paragraph. There are no new methods for generating documents (it is added in v1.3: AddTextNLW [206]), but there is a method that affects their behavior: TRichView.SetAddParagraphMode [259].

Some custom item types (such as tables [145]) do not have special methods for appending them. They can be appended by general method AddItem [201].

1.2.2.2 Scrolling

Vertical scrolling

Documents in RichView can have almost unlimited height. But ranges of scrollbars are limited, and RichView cannot use pixels as scrolling units (small step of scrollbar). Scrolling units of RVScroller (ancestor of RichView) are referred here as "RVSU" (RichView Scrolling Units). By default, 1 RVSU = 10 pixels, but the component can increase it automatically during formatting if the range of vertical scrollbar exceeds 30,000. You can set a number of pixels in 1 RVSU with VSmallStep [196] property (but the component can increase it during formatting).

For example, you may want to set RVSU to the height of one line of the normal text. After assigning a new value to VSmallStep, you should reformat document using Format [217] method).

You can get or set position of vertical scrollbar using VScrollPos [547] property (in range 0..VScrollMax [547])

You can also use method ScrollTo [548] for scrolling to the specified vertical position, measured in pixels.

ScrollTo(y) is equivalent to VScrollPos := y div VSmallStep, so ScrollTo cannot scroll exactly to the specified position.

You can show and hide vertical scrollbar with VScrollVisible [547] property.

DocumentHeight [177] property is a vertical height of a document in pixels (height of scrollable area). For example, you can use this value to set height of RichView. DocumentHeight includes values of TopMargin [194] and BottomMargin [171].

You can get vertical coordinate in document for:

- *checkpoints* [65]
- hypertext link [69] (GetJumpPointY [232])
- any item (GetItemCoords [226])

Horizontal scrolling

Horizontal scrollbar appears automatically when necessary.

You can hide it with HScrollVisible [545] property.

You can get or set the position of horizontal scrollbar using HScrollPos [545] property (in range 0..HScrollMax [545])

You can use some properties to control width of a document (a width of scrollable area minus margins), thus controlling a horizontal scrolling (see LeftMargin [180], RightMargin [185])

By default, if the document does not contain wide images, controls or tables, text is wrapped to fit the client width of RichView minus margins.

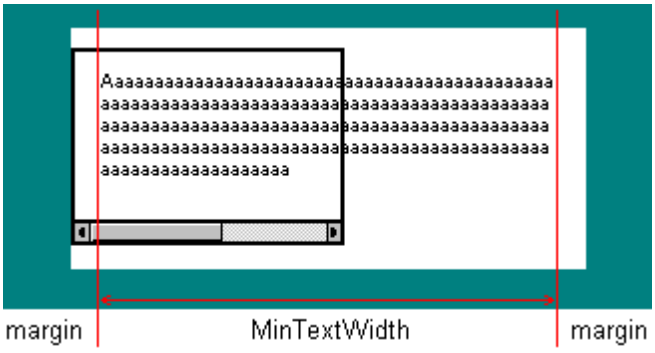
If there are pictures or controls, width of scrolling area is equal to

```
max(client width of RichView,
    "maximal width of pictures or controls" + "maximal left [485] + first line [485] + right [488]
indent" + margins),
```

and text is wrapped to fit width of the scrolling area minus margins.

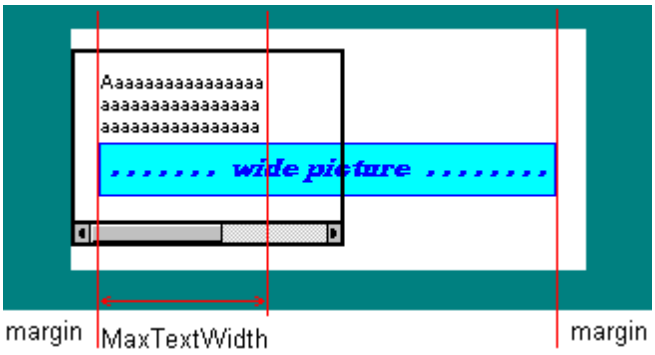
Text in paragraphs with no-wrap option[487] can also affect to width of scrolling area, in the same was as wide pictures or controls.

Use MinTextWidth[181] property to set the minimal width for text wrapping (text wraps when its length in pixels exceeds this value). Horizontal scrollable area cannot be less than MinTextWidth+margins (but can be greater because of wide pictures or wide client width of RichView). You can use MinTextWidth to speed up formatting when RichView is resized. Formatting is very quick if width of scrollable area is not changed. For example, if you set MinTextWidth to the size of screen, RichView will not perform full reformatting when user resizes it (well, it's not convenient to read text with horizontal scrollbar)



Explaining MinTextWidth

Use MaxTextWidth[181] property to set maximal width for text wrapping. This value does not affect width of scrollable area.



Explaining MaxTextWidth

A similar effect can be achieved using *rvoClientTextWidth* option from Options[182]. If this option is set, width for text wrapping is always equal to client width of RichView minus margins. It is useful if your document can have wide pictures or controls, or no-wrap paragraphs.

You can set no-wrap option[487] for paragraph style. If this option is set, this paragraph will not be wrapped automatically.

1.2.2.3 Selecting

User can select a part TRichView document with the mouse (in TRichViewEdit he/she also can use Shift +caret movement keys).

You can turn off/on a word selection mode, see TRVStyle.SelectionMode[432].

Selection Appearance

Selected text has RichView.Style.SelTextColor/InactiveSelTextColor[426] color and RichView.Style.SelColor/InactiveSelColor[426] background.

By default, it is *clHighlightText* on *clHighlight* both for active (focused) and inactive windows.

See also RVStyle.SelectionStyle[432].

Operations

You can:

- disallow making selection (clear *rvoAllowSelection* in Options[182]); please do not use in editor!
- determine if the selection is not empty (SelectionExists[258]);
- deselect TRichView (Deselect[216]; Invalidate);
- select the whole document (SelectAll[257]; Invalidate);
- get selection bounds (GetSelectionBounds[237]);
- select a part of the document (SetSelectionBounds[271];Invalidate);
- select the inserted control (SelectControl[257]);
- get the selection as a text (GetSelText, GetSelTextW[238]);
- get the selection as a Rich Text Format (SaveRTFToStream[254], SaveRTF[253]);
- get the selection as an image, if image is selected (GetSelectedImage[237]);
- save the selection to stream or file (SaveRVFToStream[254], SaveRVF[254]);
- copy the selection to the Clipboard in various formats (see RichView Clipboard Functions Overview[82])

In editor you can:

- delete selection (DeleteSelection[340]);
- select the current word[370] or line[369];
- cut selection to the Clipboard (CutDef[340]);

Note: All methods introduced in RichView which can change visual appearance require calling Invalidate or Refresh after them.

Note: All methods working with selection must be called only when the document is formatted.

See also...

Searching and replacing[83]

1.2.2.4 Clipboard

TCustomRichView and its descendants can copy information to the Clipboard as plain (ANSI and Unicode) text, image, RTF and RVF, and your own custom formats.

RichViewEdit can paste information from the Clipboard as text (ANSI and Unicode), image, RTF, and RVF (and your own formats).

Copying

The main method for copying to the Clipboard is CopyDef[212]. It can copy selection in one or more formats.

This method copies data in formats specified in RichView.Options[182]:

Option	Meaning
rvoAutoCopyText	if set, CopyDef copies selection as ANSI text
rvoAutoCopyImage	if set, CopyDef can copy selected image to clipboard
rvoAutoCopyUnicodeText	if set, CopyDef copies selection as Unicode text
rvoAutoCopyRTF	if set, CopyDef copies selection as RTF (Rich Text Format)
rvoAutoCopyRVF	if set, CopyDef copies selection as RVF

CopyDef is called automatically when the user presses Ctrl+Insert or Ctrl+C.

You can also use methods for copying:

- Copy 212 – copies selection to the Clipboard in all formats;
- CopyTextA 213 – copies the selection as ANSI text;
- CopyTextW 213 – copies selection as Unicode text
- CopyImage 212 – if there is a selected image, this method copies it to the Clipboard;
- CopyRVF 213 – copies selection in RVF format 93. This format has name 'RichView Format' and consist of two parts: <Size of RVF><RVF> where <Size of RVF> is 4-byte integer value (number of bytes in <RVF>), <RVF> is RVF data (see: About RVF 93, RVF Specification 109)
- CopyRTF 213 – copies selection as RTF (Rich Text Format).

SelectionExists 258 method answers to the question "does the selection contain some data (not empty)?"

OnCopy 276 event allows copying in your own formats.

Pasting

TRichView cannot paste. All these methods are methods of TRichViewEdit.

Pasting:

- PasteBitmap 362 – pastes bitmap from the Clipboard, if available;
- PasteMetafile 363 – pastes metafile picture from the Clipboard, if available;
- PasteGraphicFile 362 – pastes graphic file(s) the Clipboard, if available;
- PasteText 364 – pastes text from the Clipboard, if available;
- PasteRVF 364 – pastes RVF from the Clipboard, if available;
- PasteRTF 363 – pastes RTF from the Clipboard, if available;
- PasteURL 365 – pastes URL from the Clipboard, if available;
- Paste 361 – tries to paste as an RVF; if cannot – as an RTF, if cannot – as an URL, if cannot – as a text; if cannot – as graphic file(s), if cannot – as a bitmap; if cannot – as a metafile.AcceptPasteFormats 321 property is taken into account. This method is called automatically when user press Ctrl+V or Shift+Insert.

Testing is the Clipboard has the specific format:

- CanPaste 337 – "does the Clipboard contain any format what can be pasted in RichViewEdit?"
- CanPasteRVF 337 – "does the Clipboard contain RVF?"
- CanPasteRTF 337 – "does the Clipboard contain RTF?"

OnPaste 399 event allows to override the default pasting procedure. For example, you can paste data in different format, or you can allow pasting only some specific formats (for example, only a plain text).

Cutting

RichViewEdit.CutDef 340 copies the selection (like CopyDef) and then deletes it (like DeleteSelection 340). This method is called automatically when user presses Ctrl+X or Shift+Delete.

Windows messages

RichView processes *WM_COPY* message, RichViewEdit also processes *WM_PASTE, WM_CUT, EM_CANPASTE.*

See also...

Drag&drop 89

1.2.2.5 Searching and Replacing

The following main functions for text searching are available:

```
function TRichView162.SearchTextW256(s: TRVUnicodeString694; SrchOptions: TRVSearchOptions
function TRichViewEdit316.SearchTextW369(s: TRVUnicodeString694; SrchOptions: TRVESearchOpt
```

These methods search for the substring **s**. If found, they select it an make it visible (scroll the window to it).

The methods of TRichView start searching from the current selection (if exists) or from the first (or the last, when searching up) visible item. If *rvsroFromStart* is included in SrchOptions, the methods start searching from the beginning (or the end, when searching up) of the document.

The methods of TRichViewEdit start searching from the caret position.

All the methods support the following options:

- searching up or down;
- case sensitive/insensitive searching;
- searching for the whole words;
- allowing matching in a single text item or in multiple items.

s may contain a tab character (#9). However, it makes sense only if the option for matching in multiple items is set.

The unit RVMisc.pas contains the functions to convert options of search dialogs:

```
function GetRVSearchOptions[661](fo: TFindOptions): TRVSearchOptions[689];
function GetRVESearchOptions[660](fo: TFindOptions):TRVESearchOptions[672];
```

It's easy to implement replacing of text in editor. SearchText selects the old substring, InsertText[360] replaces the selected string with the new one.

1.2.2.6 Obtaining Items

There are two groups of methods that can be used to obtain information about items of RichView:

- **Get***Info** methods, introduced in RichView, can be used to get information about any item;
- **GetCurrent***Info** methods of RichViewEdit, return information about the "current" (at the caret position) item.

GetCurrent*Info** methods are equivalent to **Get***Info**(CurItemNo[322],...). But there is one important exception: when using tables[145], **GetCurrent***Info** methods return information about the item at the position of caret regardless its location (in root editor or in cell inplace editor).

Read-only property RichView.ItemCount[180] returns the number of items.

```
function RichView.GetItemStyle[229](ItemNo: Integer): Integer;
```

This method returns style (type) of item with the specified index.

If the returned value>=0 then this is a text item and the returned value is an index in the collection of text styles (RichView.Style.TextStyles[437]).

If the returned value <0 then this is a special item. Negative value could be equal to one of the following constants:

Constant	Value	Meaning
rvsBreak	-1	horizontal line[125]
rvsPicture	-3	picture[122]
rvsHotspot	-4	*hotspot*[129]
rvsComponent	-5	Delphi control[126]
rvsBullet	-6	*bullet*[128]
rvsHotImage	-10	*hot-picture*[125]
rvsListMarker	-11	list marker[131] (bullet or numbering of paragraphs)
rvsTab	-12	tabulator[121]
rvsTable	-60	table[130]

See also: rvsXXX constants 711)

When you know a type of item you can get further information about it with the methods (**Get***Info** methods):

- GetBreakInfo 218,GetBulletInfo 219, GetHotspotInfo 224, GetPictureInfo 236, GetControlInfo 222, GetTextInfo 239.
- For item of any type your can get additional property: GetItemExtraIntProperty 227 and GetItemExtraStrProperty 228.
- For item of any type your can get "tag" (GetItemTag 230), "checkpoint" (GetItemCheckpoint 226)
- For any item you can get information about its paragraph: (GetItemPara 229, IsParaStart 242, IsFromNewLine 242).
- For any item you can get its text (text for text item, name for non-text item) (GetItemText 230)
- For any item you can get object representing it in a document (GetItem 225, usually used for tables 565)

In editor you can use the methods above or GetCurrentBreakInfo 341, GetCurrentBulletInfo 342, GetCurrentHotspotInfo 344, GetCurrentPictureInfo 349, GetCurrentControlInfo 343, GetCurrentTextInfo 350 (**GetCurrent***Info** methods), and GetCurrentTag 349, GetCurrentCheckpoint 343, GetCurrentItemText 347.

The arguments of these methods are similar to the arguments of **Add***Tag 77 methods for appending items 77 .

It should be noted that these methods return references to pictures and controls owned by RichView, not to their copy. Please use them wisely. For example, you can draw something on the returned picture but you should not destroy it. The preferred way to modify items of RichView is using **Set***Info 85 (or 85 **Set***InfoEd 85 for editor) 85. Exception: items with *rvepShared* 673 property.

A table in the position of the caret (or a top-level table containing the caret) can be obtained with the method GetCurrentItemEx 345.

1.2.2.7 Modifying Items

There are several groups of methods for modifying content of RichView:

- **Set***Info** methods, introduced in TRichView (viewer-style methods)
- **Set***InfoEd** methods of RichViewEdit (editor-style methods)
- **SetCurrent***Info** methods of RichViewEdit – the same as **Set***InfoEd** but work with the "current" (in the caret position) item.

SetCurrent*Info** methods are equivalent to **Set***InfoEd**(CurItemNo 322,...) But there is one important exception – when using tables 145, **SetCurrent***Info** methods modify item in the position of caret regardless its location (in the root editor or in cell inplace editor).

You cannot change a type of item (except for conversion pictures to hot-pictures 339 and vice versa 339)

First, you need to know a type of item. You can use RichView.GetItemStyle 229(ItemNo). Or you can use CurItemStyle 323 property of RichViewEdit for the item in the position of caret.

Next, you usually need to get information about the given item.

Finally, you can modify it.

Set***Info methods

The arguments of these methods are similar to the arguments of Add*** methods 77.

For the specified items you can use

- ·SetBreakInfo 260, SetBulletInfo 261, SetHotspotInfo 265 (call Invalidate to repaint RichView after them, reformatting is not necessary)
- ·SetPictureInfo 269, SetControlInfo 262 return Boolean value: "reformatting 217 is necessary?" (call Invalidate to repaint RichView after it, reformatting is not necessary if new picture or control has the same size).

You can also use:

- SetItemTag 267 method to change tags of items,
- SetCheckpointInfo 261 to change checkpoints of items,
- SetItemExtraIntProperty 266 to change values of additional integer properties,

- SetItemExtraStrProperty[266] to change values of additional integer properties,
- SetItemText[267] to change item text (visible text for text items, names for non-text items).

Please read the topic about "tags"[68] if you want to change tags of items.

The limitations of this version are:

- you can't change ImageList of *bullets*[128] and *hotspots*[129];
- you can't replace control with the different one;

It is not recommended to use these methods in editor, see Viewer vs Editor[102]

Set***InfoEd methods (editor)

The same set of methods as **Set***Info** methods, with the same arguments, but they are editor-style methods.

SetBreakInfoEd[371], SetBulletInfoEd[372], SetHotspotInfoEd[382], SetPictureInfoEd[386], SetControlInfoEd[374];

also SetItemTagEd[384], SetCheckpointInfoEd[373], SetItemTextEd[385], SetItemExtraIntPropertyEd[383], SetItemExtraStrPropertyEd[384]

SetCurrent***Info methods (editor)

The same as **Set***InfoEd** methods, but they work with the current (in position of caret) item.

SetCurrentBreakInfo[375], SetCurrentBulletInfo[375], SetCurrentHotspotInfo[377], SetCurrentPictureInfo[381], SetCurrentControlInfo[377],

also SetCurrentTag[382], SetCurrentCheckpointInfo[376], SetCurrentItemText[380], SetCurrentItemExtraIntProperty[378], SetCurrentItemExtraStrProperty[379].

Additional methods for controls

There are some special methods for resizing inserted controls in editor.

```
procedure ResizeControl[368](ItemNo, NewWidth, NewHeight: Integer);
procedure ResizeCurrentControl[368](NewWidth, NewHeight: Integer);
```

This is the preferred way to change size of the control. These methods reformat the affected part of document and repaint it.

If your control can sometimes change its size or move itself, you need to call

```
procedure AdjustControlPlacement[330](ItemNo: Integer);
procedure AdjustControlPlacement2[330](Control: TControl);
```

after such actions. These methods restore the correct position of the control and make all necessary reformatting and repainting if the control was resized.

1.2.3 Editing

1.2.3.1 RichViewEdit: Inserting Items in Position of Caret

These methods are useful for implementing "Insert" menu in your application. All these methods:

1. check ReadOnly[328] property (and do nothing in read-only mode);
2. delete the selected items;
3. insert item(s);
4. quickly reformat the changed part of the document;
5. repaint the editor;
6. generate OnChange[390] event, set Modified[327] to True.

The methods do nothing (and return True), if the selected text is modify/delete protected (or several cells of table are selected).

Inserting Text

These methods can insert several text items of the current text style (see CurTextStyleNo[324] property):

```
procedure InsertText[360](text, CaretBefore);
```

ANSI (InsertTextA) and Unicode (InsertTextW) versions of InsertText are available.

```
function InsertTextFromFile[361](FileName, CodePage):Boolean;
function InsertTextFromFileW[361](FileName):Boolean;
function InsertOEMTextFromFile[354](FileName):Boolean;
```

These methods insert one text item (string must not contain line break, page break and tab characters) with the specified tag:

```
function InsertStringTag[359](s, Tag): Boolean;
```

ANSI (InsertStringATag) and Unicode (InsertStringWTag) versions of InsertStringTag are available.

Inserting Non-Text Items

These methods insert one item of the specified type (parameters are similar to parameters of **Add***[77] methods[77]):

```
function InsertControl[352](Name, ctrl, VAlign): Boolean;
function InsertPicture[355](Name, gr, VAlign): Boolean;
function InsertHotPicture[352](Name, gr, VAlign): Boolean;
function InsertBreak[350](Width, Style, Color): Boolean;
function InsertBullet[351](ImageIndex, ImageList): Boolean;
function InsertHotspot[353](ImageIndex, HotImageIndex, ImageList): Boolean;
```

Inserting Files and Streams

These methods insert RVF[93] file or stream

```
function InsertRVFFromStreamEd[358](Stream):Boolean;
function InsertRVFFromFileEd[357](FileName):Boolean;
```

These methods insert RTF (Rich Text Format) file or stream

```
function InsertRTFFromStreamEd[357](Stream):Boolean;
function InsertRTFFromFileEd[356](FileName):Boolean;
```

These methods insert text of the current text style (see CurTextStyleNo[324] property):

```
function InsertTextFromFile[361](FileName):Boolean;
function InsertTextFromFileW[361](FileName):Boolean;
function InsertOEMTextFromFile[354](FileName):Boolean;
```

Inserting Tables and Other Items

The most general method for inserting items is

```
function InsertItem[353](Name, Item): Boolean;
```

It's recommended to use this method only for tables[130], labels[133], numbered sequences[134], endnotes[135], footnotes[136], references to parent footnotes and endnotes[140], page numbers[141], page counts[142], equations[143], and custom item types[144].

See also...

Methods for pasting from the Clipboard[82].

1.2.3.2 Undo and Redo

(Introduced in version 1.3)

All methods for document editing that introduced in editor (TRichViewEdit[316]) can be undone/redone (see editor-style methods).

All methods for document modification that introduced in TRichView[162] can't be undone/redone (see viewer-style methods). If you wish to modify document with these methods, you must clear undo buffer before.

All methods and properties described in this topic are methods and properties of TRichViewEdit.

```
property UndoLimit[329]: Integer
```

Property UndoLimit sets the capacity of undo buffer. It can be used to disable undo/redo feature.

Basic undo/redo

Undo/redo is available in editor even without any effort from your side.

User can undo the last operation by pressing Ctrl+Z or Alt+Backspace.

User can redo the last undone operation by pressing Shift+Ctrl+Z or Shift+Alt+Backspace.

Undo/redo buffers will be cleared automatically when you call Clear[210] method, or when you use methods for loading documents (which call Clear inside).

You can clear undo and redo buffer yourself using method ClearUndo[339].

Implementing menu/toolbar buttons for undo/redo

You can undo/redo operations from the application code, using methods Undo[387] and Redo[365].

You can change enabled/disabled state of menu items or buttons comparing the returned values of functions UndoAction[388] and RedoAction[366] with *rvutNone* (*rvutNone* – menu for undo/redo should be disabled; other values – enabled).

Call these methods inside OnChange[390] event.

You can modify caption/hint of menu item or button to display the name of action that will be undone/redone next. A value identifying this undo/redo action can be obtained using UndoAction[388]/RedoAction[366].

There are no predefined names for standard undo/redo actions. The Editor demo (**Demos\Delphi\Editors\Editor 1**) has the suggested names of actions in RVUndoStr.pas.

rvutCustom constant is a special case, see below. For this value, use UndoName[388]/RedoName[366] methods.

Grouping operations for undo

You can set the mode so that all subsequent operations will be considered by the editor as a whole, and they will be undone (and redone) in one step.

First, you need to give name to this undo action. Name can be standard (not a name really, but a type of undo action) or custom.

Use BeginUndoGroup[336] to give a standard name, and BeginUndoCustomGroup[336] to give a custom name.

Next, you need to call SetUndoGroupMode[387].

```
procedure TRichViewEdit.BeginUndoGroup(UndoType: TRVUndoType);
```

– begins a new undo action. It must be followed by SetUndoGroupMode(True) ... SetUndoGroupMode(False).

UndoType can be any value of type TRVUndoType[693], except for rvutNone and rvutCustom.
This value will be returned by UndoAction/RedoAction when undo/redo.

```
procedure TRichViewEdit.BeginUndoCustomGroup(const Name: TRVUnicodeString[694]);
```

– the same as BeginUndoGroup, but allows to set your own name for the undo action. The undo action will have type = *rvutCustom*.

Name of it will be returned by UndoName/RedoName.

```
procedure TRichViewEdit.SetUndoGroupMode(GroupUndo: Boolean);
```

– sets mode where all subsequent actions will be undone and redone as one. It *must* be preceded by either BeginUndoGroup or BeginUndoCustomGroup. Typing can not be grouped. Operations in different cell inplace editors cannot be grouped.

These methods must be called for the editor where these operations are performed (usually TopLevelEditor[329])

Example (inserting picture and resizing it 100x100):

```
MyRichViewEdit.TopLevelEditor.BeginUndoGroup(rvutInsert);
MyRichViewEdit.TopLevelEditor.SetUndoGroupMode(True);
try
  if MyRichViewEdit.InsertPicture[355](...) then begin
    MyRichViewEdit.SetCurrentItemExtraIntProperty[378](
      rvepImageHeight, 100, True);
    MyRichViewEdit.SetCurrentItemExtraIntProperty(
      rvepImageWidth, 100, True);
finally
  MyRichViewEdit.TopLevelEditor.SetUndoGroupMode(False);
end;
```

Each call of SetUndoGroupMode(True) should be followed by exactly one call of SetUndoGroupMode(False).

Calls can be nested.

See also...

Methods introduced in TRichView and undo: see Viewer vs Editor[102]

1.2.3.3 Drag and Drop

You can drag data from TRichView[162] or TRichViewEdit[316]. You can drop data to TRichViewEdit[316].

Drag&drop is possible not only inside the same editor, but between different controls and even between different applications.

For example, you can drop files from Windows Explorer, hyperlinks, text and images from Internet Explorer or Word.

Dragging from TRichView (or TRichViewEdit)

TRichView provides data in the following formats:

- RichView Format[93] ('RichView Format')
- RTF ('Rich Text Format')
- ANSI text (CF_TEXT)
- Unicode[99] text (CF_UNICODETEXT)
- bitmap (CF_BITMAP), if TBitmap picture[122] is selected
- metafile (CF_ENHMETAFILE), if TMetafile picture[122] is selected.

You can disallow dragging from the given TRichView by including *rvoDisallowDrag* in the Options[182] property.

TRichView cannot initialize dragging of inserted controls[126]. If you want to drag them, use BeginOleDrag[209] method.

Dragging to TRichViewEdit

AcceptDragDropFormat[321] lists formats that can be accepted.

By default all formats are accepted except for URL (because it requires processing in OnReadHyperlink[284])

Dropped object can contain data in multiple formats. The priority is: RVF, then RTF, then files, then URL, then Unicode text, then ANSI text, then bitmap, then metafile.

If *rvoDragDropPicturesFromLinks* is included in EditorOptions 325, and the dropped URL is a link to a picture, the editor attempts to insert it as a picture, not as a link.

If *rvoAssignImageFileNames* in Options 182, file name is is stored in *rvespImageFileName* extra string property 676 for pictures; you can modify this file name in OnAssignImageFileName 274 event.

The following events may occur:

- OnReadHyperlink 284;
- OnDropFile 392;
- OnDropFiles 393.

You can customize drag&drop process using the following low level events:

- OnOleDragEnter 395;
- OnOleDragOver 396;
- OnOleDragLeave 395;
- OnOleDrop 397,
- OnBeforeOleDrop and OnAferOleDrop 388

See also...

Clipboard 82

1.2.4 Graphics

1.2.4.1 Animated Images

Overview

TRichView can animate inserted images 122.

AnimationMode 169 property defines when animation starts. By default, AnimationMode 169=*rvaniManualStart*.

rvaniManualStart is used by default because *rvaniOnFormat* may cause problems in some existing applications (created before implementing this feature), because animation must be stopped when you call viewer-style methods like DeleteItems 214. If you simply load and edit documents, set this mode to *rvaniOnFormat*.

Methods:

- StartAnimation 272 starts all animations;
- StopAnimation 273 stops all animations;
- ResetAnimation 249 rewinds all animations to the first frame.

Supported Animations

1. TGifImage included in Delphi 2007, 2009 or newer

Include RVGifAnimate2007 unit in your project to enable this animation.

2. TGifImage by Anders Melander

Downloads:

- ***http://www.torry.net/vcl/graphics/gif/gifimage.exe*** (original);
- ***http://www.trichview.com/resources/thirdparty/gifimage.zip*** (update, necessary).

Include RVGifAnimate unit in your project to enable this animation.

3. RVJvGifAnimate for TJvGifImage from JEDI's JVCL

Dowloads: ***http://jvcl.sourceforge.net***.

4. TBitmap

Bitmap is sliced into animation frames (*rvepImageWidth* x *rvepImageHeight*) arranged in rows and columns. Animation is enabled if you set nonzero *rvepAnimationInterval* (animation delay in 1/100 of second), see TRVExtraItemProperty[673] type. If one of *rvepImageWidth* and *rvepImageHeight* is not defined (zero), frame width/height is assumed to be equal to the image width/height. If both of them are not defined, animation is not played.

1.2.4.2 Smooth Image Scaling

Scaling

TRichView can display scaled images using an advanced method producing smooth scaled images.

The following images are affected:

- TCustomRichView.BackgroundBitmap[169] (if BackgroundStyle[170]=*bsStretched*);
- pictures[122] and *hot-pictures*[125] (when stretched);
- table.BackgroundImage[567] (if BackgroundStyle[568]=*rvbsStretched*);
- table cells' BackgroundImage[605] (if BackgroundStyle[606]=*rvbsStretched*).

By default, only non-transparent bitmaps and JPEG images are scaled, and not greater to 3000 pixels in width or height. These settings can be changed using RVThumbnailMaker[713] object.

Smooth scaling is applied when displaying pictures on the screen. It is not used when printing or displaying in TRVReportHelper[540].

ScaleRichView note: in TSRichViewEdit, smooth scaling is applied to the same images, but they do not need to be stretched: they can be scaled when the whole page is scaled.

Caching

Smooth scaling is not a very fast operation, so resulting images are stored for reusing in future. Each TRichView component can store up to RichViewMaxThumbnailCount[703] scaled images (+ a scaled version TCustomRichView.BackgroundBitmap[169], which is always stored).

1.2.4.3 Semitransparent Objects

Documents may contain objects having transparent and semitransparent areas, such as:

- pictures[122] (for example, Gif, Png, icons, Bitmaps with alpha-channel)
- paragraph background (with specified Opacity[501])
- table[130] background (that may include transparent pictures[567] and semitransparent[577] color filling)
- table cell background (that may include transparent pictures[605] and semitransparent[609] color filling)
- sidenote[137] and text box[139] background (with specified Opacity), displayed only while printing in TRVPrint[513] and in ScaleRichView.

Drawing on the screen

When drawing on the screen, the component uses the appropriate methods for semitransparent drawing. Some special code is used to draw backgrounds of animations[90] and backgrounds of inplace editors of table cells.

Known problems: TPngObject does not support semitransparency when displayed rotated[611].

Printing in TRVPrint[513]

The most of printers do not support semitransparent drawing, so TRVPrint draws semitransparent objects in the following steps:

1. it creates a bitmap,
2. it draws a background below the transparent object in this bitmap,
3. it draws this object in this bitmap,
4. it prints this bitmap.

TRVPrint has a limited support of transparency in sidenotes and text boxes (when drawing on the step 2):

- you cannot see a sidenote or a text box through a transparent area
- through a transparent area in a sidenote or a text box, you can see only a page background (but not any content)

TRVPrint does not support textured backgrounds in rotated objects (a plain color is used instead).

Drawing and printing in TRVReportHelper[540]

When printing, TRVReportHelper uses the same method (using bitmaps to draw semitransparent objects) as TRVPrint.

When drawing on the screen, TRVReportHelper can draw transparently or can use bitmaps, depending on AllowTransparentDrawing[541] property.

Drawing and printing in ScaleRichView

In TSRichViewEdit, drawing on the screen is similar to drawing in TRichViewEdit: the component uses the appropriate methods for semitransparent drawing.

When printing, like TRVPrint, TSRichViewEdit uses a bitmap to draw semitransparent areas. However, the step 2 (drawing a background below the transparent object in a bitmap) is implemented differently. When drawing a background below the transparent object, TSRichViewEdit draws all page layers below the object. Because of this, you can see text boxes and sidenotes through the transparent areas, and you can see main document through transparent areas in text boxes and sidenotes.

TSRichViewEdit does not have a limitation on printing backgrounds of rotated objects.

Bitmaps created to draw semitransparent areas in ScaleRichView are created in resolution 96 dpi. Because of this, when printing on high resolution printers, semitransparent areas are printed more roughly than surrounding text.

1.2.5 Saving and Loading

1.2.5.1 Saving and Loading

RichView has the following methods for export:

- SaveHTML[250], SaveHTMLEx[251]: saving HTML files (also SaveHTMLToStream[252], SaveHTMLToStreamEx[252])
- SaveText[255]: saving plain ANSI text in file; SaveTextW[255]: saving plain Unicode text; SaveTextToStream, SaveTextToStreamW[255] – the same for streams (can save the whole document or only selection);
- SaveRTF[253],SaveRTFToStream[254] – exporting the whole document or the selected part to file or stream as RTF (Rich Text Format);
- SaveDocX[249],SaveDocXToStream[250] – exporting the whole document or the selected part to file or stream as DocX (Microsoft Word format);
- SaveRVF[254],SaveRVFToStream[254] – saving the whole document or the selected part to RVF (RichView Format[93]) file or stream;

RichView has the following methods for import:

- LoadText[246]: reading plain ANSI text; LoadTextW[246]: reading plain Unicode text;
- LoadTextFromStream, LoadTexFromStreamtW[247] - the same for streams instead of files;
- LoadRTF[243], LoadRTFFromStream[244]: reading RTF (Rich Text Format) file or stream.
- LoadRVF[245], LoadRVFFromStream[245]: reading RVF (RichView Format[93]) file or stream;

See also: InsertRVFFromStream[240], AppendRVFFromStream[208].

Note: All methods for reading contents, introduced in RichView, are viewer-style methods.

RichViewEdit has the following additional methods for import (insertion):

- InsertRVFFromFileEd[357], InsertRVFFromStreamEd[358] – like InsertRVFFromFile/InsertRVFFromStream, but editor-style methods.
- InsertRTFFromFileEd[356], InsertRTFFromStreamEd[357] – inserting RTF (Rich Text Format) file or stream.
- InsertTextFromFile[361] – editor-style method;
  InsertOEMTextFromFile[354] – the same method for files in DOS code page;
  InsertTextFromFileW[361] – the same method for Unicode files.

There are also method for pasting data from the Clipboard[82].

Events on export

- HTML: OnHTMLSaveImage[277], OnSaveImage2[296], OnSaveComponentToFile[292], OnSaveHTMLExtra[295], OnSaveItemToFile[297], OnSaveParaToHTML[298], OnWriteHyperlink[302].
- RTF: OnSaveComponentToFile[292], OnSaveRTFExtra[299], OnSaveItemToFile[297], OnWriteHyperlink[302].
- DocX: OnSaveComponentToFile[292], OnSaveDocXExtra[293], OnSaveItemToFile[297], OnWriteHyperlink[302].
- Text: OnSaveComponentToFile[292], OnSaveItemToFile[297]

Events on import

- RTF: OnImportPicture[279], OnReadHyperlink[284], OnStyleTemplatesChange[302].

Common events

- OnProgress[283]

See also...

Saving HTML[96]
Reading/writing RVF[93]
Export of tables[157]
Unicode in RichView[99]

1.2.5.2 RVF (RichView Format)

What is RVF?

RVF (RichView Format) is a file format specially designed for saving/loading TRichView documents. It is quite simple and compact format. It can be text file or binary file (defined by RichView.RVFOptions[188]).

This format is used:

- in TDBRichView[407], TDBRichViewEdit[410] to save their contents to binary fields;
- RVF can be copied to the Clipboard; editor can paste RVF;
- data can be dragged&dropped[89] as RVF;
- you can load/save RVF with the methods: LoadRVFFromStream[245], InsertRVFFromStream[240], AppendRVFFromStream[208], LoadRVF[245], SaveRVFToStream[254], SaveRVF[254].

All TRichVew document features can be saved in RVF. Collections of text, paragraph, list styles,style templates, margins and background can be saved optionally (see below).

The main things you need to know about RVF

- main settings affecting RVF saving and loading can be changed in the TRichView component editor[168];
- in order to load *bullets*[128], *hotspots*[129] and list levels[506] with imagelist from RVF you need to process OnRVFImageListNeeded[288] event;
- you can save pictures[122] and controls[126] in binary mode or text mode (binary mode is much more efficient and compact);
- you can save pictures and controls in RVF, or create them on request when loading;

- if you load controls in RVF you need to register their classes with RegisterClasses procedure (for example: RegisterClasses([TButton]));
- the same for pictures; however, if picture class is not registered, TRichView will try to find the proper graphic class basing on content;
- formatting of text[437] and paragraph[431] (i.e. styles) is saved in RVF documents optionally, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions[188];
- style templates are saved in RVF, if *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included in RVFOptions[188], and UseStyleTemplates[195]=*True*.

RVF and Image Lists

TRichView can save and load all document content itself, but there is an exception: *bullets*[128], *hotspots*[129] and list markers having imagelist picture in list levels[506].

When loading, RichView needs to know which ImageList to use for the given *bullet* or *hotspot*. It asks the application about it by calling OnRVFImageListNeeded[288] event. You need to process this event if you want to load *bullets* and *hotspots*.

When saving, RichView stores Tag properties of ImageList with *bullet*[128] or *hotspot*[129]. When reading, RichView calls OnRVFImageListNeeded event and pass this value to it.

Example

If you have two ImageLists (ImageList1 and ImageList2) that you use for *bullets* in MyRichView, you can:

- set ImageList1.Tag to 1, set ImageList2.Tag to 2,
- process MyRichView.OnImageListNeeded event::

```
procedure TMyForm.MyRichViewRVFImageListNeeded(
  Sender: TCustomRichView; ImageListTag: Integer;
  var il: TCustomImageList);
begin
  case ImageListTag of
    1:
      il := ImageList1;
    2:
      il := ImageList2;
  end;
end;
```

DO NOT CONFUSE IMAGELIST.TAG WITH RICHVIEW ITEMS' TAGS[68]. IN THIS CASE RICHVIEW USES TIMAGELIST.TAG PROPERTY.

RichView does not own any ImageList, it never destroys them. It just holds references to them.

If you want to load pictures and/or controls from RVF you need to register them with RegisterClasses procedure (you need not to register TBitmap, TMetafile, TIcon and TJpegImage - they are already registered by RichView)

RVF and Pictures & Controls

You can save full information about pictures and inserted controls in RVF, but you can also use advanced feature: you can save pictures/controls yourself and include in RVF only some "identifiers" that can be used to create pictures/controls when needed (on loading). Two values can be uses as these "identifiers": RichView items tags and names.

These options do not affect loading from RVF. RichView autodetects these modes (binary/text, indices/names of styles).

When loading, you can use OnRVFPictureNeeded[289] and OnRVFPictureNeeded[289] events.

Example

```
procedure TMyForm.MyRichViewRVFControlNeeded(
  Sender: TCustomRichView; Name: TRVUnicodeString[694];
```

```
  const Tag: TRVTag[692]; var ctrl: TControl);
begin
  case StrToInt(Tag) of
    1:
      begin
        ctrl := TButton.Create(nil);
        TButton(ctrl).Caption := Name;
      end;
    2:
      begin
        ctrl := TEdit.Create(nil);
        TEdit(ctrl).Text := Name;
      end;
  end;
end;
```

As you can see, TRichView passes two parameters that can help you to identify your control:

- Tag: TRVTag[692] – TRichView item tag (do not confuse with tag properties of TComponent descendants);
- Name: TRVUnicodeString[694] – string that can be associated with any control or picture in RichView.

Example

```
// this example assumes that images may be only bitmaps
procedure MyRichViewRVFPictureNeeded(Sender: TCustomRichView[162];
    const ItemName: TRVUnicodeString[694]; Item: TRVNonTextItemInfo[563];
    Index1, Index2: Integer; var gr: TGraphic);
begin
  gr := TBitmap.Create;
  if (Index1>=0) and (Index2>=0) then
    gr.LoadFromFile((Item as TRVTableItemInfo[565]).Cells[574][Index1, Index2].Backg
  else if Item is TRVTableItemInfo then
    gr.LoadFromFile(TRVTableItemInfo(Item).BackgroundImageFileName[568])
  else if Item is TRVGraphicItemInfo then
    gr.LoadFromFile(TRVGraphicItemInfo[563](Item).ImageFileName[676]);
end;
```

Pictures may be stored directly in RVF. RVF contains names of graphic classes used for these images. If you want to use the same graphic classes for loaded pictures, you need to register these classes using RegisterClasses procedure. TRichView itself registers TBitmap, TMetafile, TIcon, TJPEGImage, TWicImage (for Delphi 2010+), TPngImage (for Delphi 2009+). If you include RVGifAnimate2007 unit in your project, TGifImage will be registered as well (for Delphi 2007+). All other graphic classes must be registered explicitly. If the graphic class specified in RVF is not available when reading, TRichView will try to detect an appropriate class by the graphic content.

If you include *rvfoIgnoreGraphicClasses* in RichView.RVFOptions[188], TRichView ignores names of graphic classes stored in RVF and always tries to find a proper graphic class by graphic content.

Options for Saving and Loading. Warnings

RichView.RVFOptions[188] define options for RVF saving and loading.

After loading you can check RichView.RVFWarnings[191] property. Depending on RichView.RVFOptions some flags in RichView.RVFWarnings can be errors or warning:

Units of measurement

RVF file may be written with different units of measurement (pixels or twips[106]) than specified in Style[418].Units[437] of TRichView which loads this file.

When inserting RVF, lengths from RVF are converted to Style[418].Units[437].

When loading RVF, the control works according to *rvfoCanChangeUnits* option. If it is excluded from RVFOptions[188], lengths from RVF are converted to Style[418].Units[437]. If it is included, and RVF contains either text or paragraph styles, existing styles are converted to units read from RVF (using Style[418].ConvertToDifferentUnits[438] method), so lengths from RVF are read without conversion.

If a conversion of units is performed while reading RVF, *rvfwConvUnits* is included in RVFWarnings[191].

See also...

RVF Specification[109]

RichView properties:

- Options[182],
- RVFOptions[188],
- RVFWarnings[191];
- UseStyleTemplates[195];
- StyleTemplateInsertMode[193].

RichView events:

- OnRVFImageListNeeded[288],
- OnVFControlNeeded[288],
- OnRVFPictureNeeded[289],
- OnControlAction[275] (ControlAction=*rvcaAfterRVFLoad*),
- OnItemAction[280] (ItemAction=*rviaInserting* and *rviaInserted*);
- OnStyleTemplatesChange[302].

RichView methods:

- LoadRVFFromStream[245], InsertRVFFromStream[240], AppendRVFFromStream[208], LoadRVF[245],
- SaveRVFToStream[254], SaveRVF[254],
- Copy[212], CopyDef[212], CopyRVF[213].

RichViewEdit methods:

- InsertRVFFromStreamEd[358], InsertRVFFromFileEd[357],
- Paste[361], PasteRVF[364], CanPasteRVF[337], CutDef[340].

1.2.5.3 Export to HTML

Overview

TRichView has methods for saving HTML files. There are no methods for reading HTML files in current version (use free add-ons: ***TrvHtmlImporter and TrvHtmlViewImporter***).

TRichView saves HTML file and a set of image files.

By default, images are saved as Jpegs (*.jpg).

If you do not want to use TJpegImage class, you can define RVDONOTUSEJPEGIMAGE conditional define when compiling, and images will be saved as bitmaps.

You can specify graphic formats that must not be converted to Jpegs using RVGraphicHandler[710].RegisterHTMLGraphicFormat. TRichView itself define PNG (TPngImage, Delphi 2009+) and GIF (TGifImage, Delphi 2007+) as HTML graphic formats.

Besides, you can save images yourself in custom formats using OnHTMLSaveImage[277] or OnSaveImage2[296] event (see demo in **Demos\Delphi\Assorted\Save HTML**)

There are two ways to save HTML in RichView:

- saving using <font>, <b>, <i> and similar html tags,
- saving using CSS (Cascading Style Sheets).

HTML files which were saved with the first way can be read by larger number of programs (including TrvHtmlImporter).

HTML files with CSS retain much more formatting properties of TRichView document (they look almost exactly like TRichView content when using a capable browser) and have much more elegant and compact HTML code. TRVHtmlImporter does not support CSS generally, but it can read HTML with CSS files saved by TRichView.

Saving Using <font> Tags

```
function RichView.SaveHTML[250](const FileName,Title,
  ImagesPrefix: TRVUnicodeString[694]; Options: TRVSaveOptions[686]):Boolean;
```

FileName – file name.

Title – title of document (tag <title>).

Return value – "saving was successful?"

ImagesPrefix – first part of images file names. For example, if **ImagesPrefix**='img' then images will be saved as **img1.jpg**, **img2.jpg** and so on (or **img1.bmp**, **img2.bmp** and so on, see above).

If *rvsoOverrideImages* in **Options** then all files with the same names will be overridden. Otherwise, the component will make unique file names for images.

Saved jpegs or bitmaps are not transparent. When converting to jpeg or bitmap, transparent color of icons or metafiles converted to the background color of RichView.

You can change saving of images using OnHTMLSaveImage[277] or OnSaveImage2[296] events.

Checkpoints[65] are saved as where # is an index of *checkpoint*. If *rvsoUseCheckpointsNames* is in **Options**, *checkpoint* name is used instead.

RichView uses Charset property of 0th text style to save language of HTML file (for Delphi 3+). There is also an option to save Unicode (UTF-8) HTML files.

When saving with **SaveHTML** method, many formatting properties are lost (such as text and paragraph background colors, borders, hover color, all margins/indents, font sizes are converted to 7 available HTML font sizes). Centered and stretched background bitmaps become tiled.

Saving Using CSS

```
function RichView.SaveHTMLEx[251](const FileName, Title, ImagesPrefix,
  ExtraStyles, ExternalCSS, CPPrefix: TRVUnicodeString[694];
  Options: TRVSaveOptions[686]):Boolean;
```

This method is not an extension of SaveHTML method – this is a completely different method. It saves HTML using style sheets (CSS). CSS can be saved inside HTML file or this method can place only a reference to external CSS file into this HTML file

External CSS file can be saved using SaveCSS[441] method of RichView.Style:

You can save your style sheet one time and use it for all saved HTML files (if they all use the same collection of styles). This method allows you to reduce size when saving several HTML files with the same text and paragraph styles.

Font families are stored in CSS for the most frequently used fonts. For example, if font name is 'Arial', the string *'Arial', 'Helvetica', sans-serif* will be saved.

ExtraStyles is a string that can contain additional styles for your own use. RichView will not use these styles, but you can use them when saving, for example, controls and URLs.

CPPrefix is a first part of name used when saving *checkpoint*[65]. If **CPPrefix**='' then RichView uses standard name ('RichViewCheckPoint'). For example, if **CPPrefix**='label' then checkpoints will be saved as where # is an index of *checkpoint*. If *rvsoUseCheckpointsNames* is in **Options**, *checkpoint* name is used instead.

Usually you do not need to use advanced parameters and can write:

```
MyRichView.SaveHTMLEx(MyFileName, MyTitle, MyImagesPrefix,
  '', '', '', []);
```

Saved HTML looks exactly like TRichView with the exceptions:

- if VShift[477] property of text style >0 then this style saved as superscript, <0 as subscript;
- only "normal" (not "hot") *hotspots*[129]' pictures are saved;
- border styles converted to available set for HTML;
- negative BorderOffsets can be saved, but browsers understand them as zero offsets;
- paragraph background offsets and border offsets can't be saved separately; maximal values of these offsets are saved.

Common for the Both Methods

By default, items tags[68] are saved as targets of hypertext links. You can customize saving using the event OnWriteHyperlink[302].

By default, inserted controls are not saved. However, you can save them in the event OnSaveComponentToFile[292].

You can save additional information in HTML in the events: OnSaveHTMLExtra[295], OnSaveParaToHTML[298].

You can insert HTML codes[472] in text.

You can completely change saving of items of any type in HTML using OnSaveItemToFile[297] event.

There is a set of methods for saving HTML files to stream: TRichView.SaveHTMLToStream[252], TRichView.SaveHTMLToStreamEx[252], TVStyle.SaveCSSToStream[442].

There is also a possibility to save only part of HTML document (opening part, middle part, closing part) – useful if you use HTML output of RichView only as a part of HTML document.

Symbol Fonts

SaveHTMLEx[251] saves all text of "Symbol" and "Wingdings" fonts as a set of HTML character entities or Unicode codes. For example, the text 'a+b=g' of "Symbol" font (α+β=γ) is saved as 'α+β=γ'

Pros:

- correct view modern browsers (not only in IE);
- correct view on non-Windows platforms

Cons:

- characters may look slightly different;
- some exotic characters are lost.

Headers and footers

If a header[264] and/or a footer[263] are defined, they can be saved before and after the main document in HTML. To do it, include *rvsoHeaderFooter* in the Options parameter of HTML saving methods.

The component uses either headers/footers for the first page or normal headers/footers, depending in TRichView.DocParameters[176].TitlePage[314].

Known problems:

Numeration of checkpoints starts from 0 in each table cell. So, it's recommended to use checkpoint names as identifiers (add *rvsoUseCheckpointsNames* in **Options**).

See also...

- TRichView methods for saving and loading[92];
- Export of tables[157];
- Typed constants related to HTML export[698];
- Example how to save MIME-encoded files (*.eml or *.mht), how to send HTML e-mail using TNMSMTP or TIdSMTP components: ***http://www.trichview.com/forums/viewtopic.php?t=11***.

1.2.6 Internationalization

1.2.6.1 Unicode in TRichView

Introduction

Unicode is a worldwide character-encoding standard. Unicode simplifies localization of software and improves multilingual text processing. By implementing it in an application, a developer can enable the application with universal data exchange capabilities for global marketing, using a single binary file for every possible character code..

For Delphi 2009 or newer, Unicode is a default encoding for strings.

Unicode and ANSI Text in TRichView

All strings in TRichView are Unicode strings.

Import and Export

Text Files

LoadText[246], LoadTextFromStream[247] load ANSI text files. A code page for conversion to Unicode is specified in the optional parameter.

LoadTextW[246], LoadTextFromStreamW[247] load Unicode text files.

Note: you can test file with the function

```
function RV_TestFileUnicode(const FileName: TRVUnicodeString[694]): TRVUnicodeTestResult
```

defined in RVUni.pas.

Return values

- *rvutNo* – the file is not Unicode (odd size);
- *rvutYes* – the file is most likely Unicode (even size, Unicode byte-order characters at the start or #0 in text (first 500 bytes checked));
- *rvutProbably* – the file can contain Unicode (even size);
- *rvutEmpty* – the file is empty;
- *rvutError* – error opening the file.

You can also use WinAPI function **IsTextUnicode** performing more advanced tests.

SaveText[255] saves ANSI text file. Unicode strings are converted basing on Style.DefCodePage[421] property.

SaveTextW[255] saves Unicode text file. ANSI strings are converted basing on the corresponding Charsets.

RTF (Rich Text Format) and DocX files

RTF and DocX files can contain Unicode text.

HTML

SaveHTML***[96] can save ANSI or Unicode (UTF-8) HTML files. In ANSI HTML files, Unicode characters are written as codes (&#NNNN;), so all Unicode characters are preserved, but file size is increased; so it's highly recommended to save HTML in UTF-8 encoding.

Selection, Search and The Clipboard

GetSelTextA[238] returns selection as an ANSI string. Unicode text is converted basing on Style.DefCodePage[421] property.

GetSelTextW[238] returns selection as a Unicode string.

Text searching methods have versions allowing to search for ANSI and for Unicode string: TRichView.SearchTextA/SearchTextW[256]; however, SearchTextA simply converts the string to Unicode (using Style.DefCodePage[421]) and calls SearchTextW.

CopyTextA[213] copies selection as ANSI text. Unicode strings are converted basing on Style.DefCodePage[421] property.

CopyTextW[213] copies selection as Unicode.

Copy[212] and CopyDef[212] are copy Unicode (Option[182]-rvoAutoCopyUnicodeText)

Editing Operations

If pasting text using Paste[361] method, and text is available in Clipboard, the method pastes Unicode text.

PasteTextA[364] pastes ANSI text, PasteTextW[364] pastes Unicode text.

InsertTextFromFile[361]: the file must be ANSI (converted, if needed)

InsertOEMTextFromFile[354]: the file must be OEM (converted, if needed)

InsertTextFromFileW[361]: the file must be Unicode (converted, if needed)

InsertText[360], InsertStringTag[359] add Unicode string in Delphi/C++Builder 2009+ and ANSI string in older versions of Delphi/C++Builder.

InsertTextA[360], InsertStringATag[359] add ANSI string (converted, if needed)

InsertTextW[360], InsertStringWTag[359] add Unicode string (converted, if needed)

RVF (RichView Format[93])

Applications compiled with older versions of RichView (version less than 1.2) will not be able to load RVF files with Unicode.

RVF files will be loaded correctly even if Unicode flags in text styles are mismatched (saved with different RVStyle then loaded), conversions will be performed if required (for example, this conversion will occur when loading old RVF files in applications compiled in Delphi/C++Builder 2009+). There are two RVF Warnings[191]: *rvfwConvToUnicode* and *rvfwConvFromUnicode*, which indicate if any conversion took place.

TRichView v11 introduces a new change in RVF files allowing to store String properties as Unicode. RVF files saved in Delphi/C++Builder 2009+ are saved as *RVF version 1.3.1*, RVF files saved in the older versions of Delphi/C++Builder are saved as *RVF version 1.3*.

See also...

- Example how to load UTF-8 files[316].

1.2.6.2 Bidirectional text

Bidirectional mode specifies the reading order of text, order of items in lines, and placement of a vertical scrollbar.

The support for bidirectional text (Arabic or Hebrew) is turned off by default. In order to turn it on, set BiDiMode[170] property of TRichView either to *rvbdLeftToRight* or to *rvbdRightToLeft*.

BiDiMode of document can be overridden by BiDiMode of paragraph[484], BiDiMode of paragraph can be overridden by BiDiMode of text item[467].

Mixed right-to-left and left-to-right text is supported, even inside the same text item.

Attaching of adjacent characters (common for Arabic languages) is supported too, but only inside the same item (not between items).

Bullets and numbering of paragraphs take BiDiMode into account: for RTL (right-to-left) paragraphs, list markers are positioned to the right.

Tab stops support BiDiMode too: they are inverted (including tab alignments) for RTL paragraphs.

Differences in Windows API and Uniscribe

The results depends on the value of TextEngine[436]:

- if Windows API is selected, in *rvbdUnspecified* mode the results are completely incorrect for RTL text (wrong selection drawing and caret positions); the results are unreliable in other modes (may depend on the chosen font);
- if Uniscribe is selected, the results are correct is all modes; in *rvbdUnspecified* mode, only rearrangement of items on lines is turned off.

Limitations

- BiDiMode works in Windows versions supporting right-to-left languages (for example, on Win2000 and newer with RTL support installed, see Control Panel, Regional and Language Options, tab Languages, Install files for complex scripts and right-to-left languages). In all other versions of Windows, right-to-left text will be displayed left-to-right.
- (if Windows API is used) BiDiMode mode for Unicode text works only under WinNT-based systems (WinNT 4, 2000, XP, 7, 8 and so on)

Demo project

Demos\Delphi\Assorted\International\RTL

Demos\CBuilder\Assorted\International\RTL

1.2.7 Spelling Check

1.2.7.1 Live spelling check

Live spelling check marks misspelled words with special underlines (wavy line of TRVStyle.LiveSpellingColor[430] color).

Spelling check is executed in background thread.

Starting and finishing spelling check

In TRichView[162]:

Spelling check is started by calling StartLiveSpelling[273] method.

In TRichViewEdit[316]:

Spelling check is started:

- by calling StartLiveSpelling[273] method or
- when user modifies document or when any editing-style method is called (if LiveSpellingMode[327] =rvlspOnChange)

Spelling check is stopped by ClearLiveSpellingResults[210] method. Clear[210] also stops checking.

Checking Words

TRichView does not have its own spelling check engine and dictionaries. You need to use third-party components.

Currently the following components are supported:

1. ***Addict 3 and 4*** by Addictive Software (shareware Delphi component)
2. ***ExpressSpellChecker*** by Developer Express Inc. (shareware Delphi component)
3. ***HunSpell*** (C++ library under GPL/LGPL/MPL tri-license)
4. ***ASpell*** by Kevin Atkinson (DLL, license: LGPL)

5. ***Polar SpellChecker Component by*** by Polar (shareware ActiveX)

The following checkers were supported in older versions, discontinued (but can be restored on request):

6. EDSSpell 7.0 by ***OneDomain, Inc*** (shareware Delphi component, discontinued)
7. VSSpell 8.0 by ***ComponentOne LLC*** (shareware ActiveX, discontinued)
8. VSSpell 3.0 by Visual Components, Inc (shipped with older versions of Delphi, ActiveX)
9. ***LS Speller*** by Luzius Schneider (ISpell + MS Office (CSAPI) spell checkers, freeware)

Let me know if you are interested in supporting other spelling checkers.

The live spelling thread calls OnSpellingCheck[301] event for each word in the document.

Word is defined as a run of characters between delimiters[174]. Additionally, '&' is treated as a delimiter. Apostrophes are processed specially, see RichViewApostropheInWord[700] global variable.

Single quote character is processed specially: if it's inside word, it's treated as a part of word (for example "it's" or "I'll"). If it's to the left/right, it's treated as a delimiter.

URLs are ignored, see RVIsCustomURL[710] function.

Making Corrections

Call LiveSpellingValidateWord[242] if user added word in a custom dictionary or ignore list.

Call GetCurrentMisspelling[348] to get or to correct the current misspelled word. For example, you can use this method to implement popup menu:

1. when context menu is about to be shown, use GetCurrentMisspelling[348] to get misspelled word in the caret position and to generate suggestions;
2. when the user has chosen a suggestion, use GetCurrentMisspelling[348] again to select the misspelled word and use InsertText[360] to insert replacement.

Alternatively, misspelled word can be selected on the step 1 (before displaying popup menu).

Spelling Check and Document Editing

When calling a editing-style method, you do not need to worry about live spelling, everything is processed automatically (the method pauses the live spelling thread, makes changes, then resumes the live spelling thread from the proper position).

However, when calling any viewer-style method for modifying a document, you must be sure that live spelling is not running. If you call Clear[210] and then create a new document[77], everything is OK, because Clear stops live spelling check. But if you make changes in existing document, call ClearLiveSpellingResults[210] before these changes.

Memory Usage

Live spelling support adds 8 extra bytes per each item. If you do not use it, you can compile your application with RVDONOTUSELIVESPELL compiler define (see RV_Defs.inc file)

1.2.8 Important Technical Information

1.2.8.1 Viewer vs Editor

Methods of TRichView

TRichView[162] has the following 3 groups of methods for modifying documents:

1. appending items to the end of the document (may be from file or stream);
2. modifying item with the given index;

3. loading files and streams.

Examples of the first group are: AddNL 203, AddPicture 204, LoadRTF 243.

Examples of the second group are SetPictureInfo 269, SetItemText 267.

Examples of the third group are: LoadRVF 245, LoadRTFFromStream 244.

All these methods have the following characteristics:

- they do not require that the document is formatted (formatting is a preparing document for displaying)
- even if the document was formatted before calling such method, it is not formatted after.

You need to call Format 217 (or FormatTail 218) to display documents created or modified with such methods.

Table cells 605 also have such methods.

These methods do not pause a live spelling thread, and it may lead to errors. Before calling these methods, make sure that live spelling is not running (by calling Clear 210 or ClearLiveSpellingResults 210).

Methods of TRichViewEdit

TRichViewEdit 316 has the following groups of methods for modifying documents:

1. all methods of TRichView;
2. methods for inserting items in the position of caret (**Insert***** methods, such as InsertText 360, InsertPicture 355);
3. methods for modifying the item at the position of caret (**SetCurrent***** methods, such as SetCurrentPictureInfo 381, SetCurrentItemText 380);
4. extended set of methods for modifying item with the given index (**Set***Ed methods, such as SetPictureInfoEd 386, SetItemTextEd 385).

Methods of the group 1 are referred as viewer-style methods, methods of the groups 2-4 are editor-style methods.

All these methods (except for the methods of the group 1, inherited from TRichView) have the following characteristics:

- document must be formatted before calling such methods;
- these methods automatically reformat the affected area of documents;
- if the document is read-only 328 they do nothing; they also respect protection of text 473 and paragraphs 487;
- they call OnChange 390 event;
- they can be safely called while a live spelling is running;
- modifications can be undone and redone 88.

In an editor, the main application of the methods of the group 1 is generating new documents from scratch (after calling Clear 210).

The main application of other methods is performing user commands. They should not be used for generation of new document (because each such method updates documents, it can cause undesirable flicker, and they are much slower).

Important: You must not mix methods of group 1 and methods of groups 2-4! Methods inherited from TRichView do not update undo and redo buffers, and results of undo and redo commands will be unpredictable!

The only applications of methods of the group 1 is generating a new document, in the sequence:

1. Clear 210; (this method also clears undo and redo buffers);
2. One or several calls of methods of the group 1;
3. Format 217 (after the document has been formatted, you can apply methods of the groups 2-4)

See also: Undo in tables 155

1.2.8.2 Controls - Documents - Items

There are three main families of classes.

1. VCL controls, descendants of TCustomRichView.

See the diagram of class hierarchy 28.

The main visual controls are: TRichView 162, TRichViewEdit 316, TDBRichView 407, TDBRichViewEdit 410.

2. Documents, objects containing items, descendants of TCustomRVFormattedData 650.

Documents are objects, but not components.

Each RichView or RichViewEdit has such document accessible as RVData 187 property.

They have the most of methods of RichView controls. For example, calling

```
MyRichView.AddNL('text',0,0)
```

is equivalent to

```
MyRichView.RVData.AddNL('text',0,0)
```

Tables 565 have cells 605. Cells are also documents containing items, cells are descendants of TCustomRVFormattedData.

3. Items of documents, descendants of TCustomRVItemInfo 563.

Items are objects, but not components.

Items are usually hidden from programmers. They can be added and modified by methods of controls or document objects. The only exception is tables (TRVTableItemInfo 565).

Documents contain items, some items (tables) can contain documents (cells).

There is an important case: cell editing. When editing cell, RichViewEdit creates a special editor (also TRichViewEdit) on the top of the cell. While editing, all items in the cell are moved in this editor. When editing is finished, items are moved back to the cell.

At any time, object containing cell items is accessible as cell.GetRVData 613 (if cell is not edited, it is the cell itself; if edited, it is RVData 187 of its inplace editor).

Example 1

This example shows that the same procedure can be applied to the main document and table cells.

This example converts all text to upper case

```
procedure AllUpperCase(RVData: TCustomRVData 650);
var i,r,c: Integer;
    s: String;
    table: TRVTableItemInfo 565;
begin
for i := 0 to RVData.ItemCount 180-1 do
  if RVData.GetItemStyle 229(i)>=0 then begin // this is a text item
    s := RVData.GetItemText 230(i);
    s := AnsiUpperCase(s);
    RVData.SetItemText 267(i,s);
    end
  else if RVData.GetItemStyle 229(i)=rvsTable 711 then begin
    table := TRVTableItemInfo 565(RVData.GetItem 225(i));
    for r := 0 to table.RowCount 580-1 do
      for c := 0 to table.ColCount 575-1 do
        if table.Cells 574[r,c]<>nil then
          AllUpperCase(table.Cells 574[r,c].GetRVData 613);
  end;
```

```
end;
```

Call:

```
AllUpperCase(MyRichView.RVData);
MyRichView.Format[217];
```

Example 2

The same task, using different methods

```
// This procedure will be called for each RichView item
procedure TForm1.EnumItemsProc(RVData: TCustomRVData[650];
  ItemNo: Integer;  var UserData1: Integer;
  const UserData2: TRVUnicodeString[694]; var ContinueEnum: Boolean);
var s: String;
begin
  if RVData.GetItemStyle[229](ItemNo)>=0 then begin
    s := RVData.GetItemText[230](i);
    s := AnsiUpperCase(s);
    RVData.SetItemText[267](ItemNo,s);
  end;
  ContinueEnum := True;
end;
```

Call:

```
var v: Integer;

v := 0;
// RVData.EnumItems calls the specified procedure (EnumItems)
// for each items in RVData and its subdocuments (table cells)
// The second and the third parameters are user defined data,
// they will be passed in the procedure as UserData1 and UserData2
MyRichView.RVData[187].EnumItems(EnumItemsProc, v, '');
MyRichView.Format[217];
```

The advantage of using RVData.EnumItems is more clear and compact code.

The disadvantage: you cannot use it if you need to add or delete items.

1.2.8.3 Valid Documents

There are some limitations on document structure.

Internally, each document item[118] has properties:

- index of paragraph style
- "this item starts a paragraph" flag
- "this item starts a line inside a paragraph" flag

Not all combinations are acceptable.

All operations in editor[316] always create a valid document (providing that it was valid before the operation).

But some operations in viewer[162] can create invalid documents (especially DeleteItems[214])

Rules of Valid Documents:

1. All items in paragraphs must have the same index of paragraph style (in most cases, TRichView provides this. It's possible to create invalid documents by incorrect using of DeleteItems[214] and SetAddParagraphMode[259])
2. The first item (in document or table cell) must start a paragraph.

3. Full-line items (*breaks* 125 and tables 130) must start a new paragraph.
4. Items after full-line items must start a new paragraph.
5. Table cells must contain at least one item (by default they contain one empty text item).
6. List markers 131 must start a new paragraph.
7. Paragraph after marker must contain at least one item (may be an empty text item). Documents where next item after marker starts a new paragraph (or a line) are incorrect
8. Empty text items can be only in empty paragraphs (the only item in paragraph, or the only item after list marker). The exception is empty text items formatted with style having EmptyWidth 469>0, they can be in any place of text.

RichViewActions includes RVNormalize unit containing NormalizeRichView procedure. This procedure can fix most of document problems.

1.2.8.4 Units of Measurement

Glossary

Pica (pc) – typographic unit of measure corresponding to 1/6 of inch. The pica contains 12 points.

Point (pt) – typographic unit of measure corresponding to 1/72 of inch. This is a measuring unit for font size 475.

Twip (tw, "twentieth of a point") is defined as 1/20 of point. It equals to 1/1440 of inch.

Pixel (px) – a single point in a raster image or device. To convert inch (or other absolute unit of measurement) to pixels, a resolution (referred as "pixels per inch" or "dots per inch") of this device must be known. For some devices (like printers), this value is precise. For other devices (like computer displays), this value is logical and can be changed by the user. For screens, a typical "pixels per inch" value is 96 (accessible as Screen.PixelsPerInch).

In TRichView, properties can be measured in **"screen pixels"**. It means that on the screen, this length includes the specified number of pixels; but when printing, it is recalculated according to the printer resolution as:

value * (printer resolution) / (screen resolution)

Screen resolution can be read from Screen.PixelsPerInch, or it can be overridden in RichViewPixelsPerInch 703 global variable.

Properties measured in points

Font size (Size 475 property of text styles 437)

Properties measured in screen pixels

The following properties of TCustomRichView 162 are measured in screen pixels:

- LeftMargin 180, RightMargin 185, TopMargin 194, BottomMargin 171 – margins;
- MaxTextWidth 181 – maximal width for word wrapping;
- MinTextWidth 181 – minimal width for word wrapping (minimal width of scrollable area, not including margins).

Properties measured in TRVUnits 694

Properties of the type TRVLength 682 are measured in TRVUnits 694. They are values of a floating point type (Extended) that can represent lengths measured in inches, centimeters, millimeters, picas, points, or screen pixels.

In TRichView, this type is used for:

- properties of TRichView 162.DocParameters 176; they are measured in TRichView 162.DocParameters 176.Units 314.
- properties of TRVPrint 513; they are measured in TRVPrint 513.Units 521.

In RichViewActions, this type is used to define units of values displayed in dialog windows. They are measured in TRVAControlPanel.UnitsDisplay.

In ScaleRichView, this type is used for properties of TSRichViewEdit.PageProperty and TSRichViewEdit.ViewProperty. They are measured in TSRichViewEdit.UnitsProgram.

Properties measured in TRVStyleUnits

Properties of the type TRVStyleLength are measured in TRVStyleUnits. They are values of an integer type that can represent lengths measured in twips or screen pixels.

The property TRVStyle.Units defines units for properties (and sub-properties) of this TRVStyle component as well as for properties of items in documents loaded in linked TRichView controls.

Properties of TRVStyle measured in TRVStyleUnits

The following properties are measured in TRVStyleUnits:

- DefTabWidth;
- CharSpacing of text style;
- FirstIndent, LeftIndent, RightIndent, SpaceBefore, SpaceAfter of paragraph style;
- Position of item in a tab stop collection of paragraph style;
- Left, Right, Top, Bottom of background padding rectangle (Background.BorderOffsets) of paragraph style;
- Left, Right, Top, Bottom of border padding rectangle (Border.BorderOffsets) of paragraph style, as well as border's Width and InternalWidth;
- MarkerIndent, FirstIndent, LeftIndent in list levels of list style.

Item properties measured in TRVStyleUnits

The following items properties are measured in TRVStyleUnits:

- *rvepSpacing, rvepImageWidth, rvepImageHeight, rvepVShift* (the latter only if *rvepVShiftAbs* is nonzero)

The following table properties are measured in TRVStyleUnits:

- BorderWidth, CellBorderWidth;
- BorderHSpacing, BorderVSpacing, CellHSpacing, CellVSpacing;
- CellHPadding, CellVPadding, CellPadding;
- HRuleWidth, VRuleWidth;
- BestWidth (only if value is positive).

The following table cell properties are measured in TRVStyleUnits:

- BestHeight (only if value is positive);
- BestWidth (only if value is positive).

A width of break (horizontal line) is measured in TRVStyleUnits as well.

The following label item properties are measured in TRVStyleUnits:

- MinWidth.

The following sidenote properties are measured in TRVStyleUnits:

- BoxProperties.Width and Height (if WidthType and HeightType are "absolute");
- BoxPosition.HorizontalOffset and VerticalOffset (only if HorizontalPositionKind and VerticalPositionKind are "absolute").

In RichViewActions, this type is used for TRVAControlPanel.UnitsProgram (units of measurement for properties of several actions).

Import and export

In RTF (Rich Text Format) files, native units are twips, so all properties are saved as twips. When loading, they are converted to TRVStyle.Units.

In HTML, native units are pixels, so all properties are saved in pixels. In HTML CSS, these properties are saved either as integer values in pixels (if TRVStyle.Units=*rvstuPixels*), or as fractional values in points (if TRVStyle.Units=*rvstuTwips*).

In RVF [93] (RichView Format) files, values are saved as TRVStyle [418].Units [437]. When loading, they are either converted to TRVStyle [418].Units [437], or vice versa, TRVStyle [418].Units [437] is changed. See "Units of measurement" in RVF Overview [93]. Alternatively, TRichView can be converted to units from RVF file, if *rvfoCanChangeUnits* is included in RVFOptions [188].

1.2.8.5 Custom drawing

There are the following events for custom drawing.

TCustomRichView [162]

- OnPaint [283] – drawing over RichView document.

TRVPrint [513]

- OnPagePrepaint [530] – drawing on page under the document;
- OnPagePostpaint [530] – drawing on page over the document;
- OnPrintComponent [554] – printing components.

TRVStyle [418]

- OnDrawCheckpoint [444] – drawing checkpoint;
- OnDrawPageBreak [446] – drawing page break;
- OnDrawParaBack [447] – drawing background of paragraph;
- OnDrawTextBack [449] – drawing background of text;
- OnApplyStyle [443] – applying text style to Canvas;
- OnApplyStyleColor [444] – applying the text style color to Canvas;
- OnDrawStyleText [448] – drawing text of the given style;
- OnStyleHoverSensitive [450] – returns "should component be repainted when user moves the mouse to a hypertext of the given style?"

TRVTableItemInfo [565]

- OnDrawBorder [603] – drawing table and cell borders.

1.2.8.6 Cursors

Image	Constant	Value	Used by	Unit
 *	*crJump*	101	TRVStyle [418] component: ▪ TRVStyle.JumpCursor [427]; ▪ TFontInfo.JumpCursor [472].	**RVStyle.pas**
	*crRVZoomIn*	102	TRVPrintPreview [417] component (ZoomInCursor [560])	**CRVPP.pas**
	*crRVZoomOut*	103	TRVPrintPreview [417] component (ZoomOutCursor [561])	**CRVPP.pas**
	*crRVSelectRow*	104	TRVTableItemInfo [565]	**RVTable.pas**
	*crRVSelectCol*	105	TRVTableItemInfo [565]	**RVTable.pas**
	*crRVFlipArrow*	106	TRVStyle [418] component (LineSelectCursor [427])	**RVStyle.pas**

	*crRVLayingIBeam*	107	TRichViewEdit[316]	**RVStyle.pas**

* in Windows2000+, RichView uses the system default hypertext cursor instead (since version 1.4)

1.2.8.7 RVF Specification

This topic contains thorough information about RichView file Format.

For the most of applications you need not to know about internals of RVF, all necessary information is in RVF Overview[93].

Syntax

{x} – zero or more repetitions of item x;

{x}+ – one or more repetitions of item x;

[x] – item x is optional;

xy – item x is followed by item y;

x | y – item x or item y;

'c' – a literal;

<n> – a nonterminal.

Data Types in RVF

<int> – an integer number, ['-']{<digit>}+;

<digit> ::= '0' | ... | '9';

Integer number is either finished by ' ' (space character) or it is followed by the end of line; for the simplification, this space character will be omitted (i.e. it will be included in <int>).

<crlf> – end of line characters; usually it is a CR+LF, CR or LF characters; before binary data or Unicode it is required to use both CR and LF characters (#13#10).

<string> – string of ANSI characters (without CR, LF and #0 characters)

<Unicode string> – string of Unicode characters. Last character in string must be a paragraph separator (code $2029).

<special string> – string containing ANSI text if RVF version is 1.3 or lower, and UTF-8 text if RVF version is 1.3.1 and newer. In this string, character #1 represents character #13, character #2 represents character #10.

<special string 2> – string containing ANSI text if RVF version is 1.3.1 or lower, and UTF-8 text if RVF version is 1.3.2 and newer. In this string, character #1 represents character #13, character #2 represents character #10.

<quoted string> – <special string> enclosed in double quotes; double quotes inside the string are doubled.

RVF Structure

<RVF> ::= [<Version Info>]{<RVF Record>}

Version Info

<Version Info> := <Version Record Type><Version><SubVersion>[<SubSubVersion>]<crlf>

<Version Record Type> ::= '-8'

<Version> ::= <int>

<SubVersion> ::= <int>

<SubSubVersion> ::= <int>

- Currently, <Version> = 1, <SubVersion> = 3, <SubSubVersion> = 2. In this version, item names, tags and string properties are stored as UTF-8.
- Previously, <Version> = 1, <SubVersion> = 3. <SubSubVersion> = 1 for Delphi 2009+, and omitted for older versions of Delphi. In this version, item names are saved as ANSI, tags and string properties are stored as UTF-8 if <SubSubVersion> = 1, or as ANSI if <SubSubVersion> = 0 or omitted.
- If there is no <Version Info> in RVF, reader implies version 1.0 (no <Item Options> in <RVF Item>)

RVF Record Structure

RVF file consists of *records*. The most of records represent items (one record = one item), but there are some special record types.

<RVF Record> ::= <Record Header><crlf>({<string><crlf>}|{<Unicode sting>})[<Binary Data>]

<Record Header> ::= <Record Type><Strings Count><Paragraph><Item Options><Query><Tag><Rest of Header>

<Record Type> ::= <int>

<Strings Count> ::= <int>

<Paragraph> ::= <int>

<Item Options>::=<int>

<Query> ::= `'0'` | `'1'` | `'2'` | `'3'`

<Tag> ::= <quoted string>

<Record Type> is a type of <RVF Record>

<Record Type>	Constant	Meaning
>=0	—	text[120]; <Item Type> is an index in the collection of styles (RVStyle.TextStyles[437])
-1,-3..-6, -10..-12, -60, ...	*rvs**** [711] constants	non-text item[118]
-2	*rvsCheckpoint*	invisible label ("checkpoint"[65])
-7	*rvsBack*	background (image, color)
-8	*rvsVersionInfo*	RVF version (only in <Version Info>)
-9	*rvsDocProperty*	text, paragraph, list styles,layout information, DocProperties[176], DocParameters[176]

<Strings Count> is a number of lines between this <Record Header> and the next <Record Header> (or the end of file). If RVF reader does not understand this record type it should skip this number of lines. <Binary Data> (if present) is counted as one of strings after the header..

<Paragraph> is an index in the collection of paragraph styles (RVStyle.ParaStyles[431]) (if this item starts a paragraph) or -1 (if the record represents item, and this item continues paragraph).

<Item Options> is a set of bits, converted to integer.

Bit Number	Meaning
0	Set if the item continues paragraph (<Paragraph>=-1); it's a redundant data;

1	Page break. Set if this item must start a new page (<Paragraph><>-1)
2	Set if the item should be displayed from the new line, but it does not start a new paragraph (in this case <Paragraph><>-1, but should be equal to the paragraph style of the previous non "-1" item)
3	Set if it is a text item containing Unicode.

<Tag>

<Tag> is a string associated with this item. RichView saves this string in double quotes. Empty tags are saved as '0' (without quotes). Tag strings are saved as UTF-8 in RVF 1.3.1 or newer, and as ANSI in older versions of RVF. On saving, #13 and #10 are changed to #1 and #2, on loading, there are restored back.

See also *tags* 68.

<Query> can be a digit in the range 0..3

<Query>	Meaning
0	The item content is saved in a text format (*rvfoSaveBinary* is excluded from RichView.RVFOptions 188)
1	The item is requested from program on loading, see below.
2	The item content is saved in a binary format (rvfoSaveBinary is in RichView.RVFOptions 188).
3	Special value for saving Unicode text as hexadecimal string

If <Query>=1, then the next line after the header is a name of item (picture or control).

Data is requested in events: OnRVFPictureNeeded 289, OnRVFControlNeeded 288.

The saving mode (to save full information about images or controls / to save only name and request it when reading) is determined by RichView.RVFOptions 188 (for the request mode, *rvfoSavePicturesBody* and/or *rvfoSaveControlsBody* options should be cleared). You need to create control or image in this event and assign it to **gr** or **ctrl** parameters. For example:

```
procedure TMyForm.MyRichViewRVFControlNeeded(
  Sender: TCustomRichView; Name: TRVUnicodeString 694;
  Tag: TRVTag 692; var ctrl: TControl);
begin
  ctrl := TButton.Create(nil);
  TButton(ctrl).Caption := 'Demo';
  TButton(ctrl).OnClick := MyClickProcedure;
end;
```

<Rest of Header> varies for different item types. Some new parameters can be added in future to <Rest of Header>. RVF reader should ignore all parameters in header after last parameter it can understand.

If <Query> is equal to 1 then <Rest of Header> is empty.

The next section is for <Query>=0 | 2 | 3.

RVF Records

Checkpoints

<Record Type> = *rvsCheckpoint*

<Paragraph> is ignored.

<String Count> = 1

<Rest of Header> ::= <Raise Event>

<Raise Event> ::= '0' | '1'

If Raise Event is equal to '1' then OnCheckPointVisible[274] event occurs when the checkpoint appears in visible area of RichView

Strings after the header:

- <Checkpoint name>

<Checkpoint name> ::= <special string>

"Checkpoint" is not an item, but it's convenient to save it like an item. "Checkpoints" are saved before the associated items.

Text[120] (Items)

<Record Type> >= 0

<Rest of Header> := [<Hint>]

<Hint> := <special string>, value of rvespHint[676] property.

<Strings Count> is a number of paragraphs in this item. All paragraphs will be of the given text and paragraph style. If <Paragraph>=-1 then this item continues the paragraph of the previous item.

Strings after the header are text lines (paragraphs). Stings can be in Unicode encoding (determined by <Item Options>). If <Query>=0, Unicode is saved as UTF-16 (WideString). If <Query>=3, Unicode is saved as hexadecimal string

Break[125] (Item)

<Record Type> = *rvsBreak*

<Paragraph> is ignored

<Rest of Header> ::= <Break Color><Break Style><Break Width>

<Break Color> ::= <int>

<Break Style> ::= <int>

<Break Width> ::= <int>

<Break Color> is a color of horizontal line; if Break Color=*clNone* then line will be of color of the 0th text style.

<Break Style> is TRVBreakStyle[669] value converted to integer.

<Break Width> is a width of line.

<String Count> >= 0.

Picture[122] (Item)

<Record Type> = *rvsPicture*

<String Count> >= 3

<Rest of Header> = <Vertical Align>

<Vertical Align> ::= <int>

<Vertical Align> is a vertical alignment of picture (TRVVAlign[695] value converted to integer)

Strings after the header:

- <Picture Name>
- <Picture Delphi Type>
- <Picture Data>

<Picture Name> ::= <special string 2>, it is an item text.

<Picture Delphi Type> is a name of picture class (such as `'TBitmap', 'TIcon', 'TMetafile', 'TJPEGImage'`).

<Picture Data> can be in text or binary format.

In text format, each byte is represented as two hexadecimal digits, so all data is encoded in one string.

In binary format, <crlf> after <Picture Delphi Type> must be CR+LF (both characters). Binary data consists of <Data Size> and <Data>. Data Size is a binary 4-byte number, the number of bytes in <Data>.

Note 1: In order to load picture of the given class from RVF, its class must be registered with **RegisterClasses** procedure. You need not to register TBitmap, TIcon, TMetafile, because they are registered by RichView. For Delphi 3+ RichView also registers TJpegImage (if the conditional define RVDONOTUSEJPEGIMAGE is not defined) In order to use third party graphics formats (descendants of TGraphic) in RVF you need to register them yourself with **RegisterClasses** procedure.

Note 2: Behavior of RichView when loading unregistered graphics format is defined in RVFOptions[188] property. If *rvfoIgnoreUnknownPicFmt* option is defined then this picture will be just skipped. If this option is not defined then the function returns False (reading failed). In both cases RVFWarnings[191] property will contain *rvfwUnknownPicFmt*.

Control[126] (Item)

<Record Type> = *rvsComponent*

See the previous section, adjusted for: *rvfoIgnoreUnknownCtrls*, *rvfwUnknownCtrls*.

Bullet[128] (Item)

<Record Type> = *rvsBullet*

<String Count> >= 1

<Rest of Header> = <ImageList#><ImageIndex>

<ImageList#> ::= <int>

<ImageIndex> ::= <int>

Strings after the header:

- <Bullet Name>

<Bullet Name> ::= <special string 2>, it is an item text.

<ImageList#> is the Tag property of the corresponding ImageList (do not confuse it with RichView item tags! It is TImageList.Tag property). On reading, OnRVFImageListNeeded[288] occurs. You need to assign your imagelist to **il** parameter. Otherwise, the bullet will be skipped. The behavior of RichView when loading too large image index is defined in RVFOptions[188] property. If *rvfoConvLargeImageIdxToZero* option is defined then index is set to 0. Otherwise the function returns False (reading failed). In both cases RVFWarnings[191] property will contain *rvfwConvLargeImageIdx*.

Hotspot[129] (Item)

<Record Type> = *rvsHotspot*

<String Count> >= 1

<Rest of Header> = <ImageList#><ImageIndex><HotImageIndex>

<ImageList#> ::= <int>

<ImageIndex> ::= <int>

<HotImageIndex> ::= <int>

Strings after the header:

- <Hotspot Name>

<Hotspot Name> ::= <special string 2>, it is an item text.

See the previous section for details.

Collections of styles, layout, DocProperties[176]

<Record Type> = *rvsDocProperty*

<Paragraph> is ignored

<Query> can be 0 (data will be saved as a hexadecimal text) or 2 (in a binary form)

<Rest of Header> ::= `'1' | '2' | '3' | '4' | '5' | '6'|'7'|'8'|'9'|'10'|'11'|'12'`

<Rest of Header>	Meaning
1	Collection of text styles (Style[193].TextStyles[437])
2	Collection of paragraph styles (Style[193].ParaStyles[431])
3	Layout (margins, BiDiMode[170], MaxTextWidth[181], MinTextWidth[181]) of TRichView
4	Collection of list styles (Style[193].ListStyles[429])
5	DocProperties[176]
6	Special information, written by TRVReportHelper[540] when saving page
7	DocParameters[176]
8	Subdocument (headers and footers for TSRichViewEdit)
9	RVStyle.Units[437]
10	Collection of style templates (Style[193].StyleTemplates[435])
11	List of names of style templates
12	DocObjects[174]

<String Count> = 2 for 1..4 and 6, 7, 10. The first line contains name of TRVStyle object for 1,2,4,10 or empty for 3,6,9,12. Hexadecimal text or binary data follow next.

When saving DocProperties, <String Count>=DocProperties.Count. Each item in DocProperties is saved as <special string>.

DocParameters are saved as lines like property=value (only for properties with non-default values).

For 7, the first line contains a name of a subdocument (such as 'header').

For 11, each style template is saved in two lines, so <String Count>=StyleTemplates.Count*2. The first line contains its name[496] (<special string>), the second line contains its id[495].

For 12, DocObjects are saved using a special TRVCollectionBuilder collection, to allow storing items of different classes.

Background

<Record Type> = *rvsBackground*

This item is similar to *rvsPicture*, but has differences:

- <Rest of Header> is a background color and style
- there is no <Picture Name> line;
- <Picture Delphi Type> is always TBitmap
- <Query>=1 is not supported

<String Count> = 2 | 0

<Paragraph> and <Tag> are ignored (0).

<Rest of Header>=<Background Style><Color>

<Background Style>::= <int> (TRVBackgroundStyle[170] value converted to integer)

<Color>::=<int> (background color)

If background style is *bsNoBitmap* (or background bitmap is empty), there are no strings after header (and <Query>=0).

Strings after the header:

- <Picture Delphi Type>
- <Picture Data>

List Markers[131] (Item)

<Record Type> = *rvsMarker*

<String Count> >= 0

<Rest of Header> ::= <List Index> <Level> <Start From> <Reset>

All items in <Rest of Header> are <int> (<Reset> is ord(boolean value))

Label[133] (Item)

<Record Type> = *rvsLabel*

<String Count> >= 6

Strings after the header:

- <Text>
- <Text Style>
- <Min Width>
- <Alignment>
- `'protect'|'no protect'`
- <Item Name>

<Text> ::= <special string>, it is Text[619] property. <Text Style> ::= <int>, it is TextStyleNo[620] property. <Min Width> ::= <int>, it is MinWidth[618] property. <Alignment> ::= <int>, it is ord(Alignment[618]). The next line depends on ProtectTextStyleNo[619]. <Item Name> ::= <string>.

Sequence[134] (Item)

<Record Type> = *rvsSequence*

<String Count> >= 10

Strings after the header:

- <Sequence Id>
- <Numbering Type>
- <Reset>
- <Start From>
- <Text Style>
- <Min Width>
- <Alignment>
- `'protect'|'no protect'`
- <Item Name>
- <Formatting String>

<Sequence Id> ::= <special string>, it is SeqName[626] property. <Numbering Type> ::= <int>, it is NumberType[625] property. <Reset> ::= <int>, it is ord(Reset[626]). <Start From> ::= <int>, it is Reset[626]

property. <Text Style>. The next five lines have the same meaning as for labels. <Formatting String> ::= <special string>, it is FormatString[625] property.

Footnote[136] and EndNote[135] (Item)

<Record Type> = *rvsFootnote* | *rvsEndnote*

<String Count> >= 8

Strings after the header:

- <Text Style>
- <Min Width>
- <Alignment>
- `'protect'|'no protect'`
- <Reset>
- <Start From>
- <Item Name>
- <Document>

The first seven lines have the same meaning as in sequences.

<Document> can be in text or binary format. It is Document[628] saved in RVF format.

In text format, each byte is represented as two hexadecimal digits, so all data is encoded in one string.

In binary format, <crlf> after <Item Name> must be CR+LF (both characters). Binary data consist of <Data Size> and <Data>. Data Size is a binary 4-byte number, the number of bytes in <Data>.

Reference to Footnote or Endnote[140] (Item)

<Record Type> = *rvsNoteReference*

<String Count> >= 5

Strings after the header:

- <Text Style>
- <Min Width>
- <Alignment>
- `'protect'|'no protect'`
- <Item Name>

Meanings of these lines are the same as for labels.

Extra Item Properties

Some items may contain extra integer[673] and string[676] properties. They are saved as lines after the header (one line per property) in the form:

```
name=value
```

If the property has the default value, it is not saved. The list of names can be found in RVItem.pas, arrays RVFExtraItemIntPropNames and RVFExtraItemStrPropNames.

Since TRichView v11, values are saved as <special string>

This specifications does not cover a case of saving style names instead of style indices (this feature is deprecated)

Major Changes in RVF Format

RVF Version 1.2

Older programs are not able to read new RVF files, but programs compiled with new versions of RichView can read old RVF files.

The most significant change is adding <Item Options> in <RVF Item>.

There are two new record types: -7 (background) and -8 (version information)

RVF reader reads version number from version information. If there is no version information item, reader understand the file as RVF version 1.0

RVF version 1.3 (used in TRichView before v17.3 for Delphi 2007 or older):

- allows saving space characters in tag strings (for Delphi 3+); tags strings are saved in double quotes;
- in text mode, Unicode text is saved as a hexadecimal string (each byte is saved as two characters – hexadecimal digits)

RVF version 1.3.1 (used in TRichView before v17.3 for Delphi 2009 or newer):

- DocProperties 176, extra string properties 676, tag strings (if *rvoTagsArePChars* in Options 182; note: since TRichView v13.2 tags are always strings), style names (if *rvfoUseStyleNames* in RVFOptions 188) are saved as UTF-8 (with CR and LF characters changed to #1 and #2)

RVF version 1.3.2 (current version):

Item names are stored as UTF-8.

More Detailed Information about Changes in RVF

Changes in RichView version 1.3:

RVF now can contain Unicode text (for text items). Older versions will be unable to load files with Unicode. Version number in RVF header was not changed, because there are no changes in the specification (additions only).

Note: Unicode is used only for saving Unicode text items, so file contains a mix of ANSI and Unicode text. Such files must be considered as binary, do not edit them in text editors.

Changes in RichView version 1.4:

Added: tables (rvsTable). This is not a change in RVF, but an addition.

Tables are saved in binary or hexadecimal text form, similar to pictures or controls.

Changes in RichView version 1.5:

RVF now can contain text and paragraph styles (rvsDocProperty). This is not a change in RVF, but addition.

Styles are saved in binary or hexadecimal text form, similar to pictures or controls.

Changes in RichView version 1.7:

RVF now can contain layout information and collection of list styles (rvsDocProperty). This is not a change in RVF, but addition.

Changes in RichView version 1.8

RVF version number is changed to 1.3

Changes in RichView version 11

RVF version number is changed to 1.3.1 for Delphi 2009 and newer. Values of extra item properties are saved as <special string>, thus allowing them to contain line break characters.

Changes in RichView version 13

Saving measuring units. Sizes can be in twips.

Changes in RichView version 17.2

RVF version number is changed to 1.3.2. Item names are saved as <special string 2>, i.e. as UTF-8.

1.3 Overview of Item Types

Item types

- RichView Item Types 118
- Text Items 120
- Tabulators 121
- Pictures 122
- *Hot-Pictures* 125
- *Breaks* 125
- Controls 126
- *Bullets* 128 (images from image-lists)
- *Hotspots* 129
- Tables 130
- List markers 131 (paragraph bullets & numbering)
- Labels 133 (non-text item looking like text)
- Numbered sequences 134
- Endnotes 135
- Footnotes 136
- Sidenotes 137 (notes displayed in floating boxes)
- Text Boxes 139 (floating boxes)
- References to parent notes 140
- "Page number" fields 141
- "Page count" fields 142
- Custom Item Types 144

1.3.1 Overview

Items

TRichView documents consist of items. Items are indexed from 0 to ItemCount 180-1. Some items may contain sub-documents which in their order contain items. Such items are:

- tables 145 (each cell is a sub-document displayed in the same TRichView control)
- notes (endnotes 135, footnotes 136, sidenotes 137) and text boxes 139 (their sub-documents are not displayed in TRichView controls; however, they are printed and displayed in ScaleRichView)

Items of subdocuments are not items of the main document, the are not included in the item range from 0 to ItemCount 180-1.

Item Types

A type (also called "style" in this manual) identifier of the given item is returned by the method GetItemStyle 229 (in editor, a style of the item at the position of caret is returned by CurItemStyle 323).

If the returned value is 0 or positive, this is a text item. In this case, the returned value defines a style of this text: it is an index in the TextStyles 437 collection of the the linked 193 TRVStyle 418 component.

If the returned value is negative, this is a constant identifying the item class.

The standard TRichView item types are:

- text 120
- tabulators 121
- pictures 122
- *hot-pictures* 125 (pictures with hyperlink)
- *breaks* 125 (horizontal lines)
- controls 126 (any Delphi/C++Builder control)
- *bullets* 128 (images from image-list)
- *hotspots* 129 (bullets with hyperlink)

- tables 130
- list markers 131 (paragraph bullets & numbering)
- labels 133 (non-text item looking like text)
- numbered sequences 134
- endnotes 135
- footnotes 136
- sidenotes 137
- references to parent notes 140
- text boxes 139
- mathematical expressions 143

Below is the overview of properties which are common for all item types.

Text

Each item has associated text. For text items, this text is displayed. For non-text items, this text is not used by the component itself; you can store any information in it. Text must not contain characters: CR (#13), LF (#10), #0.

For text-items, text can be in traditional (ANSI) or in Unicode 99 encoding. ANSI text must not contain characters (CR (#13), LF (#10), #0, TAB (#9), FF (12).

See GetItemText 230, SetItemText 267 methods of TRichView and related methods of TRichViewEdit.

Tag

Tag is an additional string value associated with each item (Unicode for Delphi/C++Builder 2009+, ANSI for older versions).

The default value of tags is '' (empty string) .

For example, you can use tags to store targets of hyperlinks.

See also: Tags in TRichView 68

Checkpoint

Checkpoint marks the item and can be used to find this item quickly. *Checkpoints* are exported in RTF and DocX as bookmarks, in HTML as anchors.

See also: Checkpoints in TRichView 65

Pagebreak-before Flag

If this flag is set, this item starts a new page when printing.

See PageBreaksBeforeItems 184.

Paragraph Attributes

Items are organized in paragraphs. Paragraph attributes are defined in the ParaStyles 431 collection of the linked 193 TRVStyle 418 component.

GetItemPara 229 returns an index of paragraph style for the given item. This method returns the same value for all items in the same paragraph.

IsParaStart 242 returns True, if the given item starts a new paragraph.

IsFromNewLine 242 returns True, if this item starts a new paragraph or a new line in this paragraph.

Vertical Alignment

These properties can be applied only to non-text item types which are positioned like a character of text. They cannot be applied to items occupying a full line (such as tables and breaks).

By default, bottom of item is aligned to the base line of text (items of some types can be aligned differently, see TRVVAlign[695]).

You can move item up or down by the specified number of pixels/twips or percent. See the following extra item properties[673]: *rvepVShift*, *rvepVShiftAbs*.

For text items, see VShift[477] property of text style.

Spacing

You can define spacing around the item. See the following extra item properties[673]: *rvepSpacing*

Coordinates

GetItemCoords[226] returns coordinates of the top left corner of the given item relative to the beginning of the document.

GetItemClientCoords[226] returns coordinates of the left top corner of the given item relative to the top left corner of RichView window.

Protection

Text have several option of protection, see Protection[473] properties of text style.

You can protect non-text items from deletion. See the following extra item integer properties[673]: *rvepDeleteProtect*.

Visibility

Items can be hidden[76]. Hidden items are displayed only if *rvoShowHiddenText* is included in the Options[182] property of TRichView[162] control.

Text items are hidden, if *rvteoHidden* is included in the Options[472] property of their style[478].

Non-text items are hidden, if a non-zero value is assigned to *rvepHidden* extra item integer property[673].

Popup Hints

The component can show a hint message when the mouse pointer is above the item.

See the following extra item string properties[676]: *rvespHint*.

Hint message can be dynamically customized in OnItemHint[281] event.

Events

OnItemAction[280] occurs when item is inserted, deleted, moved to undo list, or when item text is modified.

OnSaveItemToFile[297] occurs when saving item in HTML, RTF or text file or stream. This is a "low level" event. It allows you to change how the item is saved.

1.3.2 Text Items

Text Item

Text item displays text in traditional (ANSI) or Unicode[99] encoding.

Style[229] of this item type: >=0 (index in the TextStyles[437] collection of the linked[193] TRVStyle[418]).

Item of this collection (TFontInfo[478] object) completely defines all properties of this text.

Main properties of TFontInfo[478]:

- Jump[479] - determines if this text is hypertext[69] or not;

- properties defining font, colors, background, hypertext cursors, protection, bi-di mode[100], and others.

Text item cannot contain:

- CR and LF (#13 and #10) characters;
- TAB (#09) characters (they are inserted as tabulators[121] or replaced with several space characters, depending on SpacesInTab[435]);
- #0 characters;
- FF (#12) characters (see PageBreaksBeforeItems[184] property of TCustomRichView).

Methods of TCustomRichView[162]

The following viewer-style methods add item of this type to the end of the document:

- single item: AddNL[203] (and shortcut Add[203]), AddFmt[198]
- multiple items: AddTextNL[206].

The following viewer-style methods change text of the given item

- SetItemText -A -W[267]

The following methods return text of the given item:

- GetItemText -A -W[230]

Appending text from file/stream:

- LoadText, LoadTextW[246],
- LoadTextFromStream, LoadTextFromStreamW[247]

Methods of TCustomRichViewEdit[316]

The following editor-style methods insert item of this type in the position of caret:

- single item: InsertStringTag, InsertStringATag, InsertStringWTag[359];
- multiple items: InsertText, -A, -W[360].

The following editor-style methods change text of the given item:

- SetItemTextEd -A -W[385], SetCurrentItemText -A -W[380]

The following methods return text of item at the position of caret:

- GetCurrentItemText -A -W[347]

The following editor-style methods insert text from files in the position of caret:

- InsertTextFromFile, InsertTextFromFileW[361]

See Also...

See also methods for copying and pasting[82], selecting text[81].

Some of the methods above can insert tabulators[121] as well.

See also: classes for item types[563].

1.3.3 Tabulators

Tabulators

A special item used to align the subsequent text to the next tab stop.

Tab stops are measured from the LeftMargin[180] and defined by the following properties:

- paragraph's LeftIndent[485] (for paragraph with hanging (negative) FirstIndent[485]);
- paragraph's collection of Tabs[489]
- TRVStyle.DefTabWidth[422].

Style[229] of this item type: *rvsTab* (-12)

Properties of this item type are defined by:

- its text style (the same properties as for text items[120])

- corresponding tab stop (item in the paragraph's collection of Tabs[489]: TRVTabInfo[504])

Item of this type can be inserted in document using keyboard (Tab key), or together with text or RTF (Rich Text Format).

Value of SpacesInTab[435] property of the linked TRVStyle[418] component must be 0 (default). If it is positive, several spaces are inserted instead of tabulator.

Editor accepts Tab keys only if *rvoWantTabs* is included in EditorOption[325] property.

Tabulators are read from RVF (RichView Format) files and streams regardless of value of SpacesInTab[435] property.

Normally, tabulators are invisible (they can have background color and underline defined by its text style, though). They can be displayed as small arrows if *rvoShowSpecialCharacters* is included in TRichView.Options[182].

Methods of TCustomRichView[162]

Methods that add one text item cannot insert tabulators. They always replace them with several space characters or delete.

The following viewer-style methods can add tabulators (if text contains #09 characters):

- AddTextNL, AddTextNLA, AddTextNLW[206];
- LoadText, LoadTextW[246], LoadTextFromStream, LoadTextFromStreamW[247];
- AddTab[205] – adds tabulator regardless of SpacesInTab mode.

Methods of TCustomRichViewEdit[316]

Methods that add one text item cannot insert tabulators. They always replace them with several space characters or delete.

The following editor-style methods can insert tabulators (if text contains #09 characters):

- InsertText, -A, -W[360], InsertTextFromFile, InsertTextFromFileW[361]
- InsertTab[359] – inserts tabulator regardless of SpacesInTab mode

See also methods for pasting[82].

1.3.4 Pictures

Pictures

Picture item displays a picture stored in TGraphic object.

Delphi has the following standard graphic classes:

- TBitmap (stores bitmap (*.bmp), probably with transparency);
- TMetafile (stores 32-bit (*.emf) or 16-bit (*.wmf) metafile);
- TIcon (stores icon; this class does not work with icons having more than 16 colors properly);
- TJpegImage (D3+; stores Jpegs (*.jpg));
- TGifImage (D2007+, stores Gifs (*.gif));
- TPngImage (D2009+, stores Png (*.png));
- TWicImage (D2010+, by default used for Tiff (*.tif) images, but can be used for other graphic formats).

In addition, you can use third-party graphic classes[720].

Style[229] of this item type: *rvsPicture* (-3)

For hypertext pictures, see *hot-pictures*[125] item type.

Methods of TCustomRichView[162]

The following viewer-style methods add item of this type to the end of the document:

- AddPicture[204],

The following viewer-style method changes the main properties of item of this type:

- SetPictureInfo 269

The following method returns main properties of the given item of this type:

- GetPictureInfo 236

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type in the position of caret:

- InsertPicture 355

The following editor-style methods modify main properties of the item of this type:

- SetCurrentPictureInfo 381,
- SetPictureInfoEd 386

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentPictureInfo 349

Properties

Size

This item type has the following integer properties 673 related to size:

- *rvepImageWidth*, *rvepImageHeight* stretch the image.

Layout and appearance

This item type has the following integer properties 673 related to layout and appearance:

- *rvepSpacing* – padding (spacing between the picture and its border; if a background color is specified, this area is colored);
- *rvepColor* – background color;
- *rvepBorderWidth, rvepBorderColor* – width and color of a border;
- *rvepOuterHSpacing, rvepOuterVSpacing* – horizontal and vertical spacing around the border.

Memory usage

By default, TRichView frees a graphic object when this item is destroyed, or when a new graphic object is assigned.

However, if *rvepShared* integer property 673 is nozero, TRichView does not free it.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount 702). Shared images are never deactivated.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign 695;
- extra integer properties 673: *rvepVShift*, *rvepVShiftAbs*.

Resizing

In editor, pictures can be resized with the mouse (except for TIcon). You can forbid mouse resizing by including *rvoNoImageResize* in EditorOptions 325. You can forbid resizing for the specific image by assigning 0 to *rvepResizable* integer property 673.

Resizing with mouse does not modify the image itself, it changes *rvepImageWidth* and *rvepImageHeight*.

OnItemResize 393 event occurs after resizing.

Other properties

This item type has the following specific integer properties 673:

- *rvepTransparent*, *rvepTransparentMode*, *rvepTransparentColor* define transparency in bitmap;
- *rvepAnimationInterval* – interval for bitmap animations;
- *rvepMinHeightOnPage* allows printing this image on multiple pages (cannot be applied to metafiles);

- *rvepNoHTMLImageSize* – if nonzero, <IMG width> and <IMG height> are not saved in HTML for this image.

This item type has the following specific string properties 676:

- *rvespAlt* – text representation of the image, saved in HTML as <IMG alt> attribute;
- *rvespImageFileName* – this property can be saved in HTML file as image location.

Saving and Loading

RichView Format (RVF) 93

In order to load these items from RVF, you need to register a class of the graphics. RichView does it automatically for the standard item classes. But if you use third-party graphic classes, register them.

For example:

- Delphi: RegisterClass(TMyGifImage);
- C++Builder: RegisterClass(__classid(TMyGifImage));

Classes of the most of standard graphic formats are automatically registered by TRichView, see the topic about RVGraphicHandler 654.

If *rvfoSavePicturesBody* is excluded from RVFOptions 188, graphic data is not saved in RVF. When loading such RVF files, OnRVFPictureNeeded 289 event occurs. You can use this feature to store images in database or files.

Export to HTML 96

By default, all pictures are saved as JPEGs. If you want to save pictures of some graphic class without conversion, call RVGraphicHandler 710.RegisterHTMLGraphicFormat.

For example:

- Delphi: RVGraphicHandler 710.RegisterHTMLGraphicFormat(TMyGifImage);
- C++Builder: RVGraphicHandler 710.RegisterHTMLGraphicFormat(__classid(TMyGifImage));

The standard PNG and GIF formats may be automatically registered by TRichView, see the topic about RVGraphicHandler 654.

If you want to save pictures in another format, use OnHTMLSaveImage 277 or OnSaveImage2 296 events.

RTF export

RTFOptions 185 contain options for saving pictures in RTF. Pictures are embedded in RTF, external image files are not created.

Bitmaps are stored as bitmaps. 32-bit metafiles (EMF) are stored as 32-bit or 16-bit (WMF) metafiles.

PNG images are stored as PNG images, if PNG class is specified (see RVGraphicHandler 710 .RegisterPngGraphic).

All other graphic formats are stored as bitmaps, WMF or PNG (except for Jpegs which can be stored as Jpegs).

DocX export

All graphic formats are stored as they are, except for:

- bitmaps (they are saved as PNG, if PNG graphic class is defined, see RVGraphicHandler 710 .RegisterPngGraphic;
- icons (they are saved as PNG in Delphi 2009+ using TPngImage class)

RTF Import

You can set RTFReadProperties 187.IgnorePictures 307 to *True*, and pictures in RTF will not be added in RichView.

RTF can have embedded images and external images (in separate files). In order to load external images, a proper graphic class must be associated with the file extension.

Delphi does it for standard graphic classes automatically, but for some third-party classes you need to do it yourself.

For example:

```
TPicture.RegisterFileFormat('gif','Gif Image',TMyGifImage);
```

Read more information in the topic about third-party graphic classes 720.

See Also...

- Working in 256-color display mode 177;
- Animation 90;
- Smooth image scaling 91.

1.3.5 Hot-Pictures

"Hot-picture" is a picture 122 - hypertext link 69.

Style 229 of this item type: *rvsHotPicture* (-10)

All methods working with pictures 122 work with hot-pictures. The difference is only in methods that insert hot-picture in the document.

Methods of TCustomRichView 162

The following viewer-style methods add item of this type to the end of the document:

- AddHotPicture 199.

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type at the position of caret:

- InsertHotPicture 352

ConvertToHotPicture 339, ConvertToPicture 339 convert picture to hot-picture and vice versa.

1.3.6 Breaks

Break is a horizontal line (or rectangle) of the specified width and color.

It always occupies a full line.

Style 229 of this item type: *rvsBreak* (-1)

Methods of TCustomRichView 162

The following viewer-style methods add item of this type to the end of the document:

- AddBreak 196
- The following viewer-style method changes the main properties of item of this type:
- SetBreakInfo 260

The following method returns main properties of the given item of this type:

- GetBreakInfo 218

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type at the position of caret:

- InsertBreak 350

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBreakInfo 375,
- SetBreakInfoEd 371

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentBreakInfo 341

Properties

Main properties of *breaks*: style, width, color.

In addition to the methods listed above, they are accessible as rveipcBreakStyle, rveipcBreakWidth, rveipcBreakColor[706] properties.

Saving and loading

Export to HTML[96]

Breaks are exported as <HR>. Color of *break* is retained only in CSS version (SaveHTMLEx[251], or SaveHTML[250] with *rvsoForceNonTextCSS* in Options parameter)

RTF

Breaks are exported as two paragraphs, one of them with a border at the bottom side. Not imported (i.e. imported back as two paragraphs, one of them with a border at the bottom side)

Export to DocX

Breals are exported as a special line object.

Export to Text

When saving/copying selection, *breaks* are not included in text. When saving to file (SaveText[255]), saved as a sequence of LineWidth '-' characters.

1.3.7 Controls

You can insert any Delphi/C++Builder control in the document.

Style[229] of this item type: *rvsComponent* (-5)

Methods of TCustomRichView[162]

The following viewer-style methods add item of this type to the end of the document:

- AddControl[198].

The following viewer-style method changes the main properties of item of this type:

- SetControlInfo[262]

The following method returns main properties of the given item of this type:

- GetControlInfo[222]

See also

- FindControlItemNo[217]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type at the position of caret:

- InsertControl[352]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentControlInfo[377],
- SetControlInfoEd[374]

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentControlInfo[343]

Also

- ResizeControl[368],
- ResizeCurrentControl[368],
- AdjustControlPlacement2[330]

Events:

- OnControlAction[275] – occurs when the control is loaded from RVF[93], deleted, moved to/from undo buffer, saved to RVF.

Properties

Layout and appearance

This item type has the following integer properties[673] related to layout and appearance:

- *rvepSpacing* – padding (spacing between the control and its border; if a background color is specified, this area is colored);
- *rvepColor* – background color;
- *rvepBorderWidth, rvepBorderColor* – width and color of a border;
- *rvepOuterHSpacing, rvepOuterVSpacing* – horizontal and vertical spacing around the border.

This item type has the following additional additional integer properties[706] related to layout and appearance:

- *rveipcPercentWidth* allows resizing the control according to the document width.

Memory usage

By default, TRichView frees a control when this item is destroyed.

However, if *rvepShared* integer property[673] is nozero, TRichView does not free it.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign[695];
- extra integer properties[673]: *rvepVShift*, *rvepVShiftAbs*.

Resizing

You can allow resizing this control in editor by assigning non-zero value to its *rvepResizable* property[673].

You can forbid mouse resizing of all images and controls in the editor by including *rvoNoImageResize* in EditorOptions[325].

Resizing with mouse changes Width and Height properties of the control.

Resizing is activated if this control is selected. You can select this control (for example, in its OnClick event) by calling RichView.SelectControl[257].

OnItemResize[393] event occurs after resizing.

Other properties

This item type has the following specific integer properties[673]:

- *rvepVisible* allows to hide the control (to set its Visible property to *False*).

Note about Tags of controls

RichView uses tags of inserted controls (TComponent.Tag) for its own needs. Please do not use and do not modify tags of controls.

(this note is about Tag property of component, not about tags of items[68]; you can use item's tag as you wish)

Saving and Loading

RVF[93]

In order to load these items from RVF, you need to register a class of the control.

For example:

- Delphi: RegisterClass(TButton);
- C++Builder: RegisterClass(__classid(TButton));

If *rvfoSaveControlsBody* is excluded from RVFOptions[188], control itself is not saved in RVF. When loading such RVF files, OnRVFControlNeeded[288] event occurs.

Export to text, RTF, and HTML[96]

By default, controls are not exported. Use OnSaveComponentToFile[292] event to save controls.

Printing

RichView creates a temporal bitmap, draws the control in it, then prints this bitmap.

It's not possible to create an universal procedure for drawing all possible controls. If you are not satisfied with the results, override the default printing with OnPrintComponent[554] event.

Special Width Mode

Width of controls can be set in percent:

```
var Item: TRVControlItemInfo[563];
  ...
Item := RichView.GetItem[225](ItemNo) as TRVControlItemInfo;
Item.PercentWidth := 50; // 50% of document width (minus paragraph indents)
```

The effect will be after reformatting[217].

See also...

BeginOleDrag[209]

1.3.8 Bullets

"Bullet" is an image from TImageList. *Bullets* are useful if you want to create document where the same (small) image is used many times.

For example, they can be used for smiles in chats.

This item type has no relation to paragraph bullets and numbering[131].

Style[229] of this item type: *rvsBullet* (-6)

For hypertext *bullets*, see *hotspot*[129] item type.

Methods of TCustomRichView[162]

The following viewer-style methods add item of this type to the end of the document:

- AddBullet[197]

The following viewer-style method changes the main properties of item of this type:

- SetBulletInfo[261]

The following method returns main properties of the given item of this type:

- GetBulletInfo[219]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type at the position of caret:

- InsertBullet[351]

The following editor-style methods modify main properties of the item of this type:

- SetCurrentBulletInfo[375],
- SetBulletInfoEd[372]

The following method returns main properties of the item of this type at the position of caret:

GetCurrentBulletInfo 342

Properties

Main properties

ImageList is specified when this item is added. It cannot be changed.

Image index can be changed by the methods listed above. It is also accessible as rveipcImageIndex 706 property.

Layout and appearance

This item type has the following integer properties 673 related to layout and appearance:

- *rvepSpacing* – padding (spacing between the picture and its border; if a background color is specified, this area is colored);
- *rvepColor* – background color;
- *rvepBorderWidth, rvepBorderColor* – width and color of a border;
- *rvepOuterHSpacing, rvepOuterVSpacing* – horizontal and vertical spacing around the border.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign 695;
- extra integer properties 673: *rvepVShift*, *rvepVShiftAbs*.

Resizing

Bullets cannot be resized.

Other properties

This item type has the following specific string properties 676:

- *rvespAlt* – text representation of the image, saved in HTML as <IMG alt> attribute;

Memory Usage

RichView does not own this image-list, and it does not destroy it when clearing document.

Saving and Loading

RVF 93

ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded 288 occurs allowing you to provide image-lists for *bullets* and *hotspots*.

Export to RTF and HTML 96

Bullets are exported as pictures. If you want to write special code for exporting bullets in HTML, use OnHTMLSaveImage 277 event.

1.3.9 Hotspots

"Hotspot" is a *bullet* 128 (image from TImageList) with hypertext link 69.

Hotspots can change their image under the mouse pointer.

Style 229 of this item type: *rvsHotspot* (-4)

Methods of TCustomRichView 162

The following viewer-style methods add item of this type to the end of the document:

- AddHotspot 200.

The following viewer-style method changes the main properties of item of this type:

- SetHotspotInfo 265

The following method returns main properties of the given item of this type:

- GetHotspotInfo 224,
- GetBulletInfo 219

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type at the position of caret:

- InsertHotspot 353

The following editor-style methods modify main properties of the item of this type:

- SetCurrentHotspotInfo 377,
- SetHotspotInfoEd 382

The following method returns main properties of the item of this type at the position of caret:

- GetCurrentHotspotInfo 344,
- GetCurrentBulletInfo 342

Properties

Hotspots have all properties of *bullets*.

Image index below the mouse pointer can be changed by the methods listed above. It is also accessible as rveipcHotImageIndex 706 property.

Memory Usage

RichView does not own this image-list, and it will not destroy it when clearing document.

1.3.10 Tables

Tables arrange content in rows and columns. Each table cell is a subdocument containing its own items. Tables can be nested.

More details:

- Overview of tables 145,
- **TRVTableItemInfo** 565 (see for detailed information).

Style 229 of this item type: *rvsTable* (-60)

There is an important differences between properties of tables and properties of any other items. To change values of properties of any other item as an editing operation, you need to use special methods. But direct assignment to properties of TRVTableItemInfo is written to an undo buffer, if this table is already inserted in TCustomRichViewEdit 316.

Extra Properties

This item type has the following specific string properties 676:

- *rvespImageFileName* – value of this property can be saved in HTML file as table background image location; also accessible as table.BackgroundImageFileName 568.

Methods of TCustomRichView 162

The following viewer-style methods add item of this type to the end of the document:

- AddItem 201 (create TRVTableItemInfo 565 object, add it using AddItem 201)

The following method returns TRVTableItemInfo 565 object

- GetItem 225

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVTableItemInfo[565].object, insert it using InsertItem[353])

The following methods return TRVTableItemInfo[565] object at the position of caret:

- GetCurrentItem[345],
- GetCurrentItemEx[345].

Resizing

Users can resize table rows and columns, if *rvtoRowSizing* and/or *rvtoColSizing* are included in table.Options[577] .

Resizing changes BestWidth[607] and BestHeight[606] properties of table cells.

OnItemResize[393] event occurs after resizing.

Saving and Loading

Export to HTML[96]

The TRichView's model of tables is very similar to the HTML model of tables, so tables are saved to HTML as they are.

The most of table visual properties can be saved only with use of CSS (use SaveHTMLEx[251] method, or SaveHTML[250] with *rvsoForceNonTextCSS* in Options parameter).

RTF Import

You can set RTFReadProperties[187].IgnoreTables[307] to *True*, and tables in RTF will not be added in RichView (contents of cells will be added in the main document).

RTF tables have different structure, TRichView does a smart work for converting them.

Text export

See the following properties of TRVTableItemInfo[565]:

- TextColSeparator[581];
- TextRowSeparator[581].

1.3.11 List Markers

This is a very special item type representing paragraph bullet or number.

Style[229] of this item type: *rvsListMarker* (-11)

See also: numbered sequences[134].

Properties of List Markers

TRVStyle[418] component has a collection of list styles (ListStyles[429]). Each list style[493] contains a collection of list levels (Levels[493] property). List level[506] defines the most of list properties.

List markers is a special kind of items. They are inserted with special functions. They are always inserted at the beginning of paragraph.

They have the following properties:

- ListNo – index of list style in the collection of list styles[429];
- LevelNo – list level;
- StartFrom – starting value for list counter, if UseStartFrom=True;
- UseStartFrom – if True, list counter value for this marker is defined by StartFrom. If False, numbering is continued.

List Markers Operations

Methods of TCustomRichView:

- SetListMarkerInfo[269],
- GetListMarkerInfo[233],
- RefreshListMarkers[248].

Methods of TCustomRichViewEdit:

- ApplyListStyle[331],
- RemoveLists[367],
- ChangeListLevels[338].

Saving and Loading

RVF[93]

For list levels with ImageLists: ImageLists themselves are not saved in RVF. RichView stores TImageList.Tag. When loading, OnRVFImageListNeeded[288] occurs allowing you to provide an image-list.

HTM Export, rvsoMarkersAsText is not in Options of SaveHTML[250] or SaveHTMLEx[251]

Lists are saved using <ul> and <ol> HTML tags.

Export using CSS (SaveHTMLEx[251]):

CSS of lists are inserted directly in HTML.

- rvlstBullet, rvlstUnicodeBullet[510] are exported as bullets with default markers (disc, circle or square). FormatString[508] and Font[507] are ignored;
- rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman[510] are exported as lists with corresponding numbering. FormatString[508] and Font[507] are ignored;
- rvlstPicture,rvlstImageList,rvlstImageListCounter[510] are exported as bullets with picture. It's possible to change default image saving by OnHTMLSaveImage[277].

Indents are saved, but not very accurately.

Export without CSS (SaveHTML[250])

- rvlstBullet, rvlstUnicodeBullet,rvlstPicture,rvlstImageList[510] are exported as bullets with default markers (disc, circle or square). FormatString[508], Picture[513], ImageList[509] and Font[507] are ignored;
- rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman[510] are exported as lists with corresponding numbering. FormatString[508] and Font[507] are ignored;
- rvlstImageListCounter[510] is saved as a decimal numbering.

Indents are not saved.

HTML Export, rvsoMarkersAsText is in Options

Markers are saved without using special HTML keywords for lists (like <ol>, <ul>, <li>).

Font[507] and format strings[508] settings are respected.

Saving with CSS (SaveHTMLEx[251]): LeftIndent[510] and MarkerIndent[512] are retained. FirstIndent[507] are retained for non-hanging markers only (MarkerIndent>=LeftIndent).

Saving without CSS (SaveHTML[250]): all indents are ignored.

Generally, HTML files saved with this option look closer to the original.

Demo Projects

- **Demos\Delphi\Assorted\ListStyles**
- **Demos\CBuilder\Assorted\ListStyles**

1.3.12 Labels

"Label" is a non-text item looking like text[120]. This text cannot be wrapped (always displayed on one line).

For Delphi/C++Builder 2009 or newer, this text is Unicode[99]. For older versions, it is ANSI.

Class for this item type is **TRVLabelItemInfo**[617] (see for detailed information).

Style[229] of this item type: *rvsLabel* (-200)

The following items are inherited from labels:

- numbered sequences[134]
- footnotes[136]
- endnotes[135]
- sidenotes[137]
- text boxes[139]
- references to notes[140]

Related Properties of TRVStyle[418]

- FieldHighlightColor[423] – color for highlighting label items.
- FieldHighlightType[423] specifies when to highlight label items.

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVLabelItemInfo[617] object, add it using AddItem[201])

The following method returns TRVLabelItemInfo[617] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVLabelItemInfo[617] object, insert it using InsertItem[353])

The following method returns TRVLabelItemInfo[617] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

TextStyleNo[620] property can be modified by ApplyTextStyle[334] and ApplyStyleConversion[333] methods.

Properties

Main properties

- Text[619] – text to display;
- TextStyleNo[620] defines font and colors. This is an index in TextStyles[437] collection of the linked TRVStyle[418] component;
- ProtectTextStyleNo[619] – if *True*, editing operations cannot change TextStyleNo[620];
- MinWidth[618] – minimal width for the label item;
- Alignment[618] – horizontal alignment of text (if the item width is increased by MinWidth[618]);
- Cursor[618] – mouse cursor above this item.
- RemoveInternalLeading[619] – if *True*, the font internal leading is not included in the item height.

These properties are also accessible as *rvespcText*[709], *rveipcProtectTextStyleNo, rveipcMinWidth, rveipcAlignment, rveipcCursor, rveipcRemoveInternalLeading*[706] properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties 673 related to layout and appearance:

- *rvepSpacing* – padding (spacing between the text and its border);
- *rvepBorderWidth, rvepBorderColor* – width and color of a border;
- *rvepOuterHSpacing, rvepOuterVSpacing* – horizontal and vertical spacing around the border.

Vertical Position

This item type has the following properties affecting vertical position:

- VAlign 695;
- extra integer properties 673: *rvepVShift*, *rvepVShiftAbs*.

1.3.13 Numbered Sequences

Numbered sequence is a non-text item looking like text 120 and displaying a number. For example, this item type can be used for numbering pictures, tables or formulas. This text cannot be wrapped (always displayed on one line).

Class for this item type is **TRVSeqItemInfo** 624 (see for detailed information).

Style 229 of this item type: *rvsSequence* (-202)

The following items are inherited from numbered sequences:

- footnotes 136
- endnotes 135
- sidenotes 137
- text boxes 139

Methods of TCustomRichView 162

The following viewer-style method adds item of this type to the end of the document:

- AddItem 201 (create TRVSeqItemInfo 624 object, add it using AddItem 201)

The following method returns TRVSeqItemInfo 624 object

- GetItem 225

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type in the position of caret:

- InsertItem 353 (create TRVSeqItemInfo 624 object, insert it using InsertItem 353)

The following method returns TRVSeqItemInfo 624 object at the position of caret:

- GetCurrentItem 345.

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx 378;
- SetItemExtraIntPropertyExEd 383;
- SetCurrentItemExtraStrPropertyEx 379;
- SetItemExtraStrPropertyExEd 384.

Properties

Numbered sequences have all properties of label items 133, but Text 619 is calculated automatically.

Additionally, it has properties:

- NumberType 625 – numbering type;
- SeqName 626 – name of a sequence (this item continues numbering from the previous item having the same name of a sequence);
- Reset 626 – if *True*, a numbering is not continued but reset from StartFrom 626.
- FormatString 625 – optional format string; if defined, a text to display is formatted according to this string.

These properties are also accessible as rvespcSeqName, rvespcSeqFormatString[709], rveipcSeqStartFrom, rveipcSeqReset, rveipcSeqNumberType[706] properties. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML[96] and text

Numbered sequences are exported as a plain text.

RTF

Numbered sequences can be exported in RTF (all main properties are saved except for FormatString[625]).

You can set RTFReadProperties[187].IgnoreSequences[307] to *True*, and numbered sequences will not be imported (imported as a plain text) from RTF.

1.3.14 Endnotes

An endnote looks like a number (numbering type is specified in TRVStyle.EndnoteNumbering[422]). If FormatString[625] is defined, a number is displayed formatted according to this string (for example, if FormatString is a single character, the endnote displays a single character). Endnote contains a subdocument. This subdocument is not displayed in TCustomRichView where the endnote is inserted, but it can be printed. Subdocuments for all endnotes are printed after the main document.

ScaleRichView note: TSRichViewEdit displays and allows editing endnotes. TSRichViewEdit cannot print a text for an endnote on several pages.

Class for this item type is **TRVEndnoteItemInfo**[629] (see for detailed information).

Style[229] of this item type: *rvsEndnote* (-204)

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVEndnoteItemInfo[629] object, add it using AddItem[201])

The following method returns TRVEndnoteItemInfo[629] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVEndnoteItemInfo[629] object, insert it using InsertItem[353])

The following method returns TRVEndnoteItemInfo[629] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Endnotes have all properties of numbered sequences[134].

Differences:

- NumberType[625] is read-only; it always returns TRVStyle.EndnoteNumbering[422];
- SeqName[626] = '@endnote@' and must not be changed.

Saving and Loading

Export to HTML [96]

Subdocuments for all endnotes are saved after the main document. Before each subdocument, is inserted, where *N* is an index of this endnote (in the list of all numbered sequences [134] and notes). Endnotes are saved as hyperlinks to this anchor.

RTF

Endnotes can be exported in RTF. If FormatString [625] contains a single character, it can be exported to RTF. Reset [626] and StartFrom [626] properties cannot be exported.

You can set RTFReadProperties [187].IgnoreNotes [307] to *True*, and endnotes will not be imported from RTF.

DocX

Endnotes can be exported in DocX. If FormatString [625] contains a single character, it can be exported to DocX.

Text

When copying or saving a selected text, endnotes are not included in text. When saving the whole text, they are saved at the end of the text:

```
This is a text [i] to show endnotes [ii].
-----------------
i sequence of characters
ii note of text placed at the end of the document
```

1.3.15 Footnotes

A footnote looks like a number (numbering type is specified in TRVStyle.FootnoteNumbering [422]). If FormatString [625] is defined, a number is displayed formatted according to this string (for example, if FormatString is a single character, the footnote displays a single character).

Footnote contains a subdocument. This subdocument is not displayed in TCustomRichView where the footnote is inserted, but it can be printed. Subdocuments for all footnotes are printed at the bottom of page where this footnote occurs. TRichView cannot print text for a footnote on several pages.

ScaleRichView note: TSRichViewEdit displays and allows editing footnotes.

Class for this item type is **TRVFootnoteItemInfo** [630] (see for detailed information).

Style [229] of this item type: *rvsFootnote* (-203)

Methods of TCustomRichView [162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem [201] (create TRVFootnoteItemInfo [630] object, add it using AddItem [201])

The following method returns TRVFootnoteItemInfo [630] object

- GetItem [225]

Methods of TCustomRichViewEdit [316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem [353] (create TRVFootnoteItemInfo [630] object, insert it using InsertItem [353])

The following method returns TRVFootnoteItemInfo [630] object at the position of caret:

- GetCurrentItem [345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx [378];
- SetItemExtraIntPropertyExEd [383];

- SetCurrentItemExtraStrPropertyEx 379;
- SetItemExtraStrPropertyExEd 384.

Properties

Footnotes have all properties of numbered sequences 134.

Differences:

- NumberType 625 is read-only; it always returns TRVStyle.FootnoteNumbering 422;
- SeqName 626 = '@footnote@' and must not be changed.

Saving and Loading

Export to HTML 96

Subdocuments for all footnotes are saved after the main document. Before each subdocument, is inserted, where *N* is an index of this footnote (in the list of all numbered sequences 134 and notes). Footnotes are saved as hyperlinks to this anchor.

RTF

Footnotes can be exported in RTF. If FormatString 625 contains a single character, it can be exported to RTF.

You can set RTFReadProperties 187.IgnoreNotes 307 to *True*, and footnotes will not be imported from RTF.

DocX

Footnotes can be exported in DocX. If FormatString 625 contains a single character, it can be exported to DocX.

Text

When copying or saving a selected text, footnotes are not included in text. When saving the whole text, they are saved inside [], for example:

```
This is a text [1 sequence of characters] to show footnotes [2 note of text placed at the
```

1.3.16 Sidenotes

A sidenote looks like a number (numbering type is specified in TRVStyle.SidenoteNumbering 422). If FormatString 625 is defined, a number is displayed formatted according to this string.

Sidenote contains a subdocument. This subdocument is not displayed in TCustomRichView where the sidenote is inserted, but it can be printed inside a floating box.

ScaleRichView note: TSRichViewEdit displays and allows editing sidenotes.

Class for this item type is **TRVSidenoteItemInfo** 632 (see for detailed information).

Style 229 of this item type: *rvsSidenote* (-206)

The following items are inherited from numbered sidenotes:

- text boxes 139

Methods of TCustomRichView 162

The following viewer-style method adds item of this type to the end of the document:

- AddItem 201 (create TRVSidenoteItemInfo 632 object, add it using AddItem 201)

The following method returns TRVSidenoteItemInfo 632 object

- GetItem 225

Methods of TCustomRichViewEdit 316

The following editor-style method inserts item of this type in the position of caret:

- InsertItem 353 (create TRVSidenoteItemInfo 632 object, insert it using InsertItem 353)

The following method returns TRVSidenoteItemInfo[632] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Sidenotes have all properties of numbered sequences[134].

Differences:

- NumberType[625] is read-only; it always returns TRVStyle.SidenoteNumbering[422];
- SeqName[626] = '@sidenote@' and must not be changed.

Additionally, it has the following properties:

- BoxPosition[637] – properties relative to the position of a text box;
- BoxProperties[635] – size, border, background, content alignment of a text box.

These properties are also accessible as rveipcFloatHorizontalAnchor, rveipcFloatHorizontalPositionKind, rveipcFloatHorizontalOffset, rveipcFloatHorizontalAlign, rveipcFloatVerticalAnchor, rveipcFloatVerticalPositionKind, rveipcFloatVerticalOffset, rveipcFloatVerticalAlign, rveipcFloatRelativeToCell, rveipcFloatPositionInText, rveipcBoxWidth, rveipcBoxHeight, rveipcBoxWidthType, rveipcBoxHeightType, rveipcBoxBorderColor, rveipcBoxBorderWidth, rveipcBoxBorderInternalWidth, rveipcBoxBorderStyle, rveipcBoxBorderVisibleBorders_Left, rveipcBoxBorderVisibleBorders_Top, rveipcBoxBorderVisibleBorders_Right, rveipcBoxBorderVisibleBorders_Bottom, rveipcBoxBorderBorderOffsets_Left, rveipcBoxBorderBorderOffsets_Top, rveipcBoxBorderBorderOffsets_Right, rveipcBoxBorderBorderOffsets_Bottom, rveipcBoxColor, rveipcBoxPadding_Left, rveipcBoxPadding_Top, rveipcBoxPadding_Right, rveipcBoxPadding_Bottom, rveipcBoxVAlign properties[706]. Accessing them in this way allows changing them in editing operations.

Saving and Loading

Export to HTML[96]

Subdocuments for all sidenotes are saved after the main document. Before each subdocument, is inserted, where *N* is an index of this sidenote (in the list of all numbered sequences[134] and notes). Sidenotes are saved as hyperlinks to this anchor.

RTF

Sidenotes can be exported in RTF. They are imported as NoteText[628] followed by a text box containing Document[628]. A bookmark named '_sidenoteN' (where *N* is an index of this sidenote) is added around NoteText, to allow saving note references[140] as references to this bookmark.

Some combinations of box positioning properties are not supported by Microsoft Word*, so TRichView converts them:

- MS Word does not support a vertical alignment relative to a paragraph, so it saved as an alignment relative to a line;
- MS Word does not support a relative (percent) positioning to a line or paragraph, so it is saved as an absolute positioning (with zero offset);

Other differences*:

- when a line is chosen as an anchor object, MS Word aligns a text box not relative to a whole line but relative to a line top, so a center and a bottom alignment relative to a line are different in TRichView and in MS Word;
- MS Word does not implement all options of vertical alignment relative to a table cell, so it works differently in TRichView and MS Word, see BoxPosition[637].VerticalAnchor[641];
- MS Word does not support different padding for border and background, so BoxProperties[635].Border[635].BorderOffsets[502] are not saved to RTF.

(* this information is based on tests of MS Word 2013)

Floating boxes cannot be imported from RTF (yet).

DocX

Sidenotes can be exported in DocX, both in the old (VML) and the new (DrawingML) formats.

See the comments about RTF.

Text

When copying or saving a selected text, sidenotes are not included in text. When saving the whole text, they are saved inside [], for example:

```
This is a text [a sequence of characters] to show sidenotes [b note of text placed in a f
```

1.3.17 Text Boxes

A text box item is a simplified version of a sidenote[137]. There is no visible text at the place of insertion of this item.

If rvoShowSpecialCharacters is included in RichView.Options[182], a special marks are displayed in places of insertion of text box items.

Sidenote contains a subdocument. This subdocument is not displayed in TCustomRichView where the text box item is inserted, but it can be printed inside a floating box.

ScaleRichView note: TSRichViewEdit displays and allows editing text boxes.

Class for this item type is **TRVTextBoxItemInfo**[643] (see for detailed information).

Style[229] of this item type: *rvsTextBox* (-207)

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVSidenoteItemInfo[632] object, add it using AddItem[201])

The following method returns TRVSidenoteItemInfo[632] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVSidenoteItemInfo[632] object, insert it using InsertItem[353])

The following method returns TRVSidenoteItemInfo[632] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Text box items have all properties of sidenotes[137], but almost all of them are ignored.

Only floating box-related properties are used: BoxPosition[637] and BoxProperties[635].

Saving and Loading

Export to HTML[96]

Text boxes are not exported to HTML.

RTF and DocX

Text boxes are exported to RTF and DocX.

See the comments about export of sidenotes[137] for compatibility with Microsoft Word.

Text

When copying or saving a selected text, text boxes are not included in text. When saving the whole text, they are saved inside [], for example:

```
This is a text[sequence of characters] to show text boxes[text placed in a floating box].
```

1.3.18 References to Notes

When inserted in footnote[136]'s/endnote[135]'s/sidenote[137]'s Document[628], the note reference displays the same text as the parent note.

When inserted in TCustomRichView, it displays NoteText[182].

Class for this item type is **TRVNoteReferenceItemInfo**[644] (see for detailed information).

Style[229] of this item type: *rvsNoteReference* (-205)

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVNoteReferenceItemInfo[644] object, add it using AddItem[201])

The following method returns TRVNoteReferenceItemInfo[644] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVNoteReferenceItemInfo[644] object, insert it using InsertItem[353])

The following method returns TRVNoteReferenceItemInfo[644] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Saving and Loading

Export to HTML[96]

Note references are saved as a plain text. In future, they may be saved as hyperlinks to the parent footnote/endnote.

RTF

Note references can be exported in RTF.

You can set RTFReadProperties[187].IgnoreNotes[307] to *True*, and note references will not be imported from RTF.

Export to DocX

Note references can be exported in DocX

1.3.19 Page Numbers

When inserted in TRichView[162] or TRichViewEdit[316], this item is displayed as '{p}'.

When printing with TRVPrint[513], this item displays a page number where this item is located. Page numbers are counted from TRVPrint.PageNoFromNumber[552].

When drawing with TRVReportHelper[540], the default behavior is like in TRVPrint, but you can provide a value for the page number in OnGetPageNumber[543] event.

ScaleRichView: When inserted in TSRichViewEdit, this item displays a page number where this item is located. Page numbers are counted from TSRichViewEdit.PageProperty.PageNoFromNumber.

You can insert this item in the main document, a note, a text box, a header or a footer.

Class for this item type is **TRVPageNumberItemInfo**[647] (see for detailed information).

Style[229] of this item type: *rvsPageNumber* (-208)

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVPageNumberItemInfo[647] object, add it using AddItem[201])

The following method returns TRVPageNumberItemInfo[647] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVPageNumberItemInfo[647] object, insert it using InsertItem[353])

The following method returns TRVPageNumberItemInfo[647] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Fields have all properties of label items[133].

By default, Text[619]='{p}'. You can change this value to display another code in TRichView and TRichViewEdit.

Additionally, it has property:

- NumberType[646] – numbering type;

This property is also accessible as rveipcPageNumberType[706] property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML[96]

Fields are saved as a plain text.

RTF

Fields can saved to RTF and loaded from RTF. You can disallow loading from RTF by assinging TRichView.RTFReadProperties[187].IgnoreFields[307]=*True*.

Export to DocX

Fields can be exported in DocX

1.3.20 Page Counts

When inserted in TRichView[162] or TRichViewEdit[316], this item is displayed as '{P}'.

When printing with TRVPrint[513], this item displays the count of pages.

ScaleRichView: When inserted in TSRichViewEdit, this item displays the count of pages.

Class for this item type is **TRVPageNumberItemInfo**[647] (see for detailed information).

Style[229] of this item type: *rvsPageCount* (-209)

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVPageCountItemInfo[648] object, add it using AddItem[201])

The following method returns TRVPageCountItemInfo[648] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVPageCountItemInfo[648] object, insert it using InsertItem[353])

The following method returns TRVPageCountItemInfo[648] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Fields have all properties of label items[133].

By default, Text[619]='{P}'. You can change this value to display another code in TRichView and TRichViewEdit.

Additionally, it has property:

- NumberType[646] – numbering type;

This property is also accessible as rveipcPageNumberType[706] property. Accessing it in this way allows changing it in editing operations.

Saving and Loading

Export to HTML[96]

Fields are saved as a plain text.

RTF

Fields can saved to RTF and loaded from RTF. You can disallow loading from RTF by assinging TRichView.RTFReadProperties[187].IgnoreFields[307]=*True*.

Export to DocX

Fields can be exported in DocX

Note: If RTF or DocX file is opened in Microsoft Word, this field is displayed as "?" until fields are updated.

1.3.21 Mathematical Expressions

Items of this type display mathematical expressions.

Class for this item type is **TRVMathItemInfo**[621] (see for detailed information).

Style[229] of this item type: *rvsMathEquation* (-210)

This item requires RAD Studio XE4 or newer.

License

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.

You can contact Adit Software if you want to use Adit Math Engine under a different license or to request an exception from the restrictions above.

Methods of TCustomRichView[162]

The following viewer-style method adds item of this type to the end of the document:

- AddItem[201] (create TRVMathItemInfo[621] object, add it using AddItem[201])

The following method returns TRVMathItemInfo[621] object

- GetItem[225]

Methods of TCustomRichViewEdit[316]

The following editor-style method inserts item of this type in the position of caret:

- InsertItem[353] (create TRVMathItemInfo[621] object, insert it using InsertItem[353])

The following method returns TRVMathItemInfo[621] object at the position of caret:

- GetCurrentItem[345].

The following methods change properties of items as editing operations:

- SetCurrentItemExtraIntPropertyEx[378];
- SetItemExtraIntPropertyExEd[383];
- SetCurrentItemExtraStrPropertyEx[379];
- SetItemExtraStrPropertyExEd[384].

Properties

Main properties:

- Text – mathematical expression in LaTeX-like format,
- DisplayInline – switches between the inline mode and the display mode.

These properties are also accessible as rveipcDisplayInline[706] and rvespcText[709] properties. Accessing them in this way allows changing them in editing operations.

Layout and appearance

This item type has the following integer properties[673] related to layout and appearance:

- *rvepSpacing* – padding (spacing between the expression image and its border);
- *rvepBorderWidth, rvepBorderColor* – width and color of a border;
- *rvepOuterHSpacing, rvepOuterVSpacing* – horizontal and vertical spacing around the border
- rvepColor – background color.

Additionally, it has the properties:

- FontSizeDouble – font size in half-points,

- FontName – font name (only mathematical fonts can be used),
- TextColor – text color.

These properties are also accessible as rveipcTextColor, rveipcFontSizeDouble[706] and rvespcFontName[709] properties. Accessing them in this way allows changing them in editing operations. If values of these properties are default, the item uses global settings specified in TRVMathDocObject[656] object.

Saving and Loading

Export to HTML[96]

Equations are saved as images (PNG, if possible).

Export to RTF and DocX

Equations are saved as images.

Export to text

Text property is saved.

1.3.22 Custom Item Types

You can create your own item types.

Overview of custom item types is beyond the scope of this help file.

See examples in **Demos\Addins**

1.4 Overview of Tables

I want to thank EQL International (the UK's leading provider of e-learning and assessment systems) and personally Dr. David G Wild.

Without them I could spend years implementing tables.

Overview of Tables

- Tables in RichView[145]
- Addressing Items in Table Cells (important information)[145]
- Table Cells[146]
- Colors and Layout of Tables[148]
- Selection in Table[150]
- Cell Editing[151]
- Cell Merging[152]
- Table Operations[153]
- Undo/redo in Tables[155]
- Export of Tables[157]
- Table Resizing[158]

Examples

- Converting table to text[159]
- Dividing table[160]

Main Classes

- TRVTableItemInfo[565] – table
- TRVTableCellData[605] – table cell

1.4.1 Tables in RichView

Table is a special kind of RichView items [64], like images, controls, breaks, etc. But tables are much more complicated, because they contain RichView document in each cell. Moreover, tables can contain other tables inside, so RichView documents can be nested on any depth.

How to Create and Insert a Table

Creating table

```
uses RVTable;
...
var table:TRVTableItemInfo[565];
...
table := TRVTableItemInfo.CreateEx[586](RowCount,ColumnCount,
  RVData);
```

where RVData is a richview document where you wish to insert table. Usually this is richview.RVData[187] or richviewedit.RVData.

How to append table to RichView[162]

```
table.ParaNo := <Paragraph Style Index>
richview.AddItem[201](<Table name>, table)
```

where <table name> is any string associated with table, <paragraph style index> is the index in the collection of paragraph styles[431]. This paragraph style is used for the table itself, not for each cells.

This method does not perform reformatting and requires explicit call of Format[217] before displaying the document.

How to insert table in RichViewEdit[316]

```
richviewedit.InsertItem[353](<Table name>, table)
```

This method inserts table in the position of caret, even if caret is in a cell of another table.

This method can return False, if document is read-only or insertion is impossible because of text protection. In this case table was freed by InsertItem method.

Table Placement

Table always occupies the whole line, no items are possible to the left or to the right of it (if you need this effect, use nested tables), including list markers[131].

The table's paragraph style defines its position: Alignment[483], LeftIndent[485], RightIndent[488] (FirstIndent[485] is ignored). The paragraph style can define border[485] around the table and background[484]. All these properties are applied to the table itself, not to its cells.

1.4.2 Addressing Items in Table Cells

Main Methods

RichViewEdit.[86] **Insert***[86] methods[86] insert item(s) in the position of caret, even if the caret is in cell of other table.

RichViewEdit.**GetCurrent***Info** and **SetCurrent***Info** methods differ from their analogs with names without "Current"

(for example GetCurrentPictureInfo[349] and GetPictureInfo[236])

Get/SetCurrent*Info** methods work with the current item (item at the position of caret) either in parent editor or in cell inplace-editor, depending on where the caret is located.

Get/Set-without-Current methods cannot work with cells (because item in cell cannot be addressed using only ItemNo parameter).

If you used TRichView version 1.3 and you want to work with tables, you should check your code and replace such methods with their "Current" alternatives, or to call them for TopLevelEditor[329].

All methods (and properties) of editor working with current or selected item(s) can work with cell inplace editors (for example, ApplyTextStyle[334], SetCurrentTag[382], CurTextStyleNo[324], etc.). Programmer does not need to know if the current item or selection is in root editor or in inplace editor.

Methods with ItemNo Parameter

Methods working with the Item-th item cannot work with cells (because item in cell cannot be addressed using only ItemNo parameter). They must be called for the document containing this item (usually, for editing operations, it's TopLevelEditor[329]).

There are some methods which do not have ItemNo parameter:

- ResizeCurrentControl[368] (alternative to ResizeControl[368])
- AdjustControlPlacement2[330](Control) (alternative to AdjustControlPlacement[330](ItemNo))
- SetCurrentItemText[380](text); (alternative to SetItemTextEd[385](ItemNo, text));
- GetCurrentItemText[347] (alternative to GetItemText[230]);

Events with ItemNo Parameter

For events without RVData[650] parameter: when events occur inside cell inplace editor, ItemNo is an index of table inside the root editor (for example OnRVMouseDown[289])

For events having both RVData and ItemNo parameter: ItemNo is an index of item in this RVData

Selection

SelectionExists[258] returns True, if something is selected inside cell, or several cells are selected in table. In this case GetSelectionBound[237] can returns -1s.

In editor, you can use TopLevelEditor.GetSelectionBounds to get information about selection in the edited cell.

Hypertext[69]

Hypertext items can be not only in main document, but in subdocuments (cells[605] or cell inplace-editor).

Use GetJumpPointLocation[231] to access hypertext item.

Hypertext links are numbered sequentially through the document, including all hyperlinks cells. This number is passed as **id** parameter in hypertext events

Checkpoints[65]

Unlike hypertext links, each subdocument (main document, table cell) has its own list of *checkpoints*. So methods for *checkpoints* enumeration cannot find *checkpoints* in table cell (if called for the main document).

Page Breaks

Mandatory page breaks inside tables are not supported yet.

1.4.3 Table Cells

Table Cells

The main property of a table is Cells[574][<Row>,<Col>].

<Row>,<Col> are indices of a row and a column, from 0.

Attention: The standard Delphi grids use Cells[<Col>,<Row>] instead!

The type of Cell[r,c] is TRVTableCellData[605]. Each RichView control has a similar class inside (richview.RVData[187]). Cell and richview.RVData are descendants from the same class[650] designed to store formatted RichView document.

Cells also have additional table-related properties (discussed below).

Table Rows

Tables have another property: Rows[580].

It is a list of table rows. Each row is a list of cells.

Each row has the same number of cells (**important!**)

So:

- number of rows in table = table.Rows.Count;
- number of columns in table = table.Rows[0].Count (each row has the same number of cells as the 0-th)
- table.Cells[r,c] is the same as table.Rows[r][c].

Accessing and Modifying Cells

In editor, you need no special processing to access cells.

Methods working with current/selected item(s) do it automatically (see more info[145]).

Cells have the most of the methods which RichView (not RichViewEdit!) has.

For example:

```
with table.Cells[r,c] do begin
  Clear;
  AddNL('hello!', 0, 0);
end;
```

Read more about methods of cell[614].

Initially every cell contains one empty line of the 0-th text and paragraph style.

Cells do not have editor-style methods. RichViewEdit creates special inplace editor to edit cell.

Read more about cell editing[151].

1.4.4 Colors and Layout of Tables

Table Colors

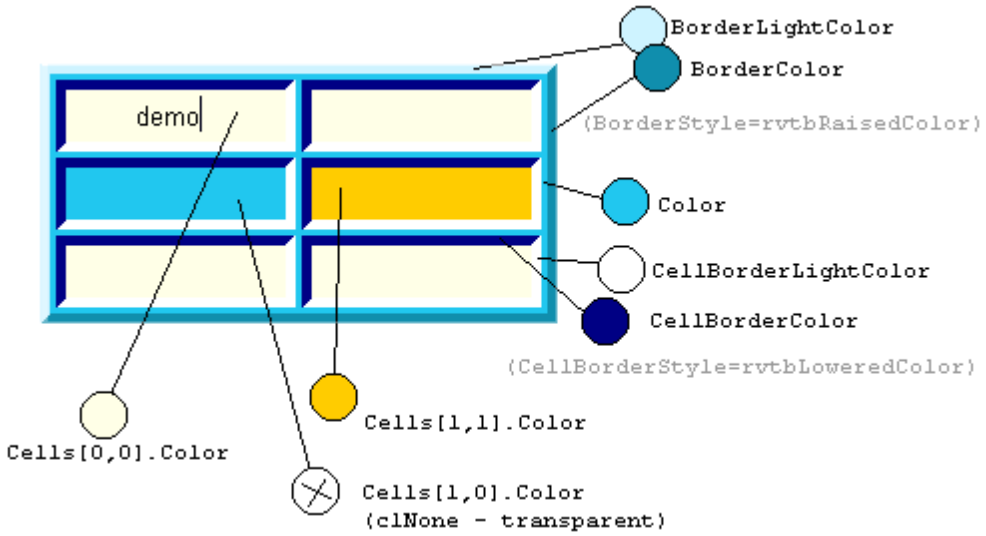


Table Colors

The main color properties of tables (Color[575], BorderColor[569], BorderLightColor[569], CellBorderColor[571], CellBorderLightColor[571]) are shown on the figure above.

There are some more color properties related to rules (lines between cells), see below[149].

Table can be transparent, if its Color[575] is set to *clNone*, or semitransparent[577].

Cells can be individually colored[608], with specified opacity[609]. You can specify default colors for groups of rows and columns[583].

Borders can be hidden, if their widths are set to 0 (see BorderWidth[571] and CellBorderWidth[572]). These are default values.

Cell borders can be 3d, like on this figure, or flat (see BorderStyle[570] and CellBorderStyle[572]).

You can hide some sides of borders of some cells (see cells' VisibleBorders[613]) and set individual color for cell border (cell.BorderColor[607] and cell.BorderLightColor[608]).

When a table with invisible border is inserted in TRichViewEdit[316], such borders are shown as dotted gray lines (see also table Options[577]).

Background Images

Tables and cells can have background images.

See properties of table: BackgroundImage[567], BackgroundStyle[568].

See properties of cell: BackgroundImage[605], BackgroundStyle[606].

Borders and Spacing

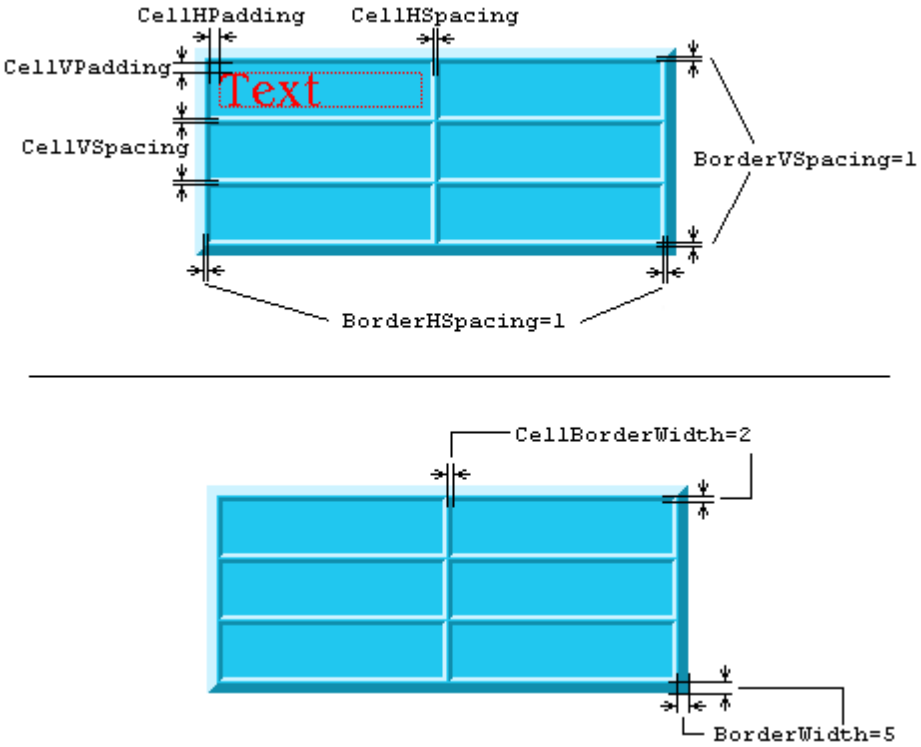


Table Borders

The figure above shows properties affecting the table layout:

- CellHSpacing[573], CellVSpacing[574],
- BorderHSpacing[569], BorderVSpacing[570],
- BorderWidth[571],
- CellBorderWidth[572],
- CellHPadding[573], CellVPadding[574].

(The dotted red line around the cell contents is imaginary).

Rules

Rules are horizontal and/or vertical lines between cells, and may be also between table borders and outermost cells.

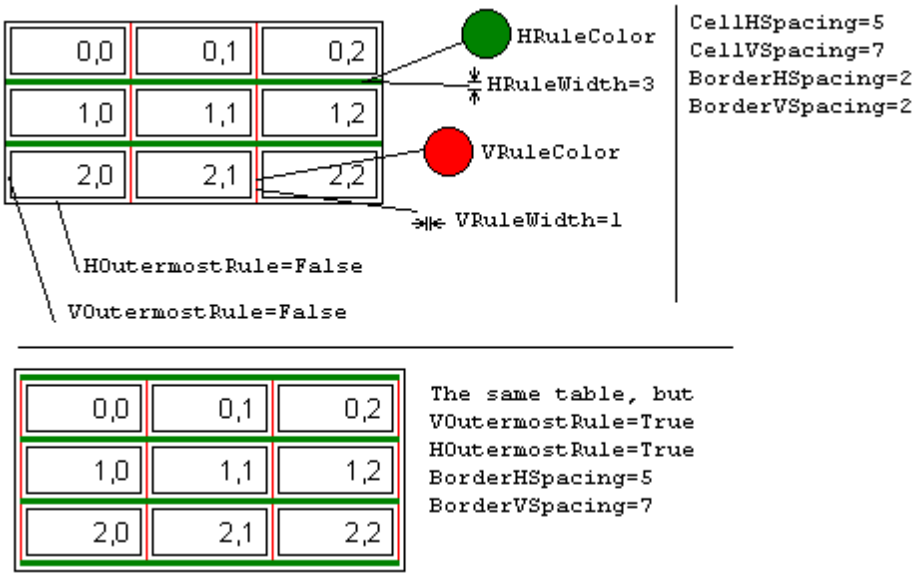


Table Rules

By default rules are invisible.

Rules are defined with properties: HRuleWidth[577], HRuleColor[576], HOutermostRule[576], VRuleWidth[582], VRuleColor[582], VOutermostRule[582].

Tip: How to make 1 pixel border

RichView tables have HTML-style layout, so making one-pixel width borders is a trick.

Even if you set CellBorderWidth[572]=1, CellHSpacing[573]=CellVSpacing[574]=0, 1-pixel width borders of adjacent cells will be visually combined in 2-pixel width borders.

Recommended solution:

setting CellBorderWidth[572]=1, CellHSpacing[573]=CellVSpacing[574]=-1. These values are specially supported by SaveHTMLEx[251] (and SaveHTML[250] with *rvsoForceNonTextCSS* in the Options parameter).

This example assumes RVStyle.Units[437]=*rvstuPixels*. For rvstuTwips, use 15 instead of 1, and -15 instead of -1.

Other solutions (not recommended):

- hiding cells border and using rules; disadvantages: rules cannot be saved in HTML and RTF;
- using VisibleBorders[613] to hide one of adjacent border sides

1.4.5 Selection in Table

Types of Selection

Table can be:

- not selected or
- selected completely (as a part of larger selection) or
- selected partially (multicell selection) or
- some part of cell content (in cell or in inplace editor) selected.

RichView does not support selection started somewhere else and finished at the middle of a table. Selection containing several items can contain tables only completely (or it does not contain them).

Any method working with selection must be called only when the document is formatted.

Tables in Larger Selection

To determine if the table is selected completely, as a part of larger selection, use RichView.GetSelectionBounds[237] (and compare its results with GetItemNo[229] for table).

Multicell Selection

Table can be selected partially: selection can starts on one cell of table, and ends on another cell.

You can determine if the table selected partially using the methods TRVTableItemInfo.GetSelectionBounds[591] or GetNormalizedSelectionBounds[590] (the latter is more convenient).

You can select some cells in table using TRVTableItemInfo.Select[595] and remove the selection using TRVTableItemInfo.Deselect[588].

You can select some rows or columns completely: TRVTableItemInfo.SelectRows[596] and SelectCols[595].

You can determine if the given cell is selected or not with the method IsCellSelected[593].

Selection Inside Cell

You can get inplace-editor and position for currently edited cell: GetEditedCell[590].

See also: TRichViewEdit TopLevelEditor[329].

Before making selection inside cells, you need to activate inplace editor for it (TRichView does not have inplace editors, but the same code must be called in TRichView too, not only in TRichViewEdit).

Example 1 (selecting table.Cells[0,0]):

```
table.EditCell(0,0);
table.Cells[0,0].GetRVData.SelectAll;
```

Example 2 (does the same)

```
uses CRVData, CRVFData;
var RVData: TCustomRVData;
...
RVData := table.Cells[0,0];
RVData := RVData.Edit;
TCustomRVFormattedData(RVData).SelectAll;
```

The method in the second example (calling RVData := RVData.Edit, then selecting) can be used for any type of RVData, not only for table cells.

Editing Operations

Some RichViewEdit[316] methods can be applied to the selected cells (regardless the type of selection):

- ApplyTextStyle[334]
- ApplyParaStyle[331]
- ApplyStyleConversion[333]
- ApplyParaStyleConversion[332]
- DeleteSelection[340].

1.4.6 Editing Cells

Editing Cells

Then the user clicks some cell in table with the mouse, or enter to some cell using keyboard, RichViewEdit creates a special editor for this cell. This editor is referred in this documentation as cell inplace editor.

(this editor will be created only if *rvtoEditing* is in table Options[577])

This editor does not have borders and located in place of cell, creating an illusion of editing directly in main RichViewEdit.

Inplace editing is possible only when table was inserted in TRichViewEdit[316], not in TRichView[162].

When inplace editor is initiated, RichViewEdit moves all content of the cell to this editor. When editor is destroyed, RichViewEdit moves modified data back to the cell (this procedure is transparent for user).

So if you wish to access directly to cell contents[614], use methods of cell's GetRVData[613] instead of methods of the cell itself.

You can initiate editing of cell from code: table.EditCell[589] (see the example in the topic about selection in tables[150]) or Cell.Edit[651].

Position of edited cell can be returned by method GetEditedCell[590] and GetNormalizedSelectionBounds[590].

Limitations

This mechanism of cell editing adds some limitations:

You must not destroy inplace editor from inside the following events: OnKeyDown, OnKeyUp, OnKeyPress, OnRVMouseMove[290], OnStartDrag, OnDragOver.

Inplace editor can be destroyed by calling RichViewEdit.Clear[210], or by the methods placing caret/selection outside the editing cell (RichViewEdit.SetSelectionBounds[271], table.EditCell[589], etc.). You must avoid calling these methods in these events.

If you need to perform such operation, use the following technique:

- from the event, post a window message to the form (PostMessage)
- in the handler procedure for this message, perform this operation.

Example 1 (reloading on double click)

Define some constant > WM_USER:

```
const WM_RELOADDOC = WM_USER+5;
```

In the form's declaration:

```
procedure WMReloadDoc(var Msg: TMessage); message WM_RELOADDOC;
```

Implementation:

```
procedure TForm1.WMReloadDoc(var Msg: TMessage);
begin
  { code for loading here }
end;
```

Double click:

```
procedure TForm3.RichViewEdit1DblClick(Sender: TObject);
begin
  PostMessage(Handle, WM_RELOADDOC, 0, 0);
end;
```

Example 2:

- **Demos\Delphi\Assorted\Graphics\DragImg\ or**
- **Demos\CBuilder\Assorted\Graphics\DragImg**

1.4.7 Cell Merging

Cells have read-only ColSpan[608] and RowSpan[612] properties, similar to html <td colspan rowspan>, 1 by default.

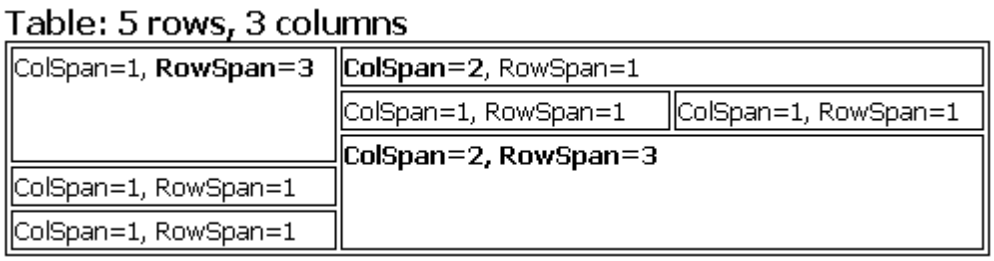
If Cells[574][r,c].ColSpan>1, this cell spans over (ColSpan-1) cells to the right:

Cell[r,c], Cell[r,c+1], ..., Cell[r,c+Cell[r,c].ColSpan-1].

In this case, all overlapped cells (Cell[r,c+1], ..., Cell[r,c+Cell[r,c].ColSpan-1]) are destroyed and equal to nil.

Number of cells in the row is still equal to the number of cells in all other rows, even with merging.

The same is with RowSpan: cell with RowSpan>1 spans over (RowSpan-1) cells below.



ColSpan and RowSpan example

Example: iterating through all cells of table:

```
for r := 0 to table.Rows[580].Count-1 do
 for c := 0 to table.Rows[580][r].Count-1 do
   if table.Cells[574][r,c]<>nil then
     table.Cells[574][r,c].Color[608] := clRed;
// ( Cells[r,c]=nil if it is overlapped by the top left cell)
```

You can get the top left cell which overlaps given cell with table.Rows[580].GetMainCell method:

```
function GetMainCell[617](ARow,ACol: Integer;
  var MRow, MCol: Integer): TRVTableCellData;
```

Methods for merging cells:

- MergeCells[594],
- MergeSelectedCells[594]

Inverse operation:

- UnmergeCells[601],
- UnmergeSelectedCells[601]

Splitting cells:

- SplitSelectedCellsHorizontally[600],
- SplitSelectedCellsVertically[601]

1.4.8 Table Operations

Operations on Selection[150]

Inserting rows and columns:

- InsertColsLeft[592],
- InsertColsRight[592],
- InsertRowsAbove[593],
- InsertRowsBelow[593].

Deleting rows and columns:

- DeleteSelectedCols[588],
- DeleteSelectedRows[588],

Cell merging and unmerging[152]:

- MergeSelectedCells[594] (see also CanMergeSelectedCells[585]),
- UnmergeSelectedCells[601],
- SplitSelectedCellsHorizontally[600],
- SplitSelectedCellsVertically[601]

Other Operations

Inserting rows and columns:

- InsertCols [591],
- InsertRows [592].

Deleting rows and columns:

- DeleteCols [586],
- DeleteRows [587].

Cell merging and unmerging [152]**:**

- MergeCells [594] (also CanMergeCells [584])
- UnmergeCells [601],
- DeleteEmptyRows [587], DeleteEmptyCols [587] (use after cell merging).

Assigning Table Properties (As Editing Operations)

Methods for assigning table properties:

- SetTableVisibleBorders [600].

Methods for assigning row properties:

- SetRowVAlign, SetRowKeepTogether, SetRowPageBreakBefore [600].

Methods for assigning cell properties:

- SetCellColor [598], SetCellBorderColor [597], SetCellBorderLightColor [598];
- SetCellVAlign [599];
- SetCellRotation [599];
- SetCellTag [599];
- SetCellBestWidth [597], SetCellBestHeight [597];
- SetCellBackgroundImage [596], SetCellBackgroundImageFileName [596], SetCellBackgroundStyle [597];
- SetCellHint [598];
- SetCellVisibleBorders [600];
- SetCellOpacity [598];
- SetCellOptions [599].

Operations in Viewer and Editor

If the operations above are performed before inserting the table, no addition actions are required.

If these operations are performed on table inserted in TRichView [162], you need to call Format [217] method after to update the document view.

If these operations are performed on table inserted in TRichViewEdit [316], a special sequence of steps is required:

1. checking RichViewEdit.CanChange [336] (especially important for data-aware versions of components);
2. obtaining table object and its position in editor;
   (position is defined as item index + editor where this table is inserted (can be root editor or cell inplace-editor, referred below as rve)
3. calling rve.BeginItemModify [335] for table
4. performing operations
5. calling rve.EndItemModify [341]
6. calling rve.Change [337]

The example is below.

You can group several actions so that they will be undone/redone as whole, using SetUndoGroupMode [387]

In order to perform operation(s) on table, you need to get table object.You can do it using method

```
function TRichView.GetItem[225](ItemNo: Integer): TCustomRVItemInfo;
```

TCustomRVItemInfo [563] is an ancestor class for all items of RichView, including table (TRVTableItemInfo [565]). This method can be used for item of any type, so you need to check if it is a table (using "is" operator, or checking RichView.GetItemStyle [229](ItemNo)=rvsTable)

But usually, when you need to perform operations on the item at the position of caret in editor, you do not know if the current item is in "root" editor, or inside cell, or inside cell of table inside other cell, and so on.

In any case, you can get item in position of caret with method

```
function TRichViewEdit.GetCurrentItem[345]: TCustomRVItemInfo;
```

But even if the caret is inside table, the current item will be not a table, but some other item in cell-inplace editor!

The problem can be solved with the method

```
function TRichViewEdit.GetCurrentItemEx[345](
  RequiredClass: TCustomRVItemInfoClass;
  var ItemRichViewEdit: TCustomRichViewEdit;
  var Item: TCustomRVItemInfo): Boolean;
```

In this method,

- **RequiredClass** – set it to TRVTableItemInfo (for C++Builder: __classid(TRVTableItemInfo)).
- **Return value**: True if there is item of the given class at the position of caret (i.e. the caret is to the left or to the right of table), or if the caret is inside item the given class (i.e. the caret is in table cell).
- **Item** receives the top-level item of the given class (i.e. table).
- **ItemRichViewEdit** receives the parent editor for **Item** (this **Item** is in this editor).

This editor can be "root" RichViewEdit or cell inplace editor.

Example

```
// MyRichViewEdit:TRichViewEdit is an editor
// placed on the form at design time.
// Note: the most of operations are performed in
// rve (editor returned by GetCurrentItemEx),
// not in MyRichViewEdit.
var item: TCustomRVItemInfo[563];
    table: TRVTableItemInfo[565];
    Data: Integer;
    rve: TCustomRichViewEdit[316];
    ItemNo: Integer;
begin
  if not MyRichViewEdit.CanChange[336] or
     not RichViewEdit1.GetCurrentItemEx[345](TRVTableItemInfo, rve, item) then
    exit;
  table := TRVTableItemInfo(item);
  ItemNo := rve.GetItemNo[229](table);
  rve.BeginItemModify[335](ItemNo, Data);
  // performing some operation, for example
  // table.InsertRowsBelow(1);
  // or table.CellPadding := 10
  rve.EndItemModify[341](ItemNo, Data);
  rve.Change[337];
end;
```

1.4.9 Undo/redo in Tables

Undo and Redo

All table operations (adding/deleting rows, merging/unmerging, assignment of table properties) can be undone/redone if table was inserted into RichViewEdit[316].

Operations will not be undone if:

- table was created, but not inserted yet;
- table was inserted in RichView[162];

- table was inserted in RichViewEdit with UndoLimit[329]=0.

Example 1

Two operations: (inserting table and changing its color) will be undone in two steps

```
table := TRVTableItemInfo.CreateEx(4,3, RichViewEdit1.RVData);
if RichViewEdit1.InsertItem('', table) then
  table.Color := clRed;
```

Example 2

One operation (inserting red table) will be undone in one step

```
table := TRVTableItemInfo.CreateEx(4,3, RichViewEdit1.RVData);
table.Color := clRed; // table is not inserted yet
RichViewEdit1.InsertItem('', table);
```

About Using Non-Editor-Style Operations in Editor for Modifying Documents

(AddNL[203], for example)

These methods must not be mixed with editor-style methods (such as InsertText[360]) because they will conflict with undo mechanism. Result of undo-redo operations is unpredictable if non-editor operations were used after editor-style one.

This rule relates not only to contents of the main document, but to cells also.

There are some cases when non-editor-style methods may be useful in editor:

- After Clear[210] method, before any editor-style methods (for generating a document before editing). Clear method clears undo and redo buffers.
- In table cells before inserting a table.
- if UndoLimit[329] of editor = 0 (undo mechanism turned off). But Format[217] must be called before calling any editor-style method.
- After ClearUndo[339] method. But Format[217] must be called before calling any editor-style method.

Undo of Assignment to Properties

Assigning values to cell properties (and to VAlign[616] property of rows) will not be undone.

But there are some special methods.

```
table.Cells[574][r,c].Color[608] := clRed; // will not be undone
table.SetCellColor[598](clRed, r,c);  // will be undone.
```

(the complete list of these methods is below)

This was made because of efficiency reasons.

These methods do not update editor view. Use them together with BeginItemModify[335]/EndItemModify[341].

For example

```
var item: TCustomRVItemInfo;
    table: TRVTableItemInfo;
    Data: Integer;
    rve: TCustomRichViewEdit;
    ItemNo: Integer;
begin
  if not RichViewEdit1.CanChange[336] or
     not RichViewEdit1.GetCurrentItemEx[345](TRVTableItemInfo, rve, item) then
    exit;
  table := TRVTableItemInfo(item);
  ItemNo := rve.GetItemNo[229](table);
  rve.BeginItemModify[335](ItemNo, Data);
  table.SetRowVAlign(rvcBottom, 1);
  table.SetCellColor(clRed,1,1);
```

```
  rve.EndItemModify(341)(ItemNo, Data);
  rve.Change(337);
end;
```

Assignments to all table properties can be undone (but still the sequence BeginItemModify-operations-EndItemModify-Change is required). The only exception is VisibleBorders(581) property, use SetTableVisibleBorders(600) if you want to implement an undoable operation.

Direct assignment to cells and rows properties cannot be undone, so use the methods below to implement an undoable operation.

Methods for assigning table properties:

- SetTableVisibleBorders(600).

Methods for assigning row properties:

- SetRowVAlign(600).

Methods for assigning cell properties:

- SetCellColor(598), SetCellBorderColor(597), SetCellBorderLightColor(598);
- SetCellVAlign(599);
- SetCellBestWidth(597), SetCellBestHeight(597);
- SetCellBackgroundImage(596), SetCellBackgroundImageFileName(596), SetCellBackgroundStyle(597);
- SetCellHint(598);
- SetCellVisibleBorders(600).

1.4.10 Export of Tables

Tables can be exported with document in the following formats:

- text (including Unicode),
- HTML,
- RTF.

Text Export

Tables have the properties:

- TextRowSeparator(581) – string to write after each row;
- TextColSeparator(581) – string to write between cells.

By default, the both of them are equal to CR+LF (#13#10).

MS Word 2000+ saves tables as text in the same way.

You can set TextColSeparator to TAB character (#09), if you wish, but do not forget that there can be CR+LFs inside cells.

RTF and DocX

RichView saves tables in RTF according to the MS Word 2000+ standard. MS Word 2000 introduced many new RTF keywords to store HTML-like formatting.

Similar keywords are saved in DocX files.

MS Word supports two algorithms of calculating widths of columns: fixed and fit-by-content. Initial widths of table columns specified in RTF/DocX (called "table grid") are adjusted by MS Word according to the specified algorithm. The algorithm takes contents and widths of cells into account.

By default, TRichView saves tables as fixed tables. Generally, MS Word's fixed tables look closer to TRichView tables; however, there are some problems. For example, if we save exact values of Cells(574)[r,c].BestWidth(568) (specified in twips or pixels), MS Word sets width of these cells equal to these values exactly, so a wide content may be truncated. What's why TRichView does not save these values to RTF/DocX. So, after saving, cells having BestWidth(568) = 0 and BestWidth(568) > 0 are undistinguished, their width is taken from a table grid (which is

saved according to table size in TRichView). You can turn on saving cells' BestWidth[568] by including *rvtoRTFSaveCellPixelBestWidth* to table.Options[577]; however, this is not recommended for fixed tables.

If you include *rvtoRTFAllowAutofit* in table.Options[577], this table will be saved as fit-by-content table (but only if table.BestWidth[568]=0). For such tables, you can safely include *rvtoRTFSaveCellPixelBestWidth* to table.Options[577].

When saving a table grid to RTF/DocX, TRichView uses the current table size. What's why TRichView must be formatted before saving to RTF/DocX, if this document contains tables.

HTML

Some properties can be saved only with CSS (with SaveHTMLEx[251], or SaveHTML[250] with *rvsoForceNonTextCSS* in Options parameter). See comments for table properties.

See Also...

RichView methods for saving and loading[92]

1.4.11 Table Resizing

Table Layout Algorithm

Tables do not have column widths and row heights specified explicitly.

Widths are calculated basing on:

- minimal width required to display document inside cell (if *rvtoIgnoreContentWidth* is not in table.Options[577])
- table.BestWidth[568]
- cells BestWidth[607]

Heights are calculated basing on:

- heights of cells content (if *rvtoIgnoreContentHeight* is not in table.Options[577])
- cells BestHeight[606]

Rules for calculations table width:

1. cells always have widths large enough to contain the widest of their item (if *rvtoIgnoreContentWidth* is not in table.Options[577]);
2. if table.BestWidth[568]<>0, it defines width of the table (but table can be wider because of the rule 1)
3. if table.BestWidth[568]=0 then
    - if all columns have cells with BestWidth[607]s in TRVStyleUnits[691], i.e. pixels or twips (and do not have cells with BestWidth[607]s in %), width of table is calculated basing on cells BestWidth[607]s and minimal widths required to show cells contents without overlapping (in this case mouse resizing works the most similar to MS Word)
    - otherwise, width of table is the same as with BestWidth[568]=-100 (100%) (in this version RichView cannot calculate width of table basing on optimal widths of cells)

Rules for calculating widths of cells are quite complicated.

The most important rules (when widths of cells are conflicting):

- widths of cells to the left have higher priority than widths of cells to the right;
- widths of cells in % have higher priority than widths in pixels/twips.

Resizing with Mouse

Resizing with mouse is possible only in editor.

If you include *rvtoRowSizing* and *rvtoColSizing* in table.Options[577], users will not be able to resize it.

When user resizes table with mouse, he/she actually changes values of BestWidth[607] or BestHeight[606] properties of cells.

When resizing columns with mouse, and columns to the left and to the right do not have cells with BestWidth[607] s in %, RichViewEdit adjusts BestWidth[607]s of cells of column to the right and to the left, trying to keep their total width unchanged.

When resizing with pressed **Shift** key, RichView sets only BestWidth[607]s of the column to the left.

1.4.12 Example: table to text

This example "removes" the table at the caret position without removing its content.

This is an editing operation, it can be undone[88] by the user.

```
procedure ConvertTableToText(RichViewEdit: TCustomRichViewEdit[316]);
var table: TRVTableItemInfo[565];
    rv: TRichView[162];
    rve: TCustomRichViewEdit[316];
    Stream: TMemoryStream;
    ItemNo,r,c: Integer;
begin
  if not RichViewEdit.GetCurrentItemEx[345](TRVTableItemInfo[565], rve, TCustomRVItemInfo[563](ta
    exit;
  rv := TRichView[162].Create[214](nil);
  try
    rv.Visible := False;
    rv.Parent := RichViewEdit.Parent;
    rv.Style[193] := RichViewEdit.Style[193];
    rv.RVFTextStylesReadMode[191] := rvf_sIgnore;
    rv.RVFParaStylesReadMode[190] := rvf_sIgnore;
    for r := 0 to table.RowCount[580]-1 do
      for c := 0 to table.ColCount[575]-1 do
        if table.Cells[574][r,c]<>nil then begin
          Stream := TMemoryStream.Create;
          try
            table.Cells[574][r,c].GetRVData[613].SaveRVFToStream[254](Stream, False, clNone, nil,
            Stream.Position := 0;
            rv.InsertRVFFromStream[240](Stream, rv.ItemCount[180])
          finally
            Stream.Free;
          end;
        end;
    ItemNo := table.GetMyItemNo; // like rve.GetItemNo[229](table), but faster
    Stream := TMemoryStream.Create;
    try
      rv.SaveRVFToStream[254](Stream, False);
      Stream.Position := 0;
      rve.SetSelectionBounds[271](ItemNo, 0, ItemNo, 1);
      rve.InsertRVFFromStreamEd[357](Stream);
    finally
      Stream.Free;
    end;
  finally
    rv.Free;
  end;
end;
```

The same for C++Builder:

```
void ConvertTableToText(TCustomRichViewEdit* RichViewEdit)
{
  TCustomRVItemInfo   *item;
  TCustomRichViewEdit *rve;
  if (! RichViewEdit->GetCurrentItemEx(__classid(TRVTableItemInfo), rve, item))
    return;
  TRVTableItemInfo*table = (TRVTableItemInfo*)item;
  TRichView* rv = new TRichView((TComponent*)NULL);
  rv->Visible = false;
  rv->Parent  = RichViewEdit->Parent;
  rv->Style   = RichViewEdit->Style;
  rv->RVFTextStylesReadMode = rvf_sIgnore;
  rv->RVFParaStylesReadMode = rvf_sIgnore;
  for (int r=0; r<table->RowCount; r++)
    for (int c=0; c<table->ColCount; c++)
      if (table->Cells[r][c])
      {
        TMemoryStream*Stream = new TMemoryStream;
        table->Cells[r][c]->GetRVData()->SaveRVFToStream(Stream, false, clNone, NULL, NULL)
        Stream->Position = 0;
        rv->InsertRVFFromStream(Stream, rv->ItemCount);
        delete Stream;
      }
  int ItemNo = table->GetMyItemNo();
  TMemoryStream*Stream = new TMemoryStream;
  rv->SaveRVFToStream(Stream, false);
  Stream->Position = 0;
  rve->SetSelectionBounds(ItemNo, 0, ItemNo, 1);
  rve->InsertRVFFromStreamEd(Stream);
  delete Stream;
  delete rv;
}
```

1.4.13 Example: dividing table

```
function SplitTable(rve: TCustomRichViewEdit[316]; table: TRVTableItemInfo[565];
  RowCountInFirstTable: Integer; PageBreak: Boolean): Boolean;
var r, c, mr, mc: Integer;
    newtable: TRVTableItemInfo[565];
    Stream: TMemoryStream;
begin
  Result := False;
  if RowCountInFirstTable>=table.RowCount[580] then
    exit;
  for c := 0 to table.ColCount[575]-1 do
    if table.Cells[574][RowCountInFirstTable, c] = nil then begin
      table.Rows[580].GetMainCell[617](RowCountInFirstTable, c, mr, mc);
      if mr<RowCountInFirstTable then
        exit;
    end;
  Stream := TMemoryStream.Create;
  table.SaveRowsToStream[595](Stream, RowCountInFirstTable,
    table.RowCount[580]-RowCountInFirstTable);
  Stream.Position := 0;
  newtable := TRVTableItemInfo[565].CreateEx(0, 0, rve.RVData);
```

```
  newtable.LoadFromStream(Stream);
  if PageBreak then
    newtable.PageBreakBefore := True;
  Stream.Free;
  rve.BeginUndoGroup(rvutModifyItem);
  rve.SetUndoGroupMode(True);
  try
    table.DeleteRows(RowCountInFirstTable, table.RowCount-RowCountInFirstTable, True
    // table.GetMyItemNo is the same as rve.GetItemNo(table), but faster
    rve.SetSelectionBounds(table.GetMyItemNo, 1, table.GetMyItemNo, 1);
    rve.InsertItem('', newtable);
  finally
    rve.SetUndoGroupMode(False);
  end;
  Result := True;
end;
```

Parameters:

rve – editor directly containing **table**;

table – the table to divide.

These two parameters are usually a result of RichViewEdit1.GetCurrentItemEx.

The table will be divided into two tables.

The first resulting table will have row count = **RowCountInFirstTable**, the second table will have row count = table.RowCount-**RowCountInFirstTable**.

If **PageBreak** parameter is *True*, a page break will be inserted between tables.

Return value: *True* if the table was divided.

This is an editing operation, it can be undone and redone by the user.

1.5 Components

Components

Icon	Class Name	Description
	TRVStyle	Component for defining visual appearance of RichView, RichViewEdit, DBRichView, DBRichViewEdit. It contains some properties which can be used by several RichViews.
	TRichView	Component for displaying hypertext documents (containing pictures, controls, tables)
	TRichViewEdit	Component for editing hypertext documents (containing pictures, controls, tables)
	TDBRichView	Data-aware version of RichView
	TDBRichViewEdit	Data-aware version of RichViewEdit
	TRVPrint	Component for printing documents contained in the components above
	TRVPrintPreview	Component allowing to show document as it will be printed by RVPrint

	TRVReportHelper 540	Component for drawing documents on different canvas (screen, printer, image, etc.)
	TRVOfficeConverter 532	Component allowing to use Microsoft Office text converters (import and export in different file formats)

1.5.1 Visual Components

1.5.1.1 TRichView

TRichView is a control for displaying documents with pictures, tables, Delphi controls and hyperlinks.

Unit RichView;

Syntax

```
TCustomRichView = class(TRVScroller 544)
TRichView = class(TCustomRichView)
```

TRichView does not display its content if it is not linked to TRVStyle component via Style 193 property.

The most important property settings can be done in the component editor 168 for TRichView. In Delphi (C++Builder) IDE, right click the RichView object on the form, choose "Settings" in the context menu.

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller 544
TCustomRichView

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

- General 162
- Background 162
- Appending Items 163
- Hypertext 163
- Style templates 163
- Scrolling, margins, sizes 163
- Selection and Clipboard 164
- Saving and loading as RVF 164
- Other methods for saving and loading 165
- Information about items 165
- Modifying items 166
- Checkpoints 166
- Mouse events 166
- Live spelling check 167
- Picture animation 167
- Working in palette mode 167
- Other properties, methods and events 167

General properties

- Style 193 – link to RVStyle component defining a lot of properties which can be used in several RichViews;
- Options 182 – some important options;
- ItemCount 180 – number of items in the document.

Properties defining background

- BackgroundBitmap 169;
- BackgroundStyle 170;

- Color[173].

Appending items (Add*** methods)

Main methods for appending items:

- AddNL[203] adds one text item;
- AddBreak[196] adds break (horizontal line);
- AddBullet[197] adds bullet (image from ImageList);
- AddHotspot[200] adds hotspot (image from ImageList - hypertext link);
- AddPicture[204] adds picture;
- AddHotPicture[199] adds picture - hypertext link;
- AddControl[198] adds Delphi/C++Builder control;
- AddTab[205] adds tabulator;
- AddItem[201] – general method for appending items (usually used for tables[145]);
- also AddCheckpoint[202] – adds checkpoint which will be associated with next added item.

Additional methods for appending items:

- Add[203] adds one text item in last paragraph;
- AddFmt[198] adds one formatted string assembled from a Pascal format string and an array arguments;
- AddTextNL, -A -W[206] add several text items from text, separated by CR+LF characters;
- also Unicode-enabled methods[162].

Also

- AppendFrom[207] copies content from another RichView;
- InsertRVFFromStream[240],
- AppendRVFFromStream[208].

See also

- Clear[210] clears document;
- SetAddParagraphMode[259];
- Format[217], FormatTail[218] format the document;
- Reformat[248] reformats the document.

Hypertext

- event OnJump[282] occurs when user clicks hyperlink;
- event OnRVMouseMove[290] occurs when mouse pointer moves over hyperlink;
- events OnWriteHyperlink[302] and OnReadHyperlink[284] occur when document is saved or loaded as RTF, DocX or HTML.
- property FirstJumpNo[179] – use to set hypertext id for the first hypertext item;
- GetJumpPointLocation[231] should be used instead of GetJumpPointItemNo in applications working with tables[145];
- GetJumpPointY[232] returns Y coordinate of item with the specified hypertext id.

Style templates

Properties:

- UseStyleTemplates[195] allows or disallows using Style[193].StyleTemplates[435].
- StyleTemplateInsertMode[193] specifies how style templates are used when inserting RTF or RVF content.

Events:

- OnStyleTemplatesChange[302] occurs then the collection of style templates has been modified.

Scrolling, margins and sizes

Properties:

- LeftMargin[180], RightMargin[185], TopMargin[194], BottomMargin[171] – margins;
- MaxTextWidth[181] – maximal width for word wrapping;
- MinTextWidth[181] – minimal width for word wrapping (minimal width of scrollable area, not including margins);
- DocumentHeight[177] – height of document (height of scrollable area);

- VAlign 195 – vertical alignment of document;
- VScrollVisible 547 – hides/shows vertical scrollbar;
- HScrollVisible 545 – hides/shows horizontal scrollbar;
- VSmallStep 196 – number of pixels in one step of vertical scrollbar;
- FirstItemVisible 179 – the first visible item;
- LastItemVisible 180 – the last visible item;
- MaxLength 181 – maximal count of characters per line;
- WordWrap 196 allows to disable word wrapping;
- Tracking 546 – whether RichView repaints while its scrollbar thumb tab is dragged with the mouse.

Methods:

- ClientToDocument 211 converts client coordinates to document coordinates

Events:

- OnVScrolled 549 occurs when document is scrolled vertically
- OnHScrolled 548 occurs when document is scrolled horizontally

Also:

- property CPEventKind 173
- event OnCheckPointVisible 274
- method GetItemCoords 226

Methods for working with selection and the Clipboard

- SelectionExists 258 returns "is something selected?"l

- Copy 212 copies selection to the Clipboard in all available formatsl
- CopyDef 212 copies selection to the Clipboard using formats specified in Options 182;
- CopyImage copies selected image to the Clipboard;
- CopyTextA 213 copies selection as text to the Clipboard;
- CopyTextW 213 copies selection to the Clipboard as Unicode text;
- CopyRVF 213 copies selection as RVF to the Clipboard;
- CopyRTF 213 copies selection as RTF (Rich Text Format);

- GetSelectedImage 237 returns selected image
- GetSelText 238 returns selection as text
- GetSelTextW 238 returns selection as Unicode string;

- GetOffsBeforeItem 235, GetOffsAfterItem 235 return offset before/after the given item; useful with SetSelectionBounds 271;

- SelectAll 257 selects the whole document for copying to the Clipboard;
- SetSelectionBounds 271 selects the specified items;
- SelectWordAt 258 selects the word at (X,Y);
- SelectControl 257 selects a visual control;
- Deselect 216 clears selection (not contents, but bounds);

- GetSelectionBounds 237 returns bounds of selection;
- event OnSelect 301 occurs when selection bounds are changed;
- event OnCopy 276 allows to copy selection to the Clipboard in custom formats;
- property SelectionHandlesVisible 192 shows or hides selection handles for a touch screen.

Properties, methods and events for loading and saving RVF

Properties:

- RVFOptions 188 – options for saving and loading RVF;
- RVFWarnings 191 describes warnings or errors occurred during the last reading of RVF.

Events:

- OnRVFControlNeeded[288] requests inserted control when reading RVF, if RVF was saved without controls' "bodies";
- OnRVFImageListNeeded[288] requests ImageList when reading bullet[128] or hotspot[129] from RVF;
- OnRVFPictureNeeded[289] requests image when reading RVF, if RVF was saved without images' "bodies".

Methods:

- SaveRVF[254] saves document or selection as RVF to the file;
- SaveRVFToStream[254] saves document or selection as RVF to the stream;
- LoadRVF[245] loads RVF from the file;
- LoadRVFFromStream[245] loads RVF from the stream;
- CopyRVF[213] copies selection as RVF to the Clipboard;
- InsertRVFFromStream[240] inserts RVF from the stream;
- AppendRVFFromStream[208] appends RVF from the stream using the specified paragraph style for the first RVF item.

Also:

- methods Copy[212], CopyDef[212].

Other saving and loading

Methods:

- SetHeader[264], SetFooter[263] specify documents for using as headers and footers; they are used to load and save headers and footers to files (RVF, RTF, DocX)
- LoadFromStream[243] loads document in RVF, RTF or text formats;
- LoadText[246] appends text form the file;
- LoadTextW[246] appends text from the file in Unicode encoding;
- LoadRTF[243], LoadRTFFromStream[244] appends the file or the stream in RTF (Rich Text Format);
- SaveHTML[250], SaveHTMLToStream[252] save the document in an HTML file/stream using a basic HTML syntax;
- SaveHTMLEx[251], SaveHTMLToStreamEx[252] save the document in an HTML or XHTML file/stream using CSS;
- SaveText[255], SaveTextToStream[255] save the document in a file/stream as ANSI text;
- SaveTextW[255], SaveTextToStreamW[255] saves the document in a file/stream as a Unicode text;
- SaveDocX[249], SaveDocXToStream[250] saves the document (or selection) in a file/stream as DocX (Microsoft Word format);
- SaveRTF[253], SaveRTFToStream[254] saves the document (or selection) in a file/stream as RTF.

Properties:

- RTFOptions[185] – options for RTF export.

Events:

- OnSaveComponentToFile[292] allows to save inserted controls in HTML and text files;
- OnWriteHyperlink[302], OnReadHyperlink[284] allow to customize saving/loading of hypertext links in HTML and RTF files;
- OnReadMergeField[286] allow importing merge fields from RTF;
- OnHTMLSaveImage[277], OnSaveImage2[296] allow to save images with HTML in custom formats and with custom file names;
- OnSaveItemToFile[297] allows to save any item in RTF, HTML or text files in your own format;
- OnImportPicture[279] allows custom processing of external images when importing RTF;
- OnAssignImageFileName[274] allows changing file names assigned to inserted images.

Information about items (Get***Info methods)

- GetItemStyle[229] returns a type of item;

- GetTextInfo[239] – information about text item;
- GetBreakInfo[218] – about "break";
- GetBulletInfo[219] – about "bullet";
- GetHotspotInfo[224] – about "hotspot";
- GetPictureInfo[236] – about picture or "hot-picture";
- GetControlInfo[222] – about inserted control;

- GetItemExtraIntProperty[Ex] 227, GetItemExtraStrProperty[Ex] 228 return value of extra item property;
- GetItem 225 – general method for obtaining items (usually used for tables 145);

- GetItemCheckpoint 226 – about item's "checkpoint";
- GetItemVAlign 230 – about item's position relative to line;
- GetItemTag 230 – about item's "tag";
- GetItemText 230 – about text/name of item;
- IsParaStart 242 returns "is this item a first item in paragraph?";
- IsFromNewLine 242 returns "is this item a first item in paragraph?";
- GetItemPara 229 returns an index of paragraph containing a specified item;
- GetItemNo 229 returns item index for object representing item (usually used for tables 145);
- GetItemCoords 226 returns coordinates of item relative to document area, GetItemClientCoords 226 returns coordinates of item relative to client area of RichView window;
- properties PageBreaksBeforeItems 184, ClearLeft 171, ClearRight 172.

Modifying items (Set***Info methods)

- SetBreakInfo 260 changes *break* 125;
- SetBulletInfo 261 changes *bullet* 128;
- SetHotspotInfo 265 changes *hotspot* 129;
- SetPictureInfo 269 changes picture 122 or *hot-picture* 125;
- SetControlInfo 262 changes inserted control 126; also
- SetItemTag 267 changes item's "tag";
- SetItemVAlign 268 – change the item position relative to the line;
- SetItemText 267 changes text/name of item;
- SetItemExtraIntProperty[Ex] 266, SetItemExtraStrProperty[Ex] 266 sets value of extra item property;
- SetCheckpointInfo 261 sets or changes item's "checkpoint";
- RemoveCheckpoint 248 removes item's "checkpoint";
- see also properties PageBreaksBeforeItems 184, ClearLeft 171, ClearRight 172.

Working with *checkpoints*

Obtaining checkpoint data:

- GetFirstCheckpoint 223, GetNextCheckpoint 234 allow to receive full list of *checkpoints*;
- GetLastCheckpoint 232, GetPrevCheckpoint 237 – the same in backward direction;
- GetCheckPointByNo 220 returns a *checkpoint* with the specified index;
- GetItemCheckpoint 226 returns the item's *checkpoint*;
- event OnCheckPointVisible 274 occurs when *checkpoint* (or "section", marked by this *checkpoint*) becomes visible as a result of vertical scrolling;
- FindCheckPointByName 217 searches for *checkpoint* with the specified name;
- FindCheckPointByTag 217 searches for *checkpoint* with the specified *tag*.

Extracting properties from *checkpoint* data:

- GetCheckpointInfo 220 returns main properties of the *checkpoint*;
- GetCheckpointItemNo 221 returns an index of item which owns the *checkpoint*;
- GetCheckpointNo 221 returns an index of the *checkpoint*;
- GetCheckpointYEx 222, GetCheckpointXY 221 returns coordinates of the *checkpoint*.

Also:

- GetCheckPointY 221 returns Y coordinate of *checkpoint* by its index.

See also methods for obtaining information about items 165 and for modifying document 166.

Mouse events

- OnRVDblClick 287 occurs on double click;
- OnRVMouseDown 289 occurs when user presses mouse button;

- OnRVMouseUp[291] occurs when user releases mouse button;
- OnRVRightClick[291] occurs on right click.
- OnItemHint[281] allows customizing hints displayed when the mouse pointer moves above an item.
- OnGetItemCursor[277] allows customizing cursors displayed when the mouse pointer moves above an item.

See also hypertext events[163].

Live spelling check

- StartLiveSpelling[273] starts live spelling thread;
- ClearLiveSpellingResults[210] clears live spelling underlines and stops the thread;
- LiveSpellingValidateWord[242] removes live spelling underlines from the specified word;
- OnSpellingCheck[301] event – the place to check word spelling.

Picture animation

Properties:

- AnimationMode[169] specifies when image animation[90] starts.

Methods:

- StartAnimation[272] starts all animations;
- StopAnimation[273] stops all animations;
- ResetAnimation[249] rewinds all animations to the first frame.

Working in palette mode

- property DoInPaletteMode[177];
- method UpdatePaletteInfo[273].

Others

Properties:

- RVData[187] – object representing document;
- Cursor[174] – cursor;
- Delimiters[174] – characters not belonging to words;
- TabNavigation[194] sets mode for navigation through hypertext links and controls;
- AnimationMode[169] defines when animated images starts animation;
- DocObjects[174] – additional object information for saving in RVF[93];
- DocProperties[176] – additional text information for saving in RVF[93];
- DocParameters[176] – store document page parameters;
- NoteText[182] – text for references to parent notes;
- UseVCLThemes[195] allows using colors of VCL themes (in Delphi XE2+)

Methods:

- DeleteUnusedStyles[216] deletes unused text, paragraph and list styles from TRVStyle;
- MarkStylesInUse[247], DeleteMarkedStyles[215] – advanced methods for deleting unused styles;
- SearchText, -A, -W[256] search for the text string;
- GetWordAt[239] returns a word at the specified point;
- DeleteItems[214], DeleteSection[215] delete the specified range of items from document;
- DeleteParas[215] deletes the specified range of paragraphs, updates the document;
- FindControlItemNo[217] returns index of item which owns the specified inserted control[126];
- GetFocusedItem[223] returns the item in the focus;
- GetLineNo[233] returns line number in the specified position;
- BeginOleDrag[209] starts drag&drop.

Events:

- OnItemAction[280];
- OnControlAction[275].

1.5.1.1.1 TRichView : Component editor

Component editor for TCustomRichView (and all inherited components: TRichView[162], TRichViewEdit[316], TDBRichView[407], TDBRichViewEdit[410]).

To display the component editor dialog in Delphi (C++Builder) IDE, right click RichView object on the form, choose "Settings" in the popup menu.

The dialog consists of 3 pages: Styles, Tags, RVF.

Styles

This page allows to select one of 3 style modes:

- Use a predefined set of styles.
- Allow adding styles dynamically.
- Allow adding styles dynamically + style templates

This setting changes the properties: RVFTextStylesReadMode[191], RVFParaStylesReadMode[190], RVFOptions[188], RTFReadProperties.TextStyleMode[309], .ParaStyleMode[308], UseStyleTemplates[195]. Changes are shown in the light yellow box below.

"Use a predefined set of styles" mode

In this mode, the component:

- does not add new items in the collection of text[437], paragraph[431] and list[429] styles of the linked[193] TRVStyle[418] component;
- does not save these styles in RVF[93] files or streams.

"Allow adding styles dynamically" mode

In this mode, the component:

- adds new items in the collection of text[437], paragraph[431] and list[429] styles, if necessary; new styles can be added as a result of RVF and RTF loading;
- saves these styles in RVF[93] files and streams.

If you want to display RTF files close to the original, or use commands for adding styles (for example, inside OnStyleConversion[400] or OnParaStyleConversion[397], or RichViewActions), you must set this mode.

Since version 1.8, "Allow adding styles dynamically" is set by default for all new TRichViews created at design time (placed on forms from the Component Palette), if the defaults were not overridden.

"Allow adding styles dynamically + style templates" mode

All properties are set to the same values as in the previous mode, but UseStyleTemplates[195]=*True*.

Additional information about these modes can be found in the topic "Styles and style templates"[74].

RVF[93]

Defines options for saving and loading RVF (RichView Format)[93] files and strings. These settings change RVFOptions[188].

Storing The Component Editor's Options

Set "Default" checkbox to store settings on the given page. Settings are stored in the Registry. They will be applied to new components created in Delphi/C++Builder IDE from the Component Palette. Stored settings do not affect component creation at run time (from your code), and components that already were added on forms.

1.5.1.1.2 Properties

1.5.1.1.2.1 TRichView.AllowSelection

```
property AllowSelection: Boolean;
```

This property still works, but obsolete, please use *rvoAllowSelection* in Options[182]

1.5.1.1.2.2 TRichView.AnimationMode

Specifies when image animation starts.

```
type
  TRVAnimationMode = (rvaniDisabled, rvaniManualStart, rvaniOnFormat);
property AnimationMode: TRVAnimationMode;
```

(introduced in version 10)

Mode	Meaning
rvaniDisabled	Animation is completely disabled. If images are added/updated in this mode, they cannot be animated (even if different mode is set later). This mode saves system resources.
rvaniManualStart	Animation starts when calling StartAnimation[272] and stops when calling StopAnimation[273].
rvaniOnFormat	Animation starts when calling Format[217] and stops when calling Clear[210]. StartAnimation[272] and StopAnimation[273] work as well.

Default value:

rvaniManualStart

See also:

- Animation in TRichView[90];
- Pictures[122];
- *Hot-Pictures*[125].

1.5.1.1.2.3 TRichView.BackgroundBitmap

Background image shown below the document

```
property BackgroundBitmap: TBitmap;
```

Assignment to this property copies the image, so you still need to free the source image yourself.

Example 1 (assigning image)

```
bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
MyRichView.BackgroundBitmap := bmp;
bmp.Free;
```

Example 2 (doing the same work)

```
MyRichView.BackgroundBitmap.LoadFromFile('c:\image.bmp');
```

Example 3 (clearing the image)

```
MyRichView.BackgroundBitmap := nil;
```

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetBackgroundImageEd[371].

See also properties:

- BackgroundStyle[170].

1.5.1.1.2.4 TRichView .BackgroundStyle

Defines how BackgroundBitmap[169] is displayed

```
type
  TBackgroundStyle = (bsNoBitmap, bsStretched,
    bsTiled, bsTiledAndScrolled, bsCentered,
    bsTopLeft, bsTopRight,
    bsBottomLeft, bsBottomRight);

property BackgroundStyle : TBackgroundStyle;
```

Note: TBackgroundStyle is declared in RVScroll unit.

Style	Meaning
bsNoBitmap	Colored background (bitmap is not used)
bsStretched	Bitmap is stretched to fit RichView window
bsTiled	Repeats bitmap until the entire RichView window is covered
bsTiledAndScrolled	Repeats bitmap until the entire scrollable area is covered (tiles are scrolled together with content)
bsCentered	Centers bitmap in RichView window
bsTopLeft,	Bitmap is in the top left corner
bsTopRight	Bitmap is in the top right corner
bsBottomLeft	Bitmap is in the bottom left corner
bsBottomRight	Bitmap is in the bottom right corner

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

See also properties:

- BackgroundBitmap[169];
- Color[173].

See also:

- TRVItemBackgroundType[682] type;
- Smooth image scaling[91].

1.5.1.1.2.5 TRichView .BiDiMode

Specifies the default bidirectional mode for the control.

```
property BiDiMode: TRVBiDiMode[668];
```

(introduced in version 1.6)

Value	Meaning
rvbdUnspecified	Turns off support of right-to-left languages. Bi-directed text will not be processed correctly. Also, bi-di properties will not be saved and loaded from files.

	Text output is more efficient in this mode, so, if you do not plan working with Arabic or Hebrew texts, you can use this mode.
rvbdLeftToRight	Sets default text flow to left-to-right
rvbdRightToLeft	Sets default text flow to right-to-left; if version of Windows allows it, a vertical scrollbar will be positioned at the left side of the TRichView window.

BiDiMode can be overridden by BiDiMode of paragraph style[484] and BiDiMode of text style[467].

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

The results depends on the value of Style[193].TextEngine[436]:

- if Windows API is selected, in *rvbdUnspecified* mode the results are completely incorrect for RTL text (wrong selection drawing and caret positions); the results are unreliable in other modes (may depend on the chosen font);
- if Uniscribe is selected, the results are correct is all modes; in *rvbdUnspecified* mode, only rearrangement of items on lines is turned off.

See also:

- Bidirectional text in TRichView[100].

Default value:

rvbdUnspecified

1.5.1.1.2.6 TRichView .BottomMargin

Bottom margin of document

```
property BottomMargin: Integer;
```

This value is measured in "screen" pixels, so when printing it's recalculated according to the printer resolution, and margins on pages look like margins on the screen.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

See also properties:

- LeftMargin[180];
- RightMargin[185];
- TopMargin[194].

See also properties of TRVPrint:

- Margins[520].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[80]

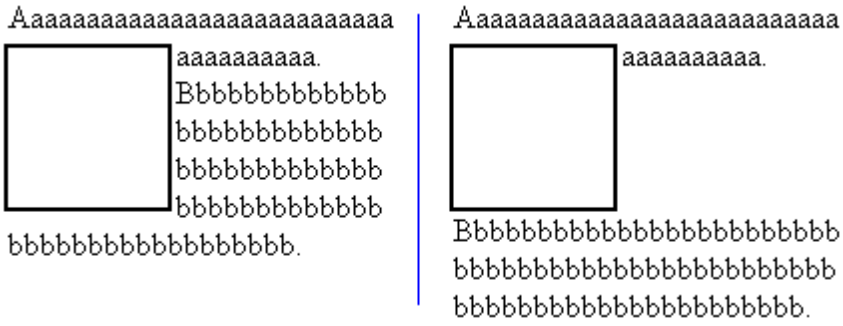
1.5.1.1.2.7 TRichView .ClearLeft

Specifies whether a text flow around left-aligned[695] items should be cleared for the paragraph started from the **Index**-th item.

```
property ClearLeft[Index: Integer]: Boolean
```

(Introduced in version 12)

Set ClearLeft[ItemNo] to *True* if you want the ItemNo-th item to be placed below any left-aligned[695] object.



This property is illustrated on the picture above. At the left side, you can see two paragraphs ("aaa" and "bbb") flow around left-aligned picture. At the right side, you can see how it looks if the first item in the "bbb" paragraph has ClearLeft[] = *True*.

This item must start a new paragraph (IsParaStart[242](ItemNo)) or a new line inside a paragraph (IsFromNewLine[242](ItemNo)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using Format[217]).

In editor you can change value of this property as an editing operation, using ClearTextFlow[339] method.

Note: if special characters are shown (i.e. *rvoShowSpecialCharacters* is included in Options[182]), this property is shown as a vertical line to the left of the paragraph (or line break) mark. For a paragraph, this line is shown if ClearLeft is True for the next paragraph.

See also:

- ClearRight[172],
- PageBreaksBeforeItems[184],
- TRVVAlign[695] type (*rvvaLeft*).

1.5.1.1.2.8 TRichView .ClearRight

Specifies whether a text flow around right-aligned[695] items should be cleared for the paragraph started from the **Index**-th item.

```
property ClearRight[Index: Integer]: Boolean
```

(Introduced in version 12)

Set ClearRight[ItemNo] to *True* if you want the ItemNo-th item to be placed below any right-aligned[695] object.

This item must start a new paragraph (IsParaStart[242](ItemNo)) or a new line inside a paragraph (IsFromNewLine[242](ItemNo)), otherwise assignment will be ignored.

You need to reformat RichView after this assignment (using Format[217]).

In editor you can change value of this property as an editing operation, using ClearTextFlow[339] method.

Note: if special characters are shown (i.e. *rvoShowSpecialCharacters* is included in Options[182]), this property is shown as a vertical line to the right of the paragraph (or line break) mark. For a paragraph, this line is shown if ClearLeft is True for the next paragraph.

See also:

- ClearLeft[171],
- PageBreaksBeforeItems[184],
- TRVVAlign[695] type (*rvvaRight*).

1.5.1.1.2.9 TRichView.Color

Background color

```
property Color: TColor;
```

This property allows to override the default background color specified in RichView.Style[193].Color[421].

RichView paints background with this color, if it is not equal to *clNone*, otherwise RichView uses the default color.

Background color is used if BackgroundBitmap[169] is empty, or BackgroundStyle[170] is not equal to *bsStretched*, *bsTiled* or *bsTiledAndScrolled.*

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

Default value:

clNone

See also properties:

- BackgroundStyle[170];
- BackgroundBitmap[169].

See also properties of TRVStyle:

- Color[421].

1.5.1.1.2.10 TRichView.CPEventKind

Specifies the mode of generating OnCheckpointVisible[274] event.

```
type
  TCPEventKind =
   (cpeNone, cpeAsSectionStart, cpeWhenVisible);

property CPEventKind: TCPEventKind;
```

Note: TCPEventKind is declared in RVScroll unit.

Value	Meaning
cpeNone	OnCheckpointVisible[274] does not occur
cpeAsSectionStart	OnCheckpointVisible[274] is called for visible *checkpoint* or the nearest *checkpoint* above the visible area (so that the last event is always called for *checkpoint* located in or above the visible area)
cpeWhenVisible	OnCheckpointVisible[274] is generated for visible *checkpoint*

This event occurs only for *checkpoints* having "raise event" flag.

This event never occurs in TRichViewEdit (in this version)

See also events:

- OnCheckpointVisible[274].

See also:

- *Checkpoints*[65] for details.

1.5.1.1.2.11 TRichView .Cursor

Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.

```
property Cursor: TCursor;
```

Default value:

- *crDefault* for TRichView;
- *crIBeam* for TRichViewEdit[316].

See also:

- RichViews' cursors[108].

1.5.1.1.2.12 TRichView .Delimiters

Set of delimiters for specifying a "word" in document

```
property Delimiters: TRVUnicodeString[694];
```

(changed in version 18)

Each character of this string is one delimiter.

Default value (changed in version 18)

```
  RVDEFAULTDELIMITERS = ' .;,:(){}"/\<>!?[]' + UNI_LEFT_SINGLE_QUOTE +
    UNI_RIGHT_SINGLE_QUOTE + UNI_LEFT_DOUBLE_QUOTE + UNI_RIGHT_DOUBLE_QUOTE +
    '-+*=' + UNI_NBSP + UNI_RIGHT_DOUBLE_LOW_QUOTE +
    UNI_LEFT_DOUBLE_ANGLE_QUOTE + UNI_RIGHT_DOUBLE_ANGLE_QUOTE + UNI_ELLIPSIS +
    UNI_ZERO_WIDTH_SPACE + UNI_EN_SPACE + UNI_EM_SPACE + UNI_FOURPEREM_SPACE;
```

See also methods:

- SelectWordAt[258];
- GetWordAt[239];
- SearchText[256].

See also events:

- OnRVDblClick[287];
- OnRVRightClick[291].

See also methods of TRichViewEdit:

- SelectCurrentWord[370].

See also:

- Live spelling check[101].

1.5.1.1.2.13 TRichView .DocObjects

A collection of objects that can be stored in RVF files or streams.

```
type // defined in RVDocParams unit
  TRVDocObjectCollection = class (TOwnedCollection);
  TRVDocObject = class (TCollectionItem);
property DocObjects: TRVDocObjectCollection;
```

(introduced in version 17)

TRVDocObjectCollection is a collection of items inherited from TRVDocObject. Items of this collection may have different type.

All published properties of items can be saved and loaded.

If *rvfoSaveDocObjects* is in RVFOptions[188], these objects are saved in RVF.

If *rvfoLoadDocObjects* is in RVFOptions 188, methods for loading RVF load these objects. Methods for inserting RVF ignore these objects.

Method TRichView.Clear 210 clears **DocObjects**.

There is one more property allowing to save custom data in RVF: DocProperties 176.

Example 1

TRVMathDocObject 656 defines default properties for items displaying mathematical expressions 143.

Example 2

Report Workshop defines TRVReportDocObject having properties DataQuery, HighlightRules and others.

So, for Report Workshop, **DocObjects** is a way of adding new properties to TRichView without modifying TRichView source code.

Example 3

Define your own classes:

```
type
  TMyDocObjectA = class (TRVDocObject)
    ...
  published
    property A: String ...
  end;

  TMyDocObjectB = class (TRVDocObject)
    ...
  published
    property B: Integer ...
  end;
```

Adding:

```
var
  ObjA: TMyDocObjectA;
  ObjB: TMyDocObjectB;
...
  ObjA := TMyDocObjectA.Create(MyRichView.DocObjects);
  ObjA.A := 'Hello';
  ObjB := TMyDocObjectB.Create(MyRichView.DocObjects);
  ObjB.B := 1;
```

RVF

How to save to RVF: include *rvfoSaveDocObjects* in RVFOptions 188.

How to load from RVF: include *rvfoLoadDocObjects* in RVFOptions 188.

See also methods for saving RVF (they can save this property):

- SaveRVF 254;
- SaveRVFToStream 254.

See also methods for loading RVF (they can load this property):

- LoadRVF 245;
- LoadRVFFromStream 245.

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd 357;
- InsertRVFFromStreamEd 357;
- PasteRVF 364.

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream 240;

- AppendRVFFromStream 208.

See also properties:

- RVFOptions 188.

See also:

- Saving and loading RichView documents 92;
- RVF 93.

1.5.1.1.2.14 TRichView.DocParameters

Page properties.

```
property DocParameters: TRVDocParameters 310;
```

(introduced in version 10)

Subproperties of this property are not used by TRichView itself, but can be written and read in RVF 93 and RTF.

Besides, they can be assigned to TRVPrint 513 using the method AssignDocParameters 522.

RVF

DocParameters are stored with DocProperties 176.

How to save to RVF: include *rvfoSaveDocProperties* in RVFOptions 188.

How to load from RVF: include *rvfoLoadDocProperties* in RVFOptions 188.

RTF and DocX

How to save to RTF/DocX: include *rvrtfSaveDocParameters* in RTFOptions 185.

How to load from RTF: set RTFReadProperties 187.ReadDocParameters 308 = *True*.

See also events:

- OnSaveRTFExtra 299;
- OnSaveDocXExtra 293.

1.5.1.1.2.15 TRichView.DocProperties

Text strings to store in RVF file or stream

```
property DocProperties: TStringList;
```

(introduced in v1.8)

This property is public,so it is not available at design time.

Items of this list can contain line break characters (#13 and #10), but cannot contain #01 and #02 characters.

If *rvfoSaveDocProperties* is in RVFOptions 188, these strings are saved in RVF.

If *rvfoLoadDocProperties* is in RVFOptions 188, methods for loading RVF load these strings. Methods for inserting RVF ignore these strings.

There is one more property allowing to save custom data in RVF: DocObjects 174.

Method TRichView.Clear 210 clears DocProperties.

See also methods for saving RVF (they can save this property):

- SaveRVF 254;
- SaveRVFToStream 254.

See also methods for loading RVF (they can load this property):

- LoadRVF 245;
- LoadRVFFromStream 245.

See also methods of TRichViewEdit for inserting RVF in the caret position (they ignore this property):

- InsertRVFFromFileEd 357;
- InsertRVFFromStreamEd 357;
- PasteRVF 364.

See also methods for inserting RVF (they ignore this property):

- InsertRVFFromStream 240;
- AppendRVFFromStream 208.

See also properties:

- RVFOptions 188.

See also:

- Saving and loading RichView documents 92;
- RVF 93.

1.5.1.1.2.16 TRichView.DocumentHeight

Height of document (height of scrollable area)

```
property DocumentHeight: Integer;
```

Document must be formatted to obtain a correct value.

Example:

```
if MyRichView.DocumentHeight>Screen.Height then
  MyRichView.Height := Screen.Height
else
  MyRichView.Height := MyRichView.DocumentHeight;
```

See also:

- Scrolling 80.

1.5.1.1.2.17 TRichView.DoInPaletteMode

Defines behavior of RichView when drawn in 256-color mode display.

```
type
  TRVPaletteAction =
    (rvpaDoNothing, rvpaAssignPalette,
     rvpaCreateCopies, rvpaCreateCopiesEx);

property DoInPaletteMode: TRVPaletteAction.
```

Note: TRVPaletteAction is declared in RVScroll unit.

Please always set DoInPaletteMode to rvpaDoNothing under Delphi 2 and C++Builder 1.Palette mode improvements are not available for these versions.

This property affects only displaying in-256 color mode and does nothing in highcolor and truecolor modes.

Windows XP does not support 256-color mode.

Mode	Meaning
rvpaDoNothing	RichView does nothing with palette. Use if you have no pictures in RichView, or all pictures are in VGA 16-color palette.
rvpaAssignPalette	When working in 256-color mode, RichView creates palette and assigns it to all pictures (including background). This will affect all pictures that already in RichView, and all pictures that will be inserted later (when in 256-color mode). This is also not very good option,

	because pictures lose their quality, and it can't be restored when switching to highcolor/truecolor mode. Output to RVF, HTML, clipboard, printer will lose quality of pictures. So use this option if you want only to display document.
rvpaCreateCopies	(default and preferred option) In 256-color mode, RichView creates copies of all pictures (including background) and assign palette to them. This will affect all pictures that already in RichView, and all pictures that will be inserted later (when in 256-color mode). This work is absolutely "transparent" for you, you will work with source pictures that remain unchanged.
rvpaCreateCopiesEx	For future versions, do not use (idea: copies of pictures may be created not in format of source picture, but in some better format)

Notes:

1. If you have other controls on form which also change palette (such as TImage, or buttons with multicolor glyphs), you can not achieve required results.
2. If you use multicolor *hotspots* or *bullets* in RichView, it's recommended to have two sets of them: 16color for 16- and 256-color modes, and multicolor for others.
3. RichView can't change palette of bitmaps inside metafiles; do not use TMetafile containing multicolor bitmaps in 256-color mode.
4. RichView can't update information about palette when user switches color resolution without restarting Windows. Please do it yourself, as it's shown below.
5. If you create RichViews, RichViewEdits, RVPrintPreviews, RVPrints at run-time (such as MyRichView := TRichView.Create(Self)), you need to call UpdatePaletteInfo methods of such controls after you have created them.

In the main form write...

In interface section:

```
procedure WMDisplayChange(var Message: TWMDisplayChange); message WM_DISPLAYCHANGE;
```

In implementation:

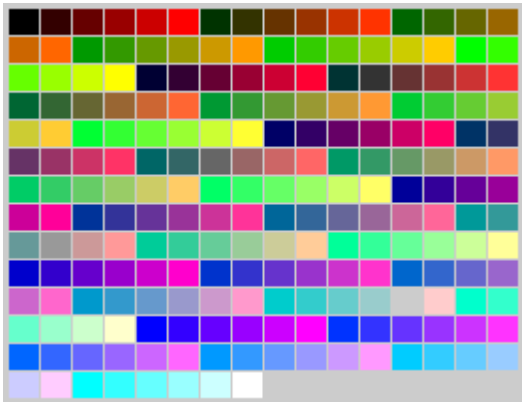
```
procedure TMyForm.WMDisplayChange(var Message: TWMDisplayChange);
begin
  // Call UpdatePaletteInfo methods of all your
  // RichViews, RichViewEdits, and RVPrints (if you use print preview)
  // for example:
  MyRichViewEdit.UpdatePaletteInfo[273];
  MyRVPrint.UpdatePaletteInfo[554];
end;
```

Default value:

rvpaCreateCopies

(or *rvpaDoNothing* in Delphi 2 and C++Builder 1)

RichView uses the palette displayed below (the same as Microsoft Internet Explorer uses):



If you need to use your own palette, let me know, I will create a public property for it. Now you can override the protected dynamic method GenerateLogPalette (introduced in TRVScroller).

See also methods:

- UpdatePaletteInfo 273.

See also methods of TRVPrint:

- UpdatePaletteInfo 554.

1.5.1.1.2.18 TRichView.FirstItemVisible

Index of the first visible (in window) item, or -1 if the document is empty.

```
property FirstItemVisible: Integer;
```

You can use this property (for example, in OnVScrolled 549 event handler) to get the index of the first visible item.

This property does not take horizontal scrolling into account, only vertical one.

Note: returning value of this property requires some calculations; so if you need to use it several times in the same procedure, assign its value to some temporary variable and then use this variable.

See also properties:

- LastItemVisible 180.

See also:

- Vertical scrolling 80.

1.5.1.1.2.19 TRichView.FirstJumpNo

Hypertext **id** for the first hyperlink in the document.

```
property FirstJumpNo: Integer;
```

When you append hypertext items (*hotspots* 129, *hot-pictures* 125 or text of hypertext style), RichView assigns hypertext identifiers to them. The first such item has **id**=FirstJumpNo, the next one – FirstJumpNo+1 and so on.

All hypertext items have hypertext **id**s. **Id**s are incremented according to their order in the document, from top to bottom.

This property is useful if you want to use one OnJump 282 event handler for several RichViews.

Default value:

0

See also:

- RichView hypertext 69.

1.5.1.1.2.20 TRichView .ItemCount

Number of items in RichView

```
property ItemCount: Integer;
```

See also:

- Item types 64;
- Adding items 77;
- Obtaining items 84;
- Modifying items 85.

1.5.1.1.2.21 TRichView .LastItemVisible

Index of the last visible (in window) item, or -1 if the document is empty.

```
property LastItemVisible: Integer;
```

You can use this property (for example, in OnVScrolled 549 event handler) to get the index of the last visible item.

This property does not take horizontal scrolling into account, only vertical one.

Note: returning value of this property requires some calculations; so if you need to use it several times in the same procedure, assign its value to some temporary variable and then use this variable.

See also properties:

- FirstItemVisible 179.

See also:

- Vertical scrolling 80.

1.5.1.1.2.22 TRichView .LeftMargin

Left margin of document

```
property LeftMargin: Integer;
```

This value is measured in "screen" pixels, so when printing it's recalculated according to the printer resolution, and margins on the page look like margins on the screen.

Left margin can be used for selecting lines with the mouse.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd 370.

See also properties:

- RightMargin 185;
- TopMargin 194;
- BottomMargin 171.

See also indents in paragraph style 482.

See also properties of TRVPrint:

- Margins 520.

See also properties of TRVStyle:

- LineSelectCursor 427.

See also:

Additional information about horizontal scrolling, margins, indents, etc. 80

1.5.1.1.2.23 TRichView.LineCount

Obsolete, use ItemCount[180].

```
property LineCount : Integer;
```

See also

- GetLineNo[233].

1.5.1.1.2.24 TRichView.MaxLength

Allows to to limit a number of characters per line.

```
property MaxLength: Integer;
```

(introduced in version 10)

Set this property to positive value to limit a number of characters per line. Non-text items are treated like one character.

This property affects only word wrapping.

This property is ignored if WordWrap[196] = *False*. This property is ignored for paragraphs having *rvpaoNoWrap* in Options[487].

Default value:

0

1.5.1.1.2.25 TRichView.MaxTextWidth

If set to positive value, it is a maximal width for text wrapping (text wraps when its length in pixels exceeds this value)

```
property MaxTextWidth: Integer;
```

This property does not affect printing.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

Default value:

0

See horizontal scrolling[80] for details.

See also properties:

- MaxLength[181].

1.5.1.1.2.26 TRichView.MinTextWidth

Minimum width for text wrapping. This value allows to increase width of scrollable area in TRichView.

```
property MinTextWidth : Integer;
```

This property does not affect printing.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

Default value:

0

See horizontal scrolling[80] for details.

1.5.1.1.2.27 TRichView .NoteText

Text for references to the parent note[140].

```
property NoteText: TRVUnicodeString[694];
```

(introduced in version 10)

Normally, references[140] must be placed in Document[628] for footnotes[136], endnotes[136] or sidenotes[137]. In this case, they display the same text as their parent note.

References[140] should not appear in TCustomRichView, but the user can copy them here. In this case, they display text specified in this property.

If this TCustomRichView is used for displaying/editing Document[628] for some note, you should set value of this property equal to NoteText[628] of this note.

Default value:

'?'

1.5.1.1.2.28 TRichView .Options

Options

```
type
  TRVOption = (rvoAllowSelection, rvoSingleClick, rvoScrollToEnd,
    rvoClientTextWidth,
    rvoShowCheckpoints, rvoShowPageBreaks,
    rvoShowSpecialCharacters, rvoShowHiddenText,
    rvoTagsArePChars,
    rvoAutoCopyText,  rvoAutoCopyUnicodeText
    rvoAutoCopyRVF, rvoAutoCopyImage,
    rvoAutoCopyRTF,
    rvoFormatInvalidate,
    rvoDblClickSelectsWord, rvoRClickDeselects,
    rvoDisallowDrag, rvoShowItemHints, rvoShowGridLines,
    rvoFastFormatting, rvoPercentEncodedURL, rvoAssignImageFileNames);
  TRVOptions = set of TRVOption;

property Options: TRVOptions;
```

Note: TRVOption and TRVOptions are declared in RVScroll unit.

Default value

For Delphi/C++Builder 2009 or newer: [*rvoAllowSelection, rvoScrollToEnd, rvoAutoCopyUnicodeText, rvoAutoCopyImage, rvoAutoCopyRVF, rvoAutoCopyRTF, rvoDblClickSelectsWord, rvoRClickDeselects, rvoFormatInvalidate, rvoShowPageBreaks, rvoShowGridLines, rvoFastFormatting*]
For older versions: the same, but with *rvoAutoCopyText* instead of *rvoAutoCopyUnicodeText*.

General Options

Option	Meaning
rvoTagsArePChars	Obsolete, does nothing.
rvoClientTextWidth	If set, text always wraps to fit client width of RichView. Also affects breaks[125] and tables[130]. See details[80]. Can be changed: any time, but you need to reformat[248] and repaint RichView after changing this option, if the document was already formatted.

rvoScrollToEnd	If set (default), RichView scrolls to the end of document when FormatTail[218] method is called. If cleared, document does not scroll in this case Can be changed: any time.
rvoSingleClick	If set, OnRVDblClick[287] event is generated on single mouse click If cleared (default), it is generated on double mouse click Can be changed: any time.
rvoFormatInvalidate	If set (default), Format[217] and FormatTail[218] methods repaint component. Can be changed: any time.
rvoDisallowDrag	If set, dragging[89] from this TRichView is not allowed. Can be changed: any time.
rvoFastFormatting	Increase performance in WinNT-based OS at the cost of some resources
rvoPercentEncodedURL	If set, it is assumed that URLs (stored in tags or provided in OnWriteHyperlink[302] event) are %-encoded, so no additional encoding/decoding is necessary when working with RTF, HTML, and DocX. By default (if not set), the component %-encodes hyperlink targets saved to RTF, HTML, DocX, and %-decodes when loading RTF.
rvoAssignImageFileNames	If set, the component assigns file names of images, when possible. This option affects: RTF loading (reading external images), HTML loading (using importer components), pasting files, accepting files and links to images as a result of drag&drop, RichViewActions (actions for inserting images and assigning background images of tables). The following properties are set: • table.BackgroundImageFileName[568] • cell.BackgroundImageFileName[606] • *rvespImageFileName*[676] item property See also: OnAssignImageFileName[274] event.

Selection Options

Can be changed: any time.

Option	Meaning
rvoAllowSelection	If set (default), user can select content of RichView using mouse or keyboard. Please do not clear this option for TRichViewEditor! See: working with selection[81].
rvoDblClickSelectsWord	If set (default), double click on word selects it (and double click on non-text item selects this item)
rvoRClickDeselects	If set (default), right click outside the selected area deselects (and moves caret to the clicked position in TRichViewEdit[316])

Show/Hide Options

Option	Meaning
rvoShowCheckpoints	If set, all *checkpoints* are shown (as dotted horizontal lines, or by OnDrawCheckpoint[444]). See details[65]. Can be changed: any time, you need to repaint RichView after changing this option
rvoShowPageBreaks	If set (default), page breaks are shown as horizontal lines. Can be changed: any time, you need to repaint RichView after changing this option
rvoShowSpecialCharacters	If set, nonprinting characters are displayed as special marks: spaces, nonbreaking spaces, paragraph breaks, line breaks, tabs[121], soft hyphens in Unicode text, lines to left- and right-aligned items, places of insertion of text boxes[139]. Can be changed: any time, you need to reformat[248] RichView after changing this option, if the document is already formatted. See also: RVVisibleSpecialCharacters[714], RichViewShowGhostSpaces[704] typed constants.
rvoShowHiddenText	If set, hidden items[76] are displayed. Can be changed: any time, you need to reformat[248] RichView after changing this option, if the document is already formatted.
rvoShowItemHints	If set, items' hints are shown. Popup hints are shown only if ShowHints property is set to True. See item properties[676] and OnItemHint[281] event.
rvoShowGridLines	If set, table[130] grid lines[425] are shown.

Clipboard Options (See Details[82])

If set , RichView copies the selection to the Clipboard in the specified formats when the user presses Ctrl+Insert or Ctrl+C. These options also affect CopyDef[212]. These options can be changed at any time.

Option	Format to Copy
rvoAutoCopyText	ANSI text
rvoAutoCopyUnicodeText	Unicode text
rvoAutoCopyRVF	RVF[93] (RichView Format)
rvoAutoCopyRTF	RTF (Rich Text Format)
rvoAutoCopyImage	Bitmap or metafile (if the image is selected)

1.5.1.1.2.29 TRichView .PageBreaksBeforeItems

Specifies if the **Index**-th item must start a new page when printing

```
property PageBreaksBeforeItems[Index: Integer]: Boolean
```

(Introduced in version 1.2)

Set PageBreaksBeforeItems[ItemNo] to *True* if you want the ItemNo-th item to start a new page. This assignment forces the item to start a new paragraph.

If PageBreaksBeforeItems[ItemNo]=*True*, a text flow for this item is cleared for both left and right side, i.e. such item is placed as if ClearLeft[171][ItemNo] and ClearRight[172][ItemNo] = *True*. Because of this, assignment to this property requires reformatting.

RichView never prints blank pages. So, for example, PageBreaksBeforeItems[0] has no effect (because it will start a new page in any case).

The editor has additional methods for working with explicit page breaks.

See also:

- Options[182] (*rvoShowPageBreaks*).

See also methods:

- AssignSoftPageBreaks[209].

See also properties and events of TRVStyle:

- PageBreakColor[431];
- OnDrawPageBreak[446].

See also methods of TRichViewEdit:

- InsertPageBreak[355];
- RemoveCurrentPageBreak[367].

See also properties of table rows:

- PageBreakBefore[616].

1.5.1.1.2.30 TRichView.RightMargin

Right margin of document

```
property RightMargin: Integer;
```

This value is measured in "screen" pixels, so when printing it's recalculated according to the printer resolution, and margins on pages look like margins on the screen.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

See also properties:

- LeftMargin[180];
- TopMargin[194];
- BottomMargin[171].

See also indents in paragraph style[482]

See also properties of TRVPrint:

- Margins[520].

See also properties of TRVStyle:

- LineSelectCursor[427].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[80]

1.5.1.1.2.31 TRichView.RTFOptions

Options for saving RTF (Rich Text Format). Some options are used for saving DocX files as well.

```
type
  TRVRTFOption = (rvrtfSaveStyleSheet,
    rvrtfDuplicateUnicode, rvrtfSaveEMFAsWMF,
    rvrtfSaveJpegAsJpeg, rvrtfSavePngAsPng,
    rvrtfSaveBitmapDefault,
    rvrtfSaveEMFDefault, rvrtfSavePicturesBinary,
    rvrtfPNGInsteadOfBitmap,
```

```
    rvrtfSaveDocParameters, rvrtfSaveHeaderFooter);
  TRVRTFOptions = set of TRVRTFOption;

property RTFOptions: TRVRTFOptions;
```

(introduced in versions 1.3, 10, 11)

Note: TRVRTFOption and TRVRTFOptions are declared in RVStyle unit.

Options for RTF

Option	Meaning
rvrtfSaveStyleSheet	Used only if UseStyleTemplates[195]=*False*. If set (not recommended), RichView saves a style sheet in RTF; this style sheet is created from text and paragraph styles having Standard[464] =*True*. If UseStyleTemplates[195]=*True*, the component always saves Style[193].StyleTemplates[435] as a style sheet.
rvrtfDuplicateUnicode	If set, RichView saves an ANSI equivalent of Unicode text[99]. A conversion is based on Charset[468] property of text styles. Turning off this option increases saving speed and produces smaller RTF file. Unicode-enabled RTF readers just ignore ANSI equivalent of Unicode characters. However, ANSI text is necessary for RTF readers that cannot read Unicode.
rvrtfSaveEMFAsWMF	If set, RichView saves EMF (enhanced (Win32) metafiles) as WMF (Windows (Win3.1) metafiles). Some RTF readers do not understand EMF in RTF.
rvrtfSaveJpegAsJpeg	If set, RichView saves TJpegImage as JPEG. Otherwise, RichView saves JPEG images in other formats (see below). Some RTF readers do not understand JPEG images in RTF.
rvrtfSavePngAsPng	If set, RichView saves PNG images (PNG class is registered with RVGraphicHandler[710].RegisterPngGraphic) as PNG. Otherwise, RichView saves PNG images in other formats (see below). Some RTF readers do not understand PNG images in RTF.
rvrtfSaveBitmapDefault	If set, RichView saves graphics formats other than bitmaps, metafiles (and optionally Jpegs) as bitmaps. If not set, as WMF or EMF (see below). See also *rvrtfPNGInsteadOfBitmap* below. (unfortunately, many RTF reader do not understand WMF with mapping mode = MM_TEXT, which RichView uses for saving other formats; some other RTF readers do not understand bitmaps).
rvrtfSaveEMFDefault	If set (and rvrtfSaveBitmapDefault is not set), RichView saves graphics formats other than bitmaps (and optionally Jpegs) as EMF. If not set, as WMF
rvrtfSavePicturesBinary	If set, pictures are saved in binary format. Less in size (twice), but such RTF becomes a binary file (and cannot be stored in MEMO field of data bases)
rvrtfPNGInsteadOfBitmap	If included, and PNG class is registered with RVGraphicHandler[710] .RegisterPngGraphic, bitmaps are saved as PNG (including saving of non-RTF picture types, if *rvrtfSaveBitmapDefault* is included). Not all RTF readers understand PNG.

Options for RTF and DocX

Option	Meaning

rvrtfSaveDocParameters	If included, properties of DocParameters[176] are saved in RTF and DocX.
rvrtfSaveHeaderFooter	If included, documents assigned by SetHeader[264] and .SetFooter[263] are saved as a header and footer in RTF and DocX.

Default value

[*rvrtfDuplicateUnicode*, *rvrtfSaveEMFAsWMF*, *rvrtfSaveJpegAsJpeg, rvrtfSavePngAsPng*]

See also methods:

- SaveRTF[253];
- SaveRTFToStream[254];
- CopyRTF[213].

See also:

- Saving and loading in RichView[92];
- Pictures[122].

1.5.1.1.2.32 TRichView.RTFReadProperties

A set of properties controlling RTF (Rich Text Format) import.

```
property RTFReadProperties: TRVRTFReaderProperties[304];
```

Read the help on TRVRTFReaderProperties for details.

Subproperties of this class affect the following methods:

- TRichView.LoadRTF[243];
- TRichView.LoadRTFFromStream[244];
- TRichViewEdit.PasteRTF[363] (and Paste[361]);
- TRichViewEdit.InsertRTFFromStreamEd[357];
- TRichViewEdit.InsertRTFFromFileEd[357];
- drag&drop[89] when dropping RTF file.

See also:

- RTFOptions[185] (options for RTF export).

1.5.1.1.2.33 TRichView.RVData

This read-only property points to an object containing RichView document.

Each TRichView control owns one such object.

Similar objects represent cells[605] in tables[565].

Many RichView methods actually call methods of this object inside, for example:

```
procedure TRichView.AddNL(const s: String; StyleNo, ParaNo: Integer);
begin
  RVData.AddNL(s, StyleNo, ParaNo);
end;
```

A class of RVData is derived from the abstract base class TCustomRVFormattedData[650], which is in its order derived from TCustomRVData.

This object can be used for creating procedures working with any document object (main TRichView or cells).

For example, see AllUpperCase procedure in Controls, Documents, Items[104].

1.5.1.1.2.34 TRichView .RVFOptions

Options for reading and saving RVF (RichView Format)

```
type
  TRVFOption =
    (rvfoSavePicturesBody, rvfoSaveControlsBody,
    rvfoIgnoreUnknownPicFmt, rvfoIgnoreUnknownCtrls,
    rvfoIgnoreUnknownCtrlProperties
    rvfoConvUnknownStylesToZero,
    rvfoConvLargeImageIdxToZero,
    rvfoSaveBinary,
    rvfoUseStyleNames,
    rvfoSaveBack, rvfoLoadBack,
    rvfoSaveTextStyles, rvfoSaveParaStyles,
    rvfoSaveLayout, rvfoLoadLayout,
    rvfoSaveDocProperties, rvfoLoadDocProperties,
    rvfoCanChangeUnits, rvfoSaveStyleTemplatesOnlyNames,
    rvfoSaveDocObjects, rvfoLoadDocObjects,
    rvfoIgnoreGraphicClasses);
  TRVFOptions = set of TRVFOption;

property RVFOptions: TRVFOptions;
```

Note: TRVFOption and TRVFOptions are defined in RVStyle unit.

Options for Saving

Option	Meaning
rvfoSavePicturesBody	▪ If set, full information about all pictures in RichView are saved in RVF. ▪ If not set, OnRVFPictureNeeded 289 event occurs on loading.
rvfoSaveControlsBody	▪ If set, full information about all controls in RichView are saved in RVF. ▪ If not set, OnRVFControlNeeded 288 event occurs loading.
rvfoSaveBinary	▪ If set, all non-text data are saved in RVF as they are (pictures, controls, tables, background, styles, Unicode text). ▪ If not set, all binary data are converted to text (hexadecimal string); RVF becomes a plain text format, like RTF, but file size is increased.
rvfoSaveBack	*(since version 1.2)* If set, background (color and image) is saved in RVF. This flag affects only saving the whole document; selection is never saved (or copied to the Clipboard) with background. (see *rvfoLoadBack* below)
rvfoUseStyleNames *(almost obsolete)*	If set, TRichView saves quoted names of text and paragraph styles instead of their indices **Pluses:** ▪ content of file remains synchronized with collections of styles even if you delete or insert new styles; **Minuses:** ▪ increased file size; ▪ names of styles must be unique; ▪ files become sensitive to changing style names. It's not recommended to use this options together with *rvfoSaveTextStyles* and *rvfoSaveParaStyles*.

rvfoSaveTextStyles	If set, collection of text styles associated with this RichView (TextStyles[437] property of the linked[193] TRVStyle component) is saved in RVF. See RVFTextStylesReadMode[191] property. This option affects saving style templates.*
rvfoSaveParaStyles	If set, collections of paragraph and list styles associated with this RichView (ParaStyles[431] and ListStyles[429] properties of the linked[193] TRVStyle component) are saved in RVF. See RVFParaStylesReadMode[190] property. This option affects saving style templates.*
rvfoSaveLayout	If set, the following properties are stored in RVF: Left[180]-, Right[185]-, Top[194]-, BottomMargin[171], MinTextWidth[181], MaxTextWidth[181], BiDiMode[170]. Layout is saved only when saving the whole document (not selection). (see *rvfoLoadLayout* below)
rvfoSaveDocProperties	If set, DocProperties[176] and DocParameters[176] are saved in RVF. (see *rvfoLoadDocProperties* below)
rvfoSaveStyleTemplatesOnlyNames	If set, when style templates must be saved in RVF*, only their names are saved. When loading such files, existing style templates are applied to loaded text and paragraph styles. This option is useful if you want to create multiple documents using the same collection of style templates. Since style templates are not saved in RVF in this mode, an application must save[462] them itself elsewhere.
rvfoSaveDocObjects	If set, DocObjects[174] are saved in RVF. (see *rvfoLoadDocObjects* below)

* If the both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* are included, and UseStyleTemplates[195]=*True*, Style[193].StyleTemplates[435] are saved as well.

Options for Loading

Option	Meaning
rvfoLoadBack	If not set, TRichView ignores background information, saved in RVF. (see *rvfoSaveBack* above)
rvfoIgnoreUnknownPicFmt	▪ If set, TRichView skips pictures of unknown (unregistered) classes. ▪ if not set, RVF reading methods return False (failure value) if pictures of unknown classes exist in RVF
rvfoIgnoreUnknownCtrls	The same as *rvfoIgnoreUnknownPicFmt*, but for controls
rvfoIgnoreUnknownCtrlProperties	If set, TRichView ignores errors occurred while reading inserted controls
rvfoConvLargeImageIdxToZero	▪ If set, too large (>=Imagelist.Count) image indices for *bullets* and *hotspots* are converted to 0. ▪ if not set, RVF reading methods return False (failure value) in such case.
rvfoConvUnknownStylesToZero	▪ If set, TRichView converts too large (>= RichView.Style.TextStyles[437].Count, or RichView.Style.ParaStyles[431].Count) style indices to 0. ▪ If not set, RVF reading methods return False (failure value) in such case.

rvfoLoadLayout	If not set, TRichView ignores layout information saved in RVF (see *rvfoSaveLayout* above)
rvfoLoadDocProperties	If set, DocProperties[176] and DocParameters[176] are loaded from RVF (but not by methods for RVF insertion) (see *rvfoSaveDocProperties* above)
rvfoCanChangeUnits	If set, the component can change Style[418].Units[437] to the value read from RVF. This change is possible only when RVF is loaded, not inserted.
rvfoLoadDocObjects	If set, DocObjects[174] are loaded from RVF. (see *rvfoSaveDocObjects* above)
rvfoIgnoreGraphicClasses	If set, names of graphic classes specified in RVF are ignored. File format is determined by content, and the default graphic class for this file format is used (see TRVGraphicHandler[654]) If not set, the component tries to use a graphic class specified in RVF. If this class is not found, TRichView works as if this option is specified. The procedure described above is used for graphic items (pictures[122] and hot pictures[125]), background images of tables[567] and table cells[605]. For pictures of list markers[513], the specified graphic class is always used.

See RVF Overview[93] for details.

Default value:

[*rvfoSavePicturesBody, rvfoSaveControlsBody, rvfoSaveBinary, rvfoSaveDocProperties, rvfoLoadDocProperties, rvfoSaveDocObjects, rvfoLoadDocObjects*];

Default value for components, placed on form at design time (can be changed in TRichView component editor)[168]:

[*rvfoSavePicturesBody, rvfoSaveControlsBody, rvfoSaveBinary, rvfoSaveDocProperties, rvfoLoadDocProperties, rvfoSaveTextStyles, rvfoSaveParaStyles, , rvfoSaveDocObjects, rvfoLoadDocObjects*];

See also:

- RVFWarnings[191]

1.5.1.1.2.35 TRichView.RVFParaStylesReadMode

Specifies how paragraph styles[431] and list styles[429] are loaded from RVF files or streams.

```
property RVFParaStylesReadMode: TRVFReaderStyleMode[679];
```

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode[679] type for details.

Paragraph styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFTextStylesReadMode[191];
- RVFOptions[188].

See also:

- RVF Overview[93]

1.5.1.1.2.36 TRichView.RVFTextStylesReadMode

Specifies how text styles[437] are loaded from RVF files or streams.

```
property RVFTextStylesReadMode: TRVFReaderStyleMode[679];
```

(Introduced in version 1.5)

See topic on TRVFReaderStyleMode[679] type for details.

Text styles are saved in RVF optionally.

Default value:

rvf_sInsertMerge

See properties:

- RVFParaStylesReadMode[190];
- RVFOptions[188].

See also:

- RVF Overview[93].

1.5.1.1.2.37 TRichView.RVFWarnings

Information about the last performed RVF reading operation.

```
type
  TRVFWarning = (
    rvfwUnknownPicFmt, rvfwUnknownCtrls,
    rvfwConvUnknownStyles, rvfwConvLargeImageIdx,
    rvfwConvToUnicode, rvfwConvFromUnicode,
    rvfwInvalidPicture, rvfwUnknownCtrlProperties, rvfwConvUnits);
  TRVFWarnings = set of TRVFWarning;

property RVFWarnings: TRVFWarnings;
```

Note: TRVFWarning and TRVFWarnings are defined in RVStyle unit.

You can use this property to know about warnings or errors that may be occurred during the last reading from RVF.

Depending on RichView.RVFOptions[188] some cases could be considered as warnings (and reading method was successfully finished) or errors.

Warning	Meaning
rvfwUnknownPicFmt	Pictures of unknown classes exist in RVF. See *rvfoIgnoreUnknownPicFmt.*
rvfwUnknownCtrls	Controls of unknown classes exist in RVF. See *rvfoIgnoreUnknownCtrls.*
rvfwConvLargeImageIdx	*Bullets* or *hotspots* with too large image indices exist in RVF. See *rvfoConvLargeImageIdxToZero.*
rvfwConvUnknownStyles	Text, paragraph or list styles with too large indices exist in RVF. See *rvfoConvUnknownStylesToZero.*
rvfwConvToUnicode *rvfwConvFromUnicode*	There was a conversion from ANSI text in RVF file to Unicode in RichView or vice versa. This is a result of loading RVF file with set of styles different from one which was used when saving. A conversion is performed automatically, without generating errors.

rvfwInvalidPicture	Some pictures were replaced with InvalidPicture[427].
rvfwUnknownCtrlProperties	There was error when reading inserted controls (for example, unknown properties)
rvfwConvUnits	RVF file has different units than Style[418].Units[437], and a conversion of units was made. This warning is not set if units were changed because of *rvfoCanChangeUnits* included in RVFOptions[188]. A conversion is performed automatically, without generating errors.

See RVF Overview[93] for details.

Methods reading RVF:

- LoadRVFFromStream[245];
- InsertRVFFromStream[240];
- AppendRVFFromStream[208];
- LoadRVF[245].

Methods of RichViewEdit reading RVF:

- InsertRVFFromFileEd[357];
- InsertRVFFromStreamEd[358];

(also PasteRVF[364] and Paste[361], but in this version you can't detect when Clipboard reading operation is finished).

1.5.1.1.2.38 TRichView.SelectionHandlesVisible

Shows or hides selection handles for using on touch screens.

```
property SelectionHandlesVisible: Boolean;
```

(introduced in version 15)

This property exists when the component is compiled in Delphi 2010 or newer.

By default, selection handles are shown if the selection was created by a double or a triple tap (double tap selects a word, triple tap selects a paragraph). They are hidden automatically when the selection becomes empty.

You can use this property to show or hide handles manually. Assigning *True* to this property can show handles even if a selection is empty.

For example, if you want to show handles even on a single tap, assign SelectionHandlesVisible=True in OnRVMouseUp[291], if ssTouch is included in the Shift parameter of this event.

Default value:

True

See also properties of TRVStyle:

- SelectionHandleKind[431]

1.5.1.1.2.39 TRichView.SingleClick

```
property SingleClick : Boolean;
```

Obsolete, please use *rvoSingleClick* in Options[182];

1.5.1.1.2.40 TRichView.Style

Style of TRichView: 

```
property Style: TRVStyle[418];
```

This property provides link to TRVStyle component.

TRVStyle define many properties for document appearance, and, the most important, it contains collections of text, paragraph and list styles for TRichView document.

RichView can display its contents only if its style is defined.

1.5.1.1.2.41 TRichView.StyleTemplateInsertMode

Controls how Style[193].StyleTemplates[435] affect insertion of RTF and RVF files or streams.

```
type
  TRVStyleTemplateInsertMode =
    (rvstimUseTargetFormatting, rvstimUseSourceFormatting, rvstimIgnoreSourceStyleTemplat
property StyleTemplateInsertMode: TRVStyleTemplateInsertMode;
```

(introduced in version 14)

This property is used only if UseStyleTemplates[195]=*True*.

Mode	Meaning
rvstimUseTargetFormatting	Use formatting of the current (target) document
rvstimUseSourceFormatting	Use formatting of the inserted (source) document
rvstimIgnoreSourceStyleTemplates	Ignore styles in the inserted (source) document.

1) rvstimUseTargetFormatting

If the source document and the target document have styles templates of the same name, text (and paragraphs) of this style template is formatted according to the target style template.

For example, let the source document has a style template named "heading 1" defining text color=red, and some text formatted as "heading 1"+bold, i.e. it looks red and bold. Let the target document has a style template of the same name, defining text color=black. When this text is inserted, it is formatted as "heading 1"+bold, i.e. it looks black+bold.

Only style templates having new names are added from the source document to the target document.

2) rvstimUseSourceFormatting

If the source document and the target document have style templates of the same name, text (and paragraphs) of this style template is formatted according to the source style template.

In the example above, this text is inserted as "heading 1"+red+bold, so it looks like in the source document, despite of linking to the target document's "heading 1" instead of the source document's "heading 1".

Only style templates having new names are added from the source document to the target document.

3) rvstimIgnoreSourceStyleTemplates

Style templates from the source document are ignored. The inserted paragraphs are linked to the target document's "Normal" style template[461] (if it exists). The inserted text is unstyled, except for hyperlinks[479] (they are linked to the target document's "Hyperlink" style, if it exists).

In the example above, this text is inserted as "Normal"+red+bold, so it looks like in the original document.

Note

When inserting unstyled paragraphs, the "Normal" style template is applied to them. To insert them unstyled in modes (1) and (2), you can assign the global variable RichViewAutoAssignNormalStyleTemplate[700] = *False*. In the mode (3), "Normal" is always applied.

Default value

rvstimUseTargetFormatting

See also:

- Styles and style templates[74].

1.5.1.1.2.42 TRichView .TabNavigation

Mode of Tab and Shift+Tab navigation

```
type
  TRVTabNavigationType =
    (rvtnNone, rvtnTab, rvtnCtrlTab)

property TabNavigation: TRVTabNavigationType;
```

(introduced in version 1.3)

Mode	Meaning
rvtnNone	No special actions for Tab (default for TRichViewEdit). A focus (dotted) rectangle is not drawn around hyperlinks, it's impossible to open hyperlink with Enter.
rvtnTab	Tab-navigation through links and controls using Tab and Shift+Tab (default for TRichView);
rvtnCtrlTab	Tab navigation through links using Ctrl+Tab and Ctrl+Shift+Tab. Has a limited application since users still need to use Tab and Shift+Tab when the focus is set on inserted control.

Note: a special tab navigation is not supported in editor. Although it has TabNavigation property, it's impossible to change its value from rvtnNone.

See also:

- GetFocusedItem[223] method.

1.5.1.1.2.43 TRichView .TopMargin

Top margin of document

```
property TopMargin: Integer;
```

This value is measured in "screen" pixels, so when printing it's recalculated according to the printer resolution, and margins on pages look like margins on the screen.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

See also properties:

- LeftMargin[180];
- RightMargin[185];
- BottomMargin[171].

See also properties of TRVPrint:

- Margins[520].

See also:

Additional information about horizontal scrolling, margins, indents, etc.[80]

1.5.1.1.2.44 TRichView.UseStyleTemplates

Allows or disallows using Style[193].StyleTemplates[435].

```
property UseStyleTemplates: Boolean;
```

(introduced in version 14)

If UseStyleTemplates=*True*:

- StyleTemplates are saved in RVF (also requires both *rvfoSaveTextStyles* and *rvfoSaveParaStyles* included in RVFOptions[188]);
- StyleTemplates are loaded from RVF (also requires RVFTextStylesReadMode[191] =RVFParaStylesReadMode[190]=*rvf_sInsertMerge*);
- StyleTemplates are saved in RTF and DocX (as a style sheet);
- StyleTemplates are loaded from RTF (also requires RTFReadProperties[187].TextStyleMode[309] =RTFReadProperties.ParaStyleMode[308]=*rvrsAddIfNeeded*) ;
- RTF and RVF files/streams are inserted according to StyleTemplateInsertMode[193];
- "Normal" style template (if it exists) is assigned to unstyled paragraphs when loading RTF and RVF files/streams (see RichViewAutoAssignNormalStyleTemplate[700]);
- the component provides that TextStyle.ParaStyleTemplateId[473]=ParaStyle.StyleTemplateId[465] for all text items[120] and non-text items linked to a text style (tabs[121], labels[133], sequences[134], footnotes[136], endnotes[135], note references[140]); if they are not equal, the component assigns another text style to this item, with the correct ParaStyleTemplateId, adding this text style to Style[418].TextStyles[437] if necessary.
- in editor, when **Enter** is pressed, the next paragraph is formatted according to NextId[496] property of the current paragraph's style template

Default value

False

See also:

- Styles and style templates[74].

1.5.1.1.2.45 TRichView.UseVCLThemes

Allows using colors of VCL themes.

```
property UseVCLThemes: Boolean;
```

(introduced in version 17)

If *True*, all colors in document are changed according to the current VCL theme.

This property is ignored in Delphi versions prior to XE2.

Default value

False

1.5.1.1.2.46 TRichView.VAlign

Vertical align of the document relative to the control window: top, center, or bottom.

```
property VAlign: TTextLayout;
```

(introduced in v1.9)

This property is in effect if height of the document is less than height of the TRichView window.

See also:

- ClientToDocument[211] method.

1.5.1.1.2.47 TRichView.VSmallStep

Number of pixels in one step of vertical scrollbar

```
property VSmallStep: Integer;
```

When VScrollPos[547] is changes by 1, the window is scrolled by VSmallStep pixels.

This is one "RVSU", see Scrolling in RichView[80]

Default value:

10

See also properties:

- VScrollPos[547];
- VScrollMax[547];
- VScrollVisible[547].

1.5.1.1.2.48 TRichView.WordWrap

Allows to disable word wrapping.

```
property WordWrap: Boolean;
```

(introduced in version 10)

If *True*, documents are formatted normally.

If *False*, all word wrapping is disabled. Unlike *rvpaoNoWrap* option[487] of paragraph style, this property should be considered as a control (view) property, not as a document property. It is not stored in RVF[93].

Default value:

True

1.5.1.1.3 Methods

1.5.1.1.3.1 TRichView.AddBreak

Adds a *break*[125] (a horizontal line) to the end of document.

```
procedure AddBreak(Width: TRVStyleLength[690] = 0;
  Style: TRVBreakStyle[669] = rvbsLine; Color: TColor = clNone;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
```

(changed in version 18)

Parameters:

Width – line width (or rectangle height). This value is measured in Style[193].Units[437]. If **Width** = 0, the new item has 1 pixel width (converted to Style[193].Units[437])

Style – break type;

Color – color of line. If it is equal to *clNone*, the *break* color is Style[193].TextStyles[437][0].Color[469].

Tag – *tag*[68] of this *break*.

For default HTML <hr> appearance, parameters must be: **Style**=*rvbs3d*, **Width**=2.

Methods type: viewer-style.

See also methods:

- Format[217];
- FormatTail[218].

See also methods of TRichViewEdit:

- InsertBreak[350].

See also:

- Item types 64;
- Other methods for appending items 77;
- "Tags" 68.

Related deprecated methods:

```
procedure AddBreakExTag(Width: TRVStyleLength690; Style: TRVBreakStyle669;
  Color: TColor; const Tag: TRVTag692);
procedure AddBreakEx(Width: TRVStyleLength690; Style: TRVBreakStyle669;
  Color: TColor);
procedure AddBreakTag(const Tag: TRVTag692);
```

These methods provide subset of functionality of AddBreak.

1.5.1.1.3.2 TRichView.AddBullet

Adds a *bullet* 128 (an image from TImageList) to the end of document.

```
procedure AddBullet(const Name: TRVUnicodeString694; ImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer = -1;
  VAlign: TRVVAlign695 = rvvaBaseline; const Tag: TRVTag692 = RVEMPTYTAG714);
```

(changed in version 18)

Alternatives:

- If you want to add a hyperlink-picture from image list, add a 200 *hotspot* 200 instead.
- If you want to add a TGraphic picture, add a picture 204 instead.
- If you want to add a TGraphic picture-hyperlink, add a 199 *hot picture* 199 instead.

Parameters:

Name – name of *bullet*, any string. **Name** must not contain CR and LF characters. TRichView does not use item names itself, they are for your own use.

ImageIndex – index of image in **ImageList**.

ImageList – image list for this *bullet*. TRichView does not own, does not copy and does not destroy image lists, it just holds pointers to them. So this image list is not destroyed when the document is cleared.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style 193.ParaStyles 431[**ParaNo**].

Tag – *tag* 68 of this *bullet*.

Method type: viewer-style.

See also methods:

- Format 217;
- FormatTail 218;
- SetAddParagraphMode 259.

See also methods of TRichViewEdit:

- InsertBullet 351.

See also properties of TRVStyle:

- ParaStyles 431.

See also:

- Item types 64;
- Other methods for appending items 77;
- "Tags" 68.

Related deprecated methods:

```
procedure AddBulletExTag(const Name: TRVUnicodeString694;
```

```
  ImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer; const Tag: TRVTag[692]);
procedure AddBulletEx(const Name: TRVUnicodeString[694]; ImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer);
```

These methods provide subset of functionality of AddBullet.

1.5.1.1.3.3 TRichView.AddControl

These methods add Delphi control[126] to the end of document.

```
procedure AddControl(const Name: TRVUnicodeString[694]; ctrl: TControl;
  ParaNo: Integer = -1; VAlign: TRVVAlign[695] = rvvaBaseline;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
```

(changed in version 18)

Parameters:

Name – name of this control item, any string. **Name** must not contain CR and LF characters. TRichView does not use item names itself, they are for your own use. Do not confuse with **ctrl**.Name property.

ctrl – control to insert.

VAlign – vertical align of this control, relative to its line, see TRVVAlign[695] for possible values.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style[193].ParaStyles[431][**ParaNo**].

Tag – *tag*[68] of this control item. Do not confuse with **ctrl**.Tag property.

Methods type: viewer-style.

See also methods:

- Format[217];
- FormatTail[218];
- SetAddParagraphMode[259].

See also methods of TRichViewEdit:

- InsertControl[352].

See also properties of TRVStyle:

- ParaStyles[431].

See also:

- Item types[64];
- Other methods for appending items[77];
- "Tags"[68].

Related deprecated methods:

```
procedure AddControlExTag(const Name: TRVAnsiString[668]; ctrl: TControl;
  ParaNo: Integer; VAlign: TRVVAlign[695]; const Tag: TRVTag[692]);
procedure AddControlEx(const Name: TRVAnsiString[668]; ctrl: TControl;
  ParaNo: Integer; VAlign: TRVVAlign[695]);
```

These methods provide subset of functionality of AddControl.

1.5.1.1.3.4 TRichView.AddFmt

Adds a formatted string assembled from a Pascal format string and an array arguments, using the **StyleNo**-th text style and the **ParaNo**-th paragraph style, to the end of document.

```
procedure AddFmt(const FormatStr: String;
  const Args: array of const; StyleNo, ParaNo: Integer);
```

AddFmt(FormatStr, Args, StyleNo, ParaNo) is equivalent to AddNL[203](Format(FormatStr,Args), StyleNo, ParaNo).
(where Format is not a method of TRichView, but SysUtils.Format)

Note: TRichView allows empty text strings only in empty paragraphs (may be with list makers). See Valid documents[105].

Parameters:

FormatStr – format string. See "Format function" in the Delphi help file for syntax. **FormatStr** must not contain CR, LF and TAB characters. To add several lines of text use AddTextNL[206].

Args – arguments for format string.

StyleNo is an index in the TextStyles[437] collection of the linked[193] RVStyle component, or rvsDefStyle[711] constant. It defines font attributes for the text.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style[193].ParaStyles[431][**ParaNo**].

Method type: viewer-style.

Unicode note: the parameter of this method is Unicode string for Delphi/C++Builder 2009 or newer, and ANSI string for older versions of Delphi/C++Builder.

Example:

```
MyRichView.AddFmt('X=%d and Y=%d', [X,Y], 0, 0);
```

If style templates[195] are used, the component may automatically change text style indices of text items, to provide a consistency of text and paragraph style templates.

See also methods:

- AddNL[203];
- AddTextNL[206];
- Format[217];
- FormatTail[218];
- SetAddParagraphMode[259].

See also properties of TRVStyle:

- TextStyles[437];
- ParaStyles[431];
- SpacesInTab[435].

See also:

- Other methods for appending items[77].

1.5.1.1.3.5 TRichView.AddHotPicture

Adds picture-hyperlink[125] to the end of document.

```
procedure AddHotPicture(const Name: TRVUnicodeString[694]; gr: TGraphic;
  ParaNo: Integer = -1; VAlign: TRVVAlign[695] = rvvaBaseline;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
```

(introduced in version 1.5; changed in version 18)

The same as AddPicture[204], but this picture will be a hypertext link.

Parameters:

Name – name of this *hot-picture* item, any string. **Name** must not contain CR and LF characters. TRichView does not use Names itself, they are for your own use.

gr – picture to insert. By default, this picture will be owned by TRichView control, and you must not free it. However, you can specify nonzero *rvepShared* extra item property 673, and this graphic object will be shared (you need to free it yourself after TRichView is cleared).

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount 702). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign 695 for possible values.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style 193.ParaStyles 431[**ParaNo**].

Tag – *tag* 68 of this *hot-picture* item..

Alternatives:

- If you want to add a simple (not a hyperlink) TGraphic picture, add a picture 204 instead.
- If you want to add a hyperlink-picture from image list, add a 200*hotspot* 200 instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a 197*bullet* 197 instead.

Methods type: viewer-style.

See also methods:

- Format 217;
- FormatTail 218;
- SetAddParagraphMode 259.

See also methods of TRichViewEdit:

- InsertHotPicture 352.

See also properties of TRVStyle:

- ParaStyles 431.

See also:

- Item types 64;
- Other methods for appending items 77;
- "Tags" 68;
- Hypertext in TRichView 69.

Related deprecated methods:

```
procedure AddHotPictureTag(const Name: TRVUnicodeString 694;
  gr: TGraphic; ParaNo: Integer;
  VAlign: TRVVAlign 695; const Tag: TRVTag 692);.
```

1.5.1.1.3.6 TRichView .AddHotspot

Adds a *hotspot* 129 (an image from TImageList – hypertext link) to the end of document.

```
procedure AddHotspot(const Name: TRVUnicodeString 694;
  ImageIndex, HotImageIndex: Integer; ImageList: TCustomImageList;
  ParaNo: Integer = -1; VAlign: TRVVAlign 695 = rvvaBaseline;
  const Tag: TRVTag 692 = RVEMPTYTAG 714);
```

(changed in version 18)

Alternatives:

- If you want to add a simple (not a hyperlink) picture from image list, add a 197*bullet* 197 instead.
- If you want to add a TGraphic picture-hyperlink, add a 199*hot picture* 199 instead.
- If you want to add a TGraphic picture, add a picture 204 instead.

Parameters:

Name – name of *hotspot*, any string. **Name** must not contain CR and LF characters. TRichView does not use item names itself, they are for your own use.

ImageIndex – index of image in **ImageList**.

HotImageIndex – index of "hot" image in **ImageList**. It is displayed when the mouse pointer is above this *hotspot*.

ImageList – image list for this *hotspot*. TRichView does not own, does not copy and does not destroy image lists, it just holds pointers to them. So this image list is not destroyed when the document is cleared.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style[193].ParaStyles[431][**ParaNo**].

Tag – *tag*[68] of this *hotspot*.

Methods type: viewer-style.

See also methods:

- Format[217];
- FormatTail[218];
- SetAddParagraphMode[259].

See also methods of TRichViewEdit:

- InsertHotspot[353].

See also properties of TRVStyle:

- ParaStyles[431].

See also:

- Item types[64];
- Other methods for appending items[77];
- "Tags"[68];
- Hypertext in RichView[69].

Related deprecated methods:

```
procedure AddHotspotExTag(const Name: TRVUnicodeString[694];
  ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer;
  const Tag: TRVTag[692]);
procedure AddHotspotEx(const Name: TRVUnicodeString[694];
  ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList; ParaNo: Integer);
```

These methods provide subset of functionality of AddHotspot.

1.5.1.1.3.7 TRichView.AddItem

Adds one item to the end of document.

```
procedure AddItem(const Text: TRVUnicodeString[694]; Item: TCustomRVItemInfo[563]);
```

(introduced in version 1.4; changed in version 18)

This is the most general method for appending items to the document. Using this method you can add items of custom types.

For items of standard types it's recommended to use other methods for appending items[77].

Usually this method is used for appending tables[130], labels[133], sequences[134], footnotes[136], endnotes[135], references to notes[140].

The detailed description of item types is beyond the scope of this help file. Briefly, standard item types are declared and implemented in RVItem unit. All item types are classes derived from one base class – TCustomRVItemInfo 563.

Parameters:

Text – item name, some text associated with this item. It must not contain CR and LF characters.

Item – the item itself. **Item**.ParaNo defines the paragraph style of item (index in Style.ParaStyles 431)

Method type: viewer-style.

See also methods:

- Format 217;
- FormatTail 218;
- SetAddParagraphMode 259;
- GetItem 225.

See also methods of TRichViewEdit:

- InsertItem 353.

See also:

- Item types 64;
- Other methods for appending items 77.

1.5.1.1.3.8 TRichView .AddCheckpoint

Adds a *checkpoint* 65 to the end of document.

```
function AddCheckpoint(const CpName: TRVUnicodeString 694 = '';
  RaiseEvent: Boolean = False; const CpTag: TRVTag 692 = RVEMPTYTAG 714): Integer;
```

Checkpoints are not items, but the methods for appending *checkpoints* are similar to methods for appending items. This *checkpoint* will be associated with the next appended item.

Parameters:

CpName – name of *checkpoint*. It must not contain #0, #1 and #2 characters.

RaiseEvent – "raise event" flag, see OnCheckpointVisible 274

Tag – *tag* of *checkpoint*.

Return value:

Index of the added checkpoint

Methods type: viewer-style.

See also methods:

- SetCheckpointInfo 261;
- RemoveCheckpoint 248.

See also methods of TRichViewEdit:

- SetCheckpointInfoEd 373;
- SetCurrentCheckpointInfo 376.

See also:

- Methods for appending items 77;
- *Checkpoints* 65;
- *Tags* 68.

Related deprecated methods:

```
function AddNamedCheckpointExTag(const CpName: TRVUnicodeString 694;
  RaiseEvent: Boolean; const Tag: TRVTag 692): Integer;
```

```
function AddCheckpoint: Integer;
function AddCheckpointTag(const Tag: TRVTag[692]): Integer;
function AddNamedCheckpoint(const CpName: TRVUnicodeString[694]): Integer;
function AddNamedCheckpointEx(const CpName: TRVUnicodeString[694];
  RaiseEvent: Boolean): Integer;
```

These methods provide subset of functionality of AddCheckpoint.

1.5.1.1.3.9 TRichView .AddNL -A -W

These methods add one text item[120] with the **StyleNo**-th text style and the **ParaNo**-th paragraph style, with the specified **Tag** to the end of document.

```
procedure AddNL(const s: String; StyleNo: Integer; ParaNo: Integer = -1;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
procedure AddNLW(const s: TRVUnicodeString[694]; StyleNo, ParaNo: Integer;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
procedure AddNLA(const s: TRVAnsiString[668]; StyleNo, ParaNo: Integer;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
procedure Add(const s: String; StyleNo:Integer);
```

(introduced in version 1.3 and 1.4; changed in version 18)

Parameters:

s is a text string to add. It must not contain CR, LF, TAB, FF characters (#13, #10, #9, #12). To add several lines of text use AddTextNL[206].

StyleNo is an index in the TextStyles[437] collection of the linked[193] RVStyle component, or *rvsDefStyle*[711] constant. It defines font attributes for the text.

If **ParaNo**=-1, the methods add an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style[193].ParaStyles[431][**ParaNo**].

Tag – *tag*[68] of this text item.

Add provides subsets of functionality of **AddNL**: Add(...) is equivalent to AddNL..., -1, '').

Note: TRichView allows empty text strings only in empty paragraphs (may be with list makers). See Valid documents[105].

Unicode notes:

Internally, text is stored as Unicode. **AddNLA** (and **AddNL** in non-Unicode versions of Delphi) converts ANSI to Unicode. A code page for conversion is calculated basing on Style[193].TextStyles[437][**StyleNo**]. Charset[468]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[193].DefCodePage[421] is used.

Methods type: viewer-style

If style templates[195] are used, the component may automatically change text style indices of text items, to provide a consistency of text and paragraph style templates.

See also methods:

- AddTab[205];
- Format[217];
- FormatTail[218];
- SetAddParagraphMode[259].

See also methods of TRichViewEdit:

- InsertStringTag, InsertStringATag, InsertStringWTag[359].

See also properties of TRVStyle:

- TextStyles[437];
- SpacesInTab[435].

See also:

- Other methods for appending items [77];
- Unicode in RichView [99].

Related deprecated methods:

```
procedure AddNLTag(const s: String; StyleNo, ParaNo: Integer; const Tag: TRVTag);
procedure AddTag(const s: String; StyleNo: Integer; const Tag: TRVTag);
procedure AddNLATag(const s: TRVAnsiString; StyleNo, ParaNo: Integer; const Tag: TRVTag);
procedure AddNLWTag(const s: TRVUnicodeString; StyleNo, ParaNo: Integer; const Tag: TRVTag)
```

1.5.1.1.3.10 TRichView .AddPicture

Adds a picture [122] to the end of document

```
procedure AddPicture(const Name: TRVUnicodeString[694]; gr: TGraphic;
  ParaNo: Integer = -1; VAlign: TRVVAlign[695] = rvvaBaseline;
  const Tag: TRVTag[692] = RVEMPTYTAG[714]);
```

Parameters:

Name – name of this picture item, any string. **Name** must not contain CR and LF characters. TRichView does not use item names itself, they are for your own use.

gr – picture to insert. By default, this picture will be owned by TRichView control, and you must not free it. However, you can specify nonzero *rvepShared* extra item property [673], and this graphic object will be shared (you need to free it yourself after TRichView is cleared).

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount [702]). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign [695] for possible values.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style [193].ParaStyles [431][**ParaNo**].

Tag – *tag* [68] of this picture item.

Alternatives:

- If you want to add a TGraphic picture-hyperlink, add a [199] *hot picture* [199] instead.
- If you want to add a hyperlink-picture from image list, add a [200] *hotspot* [200] instead.
- If you want to add a simple (not a hyperlink) picture from image list, add a [197] *bullet* [197] instead.

Methods type: viewer-style.

Example 1:

```
procedure AddBitmapFromFile(rv: TCustomRichView;
  const FileName: String; ParaNo: Integer);
var bmp: TBitmap;
begin
  bmp := TBitmap.Create;
  bmp.LoadFromFile(FileName);
  rv.AddPicture('', bmp, ParaNo);
end;
...
// creating document with 2 pictures
// in the same paragraph
rv.Clear[210];
AddBitmapFromFile(rv, 'd:\pic1.bmp', 0);
AddBitmapFromFile(rv, 'd:\pic2.bmp', -1);
rv.Format[217];
```

Example 2: adding shared images

```
procedure AddBitmapFromFile(rv: TCustomRichView;
  const FileName: String; ParaNo: Integer);

bmp := TBitmap.Create;
bmp.LoadFromFile('c:\pic.bmp');
rv.AddPicture('', bmp, 0);
rv.SetItemExtraIntProperty(rv.ItemCount-1, rvepShared, 1);
rv.AddPicture('', bmp, -1);
rv.SetItemExtraIntProperty(rv.ItemCount-1, rvepShared, 1);
rv.Format;
// you must free bmp yourself when rv is cleared.
```

See also methods:

- Format;
- FormatTail;
- SetAddParagraphMode.

See also methods of TRichViewEdit:

- InsertPicture.

See also properties of TRVStyle:

- ParaStyles.

See also:

- Item types;
- Other methods for appending items;
- "Tags".

Related deprecated methods:

```
procedure AddPictureExTag(const Name: TRVUnicodeString; gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign; const Tag: TRVTag);
procedure AddPictureEx(const Name: TRVUnicodeString; gr: TGraphic;
  ParaNo: Integer; VAlign: TRVVAlign);
```

These methods provide subset of functionality of AddPicture.

1.5.1.1.3.11 TRichView.AddTab

Adds tabulator to the end of document.

```
procedure AddTab(TextStyleNo, ParaNo: Integer);
```

(introduced in v1.9)

Parameters:

TextStyleNo – text style of tabulator. It is an index in the TextStyles collection of the linked RVStyle component, or *rvsDefStyle* constant. It defines font attributes.

If **ParaNo**=-1, the method adds an item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style.ParaStyles[**ParaNo**].

This method always adds a tabulator, regardless of RVStyle.SpacesInTab mode. If RVStyle.SpacesInTab =0, this method does the same work as AddTextNL(#9, TextStyleNo, ParaNo, ParaNo).

Methods type: viewer-style.

See also methods:

- Format;
- FormatTail;
- SetAddParagraphMode.

See also methods of TRichViewEdit:

- InsertTab 359.

See also properties of TRVStyle:

- ParaStyles 431.

See also:

- Item types 64;
- Other methods for appending items 77.

1.5.1.1.3.12 TRichView .AddTextNL -A -W

These methods add one or more text 120 lines to the end of document.

```
procedure AddTextNL(const s: String;
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag 692 = RVEMPTYTAG 714);
procedure AddTextNLA(const s: TRVAnsiString 668;
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  AsSingleParagraph: Boolean = False;
  const Tag: TRVTag 692 = RVEMPTYTAG 714;
  CodePage: TRVCodePage = CP_ACP);
procedure AddTextNLW(const S: TRVUnicodeString 694;
  StyleNo, FirstParaNo, OtherParaNo: Integer;
  DefAsSingleParagraph: Boolean = False;
  const Tag: TRVTag 692 = RVEMPTYTAG 714);
```

(changed in version 18)

Parameters:

s is a text string to add. It may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators 121, depending on value of SpacesInTab 435 property of the linked 193 RVStyle component. #12 characters add page breaks 184. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported.

StyleNo is an index in the TextStyles 437 collection of the linked 193 RVStyle component, or *rvsDefStyle* 711 constant. It defines font attributes for the text.

If **ParaNo**=-1, the method adds the first line of text to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph using attributes defined in Style 193.ParaStyles 431[**ParaNo**].

OtherParaNo defines paragraph attributes for the subsequent lines of text. Must be >=0.

DefAsSingleParagraph: If *False*, **AddTextNLW** uses the current add-paragraph mode (see SetAddParagraphMode 259). If *True*, it treats #13 and #10 as line breaks, not paragraph breaks. WideChar($2029) ("paragraph separator") always initiates a new paragraph. WideChar($2028) ("line separator") always initiates a new line.

AsSingleParagraph: If *False*, **AddTextNLA** uses the current add-paragraph mode (see SetAddParagraphMode 259). If *True*, it treats #13 and #10 as line breaks, not paragraph breaks.

Tag – *tag* 68 of this picture item.

AddTextNLW supports the Unicode byte order marks characters (if it's present, it must be the first character in **s**).

Methods type: viewer-style.

Unicode notes:

Internally, text is stored as Unicode. **AddTextNLA** (and **AddTextNL** in non-Unicode versions of Delphi) converts ANSI string to Unicode using **CodePage**.

If CodePage = CP_ACP, Style[193].DefCodePage[421] is used instead.

If style templates[195] are used, the component may automatically change text style indexes of text items, to provide a consistency of text and paragraph style templates.

See also methods:

- AddNL and others[203];
- AddTab[205];
- LoadText, LoadTextW[246];
- LoadTextFromStream, LoadTextFromStreamW[247];
- Format[217];
- FormatTail[218];
- SetAddParagraphMode[259].

See also methods of TRichViewEdit:

- InsertText, -A, -W[360].

See also properties of TRVStyle:

- TextStyles[437];
- SpacesInTab[435].

See also:

- Other methods for appending items[77];
- Unicode in RichView[99];
- Example how to load UTF-8 files[316].

1.5.1.1.3.13 TRichView.AppendFrom

Adds content of another RichView component (**Source**) to the end of document.

```
procedure AppendFrom(Source: TCustomRichView[162]);
```

This method does not copy items of some types (including inserted controls[126] and tables[130]) (it just ignores them).

This method is the fastest way to copy items from one TRichView to another, but it has the limitations listed above.

The recommended way of copying is:

```
var Stream: TMemoryStream;
begin
  Stream := TMemoryStream.Create;
  try
    Source.SaveRVFToStream[254](Stream, False);
    Stream.Position := 0;
    Dest.LoadRVFFromStream[245](Stream);
    Dest.Format[217];
  finally
    Stream.Free;
  end;
end;
```

Method type: viewer-style.

See also methods:

- Format[217];
- FormatTail[218].

See also:

- Other methods for appending items[77].

1.5.1.1.3.14 TRichView .AppendRVFFromStream

Adds data from Stream in RichView Format (RVF) 93 to the end of document.

```
function AppendRVFFromStream(Stream: TStream;
  ParaNo: Integer): Boolean;
```

This method appends content from RVF stream. It ignores the paragraph style of first paragraph in RVF and uses the **ParaNo**-th paragraph style instead of it.

Parameters:

Stream – stream containing document in RichView Format (RVF).

If **ParaNo**=-1, the method adds the first item to the end of the last paragraph. If **ParaNo**>=0, this item starts a new paragraph with the **ParaNo**-th style. (**ParaNo** is an index in the ParaStyles 431 collection of the linked 193 RVStyle component). This parameter affects only the first paragraph of inserted data. Styles of other paragraphs are defined in RVF.

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor 168.

If style templates are used 195, and RVFTextStylesReadMode 191=RVFParaStylesReadMode 190 =*rvf_sInsertMerge*, and StyleTemplateInsertMode 193<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style 193.StyleTemplates 435, and reads text and paragraph styles according to StyleTemplateInsertMode 193. The method calls OnStyleTemplatesChange 302 event.

Return value:

"Was reading successful?"

See also properties:

- RVFOptions 188;
- RVFWarnings 191;
- RVFTextStylesReadMode 191 (this method type: RVF insertion);
- RVFParaStylesReadMode 190 (this method type: RVF insertion);
- UseStyleTemplates 195.

See also events:

- OnRVFImageListNeeded 288;
- OnRVFControlNeeded 288;
- OnRVFPictureNeeded 289;
- OnControlAction 275 (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange 302.

See also methods:

- InsertRVFFromStream 240;
- LoadRVFFromStream 245;
- LoadRVF 245;
- Format 217;
- FormatTail 218.

See also:

- TRichView component editor 168;
- Saving and loading 92;
- RVF overview 93;
- Paragraphs 72.

1.5.1.1.3.15 TRichView.AssignSoftPageBreaks

Displays soft page breaks (i.e. page breaks that were made automatically)

```
procedure AssignSoftPageBreaks(RVPrint: TComponent);
```

(introduced in version 1.7)

Page breaks become visible after calling RichView.AssignSoftPageBreaks(RVPrint). **RVPrint** must be formatted (i.e. FormatPages 525 must be called before this method). Page breaks are visible only if *rvoShowPageBreaks* is included in RichView.Options 182.

In editor 316, soft page breaks will be automatically cleared after any editing operation (any action invoking OnChange 390). Page breaks will also be cleared when you call Clear 210. In all other cases, you must call ClearSoftPageBreaks 211 after each change in the document performed after calling AssignSoftPageBreaks.

Soft page breaks are drawn using RVStyle.SoftPageBreakColor 434 (or in OnDrawPageBreak 446 event).

See also:

- ClearSoftPageBreaks 211.

See also properties:

- PageBreaksBeforeItems 184.

1.5.1.1.3.16 TRichView.BeginOleDrag

Starts dragging the selection.

```
procedure BeginOleDrag;
```

(introduced in v1.9)

This method is useful if you want to implement drag&drop 89 of inserted controls 126.(all other content can be dragged automatically)

Example

This example implements drag&drop of inserted TButton controls.

1) Add the following variables in form:

```
ClickedControl: TObject;
ClickPoint: TPoint;
```

2) Add the following methods:

```
procedure TForm1.OnControlMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then begin
    RichViewEdit1.SelectControl(TControl(Sender));
    ClickedControl := Sender;
    ClickPoint := Point(X, Y);
  end;
end;

procedure TForm1.OnControlMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
  if (ssLeft in Shift) and (ClickedControl=Sender) and
    (Sqr(ClickPoint.X-X)+Sqr(ClickPoint.Y-Y)>10) then
    RichViewEdit1.BeginOleDrag;
end;
```

3) Assign events to created controls:

```
  btn := TButton.Create(Self);
  btn.Caption := 'Button';
```

```
  btn.OnMouseDown := OnControlMouseDown;
  btn.OnMouseMove := OnControlMouseMove;
  RichViewEdit1.InsertControl[352]('',btn,rvvaBaseline);
```

4) Events are not saved in RVF, so reassign them on loading. Use OnControlAction[275] event:

```
procedure TForm1.RichViewEdit1ControlAction(Sender: TCustomRichView;
  ControlAction: TRVControlAction; ItemNo: Integer; var ctrl: TControl);
begin
  if ControlAction=rvcaAfterRVFLoad then begin
    if ctrl is TButton then begin
      TButton(ctrl).OnMouseDown := OnControlMouseDown;
      TButton(ctrl).OnMouseMove := OnControlMouseMove;
     end;
  end;
end;
```

The complete example can be downloaded here: ***http://www.trichview.com/forums/viewtopic.php?t=157***

1.5.1.1.3.17 TRichView.Clear

Clears the document.

```
procedure Clear; virtual;
```

This method:

- deletes all items[64];
- turns off displaying soft page breaks[211];
- clears DocProperties[176];
- clears DocObjects[174];
- stops live spelling check[210];
- stops animations[90], if AnimationMode[169] = *rvaniOnFormat*.

This method does not change collections of text, paragraph and list styles. So, if your application adds styles while editing, call DeleteUnusedStyles[216] after Clear.

Method type: generally, it's a viewer-style method, but it is special, because it clears all editing and displaying history. It is a starting method for document generation.

This method is called by the methods: LoadRVF[245], LoadRVFFromStream[245].

See also methods:

- Format[217].

See also method of TRichViewEdit:

- Clear[338].

See also:

- Methods for building documents[77].

1.5.1.1.3.18 TRichView.ClearLiveSpellingResults

Clears all live spelling underlines and stops the live spelling thread.

```
procedure ClearLiveSpellingResults;
```

(introduced in v1.9)

Clear[210] method calls this procedure automatically.

See also: Live spelling check in RichView[101].

1.5.1.1.3.19 TRichView.ClearSoftPageBreaks

Turns off displaying soft page breaks

```
procedure ClearSoftPageBreaks;
```

See also:

- AssignSoftPageBreaks[209].

1.5.1.1.3.20 TRichView.ClientToDocument, DocumentToClient

The methods convert client coordinates to document coordinates and vice versa.

```
function ClientToDocument(const APoint: TPoint): TPoint;
```

(introduced in v1.9)

```
function DocumentToClient(const APoint: TPoint): TPoint;
```

(introduced in v14)

ClientToDocument converts **APoint** from the control's coordinate system to the document coordinate system. DocumentToClient performs an opposite conversion.

In client area coordinates, (0, 0) corresponds to the top left corner of the control's client area. For example, mouse events use this coordinate system.

In document coordinates, (0, 0) corresponds to the top left corner of the scrollable area. For example, GetItemAt[225] uses this coordinate system.

The methods take a cell rotation[611] into account.

Note:

This method must be used instead of adding HScrollPos[545] to X and VScrollPos[547]*VSmallStep[196] to Y. The old method was used in some old example code, it is correct only if the document is top-aligned[195].

See also:

Example in OnRVMouseDown[289].

1.5.1.1.3.21 TRichView.ConvertDocTo...

The methods convert all lengths in the current document to different units of measurement.

```
procedure ConvertDocToPixels;
procedure ConvertDocToTwips;
procedure ConvertDocToDifferentUnits(NewUnits: TRVStyleUnits[691]);
```

(introduced in version 13)

The methods convert all lengths in the current document from Style[193].Units[437] to different units of measurement:

- screen pixels[106],
- twips[106],
- NewUnits (twips or pixels),

respectively.

The methods convert only properties of items[118], they do not convert properties of the Style[193] itself.

To convert the document completely, you need to:

- call one of these methods (for all documents linked to the same TRVStyle[418] components);
- call one of Style[193].ConvertTo*[438] method to convert properties of Style[193] to the same units of measurement.

1.5.1.1.3.22 TRichView .Copy

Copies the selected fragment to the Clipboard in all available formats (as text, RVF, RTF and picture)

```
procedure Copy;
```

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

This method must be called only when the document is formatted.

See also:

- Working with selection 81;
- Working with Clipboard 82.

1.5.1.1.3.23 TRichView .CopyDef

Copies the selected fragment to the Clipboard in several formats according to Options 182.

```
function CopyDef: Boolean;
```

This is the main method for copying to the Clipboard. It is executed automatically when user presses Ctrl+C or Ctrl+Insert.

Calling CopyDef does the same thing as sending WM_COPY message to the control.

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

- if (*rvoAutoCopyRVF* in Options), it copies selection as RVF 93.
- if (*rvoAutoCopyImage* in Options), and there is only one picture selected, it copies selection as an image.
- if (*rvoAutoCopyUnicodeText* in Options), it copies text as Unicode 99 text.
- if (*rvoAutoCopyRTF* in Options), it copies selection as RTF.

This method must be called only when the document is formatted.

Return value:

"Is the selection not empty?" and "Is at least one of *rvoAutoCopy**** options set?"

See also:

- Working with selection 81;
- Working with Clipboard 82.

1.5.1.1.3.24 TRichView .CopyImage

Copies the selected image to the Clipboard.

```
procedure CopyImage;
```

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

If selection contains something besides one image, this method just clears the Clipboard.

This method must be called only when the document is formatted.

See also:

- Working with selection 81;
- Working with Clipboard 82.

1.5.1.1.3.25 TRichView.CopyRTF

Copies the selected part of the document to the Clipboard as RTF (Rich Text Format).

```
procedure CopyRTF;
```

This procedure does nothing if there is nothing selected.

This procedure clears Clipboard before copying.

Settings for RTF saving can be changed in the TRichView component editor[168].

Options for saving are in RTFOptions[185] property.

By default, items tags[68] are saved as hyperlinks targets. You can customize saving using OnWriteHyperlink[302] event.

This method must be called only when the document is formatted.

See also:

- SaveRTF[253], SaveRTFToStream[254] methods;
- RTFOptions[185] property;
- UseStyleTemplates[195] property;
- Working with selection[81];
- Working with Clipboard[82].

1.5.1.1.3.26 TRichView.CopyRVF

Copies the selection to the Clipboard as RVF (RichView Format)

```
procedure CopyRVF;
```

This procedure does nothing if there is nothing selected.

This procedure clears the Clipboard before copying.

Settings for RVF saving can be changed in the TRichView component editor[168].

This method must be called only when the document is formatted.

See also:

- Working with selection[81];
- Working with Clipboard[82];
- RVF[93].

1.5.1.1.3.27 TRichView.CopyText -W

These methods copy the selected fragment to the Clipboard as text.

```
procedure CopyText;
procedure CopyTextA;
procedure CopyTextW;
```

These methods do nothing if there is nothing selected.

These methods clear the Clipboard before copying.

CopyTextA copies text in ANSI encoding. Text of Unicode styles is converted basing on TRVStyle.DefCodePage[421].

CopyTextW and **CopyText** copy text in Unicode encoding.

Since modern versions of Windows converts Unicode and ANSI texts in the Clipboard to each other, using **CopyTextA** does not make sense.

These methods must be called only when the document is formatted.

See also:

- Working with selection 81;
- Working with Clipboard 82;
- Unicode 99.

1.5.1.1.3.28 TRichView .Create

Usual component constructor. Creates a new TRichView instance.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate RichView at runtime. RichViews placed on forms at design time are created automatically. Specify the owner of the new RichView using the **AOwner** parameter.

Important note: initial values of several properties of components created in code may be different from initial values of components placed on form at design time. In the latter case, initial values of these properties can be overridden by the designtime component editor 168 (and overridden by default!)

1.5.1.1.3.29 TRichView .DeleteItems

Removes the specified number of items (**Count**) from the document, starting from the **FirstItemNo**-th item.

```
procedure DeleteItems(FirstItemNo, Count: Integer);
```

Items are indexed from 0 to ItemCount 180-1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.DeleteItems.

Method type: viewer-style.

Be careful: this method can produce invalid document 105.

Example 1: deleting all empty lines

```
procedure DeleteBlankLines(RVData: TCustomRVData 650);
var i,r,c: Integer;
    table: TRVTableItemInfo 565;
begin
  for i := RVData.ItemCount 180-1 downto 0 do
    if RVData.IsParaStart 242(i) and (RVData.GetItemStyle 229(i)>=0) and
      (RVData.GetItemTextR(i)='') and (RVData.ItemCount 180>1) then
      RVData.DeleteItems(i,1)
    else if RVData.GetItem 225(i) is TRVTableItemInfo 565 then
    begin
      table := TRVTableItemInfo 565(RVData.GetItem 225(i));
      for r := 0 to table.RowCount 580-1 do
        for c := 0 to table.ColCount 575-1 do
          if table.Cells 574[r,c]<>nil then
            DeleteBlankLines(table.Cells 574[r,c].GetRVData 613);
    end;
end;
```

DeleteBlankLines cannot be undone/redone. If called for editor, the editor's undo buffer becomes incorrect, so call ClearUndo 339.

Call:

```
DeleteBlankLines(RichViewEdit1.RVData 187);
RichViewEdit1.ClearUndo 339;
RichViewEdit1.Format 217;
```

Example 2: deleting empty lines at the end of document

```
procedure DeleteTrailingBlankLines(RVData: TCustomRVData);
var i: Integer;
begin
  for i := RVData.ItemCount-1 downto 1 do
```

```
    if RVData.IsParaStart(i) and (RVData.GetItemStyle(i)>=0) and
     (RVData.GetItemTextW(i)='') then
      RVData.DeleteItems(i,1)
    else
      break;
end;
```

See also:

- DeleteSection[215];
- DeleteParas[215].

1.5.1.1.3.30 TRichView.DeleteMarkedStyles

Deletes unused text[437], paragraph[431] and list[429] styles.

```
procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData[653]);
```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents[93] (but stored in file or in the Registry). In this case, several documents share the same collections of styles. Used styles are marked with MarkStylesInUse[247] method, see TRVDeleteUnusedStylesData[653] for the examples.

Usually it's much more simple (and recommended) to use one RVStyle per one TRichView and store collections of styles inside RVF documents. In this case, use DeleteUnusedStyles[216] instead.

See also methods:

- MarkStylesInUse[247];
- DeleteUnusedStyles[216].

1.5.1.1.3.31 TRichView.DeleteParas

Deletes paragraph(s) containing items from **FirstItemNo** to **LastItemNo**, updates the document

```
procedure DeleteParas(FirstItemNo, LastItemNo: Integer);
```

(introduced in version 1.7)

This method is designed specially for chat/log windows. Together with FormatTail[218], it allows updating chat windows faster, without complete reformatting.

Items are indexed from 0 to ItemCount[180]-1..

Method type: unique method. Unlike all other methods of TRichView, it quickly reformats TRichView, so you do not need to call Format[217] after it. Like editing-style methods, this method requires document to be formatted before calling, and leaves it formatted after. But it is still a viewer-style method, because it does not update undo buffers, does not check ReadOnly property, etc.

See also:

- DeleteItems[214];
- DeleteSection[215].

1.5.1.1.3.32 TRichView.DeleteSection

Removes the "section" named **CpName** from documents

```
procedure DeleteSection(CpName: String);
```

"Section" is a part of document. The first item of "section" is the item having *checkpoint*[65] with non empty name. This name is also a name of "section". If there is no another *checkpoint* with non empty name below, the

last item in the document is also the last item of "section". Otherwise, item with that "checkpoint" starts a new section, and does not belong to the current section.

Method type: viewer-style.

Be careful: this method can produce invalid document[105].

See also:

- DeleteItems[214];
- DeleteParas[215].

1.5.1.1.3.33 TRichView.DeleteUnusedStyles

Deletes text and/or paragraph and/or list styles which do not exist in document

```
procedure DeleteUnusedStyles(
  TextStyles, ParaStyles, ListStyles: Boolean);
```

(introduced in version 1.5, updated in version 1.6, 1.7)

This procedure deletes unused items from the collections of text (if **TextStyles** parameter is True), paragraph (if **ParaStyles** parameter is True), and list (if **ListStyles** parameter is True) styles (see TextStyles[437], ParaStyles[431], ListStyles[429]) of the linked[193] TRVStyle component).

Then it updates references to these styles in the component and other styles.

Warning: be careful when using this method. Use it only if RVStyle object is used exclusively by this RichView (this method updates references to styles only in one RichView)

This method deletes only non-standard styles (see Standard[464] property of styles)

This method does not require reformatting.

See also methods:

- MarkStylesInUse[247];
- DeleteMarkedStyles[215].

1.5.1.1.3.34 TRichView.Deselect

Removes the document selection (does not change document, just nothing becomes selected for copying to the Clipboard)

```
procedure Deselect;
```

This method does not repaint document (call Invalidate)

This method must be called only when the document is formatted.

The method generates OnSelect[301] event.

See also methods:

- SetSelectionBounds[271];
- SelectAll[257].

See also:

- Selecting in RichView document[81].

1.5.1.1.3.35 TRichView.Destroy

Destroys an instance of TRichView

```
destructor Destroy;override;
```

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not nil, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

1.5.1.1.3.36 TRichView.FindCheckpointByName

Returns the first *checkpoint* having the specified **Name**, or **nil** if there is no such *checkpoint*

```
function FindCheckpointByName(const Name: TRVUnicodeString): TCheckpointData;
```

See also methods:

- FindCheckpointByTag 217;

See also:

- "Checkpoints" 65.

1.5.1.1.3.37 TRichView.FindCheckpointByTag

Returns the first checkpoint having the specified tag, or nil if there is no such checkpoint

```
function FindCheckpointByTag(const Tag: TRVTag): TCheckpointData;
```

Example:

```
var  CheckpointData: TCheckpointData;
...
  CheckpointData := MyRichView.FindCheckpointByTag('Tag_String_Example');
```

See also methods:

- FindCheckpointByName 217.

See also:

- "Tags" 68;
- "Checkpoints" 65.

1.5.1.1.3.38 TRichView.FindControlItemNo

Returns an index of the RichView item which owns the control 126 **actrl**, or -1 if **actrl** was not added in TRichView.

```
function FindControlItemNo(actrl: TControl): Integer;
```

Note: This method contains some calculations inside (iterations through items, from the 0-th to the returned one)

See also:

- RichView item types 64.

1.5.1.1.3.39 TRichView.Format

Formats the document. Only formatted documents can be displayed or edited.

```
procedure Format;
```

If *rvoFormatInvalidate* in Options 182 (default), this method also repaints RichView.

You need to call this method after you modified document (using any of viewer-style methods), its text or paragraph styles.

When resized, TRichView is reformatted automatically.

Most of methods of RichViewEdit[316] automatically quickly reformat the affected part of the document, but you still need this method in RichViewEdit when you use methods inherited from RichView (viewer-style methods).

See also methods:

- FormatTail[218];
- Reformat[248].

See also properties:

- Options[182].

See also:

- Building document[77].

1.5.1.1.3.40 TRichView .FormatTail

Formats a part of document appended after the last call of Format[217] or FormatTail.

```
procedure FormatTail;
```

This method works properly only if new items were added from new line (as new paragraphs) after the last call of Format[217] or FormatTail.

It's much more efficient to reformat only a new part of the document than making a full reformatting.

This method is designed specially for chat/log windows. Together with DeleteParas[215], it allows updating chat windows faster, without complete reformatting.

If *rvoFormatInvalidate* in Options[182], this method also repaints RichView.

See also methods:

- Format[217];
- DeleteParas[215].

See also properties:

- Options[182];
- VScrollVisible[547].

See also:

- Appending items[77].

1.5.1.1.3.41 TRichView .GetBreakInfo

Returns main properties of item of *break*[125] type[125]

```
procedure GetBreakInfo(ItemNo: Integer; var AWidth: TRVStyleLength[690];
  var AStyle: TRVBreakStyle[669]; var AColor: TColor;
  var ATag: TRVTag[692]);
```

Input parameter:

ItemNo – index of the item. The item must be of *break*[125] type (*rvsBreak*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetBreakInfo.

Output parameters:

AWidth – line width (or rectangle height). This value is measured in Style[193].Units[437].

AStyle – visual style of this *break*, see TRVBreakStyle[669] for possible values.

AColor – line color. If it is equal to *clNone*, Style[193].TextStyles[437][0].Color[469] is used.

ATag – *tag* of the item. Use SetItemTag[267] or SetBreakInfo[260] to change *tag* of item.

Instead of this method, you can use GetItemExtraIntPropertyEx[227] and GetItemTag[230] methods.

Additional item properties are returned by the methods GetItemExtraIntProperty[227] and GetItemExtraStrProperty[228].

See also methods:

- SetBreakInfo[260];
- GetItemStyle[229];
- GetItemExtraIntProperty[227];
- GetItemExtraStrProperty[228].

See also properties:

- ItemCount[180].

See also:

- Obtaining RichView items[84];
- Item types[64];
- *Tags*[68].

1.5.1.1.3.42 TRichView .GetBulletInfo

Returns main properties of item of *bullet*[128] type[128]

```
procedure GetBulletInfo(ItemNo: Integer; var AName: TRVUnicodeString[694];
  var AImageIndex: Integer; var AImageList: TCustomImageList;
  var ATag: TRVTag[692]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of *bullet*[128] or *hotspot*[129] type (*rvsBullet*[711] or *rvsHotspot*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetBulletInfo.

Output parameters:

AName – name of *bullet*[128]. It can also be read using GetItemText[230] method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using GetItemExtraIntPropertyEx[227] method.

ATag – *tag* of the item. Use SetItemTag[267] or SetBulletInfo[261] to change *tag* of item. This value can also be read using GetItemTag[230] method.

Additional item properties are returned by the methods GetItemExtraIntProperty[227] and GetItemExtraStrProperty[228].

See also methods:

- SetBulletInfo 261;
- GetItemStyle 229;
- GetItemExtraIntProperty 227;
- GetItemExtraStrProperty 228.

See also properties:

- ItemCount 180.

See also:

- Obtaining RichView items 84;
- Item types 64.
- *Tags* 68.

1.5.1.1.3.43 TRichView .GetCheckpointByNo

Returns the **No**-th *checkpoint*.

```
function GetCheckpointByNo(No: Integer): TCheckpointData667;
```

The first *checkpoint* has index = 0.

Note 1: this function contains some calculations inside (iterations through checkpoints, from the 0-th to the **No**-th *checkpoint*).

Note 2: this method does not work with *checkpoints* in subdocuments (table cells 605)

See also methods:

- GetCheckpointNo 221;
- GetItemCheckpoint 226.

See also:

- "Checkpoints" 65.

1.5.1.1.3.44 TRichView .GetCheckpointInfo

Returns information about the *checkpoint* **CheckpointData**

```
procedure GetCheckpointInfo(CheckpointData: TCheckpointData667;
  var Tag: TRVTag692; var Name: TRVUnicodeString694;
  var RaiseEvent: Boolean);
```

Output parameters

Tag – *tag* of checkpoint;

Name – name of the *checkpoint*, any string;

RaiseEvent – "raise event" flag; if set, RichView can generate OnCheckpointVisible 274 event for this *checkpoint*.

See also properties:

- CPEventKind 173.

See also events:

- OnCheckpointVisible 274.

See also:

- "Checkpoints" 65;
- "Tags" 68.

1.5.1.1.3.45 TRichView.GetCheckpointItemNo

Returns index of the RichView item which owns the checkpoint **CheckpointData**

```
function GetCheckpointItemNo(CheckpointData: TCheckpointData[667]): Integer;
```

Note: this method does not work with checkpoints in subdocuments (table cells[605])

See also:

- Item types[64];
- "Checkpoints"[65].

1.5.1.1.3.46 TRichView.GetCheckpointNo

Returns index of the *checkpoint* **CheckpointData**

```
function GetCheckpointNo(CheckpointData: TCheckpointData[667]): Integer;
```

The first *checkpoint* has index = 0.

Note 1: this function contains some calculations inside (iterations through *checkpoints*, from the 0-th to **CheckpointData**)

Note 2: this method does not work with *checkpoints* in subdocuments (table cells[605])

See also methods:

- GetCheckpointByNo[220];
- GetCheckpointItemNo[221].

See also:

- "Checkpoints"[65].

1.5.1.1.3.47 TRichView.GetCheckpointXY

Returns **X** and **Y** coordinates of the *checkpoint* (relative the top left corner of the document, i.e. top left corner of scrollable area)

```
procedure GetCheckpointXY(CheckpointData: TCheckpointData[667];
  var X,Y: Integer);
```

You can use the returned **Y** to scroll document to this *checkpoint*.

These coordinates are the coordinates of the top left corner of the item which owns this *checkpoint* (for *checkpoint* at the bottom of the document the method returns (X=0, Y=DocumentHeight[177]).

This method must be called only when the document is formatted.

See also methods:

- GetCheckpointY[221];
- GetCheckpointYEx[222];
- ScrollTo[548].

See also:

- "Checkpoints"[65];
- Scrolling[80].

1.5.1.1.3.48 TRichView.GetCheckpointY

Returns vertical coordinate of the **no**-th *checkpoint*

```
function GetCheckpointY(no: Integer): Integer;
```

The first *checkpoint* has index (i.e. **no** parameter) = 0.

The coordinate is relative to the top of document (top of scrollable area)

You can use the returned value to scroll document to this *checkpoint*.

This method must be called only when the document is formatted.

Note: this function contains some calculations inside (iterations through *checkpoints*, from the 0-th to the **no**-th checkpoint).

This function ignores *checkpoints* in table cells.

See also methods:

- GetCheckpointYEx 222;
- GetCheckpointXY 221;
- ScrollTo 548.

See also:

- "Checkpoints" 65;
- Vertical scrolling 80.

1.5.1.1.3.49 TRichView .GetCheckpointYEx

Returns vertical coordinate of the *checkpoint* **CheckpointData**

```
function GetCheckpointYEx(CheckpointData: TCheckpointData667): Integer;
```

Coordinate is relative to the top of the document (top of scrollable area)

You can use the returned value to scroll document to this *checkpoint*.

This method must be called only when the document is formatted.

See also methods:

- GetCheckpointY 221;
- GetCheckpointXY 221;
- ScrollTo 548.

See also:

- "Checkpoints" 65;
- Vertical scrolling 80.

1.5.1.1.3.50 TRichView .GetControlInfo

Returns main properties of the item of "inserted control" type 126

```
procedure GetControlInfo(ItemNo: Integer;
  var AName: TRVUnicodeString694; var Actrl: TControl;
  var AVAlign: TRVVAlign695; var ATag: TRVTag692);
```

(changes in version 18)

Input parameter:

ItemNo – index of the item. The item must be of "inserted control" type 126 (*rvsComponent* 711), otherwise the method raises ERichViewError 649 exception. Items are indexed from 0 to ItemCount 180-1, GetItemStyle 229 returns type of item. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.GetControlInfo.

Output parameters:

AName – name of control item 126. Do not confuse with **Actrl**.Name property! This value can also be read using GetItemText 230 method.

Actr – the control itself. This method returns control owned by RichView, do not not destroy it.

AVAlign – vertical alignment of the control.

ATag – *tag* of the item. Do not confuse with **Actrl**.Tag property!. Use SetItemTag[267] or SetControlInfo[262] to change *tag* of item. This value can also be read using GetItemTag[230] method.

Additional item properties are returned by the methods GetItemExtraIntProperty[227] and GetItemExtraStrProperty[228].

Unicode note: names of non-text items are ANSI strings.

See also methods:

- SetControlInfo[262];
- GetItemStyle[229];
- GetItemExtraIntProperty[227];
- GetItemExtraStrProperty[228].

See also properties:

- ItemCount[180].

See also:

- Obtaining RichView items[84];
- Item types[64];
- "Tags"[68].

1.5.1.1.3.51 TRichView.GetFirstCheckpoint

Returns value identifying the first *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetFirstCheckpoint: TCheckpointData[667];
```

GetFirstCheckpoint + GetNextCheckpoint[234] allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells[605]) are not included in *checkpoints* list of the document.

See also methods:

- GetNextCheckpoint[234], contains example;
- GetLastCheckpoint[232];
- GetPrevCheckpoint[237];

- GetCheckpointInfo[220];
- GetCheckpointXY[221].

See also:

- "Checkpoints"[65].

1.5.1.1.3.52 TRichView.GetFocusedItem

Returns the item having keyboard focus.

```
procedure GetFocusedItem(var ARVData: TCustomRVFormattedData[650];
  var AItemNo: Integer);
```

There are two types of items which can have keyboard focus:

- hypertext[69] (text or pictures)
- inserted controls[126].

If hypertext item has focus, pressing Enter invokes OnJump[282] event for the focused link.

This method returns the focused item (the **AItemNo**-th item of **ARVData** object).

ARVData can be TRichView.RVData[187] or table cell[146].

This method must be called only when the document is formatted.

If **ARVData**=**nil**, there is no focused item.

Focused items are supported only in TRichView, not in TRichViewEdit[316].

See also:

- TabNavigation[194] property.

1.5.1.1.3.53 TRichView.GetHotspotInfo

Returns main properties for item of *hotspot*[129] type[129]

```
procedure GetHotspotInfo(ItemNo: Integer; var AName: TRVUnicodeString[694];
  var AImageIndex, AHotImageIndex: Integer;
  var AImageList: TCustomImageList;
  var ATag: TRVTag[692]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of *hotspot*[129] type (*rvsHotspot*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetHotspotInfo.

Output parameters:

AName – name of *hotspot*[129]. It can also be read using GetItemText[230] method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList** for normal image.It can also be read using GetItemExtraIntPropertyEx[227] method.

AHotImageIndex – index in **AImageList** for "hot" image. This image is displayed under the mouse pointer (in TRichView, or in TRichViewEdit in hypertext mode), or when user moves the caret to this item (in TRichViewEdit). It can also be read using GetItemExtraIntPropertyEx[227] method.

ATag – *tag* of the item. Use SetItemTag[267] or SetHotspotInfo[265] to change *tag* of item. This value can also be read using GetItemTag[230] method.

Additional item properties are returned by the methods GetItemExtraIntProperty[227] and GetItemExtraStrProperty[228].

See also methods:

- SetHotspotInfo[265];
- GetItemStyle[229];
- GetItemExtraIntProperty[227];
- GetItemExtraStrProperty[228].

See also properties:

- ItemCount[180].

See also:

- Obtaining RichView items[84];
- Item types[64];
- *Tags*[68].

1.5.1.1.3.54 TRichView.GetItem

Returns object representing one item in document by its index **ItemNo** (zero-based)

```
function GetItem(ItemNo: Integer): TCustomRVItemInfo;
```

(introduced in version 1.4)

Items are indexed from 0 to ItemCount-1, GetItemStyle returns type of item. Items of subdocuments (table cells) are not included in the items range of the main document; for items in cells, use Cell.GetRVData.GetItem.

Do not free this object yourself.

Usually this method if used to get an item object for tables, labels, sequences, footnotes, endnotes, references to notes.

This method is inverse to GetItemNo.

See also methods:

- AddItem;
- GetItemStyle.

See also methods of TRichViewEdit:

- GetCurrentItem;
- GetCurrentItemEx.

See also properties:

- ItemCount.

See also events:

- OnItemAction.

See also:

- Obtaining RichView items.

1.5.1.1.3.55 TRichView.GetItemAt

Returns item at the position **X**,**Y**.

```
function GetItemAt(X, Y: Integer; var RVData: TCustomRVFormattedData;
  var ItemNo, OffsetInItem: Integer; Strict: Boolean): Boolean;
```

Input parameters:

X,**Y** – coordinates relative to the top level corner of document (scrollable area)

Strict:

- *True* – the method returns an item which is exactly at the specified location;
- *False* – the method returns a position where caret should be moved, if the user would click the mouse at (**X**,**Y**)

Output parameters:

RVData – object containing this item (RichView.RVData or table cell or RVData of cell inplace editor)

ItemNo – index of item (in the items list of RVData), or -1 if no item was found.

OffsetInItem – position of this item:

- for non-text: 0 (closer to the beginning of the item) or 1 (to the end);
- for text: position is before the **OffsetInItem**-th character of text; characters are indexed from 1, the last position is Length(text)+1.

Return value:

"item found?"

This method must be called only when the document is formatted.

See also:

- GetWordAt 239.

See also events:

- OnRVMouseDown 289 (contains example);
- OnRVMouseUp 291.

Demo projects:

- **Demos\Delphi\Assorted\PlanetQuiz**
- **Demos\CBuilder\Assorted\PlanetQuiz**

1.5.1.1.3.56 TRichView .GetItemCheckpoint

Returns *checkpoint* associated with the **ItemNo**-th item, or **nil** if this item does not have associated *checkpoint*.

```
function GetItemCheckpoint(ItemNo: Integer): TCheckpointData 667;
```

Items are indexed from 0 to ItemCount 180-1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.GetItemCheckpoint.

See also methods:

- GetCheckpointInfo 220;
- GetCheckpointXY 221.

See also:

- "Checkpoints" 65.

1.5.1.1.3.57 TRichView .GetItemCoords

The methods return coordinates of the item in RichView document.

```
function GetItemCoords(ItemNo: Integer;
  var Left,Top: Integer): Boolean;
function GetItemClientCoords(ItemNo: Integer;
  var Left,Top: Integer): Boolean;
```

(introduced in version 1.4)

Input parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount 180-1. Items of subdocuments (table cells 605) are not included in the items range of the main document.

Output parameters:

Left,**Top** – coordinates of the top left corner of the item, pixels.

GetItemCoords returns document coordinates, i.e. (0,0) is in the top left corner of the scrollable area.

GetItemClientCoords returns client coordinates, i.e. (0,0) is in the top left corner of the control window.

These methods must be called only when the document is formatted.

Limitations:

- the methods cannot be used to get coordinates of items inside table cells;

- the methods return only the top left coordinates;
- for text items occupying several lines, the methods return the coordinates for the first line;
- for left- and right-aligned objects, the methods return the position of their placeholder in a text, not the position of a floating object.

GetItemCoordsEx[227] does not have these limitations.

Return value:

True if coordinates were successfully retrieved (document must be formatted)

See also:

- ClientToDocument, DocumentToClient[211].

See also methods of TCustomRVFormattedData

- GetOriginEx[653].

1.5.1.1.3.58 TRichView.GetItemCoordsEx

Returns coordinates of the item in RichView document.

```
function GetItemCoordsEx(RVData: TCustomRVFormattedData[650]; ItemNo, Part: Integer;
  AllowFloating: Boolean; var R: TRect): Boolean;
```

(introduced in version 14)

Input parameters:

RVData, ItemNo define the position of the item. **RVData** – an object containing this item (RichView.RVData[187] or table cell[605] or RVData of cell inplace editor). **ItemNo** – an index of the item (in the items list of **RVData**).

Part – the item part. For non-text items and text items occupying a single line, it should be 0. For text items occupying more than one line, **Part** must be 0 for the first line, 1 for the second line and so on (you can call this method until it returns *False* for a non-existing part).

AllowFloating specifies which coordinates should be returned for left- and right-aligned[695] objects. If *True*, coordinates of a floating object are returned. If *False*, coordinates of a placeholder in a text are returned.

Output parameters:

R – coordinates of the item (or item part), in pixels. These are document coordinates, i.e. (0,0) is in the top left corner of the scrollable area. If you need client coordinates, convert **R**.TopLeft and **R**.BottomRight using DocumentToClient[211].

The method must be called only when the document is formatted.

This method takes cell rotation[611] into account.

Return value:

True if coordinates were successfully retrieved (document must be formatted; and **Part** must be valid).

See also:

- GetItemCoords[226].

1.5.1.1.3.59 TRichView.GetItemExtraIntProperty -Ex

The methods return a value of the specified integer property for the **ItemNo**-th item.

```
function GetItemExtraIntProperty(ItemNo: Integer;
  Prop: TRVExtraItemProperty[673]; var Value: Integer): Boolean;
function GetItemExtraIntPropertyEx(ItemNo, Prop: Integer;
```

```
  var Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetItemExtraIntProperty[Ex].

Value receives value of the property identified by **Prop**.

In **GetItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[673]. See information about this type for the list of properties.

In **GetItemExtraIntPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemProperty, **GetItemExtraIntPropertyEx** works like **GetItemExtraIntProperty**. In addition, it supports properties identified by rveipc*** constants[706]

Return value:

True, if this item has this property. *False*, if not.

See also:

- SetItemExtraIntProperty[Ex][266];
- GetItemExtraStrProperty[228].

See also methods of TRichViewEdit:

- GetCurrentItemExtraIntProperty[Ex][346].

1.5.1.1.3.60 TRichView.GetItemExtraStrProperty -Ex

The methods return a value of the specified string property for the **ItemNo**-th item

```
function GetItemExtraStrProperty(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty[676]; var Value: TRVUnicodeString[694]): Boolean;
function GetItemExtraStrPropertyEx(ItemNo, Prop: Integer;
  var Value: TRVUnicodeString[694]): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetItemExtraStrProperty[Ex].

Value receives value of the property identified by **Prop**.

In **GetItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[676]. See information about this type for the list of properties.

In **GetItemExtraStrPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemStrProperty, **GetItemExtraStrPropertyEx** works like **GetItemExtraStrProperty**. In addition, it supports properties identified by rvespc*** constants[709]

Return value:

True, if this item has this property. *False*, if not.

See also:

- SetItemExtraStrProperty[Ex][266];
- GetItemExtra[227]**Int**[227]Property[Ex][227].

See also methods of TRichViewEdit:

- GetCurrentItemExtraStrProperty[Ex][346].

1.5.1.1.3.61 TRichView.GetItemNo

Returns index of item (zero-based).

```
function GetItemNo(Item: TCustomRVItemInfo[563]): Integer;
```

(introduced in version 1.4)

Item is an object representing the item. Usually this method if used to get index if table item[565].

This method is inverse to GetItem[225].

Returned value can be used, for example, in methods:

TRichViewEdit.BeginItemModify[335]/EndItemModify[341]

See also methods:

AddItem[201].

1.5.1.1.3.62 TRichView.GetItemPara

Returns the index of paragraph style for paragraph containing the **ItemNo**-th item.

```
function GetItemPara(ItemNo: Integer): Integer;
```

(Introduced in version 1.3)

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetItemPara.

Return value:

The returned value is an index in the collection of Style[193].ParaStyles[431]. The method never returns -1, even if this item is not the first paragraph item. The method returns the same value for all items in the same paragraph.

See also:

- IsParaStart[242];
- IsFromNewLine[242].

See also:

- Paragraphs[72].

1.5.1.1.3.63 TRichView.GetItemStyle

Returns type (style) of the **ItemNo**-th item.

```
function GetItemStyle(ItemNo: Integer): Integer;
```

Parameters:

ItemNo – index of item (from 0 to ItemCount[180]-1). Items are indexed from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetItemStyle.

Return value:

If the item is a text item[120], this function returns zero or positive value: index of text style of this item in the collection of styles (Style[193].TextStyles[437]).

If the item is not a text item, this function returns type of this item. It is a negative value, see rvs**** constants[711].

See also properties of TRichViewEdit:

- CurItemStyle 323.

See also:

- Item types 64.

1.5.1.1.3.64 TRichView .GetItemTag

Returns *tag* of the **ItemNo**-th item

```
function GetItemTag(ItemNo: Integer): TRVTag 692;
```

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount 180-1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613 .GetItemTag.

Use SetItemTag 267 to change tag of item.

See also methods:

- SetItemTag 267.

See also method of TRichViewEdit:

- GetCurrentTag 349.

See also:

- RichView items' "tags" 68.

1.5.1.1.3.65 TRichView .GetItemText -A -W

Returns a text of text item, or a name of non-text item.

```
function GetItemText(ItemNo: Integer): String;
function GetItemTextA(ItemNo: Integer): TRVAnsiString 668;
function GetItemTextW(ItemNo: Integer): TRVUnicodeString 694;
```

(Introduced in version 1.3, 1.7)

ItemNo – index of the item. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.GetItemText*.

Unicode notes:

Internally, text is stored as Unicode. **GetItemTextA** (and **GetItemText** in non-Unicode versions of Delphi) converts Unicode to ANSI. For a text item, a code page for conversion is calculated basing on the its Charset 468 . If it is equal to DEFAULT_CHARSET, and for non-text items, Style 193.DefCodePage 421 is used.

See also:

- SetItemText -A -W 267.

See also method of TRichViewEdit:

- GetCurrentItemText -A -W 347.

See also:

- Obtaining Items of RichView 84;
- Unicode in RichView 99.

1.5.1.1.3.66 TRichView .GetItemVAlign

Returns vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
function GetItemVAlign(ItemNo: Integer): TRVVAlign 695;
```

(introduced in version 12)

Parameter:

ItemNo – index of the item. Items are indexed from 0 to ItemCount 180 -1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613 .GetItemVAlign.

This method must not be called for text items 120, tables 130, *breaks* 125, tabs 121, list markers 131, footnotes 136 and endnotes 135.

Use SetItemVAlign 268 to change vertical alignment.

See also methods:

- SetItemVAlign 268.

See also method of TRichViewEdit:

- GetCurrentItemVAlign 348

1.5.1.1.3.67 TRichView .GetJumpPointItemNo

Returns index of hypertext item (*hotspot* 129, *hot-picture* 125 or text hypertext) by its hypertext **id**.

```
function GetJumpPointItemNo(id: Integer): Integer;
```

Warning: do not use this function for document containing tables 145. When the link is in table cell, or in cell inplace-editor, it can not be addressed using only an index of item. Use GetJumpPointLocation 231 instead.

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo 179.

Return value

Index of the hypertext item (in range 0..ItemCount 180 -1). If there is no link with such **id**, the method returns -1.

Note: this function contains some calculations inside (iterations through hyperlinks, from the first link to the requested one)

For TCustomRichViewEdit 316, if ReadOnly 328 =False, this function always returns -1 (except for calling it from inside OnWriteHyperlink 302, OnJump 282 or OnRVMouseMove 290).

See also:

- RichView hypertext 69.

1.5.1.1.3.68 TRichView .GetJumpPointLocation

Returns information required to access item by its hypertext **id**.

```
procedure GetJumpPointLocation(id: Integer;
  var RVData: TCustomRVFormattedData 650; var ItemNo: Integer);
```

(introduced in version 1.4)

Input parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo 179.

Output parameters

RVData and **ItemNo** define location of the hypertext item with the given hypertext **id**.

Hypertext item is the **ItemNo**-th item in **RVData** object.

RVData is a RichView document. It can be RichView.RVData 187, table cell 605, or RVData of cell inplace-editor.

If there is no jump with such **id**, the method returns -1 in **ItemNo**.

Note: this function contains some calculations inside (iterations through hyperlinks, from the first link to the requested one)

For TCustomRichViewEdit[316], if ReadOnly[328]=False, this function always returns -1 (except for calling it from inside OnWriteHyperlink[302], OnJump[282] or OnRVMouseMove[290]).

See also:

- RichView hypertext[69].

1.5.1.1.3.69 TRichView.GetJumpPointY

Returns vertical coordinate of the hypertext link (*hotspot*[129], *hot-picture*[125] or text hypertext), identified by **id**

```
function GetJumpPointY(id: Integer): Integer;
```

Parameter

id – identifier of the hypertext link; links are numbered sequentially starting from FirstJumpNo[179].

Return value

You can use the returned value to scroll document to the position of this link (useful when implementing "Back" button functionality)

If there is no link with such **id**, the method returns 0 (top of the document)

This method must be called only when the document is formatted.

Note: this function contains some calculations inside (iterations through jumps, from the first jump to requested one)

For TCustomRichViewEdit[316], if ReadOnly[328]=False, this function always returns 0 (except for calling it from inside OnJump[282] or OnRVMouseMove[290]).

See also methods:

- ScrollTo[548]

See also:

- RichView Hypertext[69]

1.5.1.1.3.70 TRichView.GetLastCheckpoint

Returns value identifying the last *checkpoint* in the document, or **nil** if the document has no *checkpoints*.

```
function GetLastCheckpoint: TCheckpointData[667];
```

(introduced in version 1.5)

GetLastCheckpoint + GetPrevCheckpoint[237] allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells) are not included in the *checkpoints* list of document.

See also methods:

- GetNextCheckpoint[234], contains example;
- GetFirstCheckpoint[223];
- GetPrevCheckpoint[237];
- GetCheckpointInfo[220];

- GetCheckpointXY 221.

See also:

- "Checkpoints" 65.

1.5.1.1.3.71 TRichView .GetLineNo

Returns line of the position specified by **ItemNo** and **ItemOffs**.

```
function GetLineNo(ItemNo, ItemOffs: Integer): Integer;
```

(introduced in version 1.7)

Parameters

ItemNo – index of the item, in range from 0 to ItemCount 180-1.

ItemOffs – offset in item.

For non-text items, **ItemOffs**=0 specifies position before the item, **ItemOffs**=1 specifies position after the item (this is not important for this function, because non-text item cannot be placed on several lines).

For text items, ItemOffs specifies position before the **ItemOffs**-th character:

- **ItemOffs**=1 – position before the text item (before the first character);
- **ItemOffs**=2 – position before the second character;
- ....
- **ItemOffs**=GetOffsAfterItem 235(ItemNo) – position after the last character of the text item.

Return value

Line index. The first line has index 1. Number of lines depends on word-wrapping.

Example:

```
with MyRichView do
  r := GetLineNo(ItemCount 180-1, GetOffsAfterItem 235(ItemCount 180-1))
```

r receives number of lines in MyRichView.

Warning: this function contains calculations inside, and it may take a while to execute it. (There is an idea how to make it lightning-fast in future, but it will be implemented later).

This method must be called only when the document is formatted.

See also:

- GetItemStyle 229.

See also methods of TCustomRichViewEdit:

- GetCurrentLineCol 348.

1.5.1.1.3.72 TRichView .GetListMarkerInfo

Returns main properties of list marker 131 (paragraph bullet or numbering)

```
function GetListMarkerInfo(AItemNo: Integer;
  var AListNo, AListLevel, AStartFrom: Integer;
  var AUseStartFrom: Boolean): Integer;
```

(introduced in version 1.7)

Input parameter

AItemNo – index of the item, in range from 0 to ItemCount 180-1. This is not necessary the index of list marker, it can be index of any item in the given paragraph. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.GetListMarkerInfo.

Output parameters (valid only if the function returned non-negative value)

AListNo – style of this list maker. This is an index in the collection of list styles (Style[193].ListStyles[429]).

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[AListNo].Levels[493]).

AStartFrom: if **AUseStartFrom**=*True*, and this is a numbered list, this parameter is a value of list counter for this marker. If **AUseStartFrom**=*False*, numbering is continued.

Return value:

If this paragraph does not have a marker, this function returns -1; otherwise it returns item index of the marker.

See also:

- SetListMarkerInfo[269].

1.5.1.1.3.73 TRichView .GetNextCheckpoint

Returns value identifying the next *checkpoint* after the given **CheckpointData**, or **nil** if **CheckpointData** is the last *checkpoint* in the document.

```
function GetNextCheckpoint(CheckpointData: TCheckpointData[667]): TCheckpointData[667];
```

GetFirstCheckpoint[223] + **GetNextCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells[605]) are not included in *checkpoints* list of the document.

Example:

```
var Tag: TRVTag[692];
    Name: String;
    CheckpointData: TCheckpointData;
    RaiseEvent: Boolean;
    s: TRVUnicodeString[694];
begin
  MyListBox.Items.Clear;
  CheckpointData := MyRichView.GetFirstCheckpoint[223];
  while CheckpointData<>nil do
  begin
    MyRichView.GetCheckpointInfo[220](CheckpointData, Tag, Name, RaiseEvent);
    s := Format('Name:"%s" Tag:"%s"', [Name, Tag]);
    MyListBox.Items.Add(s);
    CheckpointData := MyRichView.GetNextCheckpoint(CheckpointData);
  end;
end;
```

This code fills listbox MyListBox with information about *checkpoints* of MyRichView.

See also methods:

- GetFirstCheckpoint[223];
- GetLastCheckpoint[232];
- GetPrevCheckpoint[237];
- GetCheckpointInfo[220];
- GetCheckpointXY[221].

See also:

- "Checkpoints"[65].

1.5.1.1.3.74 TRichView.GetOffsAfterItem

Returns offset after the **ItemNo**-th item in the document

```
function GetOffsAfterItem(ItemNo: Integer): Integer;
```

This method is useful for selecting part of document for copying in the Clipboard, or for moving caret to the specified position in editor.

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetOffsAfterItem.

For text items, the method returns length of text + 1. The exception is empty text items formatted with style having EmptyWidth[469]>0; for them, the position after the item is 2.

For non-text items, it returns 1.

Example for TRichViewEdit[316]: moving caret to the end of document

```
var ItemNo, Offs: Integer;

  ItemNo := MyRichViewEdit.ItemCount-1;
  Offs := MyRichViewEdit.GetOffsAfterItem(ItemNo);
  MyRichViewEdit.SetSelectionBounds[271](ItemNo, Offs, ItemNo, Offs);
  MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds[271];
- GetOffsBeforeItem[235].

See also properties of TRichView:

- ItemCount[180].

See also properties of TRichViewEdit:

- CurItemNo[322];
- OffsetInCurItem[328].

See also:

- Working with selection[81];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[717].

1.5.1.1.3.75 TRichView.GetOffsBeforeItem

Returns offset before the **ItemNo**-th item in the document

```
function GetOffsBeforeItem(ItemNo: Integer): Integer;
```

This method is useful for selecting a part of document for copying in the Clipboard, or for moving caret to the specified position in editor.

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetOffsBeforeItem.

For text items, the method returns 1 For non-text items, it returns 0.

Example for TRichViewEdit[316]: moving caret to the beginning of document

```
var ItemNo, Offs: Integer;

  ItemNo := 0;
  Offs := MyRichViewEdit.GetOffsBeforeItem(ItemNo);
  MyRichViewEdit.SetSelectionBounds[271](ItemNo, Offs, ItemNo, Offs);
```

```
MyRichViewEdit.Invalidate;
```

See also methods of TRichView:

- SetSelectionBounds[271];
- GetOffsAfterItem[235].

See also properties of TRichView:

- ItemCount[180].

See also properties of TRichViewEdit:

- CurItemNo[322];
- OffsetInCurItem[328].

See also:

- Working with selection[81];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[717];
- Example how to move caret to the beginning of paragraph[405].

1.5.1.1.3.76 TRichView .GetPictureInfo

Returns main properties for item of picture[122] or *hot-picture*[125] type

```
procedure GetPictureInfo(ItemNo: Integer; var AName: TRVUnicodeString[694];
  var Agr: TGraphic; var AVAlign: TRVVAlign[695];
  var ATag: TRVTag[692]);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of picture[122] or *hot-picture*[125] type (*rvsPicture*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].GetPictureInfo.

Output parameters:

AName – name of the item. This value can also be read using GetItemText[230] method.

Agr – graphic object. This method returns object owned by RichView, do not not destroy it.

AVAlign – vertical alignment of the picture.

ATag – *tag* of the item. Use SetItemTag[267] or SetPictureInfo[269] to change *tag* of item. This value can also be read using GetItemTag[230] method.

Additional item properties (including stretching, transparency for bitmaps, link to file name, alternative text for saving to HTML) are returned by the methods GetItemExtraIntProperty[227] and GetItemExtraStrProperty[228].

See also methods:

- SetPictureInfo[269];
- GetItemStyle[229];
- GetItemExtraIntProperty[227];
- GetItemExtraStrProperty[228].

See also properties:

- ItemCount[180].

See also:

- Obtaining RichView items[84];
- Item types[64];

- "Tags" 68.

1.5.1.1.3.77 TRichView.GetPrevCheckpoint

Returns value identifying the previous *checkpoint* before the given **CheckpointData**, or **nil** if **CheckpointData** is the first *checkpoint* in the document.

```
function GetPrevCheckpoint(CheckpointData: TCheckpointData 667): TCheckpointData 667;
```

(introduced in version 1.5)

GetLastCheckpoint 232 + **GetPrevCheckpoint** allow to iterate through all *checkpoints* in the document.

Checkpoints from subdocuments (table cells 605) are not included in *checkpoints* list of document.

See also methods:

- GetFirstCheckpoint 223;
- GetLastCheckpoint 232;
- GetNextCheckpoint 234;
- GetCheckpointInfo 220;
- GetCheckpointXY 221.

See also:

- "Checkpoints" 65.

1.5.1.1.3.78 TRichView.GetSelectedImage

Returns the selected image

```
function GetSelectedImage: TGraphic;
```

This method returns image, if the selection consist only of one image. If there is nothing selected, or not only image is selected, this method returns **nil**.

The method returns image owned by RichView, not a copy of it. So do not destroy this image.

This method must be called only when the document is formatted.

See also:

- Working with selection 81.

1.5.1.1.3.79 TRichView.GetSelectionBounds

Returns bounds of selected part of document.

```
procedure GetSelectionBounds(var StartItemNo, StartItemOffs,
  EndItemNo, EndItemOffs: Integer; Normalize: Boolean);
```

Parameters

StartItemNo – index of the first selected item.

StartItemOffs:

- if the first item is a text item 120, then the selection start is before the **StartItemOffs**-th character of string (characters in strings are counted from 1, the last position (after the text item) is text length+1). The exception is empty text items formatted with style having EmptyWidth 469>0; for them, the position after the item is 2.
- if the first item is not a text, then
  - if **StartItemOffs**=0, then the selection start is before the first item;
  - if **StartItemOffs**=1, then the selection start is after the first item.

EndItemNo – index of the last selected item.

EndItemOffs:

- if the last item is a text item, then the selection end is before the **EndItemOffs**-th character of string (characters in strings are counted from 1, the last position (after the text item) is text length+1). The exception is empty text items formatted with style having EmptyWidth[469]>0; for them, the position after the item is 2.
- if the last item is not a text, then
  - if **EndItemOffs**=0, then the selection end is before the last item
  - if **EndItemOffs**=1, then the selection end is after the last item.

Normalize:

- if *True*, the upper bound of the selection is returned in **StartItemNo**:**StartItemOffs**, and the lower bound is in **EndItemNo**:**EndItemOffs**;
- if *False*, bounds are returned as they were selected by user.

The selection is empty if

- **StartItemNo**=-1 or
- **StartItemNo**=**EndItemNo** and **StartItemOffs**=**EndItemOffs**.

This method must be called only when the document is formatted.

If you want to get selection bounds using richedit-like parameters (SelStart and SelLength), use RVGetSelection from RVLinear[666] unit instead.

See also methods:

- SetSelectionBounds[271].

See also:

- Working with selection.[81]

1.5.1.1.3.80 TRichView .GetSelText -A -W

These methods return the selected part of the document as a text.

```
function GetSelText: String;
function GetSelTextA: TRVAnsiString[668];
function GetSelTextW: TRVUnicodeString[694];
```

If there is nothing selected, these methods return '' (empty string)

Unicode notes:

Internally, text is stored as Unicode. **GetSelTextA** (and **GetSelText** in non-Unicode versions of Delphi) converts Unicode to ANSI using Style[193].DefCodePage[421] code page.

These methods must be called only when the document is formatted.

See also:

- Working with selection[81].

1.5.1.1.3.81 TRichView.GetTextInfo

Returns text and *tag* for the **ItemNo**-th item

```
procedure GetTextInfo(ItemNo: Integer; var AText: TRVUnicodeString;
  var ATag: TRVTag);
```

(changed in version 18)

Input parameter:

ItemNo – index of the item. The item must be of text type (GetItemStyle 229 must return zero or positive value for this item), otherwise the method raises ERichViewError 649 exception. Items are indexed from 0 to ItemCount 180 -1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.GetTextInfo.

Output parameters:

AText – item text (visible text). This is the same value as returned by GetItemText 230.

ATag – *tag* of the item. Use SetItemTag 267 to change *tag* of item. The *tag* can also be returned by GetItemTag 230.

See also methods:

- GetItemStyle 229;
- GetItemText 230, SetItemText 267;
- GetItemTag 230, SetItemTag 267.

See also properties:

- ItemCount 180.

See also:

- Obtaining items 84;
- Item types 64;
- "Tags" 68.

1.5.1.1.3.82 TRichView.GetWordAt -A -W

Returns word at the specified coordinates

```
procedure GetWordAt(X,Y: Integer;
  var RVData: TCustomRVFormattedData; var ItemNo: Integer;
  var Word: TRVUnicodeString); overload;
function GetWordAtA(X,Y: Integer): TRVAnsiString;
function GetWordAtW(X,Y: Integer): TRVUnicodeString
function GetWordAt(X,Y: Integer): String;
```

(changed in version 18)

Input parameters

(**X**,**Y**) – client coordinates (coordinates relative to the top left corner of RichView window).

Output parameters

Word (or returned value for the simplified versions) receives word (if it is a text item 120), or name of non-text item. See the Unicode note below. "Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters 174 property.

ItemNo receives index of item at (**X**,**Y**), or -1 if there is no item at this point. This is an index of item in **RVData** object (it may be the main document, cell, or cell inplace editor).

Unicode notes:

Internally, all text is stored as Unicode. **GetWordAtA** (and **GetWordAt** in non-Unicode versions of Delphi) converts it to ANSI using Style[193].DefCodePage[421].

For example, you can use this method inside OnRVMouseDown[289] and OnRVMouseUp[291].

This method must be called only when the document is formatted.

Example: displaying the clicked words (only text items!) in Label1.

We cannot use simplified version of **GetWordAt** because we cannot know if the returned text is for text item.

```
uses CRVFData;
...
procedure TForm1.RichView1RVMouseUp(Sender: TCustomRichView;
  Button: TMouseButton; Shift: TShiftState; ItemNo, X, Y: Integer);
var
  LItemNo: Integer;
  LRVData: TCustomRVFormattedData;
  LWord: TRVUnicodeString[694];
begin
  ItemNo<0 then
    exit; // we already know that the mouse is not above item
  Sender.GetWordAt(X,Y, LRVData, LItemNo, LWord);
  if LItemNo<0 then
    exit;
  if LRVData.GetItemStyle[229](LItemNo)>=0 then // text item?
    Label1.Caption := LWord;
end;
```

See also events:

- OnRVMouseDown[289];
- OnRVMouseUp[291];
- OnRVRightClick[291];
- OnRVDblClick[287].

See also properties:

- Delimiters[174].

See also methods:

- SelectWordAt[258]

1.5.1.1.3.83 TRichView.InsertRVFFromStream

Inserts RVF content from **Stream** in the position of the document specified as **Index** (the first inserted item will have this index)

```
function InsertRVFFromStream(Stream: TStream; Index: Integer): Boolean;
```

This is the only method of RichView (not RichViewEdit) inserting items in any position.

Index – index of the item, must be in range from 0 to ItemCount[180]. Items of subdocuments (table cells[605]) are not included in the items range of the main document.

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor[168].

If style templates are used[195], and RVFTextStylesReadMode[191]=RVFParaStylesReadMode[190] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style[193].StyleTemplates[435], and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

Return value: "Was reading successful?"

Example

This example shows how to combine several RVF documents in one.

```
procedure AddDoc(const FileName: String; rv: TCustomRichView);
var Stream: TFileStream;
begin
  Stream := TFileStream.Create(FileName, fmOpenRead);
  rv.InsertRVFFromStream(Stream, rv.ItemCount);
  Stream.Free;
end;
  ...
  RichView1.Clear;
  RichView1.DeleteUnusedStyles(True, True, True);
  AddDoc('c:\file1.rvf', RichView1);
  AddDoc('c:\file2.rvf', RichView1);
  AddDoc('c:\file3.rvf', RichView1);
  RichView1.Format;
```

If you want to add page breaks between documents, use the following code

```
procedure AddDoc(const FileName: String; rv: TCustomRichView);
var Stream: TFileStream;
  ItemCount: Integer;
begin
  ItemCount := rv.ItemCount;
  Stream := TFileStream.Create(FileName, fmOpenRead);
  rv.InsertRVFFromStream(Stream, rv.ItemCount);
  Stream.Free;
  if (ItemCount>0) and (ItemCount<rv.ItemCount) then
    rv.PageBreaksBeforeItems[ItemCount] := True;
end;
```

See also properties:

- RVFOptions;
- RVFWarnings;
- RVFTextStylesReadMode (this method type: RVF insertion);
- RVFParaStylesReadMode (this method type: RVF insertion).

See also events:

- OnRVFImageListNeeded;
- OnRVFControlNeeded;
- OnRVFPictureNeeded;
- OnControlAction (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange.

See also methods:

- AppendRVFFromStream;
- LoadRVFFromStream;
- LoadRVF;
- Format;
- FormatTail.

See also methods of TRichViewEdit:

- InsertRVFFromStreamEd.

See also:

- TRichView component editor;
- Saving and loading;
- RVF overview.

1.5.1.1.3.84 TRichView .IsFromNew Line

Returns "does the **ItemNo**-th item start a new line"?

```
function IsFromNewLine(ItemNo: Integer): Boolean;
```

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].IsFromNewLine.

This method returns *True* both in cases when this item starts a new paragraph (see IsParaStart[242]) or starts a new line inside the paragraph (it was moved to the new line in editor using Shift+Enter, or added in the mode SetAddParagraphMode[259](*False*)).

See also methods:

- IsParaStart[242];
- GetItemPara[229].

See also:

- Paragraphs[72].

1.5.1.1.3.85 TRichView .IsParaStart

Returns "is the **ItemNo**-th item a first paragraph item"?

```
function IsParaStart(ItemNo: Integer): Boolean;
```

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].IsParaStart.

This method returns *False* if this item starts a new line inside the paragraph (i.e. it was moved to the new line in editor using Shift+Enter, or added in the mode SetAddParagraphMode[259](*False*)).

See also methods:

- IsFromNewLine[242];
- GetItemPara[229].

See also:

- Paragraphs[72];
- Example how to move caret to the beginning of paragraph[405].

1.5.1.1.3.86 TRichView .LiveSpellingValidateWord

Removes live spelling underlines from all occurrences of **AWord**.

```
procedure LiveSpellingValidateWord(const AWord: TRVUnicodeString[694]);
```

(introduced in version 1.9; changed in version 18)

Call this method if you added this word in a dictionary or ignore list.

This procedure does not affect further processing of this word. User dictionaries and ignore list must be maintained by spelling checkers.

See also:

- Live spelling check in RichView[101].

1.5.1.1.3.87 TRichView.LoadFromStream

Loads document from **Stream**, autodetecting its format (RVF[93]/RTF/text)

```
type
  TRVYesNoAuto = ( rvynaNo, rvynaYes, rvynaAuto );
function LoadFromStream(Stream: TStream; IsTextUnicode: TRVYesNoAuto): Boolean;
```

(introduced in version 10)

This method can detect both normal RTF, and RTF converted to Unicode string.

The method does not load text if it contains characters having codes less than ord(' ').

IsTextUnicode specifies whether the text is Unicode[99]:

- *rvynaNo* – reading text as ANSI;
- *rvynaYes* – reading text as Unicode;
- *rvynaAuto* – autodetecting ANSI/Unicode (wrong detection is very possible).

Method type: viewer-style.

Return value:

"Was reading successful?"

See also methods of TRichView:

- LoadRVFFromStream[245];
- LoadRTFFromStream[244];
- LoadRTFFromStream, LoadRTFFromStreamW[247].

1.5.1.1.3.88 TRichView.LoadRTF

Appends the content of RTF (Rich Text Format) file **FileName** to the document.

```
function LoadRTF(const FileName: TRVUnicodeString[694]):Boolean;
```

(Introduced in version 1.5; changed in version 18)

Despite its name, this method does not clear the existing document before loading. To replace the current document with the content of this file, call Clear[210] method before loading (and may be DeleteUnusedStyles[216] after Clear)

Method type: viewer-style.

Parameters for loading are in RTFReadProperties[187]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[309] and ParaStyleMode[308]. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor[168].

See important note on loading external images from RTF[721].

By default, hyperlink targets are loaded in items tags[68]. You can customize loading using OnReadHyperlink[284] event.

If style templates are used[195], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method loads an RTF style sheet into Style[193].StyleTemplates[435]. If TRichView was completely empty before loading, LoadRTF replaces the existing style templates. Otherwise, it merges the RTF style sheet into the style templates, and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

Return value: "Was reading successful?" (extended information is in RTFReadProperties.ErrorCode[306])

See also methods of TRichView:

- LoadRTFFromStream 244;
- SaveRTF 253.

See also events of TRichView:

- OnImportPicture 279;
- OnReadHyperlink 284;
- OnStyleTemplatesChange 302;
- OnProgress 283;
- OnAssignImageFileName 274.

See also methods of TRichViewEdit:

- InsertRTFFromFileEd 356.

See also:

- Saving and loading in RichView 92.

1.5.1.1.3.89 TRichView .LoadRTFFromStream

Appends the content of RTF (Rich Text Format) stream **Stream** to the document.

```
function LoadRTFFromStream(Stream: TStream):Boolean;
```

(Introduced in version 1.5)

Despite its name, this method does not clear the existing document before loading. To replace the current document with the content of this file, call Clear 210 method before loading (and may be DeleteUnusedStyles 216 after Clear)

Method type: viewer-style.

Parameters for loading are in TRichView.RTFReadProperties 187. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode 309 and ParaStyleMode 308. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor 168.

See important note on loading external images from RTF 721.

By default, hyperlink targets are loaded in items tags 68. You can customize loading using OnReadHyperlink 284 event.

If style templates are used 195, and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode 193<>*rvstimIgnoreSourceStyleTemplates*, the method loads an RTF style sheet into Style 193.StyleTemplates 435. If TRichView was completely empty before loading, LoadRTFFromStream replaces the existing style templates. Otherwise, it merges the RTF style sheet into the style templates, and reads text and paragraph styles according to StyleTemplateInsertMode 193. The method calls OnStyleTemplatesChange 302 event.

Return value: "Was reading successful?" (extended information is in TRichView.RTFReadProperties.ErrorCode 306)

See also methods of TRichView:

- LoadRTF 243;
- SaveRTFToStream 254.

See also events of TRichView:

- OnImportPicture 279;
- OnReadHyperlink 284;
- OnStyleTemplatesChange 302;
- OnProgress 283.

See also methods of TRichViewEdit:

- InsertRTFFromStreamEd 357.

See also:

- Saving and loading in RichView 92.

1.5.1.1.3.90 TRichView.LoadRVF

Clears the document and loads a new document from the file **FileName** as RVF (RichView Format 93).

```
function LoadRVF(const FileName: TRVUnicodeString 694): Boolean;
```

(changed in version 18)

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor 168.

Return value: "Was reading successful?"

See also properties:

- RVFOptions 188;
- RVFWarnings 191;
- RVFTextStylesReadMode 191 (this method type: RVF loading);
- RVFParaStylesReadMode 190 (this method type: RVF loading);
- UseStyleTemplates 195.

See also events:

- OnRVFImageListNeeded 288;
- OnRVFControlNeeded 288;
- OnRVFPictureNeeded 289;
- OnControlAction 275 (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange 302;
- OnProgress 283.

See also methods:

- LoadRVFFromStream 245;
- AppendRVFFromStream 208;
- InsertRVFFromStream 240;
- Format 217;
- FormatTail 218.

See also methods of TRichViewEdit:

- InsertRVFFromFileEd 357.

- **See also:**

- TRichView component editor 168;
- Saving and loading 92;
- RVF overview 93.

1.5.1.1.3.91 TRichView.LoadRVFFromStream

Clears the document and loads a new document from the file stream **Stream** as RVF (RichView Format 93).

```
function LoadRVFFromStream(Stream: TStream): Boolean;
```

Method type: viewer-style.

Setting for RVF loading can be changed in the TRichView component editor 168.

Return value: "Was reading successful?"

See also properties:

- RVFOptions 188;
- RVFWarnings 191;

- RVFTextStylesReadMode 191 (this method type: RVF loading);
- RVFParaStylesReadMode 190 (this method type: RVF loading);
- UseStyleTemplates 195.

See also events:

- OnRVFImageListNeeded 288;
- OnRVFControlNeeded 288;
- OnRVFPictureNeeded 289;
- OnControlAction 275 (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange 302;
- OnProgress 283.

See also methods:

- LoadRVF 245;
- AppendRVFFromStream 208;
- InsertRVFFromStream 240;
- Format 217;
- FormatTail 218.

See also methods of TRichViewEdit:

- InsertRVFFromFileEd 357.

- **See also:**

- TRichView component editor 168;
- Saving and loading 92;
- RVF overview 93.

1.5.1.1.3.92 TRichView .LoadText -W

Append the content of the text file **FileName** to the document. **LoadText** loads text from ANSI text files, **LoadTextW** loads text from Unicode (UTF-16) text files.

```
function LoadText(const FileName: TRVUnicodeString 694; StyleNo, ParaNo: Integer;
  AsSingleParagraph: Boolean; CodePage: Cardinal=CP_ACP):Boolean;

function LoadTextW(const FileName: TRVUnicodeString 694; StyleNo, ParaNo: Integer;
  DefAsSingleParagraph: Boolean):Boolean;
```

(changed in version 18)

Despite their names, these methods do not clear the existing document before loading. To replace the current document with the content of this file, call Clear 210 method before loading (and may be DeleteUnusedStyles 216 after Clear).

Parameters

StyleNo – an index of text style 437 which will be used for new text. It defines font attributes for the added text.

ParaNo – an index of paragraph style 431 which will be used for new text. It defines paragraph attributes for the added text.

AsSingleParagraph (and **DefAsSingleParagraph**):

- *False* – the methods use the current add-paragraph-mode (see SetAddParagraphMode 259);
- *True* – the methods add all text as a single paragraph (add the first new item in the current add-paragraph-mode, set add-paragraph-mode to *False*, add the rest of text, restore the add-paragraph-mode). **DefAsSingleParagraph** only affects processing CR and LF characters. WideChar($2029) ("paragraph separator") always initiates a new paragraph. WideChar($2028) ("line separator") always initiates a new line.

CodePage – a code page for converting from ANSI to Unicode.

LoadTextW supports the Unicode byte order marks characters (if they are present, they must be the first characters in the file).

Methods type: viewer-style.

If style templates [195] are used, the component may automatically change text style indexes of text items, to provide a consistency of text and paragraph style templates.

Return value:"Was loading successful?"

See also methods:

- LoadTextFromStream, LoadTextFromStreamW [247];
- AddTextNLA, AddTextNLW [206];
- Format [217];
- FormatTail [218];
- SetAddParagraphMode [259];
- SaveText, SaveTextW [255].

See also methods of TRichViewEdit:

- InsertTextFromFile, InsertTextFromFileW [361].

See also properties of TRVStyle:

- TextStyles [437];
- SpacesInTab [435].

See also:

- Other methods for appending items [77];
- Saving and loading [92];
- Unicode in RichView [99];
- Example how to load UTF-8 files [316].

1.5.1.1.3.93 TRichView.LoadTextFromStream-W

Append the content of the stream **Stream** to the document. **LoadTextFromStream** reads ANSI text, **LoadTextFromStreamW** reads Unicode (UTF-16) text.

```
function LoadTextFromStream(Stream: TStream; StyleNo, ParaNo: Integer;
  AsSingleParagraph: Boolean; CodePage: Cardinal=CP_ACP):Boolean;

function LoadTextFromStreamW(Stream: TStream; StyleNo, ParaNo: Integer;
  DefAsSingleParagraph: Boolean):Boolean;
```

The same as LoadText and LoadTextW [246], but read text from streams instead of files.

1.5.1.1.3.94 TRichView.MarkStylesInUse

Stores lists of used text [437], paragraph [431] and list [429] styles.

```
procedure MarkStylesInUse(Data: TRVDeleteUnusedStylesData [653]);
```

(introduced in version 10)

This method is useful if collections of text, paragraph and list styles are not saved in RVF documents [93] (but stored in file or in the Registry). In this case, several documents share the same collections of styles. See TRVDeleteUnusedStylesData [653] for the examples.

Usually it's much more simple (and recommended) to use one RVStyle per one RichView and store collections of styles inside RVF documents. In this case, you can use DeleteUnusedStyles [216] instead.

See also methods:

- DeleteMarkedStyles [215];
- DeleteUnusedStyles [216].

1.5.1.1.3.95 TRichView.Reformat

Reformats the document.

Only formatted documents can be displayed or edited.

```
procedure Reformat;
```

(introduced in v1.9)

This method works like Format[217], but keeps the selection (and caret position in TRichViewEdit[316]).

Document must be formatted before calling this method.

This method is useful, for example, if you want to reformat document after changing margins.

1.5.1.1.3.96 TRichView.RefreshListMarkers

Recalculates list markers[131]

```
procedure RefreshListMarkers;
```

(introduced in v1.8)

Call it (before calling Format[217]) if you changed ListType[510] or FormatString[508] of list style level of marker that is already inserted in TRichView. It's usually not necessary to use this method.

1.5.1.1.3.97 TRichView.RefreshListSequences

Recalculates numbered sequences[134], footnotes[136], endnotes[135], and sidenotes[137].

```
procedure RefreshSequences;
```

(introduced in v13)

Call it (before calling Format[217]) if you changed FootnoteNumbering[422], EndnoteNumbering[422] or SidenoteNumbering[422] of TRVStyle[418] component linked[193] to this TRichView control, or a property of a numbered sequence item that was already inserted, to update existing document containing footnotes or endnotes.

1.5.1.1.3.98 TRichView.RemoveCheckpoint

Removes the *checkpoint* associated with the **ItemNo**-th item.

```
function RemoveCheckpoint(ItemNo: Integer): Boolean;
```

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].RemoveCheckpoint.

If the **ItemNo**-th item does not have a *checkpoint*, this method does nothing.

Methods type: viewer-style.

Repainting may be needed if *rvoShowCheckpoints* in Options[182].

See also methods of TRichView:

- GetItemCheckpoint[226];
- SetCheckpointInfo[261].

See also properties:

- ItemCount[180].

See also methods of RichViewEdit:

- RemoveCheckpointEd[366];
- RemoveCurrentCheckpoint[367].

See also:
- "Checkpoints 65";
- Modifying RichView items 85.

1.5.1.1.3.99 TRichView.ResetAnimation

Rewinds all inserted animated pictures to the first frame.

```
procedure ResetAnimation;
```

(introduced in version 10)

See also methods:
- StartAnimation 272;
- StopAnimation 273.

See also properties:
- AnimationMode 169.

See also:
- Animation in TRichView 90;
- Pictures 122;
- *Hot-Pictures* 125.

1.5.1.1.3.100 TRichView.SaveDocX

Exports TRichView document (or the selected part, if **SelectionOnly**=*True*) to the file **FileName** as DocX (Microsoft Word format).

```
function SaveDocX(const FileName: TRVUnicodeString 694; SelectionOnly: Boolean):Boolean;
```

(introduced in version 15; changed in version 18)

This method is available for Delphi 4+, C++Builder 2006+

The method uses *rvrtfSaveDocParameters* and *rvrtfSaveHeaderFooter* options from RTFOptions 185 property.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile 292 event.

By default, items tags 68 are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink 302 event.

You can:
- completely change DocX code for certain items using OnSaveItemToFile 297 event;
- insert OOXML codes 472 in text;
- insert additional code in DocX in OnSaveDocXExtra 293 event.

Return value: "successful saving?"

See also methods:
- SaveDocXToStream 250;

See also properties:
- RTFOptions 185;
- UseStyleTemplates 195.

See also events:
- OnWriteHyperlink 302;
- OnSaveComponentToFile 292;
- OnSaveDocXExtra 293;
- OnSaveItemToFile 297;

- OnProgress 283.

See also global variables and constants:

- RichViewSaveHyperlinksInDocXAsFields 704

See also:

- Saving and loading in RichView 92.

1.5.1.1.3.101 TRichView.SaveDocXToStream

Exports TRichView document (or the selected part, if **SelectionOnly**=*True*) to the stream **Stream** as DocX (Microsoft Word format).

```
function SaveDocXToStream(Stream: TStream; SelectionOnly: Boolean):Boolean;
```

(introduced in version 15)

This method is available for Delphi 4+, C++Builder 2006+

The same as SaveDocX 249, but saves to a stream instead of a file.

1.5.1.1.3.102 TRichView.SaveHTML

Exports TRichView document to HTML or XHTML file, using HTML tags like <font>, <b>, , etc.

```
function SaveHTML(const FileName, Title, ImagesPrefix: TRVUnicodeString 694;
  Options: TRVSaveOptions 686): Boolean;
```

(changed in version 18)

This method saves HTML file and a set of images (in separate files).

Files saved by this method can be rendered by the most of HTML applications. But many advanced formatting options are lost. Files saved by SaveHTMLEx 251 look better in modern browsers.

Parameters:

FileName – the name of the output HTML file.

Title – the title of the output HTML file.

ImagesPrefix – the first part of names of images that will be saved with HTML document.

Options – options for saving, see TRVSaveOptions 686 for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile 292 event.

By default, items tags 68 are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink 302 event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile 297 event;
- insert additional code in HTML in OnSaveHTMLExtra 295 and OnSaveParaToHTML 298 events;
- change how images are saved using OnHTMLSaveImage 277 or OnSaveImage2 296 event,
- insert HTML codes 472 in text.

Return value: "successful saving?"

See also methods:

- SaveHTMLEx 251;
- SaveHTMLToStream 252.

See also events:

- OnWriteHyperlink 302;

- OnSaveComponentToFile 292;
- OnSaveHTMLExtra 295;
- OnHTMLSaveImage 277;
- OnSaveImage2 296;
- OnSaveItemToFile 297;
- OnSaveParaToHTML 298;
- OnProgress 283.

See also:

- Export to HTML 96;
- RichViewSavePInHTML 698 typed constant;
- TFontInfo.Options 478 (HTML codes);
- Saving and loading 92.

1.5.1.1.3.103 TRichView.SaveHTMLEx

Exports TRichView document to HTML or XHTML file, using CSS (Cascading Style Sheets)

```
function SaveHTMLEx(const FileName, Title, ImagesPrefix, ExtraStyles,
  ExternalCSS, CPPrefix: TRVUnicodeString;
  Options: TRVSaveOptions): Boolean;
```

(changed in version 18)

This method saves HTML file and a set of images (in separate files).

Parameters:

FileName –- the name of the output HTML file.

Title –- the title of the output HTML file.

ImagesPrefix –- the first part of names of images that will be saved with HTML document;

ExtraStyles –- strings that can contain additional entries of CSS (usually you need not to use it, set to '').

ExternalCSS –- if this string is not empty, this method uses external CSS (saved with RichView.Style.SaveCSS 441) instead of saving CSS into HTML file).

CPPrefix –- first part of *checkpoints* 65 names when saving to HTML; set to '' for using default names ('RichViewCheckPoint'). This parameter is not used, if *rvsoUseCheckpointsNames* is in Options.

Options – options for saving, see TRVSaveOptions 686 for possible values.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile 292 event.

By default, items tags 68 are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink 302 event.

You can:

- completely change HTML code for certain items using OnSaveItemToFile 297 event;
- insert additional code in HTML in OnSaveHTMLExtra 295 and OnSaveParaToHTML 298 events;
- change how images are saved using OnHTMLSaveImage 277 or OnSaveImage2 296 event,
- insert HTML codes 472 in text.

Since v1.9, the resulting HTML conforms "HTML 4.01 Transitional" standard, see http://validator.w3.org/.

Two issues may cause the validator's warnings:

- HTML encoding is written in HTML only if Charset 468 of the 0-th text style <> DEFAULT_CHARSET;
- non-standard <TABLE bordercolor> attribute is saved for MS Internet Explorer.

Return value: "successful saving?"

See also methods:

- SaveHTML 250;
- SaveHTMLToStreamEx 252.

See also events:

- OnWriteHyperlink 302;
- OnSaveComponentToFile 292;
- OnSaveHTMLExtra 295;
- OnHTMLSaveImage 277;
- OnSaveImage2 296;
- OnSaveItemToFile 297;
- OnSaveParaToHTML 298;
- OnProgress 283.

See also:

- Export to HTML 96;
- TFontInfo.Options 478 (HTML codes);
- RichViewSaveDivInHTMLEx 698 typed constant;
- Saving and loading 92.

1.5.1.1.3.104 TRichView.SaveHTMLToStream

Exports TRichView document as HTML or XHTML to the stream **Stream**, using HTML tags like <font>, <b>, , etc.

```
function SaveHTMLToStream(Stream: TStream; const Path, Title,ImagesPrefix: TRVUnicodeString
  Options: TRVSaveOptions 686): Boolean;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTML 250, but saves to a stream instead of a file.

Parameters:

Stream –- stream where to export HTML.

Path –- path where to save images (must have '\' as the last character).

For the descriptions of other parameters, see SaveHTML 250.

1.5.1.1.3.105 TRichView.SaveHTMLToStreamEx

Exports contents of RichView to stream as HTML or XHTML, using CSS (Cascading Style Sheets)

```
function SaveHTMLToStreamEx(Stream: TStream;
  const Path, Title, ImagesPrefix, ExtraStyles,
  ExternalCSS, CPPrefix: TRVUnicodeString 694;
  Options: TRVSaveOptions 686): Boolean;
```

(changed in version 18)

This method saves HTML to **Stream** and creates files for images.

The same as SaveHTMLEx 251, but saves to a stream instead of a file.

Parameters:

Stream –- stream where to export HTML.

Path –- path where to save images (must have '\' as the last character).

For the descriptions of other parameters, see SaveHTMLEx 251.

1.5.1.1.3.106 TRichView.SavePicture

Saves picture. This method is used when saving HTML files.

```
function SavePicture(DocumentSaveFormat: TRVSaveFormat[685];
  const Path: TRVUnicodeString[694]; gr: TGraphic): TRVUnicodeString[694]; virtual;
```

(changed in version 18)

Parameters:

gr – image to save.

DocumentSaveFormat – output document format, see TRVSaveFormat[685] for possible values.

Path – path where to save the image (the last character must be '\').

If **DocumentSaveFormat**=*rvsfHTML* then the image is saved as Jpeg image (or as bitmap, if the application was compiled with RVDONOTUSEJPEGIMAGE conditional define). You can define image formats that will not be converted to Jpeg[721].

If **DocumentSaveFormat**<>*rvsfHTML*, the image is saved as bitmap.

This method must be used only inside the following events:

- OnHTMLSaveImage[277],
- OnSaveImage2[296],
- OnSaveComponentToFile[292],
- OnSaveItemToFile[297] (SaveFormat=rvsfHTML).

The image file name will consist of the image prefix (see the **ImagesPrefix** parameter of SaveHTML[250] and SaveHTMLEx[251]) and number. The file name may be unique or not (depending on *rvsoOverrideImages* in the **Options** parameter of SaveHTML[250] and SaveHTMLEx[251]).

See also:

- Export to HTML[96].

1.5.1.1.3.107 TRichView.SaveRTF

Exports TRichView document (or the selected part, if **SelectionOnly**=*True*) to the file **FileName** as RTF (Rich Text Format).

```
function SaveRTF(const FileName: TRVUnicodeString[694]; SelectionOnly: Boolean):Boolean;
```

(introduced in version 1.3; changed in version 18)

Options for saving are in RTFOptions[185] property.

By default, inserted controls are not saved. You can save them yourself using OnSaveComponentToFile[292] event.

By default, items tags[68] are saved as targets of hypertext links. You can customize saving using OnWriteHyperlink[302] event.

You can:

- completely change RTF code for certain items using OnSaveItemToFile[297] event;
- insert additional code in RTF in OnSaveRTFExtra[299] event;
- insert RTF codes[472] in text.

Return value: "successful saving?"

See also methods:

- SaveRTFToStream 254;
- CopyRTF 213;
- LoadRTF 243.

See also properties:

- RTFOptions 185;
- UseStyleTemplates 195.

See also events:

- OnWriteHyperlink 302;
- OnSaveComponentToFile 292;
- OnSaveRTFExtra 299;
- OnSaveItemToFile 297;
- OnProgress 283.

See also:

- Saving and loading in RichView 92.

1.5.1.1.3.108 TRichView.SaveRTFToStream

Exports TRichView document (or the selected part, if **SelectionOnly**=*True*) to the **Stream** as RTF (Rich Text Format)

```
function SaveRTFToStream(Stream: TStream; SelectionOnly: Boolean):Boolean;
```

The same as SaveRTF 253, but saves to a stream instead of a file.

1.5.1.1.3.109 TRichView.SaveRVF

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the file **FileName** as RVF (RichView Format 93)

```
function SaveRVF(const FileName: TRVUnicodeString 694; SelectionOnly: Boolean): Boolean;
```

(changed in version 18)

Setting for RVF saving can be changed in the TRichView component editor 168.

Return value: "successful saving?"

See also properties:

- RVFOptions 188.

See also methods:

- SaveRVFToStream 254;
- LoadRVF 245.

See also:

- Saving and loading RichView document 92;
- RVF 93.

1.5.1.1.3.110 TRichView.SaveRVFToStream

Saves the document (or the selected part, if **SelectionOnly**=*True*) to the **Stream** as RVF (RichView Format 93).

```
function SaveRVFToStream(Stream: TStream; SelectionOnly: Boolean): Boolean;
```

Setting for RVF saving can be changed in the TRichView component editor 168.

Return value: "successful saving?"

See also properties:

- RVFOptions 188.

See also methods:

- SaveRVF 254;
- LoadRVFFromStream 245;
- InsertRVFFromStream 240;
- AppendRVFFromStream 208.

See also:

- Saving and loading RichView document 92;
- RVF 93.

1.5.1.1.3.111 TRichView.SaveText -W

The methods export document to ANSI (**SaveText**) or Unicode (**SaveTextW**) text file.

```
function SaveText(const FileName: TRVUnicodeString 694; LineWidth: Integer;
  CodePage: Cardinal=CP_ACP): Boolean;
function SaveTextW(const FileName: TRVUnicodeString 694; LineWidth: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName – output file name.

LineWidth is used for saving *breaks* 125 (they are saved as **LineWidth** '-' characters)

Unicode notes:.

Internally, text is stored as Unicode. **SaveText** converts Unicode to ANSI using **CodePage** parameter. If **CodePage**=CP_ACP, Style 193.DefCodePage 421 is used instead.

Return value: "successful saving?"

See also methods:

- SaveTextToStream, SaveTextToStreamW 255;
- LoadText, LoadTextW 246.

See also events:

- OnSaveImage2 296;
- OnSaveComponentToFile 292;
- OnSaveItemToFile 297.

See also:

- Saving and loading RichView documents 92.

1.5.1.1.3.112 TRichView.SaveTextToStream -W

The methods export document or selection as ANSI (**SaveTextToStream**) or Unicode (**SaveTextToStreamW**) text.

```
function SaveTextToStream(const Path: TRVUnicodeString 694; Stream: TStream;
  LineWidth: Integer; SelectionOnly, TextOnly: Boolean;
  CodePage: Cardinal=CP_ACP):Boolean;
function SaveTextToStreamW(const Path: TRVUnicodeString 694; Stream: TStream;
  LineWidth: Integer; SelectionOnly, TextOnly: Boolean):Boolean;
```

(introduced in version 1.3; changed in version 18)

Parameters

LineWidth is used for saving *breaks* [125] (they are saved as **LineWidth** '-' characters)

Path – path for saving images and other non-text items (they are not saved by default, but this parameter is used in OnSaveImage2 [296], OnSaveComponentToFile [292], and OnSaveItemToFile [297] events);

SelectionOnly – if *True*, only selected part of the document is saved.

TextOnly – if *True*, non-text items are ignored when saving. If *False*, text representation of items is saved.

Unicode notes:.

Internally, text is stored as Unicode. **SaveTextToStream** converts Unicode to ANSI using **CodePage** parameter. If **CodePage**=CP_ACP, Style [193].DefCodePage [421] is used instead.

Return value: "successful saving?"

See also methods:

- SaveText, SaveTextW [255];
- LoadTextFromStream, LoadTextFromStreamW [247].

See also events:

- OnSaveImage2 [296];
- OnSaveComponentToFile [292];
- OnSaveItemToFile [297].

See also:

- Saving and loading RichView documents [92].

1.5.1.1.3.113 TRichView .SearchText -A -W

The methods search for the substring **s** in the document.

```
function SearchText(s: String; SrchOptions: TRVSearchOptions [689]): Boolean;
function SearchTextA(s: TRVAnsiString [668]; SrchOptions: TRVSearchOptions [689]): Boolean;
function SearchTextW(s: TRVUnicodeString [694]; SrchOptions: TRVSearchOptions [689]): Boolean;
```

When found, these methods select substring and scroll to it.

if *rvsroDown* is not included in **SrchOptions**, the methods search upwards. if *rvsroDown* is included, the methods search downwards.

If *rvsroFromStart* is included in **SrchOptions**, the search is started from the beginning/end of the document. Otherwise, the methods start searching from the current selection, if it exists. If nothing is selected, the methods start from the first visible item (when searching down) or the last visible item (when searching up).

rvsroSmartStart is used only if *rvsroFromStart* is not included. If *rvsroSmartStart* is included, and some document fragment is selected, then:

- when searching down, the search starts from the second selected character;
- when searching up, the search starts from the last but one selected character;

(selected characters are counted from the top of document, the selection direction is ignored). This option allows to avoid selecting the same fragment again when changing a search direction.

If *rvsroMultiItem* is included, the search can match substrings of several text items [120]. If not included, the text is searched in each text item separately. For example, if the document contains the text **He*llo***, the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.

Unicode note:

Internally, text is stored as Unicode. **SearchTextA** (and **SearchText** in non-Unicode versions of Delphi) converts **s** to Unicode using Style[193].DefCodePage[421]. and calls **SearchTextW**.

Note: There is incompatibility with programs written with the older version of TRichView (1.0 – 1.1.4). Now if rvsroDown is not included in SrchOptions, the methods search upwards. In the older versions there was no rvsroDown option, and SearchText searched downwards only.

These methods must be called only when the document is formatted.

Return value: Is substring found?

See also methods of TRichViewEdit:

- SearchText, -A, -W[369]

See also global functions:

- GetRVSearchOptions[661].

See also:

- Searching in RichView and RichViewEdit[83].

1.5.1.1.3.114 TRichView.SelectAll

Selects the whole document for copying to the Clipboard

```
procedure SelectAll;
```

This method does not repaint document, call Invalidate.

In RichViewEdit[316] this method moves the caret to the end of document. The method generates OnSelect[301] event.

This method must be called only when the document is formatted.

See also methods:

- SetSelectionBounds[271];
- Deselect[216].

See also:

- Selecting in RichView document[81].

See also:

- Edit[651] method of table cell.

1.5.1.1.3.115 TRichView.SelectControl

Selects the given visual control in the document.

```
function SelectControl(actrl: TControl): Boolean;
```

(introduced in version 1.8)

The method selects the item[126] containing the control **actrl**.

This method repaints document.

The method generates OnSelect[301] event.

In TRichViewEdit[316], if this control's item is resizable (see *rvepResizable*[673] extra-property[673]) and resizing with mouse is not forbidden (see *rvoNoImageResize* in EditorOptions[325]), resizing handles appear. A good idea to call this method in OnClick event of resizable controls.

This method must be called only when the document is formatted.

Return value: *True* if this control was found and selected.

See also methods:

- SetSelectionBounds[271];
- Deselect[216].

See also:

Selecting in RichView document[81].

1.5.1.1.3.116 TRichView.SelectionExists

"Is some part of document selected for copying to the Clipboard?"

```
function SelectionExists: Boolean;
```

Note: In some cases, this function can generate OnSelect[301] event, but it's safe to use it inside OnSelect event handler (infinite recursion will not occur)

This method must be called only when the document is formatted.

See also:

- Selecting in RichView document[81].

1.5.1.1.3.117 TRichView.SelectWordAt

Selects word at the specified point for copying to clipboard. If there is non text item at this point (such as picture), this item becomes selected.

```
procedure SelectWordAt(X,Y: Integer);
```

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[174] property.

This method repaints document.

In RichViewEdit[316] this method moves the caret to the end of the selected word.

The method generates OnSelect[301] event.

Parameters:

(**X**,**Y**) – client coordinates (coordinates relative to the top left corner of RichView window).

This method must be called only when the document is formatted.

Note: RichView can automatically select double-clicked word, if *rvoDblClickSelectsWord* is in Options[182].

See also events:

- OnRVRightClick[291];
- OnRVDblClick[287].

See also properties:

- Delimiters[174].

See also methods:

- SelectAll 257;
- GetWordAt 239;
- SetSelectionBounds 271;
- GetSelText 238.

See also methods of TRichViewEdit:

- SelectCurrentWord 370.

See also:

- Selecting in RichView document 81.

1.5.1.1.3.118 TRichView.SetAddParagraphMode

Sets mode for adding items from new line for all Add*** methods 77.

```
procedure SetAddParagraphMode(AllowNewPara: Boolean);
```

(Introduced in version 1.2)

By default (if **SetAddParagraphMode** was not called), all items that will be added from new line will start new paragraphs.

You can change this mode calling **SetAddParagraphMode**(False), and then return this mode again calling **SetAddParagraphMode**(True).

After **SetAddParagraphMode**(False) items that will be added from new line will not start new paragraphs but will be displayed from new line inside the previous paragraph.

Warning: be careful using **SetAddParagraphMode**(False):

- do not add the first item (or the item after *breaks* 125 or tables 130) in this mode;
- do not switch paragraph style in this mode (all items in the same paragraph must have the same paragraph style).

See: valid documents 105

Example 1:

```
with MyRichView do begin
  Clear;
  SetAddParagraphMode(True); // default mode
  AddNL('First',  0, 0); // ok, starting 1st paragraph with style 0
  AddNL('Second', 0, 0); // ok, starting 2nd paragraph with style 0
  AddNL('Third',  0, 1); // ok, starting 3rd paragraph with style 1
end;
```

Example 2:

```
with MyRichView do begin
  Clear;
  SetAddParagraphMode(False);
  AddNL('First', 0, 0); // error, do not add the first item in this mode!
end;
```

Example 3:

```
with MyRichView do begin
  Clear;
  SetAddParagraphMode(True); // default mode
  AddNL('First',  0, 0); // ok, starting 1st paragraph with style 0
  SetAddParagraphMode(False);
  AddNL('Second', 0, 0); // ok, adding this item from new line inside the 1st paragraph
  AddNL('Third',  0, 1); // error, do not add items with another paragraph style in this
```

```
end;
```

See also

- Paragraphs 72;
- Building a document 77.

1.5.1.1.3.119 TRichView.SetBreakInfo

Changes main properties of the item of *break* 125 (horizontal line) type.

```
procedure SetBreakInfo(ItemNo: Integer; AWidth: TRVStyleLength690;
  AStyle: TRVBreakStyle669; AColor: TColor; const ATag: TRVTag692);
```

Parameters:

ItemNo – index of the item. The item must be of *break* 125 type (*rvsBreak* 711), otherwise the method raises ERichViewError 649 exception. Items are indexed from 0 to ItemCount 180-1, GetItemStyle 229 returns type of item. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.SetBreakInfo.

AWidth – line width (or rectangle height). This value is measured in Style 193.Units 437.

AStyle – visual style of this *break*, see TRVBreakStyle 669 for possible values.

AColor – line color. If it is equal to *clNone*, Style 193.TextStyles 437[0].Color 469 is used.

ATag – *tag* of the item.You can use value returned by GetBreakInfo 218 or GetItemTag 230 for this item.

Instead of this method, you can use SetItemExtraIntPropertyEx 266 and SetItemTag 267 methods.

Method type: viewer-style. It's not necessary to reformat document after it, repainting is enough.

Additional item properties are assigned by the methods SetItemExtraIntProperty 266 and SetItemExtraStrProperty 266.

Example

```
RichView2.SetBreakInfo(ItemNo, 1, rvbsLine, clNone, 'Break Tag');
```

See also methods:

- GetBreakInfo 218;
- GetItemStyle 229;
- SetItemExtraIntProperty 266;
- SetItemExtraStrProperty 266.

See also properties:

- ItemCount 180.

See also methods of RichViewEdit:

- SetBreakInfoEd 371;
- SetCurrentBreakInfo 375.

See also:

- Modifying RichView items 85;
- Item types 64;
- *Tags* 68.

1.5.1.1.3.120 TRichView.SetBulletInfo

Changes main properties of the item of *bullet*[128] (image from ImageList) type.

```
procedure SetBulletInfo(ItemNo: Integer; const AName: TRVUnicodeString[694];
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag[692]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *bullet*[128] or *hotspot*[129] type (*rvsBullet or rvsHotspot*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetBulletInfo.

AName – name of *bullet*[128], any string without line break (CR, LF) characters. It can also be set using SetItemText[267] method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyEx[266] method.

ATag – *tag* of the item. You can use value returned by GetBulletInfo[219] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTag[267] method.

Method type: viewer-style. It's not necessary to reformat document after it, repainting is enough.

Additional item properties are assigned by the methods SetItemExtraIntProperty[266] and SetItemExtraStrProperty[266].

Example

```
RichView1.SetBulletInfo(ItemNo, 'my image', 0, nil, 'bullet tag');
```

See also methods:

- GetBulletInfo[219];
- GetItemStyle[229];
- SetItemExtraIntProperty[266];
- SetItemExtraStrProperty[266].

See also properties:

- ItemCount[180].

See also methods of RichViewEdit:

- SetBulletInfoEd[372];
- SetCurrentBulletInfo[375].

See also:

- Modifying RichView items[85];
- Item types[64];
- *Tags*[68].

1.5.1.1.3.121 TRichView.SetCheckpointInfo

Creates a new *checkpoint* or modifies the existing *checkpoint* associated with the **ItemNo**-th item

```
procedure SetCheckpointInfo(ItemNo: Integer; const ATag: TRVTag[692];
  const AName: TRVUnicodeString[694]; ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetCheckpointInfo. If this item already has a *checkpoint*, its properties are modified; otherwise, a new *checkpoint* is created.

ATag – *tag* of *checkpoint*;

AName – name of the *checkpoint*, any string without line break (CR, LF) characters.

ARaiseEvent – "raise event" flag; if set, RichView can generate OnCheckpointVisible[274] event for this *checkpoint*.

Method type: viewer-style. It's not necessary to reformat document after it, repainting is enough (repainting is needed if *rvoShowCheckpoints* in Options[182]).

See also methods:

- GetItemCheckpoint[226];
- RemoveCheckpoint[248].

See also properties:

- ItemCount[180].

See also methods of RichViewEdit:

- SetCheckpointInfoEd[373];
- SetCurrentCheckpointInfo[376].

See also:

- Modifying RichView items[85];
- *Checkpoints*[65];
- *Tags*[68].

1.5.1.1.3.122 TRichView.SetControlInfo

Changes main properties of the item of control[126] type.

```
function SetControlInfo(ItemNo: Integer; const AName: TRVUnicodeString[694];
  AVAlign: TRVVAlign[695]; const ATag: TRVTag[692]): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control[126] type (*rvsComponent*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetControlInfo.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemText[267] method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign[695] for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetControlInfo[222] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTag[267] method.

Return value: "is reformatting needed? (i.e., is the new vertical align not equal to the old one?)"

Method type: viewer-style.

Additional item properties are assigned by the methods SetItemExtraIntProperty[266] and SetItemExtraStrProperty[266].

Note: As you can see, current version of TRichView does not allow to change the control itself.

Example

```
RichView1.SetControlInfo(ItemNo, 'my control', rvvaBaseline, 'control tag');
```

See also methods:

- GetControlInfo[222];
- GetItemStyle[229];
- Format[217];
- SetItemExtraIntProperty[266];
- SetItemExtraStrProperty[266].

See also properties:

- ItemCount[180].

See also methods of RichViewEdit:

- SetControlInfoEd[374];
- SetCurrentControlInfo[377].

See also:

- Modifying RichView items[85];
- Item types[64];
- *Tags*[68].

1.5.1.1.3.123 TRichView .SetFooter

Assigns the specified document for using as a footer.

```
procedure SetFooter(Footer: TCustomRVData[650];
  HeaderType: TRVHFType[680] = rvhftNormal);
```

(introduced in version 17)

Parameters:

Footer – document which will be used as a footer (of the type specified in **HFType**);

HFType – footer type (normal/for the first page/for even pages).

This footer will be used when saving and loading files and streams. It will not be used for printing, use TRVPrint.SetFooter[528] for assign footers for printing.

Saving:

- **Footer** will be saved together with the main document in RVF;
- **Footer** will be saved together with the main document in RTF and DocX, if *rvrtfSaveHeaderFooter* is included in RTFOptions[185];
- **Footer** will be saved after the main document in HTML, if *rvsoHeaderFooter* is included in the Options parameter of SaveHTML[250]/SaveHTMLEx[251].

Loading:

- when loading RVF or RTF containing a footer of this type, it will be read in **Footer**;

Each document assigned as a header/footer must be linked[193] to each own TRVStyle[418] component. Values of UseStyleTemplates[195] must be the same for the main document and all headers and footers. UseStyleTemplates[195] = *True*, MainRVStyle[430] of RVStyles of headers and footers must be equal to RVStyle of the main document (to allow them using the same collections of StyleTemplates[435]).

Example:

```
RichView1.Clear[210];
RichView2.Clear[210];
```

```
RichView1.RTFReadProperties.ReadDocParameters := True;
RichView1.SetFooter(RichView2.RVData);
RichView1.LoadRTF('test.rtf');
RichView1.Format; // contains document
RichView2.Format; // contains header
RVPrint1.AssignSource(RichView1);
if RichView2.ItemCount>0 then
  RVPrint1.SetFooter(RichView2.RVData)
else
  RVPrint1.SetFooter(nil);
RVPrint1.AssignDocParameters(RichView1.DocParameters);
RVPrint1.FormatPages(rvdoAll);
RVPrint1.Print('Doc with footer', 1, False);
```

See also:

- SetHeader.

See also properties of TRVStyle:

- MainRVStyle.

1.5.1.1.3.124 TRichView .SetHeader

Assigns the specified document for using as a header.

```
procedure SetHeader(Header: TCustomRVData;
  HeaderType: TRVHFType = rvhftNormal);
```

(introduced in version 17)

Parameters:

Header – document which will be used as a header (of the type specified in **HFType**);

HFType – header type (normal/for the first page/for even pages).

This header will be used when saving and loading files and streams. It will not be used for printing, use TRVPrint.SetHeader for assign headers for printing.

Saving:

- **Header** will be saved together with the main document in RVF;
- **Header** will be saved together with the main document in RTF and DocX, if *rvrtfSaveHeaderFooter* is included in RTFOptions;
- **Header** will be saved before the main document in HTML, if *rvsoHeaderFooter* is included in the Options parameter of SaveHTML/SaveHTMLEx.

Loading:

- when loading RVF or RTF containing a header of this type, it will be read in **Header**;

Each document assigned as a header/footer must be linked to each own TRVStyle component. Values of UseStyleTemplates must be the same for the main document and all headers and footers. UseStyleTemplates = *True*, MainRVStyle of RVStyles of headers and footers must be equal to RVStyle of the main document (to allow them using the same collections of StyleTemplates).

Example:

```
RichView1.Clear;
RichView2.Clear;
RichView1.RTFReadProperties.ReadDocParameters := True;
RichView1.SetHeader(RichView2.RVData);
RichView1.LoadRTF('test.rtf');
RichView1.Format; // contains document
```

```
RichView2.Format[217]; // contains header
RVPrint1.AssignSource[522](RichView1);
if RichView2.ItemCount[180]>0 then
  RVPrint1.SetHeader[529](RichView2.RVData)
else
  RVPrint1.SetHeader[529](nil);
RVPrint1.AssignDocParameters[522](RichView1.DocParameters[176]);
RVPrint1.FormatPages[525](rvdoAll);
RVPrint1.Print[526]('Doc with header', 1, False);
```

See also:

- SetFooter[264].

See also properties of TRVStyle[418]:

- MainRVStyle[430].

1.5.1.1.3.125 TRichView.SetHotspotInfo

Changes main properties of the item of *hotspot*[129] (image from ImageList, hyperlink) type.

```
procedure SetHotspotInfo(ItemNo: Integer; const AName: TRVUnicodeString[694];
  AImageIndex, AHotImageIndex: Integer;
  AImageList: TCustomImageList;
  const ATag: TRVTag[692]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *hotspot*[129] type (*rvsHotspot*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetHotspotInfo.

AName – name of *hotspot*[129], any string without line break (CR, LF) characters. It can also be set using SetItemText[267] method.

AImageList – not used, reserved, set it to *nil*..

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyEx[266] method.

AImageIndex – index of "hot" image in image list. This image is displayed under the mouse pointer (in TRichView, or in TRichViewEdit in hypertext mode), or when user moves the caret to this item (in TRichViewEdit). It can also be set using SetItemExtraIntPropertyEx[266] method.

ATag – *tag* of the item. You can use value returned by GetHotspotInfo[224] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTag[267] method.

Method type: viewer-style. It's not necessary to reformat document after it, repainting is enough.

Additional item properties are assigned by the methods SetItemExtraIntProperty[266] and SetItemExtraStrProperty[266].

Example

```
RichView1.SetHotspotInfo(ItemNo, 'my image', 0, 0, nil, 'hotspot tag');
```

See also methods:

- GetHotspotInfo[224];
- GetItemStyle[229];
- SetItemExtraIntProperty[266];
- SetItemExtraStrProperty[266].

See also properties:

- ItemCount 180.

See also methods of RichViewEdit:

- SetHotspotInfoEd 382;
- SetCurrentHotspotInfo 377.

See also:

- Modifying RichView items 85;
- Item types 64;
- *Tags* 68.

1.5.1.1.3.126 TRichView.SetItemExtraIntProperty -Ex

The methods change a value of the specified integer property of the **ItemNo**-th item.

```
function SetItemExtraIntProperty(ItemNo: Integer;
  Prop: TRVExtraItemProperty673; Value: Integer): Boolean;
function SetItemExtraIntProperty(ItemNo, Prop,
  Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

These methods set a new **Value** of the item property identified by **Prop**.

ItemNo – index of the item, from 0 to ItemCount 180-1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613.SetItemExtraIntProperty[Ex]

In **SetItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty 673. See information about this type for the list of properties.

In **SetItemExtraIntPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemProperty, **SetItemExtraIntPropertyEx** works like **SetItemExtraIntProperty**. In addition, it supports properties identified by rveipc*** constants 706

Methods type: viewer-style. Reformatting or repainting may be needed, depending on the changed property.

Return value: *True*, if this item has this property. *False*, if not.

See also

- GetItemExtraIntProperty[Ex] 227;
- SetItemExtra 266**Str** 266Property 266;
- Format 217.

See also methods of TRichViewEdit

- SetItemExtraIntProperty[Ex]Ed 383;
- SetCurrentItemExtraIntProperty[Ex] 378.

1.5.1.1.3.127 TRichView.SetItemExtraStrProperty -Ex

The methods change a value of the specified additional string property of the **ItemNo**-th item.

```
function SetItemExtraStrProperty(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty676; const Value: TRVUnicodeString694): Boolean;
function SetItemExtraStrProperty(ItemNo, Prop: Integer;
  const Value: TRVUnicodeString694): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

This methods set a new **Value** of the item property identified by **Prop**.

In **SetItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[676]. See information about this type for the list of properties.

In **SetItemExtraStrPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemStrProperty, **SetItemExtraStrPropertyEx** works like **SetItemExtraStrProperty**. In addition, it supports properties identified by rvespc*** constants[709]

ItemNo – index of the item, from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetItemExtraStrProperty[Ex].

Method type: viewer-style. Reformatting or repainting may be needed, depending on the changed property.

Return value: *True*, if this item has this property. *False*, if not.

See also:

- GetItemExtraStrProperty[Ex][228];
- SetItemExtra[266]**Int**[266]Property[Ex][266].

See also methods of TRichViewEdit:

- SetItemExtraStrProperty[Ex]Ed[384];
- SetCurrentItemExtraStrProperty[Ex][379].

1.5.1.1.3.128 TRichView.SetItemTag

Changing *tag* of the **ItemNo**-th item

```
procedure SetItemTag(ItemNo: Integer; const ATag: TRVTag[692]);
```

Items are indexed from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetItemTag.

Method type: viewer-style. Reformatting or repainting is not necessary.

Example

```
RichView1.SetItemTag(ItemNo, 'tag #1');
```

See also methods:

- GetItemTag[230].

See also properties:

- ItemCount[180].

See also methods of RichViewEdit:

- SetItemTagEd[384];
- SetCurrentTag[382].

See also:

- Modifying RichView items[85];
- *Tags*[68].

1.5.1.1.3.129 TRichView.SetItemText -A -W

Assign text of text item or name of non-text item.

```
procedure SetItemText(ItemNo: Integer; const s: String);
procedure SetItemTextA(ItemNo: Integer; const s: TRVAnsiString[668]);
procedure SetItemTextW(ItemNo: Integer; const s: TRVUnicodeString[694]);
```

(Introduced in version 1.3, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount 180 -1, GetItemStyle 229 returns type of item. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613 .SetItemText*.

s – text assigned to the item. For text items, this is a visible text. For non-text item, this is a string value associated with the item (not displayed). **S** must not contain line break (CR and LF) characters. For text items, it must not contain CR, LF, TAB, FF characters (#13, #10, #9, #12).

Unicode notes:

Internally, text is stored as Unicode. **SetItemTextA** (and **SetItemText** in non-Unicode versions of Delphi) converts ANSI to Unicode. For a text item, a code page for conversion is calculated basing on the its Charset 468 . If it is equal to DEFAULT_CHARSET, and for non-text items, Style 193 .DefCodePage 421 is used.

Methods type: viewer-style. Reformatting and repainting are required only for text items.

See also methods:

- GetItemText -W -A 230 ;
- Format 217 .

See also methods of TCustomRichViewEdit:

- SetItemTextEd -A -W 385 ;
- SetCurrentItemText -A -W 380 .

See also:

- Modifying RichView Items 85 ;
- Unicode in RichView 99 .

1.5.1.1.3.130 TRichView .SetItemVAlign

Changing vertical alignment (position relative to the line) of the **ItemNo**-th item

```
procedure SetItemVAlign(ItemNo: Integer; VAlign: TRVVAlign 695 );
```

(introduced in version 12)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount 180 -1. Items of subdocuments (table cells 605) are not included in the items range of the main document; for items in cells, use Cell.GetRVData 613 .SetItemVAlign.

VAlign – new value of vertical alignment.

This method must not be called for text items 120 , tables 130 , *breaks* 125 , tabs 121 , list markers 131 , footnotes 136 and endnotes 135 .

Method type: viewer-style. Requires reformatting and repainting.

See also methods:

- GetItemVAlign 230 .

See also properties:

- ItemCount 180 .

See also methods of RichViewEdit:

- SetItemVAlignEd 386 ;
- SetCurrentItemVAlign 380 .

See also:

- Modifying RichView items 85

1.5.1.1.3.131 TRichView.SetListMarkerInfo

Makes the given paragraph bulleted or numbered.

```
function SetListMarkerInfo(AItemNo, AListNo,
  AListLevel, AStartFrom, AParaNo: Integer;
  AUseStartFrom: Boolean): Integer;
```

(introduced in version 1.7)

Parameters:

AItemNo – index of the item, in range from 0 to ItemCount[180]-1. This is not necessary the index of list marker, it can be index of any item in the given paragraph. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetListMarkerInfo. If this paragraph does not have list marker, a new list marker is created; otherwise, properties of the existing marker are modified.

If **AItemNo**<0 or >=ItemCount[180], a new marker is added at the end of the document. It's assumed that at least one item will be added after such marker in the same paragraph (see Valid Documents[105]).

AListNo – style of this list maker. This is an index in the collection of list styles (Style[193].ListStyles[429]).

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[**AListNo**].Levels[493]).

AStartFrom: if **AUseStartFrom**=*True*, and this is a numbered list, this parameter is a value of list counter for this marker. If **AUseStartFrom**=*False*, numbering is continued.

AParaNo is used to define paragraph style of marker (if it is added as the last item, i.e. **AItemNo**<0). This is an index in the collection of paragraph styles (Style[193].ParaStyles[431]). If this marker is added to the existing paragraph, this parameter is ignored.

Method type: viewer-style.

Important note: list marker cannot be the only item in paragraph. It must be followed by other item. If you need empty markered paragraph, call Add[203] to add empty text item after the list marker. See also: rules for valid documents[105]

Return value: item index of the marker.

See also methods:

- GetListMarkerInfo[233];
- Format[217].

See also properties:

- Style[193];
- ItemCount[180].

See also methods of TCustomRichViewEdit

- ApplyListStyle[331];
- RemoveLists[367];
- ChangeListLevels[338].

See also:

- List Markers[131].

1.5.1.1.3.132 TRichView.SetPictureInfo

Changes main properties of the item of picture[122] or *hot-picture*[125] type.

```
function SetPictureInfo(ItemNo: Integer; const  AName: TRVUnicodeString[694];
  Agr: TGraphic; AVAlign: TRVVAlign[695]; const ATag: TRVTag[692]): Boolean;
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture[122] or *hot-picture*[125] type (*rvsPicture or rvsHotPicture*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item. Items of subdocuments (table cells[605]) are not included in the items range of the main document; for items in cells, use Cell.GetRVData[613].SetPictureInfo.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemText[267] method.

Agr – graphic object. If this item does not use a shared image (by default), **Agr** must be an unique graphic object (you cannot assign the same graphic object to two or more items) or the value returned by GetPictureInfo[236] for this item (the image is shared if *rvepShared* extra item property[673] is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetPictureInfo[236] for this item, the previously used graphic object is freed.

AVAlign – vertical alignment of the picture.

ATag – *tag* of the item.You can use value returned by GetPictureInfo[236] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTag[267] method.

Return value: is reformatting needed? (are the size and the vertical align of the new picture not equal to the size and the vertical align of the old one?)

Method type: viewer-style.

Additional item properties are assigned by the methods SetItemExtraIntProperty[266] and SetItemExtraStrProperty[266] (including stretching, transparency for bitmaps, link to file name, alternative text for saving to HTML).

Example 1

```
RichView1.SetPictureInfo(ItemNo, 'my image', Bitmap2, rvvaBaseline, 'image tag');
```

Example 2 (incorrect!)

```
bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
RichView1.SetPictureInfo(ItemNo1, '', bmp, rvvaBaseline, '');
RichView1.SetPictureInfo(ItemNo2, '', bmp, rvvaBaseline, ''); // incorrect! you cannot assi
  // the same graphic object to different items
```

Example 2 (the same, but correct)

```
bmp1 := TBitmap.Create;
bmp1.LoadFromFile('c:\image.bmp');
RichView1.SetPictureInfo(ItemNo1, '', bmp1, rvvaBaseline, '');
bmp2 := TBitmap.Create;
bmp2.Assign(bmp1);
RichView1.SetPictureInfo(ItemNo2, '', bmp2, rvvaBaseline, '');
```

Example 2 (the same, using shared images)

```
bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
RichView1.SetPictureInfo(ItemNo1, '', bmp, rvvaBaseline, '');
RichView1.SetItemExtraIntProperty(ItemNo1, rvepShared, 1);
RichView1.SetPictureInfo(ItemNo2, '', bmp, rvvaBaseline, '');
RichView1.SetItemExtraIntProperty(ItemNo2, rvepShared, 1);
// you must free bmp yourself when RichView1 is cleared
```

See also methods:

- GetPictureInfo 236;
- GetItemStyle 229;
- Format 217;
- SetItemExtraIntProperty 266;
- SetItemExtraStrProperty 266.

See also properties:

- ItemCount 180.

See also methods of RichViewEdit:

- SetPictureInfoEd 386;
- SetCurrentPictureInfo 381.

See also:

- Modifying RichView items 85;
- Item types 64;
- *Tags* 68.

1.5.1.1.3.133 TRichView .SetSelectionBounds

Selects part of RichView document (for copying to the Clipboard)

```
procedure SetSelectionBounds(StartItemNo, StartItemOffs,
  EndItemNo, EndItemOffs: Integer);
```

Parameters

StartItemNo – index of the first selected item.

StartItemOffs:

- if the first item is a text item 120, then the selection start is before the **StartItemOffs**-th character of string (characters in strings are counted from 1, the last position (after the text item) is text length+1). The exception is empty text items formatted with style having EmptyWidth 469>0; for them, the position after the item is 2.
- if the first item is not a text, then
  - if **StartItemOffs**=0, then the selection start is before the first item;
  - if **StartItemOffs**=1, then the selection start is after the first item.

EndItemNo – index of the last selected item.

EndItemOffs:

- if the last item is a text item, then the selection end is before the **EndItemOffs**-th character of string (characters in strings are counted from 1, the last position (after the text item) is text length+1). The exception is empty text items formatted with style having EmptyWidth 469>0; for them, the position after the item is 2.
- if the last item is not a text, then
  - if **EndItemOffs**=0, then the selection end is before the last item
  - if **EndItemOffs**=1, then the selection end is after the last item.

Caret position

In TRichViewEdit 316, the caret is always at the end of the selection, so this method can be used to move caret.

Selection in table 130 cells

Before making selecting inside a table cell, activates its editing:

```
var RVData: TCustomRVFormattedData 650;

RVData := TCustomRVFormattedData 650(Table.Cells 574[r,c].Edit 651);
RVData.SetSelectionBounds(...);
```

Deselection

There are two ways to deselect:

- to pass **StartItemNo**=-1
- to pass **StartItemNo**=**EndItemNo** and **StartItemOffs**=**EndItemOffs** (in editor, the caret is moved to the specified position).

Please be careful and do not specify incorrect values.

Additional information

This method does not repaint the component (use Invalidate)

This method generates OnSelect[301] event.

There are two useful methods to work with SetSelectionBounds: GetOffsBeforeItem[235] and GetOffsAfterItem[235].

This method must be called only when the document is formatted.

If you want to set selection bounds using richedit-like parameters (SelStart and SelLength), use RVSetSelection from RVLinear[666] unit instead.

See also methods:

- SelectAll[257];
- Deselect[216];
- GetSelectionBounds[237].

See also events:

- OnSelect[301].

See also:

- Working with selection[81];
- How to: move a caret to the beginning or to the end of document in TRichViewEdit[717];
- Example how to move caret to the beginning of paragraph[405].

See also:

- Edit[651] method of table cell.

1.5.1.1.3.134 TRichView .StartAnimation

Starts animation of pictures.

```
procedure StartAnimation;
```

(introduced in version 10)

Alternatively, animation may start automatically when document is formatted[217], if AnimationMode[169] = *rvaniOnFormat*.

See also methods:

- ResetAnimation[249];
- StopAnimation[273].

See also properties:

- AnimationMode[169].

See also:

- Animation in TRichView[90];
- Pictures[122];
- *Hot-Pictures*[125].

1.5.1.1.3.135 TRichView.StartLiveSpelling

Starts live spelling check (runs a live spelling thread).

```
procedure StartLiveSpelling;
```

(introduced in v1.9)

OnSpellingCheck[301] event must be assigned.

If live spelling thread is already active, this method clears all live spelling underlines and restarts the thread from the beginning of document.

Note: If you use RichViewActions and want to start live spelling immediately when document is loaded, call **StartLiveSpelling** in **TrvActionOpen.OnOpenFile**.

See also property of TRichViewEdit:

- LiveSpellingMode[327].

See also:

- Live spelling check in RichView[101].

1.5.1.1.3.136 TRichView.StopAnimation

Stops animation of pictures.

```
procedure StopAnimation;
```

(introduced in version 10)

Clear[210] may stop animations as well.

See also methods:

- StartAnimation[272];
- ResetAnimation[249].

See also properties:

- AnimationMode[169].

See also:

- Animation in TRichView[90];
- Pictures[122];
- *Hot-Pictures*[125].

1.5.1.1.3.137 TRichView.UpdatePaletteInfo

Call this method when screen color resolution changed (see WM_DISPLAYCHANGE window message). You need not to call this method if DoInPaletteMode[177]=rvpaDoNothing.

```
procedure UpdatePaletteInfo;
```

Example[178]

You also need to call UpdatePaletteInfo if you create TRichView at run-time (instead of placing at design-time on form) after you have created it.

This method is necessary only in 256-color display mode (not supported by new versions of Windows).

See also properties:

- DoInPaletteMode[177].

See also methods of TRVPrint:

- UpdatePaletteInfo[554].

1.5.1.1.4 Events

1.5.1.1.4.1 TRichView.OnAddStyle

Occurs when new style is added.

```
type
  TRVAddStyleEvent = procedure (Sender: TCustomRichView[162];
    StyleInfo: TCustomRVInfo[463]) of object;

property OnAddStyle: TRVAddStyleEvent;
```

(introduced in version 11)

This event occurs when a new style is added (to TextStyles[437], ParaStyles[431], or ListStyles[429] collections of the linked TRVStyle[418] component) as a result of:

- RTF (Rich Text Format) reading (both loading and insertion)
- RVF (RichView Format) insertion;
- (if style templates are used[195]) auto-correction of ParaStyleTemplateId[473] property of text styles;
- (in editor) changing the current text style on keyboard language change (see *rvoAutoSwitchLang* in EditorOptions[325]);
- (in editor) ApplyStyleTemplate[333], ApplyParaStyleTemplate[332], ApplyTextStyleTemplate[334];
- (in editor, if style templates are used[195]) when a user presses **Enter** (because of applying NextId[496] property of a style template).

Notes about additional components:

- HTML importers call this event;
- RichViewActions do not call this event, they use TRVAControlPanel.OnAddStyle instead.

1.5.1.1.4.2 TRichView.OnAssignImageFileName

Occurs when inserting an image file in TRichView.

```
type
  TRVAssignImageFileNameEvent = procedure (Sender: TCustomRichView[162];
    var FileName: TRVUnicodeString[694]) of object;

property OnAssignImageFileName: TRVAssignImageFileNameEvent;
```

(introduced in version 16; changed in version 18)

This event occurs if *rvoAssignImageFileNames* is included in Options[182].

In this event, you can modify **FileName** (for example, change a full path to a relative path).

This event may happen on:

- RTF loading[243] (reading external images),
- HTML loading (using importer components),
- pasting files[361], accepting files and links to images as a result of drag&drop[89],
- RichViewActions (actions for inserting images and assigning background images of tables).

The following properties are set:

- table.BackgroundImageFileName[568]
- cell.BackgroundImageFileName[606]
- rvespImageFileName[676] extra item properties of pictures[122] and hot-pictures[125].

1.5.1.1.4.3 TRichView.OnCheckpointVisible

Occurs when *checkpoint* (with RaiseEvent flag = *True*) becomes visible as a result of vertical scrolling.

```
type
  TRVCheckpointVisibleEvent = procedure(Sender: TCustomRichView;
    CheckpointData: TCheckpointData) of object;
```

```
property OnCheckpointVisible: TRVCheckpointVisibleEvent;
```

This event occurs if:

- RichView.CPEventKind[173] = *cpeWhenVisible* for *checkpoint* that becomes visible;
- RichView.CPEventKind = *cpeAsSectionStart* for visible *checkpoint* or the nearest *checkpoint* above visible area.

CheckpointData parameter identifies the *checkpoint*. It's possible that **CheckpointData**=*nil* (no checkpoint visible).

This event never occurs in TRichViewEdit.

See also properties:

- CPEventKind[173].

See also methods extracting information from CheckpointData:

- GetCheckpointInfo[220];
- GetCheckpointXY[221];
- GetCheckpointYEx[222];
- GetCheckpointItemNo[221];
- GetCheckpointNo[221].

See also

- Checkpoints[65].

1.5.1.1.4.4 TRichView.OnCMHintShow

Allows to define custom message handler for hint display.

```
type
  TRVCMHintShowEvent = procedure (Sender: TCustomRichView;
    var AMessage: TCMHintShow) of object;

property OnCMHintShow: TRVCMHintShowEvent;
```

This event occurs when the control receives CM_HINTSHOW message. The event occurs before the default processing of this message.

Assign **AMessage**.Result = 0 to perform the default hint processing after the event, assign non-zero value to cancel the default processing.

Parameter:

AMessage – message data for CM_HINTSHOW

See also:

- CustomHint property
- OnItemHint[281]

1.5.1.1.4.5 TRichView.OnControlAction

Occurs when some important operation is performed on control inserted in RichView

```
type
  TRVControlAction =
    (rvcaAfterRVFLoad, rvcaDestroy,
     rvcaMoveToUndoList, rvcaMoveFromUndoList,
     rvcaDestroyInUndoList, rvcaBeforeRVFSave,
     rvcaAfterRVFSave);

  TRVControlActionEvent =
```

```
    procedure (Sender: TCustomRichView;
      ControlAction: TRVControlAction;
      ItemNo: Integer;
      var ctrl: TControl) of object;

property OnControlAction: TRVControlActionEvent;
```

(Introduced in version 1.3, changed in v1.4 and v10)

Warning: this event can occur from RichView destructor (when clearing document before destruction). Be careful if you access other controls from this event: they may be already destroyed!

This event occurs when some changes were done (or are being done) on inserted control **ctrl**.

Position of this control in the document is defined by the "hidden" parameter **Sender**.Style[193].RVData (must be typecasted to TCustomRVData[650]) and **ItemNo** parameter. **ItemNo** is an index of item (containing **ctrl**) in the document **Sender**.Style.RVData (it may be RVData[187] of the main RichView (**Sender**), cell[605], or RVData[187] of cell inplace editor).

ControlAction	Meaning
rvcaAfterRVFLoad	The control was created and loaded from RVF (file, stream, database field or Clipboard);
rvcaDestroy	The control will be destroyed soon (removing items from RichView, or destroying RichView itself)
rvcaMoveToUndoList	The control was moved to undo/redo buffer. It will not be displayed, but it will not be destroyed yet.
rvcaMoveFromUndoList	The control was moved from undo/redo list back to the editor.
rvcaDestroyInUndoList	The control will be destroyed together with undo/redo buffer (undo item); this control is already not in the editor; only **ctrl** parameter is valid in this case, **ItemNo**=-1;
rvcaBeforeRVFSave	Before saving to RVF
rvcaAfterRVFSave	After saving to RVF

Note: do not perform any changes on RichView when processing this event.

Note: TRVControlAction is defined in RVStyle unit.

See also:

- OnItemAction[280].

1.5.1.1.4.6 TRichView .OnCopy

Occurs when copying to the Clipboard

```
property OnCopy: TNotifyEvent;
```

(Introduced in version 1.4)

Occurs inside CopyDef[212] and Copy[212] methods, or when copying from keyboard.

This event allows copying to the Clipboard in your own formats (occurs between Clipboard.Open and Clipboard.Close).

See also:

- TRichViewEdit.OnPaste[399];

- Working with Clipboard 82.

1.5.1.1.4.7 TRichView.OnGetItemCursor

Allows defining custom custom cursors for items.

```
type
  TRVGetItemCursorEvent = procedure (Sender: TCustomRichView 162;
    RVData: TCustomRVData 650; ItemNo: Integer; var Cursor: TCursor) of object;

property OnGetItemCursor: TRVGetItemCursorEvent;
```

(introduced in version 12)

This event occurs when mouse pointer moves over the item.

Input Parameters

RVData – document containing this item (RichView.RVData 187, or table cell, or cell inplace editor's RVData 187)

ItemNo – index of the item inside **RVData**.

Output Parameter

Cursor – cursor to display.

See also:
- OnItemHint 281.

1.5.1.1.4.8 TRichView.OnHTMLSaveImage

Occurs when TRichView saves item containing image to HTML file or stream. This event allows to modify saving procedure for all (or some) images.

```
type
  TRVHTMLSaveImageEvent =
    procedure (Sender: TCustomRichView; RVData: TCustomRVData;
      ItemNo: Integer; const Path: TRVUnicodeString 694;
      BackgroundColor: TColor; var Location: String;
      var DoDefault: Boolean) of object;

property OnHTMLSaveImage: TRVHTMLSaveImageEvent;
```

(introduced in version 1.4; changed in version 16)

This event is called for the following item types:
- pictures 122, *hot-pictures* 125,
- *bullets* 128, *hotspots* 129,
- list markers 131 (paragraph bullets) with images,
- tables 130 (saving background image 567),
- for background image 169 and cells' background images 605.

There is more easy-to-use event: OnSaveImage2 296.

By default:

Images are saved in separate files as Jpegs (or as bitmaps for Delphi2, or if RVDONOTUSEJPEGIMAGE is defined) and have names composed as ImagesPrefix + Number + '.jpg', where
- ImagesPrefix – parameter of SaveHTML 250/SaveHTMLEx 251;
- Number – some number making this file name unique (image files can override existing ones or not, depending on parameters of SaveHTML/SaveHTMLEx). *Bullets* and *hotspots* items having the same image list, image index and background color are saved in the same file (the same is for list markers).

Transparent color of *bullets*, *hotspots* and transparent image formats (TIcon, TMetafile, gifs) is replaced with background color (which can be color of component (TRichView.Color[173]/TRVStyle.Color[421]), color of cell (TRVTableCellData.Color[608]) or color of paragraph background (TRVBackroundRect.Color[501]).

You can define formats that must not be converted to Jpeg[721].

Input Parameters

RVData – document containing item that is being saved to image; it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor.

ItemNo – index of this item inside **RVData**. If this event is called for background image (document or cell), **ItemNo**=-1.

Path – destination directory of HTML file (or Path parameter of SaveHTMLToStream[252] /SaveHTMLToStreamEx[252]).

BackgroundColor – color of background below the item (can be ignored, if you can preserve transparency of image)

Location – if *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML[250] /SaveHTMLEx[251], this is a path to the image defined in the *rvespImageFileName* item property[676], relative to **Path**. Otherwise, initial value of this parameter is empty.

Output Parameters

Location – set this parameter to file name of image where you saved it. This string will be inserted in HTML file.

DoDefault – set to *False* if you saved this item as image yourself, to disallow default saving. If **DoDefault** is *True*, then after calling this event:

- if *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML[250]/SaveHTMLEx[251] and value of *rvespImageFileName* item property[676] is not empty, this value is written in HTML (relative to **Path**); otherwise
- OnSaveImage2[296] is called (if its handler is assigned); if it is not assigned or OnSaveImage2's DoDefault parameter is *True*, the default image saving procedure is called.

Inside this event you can use:

- **Sender**.imgSaveNo – value which was used as Number for file name for the previously saved image;
- **RVData**.GetNextFileName – function returning image file name with the given image prefix and extension:

```
function GetNextFileName(const ImagesPrefix, Path, Ext: TRVUnicodeString[694];
  var imgSaveNo: Integer; OverrideFiles: Boolean): TRVUnicodeString[694];
```

where

ImagesPrefix – the first part of image file name (for example, you can set it to ImagesPrefix parameter of SaveHTML[250]/SaveHTMLEx[251]);

Path – path for the image (for example, you can use **Path** parameter of this event);

Ext – file extension, for example '.jpg' (you can use GraphicExtension(TGraphicClass(Graphic.ClassType)));

imgSaveNo – this function will use this value incremented one or several times to compose a name of image; returns the incremented value;

OverrideFiles – if set to *False*, this function will check existing files and returns an unique file name (by incrementing **imgSaveNo** several times)

Return value: full file name of the image. You can use it to save image and assign it to **Location** parameter of the event (this file name includes path, use ExtractFileName or ExtractRelativePath).

Alternatively, you can use SavePicture[253].

See also events:

- OnSaveImage2 [296];
- OnSaveItemToFile [297].

See also

- Export to HTML [96].

1.5.1.1.4.9 TRichView .OnImportPicture

Occurs when loading a file or a stream containing links to image files.

```
type
  TRVImportPictureEvent =
    procedure (Sender: TCustomRichView;
    const Location: TRVUnicodeString[694]; Width, Height: Integer;
    var Graphic: TGraphic) of object;

property OnImportPicture: TRVImportPictureEvent;
```

(introduced in version 1.8; changed in version 18)

This event may occur:

- when loading RTF. Usually pictures are saved inside RTF file/stream. But sometimes RTF contains links to external image files (file names or URLs).
- when loading HTML (using RvHtmlImporter or RvHtmlViewImporter components)
- when accepting URL to an image as a result of drag&drop, and *rvoDragDropPicturesFromLinks* is include in EditorOptions [325].

If these links are local (like 'c:\images\image.bmp'), TRichView can load these files itself. But if images are on the Internet (like 'http://www.trichview.com/images/headers/what_is.gif'), it makes sense to process this event and to download the image.

Input parameters

Location – path to the file (local or URL);

Width, Height – reserved, equal to 0;

Graphic – *nil.*

Output parameters

Graphic – set it to the loaded image. If you leave it equal to *nil*, TRichView will attempt to load the image itself.

(TRichView can load image file only if its extension is associated with some graphic class. Delphi does it automatically for TBitmap, TIcon, TMetafile, TJpegImage. If you use third-party graphic classes [720], you can register them using TPicture.RegisterFileFormat class procedure).

If you use RichViewActions, they can use downloader components to download pictures automatically, when loading is initiated by the action (Open, Insert File, Paste and Paste Special actions). This downloader can also be used in OLE drag&drop, see OnBeforeOleDrop and OnAfterOleDrop [388] events.

Example: downloading images from http location using Indy (IdHTTP1: TIdHTTP)

```
uses RVFuncs;

{ TMyRichView.OnImportPicture }
procedure TMyForm.MyRichViewImportPicture(Sender: TCustomRichView;
  const Location: TRVUnicodeString[694]; Width, Height: Integer;
  var Graphic: TGraphic);
var Stream: TMemoryStream;
begin
  if Pos('http://', AnsiLowerCase(Location))=1 then
```

```
  begin
    Stream := TMemoryStream.Create;
    try
      IdHTTP1.Get(Location, Stream);
      Stream.Position := 0;
      Graphic := RVGraphicHandler[710].LoadFromStream(Stream);
    except
      Graphic := nil;
    end;
    Stream.Free;
  end;
end;
```

See also methods of TRichView:

- LoadRTF[243];
- LoadRTFFromStream[244].

See also methods of TRichViewEdit:

- InsertRTFFromFileEd[356];
- InsertRTFFromStreamEd[357].

1.5.1.1.4.10 TRichView.OnItemAction

Occurs on insertion, deletion and some other modifications of item

```
type
  TRVItemAction = (rviaInserting, rviaInserted,
    rviaTextModifying, rviaDestroying, rviaMovingToUndoList)

  TRVItemActionEvent = procedure (Sender: TCustomRichView[162];
    ItemAction: TRVItemAction; Item: TCustomRVItemInfo[563];
    var Text: TRVUnicodeString[694]; RVData: TCustomRVData[650]) of object;

property OnItemAction: TRVItemActionEvent;
```

(introduced in version 1.7; modified in v18)

Parameters

Item – item[563] for which this event is generated.

Text – text associated with this item. If this is a text item, this text is displayed in TRichView.

RVData – document containing Item (it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor).

ItemAction identifies operation that calls this event.

ItemAction	Meaning
rviaInserting	The event occurs before the item is inserted in **Sender** (in **RVData**). You can modify **Text**. This event also occurs when moving items to cell inplace editor and back to the cell.
rviaInserted	The event occurs after the item is inserted in the **Sender** (in **RVData**). Ignore **Text** parameter. This event also occurs when moving items to cell inplace editor and back to the cell.

rviaMovingToUndoList	The event occurs when item is moved (from **RVData**) to undo list. Ignore **Text** parameter.
rviaDestroying	Occurs when item is destroyed. Ignore **Text** parameter. If the item is in undo/redo list, **RVData** is *nil*. Otherwise, this is an object containing this item.
rviaTextModifying	Occurs when item text (**Text**) is modified as a result of editing operation. Be careful when modifying text. It's highly not recommended to do any modification of **Text** except for replacing one characters to others.

Example (converting all text to upper case):

```
procedure TForm1.RichViewEdit1ItemAction(Sender: TCustomRichView;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo;
  var Text: TRVUnicodeString[694]; RVData: TCustomRVData);
begin
  if (ItemAction in [rviaInserting, rviaTextModifying]) and
     (Item.StyleNo>=0) then
    Text := AnsiUpperCase(Text);
end;
```

Warning: this event can occur from TRichView destructor (when clearing document before destruction). Be careful if you access other controls from this event, they may be already destroyed.

See also events:

- OnControlAction[275].

See also methods:

- GetItem[225];
- GetItemNo[229].

1.5.1.1.4.11 TRichView.OnItemHint

Allows to define custom popup hints for items.

```
type
  TRVItemHintEvent = procedure (Sender: TCustomRichView[162];
    RVData: TCustomRVData[650]; ItemNo: Integer;
    var HintText: TRVUnicodeString[694]) of object;

property OnItemHint: TRVItemHintEvent;
```

This event occurs when mouse pointer moves over the item.

Parameters

RVData – document containing this item (RichView.RVData[187], or table cell, or cell inplace editor's RVData[187])

ItemNo – index of the item inside **RVData**.

HintText – on input, it's a value of *rvespHint*[676] item property[676]. On output, it's a message that will be displayed.

Item popup hint can be displayed if *rvoShowItemHints* is in RichView.Options[182] and RichView.ShowHint=*True*.

Cell Hint[609]s cannot be altered in this event.

See also:

- OnGetItemCursor[277].

Example

Besides the default hint, this code displays:

- for hyperlinks: tag;
- for images: Delphi class type, width and height.

```
procedure TMyForm.MyRichViewEditItemHint(Sender: TCustomRichView[162];
  RVData: TCustomRVData[650]; ItemNo: Integer; var HintText: TRVUnicodeString[694]);
var Tag: TRVTag[692];
    VAlign: TRVVAlign[695];
    Name: TRVUnicodeString[694];
    gr: TGraphic;
begin
  if RVData.GetItem[225](ItemNo).GetBoolValueEx(rvbpJump, Sender.Style[193]) then
  begin
    if HintText<>'' then
      HintText := HintText+#13;
    HintText := HintText + 'Target: '+RVData.GetItemTag[230](ItemNo);
  end;
  if (RVData.GetItemStyle[229](ItemNo)=rvsPicture[711]) or
     (RVData.GetItemStyle[229](ItemNo)=rvsHotPicture[711]) then
  begin
    if HintText<>'' then
      HintText := HintText+#13;
    RVData.GetPictureInfo[236](ItemNo, Name, gr, VAlign, Tag);
    HintText := HintText+gr.ClassName+' ('+IntToStr(gr.Width)+','+IntToStr(gr.Height)+')';
  end;
end;
```

1.5.1.1.4.12 TRichView .OnJump

Occurs when user clicks on hypertext item.

```
type
  TJumpEvent = procedure(Sender: TObject; id: Integer) of object;

property OnJump: TJumpEvent;
```

Parameter

id is an identifier of hypertext link (the first link in the document has id=FirstJumpNo[179], the next link has **id**=FirstJumpNo[179]+1, and so on).

In editor, this numeration is available only when the editor works in hypertext mode:

- while user holds **Ctrl** key (see EditorOptions[325]) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly[328]=True.

So, when using this event in editor, perform all actions that use this **id** before any action changing document or moving focus out of editor.

In viewer, this even can also be initiated using keyboard, if TabNavigation[194]<>*rvtnNone*. User uses **Tab** to highlight the hyperlink, then **Enter** to activate it (to call this event).

Example:

```
procedure TMyForm.MyRichViewJump(Sender: TObject; id: Integer);
var ItemNo: Integer;
    RVData: TCustomRVFormattedData[650];
```

```
    s: String;
begin
  MyRichView.GetJumpPointLocation(id, RVData, ItemNo);
  s := RVData.GetItemTag(ItemNo);
  Application.MessageBox(PChar(s), 'Link', MB_OK or MB_ICONINFORMATION);
end;
```

See also events:

- OnRVMouseMove.

See also:

- RichView hypertext.

1.5.1.1.4.13 TRichView.OnPaint

Allows painting over the richview document

```
type
  TRVPaintEvent = procedure (Sender: TCustomRichView;
    Canvas: TCanvas; Prepaint: Boolean) of object;

property OnPaint: TRVPaintEvent;
```

(introduced in version 1.7)

Parameters:

Canvas – canvas for painting.

Prepaint – reserved for future use.This event is called when all standard painting is done.

This event does not affect printing (see OnPagePrepaint, OnPagePostpaint).

Demo project:

- **Demos\Delphi\Assorted\Custom Draw\CustomDraw**
- **Demos\CBuilder\Assorted\Custom Draw\CustomDraw**

1.5.1.1.4.14 TRichView.OnProgress

Occurs on long operations

```
type
  TRVLongOperation = ( // defined in RVStyle unit
    rvloRTFRead, rvloRTFWrite,
    rvloRVFRead, rvloRVFWrite,
    rvloHTMLRead, rvloHTMLWrite,
    rvloTextRead, rvloTextWrite,
    rvloConvertImport, rvloConvertExport,
    rvloXMLRead, rvloXMLWrite,
    rvloDocXRead, rvloDocXWrite,
    rvloOtherRead, rvloOtherWrite
    );
  TRVProgressStage = // defined in RVStyle unit
    (rvpstgStarting, rvpstgRunning, rvpstgEnding);
type
  TRVProgressEvent = procedure (Sender: TCustomRichView;
    Operation: TRVLongOperation; Stage: TRVProgressStage;
    PercentDone: Byte) of object;
```

```
property OnProgress: TRVProgressEvent;
```

(introduced in v1.9, changed in v14)

First, it is called with parameter **Stage**=*rvpstgStarting*.

Next, it is called several times with parameter **Stage**=*rvpstgRunning*, with **PercentDone** in the range 0..100.

Finally, it is called with parameter **Stage**=*rvpstgEnding*.

This event occurs on the following operations:

- reading/writing RTF;
- reading/writing RVF [93];
- writing HTML;
- writing DocX;
- reading/writing a plain ANSI or Unicode text;
- when calling TRVOfficeConverter.ImportRV [538] and ExportRV [537].

Note: the event is called if the document/source file is large enough.

See also:

- TRVOfficeConverter.OnConverting [539]

1.5.1.1.4.15 TRichView.OnReadHyperlink

Occurs when reading hyperlinks (for example, from RTF file)

```
type
  TRVReadHyperlink =
    procedure (Sender: TCustomRichView;
      const Target, Extras: TRVUnicodeString[694];
      DocFormat: TRVLoadFormat[668];
     var StyleNo: Integer; var ItemTag: TRVTag[692];
     var ItemName: TRVUnicodeString[694]) of object;

property OnReadHyperlink: TRVReadHyperlink;
```

(changed in version 18)

By default, hyperlink targets is saved in the item tag [68]. Use this event to customize loading.

Input parameters:

Target – target of hypertext link (for example, an Internet address, a file name, or a bookmark/anchor).

Extras – additional information stored with link. For example, if **Extras**='\o"hint_text"', then **Target** is read from RTF and the link has a popup hint 'hint_text'.

TRichView automatically writes items hints in *rvespHint* item's extra string property [676], and adds '#' to the beginning of links to bookmarks.

DocFormat – format of loaded document.

DocFormat	Meaning
rvlfRTF	The event occurs when reading RTF files containing hyperlinks (RTF loading, pasting, or accepting on drag&drop).
rvlfURL	The event occurs: 1) on drag&drop, when accepting URL dropped to TRichViewEdit [316] (*rvddURL* must be in AcceptDragDropFormats [321]).

	2) when pasting from the Clipboard, when pasting URL to TRichViewEdit[316] (*rvddURL* must be in AccepPasteFormats[321]).
rvlfHTML	The event occurs when reading HTML files using ***TRVHtmlImporter***, if *rvhtmloAutoHyperlinks* is excluded from RVHtmlImporter.Options. Or the event occurs when reading HTML files using ***TRVHtmlViewImporter***, if RVHtmlViewImporter.AutoHyperlinks=*False*.

StyleNo:

- if the item is a picture, it is initially equal to *rvsHotPicture*. You can change it to *rvsPicture*, if you want to disable this link.
- if the item is a text, it is index of style[437] of this text. You can modify it to point to another (hypertext[479]) style. If **DocFormat**=*rvlfURL*, the initial style may be not hypertext, so you need to provide the index of hypertext style in this parameter.

ItemName:

- if the item is a picture, it is initially equal to '' (empty string) . You can set it to value which will be assigned to name of this item.
- if the item is a text, it is initially a text to display. You can modify it.

Output parameters:

StyleNo – style that will be used for the loaded item. See comments for input parameters.

ItemName – value that will be assigned to item text. See comments for input parameters.

ItemTag – value that will be assigned to tag[68] of this item. A good place to store hyperlink target.

Example 1

This example does the same work as it the event is not assigned: reading the link target in the item tag[68].

```
procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[694]; DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag[692];
  var ItemName: TRVUnicodeString[694]);
begin
  ItemTag := Target;
end;
```

More complicated example

This example assumes that you included *rvddURL* in AcceptDragDropFormat[321] and want to accept hyperlinks. Besides, it's assumed that "Allow adding styles dynamically" is set in the component editor[168] (or properties are set to the proper values to allow saving the changed collection of text styles).
This code uses blue underlined font for hyperlinks.

```
procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[694]; DocFormat: TRVLoadFormat[668];
  var StyleNo: Integer; var ItemTag: TRVTag[692];
  var ItemName: TRVUnicodeString[694]);
var FontStyle: TFontInfo[478];
begin
  ItemTag := Target;
  if DocFormat=rvddURL then
  begin
```

```
    FontStyle := TFontInfo[478].Create(nil);
    FontStyle.Assign(Sender.Style[193].TextStyles[437][StyleNo]);
    FontStyle.Color[469] := clBlue;
    FontStyle.Style[475] := FontStyle.Style + [fsUnderline];
    FontStyle.Jump[479] := True;
    StyleNo := Sender.Style.FindTextStyle[440](FontStyle);
    FontStyle.Free;
  end;
end;
```

The same example using RichViewActions:

```
procedure TMyForm.MyRichViewReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString[694]; DocFormat: TRVLoadFormat[668];
  var StyleNo: Integer; var ItemTag: TRVTag[692];
  var ItemName: TRVUnicodeString[694]);
begin
  ItemTag := Target;
  if DocFormat=rvddURL then
    StyleNo := rvActionInsertHyperlink1.GetHyperlinkStyleNo(Sender as TRichViewEdit);
end;
```

See also events:

- OnWriteHyperlink[302].

See also properties of RVRTFReadProperties[187]:

- BasePathLinks[305].

1.5.1.1.4.16 TRichView .OnReadMergeField

Occurs when reading merge fields from RTF file.

```
type

  TRVReadMergeFieldEvent =
    procedure (Sender: TCustomRichView;
      const FieldName, Extras: TRVUnicodeString[694];
      DocFormat: TRVLoadFormat;
     var StyleNo: Integer; var ItemTag: TRVTag[692];
     var ItemName: TRVUnicodeString[694]) of object;

property OnReadMergeField: TRVReadMergeFieldEvent;
```

(introduced in version 18)

Input parameters:

FieldName – merge field name.

Extras – reserved, always empty.

DocFormat – format of loaded document. It is always equals to *rvlfRTF.*

StyleNo:

- if the item is a picture, it is either *rvsPicture* or *rvsHotPicture*. You can change it, but it must still be *rvsPicture* or *rvsHotPicture* (or another item type inherited from TRVGraphicItemInfo[563])
- if the item is a text, it is an index of style[437] of this text. You can modify it to point to another text style.

ItemName:

- if the item is a picture, it is initially equal to '' (empty string) . You can set it to value which will be assigned to name of this item.
- if the item is a text, it is initially a text to display. You can modify it.

Output parameters:

StyleNo – style that will be used for the loaded item. See comments for input parameters.

ItemName – value that will be assigned to item text. See comments for input parameters.

ItemTag – value that will be assigned to tag[68] of this item. You can store **FieldName** here. However, please note that targets of hyperlinks[69] are also stored here; so, if you choose to store hyperlink targets and field names in tags, you need to encode them somehow.

If this event is not assigned, merge fields are not loaded (their content is loaded as a normal text and pictures).

Example:

Loading text merge fields as field names inside {}.

```
procedure TMyForm.MyRichViewReadMergeField(Sender: TCustomRichView;
  const FieldName, Extras: TRVUnicodeString[694]; DocFormat: TRVLoadFormat[668];
  var StyleNo: Integer; var ItemTag: TRVTag[692];
  var ItemName: TRVUnicodeString[694]);
begin
  if StyleNo >= 0 then
    ItemName := '{' + FieldName + '}';
end;
```

1.5.1.1.4.17 TRichView.OnRVDblClick

Occurs when user double-clicks the primary mouse button when the mouse pointer is above the item in RichView (or on single left-click, if *rvsSingleClick* is in Options[182])

```
type
  TRVDblClickEvent =
    procedure(Sender: TCustomRichView[162]; ClickedWord: TRVUnicodeString[694];
      Style: Integer) of object;

property OnRVDblClick:TRVDblClickEvent;
```

(changed in version 18)

Parameters

Style – style (type) of the clicked item.

If **Style**<0 then the clicked item is not a text item, and **Style** is a type of this item (see Item Types[64]). In this case **ClickedWord** contains a name of this item.

If **Style**>=0, then this is a text item[120], and **Style** is an index in the collection of text styles (Style[193].TextStyles[437]). In this case **ClickedWord** is a word below the mouse pointer. "Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[174] property.

This event is not generated when the user clicks not on an item (i.e. on a background)

See also events:

- OnRVRightClick[291].

See also properties:

- Delimiters[174].

1.5.1.1.4.18 TRichView .OnRVFControlNeeded

Occurs when reading inserted control [126] from RVF file or stream, and this control was saved without its "body".

```
type
  TRVFControlNeededEvent =
    procedure(Sender: TCustomRichView[162]; Name: TRVUnicodeString[694];
      const Tag: TRVTag[692]; var ctrl: TControl) of object;

property OnRVFControlNeeded: TRVFControlNeededEvent;
```

(changed in version 18)

By default, values of all published properties of controls are saved in RVF documents.

If you exclude *rvfoSaveControlsBody* from RichView.RVFOptions [188], none of control properties are saved. Only the item name (value returned by GetItemText [230]) and *tag* (value returned by GetItemTag [230]) are saved. These values allow to identify the control on loading. When reading such RVF files, OnRVFControlNeeded is called for each control.

Input parameters:

Name – name of the control's item (this is a value returned by GetItemTextW [230]; do not confuse with Name property of controls, see Item types [64]).

Tag – *tag* of the control's item (this is a value returned by GetItemTag [230]; do not confuse with Tag property of controls, see "Tags" [68]).

Output parameter:

ctrl – control to insert in document. Create this control in this event. Do not destroy it (RichView will destroy it when necessary).

See also:

- RVF overview [93], contains example.

1.5.1.1.4.19 TRichView .OnRVFImageListNeeded

Occurs when reading *bullet* [128], *hotspot* [129] or list level with imagelist picture [509] from RVF.

```
type
  TRVFImageListNeededEvent =
    procedure(Sender: TCustomRichView[162];
      ImageListTag: Integer;
      var il: TCustomImageList) of object;

property OnRVFImageListNeeded: TRVFImageListNeededEvent;
```

If you do not process this event, RichView is not able to read *bullets* [128], *hotspots* [129] and list level with imagelist from RVF file or stream.

When saving these items, RichView saves Tag of the corresponding ImageList (do not confuse with RichView items' tags [68], this is TCustomImageList.Tag property). This value is passed as **ImageListTag** parameter of this event on reading.

Assign your ImageList to **il** parameter (note: RichView does not own ImageLists, it will not destroy it).

This event is generated for each *bullet*, *hotspot* and list levels with imagelist in RVF, so it is not recommended to perform time consuming calculations in this event.

See also:

- RVF overview [93], contains example.

1.5.1.1.4.20 TRichView.OnRVFPictureNeeded

Occurs when reading picture[122] or *hot-picture*[125] from RVF file or stream, and this picture was saved without its "body"

```
type
  TRVFPictureNeededEvent = procedure (Sender: TCustomRichView[162];
    const ItemName: TRVUnicodeString[694]; Item: TRVNonTextItemInfo[563];
    Index1, Index2: Integer; var gr: TGraphic) of object;

property OnRVFPictureNeeded: TRVFPictureNeededEvent;
```

(the parameters are changed in version 14 and 18)

By default, images are stored inside RVF documents.

If you exclude *rvfoSavePicturesBody* from RichView.RVFOptions[188], images are not saved. Only item name (value returned by GetItemText[230]) and *tag*[68] (value returned by GetItemTag[230]) are saved. These values allow to identify the image on loading. When reading such RVF files, OnRVFPictureNeeded is called for each image.

This event is called for the following objects:

- pictures[122] and hot-pictures[125];
- table[130] background images[567] (even if the saved table did not have a background image initially)
- table cell background images[605] (for each table cell, even for cells that did not have a background image initially)

Input parameters:

ItemName – the name of the item (this is a value returned by GetItemTextW[230]). This parameter is empty when the event is called for a table cell.

Item – the item that will use this image. This is an object of one of the following types: TRVGraphicItemInfo, TRVHotGraphicItemInfo, TRVTableItemInfo[563].

Index1, Index2 – integer values allowing to identify image inside the item. When called for a picture, a hot-picture, or a table, they are (-1, -1). When called for a table cell, Index1 contains the row index, Index2 contains the column index.

Output parameter:

gr – image (for example, of TBitmap type) to insert in the document. Create it in this event. Do not destroy it (RichView will destroy it when necessary)

The following values may be useful:

- Item.Tag
- if Item is TRVGraphicItemInfo: TRVGraphicItemInfo(Item).ImageWidth and TRVGraphicItemInfo(Item).ImageHeight (a size of a stretched picture)
- if Item is TRVGraphicItemInfo: TRVGraphicItemInfo(Item).ImageFileName
- if Item is TRVTableItemInfo: TRVTableItemInfo(Item).BackgroundImageFileName

See also:

- RVF overview[93], contains an example.

1.5.1.1.4.21 TRichView.OnRVMouseDown

Occurs when the user presses a mouse button with the mouse pointer over the control.

```
type
  TRVMouseEvent =
    procedure(Sender: TCustomRichView[162]; Button: TMouseButton;
    Shift: TShiftState; ItemNo, X, Y: Integer) of object;
```

```
property OnRVMouseUp: TRVMouseEvent;
```

Similar to OnMouseDown, but you also have information about index of item under the mouse pointer.

ItemNo is an index of this item, or -1 if there is no item at the position of the mouse pointer.

The OnRVMouseDown event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. **X** and **Y** are the pixel coordinates of the mouse pointer in the client area of the **Sender**.

When clicking on table, this event returns **ItemNo** of this table in the root document, even if user clicked on item inside this table.

Use the following procedure if you want to know the exact clicked item:

```
procedure TForm1.MyRichViewRVMouseDown(Sender: TCustomRichView;
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: Integer);
var LRVData: TCustomRVFormattedData;
    LItemNo, LOffs: Integer;
    pt: TPoint;
begin
  pt := MyRichView.ClientToDocument(Point(X,Y));
  if not MyRichView.GetItemAt(pt.X, pt.Y, LRVData, LItemNo, LOffs, True) then
    exit;
  // item is the LItemNo-th item in LRVData
  // example of using: LRVData.GetItemStyle (LItemNo)
  ..
end;
```

Note: if you want to create code responding on clicking on item, it's recommended to use OnRVMouseUp instead of OnRVMouseDown.

See also events:

- OnRVMouseUp.

1.5.1.1.4.22 TRichView .OnRVMouseMove

Occurs when mouse pointer is moved to the area of hypertext link.

```
type
  TRVMouseMoveEvent = procedure(Sender: TObject; id: Integer) of object

property OnRVMouseMove: TRVMouseMoveEvent
```

id is an identifier of hypertext link (the first link in the document has **id**=FirstJumpNo, the next link has id=FirstJumpNo+1, and so on).

In editor, this numeration is available only when the editor works in hypertext mode:

- while user holds Ctrl key (see EditorOptions) until he/she releases it, or the document is modified, or the editor loses focus;
- if ReadOnly=True.

So, when using this event in editor, perform all actions that use this **id** before any action changing document or moving focus out of editor.

If **id**=-1, the mouse pointer is not above hyperlink.

This event is never called two times in succession for the same hypertext link.

See also events:

- OnJump[282].

See also:

- RichView hypertext overview[69], contains example.

1.5.1.1.4.23 TRichView.OnRVMouseUp

Occurs when the user releases a mouse button that was pressed with the mouse pointer over the component.

```
type
  TRVMouseEvent = Procedure(Sender: TCustomRichView[162];
    Button: TMouseButton; Shift: TShiftState;
    ItemNo, X, Y: Integer) of object;

property OnRVMouseUp: TRVMouseEvent;
```

Similar to OnMouseUp, but you also have information about index of item under the mouse pointer.

ItemNo is an index of this item, or -1 if there is no item at the position of the mouse pointer.

The OnRVMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys. **X** and **Y** are the pixel coordinates of the mouse pointer in the client area of the **Sender**.

When clicking on table, this event returns **ItemNo** of this table in the root document, even if user clicked on item inside this table.

Use the following procedure if you want to know the exact clicked item:

```
procedure TForm1.MyRichViewMouseUp(Sender: TCustomRichView[162];
  Button: TMouseButton; Shift: TShiftState;
  ItemNo, X, Y: Integer);
var LRVData: TCustomRVFormattedData[650];
    LItemNo, LOffs: Integer;
    pt: TPoint;
begin
  pt := MyRichView.ClientToDocument[211](Point(X,Y));
  if not MyRichView.GetItemAt[225](pt.X, pt.Y, LRVData, LItemNo, LOffs, True) then
    exit;
  // item is the LItemNo-th item in LRVData
  // example of using: LRVData.GetItemStyle[229] (LItemNo)
  ..
end;
```

See also events:

- OnRVMouseDown[289];
- OnRVRightClick[291].

1.5.1.1.4.24 TRichView.OnRVRightClick

Occurs when the user releases the right mouse button over the RichView component.

```
type
  TRVRightClickEvent = procedure(Sender: TCustomRichView[162];
    ClickedWord: TRVUnicodeString[694]; Style, X, Y: Integer) of object;

property OnRVRightClick:TRVRightClickEvent;
```

This event is almost obsolete. There are more convenient events:

- OnRVMouseDown[289],
- OnRVMouseUp[291].

OnRVRightClick event occurs if user releases right mouse button over some item of TRichView.

If **Style**<0 then this item is not a text item, and **Style** is a type of this item (such as *rvsPicture*[711], [711] *rvsBreak*[711], [711] *rvsHotspot*[711] or [711] *rvsBullet*[711]). In this case **ClickedWord** contains name of this item (see Item types[64]).

If **Style**>=0, this is a text item, and **Style** is an index in the collection of text styles (Style[193].TextStyles[437]). In this case **ClickedWord** is a word which user has clicked. "Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[174] property.

X,**Y** are screen coordinates of mouse pointer (more convenient for displaying a pop-up menu than client coordinates)

This event has the following limitations (which do not exist in OnRVMouseDown[289], OnRVMouseUp[291]):

- you can not get further information about the clicked item (you do not know index of this item);
- this event is not generated when user clicks not on an item (i.e. on a background).

See also events:

- OnRVDblClick[287].

See also properties:

- Delimiters[174].

1.5.1.1.4.25 TRichView.OnSaveComponentToFile

Occurs when TRichView wants to save inserted control[126] in text, RTF, HTML, or DocX file (or stream).

```
type
  TRVSaveComponentToFileEvent = procedure(
    Sender: TCustomRichView[162]; Path: TRVUnicodeString[694];
    SaveMe: TPersistent; SaveFormat: TRVSaveFormat[685];
    var OutStr:TRVUnicodeString[694]) of object;

property OnSaveComponentToFile: TRVSaveComponentToFileEvent;
```

By default, inserted controls are not saved in HTML, RTF, DocX and text files. But you can use this event to save them.

Input parameters:

SaveMe – component to save.

SaveFormat identifies file format, one of *rvsfText*, *rvsfHTML*, *rvsfRTF, rvsfDocX*.

Path – path where the file is saved. May be used, for example, for saving images (see SavePicture[253] method).

Additional input parameters

Sender. Style[193].RVData, **Sender.**Style[193].ItemNo identify the item to save. RVData – document containing item that is being saved; it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor. ItemNo – index of this item inside RVData.

Output parameter:

OutStr – text to insert in the output file.

For DocX, **OutStr** is inserted inside a "run", i.e. between <w:r> and </w:r>.

Example:

```
procedure TMyForm.MyRichViewSaveComponentToFile(
  Sender: TCustomRichView; Path: TRVUnicodeString[694];
  SaveMe: TPersistent; SaveFormat: TRVSaveFormat;
  var OutStr: TRVUnicodeString[694]);
```

```
var bmp: TBitmap;
  ImageFileName: TRVUnicodeString[694];
begin
 case SaveFormat of
   rvsfText:
     begin
       if SaveMe is TButton then
         OutStr := '['+TButton(SaveMe).Caption+']';
       else if SaveMe is TImage then
         OutStr := '<Image of '+TImage(SaveMe).Hint+'>';
      end;
   rvsfHTML:
     begin
       if SaveMe is TButton then
         OutStr := '<FORM><INPUT type="button"' +
                   'value="' + TButton(SaveMe).Caption +
                   '" onClick="alert(''Test'')"></FORM>';
       else if SaveMe is TImage then begin
         bmp         := TBitmap.Create;
         bmp.Height  := TImage(SaveMe).Height;
         bmp.Width   := TImage(SaveMe).Width;
         bmp.Canvas.Draw(0,0, TImage(SaveMe).Picture.Bitmap);
         ImageFileName := MyRichView.SavePicture(SaveFormat,Path,bmp);
         OutStr      := '<IMG src="'+ImageFileName+'" alt='+
                            TImage(SaveMe).Hint+'>';
         bmp.Free;
       end;
     end;
    rvsfRTF:
      begin
        OutStr := '\plain\b ('+SaveMe.ClassName+')';
      end;
 end;
end;
```

This code saves

- TButton controls in text file as "[Button.Caption]" string;
- TButton controls in HTML file as form with button;
- TImage controls in text file as "<Image of Image.Hint>" string;
- TImage controls in HTML file as image (using SavePicture) with alternative text "Image.Hint";
- All controls in RTF as "(Class of control)".

See also:

- OnSaveItemToFile[297];
- Saving and loading[92];
- Export to HTML[96].

1.5.1.1.4.26 TRichView .OnSaveDocXExtra

Allows saving additional information in DocX.

```
type
  TRVDocXSaveArea = // defined in RVStyle.pas
   (rv_docxs_TextStyle, rv_docxs_ParaStyle,
    rv_docxs_ListStyle, rv_docxs_StyleTemplate,
    rv_docxs_StyleTemplateText, rv_docxs_StyleTemplatePara,
    rv_docxs_CellProps, rv_docxs_RowProps, rv_docxs_TableProps,
```

```
    rv_docxs_SectProps, rv_docxs_Settings, rv_docxs_CoreProperties);
 TRVSaveDocXExtraEvent = procedure (Sender: TCustomRichView;
   Area: TRVDocXSaveArea; Obj: TObject;
   Index1, Index2: Integer; var OOXMLCode: TRVUnicodeString) of object;

property OnSaveDocXExtra: TRVSaveDocXExtraEvent;
```

(changed in version 18)

This event occurs when saving DocX (SaveDocX, SaveDocXToStream).

Text assigned to **OOXMLCode** parameter will be saved in DocX. This text will be encoded as UTF-8.

Area defines a place where **OOXMLCode** will be inserted. Meaning of **Index1**, **Index2** and **Obj** depends on **Area**.

Area	Meaning	Parent XML Element	XML file
rv_docxs_TextStyle	Occurs when saving a text style. **Index1** is an index of the text style in TextStyles collection, **Obj** is a text style object	<w:rPr>	document files*
rv_docxs_ParaStyle	Occurs when saving a paragraph style. **Index1** is an index of the paragraph style in ParaStyles collection, **Obj** is a paragraph style object	<w:pPr>	document files*
rv_rtfs_ListStyle	Occurs when saving a list style and list levels. **Index1** is an index of the list style in ListStyles. **Index2** equals to -1 when saving the list description, or it equals to the level index when saving a list level. **Obj** is a list style object	if **Index2** = -1: <w:abstractNum>; if **Index2** >= 0: <w:lvl>	word/numbering.xml
rv_docxs_StyleTemplate	Occurs when saving a style template as a style sheet item. **Index1** is an index of the style template in StyleTemplates. **Obj** is a style template.	<w:style>	word/styles.xml
rv_docxs_StyleTemplateText	Occurs when saving text attributes of a style template inside a style sheet item. **Index1** is an index of the style template in StyleTemplates. **Obj** is a style template.	<w:rPr> inside <w:style>	word/styles.xml
rv_docxs_StyleTemplatePara	Occurs when saving paragraph attributes of a style template inside a style sheet item. **Index1** is an index of the style template in StyleTemplates. **Obj** is a style template.	<w:pPr> inside <w:style>	word/styles.xml
rv_docxs_CellProps	Occurs when saving table cell properties **Index1** = row index, **Index2** =column index, **Obj** is the table.	<w:tcPr>	document files*

rv_docxs_RowProps	Occurs when saving table row properties. **Index1** = row index, **Obj** is the table[565].	<w:trPr>	document files*
rv_docxs_TableProps	Occurs when saving table properties. **Obj** is the table[565].	<w:tblPr>	document files*
rv_docxs_SectProps	Occurs when saving page properties.	<w:sectPr>	word/document.xml
rv_docxs_Settings	Occurs when saving document settings.	<w:settings>	word/settings.xml
rv_docxs_CoreProperties	Occurs when saving document properties.	<cp:coreProperties>	docProps/core.xml

* - "document files" include:

- word/document.xml
- word/footnotes.xml
- word/endnotes.xml
- XML files used for saving headers and footers (header.xml. footer.xml, etc.)

See also properties:

- DocParameters[176].

See also events:

- OnSaveRTFExtra[299].
- OnSaveHTMLExtra[295].

1.5.1.1.4.27 TRichView .OnSaveHTMLExtra

Allows saving additional information in HTML.

```
type
  TRVHTMLSaveArea = (rv_thms_Head, rv_thms_BodyAttribute,
    rv_thms_Body, rv_thms_End);
  TRVSaveHTMLExtraEvent = procedure (
    Sender: TCustomRichView; Area: TRVHTMLSaveArea;
    CSSVersion: Boolean; var HTMLCode: TRVUnicodeString[694]) of object;

property OnSaveHTMLExtra: TRVSaveHTMLExtraEvent;
```

(changed in version 18)

This event occurs when saving HTML by methods SaveHTML[250] (with **CSSVersion** parameter = False) and SaveHTMLEx[251] (with **CSSVersion**=True).

Text assigned to **HTMLCode** parameter will be saved in HTML.

Area can be:

Area	HTMLCode is inserted...
rv_thms_Head	between <head> and </head>
rv_thms_BodyAttribute	between <body and >
rv_thms_Body	just after <body>
rv_thms_End	just before </body>

Example

```
procedure TMyForm.MyRichViewSaveHTMLExtra(
```

```
  Sender: TCustomRichView; Area: TRVHTMLSaveArea;
  CSSVersion: Boolean; var HTMLCode: TRVUnicodeString[694]);
begin
  case Area of
  rv_thms_Head:
    HTMLCode := '<script></script>';
  rv_thms_BodyAttribute:
    HTMLCode := 'alink=#ff0000';
  rv_thms_Body:
    HTMLCode := 'This document is generated by '+
    '<A href="http://www.trichview.com">RichView</A><BR>';
  end;
end;
```

Note: TRVHTMLSaveArea is defined in RVStyle.pas.

See also events:

- OnHTMLSaveImage[277];
- OnSaveImage2[296];
- OnSaveParaToHTML[298];
- OnSaveRTFExtra[299];
- OnSaveDocXExtra[293].

See also:

- Export to HTML[96].

1.5.1.1.4.28 TRichView .OnSaveImage2

Occurs when TRichView saves image to HTML file or stream.

```
type
  TRVSaveImageEvent2 = procedure (Sender: TCustomRichView;
    Graphic: TGraphic; SaveFormat: TRVSaveFormat[685];
    const Path, ImagePrefix: TRVUnicodeString[694];
    var ImageSaveNo: Integer; var Location: TRVUnicodeString[694];
    var DoDefault: Boolean) of object;

property OnSaveImage2: TRVSaveImageEvent2;
```

(introduced in version 1.8; "hidden" parameters are added in version 14; changed in version 18)

This event is alternative to OnHTMLSaveImage[277]. It is more easy to use, because the image is available in the Graphic parameter (in OnHTMLSaveImage, you need to extract image from the item or RVData yourself). Besides, this event has **ImagePrefix** and **ImageSaveNo** parameters, so you do not need to store them in global variables.

OnSaveImage2 is not called if:

- *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML[250]/SaveHTMLEx[251] and value of *rvespImageFileName* item property[676] is not empty;
- DoDefault parameter of OnHTMLSaveImage[277] was set to *False* for this image.

Input parameters:

Graphic – image to save.

SaveFormat is equal to *rvsfHTML*.

Path – destination directory of HTML file (or **Path** parameter of SaveHTMLToStream[252] /SaveHTMLToStreamEx[252])

ImagePrefix – recommended beginning of image file name (this is **ImagesPrefix** parameter of SaveHTMLToStream[252]/SaveHTMLToStreamEx[252])

ImageSaveNo – value that you can use to create unique file names (see GetNextFileName method in the description of OnHTMLSaveImage[277]).

"Hidden" input parameters:

Sender.Style.RVData (must be typecasted to TCustomRVData[650]) – document containing the item that is being saved to an image; it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor.

Sender.Style.ItemNo – index of this item inside **Sender.Style.RVData**. If this event is called for background image (document or cell), **ItemNo**=-1.

Output parameters:

Location – set this parameter to file name of image where you saved it. This string will be inserted in HTML file.

DoDefault – set to *False* if you saved this image yourself. Otherwise, the default image saving procedure will be called.

Note: when saving images from image-lists (*bullets*[128], *hotspots*[129], list markers[131] with images), TRichView creates a temporal TBitmap and passes it to this event.

See also events:

- OnHTMLSaveImage[277];
- OnSaveItemToFile[297].

See also:

- Export to HTML[96].

1.5.1.1.4.29 TRichView .OnSaveItemToFile

Allows you to change how document items are saved in text, RTF, DocX or HTML file or stream.

```
type
  TRVSaveItemToFileEvent = procedure (Sender: TCustomRichView;
    const Path: String; RVData: TCustomRVData[650]; ItemNo: Integer;
    SaveFormat: TRVSaveFormat[685]; var OutStr:TRVUnicodeString[694];
    var DoDefault: Boolean) of object;
```

(introduced in v1.8; changed in v18)

This is a "low level" event allowing to change saving of items completely.

There are events offering a subset of functionality of this event:

- OnHTMLSaveImage[277],
- OnSaveImage2[296],
- OnSaveComponentToFile[292].

These events are easier to use than this event.

Input parameters:

Path – path to output file;

RVData, **ItemNo** – item to save. **RVData** – document containing item that is being saved; it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**.

SaveFormat identifies file format, one of *rvsfText*, *rvsfHTML*, *rvsfRTF, rvsfDocX*.

OutStr:

- for non-text items: empty string;

- for text items: text that needs to be saved (it can be a part of the item's text when saving selected fragment).

DoDefault – *True.*

Output parameters:

Set values of **DoDefault** and **OutStr** according to the table below.

Output value of DoDefault	For Unicode text items	For non-text items
True	For text files: default saving, value of **OutStr** is ignored. For HTML, RTF, DocX: value of **OutStr** will be processed afterward (for example, '<' will be changed to '<' for HTML).	Default saving, value of **OutStr** is ignored.
False	**OutStr** will be inserted in the file without further processing (no control codes will be replaced).	

If you do not want to change saving of some item, leave **DoDefault** to *True*.

You cannot override some special processing for paragraph markers when saving to HTML and RTF.

Notes about DocX:

- this event is not called for tables
- this event allows changing saving of numbering sequences[134] only partially

Example: saving pictures as '[PIC]'

```
procedure TMyForm.MyRichViewSaveItemToFile(
  Sender: TCustomRichView; const Path: String;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var OutStr: TRVUnicodeString[694]; var DoDefault: Boolean);
begin
  if (RVData.GetItemStyle[229](ItemNo)=rvsPicture[711]) then
  begin
    OutStr := '[PIC]';
    DoDefault := False;
  end;
end;
```

1.5.1.1.4.30 TRichView .OnSaveParaToHTML

Allows to add your HTML code when saving paragraphs to HTML.

```
property OnSaveParaToHTML: TRVSaveParaToHTMLEvent

TRVSaveParaToHTMLEvent =
  procedure (Sender: TCustomRichView;
    RVData: TCustomRVData[650]; ItemNo: Integer;
    ParaStart, CSSVersion: Boolean;
    var HTMLCode: TRVUnicodeString[694]) of object;
```

(introduced in v1.8; changed in version 18)

Input parameters:

ParaStart:

- *True*, if the returned text will be inserted in the beginning of paragraph (before , or <li>),
- *False*, if it will be inserted in the end (after , or).

RVData, **ItemNo** define the location. if **ParaStart**=*True*, **ItemNo** is an index of the first paragraph item (in **RVData**). If *False*, it is an index of the first item of the next paragraph (or **RVData**.ItemCount[180] for the last paragraph).

RVData can be **Sender**.RVData[187], or table cell, or cell inplace editor's RVData[187].

Output parameters:

HTMLCode – output string.

Example: each paragraph in its own table

```
procedure TMyForm.MyRichViewSaveParaToHTML(Sender: TCustomRichView;
  RVData: TCustomRVData; ItemNo: Integer; ParaStart,
  CSSVersion: Boolean; var HTMLCode: TRVUnicodeString[694]);
begin
  if ParaStart then
    HTMLCode := '<table border=1 width=100%><tr><td>'
  else
    HTMLCode := '</td></tr></table>'
end;
```

This event is public, so you need to assign its handler in code at run time.

See also:

- OnSaveHTMLExtra[295];
- Export to HTML[96].

1.5.1.1.4.31 TRichView.OnSaveRTFExtra

Allows saving additional information in RTF

```
type
  TRVRTFSaveArea = // defined in RVStyle.pas
    (rv_rtfs_TextStyle, rv_rtfs_ParaStyle, rv_rtfs_ListStyle,
    rv_rtfs_StyleTemplate, rv_rtfs_StyleTemplateText,
    rv_rtfs_CellProps, rv_rtfs_RowProps, rv_rtfs_Doc);
 TRVSaveRTFExtraEvent = procedure (Sender: TCustomRichView;
   Area: TRVRTFSaveArea; Obj: TObject;
   Index1, Index2: Integer;
   InStyleSheet: Boolean; var RTFCode: TRVUnicodeString[694]) of object;

property OnSaveRTFExtra: TRVSaveRTFExtraEvent;
```

(changed in version 18)

This event occurs when saving RTF (SaveRTF[253], SaveRTFToStream[254]).

Text assigned to **RTFCode** parameter will be saved in RTF (converted to ANSI)

Area defines a place where **RTFCode** will be inserted. Meaning of **Index1**, **Index2** and **Obj** depends on **Area**.

Area	Meaning
rv_rtfs_TextStyle	Occurs when saving text style. **Index1** is an index of the text style in TextStyles[437] collection, **Obj** is a text style object[478]

	If **InStyleSheet**=*True*, this style is saved as a part of style sheet (when RTF is saved with *rvrtfSaveStyleSheet*[185] option)
rv_rtfs_ParaStyle	Occurs when saving paragraph style. **Index1** is an index of the paragraph style in ParaStyles[431] collection, **Obj** is a paragraph style object[482] If **InStyleSheet**=*True*, this style is saved as a part of style sheet (when RTF is saved with *rvrtfSaveStyleSheet*[185] option)
rv_rtfs_ListStyle	Occurs when saving list style and list levels. **Index1** is an index of the list style in ListStyles[429]. **Index2** equals to -1 when saving the list description, or it equals to the level index when saving list level. **Obj** is a list style object[493]
rv_rtfs_StyleTemplate	Occurs when saving a style template as a style sheet item. **Index1** is an index of the style template in StyleTemplates[435]. **Index2** = -1 **Obj** is a style template[494].
rv_rtfs_StyleTemplateText	Occurs when saving a style template as an additional style sheet item (containing only character properties), linked the the main style sheet item. Such additional style sheet items are created for style templates having Kind[495] = *rvstkParaText*. **Index1** is an index of the style template in StyleTemplates[435]. **Index2** = -1 **Obj** is a style template[494].
rv_rtfs_CellProps	Occurs when saving table cell properties. **Index1** = row index, **Index2** =column index, **Obj** is the table[565].
rv_rtfs_RowProps	Occurs when saving table row properties. **Index1** = row index, **Obj** is the table[565].
rv_rtfs_Doc	Allows saving additional information at the beginning of the document

Example

```
procedure TMyForm.MyRichViewSaveRTFExtra(
  Sender: TCustomRichView; Area: TRVRTFSaveArea;
  Obj: TObject; Index1, Index2: Integer;
  InStyleSheet: Boolean; var RTFCode: TRVAnsiString[668]);
begin
  if Area=rv_rtfs_Doc then
    RTFCode := '\paperw11906\paperh16838\margl567\'+
      'margr567\margt567\margb567';
end;
```

This example saves paper size = A4, and margins = 1 cm.

See also properties:

- DocParameters 176.

See also events:

- OnSaveDocXExtra 293;
- OnSaveHTMLExtra 295.

See also examples:

- ***http://www.trichview.com/forums/viewtopic.php?t=64***.

1.5.1.1.4.32 TRichView .OnSelect

Occurs when selection in the document was changed (not contents, but bounds)

```
property OnSelect:TNotifyEvent;
```

Changes in selection can occur when the user changes the selection using the mouse or keyboard, or when the selection is changed with the methods:

- SelectAll 257;
- Deselect 216;
- SetSelectionBounds 271;
- (and sometimes SelectionExists 258).

See also:

- Selection in RichView 81.

1.5.1.1.4.33 TRichView .OnSpellingCheck

Allows to mark misspelled words.

```
type
  TRVSpellingCheckEvent = procedure (Sender: TCustomRichView;
    const AWord: TRVUnicodeString 694; StyleNo: Integer;
    var Misspelled: Boolean) of object;

property OnSpellingCheck: TRVSpellingCheckEvent;
```

(introduced in v1.9; changed in version 18)

This event is called for each word in the document.

Input parameters:

AWord is a word to check (Unicode for Delphi/C++Builder 2009 or newer, ANSI for older versions of Delphi/C++Builder).

StyleNo is a style of text item 120 containing this word (index in the collection **Sender**.Style 193.TextStyles 437).

Output parameter:

Misspelled, must be set to True if this word is misspelled.

Example (using ***Addict3*** spelling checker):

```
procedure TMyForm.MyRichViewEditSpellingCheck(Sender: TCustomRichView;
  const AWord: TRVUnicodeString 694; StyleNo: Integer; var Misspelled: Boolean);
begin
  Misspelled := not MyRVAddictSpell3.WordAcceptable(AWord);
end;
```

This event is called in thread context.

See also:

- Live spelling check in RichView 101.

1.5.1.1.4.34 TRichView.OnStyleTemplatesChange

Occurs when the collection Style 193.StyleTemplates 435 was changed.

```
property OnStyleTemplatesChange: TNotifyEvent;
```

(introduced in version 14)

This event may be called only if UseStyleTemplates 195=*True*.

It is called:

- when loading or inserting RVF and RTF files or streams (and StyleTemplateInsertMode 193 <>*rvstimIgnoreSourceStyleTemplates*);
- (in editor) by ChangeStyleTemplates 338 method;
- by RichViewActions that can modify this collection (TrvActionNew, TrvActionOpen, TrvActionInsertHyperlink, and actions calling ChangeStyleTemplates 338: TrvActionStyleTemplates, TrvActionAddStyleTemplate).

1.5.1.1.4.35 TRichView.OnWriteHyperlink

Occurs when TRichView wants to save hypertext link to HTML, RTF or DocX file

```
type
  TRVWriteHyperlink = procedure (Sender: TCustomRichView;
     id: Integer; RVData: TCustomRVData 650; ItemNo: Integer;
     SaveFormat: TRVSaveFormat 685;
     var Target, Extras: TRVUnicodeString 694) of object;

property OnWriteHyperlink: TRVWriteHyperlink;
```

(introduced in v1.8; changed in version 18)

By default, items tags 68 are saved as targets of hypertext links in RTF, DocX and HTML.

But you can use this event to customize saving.

In order to do it, assign the target URL to the **Target** parameter.

Input parameters:

id – identifier of the hypertext link; this event can be called for unformatted documents; in this case, **id**=-1.

RVData – document containing hyperlink (it may be **Sender**.RVData 187, or table cell, or cell inplace editor's RVData 187)

ItemNo – index of the hyperlink-item inside **RVData**.

SaveFormat can be *rvsfHTML* or *rvsfRTF* or *rvsfDocX*

Output parameters:

Target – hyperlink target to save in a file or a stream. If you return an empty string, this item will not be saved as a hyperlink.

Extras – optional additional information for saving with the link.

Example 1:

This example does the same work as if the event is not assigned: saves the item tag 68 as a target.

```
procedure TMyForm.MyRichViewWriteHyperlink(
```

```
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString[694]);
begin
  Target := RVData.GetItemTag[230](ItemNo);
end;
```

Example 2:

The same, but in addition to saving the link target, this example saves the hint "Click here!" (both for RTF and HTML) and the attribute "target=_blank" for HTML.

```
procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString[694]);
begin
  Target := RVData.GetItemTag[230](ItemNo);
  case SaveFormat of
    rvsfHTML:
      Extras := 'target=_blank title="Click here!"';
    rvsfRTF:
      Extras := '\o "Click here!"';
  end;
end;
```

Saving hint is given here only as an example. TRichView saves hints stored in the *rvespHint* item's extra string property[676] automatically.

Example 3:

```
procedure TMyForm.MyRichViewWriteHyperlink(
  Sender: TCustomRichView; id: Integer;
  RVData: TCustomRVData; ItemNo: Integer;
  SaveFormat: TRVSaveFormat;
  var Target, Extras: TRVUnicodeString[694]);
begin
  case id-FirstJumpNo[179] of
    0: Target := 'http://www.borland.com';
    1: Target := 'http://www.inprise.com';
    2: Target := 'http://www.codegear.com';
    3: Target := 'http://www.embarcadero.com';
  end;
end;
```

See also events:

- OnReadHyperlink[284].

See also methods:

- SaveHTML[250];
- SaveHTMLEx[251];
- SaveRTF[253];
- (and methods saving RTF and HTML to streams).

See also:

- Hypertext[69];
- Saving to HTML[96].

1.5.1.1.5 Classes of Properties

1.5.1.1.5.1 TRVRTFReaderProperties

This is a type of TRichView[162].RTFReadProperties[187]. It contains properties for controlling RTF import in TRichView and TRichViewEdit. Do not create objects of this class yourself, use this property instead.

Unit RVRTFProps.

Syntax

```
TRVRTFReaderProperties = class (TPersistent)
```

Properties

The most important properties are:

- ParaStyleMode[308];
- TextStyleMode[309].

The following properties allow to ignore some part of RTF content:

- IgnoreBookmarks[306];
- IgnoreFields[307];
- IgnoreNotes[307];
- IgnorePictures[307];
- IgnoreSequences[307];
- IgnoreTables[307];
- SkipHiddenText[309];
- ReadDocParameters[308].

Reading Hyperlinks

By default, hyperlinks targets are loaded in items tags[68]. Use TCustomRichView.OnReadHyperlink[284] to customize loading.

Reading RTF Images

The following types of RTF pictures are supported (if IgnorePictures[307] = *False*):

- bitmaps (DIB and DDB), loaded as TBitmap;
- metafiles (WMF and EMF), loaded as TMetafile; if metafile contains only a bitmap, it can be loaded as TBitmap, see ExtractMetafileBitmaps[306];
- Jpegs, loaded as TJPEGImage;
- PNG, if PNG class is specified (for Delphi 2009+, TPngImage is used by default; you can specify a PNG class using RVGraphicHandler[710].RegisterPngGraphic);
- external images, loaded from files (TCustomRichView.OnImportPicture[279] event occurs).

Hides grid lines (dotted lines in place of hidden borders) for tables imported from RTF.

```
property AutoHideTableGridLines: Boolean;
```

(introduced in version 1.6)

If this property is set to *True*, RichView includes *rvtoHideGridLines* in Options[577] of tables read from RTF.

Note: it's not necessary to set this option for individual tables. You can hide and show grid for all tables using rvoShowGridLines option in TRichView.Options[182].

Default value:

False

Specifies whether to add base path (path to RTF file) to hyperlink targets.

```
property BasePathLinks: Boolean;
```

(introduced in version 10)

Set to *False* if you do not want to add the document path to relative hyperlinks on RTF loading.

For example, when loading "c:\Docs\MyFile.rtf' containing link to "MyFile2.rtf":

- if this property is *False*, TCustomRichView.OnReadHyperlink 284 event occurs with Target='MyFile2.rtf' (or, if the event is not assigned, this value is stored in them item tag 68);
- if this property is *True*, this event occurs with Target='c:\Docs\MyFile2.rtf' (or, if the event is not assigned, this value is stored in them item tag).

In ***RVHTMLImporter***, this property corresponds to *rvhtmloBasePathLinks* in RVHTMLImporter.Options.

Default value:

True

Specifies a character set 468 for Unicode text read from RTF.

```
property CharsetForUnicode: TFontCharset;
```

(introduced in version 11)

This property is used if UseCharsetForUnicode 310=*True* and TextStyleMode 309<>*rvrsUseSpecified*.

This character set is applied to Unicode text read from RTF.

Using a single specified character set for all text may be useful if you want to avoid words consisting of several text items (having the same font but different character sets).

Default value:

DEFAULT_CHARSET

Specifies the mode of import of text highlighting.

```
type
  TRVRTFHighlight = // defined in RVScroll.pas
    (rtfhlIgnore, rtfhlFixedColors, rtfhlColorTable);

property ConvertHighlight: TRVRTFHighlight;
```

(introduced in version 1.7)

RTF editors show text highlighting as a text background color. But this is not a real text background color (which is always imported by TRichView), this is not even a RTF text attribute.

 – this is a MS Word toolbar button applying text highlighting.

Mode	Meaning
rtfhlIgnore	Highlighting is ignored.
rtfhlFixedColors	Highlighting is converted to text background using a fixed set of colors (according to the old RTF specification)
rtfhlColorTable	Highlighting is converted to text background using RTF color table.

MS Word (and TRichView) always puts 16 default colors to the beginning of RTF color table, so for such documents, both *rtfhlFixedColors* and *rtfhlColorTable* work absolutely identically. TRichView itself never saves text background color as a highlight, it uses the special RTF keyword for text background.

It's not recommended to change value of this property.

Default value:

rtfhlColorTable

Contains error code for the last RTF reading operation.

```
ErrorCode: TRVRTFErrorCode;
```

```
type // defined in RVRTFErr unit.
  TRVRTFErrorCode = (
    rtf_ec_OK,             // No error
    rtf_ec_StackUnderflow, // Unmatched '}'
    rtf_ec_StackOverflow,  // Too many '{' -- memory exhausted
    rtf_ec_UnmatchedBrace, // RTF ended during an open group.
    rtf_ec_InvalidHex,     // Invalid hex character found in data
    rtf_ec_BadTable,       // RTF table invalid (internal parser error)
    rtf_ec_Assertion,      // Assertion failure (internal parser error)
    rtf_ec_EndOfFile,      // End of file reached while reading RTF
    rtf_ec_FileOpenError,  // Parser cannot open input file
    rtf_ec_Exception,      // Exception (internal parser error)
    rtf_ec_InvalidPicture, // Invalid picture in RTF
    rtf_ec_Aborted         // (internal parser error)
    );
```

Option for reading metafiles from RTF

```
property ExtractMetafileBitmaps: Boolean;
```

(introduced in version 1.9)

MS Word usually does not save bitmaps in RTF as they are, it wraps them in metafiles.

If this property is set to *False*, such bitmaps will be loaded as TMetafile. TMetafile sometimes scales bitmap images inaccurately.

If this property is set to *True*, TRichView attempts to extract bitmaps embedded in metafiles and to use them instead of metafiles. If successful, they are inserted as TBitmap.

This option does not affect RTF metafiles containing something besides bitmaps, no bitmaps, or multiple bitmaps – they will be loaded as TMetafile.

Default value:

True

Disallows loading bookmarks from RTF.

```
property IgnoreBookmarks: Boolean;
```

(introduced in version 10)

Bookmarks starts are read as *checkpoints* 65. Bookmark ends cannot be read.

Default value:

False

Disallows loading "page number" [141] and "page count" [142] fields from RTF.

```
property IgnoreFields: Boolean;
```

(introduced in version 15)

If *True*, fields are loaded as a plain text.

Default value:

False

Disallows loading footnotes [136], endnotes [135] and references to them [140] from RTF.

```
property IgnoreNotes: Boolean;
```

(introduced in version 10)

If loading of notes is allowed, their properties are also loaded, specifically Style [193].FootNotePageReset [425].

Default value:

False

Disallows loading pictures from RTF.

```
property IgnorePictures: Boolean;
```

Set it to *True* to import RTF without pictures.

TRichView supports RTF pictures in the following formats:

- Windows Bitmaps (DIB and DDB), loaded as TBitmap;
- Metafiles, loaded as TMetafile or as TBitmap (see ExtractMetafileBitmaps [306] property);
- Enhanced Metafiles, loaded as TMetafile,
- JPEG (D3+), loaded as TJpegImage,
- PNG, see RVGraphicHandler [710].RegisterPngGraphic.

Default value:

False

Disallows loading numbering sequences [134] from RTF.

```
property IgnoreSequences: Boolean;
```

(introduced in version 10)

If this property is *True*, RTF numbered sequences are loaded as a plain text.

Default value:

False

Disallow loading tables from RTF.

```
property IgnoreTables: Boolean;
```

(introduced in version 10)

If *True*, contents of tables are loaded as normal text.

Default value:

False

If set to *True*, all line breaks (added with Shift + Enter) in RTF will be treated as paragraph starts.

```
property LineBreaksAsParagraphs: Boolean;
```

(introduced in version 1.7)

Default value:

False

Defines mode of reading of paragraph properties from RTF.

```
property ParaStyleMode: TRVReaderStyleMode;
```

This property defines how RTF paragraph attributes will be converted to TRichView.Style.ParaStyles.

Mode	Meaning
rvrsUseSpecified	All paragraph formatting in RTF will be ignored, all paragraphs will have style specified in ParaStyleNo property.
rvrsUseClosest	RTF formatting will be mapped to the the most similar existing paragraph style
rvrsAddIfNeeded	This mode provides the most closer look to the original RTF. RichView will try to use existing styles as much as possible, but if style with the desired formatting does not exist, RichView will add a new one to the end of the collection of paragraph styles and use it.

List markers (bullets and numbering) are imported only if ParaStyleMode=*rvrsAddIfNeeded*.

If the both TextStyleMode=**ParaStyleMode**=*rvrsAddIfNeeded*, and TRichView.UseStyleTemplates=*True*, and StyleTemplateInsertMode<>*rvstimIgnoreSourceStyleTemplates,* the RTF style sheet will be read in TRichView.Style.StyleTemplates. For RTF loading, the RTF style sheet replaces existing style templates. For RTF insertion, the RTF style sheet is merged into the existing style templates. In this case, RTF paragraph properties are adjusted according to TRichView.StyleTemplateInsertMode.

Default value:

- *rvrsUseClosest* for components created at runtime.
- For components placed on form at designtime, initial value of this property is defined in the component editor for TRichView. By default, the component editor applies "allow adding styles dynamically" mode to new TRichView components, and this property is set to *rvrsAddIfNeeded*.

See also:

- ParaStyleNo;
- TextStyleMode.

Default paragraph style for RTF loading.

```
property ParaStyleNo: Integer
```

This property is used if ParaStyleMode=*rvrsUseSpecified*.

It is an index in the collection of paragraph styles.

Default value:

0

See also:

- ParaStyleMode;
- TextStyleNo.

Set to True to allow reading DocParameters from RTF.

```
property ReadDocParameters: Boolean;
```

(introduced in version 10)

Default value:

False

This property controls how hidden text is loaded from RTF

```
property SkipHiddenText: Boolean;
```

(introduced in version 1.6)

If *True*, hidden content is not added to RichView.

If *False*, hidden content [76] is added.

Default value:

True

Defines a mode for extraction of character formatting from RTF

```
property TextStyleMode: TRVReaderStyleMode [685];
```

This property defines how RTF text attributes will be converted to TRichView.Style [193].TextStyles [437].

Mode	Meaning
rvrsUseSpecified	All character formatting in RTF will be ignored, all text will have style specified in TextStyleNo [309] property.
rvrsUseClosest	RTF formatting will be mapped to the the most similar existing text style [437].
rvrsAddIfNeeded	This mode provides the most closer look to the original RTF. RichView will try to use existing styles as much as possible, but if style with the desired formatting does not exist, RichView will add a new one to the end of the collection of text styles [437] and use it.

If the both **TextStyMode**=ParaStyleMode [308]=*rvrsAddIfNeeded*, and TRichView.UseStyleTemplates [195]=*True*, and StyleTemplateInsertMode [193]<>*rvstimIgnoreSourceStyleTemplates,* the RTF style sheet will be read in TRichView.Style [193].StyleTemplates [435]. For RTF loading, the RTF style sheet replaces existing style templates. For RTF insertion, the RTF style sheet is merged into the existing style templates. In this case, RTF character properties are adjusted according to TRichView.StyleTemplateInsertMode [193].

Default value:

- *rvrsUseClosest* for components created at runtime.
- For components placed on form at designtime, initial value of this property is defined in the component editor [168] for TRichView. By default, the component editor applies "allow adding styles dynamically" mode to new TRichView components, and this property is set to *rvrsAddIfNeeded*.

See also:

- TextStyleNo [309];
- ParaStyleMode [308].

Default text style for RTF loading.

```
property TextStyleNo: Integer
```

This property is used if TextStyleMode [309]=*rvrsUseSpecified*.

It is an index in the collection of text styles [437].

Default value:

0

See also:

- TextStyleMode[309];
- ParaStyleNo[308].

Obsolete property, it does nothing

```
property UnicodeMode: TRVReaderUnicode;

type
  TRVReaderUnicode =
    (rvruMixed, rvruNoUnicode, rvruOnlyUnicode);
```

Specifies whether CharsetForUnicode[305] property is used as a character set for Unicode text read from RTF.

```
property UseCharsetForUnicode: Boolean;
```

Default value:

False

Specifies whether

```
property UseCharsetForUnicode: Boolean
```

Set this property to *True* to allow using existing hypertext text styles for non-hypertext text from RTF (in TextStyleMode[309] = *rvrsUseClosest* or *rvrsAddIfNeeded*).

Note: it's not recommended to change value of this property.

Default value:

False

See also:

- Hypertext in RichView[69].

Set it to *True* if you want to use the same values for CellHPadding[573] and CellVPadding[574] for loaded tables.

```
property UseSingleCellPadding: Boolean;
```

(introduced in version 10)

It may be useful if you want to maintain compatibility with older versions of your software, because applications compiled with older versions of TRichView cannot read RVF[93] files containing tables with different cell padding).

Default value:

False

1.5.1.1.5.2 TRVDocParameters

This is a type of TRichView[162].DocParameters[176]. It contains document's and page properties that can be written to or read from RVF[93] (RichView Format) and RTF (Rich Text Format). The components do not use these properties, but they can be assigned to TRVPrint[513] using the method AssignDocParameters[522].

Unit RVDocParams.

Syntax

```
TRVDocParameters = class(TPersistent)
```

(introduced in version 10)

Properties

This class has the following properties:

- Units 314 – measurement units for margins, paper size, header and footer positions;
- PageWidth 313, PageHeight 313 – page size;
- Orientation 313 – page orientation;
- LeftMargin 312, RightMargin 313, TopMargin 314, BottomMargin 311 – margins;
- MirrorMargins 313 allow to exchange left and right margins on even pages;
- HeaderY 312, FooterY 312 – position of header and footer;
- TitlePage 314 shows/hides header and footer for the first page;
- FacingPages 312 shows/hides header and footer for even pages;
- ZoomMode 314 – zooming mode;
- ZoomPercent 315 – zooming percent.
- Author 311, Title 314, Comments 311 – text properties.

Methods

This class has the following methods:

- Reset 315 resets all properties to default values;
- ResetLayout 315 resets all sizes to default values measured in Units 314;
- UnitsPerInch 315 allows to convert between measurement units;
- ConvertToUnits 315 converts all sizes to the specified units.

Author of the document

```
property Author: TRVUnicodeString 694;
```

(introduced in version 13; changed in version 18)

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd 370.

Default value:

" (empty string)

Bottom margin, measured in Units 314.

```
property BottomMargin: TRVLength 682;
```

This property is not used by TRichView components, but it can be written and read from RVF 93 (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters 522 assigns this property to TRVPrint.Margins 520.Bottom.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd 370.

Default value:

1.0

Comments to the document

```
property Comments: TRVUnicodeString 694;
```

(introduced in version 13; changed in version 18)

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd 370.

Default value:

" (empty string)

Instructs to use different headers and footers for odd and even pages.

```
property FacingPages: Boolean;
```

This property is not used by TRichView components, but it can be written and read from RVF 93 (RichView Format), RTF (Rich Text Format) and DocX files.

TRVPrint.AssignFromDocParameters 522 assigns this property to TRVPrint.FacingPages 518.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd 370.

Headers and footers are assigned to TRichView using SetHeader 264 and SetFooter 263 methods.

Default value:

False

See also:

- TitlePage 314

Distance from the bottom of page to the bottom of footer, measured in Units 314.

```
property FooterY: TRVLength 682;
```

This property is not used by TRichView components, but it can be written and read from RVF 93 (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters 522 assigns this property to TRVPrint.FooterY 519.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd 370.

Default value:

0.5

Distance from the top of page to the top of header, measured in Units 314.

```
property HeaderY: TRVLength 682;
```

This property is not used by TRichView components, but it can be written and read from RVF 93 (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters 522 assigns this property to TRVPrint.HeaderY 519.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd 370.

Default value:

0.5

Left margin, measured in Units 314.

```
property LeftMargin: TRVLength 682;
```

This property is not used by TRichView components, but it can be written and read from RVF 93 (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters 522 assigns this property to TRVPrint.Margins 520.Left.

In TRichViewEdit 316, you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd 370.

Default value:

1.25

Instructs to swap left and right margins on even pages.

```
property MirrorMargins: Boolean;
```

This property is not used by TRichView components, but it can be written and read from RVF [93] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters [522] assigns this property to TRVPrint.MirrorMargins [521].

In TRichViewEdit [316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd [370].

Default value:

False

Page orientation.

```
property Orientation: TPrinterOrientation;
```

This property is not used by TRichView components, but it can be written and read from RVF [93] (RichView Format) and RTF (Rich Text Format) files.

In TRichViewEdit [316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd [370].

Default value:

poPortrait

Page height, measured in Units [314].

```
property PageHeight: TRVLength [682];
```

This property is not used by TRichView components, but it can be written and read from RVF [93] (RichView Format) and RTF (Rich Text Format) files.

Default value:

11.692913386 (A4 height in inches)

Page width, measured in Units [314].

```
property PageWidth: TRVLength [682];
```

This property is not used by TRichView components, but it can be written and read from RVF [93] (RichView Format) and RTF (Rich Text Format) files.

Default value:

8.2677165354 (A4 width in inches)

Right margin, measured in Units [314].

```
property RightMargin: TRVLength [682];
```

This property is not used by TRichView components, but it can be written and read from RVF [93] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters [522] assigns this property to TRVPrint.Margins [520].Right.

In TRichViewEdit [316], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd [370].

Default value:

1.25

Title of the document

```
property Title: TRVUnicodeString[694];
```

(introduced in version 13; change version 18)

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetStrPropertyEd[370].

Default value:

" (empty string)

Instructs to use a special header and footer on the first page.

```
property FacingPages: Boolean;
```

This property is not used by TRichView components, but it can be written and read from RVF[93] (RichView Format), RTF (Rich Text Format) and DocX files. It also affects saving headers and footers in HTML files.

TRVPrint.AssignFromDocParameters[522] assigns this property to TRVPrint.TitlePage[521].

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

Headers and footers are assigned to TRichView using SetHeader[264] and SetFooter[263] methods.

Default value:

False

See also:

- FacingPages[312]

Top margin, measured in Units[314].

```
property TopMargin: TRVLength[682];
```

This property is not used by TRichView components, but it can be written and read from RVF[93] (RichView Format) and RTF (Rich Text Format) files.

TRVPrint.AssignFromDocParameters[522] assigns this property to TRVPrint.Margins[520].Top.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetFloatPropertyEd[370].

Default value:

1.0

Measurement units for all sizes in TRVDocParameters.

```
property Units: TRVUnits[694];
```

Assignment to this property does not change other properties. Use ConvertToUnits[315] to convert all sizes to new units.

Use UnitsPerInch[315] to convert values between units.

In TRichViewEdit[316], you can change this property as an editing operation, see TRichViewEdit.SetIntPropertyEd[370].

Default value

rvuInches

Scaling mode

```
type // defined in RVStyle unit
  TRVZoomMode = (rvzmFullPage, rvzmPageWidth, rvzmCustom);
```

```
property ZoomMode : TRVZoomMode;
```

This property is not used by TRichView components, but it can be written and read from RVF[93] (RichView Format) and RTF (Rich Text Format) files.

Default value:

rvzmCustom

View scale, % (if ZoomMode[314]=*rvzmCustom*).

```
property ZoomPercent: Integer;
```

This property is not used by TRichView components, but it can be written and read from RVF[93] (RichView Format) and RTF (Rich Text Format) files.

Default value:

100

Converts all sizes to **AUnits**.

```
procedure ConvertToUnits(AUnits: TRVUnits[314]);
```

This method converts all size properties from Units[314] to **AUnits**, then assigns Units[314] = **AUnits**.

The following properties are converted:

- PageWidth[313], PageHeight[313] – page size;
- LeftMargin[312], RightMargin[313], TopMargin[314], BottomMargin[311] – margins;
- HeaderY[312], FooterY[312] – position of header and footer.

Resets all properties to default values.

```
procedure Reset;
```

Resets all sizes to default values measured in Units[314].

```
procedure ResetLayout;
```

It makes sense to call this method on "File | New" command.

The following properties are reset:

- PageWidth[313], PageHeight[313] – page size;
- LeftMargin[312], RightMargin[313], TopMargin[314], BottomMargin[311] – margins;
- HeaderY[312], FooterY[312] – position of header and footer.

In addition, this method resets Orientation[313] to *poPortrait* and MirrorMargins[313] to *False*.

Returns the number of **Units** in one inch.

```
function UnitsPerInch(Units: TRVUnits[314]): TRVLength[682];
```

This method can be used to convert between measurement units.

For example, this code converts from points to millimeters:

```
ValueMM := ValuePoints * UnitsPerInch(rvuMillimeters) / UnitsPerInch(rvuPoints);
```

See the note about *rvuPixels* in the topic about Units[314].

1.5.1.1.6 Examples

1.5.1.1.6.1 Loading UTF-8 files

This example shows how to load Unicode UTF-8 file:

```
procedure LoadUTF8(rv: TCustomRichView[162]; const FileName: String;
  StyleNo, ParaNo: Integer);
var Stream: TFileStream;
    s: TRVRawByteString[685];
    ws: TRVUnicodeString[694];
begin
  Stream := TFileStream.Create(FileName, fmOpenRead);
  SetLength(s, Stream.Size);
  Stream.ReadBuffer(PRVAnsiChar(s)^, Stream.Size);
  Stream.Free;
  rv.Clear;
  ws := UTF8Decode(s);
  rv.AddTextNLW[206](ws, StyleNo, ParaNo, ParaNo, False);
end;
```

Call:

```
LoadUTF8(RichViewEdit1, 'test.txt', 0, 0);
RichViewEdit1.Format[217];
```

See also:

- Unicode in TRichView[99].

1.5.1.2 TRichViewEdit

TRichViewEdit is a control for editing documents with pictures, tables, Delphi controls and hyperlinks.

Unit RVEdit

Syntax

```
TCustomRichViewEdit = class(TCustomRichView[162])
TRichViewEdit = class(TCustomRichViewEdit)
```

RichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style[193] property.

The most important property settings can be done in the component editor[168] for TRichView. In Delphi (C++Builder) IDE, right click the RichViewEdit object on the form, choose "Settings" in the context menu.

Some abilities of TRichView are disabled in TRichViewEdit, such as:

- OnCheckpointVisible[274] event;
- all features related to hypertext ids:OnRVMouseMove[290], OnJump[282], GetJumpPointLocation[231] work only when user presses and holds Ctrl key, if *rvtoCtrlJumps* in EditorOptions[325] (or if ReadOnly[328]=*True*).

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[544]
TCustomRichView[162]
TCustomRichViewEdit

List of properties, events and methods

grouped in several categories (some entries can be in several groups)

- Properties 317
- Events 317
- General methods 317
- Inserting items in the position of caret 317
- Clipboard operations 318
- Advanced methods for modifying items 318
- Information about item at the position of caret 318
- Modifying item at the position of caret 319
- Style templates 319
- Undo/Redo 319
- Unicode 320
- Live spelling check 320
- "Smart Popups" 320
- Related to tables and special items 320
- Working with 320 *checkpoints* 320

Properties

- CurItemNo 322 – index of item in the position of caret, OffsetInCurItem 328 – offset of caret in this item;
- CurItemStyle 323 – style (or type) of the item at the position of caret;
- CurParaStyleNo 323 – current paragraph style index;
- CurTextStyleNo 324 – current text style index, see also ActualCurTextStyleNo 322;
- EditorOptions 325 – options for editing;
- ReadOnly 328 disables/enables editing;
- Modified 327 – all editing methods set it to *True*.
- CustomCaretInterval 324 – a blinking interval for a custom caret.
- AcceptDragDropFormats, AcceptPasteFormats 321 – lists of formats that the editor can accept.
- DefaultPictureVAlign 325 – alignment for pasted and dropped pictures.
- ForceFieldHighlight 327 affects highlighting of label items 133.

See also properties for undo/redo.

Events

- OnChange 390 occurs when the document is changed;
- OnCheckStickingItems 391 allows to add additional text protection;
- OnCurParaStyleChanged 391 occurs when the index of the current paragraph style is changed;
- OnCurTextStyleChanged 391 occurs when the index of the current text style is changed;
- OnPaste 399 allows pasting data from the Clipboard in custom formats;
- OnStyleConversion 400 occurs while executing ApplyStyleConversion 333;
- OnParaStyleConversion 397 occurs while executing ApplyParaStyleConversion 332;
- OnCaretGetOut 389 occurs when the user moves the caret "outside" the editor;
- OnCaretMove 390 occurs when the caret is moved;
- OnItemTextEdit 394 occurs when a text item 120's text is changed as a result of editing operation;
- OnSmartPopupClick 400 occurs when user clicks a "smart popup" 320 button.
- OnMeasureCustomCaret 394 and OnDrawCustomCaret 391 allow drawing your own caret instead of the system one.
- OnOleDragEnter 395, OnOleDragOver 396, OnOleDragLeave 395, OnOleDrop 397, OnBeforeOleDrop and OnAferOleDrop 388 occur on OLE drag&drop.

General Methods

- ApplyParaStyle 331, ApplyTextStyle 334 apply paragraph and text style to the selection;
- ApplyStyleConversion 333 applies custom style conversion to the selected text;
- ApplyParaStyleConversion 332 applies custom paragraph style conversion to the selected paragraphs;
- Change 337 generates OnChange 390 event;
- BeginUpdate 336, EndUpdate 341 disable and enable redrawing;
- SearchText, -A, -W 369 search for substring.

Inserting Items in the Position of Caret (Insert*** methods)

- InsertText, -A, -W 360, InsertStringTag, InsertStringATag, InsertStringWTag 359 insert text;

- InsertBreak[350] inserts *break* (horizontal line);
- InsertBullet[351] inserts *bullet*;
- InsertControl[352] inserts Delphi control;
- InsertHotspot[353] inserts *hotspot*;
- InsertPicture[355] inserts picture;
- InsertHotPicture[352] inserts picture-hypertext link;
- InsertTab[359] inserts tabulator;
- InsertRVFFromFileEd[357], InsertRVFFromStreamEd[358] insert RVF (RichView Format) from file or stream;
- InsertRTFFromFileEd[356], InsertRTFFromStreamEd[357] insert RTF (Rich Text Format) from file or stream;
- InsertTextFromFile[361],InsertOEMTextFromFile[354],InsertTextFromFileW[361] insert text from file;
- InsertItem[353] – general method for inserting items, inserts one item (usually used for tables).

Clipboard Operations

Main methods and events:

- CanPaste[337] – "can something be pasted from the Clipboard?";
- CanPasteRVF[337] – "can RVF (RichView Format) be pasted from the Clipboard?";
- CanPasteRTF[337] – "can RTF (Rich Text Format) be pasted from the Clipboard?";
- CutDef[340] cuts selection to the Clipboard;
- Paste[361] pastes from the Clipboard;
- PasteBitmap[362] pastes bitmap from the Clipboard;
- PasteMetafile[363] pastes metafile from the Clipboard;
- PasteGraphicFile[362] pastes metafile from the Clipboard;
- PasteRVF[364] pastes RVF (RichView Format) from the Clipboard;
- PasteRTF[363] pastes RTF (Rich Text Format) from the Clipboard;
- PasteTextA[364] pastes ANSI text from the Clipboard;
- PasteTextW[364] pastes Unicode text from the Clipboard;
- event OnPaste[399] allows to paste data from the Clipboard in custom formats.

Also:

- DeleteSelection[340] deletes selection;
- SelectCurrentWord[370] selects word at the position of caret;
- SelectCurrentLine[369] selects the line containing caret.

Advanced Methods for Modifying Items (Set***InfoEd methods)

Main methods:

- SetBreakInfoEd[371] – for changing *break*;
- SetBulletInfoEd[372] – for changing *bullet*;
- SetControlInfoEd[374] – for changing inserted control;
- SetHotspotInfoEd[382] – for changing *hotspot*;
- SetPictureInfoEd[386] – for changing picture or *hot picture*.

Also:

- SetItemExtraIntProperty[Ex]Ed[383], SetItemExtraStrProperty[Ex]Ed[384] – for changing additional properties of items;
- SetItemTagEd[384] – for changing *tag*;
- SetItemVAlignEd[386] – for changing item position relative to the line;
- SetItemTextEd[385] changes text (name) of item;
- SetCheckpointInfoEd[373] – for changing *checkpoint*;
- RemoveCheckpointEd[366] – for deleting *checkpoint*;
- ResizeControl[368] – for resizing inserted control;
- AdjustControlPlacement[330] returns control to the proper position;
- AdjustControlPlacement2[330] returns control to the proper position.

Information about the Item at the Position of Caret (GetCurrent***Info methods)

These methods are equivalent to Get***Info(CurItemNo[322],...) but work also in inplace editors of tables[145].

Main methods:

- GetCurrentTextInfo 350 – about text item;
- GetCurrentBreakInfo 341 – about *break*;
- GetCurrentBulletInfo 342 – about *bullet*;
- GetCurrentControlInfo 343 – about inserted control;
- GetCurrentHotspotInfo 344 – about *hotspot*;
- GetCurrentPictureInfo 349 – about picture or *hot picture*;
- GetCurrentItem 345 – as an object about any item (usually for tables 145);
- GetCurrentItemEx 345.

Also:

- GetCurrentItemExtraIntProperty[Ex] 346, GetCurrentItemExtraStrProperty[Ex] 346 return value of additional item property;
- GetCurrentTag 349 returns item's *tag*;
- GetCurrentItemVAlign 348 returns item position relative to the line;
- GetCurrentCheckpoint 343 returns item's *checkpoint*;
- GetCurrentItemText 347 returns text/name of item;
- GetCurrentLineCol 348 returns line and column at the caret position.

Modifying Item in Position of Caret (SetCurrent***Info methods)

These methods are equivalent to Set***InfoEd(CurItemNo 322,...) but work also in inplace editors of tables 145.

Main methods:

- SetCurrentBreakInfo 375 modifies *break*;
- SetCurrentBulletInfo 375 modifies *bullet*;
- SetCurrentHotspotInfo 377 modifies *hotspot*;
- SetCurrentControlInfo 377 modifies inserted control;
- SetCurrentPictureInfo 381 modifies picture or *hot picture*.

Also:

- SetCurrentItemExtraIntProperty[Ex] 378,
- SetCurrentItemExtraStrProperty[Ex] 384 change value of additional item property;
- SetCurrentTag 382 modifies item's *tag*;
- SetCurrentItemVAlign 380 changes item position relative to the line;
- SetCurrentItemText 380 changes text (name) of item;
- SetCurrentCheckpointInfo 376 modifies *checkpoint*;
- RemoveCurrentCheckpoint 367 deletes *checkpoint*;
- ResizeCurrentControl 368 returns control to the proper position in document;

- RemoveCurrentPageBreak 367 removes "explicit page break before item" flag;
- InsertPageBreak 355 sets "explicit page break before item" flag; can split current item into two parts.

Style templates

Methods:

- ApplyStyleTemplate 333, ApplyTextStyleTemplate 334, ApplyParaStyleTemplate 332 apply the specified style template to the selection;
- ChangeStyleTemplates 338 allows changing the collection of style templates as an editing operation.

Undo/Redo

Main:

- property UndoLimit 329 sets capacity of undo buffer;
- UndoAction 388 returns which operation will be undone next;
- Undo 387 undoes the last operation;
- UndoName 388 returns name of custom undo operation which will be undone next;
- RedoAction 366 returns which operation will be redone next;
- Redo 365 redoes the last undone operation;

- RedoName[366] returns name of custom redo operation which will be redone next;
- ClearUndo[339] clears undo and redo buffers.

Also:

- BeginUndoGroup[336],
- BeginUndoCustomGroup[336],
- SetUndoGroupMode[387].

Unicode

- PasteTextW[364] pastes Unicode text from the Clipboard;
- InsertTextW[360] inserts Unicode text;
- InsertTextFromFileW[361] inserts Unicode text from file.

Live Spelling Check

- GetCurrentMisspelling[348] returns misspelled word at the caret position;
- property LiveSpellingMode[327] controls when live spelling thread starts.

"Smart Popups"

"Smart popups" are buttons that can be used to set properties for the current (at the position of caret) item.

Properties:

- SmartPopupProperties[329] – popup button properties;
- SmartPopupVisible[329] – popup button properties.

Events:

- OnSmartPopupClick[400] occurs when user clicks "smart popup" button.

Related to Tables[565] and Special Items

- InsertItem[353] – general method for inserting items, inserts one item (usually used for tables[130], labels[133], sequences[134], footnotes[136], endnotes[135], references to notes[140]).
- GetCurrentItem[345] returns item at the position of caret as an object (usually used for tables[130], labels[133], sequences[134], footnotes[136], endnotes[135], references to notes[140]);
- BeginItemModify[335], EndItemModify[341] are used to reformat editor after operations on item (usually on tables);
- CanChange[336] checks possibility of performing editing operation (usually it is called before making operations on table).

Working with *Checkpoints*[65]

Editing-style analogs of methods of TRichView[162]:

- SetCheckpointInfoEd[373] – adds *checkpoint* for the specified item;
- RemoveCheckpointEd[366] – deletes *checkpoint* for the specified item.

Methods working with the item at the position of caret:

- GetCurrentCheckpoint[343] – returns *checkpoint* associated with the item at the caret position;
- SetCurrentCheckpointInfo[376] – adds/modifies *checkpoint* for the item at the caret position;
- RemoveCurrentCheckpoint[367] – deletes *checkpoint* for the item at the caret position.

Methods working with the checkpoint exactly at the position of caret:

- InsertCheckpoint[351] – inserts *checkpoint* in the caret position;
- GetCheckpointAtCaret[341] – returns the *checkpoint* exactly at the caret position;
- RemoveCheckpointAtCaret[366] – deletes the *checkpoint* exactly at the caret position.

1.5.1.2.1 Properties

1.5.1.2.1.1 TRichView Edit.AcceptDragDropFormats, AcceptPasteFormats

AcceptDragDropFormats lists formats that can be accepted when dropping data in TRichViewEdit.

AcceptPasteFormats lists formats that can be accepted when pasting in TRichViewEdit.

```
property AcceptDragDropFormats: TRVDragDropFormats;
property AcceptPasteFormats: TRVDragDropFormats;

type
  TRVDragDropFormat = (
    rvddRVF, rvddRTF, rvddText, rvddUnicodeText,
    rvddBitmap, rvddMetafile, rvddURL, rvddFiles, rvddHTML);
  TRVDragDropFormats = set of TRVDragDropFormat;
```

(introduced in v1.8 & v14)

Value	D&D Format	Meaning
rvddRVF	'RichView Format'	RichView Format 93
rvddRTF	'Rich Text Format'	RTF
rvddText	CF_TEXT	Plain ANSI text
rvddUnicodeText	CF_UNICODETEXT	Plain Unicode text
rvddBitmap	CF_DIB or CF_BITMAP	Bitmap
rvddMetafile	CF_ENHMETAFILE	Metafile
rvddURL	'UniformResourceLocator' or 'UniformResourceLocatorW'	URL. If available, URL caption is read from CFSTR_FILEDESCRIPTOR format (otherwise caption = target). OnReadHyperlink 284 event occurs with DocFormat=*rvlfURL*. If this event is not processed, URL is inserted as a plain text.
rvddFiles	CF_HDROP	By default, RichViewEdit inserts: ▪ graphic files, ▪ *.RTF (only drag&drop), ▪ *.RVF (only drag&drop), ▪ *.TXT (only drag&drop). You can modify the default processing in OnDropFiles 393 event (only drag&drop).
rvddHTML	'HTML Format'	This format is not used by TRichViewEdit itself, but can be used by RichViewActions (if TRVAControlPanel.HTMLComponent is assigned).

Example (processing OnReadHyperlink 284 with *rvddURL* support)

```
// OnReadHyperlink
procedure TMyForm.MyRichViewEditReadHyperlink(Sender: TCustomRichView;
  const Target, Extras: TRVUnicodeString 694; DocFormat: TRVLoadFormat;
  var StyleNo: Integer; var ItemTag: TRVTag 692; var ItemName: TRVUnicodeString 694);
```

```
begin
  if DocFormat=rvlfURL then
    StyleNo := GetHyperlinkStyleNo(Sender);
  ItemTag := Target;
end;

// This function returns the index of a text style having all properties
// like the current style, but hypertext, blue, underlined.
// If this style does not exist, it is added.
function TMyForm.GetHyperlinkStyleNo(rve: TCustomRichViewEdit): Integer;
var TextStyle: TFontInfo[478];
begin
  Result := rve.CurTextStyleNo[324];
  TextStyle := TFontInfo.Create(nil);
  try
    TextStyle.Assign[477](rve.Style[193].TextStyles[437][Result]);
    TextStyle.Jump[479] := True;
    TextStyle.Color[469] := clBlue;
    TextStyle.Style[475] := TextStyle.Style+[fsUnderline];
    Result := rve.Style.FindTextStyle[440](TextStyle);
  finally
    TextStyle.Free;
  end;
end;
```

Default value:

- AcceptDragDropFormats: [rvddRVF, rvddRTF, rvddText, rvddUnicodeText, rvddBitmap, rvddMetafile, rvddFiles]
- AcceptPasteFormats: [rvddRVF, rvddRTF, rvddText, rvddUnicodeText, rvddBitmap, rvddMetafile, rvddFiles, rvddHTML]

See also:

- Paste[361];
- CanPaste[337];
- Drag&Drop[89].

1.5.1.2.1.2 TRichView Edit.ActualCurTextStyleNo

Index of the current text style in the collection of text styles. Can return rvsDefStyle[711].

```
property ActualCurTextStyleNo: Integer;
```

(introduced in v1.8)

This is an index in the collection Style[193].TextStyles[437].

This property is the same as CurTextStyleNo[324], but:

- read-only;
- does not process *rvsDefStyle*[711], returns it as it is.

1.5.1.2.1.3 TRichView Edit.CurItemNo

Index of item at the position of caret

```
property CurItemNo: Integer;
```

This value is in range from 0 to ItemCount[180]-1.

If the caret is inside table cell, this property returns index of the table in the main document. Items of subdocuments (table cells[605]) are not included in the items range of the main document. If you want to know index of item at the position of caret in the table cell, use TopLevelEditor[329].CurItemNo.

This property must be accessed only when the document is formatted.

When called for an empty editor (i.e. ItemCount[180]=0), the method returns -1.

If you want to get the caret position as a number of characters from the beginning of the document, use RVGetCaretPos from RVLinear[666] unit instead.

See also properties:

- OffsetInCurItem[328];
- TopLevelEditor[329];
- CurItemStyle[323];
- CurTextStyleNo[324];
- CurParaStyleNo[323].

See also events:

- OnCaretMove[390].

See also:

- Inserting in the position of the caret[86];
- How to: move the caret to the beginning or to the end of document in TRichViewEdit[717].

1.5.1.2.1.4 TRichView Edit.CurItemStyle

Style (i.e. type) of item at the position of caret

```
property CurItemStyle: Integer;
```

The same as TopLevelEditor[329].GetItemStyle[229](TopLevelEditor[329].CurItemNo[322]).

If the current item is a text item, this function returns zero or positive value: index in the collection of text styles (Style[193].TextStyles[437]).

Otherwise, this function returns a type of this item: see rvsXXX constants[711].

This property must be accessed only when the document is formatted.

See also methods:

- GetItemStyle[229].

See also properties:

- CurItemNo[322];
- CurTextStyleNo[324].

See also:

- Item types[64].

1.5.1.2.1.5 TRichView Edit.CurParaStyleNo

Index of the current paragraph style in the collection of paragraph styles.

```
property CurParaStyleNo: Integer;
```

This is an index in the collection Style[193].ParaStyles[431].

This is a style of paragraph containing caret.

Value of this is changed when the user moves the caret to another paragraph or edits the document, or when some methods are called. When this property is changed, OnCurParaStyleChanged[391] event occurs.

Usually you do not need not to assign value to this property directly, use ApplyParaStyle[331] or ApplyParaStyleConversion[332] methods instead.

This property must be accessed only when the document is formatted.

See also methods:

- ApplyParaStyle[331];
- ApplyParaStyleConversion[332].

See also properties:

- CurTextStyleNo[324].

See also events:

- OnCurParaStyleChanged[391].

See also:

- Working with paragraphs[72].

1.5.1.2.1.6 TRichView Edit.CurTextStyleNo

Index of the current text style in the collection of styles.

```
property CurTextStyleNo: Integer;
```

This is an index in the collection Style[193].TextStyles[437].

This text style will be used when inserting new text in the document in the position of caret (both by user keyboard input and by methods, such as PasteText[364] or InsertText[360]).

This property is changed when user moves the caret to another text item or edits the document, or when some methods are called. When this property is changed, OnCurTextStyleChanged[391] event occurs.

Value of this property is always positive. To get style (i.e. type) of item at the caret position, use CurItemStyle[323] property.

The editor never sets this property to index of text style having *rvprDoNotAutoSwitch* in Protection[473]. You can do it yourself to insert a special symbol (by keyboard input, or InsertText[360]), but after that the component automatically switches this property back to the last used style without *rvprDoNotAutoSwitch* in Protection[473].

Usually you do not need to assign value to this property directly, use ApplyTextStyle[334] or ApplyStyleConversion[333] instead.

This property must be accessed only when the document is formatted.

This property never returns *rvsDefStyle*[711] constant. It "translates" it and returns the index of text style that is used for text. If you want to know if the text actually has *rvsDefStyle*[711] style, use ActualCurTextStyleNo[322] property.

See also methods:

- ApplyTextStyle[334];
- ApplyStyleConversion[333].

See also properties:

- CurItemStyle[323];
- CurParaStyleNo[323];
- ActualCurTextStyleNo[322].

See also events:

- OnCurTextStyleChanged[391].

1.5.1.2.1.7 TRichView Edit.CustomCaretInterval

If set to positive value, activates the custom caret feature. Defines the timer interval for redrawing caret, in milliseconds.

```
property CustomCaretInterval: Integer;
```

(introduced in version 10)

The caret is activated when you call Format[217] method (and is deactivated when you call Clear[210]).

It's not recommended to use the custom caret feature together with image animation[90].

Default value:

0 (the system caret is used)

See also events:

- OnMeasureCustomCaret[394];
- OnDrawCustomCaret[391].

1.5.1.2.1.8 TRichView Edit.DefaultPictureVAlign

Specifies the default alignment for pasted images, and for images inserted as a result of drag and drop operations.

```
property DefaultPictureVAlign: TRVVAlign[695];
```

(introduced in version 14)

The following methods use this property:

- PasteBitmap[362]
- PasteMetafile[363]
- PasteGraphicFile[362]
- Paste[361] (when inserting a bitmap, a metafile, or a graphic file);

This property is also used when a picture is pasted using the keyboard (when the user presses Ctrl + V or Shift + Insert), or when dropping a picture in the editor.

Default value:

rvvaBaseline

1.5.1.2.1.9 TRichView Edit.EditorOptions

Additional options affecting editing

```
type
  TRVEditorOption = (rvoClearTagOnStyleApp, rvoCtrlJumps,
    rvoDoNotWantReturns, rvoWantTabs,
    rvoAutoSwitchLang, rvoHideReadOnlyCaret,
    rvoDoNotWantShiftReturns, rvoNoImageResize,
    rvoNoCaretHighlightJumps, rvoNoReadOnlyJumps,
    rvoDragDropPicturesFromLinks)
  TRVEditorOptions = set of TRVEditorOption;

property EditorOptions: TRVEditorOptions;
```

Options for disabling/enabling certain editing operations

Option	Meaning
rvoDoNotWantReturns	If set, editor ignores Enter keys.
rvoDoNotWantShiftReturns	If set, Shift+Enter does nothing (by default, it inserts a line break inside paragraph – noticeable in paragraphs with bullets or borders)
rvoWantTabs	If set (default), editor accepts Tab key (Ctrl+Tab always works). Tabs can be inserted as several spaces, or as a tabulator[121] (special item type), depending on the value of SpacesInTab[435] property of the linked TRVStyle component.
rvoNoImageResize	Prevents resizing pictures[122] and controls[126] with mouse.

Visual options

Option	Meaning
rvoNoCaretHighlightJumps	If set, hyperlinks will not be highlighted (with HoverColor[470] and HoverBackColor[470]) when the caret is moved inside them.
rvoHideReadOnlyCaret	If set, the editor hides the caret in read-only mode[328]

Drag&drop options

Option	Meaning
rvoDragDropPicturesFromLinks	If set, when an URL is dropped in the editor, if this is a link to an image, the editor tries to insert it as an image, not as a link. The most probably, the link points to a remote image, so it needs to be downloaded before insertion. You can do it in OnImportPicture[279] event. If *rvoAssignImageFileNames* in Options[182], the component assigns a "file name" property for this image.

Hypertext options

Option	Meaning
rvoCtrlJumps	If set (default), the editor switches to the hypertext mode (highlighting hypertext items under mouse cursor, using hypertext cursors, generating OnJump[282] and OnRVMouseMove[290]) when user presses and holds Ctrl key The editor switches back to the editing mode when user releases Ctrl key or changes document, or if the editor loses focus.
rvoNoReadOnlyJumps	If not set (default), the editor automatically switches to the hypertext mode when it becomes read-only. If set, hyperlinks in a read-only mode work like in a normal mode.

Other options

Option	Meaning
rvoClearTagOnStyleApp	If set, editor clears *tags*[68] (set to '') of all text items when changing their style index using ApplyTextStyle[334] or ApplyStyleConversion[333] method.
rvoAutoSwitchLang	Obsolete, does nothing.

Default value (changed in v1.9):

[*rvoWantTabs*, *rvoCtrlJumps*]

See also:

- Options 182.

See also events:

- OnJump 282;
- OnRVMouseMove 290.

See also:

- TRVStyle.SpacesInTab 435;
- RichView Hypertext 69;
- *Tags* 68.

1.5.1.2.1.10 TRichView Edit.ForceFieldHighlight

Allows highlighting label items 133 (including numbered sequences 134, footnotes 136, endnotes 135, sidenotes 137) even when the editor is not focused.

```
property ForceFieldHighlight: Boolean;
```

(Introduced in version 15)

The editor can highlight label items with Style 193.FieldHighlightColor 423.

There are three modes possible, depending on Style 193.FieldHighlightType 423: all labels are highlighted, a current label is highlighted, nothing is highlighted.

This property affects the mode when the current (at the caret position) item is highlighted. If *False*, it is highlighted only if the editor has the input focus. If *True*, it is highlighted even in unfocused editors.

This property is useful if you implement editing properties of these items using controls on the same form. It allows to highlight the edited item.

1.5.1.2.1.11 TRichView Edit.LiveSpellingMode

Defines when live spelling thread is started.

```
type
  TRVLiveSpellingMode = (rvlspManualStart, rvlspOnChange);

property LiveSpellingMode: TRVLiveSpellingMode;
```

(introduced in v1.9)

Value	Meaning
rvlspManualStart	Live spelling check can be started only by StartLiveSpelling 273.
rvlspOnChange	Live spelling check also starts when the user modifies document (or when any editing-style method is called).

Default value:

rvlspOnChange

See also:

- Live spelling check in RichView 101.

1.5.1.2.1.12 TRichView Edit.Modified

Value of this property is *True* if document was modified

```
property Modified: Boolean.
```

This is a public (not published) property.

Initially it is *False*. Calling Clear 210 resets it to *False*.

Any editing operation, including calling any editor-style method (which invokes OnChange 390 event) sets it to *True*.

Calling any method introduced in TCustomRichView does not affect it.

If you use this property, set it to *False* after saving document to file.

1.5.1.2.1.13 TRichView Edit.OffsetInCurItem

Offset of the caret in the current item.

```
property OffsetInCurItem: Integer;
```

Index of item at the position of the caret is returned in CurItemNo 322 property.

CurItemNo and OffsetInCurItem together define position of the caret in the document.

If item is a text item, then caret is before the OffsetInCurItem-th character of string. If caret is after the last character of the text item, OffsetInCurItem = length of the item + 1.

If item is not a text, then

- if the caret is before the item, then OffsetInCurItem=0;
- if the caret is after the item, then OffsetInCurItem=1.

If selection exists, position of the caret is always equal to the second bound of selection (because the caret is always at the end of selection).

You can set the caret position using SetSelectionBounds 271.

If the caret is inside table cell, this property returns 1 (position after the table). Items of subdocuments (table cells 605) are not included in the items range of the main document. If you want to know position of the caret in the table cell, use TopLevelEditor 329.CurItemNo 322 and TopLevelEditor 329.OffsetInCurItem.

This property must be accessed only when the document is formatted.

See also properties:

- CurItemNo 322;
- TopLevelEditor 329.

See also events:

- OnCaretMove 390.

See also methods:

- SetSelectionBounds 271;
- GetSelectionBounds 237.

See also:

- Inserting in the position of the caret 86;
- How to: move the caret to the beginning or to the end of document in TRichViewEdit 717.

1.5.1.2.1.14 TRichView Edit.ReadOnly

When set to *True*, this property prevents user's input in the editor; it also blocks execution of all editing-style methods.

```
property ReadOnly: Boolean;
```

This property does not affect viewer-style methods.

Editor hides the caret in read-only mode, if *rvoHideReadOnlyCaret* is in EditorOptions 325.

Since version 1.6, hypertext in read-only mode works automatically (user does not need to press and hold Ctrl key).

Default value:

False

See also events:
- OnChanging 390.

1.5.1.2.1.15 TRichView Edit.SmartPopupProperties

Properties for "smart popups" (analog of Microsoft Office's "smart tags" buttons).

```
property SmartPopupProperties: TRVSmartPopupProperties 402;
```

(introduced in version 10)

See TRVSmartPopupProperties 402 for details.

See also properties:
- SmartPopupVisible 329.

See also events:
- OnSmartPopupClick 400.

1.5.1.2.1.16 TRichView Edit.SmartPopupVisible

Shows/hides "smart popup" 402 button for the current (at the position of caret) item.

```
property SmartPopupVisible: Boolean;
```

(introduced in version 10)

The editor hides this button automatically on document change or clearing.

Since smart popup works only for the current item, you must ensure that it is displayed for the current item: update it in OnCaretMove 390 event. Even if this property is *True*, reassigning *True* to it makes sense, because it links the smart popup to the current item.

Properties for the button are defined in SmartPopupProperties 329.

See also events:
- OnSmartPopupClick 400.

1.5.1.2.1.17 TRichView Edit.TopLevelEditor

Returns active editor component

```
property TopLevelEditor: TCustomRichViewEdit;
```

(introduced in version 1.7)

This property returns editor containing the caret.

This is either this RichViewEdit itself or InplaceEditor 546 of table cell which is being edited.

If cell of nested table is being edited, this is InplaceEditor of InplaceEditor, and so on.

Insert*, **GetCurrent***, **SetCurrent*** methods automatically work with content of TopLevelEditor.

Use this property, if you want to work with methods having ItemNo in parameters.

This property must be accessed only when the document is formatted.

1.5.1.2.1.18 TRichView Edit.UndoLimit

Limits capacity of the undo buffer.

```
property UndoLimit: Integer;
```

(Introduced in version 1.3)

Value	Meaning
0	Undo is turned off
-1	Undo buffer "unlimited" (limited by available memory)
any positive value	Limit of saved operations to undo

Note 1: although typing is undone character-by-character, it is saved as a single undo action (for more efficient memory usage).

Note 2: It's not recommended for this version to set UndoLimit to 1. Use multilevel undo instead.

Note 3: Modifying UndoLimit does not affect the current state of undo buffer (no items are deleted from it)

This property does not affect behavior of methods introduced in TRichView, they can't be undone.

Default value:

-1 (unlimited buffer)

See also:

- Undo in RichViewEdit 88.

1.5.1.2.2 Methods

1.5.1.2.2.1 TRichViewEdit.AdjustControlPlacement

Returns the control 126 owned by the **ItemNo**-th item to its position in the document.

```
procedure AdjustControlPlacement(ItemNo: Integer);
```

Usually you do not need to use this method. RichViewEdit moves and scrolls inserted controls when it's needed.

If you want to resize control, use ResizeCurrentControl 368 (or ResizeControl 368) method.

Use this method if the control has moved or resized itself. This method moves the control to the proper place in a document.

If the control has changed its size, this method also quickly reformats the affected part of the document.

It's recommended to use AdjustControlPlacement2 330 instead of this method, because it can work with controls inserted in table 145 cells.

This method must be called only when the document is formatted.

1.5.1.2.2.2 TRichViewEdit.AdjustControlPlacement2

Returns **Control** back to its position in the document. **Control** must be inserted in RichViewEdit directly or in table cell.

```
procedure AdjustControlPlacement2(Control: TControl);
```

(introduced in version 1.4)

Usually you do not need to use this method. RichViewEdit moves and scrolls inserted controls when it's needed.

If you want to resize control, use ResizeCurrentControl 368 (or ResizeControl 368) method.

Use this method if **Control** has moved or resized itself. This method moves the control to the proper place in the document.

If the control has changed its size, this method also quickly reformats the affected part of the document.

This method works only if the control is inserted in the root RichViewEdit or in cell inplace editor.

This method must be called only when the document is formatted.

See also:

- AdjustControlPlacement 330.

1.5.1.2.2.3 TRichView Edit.ApplyListStyle

Applies list style (bullets or numbering) to the selected paragraphs

```
procedure ApplyListStyle(AListNo, AListLevel, AStartFrom: Integer;
  AUseStartFrom, ARecursive: Boolean);
```

(introduced in version 1.7)

This method adds/changes list markers for the selected paragraph(s).

Parameters

AListNo defines style of this list. This is an index in the collection of list styles (Style 193.ListStyles 429)

AListLevel – list level. This is an index in the collection of list levels (Style.ListStyles[**AListNo**].Levels 493).

If **AListLevel**=-1, levels of the existing markers are not changed, new markers have level=0.

If **AUseStartFrom**=*True*, and this is a numbered list, **AStartFrom** defines value of the list counter for this marker. If **AUseStartFrom**=*False*, the numbering is continued. For bulleted lists, these parameters are ignored.

ARecursive – reserved for future use. Set it to *False*.

Method type: editor-style method (unlike SetListMarkerInfo 269).

See also:

- RemoveLists 367;
- ChangeListLevels 338.

See also methods of TCustomRichView:

- SetListMarkerInfo 269;
- GetListMarkerInfo 233.

1.5.1.2.2.4 TRichView Edit.ApplyParaStyle

Applies the specified paragraph style.

```
procedure ApplyParaStyle(ParaStyleNo: Integer);
```

ParaStyleNo – index in the collection of paragraph styles (Style 193.ParaStyles 431).

The method applies the paragraph style **ParaStyleNo** to the paragraph containing the caret and all the selected paragraphs.

If existing paragraphs are formatted with styles having *rvpaoStyleProtect* in their Options 487, this method does not change their styles.

Method type: editing-style.

If you want to implement commands like "change alignment", "increase indent" use ApplyParaStyleConversion 332 instead.

See also properties:

- CurParaStyleNo 323.

See also methods:

- ApplyTextStyle 334;
- ApplyParaStyleConversion 332.

See also:

- TParaInfo.Options 487;
- Paragraphs overview 72.

1.5.1.2.2.5 TRichView Edit.ApplyParaStyleConversion

Applies a custom conversion procedure to styles of the selected paragraphs.

```
procedure ApplyParaStyleConversion(UserData: Integer; Recursive: Boolean = True);
```

(introduced in version 1.7)

This procedure calls OnParaStyleConversion[397] event for each selected item. Using this method, you can implement commands like "change alignment" or "increase indent".

This method ignores any paragraph protection[487], it can change paragraph style even if *rvpaoStyleProtect* is included in Options[487] of existing paragraphs' styles. If you want to respect paragraph style protection, check this option yourself in OnParaStyleConversion[397] event.

If **Recursive** = *False*, OnParaStyleConversion[397] is not called for cells of selected tables[130] (except for the case of multicell selection in a single table).

Method type: editing-style.

Demo:

- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

See also:

- ApplyStyleConversion[333];
- ApplyParaStyle[331].

See also events:

- OnParaStyleConversion[397].

See also:

- Paragraphs overview[72].

1.5.1.2.2.6 TRichView Edit.ApplyParaStyleTemplate

Applies the specified style template to the selected paragraphs.

```
procedure ApplyParaStyleTemplate(TemplateNo: Integer;
  ResetAdditionalFormatting: Boolean; Recursive: Boolean=True);
```

(introduced in version 14)

This procedure assigns the specified style template to paragraph styles[431] of items in the selected paragraphs.

Parameters:

TemplateNo – index in the collection Style[193].StyleTemplates[435], or the value -1. The value -1 means a format clearing (see below). The style template must have Kind[495] = *rvstkParaText* or *rvstkPara*.

If **ResetAdditionalFormatting**=*True*, all properties of the specified style template will be applied. If **ResetAdditionalFormatting**=*False*, only "unmodified" properties (see below) will be changed. This parameter is ignored if **TemplateNo**=-1.

If **Recursive** = *False*, this method does not change styles in cells of selected tables[130] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates[195]=*False* and **TemplateNo**>=0.

If paragraphs are formatted with styles having *rvpaoStyleProtect* in their Options[487], this method skips them.

Format clearing

A format clearing means the following procedure. If "Normal" style template exists, the method works like applying this style template with **ResetAdditionalFormatting**=*True*. If it does not exist, the method resets all properties of paragraph styles to default values, and assigns StyleTemplateId[465]=-1.

If UseStyleTemplates[195]=*False* and **TemplateNo**<0, the results are identical to calling ApplyParaStyle[331](0).

Modified properties

Properties of a paragraph style[490] are considered modified, if they have values different from values of the corresponding properties of the current style template (referred in StyleTemplateId[465] property). If a style template is not assigned, the properties are considered modified if they have non-default values. The method ignores ModifiedProperties[491] property (it calculates this property itself).

Method type: editing-style.

See also:

- ApplyTextStyleTemplate[334];
- ApplyParaStyleConversion[332];
- ApplyParaStyle[331].

1.5.1.2.2.7 TRichView Edit.ApplyStyleConversion

Applies a custom conversion procedure to styles of the selected text items[120], tabs[121], label items[133] (and item of all types inherited from label items).

```
procedure ApplyStyleConversion(UserData: Integer; Recursive: Boolean = True);
```

This procedure calls OnStyleConversion[400] event for each selected text item and for the current text style[324]. Using this method, you can implement commands like "make bold" or "change font name".

This method ignores any text protection[473], it can change text style even if *rvprStyleProtect* is included in Protection[473] of existing text items' styles. If you want to respect text style protection, check this protection option yourself in OnStyleConversion[400] event.

If **Recursive** = *False*, OnStyleConversion[400] is not called for cells of selected tables[130] (except for the case of multicell selection in a single table).

Method type: editing-style.

Demo:

- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

See also:

- ApplyParaStyleConversion[332];
- ApplyTextStyle[334].

See also events:

- OnStyleConversion[400].

1.5.1.2.2.8 TRichView Edit.ApplyStyleTemplate

Applies the specified style template to the selected text or paragraphs.

```
procedure ApplyStyleTemplate(TemplateNo: Integer; Recursive: Boolean=True);
```

(introduced in version 14)

This procedure assigns the specified style template to text styles[437] or paragraph styles[431] of items in the selected paragraphs. The result depends on the style template's Kind[495] and the selection.

Parameters:

TemplateNo – index in the collection Style[193].StyleTemplates[435], or the value -1. The value -1 means a format clearing (see below).

If **Recursive** = *False*, this method does not change styles in cells of selected tables[130] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates[195]=*False* and **TemplateNo**>=0.

The method works according to the following rules:

- if Kind[495] of the specified style template = *rvstkText*, it is applied to the selected text;
- if Kind[495] of the specified style template = *rvstkPara*, it is applied to the selected paragraphs; the only exception is "Normal" style template[461];
- if **TemplateNo**=-1, or Kind[495] of the specified style template = *rvstkParaText*, or this is a "Normal" style template, the result depends on the selection. If nothing is selected, or the whole paragraph is selected, or the selection includes multiple paragraphs/cells, the method applies to the selected paragraphs. Otherwise, it applies to the selected text.

The method clears additional formatting, like ApplyTextStyleTemplate[334] or ApplyParaStyleTemplate[332] called with **ResetAdditionalFormatting**=*True*.

The method skips paragraph styles having *rvpaoStyleProtect* in their Options[487], and text styles having *rvprStyleProtect* in their Protection[473].

Format clearing

The method clears text or paragraph formatting in two cases: when it is called with **TemplateNo**=-1, or when it applies "Normal" style template. The results of format clearing is described in the topics about ApplyTextStyleTemplate[334] and ApplyParaStyleTemplate[332].

Method type: editing-style.

See also:

- ChangeStyleTemplates[338].

1.5.1.2.2.9 TRichView Edit.ApplyTextStyle

Applies the specified text style to the selected items (text items[120], tabs[121], label items[133])

```
procedure ApplyTextStyle(TextStyleNo: Integer);
```

TextStyleNo – index in the collection of text styles (Style[193].TextStyles[437]).

If selection is empty, this method only sets CurTextStyleNo[324] to **TextStyleNo**. If the selection is not empty, the method changes style of text items[120] in the selection.

If the existing text items are formatted with styles having *rvprStyleProtect* in their Protection[473], this method does not change their styles.

Method type: editing-style.

If you want to implement commands like "make bold", "change text color", "apply font", use ApplyStyleConversion[333] instead of this method.

See also properties:

- CurTextStyleNo[324];
- EditorOptions[325].

See also methods:

- ApplyParaStyle[331].

1.5.1.2.2.10 TRichView Edit.ApplyTextStyleTemplate

Applies the specified style template to the selected text items[120], tabs[121], label items[133] (and to items of all types inherited from label items).

```
procedure ApplyTextStyleTemplate(TemplateNo: Integer;
  ResetAdditionalFormatting: Boolean; Recursive: Boolean=True);
```

(introduced in version 14)

This procedure assigns the specified style template to text styles[437] of the selected items.

Parameters:

TemplateNo – index in the collection Style[193].StyleTemplates[435], or the value -1. The value -1 means a format clearing (see below). The style template must have Kind[495] = *rvstkParaText* or *rvstkText*. The only exception is "Normal" style template[461]: applying it clears text format as well, but **ResetAdditionalFormatting** is respected.

If **ResetAdditionalFormatting**=*True*, all properties of the specified style template will be applied. If **ResetAdditionalFormatting**=*False*, only "unmodified" properties (see below) will be changed. This parameter is ignored if **TemplateNo**=-1.

If **Recursive** = *False*, this method does not change styles in cells of selected tables[130] (except for the case of multicell selection in a single table).

This method does nothing if UseStyleTemplates[195]=*False* and **TemplateNo**>=0.

If items are formatted with text styles having *rvprStyleProtect* in their Protection[473], this method skips them.

Format clearing

A format clearing means resetting all text attributes to default values. If a paragraph style template (ParaStyleTemplateId[473]) is assigned, it defines these default values. A special processing is performed for hyperlinks (i.e. text styles with Jump[479]=*True*): if "Hyperlink" style template exists, it is applied (in other words, the method works like applying this style template with **ResetAdditionalFormatting**=*True*).

If UseStyleTemplates[195]=*False* and **TemplateNo**<0, the results are identical to calling ApplyTextStyle[334](0).

Modified properties

Properties of a text style[478] are considered modified, if they have values different from values of the corresponding properties of the current style templates (referred in StyleTemplateId[465] and ParaStyleTemplateId[473] properties). If none of these style templates are assigned, the properties are considered modified of they have non-default values. The method ignores ModifiedProperties[480] property (it calculates this property itself).

Method type: editing-style.

See also:

- ApplyParaStyleTemplate[332];
- ApplyStyleConversion[333];
- ApplyTextStyle[334].

1.5.1.2.2.11 TRichView Edit.BeginItemModify

Stores some information required for quick reformatting after modification of item.

```
procedure BeginItemModify(ItemNo: Integer; var ModifyData: Integer);
```

Input parameter:

ItemNo – index of item which will be modified.

Output parameter:

ModifyData receives some formatting information. Value assigned to this parameter should be passed to EndItemModify[341] after performing modifications.

Usually this function is used when modifying tables[565].

This method must be called only when the document is formatted.

Example is in Table Operations Overview[153]

1.5.1.2.2.12 TRichView Edit.BeginUndoCustomGroup

Begins a new undo action.

```
procedure BeginUndoCustomGroup(const Name: TRVUnicodeString[694]);
```

(changed in version 18)

Begins a new undo action.

Usually followed by SetUndoGroupMode[387](*True*) ... SetUndoGroupMode(*False*), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as BeginUndoGroup[336], but allows to give a custom name to this group of changes.

This method must be called only when the document is formatted.

This method must be called directly for the editor where changed are made (usually TopLevelEditor[329]).

See also:

- Undo in RichViewEdit[88].

1.5.1.2.2.13 TRichView Edit.BeginUndoGroup

Begins a new undo action.

```
procedure BeginUndoGroup(UndoType: TRVUndoType[693]);
```

Begins a new undo action.

Usually followed by SetUndoGroupMode[387](*True*) ... SetUndoGroupMode(*False*), thus allowing to group several subsequent changes in editor (they will be undone/redone as one operation).

The same as BeginUndoCustomGroup[336], but gives a standard name (undo identifier) to this group of changes.

This method must be called only when the document is formatted.

This method must be called directly for the editor where changed are made (usually TopLevelEditor[329]).

See also:

- Undo in RichViewEdit[88], includes example.

1.5.1.2.2.14 TRichView Edit.BeginUpdate

Suspends repainting until EndUpdate[341] is called.

```
procedure BeginUpdate;
```

(meaning of this method is changed since version 10)

Use BeginUpdate to speed processing and avoid flicker while multiple editing operations are applied.

Calls to BeginUpdate and EndUpdate can be nested.

1.5.1.2.2.15 TRichView Edit.CanChange

Returns: "can the document in this editor be changed now?"

```
function CanChange: Boolean;
```

This function checks ReadOnly[328] property.

Usually this function is called before performing operations on tables[565] to prevent modifying editor in read-only state.

For TDBRichViewEdit[410], this function also turns the editor's dataset in editing mode (required before operations on tables).

This method must be called only when the document is formatted.

1.5.1.2.2.16 TRichView Edit.CanPaste

Returns *True* if:

- the Clipboard contains data in at least one format which RichViewEdit can paste (and this format is included in AcceptPasteFormats[321] property), and
- the editor is not in read-only state (ReadOnly[328]=*False*)

```
function CanPaste: Boolean;
```

See also methods:

- Paste[361];
- CanPasteRVF[337];
- CanPasteRTF[337].

See also:

- Working with Clipboard[82].

1.5.1.2.2.17 TRichView Edit.CanPasteRTF

Returns *True* if:

- the Clipboard contains data in RTF (Rich Text Format), and
- the editor is not in read-only state (ReadOnly[328]=*False*)

```
function CanPasteRTF: Boolean;
```

(Introduced in version 1.5)

See also methods:

- PasteRTF[363];
- CanPaste[337].

See also:

- Working with Clipboard[82]

1.5.1.2.2.18 TRichView Edit.CanPasteRVF

Returns *True* if:

- the Clipboard contains data in RVF (RichView Format[93]), and
- the editor is not in read-only state (ReadOnly[328]=*False*)

```
function CanPasteRVF: Boolean;
```

See also methods:

- PasteRVF[364];
- CanPaste[337].

See also:

- Working with Clipboard[82];
- RVF[93].

1.5.1.2.2.19 TRichView Edit.Change

Generates OnChange[390] event.

```
procedure Change;
```

This method also assigns *True* to Modified[327] property.

1.5.1.2.2.20 TRichView Edit.ChangeListLevels

Changes list levels (levels of paragraphs bullets and numbering) for the selected paragraphs.

```
procedure ChangeListLevels(LevelDelta: Integer);
```

(introduced in version 1.7)

For example,

```
  rve.ChangeListLevels(+1) // increases level by 1
  rve.ChangeListLevels(-1) // decreases level by 1
```

Paragraphs without bullets or numbering are not affected.

Method type: editing-style.

See also:

- ApplyListStyle[331];
- RemoveLists[367].

See also methods of TCustomRichView:

- SetListMarkerInfo[269];
- GetListMarkerInfo[233].

1.5.1.2.2.21 TRichView Edit.ChangeStyleTemplates

Updates the document after editing style templates.

```
procedure ChangeStyleTemplates(NewStyleTemplates: TRVStyleTemplateCollection[460]);
```

(introduced in version 14)

The code below shows how to edit style templates for RichViewEdit1: TRichViewEdit. It assumes that you have RVStyleTmp: TRVStyle[418] component, not linked to any RichView.

```
RVStyleTmp.Units[437] := RichViewEdit1.Style.Units;
RVStyleTmp.StyleTemplates[435].AssignStyleTemplates[461](RichViewEdit1.Style.StyleTemplates, Tru
<code for changing RVStyleTmp.StyleTemplates here>
RichViewEdit1.ChangeStyleTemplates(RVStyleTmp.StyleTemplates);
```

While editing RVStyleTmp.StyleTemplates, you can:

- add new style templates;
- change any property of style templates, rename them;
- delete style templates (use RVStyleTmp.StyleTemplates.ForbiddenIds[461]).

Method type: editing-style.

ScaleRichView note: in TSRichViewEdit, this method must be called for SRichViewEdit.ActiveEditor. It may be called while the main editor, a header or a footer is being edited. It cannot be called while a footnote or an endnote is being edited.

See also:

- ApplyStyleTemplate[333];
- ApplyParaStyleTemplate[332];
- ApplyTextStyleTemplate[334].

1.5.1.2.2.22 TRichView Edit.Clear

Clears the document.

```
procedure Clear;override;
```

This method:

- deletes all content of the document (see RichView.Clear[210]);
- sets the current text style index (CurTextStyleNo[324]) and the current paragraph style index (CurParaStyleNo[323]) to zeros;

- clears undo and redo buffers (see ClearUndo 339);
- sets Modified 327 to *False*.

This method does not check ReadOnly 328, does not reformat and repaint component, does not generate OnChange 390.

This methods makes the document unformatted.

1.5.1.2.2.23 TRichView Edit.ClearTextFlow

Clears text flow around left- and right-aligned items.

```
procedure ClearTextFlow(Left, Right: Boolean);
```

(introduced in version 12)

This method changes ClearLeft 171 and ClearRight 172 property for the first selected paragraph.

Method type: editing-style.

See also:

- TRVVAlign 695 type (*rvvaLeft* and *rvvaRight*)

1.5.1.2.2.24 TRichView Edit.ClearUndo

Clears undo and redo buffers of the editor.

```
procedure ClearUndo;
```

(Introduced in version 1.3)

See also:

- Undo in RichViewEdit 88.

1.5.1.2.2.25 TRichView Edit.ConvertToHotPicture

Converts the item type from picture 122 to *hot-picture* 125.

```
procedure ConvertToHotPicture(ItemNo: Integer);
```

Hot-picture is a picture with hypertext link.

If **ItemNo**<0, this methods works with the item at the position of caret. Otherwise (if **ItemNo** in range 0..ItemCount 180-1) it works with the specified item

If this item is not a picture (GetItemStyle 229/CurItemStyle 323 <> *rvsPicture*), this method does nothing.

Method type: editing-style.

See also:

- ConvertToPicture 339.

1.5.1.2.2.26 TRichView Edit.ConvertToPicture

Converts the item type from *hot-picture* 125 to picture 122

```
procedure ConvertToPicture(ItemNo: Integer);
```

Hot-picture is a picture with hypertext link.

If **ItemNo**<0, this methods works with the item at the position of caret. Otherwise (if **ItemNo** in range 0..ItemCount 180-1) it works with the specified item.

If this item is not a *hot-picture* (GetItemStyle 229/CurItemStyle 323 <> *rvsHotPicture*), this method does nothing.

Method type: editing-style.

See also:

- ConvertToHotPicture[339].

1.5.1.2.2.27 TRichViewEdit.Create

Usual component constructor.

Creates a new RichViewEdit instance.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate RichViewEdit at runtime. RichViewEdits placed on forms at design time are created automatically. Specify the owner of the new RichViewEdit using the AOwner parameter.

Important note: initial values of several properties of components created in code may be different from initial values of components placed on form at design time. In the latter case, initial values of these properties can be overridden by the designtime component editor[168] (and overridden by default!)

1.5.1.2.2.28 TRichViewEdit.CutDef

Cuts selection to the Clipboard

```
procedure CutDef;
```

This method copies selection to the Clipboard using CopyDef[212] method. If the selection was copied (at least one of copy formats is selected in Options[182]), it calls DeleteSelection[340].

This method is executed automatically when user presses Ctrl+X or Shift+Delete.
Method type: editing-style.

See also methods:

- CopyDef[212];
- Paste[361];
- SelectionExists[258].

See also:

- Working with selection[81];
- Working with Clipboard[82].

1.5.1.2.2.29 TRichViewEdit.DeleteSelection

Deletes the selected fragment in the editor.

```
procedure DeleteSelection;
```

Method type: editing-style.

See also:

- Working with selection[81].

1.5.1.2.2.30 TRichViewEdit.Destroy

Destroys an instance of TRichViewEdit

```
destructor Destroy;override;
```

Do not call Destroy directly in an application. Instead, call Free. Free verifies that the control is not *nil*, and only then calls Destroy.

Applications should only free controls explicitly when the constructor was called without assigning an owner to the control.

1.5.1.2.2.31 TRichView Edit.EndItemModify

Quickly reformats editor after modifying item.

```
procedure EndItemModify(ItemNo: Integer; ModifyData: Integer);
```

Parameters:

ItemNo – index of item which was modified.

ModifyData – the value obtained in BeginItemModify[335]

Usually this function is used when modifying tables[565].

Example is in Table Operations Overview[153]

1.5.1.2.2.32 TRichView Edit.EndUpdate

Reenables editor repainting that was turned off with the BeginUpdate[336] method. Repaints the editor.

```
procedure EndUpdate;
```

(meaning of this method is changed since v10)

Calls to BeginUpdate are cumulative, so calling EndUpdate will reenable repainting only if every call to BeginUpdate has been matched by a call to EndUpdate.

1.5.1.2.2.33 TRichView Edit.GetCheckpointAtCaret

Returns *checkpoint* at the position of caret.

```
function GetCheckpointAtCaret: TCheckpointData;
```

(introduced in version 10)

This method returns *checkpoint* only if the caret is at the beginning of item, otherwise it returns *nil*.

The following table compares this method with GetCurrentCheckpoint[343].

Caret Position	GetCheckpointAtCaret	GetCurrentCheckpoint[343]
In the middle of text item	returns *nil*	returns *checkpoint* for this item
At the beginning of item and at the beginning of line	returns *checkpoint* for this item	returns *checkpoint* for this item
At the beginning of item and in the middle of line	returns *checkpoint* for this item	returns *checkpoint* for the previous item
At the end of item and at the end of line	returns *nil*	returns *checkpoint* for this item

This method is used in conjunction with InsertCheckpoint[351] and RemoveCheckpointAtCaret[366]. The alternative set of methods is: SetCurrentCheckpointInfo[376], GetCurrentCheckpoint[343] and RemoveCurrentCheckpoint[367].

See also:

- "Checkpoints"[65].

1.5.1.2.2.34 TRichView Edit.GetCurrentBreakInfo

Returns main properties for the item of *break*[125] type at the position of caret.

```
procedure GetCurrentBreakInfo(var AWidth: TRVStyleLength[690];
  var AStyle: TRVBreakStyle[669]; var AColor: TColor;
  var ATag: TRVTag[692]);
```

GetCurrentBreakInfo(...) is equivalent to TopLevelEditor[329].**GetBreakInfo**[218](TopLevelEditor[329].CurItemNo[322], ...).

The item must be of *break*[125] type (*rvsBreak*[711]), otherwise the method raises ERichViewError[649] exception. Type of item at the position of caret is returned by CurItemStyle[323].

Output parameters:

AWidth – line width (or rectangle height). This value is measured in Style[193].Units[437].

AStyle – visual style of this *break*, see TRVBreakStyle[669] for possible values.

AColor – line color. If it is equal to *clNone*, Style[193].TextStyles[437][0].Color[469] is used.

ATag – *tag* of the item. Use SetCurrentTag[382] or SetCurrentBreakInfo[375] to change *tag* of item as an editing operation.

Instead of this method, you can use GetCurrentItemExtraIntPropertyEx[346] and GetCurrentTag[349] methods.

This method must be called only when the document is formatted.

Additional properties of item at the position of caret are returned by the methods GetCurrentItemExtraIntProperty[346] and GetCurrentItemExtraStrProperty[346].

See also methods:

- SetCurrentBreakInfo[375] (changes properties of *break* at the position of caret, as an editing operation);
- GetBreakInfo[218] (returns properties of the specified *break*).

See also properties:

- CurItemNo[322];
- CurItemStyle[323].

See also:

- Obtaining RichView items[84];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.35 TRichView Edit.GetCurrentBulletInfo

Returns main properties for the item of *bullet*[128] type at the position of caret.

```
procedure GetCurrentBulletInfo(var AName: TRVUnicodeString[694];
  var AImageIndex: Integer; var AImageList: TCustomImageList;
  var ATag: TRVTag[692]);
```

GetCurrentBulletInfo(...) is equivalent to TopLevelEditor[329].**GetBulletInfo**[219](TopLevelEditor[329].CurItemNo[322], ...).

Output parameters:

AName – name of *bullet*[128]. It can also be read using GetCurrentItemText[347] method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList**. It can also be read using GetCurrentItemExtraIntPropertyEx[346] method.

ATag – *tag* of the item. Use SetCurrentTag[382] or SetCurrentBulletInfo[375] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[349] method.

This method must be called only when the document is formatted.

Additional properties of item at the position of caret are returned by the methods GetCurrentItemExtraIntProperty[346] and GetCurrentItemExtraStrProperty[346].

See also methods:

- SetCurrentBulletInfo 375 (changes properties of *bullet* at the position of caret, as an editing operation);
- GetBulletInfo 219 (returns properties of the specified *bullet*).

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Obtaining RichView items 84;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.36 TRichView Edit.GetCurrentCheckpoint

Returns *checkpoint* associated with the item at position of caret, or *nil* if this item does not have associated *checkpoint*.

```
function GetCurrentCheckpoint: TCheckPointData 667;
```

The same as TopLevelEditor 329.**GetItemCheckpoint** 226(TopLevelEditor 329.CurItemNo 322).

This method must be called only when the document is formatted.

See also methods:

- SetCurrentCheckpointInfo 376 (changes *checkpoint* of item at the position of caret, as an editing operation);
- GetItemCheckpoint 226 (returns *checkpoint* of the specified item).

See also properties:

- CurItemNo 322.

See also:

- "Checkpoints" 65.

1.5.1.2.2.37 TRichView Edit.GetCurrentControlInfo

Returns main properties for the item of inserted control 126 type at position of caret.

```
procedure GetCurrentControlInfo(var AName: TRVUnicodeString 694;
  var Actrl: TControl; var AVAlign: TRVVAlign 695;
  var ATag: TRVTag 692);
```

(changed in version 18)

GetCurrentControlInfo(...) is equivalent to TopLevelEditor 329.**GetControlInfo** 222(TopLevelEditor 329 .CurItemNo 322, ...).

Output parameters:

AName – name of control item 126. Do not confuse with **Actrl**.Name property! This value can also be read using GetCurrentItemText 347 method.

Actr – the control itself. This method returns control owned by RichView, do not not destroy it.

AVAlign – vertical alignment of the control.

ATag – *tag* of the item. Do not confuse with **Actrl**.Tag property!. Use SetCurrentTag 382 or SetCurrentControlInfo 377 to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag 349 method.

This method must be called only when the document is formatted.

Additional properties of item at the position of caret are returned by the methods GetCurrentItemExtraIntProperty 346 and GetCurrentItemExtraStrProperty 346.

See also methods:

- SetCurrentControlInfo 377 (changes properties of control item at the position of caret, as an editing operation);
- GetControlInfo 222 (returns properties of the specified control item).

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Obtaining RichView items 84;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.38 TRichView Edit.GetCurrentHotspotInfo

Returns main properties for the item of *hotspot* 129 type at the position of caret.

```
procedure GetCurrentHotspotInfo(var AName: TRVUnicodeString 694;
  var AImageIndex, AHotImageIndex: Integer;
  var AImageList: TCustomImageList;
  var ATag: TRVTag 692);
```

(changed in version 18)

GetCurrentHotspotInfo(...) is equivalent to TopLevelEditor 329.**GetHotspotInfo** 224(TopLevelEditor 329 .CurItemNo 322, ...).

Output parameters:

AName – name of *hotspot* 129. It can also be read using GetCurrentItemText 347 method.

AImageList – image list (a reference to image list, do not free it).

AImageIndex – index in **AImageList** for normal image. It can also be read using GetCurrentItemExtraIntPropertyEx 346 method.

AHotImageIndex – index in **AImageList** for "hot" image. This image is displayed under the mouse pointer (in hypertext mode), or when user moves the caret to this item (in TRichViewEdit). It can also be read using GetCurrentItemExtraIntPropertyEx 346 method.

ATag – *tag* of the item. Use SetCurrentTag 382 or SetCurrentHotspotInfo 377 to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag 349 method.

This method must be called only when the document is formatted.

Additional properties of item at the position of caret are returned by the methods GetCurrentItemExtraIntProperty 346 and GetCurrentItemExtraStrProperty 346.

See also methods:

- SetCurrentHotspotInfo 377 (changes properties of *hotspot* at the position of caret, as an editing operation);
- GetHotspotInfo 224 (returns properties of the specified *hotspot*).

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Obtaining RichView items 84;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.39 TRichView Edit.GetCurrentItem

Returns object representing item at position of caret.

```
function GetCurrentItem: TCustomRVItemInfo;
```

(introduced in version 1.4)

This method is usually used when working with tables, labels, sequences, footnotes, endnotes, references to notes.

The table object is returned if the caret is to the left or to the right of the table, or if several table cells are selected. If the caret is inside table cell, this method returns item contained in this cell. To get the table in this case, use GetCurrentItemEx instead of GetCurrentItem.

This method must be called only when the document is formatted.

When called for an empty editor (i.e. ItemCount=0), the method returns *nil*.

See also methods:

- GetCurrentItemEx;
- InsertItem;
- GetItem.

See also properties:

- CurItemNo;
- CurItemStyle.

See also:

- Obtaining RichView items;
- Tables in RichView.

1.5.1.2.2.40 TRichView Edit.GetCurrentItemEx

Returns object at the position of caret representing item of the given class.

```
function GetCurrentItemEx(RequiredClass: TCustomRVItemInfoClass;
  var ItemRichViewEdit: TCustomRichViewEdit;
  var Item: TCustomRVItemInfo): Boolean;
```

(introduced in version 1.4)

This method is usually used when working with tables.

Input parameter:

RequiredClass – the required class of item, set it to TRVTableItemInfo.

Output parameters:

ItemRichViewEdit – editor containing this item (table); it can be the main editor, or cell inplace-editor (if this item (table) is in cell of another table).

Item – object representing the item (table)

This method must be called only when the document is formatted.

Return value

The method returns *True* if the output parameters were assigned. In case of tables, it returns *True* if:

- the caret is to the right or to the left of the table, or
- the table has multicell selection, or
- the caret is inside table cell.

In case of nested tables, this method returns the top level table (the closest to the user in Z-order).

See also methods:

- GetCurrentItem[345];
- InsertItem[353];
- GetItem[225].

See also:

- Obtaining RichView items[84];
- Tables in RichView[145].

1.5.1.2.2.41 TRichView Edit.GetCurrentItemExtraIntProperty -Ex

The methods return a value of the specified integer property of the item at the position of caret

```
function GetCurrentItemExtraIntProperty(
  Prop: TRVExtraItemProperty[673]; var Value: Integer): Boolean;
function GetCurrentItemExtraIntPropertyEx(
  Prop: Integer; var Value: Integer): Boolean;
```

(introduced in versions 1.7 and 15)

GetCurrentItemExtraIntProperty[Ex](...) is equivalent to TopLevelEditor[329].**GetItemExtraIntProperty[Ex]**[227] (TopLevelEditor[329].CurItemNo[322], ...).

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[673]. See information about this type for the list of properties.

In **GetCurrentItemExtraIntPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemProperty, **GetCurrentItemExtraIntPropertyEx** works like **GetCurrentItemExtraIntProperty**. In addition, it supports properties identified by rveipc*** constants[706]

These methods must be called only when the document is formatted.

Return value

True, if this item has this property. *False*, if not.

See also

- SetCurrentItemExtraIntProperty[Ex][378] (set a value of an integer property for the item at the position of caret, as an editing operation);
- SetItemExtraIntProperty[Ex]Ed[383] (set a value of an integer property for the specified item, as an editing operation);
- GetCurrentItemExtra[346]**Str**[346]Property[346] (return a value of the specified *string* property of the item at the position of caret).

See also methods of TRichView

- GetItemExtraIntProperty[Ex][227].

1.5.1.2.2.42 TRichView Edit.GetCurrentItemExtraStrProperty -Ex

The methods return a value of the specified string property of the item at the position of caret

```
function GetCurrentItemExtraStrProperty(
  Prop: TRVExtraItemStrProperty[676]; var Value: TRVUnicodeString[694]): Boolean;
function GetCurrentItemExtraStrProperty(
  Prop: Integer; var Value: TRVUnicodeString[694]): Boolean;
```

(introduced in versions 1.9 and 15; changed in version 18)

GetCurrentItemExtraStrProperty[Ex](...) is equivalent to TopLevelEditor[329].**GetItemExtraStrProperty[Ex]**[228] (TopLevelEditor[329].CurItemNo[322], ...).

Value receives value of the property identified by **Prop**.

In **GetCurrentItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[676]. See information about this type for the list of properties.

In **GetCurrentItemExtraStrPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemStrProperty, **GetCurrentItemExtraStrPropertyEx** works like **GetCurrentItemExtraStrProperty**. In addition, it supports properties identified by rvespc*** constants[709]

This method must be called only when the document is formatted.

Return value

True, if this item has this property. *False*, if not.

See also:

- SetCurrentItemExtraStrProperty[Ex][379] (set a value of a string property for the item at the position of caret, as an editing operation);
- SetItemExtraStrProperty[Ex]Ed[384] (set a value of a string property for the specified item, as an editing operation);
- GetCurrentItemExtra[346]**Int**[346]Property[Ex][346] (return a value of the specified *integer* property of the item at the position of caret).

See also methods of TRichView:

- GetItemExtraStrProperty[Ex][228].

1.5.1.2.2.43 TRichView Edit.GetCurrentItemText -A -W

Returns text for the item at the position of caret.

```
function GetCurrentItemText: String;
function GetCurrentItemTextA: TRVAnsiString[668];
function GetCurrentItemTextW: TRVUnicodeString[694];
```

(Introduced in version 1.4, 1.7)

For text items, these methods return visible text. For non-text items, they return item name.

- **GetCurrentItemText** is the same as TopLevelEditor[329].**GetItemText**[230](TopLevelEditor[329].CurItemNo[322]).
- **GetCurrentItemTextA** is the same as TopLevelEditor[329].**GetItemTextA**[230](TopLevelEditor[329].CurItemNo[322]).
- **GetCurrentItemTextW** is the same as TopLevelEditor[329].**GetItemTextW**[230](TopLevelEditor[329].CurItemNo[322]).

Unicode notes:

Internally, text is stored as Unicode. **GetCurrentItemTextA** (and **GetCurrentItemText** in non-Unicode versions of Delphi) converts Unicode to ANSI. For a text item, a code page for conversion is calculated basing on the its Charset[468]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[193].DefCodePage[421] is used.

These methods must be called only when the document is formatted.

See also methods:

- SetCurrentItemText -A -W[380].

See also methods of TCustomRichView:

- GetItemText -A -W[230].

See also properties:

- CurItemNo[322];

- CurItemStyle 323.

See also:

- Obtaining RichView items 84;
- Item types 64;
- "Tags" 68;
- Unicode in RichView 99.

1.5.1.2.2.44 TRichView Edit.GetCurrentItemVAlign

Returns vertical alignment (position relative to the line) of the item at the caret position

```
function GetCurrentItemVAlign: TRVVAlign695;
```

(introduced in version 12)

The same as TopLevelEditor 329.**GetItemVAlign** 230(TopLevelEditor 329.CurItemNo 322).

This method must be called only when the document is formatted.

This method must not be called if the current item is one of: text items 120, tables 130, *breaks* 125, tabs 121, list markers 131, footnotes 136 and endnotes 135.

See also methods:

- SetCurrentItemVAlign 380.

See also properties:

- CurItemNo 322.

1.5.1.2.2.45 TRichView Edit.GetCurrentLineCol

Returns a line and column at the caret position.

```
procedure GetCurrentLineCol(var Line, Column: Integer);
```

(introduced in version 1.7)

Number of lines depends on word-wrapping.

Number of the first line is 1. Number of the leftmost caret position is 1.

Lines in table cells are calculated relative to the beginning of the cell.

Warning: this function contains calculations inside, and it may take some time to execute it

(There is an idea how to make it lightning-fast in future, but it will be implemented later)

This method must be called only when the document is formatted.

See also methods of TCustomRichView:

- GetLineNo 233.

1.5.1.2.2.46 TRichView Edit.GetCurrentMisspelling

Returns misspelled word in the position of caret, optionally selects it.

```
function GetCurrentMisspelling(SelectIt: Boolean;
  var Word: TRVUnicodeString694; var StyleNo: Integer): Boolean;
```

(changed in version 18)

Input parameters:

SelectIt – if *True*, the returned word will be selected in the editor.

Output parameters:

Word receives misspelled word.

StyleNo receives text style of misspelled word (index in TextStyles[437] collection of the linked[193] TRVStyle component).

The output parameters are defined only if this method returns *True*.

Return value

True if misspelling is found.

See also:

- Live spelling check in RichView[101].

1.5.1.2.2.47 TRichView Edit.GetCurrentPictureInfo

Returns main properties for the item of picture[122]/*hot-picture*[125] type at the position of caret.

```
procedure GetCurrentPictureInfo(var AName: TRVUnicodeString[694]; var Agr: TGraphic;
  var AVAlign: TRVVAlign[695]; var ATag: TRVTag[692]);
```

(changed in version 18)

GetCurrentPictureInfo(...) is equivalent to TopLevelEditor[329].**GetPictureInfo**[236](TopLevelEditor[329].CurItemNo[322], ...).

Output parameters:

AName – name of the item. It can also be read using GetCurrentItemText[347] method.

Agr – graphic object. This method returns object owned by RichViewEdit, do not not destroy it.

AVAlign – vertical alignment of the picture.

ATag – *tag* of the item. Use SetCurrentTag[382] or SetCurrentPictureInfo[381] to change *tag* of item as an editing operation. This value can also be read using GetCurrentTag[349] method.

This method must be called only when the document is formatted.

Additional properties of item at the position of caret are returned by the methods GetCurrentItemExtraIntProperty[346] and GetCurrentItemExtraStrProperty[346].

See also methods:

- SetCurrentPictureInfo[381] (changes properties of picture at the position of caret, as an editing operation);
- GetPictureInfo[236] (returns properties of the specified picture).

See also properties:

- CurItemNo[322];
- CurItemStyle[323].

See also:

- Obtaining RichView items[84];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.48 TRichView Edit.GetCurrentTag

Returns tag of the item at the caret position

```
function GetCurrentTag: TRVTag[692];
```

The same as TopLevelEditor[329].**GetItemTag**[230](TopLevelEditor[329].CurItemNo[322]).

Use SetCurrentTag[382] to change tag of item at the caret position, as an editing operations.

This method must be called only when the document is formatted.

See also methods:
- SetCurrentTag[382].

See also properties:
- CurItemNo[322].

See also
- "Tags"[68].

1.5.1.2.2.49 TRichView Edit.GetCurrentTextInfo

Returns text and *tag* for text item at the position of caret

```
procedure GetCurrentTextInfo(var AText: TRVUnicodeString[694]; var ATag: TRVTag[692]);
```

GetCurrentTextInfo(...) is equivalent to TopLevelEditor[329].**GetTextInfo**[239](TopLevelEditor[329].CurItemNo[322], ...).

Use SetCurrentTag[382] to change tag of item at the caret position, as an editing operations.

This method must be called only when the document is formatted.

See also properties:
- CurItemNo[322];
- CurItemStyle[323].

See also:
- Obtaining items[84];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.50 TRichView Edit.InsertBreak

Inserts *break*[125] (horizontal line) in the position of caret.

```
function InsertBreak(Width: Byte; Style: TRVBreakStyle[669]; Color: TColor): Boolean;
```

Parameters:

Width – width of line (can be in range from 1 to 10);

Style – break type;

Color – color of line. If it is equal to *clNone*, the *break* has the color of the 0th text style[437].

Method type: editing-style method for insertion.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also methods of TCustomRichView:
- AddBreak[196].

See also:
- Inserting items in position of caret[86];
- Item types[64].

1.5.1.2.2.51 TRichView Edit.InsertBullet

Inserts *bullet* [128] in the position of caret.

```
function InsertBullet(ImageIndex: Integer; ImageList: TCustomImageList): Boolean;
```

Parameters:

Name – name of *bullet*, any string. **Name** must not contain CR and LF characters. RichViewEdit does not use item names itself, they are for your own use. This is an ANSI string, regardless of Unicode mode of text styles.

ImageIndex – index of image in **ImageList**.

ImageList – image list for this *bullet*. RichView does not own, does not copy and does not destroy image lists, it just holds pointers to them. So this image list is not destroyed when the document is cleared.

Method type: editing-style method for insertion.

Unicode note: names of non-text items are ANSI strings.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also methods of TCustomRichView:

- AddBullet [197];

See also:

- Inserting items in position of caret [86];
- Item types [64].

1.5.1.2.2.52 TRichView Edit.InsertCheckpoint

Inserts *checkpoint* in the position of caret.

```
procedure InsertCheckpoint(const ATag: TRVTag [692]; const AName: TRVUnicodeString [694];
 ARaiseEvent: Boolean);
```

(introduced in version 10; changed in version 18)

This method is similar to SetCurrentCheckpointInfo [376], but inserts *checkpoint* exactly in the position of caret.

If the caret is at the beginning of item, this methods calls SetCheckpointInfoEd [373] to add *checkpoint* for this item.

If the caret is at the middle of text item [120], this method splits this item at the position of caret, and calls SetCheckpointInfoEd [373] to add *checkpoint* for the item to the right.

If the caret is at the end of paragraph, the method moves it to the beginning of the next paragraph, and calls SetCheckpointInfoEd [373] to add *checkpoint* for the first paragraph item (or the second item, if the first one is list marker [131]).

If the caret is at the end of document, this methods does nothing.

This method is used in conjunction with GetCheckpointAtCaret [341] and RemoveCheckpointAtCaret [366] (GetCurrentCheckpoint [343] and RemoveCurrentCheckpoint [367] cannot be used, because *checkpoint* added by this method is not necessary belongs to the current item (that is usually the item to the left of the caret).

Parameters.

ATag – *tag* of *checkpoint*;

AName – name of the *checkpoint*, any string without line break (CR, LF) characters.

ARaiseEvent – "raise event" flag; if set, RichView can generate OnCheckpointVisible[274] event for this *checkpoint*.

Method type: editing-style.

See also:

- Modifying RichView items[85];
- "Checkpoints"[65];
- "Tags"[68].

1.5.1.2.2.53 TRichView Edit.InsertControl

Inserts Delphi/C++Builder control[126] in the position of caret.

```
function InsertControl(const Name: TRVUnicodeString[694]; ctrl: TControl;
  VAlign: TRVVAlign[695]): Boolean;
```

(changed in version 18)

Parameters:

Name – name of this control item, any string. **Name** must not contain CR and LF characters. TRichView does not use item names itself, they are for your own use. Do not confuse with **ctrl**.Name property.

ctrl – control to insert. You should create control but should not destroy it (RichViewEdit will do it for you).

VAlign – vertical align of this control, relative to its line, see TRVVAlign[695] for possible values.

Method type: editing-style method for insertion.

Return value

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also methods of TCustomRichView:

- AddControl[198];

See also:

- Inserting items in position of caret[86];
- Item types[64].

1.5.1.2.2.54 TRichView Edit.InsertHotPicture

Inserts picture with hypetext link[125] in the position of caret.

```
function InsertHotPicture(const Name: TRVUnicodeString[694]; gr: TGraphic;
  VAlign: TRVVAlign[695]): Boolean;
```

(introduced in version 1.5)

The same as InsertPicture[355], but this picture will be a hypertext link.

Method type: editing-style method for insertion.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection). If insertion failed, TRichViewEdit destroys **gr**.

See also methods of TCustomRichView:

- AddHotPicture[199].

See also:

- Inserting items in position of caret[86];
- Hypertext in RichView[69];
- Item types[64].

1.5.1.2.2.55 TRichView Edit.InsertHotspot

Inserts *hotspot*[129] in the position of caret.

```
function InsertHotspot(ImageIndex, HotImageIndex: Integer;
  ImageList: TCustomImageList): Boolean;
```

Parameters:

Name – name of *hotspot*, any string. **Name** must not contain CR and LF characters. RichViewEdit does not use item names itself, they are for your own use.

ImageIndex – index of image in **ImageList**.

HotImageIndex – index of "hot" image in **ImageList**. It is displayed when the mouse pointer is above this *hotspot*.

ImageList – image list for this *bullet*. RichView does not own, does not copy and does not destroy image lists, it just holds pointers to them. So this image list is not destroyed when the document is cleared.

Method type: I editing-style method for insertion.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also methods of TCustomRichView:

- AddHotspot[200].

See also:

- Inserting items in position of caret[86];
- Hypertext in RichView[69];
- Item types[64].

1.5.1.2.2.56 TRichView Edit.InsertItem

Inserts item in the position of caret.

```
function InsertItem(const Name: TRVUnicodeString[694];
  Item: TCustomRVItemInfo[563]): Boolean;
```

(introduced in version 1.4; changed in version 18)

This is the most general method for inserting items. It can be used to insert items of custom types.

This method is usually used for inserting tables[130], labels[133], sequences[134], footnotes[136], endnotes[135], references to notes[140], page numbers[141], page counts[142], equations[143] (i.e. for items that does not have special insertion methods).

The detailed description of item types is beyond the scope of this help file. Briefly, standard item types are declared and implemented in RVItem unit. All item types are classes derived from one base class – TCustomRVItemInfo.

Parameters:

Text – some text associated with this item (the name of item). It must not contain CR and LF characters.

Item – item itself. **Item**.ParaNo defines paragraph style of item (index in Style.ParaStyles[431]).

Method type: editing-style method for insertion.

Return value:

If *True*, the item was inserted successfully.

If *False*, the item can not be inserted (because text is read-only[328] or protected[473], or when inserting in table having multicell selection); in this case **Item** was destroyed (freed) by InsertItem.

See also methods of TCustomRichView:

- AddItem[201].

See also:

- Inserting items in position of caret[86];
- Item types[64].

1.5.1.2.2.57 TRichView Edit.InsertOEMTextFromFile

Loads text file (in OEM-defined character set) **FileName** and inserts its content in the position of caret.

```
function InsertOEMTextFromFile(const FileName: TRVUnicodeString[694]):Boolean;
```

(changed in version 18)

"OEM" stands for "original equipment manufacturer". This is character set of MS DOS.

The inserted text (one or more text items) has current text and paragraph styles.

Method type: editing-style method for insertion.

Unicode note: File content must be in OEM charset. Text will be converted to Unicode.

Return value:

"Was reading from from file successful?"

See also methods:

- InsertTextFromFile[361];
- InsertText[360].

See also properties:

- CurParaStyleNo[323];
- CurTextStyleNo[324].

See also properties of TRVStyle:

- TextStyles[437];
- ParaStyles[431];
- SpacesInTab[435].

See also:

- Saving and loading[92];
- Inserting items in position of caret[86].

1.5.1.2.2.58 TRichView Edit.InsertPageBreak

Inserts explicit page break in the position of caret.

```
procedure InsertPageBreak
```

(introduced in version 1.2)

The method deselects the document before performing its work.

If the caret is not at the beginning of some item which is always displayed from the new line (see IsFromNewLine[242]), the method breaks the current line (just like if user pressed Enter key).

The method sets "explicit page break" flag for the item in the position of caret (see PageBreaksBeforeItems[184]).

Method type: editing-style.

See also:

- RemoveCurrentPageBreak[367].

See also properties of RichView:

- PageBreaksBeforeItems[184];
- Options[182] (*rvoShowPageBreaks*).

See also properties of RVStyle:

- PageBreakColor[431].

1.5.1.2.2.59 TRichView Edit.InsertPicture

Inserts picture[122] in the position of caret.

```
function InsertPicture(const Name: TRVUnicodeString[694]; gr: TGraphic;
  VAlign: TRVVAlign[695]): Boolean;
```

(changed in version 18)

Parameters:

Name – name of this picture item, any string. **Name** must not contain CR and LF characters. TRichView does not use items names itself, they are for your own use.

gr – picture to insert. By default, this picture will be owned by TRichView control, and you must not free it. However, you can specify nonzero *rvepShared* extra item property[673], and this graphic object will be shared (you need to free it yourself after TRichView is cleared).

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount[702]). Shared images are never deactivated.

VAlign – vertical align of this picture, relative to its line, see TRVVAlign[695] for possible values.

Method type: editing-style method for insertion.

If you want this picture to be a hypertext link, use InsertHotPicture[352] instead.

Return value

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection). If insertion failed, TRichViewEdit destroys **gr**.

Example (inserting icon from file)

```
var ico: TIcon;
...
ico := TIcon.Create;
ico.LoadFromFile('My Icon.ico');
```

```
MyRichViewEdit.InsertPicture('',ico, rvvaBaseline);
```

Example 2 (inserting picture of any supported format):

```
var gr: TGraphic;
...
if OpenPictureDialog1.Execute then
  gr := RVGraphicHandler[710].LoadFromFile[654](OpenPictureDialog1.FileName);
  if gr<>nil then
    MyRichViewEdit.InsertPicture('', gr, rvvaBaseline)
  else
    // process a picture loading error here
end;
```

See also methods of TCustomRichView:

- AddPicture[204].

See also:

- Inserting items in position of caret[86];
- Hypertext in RichView[69];
- Item types[64].

1.5.1.2.2.60 TRichViewEdit.InsertRTFFromFileEd

Inserts RTF (Rich Text Format) from file **FileName** in the position of caret.

```
function InsertRTFFromFileEd(const FileName: TRVUnicodeString[694]): Boolean;
```

(Introduced in version 1.5; changed in version 18)

Parameters for loading are in TRichView.RTFReadProperties[187]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[309] and ParaStyleMode[308]. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor[168].

Method type: editing-style method for insertion.

If style templates are used[195], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style[193].StyleTemplates[435], and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

Return value:

"successful reading?" (extended information is in TRichView.RTFReadProperties.ErrorCode[306])

See also methods of TRichViewEdit:

- InsertRTFFromStreamEd[357];
- PasteRTF[363].

See also methods of TRichView:

- LoadRTF[243].

See also events of TRichView:

- OnImportPicture[279];
- OnReadHyperlink[284];
- OnStyleTemplatesChange[302],

See also:

- Saving and loading[92];
- Inserting items in position of caret[86].

1.5.1.2.2.61 TRichView Edit.InsertRTFFromStreamEd

Inserts RTF (Rich Text Format) from the **Stream** in the position of caret.

```
function InsertRTFFromStreamEd(Stream: TStream): Boolean;
```

(Introduced in version 1.5)

Parameters for loading are in TRichView.RTFReadProperties[187]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[309] and ParaStyleMode[308]. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor[168].

Method type: editing-style method for insertion.

If style templates are used[195], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style[193].StyleTemplates[435], and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

Return value:

"successful reading?" (extended information is in TRichView.RTFReadProperties.ErrorCode[306])

See also methods of TRichViewEdit:

- InsertRTFFromFileEd[356];
- PasteRTF[363].

See also methods of TRichView:

- LoadRTFFromStream[244].

See also events of TRichView:

- OnImportPicture[279];
- OnReadHyperlink[284];
- OnStyleTemplatesChange[302].

See also:

- Saving and loading[92];
- Inserting items in position of caret[86].

1.5.1.2.2.62 TRichView Edit.InsertRVFFromFileEd

Inserts RVF (RichView Format[93]) from the file **FileName** in the position of caret.

```
function InsertRVFFromFileEd(const FileName: TRVUnicodeString[694]):Boolean;
```

(changed in version 18)

Settings for RVF loading can be changed in the TRichView component editor[168].

Method type: editing-style method for insertion.

If style templates are used[195], and RVFTextStylesReadMode[191]=RVFParaStylesReadMode[190] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style[193].StyleTemplates[435], and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

Return value: "successful reading?"

See also methods of TRichViewEdit:

- InsertRVFFromStreamEd[358];
- PasteRVF[364].

See also methods of TRichView:

- LoadRVF 245.

- **See also properties:**

- RVFOptions 188;
- RVFWarnings 191;
- RVFTextStylesReadMode 191 (this method type: RVF insertion);
- RVFParaStylesReadMode 190 (this method type: RVF insertion);
- UseStyleTemplates 195;
- StyleTemplateInsertMode 193.

See also events:

- OnRVFImageListNeeded 288;
- OnRVFControlNeeded 288;
- OnRVFPictureNeeded 289;
- OnControlAction 275 (ControlAction=*rvcaAfterRVFLoad*).

See also:

- TRichView component editor 168;
- Saving and loading 92;
- Inserting items in position of caret 86;
- RVF 93.

1.5.1.2.2.63 TRichViewEdit.InsertRVFFromStreamEd

Inserts RVF (RichView Format 93) from the **Stream** in the position of caret.

```
function InsertRVFFromStreamEd(Stream: TStream):Boolean;
```

Settings for RVF loading can be changed in the TRichView component editor 168.

Method type: editing-style method for insertion.

If style templates are used 195, and RVFTextStylesReadMode 191=RVFParaStylesReadMode 190 =*rvf_sInsertMerge*, and StyleTemplateInsertMode 193<>*rvstimIgnoreSourceStyleTemplates*, the method merges style templates from RVF into Style 193.StyleTemplates 435, and reads text and paragraph styles according to StyleTemplateInsertMode 193. The method calls OnStyleTemplatesChange 302 event.

Return value: "successful reading?"

See also methods of TRichViewEdit:

- InsertRVFFromFileEd 357;
- PasteRVF 364.

See also methods of TRichView:

- LoadRVFFromStream 244;
- InsertRVFFromStream 240;
- AppendRVFFromStream 208.

- **See also properties:**

- RVFOptions 188;
- RVFWarnings 191;
- RVFTextStylesReadMode 191 (this method type: RVF insertion);
- RVFParaStylesReadMode 190 (this method type: RVF insertion);
- UseStyleTemplates 195;
- StyleTemplateInsertMode 193.

See also events:

- OnRVFImageListNeeded 288;
- OnRVFControlNeeded 288;
- OnRVFPictureNeeded 289;

- OnControlAction 275 (ControlAction=*rvcaAfterRVFLoad*);
- OnStyleTemplatesChange 302.

See also:

- TRichView component editor 168;
- Saving and loading 92;
- Inserting items in position of caret 86;
- RVF 93.

1.5.1.2.2.64 TRichView Edit.InsertStringTag -A -W

The methods Insert text string **s** with associated **Tag** in the position of caret.

```
function InsertStringTag(const s: String; const Tag: TRVTag 692): Boolean;
function InsertStringATag(const s: TRVAnsiString 668; const Tag: TRVTag 692): Boolean;
function InsertStringWTag(const s: TRVUnicodeString 694; const Tag: TRVTag 692): Boolean;
```

(introduced in version 1.6)

Inserted text (one text item) has the current text 324 and current paragraph 323 style.

Methods type: editing-style method for insertion.

Unlike InsertText 360:

- **s** parameter must not contain CR, LF characters (#13 and #10) and page break character (#12);
- TAB characters are always replaced with spaces, even if SpacesInTab 435=0.
- the inserted string is not merged with surrounding text items, even if it has the same text style and *tag*.

Do not overuse these methods. Use InsertText 360 when possible.

Unicode note:

Internally, text is stored as Unicode. **InsertStringATag** (and **InsertStringTag** in non-Unicode versions of Delphi) converts ANSI to Unicode using Style 193.DefCodePage 421.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also methods:

- InsertText, -A, -W 360.

See also properties:

- CurParaStyleNo 323;
- CurTextStyleNo 324.

See also properties of TRVStyle:

- TextStyles 437;
- ParaStyles 431;
- SpacesInTab 435.

See also:

- Inserting items in position of caret 86;
- Tags 68.

1.5.1.2.2.65 TRichView Edit.InsertTab

Inserts tabulator 121 in the position of caret.

```
function InsertTab: Boolean;
```

(introduced in v1.9)

Inserted tabulator has the current text 324 and the current paragraph 323 style.

This method always inserts a tabulator, regardless of RVStyle.SpacesInTab[435] mode.

If RVStyle.SpacesInTab[435]=0, this method does the same work as InsertText[360](#9).

Method type: editing-style method for insertion.

Return value:

True if the insertion was successful (it can fail due to protection, or when inserting in table having multicell selection)

See also method of TRichView:

- AddTab[205].

1.5.1.2.2.66 TRichView Edit.InsertText -A -W

The methods insert **text** in the position of caret.

```
procedure InsertText(const text: String; CaretBefore: Boolean=False);
procedure InsertTextA(const text: TRVAnsiString[668]; CaretBefore: Boolean=False);
procedure InsertTextW(const text: TRVUnicodeString[694]; CaretBefore: Boolean=False);
```

Inserted text has current text[324] and current paragraph[323] style.

Text may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[121], depending on value of SpacesInTab[435] property of the linked[193] RVStyle component. #12 characters add page breaks[184]. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported.

InsertTextW supports the Unicode byte order marks characters (if it's present, it must be the first character in **text**).

If **CaretBefore**=*True*, the caret will be positioned before the inserted text after insertion (default positioning is after the text). Set it to *True* when performing search-and-replace in down-up direction (note: the standard replace dialog does not allow to choose down-up direction) to avoid infinite loop (when replacing, for example, 'a' with 'aa'). The "smart start" search option helps to solve this problem; however, this advice is still in effect.

Methods type: editing-style method for insertion.

Unicode note:

Internally, text is stored as Unicode. **InsertTextA** (and **InsertText** in non-Unicode versions of Delphi) converts ANSI to Unicode using Style[193].DefCodePage[421].

See also methods:

- InsertStringTag, InsertStringATag, InsertStringWTag[359].

See also properties:

- CurParaStyleNo[323];
- CurTextStyleNo[324].

See also properties of TRVStyle:

- TextStyles[437];
- ParaStyles[431];
- SpacesInTab[435].

See also:

- Inserting items in position of caret[86].

1.5.1.2.2.67 TRichView Edit.InsertTextFromFile -W

The methods load text file **FileName** and insert its content in the position of caret. **InsertTextFromFile** inserts text from ANSI text files, **InsertTextFromFileW** inserts text from Unicode (UTF-16) text files.

```
function InsertTextFromFile(const FileName: TRVUnicodeString[694]; CodePage: Cardinal=CP_ACP
function InsertTextFromFileW(const FileName: TRVUnicodeString[694]): Boolean;
```

Inserted text has current text[324] and current paragraph[323] style.

A file may contain special characters:CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[121], depending on value of SpacesInTab[435] property of the linked[193] RVStyle component. #12 characters add page breaks[184]. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported.

InsertTextFromFileW supports the Unicode byte order marks characters (if they are present, they must be the first character in file).

Methods type: editing-style method for insertion.

Unicode note:

Internally, text is stored as Unicode. **InsertTextFromFile** converts ANSI to Unicode using **CodePage**.

Return value: "Was reading from from the file successful?"

See also methods:

- InsertOEMTextFromFile[354];
- InsertText, -A, -W[360].

See also methods of TRichView:

- LoadText, LoadTextW[246].

See also properties:

- CurParaStyleNo[323];
- CurTextStyleNo[324].

See also properties of TRVStyle:

- TextStyles[437];
- ParaStyles[431];
- SpacesInTab[435].

See also:

- Saving and loading[92];
- Inserting items in position of caret[86].

1.5.1.2.2.68 TRichView Edit.Paste

Pastes document from the Clipboard.

```
procedure Paste;
```

First, it generates OnPaste[399] event, allowing the programmer to insert data in custom format.

If DoDefault parameter of OnPaste was set to *False*, Paste finishes.

The method takes AcceptPasteFormats[321] property into account: it tries to paste the specific format only if it is included in the property.

Next, it tries to paste as RVF, see PasteRVF[364].

If unsuccessful, it tries to paste as RTF, see PasteRTF[363].

If unsuccessful, it tries to paste as an URL, see PasteURL[365].

If unsuccessful, it tries to paste as text, see PasteTextW[364].

If unsuccessful, it tries to paste image files, see PasteGraphicFile[362] (if *rvoAssignImageFileNames* in Options[182], file name is is stored in *rvespImageFileName* extra string property[676]; you can modify this file name in OnAssignImageFileName[274] event).

If unsuccessful, it tries to paste as a bitmap, see PasteBitmap[362].

If unsuccessful, it tries to paste as a metafile picture, see PasteMetafile[363].

This method is executed automatically when user presses Ctrl+V or Shift+Insert.

Method type: editing-style method for insertion.

See also methods:

- CanPaste[337].

See also:

- TRichView and Clipboard[82].

1.5.1.2.69 TRichView Edit.PasteBitmap

Inserts Windows bitmap from the Clipboard in the position of caret

```
function PasteBitmap(TextAsName: Boolean): Boolean;
```

This method does nothing if the Clipboard does not contain Windows bitmap;

It inserts a new picture item with bitmap (TBitmap) from the Clipboard; if **TextAsName**=*True* and the Clipboard contains text, it sets name of this item to the text from the Clipboard (warning: if you want to save this document to RVF, names of items should not contain CR and LF characters).

Method type: editing-style method for insertion.

Unicode note: names of non-text items are ANSI strings.

Return value:

True, if image was inserted.

See also methods:

- Paste[361];
- InsertPicture[355].

See also properties:

- DefaultPictureVAlign[325].

See also:

- RichView and Clipboard[82].

1.5.1.2.70 TRichView Edit.PasteGraphicFiles

Inserts graphic files from the Clipboard (CF_HDROP format).

```
function TCustomRichViewEdit.PasteGraphicFiles(
  FileNamesAsNames,StoreFileNames: Boolean): Boolean;
```

(introduced in version 10)

This method does nothing if the Clipboard does not contain graphic files in CF_HDROP format. This format is copied, for example, by Windows Explorer, or by Internet Explorer, when user copies an image (a link to the file in the browser's cache is copied).

Parameters:

If **FileNamesAsNames** = *True*, the full file name is stored in the picture item name (not recommended).

If **StoreFileNames** = *True*, the full file name is is stored in *rvespImageFileName* extra string property[676]. You can modify this file name in OnAssignImageFileName[274] event.

Method type: editing-style method for insertion.

Unicode note: names of non-text items and extra item string properties are ANSI strings.

You can use OnDropFile[392] event to insert files yourself (so you can implement insertion of files of multiple formats, not only graphics)

This method may be called automatically when pasting files, if the editor chooses to insert data from the Clipboard in this format. In this case, the method is called with **FileNamesAsNames** = *False*, **StoreFileNames** = (*rvoAssignImageFileNames* in Options[182]).

Return value:

True, if at least one of pictures was inserted.

See also methods:

- Paste[361];
- InsertPicture[355].

See also properties:

- DefaultPictureVAlign[325].

See also events:

- OnAssignImageFileName[274]

See also:

- RichView and Clipboard[82].

1.5.1.2.2.71 TRichView Edit.PasteMetafile

Inserts metafile from the Clipboard in the position of caret

```
function PasteMetafile(TextAsName: Boolean): Boolean;
```

This method does nothing if the Clipboard does not contain a Windows metafile.

It inserts a new picture item with metafile (TMetafile) from the Clipboard; if **TextAsName**=*True* and the Clipboard contains text, it sets name of this item to the text from the Clipboard (warning: if you want to save this document to RVF, names of items should not contain CR and LF characters).

Method type: editing-style method for insertion.

Unicode note: names of non-text items are ANSI strings.

Return value:

True, if image was inserted.

See also methods:

- Paste[361];
- InsertPicture[355].

See also properties:

- DefaultPictureVAlign[325].

See also:

- RichView and Clipboard[82].

1.5.1.2.2.72 TRichView Edit.PasteRTF

Inserts RTF (Rich Text Format) from the Clipboard in the position of caret

```
procedure PasteRTF;
```

(Introduced in version 1.5)

This method does nothing if the Clipboard does not contain RTF.

If the Clipboard has RTF, and there was an error during reading, you can get additional information in TRichView.RTFReadProperties.ErrorCode[306] property.

Parameters for loading are in TRichView.RTFReadProperties[187]. The most important properties affecting RTF loading are RTFReadProperties.TextStyleMode[309] and ParaStyleMode[308]. The imported RTF may look absolutely different depending on values of these properties! Setting for RTF loading can be changed in the TRichView component editor[168].

If the Clipboard contains HTML format as well, RichViewEdit reads base path for external images from it.

Method type: editing-style method for insertion.

If style templates are used[195], and RTFReadProperties.TextStyleMode=RTFReadProperties.ParaStyleMode=*rvrsAddIfNeeded*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the method merges an RTF style sheet into Style[193].StyleTemplates[435], and reads text and paragraph styles according to StyleTemplateInsertMode[193]. The method calls OnStyleTemplatesChange[302] event.

See also methods:

- CanPasteRTF[337];
- InsertRTFFromFileEd[356];
- InsertRTFFromStreamEd[357].

See also:

- RichView and Clipboard[82]

1.5.1.2.2.73 TRichView Edit.PasteRVF

Inserts RVF (RichView Format) from the Clipboard in the position of caret

```
procedure PasteRVF;
```

This method does nothing if the Clipboard does not contain RVF.

Method type: editing-style method for insertion.

See also methods:

- CanPasteRVF[337];
- Paste[361];
- InsertRVFFromStreamEd[358].

See also:

- RichView and Clipboard[82];
- RVF[93].

1.5.1.2.2.74 TRichView Edit.PasteText -A -W

These methods Insert text from the Clipboard in the position of caret

```
procedure PasteText;
procedure PasteTextA;
procedure PasteTextW;
```

PasteTextA pastes text in CF_TEXT format. **PasteText** and **PasteTextW** paste text in CF_UNICODETEXT.

In modern versions of Windows, Windows automatically converts Unicode and ANSI text in the Clipboard to each other, using the language of the keyboard layout that was used at the moment of copying, so using **PasteTextA** does not make sense.

These methods do nothing if the Clipboard does not contain text.

Text is inserted in the position of caret, using the current text[324] and paragraph[323] styles.

Text from the Clipboard may contain special characters: CR, LF, TAB, FF (#13, #10, #9, #12). #9 characters may be inserted as tabulators[121], depending on value of SpacesInTab[435] property of the linked[193] RVStyle

component. #12 characters add page breaks[184]. All possible line breaks modes (CR, LF, CR+LF, LF+CR) are supported. **PasteTextW** supports the Unicode byte order marks characters (if it's present, it must be the first character).

Methods type: editing-style method for insertion.

Unicode note:

- **PasteTextA:** itext from the Clipboard will be converted to Unicode (conversion is based on the charset of the current text style).

See also methods:

- Paste[361];
- InsertText, -A, -W[360].

See also properties:

- CurParaStyleNo[323];
- CurTextStyleNo[324].

See also properties of TRVStyle:

- TextStyles[437];
- ParaStyles[431];
- SpacesInTab[435].

See also:

- RichView and Clipboard[82].

1.5.1.2.2.75 TRichView Edit.PasteURL

Inserts URL from the Clipboard in the position of caret

```
function PasteURL: Boolean;
```

(introduced in version 14)

This method does nothing and returns *False* if the Clipboard does not contain URL.

The method calls OnReadHyperlink[284] event, passing the current text style in the StyleNo parameter. The application should return the corresponding hypertext style in this parameter (if the event is not processed, URL is inserted as a plain text with URL in its tag[68]).

Method type: editing-style method for insertion.

See also methods:

- Paste[361].

See also:

- RichView and Clipboard[82].

1.5.1.2.2.76 TRichView Edit.Redo

Redoes the last undone editing operation.

```
procedure Redo;
```

(Introduced in version 1.3)

This method does nothing if there is no operation to redo.

Information about the operation to redo is returned by the methods RedoAction[366] and RedoName[366].

This method must be called only when the document is formatted.

This method is called when the user presses Shift+Ctrl+Z or Shift+Alt+Backspace.

Method type: editing-style.

See also:
- Undo 387;
- Undo in RichViewEdit 88.

1.5.1.2.2.77 TRichView Edit.RedoAction

Returns type of the operation to be redone 365 next.

```
function RedoAction: TRVUndoType 693;
```

(Introduced in version 1.3)

See TRVUndoType 693 for the list of operations. If the returned value is *rvutCustom*, you can use RedoName 366.

See also:
- UndoAction 388;
- Undo in RichViewEdit 88.

1.5.1.2.2.78 TRichView Edit.RedoName

Returns name of the operation to be redone 365 next.

```
function RedoName: TRVUnicodeString 694;
```

(Introduced in version 1.3; changed in version 18)

Returns empty string for the standard undo/redo operations, i.e. all operations but *rvutCustom* (see RedoAction 366).

Returns non-empty string only for custom redo action, initiated by BeginUndoCustomGroup 336.

See also:
- UndoName 388;
- Undo in RichViewEdit 88.

1.5.1.2.2.79 TRichView Edit.RemoveCheckpointAtCaret

Deletes *checkpoint* at the position of caret.

```
procedure RemoveCheckpointAtCaret;
```

(introduced in version 10)

This method deletes *checkpoint* only if the caret is at the beginning of item.

This method deletes the *checkpoint* that is returned by GetCheckpointAtCaret 341.

This method is used in conjunction with InsertCheckpoint 351 and GetCheckpointAtCaret 341. The alternative set of methods is: SetCurrentCheckpointInfo 376, GetCurrentCheckpoint 343 and RemoveCurrentCheckpoint 367.

Method type: editing-style, unlike RemoveCheckpoint 248.

See also:
- "Checkpoints" 65.

1.5.1.2.2.80 TRichView Edit.RemoveCheckpointEd

Editing-style method for deleting *checkpoint* associated with the **ItemNo**-th item.

```
procedure RemoveCheckpointEd(ItemNo: Integer);
```

Method type: editing-style, unlike RemoveCheckpoint 248.

See also methods of TRichViewEdit:
- RemoveCurrentCheckpoint 367;

- SetCheckpointInfoEd 373.

See also methods of TRichView:

- GetItemCheckpoint 226;
- RemoveCheckpoint 248.

See also:

- Modifying RichView items 85;
- "Checkpoints" 65.

1.5.1.2.2.81 TRichView Edit.RemoveCurrentCheckpoint

Deletes *checkpoint* associated with the item at the position of caret.

```
procedure RemoveCurrentCheckpoint;
```

RemoveCurrentCheckpoint is equivalent to TopLevelEditor 329.RemoveCheckpointEd 366(TopLevelEditor 329.CurItemNo 322).

Method type: editing-style.

See also methods:

- GetCurrentCheckpoint 343;
- SetCurrentCheckpointInfo 376;
- RemoveCheckpointEd 366.

See also properties:

- CurItemNo 322.

See also:

- Modifying RichView items 85;
- "Checkpoints" 65.

1.5.1.2.2.82 TRichView Edit.RemoveCurrentPageBreak

Removes an explicit page break before the paragraph containing the caret.

```
procedure RemoveCurrentPageBreak;
```

(Introduced in version 1.2)

The method removes "explicit page break" flag of the first item belonging to the paragraph containing the caret (see PageBreaksBeforeItems 184), if this flag was set.

Method type: editing-style.

See also:

- InsertPageBreak 355.

See also properties of RichView:

- PageBreaksBeforeItems 184;
- Options 182 (*rvoShowPageBreaks*).

See also properties of RVStyle:

- PageBreakColor 431.

1.5.1.2.2.83 TRichView Edit.RemoveLists

Removes markers 131 (bullets and numbering) from the selected paragraphs.

```
procedure RemoveLists(ARecursive: Boolean);
```

(introduced in version 1.7)

Parameter:

ARecursive – reserved for future use. Set it to *False*.

Method type: editing-style.

See also:
- ApplyListStyle 331;
- ChangeListLevels 338.

See also methods of TCustomRichView:
- SetListMarkerInfo 269;
- GetListMarkerInfo 233.

1.5.1.2.2.84 TRichView Edit.ResizeControl

Resizes control 126 in the **ItemNo**-th item, and quickly reformats the affected part of the document.

```
procedure ResizeControl(ItemNo, NewWidth, NewHeight: Integer);
```

The **ItemNo**-th item must be of "inserted control" 126 type (*rvsComponent* 711), otherwise the method raises ERichViewError 649 exception.

The control's width will be set to **NewWidth**, height to **NewHeight**.

Method type: editing-style.

See also methods:
- ResizeCurrentControl 368;
- SetControlInfoEd 374;
- AdjustControlPlacement 330;
- GetItemStyle 229.

See also properties:
- ItemCount 180.

See also:
- Modifying RichView items 85;
- Item types 64.

1.5.1.2.2.85 TRichView Edit.ResizeCurrentControl

Resizes control 126 at the position of caret and quickly reformats the affected part of the document.

```
procedure ResizeCurrentControl(NewWidth, NewHeight: Integer);
```

(introduced in version 1.4)

ResizeCurrentControl(...) is equivalent to TopLevelEditor 329.ResizeControl 368(TopLevelEditor 329.CurItemNo 322, ...).

Method type: editing-style.

See also methods:
- ResizeControl 368;
- SetControlInfoEd 374;
- AdjustControlPlacement 330;
- GetItemStyle 229.

See also properties:
- ItemCount 180.

See also:
- Modifying RichView items 85;

- Item types 64.

1.5.1.2.2.86 TRichView Edit.SearchText -W

The methods search for the substring **s** in the document.

```
function SearchText(s: String; SrchOptions: TRVESearchOptions672): Boolean;
function SearchTextA(s: TRVAnsiString668; SrchOptions: TRVESearchOptions672): Boolean;
function SearchTextW(s: TRVUnicodeString694; SrchOptions: TRVESearchOptions672): Boolean;
```

When found, these methods select substring and move the caret to the end of selection.

If there is no selected fragment, the methods start searching from the caret position, up or down depending on **SrchOptions**.

If something is selected, *rvseoSmartStart* is used. If it is included, then:

- when searching down, the search starts from the second selected character;
- when searching up, the search starts from the last but one selected character;

(selected characters are counted from the top of document, the selection direction is ignored). This option allows to avoid selecting the same fragment again when changing a search direction.

If *rvseoMultiItem* is included, the search can match substrings of several text items 120. If not included, the text is searched in each text item separately. For example, if the document contains the text **He*llo***, the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.

Unicode note:

Internally, text is stored as Unicode. **SearchTextA** (and **SearchText** in non-Unicode versions of Delphi) converts **s** to Unicode using Style 193.DefCodePage 421. and calls **SearchTextW**.

These methods can be used for search-and-replace operation as well. **SearchText** selects the found string, InsertText 360 replaces it with the new one. See important note about search and replace in down-up direction in the topic about InsertText 360.

Note: There is incompatibility with programs written with the older version of RichView (1.0 – 1.1.4). Now if rvsroDown is not included in SrchOptions, the methods search upwards. In the older versions there was no rvsroDown option, and SearchText searched downwards only.

These methods must be called only when the document is formatted.

Return value:

Is the substring found?

See also:

- TRichView.SearchText, -A, W 256;
- GetRVESearchOptions 660 global function;
- Searching in RichView and RichViewEdit 83.

1.5.1.2.2.87 TRichView Edit.SelectCurrentLine

Selects the line containing the caret.

```
procedure SelectCurrentLine;
```

(introduced in version 1.6)

The line beginning and ending position may depend on word wrapping.

This method repaints document, moves caret to the end of line and generates OnSelect 301 event.

This method must be called only when the document is formatted.

See also methods:

- SelectCurrentWord 370;

- GetSelText[238].

See also:

- Selecting in RichView document[81].

1.5.1.2.2.88 TRichView Edit.SelectCurrentWord

Selects word at the position of caret. If there is non text item at the caret position, this method does nothing.

```
procedure SelectCurrentWord;
```

This method repaints document, moves the caret to the end of word and generates OnSelect[301] event.

"Word" is defined as a part of string between delimiters. Delimiters are listed in Delimiters[174] property.

This method must be called only when the document is formatted.

See also methods:

- SelectWordAt[258];
- GetSelText[238];
- SelectCurrentLine[369].

See also:

- Selecting in RichView document[81].

1.5.1.2.2.89 TRichView Edit.Set*PropertyEd

Editing-style methods for changing properties of the editor.

```
procedure SetIntPropertyEd(Prop: TRVIntProperty[681]; Value: Integer;
  AutoReformat: Boolean);
procedure SetStrPropertyEd(Prop: TRVStrProperty[690]; const Value: TRVUnicodeString[694];
  AutoReformat: Boolean);
procedure SetFloatPropertyEd(Prop: TRVFloatProperty[678]; Value: TRVLength[682];
  AutoReformat: Boolean);
```

(introduced in version 15; changed in version 18)

Parameters:

Prop identifies the property to change. If the property is a property of enumerated type, use Ord(). If the property is a boolean property, any non-zero value means *True*.

Value – new property value.

If **AutoReformat**=*True*, the editor is reformatted (if necessary), and Change[337] is called. Normally, this method is called with **AutoReformat**=*True* if you change a single property. If you call these methods multiple times for changing multiple properties, reformat (if necessary) and call Change yourself.

Method type: editing-style (unlike direct assignment to these properties).

Example 1: calling for a property of an enumerated type (using Ord):

```
MyRichViewEdit.SetIntPropertyEd(rvipBackgroundStyle, Ord(bsTiled), True);
```

Example 2: calling for a property of a boolean type (any nonzero value means *True*):

```
MyRichViewEdit.SetIntPropertyEd(rvipDPMirrorMargins, 1, True);
```

Example 3: calling for multiple properties:

```
//var NewMargins: TRect
```

```
if MyRichViewEdit.CanChange then begin
  MyRichViewEdit.BeginUndoGroup(rvutLayout);
  MyRichViewEdit.SetUndoGroupMode(True);
  MyRichViewEdit.SetIntPropertyEd(rvipLeftMargin,   NewMargins.Left,   False);
  MyRichViewEdit.SetIntPropertyEd(rvipTopMargin,    NewMargins.Top,    False);
  MyRichViewEdit.SetIntPropertyEd(rvipRightMargin,  NewMargins.Right,  False);
  MyRichViewEdit.SetIntPropertyEd(rvipBottomMargin, NewMargins.Bottom, False);
  MyRichViewEdit.SetUndoGroupMode(False);
  MyRichViewEdit.Reformat;
  MyRichViewEdit.Change;
end;
```

See also methods:

- SetBackgroundImageEd.

1.5.1.2.2.90 TRichView Edit.SetBackgroundImageEd

An editing-style method for changing BackgroundBitmap.

```
procedure SetBackgroundImageEd(Graphic: TGraphic; AutoReformat: Boolean);
```

(introduced in version 15)

Parameters:

Graphic – a graphic to assign to BackgroundBitmap. If **Graphic** is not TBitmap, it is converted to TBitmap. After calling this method, the component uses a copy of **Graphic**, so you must free **Graphic** yourself.

If **AutoReformat**=*True*, Change is called. Normally, this method is called with **AutoReformat**=*True* if you change a single property. If you call this method together with other methods in a group, this parameter must be *False*, and you must call Change yourself.

Method type: editing-style (unlike direct assignment to BackgroundBitmap).

Example:

```
if MyRichViewEdit.CanChange then begin
  MyRichViewEdit.BeginUndoGroup(rvutBackground);
  MyRichViewEdit.SetUndoGroupMode(True);
  MyRichViewEdit.SetIntPropertyEd(rvipBackgroundStyle, Ord(bsTiled), False);
  MyRichViewEdit.SetBackgroundImageEd(Image1.Picture.Graphic, False);
  MyRichViewEdit.SetUndoGroupMode(False);
  MyRichViewEdit.Invalidate;
  MyRichViewEdit.Change;
end;
```

See also methods:

- SetIntPropertyEd, SetStrPropertyEd, SetFloatPropertyEd.

1.5.1.2.2.91 TRichView Edit.SetBreakInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a *break*.

```
procedure SetBreakInfoEd(ItemNo: Integer; AWidth: TRVStyleLength;
  AStyle: TRVBreakStyle; AColor: TColor; const ATag: TRVTag);
```

Parameters:

ItemNo – index of the item. The item must be of *break* type (*rvsBreak*), otherwise the method raises ERichViewError exception. Items are indexed from 0 to ItemCount-1, GetItemStyle returns type of item.

Items of subdocuments (table cells 605) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds 271), and call SetBreakInfoEd of this inplace-editor. Alternatively, you can use SetCurrentBreakInfo 375.

AWidth – line width (or rectangle height). This value is measured in Style 193.Units 437.

AStyle – visual style of this *break*, see TRVBreakStyle 669 for possible values.

AColor – line color. If it is equal to *clNone*, Style 193.TextStyles 437[0].Color 469 is used.

ATag – *tag* of the item. You can use value returned by GetBreakInfo 218 or GetItemTag 230 for this item.

Instead of this method, you can use SetItemExtraIntPropertyExEd 383 and SetItemTagEd 384 methods.

Method type: editing-style (unlike SetBreakInfo 260).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd 383 and SetItemExtraStrPropertyEd 384.

When possible, use SetCurrentBreakInfo 375 instead of this method.

See also methods of TCustomRichView:

- GetBreakInfo 218;
- GetItemStyle 229;
- SetBreakInfo 260.

See also methods:

- SetCurrentBreakInfo 375.

See also properties:

- ItemCount 180.

See also:

- Modifying RichView items 85;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.92 TRichView Edit.SetBulletInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of TRichView, if this item is a *bullet* 128 *or hotspot* 129.

```
procedure SetBulletInfoEd(ItemNo: Integer; const AName: TRVUnicodeString 694;
  AImageIndex: Integer; AImageList: TCustomImageList; const ATag: TRVTag 692);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *bullet* 128 or *hotspot* 129 type (*rvsBullet or rvsHotspot* 711), otherwise the method raises ERichViewError 649 exception. Items are indexed from 0 to ItemCount 180-1, GetItemStyle 229 returns type of item.

Items of subdocuments (table cells 605) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds 271), and call SetBulletInfoEd of this inplace-editor. Alternatively, you can use SetCurrentBulletInfo 375.

AName – name of *bullet* 128, any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW 385 method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyExEd 383 method.

ATag – *tag* of the item. You can use value returned by GetBulletInfo[219] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTagEd[384] method.

Method type: editing-style (unlike SetBulletInfo[261]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[383] and SetItemExtraStrPropertyEd[384].

When possible, use SetCurrentBulletInfo[375] instead of this method.

See also methods of TCustomRichView:

- GetBulletInfo[219];
- GetItemStyle[229];
- SetBulletInfo[261].

See also methods:

- SetCurrentBulletInfo[375].

See also properties:

- ItemCount[180].

See also:

- Modifying RichView items[85];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.93 TRichView Edit.SetCheckpointInfoEd

An editing-style method for creating new or modifying the existing *checkpoint* associated with the **ItemNo**-th item

```
procedure SetCheckpointInfoEd(ItemNo: Integer; const ATag: TRVTag[692];
  const AName: TRVUnicodeString[694]; ARaiseEvent: Boolean);
```

(changed in version 18)

Parameters

ItemNo – index of the item, from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetCheckpointInfoEd of this inplace-editor. Alternatively, you can use SetCurrentCheckpointInfo[376].

ATag – *tag* of *checkpoint*;

AName – name of the *checkpoint*, any string without line break (CR, LF) characters.

ARaiseEvent – "raise event" flag; if set, RichView can generate OnCheckpointVisible[274] event for this *checkpoint*.

Method type: editing-style (unlike SetCheckpointInfo[261]).

When possible, use SetCurrentCheckpointInfo[376] instead of this method.

See also methods of TCustomRichView:

- GetItemCheckpoint[226];
- SetCheckpointInfo[261].

See also methods:

- SetCurrentCheckpointInfo[376].

See also properties:

- ItemCount 180.

See also:

- Modifying RichView items 85;
- "Checkpoints" 65;
- "Tags" 68.

1.5.1.2.2.94 TRichView Edit.SetControlInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is "inserted control" 126.

```
procedure SetControlInfoEd(ItemNo: Integer;
  const AName: TRVUnicodeString 694; AVAlign: TRVVAlign 695;
  const ATag: TRVTag 692);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of control 126 type (*rvsComponent* 711), otherwise the method raises ERichViewError 649 exception. Items are indexed from 0 to ItemCount 180-1, GetItemStyle 229 returns type of item.

Items of subdocuments (table cells 605) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds 271), and call SetControlInfoEd of this inplace-editor. Alternatively, you can use SetCurrentControlInfo 377.

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW 385 method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign 695 for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetControlInfo 222 or GetItemTag 230 for this item. The *tag* can also be set by SetItemTagEd 384 method.

Method type: editing-style (unlike SetControlInfo 262).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd 383 and SetItemExtraStrPropertyEd 384.

When possible, use SetCurrentControlInfo 377 instead of this method.

See also methods of TCustomRichView:

- GetControlInfo 222;
- GetItemStyle 229;
- SetControlInfo 262.

See also methods:

- SetCurrentControlInfo 377;
- ResizeControl 368;
- AdjustControlPlacement 330.

See also properties:

- ItemCount 180.

See also:

- Modifying RichView items 85;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.95 TRichView Edit.SetCurrentBreakInfo

Changes properties for the item at the position of caret, if this item is a *break* 125

```
procedure SetCurrentBreakInfo(AWidth: TRVStyleLength 690;
  AStyle: TRVBreakStyle 669;  AColor: TColor; const ATag: TRVTag 692);
```

SetCurrentBreakInfo(...) is equivalent to TopLevelEditor 329.SetBreakInfoEd 371(TopLevelEditor 329.CurItemNo 322, ...).

This method can be used if CurItemStyle 323 returns *rvsBreak* 711.

Parameters:

AWidth – line width (or rectangle height). This value is measured in Style 193.Units 437.

AStyle – visual style of this *break*, see TRVBreakStyle 669 for possible values.

AColor – line color. If it is equal to *clNone*, Style 193.TextStyles 437[0].Color 469 is used.

ATag – *tag* of the item. You can use value returned by GetCurrentBreakInfo 341 or GetCurrentTag 349 for this item.

Instead of this method, you can use SetCurrentItemExtraIntPropertyEx 378 and SetCurrentTag 382 methods.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty 378 and SetCurrentItemExtraStrProperty 379.

See also methods:

- GetCurrentBreakInfo 341;
- SetBreakInfoEd 371.

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Modifying RichView items 85;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.96 TRichView Edit.SetCurrentBulletInfo

Changes properties for the item at the position of caret, if this item is a *bullet* 128 *or* *hotspot* 129.

```
procedure SetCurrentBulletInfo(const AName: TRVUnicodeString 694;
  AImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag 692);
```

(changed in version 18)

SetCurrentBulletInfo(...) is equivalent to TopLevelEditor 329.**SetBulletInfoEd** 372(TopLevelEditor 329.CurItemNo 322, ...).

This method can be used if CurItemStyle 323 returns *rvsBullet or rvsHotspot* 711.

Parameters:

AName – name of *bullet* 128, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW 380 method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetCurrentItemExtraIntPropertyEx[378] method.

ATag – *tag* of the item. You can use value returned by GetCurrentBulletInfo[342] or GetCurrentTag[349] for this item. The *tag* can also be set by SetCurrentTag[382] method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[378] and SetCurrentItemExtraStrProperty[379].

See also methods:

- GetCurrentBulletInfo[342];
- SetBulletInfoEd[372].

See also properties:

- CurItemNo[322];
- CurItemStyle[323].

See also:

- Modifying RichView items[85];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.97 TRichView Edit.SetCurrentCheckpointInfo

Creates or modifies *checkpoint* associated with the item at the position of caret.

```
procedure SetCurrentCheckpointInfo(const ATag: TRVTag[692];
  const AName: TRVUnicodeString[694]; ARaiseEvent: Boolean);
```

(changed in version 18)

SetCurrentCheckpointInfo(...) is equivalent to TopLevelEditor[329].**SetCheckpointInfoEd**[373](TopLevelEditor[329].CurItemNo[322], ...).

If the item at the position of caret already has a *checkpoint* (see GetCurrentCheckpoint[343]), its properties are modified; otherwise, a new *checkpoint* is created.

This method is used in conjunction with GetCurrentCheckpoint[343] and RemoveCurrentCheckpoint[367]. The alternative set of methods is: InsertCheckpoint[351], GetCheckpointAtCaret[341] and RemoveCheckpointAtCaret[366].

Parameters.

ATag – *tag* of *checkpoint*;

AName – name of the *checkpoint*, any string without line break (CR, LF) characters.

ARaiseEvent – "raise event" flag; if set, RichView can generate OnCheckpointVisible[274] event for this *checkpoint*.

Method type: editing-style.

See also methods:

- GetCurrentCheckpoint[343], RemoveCurrentCheckpoint[367];
- InsertCheckpoint[351], GetCheckpointAtCaret[341], RemoveCheckpointAtCaret[366];
- SetCheckpointInfoEd[373].

See also properties:

- CurItemNo[322].

See also:

- Modifying RichView items 85;
- "Checkpoints" 65;
- "Tags" 68.

1.5.1.2.2.98 TRichView Edit.SetCurrentControlInfo

Modifies the item at the position of caret, if this item is an inserted control 126.

```
procedure SetCurrentControlInfo(const AName: TRVUnicodeString 694;
  AVAlign: TRVVAlign 695; const ATag: TRVTag 692);
```

(changed in version 18)

SetCurrentControlInfo(...) is equivalent to TopLevelEditor 329.**SetControlInfoEd** 374(TopLevelEditor 329.CurItemNo 322, ...).

This method can be used if CurItemStyle 323 returns *rvsComponent* 711.

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemText 380 method. This is not a Name property of the control!

AVAlign – vertical alignment of this item, see TRVVAlign 695 for possible options.

ATag – *tag* of the item. Do not confuse with Tag property of the control! You can use value returned by GetCurrentControlInfo 343 or GetCurrentTag 349 for this item. The *tag* can also be set by SetCurrentTag 382 method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty 378 and SetCurrentItemExtraStrProperty 379.

See also methods:

- GetCurrentControlInfo 343;
- SetControlInfoEd 374.

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Modifying RichView items 85;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.99 TRichView Edit.SetCurrentHotspotInfo

Modifies the item at the position of caret, if this item is a *hotspot* 129.

```
procedure SetCurrentHotspotInfo(const AName: TRVUnicodeString 694;
  AImageIndex, AHotImageIndex: Integer; AImageList: TCustomImageList;
  const ATag: TRVTag 692);
```

(changed in version 18)

SetCurrentHotspotInfo(...) is equivalent to TopLevelEditor 329.**SetHotspotInfoEd** 382(TopLevelEditor 329.CurItemNo 322, ...).

This method can be used if CurItemStyle 323 returns *rvsHotspot* 711.

Parameters:

AName – name of *hotspot*[129], any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW[380] method.

AImageList – not used, reserved, set it to *nil*..

AImageIndex – index of image in the image list. It can also be set using SetCurrentItemExtraIntPropertyEx[378] method.

AHotImageIndex – index of "hot" image in the image list. This image is displayed under the mouse pointer (in TRichView, or in TRichViewEdit in hypertext mode), or when user moves the caret to this item (in TRichViewEdit). It can also be set using SetCurrentItemExtraIntPropertyEx[378] method.

ATag – *tag* of the item. You can use value returned by GetCurrentHotspotInfo[344] or GetCurrentTag[349] for this item. The *tag* can also be set by SetCurrentTag[382] method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty[378] and SetCurrentItemExtraStrProperty[379].

See also methods:

- GetCurrentHotspotInfo[344];
- SetHotspotInfoEd[382].

See also properties:

- CurItemNo[322];
- CurItemStyle[323].

See also:

- Modifying RichView items[85];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.100 TRichView Edit.SetCurrentItemExtraIntProperty -Ex

The methods change value of an integer property for the item at the position of caret.

```
procedure SetCurrentItemExtraIntProperty(Prop: TRVExtraItemProperty;
  Value: Integer; AutoReformat: Boolean);
procedure SetCurrentItemExtraIntPropertyEx(Prop, Value: Integer;
  AutoReformat: Boolean);
```

(introduced in versions 1.7 and 15)

SetCurrentItemExtraIntProperty[Ex](...) is equivalent to TopLevelEditor[329].**SetItemExtraIntProperty[Ex]Ed**[383](TopLevelEditor[329].CurItemNo[322], ...).

These methods set a new **Value** of the item's property identified by **Prop**.

In **SetCurrentItemExtraIntProperty**, **Prop**'s type is TRVExtraItemProperty[673]. See information about this type for the list of properties.

In **SetCurrentItemExtraIntPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemProperty, **SetCurrentItemExtraIntPropertyEx** works like **SetCurrentItemExtraIntProperty**. In addition, it supports properties identified by rveipc*** constants[706]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[390].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[337] and reformat the document after it yourself. A reformatting depends on the property:

- some properties do not require reformatting (for example, colors), repainting is enough (call Invalidate);

- some properties require a local reformatting (change properties between calls of BeginItemModify[335] and EndItemModify[341]);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call Reformat[248]).

Method type: editing-style.

See also methods:

- GetCurrentItemExtraIntProperty[Ex][346];
- SetItemExtraIntProperty[Ex]Ed[383];
- SetCurrentItemExtra[379]**Str**[379]Property[379];

See also methods of TCustomRichView:

- SetItemExtraIntProperty[Ex][266].

1.5.1.2.2.101 TRichView Edit.SetCurrentItemExtraStrProperty -Ex

The methods change a value of a string property for the item at the position of caret

```
procedure SetCurrentItemExtraStrProperty(Prop: TRVExtraItemStrProperty[676];
  const Value: TRVUnicodeString[694]; AutoReformat: Boolean);
procedure SetCurrentItemExtraStrPropertyEx(Prop: Integer;
  const Value: TRVUnicodeString[694]; AutoReformat: Boolean);
```

(introduced in versions 1.9 and 15; changed in version 18)

SetCurrentItemExtraStrProperty[Ex](...) is equivalent to TopLevelEditor[329].**SetItemExtraStrProperty[Ex]Ed**[384](TopLevelEditor[329].CurItemNo[322], ...).

This method sets a new **Value** of the item's property identified by **Prop**.

In **SetCurrentItemExtraStrProperty**, **Prop**'s type is TRVExtraItemStrProperty[676]. See information about this type for the list of properties.

In **SetCurrentItemExtraStrPropertyEx**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemStrProperty, **SetCurrentItemExtraStrPropertyEx** works like **SetCurrentItemExtraStrProperty**. In addition, it supports properties identified by rvespc*** constants[709]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[390].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[337] and reformat the document after it yourself. A reformatting depends on the property:

- some properties do not require reformatting or repainting;
- some properties require a local reformatting (for example, "format string" of numbered sequences) (change properties between calls of BeginItemModify[335] and EndItemModify[341]);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call Reformat[248]).

Method type: editing-style.

See also methods:

- GetCurrentItemExtraStrProperty[Ex][346];
- SetItemExtraStrProperty[Ex]Ed[384];
- SetCurrentItemExtra[378]**Int**[378]Property[Ex][378].

See also methods of TCustomRichView:

- SetItemExtraStrProperty[Ex][266].

1.5.1.2.2.102 TRichView Edit.SetCurrentItemText -A -W

Changes text of text item or a name of non-text item at the position of caret.

```
procedure SetCurrentItemText(const s: String);
procedure SetCurrentItemTextA(const s: TRVAnsiString[668]);
procedure SetCurrentItemTextW(const s: TRVUnicodeString[694]);
```

(Introduced in version 1.4, 1.7)

- **SetCurrentItemText**(s) is equivalent to TopLevelEditor[329].**SetItemTextEd**[385](TopLevelEditor[329].CurItemNo[322], s).
- **SetCurrentItemTextA**(s) is equivalent to TopLevelEditor[329].**SetItemTextEdA**[385](TopLevelEditor[329].CurItemNo[322], s).
- **SetCurrentItemTextW**(s) is equivalent to TopLevelEditor[329].**SetItemTextEdW**[385](TopLevelEditor[329].CurItemNo[322], s).

Parameters:

s – text assigned to the item. For text items, this is a visible text. For non-text item, this is a string value associated with the item (not displayed). **S** must not contain line break (CR and LF) characters. For text items, it must not contain CR, LF, TAB, FF characters (#13, #10, #9, #12).

Unicode notes:

Internally, text is stored as Unicode. **SetCurrentItemTextA** (and **SetCurrentItemText** in non-Unicode versions of Delphi) converts ANSI to Unicode. If the current item is a text item, a code page for conversion is calculated basing on the its Charset[468]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[193].DefCodePage[421] is used.

Methods type: editing-style.

See also methods:

- GetCurrentItemText -A -W[347];
- SetItemTextEd -A -W[385].

See also methods of TCustomRichView:

- SetItemText -A -W[267].

See also properties:

- CurItemNo[322];
- CurItemStyle[323].

See also:

- Obtaining RichView items[84];
- Item types[64];
- "Tags"[68];
- Unicode in RichView[99].

1.5.1.2.2.103 TRichView Edit.SetCurrentItemVAlign

Changes vertical alignment (position relative to the line) of the item at the position of caret.

```
procedure SetCurrentItemVAlign(VAlign: TRVVAlign[695]);
```

(introduced in version 12)

SetCurrentItemVAlign(VAlign) is equivalent to TopLevelEditor[329].**SetItemVAlignEd**[386](TopLevelEditor[329].CurItemNo[322], VAlign).

Method type: editing-style.

This method must not be called if the current item is one of: text items[120], tables[130], *breaks*[125], tabs[121], list markers[131], footnotes[136] and endnotes[135].

See also methods:

- GetCurrentItemVAlign 348;
- SetItemVAlignEd 386.

See also properties:

- CurItemNo 322.

See also:

- Modifying RichView items 85.

1.5.1.2.2.104 TRichView Edit.SetCurrentPictureInfo

Modifies the item at the position of caret, if this item is a picture 122 or *hot-picture* 125.

```
procedure SetCurrentPictureInfo(const AName: TRVUnicodeString 694;
  Agr: TGraphic; AVAlign: TRVVAlign 695; const ATag: TRVTag 692);
```

(changed in version 18)

SetCurrentPictureInfo(...) is equivalent to TopLevelEditor 329.**SetPictureInfoEd** 386(TopLevelEditor 329.CurItemNo 322, ...).

This method can be used if CurItemStyle 323 returns *rvsPicture* 711.

Parameters:

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetCurrentItemTextW 380 method.

Agr – graphic object. If this item does not use a shared image (by default), **Agr** must be an unique graphic object (you cannot assign the same graphic object to two or more items) or the value returned by GetCurrentPictureInfo 349 (the image is shared if *rvepShared* extra item property 673 is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetCurrentPictureInfo 349, the previously used graphic object is freed.

AVAlign – vertical alignment of the picture.

ATag – *tag* of the item. You can use value returned by GetCurrentPictureInfo 349 or GetCurrentTag 349 for this item. The *tag* can also be set by SetCurrentTag 382 method.

Method type: editing-style.

Additional item properties are assigned by the methods SetCurrentItemExtraIntProperty 378 and SetCurrentItemExtraStrProperty 379.

See also methods:

- GetCurrentPictureInfo 349;
- SetPictureInfoEd 386;
- SetCurrentItemExtraIntProperty 378.

See also properties:

- CurItemNo 322;
- CurItemStyle 323.

See also:

- Modifying RichView items 85;
- Item types 64;
- "Tags" 68.

1.5.1.2.2.105 TRichView Edit.SetCurrentTag

Changes tag of the item at the position of caret.

```
procedure SetCurrentTag(ATag: TRVTag);
```

SetCurrentTag(Tag) is equivalent to TopLevelEditor.**SetItemTagEd**(TopLevelEditor.CurItemNo, Tag).

Method type: editing-style.

See also methods:

- GetCurrentTag;
- SetItemTagEd.

See also properties:

- CurItemNo.

See also:

- Modifying RichView items;
- "Tags".

1.5.1.2.2.106 TRichView Edit.SetHotspotInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a *hotspot*.

```
procedure SetHotspotInfoEd(ItemNo: Integer; const AName: TRVUnicodeString;
  AImageIndex,  AHotImageIndex: Integer;
  AImageList: TCustomImageList; const ATag: TRVTag);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of *hotspot* type (*rvsHotspot*), otherwise the method raises ERichViewError exception. Items are indexed from 0 to ItemCount-1, GetItemStyle returns type of item.

Items of subdocuments (table cells) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds), and call SetHotspotInfoEd of this inplace-editor. Alternatively, you can use SetCurrentHotspotInfo.

AName – name of *hotspot*, any string without line break (CR, LF) characters. It can also be set using SetItemTextEdW method.

AImageList – not used, reserved, set it to *nil*.

AImageIndex – index of image in image list. It can also be set using SetItemExtraIntPropertyExEd method.

AHotImageIndex – index of "hot" image in image list. This image is displayed under the mouse pointer (in TRichView, or in TRichViewEdit in hypertext mode), or when user moves the caret to this item (in TRichViewEdit). It can also be set using SetItemExtraIntPropertyExEd method.

ATag – *tag* of the item. You can use value returned by GetHotspotInfo or GetItemTag for this item. The *tag* can also be set by SetItemTagEd method.

Method type: editing-style (unlike SetHotspotInfo).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd and SetItemExtraStrPropertyEd.

When possible, use SetCurrentHotspotInfo instead of this method.

See also methods of TCustomRichView:

- GetHotspotInfo[224];
- GetItemStyle[229];
- SetHotspotInfo[265].

See also methods:

- SetCurrentHotspotInfo[377].

See also properties:

- ItemCount[180].

See also:

- Modifying RichView items[85];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.107 TRichView Edit.SetItemExtraIntProperty-Ed -ExEd

Editing-style methods for changing a value of the specified additional integer property of the **ItemNo**-th item..

```
procedure SetItemExtraIntPropertyEd(ItemNo: Integer;
  Prop: TRVExtraItemProperty[673]; Value: Integer;
  AutoReformat: Boolean);
procedure SetItemExtraIntPropertyExEd(ItemNo, Prop,
  Value: Integer; AutoReformat: Boolean);
```

(introduced in versions 1.7 and 15)

These methods set a new **Value** of the item property identified by **Prop**.

ItemNo – index of the item, from 0 to ItemCount[180]-1. Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetItemExtraIntProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraIntProperty[Ex][378].

In **SetItemExtraIntPropertyEd**, **Prop**'s type is TRVExtraItemProperty[673]. See information about this type for the list of properties.

In **SetItemExtraIntPropertyExEd**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemProperty, **SetItemExtraIntPropertyExEd** works like **SetItemExtraIntPropertyEd**. In addition, it supports properties identified by rveipc*** constants[706]

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[390].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[337] and reformat the document after it yourself. A reformatting depends on the property:

- some properties do not require reformatting (for example, colors), repainting is enough (call Invalidate);
- some properties require a local reformatting (change properties between calls of BeginItemModify[335] and EndItemModify[341]);
- some properties require a complete reformatting (for example, "start-from" and "reset" of numbered sequences) (call Reformat[248]).

Methods type: editing-style (unlike SetItemExtraIntProperty[Ex][266]).

When possible, use SetCurrentItemExtraIntProperty[Ex][378] instead of these methods.

See also methods:

- SetItemExtra[384]**Str**[384]PropertyEd[384];
- SetCurrentItemExtraIntProperty[Ex][378].

See also methods of TCustomRichView:

- SetItemExtraIntProperty[Ex][266].

1.5.1.2.2.108 TRichView Edit.SetItemExtraStrProperty-Ed -ExEd

Editing-style methods for changing a value of the specified additional string property of the **ItemNo**-th item..

```
procedure SetItemExtraStrPropertyEd(ItemNo: Integer;
  Prop: TRVExtraItemStrProperty[676]; const Value: TRVUnicodeString[694];
  AutoReformat: Boolean);
procedure SetItemExtraStrPropertyExEd(ItemNo, Prop: Integer;
  const Value: TRVUnicodeString[694]; AutoReformat: Boolean);
```

(introduced in versions 1.9 and 15; changed in version 18)

These methods set a new **Value** of the item property identified by **Prop**.

In **SetItemExtraStrPropertyEd**, **Prop**'s type is TRVExtraItemStrProperty[676]. See information about this type for the list of properties.

In **SetItemExtraStrPropertyExEd**, **Prop**'s type is Integer. If **Prop** can be converted to TRVExtraItemStrProperty, **SetItemExtraStrPropertyExEd** works like **SetItemExtraStrPropertyEd**. In addition, it supports properties identified by rvespc*** constants[709]

ItemNo – index of the item, from 0 to ItemCount[180]-1.

Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetItemExtraStrProperty[Ex]Ed of this inplace-editor. Alternatively, you can use SetCurrentItemExtraStrProperty[379].

If **AutoReformat**=*True*, these methods automatically reformat the document and call OnChange[390].

If **AutoReformat**=*False* (this can be useful if you set several properties of an item at once), call Change[337] and reformat the document after it yourself. A reformatting depends on the property:

- some properties do not require reformatting or repainting;
- some properties require a local reformatting (for example, "format string" of numbered sequences) (change properties between calls of BeginItemModify[335] and EndItemModify[341]);
- some properties require a complete reformatting (for example, "sequence name" of numbered sequences) (call Reformat[248]).

Method type: editing-style (unlike SetItemExtraStrProperty[Ex][266]).

When possible, use SetCurrentItemExtraStrProperty[Ex][379] instead of this method.

See also methods:

- SetCurrentItemExtraStrProperty[Ex][379];
- SetItemExtra[383]**Int**[383]Property[Ex]Ed[383].

See also methods of TCustomRichView:

- SetItemExtraStrProperty[Ex][266].

1.5.1.2.2.109 TRichView Edit.SetItemTagEd

Changes tag of the **ItemNo**-th item.

```
procedure SetItemTagEd(ItemNo: Integer; const ATag: TRVTag[692]);
```

Items are indexed from 0 to ItemCount[180]-1.

Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetItemTagEd of this inplace-editor. Alternatively, you can use SetCurrentTag[382].

Tag can be an integer value or a pointer to dynamically allocated string, see *Tags*[68].

Method type: editing-style, unlike SetItemTag[267].

When possible, use SetCurrentTag[382] instead of this method.

See also methods of TCustomRichView:

- GetItemTag[230];
- SetItemTag[267].

See also methods:

- SetCurrentTag[382].

See also properties:

- ItemCount[180].

See also:

- Modifying RichView items[85];
- "Tags"[68].

1.5.1.2.2.110 TRichView Edit.SetItemTextEd -A -W

Editing-methods for changing text of text item or name of non-text item.

```
procedure SetItemTextEd(ItemNo: Integer; const s: String);
procedure SetItemTextEdA(ItemNo: Integer; const s: TRVAnsiString[668]);
procedure SetItemTextEdW(ItemNo: Integer; const s: TRVUnicodeString[694]);
```

(Introduced in version 1.4, 1.7)

Parameters:

ItemNo – index of the item. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item.

Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetItemTextEd* of this inplace-editor. Alternatively, you can use SetCurrentItemText*[380].

s – text assigned to the item. For text items, this is a visible text. For non-text item, this is a string value associated with the item (not displayed). **S** must not contain line break (CR and LF) characters. For text items, it must not contain CR, LF, TAB, FF characters (#13, #10, #9, #12).

Unicode notes:

Internally, text is stored as Unicode. **SetItemTextEdA** (and **SetItemTextEd** in non-Unicode versions of Delphi) converts ANSI to Unicode. If the **ItemNo**-th item is a text item, a code page for conversion is calculated basing on the its Charset[468]. If it is equal to DEFAULT_CHARSET, and for non-text items, Style[193].DefCodePage[421] is used.

Methods type: editing-style.

See also methods:

- SetCurrentItemText -A -W[380].

See also methods of TCustomRichView:

- GetItemText -A -W[230];
- SetItemText -A -W[267].

See also properties:

- ItemCount[180].

See also:

- Modifying RichView items[85];
- "Tags"[68];
- Unicode in RichView[99].

1.5.1.2.2.111 TRichView Edit.SetItemVAlignEd

Changes vertical alignment (position relative to the line) of the **ItemNo**-th item.

```
procedure SetItemVAlignEd(ItemNo: Integer; VAlign: TRVVAlign[695]);
```

(introduced in version 12)

Items are indexed from 0 to ItemCount[180]-1.

Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetItemVAlignEd of this inplace-editor. Alternatively, you can use SetCurrentItemVAlign[380].

Method type: editing-style, unlike SetItemVAlign[268].

This method must not be called for text items[120], tables[130], *breaks*[125], tabs[121], list markers[131], footnotes[136] and endnotes[135].

When possible, use SetCurrentItemVAlign[380] instead of this method.

See also methods of TCustomRichView:

- GetItemVAlign[230];
- SetItemVAlign[268].

See also methods:

- SetCurrentItemVAlign[380].

See also properties:

- ItemCount[180].

See also:

- Modifying RichView items[85].

1.5.1.2.2.112 TRichView Edit.SetPictureInfoEd

An editing-style method for changing properties of the **ItemNo**-th item of RichView, if this item is a picture[122] or *hot-picture*[125].

```
procedure SetPictureInfoEd(ItemNo: Integer; const AName: TRVUnicodeString[694];
  Agr: TGraphic; AVAlign: TRVVAlign[695]; const ATag: TRVTag[692]);
```

(changed in version 18)

Parameters:

ItemNo – index of the item. The item must be of picture[122] or *hot-picture*[125] type (*rvsPicture or rvsHotPicture*[711]), otherwise the method raises ERichViewError[649] exception. Items are indexed from 0 to ItemCount[180]-1, GetItemStyle[229] returns type of item.

Items of subdocuments (table cells[605]) are not included in the items range of the main document. For items in cells, activate inplace-editor, select the proper item in the cell (SelSelectionBounds[271]), and call SetPictureInfoEd of this inplace-editor. Alternatively, you can use SetCurrentPictureInfo[381].

AName – name of the item, any string without line break (CR, LF) characters. It can also be set using SetItemTextEd[385] method.

Agr – graphic object. If this item does not use a shared image (by default), **Agr** must be an unique graphic object (you cannot assign the same graphic object to two or more items) or the value returned by GetPictureInfo[236] for this item (the image is shared if *rvepShared* extra item property[673] is nonzero). If this item does not use a shared image, and **Agr** is not equal to the value returned by GetPictureInfo[236] for this item, the previously used graphic object is freed.

AVAlign – vertical alignment of the picture.

ATag – *tag* of the item. You can use value returned by GetPictureInfo[236] or GetItemTag[230] for this item. The *tag* can also be set by SetItemTagEd[384] method.

Method type: editing-style (unlike SetPictureInfo[269]).

Additional item properties are assigned by the methods SetItemExtraIntPropertyEd[383] and SetItemExtraStrPropertyEd[384].

When possible, use SetCurrentPictureInfo[381] instead of this method.

See also methods of TCustomRichView:
- GetPictureInfo[236];
- GetItemStyle[229];
- SetPictureInfo[269].

See also methods:
- SetCurrentPictureInfo[381].

See also properties:
- ItemCount[180].

See also:
- Modifying RichView items[85];
- Item types[64];
- "Tags"[68].

1.5.1.2.2.113 TRichView Edit.SetUndoGroupMode

Begins or ends group of editing operations which will be undone/redone as a single operation.

```
procedure SetUndoGroupMode(GroupUndo: Boolean);
```

(Introduced in version 1.3)

See Undo in RichViewEdit[88].

See also methods:
- BeginUndoGroup[336];
- BeginUndoCustomGroup[336].

1.5.1.2.2.114 TRichView Edit.Undo

Undoes the last editing operation.

```
procedure Undo;
```

(Introduced in version 1.3)

This method does nothing if there is no action to undo.

Information about the operation to redo is returned by the methods UndoAction[388] and UndoName[388].

This method must be called only when the document is formatted.

This method is called when the user presses Ctrl+Z or Alt+Backspace.

Method type: editing-style.

See also:
- Redo[365];
- Undo in RichViewEdit[88].

1.5.1.2.2.115 TRichView Edit.UndoAction

Returns type of operation to be undone[387] next.

```
function UndoAction: TRVUndoType[693];
```

(Introduced in version 1.3)

See TRVUndoType[693] for the list of operations. If the returned value is *rvutCustom*, you can use UndoName[388].

See also:

- RedoAction[366];
- Undo in RichViewEdit[88].

1.5.1.2.2.116 TRichView Edit.UndoName

Returns name of operation to be undone[387] next.

```
function UndoName: TRVUnicodeString[694];
```

(Introduced in version 1.3; changed in version 18)

Returns empty string for the standard undo operations, i.e. all operations but *rvutCustom* (see UndoAction[388]).

Returns non-empty string only for custom undo operation initiated by BeginUndoCustomGroup[336].

See also:

- RedoName[366];
- Undo in RichViewEdit[88].

1.5.1.2.3 Events

1.5.1.2.3.1 TRichView Edit.OnBeforeOleDrop, OnAfterOleDrop

OLE drag&drop events. They occur when the user completes drag&drop operation into the editor. These events occur before and after insertion.

```
property OnBeforeOleDrop: TNotifyEvent;
property OnBeforeOleDrop: TNotifyEvent;
```

(introduced in version 16)

When the editor accepts dropped data, the following sequence of actions happens:

1. **OnBeforeOleDrop** event
2. data insertion (OnOleDrop[397] event may occur)
3. **OnAfterOleDrop** event

These events may be used to initialize and finalize insertion as a result of OLE drag&drop.

Example

This example uses RichViewActions.

MyRVAControlPanel: TRVAControlPanel is placed on a form. A downloader component is assigned to MyRVAControlPanel.DownloadInterface property, to allow downloading remote pictures referred in inserted data (RTF or URL).

```
procedure TMyForm.MyRichViewEditBeforeOleDrop(Sender: TObject);
begin
  MyRVAControlPanel.InitImportPictures(nil, nil);
  (Sender as TCustomRichViewEdit).OnImportPicture := MyRVAControlPanel.DoImportPicture;
end;
procedure TForm3.RichViewEdit1AfterOleDrop(Sender: TObject);
begin
  (Sender as TCustomRichViewEdit).OnImportPicture := nil;
  MyRVAControlPanel.DoneImportPictures;
end;
```

See also:

- Drag&drop in TRichView' 89;

1.5.1.2.3.2 TRichView Edit.OnCaretGetOut

Occurs when user presses arrow keys at the beginning/end of document in "outside" direction.

```
type
  TRVGetOutDirection = (rvdLeft, rvdUp, rvdRight, rvdDown,
    rvdTop, rvdBottom);
  TRVOnCaretGetOutEvent =
    procedure (Sender: TCustomRichViewEdit;
     Direction: TRVGetOutDirection) of object;

property OnCaretGetOut:TRVOnCaretGetOutEvent;
```

(introduced in version 1.4, 1.5)

Direction	The event happens if...
rvdLeft	User pressed **Left** (or **Right**, if BiDiMode 170 =*rvbdRightToLeft*) while the caret is at the beginning of the document
rvdUp	User pressed **↑Up** while the caret is in the first line of the document
rvdRight	User pressed **Right** (or **Left**, if BiDiMode 170 =*rvbdRightToLeft*) while the caret is at the end of the document
rvdDown	User pressed **Down** while the caret is in the last line of the document.
rvdTop	User pressed **Ctrl**+**Home**
rvdBottom	User pressed **Ctrl**+**End**

This event can be used to move focus to another control.

1.5.1.2.3.3 TRichView Edit.OnCaretMove

Occurs when the caret is moved to the another position within the document

```
property OnCaretMove: TNotifyEvent;
```

(introduced in version 1.5)

Warning: This event may occur when document is not completely formatted! Do not use any methods relying on formatting inside this event, including methods working with selection (for example, do not use SelectionExists[258] here). If you need to update user interface reflecting selection, use OnSelect[301].

In this event, you can use the following:

- CurItemNo[322] property;
- OffsetInCurItem[328] property;
- GetCurrentLineCol[348] method.

1.5.1.2.3.4 TRichView Edit.OnChange

Occurs when document is changed.

```
property OnChange: TNotifyEvent;
```

This event occurs when:

- user edits document, or
- calling any editing-style method, or
- calling Change[337].

Viewer-style methods do not generate this event.

See also property:

- Modified[327].

1.5.1.2.3.5 TRichView Edit.OnChanging

Occurs before editing operations

```
type
  TRVChangingEvent = procedure (Sender: TCustomRichViewEdit;
    var CanEdit: Boolean) of object;

property OnChanging: TRVChangingEvent;
```

Set **CanEdit** to *False* to prevent editing.

This event occurs when:

- user tries to edit text, or
- calling any editing-style method (before changes are actually done).

Viewer-style methods do not generate this event.

See also:

- OnChange[390] event;
- ReadOnly[328] property.

1.5.1.2.3.6 TRichView Edit.OnCheckStickingItems

Allows to "stuck" items together.

```
type
  TRVStickingItemsEvent = procedure (Sender: TCustomRichViewEdit;
    RVData: TCustomRVData[650]; FirstItemNo: Integer; var Sticking: Boolean) of object;

property OnCheckStickingItems: TRVStickingItemsEvent;
```

(introduced in version 14)

This event occurs before inserting something after the **FirstItemNo**-th item in **RVData** (if **FirstItemNo**=-1, this insertion is at the beginning of **RVData**).

Assign **Sticking**=*True* to prevent this insertion.

This event occurs after all default checking for sticking is done (see the Protection[473] property of a text style).

1.5.1.2.3.7 TRichView Edit.OnCurParaStyleChanged

Occurs when the index of current paragraph style (value of CurParaStyleNo[323] property) is changed.

```
property OnCurParaStyleChange: TNotifyEvent;
```

This event can occur when user moves caret with mouse or keyboard, or when he/she edits document. Some methods of RichViewEdit can also invoke this event.

Use this event to update user interface in your application.

See also properties:

- CurParaStyleNo[323].

See also events:

- OnCurTextStyleChanged[391].

1.5.1.2.3.8 TRichView Edit.OnCurTextStyleChanged

Occurs when the index of current text style (value of CurTextStyleNo[324] property) is changed.

```
property OnCurTextStyleChange: TNotifyEvent;
```

This event can occur when user moves caret with mouse or keyboard, or when he/she edits document. Some methods of RichViewEdit can also invoke this event.

Use this event to update user interface in your application.

See also properties:

- CurTextStyleNo[324].

See also events:

- OnCurParaStyleChanged[391].

1.5.1.2.3.9 TRichView Edit.OnDraw CustomCaret

Allows to draw a custom caret.

```
type
  TRVDrawCustomCaretEvent = procedure (Sender: TCustomRichViewEdit; Canvas: TCanvas;
    const Rect: TRect) of object;
property OnDrawCustomCaret: TRVDrawCustomCaretEvent;
```

(introduced in version 10)

Custom caret is drawn if CustomCaretInterval[324]>0.

OnMeasureCustomCaret[394] event occurs before each call of this event.

Example

This example draws a rectangle/ellipse caret. In combination with the example in OnMeasureCustomCaret[394] topic, it draws rectangle/ellipse around the character to the right of the caret.

```
procedure TMyForm.MyRichViewEditDrawCustomCaret(Sender: TCustomRichViewEdit;
  Canvas: TCanvas; const Rect: TRect);
const f: Boolean=True;
var r: TRect;
begin
  if f then
    Canvas.Pen.Color := clRed
  else
    Canvas.Pen.Color := clBlue;
  f := not f;
  r := Rect;
  Canvas.Brush.Style := bsClear;
  if f then
    Canvas.Rectangle(r)
  else
    Canvas.Ellipse(r);
end;
```

1.5.1.2.3.10 TRichView Edit.OnDropFile

Occurs when dropping files in the editor (as a result of drag&drop operation, for example from Windows Explorer), or when pasting data in this format.

```
type
  TRVDropFileEvent = procedure(Sender: TCustomRichViewEdit;
    const FileName: TRVUnicodeString[694]; var DoDefault: Boolean) of object;

property OnDropFile: TRVDropFileEvent;
```

(introduced in version 17; changed in version 18)

This event occurs when accepting data in CF_HDROP format, when pasting from the Clipboard (see PasteGraphicFiles[362]), or as a result of drag&drop[89].

In this event, you can insert content of this file in the caret position.

Input parameters:

FilesName – a name of file to insert.

DoDefault = *True*.

Output parameters:

DoDefault – set it to *False* if you have processed the file yourself in this event. If not, RichViewEdit will try to insert it as a graphic, RTF, RVF, plain text.

1.5.1.2.3.11 TRichView Edit.OnDropFiles

Occurs when dropping files in the editor (as a result of drag&drop operation, for example from Windows Explorer)

```
type
  TRVDropFilesEvent =
    procedure (Sender: TCustomRichViewEdit;
      Files: TStrings; var FileAction: TRVDropFileAction;
      var DoDefault: Boolean) of object;
  TRVDropFileAction = (rvdfNone, rvdfInsert, rvdfLink);

property OnDropFiles: TRVDropFilesEvent;
```

(introduced in v1.8)

rvddFiles must be included in AcceptDragDropFormats [321] property, otherwise the editor will not accept files.

Input parameters:

Files – list of names of dropped files.

FileAction = *rvdfNone*.

DoDefault = *True*.

Output parameters:

FileAction – operation that you have performed with these files:

- *rvdfNone* – nothing,
- *rvdfInsert* – the files were inserted,
- rvdfLink – hyperlinks to the files were inserted.

FileAction is ignored if **DoDefault** is set to *True*.

DoDefault – set it to *False* if you have processed the files yourself in this event.

If the default processing is not canceled, the editor will process files one by one. For each file, it calls OnDropFile [392] event, then (if the file was not processed in this event), it tries to insert this file as a graphic file, *.RTF, *.RVF, *.TXT.

See **the example** [405] (how to insert files of all formats supported by Microsoft Office text import converters) [405].

See also:

- Drag&Drop [89].

1.5.1.2.3.12 TRichView Edit.OnItemResize

Occurs after resizing image [122], control [126] or table [130] with the mouse.

```
type
  TRVItemResizeEvent = procedure(Sender: TCustomRichViewEdit;
    RVData: TCustomRVFormattedData [650]; ItemNo, Val1, Val2: Integer) of object;
property OnItemResize: TRVItemResizeEvent;
```

(introduced in version 10)

Parameters:

RVData and **ItemNo** identify the resized item. **RVData** – document containing this item; it can be **Sender**.RVData [187], table cell [605], or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**.

(**Val1**, **Val2**) are:

- for images and controls: (-1, -1).

- for tables: (0, row index) after resizing row, or (1, column index) after resizing columns.

1.5.1.2.3.13 TRichView Edit.OnItemTextEdit

Occurs when text item[120]'s text is changed as a result of editing operation.

```
type
  TRVItemTextEditEvent = procedure (Sender: TCustomRichViewEdit;
    const OldText: TRVUnicodeString[694]; RVData: TCustomRVData[650]; ItemNo: Integer;
    var NewTag: TRVTag[692]; var NewStyleNo: Integer) of object;
property OnItemTextEdit: TRVItemTextEditEvent;
```

(introduced in version 10; changed in version 18)

This event is designed to update tags of hyperlinks.

This event is called when document is not completely formatted, so do not use any methods for changing document or methods requiring formatted document in this event.

Input Parameters:

OldText – the item text before the editing operation.

RVData, **ItemNo** – identify the item. **RVData** – document containing this item; it can be **Sender**.RVData[187], table cell[605], or RVData of cell inplace-editor. **ItemNo** – index of this item inside **RVData**. You can get text after the editing operation using **RVData**.GetItemTextW[230](**ItemNo**).

NewTag – the item *tag*[68], the same as **RVData**.GetItemTag[230](**ItemNo**)

NewStyleNo – the item style, index in the collection Style[193].TextStyles[437], the same as **RVData**.GetItemStyle[229](**ItemNo**)

Output Parameters:

NewTag – value of *tag*[68] to assign to this item.

NewStyleNo – style index to assign to this item.

1.5.1.2.3.14 TRichView Edit.OnMeasureCustomCaret

Allows to define size and position for a custom caret.

```
type
  TRVMeasureCustomCaretEvent = procedure (Sender: TCustomRichViewEdit;
    var Rect: TRect) of object;
property OnMeasureCustomCaret: TRVMeasureCustomCaretEvent;
```

(introduced in version 10)

Custom caret

A custom caret is drawn if CustomCaretInterval[324]>0.

This event is called before each call to OnDrawCustomCaret[391]. It allows to set your own size and position for the caret. The default size and position are the same as of the default system caret. The editor stores background under the **Rect**, it will be restored after the call to OnDrawCustomCaret[391].

Standard caret

If the global variable RichViewEditCustomCaretSize[699] = *True*, this even is called for the standard caret as well. If RichViewEditCaretPosition[699] <> *rvhcpRight*, the editor adjusts the position of the caret after calling this event.

1.5.1.2.3.15 TRichView Edit.OnOleDragEnter

OLE drag&drop event. It occurs when the user drags the mouse pointer into the editor. Indicates whether a drop can be accepted, and, if so, the effect of the drop.

```
type
  TRVOleDragEnterEvent = procedure (Sender: TCustomRichView[162];
    const DataObject: IDataObject; Shift: TShiftState;
    X, Y: Integer; PossibleDropEffects: TRVOleDropEffects[682];
    var DropEffect: TRVOleDropEffect[682]) of object;
property OnOleDragEnter: TRVOleDragEnterEvent;
```

(introduced in version 10)

This is a low level event. Do not use it if you are not familiar with OLE drag&drop.

After calling this event, OnOleDragOver[396] will be called while the user moves the mouse pointer inside the editor. Dragging will be finished either by calling OnOleDrop[397] (successfully) or by calling OnOleDragLeave[395] (canceled, or mouse left the editor).

Input Parameters:

DataObject – dragged data (IDataObject is declared in ActiveX unit).

Shift – keyboard state.

PossibleDropEffects – a list of allowed operations, set of *rvdeCopy*, *rvdeMove*, *rvdeLink*.

DropEffect – operation that the editor would choose by default.

Output Parameters:

DropEffect – operation that you want to perform with the dragged data (one of **PossibleDropEffects**, or *rvdeNone* if you do not want to accept it).

See also events:

- OnOleDragOver[396];
- OnOleDragLeave[395];
- OnOleDrop[397].

See also:

- Drag&drop in TRichView[89].

1.5.1.2.3.16 TRichView Edit.OnOleDragLeave

OLE drag&drop event. It occurs when the user drags the mouse cursor out of the editor or cancels the current drag&drop operation.

```
property OnOleDragLeave: TNotifyEvent;
```

(introduced in version 10)

This is a low level event. Do not use it if you are not familiar with OLE drag&drop.

The first event in sequence is OnOleDragEnter[395], it occurs when the user moves the mouse pointer into the editor while dragging. After that, OnOleDragOver[396] is called while the user moves the mouse inside the editor. Dragging is finished either by calling OnOleDrop[397] (successfully) or by calling this event.

Input Parameters:

DataObject – dragged data (IDataObject is declared in ActiveX unit).

Shift – keyboard state.

PossibleDropEffects – a list of allowed operations, set of *rvdeCopy*, *rvdeMove*, *rvdeLink*.

DropEffect – operation that the editor would choose by default.

Output Parameters:

DropEffect – operation that you want to perform with the dragged data (one of **PossibleDropEffects**, or *rvdeNone* if you do not want to accept it).

See also events:

- OnOleDragEnter[395];
- OnOleDragOver[396];
- OnOleDrop[397].

See also:

- Drag&drop in TRichView[89].

1.5.1.2.3.17 TRichView Edit.OnOleDragOver

OLE drag&drop event. It occurs when the user moves the mouse pointer over the editor. Called only if the dragging was accepted in OnOleDragEnter[395] (or automatically by the editor).

```
type
  TRVOleDragOverEvent = procedure (Sender: TCustomRichView[162];
    Shift: TShiftState; X, Y: Integer;
    PossibleDropEffects: TRVOleDropEffects[682];
    var DropEffect: TRVOleDropEffect[682]) of object;
property OnOleDragOver: TRVOleDragOverEvent;
```

(introduced in version 10)

This is a low level event. Do not use it if you are not familiar with OLE drag&drop.

OnOleDragEnter[395] occurs before this event, with DataObject in parameters. Dragging will be finished either by calling OnOleDrop[397] (successfully) or by calling OnOleDragLeave[395] (canceled, or mouse left the editor).

Input Parameters:

DataObject – dragged data (IDataObject is declared in ActiveX unit).

Shift – keyboard state.

X,**Y** – mouse cursor coordinates, relative to the top left corner of **Sender**.

PossibleDropEffects – a list of allowed operations, set of *rvdeCopy*, *rvdeMove*, *rvdeLink*.

DropEffect – operation that the editor would choose by default.

Output Parameters:

DropEffect – operation that you want to perform with the dragged data (one of **PossibleDropEffects**, or *rvdeNone* if you do not want to accept it).

See also events:

- OnOleDragEnter[395];
- OnOleDragLeave[395];
- OnOleDrop[397].

See also:

- Drag&drop in TRichView[89].

1.5.1.2.3.18 TRichView Edit.OnOleDrop

OLE drag&drop event. It occurs when the user completes drag&drop operation into the editor.This event allows you to insert data in your own format.

```
type
  TRVOleDropEvent = procedure (Sender: TCustomRichView;
    const DataObject: IDataObject; Shift: TShiftState; X, Y: Integer;
    PossibleDropEffects: TRVOleDropEffects;
    var DropEffect: TRVOleDropEffect; var DoDefault: Boolean) of object;
property OnOleDrop: TRVOleDropEvent;
```

(introduced in version 10)

This is a low level event. Do not use it if you are not familiar with OLE drag&drop

The first event in sequence is OnOleDragEnter[395], it occurs when the user moves the mouse pointer into the editor while dragging. After that, OnOleDragOver[396] is called while the user moves the mouse inside the editor. Dragging is finished either by calling this event (successfully) or by calling OnOleDragLeave[395] (canceled, or mouse left the editor).

Input Parameters:

DataObject – dragged data (IDataObject is declared in ActiveX unit).

Shift – keyboard state.

X,**Y** – mouse cursor coordinates, relative to the top left corner of **Sender**.

PossibleDropEffects – a list of allowed operations, set of *rvdeCopy*, *rvdeMove*, *rvdeLink*.

DropEffect – operation that the editor would choose by default.

Output Parameters:

DropEffect – the operation that you have performed (one of **PossibleDropEffects**, or *rvdeNone* if you canceled the insertion). If it is *rvdeMove*, it instructs to delete the dragged object from its source location. The value of this parameter is used only if you set **DoDefault**=*False*, otherwise the editor does the default processing. If you inserted some data in editor in this event, they are automatically selected afterwards.

Limitation: when dragging from the same editor, it decides by itself whether it should move or copy data, and it ignores **DropEffect**.

This event occurs between OnBeforeOleDrop and OnAfterOleDrop[388] events.

See also events:

- OnOleDragEnter[395];
- OnOleDragOver[396];
- OnOleDragLeave[395];
- OnDropFiles[393].

See also:

- Drag&drop in TRichView'[89];
- How to make one line, plain text editor[718].

1.5.1.2.3.19 TRichView Edit.OnParaStyleConversion

Occurs while executing ApplyParaStyleConversion[332] method.

```
type
  TRVStyleConversionEvent =
    procedure (Sender: TCustomRichViewEdit;
```

```
      StyleNo, UserData: Integer; AppliedToText: Boolean;
      var NewStyleNo: Integer) of object;

property OnParaStyleConversion:TRVStyleConversionEvent;
```

(introduced in version 1.7)

This event allows you to create custom conversion procedure for styles of the selected paragraphs. This event is called for all selected items allowing to change their paragraph style.

It can be useful for implementing commands like "change paragraph alignment", "change paragraph indents", etc.

Parameters:

StyleNo – current paragraph style (index in collection of styles, Style[193].ParaStyles[431]);

UserData – value passed as a parameter in ApplyParaStyleConversion[332];

AppliedToText – not used

NewStyleNo – initially equals to StyleNo; assign a new value to this parameter to change paragraph style.

Example:

```
procedure TMyForm.MyRichViewEditParaStyleConversion(Sender: TCustomRichViewEdit;
  StyleNo, UserData: Integer; AppliedToText: Boolean;
  var NewStyleNo: Integer);
var ParaInfo: TParaInfo[482];
begin
  // creating paragraph style with the desired attributes
  ParaInfo := TParaInfo.Create(nil);
  try
    ParaInfo.Assign(RVStyle1.ParaStyles[431][StyleNo]);
    case UserData of
      PARA_ALIGNMENT:
        ParaInfo.Alignment := GetAlignment;
      PARA_INDENTINC:
        begin
          ParaInfo.LeftIndent := ParaInfo.LeftIndent+20;
          if ParaInfo.LeftIndent>1000 then
            ParaInfo.LeftIndent := 1000;
        end;
      PARA_INDENTDEC:
        begin
          ParaInfo.LeftIndent := ParaInfo.LeftIndent-20;
          if ParaInfo.LeftIndent<0 then
            ParaInfo.LeftIndent := 0;
        end;
      PARA_COLOR:
        ParaInfo.Background.Color := ColorDialog1.Color;
      // add your code here....
    end;
    // searching for the style (or adding it if necessary)
    NewStyleNo := RVStyle1.FindParaStyle[440](ParaInfo);
  finally
    ParaInfo.Free;
  end;
end;
```

This example shows how to implement the following commands:

- change paragraph alignment,
- increase left indent,
- decrease left indent,
- change paragraph background color.

In this code:

- PARA_*** – user-defined integer constants with unique values; these constants identify commands; they are passed to ApplyParaStyleConversion[332] as UserData;
- RVStyle1 – TRVStyle[418] component linked with the MyRichViewEdit;
- ColorDialog1: TColorDialog;
- GetAlignment – function returning paragraph alignment chosen by the user.

You can see that for new paragraph styles added in this event, Standard[464] property is set to *False*. It is important, otherwise DeleteUnusedStyles[216] will not be able to delete this style, even if all paragraphs of this style will be removed.

See demos:

- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

See also:

- OnStyleConversion[400].

See also methods:

- ApplyParaStyleConversion[332].

1.5.1.2.3.20 TRichView Edit.OnPaste

Allows to paste data from the Clipboard in custom formats.

```
type
  TRVPasteEvent =
    procedure (Sender: TCustomRichViewEdit;
               var DoDefault: Boolean) of object;

property OnPaste: TRVPasteEvent;
```

(introduced in version 1.3)

Occurs before any data are pasted in editor (Paste[361] method, or Ctrl+V / Shift+Insert).

You can insert data in your own format here and/or forbid default pasting procedure (set **DoDefault** to *False*).

Note: see standard priorities of formats for pasting in Paste[361] method.

Example (allowing to paste only one line of plain text):

```
procedure TMyForm.MyRichViewEditPaste(Sender: TCustomRichViewEdit;
  var DoDefault: Boolean);
var s: String;
begin
  s := Clipboard.AsText;
  if (Pos(#10,s)=0) and (Pos(#13,s)=0) then
    MyRichViewEdit.InsertText[360](s,False);
  DoDefault := False;
end;
```

See also:

- RichView and Clipboard[82];
- How to make a plain text editor[718].

1.5.1.2.3.21 TRichView Edit.OnSmartPopupClick

Occurs when user clicks "smart popup" [402] button.

```
type
  TRVSmartPopupClickEvent = procedure (Sender: TCustomRichView;
    Button: TCustomControl) of object;
property OnSmartPopupClick: TRVSmartPopupClickEvent;
```

(introduced in version 10)

This event allows implementing your own effect when clicking "smart popup" button (or on pressing ShortCut [404]). If you use menu, it's not necessary to process this event.

1.5.1.2.3.22 TRichView Edit.OnStyleConversion

Occurs while executing ApplyStyleConversion [333] method.

```
type
  TRVStyleConversionEvent =
    procedure (Sender: TCustomRichViewEdit;
     StyleNo, UserData: Integer; AppliedToText: Boolean;
     var NewStyleNo: Integer) of object;

property OnStyleConversion:TRVStyleConversionEvent;
```

This event allows you to create custom conversion procedure of text styles for the selected text. This event is called for all selected text items allowing to change their text style.

It can be useful to implement commands like "make bold", "apply font", "change text color", etc.

Parameters:

StyleNo – current text style (index in the collection of styles, Style [193].TextStyles [437]);

UserData – value passed as a parameter in ApplyStyleConversion [333];

AppliedToText – *True*, if this event was called for conversion of style of text item; *False*, if it was called for conversion of the current text style (CurTextStyleNo [324]);

NewStyleNo – initially equals to **StyleNo**; assign a new value to this parameter to change text style.

Example 1

```
procedure TMyForm.MyRichViewEditStyleConversion(Sender: TCustomRichViewEdit;
  StyleNo, UserData: Integer; AppliedToText: Boolean;
  var NewStyleNo: Integer);
begin
   if not (AppliedToText and
      (rvprStyleProtect in Style.TextStyles[StyleNo].Protection [473])) then
    NewStyleNo := UserData;
end;
```

This example does exactly the same work as ApplyTextStyle [334], i.e. MyRichViewEdit.ApplyStyleConversion [333] (StyleNo) will be equivalent to MyRichViewEdit.ApplyTextStyle [334](StyleNo).

You can see that ApplyStyleConversion ignores protection of styles. If you wish to protect them, you should check protection yourself and do not change this style (like in the example above).

Example 2

```
procedure TMyForm.MyRichViewEditStyleConversion(Sender: TCustomRichViewEdit;
                  StyleNo, UserData: Integer;
                  AppliedToText: Boolean;
                  var NewStyleNo: Integer);
```

```
var FontInfo: TFontInfo[478];
begin
  // creating text style with the desired attributes
  FontInfo := TFontInfo.Create(nil);
  try
    FontInfo.Assign(RVStyle1.TextStyles[437][StyleNo]);
    case UserData of
      TEXT_BOLD:
        if btnBold.Down then
          FontInfo.Style := FontInfo.Style[475]+[fsBold]
        else
          FontInfo.Style := FontInfo.Style-[fsBold];
      TEXT_ITALIC:
        if btnItalic.Down then
          FontInfo.Style := FontInfo.Style+[fsItalic]
        else
          FontInfo.Style := FontInfo.Style-[fsItalic];
      TEXT_UNDERLINE:
        if btnUnderline.Down then
          FontInfo.Style := FontInfo.Style+[fsUnderline]
        else
          FontInfo.Style := FontInfo.Style-[fsUnderline];
      TEXT_APPLYFONTNAME:
        FontInfo.FontName := FontName;
      TEXT_APPLYFONTSIZE:
        FontInfo.Size     := FontSize;
      TEXT_APPLYFONT:
        FontInfo.Assign(FontDialog1.Font);
      TEXT_COLOR:
        FontInfo.Color := ColorDialog1.Color;
      TEXT_BACKCOLOR:
        FontInfo.BackColor := ColorDialog1.Color;
      // add your code here....
    end;
    // searching for the style (or adding it if necessary)
    NewStyleNo := RVStyle1.FindTextStyle[440](FontInfo);
  finally
    FontInfo.Free;
  end;
end;
```

This example shows how to implement the following commands:

- make bold/not bold,
- make italic/not italic,
- make underlined/not underlined,
- change font name,
- change font size,
- apply font,
- change text color,
- change text background color.

In this code:

- TEXT_*** – user defined integer constants with unique values; these constants identify commands; they are passed to ApplyStyleConversion[333] as UserData;
- RVStyle1 – TRVStyle[418] component linked with the richview editor;
- ColorDialog1: TColorDialog;
- FontDialog1: TFontDialog;

- btnBold, btnItalic, btnUnderline: TSpeedButton;
- FontName: String;
- FontSize: Integer.

See demos:
- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

See also:
- OnParaStyleConversion 397.

See also methods:
- ApplyStyleConversion 333.

1.5.1.2.4 Classes of Properties

1.5.1.2.4.1 TRVSmartPopupProperties

Properties of "smart popup" buttons (analogs of buttons used for Microsoft Office "smart tags"). Class of TCustomRichViewEdit.SmartPopupProperties 329.

Unit RVPopup.

Syntax

```
TRVSmartPopupProperties = class (TPersistent)
```

(introduced in version 10)

Hierarchy

TObject
TPersistent

Using

"Smart popups" allow to edit properties of the current (at the position of caret) item 64. Of course, you can use popup menu instead, but in some cases "smart popups" may be useful (for example, to show to user that some special commands available for the current item). Only one "smart popup" button can be displayed at the same time.

Use TCustomRichViewEdit.OnCaretMove 390 event to set properties of "smart popup" and display the button. The button is displayed when you assign True to TCustomRichViewEdit.SmartPopupVisible 329. The editor hides this button automatically on document change or clearing. Since "smart popup" works only for the current item, you must ensure that it is displayed for the current item (update or hide it when the caret moves to another item).

The following properties define visual appearance of "smart popup" button:
- ButtonType 402;
- Color 403, LineColor 404;
- HoverColor 403, HoverLineColor 403;
- ImageList 404, ImageIndex 404.

Hint 403 is displayed when the user moves the mouse pointer over the button.

Position 404 defines the button position.

When the user clicks the button or presses ShortCut 404, Menu 404 is displayed (if assigned). Instead of a menu, you can process TCustomRichViewEdit.OnSmartPopupClick 400 event to execute a command or display a dialog.

Defines how popup buttons look like in hot (under the mouse pointer) state.

```
type
  TRVSmartPopupType = (rvsptDropDown, rvsptShowDialog, rvsptSimple);
```

```
property ButtonType: TRVSmartPopupType;
```

Note: TRVSmartPopupType is defined in RVScroll unit.

Type	Meaning	Example
rvsptDropDown	Button width is increased, a triangle is shown. Recommended for buttons displaying menu[404] or popup windows.	
rvsptShowDialog	Button width is increased, an ellipsis is shown. Recommended for buttons displaying dialogs.	
rvsptSimple	Button width is not changed. Recommended for buttons executing an immediate command.	

You can change this property in TCustomRichViewEdit.OnCaretMove[390] (to create the proper button for the item at the position of caret)

Default value:

rvsptDropDown

Background color of a button.

```
property Color: TColor;
```

Default value:

clWindow

See also:

- LineColor[404];
- HoverColor[403];
- HoverLineColor[403].

Contains the text string that can appear when the user moves the mouse over a button.

```
property Hint: TRVUnicodeString[694];
```

You can change this property in TCustomRichViewEdit.OnCaretMove[390] (to create the proper button for the item at the position of caret)

Background color of a button when it is below the mouse pointer.

```
property HoverColor: TColor;
```

Default value:

clInfoBk

See also:

- Color[403];
- LineColor[404];
- HoverLineColor[403].

Color of button border and the right side sign (ellipsis or triangle)[402] when a button is under the mouse pointer.

```
property HoverLineColor: TColor;
```

Default value:

clInfoText

See also:

- Color[403];

- HoverColor 403;
- HoverColor 403.

Index of a button image in ImageList 404.

```
property ImageIndex: Integer;
```

This image is displayed on a button.

You can change this property in TCustomRichViewEdit.OnCaretMove 390 (to create the proper button for the item at the position of caret)

Default value:

0

Link to an image list containing images for buttons.

```
property ImageList: TCustomImageList;
```

The ImageIndex 404-th image is shown on button.

You can change this property in TCustomRichViewEdit.OnCaretMove 390 (to create the proper button for the item at the position of caret).

Border color of a button.

```
property LineColor: TColor;
```

Default value:

clHighlight

See also:

- Color 403;
- HoverColor 403;
- HoverLineColor 403.

Link to popup menu to display when the user clicks a button or presses ShortCut 404.

```
property Menu: TPopupMenu;
```

Alternatively, you can process TCustomRichViewEdit.OnSmartPopupClick 400 event to execute a command or display a dialog.

You can change this property in TCustomRichViewEdit.OnCaretMove 390 (to create the proper button for the item at the position of caret).

Defines the position of a smart popup button relative to the current item.

```
type
  TRVSmartPopupPosition =
    (rvsppTopLeft, rvsppTopRight, rvsppBottomRight, rvsppBottomLeft);
property Position: TRVSmartPopupPosition;
```

Note: TRVSmartPopupPosition is defined in RVScroll unit.

Specifies the key combination users can type to execute command associated with visible "smart popup" button.

```
property ShortCut: TShortCut;
```

The effect of pressing ShortCut is the same as clicking the button.

Default value:

$6028 (Shift + Ctrl + ⇩)

See also properties:

- Menu 404.

See also events of TCustomRichViewEdit:

- OnSmartPopupClick 400.

Call SetButtonState(*True*) to fix the button in the "hot" state. It's useful while you display your own drop down window for the button. Call SetButtonState(*False*) to return it back to normal.

```
procedure SetButtonState(Hot: Boolean);
```

See also events of TCustomRichViewEdit:

- OnSmartPopupClick 400.

1.5.1.2.5 Examples

1.5.1.2.5.1 Moving caret to the beginning of paragraph

This example shows how to move caret to the beginning of the ParagraphIndex-th paragraph

```
procedure GoToParagraph(rve: TCustomRichViewEdit316; ParagraphIndex: Integer);
var i: Integer;
begin
  for i := 0 to rve.ItemCount180-1 do
  begin
    if rve.IsParaStart242(i) then
      dec(ParagraphIndex);
    if ParagraphIndex<0 then begin
      rve.SetSelectionBounds271(i, rve.GetOffsBeforeItem235(i),
        i, rve.GetOffsBeforeItem235(i));
      rve.Invalidate;
      exit;
    end;
  end;
end;
```

Call:

```
GoToParagraph(RichViewEdit1, 7);
```

ParagraphIndex is zero-based.

1.5.1.2.5.2 OnDropFile Example

By default, TRichViewEdit accepts the following types of dropped files:

- graphics,
- RVF,
- RTF,
- plain text (ANSI).

This example shows how to use OnDropFiles 393 event to insert all these types of files, and all file formats supported by installed Microsoft Office text import converters.

This example assumes that RVOfficeConverter1:TRVOfficeConverter 532 is placed on the form.

```
procedure TForm1.RichViewEdit1DropFiles(Sender: TCustomRichViewEdit;
  Files: TStrings; var FileAction: TRVDropFileAction;
var DoDefault: Boolean);
  {.....................................................}
  { This function inserts one file FileName }
  function InsertFile(const FileName: String): Boolean;
```

```
  var
    gr: TGraphic;
    Ext: String;
    i: Integer;
  begin
    Result := False;
    try
      // 1. Trying to insert as a graphic
      gr := RVGraphicHandler[710].LoadFromFile(FileName);
      if gr <> nil then
      begin
        Sender.InsertPicture[355]('', gr, rvvaBaseline[695]);
        Result := True;
        exit;
      end;
      // 2. Trying to insert as RTF file
      Ext := LowerCase(ExtractFileExt(FileName));
      if Ext='.rtf' then
      begin
        Sender.InsertRTFFromFileEd[356](FileName);
        Result := True;
        exit;
      end;
      // 3. Trying to insert as RVF file
      if Ext='.rvf' then
      begin
        Sender.InsertRVFFromFileEd[357](FileName);
        Result := True;
        exit;
      end;
      // 4. Trying to insert as text file
      if Ext='.txt' then
      begin
        Sender.InsertTextFromFile[361](FileName);
        Result := True;
        exit;
      end;
      // 5. Trying to insert using office converters
      for i := 0 to RVOfficeConverter1.ImportConverters[535].Count-1 do
        if Pos(Ext, RVOfficeConverter1.ImportConverters[i].Filter)>0 then
          if RVOfficeConverter1.ImportRTF[538](FileName, i) then
          begin
            RVOfficeConverter1.Stream[536].Position := 0;
            Sender.InsertRTFFromFileEd[356](RVOfficeConverter1.Stream);
            RVOfficeConverter1.Stream.SetSize(0);
            Result := True;
            exit;
          end;
    except
    end;
  end;
  {.....................................................}
var i: Integer;
begin
  for i := 0 to Files.Count-1 do
    InsertFile(Files[i]);
```

```
  DoDefault := False;
end;
```

1.5.1.3 TDBRichView

Use TDBRichView to display hypertext documents stored in database field.

Unit DBRV;

Syntax

```
TDBRichView = class(TCustomRichView)
```

DBRichView does not display its content if it is not linked to RVStyle component via Style[193] property.

The most important property settings can be done in the component editor[168] for TRichView. In Delphi (C++Builder) IDE, right click the DBRichView object on the form, choose "Settings" in the context menu.

If the application doesn't require the data-aware capabilities of TDBRichView, use TRichView[162] instead, to conserve system resources.

For editing documents in database fields, use TDBRichViewEdit[410].

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[544]
TCustomRichView[162]

Field Type

This component must be linked to BLOB field.

If you use RVF[93] (RichView Format), the database field must have binary BLOB type (i.e. must have ability to store arbitrary binary data, without conversion and filtering of characters). For example, in Paradox Tables it is "Binary", in Microsoft Access tables it is "OLE Object".

You can also use text BLOB field type (memo). Memo fields cannot store binary data, so it is impossible to store documents in RVF format in them (unless you exclude *rvfoSaveBinary* from RVFOptions[188]).

However, you can use memo fields:

- to display text or RTF (Rich Text Format) fields,
- to display text mode RVF (that was saved with *rvfoSaveBinary* excluded from RVFOptions[188]).

Starting from Delphi 2006, Unicode memo fields (Field[409].DataType=*ftWideMemo*) are supported specially. TDBRichView does not attempt loading RVF from these fields. It tries to load Unicode RTF (RTF code converted from String to WideString) or Unicode text from them.

If *rvfoLoadBack* is in RVFOptions[188], the component clears BackgroundBitmap[169] before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF[93] (RichView Format),
- RTF (Rich Text Format),
- text,
- others (using events).

For all field types except for Unicode memos:

1. OnLoadCustomFormat[410] event occurs.
2. If this event is not processed, TDBRichView tries to load RVF.
3. If the field does not contain RVF, it tries to load RTF.
4. If the field does not contain RTF, it loads plain ANSI text.

For Unicode memos:

1. OnLoadCustomFormat[410] event occurs.
2. If this event is not processed, TDBRichView tries to load Unicode RTF (RTF code stored as WideString).
3. If the field does not contain RTF, it loads a plain Unicode text.

Events

OnNewDocument[410] event occurs when creating a new document or before loading an existing one from the field.

When the document is loaded from the field, OnLoadDocument[410] occurs.

OnLoadCustomFormat[410] allows loading documents in custom formats.

1.5.1.3.1 Properties

1.5.1.3.1.1 TDBRichView .AutoDisplay

Determines whether to automatically display the contents of field in the TDBRichView control.

```
property AutoDisplay: Boolean;
```

If AutoDisplay is *True*, the TDBRichView control automatically displays new data when the underlying BLOB field changes (such as when moving to a new record).

If AutoDisplay is *False*, the control shows only the field name whenever the underlying BLOB field changes. To display the data, the user can double-click on the control.

The effect of AutoDisplay is not purely cosmetic. When AutoDisplay is *False*, if the data changes, the content of TDBRichView document changes to the name of the field (written in the 0th text and paragraph style). Thus, if AutoDisplay is *False*, applications should be cautious about using the methods for saving contents to ascertain the value of the underlying field.

Calling the LoadField[409] method causes component to update to the current value of the BLOB field. This change will also be reflected in the appearance of the control on screen.

Change the value of AutoDisplay to *False* if the automatic loading of BLOB fields seems to take too long.

1.5.1.3.1.2 TDBRichView .DataField

Specifies the field from which the TDBRichView control displays data.

```
property DataField: String;
```

Use DataField to bind the TDBRichView control to the field in the dataset. To fully specify database field, both the dataset and the field within that dataset must be defined. The DataSource[408] property of the TDBRichView control specifies the dataset which contains the DataField.

Please use only BLOB fields with TDBRichView.

1.5.1.3.1.3 TDBRichView .DataSource

Links the TDBRichView control to the dataset that contains the field it represents.

```
property DataSource : TDataSource;
```

Use DataSource to link the TDBRichView control to the dataset in which the data can be found. To fully specify database field for the DBRichView control, both the dataset and the field within that dataset must be defined. Use the DataField 408 property to specify the particular field within the dataset.

1.5.1.3.1.4 TDBRichView.Field

Indicates the TField object for the database field the TDBRichView control represents.

```
property Field : TField;
```

Read Field to get direct access to the content and properties of the database field.

If you use RVF 93 (RichView Format), the database field must have binary BLOB type (i.e. must have ability to store arbitrary binary data, without conversion and filtering of characters).

Using memo is not recommended (and impossible if you want to save styles, controls, pictures and tables in binary mode, or use Unicode).

However, you can use memo fields:

- to display text or RTF (Rich Text Format) fields,
- to display text mode RVF (that was saved with *rvfoSaveBinary* excluded from RVFOptions 188).

1.5.1.3.2 Methods

1.5.1.3.2.1 TDBRichView.Create

A constructor, creates new DBRichView

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate DBRichView at runtime. DBRichViews placed on forms at design time are created automatically. Specify the owner of the new DBRichView using the **AOwner** parameter.

Important note: initial values of several properties of components created in code may be different from initial values of components placed on form at design time. In the latter case, initial values of these properties can be overridden by the designtime component editor 168 (and overridden by default!)

1.5.1.3.2.2 TDBRichView.Destroy

A destructor, frees an instance of a DBRichView object.

```
destructor Destroy;override;
```

Destroy seldom needs to be called. When DBRichView is constructed without an owner by the Create 409 method, Free should be called to release memory and dispose of the object. Free checks to see if the pointer is *nil* before calling Destroy.

1.5.1.3.2.3 TDBRichView.LoadField

LoadField loads the document from the field in the database into the TDBRichView control.

```
procedure LoadField;
```

LoadField allows an application to control when the TDBRichView control displays the contents of the BLOB field. Use LoadField to update the document in the TDBRichView control to the value of the field in the current record. LoadField only works when the AutoDisplay 408 property is *False*, so that the value of the field is not loaded automatically.

Similar to LoadMemo in TDBRichEdit or TDBMemo.

1.5.1.3.3 Events

1.5.1.3.3.1 TDBRichView.OnLoadCustomFormat

This event allows loading data from field in your own format.

```
type
  TRVCustomFormatEvent = procedure (Sender: TCustomRichView;
    Stream: TStream; var DoDefault: Boolean) of object;
property OnLoadCustomFormat: TRVCustomFormatEvent;
```

(introduced in version 10)

Input parameters:

Stream contains data from DataField. Data should be loaded from **Stream** to **Sender**.

DoDefault is *True*.

Output parameters:

If data are loaded successfully, set **DoDefault** to *False*, otherwise the component will attempt to load RVF/RTF/text.

1.5.1.3.3.2 TDBRichView.OnLoadDocument

Occurs after loading document from the database field

```
property OnLoadDocument: TNotifyEvent;
```

(introduced in v1.9)

This event allows to preprocess the loaded document before displaying.

See also:

- OnNewDocument.

1.5.1.3.3.3 TDBRichView.OnNew Document

Occurs when creating or before loading document

```
property OnNewDocument: TNotifyEvent;
```

(introduced in v1.9)

Occurs:

- on creating new document (before displaying field from newly added record);
- before loading document from the field.

Use this event to assign default values to properties (such as background, margins, styles).

See also:

- OnLoadDocument.

1.5.1.4 TDBRichViewEdit

Use TDBRichView to edit hypertext documents stored in database field.

Unit DBRV;

Syntax

```
TDBRichViewEdit = class(TCustomRichViewEdit)
```

DBRichViewEdit does not display or edit its content if it is not linked to RVStyle component via Style property.

The most important property settings can be done in the component editor[168] for TRichView. In Delphi (C++Builder) IDE, right click the DBRichViewEdit object on the form, choose "Settings" in the context menu. If you decide to choose "Allow adding styles dynamically" mode, please set AutoDeleteUnusedStyles[412] to *True*.

If the application doesn't require the data-aware capabilities of TDBRichViewEdit, use TRichViewEdit[316] instead, to conserve system resources.

If the application doesn't require the editing capabilities of TDBRichViewEdit, use TDBRichView[407] instead, to conserve system resources.

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[544]
TCustomRichView[162]
TCustomRichViewEdit[316]

Field Type

This component must be linked to BLOB field.

If you use RVF[93] (RichView Format), the database field must have binary BLOB type (i.e. must have ability to store arbitrary binary data, without conversion and filtering of characters). For example, in Paradox Tables it is "Binary", in Microsoft Access tables it is "OLE Object".

You can also use text BLOB field type (memo). Memo fields cannot store binary data, so it is impossible to store documents in RVF format in them (unless you exclude *rvfoSaveBinary* from RVFOptions[188]).

However, you can use memo fields:

- to display text or RTF (Rich Text Format) fields,
- to display text mode RVF (that was saved with *rvfoSaveBinary* excluded from RVFOptions[188]).

Starting from Delphi 2006, Unicode memo fields (Field[413].DataType=*ftWideMemo*) are supported specially. TDBRichView does not attempt loading RVF from these fields. It tries to load Unicode RTF (RTF code converted from String to WideString) or Unicode text from them.

If *rvfoLoadBack* is in RVFOptions[188] and FieldFormat[413]=*rvdbRVF*, the component clears BackgroundBitmap[169] before loading data from the field.

Supported Data Formats

The supported field formats are:

- RVF[93] (RichView Format),
- RTF (Rich Text Format),
- text,
- others (using events).

For all field types except for Unicode memos:

1. OnLoadCustomFormat[415] event occurs.
2. If this event is not processed, TDBRichView tries to load RVF.
3. If the field does not contain RVF, it tries to load RTF.
4. If the field does not contain RTF, it loads plain ANSI text.

For Unicode memos:

1. OnLoadCustomFormat[415] event occurs.
2. If this event is not processed, TDBRichView tries to load Unicode RTF (RTF code stored as WideString).
3. If the field does not contain RTF, it loads plain Unicode text.

Saving data to the field:

OnSaveCustomFormat[416] event allows saving data in your own format. If this event is not processed, documents are saved in FieldFormat[413].

Events

OnNewDocument[416] event occurs when creating a new document or before loading an existing one from the field.

When the document is loaded from the field, OnLoadDocument[415] occurs.

OnLoadCustomFormat[415] and OnSaveCustomFormat[416] allow saving and loading documents in custom formats.

Examples

Example 1: Using viewer-style methods in DBRichViewEdit[416] (such as LoadRTF[243])

Example 2: Inserting controls what can modify themselves in DBRichViewEdit[417] (for example, inserting TEdit; user can modify its text)

1.5.1.4.1 Properties

1.5.1.4.1.1 TDBRichViewEdit.AutoDeleteUnusedStyles

If this property is set to *True*, the editor deletes unused text[437], paragraph[431] and list[429] styles before posting to the database field.

```
property TDBRichViewEdit.AutoDeleteUnusedStyles: Boolean;
```

(introduced in version 1.7)

Assign *True* to this property only if:

- this editor uses TRVStyle[418] object exclusively (no other RichViews are linked to this RVStyle via Style[193] property) and
- collections of text, paragraph and list styles are saved together with documents in the RVF field (see RVFOptions[188], *rvfoSaveTextStyles* and *rvfoSaveParaStyles* must be included)

If FieldFormat[413] is a plain text, this property is ignored,

Default value:

False

1.5.1.4.1.2 TDBRichViewEdit.AutoDisplay

Determines whether to automatically display the contents of field in the TDBRichViewEdit control.

```
property AutoDisplay: Boolean;
```

If AutoDisplay is *True*, the TDBRichViewEdit control automatically displays new data when the underlying BLOB field changes (such as when moving to a new record).

If AutoDisplay is *False*, the control shows only the field name whenever the underlying BLOB field changes. To display the data, the user can double-click on the control.

The effect of AutoDisplay is not purely cosmetic. When AutoDisplay is *False*, if the data changes, the content of TDBRichViewEdit document changes to the name of the field (written in the 0th text and paragraph style). Thus, if AutoDisplay is *False*, applications should be cautious about using the methods for saving contents to ascertain the value of the underlying field.

Calling the LoadField[415] method causes component to update to the current value of the BLOB field. This change will also be reflected in the appearance of the control on screen.

Change the value of AutoDisplay to *False* if the automatic loading of BLOB fields seems to take too long.

1.5.1.4.1.3 TDBRichView Edit.DataField

Specifies the field from which the TDBRichViewEdit control displays data.

```
property DataField: String;
```

Use DataField to bind the TDBRichViewEdit control to the field in the dataset. To fully specify database field, both the dataset and the field within that dataset must be defined. The DataSource[413] property of the TDBRichView control specifies the dataset which contains the DataField.

Please use only BLOB fields with TDBRichViewEdit.

1.5.1.4.1.4 TDBRichView Edit.DataSource

Links the TDBRichViewEdit control to the dataset that contains the field it represents.

```
property DataSource : TDataSource;
```

Use DataSource to link the TDBRichViewEdit control to the dataset in which the data can be found. To fully specify database field for the DBRichViewEdit control, both the dataset and the field within that dataset must be defined. Use the DataField[413] property to specify the particular field within the dataset.

1.5.1.4.1.5 TDBRichView Edit.Field

Indicates the TField object for the database field the TDBRichViewEdit control represents.

```
property Field : TField;
```

Read Field to get direct access to the content and properties of the database field.

If you use RVF[93] (RichView Format), the database field must have binary BLOB type (i.e. must have ability to store arbitrary binary data, without conversion and filtering of characters).

Using memo is not recommended (and impossible if you want to save styles, controls, pictures and tables in binary mode, or use Unicode).

However, you can use memo fields:

- to display text or RTF (Rich Text Format) fields;
- to display text mode RVF (that was saved with *rvfoSaveBinary* excluded from RVFOptions[188]).

1.5.1.4.1.6 TDBRichView Edit.FieldFormat

Format for saving data in the database field

```
property FieldFormat: TRVDBFieldFormat;

type
  TRVDBFieldFormat = (rvdbRVF, rvdbRTF, rvdbText);
```

(introduced in version 1.6)

This property allows changing the field saving format to RTF or plain text.

Note: It is not guaranteed that document saved in RTF and text will be the same after reloading (some item types, text and paragraph attributes cannot be saved in RTF or plain text).

You can save data in your own format using OnSaveCustomFormat[416] event.

Starting from Delphi 2006, Unicode memo fields (Field[413].DataType=*ftWideMemo*) are supported specially. For these fields, if FieldFormat is *rvdbRVF* or *rvdbRTF*, documents are stored as Unicode RTF (RTF code converted from String to WideString). If FieldFormat is *rvdbText*, documents are stored as Unicode text.

Default value:

rvdbRVF

1.5.1.4.1.7 TDBRichView Edit.IgnoreEscape

Disallows processing of **Escape** key

```
property IgnoreEscape: Boolean;
```

(introduced in version 1.8)

By default, **Escape** cancels editing in TDBRichViewEdit.

If you set this property to *True*, **Escape** will do nothing.

Default value:

False

1.5.1.4.1.8 TDBRichView Edit.ReadOnly

Determines whether the user can use the DBRichVewEdit control to change the value of the field in the current record.

```
property ReadOnly : Boolean;
```

Set ReadOnly to *True* to prevent the user from changing the content of the database field. When ReadOnly is *True*, the edit control is used only to display the content of the field. If ReadOnly is *False*, the user can change the field's value as long as the dataset is in edit mode.

Note: TRichViewEdit.ReadOnly[328] property is overridden in TDBRichViewEdit to get/set read-only mode in the linked dataset.

1.5.1.4.2 Methods

1.5.1.4.2.1 TDBRichView Edit.Change

Generates OnChange[390] event.

```
procedure Change; override;
```

TCustomRichViewEdit.Change[337] method was overridden to support data-aware features of TDBRichViewEdit.

In addition to the actions inherited from RichViewEdit, this method also informs the linked dataset that data were modified.

Change can be useful if you need to modify document using viewer-style methods, see the example: Using viewer-style methods in DBRichViewEdit[416].

1.5.1.4.2.2 TDBRichView Edit.Create

A constructor, creates new DBRichViewEdit.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate DBRichViewEdit at runtime. DBRichViewEdits placed on forms at design time are created automatically. Specify the owner of the new DBRichViewEdit using the **AOwner** parameter.

Important note: initial values of several properties of components created in code may be different from initial values of components placed on form at design time. In the latter case, initial values of these properties can be overridden by the designtime component editor[168] (and overridden by default!)

1.5.1.4.2.3 TDBRichView Edit.Destroy

A destructor, frees an instance of a DBRichViewEdit object

```
destructor Destroy;override;
```

Destroy seldom needs to be called. When DBRichViewEdit is constructed without an owner by the Create[414] method, Free should be called to release memory and dispose of the object. Free checks to see if the pointer is *nil* before calling Destroy.

1.5.1.4.2.4 TDBRichView Edit.LoadField

LoadField loads the document from the field in the database into the TDBRichViewEdit control.

```
procedure LoadField;
```

LoadField allows an application to control when the TDBRichViewEdit control displays the contents of the BLOB field. Use LoadField to update the document in the TDBRichViewEdit control to the value of the field in the current record. LoadField only works when the AutoDisplay[412] property is *False*, so that the value of the field is not loaded automatically.

Similar to LoadMemo in TDBRichEdit or TDBMemo.

1.5.1.4.3 Events

1.5.1.4.3.1 TDBRichView Edit.OnLoadCustomFormat

This event allows loading data from field in your own format.

```
type
  TRVCustomFormatEvent = procedure (Sender: TCustomRichView[162];
    Stream: TStream; var DoDefault: Boolean) of object;
property OnLoadCustomFormat: TRVCustomFormatEvent;
```

(introduced in version 10)

Input parameters:

Stream contains data from DataField[413]. Data should be loaded from **Stream** to **Sender**.

DoDefault is *True*.

Output parameters:

If data are loaded successfully, set **DoDefault** to *False*, otherwise the component will attempt to load RVF/RTF/text.

See also events:

- OnSaveCustomFormat[416].

1.5.1.4.3.2 TDBRichView Edit.OnLoadDocument

Occurs after loading document from the database field

```
property OnLoadDocument: TNotifyEvent;
```

(introduced in v1.9)

This event allows to preprocess the loaded document before displaying

See also:

- OnNewDocument[416].

1.5.1.4.3.3 TDBRichView Edit.OnNew Document

Occurs when creating or before loading document

```
property OnNewDocument: TNotifyEvent;
```

(introduced in v1.9)

Occurs:

- on creating new document (before displaying field from newly added record);
- before loading document from the field.

Use this event to assign default values to properties (such as background, margins, styles).

See also:

- OnLoadDocument[415].

1.5.1.4.3.4 TDBRichView Edit.OnSaveCustomFormat

This event allows saving data to field in your own format.

```
type
  TRVCustomFormatEvent = procedure (Sender: TCustomRichView[162];
    Stream: TStream; var DoDefault: Boolean) of object;
property OnSaveCustomFormat: TRVCustomFormatEvent;
```

(introduced in version 10)

Stream is empty initially. Save the content of **Sender** in Stream and set **DoDefault** to *False*, otherwise the component will save document according to the FieldFormat[413] property.

See also events:

- OnLoadCustomFormat[415].

1.5.1.4.4 Examples

1.5.1.4.4.1 Using viewer-style methods in DBRichView Edit

This example shows how to use viewer-style methods in DBRichViewEdit (i.e. methods, introduced in TCustomRichView[162]).

DBRichViewEdit1 edits a field of table Table1.We want to load RTF file.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenDialog1.Execute and DBRichViewEdit1.CanChange[336] then begin
    DBRichViewEdit1.Clear[338];
    DBRichViewEdit1.LoadRTF[243](OpenDialog1.FileName);
    DBRichViewEdit1.Change[414];
    DBRichViewEdit1.Format[217];
  end;
end;
```

Calling CanChange has the same effect as calling Table1.Edit.

If you want to save changes in the database immediately, call Table1.Post instead of DBRichViewEdit1.Format.

This sequence of actions (CanChange - modifications - Change - Format) is required only for viewer-style methods.

Editing-style methods do this work automatically:

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
  if OpenDialog1.Execute then
    DBRichViewEdit1.InsertRTFFromFileEd 356(OpenDialog1.FileName);
end;
```

1.5.1.4.4.2 Inserting controls w hat can modify themselves in DBRichView Edit

This example shows how to work with TEdit inserted in TDBRichViewEdit 410, if you want to store its text in RVF field.

The problem: when user changes text in TEdit, the dataset is not informed that the document is changed, and these changes are not saved.

In this example, we use DBRichViewEdit1:TDBRichViewEdit placed on Form1.

1) Add in the private section of Form1:

```
procedure EditChange(Sender: TObject);
```

Implementation:

```
procedure TForm1.EditChange(Sender: TObject);
begin
  if DBRichViewEdit1.CanChange 336 then
    DBRichViewEdit1.Change 414
  else
    Beep;
end;
```

2) When creating a new TEdit, assign its OnChange event:

```
var Edit: TEdit;
begin
  Edit := TEdit.Create(nil);
  Edit.OnChange := EditChange;
  DBRichViewEdit1.InsertControl 352('', Edit, rvvaBaseline);
end;
```

3) You'll need to assign this event to all edits loaded from the database field with RichView documents. Use OnControlAction 275 of DBRichViewEdit:

```
procedure TForm1.DBRichViewEdit1ControlAction(Sender: TCustomRichView;
  ControlAction: TRVControlAction; ItemNo: Integer; var ctrl: TControl);
begin
  if (ControlAction=rvcaAfterRVFLoad) and (ctrl is TEdit) then
    TEdit(ctrl).OnChange := EditChange;
end;
```

And of course, do not forget to register TEdit in the initialization section of unit:

```
initialization
  RegisterClasses([TEdit]);
```

1.5.1.5 TRVPrintPreview

This component displays a print preview for TRichView 162, TRichViewEdit 316, TDBRichView 407, or TDBRichViewEdit 410.

TRVPrintPreview control can display preview only if it is linked to RVPrint 418, and RVPrint is ready (it has a document assigned and pages formatted)

Unit RVPP;

Syntax

```
TRVPrintPreview = class(TCustomRVPrintPreview 556)
```

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[544]
TCustomRVPrintPreview[556]

Information

See more info in the topic about the ancestor of TRVPrintPreview, TCustomRVPrintPreview[556].

The main property of this component is RVPrint[418], allowing to link it with TRVPrint[513] object.

1.5.1.5.1 Properties

1.5.1.5.1.1 TRVPrintPreview .RVPrint

This property links this preview control to  TRVPrint component.

```
property RVPrint: TRVPrint[513];
```

TRVPrintPreview control can display preview only if it is linked to RVPrint, and RVPrint is ready (it has a document assigned and pages formatted).

1.5.2 Additional Components

1.5.2.1 TRVStyle

This invisible at run-time component is used for defining a visual appearance of RichView[162] and its descendant components.

Unit RVStyle

Syntax

```
TRVStyle = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

Using

If you want to display some document in RichView, associate this RichView with some RVStyle component: create TRVStyle component and assign it to the Style[193] property of RichView. You can do it at design-time using the the Object Inspector.

The same RVStyle object can be used by several RichView[162], RichViewEdit[316], DBRichView[407], DBRichViewEdit[410] controls.

But if you use the following:

- TCustomRichView.DeleteUnusedStyles[216];
- TDBRichViewEdit.AutoDeleteUnusedStyles[412]=*True*;
- TCustomRichView.LoadRVF[245], LoadRVFFromStream[244], and RVFTextStylesReadMode[191] or RVFParaStylesReadMode[190] <> *rvf_sIgnore*;

- TDBRichView or TDBRichViewEdit, and RVFTextStylesReadMode[191] or RVFParaStylesReadMode[190] <> *rvf_sIgnore*

you must use one RVStyle for one RichView.

Design-Time Component Editor

TRVStyle has a design-time component editor (unit RVSEdit). This editor:

- adds 3 items into the context menu for component:"Edit Text Styles", "Edit Paragraph Styles", "Edit List Styles" which invoke property editors for TextStyles[437], ParaStyles[431], or ListStyles[429];
- adds "Convert to twips" or "Convert to pixels" item in the context menu, allowing to convert all RVStyle properties measured in Units[437].
- adds 3 items into the context menu for component:"Edit Style Templates", "Save Style Templates...", "Load Style Templates..." which invoke property editors for StyleTemplates[435], save them to a file, and load them from a file.
- invokes property editor for TextStyles on double-click.

Properties

Three main properties of RVStyle are:

- TextStyles[437] – collection of text attributes (styles), a collection of TFontInfo[478] items;
- ParaStyles[431] – collection of paragraph attributes (styles), a collection of TParaInfo[490] items;
- ListStyles[429] – collection of paragraph list styles (bullets and numbering), a collection of TRVListInfo[493] items.

Style templates ("real" styles)

- StyleTemplates[435] – collection of "real" styles, controlling properties of TextStyles[437] and ParaStyles[431];
- MainRVStyle[430] allows using a single StyleTemplates[435] in multiple RVStyle components.

Units of measurement

- Units[437] defines measure units for properties of this component and documents in linked controls.

RVStyle allows to set some colors for associated controls:

- Color[421] – background color;
- SelColor, InactiveSelColor[426] – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor[426] – colors of selected text (with and without focus);
- DisabledFontColor[422] – text color for disabled controls.
- HoverColor[426] – default color of hypertext links under the mouse pointer;
- CheckpointColor, CheckpointEvColor[420] – colors of *checkpoints* (if visible);
- PageBreakColor[431] – color of explicit page breaks (if visible);
- FloatingLineColor[423] – color of placeholders for left- and right-aligned items (if visible).
- GridColor, GridReadOnlyColor[425] – colors of table grid lines (if visible); GridStyle, GridReadOnlyStyle[425] define their pen styles.

RVStyle allows to set cursors:

JumpCursor[427] – hypertext cursor for *hotspots*[129] and *hot-pictures*[125] (and default cursor for text hypertext items);
LineSelectCursor[427] – cursor for the left margin.

Tabs:

- SpacesInTab[435] – If positive, **Add*** ** methods of RichView, methods for loading text and RTF, keyboard input will replace tab characters with the specified number of spaces. If 0 (default), a special tabulator item type[121] is used;
- DefTabWidth[422] – default distance between tab stops.

Properties related to label items[133] and item types inherited from them:

- FieldHighlightColor[423] – color for highlighting label items.
- FieldHighlightType[423] specifies when to highlight label items.
- FootnoteNumbering[422] – numbering type for footnotes.
- FootnotePageReset[425] – numbering mode for footnotes.
- EndnoteNumbering[422] – numbering type for endnotes.

- SidenoteNumbering 422 – numbering type for sidenotes.

Properties related to selection:

- SelectionStyle 432 – visual appearance of selection;
- SelectionMode 432 – modes of selection;
- SelectionHandleKind 431 – visual appearance of touch screen selection handles;
- SelColor, InactiveSelColor 426 – background colors of selected text (with and without focus);
- SelTextColor, InactiveSelTextColor 426 – colors of selected text (with and without focus).

Other properties

- TextEngine 436 specifies which API is used for drawing and measuring text;
- FontQuality 424 – font quality and anti-aliasing;
- DefCodePage 421 – default code page for internal ANSI <->Unicode conversion;
- LineWrapMode 428 specifies a line wrapping algorithm;
- UseSound 437.

Methods

You can save and load all information in RVStyle in ini-file:

- SaveINI 442 saves RVStyle properties in ini-file;
- LoadINI 440 loads RVStyle properties from ini-file;

or registry:

- SaveReg 442 saves RVStyle properties in ini-file;
- LoadReg 441 loads RVStyle properties from ini-file.

You can export styles in style sheet (for saving HTML files):

- SaveCSS 441;
- SaveCSSToStream 442.

You can search for a style with the specified properties:

- FindTextStyle, FindParaStyle 440

Events

Custom drawing:

- OnDrawCheckpoint 444 – drawing checkpoint;
- OnDrawPageBreak 446 – drawing page break;
- OnDrawParaBack 447 – drawing background of paragraph;
- OnDrawTextBack 449 – drawing background of text;
- OnApplyStyle 443 – applying text style to Canvas;
- OnApplyStyleColor 444 – applying text style color to Canvas;
- OnDrawStyleText 448 – drawing text of the given style;
- OnStyleHoverSensitive 450 – "should the component be repainted when user moves the mouse pointer over hyperlink of the given style?".

1.5.2.1.1 Properties

1.5.2.1.1.1 TRVStyle.CheckpointColor, CheckpointColor

A color of dotted lines showing positions of *checkpoints* in document.

```
property CheckpointColor: TColor;
property CheckpointEvColor: TColor;
```

If TRichView shows its *checkpoints* (*rvoShowCheckpoints* in TRichView.Options 182), they are displayed as dotted horizontal lines with a small circle at the top left corner of the associated item.

Normal *checkpoints* are shown using **CheckpointColor**. *Checkpoints* with "RaiseEvent" flag are shown using CheckpointEvColor.

You can draw *checkpoints* yourself in OnDrawCheckpoint 444 event.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

Default value:

CheckpointColor: clGreen
CheckpointEvColor: clLime

See also events:

- OnDrawCheckpoint[444].

See also properties of TCustomRichView:

- Options[182].

See also:

- *Checkpoints*[65] overview[65].

1.5.2.1.1.2 TRVStyle.Color

A default background color for TRichView component linked to this TRVStyle component.

```
property Color: TColor;
```

This property may be overridden with TRichView.Color[173] property.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

See also properties of TCustomRichView:

- Style[193];
- Color[173];
- BackgroundStyle[170];
- BackgroundBitmap[169].

1.5.2.1.1.3 TRVStyle.CurrentItemColor

A color for drawing frame around the current item in TRichViewEdit[316].

```
property CurrentItemColor: TColor;
```

(Introduced in v1.9)

Current item is an item at the position of caret. All RichViewEdit's **GetCurrent***** and **SetCurrent***** methods work with this item.

Set this property to *clNone* to disable this effect.

TRichViewEdit must be linked with this TRVStyle (see TRichView.Style[193]).

Default value:

clNone

1.5.2.1.1.4 TRVStyle.DefCodePage

A default code page used inside TRichView for automatic conversions of ANSI strings to Unicode and vice versa

```
property DefCodePage: TRVCodePage[671];
```

(Introduced in version 1.4)

This property defines the code page (language) which will be used for ANSI↔Unicode conversions.

Usually, if conversion includes only one TRichView text item, the code page for conversion is defined by the character set[468] of the text style. If the conversion is performed on a block of text consisting of several items, TRichView gets a code page for the conversion from this property.

Note: there is a special property editor for this property allowing to select code page from drop-down list.

Default value:

0 (Windows default code page, *CP_ACP*)

See also:

- Unicode in RichView [99].

1.5.2.1.1.5 TRVStyle.DefTabWidth

A default distance between tab stops

```
property DefTabWidth: TRVStyleLength [690];
```

(introduced in v1.9)

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units [437].

Tab stops are started from the LeftMargin [180] (or from the RightMargin [185] for right-to-left [100] paragraphs).

Note: tabulators [121] can be inserted from text/RTF files or keyboard only if value of SpacesInTab [435] property is 0 (otherwise tabs are inserted as several space characters).

Default value:

48

See also properties:

- ParaStyles [431].Tabs [489].

See also properties:

- Units of measurement in TRichView [106]

1.5.2.1.1.6 TRVStyle.DefUnicodeStyle

```
property DefUnicodeStyle: Integer;
```

Obsolete property.

Default value:

-1

1.5.2.1.1.7 TRVStyle.DisabledFontColor

Defines a font color to use in disabled TRichView [162] controls.

```
property DisabledFontColor: TColor;
```

(Introduced in version 13)

If this property = *clNone*, a disabled control uses the same text color as enabled control.

Default value:

clNone

1.5.2.1.1.8 TRVStyle.EndnoteNumbering

A numbering type for endnotes [135], footnotes [136], and sidenotes [137].

```
property EndnoteNumbering: TRVSeqType
property FootnoteNumbering: TRVSeqType
property SidenoteNumbering: TRVSeqType
```

(introduced in version 10 and 15)

See TRVSeqType [689] for possible values.

If you changed value of these properties while the document in the linked [193] TRichView [162] control is displayed, call RichView.RefreshSequences [248] and RichView.Format [217].

Default values:

- EndnoteNumbering: *rvseqLowerRoman*
- FootnoteNumbering: *rvseqDecimal*
- SidenoteNumbering: *rvseqLowerAlpha*

See also:

- FootnotePageReset[425].

1.5.2.1.1.9 TRVStyle.FieldHighlightColor

A color for highlighting labels[133], numbered sequences[134], endnotes[135], footnotes[136], references to parent footnotes and endnotes[140].

```
property FieldHighlightColor: TColor;
```

(introduced in version 10)

See FieldHighlightType[423] for details.

Default value:

clBtnFace

1.5.2.1.1.10 TRVStyle.FieldHighlightType

Specifies when FieldHighlightColor[423] must be used for highlighting labels[133], numbered sequences[134], endnotes[135], footnotes[136], references to parent footnotes and endnotes[140].

```
type
  TRVFieldHighlightType =
    (rvfhNever, rvfhCurrent, rvfhAlways);
property FieldHighlightType;
```

(introduced in version 10)

Type	Meaning
rvfhNever	Items are never highlighted.
rvfhCurrent	Item at the position of the caret is highlighted. Either the editor must have the input focus, or its ForceFieldHighlight[327] property must be *True*.
rvfhAlways	All items are always highlighted.

Default value:

rvfhCurrent

1.5.2.1.1.11 TRVStyle.FloatingLineColor

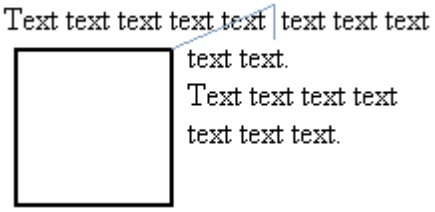
Color for displaying places of insertion for left- and right-aligned[695] items, and markers for text box items[139].

```
property FloatingLineColor: TColor;
```

(introduced in version 12)

Left- and right-aligned items

Two lines are drawn: a vertical line in place of the placeholder (where this object is inserted), and a line from the top of placeholder to the nearest corner of the floating object.



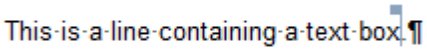
The lines are displayed in the following cases:

- in TCustomRichViewEdit[316], for one item, when this item is current (at the position of the caret)
- for all side-aligned items, if *rvoShowSpecialCharacters* is included in Options[182], and *rvcFloatingLines* is included in RVVisibleSpecialCharacters[714]

Assign *clNone* to prevent displaying these lines.

Text box items

A special mark is inserted in places of insertion of text box items[139]:



The marks are displayed in the following cases:

- in TCustomRichViewEdit[316], for one item, when this item is current (at the position of the caret)
- for all text box items, if *rvoShowSpecialCharacters* is included in Options[182], and *rvcPlaceholders* is included in RVVisibleSpecialCharacters[714].

If FloatingLineColor=*clNone*, SpecialCharactersColor[435] is used. If SpecialCharactersColor=*clNone*, *clBtnShadow* is used.

ScaleRichView note: in TSRichViewEdit, this color is also used to draw lines to places of insertion of sidenotes[137] and text box items, and around the edited floating box. Assign *clNone* to prevent displaying these lines.

Default value:

$BDA28A

1.5.2.1.1.12 TRVStyle.FontQuality

Specifies the font quality.

```
type
  TFontQuality = (fqDefault, fqDraft, fqProof, fqNonAntialiased, fqAntialiased,
    fqClearType, fqClearTypeNatural);
property FontQuality: TFontQuality;
```

(introduced in version 15)

Type	Meaning
fqDefault	The quality is determined by the system settings.
fqDraft	For raster fonts, scaling is enabled; the font size can be increased but the quality may be lower. Also the font supports Bold, Italic, Underline or Strikeout if necessary. The quality is less important than when Proof is used.
fqProof	The quality of the characters is important, so for the raster fonts scaling is disabled and the font closest in size is chosen.
fqNonAntialiased	Fonts are never antialiased.

fqAntialiased	Fonts are always antialiased if they support it. The size of the font cannot be too small or too large.
fqClearType	Fonts are rendered using the ClearType antialiasing method.
fqClearTypeNatural	Fonts are rendered using the ClearTypeNatural antialiasing method.

Note: TFontQuality is defined in RVStyle.pas for Delphi versions prior to XE; for newer versions of Delphi, UITypes.TFontQuality is used instead.

Default value:

fqDefault

1.5.2.1.1.13 TRVStyle.FootnotePageReset

Specifies whether to reset numbering of footnotes[136] on each page.

```
property FootnotePageReset: Boolean;
```

(introduced in version 10)

This property affects only printing. Footnotes in TRichView are still numbered sequentially.

This property can be changed when RTF file is loaded, if RichView.RTFReadProperties[187].IgnoreNotes[307] =*False*.

Default value:

True

See also:

- FootnoteNumbering[422]

1.5.2.1.1.14 TRVStyle.FullRedraw

For internal use

```
property FullRedraw: Boolean;
```

This property is temporal.

If FullRedraw=*True*, TRichView is redrawn completely when user moves the mouse pointer to the hyperlink area that needs to be highlighted with hovercolor[470].

If FullRedraw=*False*, TRichView redraws only the highlighted part (with additional 5 pixels border).

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

Default value:

False

1.5.2.1.1.15 TRVStyle.GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle

Colors and styles of table[130] grid lines.

```
property GridColor: TColor;
property GridReadOnlyColor: TColor;
property GridStyle: TPenStyle;
property GridReadOnlyStyle: TPenStyle;
```

(introduced in version 14)

GridColor and GridStyle are used for drawing invisible borders of tables in editing mode.

GridReadOnlyColor and GridReadOnlyStyle are used for drawing invisible borders of tables in TRichView[162] or in TRichViewEdit[316] in read-only[328] mode.

Table borders can be invisible in the following cases:

- their width is 0 (see BorderWidth[571] and CellBorderWidth[572] properties);
- some sides are hidden (see VisibleBorders[581] and Cell[605].VisibleBorders[613] properties).

If a pen style is *psClear*, or a color is *clNone*, a grid is not shown. But the recommended way to hide grid lines in all tables in the component is excluding *rvoShowGridLines* from TRichView.Options[182].

You can hide/show table grid for the specific table, see *rvtoHideGridLines* in Table.Options[577] property.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

Default values:

- GridColor: *clBtnFace;*
- GridReadOnlyColor: *clBtnFace;*
- GridStyle: *psDot;*
- GridReadOnlyStyle: *psClear.*

1.5.2.1.1.16 TRVStyle.HoverColor

Default text hover color.

```
property HoverColor: TColor;
```

Text hover color is a text color under the mouse pointer.

All hypertext styles have their own hover color (see TFontInfo.HoverColor[470]). If they are equal to *clNone*, this HoverColor is used.

Set to *clNone* to disable hover color effect.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

Default value:

clNone

1.5.2.1.1.17 TRVStyle.InactiveSelColor, InactiveSelTextColor, SelColor, SelTextColor

Color of text and background of selected text in TRichView components.

```
property SelColor: TColor;
property SelTextColor: TColor;
property InactiveSelColor: TColor;
property InactiveSelTextColor: TColor;
```

("Inactive" properties are introduced in version 1.6)

SelColor and SelTextColor are used when the control is focused. InactiveSelColor and InactiveSelTextColor are used otherwise.

Background colors (SelColor and InactiveSelColor) are used for:

- drawing background for the selected text;
- shading the selected *bullets*[128] and *hotspots*[129],
- drawing a frame around the selected *breaks*[125],
- drawing background for the selected pictures[122] and controls[126],
- drawing background for selected table[130] cells.

Set InactiveSelColor and InactiveSelTextColor to *clNone* to hide a selection when the input focus shifts to another control.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

Default values:

- SelColor: *clHighlight*
- SelTextColor: *clHighlightText*
- InactiveSelColor: *clHighlight*

- InactiveTextColor: *clHighlightText*

See also properties:

- SelectionStyle 432.

1.5.2.1.1.18 TRVStyle.InvalidPicture

Picture that will be inserted when reading RTF or RVF containing invalid or missing pictures.

```
property InvalidPicture: TPicture;
```

(Introduced in version 1.7)

Default value:

 (TBitmap)

See also properties of TCustomRichView:

- RVFWarnings 191.

1.5.2.1.1.19 TRVStyle.JumpCursor

Mouse cursor for hypertext indication.

```
property JumpCursor : TCursor
```

This cursor appears when user moves the mouse pointer to *hotspot* 129 or *hot-picture* 125 item. In older versions of TRichView this value was also used for text hypertext items, but now all text styles can have their own hypertext cursors (see TFontInfo.JumpCursor 472).

TRichView must be linked with this TRVStyle (see TRichView.Style 193).

Default value:

crJump (101) (system hypertext cursor or )

See also:

- Cursors used by the components of RichView family 108.

1.5.2.1.1.20 TRVStyle.LineSelectCursor

Mouse cursor for line selection

```
property LineSelectCursor : TCursor
```

(introduced in v1.8)

This cursor is displayed when the mouse pointer is above the left margin.

This cursor is used in the special text selection mode: line selection. Line selection is started if the user started selecting with the mouse from the left margin.

Line selection does not work for right-to-left documents 170.

TRichView must be linked with this TRVStyle (see TRichView.Style 193).

You can disable line selection if you set this property to *crNone* (-1).

Default value:

 *crRVFlipArrow* (106)

See also properties of TCustomRichView:

- LeftMargin 180.

See also:

- Cursors used by the components of RichView family [108].

1.5.2.1.1.21 TRVStyle.LineWrapMode

Specifies how text lines are wrapped.

```
type
  TRVLineWrapMode = (rvWrapAnywhere, rvWrapSimple, rvWrapNormal);
property LineWrapMode: TRVLineWrapMode
```

(introduced in version 13)

Value	Line wrapping algorithm
rvWrapAnywhere	Line break can occur in any place of text
rvWrapSimple	Line breaks inside text items [120] are added using line breaking rules. Line breaks between text items [120] can be added between any text items.
rvWrapNormal	Line breaks both inside and between text items are added using line breaking rules.

Line breaking rules are different for ANSI and Unicode [99] text items.

For ANSI text, line breaks occur only on space characters.

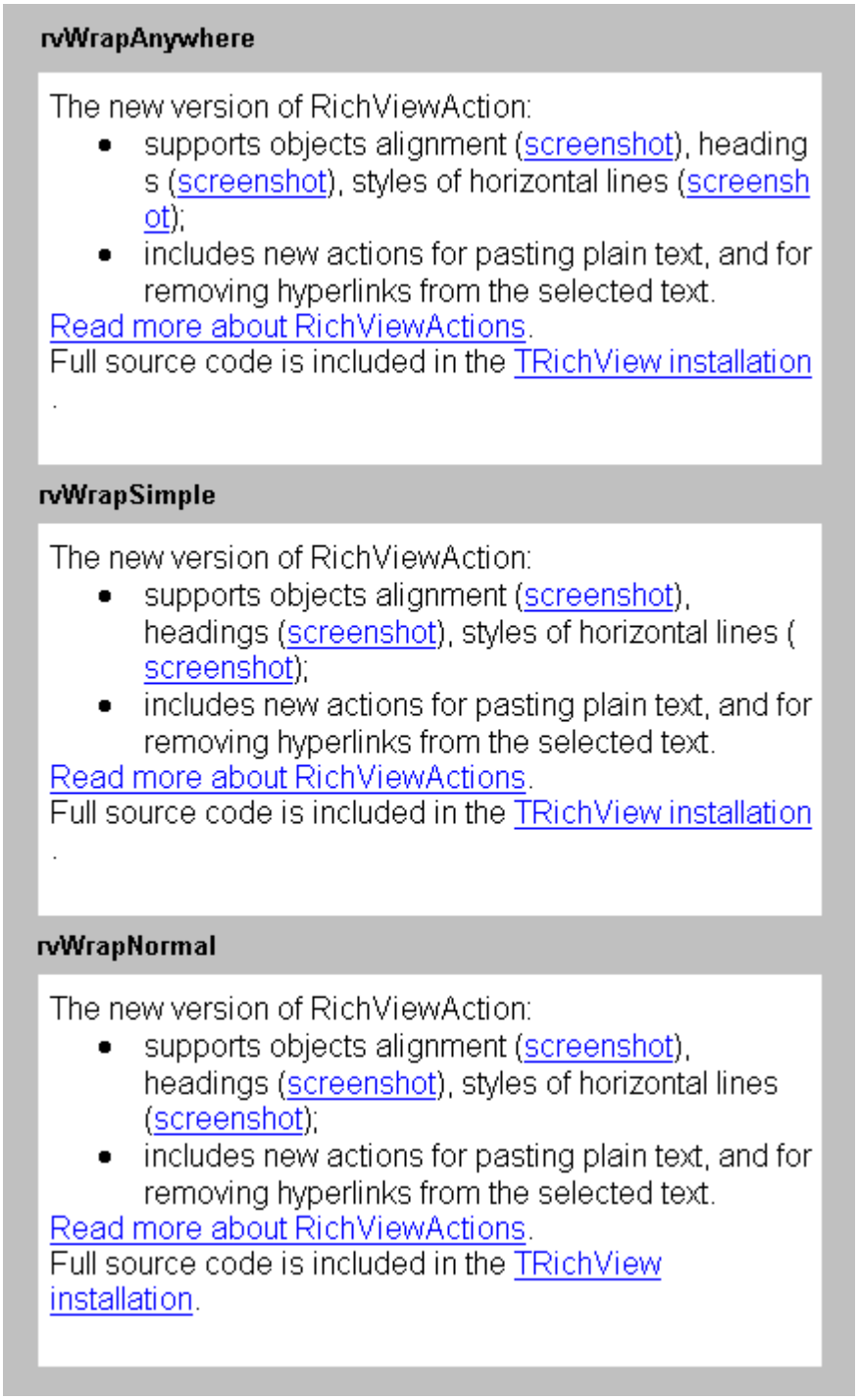
For Unicode text, line breaks are added using advanced rules. For example, in the string '(a)(b)' the only possible line break is between ')' and '('.

rvWrapAnywhere is a special mode.

rvWrapSimple was used in TRichView versions prior to 13. It is faster than *rvWrapNormal*.

rvWrapNormal is the default and recommended value.

Example



Default value:

rvWrapNormal

1.5.2.1.1.22 TRVStyle.ListStyles

This is one of the key properties of TRVStyle: collection of paragraph list styles (bullets and numbering).

```
property ListStyles : TRVListInfos[458];
```

Bullets/numbering (lists) are implemented in the following way:

1. TRVStyle has a collection of list styles (this property). Each list style[493] contains a collection of list levels (Levels[493] property). A list level[506] defines the most of list properties.

2. List markers[131] is a special type of items. They are inserted with the special functions. They are always inserted at the beginning of paragraphs. They have the following properties:

- ListNo – index of list style in the collection of list styles (i.e. in this property);
- LevelNo – list level;
- StartFrom – starting value for list counter, if UseStartFrom=*True*;
- UseStartFrom – if *True*, list counter value for this marker is defined by StartFrom. If *False*, numbering is continued.

See also properties:

- TextStyles[437];
- ParaStyles[431].

See also types:

- TRVListInfos[458] (collection of list styles);
- TRVListInfo[493] (item in the collection of list styles).

See also methods of TCustomRichView:

- SetListMarkerInfo[269],
- GetListMarkerInfo[233].

See also methods of TCustomRichViewEdit:

- ApplyListStyle[331],
- RemoveLists[367],
- ChangeListLevels[338].

See also:

- List markers item type[131].

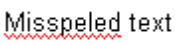
1.5.2.1.1.23 TRVStyle.LiveSpellingColor

Color of live spelling wavy underlines.

```
property LiveSpellingColor: TColor;
```

(introduced in v1.9)

Example:



Default value:

clRed

See also:

- Live spelling check in RichView[101].

1.5.2.1.1.24 TRVStyle.MainRVStyle

Defines the master RVStyle component for this component.

```
property MainRVStyle: TRVStyle[418];
```

(introduced in version 14)

If this **MainRVStyle** is assigned, StyleTemplates[435] returns **MainRVStyle**.StyleTemplates[435]. It allows several editors working with the same collection of style templates.

If some TRichView component is used as a header[264]/footer[263] (and style templates are used[195]) it must have Style[193].MainRVStyle assigned equal to Style[193] of the main editor.

ScaleRichView note: TSRichViewEdit provides that srve.RichViewEdit.Style = srve.RVHeader.Style.MainRVStyle = srve.RVFooter.Style.MainRVStyle.

1.5.2.1.1.25 TRVStyle.PageBreakColor

Color of explicit (user-defined) page breaks in TRichView components.

```
property PageBreakColor: TColor;
```

This color is used for drawing horizontal lines (with "folded paper" icon) specifying explicit (user-defined) page breaks. You can change this drawing in OnDrawPageBreak 446 event.

Explicit page breaks are added by assigning *True* to TRichView.PageBreaksBeforeItems 184[], or using TRichViewEdit.InsertPageBreak 355.

TRichView must be linked with this TRVStyle (see TRichView.Style 193).

Default value:

clBtnShadow

See also properties:

- SoftPageBreakColor 434.

See also events:

- OnDrawPageBreak 446.

See also properties of TCustomRichView:

- Options 182 (*rvoShowPageBreaks*).

1.5.2.1.1.26 TRVStyle.ParaStyles

This is one of the key properties of TRVStyle: collection of paragraph styles.

```
property ParaStyles : TParaInfos 456;
```

Paragraph styles define paragraph attributes in documents.

See also properties:

- TextStyles 437;
- ListStyles 429.

See also types:

- TParaInfos 456 (collection of paragraph styles);
- TParaInfo 482 (item in collection of paragraph styles).

See also:

- Paragraphs in RichView 72.

1.5.2.1.1.27 TRVStyle.SelectionHandleKind

Defines visual appearance of selection handles for touch screens

```
property SelectionHandleKind: TRVSelectionHandleKind;

type
  TRVSelectionHandleKind = (rvshkWedge, rvshkCircle);
```

(introduced in v15)

This property exists when the component is compiled in Delphi 2010 or newer.

Value	Appearance
rvshkWedge	

rvshkCircle	

rrvshkCircle is not recommended, because it may overlap selected text in table cells; you can use it in ScaleRichView, because TSRichViewEdit does not shift handles to fit in cells.

Default value:

rvshkWedge

See also properties of TCustomRichView:

- SelectionHandlesVisible 192

1.5.2.1.1.28 TRVStyle.SelectionMode

Defines how the selection is guided by TRichView components.

```
property SelectionMode: TRVSelectionMode;

type
  TRVSelectionMode =
    (rvsmChar, rvsmWord, rvsmParagraph);
```

(introduced in v 1.8)

Selection Mode	Meaning
rvsmChar	No special processing for selection: selection starts where the user pressed the left mouse button, selection ends where the user released it.
rvsmWord	Selecting by words. Selection is smart: you can still select a part of word by moving mouse pointer in backward direction. This mode affects only selecting by moving mouse.
rvsmParagraph	Special selecting mode: if items of multiple paragraphs are selected, the component selects these paragraphs completely. The mode affects selection by moving mouse, Shift+click, keyboard.

Default value:

rvsmWord

1.5.2.1.1.29 TRVStyle.SelectionStyle

Defines visual appearance of selection in TRichView components.

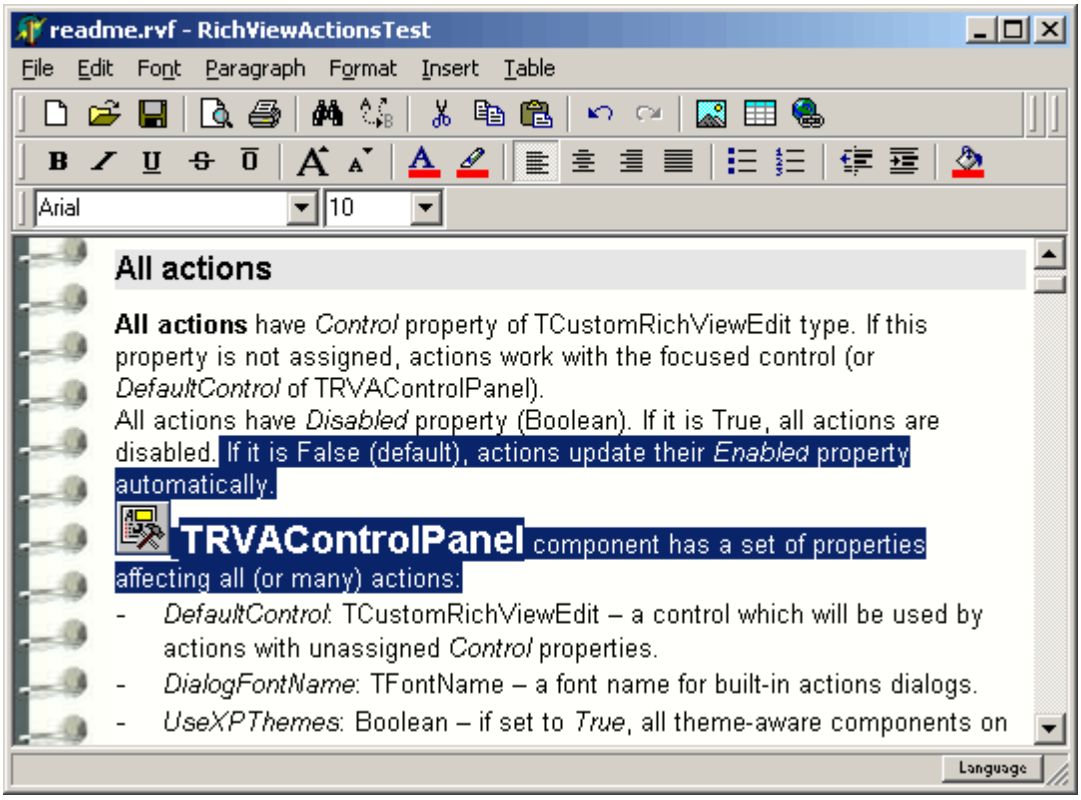
```
property SelectionStyle: TRVSelectionStyle;

type
  TRVSelectionStyle = (rvssItems, rvssLines);
```

(introduced in v 1.8)

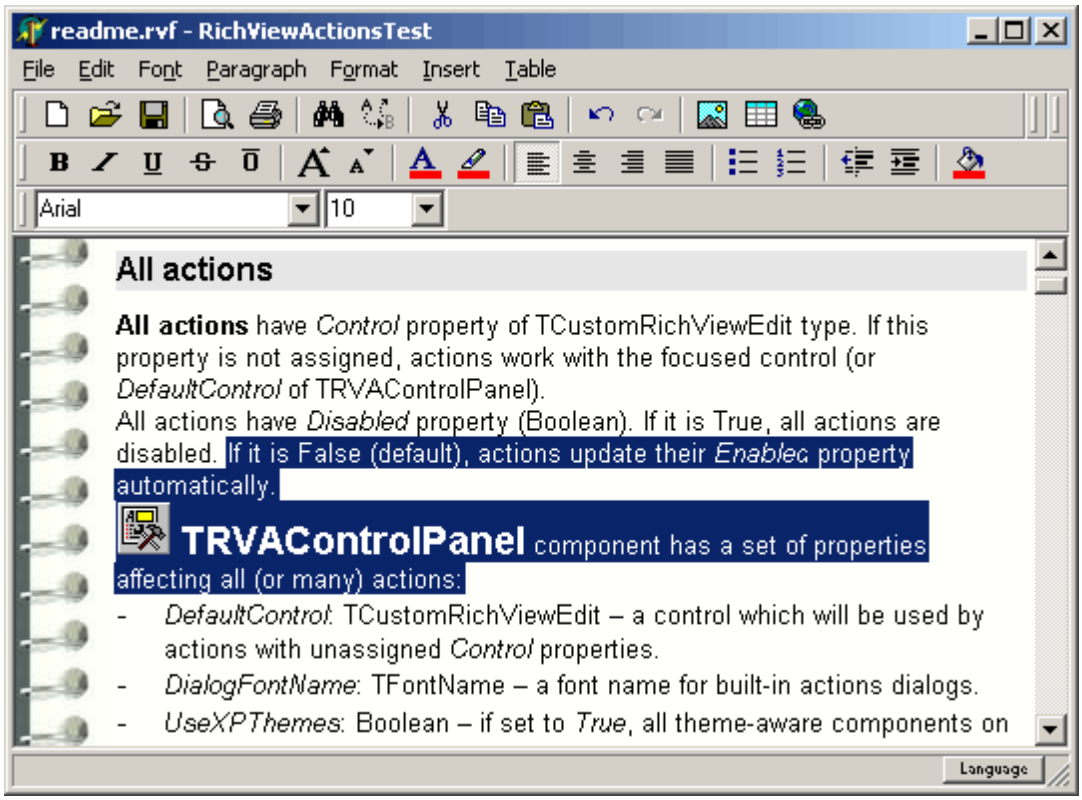
Selection Style	Meaning
rvssItems	Each item draws its selection separately.

rvssLines	The selection looks more like in MS Word (selection has height of the highest item in the line, including spacing); this mode works only if BiDiMode[100]=*rvbdUnspecified*.

rvssItems:



rvssLines:



Default value:

rvssItems

See also properties:

- SelColor, SelTextColor, InactiveSelColor, InactiveSelTextColor 426;
- SelectionMode 432.

1.5.2.1.1.30 TRVStyle.SoftPageBreakColor

Color of soft page breaks in TRichView components.

```
property PageBreakColor: TColor;
```

This color is used for drawing horizontal lines (with "folded paper" icon) specifying soft (automatic) page breaks. You can change this drawing in OnDrawPageBreak 446 event.

TRichView must be linked with this TRVStyle (see TRichView.Style 193). Soft page breaks displaying is turned on by TRichView.AssignPageBreaks 209.

Default value:

clBtnFace

See also properties:

- PageBreakColor 431.

See also events:

- OnDrawPageBreak 446.

See also properties of TCustomRichView:

- Options 182 (*rvoShowPageBreaks*).

1.5.2.1.1.31 TRVStyle.SpacesInTab

If set to positive value, TRichView components will replace TAB characters with the specified number of space characters.

```
property SpacesInTab: SmallInt;
```

This property is used in all methods of TRichView[162] and TRichViewEdit[316] for adding/inserting multiline text and RTF, or on typing.

TRichView must be linked with this TRVStyle (see TRichView.Style[193]).

If this property is 0, TABs are inserted as a special tabulator item type[121].

Note: default value is changed to 0 since v1.9

Default value:

0

See also properties:

- DefTabWidth[422].

1.5.2.1.1.32 TRVStyle.SpecialCharactersColor

Color of special characters.

```
property SpecialCharactersColor: TColor;
```

(introduced in version 12)

if *rvoShowSpecialCharacters* is included in TRichView.Options[182], this color is used for drawing marks denoting space characters and paragraph breaks.

If it equals to *clNone*, a normal text color is used instead.

Default value:

clNone

See also:

- RVVisibleSpecialCharacters[714] global typed constant
- FloatingLineColor[423]

1.5.2.1.1.33 TRVStyle.StyleTemplates

A collection of real styles, controlling text attributes from TextStyles[437] and paragraph attributes from ParaStyles[431].

```
property StyleTemplates : TRVStyleTemplateCollection[460];
```

Using style templates is optional, they are used only for richview controls having UseStyleTemplates[195]=*True*.

If MainRVStyle[430] is assigned, this property returns MainRVStyle[430].**StyleTemplates**.

See also

- Styles and style templates[74].

1.5.2.1.1.34 TRVStyle.TextBackgroundKind

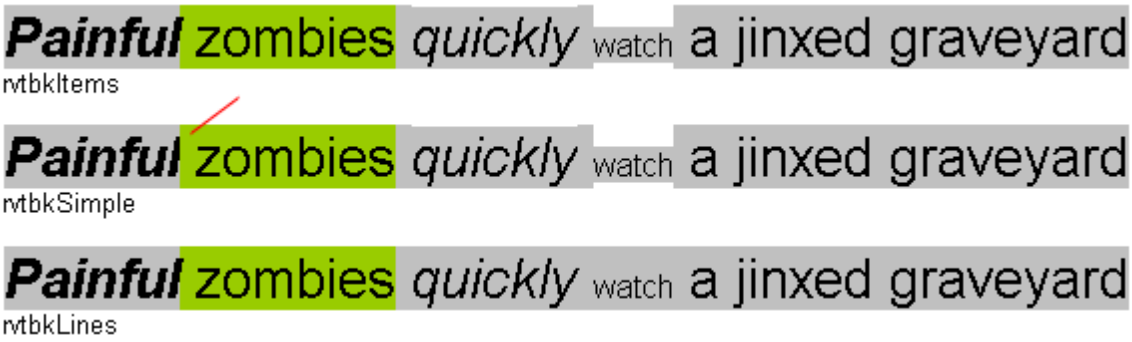
Defines visual appearance of text background in TRichView components.

```
type
  TRVTextBackgroundKind = (rvtbkSimple, rvtbkItems, rvtbkLines);
property TextBackgroundKind: TRVTextBackgroundKind;
```

(introduced in v14)

Selection Style	Background is drawn...	Background height =
rvtbkItems	before text	item height
rvtbkSimple	together with text	item height
rvtbkLines	before text	line height

Example:



rvtbkSimple is not recommended, because the item text may be overlapped by a background of the next item, especially for italic fonts (see the red line in the example above).

Default value:

rvtbkItems

See also:

- OnDrawTextBack 449

See also properties of text style:

- BackColor 467
- HoverBackColor 470;

1.5.2.1.1.35 TRVStyle.TextEngine

Specifies which set of functions are used for text measuring and drawing.

```
type
  TRVTextEngine = (rvteWindows, rvteUniscribe);
property TextEngine: TRVTextEngine;
```

(introduced in v16)

Value	Meaning	Comments
rvteWindows	Windows API	**Pros:** • faster, especially if the support of bidirected text is turned off. **Cons:** • completely incorrect results for right-to-left text (Arabic and Hebrew), if the support of bidirected text is turned off; • results may be not completely correct when text includes some "exotic" characters; • unreliable results when the support of bidirected text is turned on (may be correct for one font, but incorrect for another font).
rvteUniscribe	***Uniscribe***	**Pros:** • reliable processing of complex scripts and "exotic" characters;

		• acceptable results for right-to-left text even if support of bidirected text is turned off **Cons:** • usually slower

Reformat documents in linked TRichView controls after changing value of this property.

Default value:

rvteUniscribe

1.5.2.1.1.36 TRVStyle.TextStyles

This is one of key properties of TRVStyle: a collection of text attributes (styles).

```
property TextStyles : TFontInfos[451];
```

Text styles define fonts and other text attributes in documents.

The first text style in the collection (TextStyles[0]) has some additional meaning:

- *breaks*[64] with color = *clNone* (default value) have the color of this style;
- Charset[468] of this style is used for saving language information in HTML files (D3+);
- in some cases this style is used as default, so it's recommended to use it as a style of normal text; please do not set *rvprDoNotAutoSwitch* in Protection[473] for this style;
- this style can be hypertext, but it is not recommended.

See also properties:

- ParaStyles[431];
- ListStyles[429].

See also types:

- TFontInfos[451] (collection of text styles);
- TFontInfo[478] (item in collection of text styles).

1.5.2.1.1.37 TRVStyle.Units

Defines units of measurement for properties of TRVStyle and linked[193] controls.

```
property Units: TRVStyleUnits[691]
```

(introduced in version 13)

All values of type TRVStyleLength[690] are measured in units specified in this property.

Assignment to this property does not convert properties to new units. To change units of measurement, at first call one of ConvertDocTo*[211] methods of all TRichView[162] components linked[193] to this TRVStyle, then call one of ConvertTo*[438] methods to convert properties of this TRVStyle component to the same units of measurement.

At design time, you can convert units using TRVStyle's context menu ("Convert to Pixels" or "Convert to Twips" commands).

Default value:

rvstuPixels

See also properties:

- Units of measurement in TRichView[106] (contains a list of properties measured in Units)

1.5.2.1.1.38 TRVStyle.UseSound

Defines whether RichViewEdit[316] will use sound.

```
property UseSound: Boolean
```

(introduced in version 1.3)

If *True*, RichViewEdit plays a system default sound when user tries to modify read-only[328] or protected[473] text.

Default value:

True

1.5.2.1.2 Methods

1.5.2.1.2.1 TRVStyle.AddTextStyle

```
function AddTextStyle: Integer;
```

Obsolete. In order to add item in collection of text styles, use RVStyle.TextStyles.Add[452]

1.5.2.1.2.2 TRVStyle.Create

A constructor, creates a new TRVStyle component.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate TRVStyle component at runtime. TRVStyle components placed on forms at design time are created automatically. Specify the owner of the new RVStyle using the AOwner parameter.

1.5.2.1.2.3 TRVStyle.ConvertTo*

The methods convert all properties of this TRVStyle component to new units of measurement.

```
procedure ConvertToPixels;
procedure ConvertToTwips;
procedure ConvertToDifferentUnits(NewUnits: TRVStyleUnits[691]);
```

(introduced in version 13)

The methods convert all properties of this TRVStyle (including all sub-properties) from Units[437] to:

- screen pixels[106],
- twips[106],
- NewUnits (twips or pixels),

respectively.

After the conversion, a new value is assigned to Units[437] property.

If there is a document in TRichView control linked to this TRVStyle component, this document must be converted before calling this method.

When converting to/from pixels, RichViewPixelsPerInch[703] is taken into account.

1.5.2.1.2.4 TRVStyle.DeleteTextStyle

```
procedure DeleteTextStyle(Index: Integer);
```

Obsolete. In order to delete the i-th item from text styles, use TRVStyle.TextStyles[437][i].Free

1.5.2.1.2.5 TRVStyle.Destroy

A destructor, frees an instance of a TRVStyle component,

```
destructor Destroy;override;
```

Destroy seldom needs to be called. When TRVStyle is constructed without an owner by the Create method, Free should be called to release memory and dispose the object. Free checks to see if the pointer is nil before calling Destroy.

1.5.2.1.2.6 TRVStyle.GetAs*

The methods return **Value** converted to different units of measurement.

```
function GetAsPixels(Value: TRVStyleLength[690]): Integer;
function GetAsTwips(Value: TRVStyleLength[690]): Integer;
function GetAsDifferentUnits(Value: TRVStyleLength[690]; Units: TRVStyleUnits[691]): TRVStyleL
function GetAsRVUnits(Value: TRVStyleLength[690]; RVUnits: TRVUnits[694]): TRVLength[682];
```

(introduced in version 13)

The methods return **Value** converted from Units[437] to

- screen pixels[106],
- twips[106],
- another Units (twips or pixels),
- RVUnits (inches, mm, points, etc.)

respectively.

When converting to/from pixels, RichViewPixelsPerInch[703] is taken into account.

See also:

- *ToUnits[439] methods;
- ConvertTo*[438] methods.

1.5.2.1.2.7 TRVStyle.*ToUnits

The methods return **Value** converted from different units of measurement.

```
function PixelsToUnits(Value: Integer): TRVStyleLength[690];
function TwipsToUnits(Value: Integer): TRVStyleLength[690];
function DifferentUnitsToUnits(Value: TRVStyleLength[690]; Units: TRVStyleUnits[691]): TRVStyl
function RVUnitsToUnits(Value: TRVLength[682]; RVUnits: TRVUnits[694]): TRVStyleLength[690];
```

(introduced in version 13)

The methods return **Value** converted to Units[437] from

- screen pixels[106],
- twips[106],
- another Units (twips or pixels),
- RVUnits (inches, mm, points, etc.)

respectively.

When converting to/from pixels, RichViewPixelsPerInch[703] is taken into account.

See also:

- GetAs*[439] methods;
- ConvertTo*[438] methods.

1.5.2.1.2.8 TRVStyle.GetListStyleClass

Returns the class type representing a style of paragraph lists (bullets and numbering).

```
type
  TRVListInfoClass = class of TRVListInfo[493];

function GetListStyleClass: TRVListInfoClass; virtual;
```

(introduced in version 1.7)

This method can be used to create components inherited from TRVStyle. You can create a new class (inherited from TRVListInfo[493]) and use it in this descendant component.

See also:

- GetParaStyleClass[440];
- GetTextStyleClass[440].

1.5.2.1.2.9 TRVStyle.GetParaStyleClass

Returns the class type representing a style of paragraphs.

```
type
  TRVParaInfoClass = class of TParaInfo[482];

function GetParaStyleClass: TRVParaInfoClass; virtual;
```

(introduced in version 1.7)

This method can be used to create components inherited from TRVStyle. You can create a new class (inherited from TParaInfo[482]) and use it in this descendant component.

See also:

- GetListStyleClass[439];
- GetTextStyleClass[440].

1.5.2.1.2.10 TRVStyle.GetTextStyleClass

Returns the class type representing a style of text.

```
type
  TRVFontInfoClass = class of TFontInfo[478];

function GetTextStyleClass: TRVFontInfoClass; virtual;
```

(introduced in version 1.7)

This method can be used to create components inherited from TRVStyle. You can create a new class (inherited from TFontInfo[478]) and use it in this descendant component.

See also:

- GetListStyleClass[439];
- GetParaStyleClass[440].

1.5.2.1.2.11 TRVStyle.LoadINI

Loads all properties of TRVStyle component from ini-file

```
procedure LoadINI(const FileName, Section: String);
```

Parameters:

FileName – name of the ini-file;

Section – section of the ini-file.

See also:

- SaveINI[442];
- LoadReg[441].

1.5.2.1.2.12 TRVStyle.FindTextStyle, FindParaStyle

The methods search for a style like the style specified in the parameter, and add it if necessary.

```
function FindTextStyle(TextStyle: TFontInfo[478]): Integer;
function FindParaStyle(ParaStyle: TParaInfo[490]): Integer;
```

(introduced in version 14)

The methods are equivalent to:

```
TextStyles[437].FindSuchStyleEx[455](0, TextStyle, RVAllFontInfoProperties[679])
ParaStyles[431].FindSuchStyleEx[458](0, ParaStyle, RVAllParaInfoProperties[684])
```

1.5.2.1.2.13 TRVStyle.LoadReg

Loads all properties of TRVStyle component from the Registry (for Delphi4+).

```
procedure LoadReg(const BaseKey: String);
```

(Introduced in version 1.2)

Parameter:

BaseKey – the name of the HKEY_CURRENT_USER subkey to loads data from. LoadReg appends 'RVStyle' to this subkey. For example, if **BaseKey** is 'Software\My Company\My Program', it loads data from the 'Software\My Company\My Program\RVStyle' key.

Warning: these data in this section (BaseKey+'\RVStyle') must exists, otherwise all properties will be reset to default values.

See also:

- SaveReg[442];
- LoadINI[440].

1.5.2.1.2.14 TRVStyle.SaveCSS

Saves text and paragraph styles as CSS (Cascading Style Sheets) file.

```
type
  TRVSaveCSSOption = (
    rvcssOnlyDifference, rvcssIgnoreLeftAlignment,
    rvcssNoDefCSSStyle, rvcssUTF8, rvcssDefault0Style,
    rvcssMarkersAsText);
  TRVSaveCSSOptions = set of TRVSaveCSSOption;

function SaveCSS(const FileName: TRVUnicodeString[694];
  Options: TRVSaveCSSOptions): Boolean;
```

(changed in version 18)

Use this method if you want to save content of TRichView as HTML file (RichView.SaveHTMLEx[251]) with external CSS.

You can save CSS only once and use it for several HTML files (if they are linked to the same or compatible RVStyle components) .

This method allows to reduce size of HTML files.

The following options may be included in the **Options** parameter.

Option	Meaning
rvcssNoDefCSSStyle	Corresponds to *rvsoNoDefCSSStyle*[686] in the Options parameters of SaveHTMLEx[251]. It's not recommended to use this option.
rvcssDefault0Style	Properties of TextStyles[437][0] are not saved. Corresponds to *rvsoDefault0Style*[686] in the Options parameter of SaveHTMLEx[251].
rvcssMarkersAsText	Lists will saved as text, so this method creates an additional styles for text bullets.

	Corresponds to *rvsoMarkersAsText* 686 in the Options parameter of SaveHTMLEx 251.
rvcssUTF8	CSS is saved in UTF-8 encoding.

Please do not use other options.

There are no methods for loading CSS files.

See also:

- RichView work with HTML files 96.

1.5.2.1.2.15 TRVStyle.SaveCSSToStream

The same as SaveCSS 441, but accepts **Stream** parameter

```
procedure SaveCSSToStream(Stream: TStream; Options: TRVSaveCSSOptions);
```

(introduced in v1.1.4)

1.5.2.1.2.16 TRVStyle.SaveINI

Saves all properties of TRVStyle component in the ini-file

```
procedure SaveINI(const FileName, Section: String);
```

Parameters:

FileName – name of the ini-file;

Section – section of the ini-file

Warning: the section is cleared before saving.

Alternative way to store/load TRVStyle: using TStream.WriteComponent and ReadComponent.

See also:

- LoadINI 440;
- SaveReg 442;
- SaveCSS 441.

1.5.2.1.2.17 TRVStyle.SaveReg

Saves all properties of component to the Registry (for Delphi4+)

```
procedure SaveReg(const BaseKey: String);
```

(Introduced in version 1.2)

Parameter:

BaseKey – the name of the HKEY_CURRENT_USER subkey to saves data to. SaveReg appends 'RVStyle' to this subkey. For example, if **BaseKey** is 'Software\My Company\My Program', the method saves data to the 'Software\My Company\My Program\RVStyle' key.

Warning: all data in this key (BaseKey+'\RVStyle') will be removed before saving!

The saving format is the same as in SaveINI 442.

See also:

- LoadReg 441;
- SaveINI 442.

1.5.2.1.3 Events

1.5.2.1.3.1 TRVStyle.OnAfterApplyStyle

Occurs after RichView applies attributes of the **StyleNo**-th style to Canvas.

```
property OnAfterApplyStyle: TRVAfterApplyStyleEvent;

type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      StyleNo: Integer) of object;
```

(introduced in version 12)

In this event you can apply font (excluding colors) for the text style **StyleNo** to **Canvas**. Nothing should be drawn inside this event.

This event is similar to OnApplyStyle[443], but occurs after the default procedure. This event does not occur if OnApplyStyle[443] returned *False* in DoDefault parameter.

Input parameters:

Sender – TRVStyle generating this event;

Canvas – canvas for applying font attributes to.

See also:

- OnApplyStyleColor[444];
- OnDrawStyleText[448].

See also properties:

- TextStyles[437].

1.5.2.1.3.2 TRVStyle.OnApplyStyle

Occurs when RichView applies attributes of the **StyleNo**-th style to Canvas.

```
property OnApplyStyle: TRVApplyStyleEvent;

type
  TRVApplyStyleEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      StyleNo: Integer; var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

In this event you can apply font (excluding colors) for the text style **StyleNo** to **Canvas**.

Nothing should be drawn inside this event.

Input parameters:

Sender – TRVStyle generating this event;

Canvas – canvas for applying font attributes to.

DoDefault is equal to True.

Output parameters:

DoDefault – set to *False* to prevent the default style applying procedure and OnAfterApplyStyle[443] event.

See also:

- OnApplyStyleColor[444];
- OnDrawStyleText[448].

See also properties:

- TextStyles 437.

1.5.2.1.3.3 TRVStyle.OnApplyStyleColor

Occurs when RichView applies the color of the **StyleNo**-th style to **Canvas**.

```
property OnApplyStyleColor: TRVApplyStyleColorEvent;

type
  TRVApplyStyleColorEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      StyleNo: Integer; DrawState: TRVTextDrawStates 692;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

In this event you can apply font color for the **StyleNo**-th text style to **Canvas**.

Nothing should be drawn inside this event.

Input parameters:

Sender – TRVStyle generating this event.

Canvas – canvas for applying background and text colors to.

DrawState – states of this text item, see TRVTextDrawStates 692.

DoDefault is equal to *True*.

Output parameters:

DoDefault – set to *False* to prevent the default colors applying procedure.

See also:

- OnApplyStyle 443;
- OnDrawStyleText 448.

See also properties:

- TextStyles 437.

1.5.2.1.3.4 TRVStyle.OnDraw Checkpoint

Occurs when drawing *checkpoint* 65.

```
property OnDrawCheckpoint: TRVDrawCheckpointEvent;

type
  TRVDrawCheckpointEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      X,Y, ItemNo, XShift: Integer;
      RaiseEvent: Boolean; Control: TControl;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

You can use this event to draw your own visual representation of *checkpoints*.

This event occurs only if *rvoShowCheckpoints* is included in TCustomRichView(**Control**).Options 182.

Input parameters:

Sender – TRVStyle generating the event.

Canvas – canvas to paint.

X,**Y** – coordinates of the top left corner of the item marked with this *checkpoint*. If this is the last *checkpoint* in document (that is not associated with any item), **X** = -1.

ItemNo – index of this item in **Sender**.**RVData**. (**ItemNo** = -1 for *checkpoint* without item).

XShift – how much TRichView is scrolled horizontally (>=0);

RaiseEvent – "raise event" flag of the *checkpoint*.

Control – TRichView control.

DoDefault is equal to *True*.

"Hidden" input parameter:

Sender.**RVData**: TPersistent, document containing the item with this *checkpoint*. It should be typecasted to TCustomRVFormattedData[650] before using.

Output parameters:

DoDefault – set to *False* to prevent default drawing. The default procedure draws horizontal dotted line with small circle in (**X**,**Y**). The line color is either CheckpointColor or CheckpointEvColor[420], depending on **RaiseEvent**.

Using **ItemNo** and **Sender**.**RVData** you can get additional information about the item and the *checkpoint*.

For example:

```
uses ..., CRVFData;
...
var
  CheckpointData: TCheckpointData[667];
  CheckpointTag: TRVTag[692];
  CheckpointName: String;
begin
  if ItemNo>=0 then
  begin
    CheckpointData :=
      TCustomRVFormattedData[650](Sender.RVData).GetItemCheckpoint[226](ItemNo));
    TCustomRVFormattedData(Sender.RVData).GetCheckpointInfo[220](
      CheckpointTag, CheckpointName, RaiseEvent);
    ...
  end;
end;
```

See also properties:

- CheckpointColor, CheckpointEvColor[420].

See also:

- Checkpoints overview[65].

Demo project:

- Demos\Delphi\Assorted\Custom Draw\CustomDraw\
- Demos\CBuilder\Assorted\Custom Draw\CustomDraw\

1.5.2.1.3.5 TRVStyle.OnDrawPageBreak

Occurs when drawing page breaks.

```
property OnDrawPageBreak: TRVDrawPageBreakEvent;

type
  TRVPageBreakType = (rvpbSoftPageBreak, rvpbPageBreak);

  TRVDrawPageBreakEvent =
    procedure (Sender: TRVStyle;
      Canvas: TCanvas;
      Y, XShift: Integer;
      PageBreakType: TRVPageBreakType;
      Control: TControl;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3; modified in 1.7)

You can use this event to draw your own visual representation of page breaks.

This event occurs only if *rvoShowPageBreaks* is included in TCustomRichView(**Control**).Options[182].

Input parameters:

Sender – TRVStyle generating the event.

Canvas – canvas to paint.

PageBreakType – type of page break (either soft (automatic) page break or explicit (user-defined) page break).

Y – y coordinate of page break in TRichView window.

XShift – how much TRichView is scrolled horizontally (>=0).

Control – TRichView control.

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document where page break occurs. It should be typecasted to TCustomRVFormattedData[650] before using.

Sender.**ItemNo**: Integer, index of the item in **Sender**.**RVData**.

These parameters define one of the following:

- item starting a new page;
- table[565] with page breaks between rows.

Output parameter:

DoDefault – set to *False* to prevent default drawing. The default procedure draws horizontal line horizontal line with "dog-ear" effect. The line color is either PageBreakColor[431] or SoftPageBreakColor[434], depending on **PageBreakType**.

Using **Sender**.**RVData** you can get additional information about the item:

For example:

```
uses ..., CRVFData;
...
var ParaNo: Integer;
begin
  ParaNo := TCustomRVFormattedData[650](Sender.RVData).GetItemPara[229](Sender.ItemNo));
```

```
  ...
end;
```

See also properties:

- PageBreakColor 431;
- SoftPageBreakColor 434.

See also properties of TCustomRichView 162:

- PageBreaksBeforeItem 184.

See also methods of TCustomRichView 162:

- AssignSoftPageBreaks 209.

See also methods of TCustomRichViewEdit 316:

- InsertPageBreak 355;
- RemoveCurrentPageBreak 367.

See also properties of table 565:

- PrintOptions 579.

See also properties of table row 615:

- PageBreakBefore 616.

Demo project:

- Demos\Delphi\Assorted\Custom Draw\CustomDraw\
- Demos\CBuilder\Assorted\Custom Draw\CustomDraw\

1.5.2.1.3.6 TRVStyle.OnDrawParaBack

Occurs when painting a background of paragraph of the **ParaNo**-th style.

```
property OnDrawParaBack: TRVDrawParaRectEvent;

type
  TRVDrawParaRectEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      ParaNo: Integer; ARect: TRect;
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

You can use this event to draw your own background and/or border for some or all paragraphs.

Input parameters:

Sender – TRVStyle generating the event.

Canvas – canvas to paint.

ARect – rectangle where to paint.

ParaNo – index in the collection of paragraph styles (ParaStyles 431).

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData 650 before using.

Sender.**ItemNo**: Integer, index of the last item in the paragraph, in **Sender**.**RVData**.

Output parameters:

DoDefault – set to *False* to prevent default drawing.

See also properties:

- ParaStyles[431];
- TParaInfo[482].Background[484].

See also:

- RichView Paragraphs[72].

Demo project:

- Demos\Delphi\Assorted\Custom Draw\CustomDraw\
- Demos\CBuilder\Assorted\Custom Draw\CustomDraw\

1.5.2.1.3.7 TRVStyle.OnDrawStyleText

Allows to perform custom drawing of a text item of the **StyleNo**-th text style.

```
property OnDrawStyleText: TRVDrawStyleTextEvent

type
  TRVDrawStyleTextEvent = procedure (Sender: TRVStyle[418];
    const s: TRVUnicodeString[694]; Canvas: TCanvas; StyleNo: Integer;
    SpaceAtLeft, Left, Top, Width, Height
    {$IFDEF RVUSEBASELINE},BaseLine{$ELSE}{$IFDEF RVHIGHFONTS},DrawOffsY{$ENDIF}{$ENDIF}: I
    TextWidth, SpaceCount: Integer;
    DrawState: TRVTextDrawStates[692];
    Printing, PreviewCorrection, ForMetafile: Boolean;
    ColorMode: TRVColorMode[671]; DefBiDiMode: TRVBiDiMode[668];
    RefCanvas: TCanvas;
    LeftNoExpIndex, RightNoExpIndex: Integer;
    var DoDefault: Boolean) of object;
```

(introduced in version 1.3; new states in TRVTextDrawStates[692] since 1.4; new parameters since v16; changed in v18)

You can use this event to modify drawing of text of some or all text styles.

Input parameters:

The same is in TFontInfo[478].Draw[481]. You can call **Sender**.TextStyles[437][**StyleNo**].Draw(..) passing the even parameters.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData[650] before using.

Sender.**ItemNo**: Integer, index of the item in **Sender**.**RVData**.

Self.**OffsetInItem**: Integer, index of the first character in the item to draw.

These parameters allow to get additional information about the item being painted.

For example:

```
uses ..., CRVFData;
...
var ItemTag: TRVTag[692];
...
ItemTag := TCustomRVFormattedData[650](Sender.RVData).GetItemTag[230](Sender.ItemNo));
```

Output parameters:

DoDefault – set to *False* to prevent default drawing.

See also events:

- OnApplyStyle[443];
- OnApplyStyleColor[444];
- OnDrawTextBack[449].

See also properties:

- TextStyles[437].

Demo projects:

- Demos*\Assorted\Custom Draw\

1.5.2.1.3.8 TRVStyle.OnDrawTextBack

Occurs when drawing background of text of the **StyleNo**-th text style.

```
property OnDrawTextBack: TRVDrawTextBackEvent;

type
  TRVDrawTextBackEvent =
    procedure (Sender: TRVStyle; Canvas: TCanvas;
      StyleNo: Integer; Left, Top, Width, Height: Integer;
      DrawState: TRVTextDrawStates[692];
      var DoDefault: Boolean) of object;
```

(introduced in version 1.3)

You can use this event to modify drawing of text background for some or all text styles.

Unlike OnDrawStyleText[448], this event is called only once for each drawing item (OnDrawStyleText[448] can be called several times if part of text item is selected). In paragraphs having Alignment[483]=*rvaJustify*, each word is a drawing item, but OnDrawTextBack is called only once per line, covering all area below this text item.

Input parameters:

Sender – TRVStyle generating the event.

Canvas – canvas to paint.

StyleNo – index of text style in TextStyles[437] collection.

Left, **Top**, **Width**, **Height** – rectangle where to paint.

DrawState may include *rvtsControlFocused* and *rvtsHover*, see , see TRVTextDrawStates[692].

DoDefault is equal to *True*.

"Hidden" input parameters:

Sender.**RVData**: TPersistent, document containing this paragraph. It should be typecasted to TCustomRVFormattedData[650] before using.

Sender.**ItemNo**: Integer, index of the item in **Sender**.**RVData**.

These parameters allow to get additional information about the item being painted.

For example:

```
uses ..., CRVFData;
...
var ItemTag: TRVTag[692];
...
ItemTag := TCustomRVFormattedData[650](Sender.RVData).GetItemTag[230](Sender.ItemNo));
```

Output parameters:

DoDefault – set to *False* to prevent default drawing. It is supported only if TextBackgroundKind[435] <>*rvtbkSimple*.

See also events:

- OnDrawStyleText[448]
- OnApplyStyle[443];
- OnApplyStyleColor[444];

See also properties:

- TextStyles[437].

Demo projects:

- Demos\Delphi\Assorted\Custom Draw\
- Demos\CBuilder\Assorted\Custom Draw\

1.5.2.1.3.9 TRVStyle.OnStyleHoverSensitive

In this event you can define whether TRichView should redraw itself when the mouse pointer moves in or out of the area of text of the **StyleNo**-th style.

```
property OnStyleHoverSensitive: TRVStyleHoverSensitiveEvent;

type
  TRVStyleHoverSensitiveEvent =
    procedure (Sender: TRVStyle;
      StyleNo: Integer;
      var Sensitive: Boolean) of object;
```

(introduced in version 1.3)

This event occurs only for hypertext[479] styles.

Input parameters:

Sender – TRVStyle generating the event.

StyleNo – index of text style in TextStyles[437] collection.

Sensitive – *True* if for the given text style (HoverEffects[471]=[]) or (HoverBackColor[470]<>BackColor[467]) or (HoverColor[470]<>*clNone*) or (HoverUnderlineColor[471]<>*clNone*), or if HoverColor[426]<>*clNone*. *False* otherwise.

Output parameters:

Set **Sensitive** to *True* is you want to draw a special effect for "hot" (under the mouse pointer) text for this text style.

See also events:

- OnDrawStyleText[448];
- OnDrawTextBack[449].

See also properties:

- TextStyles[437].

1.5.2.1.4 Classes of Properties - Style Collections

1.5.2.1.4.1 TFontInfos

Collection of text attributes (styles). Class of TRVStyle.TextStyles[437] property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
TFontInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items[451] – array of TFontInfo[478].

InvalidItem[451] is returned when accessing item which is out of index range.

Methods for Text Style Search

- FindStyleWithFontStyle[455]
- FindStyleWithFontSize[454]
- FindStyleWithColor[453]
- FindStyleWithFontName[454]
- FindSuchStyle, FindSuchStyleEx[455]
- FindStyleWithFont[454]
- FindStyleWithCharset[453]

See Also

- TParaInfos[456] – class of collection of paragraph attributes (styles);
- TRVListInfos[458] – class of collection of list attributes (i.e. styles of bullets and numbering).

This item is returned when accessing Items[451][ItemNo], if (ItemNo<0) or (ItemNo>=TFontInfos.Count)

```
property InvalidItem: TFontInfo[478];
```

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value:

Default value has all properties Items[451][0], but red background[467] and white text color[469].

See also:

- TParaInfos.InvalidItem[456].

Provides indexed access to the items in the collection of text attributes (styles)

```
property Items[Index:Integer]: TFontInfo[478];
```

This is the default property of TFontInfos, so you can write

```
MyRVStyle.TextStyles[437][i]
```

instead of

```
MyRVStyle.TextStyles.Items[i]
```

Many methods of components inherited from TCustomRichView[162] have StyleNo parameter, which is an index in TFontInfos.Items collection (more specifically, in TCustomRichView.Style[193].TextStyles[437].Items).

Example:

```
MyRichView.AddNL[203]('Hello', 0, 0);
```

Unlike normal collections, when accessing item with invalid Index (for example, Index<0 or Index>=Count), no exception occurs. InvalidItem[451] is returned instead.

Indicates the number of pixels that make up a logical inch in the vertical direction.

```
PixelsPerInch: Integer;
```

This property affects how text is displayed in components inherited from TCustomRichView[162] and TRVReportHelper[540] (this property is ignored by TRVPrint[513]).

If this property is equal to 0, Screen.PixelsPerInch is used to display text.

You can assign positive value to this property to make your application independent of screen font mode. For example, if you assign 96, text will always look like in "Small Fonts" mode; if you assign 120, text will always look like in "Large Fonts" mode.

This property does not affect non-text elements, such as line widths or picture sizes (see RichViewPixelsPerInch[703] typed constant).

Default value:

0

Adds a new style in the collection of text styles

```
function Add: TFontInfo[478];
```

This method overrides TCollection.Add method to return TFontInfo instance.

Call Add to create a text style in the collection. The new item is placed at the end of the Items[451] array.

Return value:

Created collection item.

See also:

- TFontInfo.Create[481].

Adds a new item in the collection of text styles, initializes it.

```
function AddFont(Name: TFontName; Size: Integer;
  Color, BackColor: TColor; Style:TFontStyles): TFontInfo[478];
```

Please use Add[452] instead of this method.

Adds a new item in the collection of text styles, initializes it.

```
function AddFontEx(Name: TFontName; Size: Integer;
  Color, BackColor: TColor; Style:TFontStyles;
  Charset: TFontCharset): TFontInfo[478];
```

Please use Add[452] instead of this method.

Copies properties of this collection to the destination object **Dest**.

```
procedure AssignTo(Dest: TPersistent); override;
```

TCollection AssignTo method is overridden to make it possible to assign names of styles (TFontInfo.StyleName[465]) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.TextStyles[437]);
```

A constructor, creates and initializes a collection of text styles.

```
constructor Create(Owner: TPersistent);
```

Do not create collection of styles yourself. It is created and destroyed by TRVStyle component as TextStyles[437] property.

Searches for style with the specified **Charset**.

```
function FindStyleWithCharset(BaseStyle: Integer; Charset: TFontCharset): Integer;
```

(Introduced in version 1.4)

Searches for style having all properties of the **BaseStyle**-th style, but Charset[468]=**Charset**.

BaseStyle is an index in Items[451].

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for style with the specified colors.

```
function FindStyleWithColor(BaseStyle: Integer; Color, BackColor: TColor): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but Color[469]=**Color** and BackColor[467] =**BackColor**.

BaseStyle is an index in Items[451].

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for style having the specified font attributes.

```
function FindStyleWithFont(BaseStyle: Integer; Font: TFont): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, except for all font attributes defined in **Font**.

BaseStyle is an index in Items[451].

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for style having specified font name.

```
function FindStyleWithFontName(BaseStyle: Integer; const FontName: TFontName): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but FontName[470]=**FontName**.

BaseStyle is an index in Items[451].

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for style having the specified font size.

```
function FindStyleWithFontSize(BaseStyle: Integer; Size: Integer): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but font Size[475]=**Size**.

This method does not return items having odd values of SizeDouble[475].

BaseStyle is an index in Items[451].

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find*** ** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for style having the specified font style.

```
function FindStyleWithFontStyle(BaseStyle: Integer; Value, Mask: TFontStyles): Integer;
```

(Introduced in version 1.3)

Searches for style having all properties of the **BaseStyle**-th style, but the font styles defined by **Value** and **Mask**.

Mask is a set of styles for check.

Value is a set of styles what must exist.

BaseStyle is an index in Items[451].

For example:

- **Mask** = [*fsBold*], **Value** = [*fsBold*] searches for bold style (all other font styles are not checked).
- **Mask** = [*fsBold*], **Value** = [] searches for not bold style (all other font styles are not checked).

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found.

See also:

- other TFontInfos.**Find***** methods; the most universal method is FindSuchStyle[455];
- TFontInfo.IsEqual[478].

Searches for a style.

```
function FindSuchStyle(BaseStyle: Integer; Style: TFontInfo[478];
  Mask: TRVFontInfoProperties[679]): Integer;
function FindSuchStyleEx(BaseStyle: Integer; Style: TFontInfo[478];
  Mask: TRVFontInfoProperties[679]): Integer;
```

(Introduced in version 1.3 and version 14)

Searches for a style having properties like **Style**.

The **BaseStyle**-th style is checked first. **BaseStyle** is an index in Items[451].

Mask is a set of properties to compare (other properties are ignored). Use RVAllFontInfoProperties[679] constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

If FindSuchStyle cannot find the desired style, it returns -1.

If FindSuchStyleEx cannot find the desired style, it adds a new style to the end of Items, assigns properties of **Style** to this new item, sets its Standard[464]=*False*, returns the index of this new item.

Return value:

Index of the found style; FindSuchStyle returns -1 if not found.

See also:

- other TFontInfos.**Find***** methods;
- TRVStyle.FindTextStyle[440];
- TFontInfo.IsEqual[478];
- TParaInfos.FindSuchStyle[458];
- TRVListInfos.FindSuchStyle[459].

Demos:

- Demos\Delphi\Editors\Editor 2\

- Demos\CBuilder\Editors\Editor 2\

(see procedure TForm1.rveStyleConversion, rve.OnStyleConversion[400] event)

1.5.2.1.4.2 TParaInfos

Collection of paragraph attributes (styles). Class of TRVStyle.ParaStyles[431] property.

Unit RVStyle;

Syntax

```
TCustomRVInfos = class(TCollection)
TParaInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items[456] – array of TParaInfo[482].

InvalidItem[456] is returned when accessing item which is out of index range.

Methods for Paragraph Style Search

- FindStyleWithAlignment[457];
- FindSuchStyle, FindSuchStyleEx[458].

See Also

- TFontInfos[451] – class of collection of text attributes (styles);
- TRVListInfos[458] – class of collection of list attributes (i.e. styles of bullets and numbering).

This item is returned when accessing Items[456][ItemNo], if (ItemNo<0) or (ItemNo>=TParaInfos.Count)

```
property InvalidItem: TParaInfo[482];
```

(introduced in version 1.7)

This feature is for debugging, this situation should not occur.

Default value

Default value has all properties Items[456][0], but (LeftIndent[485],SpaceBefore[489],RightIndent[488],SpaceAfter[489]) =(1,1,1,1) and red border[485].

See also

- TFontInfos.InvalidItem[451].

Provides indexed access to the items in the collection of paragraph attributes (styles)

```
property Items[Index: Integer]: TParaInfo[482];
```

This is default property of TParaInfos, so you can write

```
MyRVStyle.ParaStyles[431][i]
```

instead of

```
MyRVStyle.ParaStyles.Items[i]
```

Many methods of components inherited from TCustomRichView[162] have ParaNo parameter, which is an index in TParaInfos.Items collection (more specifically, in TCustomRichView.Style[193].ParaStyles[431].Items).

Example:

```
  MyRichView.AddNL('Hello', 0, 0);
```

Unlike normal collections, when accessing item with invalid Index (for example, Index<0 or Index>=Count), no exception occurs. InvalidItem is returned instead.

Adds a new style to the collection of paragraph styles.

```
function Add: TParaInfo;
```

This method overrides TCollection.Add method to return TParaInfo instance.

Call Add to create a paragraph style in the collection. The new item is placed at the end of the Items array.

Return value:

Created collection item.

See also:

- TParaInfo.Create.

Copies properties of this collection to the destination object **Dest**.

```
procedure AssignTo(Dest: TPersistent); override;
```

TCollection AssignTo method is overridden to make it possible to assign names of styles (TParaInfo.StyleName) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.ParaStyles);
```

A constructor, creates and initializes a collection of paragraph styles.

```
constructor Create(Owner: TPersistent);
```

Do not create collection of styles yourself. It is created and destroyed by TRVStyle component as ParaStyles property.

Searches for style having the specified alignment

```
function FindStyleWithAlignment(BaseStyle: Integer;
  Alignment: TRVAlignment): Integer;
```

(Introduced in version 1.5)

Searches for style having all properties of the **BaseStyle**-th style, but Alignment=**Alignment**.

BaseStyle is an index in Items.

The following properties are ignored when comparing styles:

- Standard;
- StyleName (if RichViewCompareStyleNames=*False* (default)).

Return value:

Index of style or -1, if not found.

See also methods:

- FindSuchStyle.

Searches for a style having properties like **Style**.

```
function FindSuchStyle(BaseStyle: Integer; Style: TParaInfo[482];
  Mask: TRVParaInfoProperties[684]): Integer;
function FindSuchStyleEx(BaseStyle: Integer; Style: TParaInfo[482];
  Mask: TRVParaInfoProperties[684]): Integer;
```

(Introduced in version 1.5 and version 14)

The **BaseStyle**-th style is checked first. **BaseStyle** is an index in Items[456].

Mask is a set of properties to compare (other properties are ignored). Use RVAllParaInfoProperties[684] constant to compare all properties.

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

If **FindSuchStyle** cannot find the desired style, it returns -1.

If **FindSuchStyleEx** cannot find the desired style, it adds a new style to the end of Items, assigns properties of **Style** to this new item, sets its Standard[464]=*False*, returns the index of this new item.

Return value:

Index of the found style; **FindSuchStyle** returns -1 if not found.

See also methods:

- FindStyleWithAlignment[457];
- TRVStyle.FindParaStyle[440];
- TParaInfo.IsEqual[490];
- TFontInfos.FindSuchStyle[455];
- TRVListInfos.FindSuchStyle[459].

Demos:

- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

(see procedure TForm1.rveParaStyleConversion, rve.OnParaStyleConversion[397] event)

1.5.2.1.4.3 TRVListInfos

Collection of paragraph list attibutes (bullets and numbering). Class of TRVStyle.ListStyles[429] property.

Unit RVStyle.

Syntax

```
TCustomRVInfos = class(TCollection)
TRVListInfos = class(TCustomRVInfos)
```

Hierarchy

TObject
TPersistent
TCollection
TCustomRVInfos

Main Properties

Items[459] – array of TRVListInfo[493].

Main Methods

- FindSuchStyle[459]
- FindStyleWithLevels[459]

See Also

- TFontInfos 451 – class of collection of text attributes (styles);
- TParaInfos 456 – class of collection of paragraph attributes (styles);
- List markers item type 131.

Provides indexed access to items in the collection of paragraph list attributes (bullets and numbering)

```
property Items[Index:Integer] : TRVListInfo 493;
```

This is default property of TRVListInfos, so you can write

```
MyRVStyle.ListStyles 429[i]
```

instead of

```
MyRVStyle.ListStyles.Items[i]
```

Adds a new style to the collection of paragraph list styles (bullets and numbering)

```
function Add: TRVListInfo 493;
```

This method overrides TCollection.Add method to return TRVListInfo instance.

Call Add to create a new list style in the collection. The new item is placed at the end of the Items 459 array.

Add returns the new collection item.

Copies properties of this collection to the destination object **Dest**.

```
procedure AssignTo(Dest: TPersistent); override;
```

TCollection AssignTo method is overridden to make it possible to assign names of styles (TParaInfo.StyleName 465) to TStrings.

Do not call AssignTo directly, this method is called from **Dest**.Assign method.

Example:

```
MyListBox: TListBox;
...
MyListBox.Items.Assign(MyRVStyle.ListStyles 429);
```

Searches for the style having the specified levels. Adds if not found (optionally)

```
function FindStyleWithLevels(Levels: TRVListLevelCollection;
  const StyleNameForAdding: String; AddIfNotFound: Boolean): Integer;
```

This method searches for the style having levels with properties like in **Levels**.

If **AddIfNotFound**=*True* and style is not found, it adds a new style to the end of collection, assigns **Levels** to its Levels 493 and **StyleNameForAdding** to its StyleName 465. The created style has Standard 464 property = *False*.

Return value:

Index of style, or -1 if not found (and not added).

See also:

- FindSuchStyle 459.

Searches for the style. Adds if not found (optionally)

```
function FindSuchStyle(Style: TRVListInfo 493; AddIfNotFound: Boolean): Integer;
```

This method searches for the style having all properties of **Style**.

If **AddIfNotFound**=*True* and style is not found, it adds a new style to the end of collection and assigns all properties of **Style** to it. The created style has Standard[464] property = *False*.

The following properties are ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

Return value:

Index of style, or -1 if not found (and not added).

See also:

- FindStyleWithLevels[459];
- TFontInfos.FindSuchStyle[455];
- TParaInfos.FindSuchStyle[458].

1.5.2.1.4.4 TRVStyleTemplateCollection

Collection of "real" styles of text and paragraphs. Class of TRVStyle.StyleTemplates[435] property.

Unit RVStyle.

Syntax

```
TRVStyleTemplateCollection = class(TCollection)
```

(introduced in version 14)

Hierarchy

TObject
TPersistent
TCollection

Main Properties

- Items[461] – array of TRVStyleTemplate[494].
- DefStyleName[460] – a format string used to generate names of new items.

Main Methods

- AssignStyleTemplates[461] assigns another collection to this collection;
- FindById, FindItemById[462] search for items by their identifiers;
- FindByName, FindItemByName[462] search for items by their names;
- LoadFromRVST, SaveToRVST[462] save and load the collection to a file;
- ResetNameCounter[463] resets a counter used to generate names for new items.

See also

- Styles and style templates[74].

Specifies the name for new style templates.

```
property DefStyleName: TRVUnicodeString[694];
```

(changed in version 18)

This string must contain '%d' as a substring. It will be substituted by a number to generate an unique name. This name will be assigned to Name[496] of new items in this collection.

Default value:

'Style %d'

See also:

- ResetNameCounter[463]

Returns a list of identifiers that cannot be assigned to new items.

```
property ForbiddenIds: TRVIntegerList;
```

This property is useful to implement editing styles (in a user interface) before calling TRichViewEdit.ChangeStyleTemplates[338]. When the user deletes a style template, add its Id[495] to this list. Otherwise, if a new style template accidentally receives the same Id as a deleted style template, text and paragraphs linked to the deleted style template will be linked to the new style template.

Provides indexed access to items in the collection of style templates.

```
property Items[Index:Integer]: TRVStyleTemplate[494];
```

This is default property of TRVStyleTemplateCollection, so you can write

```
MyRVStyle.StyleTemplates[435][i]
```

instead of

```
MyRVStyle.StyleTemplates.Items[i]
```

Returns the item having Name[496]='Normal'.

```
property NormalStyleTemplate: TRVStyleTemplate[494];
```

If such an item does not exist in Items[461], the property returns *nil*.

Copies items from **Source** to this collection.

```
procedure AssignStyleTemplates(Source: TRVStyleTemplateCollection; CopyIds: Boolean);
```

If **CopyIds**=*True*, Id[495] properties of **Source**.Items[461] are copied to new items.

If **CopyIds**=*False*, the method generates new Id for new items, retaining NextId[496] and ParentId[497] relations.

Note:

StyleTemplates1.Assign(StyleTemplates2) is equivalent to StyleTemplates1.AssignStyleTemplates(StyleTemplates2, False).

See also:

- TCustomRichViewEdit.ChangeStyleTemplates[338].

Assign style templates to a string list.

```
type
  TRVStyleTemplateKinds = set of TRVStyleTemplateKind[495];
procedure AssignToStrings(Strings: TStrings; AssignIds: Boolean;
  Kinds: TRVStyleTemplateKinds; ExcludeThisStyle: TRVStyleTemplate[494];
  OnlyPotentialParents: Boolean);
```

Parameters:

Strings – a destination string list. Names[496] of style templates are added as items to this string list.

If **AssignIds**=*True*, the method saves Ids[495] of style templates as Objects of **Strings** (type-casted to TObject).

Only style templates having Kinds[495] belonging to **Kinds** are added.

If **ExcludeThisStyle**<>*nil*, it is not added to **Strings**.

If **OnlyPotentialParents**=*True*, and **ExcludeThisStyle**<>*nil*, only style templates that can be parents of **ExcludeThisStyle** are added to **Strings** (direct or indirect children of **ExcludeThisStyle** are not added).

Note: this method may be useful only if you display Names[496] of all style templates to users (for example, you may implement localization of names of standard styles; in this case, you need to implement a similar function yourself).

The methods reset formatting of the specified text/paragraph attributes (styles).

```
procedure ClearTextFormat(ATextStyle: TCustomRVFontInfo[466]);
procedure ClearParaFormat(AParaStyle: TCustomRVParaInfo[482]);
```

ClearTextFormat clears formatting from **ATextStyle**. The method resets all its properties to default values, then:

- if **ATextStyle** is TFontInfo[478], and **ATextStyle**.Jump[479]=True, and "Hyperlink" style template exists in Items[461], the method applies it to **ATextStyle**;
- otherwise, the method assigns **ATextStyle**.StyleTemplateId = -1.

ClearParaFormat clears formatting from **AParaStyle**. The method resets all its properties to default values, then:

- if "Normal" style template exists in Items[461], the method applies it to **AParaStyle**;
- otherwise, the method assigns **AParaStyle**.StyleTemplateId = -1.

Searches for the item having the specified identifier[495] **Id**.

```
function FindById(Id: TRVStyleTemplateId): Integer;
function FindItemById(Id: TRVStyleTemplateId): TRVStyleTemplate;
```

FindById returns the index of the found item in Items[461], or -1 if not found.

FindItemById returns the the found item, or *nil* if not found.

See also:

- FindByName, FindItemByName[462]

Searches for the item having the specified name[496] **Name**

```
function FindByName(const Name: TRVStyleTemplateName[691]): Integer;
function FindItemByName(const Name: TRVStyleTemplateName[691]): TRVStyleTemplate;
```

FindByName returns the index of the found item in Items[461], or -1 if not found.

FindItemByName returns the the found item, or nil if not found.

See also:

- FindById, FindItemById[462]

The methods load and save style templates to files.

```
function LoadFromRVST(const FileName: TRVUnicodeString[694]; Units: TRVStyleUnits[691]): Boolean
function SaveToRVST(const FileName: TRVUnicodeString[694]; Units: TRVStyleUnits[691]): Boolean;
```

(changed in version 18)

Files of "RichView Styles" format contain style templates. Normally they have "rvst" extension. Files of these formats can be imported and exported in the RichViewActions' style management dialog window (displayed by TrvActionStyleTemplates action).

When saving, specify **Units** used by the style templates:

- for RVStyle.StyleTemplates[435], this is RVStyle.Units[437];
- for rvActionNew.StyleTemplates or rvActionStyleTemplates.StandardStyleTemplates, this is RVAControlPanel.UnitsProgram.

When loading, **Units** are specified similarly. **LoadFromRVST** converts the loaded style templates to the specified **Units**.

Note: after loading to RVStyle.StyleTemplates[435], clear RVStyle.TextStyles[437] and ParaStyles[431] (because the old items may contain references[465] to previous style templates). Then you can add default text and paragraph styles (see the example[498]).

Example:

Demos*\Editors\StyleTemplates demo loads "default.rvst" in "File | New" command.

See also

- InsertFromRVST[463]

Applies styles from the file to the collection.

```
function InsertFromRVST(const FileName: TRVUnicodeString[694]; Units: TRVStyleUnits[691]): Boo
```

Files of "RichView Styles" format contain style templates. Normally they have "rvst" extension. They can be created by SaveToRVST[462] method.

This method loads style templates from the file. If style templates of the same names already exist, properties of existing style templates are overridden from the file. Otherwise, new style templates are added to the collection. Units of loaded style templates are converted to **Units**.

Example: applying styles from a file to RichViewEdit1

```
var
  StyleTemplates: TRVStyleTemplateCollection[460];
begin
  if not OpenDialog1.Execute then
    exit;
  StyleTemplates := TRVStyleTemplateCollection[460].Create(nil);
  try
    StyleTemplates.AssignStyleTemplates[461](rvs.StyleTemplates, True);
    if not StyleTemplates.InsertFromRVST(OpenDialog1.FileName, RichViewEdit1.Style[193].Uni
      exit;
    RichViewEdit1.ChangeStyleTemplates[338](StyleTemplates);
  finally
    StyleTemplates.Free;
  end;
end;
```

Resets the counter used to generate unique names for new Items[461].

```
procedure ResetNameCounter;
```

Names are generated using DefStyleName[460] property. By default, new items receive names: 'Style 1', 'Style 2', ..., with the name counter increased: 1, 2...

This method assigns 1 to the name counter, so the next generated name will include a minimal number providing its uniqueness.

1.5.2.1.5 Classes of Properties - Styles

1.5.2.1.5.1 TCustomRVInfo, TCustomRVTextOrParaInfo

These are ancestor classes for all TRichView attributes (styles): text attributes, paragraph attributes, list attributes.

Unit RVStyle.

Syntax

```
TCustomRVInfo = class(TCollectionItem)
TCustomRVFontOrParaInfo = class(TCustomRVInfo)
```

Hierarchy

TObject
TPersistent
TCollectionItem

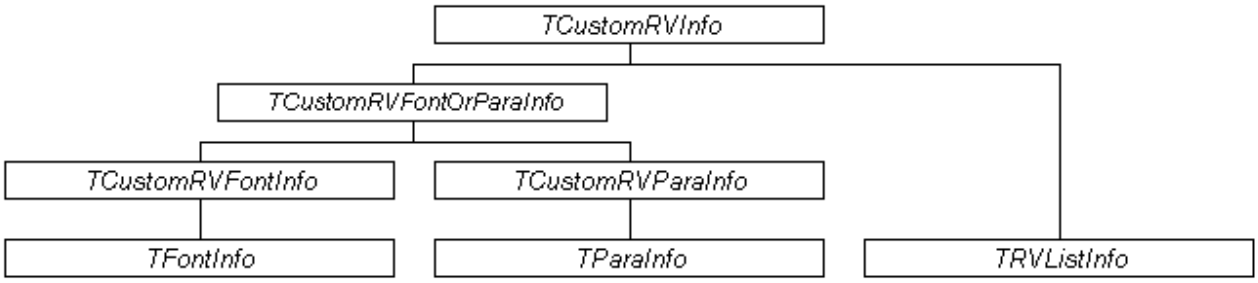
Using

These classes are not used directly. Classes inherited from them are used instead.

The following classes are inherited from TCustomRVInfo (all defined in RVStyle unit).

Class	Meaning
TCustomRVFontInfo 466	This class introduces the most of properties related to text attributes. Ancestor of TFontInfo.
TCustomRVParaInfo 482	This class introduces the most of properties related to paragraph attributes. Ancestor of TParaInfo.
TFontInfo 478	Text style, defines text attributes. Class of items in the collection TRVStyle 418.TextStyles 437.
TParaInfo 490	Paragraph style, defines paragraph attributes. Class of items in the collection TRVStyle 418.ParaStyles 431.
TRVListInfo 493	List style (i.e. style of paragraphs' bullets & numbering). It has Levels 493 collection defining attributes of paragraph list markers 131. Class of items in the collection TRVStyle 418.ListStyles 429.

The class hierarchy is explained on the diagram below.



Properties

TCustomRVInfo introduces the following properties:

- Standard 464;
- StyleName 465.

TCustomRVFontOrParaInfo introduces the following properties:

- StyleTemplateId 465.

Identifies standard (essential) styles

```
property Standard: Boolean;
```

(introduced in version 1.6)

Description

Standard styles are not deleted by DeleteUnusedStyles 216.

This property is ignored when comparing and searching styles.

All styles which are added automatically while loading from RTF or inserting from RVF are not standard.

All styles added by TRVStyle.FindTextStyle 440, TRVStyle.FindParaStyle 440, TFontInfos.FindSuchStyleEx 455, TParaInfos.FindSuchStyleEx 458 are not standard.

Styles which are added automatically while loading from RVF preserve value of this property.

Usage without StyleTemplates[435]

If StyleTemplates are not used[195], standard styles may represent "real" styles, while nonstandard styles may represent additional text and paragraph attributes. Initially, you can fill the collection with standard styles, which will never be removed (unless replaced when loading an RVF file). When working with a fixed set of styles[168], it's recommended to make all styles standard. When working with a dynamic set of styles, it is usually enough to have only one standard text and paragraph style initially (all other styles will be added on editing).

Usage with StyleTemplates[435]

If StyleTemplates are used, it's recommended to have only one standard paragraph style initially, linked[465] to NormalStyleTemplate[461], and only one standard text style not linked to any style template (but its ParaStyleTemplateId[473] should point to NormalStyleTemplate[461] as well). All other styles will be added on editing.

Default value:

True

See also:

- TFontInfo[478] – class of text attributes (style);
- TParaInfo[482] – class of paragraph attributes (style);
- TRVListInfo[493] – class of paragraph list attributes (style);
- RichViewResetStandardFlag[703] typed constant.

Name of this style.

```
property StyleName: TRVUnicodeString[694];
```

(changed in version 18)

This name is used:

- when displaying a design-time editor for collections of styles;
- when assigning text styles to TStrings[453].
- when saving RTF without using StyleTemplates[195] and with *rvrtfSaveStyleSheet*[185] option;

This property is ignored when comparing and searching styles (see also RichViewCompareStyleNames[701] typed constant).

Names of styles do not need to be unique in the collection (unless you save RVF with *rvfoUseStyleNames*[188] option).

This property makes sense if you use a predefined set of styles[168]. If you add styles dynamically, you can ignore this property.

Links this style to a style template.

```
property StyleTemplateId: TRVStyleTemplateId[690];
```

(introduced in version 14)

This property is used only if style templates are used[195].

TCustomRVFontInfo[466] may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkText*.

TCustomRVParaInfo[482] may be linked to style templates having Kind equal to *rvstkParaText* or *rvstkPara*.

See also:

- TCustomRVFontInfo.ParaStyleTemplateId[473]

1.5.2.1.5.2 TCustomRVFontInfo

This is an ancestor class for TFontInfo[478] (class of item in collection of text styles[437]).

TCustomRVFontInfo introduces the most of properties of text.

Unit RVStyle.

Syntax

```
TCustomRVFontInfo = class(TCustomRVFontOrParaInfo[463])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVFontOrParaInfo[463]
TCustomRVInfo[463]

Properties

Font attributes (similar to properties of TFont):

- FontName[470] – font name;
- Size[475] and SizeDouble[475] – font size;
- Color[469] – text color;
- Style[475] – text styles (bold, italic, underline, strikeout);
- Charset[468] – font character set (for non-Unicode text styles).

Additional properties affecting text appearance and layout:

- BackColor[467] – text background color;
- UnderlineColor[476] – underline color;
- UnderlineType[476] – underline style (normal, double, dotted, etc.);
- StyleEx[475] – additional text styles (overline, all-caps);
- SubSuperScriptType[476] – normal text/subscript/superscript;
- VShift[477] moves text up or down from its base line;
- CharScale[467] – horizontal character scaling;
- CharSpacing[468] – spacing between characters;
- BiDiMode[467] – default text direction.
- EmptyWidth[469] – width for empty text item

Hypertext properties:

- HoverColor[470] – text highlight color;
- HoverBackColor[470] – text background highlight color;
- HoverUnderlineColor[471] – underline highlight color;
- HoverEffects[471] – additional highlight effects (underline);
- JumpCursor[472] – mouse cursor for this text.

Properties related to editing in TCustomRichViewEdit[316]:

- Protection[473] – text protection options;
- EmptyWidth[469] – width for empty text item, affects editing.

Properties related to text export:

- Options[472] allows to create RTF and HTML codes.

Methods

IsEqual[478] compares this text style with another one.

You can use Assign[477] method to assign:

- TCustomRVFontInfo to TCustomRVFontInfo;

- TFont to TCustomRVFontInfo;
- TCustomRVFontInfo to TFont.

See Also

Classes:

- TFontInfo[478] – class inherited from this class, text attributes (style);
- TFontInfos[451] – collection of text attributes (styles) (TFontInfo[478]).

Background color for text of this style

```
property BackColor: TColor;
```

Set BackColor=*clNone* for transparent text background.

Default value:

clNone

See also:

- HoverBackColor[470];
- Color[469];
- HoverColor[470].

See also properties of TRVStyle:

- TextBackgroundKind[435].

Specifies default direction for the text of this style.

```
property BiDiMode: TRVBiDiMode[668];
```

(introduced in version 1.6)

Value	Meaning
rvbdUnspecified	Text direction is inherited from higher level, i.e. from paragraph (see BiDiMode[484] of paragraph style)
rvbdLeftToRight	Default text reading order is left-to-right
rvbdRightToLeft	Default text reading order is right-to-left

Default value:

rvbdUnspecified

See also:

- Bidirectional text in TRichView[100].

Horizontal character scaling value for text of this style, %

```
property CharScale: Integer;
```

(Introduced in version 1.5)

By default, value of this property is equal to 100%, and text of this style is displayed with normal proportions. Set it to larger value to make characters wider, and to lesser value to make them narrower.

This value does not affect the height of text.

Example:

This example shows text formatted with CharScale = 100, 150, and 80.



CharScale Example

Warning: the standard Delphi TFont class does not support character scaling. So be careful if you want to use this value in combination with custom drawing[448]. Any changes of RichView's Canvas.TFont will reset font scaling to default.

Default value:

100

Specifies character set for text of this style.

```
property Charset : TFontCharset;
```

Similar to TFont.Charset.

Since all text in TRichView is stored as Unicode, this Charset is not used when drawing text (but still used inside some methods performing conversion ANSI↔Unicode, for determining the code page (language) of text of this style). However, it's highly recommended to assign SYMBOL_CHATSET when using symbol fonts (such as "Symbol", "Wingdings", "Webdings").

Default value:

DEFAULT_CHARSET

Specifies intercharacter spacing for text of this style.

```
property CharSpacing: TRVStyleLength[690];
```

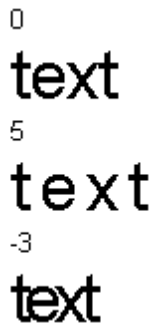
(introduced in version 1.6)

This property defines expansion (if positive) or compression (if negative) of space between characters.

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this text style.

Example:

This example shows text formatted with CharSpacing = 0, 5, and -3 pixels.



CharSpacing Example

Note: if TextEngine[436] is Windows API, all expanded characters are aligned to the left side (space is added to the right side of the characters).

if TextEngine[436] is Uniscribe:

- characters are aligned to the leading side (left side for left-to-right text, right side for right-to-left text)
- kashidas are used for expanded Arabic text
- some characters may be grouped in clusters, a total spacing will be applied to the whole cluster; however, the caret positions are calculated as if there are no clusters.

Default value:

0

Specifies color for text of this style.

```
property Color: TColor;
```

Similar to TFont.Color.

Default value:

clWindowText

See also:

- BackColor[467];
- HoverColor[470];
- HoverBackColor[470].

Specifies the width of a text item if its text is empty.

```
property EmptyWidth: TRVStyleLength[690];
```

(introduced in version 17)

If a positive value is assigned to this property, text items formatted with this style have special behavior. Editing operations (**Delete** , **Backspace** keys, deleting selection) can delete all text of this item without deleting this item: it becomes empty. Empty items have the specified width. Unless protected, this empty item can be deleted using **Delete** or **Backspace** keys.

Such text items can be used to implement text fields in document. It's recommended to define background color[467] for them.

For all normal text items, selection[81]/caret position after the item equals to length of text + 1. But for items of styles having **EmptyWidth** > 0, the position after empty text item is 2.

Recommendations:

- you can assign a small value to this property, less than width of space character; or
- you can assign a large value to this property, to attract attention to unfilled fields.

Default value:

0

See also:

- Protection[473].

Specifies font typeface for text of this style.

```
type TFontName = type string;
property FontName: TFontName
```

Similar to TFont.Name.

Default value:

'Arial'

Specifies background highlight color for hypertext links of this style.

```
property HoverColor: TColor;
```

(introduced in version 1.2)

If this style is a hypertext style (Jump[479]=*True*), background of text of this style can be highlighted with this color when user moves the mouse pointer above it. Set HoverBackColor=BackColor[467] to disable effect.

By default, RichViewEdit[316] also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[325] property.

This property has no effect for non-hypertext styles.

Default value:

clNone

See also properties:

- BackColor[467];
- HoverColor[470];
- Jump[479];
- JumpCursor[472];
- HoverEffects[471];
- HoverUnderlineColor[471].

See also properties of TRVStyle:

- HoverColor[426];
- FullRedraw[425];
- TextBackgroundKind[435].

Specifies highlight color for hypertext links of this style.

```
property HoverColor: TColor;
```

If this style is a hypertext style (Jump[479]=*True*), text of this style can be highlighted with this color when user moves the mouse pointer above it.

If this property is equal to *clNone*, TRVStyle.HoverColor[426] is used. If it is, in its order, equal to *clNone*, no hover color effect is used.

By default, RichViewEdit[316] also highlights hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[325] property.

This property has no effect for non-hypertext styles.

Default value:

clNone

See also properties:

- Color[469];
- HoverBackColor[470];
- Jump[479];
- JumpCursor[472];
- HoverEffects[471];
- HoverUnderlineColor[471].

See also properties of TRVStyle:

- HoverColor[426];
- FullRedraw[425].

Specifies additional effects for hyperlinks.

```
type
  TRVHoverEffect = (rvheUnderline);
  TRVHoverEffects = set of TRVHoverEffect;
property HoverEffects: TRVHoverEffects;
```

(introduced in version 10)

If this style is a hypertext style (Jump[479]=*True*), these effects are applied to text of this style when user moves the mouse pointer above it.

By default, RichViewEdit[316] also applies these effects to hyperlinks when the caret is moved inside them. This effect can be disabled by including *rvoNoCaretHighlightJumps* in RichViewEdit.EditorOptions[325] property.

This property has no effect for non-hypertext styles.

Effect	Meaning
rvheUnderline	Text is underlined.

This underline may have a custom color, see HoverUnderlineColor[471] property.

Default value:

[]

See also properties:

- Jump[479];
- HoverColor[470];
- HoverBackColor[470];
- JumpCursor[472].

See also properties of TRVStyle:

- FullRedraw[425].

Specifies underline highlight color for hypertext links of this style.

```
property UnderlineColor: TColor;
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[475], or *rvheUnderline* is included HoverEffects[471].

If this style is a hypertext style (Jump[479]=*True*), underline of text of this style can be highlighted with this color when user moves the mouse pointer above it.

If this property is equal to *clNone*, UnderlineColor[476] is used. If it is, in its order, equal to *clNone*, HoverColor[470] is used. If it is, in its order, equal to *clNone*, Color[469] is used.

RTF Note:

This attribute cannot be stored in RTF files.

HTML Note:

Colored underlines are not supported in HTML and CSS.

SaveHTML[250] ignores this attribute. SaveHTMLEx[251] saves colored underline as a bottom border side of the text's element.

Default value:

clNone

See also:

- UnderlineType[476].

Specifies mouse cursor for hypertext of this style.

```
property JumpCursor: TCursor;
```

This cursor appears when user moves the mouse pointer to text item of this style in RichView. This property has effect only for hypertext items (Jump[479]=*True*).

Default value:

crJump (101) (system hypertext cursor or )

See also properties of TRVStyle:

- JumpCursor[427].

See also:

- Cursors used by components of RichView family[108].

Options for exporting text of this style to RTF and HTML.

```
type
  TRVTextOption = (rvteoHTMLCode, rvteoRTFCode, rvteoDocXCode, rvteoDocXInRunCode, rvteoHid
  TRVTextOptions = set of TRVTextOption;

property NextStyleNo: Integer;
```

(introduced in version 1.6)

Option	Meaning
rvteoRTFCode	If set, '{', '}', '\' are saved in RTF as they are. **Example:** set this option, write text of this style: '\animtext1 Viva Las Vegas', copy to RTF and paste in MS Word.
rvteoHTMLCode	If set, '&', '<', '>', and multiple spaces are saved in HTML as they are. **Example:** set this option, write text of this style: '<marquee width=200>Attention. Spaceship is arriving at the Second Terminal.</marquee>', save as HTML and open in web browser.
rvteoDocXCode	If set, text of this style is saved to DocX as it is (converted to UTF-8), in place of <w:r>. No character properties are saved.
rvteoDocXInRunCode	If set, text of this style is saved to DocX as it is (converted to UTF-8) in place of <w:t> (inside <w:r>).
rvteoHidden	If set, this text is hidden. Hidden text is displayed only if *rvoShowHiddenText* is included in RichView.Options[182]

All "code" options are mutually exclusive. If two or more different "code" options are included, text of this style is not saved in any of RTF, HTML, DocX.

If the both *rvteoDocXCode* and *rvteoDocXInRunCode* are specified, *rvteoDocXCode* is used.

Default value:

[]

See also:

- TCustomRichView.SaveRTF[253];
- TCustomRichView.SaveDocX[249];
- About HTML export[96].

Links this text style to the style template of paragraph.

```
property ParaStyleTemplateId: TRVStyleTemplateId[690];
```

(introduced in version 14)

This property is used only if style templates are used[195].

This property is redundant: the style template of paragraph is linked using TParaInfo[490].StyleTemplateId[465]. However, it is necessary to distinguish text styles having all identical properties but used for text in paragraphs linked to different style templates.

TRichView provides for all text items that their TextStyle.ParaStyleTemplateId=ParaStyle.StyleTemplateId. If not, it automatically links the item to another text style with the correct properties (adding it to the end of TextStyles[437], if necessary).

Protects text of this style from specific operations in editor.

```
property Protection: TRVProtectOptions;

type
  TRVProtectOption =
    (rvprStyleProtect, rvprStyleSplitProtect,
     rvprModifyProtect,
     rvprDeleteProtect, rvprConcateProtect,
     rvprRVFInsertProtect, rvprDoNotAutoSwitch,
     rvprParaStartProtect,
     rvprSticking, rvprSticking2, rvprSticking3,
     rvprStickToTop, rvprStickToBottom);
  TRVProtectOptions = set of TRVProtectOption;
```

Used in RichViewEdit[316]. Text of this style has special behavior when editing.

General Protection Options

Option	Meaning
rvprStyleProtect	Protects text from changing style with ApplyTextStyle[334], ApplyTextStyleTemplate[334], ApplyStyleTemplate[333]. This protection is ignored by ApplyStyleConversion[333], but you can take it into account manually in OnStyleConversion[400] event.
rvprStyleSplitProtect	Protects from applying text style to the part of text item. ApplyTextStyle[334] and ApplyStyleConversion[333] methods apply new style to the whole item having this protection, even if it is selected only partially.
rvprModifyProtect	Protects text items of this style from modification, but does not protect from deleting as a whole.

rvprDeleteProtect	Protects text items of this style from deletion as a whole, but does not protect from modification. Normally, this option blocks all operations deleting the last character of the protected text item. But if EmptyWidth[469] > 0, all characters can be deleted, but the item is not deleted. See also: *rvepDeleteProtect*[673] item property[673].
rvprConcateProtect	Protects texts items from concatenation with adjacent item of the same style.
rvprRVFInsertProtect	If set, items of this style will not be inserted when inserting RVF. Affects methods introduced in editor only (**InsertRV***Ed**, pasting).
rvprDoNotAutoSwitch	If set, editor never switches the current text style to this one automatically. If this style was made current by the user (by assigning CurTextStyleNo[324] property), the editor switches back to another (previously used) style after the first performed operation.
rvprParaStartProtect	Disallows moving text item of this style to/from new line; blocks Enter key when the caret is in text of this style.

Sticking

Option	Meaning
rvprSticking	Disallows insertion between two text items having this protection (allows to stick together several text items of different text styles).
rvprSticking2	The same.
rvprSticking3	The same.
rvprStickToTop	Disallows insertion before the text of this style, if this text is a first document item.
rvprStickToBottom	Disallows insertion after the text of this style, if this text is a last document item.

rvprSticking, *rvprSticking2*, and *rvprSticking3* allow to create several groups of stuck items. It's not possible to insert something between items having the same protection option, but it's possible to insert between items having different options, for example *rvprSticking* and *rvprSticking2*. In addition to these options, you can use OnCheckStickingItems[391] event.

Note:

TFontInfo.SingleSymbols property is removed in version 1.3. When loading forms saved with an older version, or when loading styles from ini-files or registry, SingleSymbols=*True* is converted to Protection=[*rvprStyleProtect*, *rvprDoNotAutoSwitch*].

Default value:

[]

See also:

- Options[487] of paragraph style;
- EmptyWidth[469] property

Demo projects:

- Demos\Delphi\Assorted\Fields\
- Demos\CBuilder\Assorted\Fields\

The properties specify the font height.

```
property Size: Integer;
property SizeDouble: Integer;
```

(SizeDouble is introduced in version 15)

Size specifies the font size in points [106], similarly to TFont.Size.

SizeDouble specifies the font size in half-points, i.e. its value is twice greater than **Size**.

Assigning to one of these properties changes value another one. **SizeDouble** allows to specify a font size with higher precision.

Default values:

- Size: 10
- SizeDouble: 20

Specifies whether the text of this style is normal, italic, underlined, bold, and so on.

```
type  TFontStyle = (fsBold, fsItalic, fsUnderline, fsStrikeOut);
type  TFontStyles = set of TFontStyle;
property Style: TFontStyles;
```

Similar to TFont.Style.

Default value:

[]

See also properties:

- StyleEx [475];
- HoverEffects [471].

Additional attributes for font of text of given style in RichView or its descendants.

```
type TRVFontStyle = (rvfsOverline, rvfsAllCaps);
type TRVFontStyles = set of TRVFontStyle
property StyleEx: TRVFontStyles;
```

Option	Meaning
rvfsOverline	Vertical line above the text
rvfsAllCaps	All capitals

Note:

All-Caps is introduced in version 1.5. It has the following limitations:

- for Unicode text, capitalization works only under WinNT/2000/XP/Vista;
- for ANSI text capitalization, TRichView always uses the system default language/code page (i.e. the result of AnsiUpperCase).

Default value:

[]

Specifies whether this style is a normal text, a subscript, or a superscript.

```
type
  TRVSubSuperScriptType = (rvsssNormal,  rvsssSubscript,  rvsssSuperScript);
property SubSuperScriptType: TRVSubSuperScriptType;
```

(introduced in version 10)

Subscripts and superscripts are displayed with smaller font than specified in Size[475].

Default value:

rvsssNormal

See also properties:

- VShift[477].

Specifies underline color.

```
property UnderlineColor: TColor;
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[475] (or *rvheUnderline* is included HoverEffects[471], under the mouse pointer).

If UnderlineColor=*clNone*, text color is used for underline (Color[469] in normal state, HoverColor[470] for hyperlinks under the mouse pointer).

Values other than *clNone* define underline color. HoverUnderlineColor[471] property may change hyperlink color under the mouse pointer.

RTF Note:

This attribute can be saved and loaded in RTF files.

HTML Note:

Colored underlines are not supported in HTML and CSS.

SaveHTML[250] ignores this attribute. SaveHTMLEx[251] saves colored underline as a bottom border side of the text's element.

Default value:

clNone

See also:

- UnderlineType[476].

Specifies the style of underline.

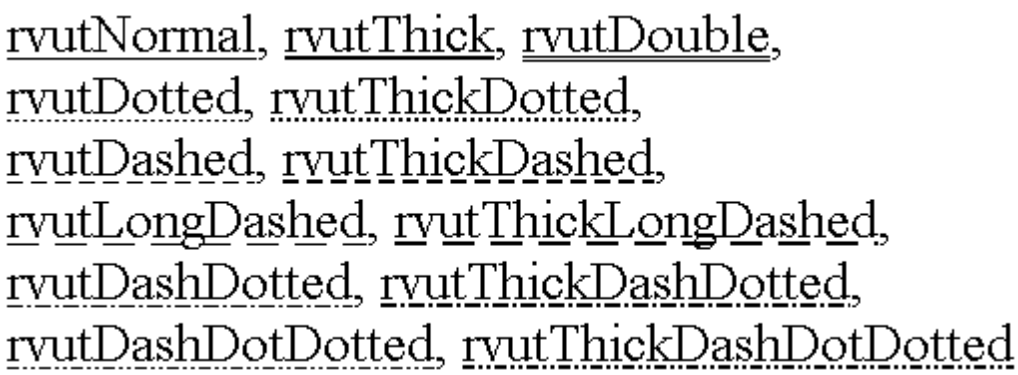
```
type
  TRVUnderlineType = (rvutNormal, rvutThick, rvutDouble,
    rvutDotted, rvutThickDotted,
    rvutDashed, rvutThickDashed,
    rvutLongDashed, rvutThickLongDashed,
    rvutDashDotted, rvutThickDashDotted,
    rvutDashDotDotted, rvutThickDashDotDotted);
property UnderlineType: TRVUnderlineType;
```

(introduced in version 10)

Underline is displayed only if *fsUnderline* is included in Style[475] (or *rvheUnderline* is included HoverEffects[471], under the mouse pointer).

By default, underline color is defined in Color[469] property, but it can be altered by UnderlineColor[476] property (see also HoverColor[470]/HoverUnderlineColor[471] for hyperlink colors under the mouse pointer).

Sample:



UnderlineType Example

RTF Note:

This attribute can be saved and loaded in RTF files.

HTML Note:

Custom underlines are not supported in HTML and CSS.

SaveHTML[250] ignores this attribute. SaveHTMLEx[251] saves custom underline as a bottom border side of the text's element (single, double, dotted and dashed lines are supported).

Default value:

rvutNormal

Vertical offset of text item of this style, in percents of text height.

```
property VShift: Integer;
```

Vertical offset is counted relative to the base line of text. Normal text items have VShift=0.

Use positive values to move text up, negative values to move text down.

This property can be applied not only to normal text, but to subscripts and superscripts[476] (the effect is cumulative).

Default value:

0

See also:

- TRVExtraItemProperty[673] type (*rvepVShift*)

Copies the contents of **Source** to this text style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TFont, TCustomRVInfo[463], or TCustomRVFontInfo to this text style.

See also:

- TFontInfo.Assign[480] (TFontInfo[478] is inherited from TCustomRVFontInfo).

Copies the properties of this text style to the destination object **Dest**.

```
procedure AssignTo(Dest: TPersistent);override;
```

Do not call AssignTo directly, this method is called from Assign method of a destination object.

AssignTo allows to assign text style to TFont.

Example:

```
MyEdit.Font.Assign(MyRVStyle.TextStyles[437][0]);
```

A constructor, creates and initializes a new instance of TCustomRVFontInfo.

```
constructor Create(Collection: TCollection);override;
```

The constructor initializes:

```
// Properties inherited from TCustomRVInfo[463]:
Standard[464]     := True;
StyleName[465]    := 'Font Style';
// Properties defined in TCustomRVFontInfo:
Size[475]         := 10;
FontName[470]     := 'Arial';
Color[469]        := clWindowText;
UnderlineColor[476] := clNone;
BackColor[467]    := clNone;
HoverColor[470]   := clNone;
HoverUnderlineColor[471] := clNone;
Charset[468]      := DEFAULT_CHARSET;
JumpCursor[472]   := crDefault;
CharScale[467]    := 100;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a text style in code, you should assign Standard[464] := *False*.

Compares this text style with **Value**.

```
function IsEqual(Value: TCustomRVFontInfo; IgnoreList: TRVFontInfoProperties[679]): Boolean; d
```

(introduced in version 1.3)

Parameters:

Value – object to compare with this text style. The class of this object is TCustomRVFontInfo or inherited from it.

IgnoreList – set of values identifying properties to ignore when comparing, see TRVFontInfoProperties[679].

The following properties are always ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

See also:

- TFontInfo.IsEqual[482] (TFontInfo[478] is inherited from this class);
- TCustomRVParaInfo.IsEqual[490].

1.5.2.1.5.3 TFontInfo

This class defines attributes of text items[120] in TRichView[162], TRichViewEdit[316], TDBRichView[407], TDBRichViewEdit[410] controls.

This is an item in the collection of TRVStyle[418].TextStyles[437]. The class of the collection itself is TFontInfos[451].

Unit RVStyle.

Syntax

```
TFontInfo = class(TCustomRVFontInfo[466])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[463]
TCustomRVFontOrParaInfo[463]
TCustomRVFontInfo[466]

Properties

The most of properties are inherited from TCustomRVFontInfo[466].

The new properties are:

- Jump[479] – "is this a hyperlink?"
- NextStyleNo[480] – index of text style to apply after pressing Enter in editor (if style templates are not used[195]);
- ModifiedProperties[480] may (temporarily) contain a list of properties having values different from values defined in a style templates (of this text style[465] and paragraph[473]).

Methods

IsEqual[482] compares this text style with another one.

You can use Assign[480] method to assign:

- TFontInfo to TFontInfo;
- TCustomRVFontInfo[466] to TFontInfo and vice versa;
- TFont to TFontInfo and vice versa.

See Also

- TFontInfos[451] – class of collection of text styles;
- TParaInfo[482] – class of paragraph style;
- TRVListInfo[493] – class of list style.

Specifies whether this style is a hypertext style.

```
property Jump: Boolean
```

If Jump=*True*, text items of this style are hyperlinks.

The following properties are applied only to hypertext styles:

- HoverColor[470];
- HoverBackColor[470];
- HoverUnderlineColor[471];
- HoverEffects[471];
- JumpCursor[472].

See also:

- RichView hypertext overview[69].

Demo projects:

- Demos\Delphi\Assorted\Hypertext\
- Demos\CBuilder\Assorted\Hypertext\

This property may contain a list of properties not defined in style templates[435] of this style.

```
property ModifiedProperties: TRVFontInfoProperties[679];
```

(introduced in version 14)

This property is used only if style templates are used[195].

This is a redundant property: the difference in properties is determined not by this property, but by the actual difference in values of properties of this style and its style template(s).

This property is ignored when searching for or comparing text styles.

Some methods may change value of this property, so do not use this property to store any information. Value of this property should be calculated just before using.

See also properties:

- StyleTemplateId[465];
- ParaStyleTemplateId[473].

See also properties of TParaInfo:

- ModifiedProperties[491].

See also methods of TRVStyleTemplate:

- UpdateModifiedTextStyleProperties[500];
- ApplyToTextStyle[499] (contains an example).

Defines the text style for following paragraphs.

```
property NextStyleNo: Integer;
```

This property may contain either an index in Style[193].TextStyles[437], or the value -1.

This property is used in RichViewEdit[316], only if UseStyleTemplates[195]=*False*. When a user presses Enter key at the end of paragraph, and the current text has this style, a text in a new empty paragraph will be created using this property. If NextStyleNo=-1, a new paragraph will have the same text style as in the current paragraph.

If UseStyleTemplates[195]=*True*, this property is ignored. See the the description NextId[496] property of the paragraph's style template[473] about defining text properties for new paragraphs.

Default value:

-1

See also

- TParaInfo.NextParaNo[492].

```
property Unicode: Boolean;
```

Obsolete property. Always *True*.

Default value:

True

See also:

- Unicode in RichView[99].

Copies the contents of **Source** to this text style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TFont, TCustomRVInfo[463], TCustomRVFontInfo[466], or TFontInfo to this text style.

Example:

```
// Copying TFont to text style:
```

```
MyRVStyle.TextStyles[437][0].Assign(MyEdit.Font);

// Copying text style to text style:
MyRVStyle.TextStyles[437][0].Assign(MyRVStyle.TextStyles[437][1]);
// or, the same:
MyRVStyle.TextStyles[437][0] := MyRVStyle.TextStyles[437][1];
```

A constructor, creates and initializes a new instance of TFontInfo.

```
constructor Create(Collection: TCollection);override;
```

In addition to initializations performed by the inherited constructor[478], this methods initializes:

```
NextStyleNo[480]     := -1;
```

Other properties introduced in TFontInfo are initialized with zeros/*False*/empty values.

Note: usually, when you create a text style in code, you should assign Standard[464] := *False*.

Draws **s** on **Canvas**.

```
procedure Draw(const s: TRVUnicodeString[694]; Canvas: TCanvas; ThisStyleNo: Integer;
  SpaceAtLeft, Left, Top, Width, Height
  {$IFDEF RVUSEBASELINE},BaseLine{$ELSE}{$IFDEF RVHIGHFONTS},DrawOffsY{$ENDIF}{$ENDIF}: I
  TextWidth, SpaceCount: Integer;
  RVStyle: TRVStyle;
  DrawState: TRVTextDrawStates[692];
  Printing, PreviewCorrection, ForMetafile: Boolean;
  ColorMode: TRVColorMode[671]; DefBiDiMode: TRVBiDiMode[668];
  RefCanvas: TCanvas; GraphicInterface: TRVGraphicInterface;
  LeftNoExpIndex, RightNoExpIndex: Integer);
```

(changed in version 18)

Normally, this procedure is called from TRVStyle[418].OnDrawStyleText[448] event. You just need to pass all the parameters of this event to this method. The parameters that do not exist in OnDrawStyleText are marked with "*".

This method assumes that correct settings (font, colors, character spacing etc.) are already applied to **Canvas**.

Parameters:

s – string to draw.

Canvas – canvas to paint.

ThisStyleNo must be equal to the index of this text style in **RVStyle**.TextStyles[437].

SpaceAtLeft – distance from **Left** to the beginning of text. It must be 0 (unless you define RVOLDJUSTIFY in RV_Defs.inc).

Left, Top, Width, Height – rectangle where the text must be drawn.

BaseLine – Y coordinate of the text base line. This parameter exists only if RVUSEBASELINE is defined in RV_Defs.inc; in this case, **Canvas**'s text align must include TA_BASELINE.

DrawOffsY: this parameter exists only if RVHIGHFONTS is defined and RVUSEBASELINE is not defined in RV_Defs.inc; in this case, the text top is drawn at **Top+DrawOffsY**.

TextWidth – width of **s**; this parameter is used only if **SpaceCount**<>0. This parameter is ignored, if **PreviewCorrection**=*True* (text width is recalculated before drawing).

SpaceCount:

- if 0, the text is drawn as it is (unless **PreviewCorrection**=*True*);
- if positive, it must be equal to the count of space characters in **s**; in this case, **Width-SpaceAtLeft-TextWidth** is distributed between space characters;

- if equal to -100, **Width-SpaceAtLeft-TextWidth** is distributed between all characters in **s** that can be expanded.

RVStyle* – TRVStyle[418] component containing this text style.

Printing is *True* if the text is drawn while printing or print previewing.

PreviewCorrection is *True* if the text is drawn while print previewing[417], and TRVPrint.PreviewCorrection[552] =*True*. In this case, **s** is fit to **Width-SpaceLeft** by adjusting character spacing. This parameter is ignored, of **Canvas**<>**RefCanvas**.

ForMetafile is *True* if drawing must be compatible with metafiles.

ColorMode allows printing text as black and white (although colors must already be assigned to **Canvas**).

DefBiDiMode is a bi-di mode of paragraph containing this text.

RefCanvas – canvas used to measure text:

- when drawing in TRichView[162], TRichViewEdit[316], TRVReportHelper[540], TRVPrint[513], it must be equal to **Canvas;**
- when drawing in TSRichViewEdit (ScaleRichView), it must be canvas returned by SRichViewEdit.RichViewEdit.RVData.GetScaleRichViewInterface.GetFormatCanvas.

GraphicInterface* – **RVStyle**.GraphicInterface.

LeftNoExpIndex, RightNoExpIndex are used if **SpaceCount**=-100. If not negative, they specify indexes of characters in **s** (from 0) which must not be expanded.

Compares this text style with **Value**.

```
function IsEqual(Value: TCustomRVFontInfo; IgnoreList: TRVFontInfoProperties[679]): Boolean; c
```

(introduced in version 1.3)

Parameter:

Value – object to compare with this text style. The class of this object is TCustomRVFontInfo[466] or inherited from it. If **Value** is TFontInfo, values of NextStyleNo[480] property can be compared as well.

IgnoreList – set of values identifying properties to ignore when comparing.

The following properties are always ignored when comparing styles:

- Standard[464];
- ModifiedProperties[480];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

See also:

- TCustomRVFontInfo.IsEqual[478] (method of the ancestor class);
- TFontInfos.FindSuchStyle[455];
- TParaInfo.IsEqual[492].

1.5.2.1.5.4 TCustomRVParaInfo

This is an ancestor class for TParaInfo[490] (class of item in collection of paragraph styles[431]).

TCustomRVParaInfo introduces the most of properties of paragraph style.

Unit RVStyle.

Syntax

```
TCustomRVParaInfo = class(TCustomRVFontOrParaInfo[463])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVFontOrParaInfo 463
TCustomRVInfo 463

Properties

- Alignment, LastLineAlignment 483 – alignment of paragraph (left, right, center, justify or distribute);
- FirstIndent 485 – indent of the first line of paragraph, summarized with LeftIndent;
- LeftIndent 485 – indent from left margin;
- RightIndent 488 – indent from right margin;
- SpaceBefore 489 – spacing above the paragraph;
- SpaceAfter 489 – spacing below the paragraph;
- Border 485 – border around the paragraph;
- Background 484 – background of paragraph;
- Tabs 489 – collection of tab stops;
- LineSpacing 486, LineSpacingType 486 – line spacing;
- BiDiMode 484 – default text direction;
- Options 487 – protection, wrapping.
- OutlineLevel 488 allows defining headings.

Methods

IsEqual 490 compares this paragraph style with another one.

You can use Assign 489 method to assign TCustomRVParaInfo to TCustomRVParaInfo.

See Also

Classes:

- TParaInfo 490 – class inherited from this class, a paragraph style;
- TParaInfos 456 – collection of text styles (TParaInfo 490).

Overview:

- RichView Paragraphs 72

Alignment of paragraph of this style.

```
type
  TRVAlignment =
    (rvaLeft, rvaRight, rvaCenter, rvaJustify, rvaDistribute);
  TRVLastLineAlignment =
    (rvllaDefault, rvllaLeft, rvllaRight, rvllaCenter, rvllaJustify);

property Alignment : TRVAlignment;
property LastLineAlignment: TRVLastLineAlignment;
```

Possible values of **Alignment**:

Value	Meaning
rvaLeft	The paragraph is aligned to the left side.
rvaRight	The paragraph is aligned to the right side.
rvaCenter	The paragraph is centered.

rvaJustify	All lines of the paragraph (except for the last line) are aligned to the both left and right sides by adding space between words (i.e. by increasing widths of space characters). The last line is aligned according to **LastLineAlignment** property.
rvaDistribute	All lines of the paragraph (except for the last line) are aligned to the both left and right sides by adding space between all characters. The last line is aligned according to **LastLineAlignment** property.

LastLineAlignment is used if **Alignment** is equal to *rvaJustify* or *rvaDistribute*. It is applied to the last line of the paragraph (even if the paragraph has the only line). Additionally it is applied to lines that cannot be aligned to the both sides (for example, if a line contains only a single picture).

Possible values of **LastLineAlignment**:

Value	Meaning
rvllaDefault	The line is aligned to the left side if the paragraph is aligned from left to right [484], or to the right side otherwise.
rvllaLeft	The line is aligned to the left side.
rvllaRight	The line is aligned to the right side.
rvllaCenter	The line is centered.
rvllaJustify	The line is aligned according to **Alignment** property.

Note: Microsoft Word 2013 uses the following combinations:

- *rvaJustify* + *rvllaDefault*
- *rvaDistribute* + *rvllaJustify*

Default value:

- Alignment: *rvaLeft*
- LastLineAlignment: *rvllaDefault*

Defines background for paragraphs of this style (transparent by default)

```
property Background:TRVBackgroundRect[500];
```

(Introduced in version 1.2)

See TRVBackgroundRect [500] for additional information.

Note: RTF and HTML support only equal Background.BorderOffsets [501] and Border [485].BorderOffsets [502]. RTF and DocX does not support Opacity [501].

See also:

- Paragraph Background [73].

Specifies default text direction for paragraphs of this style.

```
property BiDiMode: TRVBiDiMode[668];
```

(introduced in version 1.6)

Value	Meaning
rvbdUnspecified	Text direction is inherited from higher level, i.e. from TCustomRichView (see BiDiMode[170])
rvbdLeftToRight	Default text reading order is left-to-right
rvbdRightToLeft	Default text reading order is right-to-left

If two adjacent text items have adjacent LTR (RTL) characters, these items will be arranged LTR (RTL), regardless of the paragraph's BiDiMode.

Default value:

rvbdUnspecified

See also:

- Bidirectional text in TRichView[100].

Defines a border around paragraphs of this style (none by default)

```
property Border:TRVBorder[501];
```

(Introduced in version 1.2)

See TRVBorder[501] for additional information.

Note: RTF and HTML support only equal Background[484].BorderOffsets[501] and Border.BorderOffsets[502].

See also:

- Paragraph Background[73].

Indent of the first line for paragraphs of this style.

```
property FirstIndent: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

This value can be negative.

Indent of first paragraph line from the left border (assuming that indents are converted to pixels) = TCustomRichView.LeftMargin[180]+LeftIndent[485]+FirstIndent.

In paragraphs with bullets or numbering, this property is overridden by FirstIndent[507] of list level.

Default value:

0

See also properties:

- LeftIndent[485], RightIndent[488];
- SpaceBefore[489], SpaceAfter[489].

See also properties of TRVListLevel:

- FirstIndent[507].

Left indent for paragraphs of this style.

```
property LeftIndent: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

Indent for paragraph from left border (assuming that indents are converted to pixels) = TCustomRichView.LeftMargin[180]+LeftIndent [+ FirstIndent[485] for the first line]

In paragraphs with bullets or numbering, this property is overridden with LeftIndent[510] of list level (if BiDiMode[484] <>*rvbdRightToLeft*).

Default value:

0

See also properties:

- RightIndent[488];
- SpaceBefore[489], SpaceAfter[489].

See also properties of TRVListLevel:

- LeftIndent[510].

Defines spacing between lines in paragraphs of this style

```
type
  TRVLineSpacingValue = type Integer;
property LineSpacing: TRVLineSpacingValue;
```

(Introduced in version 1.5)

The meaning of the property depends on the value of LineSpacingType[486] property.

If LineSpacingType is *rvlsPercent*, this property defines line height in percent of normal line height (a height of the highest character in a line, heights of pictures and controls are not taken into account). Additional space will be added below the line (including the last paragraph line). For example, LineSpacing=100 means single spacing (no additional space will be added), LineSpacing=200 means double spacing. Values lesser than 50 are ignored.

With other values LineSpacingType, this value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

If LineSpacingType is *rvlsSpaceBetween*, this value will be added below each line (including the last line in a paragraph). Negative values are ignored.

If LineSpacingType is *rvlsLineHeightAtLeast*, this property defines minimal line height. If the actual line height (including all pictures and controls) is less than this value, additional spacing is added (below and above the line).

If LineSpacingType is *rvlsLineHeightExact*, this property defines line height. If the actual line height (including all pictures and controls) is not equal to this value, additional spacing is added/subtracted (below and above the line).

Default value:

100

Defines type of line spacing in paragraphs of this style.

```
type
  TRVLineSpacingType = (rvlsPercent, rvlsSpaceBetween);
property LineSpacingType: Integer;
```

(Introduced in version 1.5, 11)

Value	Meaning
rvlsPercent	LineSpacing[486] defines spacing in % of text height. Extra spacing is added/subtracted below lines.
rvlsSpaceBetween	LineSpacing[486] is a value which will be added between lines, measured in RVStyle.Units[437]. This type of line

	spacing cannot be exported in RTF and HTML. Extra spacing is added below lines.
rvlsLineHeightAtLeast	LineSpacing[486] defines minimal line height, measured in RVStyle.Units[437]. Extra spacing is added above and below lines.
rvlsLineHeightExact	LineSpacing[486] defines line height, measured in RVStyle.Units[437]. Extra spacing is added/subtracted above and below lines.

Default value:

rvlsPercent

Several options for paragraph

```
type
  TRVParaOption =
    (rvpaoNoWrap, rvpaoReadOnly,
      rvpaoStyleProtect, rvpaoDoNotWantReturns,
      rvpaoKeepLinesTogether, rvpaoKeepWithNext,
      rvpaoWidowOrphanControl);
  TRVParaOptions = set of TRVParaOption;

property Options: TRVParaOptions;
```

(Introduced in version 1.5 - 10)

Layout Options

Option	Meaning
rvpaoNoWrap	Disallows automatic word wrapping in paragraphs of this style. If you use no-wrap paragraphs, it's recommended to add *rvoClientTextWidth* in RichView.Options[182]

Editing and Protection Options:

Option	Meaning
rvpaoReadOnly	Disallows editing paragraphs of this style in TRichViewEdit[316]. This option does not protect from adding new paragraph when the user presses Enter at the beginning or at the end of the paragraph. This option does not protect this paragraph from deletion together with the larger selection.
rvpaoStyleProtect	If set, ApplyParaStyle[331], ApplyParaStyleTemplate[332], ApplyStyleTemplate[333] cannot change style of paragraphs of this style. This option does not protect from ApplyParaStyleConversion[332], but you can take it into account in OnParaStyleConversion[397] event.

rvpaoDoNotWantReturns	If set, Enter key will be blocked if the caret is inside paragraph of this style.

Printing Options

Option	Meaning
rvpaoKeepLinesTogether	Paragraph of this style is printed on one page, if possible.
rvpaoKeepWithNext	Paragraph of this style is printed on the same page as the next paragraph, if possible. Additionally, this option works like *rvpaoKeepLinesTogether*
rvpaoWidowOrphanControl	Prevents page breaks after the first line and before the last line in paragraphs of this style.

Default value:

[]

See also:

- TFontInfo.Protection[473];
- TCustomRichView.WordWrap[196].
- TRVTableRow.KeepTogether[615]

Defines outline level of paragraph.

```
property OutlineLevel: Integer;
```

(introduced in v12)

0 – no outline level (body text).

Positive values define levels of heading (1 – heading level 1; 2 – heading level 2, and so on).

Default value:

0

Right indent for paragraphs of this style.

```
property RightIndent: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

Indent for paragraph from the right border (assuming that indents are converted to pixels) = TCustomRichView.RightMargin[185]+RightIndent.

In paragraphs with bullets or numbering, this property is overridden with LeftIndent[510] of list level, if BiDiMode[484] =*rvbdRightToLeft*.

Default value:

0

See also properties:

- LeftIndent[485];
- SpaceBefore[489], SpaceAfter[489].

See also properties of TRVListLevel:

- LeftIndent[510].

Specifies the size of the spacing below the paragraph.

```
property SpaceAfter: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

This value is greater than or equals to zero (assigning a negative value assigns zero).

Default value:

0

See also properties:

- SpaceBefore[489];
- LeftIndent[485], RightIndent[488].

Specifies the size of the spacing above the paragraph.

```
property SpaceBefore: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

This value is greater than or equals to zero (assigning a negative value assigns zero).

Default value:

0

See also properties:

- SpaceAfter[489];
- LeftIndent[485], RightIndent[488].

Defines tab stops and their properties for paragraphs of this style.

```
property Tabs: TRVTabInfos[503];
```

(introduced in v1.9)

This is a sorted collection of TRVTabInfo[504] (sorted by Position[505] in ascending order).

Default value:

Empty collection.

See also properties of TRVStyle:

- DefTabWidth[422].

Copies the contents of **Source** to this paragraph style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TCustomRVInfo[463] or TCustomRVParaInfo to this paragraph style.

See also:

- TParaInfo.Assign[492] (TParaInfo[490] is inherited from TCustomRVParaInfo).

A constructor, creates and initializes a new instance of TCustomRVParaInfo.

```
constructor Create(Collection: TCollection);override;
```

The constructor initializes:

```
// Properties, inherited from TCustomRVInfo[463]:
Standard[464]        := True;
StyleName[465]       := 'Paragraph Style';
// Properties defined in TCustomRVParaInfo:
Alignment[483]       := rvaLeft;
```

```
LineSpacingType[486] := 100;
LineSpacing[486]     := rvlsPercent;
```

Other properties are initialized with zeros/*False*/empty values.

Note: usually, when you create a paragraph style in code, you should assign Standard[464] := *False*.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties[684]): Boolean; dynamic;
```

(introduced in version 1.5)

Parameters:

Value – object to compare with this paragraph style. The class of this object is TCustomRVParaInfo or inherited from it.

IgnoreList – set of values identifying properties to ignore when comparing, see TRVParaInfoProperties[684].

The following properties are always ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default)).

See also:

- TParaInfo.IsEqual[492] (TParaInfo[490] is inherited from this class);
- TCustomRVFontInfo.IsEqual[478].

1.5.2.1.5.5 TParaInfo

This is a paragraph style. It defines attributes of paragraphs[72] in TRichView[162], TRichViewEdit[316], TDBRichView[407], TDBRichViewEdit[410] controls.

This is an item in the collection of TRVStyle[418].ParaStyles[431]. The class of the collection itself is TParaInfos[456].

Unit RVStyle.

Syntax

```
TParaInfo = class(TCustomRVParaInfo[482])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[463]
TCustomRVFontOrParaInfo[463]
TCustomRVParaInfo[482]

Properties

The most of properties are inherited from TCustomRVParaInfo[482].

The new properties are:

- DefStyleNo[491] – default text style for paragraph;
- NextParaNo[492] – index of paragraph style to apply after pressing Enter in editor (if style templates are not used[195]);
- ModifiedProperties[491] may (temporarily) contain a list of properties having values different from values defined in a style template[465].

Methods

IsEqual[492] compares this paragraph style with another one.

You can use Assign[492] method to assign:

- TParaInfo to TParaInfo;
- TCustomRVParaInfo[482] to TParaInfo and vice versa.

See Also

- TParaInfos[456] – class of collection of paragraph styles;
- TFontInfo[478] – class of text style;
- TRVListInfo[493] – class of list style;
- RichView Paragraphs[72].

Specifies the default text style for paragraphs of this style.

```
property DefStyleNo: Integer;
```

(introduced in version 1.8)

Deprecated! Use style templates[435] to link text and paragraph properties together.

You can use the rvsDefStyle[711] constant (= MaxInt, a large positive integer value) as a StyleNo parameter of AddNL[203] and other methods for adding text.

It means using the paragraph's default text style, i.e. DefStyleNo property (or the 0-th style, if DefStyleNo=-1).

GetItemStyle[229] method and CurTextStyleNo[324] return a translated text style (they never return rvsDefStyle).

Use GetItem[225](i).StyleNo and ActualCurTextStyleNo[322] to get the text style as is.

Default value:

-1

This property may contain a list of properties not defined in the style template[435] of this style.

```
property ModifiedProperties: TRVParaInfoProperties[684];
```

(introduced in version 14)

This property is used only if style templates are used[195].

This is a redundant property: the difference in properties is determined not by this property, but by the actual difference in values of properties of this style and its style template.

This property is ignored when searching for or comparing paragraph styles.

Some methods may change value of this property, so do not use this property to store any information. Value of this property should be calculated just before using.

See also properties:

- StyleTemplateId[465].

See also properties of TFontInfo:

- ModifiedProperties[480].

See also methods of TRVStyleTemplate:

- UpdateModifiedParaStyleProperties[500];
- ApplyToParaStyle[498] (contains an example).

Defines the paragraph style for following paragraphs.

```
property NextParaNo: Integer;
```

(Introduced in version 1.4)

This property may contain either an index in Style[193].ParaStyles[431], or the value -1.

This property is used in RichViewEdit[316], only if UseStyleTemplates[195]=*False*. When a user presses Enter key at the end of a paragraph of this style, a new empty paragraph will be created using this property. If NextParaNo=-1 then this new line will have the same paragraph style as the current paragraph.

If UseStyleTemplates[195]=*True*, this property is ignored, NextId[496] property of the paragraph's style template[465] is used instead.

Default value:

-1

See also:

- TFontInfo.NextStyleNo[480].

Copies the contents of **Source** to this paragraph style.

```
procedure Assign(Source: TPersistent); override;
```

Call Assign to copy properties of TCustomRVInfo[463], TCustomRVParaInfo[482], or TParaInfo to this paragraph style.

Example:

```
// Copying paragraph style to paragraph style:
MyRVStyle.ParaStyles[431][0].Assign(MyRVStyle.ParaStyles[431][1]);
// or, the same:
MyRVStyle.ParaStyles[431][0] := MyRVStyle.ParaStyles[431][1];
```

A constructor, creates and initializes a new instance of TParaInfo.

```
constructor Create(Collection: TCollection);override;
```

In addition to initializations performed by the inherited constructor[489], this methods initializes:

```
DefStyleNo[491] := -1;
NextParaNo[492] := -1;
```

Note: usually, when you create a paragraph style in code, you should assign Standard[464] := *False*.

Compares this paragraph style with **Value**.

```
function IsEqual(Value: TCustomRVParaInfo;
  IgnoreList: TRVParaInfoProperties[684]): Boolean; override;
```

(introduced in version 1.5)

Parameters:

Value – object to compare with this paragraph style. The class of this object is TCustomRVParaInfo or inherited from it.

IgnoreList – set of values identifying properties to ignore when comparing, see TRVParaInfoProperties[684].

The following properties are always ignored when comparing styles:

- Standard[464];
- StyleName[465] (if RichViewCompareStyleNames[701]=*False* (default));
- ModifiedProperties[491].

See also:

- TCustomRVParaInfo.IsEqual[490] (method of the ancestor class);
- TParaInfos.FindSuchStyle[458];
- TFontInfo.IsEqual[482].

1.5.2.1.5.6 TRVListInfo

This is a paragraph list style. It defines attributes of list markers[131] in TRichView[162], TRichViewEdit[316], TDBRichView[407], TDBRichViewEdit[410] controls.

This is an item in the collection of TRVStyle[418].ListStyles[429]. The class of the collection itself is TRVListInfos[458].

Unit RVStyle.

Syntax

```
TRVListInfo = class(TCustomRVInfo[463])
```

Hierarchy

TObject
TPersistent
TCollectionItem
TCustomRVInfo[463]

Properties

- Levels[493] – collection of list levels (TRVListLevel[506])
- OneLevelPreview[493] suggests appearance of thumbnail.

Other properties are inherited from TCustomRVInfo[463].

Methods

- AllNumbered[494] returns *True* if all levels represent numbered lists;
- HasNumbering[494] returns *True* if at least one of list levels represents a numbered list.

See Also

- TRVListInfos[458] – class of collection of paragraph list styles;
- TFontInfo[478] – class of text style;
- TParaInfo[482] – class of paragraph style;
- List markers item type[131].

Levels of paragraph list style, collection of TRVListLevel[506] objects.

```
property Levels: TRVListLevelCollection[506];
```

Each level defines properties for bullet or numbering of paragraphs.

Can be used by application to define appearance of preview thumbnail for this list style.

```
property OneLevelPreview: Boolean
```

This property is not used by the components themselves.

If application creates thumbnails in dialog for selecting list style, this property can be used to define thumbnail image.

If it is *True*, only one level of this list should be shown on the thumbnail.

See also:

- Levels[493].

Returns *True* if all levels[493] represent numbered lists

```
function AllNumbered: Boolean;
```

See also:

- HasNumbering[494].

Returns *True* if at least one of list levels[493] represents a numbered list.

```
function HasNumbering: Boolean;
```

See also:

- AllNumbered[494].

1.5.2.1.5.7 TRVStyleTemplate

This class represents a "real" style controlling properties of TFontInfo[478] and TParaInfo[478] objects.

This is an item in the collection of TRVStyle[418].StyleTemplates[435]. The class of the collection itself is TRVStyleTemplateCollection[460].

Unit RVStyle.

Syntax

```
TRVStyleTemplate = class(TCollectionItem)
```

(introduced in version 14)

Hierarchy

TObject
TPersistent
TCollectionItem

Main Properties

- Kind[495] – kind of this style templates (defines it applicability to text or paragraphs);
- Name[496] – unique name;
- Id[495] – unique identifier;
- TextStyle[497], ValidTextProperties[498] – text properties;
- ParaStyle[497], ValidParaProperties[498] – paragraph properties;
- ParentId[497] – a reference to a parent style template;
- NextId[496] – a reference to a style template for following paragraphs;
- QuickAccess[497] – recommends to provide a quick access to this style template in applications UI.

Main Methods

- UpdateModifiedTextStyleProperties[500] fills the set of properties of the specified text style that were changed after applying this style template;
- UpdateModifiedParaStyleProperties[500] fills the set of properties of the specified paragraph style that were changed after applying this style template;
- ApplyToTextStyle[499] applies this style template to the specified text style;
- ApplyToParaStyle[498] applies this style template to the specified paragraph style.

Linking

Each style template has two properties having unique values: Id[495] and Name[496]. Id is read-only, is assigned automatically, the collection provides its uniqueness. Name is assigned by the user, you (the programmer) must provide its uniqueness.

Links[465] from TextStyles[437], ParaStyles[431], and between StyleTemplates are made via Id property.

Name is used when merging style templates of two different documents.

Unlike items in TRVStyle.TextStyles[437] and TRVStyle.ParaStyles[431], style templates are never referred by their index in the collection (except for indices returned after searching by a name or an identifier, and the editor methods for applying to the selection[333]).

See also

- Styles and style templates[74].

An identifier of this style template.

```
property Id: TRVStyleTemplateId[690];
```

Value of this property is generated automatically using a random number generator. Each item has an identifier that is unique in the collection[460].

Different style template collections may have equal values of identifiers of some items by a coincidence, however it does not mean that these style templates are related (unless the collection was duplicated using AssignStyleTemplates[461] with CopyIds=*True*).

This property is used to refer to this style template within a single TRVStyle[418] component:

- another style template may refer to this style-template using ParentId[497] property;
- another style template may refer to this style-template using NextId[496] property;
- text[437] and paragraph[431] styles of the same TRVStyle may refer to this style template using StyleTemplateId[465] property.

Why do style templates need two unique properties, **Id** and Name[496]? There are the following reasons:

- editing style templates in a dialog window: if a user renames a style template, all references to this style template must stay unchanged;
- working with integer values is more efficient than with strings.

This property is not copied when assigning one style template to another.

See also methods of TRVStyleTemplateCollection[460]:

- FindById, FindItemById[462].

Specifies the kind of this style template.

```
type
  TRVStyleTemplateKind = (rvstkParaText, rvstkPara, rvstkText);
property Kind: TRVStyleTemplateKind;
```

Kind	Meaning
rvstkParaText	This style template can be applied both to text and paragraph styles.
rvstkPara	This style template can be applied to paragraphs styles[431] (i.e. paragraph styles can be linked[465] to this style template)
rvstkParaText	This style template can be applied to text styles[437] (i.e. text styles can be linked[465] to this style template)

Assigning a new value to Name[496] may change Kind.

Default value:

rvstkParaText.

Name of the style template.

```
property Name: TRVStyleTemplateName;
```

You must provide that each item in the collection has an unique name, otherwise style templates will be saved to RTF and RVF incorrectly.

Special names:

- Assigning 'Normal' changes Kind[495] to *rvstkPara*.
- Assigning 'heading 1'...'heading 9' changes Kind to *rvstkParaText*.
- Assigning 'Hyperlink' changes Kind to *rvstkText*.

Names are used when merging style templates from different documents.

Some names may be reserved for standard style templates. Actually, there is no strict definition of a "standard style template". A "standard style template" is a style template having a special meaning for your application. For example, your application may have a command "Insert Abracadabra" inserting some text and formatting using the style template named "Abracadabra Text". In this case, "Abracadabra Text" is a standard style template for your application.

However, there are style templates having a special meaning for TRichView: "Normal" and "Hyperlink".

"Normal" is a paragraph style template. This style template is special, because it uses some "magic". All paragraph style templates are applied to paragraphs and cannot be applied to text. But applying "Normal" to text clears this text format.

The format clearing command does the following. For paragraphs, it assigns the style template "Normal" and clears all additional paragraph attributes. For non-hypertext text, it removes its style template and clears all additional text attributes. For hypertext links, it assigns the style template "Hyperlink" and clears all additional text attributes.

 RichViewActions reserve some other names for standard style templates, but only the following are used specially, and only in ScaleRichView:

- "header" to apply to headers of new documents;
- "footer" to apply to footers of new documents;
- "endnote reference"/"footnote reference" to apply to new footnote[136]/endnote[135] characters, and to note references[140];
- "endnote text", "footnote text" to apply to text in Document[628] of new footnote/endnote.

See also members of TRVStyleTemplateCollection[460]:

- FindByName, FindItemByName[462];
- DefStyleName[460].

Defines the style template to be used in the next paragraph after the paragraph marked by this style template (when a user presses Enter)

```
property NextId: TRVStyleTemplateId[690];
property Next: TRVStyleTemplate; // read-only
```

This property is used only for style templates applied to paragraphs, so it is ignored if Kind[495]=*rvstkText*.

To make some style template a next style template, assign its Id[495] to **NextId** of this style template.

NextId equals to Id[495] of the next style (If **NextId**=-1, the next style-template is this style template itself)

Next returns the next style (or *nil* if **NextId**=-1), i.e. **Next**.Id[495]=**NextId**.

When Enter is pressed, and style templates are used[195]:

- the new paragraph is linked to **Next** (or to this style template, if **Next**=*nil*); its attributes are changed accordingly;
- if the current text is a hyperlink (i.e. its Jump[479]=*True*), the new text is not linked to any style template, its attributes are changed accordingly, its Jump[479]=*False*;
- if the current text is not a hyperlink, the new text is linked to the same style template as the current text.

New TRVStyle.TextStyles[437] and ParaStyles[431] are added, if necessary.

These properties are copied when assigning one style template to another, only if the source and target style templates belong to the same collection.

Default value:

NextId: -1

Paragraph properties.

```
type
  TRVSTParaInfo = class (TCustomRVParaInfo[482]);
property ParaStyle: TRVSTParaInfo;
```

ParaStyle is ignored if Kind[495]=*rvstkText*

Only sub-properties listed in ValidParaProperties[498] are taken into account. Values of other sub-properties are ignored.

Defines the style template the current style template is based on.

```
property ParentId: TRVStyleTemplateId[690];
property Parent: TRVStyleTemplate; // read-only
```

To make some style template a parent of this style template, assign its Id[495] to **ParentId** of this style template.

To make this style template a root style template, assign -1.

Parent returns the parent style template (or *nil* if **ParentId**=-1), i.e. **Parent**.Id[495]=**ParentId**.

If Kind[495]=*rvstkText*, only style templates having Kind[495]=*rvstkText* can be its parents.

If Kind[495]<>*rvstkText*, only style templates having Kind[495]<>*rvstkText* can be its parents.

Circular references are not allowed.

The style template inherits (from parents) properties not listed in its ValidTextProperties[498] and ValidParaProperties[498].

These properties are copied when assigning one style template to another, only if the source and target style templates belong to the same collection.

Default value:

ParentId: -1

Specifies that this style template should be readily available for a user via the application's user interface.

```
property QuickAccess: Boolean;
```

For example, when RichViewActions show style templates in TRVStyleTemplateComboBox/TRVStyleTemplateListBox, quick-access styles are always shown, even if they are not used in the document.

Default value

True

Text properties.

```
type
  TRVSTFontInfo = class (TCustomRVFontInfo[466]);
property TextStyle: TRVSTFontInfo;
```

Only sub-properties listed in ValidTextProperties[498] are taken into account. Values of other sub-properties are ignored.

A set of properties of ParaStyle[497] defined in this style template.

```
property ValidParaProperties: TRVParaInfoProperties[684];
```

Only properties included in this set are used. Values of other ParaStyle[497]'s properties are ignored.

This property is used only if Kind[495]<>*rvstkText* (otherwise, none of paragraph properties are used).

Although this property is public, not published, it may be defined at design-time in the Object Inspector (where it is displayed as ValidParaPropertiesEditor). It was made because of a Delphi limitation on the count of values in published set properties.

See also:

- ParentId[497].

A set of properties of TextStyle[497] defined in this style template.

```
property ValidTextProperties: TRVFontInfoProperties[679];
```

Only properties included in this set are used. Values of other TextStyle[497]'s properties are ignored.

Although this property is public, not published, it may be defined at design-time in the Object Inspector (where it is displayed as ValidTextPropertiesEditor). It was made because of a Delphi limitation on the count of values in published set properties.

See also:

- ParentId[497].

Changes properties of **AParaStyle** according to the properties of this style template.

```
procedure ApplyToParaStyle(AParaStyle: TCustomRVParaInfo[482]);
```

The method changes only properties of **AParaStyle** not listed in **AParaStyle**.ModifiedProperties[491]. If you want to change all properties, assign **AParaStyle**.ModifiedProperties[491]=[] before calling this method. If you want to change only properties controlled by style templates, call UpdateModifiedParaStyleProperties[500] before calling this method.

Applying a style template means applying properties of its ParaStyle[497] listed in ValidParaProperties[498], and properties inherited from Parent[497].

Kind[495] of this style template must be either *rvstkPara* or *rvstkParaText*.

The sequence of operations:

1. The method applies this style template (and its parents) to **AParaStyle**. The method assigns **AParaStyle**.StyleTemplateId[465] = Id[495]. This method may be called for *nil*-objects; in this case, it just assigns **AParaStyle**.StyleTemplateId[465] = -1.
2. Then it resets remaining properties of **AParaStyle** to default values.

Example: resetting TextStyles and ParaStyles for using in new documents, using 'Normal' style template.

```
var StyleTemplate: TRVStyleTemplate[494];
begin
  RVStyle.ListStyles[429].Clear;
  RVStyle.TextStyles[437].Clear;
  RVStyle.TextStyles.Add;
  RVStyle.ParaStyles[431].Clear;
  RVStyle.ParaStyles.Add;
  StyleTemplate := RVStyle.StyleTemplates[435].NormalStyleTemplate[461];
  if StyleTemplate<>nil then begin
    RVStyle.ParaStyles[0].ModifiedProperties[491] := [];
```

```
    StyleTemplate.ApplyToParaStyle(RVStyle.ParaStyles[0]);
    RVStyle.TextStyles[0].ModifiedProperties := [];
    TRVStyleTemplate(nil).ApplyToTextStyle(RVStyle.TextStyles[0], StyleTemplate);
    RVStyle.TextStyles[0].ParaStyleTemplateId := StyleTemplate.Id;
  end;
end;
```

See also:

- ApplyToTextStyle

Changes properties of **ATextStyle** according to the properties of this style template.

```
procedure ApplyToTextStyle(ATextStyle: TCustomRVFontInfo;
  ParaStyleTemplate: TRVStyleTemplate);
```

The method changes only properties of **ATextStyle** not listed in **ATextStyle**.ModifiedProperties. If you want to change all properties, assign **ATextStyle**.ModifiedProperties=[] before calling this method. If you want to change only properties controlled by style templates, call UpdateModifiedTextStyleProperties before calling this method.

Applying a style template means applying properties of its TextStyle listed in ValidTextProperties, and properties inherited from Parent.

Kind of this style template must be either *rvstkText* or *rvstkParaText*.

Up to two style templates can be applied to **ATextStyle**: this style template and **ParaStyleTemplate**. This style template has a higher priority than **ParaStyleTemplate**: if some property is defined both in this style template and in **ParaStyleTemplate**, properties of this style template are applied. Both this style template and **ParaStyleTemplate** may be *nil*.

The sequence of operations:

1. The method applies this style template (and its parents) to **ATextStyle**. The method assigns **ATextStyle**.StyleTemplateId = Id. This method may be called for *nil*-objects; in this case, it just assigns **ATextStyle**.StyleTemplateId = -1.
2. Then it applies **ParaStyleTemplate** (if not *nil*) to remaining properties. The method does NOT assign **ATextStyle**.ParaStyleTemplateId.
3. Then it resets remaining properties of **ATextStyle** to default values.

Example: inserting a hyperlink in rve:TRichViewEdit. If possible, this hyperlink is formatted using 'Hyperlink' style template.

```
var
  OldStyleTemplate, NewStyleTemplate, ParaStyleTemplate: TRVStyleTemplate;
  TextStyle: TFontInfo;
begin
  // making a copy of the current text attributes (of the current text style)
  TextStyle := TFontInfo.Create(nil);
  TextStyle.Assign(rve.Style.TextStyles[rve.CurTextStyleNo]);
  // searching for 'Hyperlink' style template
  NewStyleTemplate := rve.Style.StyleTemplates.FindItemByName('Hyperlink');
  if NewStyleTemplate<>nil then begin
    // 'Hyperlink' style template exists. applying it to TextStyle
    OldStyleTemplate := rve.Style.StyleTemplates.FindItemById(TextStyle.StyleTemplateId
    ParaStyleTemplate := rve.Style.StyleTemplates.FindItemById(TextStyle.ParaStyleTemplat
    OldStyleTemplate.UpdateModifiedTextStyleProperties(TextStyle, ParaStyleTemplate);
    NewStyleTemplate.ApplyToTextStyle(FontInfo, ParaStyleTemplate);
    end
  else begin
    // 'Hyperlink' style template does not exists. applying default hyperlink properties
    TextStyle.Color := clBlue;
    TextStyle.Style := [fsUnderline];
```

```
  end;
  TextStyle.Jump := True;
  // inserting
  rve.CurTextStyleNo := rve.Style.FindTextStyle(TextStyle);
  rve.InsertStringTag('TRichView.com', 'http://www.trichview.com');
  // cleaning up
  TextStyle.Free;
end;
```

See also:

- ApplyToParaStyle (contains one more example)

The methods fill the list of properties having different values in **ATextStyle/AParaStyle** and in this style template.

```
procedure UpdateModifiedTextStyleProperties(ATextStyle: TFontInfo;
  ParaStyleTemplate: TRVStyleTemplate);
procedure UpdateModifiedParaStyleProperties(AParaStyle: TParaInfo);
```

UpdateModifiedTextStyleProperties compares values of properties of **ATextStyle** and properties of this style template (i.e. properties of TextStyle listed in ValidTextProperties and properties inherited from Parent). Properties undefined in this style template (and its parents) are compared with properties of **ParaStyleTemplate** (if **ParaStyleTemplate**<>*nil*). Properties having different values are added in **ATextStyle**.ModifiedProperties.

UpdateModifiedParaStyleProperties compares values of properties of **AParaStyle** and properties of this style template (i.e. properties of ParaStyle listed in ValidParaProperties and properties inherited from Parent). Properties having different values are added in **AParaStyle**.ModifiedProperties.

These methods may be called for *nil*- style templates. In this case, properties of **ATextStyle/AParaStyle** are compared with default values.

The results are used in ApplyToTextStyle and ApplyToParaStyle methods.

1.5.2.1.6 Classes of Properties of Paragraph Style

1.5.2.1.6.1 TRVBackgroundRect

This class defines background of paragraph (colored rectangle). This is a class of Background property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBackgroundRect = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- BorderOffsets;
- Color;
- Opacity.

See also:

- TCustomRVParaInfo.Background;
- Paragraph Borders and Background.
- TRVBoxBackgroundRect (background of a floating box)

Distance between the paragraph content and the border of the colored area.

```
property BorderOffsets: TRVRect[657];
```

Sides of this rectangle are measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns the paragraph style.

Lager positive values increase padding. Negative values force border to be inside the paragraph content.

Warning:

- Background[484].BorderOffsets.Top must not exceed SpaceBefore[489] property of the same paragraph.
- Background.BorderOffsets.Bottom must not exceed SpaceAfter[489] property of the same paragraph.
- Background.BorderOffsets.Left should not exceed LeftIndent[485] property of the same paragraph.
- Background.BorderOffsets.Right should not exceed RightIndent[488] property of the same paragraph.

Default value:

0,0,0,0 (border is drawn around the paragraph without gaps)

See also:

- Paragraph Background[73]

Color of paragraph background.

```
property Color: TColor
```

Default value:

clNone (transparent background)

Opacity of paragraph background

```
property Opacity: TRVOpacity[683]
```

(introduced in version 16)

This property is used if Color[501]<>*clNone*.

Default value:

100000 (i.e. 100%)

1.5.2.1.6.2 TRVBorder

This class defines border around paragraph. This is a class of Border[485] property of paragraph style.

Unit RVStyle.

Syntax

```
TRVBorder = class (TPersistent)
```

(Introduced in version 1.2)

Properties:

- BorderOffsets[502] – distance between paragraph content and border;
- Color[502] – border color;
- InternalWidth[502] – distance between border lines (for double and triple borders);
- Style[502] – border style;
- VisibleBorders[502] hides/shows border sides;
- Width[503] – line width.

See also:

- TCustomRVParaInfo.Border[485];
- Paragraph Borders and Background[73].

Spacing between paragraph content and border.

```
property BorderOffsets: TRVRect[657];
```

Sides of this rectangle are measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns the paragraph style.

Lager positive values increase spacing. Negative values force the border to be inside the paragraph content.

Warning:

- Border[485].BorderOffsets.Top must not exceed SpaceBefore[489] property of the same paragraph.
- Border.BorderOffsets.Bottom must not exceed SpaceAfter[489] property of the same paragraph.
- Border.BorderOffsets.Left should not exceed LeftIndent[485] property of the same paragraph.
- Border.BorderOffsets.Right should not exceed RightIndent[488] property of the same paragraph.

Default value:

0,0,0,0

See also:

- Paragraph Borders[73].

Color of border

```
property Color: TColor
```

This property cannot be used to hide the border. To hide border, set Style[502]=*rvbsNone*.

Default value:

clWindowText

Spacing between lines in double and triple borders.

```
property InternalWidth: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns this paragraph style.

For example, if Style[502]=*rvbDouble*, InternalWidth=1, and Width[503]=1, and RVStyle.Units[437]=*rvstuPixels*, you'll see two lines having 1-pixel width, with 1-pixel spacing between them.

Default value:

1
Note: this default value assumes that RVStyle.Units[437]=*rvstuPixels*, otherwise it could be too small.

Style of a border

```
property Style: TRVBorderStyle[669];
```

See TRVBorderStyle[669] for possible values.

Default value:

rvbNone (no border)

Defines which sides of border are visible.

```
property VisibleBorders: TRVBooleanRect[653];
```

Default value:

True, *True*, *True*, *True* (all sides are visible, but border is still invisible because its Style[502]=*rvbNone* by default)

Width of border lines.

```
property Width: TRVStyleLength;
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units of TRVStyle component which owns this paragraph style.

It's not necessary a total width of a border, It's a width of lines in the border. This property is equal to a width of border only if Style=*rvbSingle*.

Thick lines (Style=*rvbThickInside* or *rvbThickOutside*) have double width.

Default value:

1

Note: this default value assumes that RVStyle.Units=*rvstuPixels*, otherwise it could be too small.

1.5.2.1.6.3 TRVTabInfos

Collection of tab stops (TRVTabInfo) and their properties.

This is a type of TRVStyle.ParaStyles[i].Tabs property.

Unit RVStyle.

Syntax

```
TRVTabInfos = class(TCollection)
```

(introduced in version 1.9)

Hierarchy

TObject
TPersistent
TCollection

Using

This collection is sorted in ascending order by Position properties of its items.

When you add a new tab stop (using Add method), it is added to the end of the collection. When you assign its Position, it is moved to the proper place.

Main properties:

- Items;
- Count.

Provides indexed access to the items in the collection of tab stops for the paragraph.

```
property Items[Index:Integer]: TRVTabInfo;
```

This is the default property of TRVTabInfos, so you can write

```
MyRVStyle.ParaStyles[i].Tabs[j]
```

instead of

```
MyRVStyle.ParaStyles[i].Tabs.Items[j].
```

Adds tab stops from **Source**.

```
procedure AddFrom(Source: TRVTabInfos);
```

New tab stops are inserted, existing tab stops are updated.

(tab stops are compared by their Position property)

The collection remains sorted.

Deletes tab stops at the specified **Positions**

```
procedure DeleteList(Positions: TRVIntegerList656);
```

All tab stops having Position505 property equal to one of values in **Positions** are deleted.

Returns the index of tab stop with the given **Position** (or -1 if not found).

```
function Find(Position: Integer): Integer;
```

This function compares value of **Position** parameter with value of Position505 property of tab stops.

Deletes all tab stops that do not present in **Value**. Only tab stops with all common properties are not deleted.

```
procedure Intersect(Value: TRVTabInfos503);
```

1.5.2.1.6.4 TRVTabInfo

Defines properties for tab stop. This is a type of item in TRVStyle418.ParaStyles431[i].Tabs489 collection.

Unit RVStyle.

Syntax

```
TRVTabInfo = class(TCollectionItem)
```

(introduced in version 1.9)

Hierarchy

TObject
TPersistent
TCollectionItem

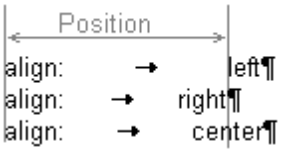
Properties

Position505 – distance of this tab stop from LeftMargin180;
Align504 – alignment of tab relating to the followed text;
Leader505 – text to fill this tabulator.

Defines tab stop alignment (relative to the text after this tab stop)

```
type
  TRVTabAlign = ( rvtaLeft, rvtaRight, rvtaCenter );

property Align: TRVTabAlign;
```

Image:



Alignments are inverted for right-to-left paragraphs100.

Default value:

rvtaLeft

See also:

- Position505.

Repeated text to fill the tabulator space.

```
property Leader: TRVUnicodeString[694];
```

(changed in version 18)

Only the following values of this property can be saved in RTF

Character	Description
.	dot
-	hyphen
_	underscore character
·	#$00B7, middle dot
=	equals sign

Default value:

" (empty string)

Defines distance of this tab stop from LeftMargin[180].

```
property Position: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns the paragraph style with this tab stop.

For right-to-left paragraphs[100], the distance is measured from RightMargin[185].

Collections of tab stops (TRVStyle[418].ParaStyles[431][i].Tabs[489]) are sorted by this property in ascending order. They are resorted automatically when you assign value to this property.

Do not rely on index of item in this collection after assigning this property:

Wrong:

```
ParaStyle.Tabs[489][i].Position := 10;
ParaStyle.Tabs[489][i].Align[504] := rvtaRight; // index of this tab stop may be changed!
```

Correct:

```
ParaStyle.Tabs[489][i].Align[504] := rvtaRight;
ParaStyle.Tabs[489][i].Position := 10;
```

or

```
with ParaStyles.Tabs[489][i] do
begin
  Position := 10;
  Align[504] := rvtaRight;
end;
```

Default value:

0

1.5.2.1.7 Classes of Properties of List Style

1.5.2.1.7.1 TRVListLevelCollection

Collection of list levels. Each level (TRVListLevel[506]) defines properties for bullet or numbering of paragraphs.

This is a class of Levels[493] property of list styles[493].

Unit RVStyle.

Syntax

```
TRVListLevelCollection = class(TCollection)
```

Hierarchy

TObject
TPersistent
TCollection

Main Properties

Items[506] – array of TRVListLevel[506].

Provides indexed access to the items in the collection of paragraph list levels (bullets and numbering)

```
property Items[Index:Integer] : TRVListLevel[506];
```

Each level defines properties for bullet or numbering of paragraphs.

Adds a new style to the collection of list levels.

```
function Add: TRVListLevel[506];
```

This method overrides TCollection.Add method to return TRVListLevel instance.

Call Add to create a new level in the collection. The new item is placed at the end of the Items[506] array.

Add returns the new collection item.

1.5.2.1.7.2 TRVListLevel

One level of paragraph list, defines properties for bullet or numbering. This is a class of item in the Levels[493] collection of list style[493].

Unit RVStyle.

Syntax

```
TRVListLevel = class(TCollectionItem)
```

Hierarchy

TObject
TPersistent
TCollectionItem

Properties

Layout:

- FirstIndent[507], LeftIndent[510] override the corresponding properties of paragraph style;
- MarkerIndent[512] – indent of marker;
- MarkerAlignment[511] – position of marker relative to MarkerIndent.

List marker:

- ListType[510] – type of marker (bullet, number, image, etc.);

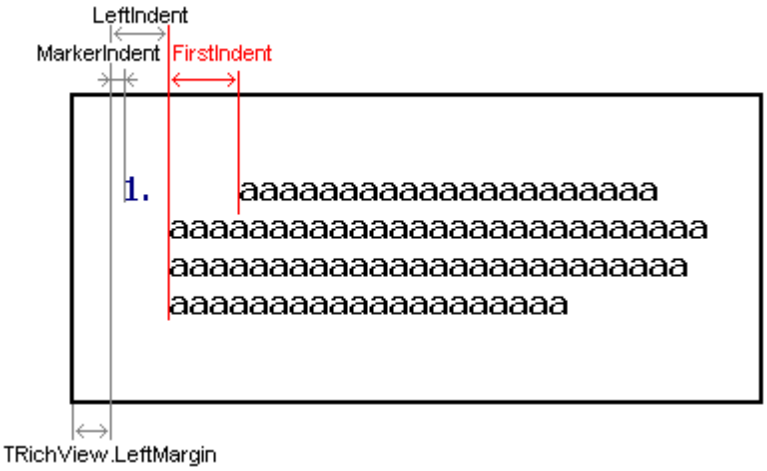
- Font 507 – font for text markers;
- Picture 513 – picture, used if ListType 510 =rvlstPicture;
- ImageList 509, ImageIndex 509 – used if ListType 510 in [rvlstImageList, rvlstImageListCounter];
- FormatString 508 – format string for displaying text markers

Numbering:

- StartFrom 513 – initial counter value for numbered lists;
- Options 512 – numbering options for this list level.

Indent for the first line of paragraph text, relative to LeftIndent 510.

```
property FirstIndent: TRVStyleLength 690;
```



FirstIndent Example

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units 437 of TRVStyle 418 component which owns the list style.

This value can be negative.

This property overrides FirstIndent 485 of paragraph style for bulleted/numbered paragraphs.

If this value is less than width of marker, text is shifted to the right beyond the FirstIndent to avoid overlapping.

Default value:

10

Note: this default value assumes that RVStyle.Units 437=*rvstuPixels*, otherwise it could be too small.

See also properties:

- LeftIndent 510,
- MarkerIndent 512.

See also properties of TCustomRichView:

- LeftMargin 180.

Font for text markers.

```
type
 TRVMarkerFont = class (TFont);

 property Font: TRVMarkerFont;
```

This font is used for the following list types 510:

- *rvlstBullet,*
- *rvlstDecimal,*
- *rvlstLowerAlpha,*

- *rvlstUpperAlpha,*
- *rvlstLowerRoman,*
- *rvlstUpperRoman,*
- *rvlstUnicodeBullet.*

Default value:

'Arial', 8

Defines format for text list markers.

```
property FormatString: TRVUnicodeString[694];
```

This property is used for the following list types[510]:

- *rvlstBullet,*
- *rvlstDecimal,*
- *rvlstLowerAlpha,*
- *rvlstUpperAlpha,*
- *rvlstLowerRoman,*
- *rvlstUpperRoman.*

For *rvlstBullet* list type, it completely defines a text to display.

For *rvlstDecimal..rvlstUpperRoman*, it defines how to display numbers. The actual string = Format(FormatString, [CL0, CL1, ..., CLN])

where

- CL0 - string containing value of counter of the 0-th level marker;
- CL1 - ----------"" ---------- 1st level marker;
- ...
- CLN - string containing value of counter of this level marker.

Example1:

FormatString = '*' // no numbering

```
* aaaaa
* aaaaa
* aaaaa
```

Example2:

For 0-th level: FormatString = '%s.', ListType=*rvlstDecimal*

For 1-th level: FormatString = '%s.%s.', ListType=*rvlstDecimal*

```
1. aaaaa
2. aaaaa
  2.1. aaaaa
  2.2. aaaaa
3. aaaaa
```

The same result will be with '%0:s.' and '%0:s.%1:s.'

Example3:

For 0-th level: FormatString = 'Chapter %s.', ListType=*rvlstDecimal*

For 1-th level: FormatString = '%1:s)', ListType=*rvlstLowerAlpha*

```
Chapter 1. aaaaa
Chapter 2. aaaaa
  a) aaaaa
  b) aaaaa
Chapter 3. aaaaa
```

The same result will be with 'Chapter %0:s.' and '%1:s)'

Default value:

· (#$00B7, middle dot)

Obsolete property. Text for Unicode text markers

```
property FormatStringW: WideString;
```

If ListType[510]=*rvlstUnicodeBullet*, this text is displayed.

Since version 18, this property does not make sense, because FormatString[508] is Unicode as well.

Default value:

" (empty string)

See also properties:

- FormatString[508].

Image index for bullets that show image from the image list

```
property ImageIndex: Integer;
```

If ListType[510]=*rvlstImageList*, the marker of this list level is the ImageIndex-th image from ImageList[509].

If ListType[510]=*rvlstImageListCounter*, the marker of this list level is the (Counter+ImageIndex-1)-th image from ImageList[509], there Counter is a value of counter for the given list marker.

List counter is numbered from StartFrom[513].

For example:

Value of Counter	ImageList Index
1	ImageIndex
2	ImageIndex+1
3	ImageIndex+2
4	ImageIndex+3

Default value:

0

See also properties:

- StartFrom[513].

Image list for image of list marker

```
property ImageList: TCustomImageList;
```

ImageList is used If ListType[510] is *rvlstImageList* or *rvlstImageListCounter*.

When saving lists of this type in RVF[93], RichView stores value of ImageList.Tag. When loading, OnRVFImageListNeeded[288] occurs.

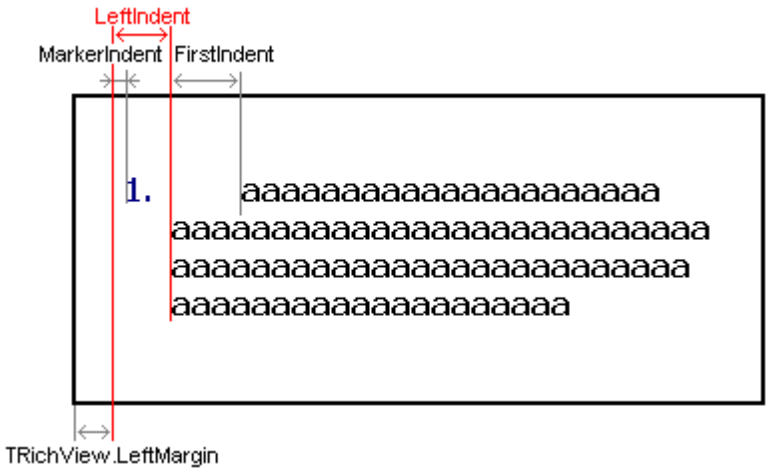
List level does not own this imagelist: it does not create and does not destroy it. It just has a link to it.

See also properties:

- ImageIndex[509];
- Picture[513].

Left indent of paragraph text.

```
property LeftIndent: TRVStyleLength[690];
```



LeftIndent Example

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns the list style.

This value can be negative, but it's not recommended.

This property overrides LeftIndent[485] of paragraph style for bulleted/numbered paragraphs. For right-to-left paragraphs[100], list marker is positioned to the right side of paragraph, and this property overrides RightIndent[488] of paragraph style instead.

Default value:

0

See also properties:

- FirstIndent[507],
- MarkerIndent[512].

See also properties of TCustomRichView:

- LeftMargin[180].

Type of list marker for this list level

```
type
  TRVListType = (rvlstBullet, rvlstPicture,
    rvlstImageList, rvlstDecimal,
    rvlstLowerAlpha, rvlstUpperAlpha,
    rvlstLowerRoman, rvlstUpperRoman,
    rvlstImageListCounter,rvlstUnicodeBullet);

property ListType: TRVListType;
```

Unnumbered List Types

List Type	Meaning
rvlstBullet	Unnumbered text (from FormatString[508] property). See also Font[507].
rvlstPicture	Picture (from Picture[513] property)
rvlstImageList	Image from an image list (ImageList[509] and ImageIndex[509] properties)

Numbered List Types

List Type	Meaning
rvlstDecimal	1, 2, 3,... (see FormatString[508], Font[507])
rvlstLowerAlpha	a, b, c, ..., z, aa, ab, ... (see FormatString[508], Font[507])
rvlstUpperAlpha	A, B, C, ..., Z, AA, AB, ... (see FormatString[508], Font[507])
rvlstLowerRoman	i, ii, iii, iv, ... (see FormatString[508], Font[507])
rvlstUpperRoman	I, II, III, IV, ... (see FormatString[508], Font[507])
rvlstImageListCounter	Image from an image list, depending on a counter (ImageList[509] and ImageIndex[509] properties)

Obsolete List Types

List Type	Meaning
rvlstUnicodeBullet	Unnumbered Unicode text (from FormatStringW[509] property).

Default value:

rvlstBullet

Defines marker alignment relative to the MarkerIndent[512]

```
type
  TRVMarkerAlignment = (rvmaLeft, rvmaRight, rvmaCenter);

property MarkerAlignment: TRVMarkerAlignment;
```



MarkerAlignment Example

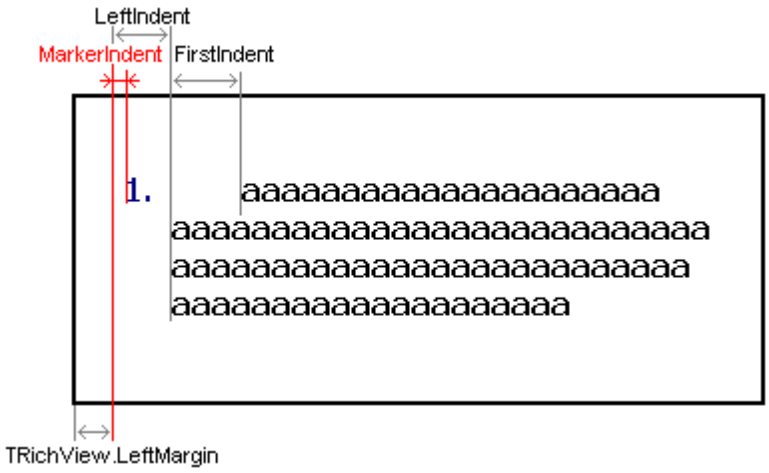
For right-to-left paragraphs[100], list marker position is reversed.

Default value:

rvmaLeft

Indent of the list marker

```
property MarkerIndent: TRVStyleLength[690];
```



MarkerIndent Example

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of TRVStyle[418] component which owns the list style.

This value can be negative, but it's not recommended.

The image above shows the position of left-aligned marker. See MarkerAlignment[511] property.

Default value:

0

See also properties:

- FirstIndent[507];
- LeftIndent[510].

See also properties of TCustomRichView:

- LeftMargin[180].

List level options

```
type
  TRVListLevelOption =
    (rvloContinuous, rvloLevelReset, rvloLegalStyleNumbering);
  TRVListLevelOptions = set of TRVListLevelOption;
```

If *rvloLevelReset* is in Options, numbering will be like this:

```
  1. aaaa
    a) aaaa
    b) aaaa
  2. aaaa
    a) aaaa
    b) aaaa
```

If it is excluded, numbering will be like this:

```
  1. aaaa
    a) aaaa
    b) aaaa
  2. aaaa
    c) aaaa
    d) aaaa
```

If *rvloLegalStyleNumbering* is in Options, all roman and alpha numbering of higher levels will be changed to decimals, like this:

```
I. aaaa
  1.1 aaaa
  1.2 aaaa
II. aaa
```

Otherwise:

```
I. aaaa
  I.1 aaaa
  I.2 aaaa
II. aaa
```

rvloContinuous is reserved for future use.

Default value:

[*rvloContinuous*, *rvloLevelReset*]

Picture for list marker

```
property Picture: TPicture;
```

This property is used if ListType[510]=*rvlstPicture*.

Starting value for marker counter

```
property StartFrom: Integer;
```

For example, if StartFrom for Levels[493][1]=0:

```
1. aaaaa
  0) aaaaa
  1) aaaaa
2. aaaaa
```

Default value:

1

See also:

- ListType[510].

Returns *True* if this list level is numbered

```
function HasNumbering: Boolean;
```

This method returns

Result := ListType[510] in [*rvlstDecimal, rvlstLowerAlpha, rvlstUpperAlpha*,

rvlstLowerRoman, rvlstUpperRoman, rvlstImageListCounter].

1.5.2.2 TRVPrint

This component allows printing TRichView[162], TRichViewEdit[316], TDBRichView[407], or TDBRichViewEdit[410]. It is Invisible at run-time.

Unit PtblRV;

Syntax

```
TRVPrint = class(TCustomRVPrint[549])
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVPrint[549]

How to Use

1. Assign the source component for printing (AssignSource[522] method).
2. (optionally) Assign headers and/or footers (SetHeader[529], SetFooter[528]).
3. Format pages (FormatPages[525] method).
4. Now you can:

- get a number of pages (PagesCount[552] property);
- display a print preview (using TRVPrintPreview[417] component; see also DrawPreview[524], MakePreview[526], MakeScaledPreview[526], DrawPage, DrawPageAt[524] methods);
- print the whole document (Print[526] method) or range of pages (PrintPages[527] method).

5. When finished, you can release some temporally allocated memory (Clear[553]).

Printer

By default, the component prints on the default printer, using its default properties. The global Printer object provides access to the available printers. You can switch the printer by assigning Printer.PrinterIndex. You can change orientation of pages by assigning Printer.Orientation. After these changes, before printing or previewing, call FormatPages[525].

Alternatively, you can use "virtual printer" to draw pages on any canvas. Assign VirtualPrinter[522].Active[532] = *True* to use parameters (page size and DPI) specified in VirtualPrinter[522], instead of parameters of the current printer.

Events

You can display some information about the process of paginating and printing:

- OnFormatting[529] occurs during paginating (FormatPages[525] method);
- OnSendingToPrinter[531] occurs during printing (Print[526] and PrintPages[527] methods).

You can draw additional content using the following events:

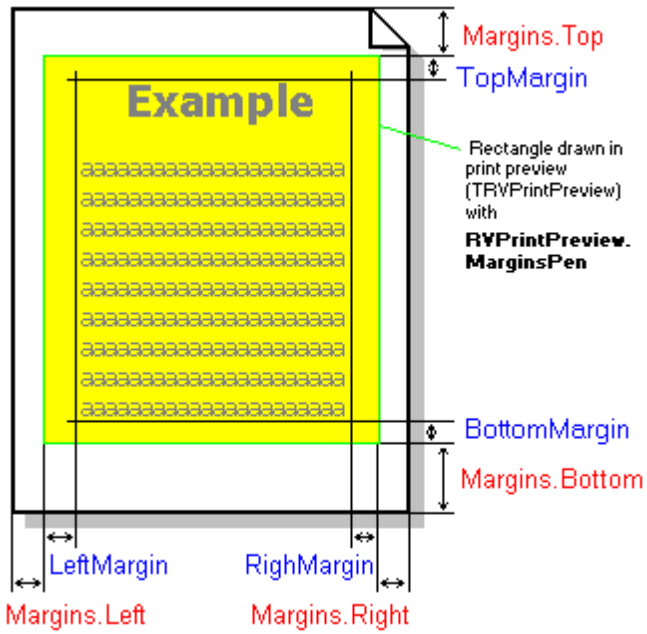
- OnPagePrepaint[530] and OnPagePostpaint[530] allow drawing on each page below/over TRichView document;
- OnPrintComponent[554] requests an image of an inserted control for printing.

There are also informational events; they occurred when some special item is drawn. You can store coordinates of these items and implement special processing (for example, adding hyperlinks when printing to PDF using some third-party library).

- OnDrawCheckpoint[556]
- OnDrawHyperlink[556]

Margins

Margins are specified in Margins[520] property.



Margins in TRVPrint

This picture illustrates the meaning of margin properties. Paragraph indents are not shown for simplification.

Margins[520] are measured in Units[521] (shown in red).

TRichView.LeftMargin[180], TRichView.RightMargin[185], TRichView.TopMargin[194] and TRichView.BottomMargin[171] are measured in "screen pixels" (they are recalculated according to the printer resolution, and margins on the page look like margins on the screen; shown in blue).

Left and right margins can be swapped for even pages, if MirrorMargins[521] is set to *True*.

See also ClipMargins[518] property.

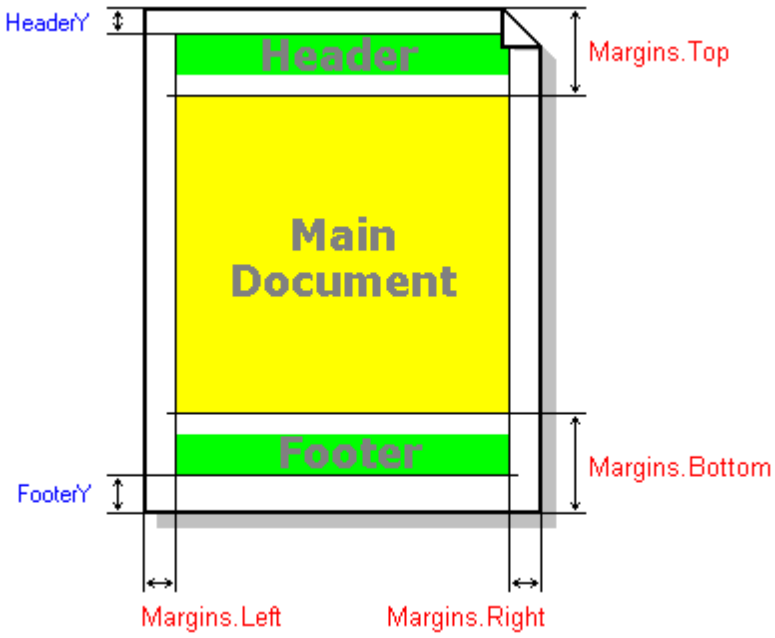
Headers and Footers

TRVPrint supports 3 types of headers and footers:

- normal header and footer
- header and footer for the first page (used only if TitlePage[521]=*True*)
- header and footer for even pages (used only if FacingPages[518]=*True*)

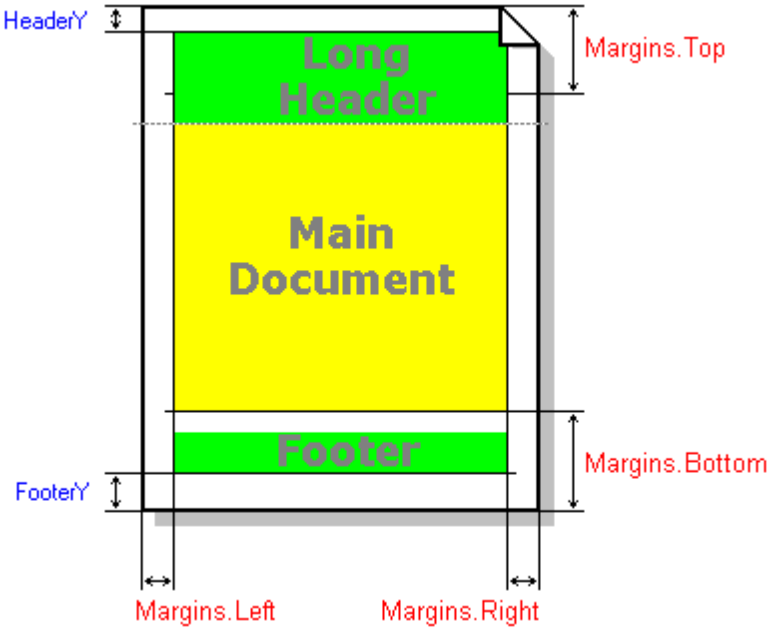
Headers are assigned using SetHeader[529], footers are assigned using SetFooter[528] methods.

Any TRichView document can be assigned as a header or footer. Positions of headers and footers are defined by HeaderY[519] and FooterY[519].



Header and Footer

If a header and/or a footer is too long, size of the main document is adjusted to avoid overlapping:



Long Header Example

If you want more flexible headers / footers, consider:

- printing document in several parts, each of them with its own header/footer;
- using OnPagePrepaint[530] event;
- using TRVReportHelper[540].

See also: GetHeaderRect[525], GetFooterRect[525].

Compatibility options

Some printers may have problems with advanced content (especially when you print to create PDF or metafile files). In this case, you can use the properties:

- MetafileCompatibility[551];
- NoMetafiles[551].

Document inside RVPrint

RVPrint contains an instance of TPrintableRV (descendant of TRichView) inside. This variable (named rv) is public and you can change some properties of it:

- Color[173],
- BackgroundStyle[170],
- BackgroundBitmap[169].

Changing other its properties is not recommended.

Example:

```
MyRVPrint.rv.Color := clWhite;
```

Note: this richview does not contain the full information about the printed document, just a formatting information; so you must provide that the source RichView is not destroyed while you are printing.

Incremental printing

You can define the range of items to print, using MinPrintedItemNo[551] and MaxPrintedItemNo[550] properties. If these properties define a sub-range of items, only this sub-range (and only pages containing it) is printed.

These properties allow implementing an incremental printing: when the next portion of the document is ready, it can be printed below the previously printed fragment. This feature can be used for printing accounting journals, logs, etc.

Order of drawing

The component draws a page content in the following order:

1. page background
2. OnPagePrepaint[530] event
3. Header's text boxes[139] and sidenotes[137] having PositionInText[640]=*rvpitBelowText*
4. Header[529]
5. Header's text boxes and sidenotes having PositionInText=*rvpitAboveText*
6. Footer's text boxes and sidenotes having PositionInText=*rvpitBelowText*
7. Footer[529]
8. Footer's text boxes and sidenotes having PositionInText=*rvpitAboveText*
9. Main document's text boxes and sidenotes having PositionInText=*rvpitBelowText*
10. Main document, including footnotes and endnotes
11. Main document's text boxes and sidenotes having PositionInText=*rvpitAboveText*
12. OnPagePostPaint[530] event

Text boxes and sidenotes are drawn in the order of their positions in the parent document.

Tips and Tricks

See ***http://www.trichview.com/forums/viewtopic.php?p=250***.

See Also

See also:

- UpdatePaletteInfo[554] method;
- TRichView.PageBreaksBeforeItems[184];
- TRichView.AssignSoftPageBreaks[209];
- TRVReportHelper[540] components;
- RichViewAlternativePicPrint[700] typed constant;
- RichViewPixelsPerInch[703] typed constant.

Demo projects:

- Demos\Delphi\Assorted\Printing\
- Demos\CBuilder\Assorted\Printing\

1.5.2.2.1 Properties

1.5.2.2.1.1 TRVPrint.ClipMargins

Disallows printing on margins.

```
property ClipMargins: Boolean
```

(introduced in version 1.7)

If *True*, all drawing that does not fit the printable area (for example, large pictures) is cropped.

Default value:

False

1.5.2.2.1.2 TRVPrint.FacingPages

Enables using different headers and footers on odd and even pages.

```
property FacingPages: Boolean;
```

(introduced in version 15)

If *False*, normal headers and footers are used on all pages.

If *True*, normal headers and footers are used on odd pages, special headers and footers are used on even pages.

If some header or footer is not defined (*nil*), it is not shown.

Default value:

False

See also properties:

- TitlePage[521];
- MirrorMargins[521].

See also methods:

- SetHeader[529];
- SetFooter[528];
- AssignDocParameters[522].

1.5.2.2.1.3 TRVPrint.FixMarginsMode

Specifies how to work with too small Margins[520] (less than the printer supports).

```
type
  TRVFixMarginsMode = (rvfmmAutoCorrect, rvfmmIgnore);
property FixMarginsMode: TRVFixMarginsMode;
```

(introduced in version 10)

Mode	Meaning
rvfmmAutoCorrect	Minimal supported margins are used (if the left and/or the top margin were too small, the output is moved to the right and/or down)
rvfmmIgnore	The specified values of margins are used (document may be cropped)

Default value:

rvfmmAutoCorrect

1.5.2.2.1.4 TRVPrint.FooterY

Vertical position of a page footer.

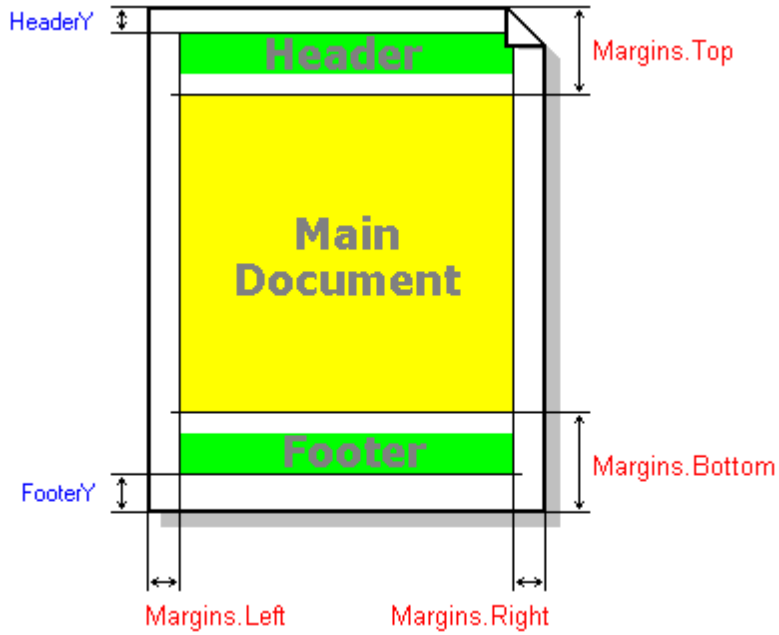
```
property FooterY: TRVLength[682];
```

(introduced in version 15)

This is a distance from the bottom of a page to the bottom of a footer. This value is measured in Units[521].

If a footer is not assigned, this property is ignored.

This property is used in all types of headers (normal, first page, even pages).



Header and Footer

Default value:

10.0

See also:

- HeaderY[519].

See also methods:

- SetFooter[528];
- AssignDocParameters[522].

1.5.2.2.1.5 TRVPrint.HeaderY

Vertical position of a page header.

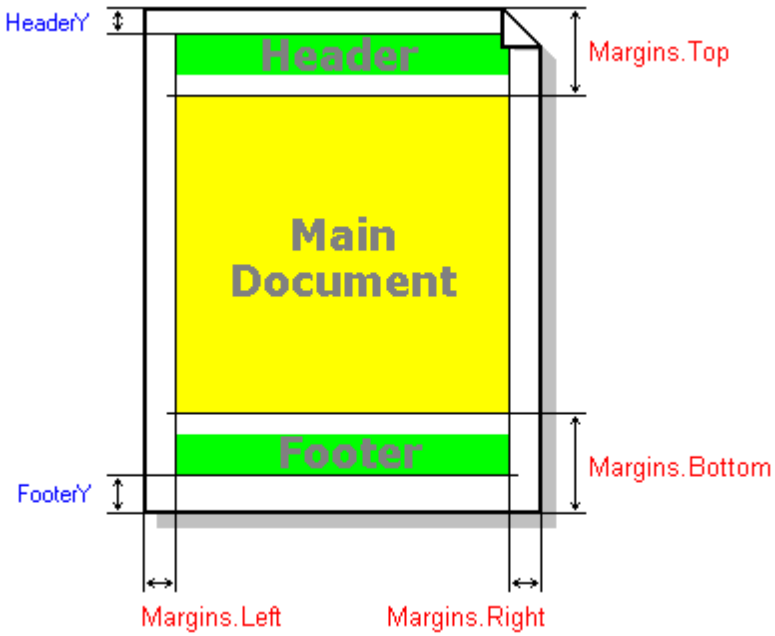
```
property HeaderY: TRVLength[682];
```

(introduced in version 15)

This is a distance from the top of a page to the top of a header. This value is measured in Units[521].

If a header is not assigned, this property is ignored.

This property is used in all types of headers (normal, first page, even pages).



Header and Footer

Default value:

10.0

See also:

- FooterY [519].

See also methods:

- SetHeader [529];
- AssignDocParameters [522].

1.5.2.2.1.6 TRVPrint.Margins

Page margins.

```
property Margins: TRVUnitsRect [660];
```

(introduced in version 15)

Margins are measured in Units [437].

If MirrorMargins [521]=*True*, Margins.Left and Margins.Right are swapped for even pages.

You need to reformat pages after changing values of this property, see FormatPages [525].

See also the picture in the TRVPrint [513] main topic.

Default value:

(20.0, 20.0, 20.0, 20.0)

See also properties:

- HeaderY [519];
- FooterY [519];

See also methods:

- AssignDocParameters [522].

1.5.2.2.1.7 TRVPrint.MirrorMargins

If set to *True*, Margins[520].Left and Margins[520].Right are swapped for even pages.

```
property MirrorMargins: Boolean;
```

(introduced in version 1.6)

Default value:

False

See also properties:

- FacingPages[518].

See also methods:

- AssignDocParameters[522].

1.5.2.2.1.8 TRVPrint.Preview100PercentHeight

Height of a page when drawing a preview on the screen at 100% scale, pixels.

```
property Preview100PercentHeight: Integer;
```

Mainly for internal use.

1.5.2.2.1.9 TRVPrint.Preview100PercentWidth

Width of a page when drawing a preview on the screen at 100% scale, pixels.

```
property Preview100PercentWidth: Integer;
```

Mainly for internal use.

1.5.2.2.1.10 TRVPrint.TitlePage

Enables using different headers on the first page.

```
property FacingPages: Boolean;
```

(introduced in version 15)

If *False*, normal headers and footers are used on the first page.

If *True*, special headers and footers are used on the first pages.

If some header or footer is not defined (*nil*), it is not shown.

Default value:

False

See also properties:

- FacingPages[518].

See also methods:

- SetHeader[529];
- SetFooter[528];
- AssignDocParameters[522].

1.5.2.2.1.11 TRVPrint.Units

Defines measurement units for sizes in TRVPrint.

```
property Units: TRVUnits[694];
```

(introduced in version 15)

The following properties are measured in Units:

- HeaderY[519];

- FooterY[519];
- Margins[520].

A direct assignment to this property does not change values of these properties. To convert properties, use ConvertToUnits[523] method.

Default value:

rvuMillimeters

1.5.2.2.1.12 TRVPrint.VirtualPrinter

Defines properties of a "virtual printer"

```
property VirtualPrinter: TRVVirtualPrinterProperties[532];
```

(introduced in version 17)

If **VirtualPrinter**.Active[532] = *True*, TRVPrint prepares pages for drawing on any canvas, using page size[532] and DPI value[532] specified in this property. Real printers are not used in this mode.

In this mode, you cannot call methods that print multiple pages, but you can draw pages on the specified canvas using DrawPage and DrawPageAt[524] methods.

"Virtual printer" functionality is similar to the functionality provided by TRVReportHelper[540] component. Differences:

- a report helper draws only a directly at the specified location; a virtual printer draws a page having the specified margins;
- a report helper does not support headers[529] and footers[528];
- a report helper does not support printing text boxes[139] and side notes[137];
- a report helper uses DPI value read from the specified Canvas; a virtual printer uses DPI specified in its property.

1.5.2.2.2 Methods

1.5.2.2.2.1 TRVPrint.AssignDocParameters

Assigns properties from **DocParameters**.

```
procedure AssignDocParameters(DocParameters: TRVDocParameters[310]);
```

(introduced in version 10)

This procedure assigns the following properties:

- Margins[520];
- HeaderY[519], FooterY[519];
- MirrorMargins[521];
- FacingPages[518];
- TitlePage[521].

Sizes are converted from **DocParameters**.Units[314] to Units[521] (the value of Units[521] property itself is not changed).

This method does not assign printer properties (page size and orientation).

1.5.2.2.2.2 TRVPrint.AssignSource

Assigns document for printing.

```
procedure AssignSource(PrintMe: TCustomRichView[162]);
```

You can print document in the following components:

- TRichView[162],
- TRichViewEdit[316],
- TDBRichView[407],
- TDBRichViewEdit[410].

Document is not copied to RVPrint, so do not modify and do not destroy it until printing is completed.

See also:

- SetHeader[529];
- SetFooter[528].

1.5.2.2.2.3 TRVPrint.ContinuousPrint

Prints the whole document in the current printing job.

```
procedure ContinuousPrint;
```

This method is useful if you need to print several documents in one printing job. Otherwise, use Print[526] or PrintPages[527] methods.

Do not call this method if VirtualPrinter[522].Active[532] = *True*

Example 1: How to print several documents in one printing job

In this example, the same RVPrint is used to print RichView1 and RichView2. This code starts printing RichView2 on the page where RichView1 ends.

```
Printer.BeginDoc;
// printing RichView1
RVPrint.StartAt[552] := 0;
RVPrint.TransparentBackground[553] := True;
RVPrint.AssignSource[522](RichView1);
RVPrint.FormatPages[525](rvdoALL);
RVPrint.ContinuousPrint;
// printing RichView2
RVPrint.StartAt[552] := RVPrint.EndAt[549];
RVPrint.AssignSource[522](RichView2);
RVPrint.FormatPages[525](rvdoALL);
RVPrint.ContinuousPrint;
Printer.EndDoc;
```

Example 2: How to mix portrait and landscape orientations in one printing job

In this example, the same RVPrint is used to print RichView1 and RichView2 in different orientations.

```
Printer.BeginDoc;
// printing RichView1 in portrait orientation
Printer.Orientation := poPortrait;
RVPrint.AssignSource[522](RichView1);
RVPrint.FormatPages[525](rvdoALL);
RVPrint.ContinuousPrint;
// printing RichView2 in landscape orientation
Printer.NewPage;
Printer.Orientation := poLandscape;
RVPrint.AssignSource[522](RichView2);
RVPrint.FormatPages[525](rvdoALL);
RVPrint.ContinuousPrint;
Printer.EndDoc;
```

1.5.2.2.2.4 TRVPrint.ConvertToUnits

Converts properties to new units.

```
procedure ConvertToUnits(AUnits: TRVUnits[694]);
```

(introduced in version 15)

The method converts HeaderY[519], FooterY[519], Margins[520] from Units[521] to **AUnits**, then assigns **AUnits** to Units[521].

1.5.2.2.2.5 TRVPrint.Create

A constructor, creates and initializes a TRVPrint instance.

```
constructor Create(AOwner: TComponent);override;
```

Use Create to instantiate TRVPrint object at runtime. TRVPrint objects placed on forms at design time are created automatically. Specify the owner of the new TRVPrint object using the **AOwner** parameter.

1.5.2.2.2.6 TRVPrint.Draw Preview

Draws print preview of the **pgNo**-th page to **PageRect** rectangle of **Canvas**.

```
procedure DrawPreview(pgNo: Integer; Canvas:  TCanvas; const PageRect: TRect);
```

There is more convenient way to display print preview: TRVPrintPreview[417] component.

The page image is scaled to fit **PageRect**. To maintain correct proportions, you can use Preview100PercentWidth[521] and Preview100PercentHeight[521] properties.

pgNo must be in range from 1 to PagesCount[552].

You need to assign source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

To draw unscaled page, use DrawPage and DrawPageAt[524] methods.

See also methods:

- MakePreview[526];
- MakeScaledPreview[526];
- DrawPage, DrawPageAt[524].

See also properties:

- PreviewCorrection[552];
- PagesCount[552].

See also events:

- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.7 TRVPrint.Draw Page, Draw PageAt

Draws the **APageNo**-th page on **ACanvas**.

```
procedure DrawPage(APageNo: Integer; ACanvas: TCanvas; APreview: Boolean);
procedure DrawPageAt(Left, Top, APageNo: Integer; ACanvas: TCanvas; APreview: Boolean);
```

(introduced in version 17)

DrawPage draws the page at (0, 0) coordinates.

DrawPageAt draws the page at (**Left**, **Top**) coordinates.

APageNo must be in range from 1 to PagesCount[552].

You need to assign source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

See also methods:

- DrawPreview[524].

See also properties:

- PreviewCorrection[552];
- PagesCount[552].

See also events:

- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.8 TRVPrint.FormatPages

Prepare document to formatting

```
function FormatPages(PrintOptions:TRVDisplayOptions[671]): Integer;
```

Please use *rvdoAll* constant for **PrintOptions** parameter:

```
FormatPages(rvdoAll);
```

Document must be assigned to this TRVPrint object using AssignSource[522] method before calling this method.

This method allocates some temporal memory what is used when drawing preview and printing. When you finish, call Clear[553] to release this memory.

Note for C++Builder users:

C++Builder (at least version 5) incorrectly translates rvdoAll constant.

The recommended call is

```
RVPrint->FormatPages(TRVDisplayOptions())
```

(i.e. initialization with the empty set; this parameter is for compatibility, and it is not used any more)

See also methods:

- Clear[553].

See also events:

- OnFormatting[529].

1.5.2.2.2.9 TRVPrint.GetFooterRect

Returns coordinates of the footer for the page **PageNo** (from 1), in printer pixels

```
function GetFooterRect(PageNo: Integer): TRect;
```

(introduced in version 1.7; the parameter is added in version 15)

If the footer is empty, the function returns an empty rectangle (Rect(0,0,0,0)).

This function may be called only when the document is prepared for printing (see FormatPages[525]).

This function can be used in OnPagePrepaint[530] and OnPagePostpaint[530] events.

See also:

- SetFooter[528];
- GetHeaderRect[525].

1.5.2.2.2.10 TRVPrint.GetHeaderRect

Returns coordinates of the header for the page **PageNo** (from 1), in printer pixels

```
function GetHeaderRect(PageNo: Integer): TRect;
```

(introduced in version 1.7; the parameter is added in version 15)

If the header is empty, the function returns an empty rectangle (Rect(0,0,0,0)).

This function may be called only when the document is prepared for printing (see FormatPages[525])

This function can be used in OnPagePrepaint[530] and OnPagePostpaint[530] events.

See also:

- SetHeader[529];
- GetFooterRect[525].

1.5.2.2.2.11 TRVPrint.MakePreview

Draws a preview of the **pgNo**-th page to bitmap **bmp**.

```
procedure MakePreview(pgNo: Integer; bmp: TBitmap);
```

There is more convenient way to display print preview: TRVPrintPreview[417] component.

pgNo must be in range from 1 to PagesCount[552].

This method changes size of bitmap to (Preview100PercentWidth[521] x Preview100PercentHeight[521]) and draws print preview of the **pgNo**-th page into it.

You need to assign source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

See also methods:

- MakeScaledPreview[526];
- DrawPreview[524].

See also properties:

- PreviewCorrection[552];
- PagesCount[552].

See also events:

- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.12 TRVPrint.MakeScaledPreview

Draws print preview of the **phNo**-th page to bitmap **bmp**.

```
procedure MakeScaledPreview(pgNo: Integer; bmp: TBitmap);
```

There is more convenient way to display print preview: TRVPrintPreview[417] component.

pgNo must be in range from 1 to PagesCount[552].

This method does not change size of bitmap, and draws the page in full bitmap size. To maintain correct proportions, you can use Preview100PercentWidth[521] and Preview100PercentHeight[521] properties.

You need to assign source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

See also methods:

- MakePreview[526];
- DrawPreview[524].

See also properties:

- PreviewCorrection[552];
- PagesCount[552].

See also events:

- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.13 TRVPrint.Print

Prints the whole document

```
procedure Print(Title: String; Copies: Integer; Collate: Boolean;
  PageSet: TRVPageSet[683] = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo[551] and MaxPrintedItemNo[550] properties into account.

Parameters

Title – the text that is listed in the Print Manager when printing.

Copies – specifies the number of copies to print.

Collate – if more than one copy is selected, specifies whether you want the copies to be collated (*True*: 1, 2, 3, 1, 2, 3; *False*: 1, 1, 2, 2, 3, 3).

PageSet – set of pages for printing (all/odd/even).

Ascending switches normal/reverse order of printing (*True*: 1, 2, 3; *False*: 3, 2, 1).

Assign the source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

```
Print(Title, Copies, Collate, PageSet, Ascending)
```

is equivalent to

```
PrintPages(1, PagesCount, Title, Copies, Collate, PageSet, Ascending)
```

Do not call this method if VirtualPrinter[522].Active[532] = *True.*

How to print on two sides of paper:

1. Print odd pages in normal order.
2. Flip the sheets over, then reinsert them into the printer. If PagesCount[552] is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- PrintPages[527];

See also events:

- OnSendingToPrinter[531];
- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.14 TRVPrint.PrintPages

Prints the range of pages, from **FirstPgNo** to **LastPgNo**

```
procedure PrintPages(FirstPgNo, LastPgNo: Integer; Title: String;
  Copies: Integer; Collate: Boolean;
  PageSet: TRVPageSet[683] = rvpsAll; Ascending: Boolean = True);
```

(new parameters are added in version 17)

This method takes MinPrintedItemNo[551] and MaxPrintedItemNo[550] properties into account.

Parameters

FirstPgNo – the first page to print, must be in range 1..PagesCount[552].

LastPgNo – the last page to print, must be in range 1..PagesCount[552].

Title – the text that is listed in the Print Manager when printing.

Copies – specifies the number of copies to print.

Collate – if more than one copy is selected, specifies whether you want the copies to be collated (*True*: 1, 2, 3, 1, 2, 3; *False*: 1, 1, 2, 2, 3, 3).

PageSet – set of pages for printing (all/odd/even).

Ascending switches normal/reverse order of printing (*True*: 1, 2, 3; *False*: 3, 2, 1).

You need to assign source document for printing (AssignSource[522] method) and format pages (FormatPages[525] method) before calling this method.

Do not call this method if VirtualPrinter[522].Active[532] = *True*.

How to print on two sides of paper:

1. Print odd pages in normal order.
2. Flip the sheets over, then reinsert them into the printer. If PagesCount[552] is odd, do not insert the last page.
3. Print even pages in reverse order.

See also methods:

- Print[526].

See also events:

- OnSendingToPrinter[531];
- OnPrintComponent[554];
- OnPagePrepaint[530];
- OnPagePostpaint[530].

1.5.2.2.2.15 TRVPrint.SetFooter

Assigns a page footer.

```
procedure SetFooter(RVData: TCustomRVData[650]; FooterType: TRVHFType[680]=rvhftNormal);
```

(introduced in version 1.7; modified in version 15)

Parameters:

RVData – document to display as a footer of the type specified in **HFType**; the document is not copied to RVPrint, so do not modify and do not destroy it until printing is completed. If **RVData** is *nil*, a footer of this type is not displayed.

HFType – footer type (normal/for the first page/for even pages).

This method must be called before FormatPages[525].

Using different footers

A footer for the first page is used only if TitlePage[521]=*True*, otherwise it is ignored, and a normal footer is displayed on the first page.

A special footer for even pages is used only if FacingPages[518]=*True*, otherwise it is ignored, and a normal footer is used on both odd and even pages.

Example:

```
var
  rvMain, rvFooter: TRichView[162];
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetFooter(rvFooter.RVData[187]);
RVPrint1.TitlePage[521] := True;
RVPrint1.SetFooter(nil, rvhftFirstPage); // hiding footer on the first page
```

See also:

- SetHeader[529];
- AssignSource[522].

See also properties:

- FooterY[519].

See also:

- TRichView[162].SetFooter[263].

1.5.2.2.2.16 TRVPrint.SetHeader

Assigns a page header.

```
procedure SetHeader(RVData: TCustomRVData[650]; FooterType: TRVHFType[680]=rvhftNormal);
```

(introduced in version 1.7; modified in version 15)

Parameters:

RVData – document to display as a header of the type specified in **HFType**; the document is not copied to RVPrint, so do not modify and do not destroy it until printing is completed. If **RVData** is *nil*, a header of this type is not displayed.

HFType – header type (normal/for the first page/for even pages).

This method must be called before FormatPages[525].

Using different headers

A header for the first page is used only if TitlePage[521]=*True*, otherwise it is ignored, and a normal header is displayed on the first page.

A special header for even pages is used only if FacingPages[518]=*True*, otherwise it is ignored, and a normal header is used on both odd and even pages.

Example:

```
var
  rvMain, rvHeader: TRichView[162];
  RVPrint1: TRVPrint;
...
RVPrint1.AssignSource(rvMain);
RVPrint1.SetHeader(rvHeader.RVData[187]);
RVPrint1.TitlePage[521] := True;
RVPrint1.SetHeader(nil, rvhftFirstPage); // hiding header on the first page
```

See also:

- SetFooter[528];
- AssignSource[522].

See also properties:

- HeaderY[519].

See also:

- TRichView[162].SetHeader[264].

1.5.2.2.3 Events

1.5.2.2.3.1 TRVPrint.OnFormatting

Occurs while FormatPages[525] method is executed.

```
type
  TRVPrintingEvent = procedure(Sender: TCustomRichView[162];
    PageCompleted: Integer;
    Step:TRVPrintingStep[684]) of object;

property OnFormatting: TRVPrintingEvent;
```

In the first time, this event is generated with

PageCompleted=0, **Step**=*rvpsStarting*.

Next, this event occurs several times with

PageCompleted=<number of page which was formatted>, **Step**=*rvpsProceeding*.

In the last time, it occurs with

PageCompleted=0, **Step**=*rvpsFinished.*

Note1: Sender is a special RichView, internally contained in TRVPrint object.

Note2: The first step (preliminary formatting) takes 90% of time; subsequent repagination is quick.

See also methods:

- FormatPages 525.

See also events:

- OnSendingToPrinter 531.

1.5.2.2.3.2 TRVPrint.OnPagePostpaint

Allows to draw on a page (paper and print preview)

```
type
  TRVPagePrepaintEvent = procedure (Sender: TRVPrint;
    PageNo: Integer; Canvas: TCanvas; Preview: Boolean;
    PageRect, PrintAreaRect: TRect) of object;

property OnPagePostpaint: TRVPagePrepaintEvent;
```

(introduced in version 1.7)

Sequence of actions:

1. drawing background (if not transparent);
2. OnPagePrepaint 530;
3. drawing document;
4. OnPagePostpaint.

Parameters

PageNo – page number (starting from 1);

Canvas – canvas where to draw;

Preview – "is this a preview drawing?"

PageRect – rectangle around the full page, pixels.

PrintAreaRect – rectangle around the document on the page (**PageRect** minus marginsMM), pixels.

See also:

- GetHeaderRect 525;
- GetFooterRect 525;
- the picture in the TRVPrint 513 main topic.

1.5.2.2.3.3 TRVPrint.OnPagePrepaint

Allows to draw on a page (paper and print preview)

```
type
  TRVPagePrepaintEvent = procedure (Sender: TRVPrint;
    PageNo: Integer; Canvas: TCanvas; Preview: Boolean;
    PageRect, PrintAreaRect: TRect) of object;

property OnPagePrepaint: TRVPagePrepaintEvent;
```

(introduced in version 1.3)

Sequence of actions:

1. drawing background (if not transparent);
2. OnPagePrepaint;
3. drawing document;
4. OnPagePostpaint[530].

Parameters

PageNo – page number (starting from 1);

Canvas – canvas where to draw;

Preview – "is this a preview drawing?"

PageRect – rectangle around the full page, pixels.

PrintAreaRect – rectangle around the document on the page (**PageRect** minus marginsMM), pixels.

See also:

- GetHeaderRect[525];
- GetFooterRect[525];
- the picture in the TRVPrint[513] main topic;
- example how to print pictures in this event: ***http://www.trichview.com/forums/viewtopic.php?t=58***.

1.5.2.2.3.4 TRVPrint.OnSendingToPrinter

Occurs while methods Print[526] or PrintPages[527] are executed.

```
type
  TRVPrintingEvent = procedure(Sender: TCustomRichView[162];
    PageCompleted: Integer; Step:TRVPrintingStep[684]) of object;

property OnSendingToPrinter: TRVPrintingEvent;
```

In the first time, this event is generated with

PageCompleted=0, **Step**=*rvpsStarting*.

Next, this event occurs several times with

PageCompleted=<number of page that was just printed>, **Step**=*rvpsProceeding*.

In the last time, it occurs with

PageCompleted=0, **Step**=*rvpsFinished.*

Note1: Sender is a special RichView, internally contained in TRVPrint object.

Note2: PageCompleted is an index of page (from 1), not a number of printed pages. The same value will be repeated several times in one call of Print or PrintPages when printing more than one copy.

See also methods:

- Print[526];
- PrintPages[527].

See also events:

- OnFormatting[529].

1.5.2.2.4 Classes of Properties

1.5.2.2.4.1 TRVVirtualPrinterProperties

This is a class of VirtualPrinter[522] property of TRVPrint[513].

Unit RVStyle.

Syntax

```
TRVVirtualPrinterProperties = class(TPersistent)
```

Hierarchy

TObject
TPersistent

Main Properties

- Active[532] enables/disables a virtual printer
- PageWidth, PageHeight[532]
- PixelsPerInch[532]

Enables the virtual printer.

```
property Active: Boolean;
```

TRVPrint[513] uses the virtual printer only if **Active** = *True*. Otherwise, TRVPrint prints on the current printer (specified in the global Printer object).

Default value

False

The number of screen pixels for an virtual inch (DPI)

```
property PixelsPerInch: Boolean;
```

If this value is positive, it is used as a DPI of the virtual printer.

Otherwise, if RichViewPixelsPerInch[703] is positive, it is used as DPI.

Otherwise, the actual screen DPI is used (Screen.PixelsPerInch).

Default value

0

Page size of the virtual printer, in pixels

```
property PageWidth: Integer;
property PageHeight: Integer;
```

Default values

- PageWidth: 794
- PageHeight: 1123

(these values correspond to A4 paper format in 96 dpi)

1.5.2.3 TRVOfficeConverter

TRVOfficeConverter allows using text converters from Microsoft Office (32-bit versions) with RichViews, for importing and exporting files in different formats.

Unit RVOfficeCnv;

Syntax

```
TRVOfficeConverter = class (TComponent)
```

(introduced in version 1.6)

Hierarchy

TObject
TPersistent
TComponent

How to Use

There are two types of converters: for importing and for exporting files.

Export converters convert RTF (Rich Text Format) to other formats, import converters convert different formats to RTF.

Lists of available converters are in ImportConverters[535] and ExportConverters[535] properties.

If there are no converters installed, converters lists are empty.

Note: some converters are shipped with Windows.

GetImportFilter[537] and GetExportFilter[537] return strings for assigning to Filter property of file-selection dialogs.

There are two modes for using this components.

1. You can convert RTF from Stream[536] to file in another format (ExportRTF[536]), or you can convert file to RTF Stream (ImportRTF[538]).
2. You can export content of TRichView component in a file (ExportRV[537]) or import from file (ImportRV[538]). Such conversion still uses Stream[536] internally.

While converting, the component generates OnConverting[539] event.

If PreviewMode[535] property is set to *True*, converters do their work silently, without displaying any dialogs. They can also produce lower quality results in this mode (that depends on a converter).

Platform notes

This component is useful only in 32-bit applications, because converters are 32-bit DLLs.

Demo projects

- Demos\Delphi\OfficeConverters\
- Demos\CBuilder\OfficeConverters\

Additional resources

- ***Microsoft Office Compatibility Pack for Word, Excel, and PowerPoint File Formats*** contains a set of converters for new formats of Microsoft Office. Alternatively, you can ***download them from our web site***.

1.5.2.3.1 Properties

1.5.2.3.1.1 TRVOfficeConverter.ErrorCode

Returns error code of the last performed import or export operation (if it failed)

```
property ErrorCode: Integer;
```

(introduced in version 1.7)

Error codes returned by the component:

Constant	Value	Meaning
rvceCnvLoadError	1	error loading converter's DLL

rvceFuncError	2	required function is not found in the converter's DLL
rvceInitError	3	converter initialization failure

Some (not all) error codes returned by converters:

Constant	Value	Meaning
rvceOpenInFileErr	-1	could not open input file
rvceReadErr	-2	error during read
rvceOpenConvErr	-3	error opening conversion file
rvceWriteErr	-4	error during write
rvceInvalidFile	-5	invalid data in conversion file
rvceOpenExceptErr	-6	error opening exception file
rvceWriteExceptErr	-7	error writing exception file
rvceNoMemory	-8	out of memory
rvceInvalidDoc	-9	invalid document
rvceDiskFull	-10	out of space on output
rvceDocTooLarge	-11	conversion document too large for target
rvceOpenOutFileErr	-12	could not open output file
rvceUserCancel	-13	conversion canceled by user
rvceWrongFileType	-14	wrong file type for this converter

1.5.2.3.1.2 TRVOfficeConverter.ExcludeDocX-Converter

These properties allow to exclude DocX converters from the list.

```
property ExcludeDocXImportConverter: Boolean;
property ExcludeDocXExportConverter: Boolean;
```

(introduced in version 15)

If ExcludeDocXImportConverter=*True*, DocX import converter is excluded from the list of ImportConverters[535].
If ExcludeDocXExportConverter=*True*, DocX export converter is excluded from the list of ExportConverters[535] (TRichView has SaveDocX[249] method allowing to save DocX better than this converter).

Default value

False

1.5.2.3.1.3 TRVOfficeConverter.ExcludeHTML-Converter

These properties allow to exclude HTML converters from the list.

```
property ExcludeHTMLImportConverter: Boolean;
property ExcludeHTMLExportConverter: Boolean;
```

(introduced in version 1.7)

If ExcludeHTMLImportConverter=*True*, HTML import converter is excluded from the list of ImportConverters[535] (useful, if you have other function for HTML import).
If ExcludeHTMLExportConverter=*True*, HTML export converter is excluded from the list of ExportConverters[535] (TRichView has methods (SaveHTML[250] and SaveHTMLEx[251]) allowing to save HTML better than this converter).

Default value

False

1.5.2.3.1.4 TRVOfficeConverter.ExportConverters

Collection of installed export converters. Export converters convert RTF to foreign formats.

```
property ExportConverters: TRVOfficeCnvList[539];
```

This is a list of TRVOfficeConverterInfo[539].

Order of converters in the list is arbitrary: on another computer, even if the same converters are installed, they may be listed in another order.

See also:

- ImportConverters[535] property;
- GetExportFilter[537] method.

1.5.2.3.1.5 TRVOfficeConverter.ExtensionsInFilter

Allows including file extensions in the file filter (GetImportFilter[537] and GetExportFilter[537])

```
property ExtensionsInFilter: Boolean;
```

(introduced in version 1.7)

For example:

- if ExtensionsInFilter=*False*, file filter entry would be 'Windows Write';
- if ExtensionsInFilter=*True*, file filter entry would be 'Windows Write (*.wri)';

Default value:

False

1.5.2.3.1.6 TRVOfficeConverter.ImportConverters

Collection of installed import converters. Import converters convert foreign formats to RTF.

```
property ImportConverters: TRVOfficeCnvList[539];
```

This is a list of TRVOfficeConverterInfo[539].

Order of converters in the list is arbitrary: on another computer, even if the same converters are installed, they may be listed in another order.

See also:

- ExportConverters[535] property;
- GetImportFilter[537] method.

1.5.2.3.1.7 TRVOfficeConverter.PreviewMode

Instructs converters to do their work without displaying dialogs

```
property PreviewMode: Boolean;
```

(introduced in v1.9)

If set to *True*, converters do their work silently, without displaying any dialogs. They can also produce lower quality results in this mode (that depends on converter).

Default value:

False

1.5.2.3.1.8 TRVOfficeConverter.Stream

A stream to store RTF (Rich Text Format) data.

```
property Stream: TMemoryStream;
```

Export converters convert RTF from this stream to file, import-converters convert file to this RTF stream.

The methods ExportRTF[536] and ImportRTF[538] work with this stream directly.

The methods ExportRV[537] and ImportRV[538] use this stream internally.

1.5.2.3.2 Methods

1.5.2.3.2.1 TRVOfficeConverter.Create

A constructor, creates a new TRVOfficeConverter component.

```
constructor Create(AOwner: TComponent);
```

Use Create to instantiate TRVOfficeConverter object at runtime. RVOfficeConverters placed on forms at design time are created automatically. Specify the owner of the new RVOfficeConverter using the **AOwner** parameter.

1.5.2.3.2.2 TRVOfficeConverter.Destroy

Destroys the RVOfficeConverter

```
destructor Destroy;
```

1.5.2.3.2.3 TRVOfficeConverter.ExportRTF

Exports RTF from Stream[536] to file in foreign format.

```
function ExportRTF(const FileName: TRVUnicodeString[694]; ConverterIndex: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName is a name of file to save.

ConverterIndex is an index of export converter (in ExportConverters[535] list).

When calling this method, Stream[536] must contain RTF (Rich Text Format) document. The method start conversion from Stream[536].Position and stopped when reached the end of stream.

You can provide a visual indication for the conversion progress using OnConverting[539] event.

Note: Internally, the component converts the file name to OEM (DOS) encoding to pass it to the converter. If **FileName** contains characters that cannot be converted to OEM encoding, this method fails.

Return value:

True if the export was successful, *False* if not.

See also:

- ImportRTF[538];
- ExportRV[537].

See also properties:

- ErrorCode[533].

1.5.2.3.2.4 TRVOfficeConverter.ExportRV

Exports content of **rv** to file in foreign format.

```
function ExportRV(const FileName: TRVUnicodeString[694]; rv: TCustomRichView[162];
  ConverterIndex: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName is a name of file to save.

ConverterIndex is an index of export converter (in ExportConverters[535] list)

This method:

1. clears Stream[536];
2. saves RTF to Stream[536] using rv.SaveRTFToStream[254];
3. performs converting;
4. clears Stream[536].

You can provide a visual indication for the conversion progress using OnConverting[539] event.

Note: Internally, the component converts the file name to OEM (DOS) encoding to pass it to the converter. If **FileName** contains characters that cannot be converted to OEM encoding, this method fails.

Return value:

True if the export was successful, *False* if not.

See also:

- ImportRTF[538];
- ExportRV[537].

See also properties:

- ErrorCode[533].

1.5.2.3.2.5 TRVOfficeConverter.GetExportFilter

Returns string that can be assigned to the Filter property of TSaveDialog.

```
function GetExportFilter: TRVUnicodeString[694];
```

(changed in version 18)

If ExtensionsInFilter[535] is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- GetImportFilter[537] method;
- ExportConverters[535] property.

1.5.2.3.2.6 TRVOfficeConverter.GetImportFilter

Returns string that can be assigned to the Filter property of TOpenDialog

```
function GetImportFilter: TRVUnicodeString[694];
```

(changed in version 18)

If ExtensionsInFilter[535] is *True*, file extensions are included in the visible part of the returned file filter.

See also:

- GetExportFilter[537] method;
- ImportConverters[535] property.

1.5.2.3.2.7 TRVOfficeConverter.ImportRTF

Converts data from the file in foreign format to RTF data in Stream[536]

```
function ImportRTF(const FileName: TRVUnicodeString[694];
  ConverterIndex: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName is a name of file to import.

ConverterIndex is an index of import converter (in ImportConverters[535] list)

The method clears Stream[536] and writes RTF (Rich Text Format) document in it.

You can provide a visual indication for the conversion progress using OnConverting[539] event.

Note: Internally, the component converts the file name to OEM (DOS) encoding to pass it to the converter. If **FileName** contains characters that cannot be converted to OEM encoding, this method fails.

Return value:

True if the import was successful, *False* if not.

See also:

- ExportRTF[536];
- ImportRV[538].

See also properties:

- ErrorCode[533].

1.5.2.3.2.8 TRVOfficeConverter.ImportRV

Imports content of file in foreign format to **rv**.

```
function ImportRV(const FileName: TRVUnicodeString[694]; rv: TCustomRichView[162];
  ConverterIndex: Integer): Boolean;
```

(changed in version 18)

Parameters:

FileName is a name of file to import.

ConverterIndex is an index of import converter (in ImportConverters[535] list)

This method:

1. clears Stream[536];
2. imports data from the file to Stream[536];
3. reads data from Stream[536] in rv, using rv.LoadRTFFromStream[244],
4. clears Stream[536].

You can provide a visual indication for the conversion progress using OnConverting[539] event.

Note: Internally, the component converts the file name to OEM (DOS) encoding to pass it to the converter. If **FileName** contains characters that cannot be converted to OEM encoding, this method fails.

Return value:

True if the import was successful, *False* if not.

See also:

- ImportRTF[538];
- ExportRV[537].

See also properties:

- ErrorCode[533].

1.5.2.3.2.9 TRVOfficeConverter.IsValidImporter

Checks if the specified file can be imported with the specified converter.

```
function IsValidImporter(const FileName: TRVUnicodeString[694]; Index: Integer): Boolean;
```

(introduced in version 12; changed in version 18)

Parameters:

FileName is a name of file to test.

Index is an index of import converter (in ImportConverters[535] list)

Return value:

If this method returns *False*, this file cannot be imported by this converter. If this method returns *True*, this file **probably** can be imported by this converter.

See also properties:

- ErrorCode[533].

1.5.2.3.3 Events

1.5.2.3.3.1 TRVOfficeConverter.OnConverting

Occurs while converting (in ImportRTF[538], ExportRTF[536], ImportRV,[538] ExportRV[537] methods)

```
property OnConverting: TConvertingEvent;
type
  TConvertingEvent = procedure (Sender:TObject; Percent: Integer) of object;
```

Percent is a value of completeness, from 0 to 100.

See also event of TRichView:

- OnProgress[283].

1.5.2.3.4 Classes of Properties

1.5.2.3.4.1 TRVOfficeCnvList

This class represents a list of installed import[535] or export[535] converters.

Unit RVOfficeCnv;

Syntax

```
TRVOfficeCnvList = class (TList)
```

Hierarchy

TObject
TList

Main Properties

The main properties are **Count** (count of converters) and **Items** (array of TRVOfficeConverterInfo[539]).

1.5.2.3.4.2 TRVOfficeConverterInfo

This class represents an item in list of installed import[535] and export[535] converters.

Unit RVOfficeCnv;

Syntax

```
TRVOfficeConverterInfo = class;
```

Hierarchy

TObject

Main Properties

This class has the following main properties:

- **Name**: String – name of format (for example, 'HTML Document');
- **Path**: String – path to the converter's DLL;
- **Filter**: String – filter (file mask, for example '*.htm;*.html;*.htx;*.otm').

1.5.2.4 TRVReportHelper

This component is similar to TRVPrint[513], but:

- it can be used to draw on any Canvas (not only on the printer's canvas) pages of any size (not limited to printer page sizes); however, a similar functionality can be implemented using TRVPrint.VirtualPrinter[522];
- it contains a document, not a reference to it.

Unit RVReport;

Syntax

```
TRVReportHelper = class(TCustomRVPrint[549])
```

Hierarchy

TObject
TPersistent
TComponent
TCustomRVPrint[549]

How to Use

1. Load document in RVReportHelper.RichView[541].
2. Initialize formatting (Init[543]). Specify the page width and canvas.
3. Format the document page by page (FormatNextPage[542]). Pages may have different heights.
4. Draw pages on canvas (DrawPage[541] or DrawPageAt[542]). If you need to know positions of hyperlinks and checkpoints, use OnDrawHyperlink[556] and OnDrawCheckpoint[556] events.

Example (drawing document in metafile)

```
procedure TForm1.DrawToMetafile(rvh: TRVReportHelper;
  wmf: TMetafile; Width: Integer);
const VERYLARGEVALUE = $FFFFFFF;
var Canvas: TMetafileCanvas;
begin
  rvh.IgnorePageBreaks[550] := True;
  rvh.Init[543](Self.Canvas, Width);
  while rvh.FormatNextPage[542](VERYLARGEVALUE) do;
  wmf.Width := Width;
  wmf.Height := rvh.GetLastPageHeight[543];
  Canvas := TMetafileCanvas.Create(wmf, 0);
  rvh.DrawPage[541](1,Canvas,True,rvh.GetLastPageHeight[543]);
  Canvas.Free;
end;
```

Compatibility options

Some canvases may have problems with advanced content (especially when you draw to create PDF or metafile files). In this case, you can use the properties:

- MetafileCompatibility[551];
- NoMetafiles[551].

Demo projects

- Demos\Delphi\Assorted\Graphics\ToImage\
- Demos\Delphi\Assorted\Printing\ReportHelper\

(for C++Builder demos, change "Delphi" to "CBuilder" in the paths above)

1.5.2.4.1 Properties

1.5.2.4.1.1 TRVReportHelper.AllowTransparentDrawing

Allows using methods drawing semitransparent images

```
property AllowTransparentDrawing: Boolean;
```

(introduced in version 16)

This property is not used when printing, i.e. it is used when **Preview** parameter of DrawPage[541] /DrawPageAt[542] is *True*.

AllowTransparentDrawing	"Preview"	Result
True	*True*	Drawing using semitransparent methods
False	*True*	Drawing semitransparent areas in a bitmap, then drawing this bitmap
True or False	*False*	Drawing semitransparent areas in a bitmap, then printing this bitmap (a special printer-friendly method of bitmap drawing is used)

Default value

False

1.5.2.4.1.2 TRVReportHelper.RichView

This is the object containing document of the report helper.

```
property RichView: TRichView[162];
```

This RichView object is invisible. Use it to load document and to set properties for loading. This RichView must be linked[193] with some TRVStyle[418] object.

In D6+, you can set properties and events of this object at designtime using the Object Inspector.

Do not use formatting methods of this RichView. Use Init[543] and FormatNextPage[542] instead.

1.5.2.4.2 Methods

1.5.2.4.2.1 TRVReportHelper.DrawPage

Draws the specified page

```
procedure DrawPage(APageNo: Integer; ACanvas: TCanvas;
  APreview: Boolean; AHeight: Integer);
```

Parameters:

APageNo – index of the page to draw. The first page has index = 1.

ACanvas – Canvas where to render the page. It's not necessary the same canvas as was used in Init[543], but must have the same resolution (and the same value of Font.PixelsPerInch)

The top left corner of the page is drawn at (0,0) coordinates.

APreview – should be set to *True* when drawing on the screen canvas, to *False* when drawing on printer canvas.

AHeight – height of this page; this value is used for drawing background.

This page must be formatted before drawing, see FormatNextPage 542.

See also:

- DrawPageAt 542.

1.5.2.4.2.2 TRVReportHelper.DrawPageAt

Draws the specified page at the specified coordinates

```
procedure DrawPageAt(Left, Top, APageNo: Integer;
  ACanvas: TCanvas; APreview: Boolean; AHeight: Integer);
```

Parameters:

APageNo – index of the page to draw. The first page has index = 1.

ACanvas – Canvas where to render the page. It's not necessary the same canvas as was used in Init 543, but must have the same resolution (and the same value of Font.PixelsPerInch)

The top left corner of the page is drawn at (**Left**,**Top**) coordinates.

APreview – should be set to *True* when drawing on the screen canvas, to *False* when drawing on printer canvas.

AHeight – height of this page; this value is used for drawing background.

This method uses SetWindowOrgEx to move origin to (**Left**,**Top**).

This page must be formatted before drawing, see FormatNextPage 542.

See also:

- DrawPage 541.

1.5.2.4.2.3 TRVReportHelper.Finished

Returns *True* if all pages were formatted.

```
function Finished: Boolean;
```

You can use the value returned by FormatNextPage 542 instead.

1.5.2.4.2.4 TRVReportHelper.FormatNextPage

Formats next page(s) of document

```
function FormatNextPage(AMaxHeight: Integer): Boolean;
```

Call this method after Init 543 several times, while it returns *True*.

Parameter:

AMaxHeight – page height, pixels; page width is specified as a parameter of Init 543.

When the page is formatted, it can be drawn (DrawPage 541, DrawPageAt 542).

Sometimes, this method formats more than one page.

This method is fast. The most time consuming method is Init 543.

Return value:

- *True* if not all pages were formatted;
- *False*, if formatting is complete.

1.5.2.4.2.5 TRVReportHelper.GetLastPageHeight

Returns a real height of the last page, in device pixels

```
function GetLastPageHeight: Integer;
```

This function returns height of document on the last page, when the document is completely formatted (all calls to FormatNextPage[542] are made).

The returned value is measured in **ACanvas** pixels, where **ACanvas** is the parameter of Init[543].

This method is useful if you draw/print several documents one after another.

1.5.2.4.2.6 TRVReportHelper.Init

Prepares document for formatting.

```
procedure Init(ACanvas: TCanvas; APageWidth: Integer);
```

Parameters:

ACanvas – canvas object used for formatting.

APageWidth – page width, pixels; all pages have the same width.

See also:

- FormatNextPage[542].

1.5.2.4.2.7 TRVReportHelper.UpdatePageNumbers

Recalculates page numbers.

```
procedure UpdatePageNumbers;
```

(introduced in version 15)

This method recalculates values of "page number"[141] and "page count"[142] fields.

Normally, values of these fields are calculated when the last page is formatted.

This procedure needs to be called if values of "page number" field are changed. For example, if this RVReportHelper is used to draw the same content on each page, you can provide a number of this page in OnGetPageNumber[543] event, and call UpdatePageNumber before drawing.

1.5.2.4.3 Events

1.5.2.4.3.1 TRVReportHelper.OnGetPageNumber

Request a page number for displaying in "page number" fields[141].

```
type
  TRVGetPageNumberEvent = procedure (Sender: TRVReportHelper[540]; PageNo: Integer;
    var DispPageNo: Integer) of object;;

property OnGetPageNumber: TRVGetPageNumberEvent;
```

(introduced in v15)

Parameters:

PageNo – a number of RVReportHelper's page (from 1 to PagesCount[552]), where the "page number" field is located.

DispPageNo – a number that will be displayed in this field. Initially, it is equal to **PageNo**+PageNoFromNumber[552]-1. You can assign another value to this parameter.

See also:

- UpdatePageNumbers[543]

1.5.3 Components Ancestor Classes

1.5.3.1 TRVScroller

This class is an ancestor of all visual components in TRichView family (TRichView[162], TRichViewEdit[316], TDBRichView[407], TDBRichViewEdit[316], TRVPrintPreview[417]).

Unit RVScroll;

Syntax

```
TCustomRVControl = class (TCustomControl)
TRVScroller = class(TCustomRVControl)
```

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TCustomRVControl

Main Features

This is a control what can scroll its content.

Due to limitation of scrollbar ranges, its vertical scrollbar units are not pixels but special units, "RVSU" (10 pixels by default).

Do not use this class directly in your program. Use its descendants, such as TRichView[162].

TRVScroller is not registered as a component itself. It introduces some properties and methods used in descendant components.

Main Properties and Methods

The most important properties of TRVScroller are:

- VScrollPos[547] –- position of vertical scrollbar, measured in "RVSU";
- VScrollMax[547] (read-only) –- maximum value of VScrollPos;
- NoHScroll[546] –- disallows horizontal scrolling;
- NoVScroll[546] –- disallows vertical scrolling;
- HScrollPos[545] –- position of horizontal scrollbar, measured in pixels;
- HScrollMax[545] (read-only) –- maximum value of HScrollPos;
- WheelStep[548] defines speed of mouse wheel scrolling;
- InplaceEditor[546] –- returns "inplace" editor of some item in document (used in tables for editing cells).

The most important method is:

- ScrollTo[548], scrolls content of the component vertically to the specified position (in pixels).

1.5.3.1.1 Properties

1.5.3.1.1.1 TRVScroller.BorderStyle

Determines whether the control has border.

```
type TBorderStyle = bsNone..bsSingle;

property BorderStyle: TBorderStyle;
```

Set BorderStyle to specify whether the control should be outlined.

Value	Meaning
bsNone	No visible border
bsSingle	Border, 3D or themed [546]

1.5.3.1.1.2 TRVScroller.HScrollMax

Maximal possible value for HScrollPos [545]

```
property HScrollMax: Integer;
```

See also properties:

- HScrollPos [545];
- VScrollMax [547].

See also:

- Scrolling in RichView [80]

1.5.3.1.1.3 TRVScroller.HScrollPos

Position of horizontal scrollbar. This value is measured in pixels. You can use this property to scroll document to specified horizontal position.

```
property HScrollPos: Integer;
```

This value is in range 0..HScrollMax [545].

See also properties:

- VScrollPos [547].

See also:

- Scrolling in RichView [80].

1.5.3.1.1.4 TRVScroller.HScrollVisible

Set to *False* to hide horizontal scrollbar

```
property HScrollVisible: Boolean;
```

(introduced in version 1.4)

Default value:

True

See also properties:

- VScrollVisible [547].

See also:

- Scrolling in RichView [80].

1.5.3.1.1.5 TRVScroller.InplaceEditor

This property provides access to editor created for editing one of document items.

```
property InplaceEditor: TWinControl;
```

(introduced in version 1.4)

Inplace editor can be used to edit some cell in a table[565] (see "Cell Editing[151]"). In future versions, inplace editors can be used for some other types of items.

Inplace editors are derived from TCustomRichViewEdit[316] type.

Inplace editor in its order can have its own inplace editor, and so on. If there is no inplace editor, this property is *nil*. At any time the control can have no more than one inplace editor.

Example: (obtaining top level editor)

```
var
  rve: TCustomRichViewEdit;
...
  rve := MyRichViewEdit;
  while rve.InplaceEditor<>nil do
    rve := rve.InplaceEditor as TCustomRichViewEdit;
  // Now rve points to the top level inplace editor
  // (or MyRichViewEdit itself, if there are no
  // inplace editors).
```

See also:

- TCustomRichViewEdit.TopLevelEditor[329].

1.5.3.1.1.6 TRVScroller.NoVScroll

Disables vertical and/or horizontal scrolling

```
property NoVScroll: Boolean;
property NoHScroll: Boolean;
```

(introduced in versions 11, 15)

Default value

False

1.5.3.1.1.7 TRVScroller.Tracking

Determines whether the control is updated when the user drags the thumb tab of scrollbar.

```
property Tracking: Boolean;
```

Repainting while the thumb tab is dragged can provide useful feedback to the user about how far the control has scrolled, but slows performance.

Default value:

True

1.5.3.1.1.8 TRVScroller.UseXPThemes

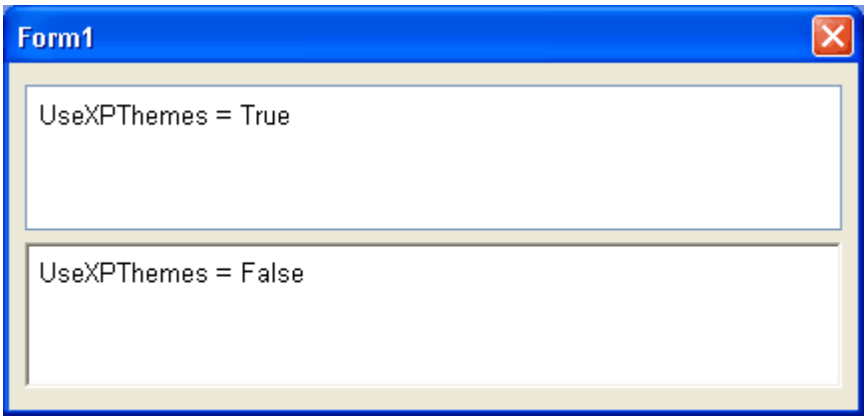
Allows using Windows themes (visual styles)

```
property UseXPThemes: Boolean
```

(introduced in version 1.7)

Themes (visual styles) were introduced in Windows XP and available in newer versions of Windows as well (Windows Vista, 7, 8, 10, etc.)

If this property is set to *True*, the control will use themes if they are available.



UseXPThemes Example

See also:

- BorderStyle 545.

1.5.3.1.1.9 TRVScroller.VScrollMax

Maximal possible value for VScrollPos 547

```
property VScrollMax: Integer;
```

See also properties:

- VScrollPos 547;
- HScrollMax 545.

See also methods:

- ScrollTo 548.

See also:

- Scrolling in RichView 80.

1.5.3.1.1.10 TRVScroller.VScrollPos

Position of vertical scrollbar. This value is measured in "RVSU" 80, not in pixels because of Windows limitation on scrollbar scrolling range. You can use this property to scroll document to specified position.

```
property VScrollPos: Integer;
```

This value is in range 0..VScrollMax 547.

See also properties:

- HScrollPos 545.

See also methods:

- ScrollTo 548.

See also:

- Scrolling in RichView 80.

1.5.3.1.1.11 TRVScroller.VScrollVisible

Set to *False* to hide vertical scrollbar

```
property VScrollVisible: Boolean;
```

Known problem: scrollbars do not reappear correctly in TRichViewEdit 316 if set VScrollVisible from *False* to *True* when document is not empty.

Default value:

True

See also properties:

- HScrollVisible 545.

See also:

- Scrolling in RichView 80.

1.5.3.1.1.12 TRVScroller.WheelStep

Defines how much the document will be scrolled when the user turns a mouse wheel.

```
property WheelStep: Integer;
```

If set to 0, the control ignores mouse wheel messages (they will be sent to the parent control).

Mouse wheel scrolling in TRVScroller depends on the system settings. The system suggests the default scrolling speed. If the suggested value is "one page", this property is ignored (if nonzero). If the suggested value is "no scrolling", no scrolling occurs. Otherwise, the component is scrolled by Round(VSmallStep 196 *WheelStep*suggested/3) pixels.

Default value:

2

1.5.3.1.2 Methods

1.5.3.1.2.1 TRVScroller.Create

A constructor, creates and initializes a TRVScroller instance.

```
constructor Create(AOwner: TComponent);override;
```

Do not create TRVScroller objects in your applications directly. Create components inherited from it:

- TRichView 162;
- TRichViewEdit 316;
- TDBRichView 407;
- TDBRichViewEdit 316;
- TRVPrintPreview 417.

1.5.3.1.2.2 TRVScroller.ScrollTo

Scrolls content of the document vertically to the specified position (in pixels).

```
procedure ScrollTo(y: Integer);
```

Vertical scrollbar units are not pixels, so scrolling will be not exactly to specified position (but a little bit higher).

See also properties:

- VScrollPos 547;
- VScrollMax 547.

See also:

- Scrolling in RichView 80.

1.5.3.1.3 Events

1.5.3.1.3.1 TRVScroller.OnHScrolled

Occurs when the document is scrolled horizontally

```
property OnHScrolled: TNotifyEvent;
```

See also events:

- OnVScrolled 549.

See also:

- Scrolling in RichView 80.

1.5.3.1.3.2 TRVScroller.OnVScrolled

Occurs when document is scrolled vertically

```
property OnVScrolled: TNotifyEvent;
```

See also events:

- OnHScrolled[548].

See also properties of TRichView:

- FirstItemVisible[179];
- LastItemVisible[180].

See also:

- Scrolling in RichView[80].

1.5.3.2 TCustomRVPrint

TCustomRVPrint is an Ancestor of TRVPrint[513] and TRVReportHelper[540] components.

This is an intermediate class. Do not use it directly in your programs.

Unit PtblRV;

Syntax

```
TCustomRVPrint = class(TComponent)
```

Hierarchy

TObject
TPersistent
TComponent

1.5.3.2.1 Properties

1.5.3.2.1.1 TCustomRVPrint.ColorMode

Sets color mode for printing and drawing preview

```
property ColorMode: TRVColorMode[671];
```

(Introduced in version 1.8)

Default values:

- *rvcmColor* for TRVReportHelper[540];
- *rvcmPrinterColor* for TRVPrint[513].

1.5.3.2.1.2 TCustomRVPrint.EndAt

Height of document on the last page, in device pixels.

```
property EndAt: Integer;
```

This property does not include margins.

You can use this property only when document is completely formatted (see FormatPages[525] for TRVPrint, Init[543] and FormatNextPage[542] for TRVReportHelper).

For TRVReportHelper, you can use GetLastPageHeight[543] instead (it's more convenient, because it includes top and bottom margins).

See also properties:

- StartAt[552].

1.5.3.2.1.3 TCustomRVPrint.FirstPageNo

Returns the index of the first page that will be printed, from 1.

```
property FirstPageNo: Integer;
```

(Introduced in version 13)

This property takes the properties MinPrintedItemNo[551] and MaxPrintedItemNo[550] into account. By default, when these properties define the range including the whole document, FirstPageNo returns 1. If MinPrintedItemNo[551]>0, this property returns the index of the page containing the MinPrintedItemNo[551]-th item (see GetPageNo[553]). If MinPrintedItemNo[551]>=ItemCount, FirstPageNo returns PagesCount[552]+1 (and nothing will be printed).

1.5.3.2.1.4 TCustomRVPrint.IgnorePageBreaks

Allows ignoring page breaks in a document.

```
property IgnorePageBreaks: Boolean;
```

(Introduced in version 14)

If *True*, the component ignores the following properties when printing:

- TRichView.PageBreaksBeforeItems[184][];
- TRVTableItemInfo.Rows[].PageBreakBefore[616].

This property is useful in TRVReportHelper[540] component, if you want to display the loaded document completely in a single rectangle.

Default value

False

1.5.3.2.1.5 TCustomRVPrint.LastPageNo

Returns the index of the last page that will be printed, from 1.

```
property LastPageNo: Integer;
```

(Introduced in version 13)

This property takes the properties MinPrintedItemNo[551] and MaxPrintedItemNo[550] into account. By default, when these properties define the range including the whole document, LastPageNo returns PagesCount[552]. If MaxPrintedItemNo[550] is inside the items range (0..ItemCount-1), this property returns the index of the page containing the MaxPrintedItemNo[550]-th item (see GetPageNo[553]).

1.5.3.2.1.6 TCustomRVPrint.MaxPrintedItemNo

Defines the index of the last item to print.

```
property MaxPrintedItemNo: Integer;
```

(Introduced in version 13)

The component prints items from MinPrintedItemNo[551] to MaxPrintedItemNo. By default, this range includes all items.

The value -1 is treated like ItemCount-1, i.e. the document is printed to the end.

Items outside of this range are not printed (but they still occupy their spaces). The pages that do not contain items in this range are skipped (the component prints pages from FirstPageNo[550] to LastPageNo[550]).

These properties allow implementing an incremental printing: when the next portion of the document is ready, it can be printed below the previously printed fragment. This feature can be used for printing accounting journals, logs, etc.

Default value:

-1

1.5.3.2.1.7 TCustomRVPrint.MetafileCompatibility

Specifies whether printing output is compatible with metafiles.

```
property MetafileCompatibility: Boolean;
```

(introduced in version 16)

If RVStyle.TextEngine[436] = *rvteUniscribe*, or for bidirectional text, the component may draw/print text using glyph indexes in fonts instead of character codes. When printing output is captured to a metafile, this metafile can be viewed correctly only on computers having the same font installed. On computers without this font, or with different version of this font, such metafiles may look completely incorrect. Also, it may cause problems for some virtual printers.

You can set **MetafileCompatibility**=*True* to switch to metafile-compatible output (less efficient, and in some cases of a lower quality). Some items (for example equations[143]) will print themselves as bitmaps.

Default value

False

1.5.3.2.1.8 TCustomRVPrint.MinPrintedItemNo

Defines the index of the first item to print.

```
property MinPrintedItemNo: Integer;
```

(Introduced in version 13)

The component prints items from MinPrintedItemNo to MaxPrintedItemNo[550]. By default, this range includes all items.

Items outside of this range are not printed (but they still occupy their spaces). The pages that do not contain items in this range are skipped (the component prints pages from FirstPageNo[550] to LastPageNo[550]).

These properties allow implementing an incremental printing: when the next portion of the document is ready, it can be printed below the previously printed fragment. This feature can be used for printing accounting journals, logs, etc.

Default value:

0

1.5.3.2.1.9 TCustomRVPrint.NoMetafiles

Allows drawing metafiles as bitmaps.

```
property NoMetafiles: Boolean;
```

(introduced in version 15)

If *True*, metafiles (in pictures[122] and *hot-pictures*[125]) and equations[143] are drawn as bitmap images.

It may be useful when creating PDF using third-party libraries, if they have problems with metafiles (especially when PDF pixel density) is different from the screen pixels density).

Default value

False

See also

- MetafilePixelsPerInch[551]

1.5.3.2.1.10 TCustomRVPrint.MetafilePixelsPerInch

Specifies the "pixels per inch" value used when converting metafiles to bitmaps.

```
property MetafilePixelsPerInch: Integer;
```

(introduced in version 17)

Metafiles (in pictures[122] and *hot-pictures*[125]) are printed as bitmaps, if NoMetafiles[551] = *True*. This property allows changing a pixel density of these bitmaps. Larger values produce results of higher quality, but require more memory (when printing to PDF using some third-party library, larger values produce larger PDF files).

For example, you can assign 300 to this property.

Default value

0 (default bitmap sizes).

1.5.3.2.1.11 TCustomRVPrint.PageNoFromNumber

Specifies the number of the first page for "page number" fields[141].

```
property PageNoFromNumber: Integer;
```

(introduced in version 15)

Default value:

1

See also:

- TRVReportHelper.OnGetPageNumber[543]

1.5.3.2.1.12 TCustomRVPrint.PagesCount

Total number of pages in the document.

```
property PagesCount: Integer;
```

You can use this property only when document is formatted (see FormatPages[525] for TRVPrint, Init[543] and FormatNextPage[542] for TRVReportHelper).

1.5.3.2.1.13 TCustomRVPrint.PreviewCorrection

Determines whether the component will make correction when drawing print preview.

```
property PreviewCorrection: Boolean;
```

This property does not affect to the printing itself, only to the preview.

Due to different resolutions on the screen and on the printer, print preview may be inaccurate (because text not always can be scaled proportionally).

If this property is *True*, the component tries to perform some corrections to improve preview quality (for example, it adjusts intercharacter spacing).

Default value:

True

1.5.3.2.1.14 TCustomRVPrint.StartAt

Offset from the top of the first page (from the top margin) to the beginning of document on it, in device pixels.

```
StartAt: Integer;
```

This property is useful (in conjunction with EndAt[549]) in TRVPrint.ContinuousPrint[523] method.

Default value:

0

1.5.3.2.1.15 TCustomRVPrint.TransparentBackground

Allows to turn off printing of the document background

```
property TransparentBackground: Boolean;
```

If set to *True*, background will not be printed.

Default value:

False

See also:

- *rvtoWhiteBackground* option in PrintOptions[579] of tables.

1.5.3.2.2 Methods

1.5.3.2.2.1 TCustomRVPrint.Clear

Clears memory allocated for printing (in FormatPages[525])

```
procedure Clear;
```

After calling Clear, TRVPrint will not be able to print and display preview until the next call of AssignSource[522] and FormatPages[525] (or Init[543] and FormatNextPage[542] in TRVReportHelper).

1.5.3.2.2.2 TCustomRVPrint.GetFirstItemOnPage

Returns index of the first item on the page

```
procedure GetFirstItemOnPage(PageNo: Integer; var ItemNo, OffsetInItem: Integer);
```

Input parameter:

PageNo – number of page (from 1).

Output parameters:

ItemNo – index of the first item on the page (from 0)

OffsetInItem:

- for non-text items: 0;
- for text items: index of the first character on the page (from 1)

Note: this information is not enough when using tables[565]. Tables can be printed on several pages, and using this information you cannot determine which part of table starts the page.

See also:

- GetPageNo[553].

See also properties:

- PagesCount[552].

1.5.3.2.2.3 TCustomRVPrint.GetPageNo

Returns index of page containing the specified place of the document.

```
function GetPageNo(RVData: TCustomRVData; ItemNo, OffsetInItem: Integer): Integer;
```

Parameters:

The parameters define a position in the source document,

RVData is a document/subdocument. It can be either RVData[187] of the source TRichView control, or table cell, or RVData[187] of a cell inplace editor.

For TRVPrint[513], the source TRichView control is a control assigned by AssignSource[522]. For TRVReportHelper, the source TRichView control is RichView[541] property.

ItemNo is the index of item in this **RVData**.

OffsetInItem is the position inside this item. For non-text items, 0 is a position before the item, 1 is a position after the item. For text items[120], offsets are counted from 1 (1 - before the first character, 2 - before the second character, ... Length+1 - after the last character).

Return value:

Index of the page for this position, from 1.

See also:

- GetFirstItemOnPage[553].

See also methods of TCustomRichView:

- GetOffsAfterItem[235];
- GetOffsBeforeItem[235].

1.5.3.2.2.4 TCustomRVPrint.UpdatePaletteInfo

Call this method when the screen color resolution changed (on WM_DISPLAYCHANGE), if you are using this RVPrint for displaying print preview.

This method is necessary (only) if you need to support 256 color display mode.

```
procedure UpdatePaletteInfo;
```

Example[178]

TRVPrint uses the same method of working in 256 color mode as its source RichView (see RichView.DoInPaletteMode[177]). Information about source RichView.DoInPaletteMode is updated when executing AssignSource[522] method.

It's not necessary to call this method if RichView.DoInPaletteMode=*rvpaDoNothing,* or if you are not using this TRVPrint object for displaying print preview in TRVPrintPreview[417].

Call UpdatePaletteInfo if you created TRVPrint at run-time (instead of placing at design-time on a form) after you have created it.

TRVPrintPreview controls do not have public UpdatePaletteInfo method. They use information from the linked RVPrint. They update their palettes when you assign their PageNo[559] or RVPrint[418] properties.

See also methods of TRichView:

- UpdatePaletteInfo[273].

1.5.3.2.3 Events

1.5.3.2.3.1 TCustomRVPrint.OnPrintComponent

Occurs when printing inserted control[126]

```
property OnPrintComponent: TRVPrintComponentEvent;

TRVPrintComponentEvent = procedure(Sender: TCustomRVPrint[549];
  PrintMe: TControl; var ComponentImage: TBitmap) of object;
```

If you are not satisfied how RichView prints some classes of controls, you can print them yourself.

This event requests image of control, in screen resolution, in bitmap.

Input parameters:

PrintMe – the control for printing.

ComponentImage – *nil*.

Output parameter:

ComponentImage – image of **PrintMe**. You should create this bitmap in this event and should not destroy it yourself (RVPrint will do it for you).

Create the bitmap having width and height of PrintMe (you can return larger bitmap and it will be stretched, allowing to create better print output).

There are several methods for creating bitmap for components in **CtrlImage** unit: DrawControl and others [660]. The component use them for default controls printing.

Example:

```
procedure MyForm.MyRVPrintPrintComponent(Sender: TCustomRVPrint;
  PrintMe: TControl; var ComponentImage: TBitmap);
var r: TRect;
    State: Cardinal
begin
  if PrintMe is TImage then
  begin
    // printing TImage
    ComponentImage := TBitmap.Create;
    ComponentImage.Assign(TImage(PrintMe).Picture.Graphic);
  end
  else if PrintMe is TCheckBox then
  begin
    // printing TCheckBox
    ComponentImage := TBitmap.Create;
    ComponentImage.Width := PrintMe.Width;
    ComponentImage.Height := PrintMe.Height;
    ComponentImage.Canvas.Brush.Color := TCheckBox(PrintMe).Color;
    r := Rect(0,0, PrintMe.Width, PrintMe.Height);
    ComponentImage.Canvas.FillRect(r);
    if TCheckBox(PrintMe).Checked then
      State := DFCS_CHECKED
    else
      State := 0;
    r.Right := r.Bottom;
    DrawFrameControl(ComponentImage.Canvas.Handle, r, DFC_BUTTON,
      DFCS_BUTTONCHECK or State);
    r := Rect(r.Right,0, PrintMe.Width, PrintMe.Height);
    ComponentImage.Canvas.Font := TCheckBox(PrintMe).Font;
    DrawText(ComponentImage.Canvas.Handle, PChar(' '+TCheckBox(PrintMe).Caption), -1, r,
      DT_VCENTER or DT_SINGLELINE);
  end
  else
    // printing other controls
    ComponentImage := DrawControl[660](PrintMe)
end;
```

In TRVPrint, this event is generated by:

- Print [526];
- PrintPages [527];
- MakePreview [526];
- MakeScaledPreview [526];
- DrawPreview [524];
- when drawing TRVPrintPreview [417] component.

1.5.3.2.3.2 TCustomRVPrint.OnDraw Hyperlink

Occurs when the component draws a hyperlink

```
type
  TRVDrawHyperlinkEvent =
    procedure (Sender: TCustomRVPrint[549];
      RVData: TCustomRVData[650]; ItemNo: Integer;
      R: TRect) of object;

property OnDrawHyperlink: TRVDrawHyperlinkEvent;
```

(introduced in v1.8, modified in v17)

This method informs when some hypertext item is drawn.

Parameters:

RVData – document containing the hyperlink.

ItemNo – index of the hypertext item inside this document.

R – hyperlink rectangle (coordinate of the top left corner of the document is (0,0)).

This is not a custom drawing event but an informational event.

See also:

- OnDrawCheckpoint[556].

1.5.3.2.3.3 TCustomRVPrint.OnDraw Checkpoint

Occurs when the component draws (or prints) an item having a *checkpoint*[65]

```
type
  TRVDrawCheckpointEvent = procedure (Sender: TCustomRVPrint[549];
    RVData: TCustomRVData[650]; ItemNo: Integer; X, Y: Integer) of object;

property OnDrawCheckpoint: TRVDrawCheckpointEvent;
```

(introduced in v1.9, modified in v17)

This method informs when some item with *checkpoint* is drawn. It's similar to OnDrawHyperlink[556].

Parameters:

RVData – document containing the item.

ItemNo – index of the checkpoint's item inside this document.

(**X**,**Y**) – top left corner of this item on page (coordinate of the top left corner of the document is (0,0)).

This is not a custom drawing event but an informational event.

For example, this method can be used to create links inside document in generated PDF files (note: PDF generation requires other third-party components).

1.5.3.3 TCustomRVPrintPreview

TCustomRVPrintPreview is an ancestor of TRVPrintPreview[417] component. This class draws a blank page, introduces mechanisms of page switching and scaling.

Unit CRVPP;

Syntax

```
TCustomRVPrintPreview = class(TRVScroller[544])
```

Hierarchy

TObject
TPersistent
TComponent
TControl
TWinControl
TCustomControl
TRVScroller[544]

Cursors

Print preview uses two cursors:

- ZoomInCursor[560],
- ZoomOutCursor[561].

Scaling

Scaling is defined by:

- ZoomMode[561] property – mode of scaling;
- ZoomPercent[562] property – scaling percent;
- SetZoom[563] method changes ZoomPercent[562];
- OnZoomChanged[563] event occurs when ZoomPercent[562] changes;
- ClickMode[558] property allows/disallows zooming.

Page switching

First[562], Last[562], Prev[563], Next[563] methods, PageNo[559] property.

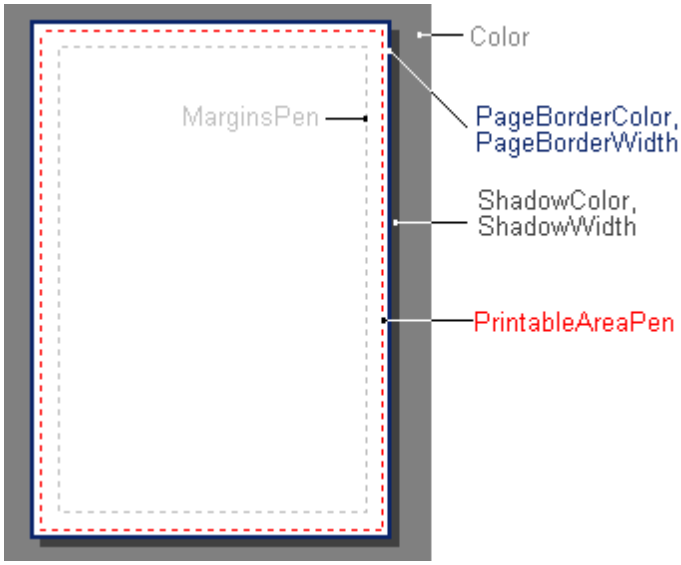
Displaying margins

MarginsPen property[559]

Colors and widths

- Color[558] – background color;
- PageBorderColor[559], PageBorderWidth[559] define border around the page;
- ShadowColor[560], ShadowWidth[560] define shadow.

Scheme



TCustomRVPrintPreview

1.5.3.3.1 Properties

1.5.3.3.1.1 TCustomRVPrintPreview.ClickMode

Defines the action occurring when the user clicks the preview.

```
property ClickMode: TRVClickMode;

type
  TRVClickMode = (rvcmNone, rvcmSwitchZoom);
```

(introduced in v1.8)

Click Mode	Meaning
rvcmNone	Clicking does nothing, zoom-in[560] and zoom-out[561] cursors are not used.
rvcmSwitchZoom	Clicking switches between 100% and custom zoom.

Default value:

rvcmSwitchZoom

1.5.3.3.1.2 TCustomRVPrintPreview.Color

Background color

```
property Color: TColor;
```

(introduced in v1.8)

Note: In Delphi XE2+, if this value contains a system color, and a custom style is applied to the application, the component uses the corresponding style color.

Default value:

clBtnShadow

See also:

- Picture at the bottom of TCustomRVPrintPreview[556] topic.

1.5.3.3.1.3 TCustomRVPrintPreview.MarginsPen

This pen is used for drawing rectangle around the document area on the page preview.

```
property MarginsPen: TRVMarginsPen;
```

By default, MarginsPen.Style=*psClear*, so the rectangle is invisible:

```
type
  TRVMarginsPen = class (TPen)
    property Style default psClear;
    property Color default clSilver;
  end;
```

See also properties:

- PrintableAreaPen [560].

See also properties of TRVPrint [513]:

- Margins [520],

See also:

- Picture in TRVPrint topic [513].

1.5.3.3.1.4 TCustomRVPrintPreview.PageBorderColor

Color of frame around the page.

```
property PageBorderColor: TColor;
```

(introduced in v1.8)

Note: In Delphi XE2+, if this value contains a system color, and a custom style is applied to the application, the component uses the corresponding style color.

Default value:

clHighlight

See also:

- Picture at the bottom of TCustomRVPrintPreview [556] topic.

1.5.3.3.1.5 TCustomRVPrintPreview.PageBorderWidth

Width of frame around the page, pixels

```
property PageBorderWidth: Integer;
```

(introduced in v1.8)

Default value:

2

See also:

- Picture at the bottom of TCustomRVPrintPreview [556] topic.

1.5.3.3.1.6 TCustomRVPrintPreview.PageNo

Index of displayed page (from 1)

```
property PageNo: Integer;
```

See also methods:

- Prev [563];
- Next [563];
- First [562];
- Last [562].

1.5.3.3.1.7 TCustomRVPrintPreview.PrintableAreaPen

This pen is used for drawing rectangle around the printable area on the page preview.

```
property PrintableAreaPen: TRVPAPen;
```

(introduced in version 10)

By default, PrintableAreaPen.Style=*psClear*, so the rectangle is invisible:

```
type
  TRVPAPen = class (TPen)
    property Style default psClear;
    property Color default clRed;
  end;
```

Printer cannot print beyond the area shown on this rectangle, see WinAPI function GetDeviceCaps, PHYSICAL*** constants.

See also properties:

- MarginsPen [559].

See also:

- Picture in TRVPrint topic [513].

1.5.3.3.1.8 TCustomRVPrintPreview.Shadow Color

Color of the page shadow.

```
property ShadowColor: TColor;
```

(introduced in v1.8)

Note: In Delphi XE2+, if this value contains a system color, and a custom style is applied to the application, the component uses the corresponding style color.

Default value:

cl3DDkShadow

See also:

- Picture at the bottom of TCustomRVPrintPreview [556] topic.

1.5.3.3.1.9 TCustomRVPrintPreview.Shadow Width

Width of the page shadow, pixels.

```
property ShadowWidth: TColor;
```

(introduced in v1.8)

Default value:

20

See also:

- Picture at the bottom of TCustomRVPrintPreview [556] topic.

1.5.3.3.1.10 TCustomRVPrintPreview.ZoomInCursor

Cursor appeared when the current scale<100%, or the current scale=100% switched from the scale>100%

```
property ZoomInCursor: TCursor;
```

If ZoomPercent [562]<>100%, clicking on component changes ZoomPercent [562] to 100%.

If ZoomPercent [562]=100%, clicking on component changes ZoomPercent [562] to previous scale.

Default value:

 *crRVZoomIn* (102)

(this cursor is registered automatically)

See also properties:

- ZoomOutCursor[561];
- ZoomPercent[562].

See also:

- RichView cursors[108].

1.5.3.3.1.11 TCustomRVPrintPreview.ZoomMode

Scaling mode

```
type // defined in RVStyle unit
  TRVZoomMode = (rvzmFullPage, rvzmPageWidth, rvzmCustom);
property ZoomMode : TRVZoomMode;
```

Assigning value to this property has an immediate effect.

Zoom Mode	Meaning
rvzmFullPage	Changing ZoomPercent[562] to display full page, keeping the full-page view during resizing (changing ZoomPercent[562] when needed)
rvzmPageWidth	Changing ZoomPercent[562] to fit width of page, keeping the page-width view during resizing (changing ZoomPercent[562] when needed)
rvzmCustom	Scaling according to the ZoomPercent[562] value

Default value:

rvzmCustom

1.5.3.3.1.12 TCustomRVPrintPreview.ZoomOutCursor

Cursor appeared when the current scale>100%, or the current scale=100% switched from the scale<100%

```
property ZoomOutCursor: TCursor;
```

If ZoomPercent[562]<>100%, clicking on component changes ZoomPercent[562] to 100%.

If ZoomPercent[562]=100%, clicking on component changes ZoomPercent[562] to previous scale.

Default value:

 *crRVZoomOut* (103)

(this cursor is registered automatically)

See also properties:

- ZoomInCursor[560];
- ZoomPercent[562].

See also:

- RichView cursors[108].

1.5.3.3.1.13 TCustomRVPrintPreview.ZoomPercent

Scale of preview, %

```
property ZoomPercent: Integer;
```

Although assigning to this property is possible, the preferable way to change preview scale is calling SetZoom[563] method.

This property is changed automatically when resizing the component, if ZoomMode[561] = *rvzmFullPage* or *rvzmPageWidth*.

It's recommended to keep this property in 10%–500% range.

See also methods:

- SetZoom[563].

See also properties:

- ZoomMode[561].

See also events:

- OnZoomChanged[563].

1.5.3.3.2 Methods

1.5.3.3.2.1 TCustomRVPrintPreview.Create

A constructor, creates a new TCustomRVPrintPreview object.

```
constructor Create(AOwner: TComponent);override;
```

Do not to create TCustomRVPrintPreview objects, create TRVPrintPreview[417] components instead.

1.5.3.3.2.2 TCustomRVPrintPreview.First

Switches the current page (PageNo[559] property) to the first page.

```
procedure First;
```

See also properties:

- PageNo[559].

See also methods:

- Prev[563];
- Next[563];
- Last[562].

1.5.3.3.2.3 TCustomRVPrintPreview.Last

Switches the current page (PageNo[559] property) to the last page.

```
procedure Last;
```

See also properties:

- PageNo[559];
- TRVPrintPreview.RVPrint[418].PagesCount[552].

See also methods:

- Prev[563];
- Next[563];
- First[562].

1.5.3.3.2.4 TCustomRVPrintPreview.Next

Switches the current page (PageNo[559] property) to the next page, if possible.

```
procedure Next;
```

See also properties:

- PageNo[559];
- TRVPrintPreview.RVPrint[418].PagesCount[552].

See also methods:

- Prev[563];
- First[562];
- Last[562].

1.5.3.3.2.5 TCustomRVPrintPreview.Prev

Switches the current page (PageNo[559] property) to the previous page, if possible.

```
procedure Prev;
```

See also properties:

- PageNo[559].

See also methods:

- Next[563];
- First[562];
- Last[562].

1.5.3.3.2.6 TCustomRVPrintPreview.SetZoom

Changes ZoomMode[561] to *rvzmCustom* and ZoomPercent[562] to **Percent**.

```
procedure SetZoom(Percent: Integer);
```

Calling this method has an immediate effect.

It's recommended to keep **Percent** in 10%–500% range.

See also properties:

- ZoomMode[561];
- ZoomPercent[562].

1.5.3.3.3 Events

1.5.3.3.3.1 TCustomRVPrintPreview.OnZoomChanged

Occurs when ZoomPercent[562] changes.

```
property OnZoomChanged: TNotifyEvent;
```

1.6 Item Types

TCustomRVItemInfo is a basic type for all items[118] in TRichView documents.

Detailed description of all classes for item types is beyond the scope of this help file.

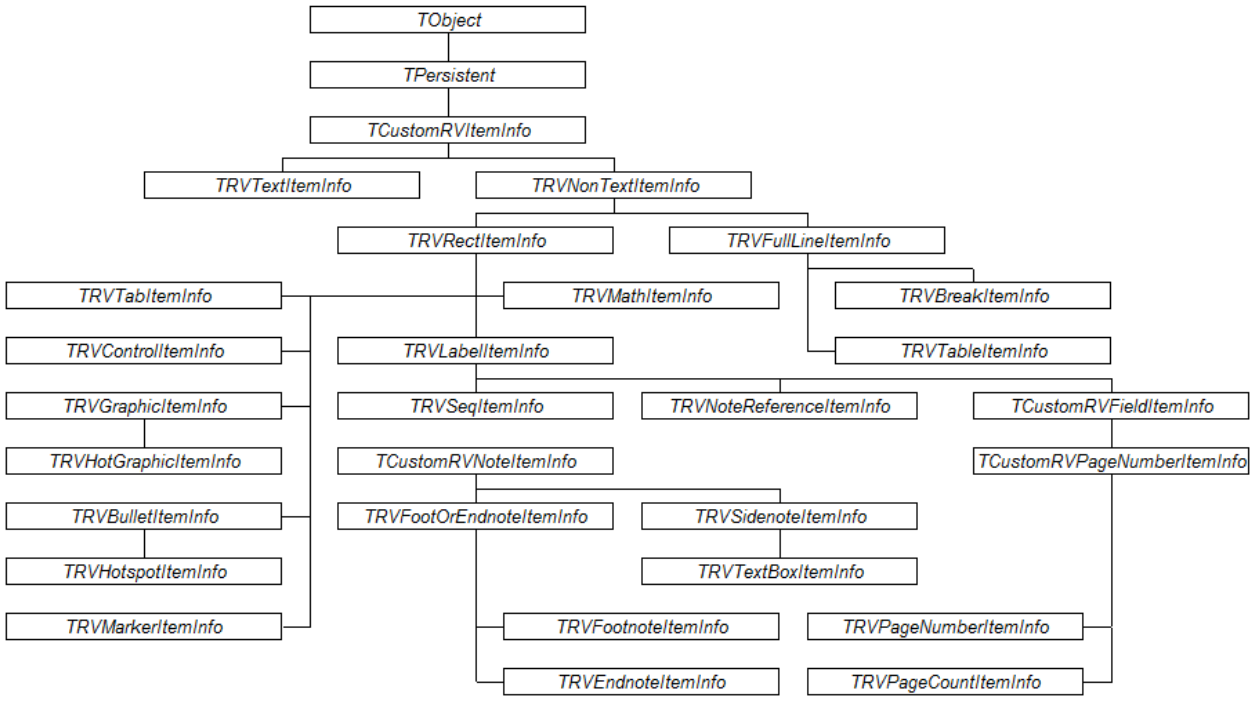
You can add, access and modify all items using methods of TRichView[162] and TRichViewEdit[316] instead of working with the item classes directly, so details of implementation are hidden from programmers. The exceptions are:

- TRVTableItemInfo[565];
- TRVLabelItemInfo[617] and classes, inherited from it.

Properties of tables must be accessed and modified directly. Properties of labels (and inherited classes) can be modified either directly or using methods of TRichView and TRichViewEdit.

Class	Unit	Meaning
TCustomRVItemInfo	RVItem	Basic class for TRichView items.
TRVTextItemInfo	RVItem	Text item 120.
TRVNonTextItemInfo	RVItem	Basic class for non-text items.
TRVRectItemInfo	RVItem	Basic class for rectangular items.
TRVFullLineItemInfo	RVItem	Basic class for items occupying the full line.
TRVTabItemInfo	RVItem	Tabulator 121.
TRVControlItemInfo	RVItem	Item containing Delphi control 126.
TRVGraphicItemInfo	RVItem	Picture 122.
TRVHotGraphicItemInfo	RVItem	*Hot-picture* 125 (picture-hyperlink).
TRVBulletItemInfo	RVItem	*Bullet* 128 (image from TImageList).
TRVHotspotItemInfo	RVItem	*Hotspot* 129 (*bullet*-hyperlink).
TRVMarkerItemInfo	RVMarker	Paragraph marker 131.
TRVTableItemInfo 565	RVTable	Table 130.
TRVBreakItemInfo	RVItem	*Break* 125 (horizontal line).
TRVLabelItemInfo 617	RVLabelItem	Label 133 (non-text item looking like text).
TRVSeqItemInfo 624	RVSeqItem	Numbered sequence 134.
TCustomRVNoteItemInfo 627	RVNote	Basic class for notes and text boxes.
TRVFootOrEndnoteItemInfo 627	RVNote	Basic class for footnotes and endnotes
TRVEndnoteItemInfo 629	RVNote	Endnote 135.
TRVFootnoteItemInfo 630	RVNote	Footnote 136.
TRVSidenoteItemInfo 632	RVSidenote	Sidenote 137
TRVTextBoxItemInfo 643	RVSidenote	Text box 139
TRVNoteReferenceItemInfo 644	RVNote	Reference to the parent note 140
TRVPageNumberItemInfo 647	RVFieldItems	"Page number" field 141
TRVPageCountItemInfo 648	RVFieldItems	"Page count" field 142
TRVMathItemInfo	RVMathItem	Mathematical expression

Hierarchy



Other Types

```
TCustomRVItemInfoClass = class of TCustomRVItemInfoClass;
```

1.6.1 Table - TRVTableItemInfo

TRVTableItemInfo is a class representing table [130] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem [201]) or inserted into TRichViewEdit (see InsertItem [353]). Style [229] of this item type: *rvsTable* (-60)

Unit RVTable.

Syntax

```
TRVTableItemInfo = class (TRVFullLineItemInfo)
```

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [563]
TRVNonTextItemInfo
TRVFullLineItemInfo

Overview

See overview topics about tables [145].

Properties

The most of properties of this class are published. These properties can be saved/loaded in RVF [93] files or streams.

The main property of TRVTableItemInfo is Cells[574]. The number of rows is RowCount[580], number of columns is ColCount[575].

Properties of rows are available via Rows[580] property.

Width of table can be defined using BestWidth[568] property.

Properties defining border around the table:

- BorderWidth[571];
- BorderStyle[570];
- BorderColor[569];
- BorderLightColor[569];
- VisibleBorders[581];
- also BorderHSpacing[569], BorderVSpacing[570].

Properties defining default border around cells:

- CellBorderWidth[572];
- CellBorderStyle[572];
- CellBorderColor[571];
- CellBorderLightColor[571].

Properties defining default colors of cells in rows and columns:

- HeadingRowColor[583], LastRowColor, FirstColumnColor, LastColumnColor, EvenColumnsColor, OddRowsColor, OddColumnsColor, EvenRowsColor

Properties defining spacing between cells:

- CellHSpacing[573];
- CellVSpacing[574];
- BorderHSpacing[569];
- BorderVSpacing[570];
- also CellHPadding[573], CellVPadding[574].

Background properties:

- Color[575];
- Opacity[577];
- BackgroundImage[567];
- BackgroundStyle[568];
- BackgroundImageFileName[568].

Border colors:

- BorderColor[569];
- BorderLightColor[569];
- CellBorderColor[571];
- CellBorderLightColor[571].

Rules:

- HRuleWidth[577];
- HRuleColor[576];
- HOutermostRule[576];
- VRuleWidth[582];
- VRuleColor[582];
- VOutermostRule[582].

Export:

- TextColSeparator[581];
- TextRowSeparator[581];
- BackgroundImageFileName[568].

Other properties:

- Options[577];
- PrintOptions[579];

- HeadingRowCount 576.

Methods

See operations on tables 153.

Events

Tables are not components, so events cannot be assigned at design time in the Object Inspector. They must be assigned from code, when the table is created.

Events cannot be saved with tables in RVF 93 files or streams. Reassign them when tables are loaded, use TRichView.OnItemAction 280 event.

Events:

- OnCellEditing 602;
- OnDrawBorder 603.

Related

- Drawing table grid 425;
- New row auto insertion 705.

1.6.1.1 Properties

1.6.1.1.1 TRVTableItemInfo.BackgroundImage

Background image for the table.

```
property BackgroundImage: TGraphic;
```

(introduced in v1.8)

Position of the image is defined by the BackgroundStyle 570 property.Transparent images are supported.

Assignment to this property copies the image, so you still need to free the source image yourself.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount 702).

Example 1 (assigning image)

```
bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
table.BackgroundImage := bmp;
bmp.Free;
```

Example 2 (clearing the image)

```
table.BackgroundImage := nil;
```

See also:

- Undo of assignments to table properties.

See also properties:

- BackgroundStyle 568;
- Color 575.

See also properties of table cell:

- BackgroundImage 605;
- BackgroundStyle 606.

See also properties of TRichView:

- BackgroundBitmap 169;
- BackgroundStyle 170.

1.6.1.1.2 TRVTableItemInfo.BackgroundImageFileName

A string storing the file name associated with BackgroundImage[567].

```
property BackgroundImageFileName: TRVUnicodeString[694];
```

(introduced in v1.9; changed in version 18)

If *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML[250]/SaveHTMLEx[251], images with defined (non-empty) file names will not be saved, but their file names will be written in HTML. This value is also accessible as extra string property[676] *rvespImageFileName*.

Assignment to this property cannot be undone by the user[88]; to make an undoable assignment, use *rvespImageFileName* property.

See also properties of table cell:

- BackgroundImageFileName[606].

1.6.1.1.3 TRVTableItemInfo.BackgroundStyle

Defines position of BackgroundImage[567].

```
property BackgroundStyle: TRVItemBackgroundStyle[682];
```

(introduced in v1.8)

See also:

- Undo of assignments to table properties.

See also properties:

- BackgroundImage[567];
- Color[575].

See also properties of table cell:

- BackgroundStyle[606];
- BackgroundImage[605].

See also properties of TRichView:

- BackgroundStyle[170];
- BackgroundBitmap[169].

1.6.1.1.4 TRVTableItemInfo.BestWidth

Preferred width for the table.

```
property BestWidth: TRVHTMLLength[680];
```

This is a preferred width for the table, measured either in Units[437] of the linked TRVStyle[418] component (pixels or twips) or in percent.

The table can be wider than this value if it is required to display its content without overlapping.

If BestWidth is undefined (=0), RichView tries to calculate the table width basing on its cells. In this version RichView can do it only if all columns have widths defined in TRVStyleUnits[691] (have at least one cell with BestWidth[607] defined in TRVStyleUnits[691], and do not have cells with BestWidth[607] defined in percents). Otherwise BestWidth=0 works like BestWidth=-100 (100%).

This property corresponds to HTML <table width> attribute.

See also undo of assignments to table properties.

Default value:

0

See also:

- Table Resizing[158];
- Undo of assignments to table properties.

See also properties of table cell:
- BestWidth[607];
- BestHeight[606].

1.6.1.1.5 TRVTableItemInfo.BorderColor

Defines color of the table border.

```
property BorderColor: TColor
```

This property defines either color of flat border or "dark" color of 3d border around the table.

See BorderStyle[570] for details.

This property corresponds to HTML <table bordercolor> or <table bordercolordark> attributes (specific to Microsoft Internet Explorer).

Default value:

clWindowText

See also:
- Undo of assignments to table properties;
- Colors and layout of tables[148] (with pictures).

See also properties
- BorderLightColor[569];
- CellBorderColor[571];
- CellBorderLightColor[571].

1.6.1.1.6 TRVTableItemInfo.BorderHSpacing

Horizontal spacing between the table border and the outermost cells.

```
property BorderHSpacing: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

BorderHSpacing be saved in HTML+CSS if:
- BorderHSpacing=CellHSpacing[573], or
- BorderHSpacing>=CellHSpacing[573] and CellHSpacing[573]>=0

Default value:

2

Note: this default value assumes that RVStyle.Units[437]=*rvstuPixels*, otherwise it could be too small.

See also:
- Undo of assignments to table propeties;
- Colors and layout of tables[148] (with pictures).

See also properties:
- BorderVSpacing[570];
- CellHSpacing[573];
- CellVSpacing[574].

1.6.1.1.7 TRVTableItemInfo.BorderLightColor

This property defines "light" color of 3d border around the table.

```
property BorderLightColor: TColor;
```

See BorderStyle[570] for details.

This property corresponds to HTML <table bordercolorlight> attribute (specific to Microsoft Internet Explorer).

Default value:

clBtnHighlight

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- BorderColor [569];
- CellBorderColor [571];
- CellBorderLightColor [571].

1.6.1.1.8 TRVTableItemInfo.BorderStyle

This property defines style of border around the table.

```
property BorderStyle: TRVTableBorderStyle [691];
```

If you want to hide table border, set BorderWidth [571] = 0.

Default value:

rvtbRaised

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- CellBorderStyle [572];
- BorderColor [569];
- BorderLightColor [569];
- BorderWidth [571];
- VisibleBorders [581].

1.6.1.1.9 TRVTableItemInfo.BorderVSpacing

Vertical spacing between border and the outermost cells.

```
property BorderVSpacing: TRVStyleLength [690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units [437] of the linked TRVStyle [418] component.

BorderVSpacing be saved in HTML+CSS if:

- BorderVSpacing=CellVSpacing [574], or
- BorderVSpacing>=CellVSpacing [574] and CellVSpacing [574]>=0

Default value:

2

Note: this default value assumes that RVStyle.Units [437]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- BorderHSpacing [569];
- CellHSpacing [573];
- CellVSpacing [574].

1.6.1.1.10 TRVTableItemInfo.BorderWidth

Width of border around the table.

```
property BorderWidth: TRVStyleLength;
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units of the linked TRVStyle component.

Set to 0 to hide border.

This property corresponds to HTML <table border>.

Default value:

0

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables (with pictures).

See also properties:

- CellBorderWidth;
- VisibleBorders.

1.6.1.1.11 TRVTableItemInfo.CellBorderColor

Defines default border color for all table cells.

```
property CellBorderColor: TColor
```

This property defines either default color of flat border or default "dark" color of 3d border around the table cells.

See CellBorderStyle for details.

This value can be overridden with BorderColor of cell.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

Default value:

clWindowText

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables (with pictures).

See also properties:

- CellBorderLightColor;
- BorderLightColor;
- BorderColor.

See also properties of table cell:

- BorderColor;
- BorderLightColor.

1.6.1.1.12 TRVTableItemInfo.CellBorderLightColor

Defines default "light" color of 3d border for all table cells.

```
property CellBorderLightColor: TColor;
```

See CellBorderStyle for details. To hide cell borders, set CellBorderWidth = 0.

This table attribute can be saved in HTML only with use of CSS (Cascading Style Sheet).

This value can be overridden with BorderLightColor of cell.

Default value:

clBtnHighlight

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- CellBorderColor [571];
- BorderLightColor [569];
- BorderColor [569].

See also properties of table cell:

- BorderColor [607];
- BorderLightColor [608].

1.6.1.1.13 TRVTableItemInfo.CellBorderStyle

Style of border around the table cells

```
property BorderStyle: TRVTableBorderStyle [691];
```

To hide cell borders, set CellBorderWidth [572] = 0.

Default value:

rvtbLowered

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- BorderStyle [570];
- CellBorderWidth [572];
- CellBorderColor [571];
- CellBorderLightColor [571].

1.6.1.1.14 TRVTableItemInfo.CellBorderWidth

Width of border around the cells of table, pixels

```
property CellBorderWidth: TRVStyleLength [690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units [437] of the linked TRVStyle [418] component.

Set to 0 to hide border.

If value of this property <> BorderWidth [571], or if it <> 1 pixel, it can be saved to HTML only with use of CSS

Default value:

0

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables [148] (with pictures).

See also properties:

- BorderWidth [571];
- CellBorderStyle [572];
- CellBorderColor [571];
- CellBorderLightColor [571].

1.6.1.1.15 TRVTableItemInfo.CellHPadding

Horizontal distance from the cell content to the cell border.

```
property CellHPadding: TRVStyleLength[690];
```

(introduced in v10)

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

This value affects all cells in the table.

If the cell is rotated[611], CellHPadding and CellVPadding[574] are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units[437]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[148] (with pictures).

See also properties:

- CellVPadding[574];
- CellPadding[574].

1.6.1.1.16 TRVTableItemInfo.CellHSpacing

Horizontal spacing between cells.

```
property CellHSpacing: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

This value can be negative. For example, you can assign CellHSpacing=-CellBorderWidth[572] to collapse cell borders horizontally.

Default value:

2

Note: this default value assumes that RVStyle.Units[437]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[148] (with pictures).

See also properties:

- CellVSpacing[574];
- BorderHSpacing[569];
- BorderVSpacing[570].

1.6.1.1.17 TRVTableItemInfo.CellOverrideColor

A color to override cell background colors.

```
property CellOverrideColor: TColor;
```

(introduced in v17)

If **CellOverrideColor**<>*clNone*, all cells are drawn using this color. Colors of individual cells[608] and colors of bands of rows and columns[583] are ignored.

Default value:

clNone

1.6.1.1.18 TRVTableItemInfo.CellPadding

Distance between cell content and cell border.

```
property CellPadding: TRVStyleLength;
```

This property is maintained for backward compatibility. Assigning to this property assigns values to CellHPadding and CellVPadding. Reading this property returns Round((CellHPadding +CellVPadding)/2).

1.6.1.1.19 TRVTableItemInfo.Cells

Provides indexed access to all cells in the table.

```
property Cells[Row, Col: Integer]: TRVTableCellData;
```

This is the key property of table.

Row can be in range from 0 to RowCount-1. **Col** can be in range from 0 to ColCount-1.

Each row has the same number of cells.

Attention: the Borland's grid uses Cells[Col,Row] instead.

Some cells can be equal to *nil* due to cell-merging.

See also properties:

- Rows.

See also:

- Table Cells Overview.

1.6.1.1.20 TRVTableItemInfo.CellVPadding

Vertical distance from the cell content to the cell border.

```
property CellVPadding: TRVStyleLength;
```

(introduced in v10)

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units of the linked TRVStyle component.

This value affects all cells in the table.

If the cell is rotated, CellHPadding and CellVPadding are not switched.

Default value:

1

Note: this default value assumes that RVStyle.Units=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables (with pictures).

See also properties:

- CellHPadding;
- CellPadding.

1.6.1.1.21 TRVTableItemInfo.CellVSpacing

Vertical spacing between cells.

```
property CellVSpacing: TRVStyleLength;
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units of the linked TRVStyle component.

This value can be negative. For example, you can assign CellVSpacing=-CellBorderWidth[572] to collapse cell borders vertically.

Default value:

2

Note: this default value assumes that RVStyle.Units[437]=*rvstuPixels*, otherwise it could be too small.

See also:

- Undo of assignments to table propeties;
- Colors and layout of tables[148] (with pictures).

See also properties:

- CellHSpacing[573],
- BorderHSpacing[569];
- BorderVSpacing[570].

1.6.1.1.22 TRVTableItemInfo.ColBandSize, Row BandSize

The properties specify the number of rows/columns in alternating row bands

```
property RowBandSize: Integer;
property ColBandSize: Integer;
```

(introduced in version 17)

RowBandSize specify the count of rows in row bands.

ColBandSize specify the count of columns in column bands.

Bands are used to implement alternate colors, see the topic about color properties of rows and columns[583].

The heading (or the first row) and the first column may be excluded from bands.

Default values

1

1.6.1.1.23 TRVTableItemInfo.ColCount

Returns the number of columns in the table.

```
property ColCount: Integer
```

(introduced in v10)

If RowCount[580]>0, this property returns Rows[580][0].Count (all rows have the same number of cells). If RowCount[580]=0, this property returns 0.

See also properties:

- Cells[574];
- Rows[580].

1.6.1.1.24 TRVTableItemInfo.Color

Background color of the table

```
property Color: TColor;
```

Set to *clNone* for transparent background. You can specify the background Opacity[577].

If the table has a background image[567], this image is drawn on top of color filling.

Default value:

clWindow

See also:

- Undo of assignments to table properties;
- Colors and layout of tables [148] (with pictures).

1.6.1.1.25 TRVTableItemInfo.HeadingRowCount

Specifies the number of rows that will be repeated on each printer page with this table.

```
property HeadingRowCount: Integer;
```

(introduced in version 1.7)

If these rows overlap next rows (due to cell merging), this property is ignored on pagination.

A special color (HeadingRowColor [583]) can be applied to heading rows (even if they overlap next rows). If **HeadingRowCount**=0, this color is applied to the first row.

Default value:

0

See also:

- PrintOptions [579],
- ColBandSize, RowBandSize [575]

1.6.1.1.26 TRVTableItemInfo.HOutermostRule

Specifies if the horizontal rules should be drawn between the table border and the outermost cells.

```
property HOutermostRule: Boolean;
```

These rules can be visible only of other rules are visible (HRuleWidth [577]>0)

Note about rules.

Default value:

False

See also:

- Undo of assignments to table properties;
- Table rules [149] (with pictures).

See also properties:

- VOutermostRule [582];
- HRuleWidth [577];
- HRuleColor [576].

1.6.1.1.27 TRVTableItemInfo.HRuleColor

Specifies the color of horizontal rules (horizontal lines between cells)

```
property HRuleColor: TColor;
```

Horizontal rules are visible only if HRuleWidth [577]>0

Note about rules.

Default value:

clWindowText

See also:

- Undo of assignments to table properties;
- Table rules [149] (with pictures).

See also properties:
- VRuleColor[582];
- HRuleWidth[577];
- HOutermostRule[576].

1.6.1.1.28 TRVTableItemInfo.HRuleWidth

Specifies the width of horizontal rules (horizontal lines between cells)

```
property HRuleWidth: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

Set to 0 to hide horizontal rules (hidden by default).

Note about rules.

Default value:

0

See also
- Undo of assignments to table properties;
- Table rules[149] (with pictures).

See also properties
- VRuleWidth[582];
- HRuleColor[576];
- HOutermostRule[576].

1.6.1.1.29 TRVTableItemInfo.Opacity

Opacity of table background.

```
property Opacity: TRVOpacity[683];
```

(introduced in version 16)

This value is used if Color[575]<>*clNone*. The opacity is applied only to background color filling, not to background image.

Default value:

100000 (i.e. 100%)

See also:
- Colors and layout of tables[148].

1.6.1.1.30 TRVTableItemInfo.Options

Specifies options for the table.

```
type
  TRVTableOption = (rvtoEditing,
    rvtoRowSizing, rvtoColSizing,
    rvtoRowSelect, rvtoColSelect,
    rvtoNoCellSelect,
    rvtoRTFSaveCellPixelBestWidth,
    rvtoHideGridLines,
    rvtoCellBelowBorders,
    rvtoOverlappingCorners,
    rvtoRTFAllowAutofit,
    rvtoIgnoreContentWidth,
```

```
    rvtoIgnoreContentHeight);
  TRVTableOptions = set of TRVTableOption;

property Options: TRVTableOptions;
```

Default value

[*rvtoEditing, rvtoRowSizing, rvtoColSizing, rvtoRowSelect, rvtoColSelect*]

Options for Editing

Option	Meaning
rvtoEditing	If set (default), users can edit content of cells using inplace editor. (table must be inserted in TRichViewEdit[316] or TDBRichViewEdit[410])

Options for Resizing and Selection

Option	Meaning
rvtoRowSizing *rvtoColSizing*	If set (default), users can resize individual rows/columns with the mouse. (table must be inserted in TRichViewEdit[316] or TDBRichViewEdit[410])
rvtoRowSelect, *rvtoColSelect*	If set (default), user can select rows/columns by clicking on the left/top border of the table (when cursor has a special vertical/horizontal arrow shape) The options have no effect if the table is inserted in RichView with *rvoAllowSelection* excluded from Options[182].
rvtoNoCellSelect	If set, multicell selection by mouse is not allowed (remove also *rvtoRowSelect* and *rvtoColSelect* to disallow making multicell selection completely.

Layout Options

Option	Meaning
rvtoIgnoreContentWidth	If set, contents of cells are completely ignored when calculating widths of table columns.Wide pictures, controls, etc. are clipped.
rvtoIgnoreContentHeight	If set, heights of rows are calculated basing only on BestHeight[606] properties of cells. But (unlike *rvtoIgnoreContentWidth*), heights of cells having BestHeight=0 are still calculated basing on their content.

Visual Options

Option	Meaning

rvtoHideGridLines	If not set (default), invisible borders are shown using GridColor, GridReadOnlyColor, GridStyle, GridReadOnlyStyle[425] properties of the linked TRVStyle. It's not recommended to use this option. Use rvoShowGridLines option in TRichView.Options[182] instead.
rvtoCellBelowBorders	If not set (default), space below invisible side[613] of border has color[575] of tables. If set, color of cell[608] is used. 
rvtoOverlappingCorners	If not set (default), colored borders (CellBorderStyle[572] =*rvtbColor*) have three-dimensional appearance. If not set, they are flat. The effect can be noticed for wide borders with some invisible sides[613]. 

RTF and DocX Export Options

Option	Meaning
rvtoRTFAllowAutofit	If set, tables with default width (BestWidth[568]=0) are exported to RTF and DocX as fit-by-content tables. Off by default. See also: RichViewTableDefaultRTFAutofit[705]
rvtoRTFSaveCellPixelBestWidth	If set, exact sizes of cells are saved to RTF and DocX. Off by default.

The options above are explained in the topic about table RTF and DocX export[157].

1.6.1.1.31 TRVTableItemInfo.PrintOptions

Specifies options for table printing.

```
type
  TRVTablePrintOption =
    (rvtoHalftoneBorders, rvtoRowsSplit,
     rvtoWhiteBackground, rvtoContinue);
  TRVTablePrintOptions = set of TRVTablePrintOption;

property PrintOptions: TRVTablePrintOptions;
```

Option	Meaning
rvtoHalftoneBorders	If set (default), table and cells borders will be printed in dithered colors, but corners will be rounded. If cleared, borders will be printed in solid colors (usually black or white), but corners will be strictly defined, like on screen. Affects to 3d borders (see BorderStyle[570] and CellBorderStyle[572]) and borders with some invisible lines. Forced compromise.
rvtoRowsSplit	Clear this option to forbid printing this table on several pages.
rvtoWhiteBackground	If set, colors of table and cells backgound will be ignored and printed in white. See also TRVPrint.TransparentBackground[553]
rvtoContinue	If set, and the table has heading rows[576], heading rows are displayed only from the second table page

Default value:

[*rvtoHalftoneBorders*,*rvtoRowsSplit*]

See also:

- HeadingRowCount[576].

1.6.1.1.32 TRVTableItemInfo.RowCount

Returns the number of rows in the table.

```
property RowCount: Integer
```

(introduced in v10)

See also properties:

- ColCount[575];
- Cells[574];
- Rows[580].

1.6.1.1.33 TRVTableItemInfo.Rows

Provides indexed access to all rows in the table.

```
property Rows: TRVTableRows[616];
```

Rows is a list of table rows (TRVTableRow[615]), each row is a list of cells (TRVTableCellData[605]). Each row has the same number of cells.

The specific cell can be accessed as Rows[<Row>][<Col>] as well as Cells[574][<Row>,<Col>].

Use Rows to access properties of rows, for example Rows[<Row>].VAlign[616].Otherwise, Cells[574] is more convenient.

See also:

- Table Cells Overview[146]

1.6.1.1.34 TRVTableItemInfo.TextColSeparator

Value of this property is used when saving table in text file. This string will be written between cells.

```
property TextColSeparator: TRVUnicodeString[694];
```

(changed in version 18)

Default value

#13#10 (CR+LF)

You can set TextColSeparator to TAB, if you wish, but do not forget that there can be CR+LFs inside cells.

See also properties:

- TextRowSeparator[581].

See also methods of TRichView:

- SaveText[255];
- GetSelText[238].

See also:

- Export of Tables[157].

1.6.1.1.35 TRVTableItemInfo.TextRow Separator

Value of this property is used when saving table in text file. This string will be written between rows.

```
property TextRowSeparator: TRVUnicodeString[694];
```

(changed in version 18)

Default value:

#13#10 (CR+LF)

See also properties:

- TextColSeparator[581].

See also methods of TRichView:

- SaveText[255];
- GetSelText[238].

See also:

- Export of Tables[157].

1.6.1.1.36 TRVTableItemInfo.VisibleBorders

Allows to hide some sides of table border.

```
property VisibleBorders: TRVBooleanRect[653];
```

(introduced in v10)

Direct assignment to this property cannot be undone/redone by user (use SetTableVisibleBorders[600])

Default value:

True, *True*, *True*, *True* (all sides are visible)

See also properties:

- BorderWidth[571];
- BorderColor[569];
- BorderLightColor[569];
- BorderStyle[570].

See also properties of table cell:

- VisibleBorders[613].

1.6.1.1.37 TRVTableItemInfo.VOutermostRule

Specifies if the vertical rules should be drawn between the table border and the outermost cells.

```
property VOutermostRule: Boolean;
```

These rules can be visible only of other rules are visible (VRuleWidth[582]>0)

Note about rules.

Default value:

False

See also:

- Undo of assignments to table properties;
- Table rules[149] (with pictures).

See also properties:

- HOutermostRule[576];
- VRuleWidth[582];
- VRuleColor[582].

1.6.1.1.38 TRVTableItemInfo.VRuleColor

Specifies the color of vertical rules (vertical lines between cells)

```
property VRuleColor: TColor;
```

Vertical rules are visible only if VRuleWidth[582]>0

Note about rules.

Default value:

clWindowText

See also:

- Undo of assignments to table properties;
- Table rules[149] (with pictures).

See also properties:

- HRuleColor[576];
- VRuleWidth[582];
- VOutermostRule[582].

1.6.1.1.39 TRVTableItemInfo.VRuleWidth

Specifies the width of vertical rules (vertical lines between cells)

```
property VRuleWidth: TRVStyleLength[690];
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

Set to 0 to hide vertical rules (hidden by default).

Note about rules.

Default value:

0

See also:

- Undo of assignments to table properties;
- Table rules 149 (with pictures).

See also properties:

- HRuleWidth 577;
- VRuleColor 582;
- VOutermostRule 582.

1.6.1.1.40 TRVTableItemInfo'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;
```

(introduced in version 17)

By default, all cells are transparent, and you can see the table background through cells (see Color 575 property).

Using the properties listed in this topic, you can define default colors of cells in the corresponding rows and columns.

Finally, you can define the color of the specified cell in Cell.Color 608 property.

Property	Meaning
HeadingRowColor	Color of heading rows 576. If this table has no heading rows, but **HeadingRowColor**<>*clNone*, this color is used for the first row.
LastRowColor	Color of the last row
OddRowsColor	Color of odd row bands
EvenRowsColor	Color of even row bands
FirstColumnColor	Color of the first column
LastColumnColor	Color of the last column
OddColumnsColor	Color of odd row bands
EvenColumnsColor	Color of even row bands

if **FirstColumnColor**<>*clNone*, column bands are started from the second column, otherwise from the first column. Each band has ColBandSize 575 columns.

Row bands are started from the first column after the table heading rows. If HeadingRowCount 576>0, it defines the number of heading rows. Otherwise, if HeadingRowColor<>clNone, one heading row is assumed. Otherwise, row bands start from the first row. Each band has RowBandSize 575 rows.

Colors have the following priorities (in descending order) : HeadingRowColor, LastRowColor, FirstColumnColor, LastColumnColor, OddColumnsColor and EvenColumnsColor, OddRowsColor and EvenRowsColor. Only properties having values different from *clNone* are taken into account.

Default values

clNone

See also

- Color 575
- CellOverrideColor 573;
- HeadingRowCount 576
- ColBandSize, RowBandSize 575
- Cell.Color 608
- Cell.Opacity 609

1.6.1.2 Methods

1.6.1.2.1 TRVTableItemInfo.AssignProperties

Assigns all table properties from the **Source**.

```
procedure AssignProperties(Source: TRVTableItemInfo565);
```

(introduced in v1.9)

Cells and their contents are not assigned.

1.6.1.2.2 TRVTableItemInfo.CanMergeCells

Returns: is cell-merging of the specified cells possible?

```
function CanMergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer;
  AllowMergeRC: Boolean): Boolean;
```

Potential area for cell-merging is specified in **TopRow**, **TopCol**, **ColSpan**, **RowSpan**.

TopRow, **TopCol** – position of the top left cell for cell-merging.

ColSpan – number of columns for merging.

RowSpan – number of rows for merging.

If merging will be performed, the resulting cell will be Cells 574[**TopRow**, **TopCol**] with ColSpan 608=**ColSpan** and RowSpan 612=**RowSpan**.

Merging can be done, if

1. Cells 574[**TopRow**, **TopCol**]<>*nil*.
2. This area defines a rectangular piece of table (there are no cells overlapping its border due to previous cell-merging). This requirement covers the 1st one.
3. Either Cells 574[**TopRow**, **TopCol**].ColSpan 608 < **ColSpan** or Cells 574[**TopRow**, **TopCol**].RowSpan 612 < **RowSpan**.
4. If **AllowMergeRC**=*True*, this function returns *False*, if one or more rows or columns will contain only *nil*s after merging. It's not recommended to have such rows/columns, so you can either forbid such operation or (better) delete them with DeleteEmptyRows 587 and DeleteEmptyCols 587.

See also:

- CanMergeSelectedCells 585;
- MergeCells 594;
- MergeSelectedCells 594.

See also properties:

- ColCount 575;
- RowCount 580.

1.6.1.2.3 TRVTableItemInfo.CanMergeSelectedCells

Returns: is cell-merging of the selected cells possible?

```
function CanMergeSelectedCells(AllowMergeRC: Boolean): Boolean;
```

The same as CanMergeCells 584, but potential area for cell-merging is the selection in the table.

If **AllowMergeRC**=*True*, this function returns *False*, if one or more rows or columns will contain only *nil*s after merging. It's not recommended to have such rows/columns, so you can either forbid such operation or (better) delete them with DeleteEmptyRows 587 and DeleteEmptyCols 587.

This method can be used to determine if the selection in the table has a rectangular shape.

This method must be called when the document is formatted.

See also:

- CanMergeCells 584;
- MergeSelectedCells 594;
- MergeCells 594.

1.6.1.2.4 TRVTableItemInfo.Changed

"Informs" table about updates in its content.

```
procedure Changed;
```

After performing operations on table which was already inserted in document you should reformat this document.

It is usually done by BeginItemModify 335/EndItemModify 341 (if the table was inserted in TRichViewEdit 316) or Format 217 (if the table was inserted in TRichView 162).

But you can realize that in some cases table will not be updated even after formatting. This happens because the table does not "know" about changes in it, so it reuses the previous formatting.

The most of operations on table (including all methods of TRVTableItemInfo, and editing operations in cell inplace-editor) "inform" table about changes in its content and thus allowing subsequent reformatting. But viewer-style operations performed on cells do not "inform" table, and their result cannot be shown after reformatting. You can inform table about changes in its content by this method.

Example

```
table := TRVTableItemInfo(MyRichView.GetItem(ItemNo));
table.Cells[0,0].Clear;
table.Cells[0,0].AddNL('One more line',0,0);
table.Changed;
MyRichView.Format;
```

Note: since version 1.7, it's not required to call table.Changed before Format 217. But it is still useful when using BeginItemModify 335/EndItemModify 341.

1.6.1.2.5 TRVTableItemInfo.Create

Creates a new TRVTableItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData 187 property of TRichView 162 or TRichViewEdit 316, where you wish to insert this table.

This constructor creates a new table with one row and one column.

There is more convenient constructor, CreateEx[586].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.1.2.6 TRVTableItemInfo.CreateEx

Creates a new TRVTableItemInfo object having **nRows** rows and **nCols** columns.

```
constructor CreateEx(nRows, nCols: Integer; AMainRVData: TCustomRVData);
```

Parameters:

nRows – count of rows.

nCols – count of columns.

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this table.

If **nRows**=0, then **nCols** is ignored, and 0x0 table is created. A number of columns is stored only if table.RowCount[580]>0.

Example

```
var table:TRVTableItemInfo;
...
table := TRVTableItemInfo.CreateEx(5, 5, MyRichViewEdit.RVData[187]);
table.Color[575] := clYellow;
table.Cells[574][0,0].AddNL[614]('Hello',0,0);
MyRichViewEdit.InsertItem[353]('5x5 table', table);
```

See also methods:

- Create[585].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.1.2.7 TRVTableItemInfo.DeleteCols

Deletes **Count** columns starting from the **Index**-th column.

```
procedure DeleteCols(Index, Count: Integer; DecreaseWidth: Boolean;
  EditCellAfter: Boolean = True);
```

Parameters:

Index – index of the first column to delete, in range 0..ColCount[575]-1.

Count – number of columns to delete.

If **DecreaseWidth** = *True*, BestWidth[607]s of cells overlapping the deleted columns (due to cell-merging) will be decreased (recommended).

If **EditCellAfter** = *True*, and this table is inserted in an editor, the edited cell remains edited after this operation; or, if no cell was edited, or the edited cell was deleted, an appropriate cell in the next column becomes edited. If **EditCellAfter** = *False*, no cell is edited after this operation.

See also:

- Undo note.

See also methods:

- DeleteSelectedCols 588;
- DeleteRows 587;
- DeleteEmptyCols 587.

1.6.1.2.8 TRVTableItemInfo.DeleteEmptyCols

Deletes all columns containing only *nil*-cells.

```
procedure DeleteEmptyCols;
```

Nil-cells occur due to cell-meging.

All-*nil* columns occur after cell-merging or deleting rows.

This method calls DeleteCols 586 with *DecreaseWidth* = *False*.

See also:

- Undo note.

See also methods:

- DeleteEmptyRows 587;
- DeleteSelectedCols 588;
- DeleteCols 586.

1.6.1.2.9 TRVTableItemInfo.DeleteEmptyRows

Deletes all rows containing only *nil*-cells.

```
procedure DeleteEmptyRows;
```

Nil-cells occur due to cell-meging.

All-nil rows occur after cell-merging or deleting columns.

This method calls DeleteRows 587 with DecreaseHeight = *False*.

See also:

- Undo note.

See also:

- DeleteEmptyCols 587;
- DeleteSelectedRows 588;
- DeleteRows 587.

1.6.1.2.10 TRVTableItemInfo.DeleteRows

Deletes **Count** rows starting from the **Index**-th row.

```
procedure DeleteRows(Index, Count: Integer; DecreaseHeight: Boolean;
  EditCellAfter: Boolean = True);
```

Parameters:

Index – index of the first row to delete, in range 0..RowCount 580-1.

Count – number of rows to delete.

if **DecreaseHeight** = *True*, BestHeight 606s of cells overlapping the deleted rows (due to cell-merging) will be decreased (recommended).

If **EditCellAfter** = *True*, and this table is inserted in an editor, the edited cell remains edited after this operation; or, if no cell was edited, or the edited cell was deleted, an appropriate cell in the next row becomes edited. If **EditCellAfter** = *False*, no cell is edited after this operation.

See also:

- Undo note.

See also methods:

- DeleteSelectedRows 588;
- DeleteCols 586;
- DeleteEmptyRows 587.

1.6.1.2.11 TRVTableItemInfo.DeleteSelectedCols

Deletes all columns containing selected cells

```
procedure DeleteSelectedCols;
```

This method deletes all columns containing the selected cells, even if not all cells in these columns are selected.

The cell that is being edited is considered selected.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- DeleteSelectedRows 588;
- DeleteCols 586;
- DeleteEmptyCols 587.

1.6.1.2.12 TRVTableItemInfo.DeleteSelectedRows

Deletes all rows containing selected cells.

```
procedure DeleteSelectedRows;
```

This method deletes all rows containing selected cells, even if not all cells in these rows are selected.

The cell that is being edited is considered selected.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- DeleteSelectedCols 588;
- DeleteCols 586;
- DeleteEmptyCols 587.

1.6.1.2.13 TRVTableItemInfo.Deselect

Removes selection from the table.

```
procedure Deselect;
```

All selected cells become not selected.

See also:

- Overview of selection in tables 150.

See also methods:

- Select 595;
- SelectRows 596;
- SelectCols 595.

1.6.1.2.14 TRVTableItemInfo.Destroy

Destroys an instance of TRVTableItemInfo.

```
destructor Destroy; override;
```

Do not call Destroy directly in an application. Tables are freed by the control where they were inserted.

1.6.1.2.15 TRVTableItemInfo.EditCell

Activates editing of the specified cell..

```
procedure EditCell(Row,Col: Integer);
```

Parameters:

Row and **Col** defines the position of cell to edit. **Row** must be in range 0..RowCount 580-1, **Col** must be in range 0..ColCount 575-1.

Cells 589[**Row**, **Col**] must not be *nil*.

This method creates inplace-editor for this cell.

If this table is nested in another table, this method also initializes inplace-editor in the parent table.

This is an equivalent for Cells 589[Row, Col].Edit 651.

This method must be called when the document is formatted.

See also:

- Overview of cell editing 151.

See also methods:

- GetNormalizedSelectionBounds 590;
- GetEditedCell 590.

1.6.1.2.16 TRVTableItemInfo.GetCellAt

Returns cell at the specified coordinates.

```
function GetCellAt(X,Y: Integer; var Row,Col: Integer): Boolean;
```

Input parameters:

X, **Y** – coordinates, relative to the top left corner of the table.

Output parameters:

Row, **Col** – receive the position of cell.

This method must be called when the document is formatted.

See also methods of TCustomRichView:

- GetItemCoords 226.

1.6.1.2.17 TRVTableItemInfo.GetEditedCell

Returns inplace editor for the currently edited cell, or *nil* if there is no such cell.

```
function GetEditedCell(var Row,Col: Integer): TCustomRichViewEdit;
```

Output parameters:

Row and **Col** receive position of the cell that is being edited.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables;
- Overview of cell editing.

See also methods:

- GetNormalizedSelectionBounds;
- EditCell.

1.6.1.2.18 TRVTableItemInfo.GetNormalizedSelectionBounds

Returns selection in table in convenient form.

```
function GetNormalizedSelectionBounds(IncludeEditedCell: Boolean;
         var TopRow, LeftCol, ColSpan, RowSpan: Integer): Boolean;
```

Input parameter:

If *IncludeEditedCell* = *True*, this function returns position of the currently edited cell (if there is no selection and some cell is currently in editing state).

Output parameters:

TopRow, **TopCol** – receive the top left corner of the selection (warning: selection is not always rectangular!)

ColSpan, **RowSpan** – how many columns and rows the selection includes. Both values are >= 1.

Warning: it will be wrong to iterate through selected cells as:

```
for r := TopRow to  TopRow+RowSpan-1 do
  for c := LeftCol to TopCol+ColSpan-1 do
    if table.Cells[r,c]<>nil then ...
```

This cycle can miss some selected cells, because there can be some cells to the left of **LeftCol** and to the top of **TopRow**, which are also in selection because of spanning.

The correct example is in the topic about IsCellSelected.

Some operations are possible only on rectangular selection (for example, cell merging). It can be checked using CanMergeSelectedCells.

This method must be called when the document is formatted.

Return value:

True, if selection exists, *False* otherwise.

See also:

- Overview of selection in tables

See also methods:

- GetSelectionBounds;
- GetEditedCell.

1.6.1.2.19 TRVTableItemInfo.GetSelectionBounds

Returns selection in table.

```
function GetSelectionBounds(var StartRow, StartCol, RowOffs, ColOffs: Integer): Boolean;
```

Output parameters:

Cells[574][**StartRow**, **StartCol**] is a cell where selection is started. Selection always includes this cell.

if **RowOffs**>0, the selection includes **RowOffs** rows below. If **RowOffs**<0, the selection includes abs(**RowOffs**) rows above.

The same is for **ColOffs**.

Selection includes all cells intersected with imaginary rectangle that fully includes Cells[**StartRow**, **StartCol**] and Cells[**StartRow**+**RowOffs**, **StartCol**+**ColOffs**]. Warning: these cells can be *nil* because of merging.

For example, the returned value (3,3,0,0) means that only one cell is selected: Cells[3,3].

Selection can have a rectangular shape or not (because of merging). Some operations are possible only on rectangular selection (for example, cell merging). It can be checked using CanMergeSelectedCells[585].

This method returns selection as it is stored internally. For practical use, GetNormalizedSelectionBounds[590] is more convenient.

This method must be called when the document is formatted.

Return value:

True, if selection exists, *False* otherwise.

See also:

- Overview of selection in tables[150].

See also methods:

- GetNormalizedSelectionBounds[590];
- Select[595];
- Deselect[588].

1.6.1.2.20 TRVTableItemInfo.InsertCols

Inserts **Count** columns in the position specified by **Index**.

```
procedure InsertCols(Index, Count, CopyIndex: Integer);
```

Parameters:

Index – index of the column to insert before (or ColCount[575] to insert after the last column).

Count – number of columns to insert.

CopyIndex – if <>-1, this is an index (before the insertion) of column to copy attributes for new columns from. Colors, background, border, size, text and paragraph styles are copied.

See also:

- Undo note.

See also methods:

- InsertRows[592];
- InsertColsLeft[592], InsertColsRight[592];
- DeleteCols[586].

1.6.1.2.21 TRVTableItemInfo.InsertColsLeft

Inserts **Count** columns to the right of the selected cells (or the cell being edited)

```
procedure InsertColsLeft(Count: Integer);
```

Does nothing if the table has no selection or editing cell.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- InsertColsRight 592;
- InsertCols 591;
- InsertRowsAbove 593, InsertRowsBelow 593;
- DeleteSelectedCols 588.

1.6.1.2.22 TRVTableItemInfo.InsertColsRight

Inserts **Count** columns to the right of the selected cells (or the cell being edited)

```
procedure InsertColsRight(Count: Integer);
```

Does nothing if the table has no selection or editing cell.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- InsertColsLeft 592;
- InsertCols 591;
- InsertRowsAbove 593, InsertRowsBelow 593;
- DeleteSelectedCols 588.

1.6.1.2.23 TRVTableItemInfo.InsertRows

Inserts **Count** rows in the position specified by **Index**.

```
procedure InsertRows(Index, Count, CopyIndex: Integer);
```

Parameters:

Index – index of the row to insert before (or RowCount 580 to insert after the last row).

Count – number of rows to insert.

CopyIndex – if <>-1, this is an index (before the insertion) of row to copy attributes for new rows from. Colors, background, border, size, text and paragraph styles are copied.

See also:

- Undo note.

See also methods:

- InsertCols 591;
- InsertRowsAbove 593, InsertRowsBelow 593;
- DeleteRows 587.

1.6.1.2.24 TRVTableItemInfo.InsertRowsAbove

Inserts **Count** rows above the selected cells (or the cell being edited)

```
procedure InsertRowsAbove(Count: Integer);
```

Does nothing if the table has no selection or editing cell.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- InsertRowsBelow[593];
- InsertRows[592];
- InsertColsLeft[592], InsertColsRight[592];
- DeleteSelectedRows[588].

1.6.1.2.25 TRVTableItemInfo.InsertRowsBelow

Inserts **Count** rows below the selected cells (or the cell being edited)

```
procedure InsertRowsBelow(Count: Integer);
```

Does nothing if the table has no selection or editing cell.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- InsertRowsAbove[593];
- InsertRows[592];
- InsertColsLeft[592], InsertColsRight[592];
- DeleteSelectedRows[588].

1.6.1.2.26 TRVTableItemInfo.IsCellSelected

Returns: is the specified cell selected or not.

```
function IsCellSelected(Row, Col: Integer): Boolean;
```

Example: iteration through all selected cells

```
for r := 0 to table.RowCount[580]-1 do
  for c := 0 to table.ColCount[575]-1 do
    if (table.Cells[574][r,c]<>nil) and
       table.IsCellSelected(r,c) then ...
```

This method must be called when the document is formatted.

See also:

- Overview of selection in tables[150].

See also methods:

- GetNormalizedSelectionBounds[590].

1.6.1.2.27 TRVTableItemInfo.LoadFromStream

Loads table from **Stream**.

```
procedure LoadFromStream(Stream: TStream);
```

Table must be saved using SaveToStream[595] or SaveRowsToStream[595] methods. Table can be loaded correctly only if collections of text, paragraph and list styles in TRVStyle[418] are the same as at the moment of saving.

Applications rarely, if ever, call LoadFromStream directly. Usually tables are loaded from RVF files or streams [93] together with main documents.

This method must be used in conjunction with BeforeLoading and AfterLoading. AfterLoading must be called when the table is inserted in the document.

```
Table.BeforeLoading(rvlfRVF);
Table.LoadFromStream(Stream);
if RichViewEdit1.InsertItem[353]('', Table) then
  Table.AfterLoading(rvlfRVF);
```

1.6.1.2.28 TRVTableItemInfo.MergeCells

Merges the specified cells in one.

```
procedure MergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer; AllowMergeRC: Boolean);
```

Parameters:

TopRow, **LeftCol** – position of the top left cell for cell-merging.

ColSpan – number of columns for merging.

RowSpan – number of rows for merging.

The resulting cell is Cells [574][**TopRow**, **TopCol**] having ColSpan [608]=**ColSpan** and RowSpan [612]=**RowSpan**.

This method fails (does nothing) if the parameters do not define a rectangle (because of previous cell merging). You can check the possibility of merging using CanMergeCells [584].

AllowMergeRC = *False* forbids to perform merging, if it will cause appearing rows or columns containing only *nils*. It's not recommended to have such rows/columns, so you can either forbid such operation or to delete them using DeleteEmptyRows [587] and DeleteEmptyCols [587] (the latter is recommended).

See also:

- Undo note.

See also methods:

- CanMergeCells [584];
- MergeSelectedCells [594];
- GetNormalizedSelectionBounds [590];
- UnmergeCells [601].

1.6.1.2.29 TRVTableItemInfo.MergeSelectedCells

Merges the selected cells.

```
procedure MergeSelectedCells(AllowMergeRC: Boolean);
```

This method fails (does nothing) if the selection does not have a rectangular shape (because of previous cell merging). You can check possibility of merging using CanMergeSelectedCells [585].

AllowMergeRC = *False* forbids to perform merging, if it will cause appearing rows or columns containing only *nils*. It's not recommended to have such rows/columns, so you can either forbid such operation or delete them with DeleteEmptyRows [587] and DeleteEmptyCols [587] (the latter is recommended).

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- CanMergeSelectedCells [585];
- MergeCells [594];
- GetNormalizedSelectionBounds [590];
- UnmergeSelectedCells [601].

1.6.1.2.30 TRVTableItemInfo.SaveRowsToStream

Saves the specified rows of table (from **Index** to **Index**+**Count**-1) to **Stream**.

```
procedure SaveRowsToStream(Stream: TStream; Index, Count: Integer);
```

(introduced in version 10)

The output is compatible with SaveToStream[595].

The result will be correct even if there are cells that overlap these rows because of RowSpan[612]>1.

See also method:

- LoadFromStream[593].

See also:

- Example how to divide table in two parts[160].

1.6.1.2.31 TRVTableItemInfo.SaveToStream

Saves table to **Stream**.

```
procedure SaveToStream(Stream: TStream);
```

Applications rarely, if ever, call SaveToStream directly. Usually tables are saved to RVF files or streams[93] with main documents.

See also method:

- LoadFromStream[593];
- SaveRowsToStream[595].

1.6.1.2.32 TRVTableItemInfo.Select

Selects the range of cells in the table

```
procedure Select(StartRow, StartCol, RowOffs, ColOffs: Integer);
```

This method selects range (rectangle, if there is no cells-merging) of cells, starting from the cell (**StartRow**, **StartCol**). The parameters have the same meaning as in GetSelectionBounds[591].

Selection includes all cells intersected with imaginary rectangle that fully includes Cells[574][**StartRow**, **StartCol**] and Cells[574][**StartRow**+**RowOffs**, **StartCol**+**ColOffs**].

You cannot remove selection with this method, because Select(StartRow, StartCol, 0,0) selects one cell, use Deselect[588].

This method must be called when the document is formatted.

See also:

- Overview of selection in tables[150]

See also methods:

- SelectRows[596], SelectCols[595];
- GetSelectionBounds[591];
- Deselect[588].

1.6.1.2.33 TRVTableItemInfo.SelectCols

Selects **Count** columns starting from **StartCol** (to the right)

```
procedure SelectCols(StartCol, Count: Integer);
```

Parameters:

StartCol – index of the first column to select, in range 0..ColCount[575]-1.

Count – number of columns to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables 150

See also methods:

- SelectRows 596,
- Select 595.

1.6.1.2.34 TRVTableItemInfo.SelectRows

Selects **Count** rows starting from **StartRow** (down)

```
procedure SelectRows(StartRow, Count: Integer);
```

Parameters:

StartRow – index of the first row to select, in range 0..RowCount 580-1.

Count – number of rows to select.

This method must be called when the document is formatted.

See also:

- Overview of selection in tables 150

See also methods:

- SelectCols 595;
- Select 595.

1.6.1.2.35 TRVTableItemInfo.SetCellBackgroundImage

Assigns **Value** to Cells 574[**Row**, **Col**].BackgroundImage 605, as an editing operation.

```
procedure SetCellBackgroundImage(Value: TGraphic; Row,Col: Integer);
```

Unlike a direct assignment to Cells 574[**Row**, **Col**].BackgroundImage 605 property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion 353).

Image in **Value** is copied, so you still need to free memory for this it.

Use this method in conjunction with methods for updating editor window, see example 156.

See also:

- Undo/redo in tables 155;
- Undo/redo in TRichViewEdit 88.

1.6.1.2.36 TRVTableItemInfo.SetCellBackgroundImageFileName

Assigns **Value** to Cells 574[**Row**, **Col**].BackgroundImageFileName 606, as an editing operation.

```
procedure SetCellBackgroundImageFileName(const Value: TRVUnicodeString 694; Row,Col: Integer)
```

(changed in version 18)

Unlike a direct assignment to Cells 574[**Row**, **Col**].BackgroundImageFileName 606 property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion 353).

Use this method in conjunction with methods for updating editor window, see example 156.

See also:

- Undo/redo in tables 155;
- Undo/redo in TRichViewEdit 88.

1.6.1.2.37 TRVTableItemInfo.SetCellBackgroundStyle

Assigns **Value** to Cells[574][**Row**, **Col**].BackgroundStyle[606], as an editing operation.

```
procedure SetCellBackgroundStyle(Value: TRVItemBackgroundStyle[682]; Row,Col: Integer);
```

Unlike a direct assignment to Cells[574][**Row**, **Col**].BackgroundStyle[606] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156]

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.38 TRVTableItemInfo.SetCellBestHeight

Assigns **Value** to Cells[574][**Row**, **Col**].BestHeight[606], as an editing operation.

```
procedure SetCellBestHeight(Value: TRVStyleLength[690]; Row,Col: Integer);
```

Unlike a direct assignment to Cells[574][**Row**, **Col**].BestHeight[606] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156]

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.39 TRVTableItemInfo.SetCellBestWidth

Assigns **Value** to Cells[574][**Row**, **Col**].BestWidth[607], as an editing operation.

```
procedure SetCellBestWidth(Value: Integer; Row,Col: Integer);
```

Unlike a direct assignment to Cells[574][**Row**, **Col**].BestWidth[607] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.40 TRVTableItemInfo.SetCellBorderColor

Assigns **Value** to Cells[574][**Row**, **Col**].BorderColor[607], as an editing operation.

```
procedure SetCellBorderColor(Value: TColor; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells[574][**Row**, **Col**].BorderColor[607] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.41 TRVTableItemInfo.SetCellBorderLightColor

Assigns **Value** to Cells[574][**Row**, **Col**].BorderLightColor[608], as an editing operation.

```
procedure SetCellBorderLightColor(Value: TColor; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells[574][**Row**, **Col**].BorderLightColor[608] property, this method can be undone/redone, if it was called for table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.42 TRVTableItemInfo.SetCellColor

Assigns **Value** to Cells[574][**Row**, **Col**].Color[608], as an editing operation.

```
procedure SetCellColor(Value: TColor; Row,Col: Integer);
```

Unlike a direct assignment to Cells[574][**Row**, **Col**].Color[608] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.43 TRVTableItemInfo.SetCellHint

Assigns **Value** to Cells[574][**Row**, **Col**].Hint[609], as an editing operation.

```
procedure SetCellHint(const Value: String; Row,Col: Integer);
```

Unlike a direct assignment to Cells[574][**Row**, **Col**].Hint[609] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.44 TRVTableItemInfo.SetCellOpacity

Assigns **Value** to Cells[574][**Row**, **Col**].Opacity[609], as an editing operation.

```
procedure SetCellOpacity(Value: TRVOpacity[683]; Row,Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells[574][**Row**, **Col**].Opacity[609] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.45 TRVTableItemInfo.SetCellOptions

Assigns **Value** to Cells[574][**Row**, **Col**].Options[609], as an editing operation.

```
procedure SetCellOpacity(Value: TRVTableCellOptions[609]; Row,Col: Integer);
```

(introduced in version 16)

Unlike a direct assignment to Cells[574][**Row**, **Col**].Options[609] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.46 TRVTableItemInfo.SetCellRotation

Assigns **Value** to Cells[574][**Row**, **Col**].Rotation[611], as an editing operation.

```
procedure SetCellRotation(Value: TRVDocRotation[672]; Row,Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells[574][**Row**, **Col**].Rotation[611] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.47 TRVTableItemInfo.SetCellTag

Assigns **Value** to Cells[574][**Row**, **Col**].Tag[612], as an editing operation.

```
procedure SetCellVAlign(Value: Tag[612]; Row,Col: Integer);
```

(introduced in version 14)

Unlike a direct assignment to Cells[574][**Row**, **Col**].Tag[612] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88]
- Tags[68]

1.6.1.2.48 TRVTableItemInfo.SetCellVAlign

Assigns **Value** to Cells[574][**Row**, **Col**].VAlign[613], as an editing operation.

```
procedure SetCellVAlign(Value: TRVCellVAlign[670]; Row,Col: Integer);
```

(introduced in version 1.6)

Unlike a direct assignment to Cells[574][**Row**, **Col**].VAlign[613] property, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.49 TRVTableItemInfo.SetCellVisibleBorders

Assigns values to Cells[574][**Row**, **Col**].VisibleBorders[613], as an editing operation.

```
procedure SetCellVisibleBorders(Left, Top, Right, Bottom: Boolean;
  Row,Col: Integer);
```

(introduced in version 1.6)

Unlike direct assignment to Cells[574][**Row**, **Col**].VisibleBorders[613] subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.50 TRVTableItemInfo.SetRow*

Assigns **Value** to Rows[580][**Row**].VAlign[616] (or PageBreakBefore[616], or KeepTogether[615]), as an editing operation.

```
procedure SetRowVAlign(Value: TRVCellVAlign[670]; Row: Integer);
procedure SetRowPageBreakBefore(Value: Boolean; Row: Integer);
procedure SetRowKeepTogether(Value: Boolean; Row: Integer);
```

(introduced in version 1.6 and 14)

Unlike direct assignments to a Rows[580][**Row**] property, these methods can be undone/redone, if they were called for a table inserted in TRichViewEdit (after insertion[353]).

Use these methods in conjunction with the methods for updating an editor window, see the example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.51 TRVTableItemInfo.SetTableVisibleBorders

Assigns values to VisibleBorders[581], as an editing operation.

```
procedure SetTableVisibleBorders(Left, Top, Right, Bottom: Boolean);
```

(introduced in version 10)

Unlike direct assignment to VisibleBorders[581] subproperties, this method can be undone/redone, if it was called for the table inserted in TRichViewEdit (after insertion[353]).

Use this method in conjunction with methods for updating editor window, see example[156].

See also:

- Undo/redo in tables[155];
- Undo/redo in TRichViewEdit[88].

1.6.1.2.52 TRVTableItemInfo.SplitSelectedCellsHorizontally

Splits each selected cell into **RowCount** cells (inserting horizontal splitters).

```
procedure SplitSelectedCellsHorizontally(RowCount: Integer);
```

This method splits cells calling UnmergeCells[601], InsertRows[592] and MergeCells[594]. This method can change number of rows.

It's recommended to call this method only when selection has a rectangular shape, see CanMergeSelectedCells[585].

If no cells are selected, it splits the cell that is being edited.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- SplitSelectedCellsVertically 601;
- UnmergeSelectedCells 601;
- GetNormalizedSelectionBounds 590.

1.6.1.2.53 TRVTableItemInfo.SplitSelectedCellsVertically

Splits each selected cell into **ColCount** cells (inserting vertical splitters).

```
procedure SplitSelectedCellsVertically(ColCount: Integer);
```

This method splits cells calling UnmergeCells 601, InsertCols 591 and MergeCells 594. This method can change number of columns.

It's recommended to call this method only when selection has a rectangular shape, see CanMergeSelectedCells 585.

If no cells are selected, it splits the cell that is being edited.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- SplitSelectedCellsVertically 601;
- UnmergeSelectedCells 601;
- GetNormalizedSelectionBounds 590.

1.6.1.2.54 TRVTableItemInfo.UnmergeCells

Unmerges cells.

```
procedure UnmergeCells(TopRow, LeftCol, ColSpan, RowSpan: Integer; UnmergeRows, UnmergeCo
```

Cells can be merged using the methods MergeCells 594 or MergeSelectedCells 594.

This method unmerges all cells in range from Cells 574[**TopRow**, **LeftCol**] to Cells 574[**TopRow**+**RowSpan**, **LeftCol**+**ColSpan**].

If **UnmergeRows**=*True*, it unmerges each cell into RowSpan 612 rows.

If **UnmergeCols**=*True*, it unmerges each cells into ColSpan 608 columns.

UnmergeRows and *UnmergeCells* can both be *True*.

See also:

- Undo note.

See also methods:

- UnmergeSelectedCells 601;
- MergeCells 594.

1.6.1.2.55 TRVTableItemInfo.UnmergeSelectedCells

Unmerges the selected cells.

```
procedure UnmergeSelectedCells(UnmergeRows, UnmergeCols: Boolean);
```

Cells can be merged using the methods MergeCells 594 or MergeSelectedCells 594.

If **UnmergeRows**=*True*, it unmerges each selected cell into RowSpan[612] rows.

If **UnmergeCols**=*True*, it unmerges each selected cells into ColSpan[608] columns.

UnmergeRows and **UnmergeCells** can both be *True*.

This method must be called when the document is formatted.

See also:

- Undo note.

See also methods:

- UnmergeCells[601];
- MergeSelectedCells[594];
- GetNormalizedSelectionBounds[590].

1.6.1.3 Events

1.6.1.3.1 TRVTableItemInfo.OnCellEditing

Occurs before starting editing of this cell and allows to prevent it.

```
type
  TRVCellEditingEvent = procedure (Sender: TRVTableItemInfo;
    Row, Col : Integer; Automatic: Boolean;
    var AllowEdit: Boolean) of object;
```

(introduced in version 1.7)

Input parameters

This event occurs when editor attempts to change content of **Sender**.Cells[574][**Row**, **Col**].

If **Automatic**=*True*, this is not actually a cell editing, but a multicell operation (clearing, applying style, etc.)

Output parameters

Set **AllowEdit** to *False* to disallow this operation.

Tables are not components, so their events cannot be assigned at design time in the Object Inspector. They must be assigned in code. Events are not saved in RVF[93] files, so you need to reassign them when tables are loaded. The best place to assign this event is TRichView.OnItemAction event.

Example

```
// This procedure is assigned to OnCellEditing of all tables
procedure TMyForm.DoCellEditing(Sender: TRVTableItemInfo; Row, Col: Integer;
  Automatic: Boolean; var AllowEdit: Boolean);
begin
  // Preventing editing all cells in the top rows of all tables
  if Row=0 then
    AllowEdit := False;
end;

// MyRichViewEdit.OnItemAction. Assigns DoCellEditing to
// OnCellEditing events of all tables inserted in MyRichViewEdit.
// (this assignment occurs when table is inserted using
// AddItem[201] or InsertItem[353] methods, or is loaded from RVF or RTF file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo[563]; var Text: String;
  RVData: TCustomRVData);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
```

```
    TRVTableItemInfo[565](Item).OnCellEditing := DoCellEditing;
end;
```

See also:

- TParaInfo.Options[487] (*rvpaoReadOnly*, *rvpaoStyleProtect*, *rvpaoDoNotWantReturns* – protection options for paragraphs);
- TFontInfo.Protection[473] (protection options for text);
- TRVExtraItemProperty[673] (*rvepDeleteProtect*, *rvepResizable* – protection option for nontext items).

1.6.1.3.2 TRVTableItemInfo.OnDraw Background

Allows custom drawing of table backgrounds

```
type
  TRVTableDrawBackEvent = procedure(Sender: TRVTableItemInfo[565]; Canvas: TCanvas;
    Left, Top, Right, Bottom: Integer;
    TableSelected, CellSelected, Printing: Boolean; Row, Col: Integer;
    var DoDefault: Boolean) of object;

property OnDrawBackground: TRVTableDrawBackEvent;
```

(introduced in version 17)

Input parameters

Sender – the table to draw background for (background can be drawn for the table itself or for the table cells).

Canvas – canvas where to draw.

Left, **Top**, **Right**, **Bottom** – rectangle for the background.

TableSelected – *True* if this table is selected completely.

CellSelected – *True* if this cell is selected.

Printing – *True* if this drawing is for printing; *False* if on screen.

Row, **Col** – row and column of the cell (or -1 if this is a table background).

Output parameters

DoDefault – set to *False* to cancel the default background drawing.

See also

- OnDrawBorder[603]

1.6.1.3.3 TRVTableItemInfo.OnDraw Border

Allows custom drawing of table borders

```
type
  TRVTableDrawBorderEvent = procedure (Sender: TRVTableItemInfo;
    Canvas: TCanvas; Left,Top,Right,Bottom: Integer;
    Width: TRVStyleLength[690];
    LightColor, Color, BackgroundColor: TColor;
    Style: TRVTableBorderStyle[691]; Printing: Boolean;
    VisibleBorders: TRVBooleanRect[653]; Row, Col: Integer;
    var DoDefault: Boolean) of object;

property OnDrawBorder: TRVTableDrawBorderEvent;
```

(introduced in version 1.7, modified in 1.8)

Input parameters

Sender – the table to draw border for (border can be drawn for the table itself or for the table cells).

Canvas – canvas where to draw.

Left, **Top**, **Right**, **Bottom** – rectangle for the border.

Width – border width, in **Sender**.Style[193].Units[437].

LightColor, **Color**, **BackgroundColor** – colors for border and for background (be careful - cell border is drawn after drawing cell content)

Style – border style.

Printing – *True* if this drawing is for printing; *False* if on screen.

VisibleBorders – VisibleBorders[581] property for the table, or VisibleBorders[613] property for the cell.

Row, **Col** – row and column of the cell (or -1 if this is a table border).

Output parameters

DoDefault – set to *False* to cancel the default border drawing.

For example,

```
{ Drawing a hairline border (both on screen and printer),
  assuming that CellBorderWidth=1, CellVSpacing=0, CellHSpacing=0.
  This example ignores VisibleBorders.
}
procedure TMyForm.DoDrawBorder(Sender: TRVTableItemInfo; Canvas: TCanvas;
  Left, Top, Right, Bottom: Integer; Width: TRVStyleLength;
  LightColor, Color, BackgroundColor: TColor; Style: TRVTableBorderStyle;
  Printing: Boolean; VisibleBorders: TRVBooleanRect; var DoDefault: Boolean);
begin
  if Width=0 then
    exit;
  inc(Right);
  inc(Bottom);
  Canvas.Pen.Color := Color;
  Canvas.Pen.Style := psInsideFrame;
  Canvas.Pen.Width := 1;
  Canvas.Brush.Style := bsClear;
  Canvas.Rectangle(Left,Top,Right,Bottom);
  DoDefault := False;
end;
```

Tables are not components, so their events cannot be assigned at design time in the Object Inspector. They must be assigned in code. Events are not saved in RVF[93] files, so you need to reassign them when tables are loaded. The best place to assign this event is TRichView.OnItemAction event.

```
// MyRichViewEdit.OnItemAction. Assigns DoDrawBorder to
// OnDrawBorder events of all tables inserted in MyRichViewEdit.
// (this assignment occurs when table is inserted using
// AddItem[201] or InsertItem[353] methods, or is loaded from RVF or RTF file)
procedure TMyForm.MyRichViewEditItemAction(Sender: TCustomRichView;
  ItemAction: TRVItemAction; Item: TCustomRVItemInfo[563]; var Text: String;
  RVData: TCustomRVData);
begin
  if (Item.StyleNo = rvsTable) and (ItemAction = rviaInserting) then
    TRVTableItemInfo[565](Item).OnDrawBorder := DoDrawBorder;
end;
```

See also

- OnDrawBackground 603

1.6.1.4 Classes of Properties

1.6.1.4.1 TRVTableCellData

TRVTableCellData represents one cell in a table (see TRVTableItemInfo 565). Cells can be accessed using Cells 574 property of table.

Unit RVTable.

Syntax

```
TRVTableCellData = class(TRVItemFormattedData 650)
```

Hierarchy

TObject
TPersistent
TCustomRVData 650
TCustomRVFormattedData
TRVItemFormattedData

Methods and Properties

TRVTableCellData has many properties and methods corresponding to properties and methods 614 of TRichView 162.

TRVTableCellData and class of object TRichView.RVData 187 are derived from the same base class.

In addition, this class has properties affecting the cell view and layout:

- Color 608 – background color of cell;
- BackgroundImage 605, BackgroundStyle 606 – background image and its position;
- BackgroundImageFileName 606 – can be used when exporting to HTML;
- BorderColor 607, BorderLightColor 608 – colors of border;
- BestWidth 607, BestHeight 606 – preferred size of cell,
- VAlign 613 – vertical alignment of contents;
- Rotation 611 – rotation of contents;
- VisibleBorders 613 – defines visible sides of cell border.
- Tag 612 – a string value (not used by the component itself).

These properties can be saved with cell in RVF 93 file or stream.

Also, this class has read-only properties:

- ColSpan 608, RowSpan 612.

Important methods:

- GetRVData 613.

See also

Table Cells Overview 146

1.6.1.4.1.1 Properties

Background image for the table cell

```
property BackgroundImage: TGraphic;
```

(introduced in v1.8)

Assignment to this property copies the image, so you still need to free the source image yourself.

Rarely used images may be "deactivated", i.e. stored in a memory stream and destroyed (see RichViewMaxPictureCount 702).

Example 1 (assigning image):

```
bmp := TBitmap.Create;
bmp.LoadFromFile('c:\image.bmp');
table.Cells[r,c].BackgroundImage := bmp;
bmp.Free;
```

Example 2 (clearing the image):

```
table.Cells[r,c].BackgroundImage := nil;
```

See also properties:

- BackgroundImageFileName 606;
- BackgroundStyle 606;
- Color 608.

See also properties of table:

- BackgroundImage 567;
- BackgroundStyle 570.

See also properties of TRichView:

- BackgroundBitmap 169;
- BackgroundStyle 170.

This is a string for storing file name associated with BackgroundImage 605

```
property BackgroundImageFileName: TRVUnicodeString 694;
```

(introduced in v1.9; changed in version 18)

If *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML 250/SaveHTMLEx 251, images with defined (non-empty) file names will not be saved, but their file names will be written in HTML (unlike pictures and tables, cells are not items, so this property cannot be accessed using item string properties 676).

See also properties of table:

- BackgroundImageFileName 568.

Defines the position of BackgroundImage 605.

```
property BackgroundStyle: TRVItemBackgroundStyle 682;
```

(introduced in v1.8)

See also:

- BackgroundImage 605;
- Color 608.

See also properties of table:

- BackgroundStyle 570;
- BackgroundImage 567.

See also properties of TRichView:

- BackgroundStyle 170;
- BackgroundBitmap 169.

Minimal height for the cell.

```
property BestHeight: TRVStyleLength 690;
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units 437 of the linked TRVStyle 418 component.

This property corresponds to <td height> in HTML, but there is a difference. **BestHeight** does not include CellVPadding[574], while <td height> includes vertical cell padding.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBestHeight[597].

The cell can be higher if it's required to show all its content (if *rvtoIgnoreContentHeight* is not included in table.Options[577]).

Default value:

0

See also properties:

- BestWidth[607].

See also properties of table:

- Options[577] (*rvtoIgnoreContentHeight* option).

See also:

- Table Resizing[158].

Preferred width for the cell

```
property BestWidth: TRVHTMLLength[680];
```

This is a preferred width for the cell, measured either in Units[437] of the linked TRVStyle[418] component (pixels or twips) or in percent.

It corresponds to <td width> in HTML.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBestWidth[597].

The cell can be wider than this value, if it's required to display its contents without overlapping. Cell always has enough width for displaying the widest its item + CellHPadding[573]*2 + paragraph indents (FirstIndent[485], LeftIndent[485], RightIndent[488]) (if *rvtoIgnoreContentWidth* is not included in table.Options[577]).

The cell can be narrower than specified in this property, if the table is not wide enough.

Default value:

0

See also properties:

- BestHeight[606].

See also properties of table:

- BestWidth[568];
- Options[577] (*rvtoIgnoreContentWidth* option).

See also:

- Table Resizing[158].

Color of border for this cell.

```
property BorderColor: TColor
```

(introduced in version 1.6)

If this value of this property is *clNone*, the cell uses table.CellBorderColor[571].

This value defines color of flat border around the cell or "dark" color of 3d border.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBorderColor[597].

Default value:

clNone

See also properties:

- BorderLightColor[608];
- Color[608].

See also properties of table:

- CellBorderColor[571];
- BorderColor[569];
- CellBorderStyle[572].

"Light" color of 3d border for this cell

```
property BorderLightColor: TColor;
```

(introduced in version 1.6)

If this value of this property is *clNone*, the cell uses table.CellBorderLightColor[571].

This value defines light color of 3d border.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellBorderLightColor[598].

Default value:

clNone

See also properties:

- BorderColor[607];
- Color[608].

See also properties of table:

- CellBorderColor[571];
- BorderColor[569];
- CellBorderStyle[572].

Background color of the cell

```
property Color: TColor;
```

Set to *clNone* for default background.

This property corresponds to <td bgcolor> in HTML.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellColor[598].

If this property is equal to clNone, a color of a row/column[583] can be used.

Default value:

clNone

See also properties:

- BorderColor[607];
- BorderLightColor[608].

See also properties of table:

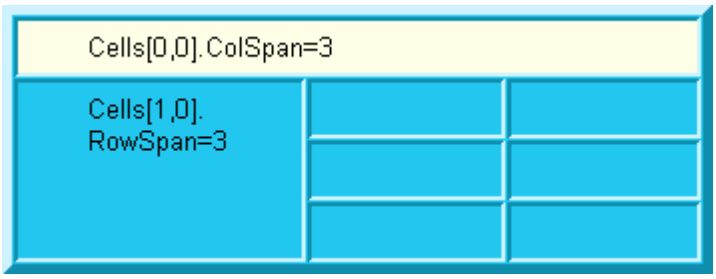
- Color[575];
- CellOverrideColor[573];
- CellBorderColor[571];
- CellBorderLightColor[571].

Read-only positive integer value specifying the number of columns spanned by this cell.

```
property ColSpan: Integer;
```

Initially this property is equal to 1 for all cells. This value can be changed due to cell-merging

Example:



ColSpan Example

This property corresponds to <td colspan> in HTML.

See also:

- RowSpan[612].

Contains the text string that can appear when the user moves the mouse pointer over the cell.

```
property Hint: String;
```

(introduced in version 10)

Default value:

" (empty string)

See also:

- TRVExtraItemStrProperty[676] (*rvespHint* defines items hints, including table hints);
- TCustomRichView.OnItemHint[281] (unlike item hints, cell hints cannot be altered in this event).

Opacity of cell background

```
property Opacity: TRVOpacity[683];
```

(introduced in version 16)

This value is used if the cell color<>*clNone*. The opacity is applied only to the background color filling, not to the background image.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellOpacity[598].

The cell color is specified in Color[608] property. If Color[608]=*clNone*, the default row/column color[583] is used.

Default value:

100000 (i.e. 100%)

See also properties of table:

- Opacity[577]

Options for table cell

```
type
  TRVTableCellOption = (rvtcoReverseLineOrder, rvtcoUseVerticalFonts);
  TRVTableCellOptions = set of TRVTableCellOption;
property Options: TRVTableCellOptions;
```

(introduced in version 16)

Option	Meaning
rvtcoReverseLineOrder	If set, lines in this cell are have reverse order. For example: • if Rotation[611]=*rvrotNone*, lines are ordered from bottom to top;

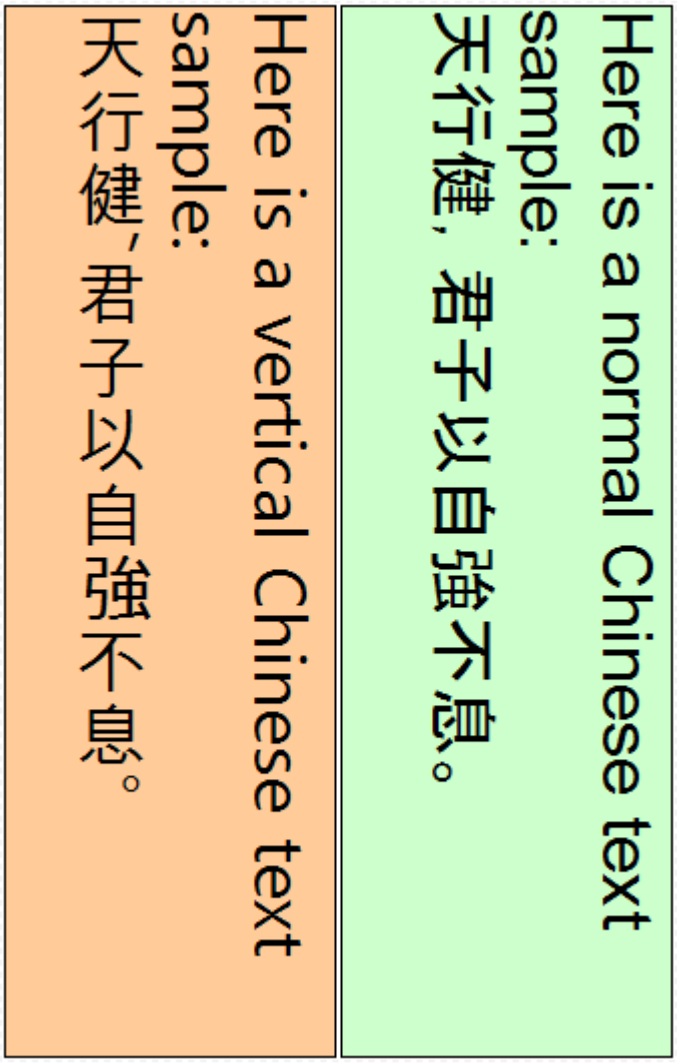
	• if Rotation[611]=*rvrot90*, lines are ordered from left to right
rvtcoUseVerticalFonts	If set, the component adds '@' to the beginning of font names to when applying fonts to canvas, if Rotation[611] is *rvrot90* or *rvrot270.*

In vertical fonts, East Asian characters are rotated by 90° counterclockwise, so, if the cell is rotated by 90° clockwise, these characters are not rotated.

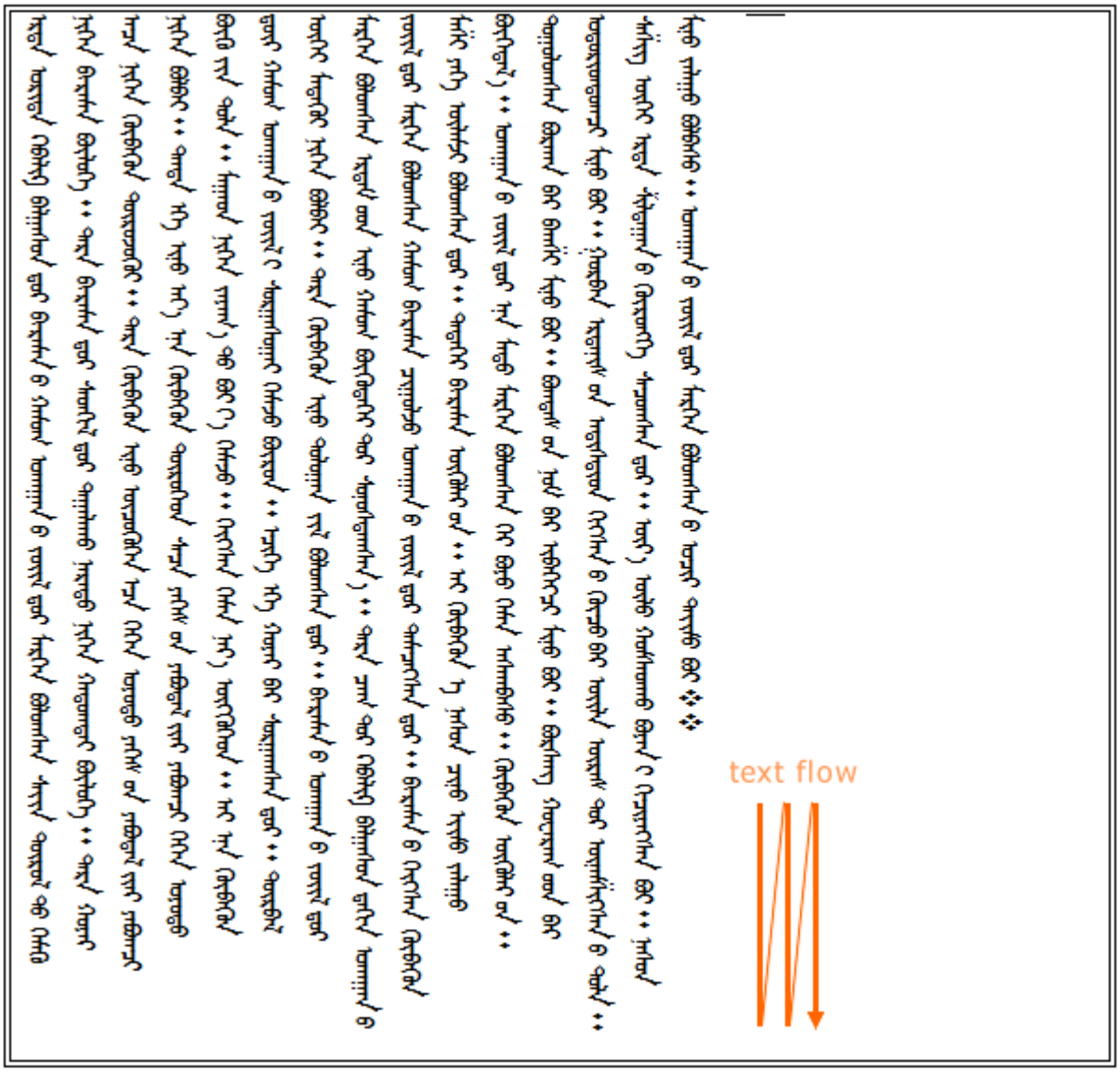
Primary meaning of these options is supporting vertical East Asian scripts.

For example:

- For vertical Chinese, Japanese or Korean text, use Rotation[611]=*rvrot90* and Options=[*rvtcoUseVerticalFonts*]. In these scripts, characters in vertical and horizontal writing must have the same orientation, columns are ordered from right to left. On the image below, the left cell has this option included.



- For vertical traditional Mongolian text, use Rotation[611]=*rvrot90*, VAlign[613]=*rvcBottom*, Options=[*rvtcoReverseLineOrder*]. In this script, columns are ordered from left to right.



Default value:

[]

Defines the rotation of the cell content.

```
property Rotation:TRVDocRotation[672];
```

(introduced in version 14)

Rotates the cell content by the specified degree.

A direct assignment to this property cannot be undone/redone by user, use table.SetCellRotation[599].

A content rotated by 90° and by 270° does not affect to width of table columns and height of table rows (if it goes beyond the cell, it is clipped).

A rotation by 90° can be used to implement vertical text, including East Asian scripts. Use it in conjunction with Options[609].

Limitations:

- this property is ignored when the application is run in Windows 95, 98, or ME;

- TPngObject/TPngImage cannot show transparency correctly if rotated;
- transparency limitation when printing (you cannot see a bitmapped background through a rotated picture further than the background of its table) (this limitation does not apply to ScaleRichView);
- no corrections are made to drawing procedures to compensate differences in drawing graphics in the standard and the advanced modes (when drawing rotated content, Canvas is switched to the advanced graphic mode);
- controls[126] inserted in rotated cells are not shown (this limitation does not apply to ScaleRichView).

Saving and loading:

- this property can be saved and loaded from RVF[93];
- rotations by 90° and by 270° are saved to RTF as the corresponding text flow in the cell, and are loaded accordingly;
- this property is not saved and not read from HTML.

See also notes about rotation in the topics about:

- Cell.VAlign[613] property;
- Table.CellPadding[574]

Default value:

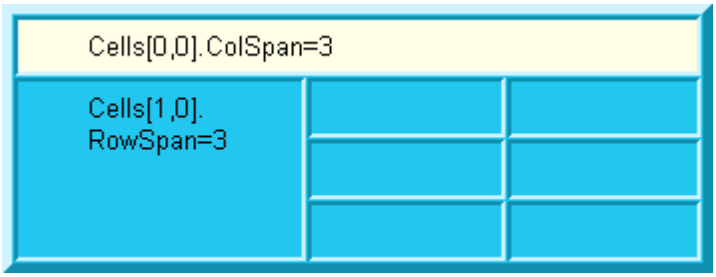
rvrotNone

Read-only positive integer value specifying the number of rows spanned by this cell

```
property RowSpan: Integer;
```

Initially this property is equal to 1 for all cells. This value can be changed due to cell-merging.

Example:



RowSpan Example

This property corresponds to <td rowspan> in HTML.

See also:

- ColSpan[608].

Contains additional string value associated with this table cell.

```
property Tag: TRVTag[692];
```

(introduced in version 14)

This value is not used by the component itself. *Tags* are provided for the convenience of storing additional string value for special needs in your application.

Default value:

'' (empty string)

See also:

- Tags[68]

Defines vertical alignment of the cell content.

```
property VAlign: TRVCellVAlign[670];
```

(introduced in version 1.6)

If this property is set to *rvcVDefault*, the cell uses vertical alignment of its row, see TRVTableRow.VAlign[616].

A direct assignment to this property cannot be undone/redone by user, use table.SetCellVAlign[599].

If the cell is rotated[611], VAlign works relative to the cell content, not relative to the cell. For example, if the cell is rotated by 90°, *rvcTop* aligns the cell content to the right side, *rvcMiddle* centers vertically, *rvcBottom* aligns to the left side.

Default value:

rvcVDefault

Allows to hide some sides of cell border.

```
property VisibleBorders: TRVBooleanRect[653];
```

A direct assignment to this property cannot be undone/redone by user, use table.SetCellVisibleBorders[600].

Default value:

True, *True*, *True*, *True* (all sides are visible)

See also properties of table:

- VisibleBorders[581];
- CellBorderWidth[572];
- CellBorderColor[571];
- CellBorderLightColor[571].

1.6.1.4.1.2 Methods

Returns object containing cell data (document items)

```
function GetRVData: TCustomRVData[650];
```

The cell is "an object containing data" itself. But when cell is in editing state[151], all its data are moved to inplace editor, and the cell is empty.

This method returns:

- the cell itself, if it is not in editing state;
- RVData[187] property of inplace editor, otherwise.

So when working with content of cell directly[614], you can use this method and do not worry about editing state of a cell.

Example (retrieving text from table):

```
function GetTableText(table: TRVTableItemInfo): String;
var
  i, r, c: Integer;
  RVData: TCustomRVData[650];
begin
  Result := '';
  for r := 0 to table.RowCount[580]-1 do
  begin
    for c := 0 to table.ColCount[575]-1 do
      if table.Cells[574][r,c]<>nil then
      begin
        RVData := table.Cells[r,c].GetRVData;
        for i := 0 to RVData.ItemCount[180]-1 do
        begin
          if (i>0) and RVData.IsFromNewLine[242](i) then
```

```
            Result := Result + #13#10;
          if RVData.GetItemStyle[229](i)=rvsTab then
            Result := Result + #9
           else if RVData.GetItemStyle(i)>=0 then
            Result := Result + RVData.GetItemText[230](i);
        end;
        Result := Result + table.TextColSeparator[581];
      end;
    Result := Result + table.TextRowSeparator[581];
  end;
end;
```

See also methods:

- GetSourceRVData[652].

1.6.1.4.1.3 Additional properties of TRVTableCellData

TRVTableCellData[605] has many properties and methods, similar to properties and methods of TRichView[162].

Some of them are listed below.

But when the cell is in editing state, RichViewEdit[316] creates special inplace-editor and temporary moves all cell content to this editor.

To provide correct work with cell content both in normal and in editing state, call methods of cell.GetRVData[613] instead of methods of the cell itself.

When working in editor, you seldom need to access cells' data directly, the most of operations can be performed using the methods that work with item at the caret position, read more[145].

See also: TRVTableItemInfo.Changed[585]

Cell methods for appending items

(links are to the corresponding methods of TCustomRichView)

- AddBreak[196];
- AddBullet[197];
- AddControl[198];
- AddHotspot[200];
- AddFmt[198], AddItem[201];
- AddNL[203];
- AddPicture[204];

(note: it's not recommended to use the methods of the group above for cells of table inserted in TRichViewEdit[316], if editor's UndoLimit[329]<>0)

Obtaining information from cell

- GetItemStyle[229];
- GetBreakInfo[218], GetBulletInfo[219], GetControlInfo[222], GetHotspotInfo[224], GetPictureInfo[236], GetTextInfo[239];
- GetItemTag[230], IsParaStart[242], GetItemPara[229], IsFromNewLine[242], GetOffsBeforeItem[235], GetOffsAfterItem[235].

Modifying items

SetBreakInfo[260], SetBulletInfo[261], SetControlInfo[262], SetHotspotInfo[265], SetPictureInfo[269], SetItemTag[267].

(note: it's not recommended to use the methods of the group above for cells of table inserted in TRichViewEdit[316], if editor's UndoLimit[329]<>0)

1.6.1.4.2 TRVTableRow

This class represents one row in table (TRVTableItemInfo)[565]

Unit RVTable;

Syntax

```
TRVTableRow = class(TRVDataList)
```

Hierarchy

TObject
TList
TRVList
TRVDataList

Using

Table has Rows[580] property of class TRVTableRows[616]. TRVTableRows is a list of objects of class TRVTableRow. TRVTableRow object is rarely, if ever, created directly in an application. It used only as an item of Rows property of tables.

Please do not use methods of this class for modifying (adding, deleting, replacing) cells. Use the proper methods of TRVTableItemInfo[565] instead.

Properties

The key properties of this class are Items[615] and Count. Count is the number of columns in the table (all rows in the table have the same number of cells).

VAlign[616] defines the default vertical alignment for the table cells in this row.

PageBreakBefore[616] allows adding a page break before this row.

KeepWithNext[615] allows keeping the content of this row on a single page, if possible.

See Also

Table Cells Overview[146]

1.6.1.4.2.1 Properties

Provides indexed access to the cells of table

```
property Items[Index: Integer]:TRVTableCellData[605]; default;
```

Each item represents one cell of the table.

Usually cells are accessed as Cells[574] property of table instead.

Allows to keep the content of this row on the same page, if possible.

```
property KeepTogether: Boolean;
```

(introduced in version 14)

This property is ignored if *rvtoRowsSplit* is not included in table.PrintOptions[579].

The effect is similar to including *rvpaoKeepWithNext* in Options[487] of all paragraphs of all cells of this row.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowKeepTogether[600].

Default value:

False

Allows to set explicit page break before this row.

```
property PageBreakBefore: Boolean;
```

(introduced in version 10)

If at least one row has PageBreaksBefore=*True*, an absence of *rvtoRowsSplit* in table.PrintOptions[579] is ignored.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowPageBreakBefore[600].

Default value:

False

Defines default vertical alignment for contents of cells in the row

```
property VAlign:TRVCellVAlign[670];
```

(introduced in version 1.6)

rvcVDefault works like *rvcMiddle*.

A direct assignment to this property cannot be undone/redone by the user, use table.SetRowVAlign[600].

Default value:

rvcTop

See also properties of cell:

- VAlign[613]

1.6.1.4.3 TRVTableRows

TRVTableRows represents contents of table (see TRVTableItemInfo[565])

Unit RVTable.

Syntax

```
TRVTableRows = class(TRVList)
```

Hierarchy

TObject
TList
TRVList

Using

This is the type of Rows[580] properties of table. TRVTableRows is a list of TRVTableRow[615]s which are in their order are lists of TRVTableCellData[605] (i.e. table cells)

TRVTableRows object is rarely, if ever, created directly in an application. It used only as Rows property of tables.

Please do not use methods of this class for modifying (adding, deleting, replacing) rows. Use the proper methods of TRVTableItemInfo[565] instead.

The key properties of this class are Items[617] and Count. Count is the number of rows in the table.

This class has method for finding main cell for given merged cell: GetMainCell[617].

See Also

Table Cells Overview[146]

1.6.1.4.3.1 Properties

Provides indexed access to the rows of table

```
property Items[Index: Integer]: TRVTableRow[615]; default;
```

Each item represents one row of the table.

1.6.1.4.3.2 Methods

Returns the main (top left) cell for given merged cell.

```
function GetMainCell(ARow,ACol: Integer; var MRow, MCol: Integer): TRVTableCellData[605];
```

Some cells[574] in table[565] can be equal to *nil* due to cell-merging. This method allows to find the top left cell overlapping the given merged cell.

Input parameters:

ARow and **ACol** specify some cell in the table. Table.Cells[574][**ARow**, **ACol**] can be *nil*.

Output parameters:

MRow and **MCol** receive position of the main cell. This cell is never equal to *nil*.

if Table.Cells[**ARow**, **ACol**]<>*nil* then **MRow** will be equal to **ARow** and **MCol** will be equal to **ACol**.

Return value:

Table.Cells[574][**MRow**, **MCol**]

1.6.2 Label - TRVLabelItemInfo

TRVLabelItemInfo is a class representing "label"[133] in TRichView documents. This is a non-text item looking like text (but it cannot be wrapped). This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsLabel* (-200)

Unit RVLabelItem.

Syntax

```
TRVLabelItemInfo = class(TRVRectItemInfo)
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo

Properties

This item displays Text[619] using font defined in TextStyleNo[620]. ProtectTextStyleNo[619] can be used to protect TextStyleNo[620] from changing.

You can set minimal possible width for this item (MinWidth[618]) and horizontal text Alignment[618].

You can define Cursor[618] for this item.

Use SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change properties of label items as editing operations.

Related Properties of TRVStyle[418]

- FieldHighlightColor[423] – color for highlighting label items (overrides background color[467] specified in the text style)
- FieldHighlightType[423] specifies when to highlight label items.

Inherited Classes

The following item types are inherited from this class:

- TRVSeqItemInfo[624] (numbered sequences);
- TRVEndNoteItemInfo[629] (endnote);
- TRVFootnoteItemInfo[630] (footnote);
- TRVNoteReferenceItemInfo[644] (reference to the parent footnote or endnote).

See Also...

Demos:

- Demos\Delphi\Assorted\Fields\MailMerge-LabelItems\
- Demos\CBuilder\Assorted\Fields\MailMerge-LabelItems\

1.6.2.1 Properties

1.6.2.1.1 TRVLabelItemInfo.Alignment

Defines a horizontal alignment of text relative to the item.

```
property Alignment: TAlignment;
```

This property makes sense if Text[619] width is less than MinWidth[618].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation (*rveipcAlignment*[706]).

Default value:

taLeftJustify

1.6.2.1.2 TRVLabelItemInfo.Cursor

Cursor for the item.

```
Cursor: TCursor;
```

Value of this property is not stored in RVF[93].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation (*rveipcCursor*[706]).

Default value:

crDefault (using default cursor)

1.6.2.1.3 TRVLabelItemInfo.MinWidth

Defines a minimal width of this item.

```
property MinWidth: Integer;
```

This value is measured either in screen pixels (when printing, it is recalculated according to the printer resolution) or in twips, depending on Units[437] of the linked TRVStyle[418] component.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation (*rveipcMinWidth*[706]).

Default value:

0

See also properties:

- Alignment[618];
- Text[619].

1.6.2.1.4 TRVLabelItemInfo.ProtectTextStyleNo

If set to *True*, TCustomRichViewEdit.ApplyTextStyle[334] and ApplyStyleConversion[333] will not be able to change TextStyleNo[620] of this item.

```
ProtectTextStyleNo: Boolean;
```

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation (*rveipcProtectTextStyleNo*[706]).

Default value:

False

1.6.2.1.5 TRVLabelItemInfo.RemoveInternalLeading, RemoveSpaceBelow

Allows removing spacing above and below the text.

```
RemoveInternalLeading: Boolean;
RemoveSpaceBelow: Boolean;
```

If **RemoveInternalLeading**=*True*, the label item is displayed without leading (space) above the text. Accent marks and other diacritical characters may occur in this area, so do not use this property if Text[619] contains accented characters.

If **RemoveSpaceBelow**=*True*, the label item is displayed without space below the text.

These properties are useful if this label item is used as a drop cap (the first letter in a chapter, with vertical alignment[695] = *rvvaLeft*).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change values of these properties as an editing operation (*rveipcRemoveInternalLeading*[706]).

Default value:

- RemoveInternalLeading: *False*
- RemoveSpaceBelow: *False*
- UseTypoMetric: *True*

1.6.2.1.6 TRVLabelItemInfo.Text

Visible text

```
Text: TRVUnicodeString[694];
```

(changed in version 18)

Unlike text items[120], this is not the text returned by TCustomRichView.GetItemText[230].

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation (RVLabelItem.*rvespcText*[709]).

See also properties:

- Alignment[618];
- TextStyleNo[620].

1.6.2.1.7 TRVLabelItemInfo.TextStyleNo

Defines text attributes.

```
property TextStyleNo: Integer;
```

This is the index in the TextStyles[437] collection of the TRVStyle[418] component linked to the TCustomRichView[162] component containing this item.

If this item is inserted in editor[316], value of this property may be changed when ApplyTextStyle[334] or ApplyStyleConversion[333] method is called. You can prevent this by assigning *True* to ProtectTextStyleNo[619] property.

If style templates[195] are used, the component may automatically change value of this property to provide a consistency of text and paragraph style templates (even if ProtectTextStyleNo[619]=*True*).

See also properties:

- Text[619].

1.6.2.2 Methods

1.6.2.2.1 TRVLabelItemInfo.Create

A constructor, creates a new TRVLabelItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new label with empty Text[619] and TextStyleNo[620]=0.

There is more convenient constructor, CreateEx[620].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.2.2.2 TRVLabelItemInfo.CreateEx

A constructor, creates a new TRVLabelItemInfo object with the specified Text[619] and TextStyleNo[620].

```
constructor CreateEx(RVData: TPersistent;
  TextStyleNo: Integer; const Text: TRVUnicodeString[694]);
```

(changed in version 18)

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

TextStyleNo – value for TextStyleNo[620] property.

Text – value for Text[619] property.

See also methods:

- Create[620].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.3 Math Expression - TRVMathItemInfo

TRVMathItemInfo is a class representing mathematical expressions [143] (equations, formulas) in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem [201]) or inserted into TRichViewEdit (see InsertItem [353]). Style [229] of this item type: *rvsMathEquation* (-210)

This item requires RAD Studio XE4 or newer.

Unit RVMathItem.

Syntax

```
TRVMathItemInfo = class(TRVRectItemInfo)
```

(introduced in version 17)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo [563]
TRVNonTextItemInfo
TRVRectItemInfo

License

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.

You can contact Adit Software if you want to use Adit Math Engine under a different license or to request an exception from the restrictions above.

Properties

The expression is defined in LaTeX-like language; it must be assigned to Text [622] property.

By default, the item uses font name, size and color defined in TRVMathDocObject [656] item of DocObjects [174] collection (or "Cambria Math, 10pt, *clWindowText*, if this object does not exist). You can override them in FontName [622], FontSizeDouble [622], TextColor [623] properties.

Use SetCurrentItemExtraIntPropertyEx [378] and SetCurrentItemExtraStrPropertyEx [379] to change properties of mathematical items as editing operations.

1.6.3.1 Properties

1.6.3.1.1 TRVMathItemInfo.DisplayInline

Switches between inline mode and display mode.

```
property DisplayInline: Boolean;
```

False: display mode, recommended for formulas inserted on separate lines

True: inline mode, recommended for formulas inserted in text.

Note: this property does not affect line breaks.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [378] to change value of this property as an editing operation (*rveipcDisplayInline* [706]).

Default value:

False

1.6.3.1.2 TRVMathItemInfo.FontName

The font name for displaying the mathematical expression.

```
property FontName: TFontName;
```

The default value of this property is '' (empty string). This means the item uses the default font name defined in TRVMathDocObject[656] item of DocObjects[174] collection (or "Cambria Math", if the object does not exist).

Only mathematical fonts can be used. Examples of such fonts:

- Cambria Math (included in Windows Vista and newer; the default font for the Microsoft Word equations)
- ***XITS Math***
- ***Asana Math***
- ***Latin Modern***

You can create a string list of mathematical fonts available on the computer using TMathTable.EnumMathFonts (from MathTable unit).

The example below assigns a list of available mathematical fonts to TComboBox items:

```
var
  SL: TStringList;
begin
  SL := TMathTable.EnumMathFonts;
  ComboBox1.Items.Assign(SL);
  SL.Free;
end;
```

If you assign a non-mathematical (or non-existent) font to this property, the item uses "Cambria Math" instead (or, if "Cambria Math" is not available, the first found available mathematical font). If no mathematical fonts are available on the computer, the item is displayed in an erroneous state.

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation (*rvespcFontName*[709]).

See also:

- FontSizeDouble[622]

1.6.3.1.3 TRVMathItemInfo.FontSizeDouble

The font size, in half-points

```
property FontSizeDouble: Integer;
```

The default value of this property is 0. This means the item uses the default font size defined in TRVMathDocObject[656] item of DocObjects[174] collection.

This value is measured in half-points, i.e. FontSizeDouble = 20 means 10pt, FontSizeDouble=21 means 10.5pt.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation (*rveipcFontSizeDouble*[706]).

See also:

- FontName[622]

1.6.3.1.4 TRVMathItemInfo.Text

Mathematical expression in LaTeX-like format.

```
property Text: TRVUnicodeString[694];
```

See ***https://en.wikibooks.org/wiki/LaTeX/Mathematics***

Differences from LaTeX:

- the symbols "[]{}" after "\left" and "\right" must be prefixed with "\" (i.e. "\left\[" instead of "\left[" in LaTeX);
- matrices are defined as "\matrix{ a_11 & a_12 \\ a_21 & a_22}"
- conditions are defined as "\case{ a_11 & a_12 \\ a_21 & a_22}"

Unlike text items [120], this is not the text returned by TCustomRichView.GetItemText [230].

Unicode note: since this item is available only for Unicode versions of Delphi, this is a Unicode text string.

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx [379] to change value of this property as an editing operation (RVMathItem.*rvespcText* [709]).

Examples of values and results:

```
\forall x \in X, \quad \exists y \leq \epsilon
```

$$\forall x \in X, \quad \exists y \leq \varepsilon$$

```
k_{n+1} = n^2 + k_n^2 - k_{n-1}
```

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

```
x = a_0 + \frac{1}{\displaystyle a_1 + \frac{1}{\displaystyle a_2 + \frac{1}{\displaystyl
```

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}$$

If the expression is incorrect, the item is displayed in an erroneous state.

1.6.3.1.5 TRVMathItemInfo.TextColor

The text color.

```
property TextColor: TColor;
```

The default value of this property is *clNone*. This means the item uses the default text color defined in TRVMathDocObject [539] item of DocObjects [174] collection.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx [378] to change value of this property as an editing operation (*rveipcTextColor* [706]).

Default value:

clNone

1.6.3.2 Methods

1.6.3.2.1 TRVMathItemInfo.Create

A constructor, creates a new TRVMathItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData [187] property of TRichView [162] or TRichViewEdit [316], where you wish to insert this item.

This constructor creates a new item with empty Text [622].

See also methods of TCustomRichView:

- AddItem [201].

See also methods of TCustomRichViewEdit:

- InsertItem [353].

1.6.4 Numbered Sequence - TRVSeqItemInfo

TRVSeqItemInfo is a class representing numbered sequence[134] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsSequence* (-202)

Unit RVSeqItem.

Syntax

```
TRVSeqItemInfo = class(TRVLabelItemInfo[617])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]

Properties

The main property of this class is SeqName[626]. All items having the same value of this property are numbered continuously. You can restart numbering from the specified value, see StartFrom[626] and Reset[626] properties.

Numbering type is defined in NumberType[625] property.

Font for this item is defined in TextStyleNo[620]. ProtectTextStyleNo[619] can be used to protect TextStyleNo[620] from changing.

Text[619] for this item is calculated automatically.

Use SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change properties of label items as editing operations.

Related Properties of TRVStyle[418]

- FieldHighlightColor[423] – color for highlighting numbering sequences (overrides background color[467] specified in the text style)
- FieldHighlightType[423] specifies when to highlight numbering sequences.

Numbering Sequences and List Markers

Numbered sequences are similar to list markers[131]. The table below shows differences between them.

Numbered sequences	List markers
Items can be inserted in any place of document.	List markers start paragraphs.
Counter for the item is calculated using SeqName[626] property.	Counter for the item is calculated using list index and list level.
Simple numbering.	Multilevel numbering.
Visual appearance is defined using the item properties.	Visual appearance is defined in list styles[429].
Items in the same sequence may have different fonts, numbering type, format	All list markers linked to the same list level have the same font, numbering type and format string.

strings.	
TCustomRichViewEdit.ApplyTextStyle[334] and ApplyStyleConversion[333] can change item font.	TCustomRichViewEdit.ApplyTextStyle[334] and ApplyStyleConversion[333] cannot change item font.

Inherited Classes

The following item types are inherited from this class:

- TRVEndNoteItemInfo[629] (endnote);
- TRVFootnoteItemInfo[630] (footnote);

See Also...

Demos:

- Demos\Delphi\Assorted\Fields\MailMerge-LabelItems\
- Demos\CBuilder\Assorted\Fields\MailMerge-LabelItems\

1.6.4.1 Properties

1.6.4.1.1 TRVSeqItemInfo.FormatString

Text format.

```
FormatString: TRVUnicodeString[694];
```

(changed in version 18)

If value of this property is not empty, it defines format for text. Use '%s' to specify where to insert numbering.

For example, if FormatString = 'Figure %s.', NumberType[625] = *rvseqUpperRoman*, and counter value is 2, this item displays 'Figure II.'.

You can assign this property before inserting this item in TRichView.

This property is ignored when saving documents to RTF.

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences[248].

Use TRichViewEdit.SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Default value:

'' (empty string)

1.6.4.1.2 TRVSeqItemInfo.NumberType

Numbering type.

```
property NumberType: TRVSeqType[689];
```

See TRVSeqType[689] for possible values.

This property is initialized in the constructor CreateEx[627].

This property is ignored in inherited classes: TRVFootnoteItemInfo[630], TRVEndnoteItemInfo[629], TRVSidenoteItemInfo[632], TRVTextBoxItemInfo[643].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences[248].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation.

See also properties:

- FormatString[625].

1.6.4.1.3 TRVSeqItemInfo.Reset

Resets counter to StartFrom[626].

```
Reset: Boolean;
```

This property is initialized in constructor CreateEx[627].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences[248].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation.

1.6.4.1.4 TRVSeqItemInfo.SeqName

Name of sequence. Items having the same value of this property are numbered continuously.

```
SeqName: TRVUnicodeString[694];
```

(changed in version 18)

Character case is ignored for this property.

The following values are reserved and cannot be used for this property for TRVSeqItemInfo:

- '@footnote@' (used in TRVFootnoteItemInfo[630]);
- '@endnote@' (used in TRVEndnoteItemInfo[629]).
- '@sidenote@' (used in TRVSidenoteItemInfo[632]).
- '@textbox@' (used in TRVTextBoxItemInfo[643]).

This property is initialized in constructor CreateEx[627].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation.

1.6.4.1.5 TRVSeqItemInfo.StartFrom

Value of counter for this item, if Reset[626] = *True*.

```
StartFrom: Integer;
```

This property is initialized in constructor CreateEx[627].

If you change this property for an item that was already inserted in a document, call RichView.RefreshSequences[248].

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation.

1.6.4.2 Methods

1.6.4.2.1 TRVSeqItemInfo.Create

A constructor, creates a new TRVSeqItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Please do not use this constructor, because it does not allow to set values of properties (including read-only properties).

Use CreateEx[627] instead.

1.6.4.2.2 TRVSeqItemInfo.CreateEx

A constructor, creates a new TRVSeqItemInfo object.

```
constructor CreateEx(RVData: TPersistent; const ASeqName: TRVUnicodeString;
  ANumberType: TRVSeqType; ATextStyleNo, AStartFrom: Integer;
  AReset: Boolean);
```

(changed in version 18)

Parameters:

RVData – RVData property of TRichView or TRichViewEdit where you wish to insert this item.

ASeqName – value for SeqName property.

ANumberType – value for NumberType property.

ATextStyleNo – value for TextStyleNo property.

AStartFrom – value for StartFrom property.

AReset – value for Reset property.

See also methods of TCustomRichView:

- AddItem.

See also methods of TCustomRichViewEdit:

- InsertItem.

1.6.5 Ancestor for Notes - TCustomRVNoteItemInfo

TCustomRVNoteItemInfo is an ancestor class for footnote, endnote, sidenote and text box items.

Objects of this class must not be created directly, create objects of the following inherited classes:

- TRVFootnoteItemInfo;
- TRVEndnoteItemInfo;
- TRVSidenoteItemInfo;
- TRVTextBoxItemInfo.

Unit RVNote.

Syntax

```
TCustomRVNoteItemInfo = class(TRVSeqItemInfo)
TRVFootOrEndnoteItemInfo = class(TCustomRVNoteItemInfo)
```

(introduced in version 10 and 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo
TRVSeqItemInfo

Properties

This class introduces Document property – footnote/endnote/sidenote/textbox text. This text is not displayed in TCustomRichView, but it is displayed on printing and in ScaleRichView.

Document uses the same TRVStyle component as the main document (where a note is inserted).

You can use ReplaceDocumentEd[628] method to change Document[627] as an editing operation.

See Also...

Demos:

- Demos\Delphi\Editors\Notes
- Demos\CBuilder\Editors\Notes

1.6.5.1 Properties

1.6.5.1.1 TCustomRVNoteItemInfo.Document

Footnote/endnote text (displayed on printing)

```
property Document: TRVNoteData[650];
```

You can use ReplaceDocumentEd[628] method to change Document as an editing operation.

1.6.5.1.2 TCustomRVNoteItemInfo.NoteText

Text for assigning to NoteText[182] property of TRichViewEdit[316].

```
property NoteText: TRVUnicodeString[694];
```

(introduced in version 15; changed in version 18)

This property returns Text[619].

If TRichViewEdit is used as an editor for Document[628], you should assign this property to its NoteText[182] property, to allow a proper displaying of note references[140].

1.6.5.2 Methods

1.6.5.2.1 TCustomRVNoteItemInfo.CreateEx

A constructor, creates a new TCustomRVNoteItemInfo object

```
constructor CreateEx(RVData: TPersistent;
  ATextStyleNo, AStartFrom: Integer; AReset: Boolean); virtual;
```

Do not create objects of this class. Create objects of inherited classes:

- TRVFootnoteItemInfo[630];
- TRVEndnoteItemInfo[629];
- TRVSidenoteItemInfo[632];
- TRVTextBoxItemInfo[643].

Parameters

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[620] property. It's recommended to use RVGetNoteTextStyleNo[663] function for assigning this parameter.

AStartFrom – value for StartFrom[626] property.

AReset – value for Reset[626] property.

1.6.5.2.2 TCustomRVNoteItemInfo.ReplaceDocumentEd

Replaces content of Document[628] as an editing operation.

```
procedure ReplaceDocumentEd(Stream: TStream);
```

Stream must contain new document in RVF format[93].

1.6.6 Endnote - TRVEndnoteItemInfo

TRVEndnoteItemInfo is a class representing an endnote[135] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsEndnote* (-204)

Unit RVNote.

Syntax

```
TRVEndnoteItemInfo = class(TRVFootOrEndnoteItemInfo[627])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]
TRVSeqItemInfo[624]
TCustomRVNoteItemInfo[627]
TRVFootOrEndnoteItemInfo[627]

Using

Numbering type for endnotes is defined in EndnoteNumbering[422] property of TRVStyle component (NumberType[625] property, inherited from TRVSeqItemInfo[624], is ignored).

You can enumerate all endnotes in documents using RVGetFirstEndnote and RVGetNextEndnote[662] functions.

Use RVGetNoteTextStyleNo[663] to assign TextStyleNo[620] property (in the constructor).

Limitation: in TRVReportHelper[540], all pages consisting only of endnotes' texts have the same height as the last document's page.

Use SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change properties of label items as editing operations.

Example

This example inserts a new endnote in rveMain (TCustomRichViewEdit). TextStyleNo[620] for this endnote is returned by RVGetNoteTextStyleNo[663] function basing on the rveMain.CurTextStyleNo[324]-th style).

Document[628] for this endnote contains one reference[140] (TextStyleNo[620] for this reference is returned by RVGetNoteTextStyleNo[663] function basing on the 0th style) and one space character of the 0th style.

```
var EndNote: TRVEndnoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo[644];
begin
  EndNote := TRVEndnoteItemInfo.CreateEx[630](rveMain.RVData[187],
    RVGetNoteTextStyleNo[663](rvs, rveMain.CurTextStyleNo[324]), 1, False);
  NoteRef := TRVNoteReferenceItemInfo[644].CreateEx[646](EndNote.Document[628], RVGetNoteTextSty
  EndNote.Document[628].AddItem[201]('', NoteRef);
  EndNote.Document[628].AddNL[203](' ', 0, -1);
  if rveMain.InsertItem[353]('', EndNote) then
    ...
end;
```

The topic about TRVNoteReferenceItemInfo[644] contains example showing how to edit Document[628].

See Also...

Demos:

- Demos\Delphi\Editors\Notes
- Demos\CBuilder\Editors\Notes

1.6.6.1 Methods

1.6.6.1.1 TRVEndnoteItemInfo.Create

A constructor, creates a new TRVEndnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new endnote with TextStyleNo[620]=0.

There is more convenient constructor, CreateEx[630].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.6.1.2 TRVEndnoteItemInfo.CreateEx

A constructor, creates a new TRVEndnoteItemInfo object with the specified properties

```
constructor CreateEx(RVData: TPersistent;
  ATextStyleNo, AStartFrom: Integer; AReset: Boolean); override;
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[620] property. It's recommended to use RVGetNoteTextStyleNo[663] function for assigning this parameter.

AStartFrom – value for StartFrom[626] property.

AReset – value for Reset[626] property.

See also methods:

- Create[630].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.7 Footnote - TRVFootnoteItemInfo

TRVFootnoteItemInfo is a class representing a footnote[136] in TRichView documents. This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsFootnote* (-203)

Unit RVNote.

Syntax

```
TRVFootnoteItemInfo = class(TRVFootOrEndnoteItemInfo 627)
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo 563
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo 617
TRVSeqItemInfo 624
TCustomRVNoteItemInfo 627
TRVFootOrEndnoteItemInfo 627

Using

Numbering type for footnotes is defined in FootnoteNumbering 422 property of TRVStyle 418 component (NumberType 625 property, inherited from TRVSeqItemInfo 624, is ignored).

You can enumerate all footnotes in documents using RVGetFirstFootnote and RVGetNextFootnote 662 functions.

Use RVGetNoteTextStyleNo 663 to assign TextStyleNo 620 property (in the constructor).

Footnotes are numbered continuously in TCustomRichView. But when printing, footnote numbering can be restarted on each page, see FootnotePageReset 425 property of TRVStyle 418 component.

Limitation: footnote's Document 628 must not be higher than the page height. Unlike endnotes 135, footnote is printed on a single page.

Use SetCurrentItemExtraIntPropertyEx 378 and SetCurrentItemExtraStrPropertyEx 379 to change properties of label items as editing operations.

Example

This example inserts a new footnote in rveMain (TCustomRichViewEdit). TextStyleNo 620 for this footnote is returned by RVGetNoteTextStyleNo 663 function basing on the rveMain.CurTextStyleNo 324-th style).

Document 628 for this footnote contains one reference 140 (TextStyleNo 620 for this reference is returned by RVGetNoteTextStyleNo 663 function basing on the 0th style) and one space character of the 0th style.

```
var FootNote: TRVFootnoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo 644;
begin
  FootNote := TRVFootnoteItemInfo.CreateEx 632(rveMain.RVData 187,
    RVGetNoteTextStyleNo 663(rvs, rveMain.CurTextStyleNo 324), 1, False);
  NoteRef := TRVNoteReferenceItemInfo 644.CreateEx 646(FootNote.Document 628, RVGetNoteTextSty
  FootNote.Document 628.AddItem 201('', NoteRef);
  FootNote.Document 628.AddNL 203(' ', 0, -1);
  if rveMain.InsertItem 353('', FootNote) then
    ...
end;
```

The topic about TRVNoteReferenceItemInfo 644 contains example showing how to edit Document 628.

See Also...

Demos:

- Demos\Delphi\Editors\Notes
- Demos\CBuilder\Editors\Notes

1.6.7.1 Methods

1.6.7.1.1 TRVFootnoteItemInfo.Create

A constructor, creates a new TRVFootnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new footnote with TextStyleNo[620]=0.

There is more convenient constructor, CreateEx[632].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.7.1.2 TRVFootnoteItemInfo.CreateEx

A constructor, creates a new TRVFootnoteItemInfo object with the specified properties

```
constructor CreateEx(RVData: TPersistent;
  ATextStyleNo, AStartFrom: Integer; AReset: Boolean); override;
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[620] property. It's recommended to use RVGetNoteTextStyleNo[663] function for assigning this parameter.

AStartFrom – value for StartFrom[626] property.

AReset – value for Reset[626] property.

See also methods:

- Create[632].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.8 Sidenote - TRVSidenoteItemInfo

TRVSidenoteItemInfo is a class representing a sidenote[137] in TRichView documents. Sidenote is a note displayed in a floating box.

This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsSidenote* (-206)

Unit RVSidenote.

Syntax

```
TRVSidenoteItemInfo = class(TCustomRVNoteItemInfo[627])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo 563
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo 617
TRVSeqItemInfo 624
TCustomRVNoteItemInfo 627

Using

Numbering type for sidenotes is defined in SidenoteNumbering 422 property of TRVStyle 418 component (NumberType 625 property, inherited from TRVSeqItemInfo 624, is ignored).

You can enumerate all sidenotes in documents using RVGetFirstSidenote and RVGetNextSidenote 662 functions (these functions return text box items 139 as well)

Use RVGetNoteTextStyleNo 663 to assign TextStyleNo 620 property (in the constructor).

This class introduces the following properties:

- BoxPosition 634 – contains properties defining the position of the floating box
- BoxProperties 634 – contains properties describing size, background and border of the floating box.

Use SetCurrentItemExtraIntPropertyEx 378 and SetCurrentItemExtraStrPropertyEx 379 to change properties of sidenotes as editing operations.

Example

This example inserts a new sidenote in rveMain (TCustomRichViewEdit). TextStyleNo 620 for this sidenote is returned by RVGetNoteTextStyleNo 663 function basing on the rveMain.CurTextStyleNo 324-th style).

Document 628 for this sidenote contains one reference 140 (TextStyleNo 620 for this reference is returned by RVGetNoteTextStyleNo 663 function basing on the 0th style) and one space character of the 0th style.

A text box for this sidenote has a yellow background and a double border, and positioned by 50 pixels below the sidenote character line's top side:

```
var SideNote: TRVSidenoteItemInfo;
    NoteRef: TRVNoteReferenceItemInfo;
begin
  SideNote := TRVSidenoteItemInfo.CreateEx(rveMain.RVData,
    RVGetNoteTextStyleNo(rvs, rveMain.CurTextStyleNo), 1, False);
  NoteRef := TRVNoteReferenceItemInfo.CreateEx(SideNote.Document, RVGetNoteTextSty
  SideNote.Document.AddItem('', NoteRef);
  SideNote.Document.AddNL(' ', 0, -1);
  SideNote.BoxProperties.Background.Color := clYellow;
  SideNote.BoxProperties.Border.Style := rvbDouble;
  SideNote.BoxPosition.VerticalOffset := rveMain.Style.PixelsToUnits(50);
  if rveMain.InsertItem('', SideNote) then
    ...
end;
```

The topic about TRVNoteReferenceItemInfo 644 contains example showing how to edit Document 628.

See also

- TRVTexBoxItemInfo 643

1.6.8.1 Properties

1.6.8.1.1 TRVSidenoteItemInfo.BoxProperties

Properties of a floating box: size, border, background, vertical alignment of content.

```
property BoxProperties: TRVBoxProperties[635];
```

1.6.8.1.2 TRVSidenoteItemInfo.BoxPosition

Properties of a floating box: horizontal and vertical positions, z-order.

```
property BoxPosition: TRVBoxPosition[637];
```

1.6.8.2 Methods

1.6.8.2.1 TRVSidenoteItemInfo.Create

A constructor, creates a new TRVSidenoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new sidenote with TextStyleNo[620]=0.

There is more convenient constructor, CreateEx[634].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.8.2.2 TRVSidenoteItemInfo.CreateEx

A constructor, creates a new TRVSidenoteItemInfo object with the specified properties

```
constructor CreateEx(RVData: TPersistent;
  ATextStyleNo, AStartFrom: Integer; AReset: Boolean); override;
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

ATextStyleNo – value for TextStyleNo[620] property. It's recommended to use RVGetNoteTextStyleNo[663] function for assigning this parameter.

AStartFrom – value for StartFrom[626] property.

AReset – value for Reset[626] property.

See also methods:

- Create[634].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.8.3 Classes of Properties

1.6.8.3.1 TRVBoxProperties

TRVBoxProperties contains properties of a floating box: size, border, background, etc.

Unit RVFloatingBox.

Syntax

```
TRVBoxProperties = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Using

This is a class of BoxProperties[634] property of sidenotes[137] and text boxes[139].

Properties:

- Background[635];
- Border[635];
- Height, HeightType[636];
- Width, WidthType[636];
- VAlign[636].

1.6.8.3.1.1 Properties

Defines background properties.

```
type
  TRVBoxBackgroundRect = class (TRVBackgroundRect[500])
property Background: TRVBoxBackgroundRect;
```

TRVBoxBackgroundRect changes the initial values of properties introduced in TRVBackgroundRect[500]:

- Color[501]=*clWindow*
- BorderOffsets[501]: Left=Right=10, Top=Bottom=5.

Unlike in TRVBackgroundRect, in TRVBoxBackgroundRect negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of subproperties of this property as an editing operation.

Defines background properties.

```
type
  TRVBoxBorder = class (TRVBorder[501])
property Border: TRVBoxBorder;
```

TRVBoxBorder changes the initial values of properties introduced in TRVBorder[501]:

- Style[502]=*rvbSingle*
- BorderOffsets[502]: Left=Right=10, Top=Bottom=5.

Unlike in TRVBorder, in TRVBoxBorder negative values of BorderOffsets are ignored (treated as zeros).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change values of subproperties of this property as an editing operation.

These properties define a height of a floating box.

```
type
  TRVBoxHeightType = (rvbhtAbsolute, rvbhtRelPage, rvbhtRelMainTextArea,
    rvbhtRelTopMargin, rvbhtRelBottomMargin, rvbhtAuto);
property HeightType: TRVBoxHeightType;
property Height: TRVFloatSize[678];
```

Value of HeightType	Meaning
rvbhtAbsolute	**Height** specifies the box height in Units[437] of TRVStyle (pixels or twips)
rvbhtRelPage	**Height** specifies the box height in (1/1000)% of the page height
rvbhtRelMainTextArea	**Height** specifies the box height in (1/1000)% of the main text area height
rvbhtRelTopMargin	**Height** specifies the box height in (1/1000)% of the top margin height
rvbhtRelBottomMargin	**Height** specifies the box height in (1/1000)% of the bottom margin height
rvbhtAuto	The box height is calculated based on the box content. **Height** property is ignored.

A main text area is an area on a page between margins, where the main document is printed.

Heights of the main text area, the top margin and the bottom margins are calculated after adjusting by the header height and the footer height (too long header may increase a height the top margin and decrease a height of the main text area; too long footer may increase a height the bottom margin and decrease a height of the main text area). Since headers and footers may be different on first/odd/even pages, these height may be different as well.

These properties define a height of a text box including contents, padding and border.

Unlike position properties, these values are always calculated relative to a page (and never calculated relative to a table cell).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change values of these properties as an editing operation.

Example:

HeightType=*rvbhtRelPage*, **Height**=20000: the height of box is equal to 20% of the page height.

Default values:

- HeightType: *rvbhtAuto*
- Height: 100

Specifies a vertical alignment of a box content inside a box.

```
property VAlign: TTextLayout;
```

This property makes sense if HeightType[636]<>*rvbhtAuto.*

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Default value:

tlTop

These properties define a width of a floating box.

```
type
  TRVBoxWidthType = (rvbwtAbsolute, rvbwtRelPage, rvbwtRelMainTextArea,
    rvbwtRelLeftMargin, rvbwtRelRightMargin,
```

```
    rvbwtRelInnerMargin, rvbwtRelOuterMargin);
property WidthType: TRVBoxWidthType;
property Width: TRVFloatSize[678];
```

Value of WidthType	Meaning
rvbwtAbsolute	**Width** specifies the box width in Units[437] of TRVStyle (pixels or twips)
rvbwtRelPage	**Width** specifies the box width in (1/1000)% of the page width
rvbwtRelMainTextArea	**Width** specifies the box width in (1/1000)% of the main text area width
rvbwtRelLeftMargin	**Width** specifies the box width in (1/1000)% of the left margin width
rvbwtRelRightMargin	**Width** specifies the box width in (1/1000)% of the right margin width
rvbwtRelInnerMargin	**Width** specifies the box width in (1/1000)% of the inner margin width (i.e. left margin for odd pages, right margin for even pages)
rvbwtRelOuterMargin	**Width** specifies the box width in (1/1000)% of the outer margin width (i.e. right margin for odd pages, left margin for even pages)

A main text area is an area on a page between margins, where the main document is printed.

"Mirror margins" option affects calculation of margin widths on even pages.

For text boxes included in a header/footer that is used both for odd and even pages, a calculation is performed for odd pages (the same width is used for all pages).

These properties define a width of a text box including contents, padding and border.

Unlike position properties, these values are always calculated relative to a page (and never calculated relative to a table cell).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change values of these properties as an editing operation.

Example:

WidthType=*rvbwtRelPage*, **Width**=20000: the width of box is equal to 20% of the page width.

Default values:

- WidthType: *rvbwtAbsolute*
- Height: 100

1.6.8.3.2 TRVBoxPosition

TRVBoxPosition contains properties defining a position of a floating box.

Unit RVFloatingPos.

Syntax

```
TRVBoxPosition = class (TPersistent)
```

Hierarchy

TObject
TPersistent

Using

This is a class of BoxPosition[634] property of sidenotes[137] and text boxes[139].

Properties:

- HorizontalAnchor[638] and HorizontalPositionKind, HorizontalOffset, HorizontalAlignment[639];
- VerticalAnchor[641] and VerticalPositionKind, VerticalOffset, VerticalAlignment[642];
- RelativeToCell[640];
- PositionInText[640].

1.6.8.3.2.1 Properties

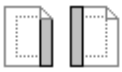
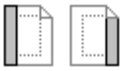
Specifies a horizontal anchor area for a floating box.

```
type
  TRVHorizontalAnchor = (rvhanCharacter, rvhanPage, rvhanMainTextArea,
    rvhanLeftMargin, rvhanRightMargin, rvhanInnerMargin, rvhanOuterMargin);
property HorizontalAnchor: TRVHorizontalAnchor;
```

A text box is positioned horizontally relative to the area specified in this property.

It is important to understand that each area has a width; for example, when printing using TRVPrint[513], a left margin starts from the page's left edge and ends at Margins[520].Left.

If RelativeToCell[640]=*True*, and a sidenote[137] (or a text box item[139]) is inserted in a table cell[146], then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
rvhanCharacter	A sidenote[137] character, or a place of insertion of a text box item[139].	
rvhanPage	A whole page (or a whole table cell)	
rvhanMainTextArea	A main text area, i.e. an area between margins (or a table cell area minus cellpadding)	
rvhanLeftMargin	A left page margin (or a left cell padding area)	
rvhanRightMargin	A right page margin (or a right cell padding area)	
rvhanInnerMargin	An inner page margin: a left margin for odd pages, a right margin for even pages (or a left cell padding area on odd pages, a right cell padding area on even pages)	
rvhanOuterMargin	An outer page margin: a right margin for odd pages, a left margin for even pages (or a right cell padding area on odd pages, a left cell padding area on even pages)	

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Differences from Microsoft Word*:

- In MS Word, a "character" anchor does not have a width, it is just a position. In TRichView, sidenote character has a width, so a center and a right alignments relative to a character of a sidenote are slightly different in TRichView and in MS Word (for simple text boxes, they are identical).
- MS Word does not support a relative (percent) alignment relative to a character (so TRichView exports it as an absolute position with zero offset).

(* this information is based on test of MS Word 2013)

Default value:

rvhanCharacter

These properties define a horizontal position of a floating box.

```
type
  TRVFloatHorizontalAlignment = (rvfphalLeft, rvfphalCenter, rvfphalRight);
property HorizontalPositionKind: TRVFloatPositionKind[678];
property HorizontalOffset: TRVFloatPosition[677];
property HorizontalAlignment: TRVFloatHorizontalAlignment;
```

A position is calculated relative to the area specified in HorizontalAnchor[638].

Note 1: these properties define the position of the box in the way similar to Microsoft Word. However, there are differences, see the notes in the topic about HorizontalAnchor[638].

Note 2: printers cannot print at areas close to paper edges; is the box is positioned close to a left or a right side of the page, it may overlap these areas.

HorizontalPositionKind=rvfpkAlignment

The position is defined in **HorizontalAlignment**. **HorizontalOffset** is ignored.

Value of HorizontalAlignment	Meaning	Scheme
rvfphalLeft	The left side of the box is at the left side of the anchor area.	
rvfphalCenter	The center of the box is at center of the anchor area.	
rvfphalRight	The right side of the box is at the right side of the anchor area.	

HorizontalPositionKind=rvfpkAbsPosition

The position is defined in **HorizontalOffset**. **HorizontalAlignment** is ignored.

HorizontalOffset is measured in pixels or twips, depending on Units[437] property of the linked TRVStyle[418].

A position is counted from the left side of the anchor area. Positive values shift the box to the right, negative values shift the box to the left.

HorizontalPositionKind=rvfpkPercentPosition

The position is defined in **HorizontalOffset**. **HorizontalAlignment** is ignored.

HorizontalOffset is measured in (1/1000)% of the anchor area width.

A position is counted from the left side of the anchor area. Positive values shift the box to the right, negative values shift the box to the left.

Examples:

Value of HorizontalOffset	Meaning	Scheme

0	0%	
100000	100%	
50000	50%	
-100000	-100%	

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change values of these properties as an editing operation.

Default values:

- HorizontalPositionKind: *rvfpkAbsPosition*
- HorizontalOffset: 0
- HorizontalAlignment: *rvfphalLeft*

Specifies how the text box is positioned relative to a text where a sidenote[137] (or a text box item[139]) is inserted.

```
type
  TRVFloatPositionInText = (rvpitAboveText, rvpitBelowText);
property PositionInText: TRVFloatPositionInText;
```

Value	Meaning
rvpitAboveText	The box is above a text.
rvpitBelowText	The box is below a text.

Note: boxes inserted in a header or a footer are always below the main text (and below boxes inserted in the main text).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Default value:

rvpitAboveText

Allows anchoring a text box to a table cell[146].

```
property RelativeToCell: Boolean;
```

If *True*, and a sidenote[137] (or a text box item[139]) is placed in a table cell, then horizontal[638] and vertical[641] positions are calculated relative to this cell (instead of a page).

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Default value:

True

Specifies a vertical anchor area for a floating box.

```
type
  TRVVerticalAnchor = (rvvanLine, rvvanParagraph, rvvanPage,
    rvvanMainTextArea, rvvanTopMargin, rvvanBottomMargin);
property VerticalAnchor: TRVVerticalAnchor;
```

A text box is positioned vertically relative to the area specified in this property.

It is important to understand that each area has a height; for example, when printing using TRVPrint[513], a top margin starts from the page's top edge and ends at Margins[520].Top.

If RelativeToCell[640]=*True*, and a sidenote[137] (or a text box item[139]) is inserted in a table cell[146], then this property defines an area of this cell instead of a page.

Value	Anchor Area	Scheme
rvvanLine	A line where a sidenote[137] character (or a text box item[139]) is inserted.	
rvvanParagraph	A paragraph where a sidenote[137] character (or a text box item[139]) is inserted.	
rvvanPage	A whole page (or a whole table cell)	
rvvanMainTextArea	A main text area, i.e. an area between margins (or a table cell area minus cellpadding)	
rvvanTopMargin	A top page margin (or a top cell padding area)	
rvvanBottomMargin	A bottom page margin (or a bottom cell padding area)	

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change value of this property as an editing operation.

Differences from Microsoft Word*:

- In MS Word, a "line" anchor does not have a height. MS Word uses only a top line coordinate, so a center and a bottom alignments relative to a line are different in TRichView and MS Word.
- MS Word does not support alignment relative to a paragraph (so TRichView exports it as an alignment to a line).
- MS Word does not support a relative (percent) alignment relative to a line or a paragraph (so TRichView exports them as absolute positions with zero offsets).
- MS Word does not implement all options for vertical positioning relative to a cell:
  - absolute positions for all options (except for a line and a paragraph) are calculated relative to the cell area minus cellpadding (i.e. top of cell + top cell padding).
  - any alignment or percent position moves the box exactly to the cell area minus cellpadding (i.e. top of cell + top cell padding).

(* this information is based on tests of MS Word 2013)

Default value:

rvvanLine

These properties define a vertical position of a floating box.

```
type
  TRVFloatVerticalAlignment = (rvfpvalTop, rvfpvalCenter, rvfpvalBottom);
property VerticalPositionKind: TRVFloatPositionKind[678];
property VerticalOffset: TRVFloatPosition[677];
property VerticalAlignment: TRVFloatVerticalAlignment;
```

A position is calculated relative to the area specified in VerticalAnchor[641].

Note 1: these properties define the position of the box in the way similar to Microsoft Word. However, there are differences, see the notes in the topic about VerticalAnchor[641].

Note 2: printers cannot print at areas close to paper edges; is the box is positioned close to a top or a bottom side of the page, it may overlap these areas.

VerticalPositionKind=rvfpkAlignment

The position is defined in **VerticalAlignment**. **VerticalOffset** is ignored.

Value of VerticalAlignment	Meaning
rvfphalTop	The top side of the box is at the top side of the anchor area.
rvfphalCenter	The center of the box is at center of the anchor area.
rvfphalBottom	The bottom side of the box is at the bottom side of the anchor area.

VerticalPositionKind=rvfpkAbsPosition

The position is defined in **VerticalOffset**. **VerticalAlignment** is ignored.

VerticalOffset is measured in pixels or twips, depending on Units[437] property of the linked TRVStyle[418].

A position is counted from the top side of the anchor area. Positive values shift the box down, negative values shift the box up.

VerticalPositionKind=rvfpkPercentPosition

The position is defined in **VerticalOffset**. **VerticalAlignment** is ignored.

VerticalOffset is measured in (1/1000)% of the anchor area height.

A position is counted from the top side of the anchor area. Positive values shift the box down, negative values shift the box up.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change values of these properties as an editing operation.

Default values:

- VerticalPositionKind: *rvfpkAbsPosition*
- VerticalOffset: 0
- VerticalAlignment: *rvfpvalTop*

1.6.9 TextBox - TRVTextBoxItemInfo

TRVTextBoxItemInfo is a class representing text box item[139] in TRichView documents. This item is printed as a floating box.

This is not a component. Objects of this class are created at runtime and appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsTextBox* (-207)

Unit RVSidenote.

Syntax

```
TRVTextBoxItemInfo = class(TRVSidenoteItemInfo[632])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]
TRVSeqItemInfo[624]
TCustomRVNoteItemInfo[627]
TRVSidenoteItemInfo[632]

Using

This is a simplified version of a sidenote[137]. It has all properties of a sidenote, but the most of them are ignored, except for BoxPosition[634] and BoxProperties[634].

A special mark can be drawn in places of insertion of text box items, see TRVStyle.FloatingLineColor[423].

1.6.9.1 Methods

1.6.9.1.1 TRVTextBoxItemInfo.Create

A constructor, creates a new TRVFootnoteItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new text box item with TextStyleNo[620]=0.

See also CreateEx[643].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.9.1.2 TRVTextBoxItemInfo.CreateEx

A constructor, creates a new TRVTextBoxItemInfo object.

```
constructor CreateEx(RVData: TPersistent;
  ATextStyleNo, AStartFrom: Integer; AReset: Boolean); override;
```

The most of parameters of this constructor are not used. You can use a simpler constructor: Create[643].

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

ATextStyleNo – ignored.

AStartFrom – value for StartFrom[626] property. Assigned, but not used in this type of item.

AReset – value for Reset[626] property. Assigned, but not used in this type of item.

See also methods:

- Create[643].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.10 Reference to the Parent Note - TRVNoteReferenceItemInfo

TRVNoteReferenceItemInfo is an item type referring to[140] the parent footnote[136], endnote[135], or sidenote[137] item. Objects of this class are usually appended to the note's Document[628] (see AddItem[201]). Besides, they can be appended to TRichView (see AddItem[201]) or inserted into TRichViewEdit (see InsertItem[353]). Style[229] of this item type: *rvsNoteReference* (-205).

Unit RVNote.

Syntax

```
TRVNoteReferenceItemInfo = class(TRVLabelItemInfo[617])
```

(introduced in version 10)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]

Using

When inserted in a note's Document[628], the note reference displays the same text as the parent note.

When inserted in TCustomRichView, it displays NoteText[182].

Text[619] for this item is calculated automatically.

Use RVGetNoteTextStyleNo[663] to assign TextStyleNo[620] property (in the constructor).

Use SetCurrentItemExtraIntPropertyEx[378] and SetCurrentItemExtraStrPropertyEx[379] to change properties of label items as editing operations.

Example

See examples in TRVFootnoteItemInfo[630] and TRVEndNoteItemInfo[629] showing how to insert footnote/endnote with document containing note reference.

This example shows how to edit note document in rveNote (TCustomRichViewEdit[316]). The footnote/endnote is NoteItem.

This code copies document from footnote/endnote to the editor:

```
var Stream: TMemoryStream;
```

```
begin

  rveNote.NoteText[182] := NoteItem.Text[619];
  rveNote.Clear[210];
  Stream := TMemoryStream.Create;
  try
    NoteItem.Document[628].SaveRVFToStream[254](Stream);
    Stream.Position := 0;
    rveNote.LoadRVFFromStream[243](Stream);
  finally
    Stream.Free;
  end;
  with rveNote do begin
    Format[217];
    SetSelectionBounds[271](ItemCount-1, GetOffsAfterItem(ItemCount-1),
      ItemCount-1, GetOffsAfterItem(ItemCount-1));
  end;
```

You can see that the text of footnote/endnote is assigned to the editor's NoteText[182]. If this document contains note references, they will display this text.

After editing, the document can be saved back to the note:

```
  if not rveNote.Modified[327] then
    exit;
  Stream := TMemoryStream.Create;
  try
    rveNote.SaveRVFToStream[254](Stream, False);
    NoteItem.ReplaceDocumentEd[628](Stream);
  finally
    Stream.Free;
  end;
  rveNote.Modified[327] := False;
```

See Also...

Demos:

- Demos\Delphi\Editors\Notes
- Demos\CBuilder\Editors\Notes

1.6.10.1 Methods

1.6.10.1.1 TRVNoteReferenceItemInfo.Create

A constructor creating a new TRVNoteReferenceItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new note reference with TextStyleNo[620]=0.

There is more convenient constructor, CreateEx[646].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.10.1.2 TRVNoteReferenceItemInfo.CreateEx

A constructor creating a new TRVNoteReferenceItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; TextStyleNo: Integer);
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

TextStyleNo – value for TextStyleNo[620] property. It's recommended to use RVGetNoteTextStyleNo[663] function for assigning this parameter.

See also methods:

- Create[645].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.11 Ancestor for Text Fields - TCustomRVFieldItemInfo

TCustomRVFieldItemInfo is an ancestor class for page number and page count fields.

Objects of this class must not be created directly, create objects of the following inherited classes:

- TRVPageNumberItemInfo[647];
- TRVPageCountItemInfo[648].

Unit RVFieldItems.

Syntax

```
TCustomRVFieldItemInfo = class (TRVLabelItemInfo[617]);
TCustomRVPageNumberItemInfo = class(TCustomRVFieldItemInfo);
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]

TCustomRVPageNumberItemInfo introduces a new property: NumberType[646].

1.6.11.1 Properties

1.6.11.1.1 TCustomRVPageNumberItemInfo.NumberType

Specifies how the page number (or page count) is displayed.

```
NumberType: TRVPageNumberType[683];
```

This property is initialized in the constructor.

Use TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378] to change value of this property as an editing operation.

Default value:
rvpntDefault

1.6.12 Page Number - TRVPageNumberItemInfo

TRVPageNumberItemInfo displays the current page number[141].

Unit RVFieldItems.

Syntax

```
TRVPageNumberItemInfo = class(TCustomRVPageNumberItemInfo[646])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]

Using

When inserted in TRichView[162], it is displayed as Text[619] (by default, as '{p}').

When printed with TRVPrint[513], it displays a number of the page where this item is located. Page numbers are counted from TRVPrint.PageNoFromNumber[552].

When drawing with TRVReportHelper[540], the default behavior is like in TRVPrint, but you can provide a value for the page number in OnGetPageNumber[543] event.

ScaleRichView: when inserted in TSRichViewEdit, it displays a number of the page where this item is located. Page numbers are counted from TSRichViewEdit.PageProperty.PageNoFromNumber.

A displayed page number is formatted according to NumberType[646] property.

1.6.12.1 Methods

1.6.12.1.1 TRVPageNumberItemInfo.Create

A constructor creating a new TRVPageNumberItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new page number field with TextStyleNo[620]=0, Text[619]='{p}'.

There is more convenient constructor, CreateEx[648].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.12.1.2 TRVPageNumberItemInfo.CreateEx

A constructor creating a new TRVPageNumberItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType[683];
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

NumberType – value for NumberType[646] property.

TextStyleNo – value for TextStyleNo[620] property.

This constructor creates a new page number field with Text[619]='{p}'.

See also methods:

- Create[647].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.13 Page Count - TRVPageCountItemInfo

TRVPageCountItemInfo displays the count of pages[142].

Unit RVFieldItems.

Syntax

```
TRVPageCountItemInfo = class(TCustomRVPageNumberItemInfo[646])
```

(introduced in version 15)

Hierarchy

TObject
TPersistent
TCustomRVItemInfo[563]
TRVNonTextItemInfo
TRVRectItemInfo
TRVLabelItemInfo[617]

Using

When inserted in TRichView[162], it is displayed as Text[619] (by default, as '{P}').

When printed with TRVPrint[513], it displays a count of pages.

ScaleRichView: when inserted in TSRichViewEdit, it displays a count of pages.

A displayed page number is formatted according to NumberType[646] property.

1.6.13.1 Methods

1.6.13.1.1 TRVPageCountItemInfo.Create

A constructor creating a new TRVPageCountItemInfo object

```
constructor Create(RVData: TPersistent); override;
```

Parameter:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316], where you wish to insert this item.

This constructor creates a new page count field with TextStyleNo[620]=0, Text[619]='{P}'.

There is more convenient constructor, CreateEx[649].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.6.13.1.2 TRVPageCountItemInfo.CreateEx

A constructor creating a new TRVPageCountItemInfo object with the specified properties.

```
constructor CreateEx(RVData: TPersistent; NumberType: TRVPageNumberType[683];
  TextStyleNo: Integer); reintroduce;
```

Parameters:

RVData – RVData[187] property of TRichView[162] or TRichViewEdit[316] where you wish to insert this item.

NumberType – value for NumberType[646] property.

TextStyleNo – value for TextStyleNo[620] property.

This constructor creates a new page count field with Text[619]='{P}'.

See also methods:

- Create[648].

See also methods of TCustomRichView:

- AddItem[201].

See also methods of TCustomRichViewEdit:

- InsertItem[353].

1.7 Other Classes

1.7.1 ERichViewError

This is an exception class used by TRichView Family of components. It does not introduce new methods and properties.

Unit RichView;

Syntax

```
ERichViewError = class(Exception)
```

Hierarchy

TObject
Exception

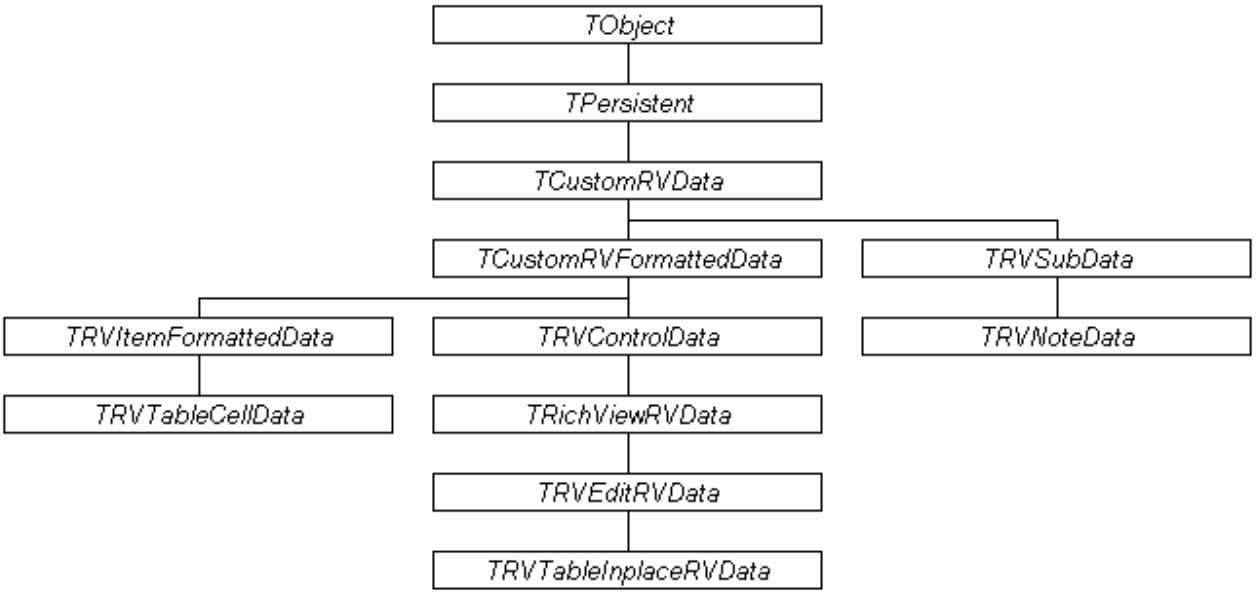
1.7.2 TCustomRVData and Others (TRichView Documents)

These classes represent RichView documents. Documents contain a list of items. Some items may, in each order, contain subdocuments.

Class	Unit	Meaning
TCustomRVData	CRVData	Basic class for TRichView documents.
TCustomRVFormattedData	CRVFData	Basic class for formatted TRichView documents (allows storing items coordinates).
TRVControlData	RVCtrlData	Basic class for documents owned by windowed control.
TRichViewRVData	RVRVData	Document in TRichView 162 and TDBRichView 407, a class of RVData 187 property of these components.
TRVEditRVData	RVERVData	Document in TRichViewEdit 316 and TDBRichViewEdit 410, a class of RVData 187 property of these components.
TRVTInplaceRVData	RVTInplace	Document in cell inplace editor, a class of of RVData 187 property of cell inplace editors.
TRVItemFormattedData	RVDataList	Basic class for formatted subdocuments owned by items.
TRVTableCellData 605	RVTable	Table 144 cell.
TRVSubData	RVSubData	Basic class for invisible documents owned by items.
TRVNoteData	RVNote	Document in footnote or endnote.

This list does not include document classes used for printing.

Hierarchy



Overview

Special methods for **TCustomRVData**:

- Edit 651;
- GetRVData 652;
- GetSourceRVData 652.

Special methods for **TCustomRVFormattedData**:

- GetOriginEx 653.

These classes have many methods and properties similar to methods and properties in TRichView (what's why in this manual many links to RVData's methods refer to TRichView's methods).

For example, the methods for appending items:

(the links are to the corresponding methods of TRichView)

- AddBreak 196,
- AddBullet 197,
- AddControl 198,
- AddHotspot 200,
- AddFmt 198, AddItem 201,
- AddNL 203, AddTextNL 206,
- AddPicture 204.

Obtaining information from cell:

- GetItemStyle 229, GetItemText -A -W 230,
- GetBreakInfo 218, GetBulletInfo 219, GetControlInfo 222, GetHotspotInfo 224, GetPictureInfo 236
- GetItemTag 230, GetItemVAlign 230, IsParaStart 242, GetItemPara 229, IsFromNewLine 242

Modifying items:

- SetBreakInfo 260, SetBulletInfo 261, SetControlInfo 262, SetHotspotInfo 265, SetPictureInfo 269, SetItemTag 267, SetItemVAlign 268, SetItemText -A -W 267

See also

- Controls, Documents, Items 104

1.7.2.1 TCustomRVData.Edit

Initializes editor for this RVData.

```
function Edit: TCustomRVData;
```

For RVData 187 of editor (main editor, or cell inplace editor), this method does nothing.

For table cell 605, this method creates cell inplace editor and returns its RVData 187.

If you want to select part of document in table cell, call Cell.Edit (or table.EditCell 589) before calling cell.SetSelectionBounds 271 or cell.SelectAll 257.

This method must be called when the document is formatted.

Return value

GetRVData 652 of the given RVData after initialization of editing.

Example

This example deletes all controls from rve: TRichViewEdit 316 as an editing operation (that can be undone and redone by the user).

In this example, all references to the methods and properties of TCustomRVFormattedData point to the methods and properties of TRichView 162. TCustomRVFormattedData has all these methods and properties, with the same meaning.

```
procedure DeleteAllControls(RVData: TCustomRVData 650; rve: TCustomRichViewEdit 316);
var i,r,c: Integer;
    table: TRVTableItemInfo 565;
```

```
begin
  i := 0;
  while i<RVData.ItemCount[180] do begin
    case RVData.GetItemStyle[229](i) of
    rvsComponent[711]:
      begin
        RVData := RVData.Edit;
        TCustomRVFormattedData(RVData).SetSelectionBounds[271](i, 0, i, 1);
        rve.DeleteSelection[340];
        i := rve.TopLevelEditor[329].CurItemNo[322]-1;
      end;
    rvsTable[711]:
      begin
        table := TRVTableItemInfo[565](RVData.GetItem[225](i));
        for r := 0 to table.RowCount[580]-1 do
          for c := 0 to table.ColCount[575]-1 do
            if table.Cells[574][r,c]<>nil then
              DeleteAllControls(table.Cells[574][r,c].GetRVData[613], rve);
      end;
    end; // of case
    inc(i);
  end; // of while
end;
```

Call:

```
DeleteAllControls(rve.RVData[187], rve);
```

1.7.2.2 TCustomRVData.GetRVData

For the given document, returns the document containing its items.

```
function GetRVData: TCustomRVData;
```

For the most of document classes, this method returns Self (RVData.GetRVData = RVData).

The only exception is table cells (TRVTableCellData[605]). When the cell is not edited, Cell.GetRVData = Cell. But when the cell is edited, this method returns RVData[187] of the cell inplace editor.

Using RVData.GetRVData instead of RVData you can always access the document's items.

See also

- GetSourceRVData[652];
- Edit[651].

1.7.2.3 TCustomRVData.GetSourceRVData

This is an inverse method to GetRVData[652].

```
function GetSourceRVData: TCustomRVData;
```

For the most of document classes, this method returns Self, i.e. RVData.GetSourceRVData = RVData.

The only exception is RVData[187] of the cell inplace editor, it returns the source cell (TRVTableCellData[605]) for this editor.

See also

- GetRVData[652].

1.7.2.4 TCustomRVFormattedData.GetOriginEx

Returns coordinates of the given subdocument relative to the main document.

```
procedure GetOriginEx(var ALeft, ATop: Integer);
```

Output parameters

ALeft, **ATop** – coordinates of the top left corner of this RVData relative to the top left corner of the main document.

Coordinates of the given item relative to the main document are calculated as a sum of values returned by RVData.GetItemCoords [226] and RVData.GetOriginEx.

This method does not take cell rotation into account, the coordinates are returned as if none of cells are rotated.

1.7.3 TPrintableRV

TRichView Reference [27] | Classes

TPrintableRV is used internally by TRVPrint

1.7.4 TRVBooleanRect

This class contains 4 boolean properties: Left, Top, Right, Bottom

Unit RVStyle;

Syntax

```
TRVRect = class (TPersistent)
```

(Introduced in version 1.2)

This class is used for defining visible sides for paragraph, cell and table borders.

See also:

- TRVBorder.VisibleBorders [502];
- TRVTableCellData.VisibleBorders [613];
- TRVTableItemInfo.VisibleBorders [581].

1.7.5 TRVDeleteUnusedStylesData

This class allows to delete unused text [437], paragraph [431] and list [429] styles in multiple documents (to delete styles that do not present in at least one of the documents)

Unit RVItem.

Syntax

```
TRVDeleteUnusedStylesData = class;
```

(introduced in version 10)

The only method of this class you need to use is the constructor:

```
constructor Create(ATextStyles, AParaStyles, AListStyles: Boolean);
```

Its parameters specify types of unused styles you want to delete.

The examples below assume that collections of text, paragraph and list styles are not saved in RVF documents [93] (but stored in file or in the Registry). In this case, several documents share the same collections

of styles. TCustomRichView.DeleteUnusedStyles[216] cannot be used in this case, MarkStylesInUse[247] and DeleteMarkedStyles[215] must be used instead.

Note: usually it's much more simple (and recommended) to use one RVStyle per one RichView and store collections of styles inside RVF documents.

Example 1

How to remove unused styles in RichView1 and RichView2, linked to the same RVStyle.

```
uses RVItem;

  data := TRVDeleteUnusedStylesData.Create(True, True, True);
  RichView1.MarkStylesInUse[247](data);
  RichView2.MarkStylesInUse[247](data);
  RichView1.DeleteMarkedStyles[215](data);
  RichView2.DeleteMarkedStyles[215](data);
  data.Free;
```

Example 2

How to remove unused styles in several documents.

```
uses CRVData, RVItem;

  data := TRVDeleteUnusedStylesData.Create(True, True, True);
  RichView1.LoadRVF[245]('Doc1.rvf');
  RichView1.MarkStylesInUse[247](data);
  RichView1.LoadRVF[245]('Doc2.rvf');
  RichView1.MarkStylesInUse[247](data);
  ...
  RichViewDoNotCheckRVFStyleRefs[701] := True;
  RichView1.LoadRVF[245]('Doc1.rvf');
  RichView1.DeleteMarkedStyles[215](data);
  RichView1.SaveRVF[254]('Doc1.rvf', False);
  RichView1.LoadRVF[245]('Doc2.rvf');
  RichView1.DeleteMarkedStyles[215](data);
  RichView1.SaveRVF[254]('Doc2.rvf', False);
  ...
  RichViewDoNotCheckRVFStyleRefs[701] := False;
  data.Free;
```

1.7.6 TRVGraphicHandler

This class provides an abstraction layer for managing graphic in TRichView components. It is designed to make TRichView components independent of specific graphic classes.

Unit RVGrHandler;

Syntax

```
TRVGraphicHandler = class;
```

(introduced in version 15)

Using

A single variable of this class is created: RVGraphicHandler[710]. You can call methods of the object returned in this variable.

Additionally, you can create a class inherited from TRVGraphicHandler and assign it to this variable (however, this possibility is not discussed in this help file).

Methods

A complete description of all methods of this class is beyond the scope of this help file.

The following methods are useful:

```
type
  TRVGraphicType = // defined in RVFuncs unit
    (rvgtBitmap, rvgtIcon, rvgtMetafile, rvgtJPEG, rvgtPNG, rvgtGIF, rvgtTIFF, rvgtOther)
procedure SetDefaultGraphicClass(GraphicType: TRVGraphicType; GraphicClass: TGraphicClass
procedure RegisterPngGraphic(GraphicClass: TGraphicClass);
procedure RegisterJpegGraphic(GraphicClass: TGraphicClass);
procedure RegisterHTMLGraphicFormat(GraphicClass: TGraphicClass);
function CreateGraphic(GraphicClass: TGraphicClass): TGraphic;
function CreateGraphicByType(GraphicType: TRVGraphicType): TGraphic;
function LoadFromFile(const FileName: String): TGraphic;
function LoadFromStream(Stream: TStream): TGraphic;
```

Choosing default graphic classes

RegisterPngGraphic registers a PNG graphic class for using in TRichView. Call this method one time when the application starts (for example, in the initialization section of the main form's unit). If PNG class is registered, TRichView controls can save and load PNG images in RTF and and save in DocX files. Optionally, bitmaps can be saved as PNG in RTF too, see TCustomRichView.RTFOptions[185] property (*rvrtfPNGInsteadOfBitmap*).

By default, in Delphi 2009+, TRichView registers a standard TPngImage as PNG graphic format, so, unless you want to use another class for PNG images, this method is necessary only for Delphi 2007 or older. For example, you can use Gustavo Huffenbacher Daud's TPngObject: ***download file*** (free PNG graphic class).

RegisterJpegGraphic registers a JPEG graphic class for using in TRichView. If this method is not called, TRichView uses TJPEGImage. This class is used when loading Jpeg images embedded in RTF.

There is an universal method allowing to define graphic classes for the specified graphic type: **SetDefaultGraphicClass**.

By default, the classes are:

- bitmaps: TBitmap,
- icons: TIcon,
- metafiles: TMetafile,
- jpeg: TJPEGImage,
- png: TPngImage (for Delphi 2009+, undefined otherwise)
- gif: TGifImage (for Delphi 2007+, if RVGifAnimate2007 is included in the project; for previous versions of Delphi, if RVGifAnimate is included in the project; undefined otherwise
- tiff: TWicImage (for Delphi 2010+, undefined otherwise)

HTML export

RegisterHTMLGraphicFormat allows to define HTML graphic formats. Call this method one time when the application starts (for example, in the initialization section of the main form's unit). Images of these formats are saved in HTML[96] as they are, without conversion to Jpegs. It's not necessary to call it for TJpegImage. These formats must have file extension associated, see TPicture.RegisterFileFormat. TRichView registers the following classes as HTML graphic formats:

- standard TPngImage (for Delphi 2009+);
- standard TGifImage (for Delphi 2007+), if you include RVGifAnimate2007 unit in your project;
- Anders Melander's TGifImage, if you include RVGifAnimate unit in your project;
- TJvGifImage from JEDI's JVCL, if you include RVJvGifAnimate unit in your project.

Creating graphics

CreateGraphic creates a graphic of this specified class. For Delphi/C++Builder 6 or newer, this method is equivalent to calling GraphicClass.Create. For older version of Delphi, this method contains a workaround for TGraphic bug.

CreateGraphicByType creates a graphic of the specified type, see "Choosing default graphic classes" above.

LoadFromFile creates a graphic object, loads the file FileName in it, and returns this graphic. A TGraphic class of a created object is determined by the file extension of the file. If the file extension is not recognized, or loading failed, the method returns *nil*.

LoadFromStream creates a graphic object, loads the stream content in it, and returns this graphic. A graphic format is recognized by content. If this procedure cannot recognize a graphic format, or no graphic class is defined for this graphic type,

See also

how to use third-party graphic classes in TRichView[720]

1.7.7 TRVIntegerList

List of integer values, used internally by TRichView classes.

Unit RVClasses;

```
type
  TRVIntegerList = class(TList)
```

Main properties:

Items[Index: Integer]: Integer.

Count: Integer

1.7.8 TRVMathDocObject

This is a class of a document object defining default properties for items-mathematical expressions[143].

Unit RVMathItem;

Syntax

```
TRVMathDocObject = class (TRVDocObject[174]);
```

(introduced in version 17)

Properties

This class has properties:

- FontName: TFontName – default font name for all mathematical formulas in this document. Default value: "Cambria Math".
- FontSizeDouble: Integer – default font size (in half-points) for all mathematical formulas in this document. Default value: 20.
- TextColor: TColor – default text color for all mathematical formulas in this document. Default value: *clWindowText*.

Items may override these properties in their FontName[622], FontSizeDouble[622], TextColor[623] properties.

See the explanations about mathematical fonts in the help topics about TRVMathItemInfo.FontName[622]

How to use

It's not recommended to use this object directly. Use RVGetMathProperties and RVSetMathProperties[667] procedures instead.

To save and load these properties in RVF[93], include *rvfoSaveDocObjects* and *rvfoLoadDocObjects* in RVFOptions[188].

1.7.9 TRVRect

This is a class similar to TRect record.

Unit RVStyle;

Syntax

```
TRVRect = class (TPersistent)
```

(Introduced in version 1.2)

This class is used for representing offsets (padding) of colored background and border of paragraphs.

The class has Left, Top, Right and Bottom properties of TRVStyleLength[690] type.

1.7.10 TRVReportHelperWithHeaderFooters

This class contains TRVReportHelper[540] components representing a main document and all six possible headers and footers.

Unit RVLabelItem.

Syntax

```
TRVReportHelperWithHeaderFooters = class;
```

(introduced in version 17)

Hierarchy

TObject

Descriptions

An object of this class contains:

- TRVReportHelper[540] component used for the main document (it is accessible as MainDoc[657])
- TRVReportHelper components used for all six headers and footers (they are accessible as GetHeader[659] and GetFooter[658])
- TRVStyle[418] components for each TRVReportHelper.

See details in the help topic about the constructor[658].

Use AssignToRVPrint[658] to initialize TRVPrint[513] components for printing this document.

Use

This class may be used for importing and exporting files. For example, it is used in the demo projects in <TRichView Dir>**\ThirdParty\Export**:

- **LLPDFLib\Demos\RV2PDF** (converting RTF and RVF files to PDF using ***LLPDFLib***)
- **SynPDF\Demos\RV2PDF** (converting RTF and RVF files to PDF using ***Synopse PDF Engine***)
- **eDocEngine\Demos\TRichView** (converting RTF and RVF files to PDF and other formats using ***eDocEngine***)

1.7.10.1 Properties

1.7.10.1.1 TRVReportHelperWithHeaderFooters.MainDoc

Returns TRVReportHelper component used to store the main document.

```
property MainDoc: TRVReportHelper[540];
```

See also

- GetHeader[659]

- GetFooter 658

1.7.10.1.2 TRVReportHelperWithHeaderFooters.UseStyleTemplates

Allows or disallows using MainDoc 657.RichView 541.Style 193.StyleTemplates 435.

```
property UseStyleTemplates: Boolean;
```

This property provides access to UseStyleTemplates 195 property of all RichViews of all TRVReportHelpers stored in the component.

Default value

False

1.7.10.2 Methods

1.7.10.2.1 TRVReportHelperWithHeaderFooters.AssignToRVPrint

Prepares the document stored in this component to printing by **RVPrint**.

```
procedure AssignToRVPrint(RVPrint: TRVPrint 513);
```

This procedure prepares **RVPrint** by calling its AssignSource 522, SetHeader 529, SetFooter 528, AssignDocParameters 522 methods.

This procedure does not format RVPrint (it does not call its FormatPages 525).

This procedure does not assign page size (only margins and header&footer positions)

1.7.10.2.2 TRVReportHelperWithHeaderFooters.Create

Creates an instance of TRVReportHelperWithHeaderFooters 657 component.

```
constructor Create;
```

The constructor:

- creates TRVReportHelper 540 component which will be used for the main document (it is accessible as MainDoc 657)
- creates TRVReportHelper components which will be used for all six headers and footers (they are accessible as GetHeader 659 and GetFooter 658), assigns them as headers/footers to the main document;
- creates TRVStyle 418 components for each created TRVReportHelper and assigns them;
- assigns properties of TRVReportHelper components so that they load RTF and RVF documents with maximum possible quality;
- for TRVStyle of all headers and footers, assigns their MainRVStyle 430 properties to MainDoc 657.RichView 541.Style 193 (so, if you decide to use style templates 658, they all will use MainDoc 657.RichView 541.Style 193.StyleTemplates 435.

Default value

False

1.7.10.2.3 TRVReportHelperWithHeaderFooters.GetFooter

Returns TRVReportHelper component representing a footer of the specified type.

```
function GetFooter(FooterType: TRVHFType 680): TRVReportHelper 540;
```

See also

- GetHeader 659
- MainDoc 657

1.7.10.2.4 TRVReportHelperWithHeaderFooters.GetHeader

Returns TRVReportHelper component representing a header of the specified type.

```
function GetHeader(FooterType: TRVHFType[680]): TRVReportHelper[540];
```

See also

- GetFooter[658]
- MainDoc[657]

1.7.10.2.5 TRVReportHelperWithHeaderFooters.Reset

Completely clears all documents stored in this component.

```
procedure Reset;
```

This procedure clears[210] all documents (the main document, headers, footers), deletes all text[437], paragraph[431] and list[429] styles from all TRVStyles, adds one text and one text and paragraph style having default properties.

If UseStyleTemplates[658] = True, it also clears StyleTemplates[435] for the main document, adds a single normal style template[461], and assign it to paragraph styles.

Additionally, it assigns 5 pixels to all margins (LeftMargin[180], RightMargin[185], TopMargin[194], BottomMargin[171]), and resets DocParameters[176] (keeping their Units[314]).

1.7.11 TRVThumbnailMaker

This class creates smooth scaled copies of images.

Unit RVThumbMaker;

Syntax

```
TRVThumbnailMaker = class;
```

(introduced in version 15)

Using

A single variable of this class is created: RVThumbnailMaker[713]. You can change properties and call methods of the object returned in this variable.

Properties

MaxThumbnailSize: Integer. Maximal possible width and height of scaled images.

Methods

```
procedure AllowThumbnailsFor(Cls: TGraphicClass);
procedure DisallowThumbnailsFor(Cls: TGraphicClass);
function IsAllowedFor(Gr: TGraphic): Boolean;
function MakeThumbnail(Gr: TGraphic; Width, Height: Integer): TBitmap;
```

AllowThumbnailsFor allows making scaled images for the specified graphic class. By default, scaling is allowed for TBitmap, TPngImage and TJPEGImage. For Delphi 2010 or newer, scaling is allowed for TWicImage as well (used for TIFF images). Thumbnails are not created for transparent images.

AllowThumbnailsFor disallows scaling for the specified graphic class.

IsAllowedFor tests if scaling allowed for the specified graphic.

MakeThumbnail creates and returns a bitmap made from **Gr** and scaled to **Width** x **Height**. If Width or Height exceeds **MaxThumbnailSize**, they are reduced proportionally. The method returns *nil* in the following cases:

- making scaled images is not allowed for **Gr**;
- **Gr** is empty (0 x 0);

- **Gr**.Width=**Width** and **Gr**.Height=**Height**;
- both max(**Gr**.Width, **Gr**.Height) and max(**Width**, **Height**) exceed **MaxThumbnailSize**, and **Gr** is smaller than **Width** x **Height** (because we want to enlarge the image, but it would be shrunken instead)

See also

See also:

- smooth image scaling [91]

1.7.12 TRVUnitsRect

This class contains 4 properties of TRVLength [682] type: Left, Top, Right, Bottom

Unit RVStyle;

Syntax

```
TRVFloatRect = class (TPersistent)
```

(Introduced in version 15)

This class is used for Margins [520] property of TRVPrint [513].

1.8 Procedures and Functions

1.8.1 DrawControl and Others

These functions create bitmap having the size of the specified control, draw image of this control into this bitmap, return this bitmap.

Unit CtrlImg.

Syntax

```
function DrawControl(ctrl: TControl): TBitmap;
```

Also

```
function DrawButton(ctrl: TButton): TBitmap;
function DrawEdit(ctrl: TEdit): TBitmap;
function DrawMemo(ctrl: TMemo): TBitmap;
function DrawPanel(ctrl: TPanel): TBitmap;
```

DrawControl has the functionality of all other functions of this group.

TRVPrint use these functions for default printing of inserted controls [126], see OnPrintComponent [554] event.

1.8.2 GetRVESearchOptions

Converts options of TFindDialog/TReplaceDialog to TRichViewEdit [316] search options.

Unit RVMisc.

Syntax

```
function GetRVESearchOptions(fo: TFindOptions): TRVESearchOptions [672];
```

This function always includes *rvseoMultiItem* and *rvseoSmartStart* in the result.

See also

- TRichViewEdit.SearchText [369].

1.8.3 GetRVSearchOptions

Converts options of TFindDialog to TRichView[162] search options

Unit RVMisc.

Syntax

```
function GetRVSearchOptions(fo: TFindOptions): TRVSearchOptions[689];
```

This function always includes *rvsroMultiItem* and *rvsroSmartStart* in the result.

See also

- TRichView.SearchText[256].

1.8.4 RV_GetPrinterDC

Returns handle compatible with the the current printer object.

Unit PtblRV.

Syntax

```
function RV_GetPrinterDC: HDC;
```

After using this handle, delete it with DeleteDC.

Alternatively, you can use Printer.Handle.

Example

The procedure below returns size of document area on page, converted to screen pixels. Document area is paper size minus margins.

For conversion, it uses screen resolution (RichViewPixelsPerInch[703] is ignored).

It does not take headers[529] and footers[528] into account.

```
procedure GetPageSize(RVPrint: TRVPrint[513]; var Width, Height: Integer);
var
  DC: HDC;
  phoX, phoY, phW, phH, lpy, lpx, LM, TM, RM, BM: Integer;
begin
  DC := RV_GetPrinterDC;

  Width := GetDeviceCaps(DC, HORZRES);
  Height := GetDeviceCaps(DC, VERTRES);

  lpy := GetDeviceCaps(DC, LOGPIXELSY);
  lpx := GetDeviceCaps(DC, LOGPIXELSX);

  phoX := GetDeviceCaps(DC, PHYSICALOFFSETX);
  phoY := GetDeviceCaps(DC, PHYSICALOFFSETY);
  phW := GetDeviceCaps(DC, PHYSICALWIDTH);
  phH := GetDeviceCaps(DC, PHYSICALHEIGHT);

  LM := RV_UnitsToPixels[667](RVPrint.Margins[520].Left,   FUnits, lpx)- phoX;
  TM := RV_UnitsToPixels(RVPrint.Margins.Top,    FUnits, lpy)- phoY;
  RM := RV_UnitsToPixels(RVPrint.Margins.Right,  FUnits, lpx)- (phW-(phoX+Width));
  BM := RV_UnitsToPixels(RVPrint.Margins.Bottom, FUnits, lpy)- (phH-(phoY+Height));

  if RVPrint.FixMarginsMode[518]=rvfmmAutoCorrect then
  begin
    if LM<0 then
    begin
```

```
      inc(RM, LM);
      LM := 0;
    end;
    if TM<0 then
    begin
      inc(BM, TM);
      TM := 0;
    end;
    if RM<0 then
      RM := 0;
    if BM<0 then
      BM := 0;
  end;

  dec(Width, LM+RM);
  dec(Height, TM+BM);

  DeleteDC(DC);

  DC := GetDC(0);
  Width := MulDiv(Width, GetDeviceCaps(DC, LOGPIXELSX), lpx);
  Height := MulDiv(Height, GetDeviceCaps(DC, LOGPIXELSY), lpy);
  ReleaseDC(0, DC);
end;
```

Example of using (making document width in editor equal to page width, taking difference in resolutions into account):

```
var w, h: Integer;

// rvoClientTextWidth must be excluded from RichViewEdit1.Options.

GetPageSize(RVPrint1, w,h);
RichViewEdit1.MaxTextWidth := w-RichViewEdit1.LeftMargin-RichViewEdit1.RightMargin;
RichViewEdit1.MinTextWidth := RichViewEdit1.MaxTextWidth;
RichViewEdit1.Format;
```

1.8.5 RVGetFirstEndnote and Others

These functions allow to enumerate endnotes, footnotes, sidenotes and text box items in a document.

Unit RVNote.

```
function RVGetFirstEndnote(RichView: TCustomRichView;
  AllowHidden: Boolean): TRVEndnoteItemInfo;
function RVGetNextEndnote(RichView: TCustomRichView;
  Endnote: TRVEndnoteItemInfo; AllowHidden: Boolean): TRVEndnoteItemInfo;

function RVGetFirstFootnote(RichView: TCustomRichView;
  AllowHidden: Boolean): TRVFootnoteItemInfo;
function RVGetNextFootnote(RichView: TCustomRichView;
  Footnote: TRVFootnoteItemInfo; AllowHidden: Boolean): TRVFootnoteItemInfo;
```

(introduced in version 10)

Unit RVSidenote.

```
function RVGetFirstSidenote(RichView: TCustomRichView;
  AllowHidden: Boolean): TRVSidenoteItemInfo;
function RVGetNextSidenote(RichView: TCustomRichView;
```

```
  Sidenote: TRVSidenoteItemInfo[632]; AllowHidden: Boolean): TRVSidenoteItemInfo[632];
```

RVGetFirstEndnote returns the first endnote in **RichView** (or *nil*, if **RichView** has no endnotes).

RVGetNextEndnote returns the next endnote after **Endnote** (or *nil*, if this is the last endnote).

RVGetFirstFootnote returns the first footnote in **RichView** (or *nil*, if **RichView** has no footnotes).

RVGetNextFootnote returns the next footnote after **Footnote** (or *nil*, if this is the last footnote).

RVGetFirstSidenote returns the first sidenote or text box item in **RichView** (or *nil*, if **RichView** has no sidenote and text box items).

RVGetNextFootnote returns the next sidenote or text box item after **Sidenote** (or *nil*, if this is the last sidenote or text box item).

If **AllowHidden** parameter is *True*, the functions enumerate all notes/text boxes.

If **AllowHidden** parameter is *False*, the functions enumerate only non-hidden[76] notes/text boxes (see *rvepHidden*[673] property[673]). To enumerate only visible notes/text boxes, use *rvoShowHiddenText* in **RichView**.Options[182] as a value for this parameter.

1.8.6 RVGetNoteTextStyleNo

Returns index of text style for endnote[135] or footnote[135].

Unit RVNote.

```
function RVGetNoteTextStyleNo(RVStyle: TRVStyle[418]; StyleNo: Integer;
  StyleTemplateName: TRVStyleTemplateName[691]=''): Integer;
```

(introduced in version 10, changed in version 14)

This function returns text style (index in **RVStyle**.TextStyles[437]) which can be used for footnote or endnote characters.

If **StyleTemplateName** is not empty, and a style template with this name[496] exists in **RVStyle**.StyleTemplates[435], the function returns the index of a text style having properties of the **StyleNo**-th text style after applying this style template. Otherwise, it returns the index of a text style having all properties of RVStyle.TextStyles[437][StyleNo], but SubSuperScriptType[476] = *rvsssSuperScript*.

In all cases, Protection[473] of the returned text style is [*rvprDoNotAutoSwitch*].

If such text style does not exist, this function adds it to the end of **RVStyle**.TextStyles[437].

The recommended names of style templates to use with this function:

- 'endnote reference' for endnotes[135] and note references[140] inside endnote text[628];
- 'footnote reference' for footnotes[136] and note references[140] inside footnote text[628].

1.8.7 RVIsURL, RVIsEmail

These function return *True* if the string is URL or e-mail address.

Unit RVFuncs.

```
function RVIsURL(const s: TRVUnicodeString[694]): Boolean;
function RVIsEmail(const s: TRVUnicodeString[694]): Boolean;
```

(introduced in version 10; changed in version 18)

First, RVIsURL calls RVIsCustomURL[710]. If it returns *False*, RVIsURL checks for the following prefixes:

- http://
- ftp://
- file://
- gopher://
- mailto:

- https://
- news:
- telnet:
- wais:
- 'www.
- ftp.

In TRichView, this function is used to ignore URLs in live spelling check[101].Besides, it is used in demo projects and in ***RichViewActions***.

See also: how to create a chat window[722].

1.8.8 Unicode Conversion

Various Unicode[99]-related functions.

Unit RVUni.

```
type
  TRVCodePage = type cardinal; // defined in RVStyle unit
function RVU_AnsiToUnicode(CodePage: TRVCodePage; const s: TRVAnsiString[668]): TRVUnicodeStri
function RVU_UnicodeToAnsi(CodePage: TRVCodePage; const s: TRVUnicodeString[694]): TRVAnsiStri
function RVU_AnsiToUTF8(CodePage: TRVCodePage; const s: TRVAnsiString[668]): TRVRawByteString
```

RVU_AnsiToUnicode converts ANSI string (in the specified CodePage) to Unicode string.

RVU_UnicodeToAnsi converts Unicode string to ANSI string (in the specified CodePage).

RVU_AnsiToUTF8 converts ANSI string (in the specified CodePage) to UTF-8 string.

```
type
  TRVUnicodeTestResult = (rvutNo, rvutYes, rvutProbably, rvutEmpty, rvutError);
function RV_TestStreamUnicode(Stream: TStream): TRVUnicodeTestResult;
function RV_TestFileUnicode(const FileName: String): TRVUnicodeTestResult;
function RV_TestStringUnicode(const s: TRVRawByteString[685]): TRVUnicodeTestResult;
```

These functions do very basic test of stream, file or string content to determine if they contain Unicode or ANSI text.

Possible results:

- *rvutNo* – it is ANSI text;
- *rvutYes* – it is (very very probably) Unicode text;
- *rvutProbably* – it may be either ANSI or Unicode;
- *rvutEmpty* – empty content;
- *rvutError* – error (reading file or stream).

First, these function check number of bytes in content. If it is odd, this is not Unicode text (*rvutNo*).

Next, they check the first 2 bytes of text. If it is a Unicode byte order mark, this is very probably Unicode (*rvutYes*).

Next, they check if there are 0-bytes in the first 500 bytes. If yes, it is not an ANSI (*rvutYes*), otherwise it may be any (*rvutProbably*).

You can also use WinAPI function **IsTextUnicode** performing more advanced tests.

See also:

- Unicode in TRichView[99].

1.8.9 Functions from RVGetText Unit

The units RVGetText and RVGetTextW include the identical sets of functions returning text from TRichView.

Functions from RVGetText return ANSI strings, functions from RVGetTextW return Unicode strings.

RVGetText:

```
function GetCurrentLineText(rve: TCustomRichViewEdit[316];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetVisibleText(rv: TCustomRichView[162];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetAllText(rv: TCustomRichView[162];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetRVDataText(RVData: TCustomRVData[650];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetCurrentParaSectionText(rve: TCustomRichViewEdit[316];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetCurrentParaText(rve: TCustomRichViewEdit[316];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetCurrentChar(rve: TCustomRichViewEdit[316];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
function GetCurrentWord(rve: TCustomRichViewEdit[316];
  CodePage: TRVCodePage = CP_ACP): TRVAnsiString[668];
```

RVGetTextW:

```
function GetCurrentLineText(rve: TCustomRichViewEdit[316];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetVisibleText(rv: TCustomRichView[162];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetAllText(rv: TCustomRichView[162];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetRVDataText(RVData: TCustomRVData[650];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetCurrentParaSectionText(rve: TCustomRichViewEdit[316];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetCurrentParaText(rve: TCustomRichViewEdit[316];
  ForANSI: Boolean = False): TRVUnicodeString[694];
function GetCurrentChar(rve: TCustomRichViewEdit[316]): TRVUnicodeString[694];
function GetCurrentWord(rve: TCustomRichViewEdit[316]): TRVUnicodeString[694];
```

(changed in version 18)

GetCurrentLineText returns the current line of text (the line containing the caret).

GetAllText returns all text in rv, **GetRVDataText** returns all text in RVData.

GetCurrentParaSectionText returns text of the paragraph section with the caret (paragraph section is a part of document limited either by paragraph breaks or by new line breaks (added with Shift+Enter).

GetCurrentParaText returns text of the paragraph with the caret.

GetCurrentChar returns character to the left of the caret. If the caret is at the beginning of line, returns the character to the right. If no character, returns empty string. Length of returned string may be more than 1, if several characters compose a single glyph.

GetCurrentWord returns the word with the caret.

For non-text items, these function return their text representation.

ANSI versions of the functions have an optional CodePage parameter. It is used to convert internal Unicode text representation to ANSI. Internally, all ANSI functions call the corresponding Unicode function and convert result to Unicode.

Unicode versions of the functions have an optional ForANSI parameter. Use the default value (*False*). ANSI functions call Unicode functions passing *True* to this parameter. In this mode, some items may return simplified versions of their text, avoiding characters that cannot be converted to ANSI.

See also:

- RVGetTextRange 666

1.8.10 Functions from RVLinear Unit

```
const RVCharsPerLineBreak: Integer = 1;
      RVNonTextCharacter: TRVUnicodeChar = ' ';
```

```
function RVGetLinearCaretPos(rve: TCustomRichViewEdit316): Integer;
procedure RVSetLinearCaretPos(rve: TCustomRichViewEdit316; LinearPos: Integer);
```

RVGetLinearCaretPos returns the caret position (as a number of characters before the caret; non-text objects (except for the tables 130 and tabulators 121) are treated as one character, line breaks are treated as RVCharsPerLineBreak characters). **RVSetLinearCaretPos** moves the caret to the specified position.

```
procedure RVGetSelection(rv: TCustomRichView162; var SelStart, SelLength: Integer);
procedure RVSetSelection(rv: TCustomRichView162; SelStart, SelLength: Integer);
```

RVGetSelection returns the selection position in richedit-like parameters (SelStart and SelLength). **RVSetSelection** sets selection by these values.

```
type
  TRVSelection = record
    SelStart, SelLength: Integer;
    MultiCell: Boolean;
    StartRow, StartCol, RowOffs, ColOffs: Integer;
  end;
procedure RVGetSelectionEx(rv: TCustomRichView162; var Selection: TRVSelection);
procedure RVSetSelectionEx(rv: TCustomRichView162; const Selection: TRVSelection);
```

RVGetSelectionEx returns the selection position in TRVSelection record. **RVSetSelectionEx** sets selection by this record. These function can handle not only a normal selection, but a multicell selection in tables as well.

```
function RichViewToLinear(rv: TCustomRichView162; CurRVData, RVData: TCustomRVData650;
  ItemNo, ItemOffs: Integer; var LinearPos: Integer): Boolean;
function LinearToRichView(rv: TCustomRichView162; CurRVData: TCustomRVData650;
  var LinearPos: Integer; var RVData: TCustomRVData650;
  var ItemNo, ItemOffs: Integer): Boolean;
```

RichViewToLinear converts the position specified in RichView coordinates (RVData – document, ItemNo – index of item in this document, ItemOffs – offset in this item) to linear coordinate (a number of characters from the beginning of CurRVData before this position; non-text objects (except for the tables 130 and tabulators 121) are treated as one character, line breaks are treated as RVCharsPerLineBreak characters). **LinearToRichView** performs an opposite conversion.

```
function RVGetTextRange(rv: TCustomRichView162; RangeStart, RangeLength: Integer): TRVUnicod
function RVGetTextLength(rv: TCustomRichView162): Integer;
```

RVGetTextRange returns text from rv, from RangeStart to RangeLength position. Non-text items (except for the tables 130 and tabulators 121) are returned as RVNonTextCharacter. If RVCharsPerLineBreak=1, #13 is used as line break. If RVCharsPerLineBreak=2, #13#10 is used as line break.

The call RVGetTextRange(rv, 0, RVGetTextLength(rv)) returns the whole text.

All these functions are compatible, they calculate position identically. The text returned by RVGetTextRange has one-to-one correspondence to RichView document (unlike text returned by the functions from RVGetText[665] unit).

1.8.11 Functions from RVMathItem Unit

```
procedure RVGetMathProperties(rv: TCustomRichView[162]; var FontName: TFontName;
  var FontSizeDouble: Integer; var TextColor: TColor);
procedure RVSetMathProperties(rv: TCustomRichView[162];
  const FontName: String; FontSizeDouble: Integer; TextColor: TColor;
  EditMode: Boolean);
```

RVGetMathProperties returns default values of properties of mathematical expressions[143]. If **rv**.DocObjects[174] has an item of TRVMathDocObject[656] class, it returns values of its properties. Otherwise, it returns 'Cambria Math', 20, *clWindowText*.

If **rv**.DocObjects has several TRVMathDocObject items, the procedure returns properties of the first of them. Normally, the collection must have no more than one TRVMathDocObject item.

RVSetMathProperties assigns default values of properties of mathematical expressions. It does it by assigning properties of TRVMathDocObject item of **rv**.DocObjects[174] collection. If such item does not exist, the procedure creates it. If **EditMode** = *True* and (**rv** is TCustomRichViewEdit[316]), this assignment is performed as an editing operation (undoable).

```
procedure RVApplyMathProperties(rv: TCustomRichView[162]);
```

RVApplyMathProperties updates mathematical expressions after changing properties of TRVMathDocObject[656] item. You must call this function if you assigned properties of this item directly. It is not needed if you use **RVSetMathProperties**.

1.8.12 Functions for TRVUnits conversion

unit RVFuncs;

```
function RV_UnitsPerInch(Units: TRVUnits[694]): TRVLength[682];
function RV_UnitsToTwips(Value: TRVLength[682]; Units: TRVUnits[694]): Integer;
function RV_TwipsToUnits(Value: Integer; Units: TRVUnits[694]): TRVLength[682];
function RV_UnitsToUnits(Value: TRVLength; OldUnits, NewUnits: TRVUnits[694]): TRVLength[682];
function RV_UnitsToPixels(Value: TRVLength[682]; Units: TRVUnits[694]; PixelsPerInch: Integer)
```

RV_UnitsPerInch returns the amount of Units in 1 inch.

RV_UnitsToTwips converts Value from Units to twips.

RV_TwipsToUnits converts Value from twips to Units.

RV_UnitsToUnits converts Value from OldUnits to NewUnits.

RV_UnitsToPixels converts Value from Units to pixels in the specified resolution.

See also

- Units of measurement in TRichView[106]

1.9 Types

1.9.1 TCheckpointData

Unit RVStyle;

```
type
  TCheckpointData = type Pointer;
```

Value of this type identifies one *checkpoint* [65].

It can be obtained as a parameter of several events, or as a return value of several methods of TRichView.

All properties of checkpoint can be extracted from this value using other RichView methods.

See "Checkpoints" [65] for details.

1.9.2 TRVLoadFormat

Unit RVStyle;

```
type
  TRVLoadFormat =
    (rvlfText, rvlfHTML, rvlfRTF, rvlfRVF, rvlfURL, rvlfOther);
```

Loading format.

See also:

- TRichView.OnReadHyperlink [284]

1.9.3 TRVAnsiString

Unit RVTypes.

Declaration for Delphi 2009 or newer:

```
type
  TRVAnsiString = type AnsiString;
```

Declaration for older versions of Delphi:

```
type
  TRVAnsiString = type String;
```

Strings of this type contain ANSI text.

See also

- TRVUnicodeString [694]
- TRVRawByteString [685]

1.9.4 TRVBiDiMode

Unit RVStyle;

```
type
  TRVBiDiMode =
    (rvbdUnspecified, rvbdLeftToRight, rvbdRightToLeft);
```

Value of this type defines default text reading direction for document (TRichView.BiDiMode [170]), paragraph (TParaInfo.BiDiMode [484]), or text (TFontInfo.BiDiMode [467]).

Value	Meaning
rvbdUnspecified	Bidirectional text is not supported (if specified for document), or text direction is inherited from higher level: ▪ for text – from paragraph; ▪ for paragraph – from document.
rvbdLeftToRight	Default text reading order is left-to-right.
rvbdRightToLeft	Default text reading order is right-to-left.

See also:

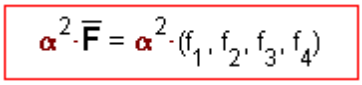
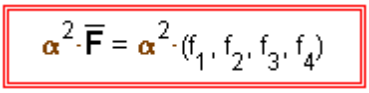
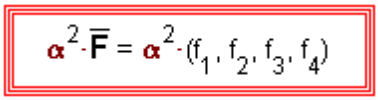
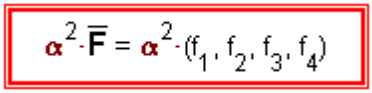
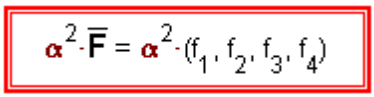
Bidirectional text in TRichView[100]

1.9.5 TRVBorderStyle

Unit RVStyle

```
type
  TRVBorderStyle = (rvbNone, rvbSingle, rvbDouble, rvbTriple,
    rvbThickInside, rvbThickOutside);
```

Style of border around paragraph[73] or around a text box (of a sidenote[137] or a text box item[139]). Type for Border[485].Style[502] property.

Style	Border
rvbNone	No border
rvbSingle	
rvbDouble	
rvbTriple	
rvbThickInside	
rvbThickOutside	

(examples are made with Color[502]=*clRed*, Width[503]=1, InternalWidth[502]=1)

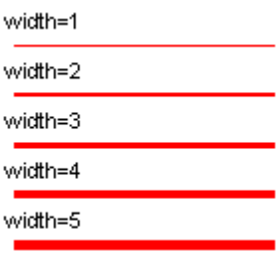
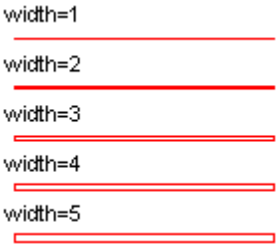
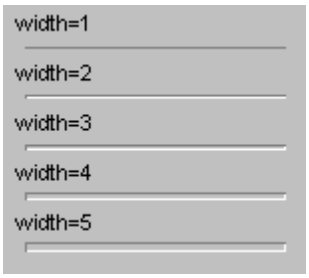
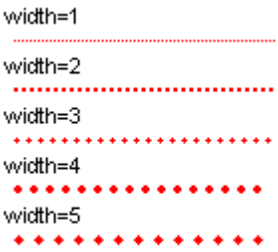
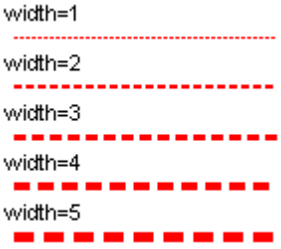
1.9.6 TRVBreakStyle

Unit RVStyle

```
type
  TRVBreakStyle =
    (rvbsLine, rvbsRectangle,
     rvbs3d, rvbsDotted, rvbsDashed);
```

Style of *break*[125] item.

Style	Example

rvbsLine	
rvbsRectangle	
rvbs3d	 (break color is ignored)
rvbsDotted	
rvbsDashed	

1.9.7 TRVCellVAlign

Unit RVTable;

```
type
  TRVCellVAlign =
    (rvcTop,  rvcMiddle, rvcBottom, rvcVDefault);
```

Types vertical alignment in cells.

See VAlign[616] property of table rows and VAlign[613] property of cells.

1.9.8 TRVCodePage

Unit RVStyle;

```
type
  TRVCodePage = type Cardinal;
```

A type for code page identifiers.

1.9.9 TRVColorMode

Unit RVStyle;

```
type
  TRVColorMode = (rvcmColor, rvcmVCLThemeColor,rvcmPrinterColor,
    rvcmGrayScale, rvcmBlackAndWhite, rvcmBlackOnWhite);
```

Mode	Meaning
rvcmColor	Color output.
rvcmVCLThemeColor	For Delphi XE2+, use system colors of the active VCL theme instead of Windows system colors. For older version of Delphi, the same as *rvcmColor*.
rvcmPrinterColor	Color output, but the following colors are changed: ▪ *clWindow*, *clBtnHighlight* to *clWhite*, ▪ *clWindowText* to *clBlack*, ▪ *clBtnShadow* to *clGray*, ▪ *clBtnFace*, *clScrollbar* to *clSilver*.
rvcmGrayScale	Colors are converted to shades of gray.
rvcmBlackAndWhite	Paragraphs backgrounds are not printed; text is printed black on white or white on black depending on what is darker.
rvcmBlackOnWhite	Paragraphs backgrounds are not printed; text is printed black on white.

In the current version of the components, color mode affects drawing and printing text, paragraph background, *breaks*[125], list markers[131], tables[130].

It does not affect drawing and printing pictures.

1.9.10 TRVDisplayOptions

Unit RVScroll

```
type
  TRVDisplayOption = (rvdoImages, rvdoComponents, rvdoBullets);
  TRVDisplayOptions = set of TRVDisplayOption;
```

```
const
  rvdoALL = [rvdoImages, rvdoComponents, rvdoBullets];
```

1.9.11 TRVDocRotation

Unit RVStyle;

```
type
  TRVDocRotation = (rvrotNone, rvrot90, rvrot180, rvrot270);
```

Possible values for a table cell rotation.

Value	Meaning
rvrotNone	Not rotated (0°)
rvrot90	Rotated by 90°
rvrot180	Rotated by 180°
rvrot270	Rotated by 270°

1.9.12 TRVESearchOptions

Unit RVEdit;

```
type
  TRVESearchOption = (rvseoMatchCase, rvseoDown, rvseoWholeWord,
    rvseoMultiItem, rvseoSmartStart);
  TRVESearchOptions = set of TRVESearchOption;
```

Options for searching in RichViewEdit, see TCustomRichViewEdit.SearchText [369].

Value	Meaning
rvseoMatchCase	If included, character case is taken into account when comparing string.
rvseoDown	If included, search is performed to the end of document. If not included, it is performed to the top of document.
rvseoWholeWord	If included, the searched string matches only whole words.
rvseoMultiItem	If included, the search can match substrings of several text items [120]. If not included, the text is searched in each text item separately. For example, if the document contains the text **He*llo***, the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.
rvseoSmartStart	If included, and a document fragment is selected, then: ▪ when searching down, the search starts from the second selected character; ▪ when searching up, the search starts from the last but one selected character (selected characters are counted from the top of the document, the selection direction is ignored)

1.9.13 TRVExtraItemProperty

Unit RVItem;

```
type
 TRVExtraItemProperty =
  (rvepUnknown, rvepVShift, rvepVShiftAbs,
   rvepImageWidth, rvepImageHeight,
   rvepTransparent, rvepTransparentMode,
   rvepTransparentColor, rvepMinHeightOnPage,
   rvepSpacing, rvepResizable,
   rvepDeleteProtect, rvepNoHTMLImageSize,
   rvepAnimationInterval, rvepVisible,
   rvepHidden, rvepShared,
   rvepBorderWidth, rvepColor, rvepBorderColor,
   rvepOuterHSpacing, rvepOuterVSpacing,
   rvepVAlign);
```

Identifies an item property of integer type.

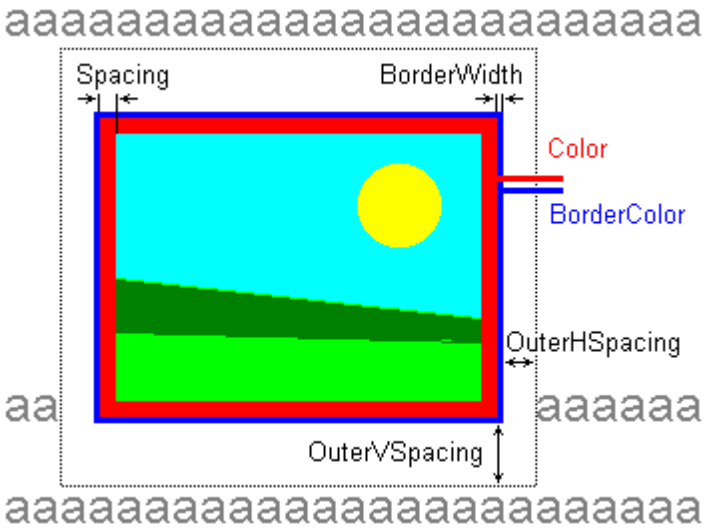
Positioning

Property	Applicable To	Default Value	Meaning
rvepVShift	rect.items*	0	Vertical offset, in % of item height (default), or in RVStyle.Units[437] (see *rvepVShiftAbs*). Negative values move the item down, positive move it up (relative to the position specified in VAlign[695])
rvepVShiftAbs	rect.items*	0	If non-zero, *rvepVShift* defines vertical offset in RVStyle.Units[437] instead of %.
rvepVAlign	rect.items*	ord(*rvvaBaseline*)	Vertical alignment, TRVVAlign[695] converted to integer.

* "rect.items" include pictures[122], *hot-pictures*[125], controls[126], *bullets*[128], *hotspots*[129], labels[133], sequences[134], footnotes[136], endnotes[135], note references[140].

Borders and spacing

Scheme:



Property	Applicable To	Default Value	Meaning
rvepSpacing	rect.items*	1	Spacing around the item content, in RVStyle.Units[437]. Must be a zero or a positive value. This is a padding, a spacing between the item content and the item border. This area may be colored with *rvepColor*.
rvepColor	rect.items*	*clNone*	Background color. It is drawn below the image (and around the image, if *rvepSpacing* is positive).
rvepBorderWidth	rect.items*	0	Border width, in RVStyle.Units[437]. Must be a zero or a positive value.
rvepBorderColor	rect.items*	*clBlack*	Border color.
rvepOuterHSpacing *rvepOuterVSpacing*	rect.items*	0	Spacing around the border (horizontal and vertical), in RVStyle.Units[437]. Must be a zero or a positive value.

* "rect.items" include pictures[122], *hot-pictures*[125], controls[126], *bullets*[128], *hotspots*[129], labels[133], sequences[134], footnotes[136], endnotes[135], note references[140].

Pictures

Property	Applicable To	Default Value	Meaning
rvepImageWidth, *rvepImageHeight*	▪ pictures[122] ▪ *hot-pictures*[125]	0	If set to positive value, define size of the image in RVStyle.Units[437]. The actual image is not modified, it is just drawn stretched. See also *rvepAnimationInterval* property below.
rvepTransparent, *rvepTransparentMode*, *rvepTransparentColor*	▪ pictures[122] ▪ *hot-pictures*[125]		Correspond to properties of TBitmap: Transparent, TransparentMode, TransparentColor. Limitations:

	(if they are TBitmap)		▪ transparency is turned off in 256-color mode; ▪ transparency is not supported when printing (although print preview shows transparent bitmaps) ▪ TBitmap implementation of transparency is not ideal; sometimes it may cause weird colors for 256-color images (probably, it was fixed in last versions of Delphi)
rvepMinHeightOnPage	▪ pictures 122 ▪ *hot-pictures* 125	0	Affects printing, allows printing large pictures on several pages. Positive value defines minimal height of part of image allowed to be on page. For example: ▪ 1 – the image can be printed on two or more pages, it can separated between pages in any place; ▪ 100 – if the rest of page can contain >= 100 pixels of the image, the image will be separated, otherwise it will be printed on single page; ▪ 0 – the image/control will be printed on single page. Do not overuse this feature, it has limitations: ▪ such images/controls are always printed from new line; ▪ for control, OnPrintComponent 554 must return bitmap having exactly the size of the control (if this property = 0, you can return larger bitmap and it will be stretched allowing to create a better printing output)
rvepNoHTMLImageSize	▪ pictures 122 ▪ *hot-pictures* 125 ▪ *bullets* 128 ▪ *hotspots* 129 ▪ list markers 131	0	If non-zero, image size is not saved in HTML, even if *rvsoImageSizes* is included in the Options parameter for SaveHTML 250/SaveHTMLEx 251 This option is ignored for pictures 122 and *hot-pictures* 125, if *rvepImageWidth* or *rvepImageHeight* properties are non-zero.
rvepAnimationInterval	▪ pictures 122 ▪ *hot-pictures* 125 (if they are TBitmap)	0	Animation delay in 1/100 of second. If nonzero and *rvepImageWidth* and/or *rvepImageHeight* are defined, playing bitmap animation 90 (in imagewidth x imageheight frame)

Editing

Property	Applicable To	Default Value	Meaning
rvepResizable	▪ pictures 122 ▪ *hot-pictures* 125 ▪ controls 126	1 (for pictures and *hot-pictures*), 0 (for controls).	Allows resizing with mouse ▪ 0 – no resizing; ▪ non-zero value – resizing is allowed.
rvepDeleteProtect	all non-text items	0	▪ 0 – no protection; ▪ non-zero value – item cannot be deleted in editing operations.

			See also: *rvprDeleteProtect* option of Protection[473] for text style.

Other

Property	Applicable To	Default Value	Meaning
rvepVisible	▪ controls[126]	1	Set to 0 to hide the control (it still occupies its place, but is not displayed)
rvepHidden	all non-text items	0	If non-zero, this item is hidden[76].
rvepShared	▪ pictures[122] ▪ *hot-pictures*[125] ▪ controls[126]	0	If non-zero, the item content (picture or control) is not deleted when the item is deleted.

See also

- TRVExtraItemStrProperty[676]

See also methods of TRichView

- SetItemExtraIntProperty[266],
- GetItemExtraIntProperty[227].

See also methods of TRichViewEdit

- SetItemExtraIntPropertyEd[383],
- SetCurrentItemExtraIntProperty[378],
- GetCurrentItemExtraIntProperty[346].

1.9.14 TRVExtraItemStrProperty

Unit RVItem;

```
type
  TRVExtraItemStrProperty = (
    rvespUnknown,
    rvespHint,
    rvespAlt,
    rvespImageFileName,
    rvespTag);
```

Identifies item property of string type.

Default value for all properties is '' (empty string).

Property	Applicable To	Meaning
rvespUnknown		Not a property. For internal use.
rvespAlt	▪ pictures[122] ▪ *hot-pictures*[125] ▪ *bullets*[128] ▪ *hotspots*[129]	Text representation of image. Saved as "<img alt>" attribute in HTML files. Must not contain line breaks and double quotes.
rvespHint	all items (but not shown for controls and tables)	Popup hint can be displayed for items if *rvoShowItemHints* is included in RichView.Options[182] and RichView.ShowHint=*True*. Hints are displayed by assigning to RichView.Hint. See also OnItemHint[281] event.

		Note: this feature adds 4 bytes in memory to each item. If you do not need it, enable RVDONOTUSEITEMHINTS compiler define in RV_Defs.inc. After enabling this define, you can still use OnItemHint[281]. Hint string must not contain line breaks. When exporting to HTML, it is exported as "title" attribute. For RTF, hints are exported and imported only for hyperlinks. In OnWriteHyperlink[302] event, the proper value of Extras parameter is set by default.
rvespImageFileName	▪ pictures[122] ▪ *hot-pictures*[125] ▪ tables[130]	Provides a place for storing image file name. For tables, it is a background image, also accessible as table.BackgroundImageFileName[568]. See also: TRVTableCellData.BackgroundImageFileName[606] If *rvsoUseItemImageFileNames* is included in the Options parameter of SaveHTML[250]/SaveHTMLEx[251], images with defined (non-empty) file names will not be saved, but their file names will be written in HTML (paths will be converted to relative paths). In some cases, the component may assign this property, if *rvoAssignImageFileNames* is included in TRichView.Options[182].
rvespTag	all items	tag[68]

See also

- TRVExtraItemProperty[673]

See also methods of TRichView

- SetItemExtraStrProperty[266],
- GetItemExtraStrProperty[228].

See also methods of TRichViewEdit

- SetItemExtraStrPropertyEd[384],
- SetCurrentItemExtraStrProperty[379],
- GetCurrentItemExtraStrProperty[346].

1.9.15 TRVFloatPosition

Unit RVFloatingPos;

```
type
  TRVFloatPosition = type Integer;
```

This is a type of HorizontalOffset[639] and VerticalOffset[642] properties of TRVBoxPosition[637].

Meaning of these values depend on HorizontalPositionKind[639] and VerticalPositionKind[642] properties, and may mean either values in TRVStyleUnits[691] or in (1/1000)%.

1.9.16 TRVFloatPositionKind

Unit RVFloatingPos;

Values of this type define how a position of a floating box is calculated.

```
type
  TRVFloatPositionKind = (rvfpkAlignment, rvfpkAbsPosition, rvfpkPercentPosition);
```

This is a type of the following properties of TRVBoxPosition[637]:

- HorizontalPositionKind[639]
- VerticalPositionKind[642]

Value	Meaning
rvfpkAlignment	A position is specified in HorizontalAlignment[639]/VerticalAlignment[642]
rvfpkAbsPosition	HorizontalOffset[639]/VerticalOffset[642] contains offset in RVStyle.Units[437] (pixels or twips)
rvfpkPercentPosition	HorizontalOffset[639]/VerticalOffset[642] contains offset in (1/1000)% of the anchor area size

1.9.17 TRVFloatProperty

Unit RichView.

Identifies a property of TRichView[162] of a TRVLength[682] type.

```
type
  TRVFloatProperty = (rvfpDPLeftMargin, rvfpDPRightMargin, rvfpDPTopMargin,
    rvfpDPBottomMargin, rvfpDPHeaderY, rvfpDPFooterY);
```

Value	Corresponding property of TRichView
rvfpDPLeftMargin	DocParameters[176].LeftMargin[312]
rvfpDPRightMargin	DocParameters.RightMargin[313]
rvfpDPTopMargin	DocParameters.TopMargin[314]
rvfpDPBottomMargin	DocParameters.BottomMargin[311]
rvfpDPHeaderY	DocParameters.HeaderY[312]
rvfpDPFooterY	DocParameters.FooterY[312]

A parameter of this type is used in TRichViewEdit[316].SetFloatPropertyEd[370].

See also:

- TRVIntProperty[681];
- TRVStrProperty[690].

1.9.18 TRVFloatSize

Unit RVFloatingBox;

```
type
  TRVFloatSize = type Integer;
```

This is a type of Width[636] and Height[636] properties of TRVBoxProperties[635].

Meaning of these values depend on WidthType[636] and HeightType[636] properties, and may mean either values in TRVStyleUnits[691] or in (1/1000)%.

1.9.19 TRVFontInfoProperties

Unit RVStyle;

```
type
TRVFontInfoProperty =
   (rvfiFontName, rvfiSize, rvfiCharset, rvfiUnicode,
    rvfiBold, rvfiItalic, rvfiUnderline, rvfiStrikeout, rvfiOverline,
    rvfiAllCaps, rvfiSubSuperScriptType, rvfiVShift, rvfiColor, rvfiBackColor,
    rvfiJump, rvfiHoverBackColor, rvfiHoverColor, rvfiHoverUnderline,
    rvfiJumpCursor, rvfiNextStyleNo, rvfiProtection, rvfiCharScale,
    rvfiBaseStyleNo, rvfiBiDiMode, rvfiCharSpacing, rvfiHTMLCode, rvfiRTFCode,
    rvfiDocXCode, rvfiHidden,
    {$IFDEF RVLANGUAGEPROPERTY}
    rvfiLanguage,
    {$ENDIF}
    rvfiUnderlineType, rvfiUnderlineColor, rvfiHoverUnderlineColor,
    rvfiEmptyWidth,
    rvfiCustom, rvfiCustom2, rvfiCustom3, rvfiCustom4, rvfiCustom5);

TRVFontInfoProperties = set of TRVFontInfoProperty;
```

Identifies properties of TFontInfo[478].

This is a type of the following properties:

- TRVStyleTemplate.ValidTextProperties[498];
- TFontInfo.ModifiedProperties[480].

Parameters of this type are used in the following methods:

- TFontInfos.FindSuchStyle, FindSuchStyleEx[455];
- TFontInfo.IsEqual[482].

```
const
  RVAllFontInfoProperties: TRVFontInfoProperties =
    [Low(TRVFontInfoProperty)..High(TRVFontInfoProperty)];
```

Constant, lists all TFontInfo properties.

1.9.20 TRVFReaderStyleMode

Unit RVStyle;

```
type
  TRVFReaderStyleMode =
  (rvf_sIgnore, rvf_sInsertMap, rvf_sInsertMerge)
```

Defines a mode for loading collection of styles from RVF[93] (if they are contained there, see *rvfoSaveTextStyles* and *rvfoSaveParaStyles* RVFOptions[188]).

This is a type of the following properties of TCustomRichView:

- RVFTextStylesReadMode[191] (loading Style[193].TextStyles[437]);
- RVFParaStylesReadMode[190] (loading Style.ParaStyles[431] and Style.ListStyles[429]).

Mode	When loading RVF...	When Inserting RVF...

rvf_sIgnore	Collections of styles in RVF are ignored. For correct loading, this RVF file must be saved with the same collections of styles.	
rvf_sInsertMerge	Current collection of styles will be replaced with values from RVF.	Current styles will be merged with styles from RVF (new styles will be added to the collection, if necessary).
rvf_sInsertMap	Current collection of styles will be replaced with values from RVF.	Styles from RVF will be mapped to the most similar existing styles. New styles will not be added.

You should be careful with the last two modes: it's recommended to use them only if TRVStyle[418] component is used exclusively by the given TRichView[Edit].

If style templates are used[195], and RVFTextStylesReadMode[191]=RVFParaStylesReadMode[190] =*rvf_sInsertMerge*, and StyleTemplateInsertMode[193]<>*rvstimIgnoreSourceStyleTemplates*, the RVF loading methods replace the existing Style[193].StyleTemplates[435] with the style templates read from RVF; the RVF insertion methods merge the collection of style templates from RVF into the existing style templates, and read text and paragraph styles according to StyleTemplateInsertMode[193].

Loading operations include:

- LoadRVF[245],
- LoadRVFFromStream[244],
- loading from database fields[407].

Insertion operations include:

- InsertRVFFromStream[240],
- AppendRVFFromStream[208],
- TRichViewEdit.InsertRVFFromFileEd[357],
- TRichViewEdit.InsertRVFFromStreamEd[358],
- pasting[82] from the Clipboard,
- drag&drop[89].

1.9.21 TRVHFType

Type of header or footer

```
type
  TRVHFType = (rvhftNormal, rvhftFirstPage, rvhftEvenPages);
```

Value	Meaning
rvhftNormal	Header/footer for all pages (except for pages having a different header/footer specified)
rvhftFirstPage	Header/footer for the first page
rvhftEvenPages	Header/footer for even pages

1.9.22 TRVHTMLLength

Unit RVTable;

```
type
  TRVHTMLLength = type Integer;
```

Represents length either in percent or in TRVStyleUnits[691].

Range	Meaning
>0	Length in TRVStyleUnits 691 (twips or screen pixels)
<0	Length in percent (for example, -50 means 50%)
=0	Length is undefined (default)

See also

- TRVTableItemInfo.BestWidth 568
- TRVTableCellData.BestWidth 607

1.9.23 TRVIntProperty

Unit RichView.

Identifies a property of TRichView 162 of an integer or an ordinal type.

```
type
  TRVIntProperty = (rvipLeftMargin, rvipRightMargin, rvipTopMargin,
    rvipBottomMargin, rvipMaxTextWidth, rvipMinTextWidth, rvipBackgroundStyle,
    rvipColor, rvipBiDiMode, rvipDPUnits, rvipDPOrientation,
    rvipDPTitlePage, rvipDPFacingPages, rvipDPMirrorMargins);
```

Value	Corresponding property of TRichView
rvipLeftMargin	LeftMargin 180
rvipRightMargin	RightMargin 185
rvipTopMargin	TopMargin 194
rvipBottomMargin	BottomMargin 171
rvipMaxTextWidth	MaxTextWidth 181
rvipMinTextWidth	MinTextWidth 181
rvipBackgroundStyle	BackgroundStyle 170
rvipColor	Color 173
rvipBiDiMode	BiDiMode 170
rvipDPUnits	DocParameters 176.Units 314
rvipDPOrientation	DocParameters.Orientation 313
rvipDPTitlePage	DocParameters.TitlePage 314
rvipDPFacingPages	DocParameters.FacingPages 312
rvipDPMirrorMargins	DocParameters.MirrorMargin 313

A parameter of this type is used in TRichViewEdit 316.SetIntPropertyEd 370.

See also:

- TRVFloatProperty 678;
- TRVStrProperty 690.

1.9.24 TRVItemBackgroundStyle

Unit RVStyle;

```
type
  TRVItemBackgroundStyle =
    (rvbsColor, rvbsStretched, rvbsTiled, rvbsCentered);
```

Defines background style (position of background image).

Value	Meaning
rvbsColor	No image.
rvbsStretched	Image stretched to fill the whole area. Not all image types can be stretched.
rvbsTiled	Image is repeated to fill the whole area.
rvbsCentered	Image is located at the center.

See also:

- TRVTableItemInfo.BackgroundStyle[568] (background of table)
- TRVTableCellData.BackgroundStyle[606] (background of cell)
- Smooth image scaling[91]

1.9.25 TRVLength

Unit RVStyle;

```
type
  TRVLength = type Extended;
```

This is a type of values measured in TRVUnits[694].

1.9.26 TRVOleDropEffects

Type of drag&drop operation.

```
type
  TRVOleDropEffect = (rvdeNone, rvdeCopy, rvdeMove, rvdeLink);
  TRVOleDropEffects = set of TRVOleDropEffect;
```

This type defines how drag&drop cursor looks like.

Effect	Meaning
rvdeNone	Drag&drop is not possible
rvdeCopy	The source object will be copied to the target
rvdeMove	The source object will be moved to the target
rvdeLink	Link to the source object will be inserted in the target

See events of TCustomRichViewEdit:

- OnOleDragEnter[395];
- OnOleDragOver[396];
- OnOleDrop[397].

1.9.27 TRVOpacity

Unit RVStyle;

```
type
  TRVOpacity = type Integer;
```

Specifies the object opacity in 1/1000 of percent. Valid values are in range 0..100 000. The larger the value, the less transparent the object is.

For example:

- 0 means full transparency, the object is invisible
- 100 000 means that the object is opaque
- 50 500 means 50.5% opacity.

Physically, only 256 gradations of transparency is supported by screen and other devices. The higher precision range is chosen for the convenience of specifying opacity in fractions of percent.

1.9.28 TRVPageNumberType

Unit RVFieldItems.

```
type
  TRVPageNumberType = (rvpntDefault, rvpntDecimal, rvpntDecimalDash,
    rvpntLowerAlpha, rvpntUpperAlpha,
    rvpntLowerRoman, rvpntUpperRoman);
```

Numbering type for "page number" field[141].

Type	Meaning
rvpntDefault	Default numbering. Displayed as decimal.
rvpntDecimal	1, 2, 3,...
rvpntDecimalDash	- 1 -, - 2 -, - 3 -
rvpntLowerAlpha	a, b, c, ..., z, aa, ab, ...
rvpntUpperAlpha	A, B, C, ..., Z, AA, AB, ...
rvpntLowerRoman	i, ii, iii, iv, ...
rvpntUpperRoman	I, II, III, IV, ...

This is a type of the following properties:

- TRVPageNumberType.NumberType[646].

1.9.29 TRVPageSet

Unit RVStyle;

```
type
  TRVPageSet = (rvpsAll, rvpsOdd, rvpsEven);
```

Set of pages.

Value	Meaning
rvpsAll	All pages
rvpsOdd	Odd pages

rvpsEven	Even pages

See also:

- TRVPrint[513].Print[526]
- TRVPrint.PrintPages[527]

1.9.30 TRVParaInfoProperties

Unit RVStyle;

```
type
  TRVParaInfoProperty = (
    rvpiFirstIndent, rvpiLeftIndent, rvpiRightIndent,
    rvpiSpaceBefore, rvpiSpaceAfter, rvpiAlignment, rvpiLastLineAlignment,
    rvpiNextParaNo, rvpiDefStyleNo, rvpiLineSpacing, rvpiLineSpacingType,
    rvpiBackground_Color,
    rvpiBackground_BO_Left, rvpiBackground_BO_Top,
    rvpiBackground_BO_Right, rvpiBackground_BO_Bottom,
    rvpiBorder_Color, rvpiBorder_Style,
    rvpiBorder_Width, rvpiBorder_InternalWidth,
    rvpiBorder_BO_Left, rvpiBorder_BO_Top,
    rvpiBorder_BO_Right, rvpiBorder_BO_Bottom,
    rvpiBorder_Vis_Left, rvpiBorder_Vis_Top,
    rvpiBorder_Vis_Right, rvpiBorder_Vis_Bottom,
    rvpiNoWrap, rvpiReadOnly, rvpiStyleProtect, rvpiDoNotWantReturns,
    rvpiKeepLinesTogether, rvpiKeepWithNext, rvpiWidowOrphanControl,
    rvpiTabs, rvpiBiDiMode, rvpiOutlineLevel,
    rvpiCustom, rvpiCustom2, rvpiCustom3, rvpiCustom4, rvpiCustom5);

TRVParaInfoProperties = set of TRVParaInfoProperty;
```

Identifies properties of TParaInfo[482].

This is a type of the following properties:

- TRVStyleTemplate.ValidParaProperties[498];
- TParaInfo.ModifiedProperties[491].

Parameters of this type are used in the following methods:

- TParaInfos.FindSuchStyle, FindSuchStyleEx[458];
- TParaInfo.IsEqual[492].

```
const
  RVAllParaInfoProperties: TRVParaInfoProperties =
    [Low(TRVParaInfoProperty)..High(TRVParaInfoProperty)];
```

A constant, lists all TParaInfo properties.

1.9.31 TRVPrintingStep

Unit PtblRV

```
type
  TRVPrintingStep = (rvpsStarting, rvpsProceeding, rvpsFinished);
```

Parameter of this type is used in OnSendingToPrinter[531] and OnFormatting[529] events.

1.9.32 TRVRawByteString

Unit RVTypes.

Declaration for Delphi 2009 or newer:

```
type
  TRVRawByteString = type RawByteString;
```

Declaration for older versions of Delphi:

```
type
  TRVRawByteString = type String;
```

String of this type may contain arbitrary data, not necessary an ANSI text.

See also

- TRVAnsiString 668
- TRVUnicodeString 694

1.9.33 TRVReaderStyleMode

Unit RVStyle;

```
type
  TRVReaderStyleMode =
    (rvrsUseSpecified, rvrsUseClosest, rvrsAddIfNeeded);
```

Defines modes for loading formatting from RTF.

Mode	Meaning
rvrsUseSpecified	All character/paragraph formatting in RTF will be ignored, all text/paragraphs will be formatted with the specified style.
rvrsUseClosest	RTF formatting will be mapped to the the most similar existing style. New styles will not be added.
rvrsAddIfNeeded	This mode provides the most closer look to the original RTF. RichView will try to use existing styles as much as possible, but if style with the desired formatting does not exist, RichView will create and use a new style.

See TRichView.RTFReadProperties.TextStyleMode 309 and ParaStyleMode 308.

1.9.34 TRVSaveFormat

Unit RVStyle;

```
type
  TRVSaveFormat =
  (rvsfText, rvsfHTML, rvsfRTF, rvsfRVF, rvsfDocX);
```

Identifies saving format.

This type is used in some events of RichView, such as OnSaveComponentToFile 292 and OnSaveItemToFile 297, and SavePicture 253 method.

1.9.35 TRVSaveOptions

Unit RVStyle;

```
type
  TRVSaveOption = (rvsoOverrideImages,
    rvsoFirstOnly, rvsoMiddleOnly, rvsoLastOnly,
    rvsoDefault0Style, rvsoNoHypertextImageBorders,
    rvsoImageSizes, rvsoForceNonTextCSS, rvsoUseCheckpointsNames,
    rvsoMarkersAsText, rvsoNoDefCSSStyle, rvsoInlineCSS,
    rvsoUseItemImageFileNames,
    rvsoXHTML, rvsoUTF8, rvsoHeaderFooter);
  TRVSaveOptions = set of TRVSaveOption;
```

Options for HTML export. Used for the parameter of SaveHTML[250], SaveHTMLEx[251], SaveHTMLToStream[252], SaveHTMLToStreamEx[252].

Options for saving images, both for SaveHTML[250], SaveHTMLEx[251]

Option	Meaning
rvsoOverrideImages	If set, TRichView can override existing images. If not set, TRichView will generate unique names for images.
rvsoNoHypertextImageBorders	By default, browsers add borders around hypertext images. This option suppresses borders for *hotspots*[129] and *hot-pictures*[125].
rvsoImageSizes	If set, TRichView saves widths and heights of images in HTML explicitly. This option provides more smooth (and sometimes faster) loading by browsers. This option does not affect pictures having defined *rvepImageWidth* and/or *rvepImageHeight* item extra integer properties[673]. Sizes are always saved for such images. You can override this setting for the specific item using *rveoNoHTMLImagesSize* item extra integer properties[673] .
rvsoUseItemImageFileNames	If included, images with defined (non-empty) file names will not be saved, but their file names will be written in HTML. If this option is included, OnHTMLSaveImage[277] and OnSaveImage2[296] events have initial value of the Location parameter equal to the image file name. File names for items are defined via *rvespImageFileName*[676] string property[676].
rvsoHeaderFooter	If included, a header is saved at the beginning of HTML document, and a footer is saved at the end. Headers and footers are defined by SetHeader[264]/SetFooter[263] methods. If DocParameters[176].TitlePage[314] = *True*, the component uses a header and a footer for the first page. Otherwise, a normal header and footer are used. For SaveHTMLEx, headers and footers are always use inline CSS (as if *rvsoInlineCSS* is included).

Options for saving list markers, both for SaveHTML[250], SaveHTMLEx[251]

Option	Meaning
rvsoMarkersAsText	If included, paragraph lists[131] are saved as normal text or images. Usually, HTML files saved with this option look closer to the original document. If not included, lists are saved as and HTML tags.

If *rvsoMarkersAsText* is not included

Export using CSS (SaveHTMLEx[251]); CSS of lists are inserted directly in HTML:

- *rvlstBullet, rvlstUnicodeBullet*[510] – exported as bullets with default markers (disc, circle or square). FormatString[508] and Font[507] are ignored;
- *rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman*[510] – exported as lists with the proper numbering. FormatString[508] and Font[507] are ignored;
- *rvlstPicture,rvlstImageList,rvlstImageListCounter*[510] – exported as a bullet with picture. It's possible to change default image saving by OnHTMLSaveImage[277];
- indents are saved, but not very accurately.

Export without CSS (SaveHTML[250]):

- *rvlstBullet, rvlstUnicodeBullet,rvlstPicture,rvlstImageList*[510] – exported as bullets with default markers (disc, circle or square). FormatString[508], Picture[513], ImageList[509] and Font[507] are ignored;
- *rvlstDecimal,rvlstLowerAlpha,rvlstUpperAlpha,rvlstLowerRoman,rvlstUpperRoman*[510] – exported as lists with the proper numbering. FormatString[508] and Font[507] are ignored;
- *rvlstImageListCounter*[510] – saved as a decimal numbering;
- indents are not saved.

If *rvsoMarkersAsText* is included

Markers are saved without special HTML keywords for lists (like <ol>, <ul>, <li>).

Font[507] and format strings[508] settings are respected.

Saving without CSS (SaveHTML[250]): all indents are ignored.

Saving with CSS (SaveHTMLEx[251]): LeftIndent[510] and MarkerIndent[512] are retained. FirstIndent[507] are retained for non-hanging markers only (MarkerIndent>=LeftIndent)..

Generally, HTML files saved with this option look closer to the original.

Other options, both for SaveHTML[250], SaveHTMLEx[251]

Option	Meaning
rvsoUseCheckpointsNames	*Checkpoints*[65]' names are used instead of checkpoint indices. Recommended.
rvsoXHTML	If included, XHTML is saved instead of HTML
rvsoUTF8	If included, Unicode (UTF-8) HTML file is saved. Highly recommended for multilingual documents.
rvsoDefault0Style	If set, no formatting will be saved for text of the 0-th style (TextStyles[437][0]). Browsers will use a default font for it. This option affects the appearance of all styles, because only attributes different from attributes of the 0th style are saved.

Options for SaveHTML [250]

Option	Meaning
rvsoForceNonTextCSS	CSS is used for exporting non-text items. This option allows to retain colors of breaks and layout of tables saved with SaveHTML (without using CSS for text and paragraph styles). Note: SaveHTMLEx always uses CSS in this case.

Options for SaveHTMLEx [251]

Option	Meaning
rvsoNoDefCSSStyle	By default, SaveHTMLEx assigns attributes of TextStyles [437][0] to <body> and <table>, and attributes of all other styles are saved relative to attributes of TextStyles[0]. If you include this option, TextStyles[0] will be treated like any other style. The same for ParaStyles [431][0]. It's not recommended to use this option: it generates much larger CSS and HTML. This option is useful if HTML generated by TRichView is used as a part of larger HTML, and you do not want to change properties of <body> and <table>.
rvsoInlineCSS	If included, CSS is inserted directly in <p>, and tags (instead of generating CSS table at the beginning of HTML). This option generates highly overbloated HTML. Not recommended.

Options for saving only part of document, both for SaveHTML [250], SaveHTMLEx [251]

These options are useful when generating HTML documents what will be inserted in larger HTML document.

Option	Meaning
rvsoFirstOnly	If set, RichView saves only opening part of HTML (opens <html> tag, saves full <head> tag, opens <body> tag)
rvsoMiddleOnly	If set, RichView saves only body of HTML document (content inside <body> tag)
rvsoLastOnly	If set, RichView saves closing part of HTML document (closes <body> and <html> tags)

If none of these three options specified, full HTML document is saved.

You cannot set two or all of these flags at the same time (nothing will be saved in such case).

1.9.36 TRVSearchOptions

Unit RVScroll.

```
type
  TRVSearchOption = (rvsroMatchCase, rvsroDown, rvsroWholeWord, rvsroFromStart,
    rvsroMultiItem, rvsroSmartStart);
  TRVSearchOptions = set of TRVSearchOption;
```

Options for searching in RichView, see TCustomRichView.SearchText[256].

Value	Meaning
rvsroMatchCase	If included, a character case is taken into account when comparing strings.
rvsroDown	If included, the search is performed to the end of the document. If not included, it is performed to the top of the document.
rvsroWholeWord	If included, the searched string matches only whole words.
rvsroFromStart	If included, the search starts from the beginning/end of the document. If not included, it starts from the end of selection or the first/last visible item.
rvsroMultiItem	If included, the search can match substrings of several text items[120]. If not included, the text is searched in each text item separately. For example, if the document contains the text **He*llo***, the substring 'Hello' can be found only if this option is included, because 'He' and 'llo' belong to different items.
rvsroSmartStart	This option is used only when *rvsroFromStart* is not included. If included, and a document fragment is selected, then: ▪ when searching down, the search starts from the second selected character; ▪ when searching up, the search starts from the last but one selected character (selected characters are counted from the top of the document, the selection direction is ignored)

1.9.37 TRVSeqType

Unit RVStyle.

```
type
  TRVSeqType = (rvseqDecimal, rvseqLowerAlpha,
    rvseqUpperAlpha, rvseqLowerRoman, rvseqUpperRoman);
```

Numbering type for sequences[134], endnotes[135], footnotes[136], sidenotes[137].

Type	Meaning
rvseqDecimal	1, 2, 3,...

rvseqLowerAlpha	a, b, c, ..., z, aa, ab, ...
rvseqUpperAlpha	A, B, C, ..., Z, AA, AB, ...
rvseqLowerRoman	i, ii, iii, iv, ...
rvseqUpperRoman	I, II, III, IV, ...

This is a type of the following properties:

- TRVStyle.EndnoteNumbering[422];
- TRVStyle.FootnoteNumbering[422];
- TRVStyle.SidenoteNumbering[422];
- TRVSeqItemInfo.NumberType[625].

1.9.38 TRVStrProperty

Unit RichView.

Identifies a property of TRichView[162] of a string type.

```
type
  TRVStrProperty = (rvspDPTitle, rvspDPAuthor, rvspDPComments);
```

Value	Corresponding property of TRichView
rvspDPTitle	DocParameters[176].Title[314]
rvspDPAuthor	DocParameters.Author[311]
rvspDPComments	DocParameters.Comments[311]

A parameter of this type is used in TRichViewEdit[316].SetStrPropertyEd[370].

See also:

- TRVFloatProperty[678];
- TRVIntProperty[681].

1.9.39 TRVStyleLength

Unit RVStyle;

```
type
  TRVStyleLength = type Integer;
```

This is a type of values measured in TRVStyleUnits[691] (i.e. either twips or screen pixels).

See also:

Units of measurement in TRichView[106]

1.9.40 TRVStyleTemplateId

Unit RVStyle;

```
type
  TRVStyleTemplateId = type Integer;
```

Values of this type identify items in TRVStyleTemplateCollection[460],

This is a type of the following properties:

- Id[495] of TRVStyleTemplate[494] (unique style template identifier);
- ParentId[497] of TRVStyleTemplate[494] (a reference to the parent style template);

- NextId[496] of TRVStyleTemplate[494] (a reference to the style template for following paragraphs);
- StyleTemplateId[465] of text and paragraph attributes (styles), linking them to a style template.
- ParaStyleTemplateId[473] of text attributes (styles), linking them to a style template.

Valid identifiers are positive values. When a value of this type is used as a reference, the value -1 means "no style template". The value 0 is reserved.

1.9.41 TRVStyleTemplateName

Unit RVStyle;

```
type
  TRVStyleTemplateName = type TRVUnicodeString[694];
```

(changed in version 18)

This is a type of the following properties:

- Name of TRVStyleTemplate[494] (unique style template name)

1.9.42 TRVStyleUnits

Unit RVStyle.

```
type
  TRVStyleUnits = (rvstuPixels, rvstuTwips);
```

Value	Meaning
rvstuPixels	Screen pixels (see below)
rvstuTwips	Twips. 1 twip = 1/20 of point = 1/1440 inch

If RichViewPixelsPerInch[703] is positive, it is used for calculation of number of screen pixels in one inch. Otherwise, Screen.PixelsPerInch is used. In the latter case, values in *rvstuPixels* depend on the screen resolution and cannot be used in WYSIWYG editors.

Values measured in TRVUnits have type TRVStyleLength[690].

This is a type of TRVStyle[418].Units[437].

See also:

Units of measurement in TRichView[106]

1.9.43 TRVTableBorderStyle

Unit RVTable;

```
type
  TRVTableBorderStyle =
    (rvtbRaised, rvtbLowered, rvtbColor,
     rvtbRaisedColor, rvtbLoweredColor);
```

Defines style of border in table. This is a type of BorderStyle[570] and CellBorderStyle[572] properties of table.

Value	Meaning
rvtbRaised	Raised 3d border, with default colors.
rvtbLowered	Sunken 3d border, with default colors.
rvtbColor	Flat border of BorderColor[569]/CellBorderColor[571] color.

rvtbRaisedColor	Raised 3d border: ▪ left and top sides: BorderLightColor[569]/CellBorderLightColor[571]; ▪ bottom and right sides: BorderColor[569]/CellBorderColor[571]).
rvtbLoweredColor	Sunken 3d border: ▪ left and top sides: BorderColor[569]/CellBorderColor[571]); ▪ bottom and right sides: BorderLightColor[569]/CellBorderLightColor[571]).

See also

- Colors and layout of tables[148]

1.9.44 TRVTag

Unit RVStyle;

```
type
  TRVTag = type String;
```

This is the type of item tags – additional value associated with any item[118] in TRichView[162] document, and with any table cell[612].

See also:

- Tags[68]

1.9.45 TRVTextDrawStates

Unit RVStyle;

```
type
  TRVTextDrawState = (rvtsSelected, rvtsHover,
    rvtsItemStart, rvtsItemEnd,
    rvtsDrawItemStart, rvtsDrawItemEnd,
    rvtsControlFocused, rvtsSpecialCharacters);
  TRVTextDrawStates = set of TRVTextDrawState;
```

States of text, used for drawing.

See:

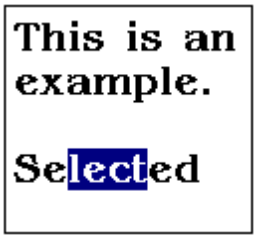
- TRVStyle.OnDrawStyleText[448], OnDrawTextBack[449];
- TFontInfo.Draw[481]

State	Meaning
rvtsSelected	text is selected (for copying to the Clipboard)
rvtsHover	[hyper]text is below a mouse pointer
rvtsItemStart	this call of event draws text item from the beginning
rvtsItemEnd	this call of event finishes drawing of text item
*rvtsDrawItemStart**	this call of event draws part of text item (called "drawing item") from the beginning
*rvtsDrawItemEnd**	this call of event finishes drawing of part of text item.

rvtsControlFocused	TRichView has input focus
rvtsSpecialCharacters	*rvoShowSpecialCharacters* is included in TRichView.Options[182].

* old versions of the components might draw selection as a part of a string belonging to a drawing item; the new version always draws the whole drawing item, setting the proper clipping region before drawing, so these states can be ignored.

For example



(there are two items: 'This is an example.' and 'Selected',

and three "drawing items": 'This is an', 'example.' and 'Selected'

Calls of TRVStyle.OnDrawStyleText[448] will be:

'This is an' with [*rvtsItemStart,rvtsDrawItemStart,rvtsDrawItemEnd*]

'example.' with [*rvtsItemEnd, rvtsDrawItemStart, rvtsDrawItemEnd*]

'Selected' with [*rvtsItemStart, rvtsItemEnd, rvtsDrawItemStart, rvtsDrawItemEnd*]

'Selected' with [*rvtsItemStart, rvtsItemEnd, rvtsDrawItemStart, rvtsDrawItemEnd, rvtsSelected*] (a clipping region is set so that only 'lect' is visible).

(Note: in older versions of the components, the last cal was 'lect' with [*rvtsSelected*]).

1.9.46 TRVUndoType

Unit RVEdit

```
type
TRVUndoType = (
    rvutNone, rvutDelete, rvutInsert, rvutPara, rvutMiscTyping,
    rvutInsertPageBreak, rvutRemovePageBreak,
    rvutTyping, rvutTag, rvutStyleNo,
    rvutAddCheckpoint, rvutRemoveCheckpoint, rvutModifyCheckpoint,
    rvutModifyItem, rvutToHypertext,  rvutFromHypertext,
    rvutTextFlow, rvutList, rvutProperty, rvutLayout,
    rvutBackground, rvutStyleTemplate, rvutSubDoc,
    rvutResize, rvutTableCell,   rvutTableRow,
    rvutTable, rvutTableInsertRows, rvutTableInsertCols,
    rvutTableDeleteRows, rvutTableDeleteCols,
    rvutTableMerge, rvutTableSplit,
    rvutCustom);
```

The values (except for *rvutNone* and *rvutCustom*) identify types of standard undo operations. They are useful if you wish to show (in a menu or a hint) which operation is about to be undone.

There are no hard-coded names for these operations, because they may be application specific. For example, *rvutTag* (changing item's *tag*[68]) can have special meaning for an application (editing a hyperlink or another item property).

The demo **Demos\Delphi\Editors\Editor 1** has suggested names of actions in **RVUndoStr.pas**.

There are two special undo types:

Type	Meaning
rvutNone	No undo/redo available (disable an undo or a redo menu item or button)
rvutCustom	The undo action has a special name (see BeginUndoCustomGroup[336]).

1.9.47 TRVUnicodeString

Unit RVTypes.

Declaration for Delphi 2009 or newer:

```
type
  TRVUnicodeString = type UnicodeString;
```

Declaration for older versions of Delphi:

```
type
  TRVUnicodeString = type WideString;
```

Strings of this type contain Unicode text (UTF-16).

See also

- TRVAnsiString[668]
- TRVRawByteString[685]

1.9.48 TRVUnits

Unit RVStyle.

Units of measurement.

```
type
  TRVUnits = (rvuInches, rvuCentimeters, rvuMillimeters, rvuPicas, rvuPixels, rvuPoints);
```

Value	Meaning
rvuInches	Inches. 1 inch = 2.54 cm.
rvuCentimeters	Centimeters.
rvuMillimeters	Millimeters.
rvuPicas	Picas (used in typography). 1 pica = 1/6 inch = 12 points.
rvuPixels	Screen pixels (see below).
rvuPoints	Points (used in typography). 1 point = 1/72 inch.

If RichViewPixelsPerInch[703] is positive, it is used for calculation of number of screen pixels in one inch. Otherwise, Screen.PixelsPerInch is used. In the latter case, values in *rvuPixels* depend on the screen resolution and cannot be used in WYSIWYG editors.

This is a type of the following properties:

- TRichView[162].DocParameters[176].Units[314].

- TRVPrint[513].Units[521]

Values measured in TRVUnits have the type TRVLength[682].

See also:

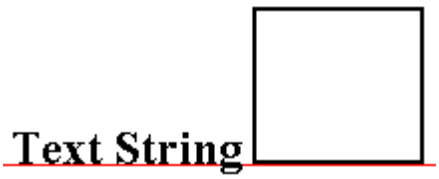
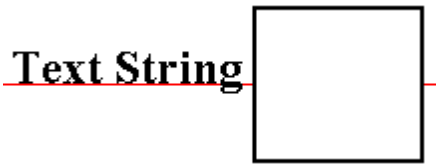
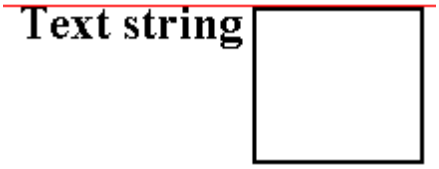
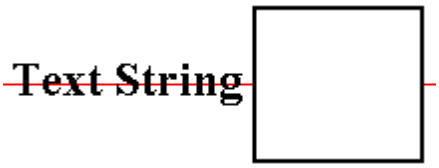
Units of measurement in TRichView[106]

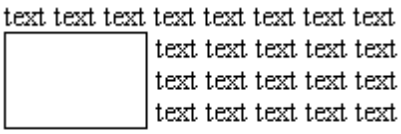
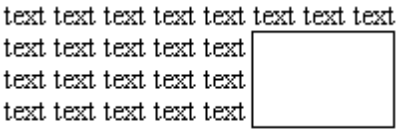
1.9.49 TRVVAlign

Unit RVStyle;

```
type
  TRVVAlign = (
    rvvaBaseline,  // bottom of picture -> baseline
    rvvaMiddle,    // center of picture -> baseline
    rvvaAbsTop,    // top of picture    -> top of line
    rvvaAbsBottom, // bottom of picture -> bottom of line
    rvvaAbsMiddle, // center of picture -> center of line
    rvvaLeft,      // align to left side
    rvvaRight      // align to right side
    );
```

Vertical align of pictures and controls in RichView

Align	Scheme
rvvaBaseline	
rvvaMiddle	
rvvaAbsTop	
rvvaAbsBottom	
rvvaAbsMiddle	

rvvaLeft	
rvvaRight	

Note: non-standard values of HTML align attributes (abstop, absbottom, absmiddle) are used for export of rvvaAbs*** values. Microsoft Internet Explorer and Mozilla understand them, but not Opera.

This property can be applied to:

- pictures 122 and *hot-pictures* 125
- *bullets* 128 and *hotspots* 129
- controls 126
- label items 133
- numbered sequences 134

See also methods of TCustomRichView 162:

- GetItemVAlign 230,
- SetItemVAlign 268.

See also properties of TCustomRichView 162:

- ClearLeft 171,
- ClearRight 172.

See also methods of TCustomRichViewEdit 316:

- GetCurrentItemVAlign 348,
- SetItemVAlignEd 386,
- SetCurrentItemVAlign 386,
- ClearTextFlow 339.

See also properties of TRVStyle 418:

- FloatingLineColor 423

1.10 Global Constants and Variables

Global Variables and Typed Constants

Editing, Caret

- RichViewEditCaretWidthMode, RichViewEditCaretWidth 699 define the caret width;
- RichViewEditCaretHeightExtra 699: Integer, allows making the caret shorter or longer;
- RichViewEditMinCaretHeight, RichViewEditMaxCaretHeight 699: Integer – minimum/maximum possible caret height;
- RichViewEditCustomCaretSize 699: Boolean allows changing caret size in an event.
- RichViewEditCaretPosition 699: TRVHorizontalCaretPosition defines how the caret is aligned horizontally relative to the insertion point.

Editing, Other

- RichViewEditDefaultProportionalResize 702: Boolean, specifies whether images are resized proportionally with the mouse;

- RichViewEditEnterAllowsEmptyMarkeredLines[702]: Boolean, enables or disables autodeletion of list marker from new empty lines;
- RichViewUnicodeInput[705]: TRichViewUnicodeInput, specifies how types Unicode characters are processed (in non-Unicode versions of Delphi);
- RichViewTableAutoAddRow[705]: Boolean, allows or disallows adding new table rows using Tab key.

Saving and Loading

- RichViewSavePInHTML[698]: Boolean, instructs always using <p> for saving in HTML;
- RichViewSaveDivInHTMLEx[698]: Boolean, instructs always using for saving in HTML;
- RichViewSavePageBreaksInText[698]: Boolean, allows or disallows saving page breaking characters in plain text;
- RichViewDoNotCheckRVFStyleRefs[701]: Boolean, allows turning off checking for invalid style indices when loading RVF;
- RichViewResetStandardFlag[703]: Boolean, specifies whether to reset Standard[464] property to False when inserting RVF;
- RichViewPixelsPerInch[703]: Integer, allows overriding the screen resolution;
- RichViewAutoAssignNormalStyleTemplate[700]: Boolean, specifies whether to assign "Normal" style template when loading unstyled paragraphs from RVF and RTF.
- RichViewTableDefaultRTFAutofit[705]: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.
- RichViewSaveHyperlinksInDocXAsFields[704] defines how hyperlinks are saved in DocX

Tables

- RichViewTableAutoAddRow[705]: Boolean, allows or disallows adding new table rows using Tab key;
- RichViewAllowCopyTableCells[700]: Boolean allows preventing copying multicell selection in a table to the Clipboard.
- RichViewTableDefaultRTFAutofit[705]: Boolean, allows saving tables as fit-by-content tables in RTF and DocX.

Formatting and Displaying

- RichViewJustifyBeforeLineBreak[702]:Boolean, specifies whether lines before line breaks are justified (expanded);
- RichViewStringFormatMethod[704]:Boolean, allows activating an alternative text measuring method;
- RichViewShowGhostSpaces[704]: Boolean, allows showing space characters "hidden" between lines.
- RVVisibleSpecialCharacters[714]: TRVSpecialCharacters specifies which special characters can be shown.
- RVThumbnailMaker[713]: TRVThumbnailMaker[659], provides access to an object used to create scaled images;
- RichViewMaxPictureCount[702]: Integer – maximal count of "active" images in TRichView control.
- RichViewMaxThumbnailCount[703]: Integer – maximal count of scaled images stored in TRichView control.

Printing

- RichViewPixelsPerInch[703]: Integer, allows overriding the screen resolution;
- RichViewAlternativePicPrint[700]: Boolean, allows using an alternative picture printing method.

Live spelling check

- RichViewApostropheInWord[700]: Boolean, specifies how apostrophe characters are processed when finding word boundaries.

Classes

- RVThumbnailMaker[713]: TRVThumbnailMaker[659], provides access to an object used to create scaled images;
- RVGraphicHandler[710]: TRVGraphicHandler[654], provides access to an object used to manage graphic classes.

Other

- RichViewCompareStyleNames[701]: Boolean, specifies whether StyleName[465] property is used when comparing (and searching for) text, paragraph or list styles.

Global Functional Variables

- RVIsCustomURL[710]: TCustomRVIsURLFunction, allows defining your own URL prefixes.

Constants

- **rvs*****[711] constants[711]: item types;
- **rveipc*****[706] constants[706]: integer properties of items;
- **rvespc*****[709] constants[706]: string properties of items;
- Constants and variables related to Clipboard[698].

1.10.1 Clipboard Related Constants and Variables

Unit RVStr.

Clipboard Format Names

Names of the Clipboard formats for copying RVF[93], RTF, URL, HTML.

Syntax

```
const
  RVFormatName = 'RichView Format';
  RTFormatName = 'Rich Text Format';
  URLFormatName = 'UniformResourceLocator';
  HTMLFormatName = 'HTML Format';
```

Clipboard Format Identifiers

```
var CFRV_RVF, CFRV_RTF, CFRV_HTML, CFRV_URL: Word;
  ...
  CFRV_RVF := RegisterClipboardFormat(RVFormatName);
  CFRV_RTF := RegisterClipboardFormat(RTFormatName);
  CFRV_URL := RegisterClipboardFormat(URLFormatName);
  CFRV_HTML := RegisterClipboardFormat(HTMLFormatName);
```

1.10.2 Constants Related to HTML and Text

Unit CRVData.

Syntax

```
var
  RichViewSavePInHTML: Boolean = False;
  RichViewSaveDivInHTMLEx: Boolean = False;
  RichViewSavePageBreaksInText: Boolean = False;
```

Set **RichViewSavePInHTML** to *True* is you want SaveHTML[250] to write <p> instead of in HTML. This variable does not affect SaveHTMLEx[251]. This setting changes appearance of resulting HTML documents (usually browsers add additional spacing between <p>).

Set **RichViewSaveDivInHTMLEx** to *True* is you want SaveHTMLEx[251] to write instead of <p> in HTML. This variable does not affect SaveHTML[250]. This setting does not change appearance of resulting HTML documents (because all <p> or visual attributes are defined in CSS).

Set **RichViewSavePageBreaksInText** to *True* if you want to save explicit page breaks as #$0C characters. This typed constant affects all plain text export methods.

1.10.3 Constants Related to the Caret

These typed constants allow to control the caret appearance in RichViewEdit[316].

Unit RVERVData.

Syntax

```
type
  TRVHorizontalCaretPosition =
    (rvhcpDefault, rvhcpCenter, rvhcpRight);
  TRVCaretWidthMode =
    (rvcwmDefault, rvcwmCharacter);
const
  RichViewEditCaretWidth: Integer                    = 0;
  RichViewEditCaretWidthMode: TRVCaretWidthMode      = rvcwmDefault;
  RichViewEditCaretHeightExtra: Integer              = 0;
  RichViewEditMaxCaretHeight: Integer                = 1000;
  RichViewEditMinCaretHeight: Integer                = 0;
  RichViewEditCaretPosition: TRVHorizontalCaretPosition = rvhcpDefault;
  RichViewEditCustomCaretSize: Boolean               = False;
```

Caret width

RichViewEditCaretWidthMode defines how the caret width (thickness) is calculated. The meaning of possible values are explained in the table below.

Value	Meaning
rvcwmDefault	The caret width is calculated according to **RichViewEditCaretWidth**, see below.
rvcwmCharacter	The caret has width of the next character/item. When the caret is at the end of line, its width is calculated like in *rvcwmDefault* mode.

RichViewEditCaretWidth defines the width of the caret. The value 0 (default) is special, it means using the system caret thickness (defined in the "Ease of Access" of the Settings app (or the Control Panel, depending in the Windows version)).

ScaleRichView note: TSRichViewEdit uses PageProperty.CaretPen.Width like **RichViewEditCaretWidth**.

Caret height

Value of **RichViewEditCaretHeightExtra** is added to the top and the bottom of the caret (negative values make the caret shorter, positive values make it longer).

RichViewEditMaxCaretHeight – maximum possible caret height.

RichViewEditMinCaretHeight – minimum possible caret height.

Custom caret size

If **RichViewEditCustomCaretSize** = *True*, OnMeasureCustomCaret[394] event is called not only for a custom caret, but for the standard caret as well.

The event is called after the standard caret size is calculated, but before the caret is repositioned (initially, it is positioned to the right side of the insertion position.

Caret position

RichViewEditCaretPosition specifies how a wide caret is positioned relative to the insertion position.

Value	Caret position
rvhcpDefault	To the left side of the insertion position for items having BiDiMode = rvbdRightToLeft[668], to the right side otherwise.
rvhcpCenter	Centered at the insertion position
rvhcpRight	To the right side of the insertion position

ScaleRichView note: RichViewEditCaretPosition affects ScaleRichView as well.

1.10.4 RichViewAllowCopyTableCells

Unit CRVData.

Syntax

```
var
  RichViewAllowCopyTableCells: Boolean = True;
```

Set to *False* to disallow copying multicell selection (it will be copied as empty RVF[93]).

1.10.5 RichViewAlternativePicPrint

Unit RVFuncs.

Syntax

```
const
  RichViewAlternativePicPrint: Boolean = False;
```

If set to *True*, alternative picture printing procedure is used. The alternative procedure is less reliable, but does not depend on the screen color resolution. It may be useful if your application work on server in 16 or 256 color mode.

1.10.6 RichViewApostropheInWord

Unit RVThread;

Syntax

```
var
  RichViewApostropheInWord: Boolean = True;
```

Specifies how apostrophe characters ("" and #146) are processed in live spelling.

If *True*: apostrophes are allowed in middle of words, even if they are included in Delimiters[174] property. For example, "doesn't" is processed as one word instead of "doesn" and "t".

If *False*: apostrophes are processed like all other delimiters. For example, when checking French text with Addict3, it is recommended to include "" in Delimiters[174] and assign RichViewApostropheInWord=*False*.

Note: apostrophe is not included in Delimiters[174] by default.

1.10.7 RichViewAutoAssignNormalStyleTemplate

Unit RVStyle.

Syntax

```
const
  RichViewAutoAssignNormalStyleTemplate: Boolean = True;
```

(introduced in version 14)

This constant controls how unstyled paragraphs (i.e. paragraphs not linked[465] to any style template[494]) are loaded from RTF and RVF files/streams, if style templates are used[195].

If *True*, "Normal" style template[461] (if it exists) is assigned to paragraph styles[490] of the loaded document.

If *False*, -1 is assigned to paragraph styles of the loaded document.

This constant affects the both loading and insertion. See also TRichView.StyleTemplateInsertMode[193] property.

See also:

- Styles and style templates[74].

1.10.8 RichViewCompareStyleNames

Unit RVStyle.

Syntax

```
const
  RichViewCompareStyleNames: Boolean = False;
```

If set to *True*, methods for searching and comparing styles take StyleName[465] property into account.

1.10.9 RichViewDoNotCheckRVFStyleRefs

Unit CRVData.

Syntax

```
const
  RichViewDoNotCheckRVFStyleRefs: Boolean = False;
```

(introduced in version 10)

If set to *True*, TRichView ignore invalid style references when loading RVF documents[93].

It is useful only if you do not store collections of styles in documents and want to delete unused styles in multiple documents, see TRVDeleteUnusedStylesData[653].

1.10.10 RichViewDoNotMergeNumbering

Unit RVStyle.

Syntax

```
const
  RichViewDoNotMergeNumbering: Boolean = False;
```

RichViewDoNotMergeNumbering affects inserting RVF[93]. By default, when inserting RVF files containing numbered lists, TRichView tries to map the inserted list styles to the existing list styles: if the inserted list style A has the same properties (including a *list identifier*) as the existing style B, then A is mapped to B; so inserted numbered lists of the style A continue existing numbered lists of the style B. *List identifier* is a random number assigned to any list style on creation. Because of list identifiers, numbered lists that are copy-pasted from the same document continue the existing numbering (because they have the same identifiers), while numbered lists from another document are inserted as new lists.

Assign *True* to **RichViewDoNotMergeNumbering** to modify the standard behavior. In this mode, all numbered lists from RVF are inserted as new lists, even if the document has list styles with the same properties and identifiers.

This property is useful when generating a big document from several small documents.

1.10.11 RichViewEditDefaultProportionalResize

Unit RVERVData.

Syntax

```
const
  RichViewEditDefaultProportionalResize: Boolean = True;
```

By default, dragging corner handles resizes picture proportionally (like in MS Word).

Set to *False* to enable free resizing using the corner handles (use Shift key to resize proportionally in this mode)

1.10.12 RichViewEditEnterAllowsEmptyMarkeredLines

Unit RVERVData.

Syntax

```
const
  RichViewEditEnterAllowsEmptyMarkeredLines: Boolean = False;
```

Set to *True* if you do not want to autoremove bullets & numbering from blank lines when the user presses Enter .

1.10.13 RichViewJustifyBeforeLineBreak

Unit CRVFData.

Syntax

```
const
  RichViewJustifyBeforeLineBreak: Boolean = False;
```

(introduced in version 15)

If *False*, the last line before a line break (added with Shift+Enter) is not justified by expanding space characters.

Set to *True* to justify such lines.

1.10.14 RichViewMaxPictureCount

Unit RvGrCache.

Syntax

```
var
  RichViewMaxPictureCount: Integer  = 100;
```

(introduced in version 16)

Specifies the maximal count of images stored in TCustomRichView[162] in an "active" form, i.e. as TGraphic objects. Other pictures are "deactivated", i.e. stored in a memory stream until they are requested. This activation/deactivation allows saving GDI resources and memory.

Changes of value of this property do not affect TRichView components created before the change.

See also:

- Smooth image scaling[91]
- RichViewMaxThumbnailCount[703]

1.10.15 RichViewMaxThumbnailCount

Unit CRVData.

Syntax

```
var
  RichViewMaxThumbnailCount = 50;
```

(introduced in version 15)

Specifies the maximal count of scaled images stored in TCustomRichView[162].

If RichViewMaxThumbnailCount=-1, all scaled images are stored.

Scaled images are stored for efficiency (to prevent recreating thumbnails on each redrawing).

Changes of value of this property do not affect TRichView components created before the change.

See also:

- Smooth image scaling[91]
- RichViewMaxPictureCount[702]

1.10.16 RichViewPixelsPerInch

Number of pixels in one screen inch.

Unit RVFuncs.

Syntax

```
const
  RichViewPixelsPerInch : Integer = 0;
```

If 0 (default), RichView uses the screen resolution (Screen.PixelsPerInch).

If non zero, this value is used.

This value is used when converting pixels on the screen to mm, inches, twips, pixels on the printer, and vice versa.

The most important applications are:

- printing;
- RTF export&import;
- calculating sizes of objects when they are measured in twips[106].

This constant does not affect text (TRVStyle[418].TextStyles[437].PixelsPerInch[452] property is used for text, it has the same meaning). This constant affects pictures and tables.

The main possible values: 96 (small font mode), 120 (large font mode).

Assigning positive value this constant makes printing and RTF export independent of the screen resolution (small font/large fonts mode).

If you use TRVRuler component (from RichViewActions) and if you assign a positive value to RichViewPixelsPerInch, assign the same value to RVRuler1.ScreenRes property,

1.10.17 RichViewResetStandardFlag

Unit RVStyle.

Syntax

```
const
```

```
  RichViewResetStandardFlag: Boolean = True;
```

If *True*, Standard[464] properties of styles which were added from inserted RVF[93] will be reset to *False* (it only affects RVF insertion, not loading).

If this flag is *True*, new styles having Standard[464]=*True* may appear only after calling LoadRVF[245] or LoadRVFFromStream[245].

1.10.18 RichViewSaveHyperlinksInDocXAsFields

Unit CRVData.

Syntax

```
const
    RichViewSaveHyperlinksInDocXAsFields = False;
```

This constant switches between two methods of hyperlinks saving in DocX files.

If *False*:

SaveDocX[249] saves hyperlinks as <w:hyperlink>. This way of saving is more compatible, it must be understood by all DocX readers.

Minus: MS Word checks hyperlink targets more strictly, if it contains an invalid URL, it may treat the whole file as erroneous.

If *True*:

SaveDocX saves hyperlinks as HYPERLINK field, similar to RTF.

1.10.19 RichViewShowGhostSpaces

Unit CRVFData.

Syntax

```
const
  RichViewShowGhostSpaces: Boolean = False;
```

(introduced in version 10)

Set to *True* if you want to show space characters at the end of lines when *rvoShowSpecialCharacters* is included in TCustomRichView.Options[182]. Caret cannot be placed after these spaces.

1.10.20 RichViewStringFormatMethod

Unit CRVFData.

Syntax

```
type
  TRVStringFormatMethod =
    (rvsfmAuto, rvsfmEngine, rvsfmTRichView);
const
  RichViewStringFormatMethod: TRVStringFormatMethod = rvsfmAuto;
```

This variable specifies formatting methods used in all TRichView and ScaleRichView components. The main question is the method of calculation of the count of characters in the string that fit to the specified width.

Value	Meaning

rvsfmAuto	The components tries to choose the fastest method
rvsfmEngine	The component uses the function provided by API
rvsfmTRichView	The component uses its own method (a binary search on string widths).

About the "auto" method

If TextEngine[436]=*rvteUniscribe*, or in ScaleRichView, the components choose the TRichView method only if the string length is much greater than an estimated count of character fitting to the specified width.

Otherwise, the components always choose the TRichView method.

1.10.21 RichViewTableAutoAddRow

Unit RVTable.

Syntax

```
const
  RichViewTableAutoAddRow: Boolean = True;
```

(introduced in version 10)

Set to *False* to disable auto insertion of new table row when user presses **Tab** key in the last table[130] cell.

The row auto insertion works only in TRichViewEdit[316] and TDBRichViewEdit[410].

1.10.22 RichViewTableDefaultRTFAutofit

Unit RVTable.

Syntax

```
const
  RichViewTableDefaultRTFAutofit: Boolean = False;
```

Assigning *True* to this variable works like inclusion of *rvtoRTFAllowAutofit* in Options[577] properties of all tables[130].

1.10.23 RichViewUnicodeInput

Unit RVEdit.

Syntax

```
type
  TRichViewUnicodeInput = (rvuiStandard, rvuiAlternative);
const RichViewUnicodeInput: TRichViewUnicodeInput = rvuiStandard;
```

This constant is ignored for Delphi/C++Builder 2009 or newer (they provide true Unicode text input).

RichViewEdit uses two methods to process typing of unicode[99] text:

- "Standard" method processes WM_CHAR and converts typed characters to Unicode.
- "Alternative" method uses ToUnicode procedure. It cannot process "dead keys" correctly, so it should not be used for keyboard layouts having "dead keys".

If RichViewUnicodeInput=*rvuiStandard*, the "alternative" method is used only for the following languages:

- languages without their own code page (so it's impossible to use WM_CHAR);
- Kazakh, Tatar, Mongolian, Azerbaijani (cyrillic&latin), Uzbek, Kyrgyz, Serbian (these languages include some characters that do not belong to code page).

Generally, it works very well. But for some "exotic" keyboard layouts the "standard" method does not provide completely correct input. Set this variable to *rvuiAlternative* to always use the "alternative" method. Do it only as an option, if you have troubles with the specific keyboard layout.

1.10.24 rveipcXXX constants

These constants identify integer properties of TRichView items[118].

They are used in the following methods:

- TRichView.GetExtraItemIntPropertyEx[227]
- TRichView.SetExtraItemIntPropertyEx[266]
- TRichViewEdit.SetItemExtraIntPropertyExEd[383]
- TRichViewEdit.GetCurrentItemExtraIntPropertyEx[346]
- TRichViewEdit.SetCurrentItemExtraIntPropertyEx[378]

Properties of Bullets[128] and Hotspots[129]

These properties are defined in RVItem unit.

These properties provides an alternative to the following methods:

- TRichView: SetBulletInfo[261], GetBulletInfo[219], GetHotspotInfo[224], SetHotspotInfo[265];
- TRichViewEdit: SetBulletInfoEd[372], SetCurrentBulletInfo[375], GetCurrentBulletInfo[342], SetHotspotInfoEd[382], SetCurrentHotspotInfo[377], GetCurrentHotspotInfo[344].

Constant	Value	Property
rveipcImageIndex	200	Image index (for *bullets* and *hotspots*)
rveipcHotImageIndex	201	Image index below the mouse pointer (for *hotspots*)

Properties of Controls[126]

These properties are defined in RVItem unit.

Constant	Value	Property
rveipcPercentWidth	200	If the value is in the range 1..100, it defines the width of the control in % of document width.

Properties of Breaks[125]

These properties are defined in RVItem unit.

These properties provide an alternative to the following methods:

- TRichView: SetBreakInfo[260], GetBreakInfo[218];
- TRichViewEdit: SetBreakInfoEd[371], SetCurrentBreakInfo[375], GetCurrentBreakInfo[341].

Constant	Value	Property
rveipcBreakWidth	200	Line width
rveipcBreakStyle	201	Line style*

rveipcBreakColor	202	Line color

* the property is type-casted to Integer

Properties of Label items [133]

These properties are defined in RVLabelItem unit.

They identify properties of label items [133] and items of all types inherited from them (numbered sequences [134], footnotes [136], endnotes [135], references to notes [140]).

Constant	Value	Property
rveipcProtectTextStyleNo	200	ProtectTextStyleNo [619]*
rveipcRemoveInternalLeading	201	RemoveInternalLeading [619]*
rveipcCursor	202	Cursor [618]
rveipcMinWidth	203	MinWidth [618]
rveipcAlignment	204	Alignment [618]*
rveipcRemoveSpaceBelow	205	RemoveSpaceBelow [619]*

* the property is type-casted to Integer

Properties of Numbered sequences [134]

These properties are defined in RVSeqItem unit.

They identify properties of numbered sequences [134] and items of all types inherited from them (footnotes [136], endnotes [135]).

Constant	Value	Property
rveipcSeqStartFrom	300	StartFrom [626]
rveipcSeqReset	301	Reset [626]*
rveipcSeqNumberType	302	NumberType [625]*,**

* the property is type-casted to Integer

** read-only for footnotes [136], endnotes [135], sidenotes [137] and text box items [139].

Properties of Sidenotes [137]

Position properties

These properties are defined in RVFloatingPos unit.

They identify sub-properties of BoxPosition [634] property of sidenotes [137] and text box items [139].

Constant	Value	Property
rveipcFloatHorizontalAnchor	500	HorizontalAnchor [638]*
rveipcFloatHorizontalPositionKind	501	HorizontalPositionKind [639]*
rveipcFloatHorizontalOffset	502	HorizontalOffset [639]

rveipcFloatHorizontalAlign	503	HorizontalAlignment 639*
rveipcFloatVerticalAnchor	504	VerticalAnchor 641*
rveipcFloatVerticalPositionKind	505	VerticalPositionKind 642*
rveipcFloatVerticalOffset	506	VerticalOffset 642
rveipcFloatVerticalAlign	507	VerticalAlignment 642*
rveipcFloatRelativeToCell	508	RelativeToCell 640*
rveipcFloatPositionInText	509	PositionInText 640*

* the property is type-casted to Integer

Other properties

These properties are defined in RVFloatingBox unit.

They identify sub-properties of BoxProperties 634 property of sidenotes 137 and text box items 139.

Constant	Value	Property
rveipcBoxWidth	600	Width 636
rveipcBoxHeight	601	Height 636
rveipcBoxWidthType	602	WidthType 636*
rveipcBoxHeightType	603	HeightType 636*
rveipcBoxBorderColor	604	Border 635.Color 502
rveipcBoxBorderWidth	605	Border 635.Width 503
rveipcBoxBorderInternalWidth	606	Border 635.InternalWidth 502
rveipcBoxBorderStyle	607	Border 635.Style 502*
rveipcBoxBorderVisibleBorders_Left	608	Border 635.VisibleBorders 502.Left*
rveipcBoxBorderVisibleBorders_Top	609	Border 635.VisibleBorders 502.Top*
rveipcBoxBorderVisibleBorders_Right	610	Border 635.VisibleBorders 502 .Right*
rveipcBoxBorderVisibleBorders_Bottom	611	Border 635.VisibleBorders 502 .Bottom*
rveipcBoxBorderBorderOffsets_Left	612	Border 635.BorderOffsets 501.Left
rveipcBoxBorderBorderOffsets_Top	613	Border 635.BorderOffsets 501.Top
rveipcBoxBorderBorderOffsets_Right	614	Border 635.BorderOffsets 501.Right
rveipcBoxBorderBorderOffsets_Bottom	615	Border 635.BorderOffsets 501 .Bottom
rveipcBoxColor	616	Background 635.Color 501

rveipcBoxPadding_Left	617	Background[635].BorderOffsets[501].Left
rveipcBoxPadding_Top	618	Background[635].BorderOffsets[501].Top
rveipcBoxPadding_Right	619	Background[635].BorderOffsets[501].Right
rveipcBoxPadding_Bottom	620	Background[635].BorderOffsets[501].Bottom
rveipcBoxVAlign	621	VAlign[636]*
rveipcBoxOpacity	622	Background[635].Opacity[501]

* the property is type-casted to Integer

Properties of "Page number"[141] and "Page count"[142] fields

These properties are defined in RVFieldItems unit.

Constant	Value	Property
rveipcPageNumberType	600	NumberType[646]*

* the property is type-casted to Integer

Properties of mathematical expressions[143]

These properties are defined in RVMathItem unit.

Constant	Value	Property
rveipcFontSizeDouble	200	FontSizeDouble[622]
rveipcTextColor	201	TextColor[623]
rveipcDisplayInline	202	DisplayInline[621]*

* the property is type-casted to Integer

See also

- rvespc*** constants[709]

1.10.25 rvespcXXX constants

These constants identify string properties of TRichView items[118].

They are used in the following methods:

- TRichView.GetExtraItemStrPropertyEx[228]
- TRichView.SetExtraItemStrPropertyEx[266]
- TRichViewEdit.SetItemExtraStrPropertyExEd[384]
- TRichViewEdit.GetCurrentItemExtraStrPropertyEx[346]
- TRichViewEdit.SetCurrentItemExtraStrPropertyEx[379]

Properties of Label items [133]

These properties are defined in RVLabelItem unit.

They identify properties of label items [133] and items of all types inherited from them (numbered sequences [134], footnotes [136], endnotes [135], sidenotes [137], references to notes [140], text boxes [139], page numbers [141], page counts [142]).

Constant	Value	Property
rvespcText	200	Text [619] *

* read-only for numbered sequences [134], footnotes [136], endnotes [135], sidenotes [137], references to notes [140]; ignored for text boxes [139]

Properties of Numbered sequences [134]

These properties are defined in RVSeqItem unit.

Constant	Value	Property
rvespcSeqName	300	SeqName [626]
rvespcSeqFormatString	301	FormatString [625]

Properties of mathematical expressions [143]

These properties are defined in RVMathItem unit.

Constant	Value	Property
rvespcText	200	Text [622]
rvespcFontName	201	FontName [622]

See also

- rveipc*** constants [706]

1.10.26 RVGraphicHandler

Unit RVGrHandler.

Syntax

```
var RVGraphicHandler: TRVGraphicHandler [654];
```

(introduced in version 15)

This variable returns an object used by TRichView components to manage graphic classes.

1.10.27 RVIsCustomURL

Unit RVFuncs.

Syntax

```
type
  TCustomRVIsURLFunction = function (const Word: TRVUnicodeString [694]): Boolean;
const
  RVIsCustomURL: TCustomRVIsURLFunction = nil;
```

(changed in version 18)

You can assign your function to this variable. This function must return *True* if **Word** parameter contains URL. For example, you can create function returning *True* if **Word** is started from 'icq:'. It's not necessary to detect standard URL types here, detect only your own types of URLs.

This procedure is called by RVIsURL 663 function.

See demos:

- Demos\Delphi\Assorted\Hypertext\URLs\
- Demos\CBuilder\Assorted\Hypertext\URLs\

1.10.28 RVParagraphMarks

Unit RVStyle.

Syntax

```
type
  TRVParagraphMarks = (rvpmStandard, rvpmArrows);
const
  RVParagraphMarks: TRVParagraphMarks = rvpmStandard;
```

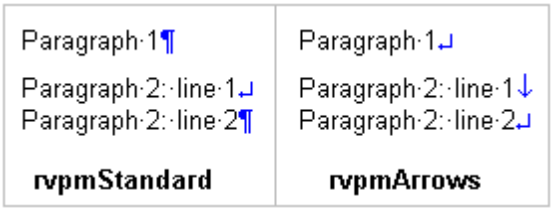
(introduced in version 13)

Defines a set of symbols used to display ends of paragraphs and lines.

These symbols are displayed if *rvoShowSpecialCharacters* is included in TCustomRichView.Options 182, and *rvscParagraph* is included in RVVisibleSpecialCharacters 714 variable.

Value	Meaning
rvpmStandard	Pilcrow for paragraph breaks, bended arrow for line breaks
rvpmArrows	Bended arrow for paragraph breaks, arrow down for line breaks

Example:



See also:

- TRVStyle.SpecialCharactersColor 435

1.10.29 rvsXXX constants

There are three groups of **rvs***** constants:

- identifiers of types of RichView 162 items 64;
- indexes in the collection of text styles (RVStyle.TextStyles 437) corresponding to the initial entries;
- values used by TRichView internally when saving and loading RVF 93 files (see RVF specification 109).

Identifiers of Item Types

Use the first group of constants to compare with values returned by RichView.GetItemStyle 229 or TRichViewEdit.CurItemStyle 323.

Constant	Value	Item Type	Unit
rvsBreak	-1	break 125 (horizontal line)	RVStyle
rvsPicture	-3	picture 122	RVStyle
rvsHotspot	-4	hotspot 129 (picture from ImageList - hyperlink)	RVStyle
rvsComponent	-5	inserted Delphi control 126	RVStyle
rvsBullet	-6	bullet 128 (picture from ImageList)	RVStyle
rvsHotPicture	-10	hot picture 125 (picture - hyperlink)	RVStyle
rvsListMarker	-11	list marker 131 (bullet or numbering of paragraphs)	RVStyle
rvsTab	-12	tabulator 121	RVStyle
rvsTable	-60	table 145	RVTable
rvsLabel	-200	label 133 (non-text item looking like text)	RVLabelItem
rvsSequence	-202	numbered sequence 134	RVSeqItem
rvsFootnote	-203	footnote 136	RVNote
rvsEndnote	-204	endnote 135	RVNote
rvsSidenote	-206	sidenotes 137	RVSidenote
rvsTextBox	-207	text boxes 139	RVSidenote
rvsNoteReference	-205	reference to the parent note 140	RVNote
rvsPageNumber	-208	"page number" field 141	RVFieldItems
rvsPageCount	-209	"page count" field 142	RVFieldItems
rvsMathEquation	-210	mathematical expression 143	RVMathItem

Item types included in ***Report Workshop***:

Constant	Value	Item Type	Unit
rvsReportTable	-61	report table	RVReportTable
rvsShape	-211	geometric shape	RVReportShapeItem

Types for demo items (in TRichView\Demos\Addins\)

Constant	Value	Item Type	Unit
rvsBlendBitmap	-50	semitransparent bitmap image	RVBlendBitmap
rvsHotBlendBitmap	-51	hypertext semitransparent bitmap image	RVBlendBitmap
rvsChart	-1001	TChart item	ChartItem

rvsCombo	-201	label item showing combo box	ComboItem

All these constants are negative so you can distinguish them from indexes in the collection of text styles.

See also: TRichView item types 64.

Indexes in The Collection of Text Styles

The second group of constants is almost obsolete, because the status of these text styles is changed from "standard" (in older versions) to "initial". You can completely customize collections of text styles by removing initial items and adding new items.

All these constants are defined in RVStyle unit.

Constant	Value	Suggested Using
rvsNormal	0	Normal text
rvsHeading	1	Heading 1
rvsSubheading	2	Heading 2
rvsKeyword	3	Keywords
rvsJump1	4	Hypertext 1
rvsJump2	5	Hypertext 2

It's still highly recommended to use the 0-th style as the default (normal) text. For example, this style is used as a default style in exported HTML files.

There is a special constant *rvsDefStyle*.

Constant	Value	Meaning
rvsDefStyle	MaxInt	"Use default text style" **(deprecated!)**

This constant can be used as a StyleNo parameter for methods that add text in TRichView.

When this value is specified instead of the index in the collection of text styles, the paragraph's default text style 491 will be used.

1.10.30 RVThumbnailMaker

Unit RVThumbMaker.

Syntax

```
var RVThumbnailMaker: TRVThumbnailMaker 659;
```

(introduced in version 15)

This variable returns an object used by TRichView components to create smooth scaled images. This object has methods and properties allowing to restrict size of scaled images, specify classes of scaled images, etc.

See also:

- smooth image scaling 91

1.10.31 RVVisibleSpecialCharacters

Unit RVStyle.

Syntax

```
type
  TRVSpecialCharacter = (rvscSpace, rvscNBSP, rvscTab, rvscParagraph,
    rvscSoftHyphen, rvscPlaceholders, rvcFloatingLines);
  TRVSpecialCharacters = set of TRVSpecialCharacter;
const
  RVVisibleSpecialCharacters: TRVSpecialCharacters = [rvscSpace..rvcFloatingLines];
```

(introduced in version 10)

Lists characters which will be shown when *rvoShowSpecialCharacters* is included in TCustomRichView.Options[182].

Value	Meaning
rvscSpace	If included, space characters are shown (as middle dots). Hidden spaces at the ends of lines are not shown, see RichViewShowGhostSpaces[704]
rvscNBSP	If included, non breaking space characters are shown (as circles)
rvscTab	If included, tab characters are shown (as arrows)
rvscParagraph	If included, end-of-paragraph and end-of-line marks are shown (as pilcrows and arrows, respectively)
rvscSoftHyphen	If included, soft hyphen characters are shown in Unicode text.
rvscParagraphAttrs	If included together with *rvscParagraph*, additional icons are drawn next to end-of-paragraph marks: ▪ triangle-arrow showing paragraph direction[484] (if defined); ▪ rectangle if any non-default printing option[487] is defined (never displayed for headings 1...9) ▪ heading level (1...9) ▪ text flow clearing[171] (for the next paragraph or this line break)
rvscPlaceholders	If included, special marks are painted at places of insertion of text box items[139]
rvcFloatingLines	If included, floating lines are displayed for left-and right-aligned items[695], see TRVStyle.FloatingLineColor[423]

See also:

- RVParagraphMarks[711]
- TRVStyle.SpecialCharactersColor[435]

1.10.32 RVEMPTYTAG

unit RVStyle;

```
const
  RVEMPTYTAG = '';
```

1.11 How To...

How to...

(printing)

➢ Change page size and orientation for printing RichView[715]

➢ Draw page numbers, headers and footers 715
(editing)
➢ Implement commands like "make bold", "apply font", etc. 716
➢ Implement Find and Replace dialogs 716
➢ Switch insert/overtype mode 717
➢ Move the caret to the beginning or to the end of document in editor 717
➢ Make Unicode editor 718
➢ Make a plain text editor 718
➢ Implement smart indents 720
(miscellaneous)
➢ Use third-party graphic classes with RichView 720
➢ Combine several documents in one 720
➢ Remove all formatting 722

1.11.1 change page size and orientation

TRVPrint 513 uses global printer settings of application.

These settings can be modified by standard Delphi/C++Builder components: TPrintDialog, TPrinterSetupDialog (located on Dialogs page of the Component Palette).

You can change them from code using the global Printer object (unit Printers). For example, you can choose printer using Printer.PrinterIndex, or change page orientation using Printer.Orientation.

Changes will be in effect after calling TRVPrint.FormatPages 525 method.

1.11.2 draw page numbers, headers and footers

Complex headers and footers, the same for all pages

The component supports complex headers and footers, they can be multiline, contain images, tables, etc.

They must be assigned to TRVPrint 513 component by the methods SetHeader 529 and SetFooter 528.

See the example:

- Demos\Delphi\Assorted\Printing\Headers\
- Demos\CBuilder\Assorted\Printing\Printing\

There is a limitation of this method: the same header and footer for all pages, with no dynamic fields (like page number).

Drawing headers and footers yourself

If you need different headers for different pages, you need to draw them yourself, using OnPagePrepaint 530 event.

This demo shows how to draw a simple text as a header:

- Demos\Delphi\Assorted\Printing\Printing\
- Demos\CBuilder\Assorted\Printing\Printing\

Since you draw this text yourself, you can draw as many lines as you want.

RichViewActions process this event themselves and draw one plain text line. They understand several codes like page number and pages count.

Drawing complex headers and footers using TRVReportHelper

If you need to draw more complex headers (different for different pages), you can use TRVReportHelper 540 component for this:

http://www.trichview.com/support/files/printheader.zip (for Delphi)

This demo uses TRVReportHelper to draw header in TRVPrint.OnPagePrepaint[530] event.

This demo draws the same header for all pages, but can be easily changed to draw different ones.

It is is shown in this demo:

http://www.trichview.com/support/files/cppheader.zip (for C++Builder).

The next demo uses TRVReportHelper for drawing both header/footer and the main document:

http://www.trichview.com/support/files/a4.zip (for Delphi)

It draws the result on the metafile, but can be easily changed to draw on the printer canvas.

The most complex case:

- headers are complex documents;
- headers have different heights on different pages.

http://www.trichview.com/support/files/printheaderex.zip (for Delphi)

Online updates

New information on this topic will be posted here: ***http://www.trichview.com/forums/viewtopic.php?t=327***

1.11.3 implement commands like "make bold", "apply font", etc.

The key method is ApplyStyleConversion[333].

Example:

- Demos\Delphi\Editors\Editor 2\
- Demos\CBuilder\Editors\Editor 2\

Alternatively, you can use ***RichViewActions***.

1.11.4 implement Find and Replace dialogs

Find

The key method is TRichView.SearchText[256] or TRichViewEdit.SearchText[369].

Search can be case sensitive or not, up or down. "Whole word" option is also supported.

You can convert options from Find dialog to SrchOptions parameter of these methods using GetRVSearchOptions[661] or GetRVESearchOptions[660].

Replace

Use TRichViewEdit.SearchText[369] to select the old string.

Then use InsertText[360] to replace it with the new one. This method removes the selected part of document before performing insertion. If you search in backward direction, set CaretBefore parameter to *True*.

Examples

Demos:

Find dialog is implemented in the main editor demo:

- Demos\Delphi\Editors\Editor 1\
- Demos\CBuilder\Editors\Editor 1\

Both Find and Replace dialogs are implemented in the demo:

- Demos\Delphi\Assorted\Search and Replace\
- Demos\CBuilder\Assorted\Search and Replace\

Alternatively, you can use ***RichViewActions***.

1.11.5 switch insert/overtype mode

The overwrite mode is not supported directly, but you can implement it using events.

The code is below.

```
var IgnoreNextChar: Boolean = False;

procedure TForm1.RichViewEdit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  IgnoreNextChar := RichViewEdit1.SelectionExists[258];
end;

procedure TForm1.RichViewEdit1KeyPress(Sender: TObject; var Key: Char);
var rve: TCustomRichViewEdit[316];
    ItemNo, Offs: Integer;
begin
  if IgnoreNextChar then begin
    IgnoreNextChar := False;
    exit;
  end;
  IgnoreNextChar := False;
  if not ((Key=#9) or (Key>=' ')) then
    exit;
  rve := RichViewEdit1.TopLevelEditor[329];
  if rve.SelectionExists[258] then
    exit;
  ItemNo := rve.CurItemNo[322];
  Offs  := rve.OffsetInCurItem[328];
  if (Offs>=rve.GetOffsAfterItem[235](ItemNo)) then begin
    if (ItemNo+1<rve.ItemCount[180]) and
       not rve.IsFromNewLine[242](ItemNo+1) then begin
      inc(ItemNo);
      Offs := rve.GetOffsBeforeItem[235](ItemNo);
      end
    else
      exit;
  end;
  rve.SetSelectionBounds[271](ItemNo, Offs, ItemNo, Offs+1);
  rve.Invalidate;
end;
```

More details and demo: ***http://www.trichview.com/forums/viewtopic.php?t=171***

1.11.6 move the caret to the beginning or to the end of document in editor

The caret is always at the end of selection.

Selection can be changed using method TRichViewEdit.SetSelectionBounds[271].

Moving to the beginning:

```
var
  ItemNo, Offs: Integer;
...
ItemNo := 0;
Offs   := MyRichViewEdit.GetOffsBeforeItem(ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo,Offs,ItemNo,Offs);
```

Moving to the end:

```
var
  ItemNo, Offs: Integer;
...
ItemNo := MyRichViewEdit.ItemCount-1;
Offs   := MyRichViewEdit.GetOffsAfterItem(ItemNo);
MyRichViewEdit.SetSelectionBounds(ItemNo,Offs,ItemNo,Offs);
```

See also:

- TRichViewEdit.CurItemNo and OffsetInCurItem;
- Example how to move caret to the beginning of paragraph.

1.11.7 make Unicode editor

See in the topic Unicode in TRichView..

1.11.8 make a plain text editor

How to make a plain text editor based on TRichViewEdit

- Exclude all but text from AcceptDragDropFormat property.
- Process OnPaste event (see below).
- Disable all UI commands changing text and paragraph attributes.

```
procedure TMyForm.MyRichViewEditPaste(Sender: TCustomRichViewEdit;
  var DoDefault: Boolean);
var s: String;
begin
  if Clipboard.HasFormat(CF_TEXT) then begin
    s := Clipboard.AsText;
    Sender.InsertText(s, False);
  end;
  DoDefault := False;
end;
```

How to make ONE LINE plain text editor

- Exclude all but text from AcceptDragDropFormat property.
- Process OnPaste event (see below).
- Process OnOleDrop event (see below).
- Disable all UI commands changing text and paragraph attributes or inserting multiline text.
- Set WordWrap property = *False*.
- Include *rvoDoNotWantReturns* in EditorOptions.
- Set VSmallStep property = 1, before calling Format.
- If you do not want tabulators as well, set RVStyle.SpacesInTab = 4.

```
uses Clipbrd, ActiveX;
// Returning one line string made from s.
// This function replaces all line breaks with ' | '.
// You can change this function, for example to return the first line
function MakeOneLineString(const s: String): String;
var p: Integer;
```

```
begin
  Result := AdjustLineBreaks(s);
  while True do begin
    p := Pos(#13#10, Result);
    if p=0 then
      break;
    Delete(Result, p, 2);
    Insert(' | ', Result, p);
 end;
end;

procedure TMyForm.MyRichViewEditOleDrop(Sender: TCustomRichView[162];
  const DataObject: IDataObject; Shift: TShiftState; X, Y: Integer;
  PossibleDropEffects: TRVOleDropEffects[682]; var DropEffect: TRVOleDropEffect[682];
  var DoDefault: Boolean);
var FmtEtc: TFormatEtc;
  StgMedium: TStgMedium;
  ptr: Pointer;
  s: String;
  p, ByteCount, CharCount: Integer;
begin
  DoDefault := False;

  FillChar(StgMedium, sizeof(StgMedium), 0);
  FillChar(FmtEtc, sizeof(FmtEtc), 0);
  FmtEtc.cfFormat := CF_TEXT; // CF_UNICODETEXT for Delphi 2009+
  FmtEtc.dwAspect := DVASPECT_CONTENT;
  FmtEtc.lindex := -1;
  FmtEtc.tymed := TYMED_HGLOBAL;
  if DataObject.GetData(FmtEtc, StgMedium)<>S_OK then
    exit;
  if StgMedium.tymed=TYMED_HGLOBAL then begin
    ptr := GlobalLock(StgMedium.HGlobal);
    try
      ByteCount := GlobalSize(StgMedium.HGlobal);
      CharCount := ByteCount;
      // CharCount := CharCount div 2; // uncomment for Delphi 2009+
      SetLength(s, CharCount);
      Move(ptr^, PChar(s)^, ByteCount);
    finally
      GlobalUnlock(StgMedium.HGlobal);
    end;
    p := Pos(#0, s);
    if p>0 then
      s := Copy(s, 1, p-1);
    MyRichViewEdit.InsertText[360](MakeOneLineString(s), False);
  end;
  ReleaseStgMedium(StgMedium);
end;

procedure TMyForm.MyRichViewEditPaste(Sender: TCustomRichViewEdit[316];
  var DoDefault: Boolean);
var s: String;
begin
 if Clipboard.HasFormat(CF_TEXT) then begin
    s := Clipboard.AsText;
```

```
    Sender.InsertText(MakeOneLineString(s), False);
  end;
  DoDefault := False;
end;
```

1.11.9 implement smart indenting

You can find the examples in the support forum: ***http://www.trichview.com/forums/viewtopic.php?t=2044***

There are two examples available.

Indenting in programming style

When the user presses **Enter**, the code adds space characters at the beginning of the new paragraph (as many spaces as at the beginning of the current paragraph).

Indenting in Microsoft Word style

The code changes FirstIndent 485 and LeftIndent 485 of paragraph, if **Tab** / **Backspace** is pressed when the caret is at the beginning of paragraph. First, it changes FirstIndent 485. Next, if FirstIndent 485 is already changed, it changes LeftIndent 485. The code changes indents only if the caret is at the beginning of non-empty paragraph.

1.11.10 combine several TRichView documents in one

See in the topic TCustomRichView.InsertRVFFromStream 240.

1.11.11 use third-party graphic classes with RichView

Using third party graphic classes

You can use graphic classes created by third party developers.

These classes can be used just like standard classes (TBitmap, TIcon, TMetafile, TJpegImage).

You can add picture in a document using AddPictureEx 204 or InsertPicture 355.

Example:

```
gif: TGifImage;
...
gif := TGifImage.Create;
gif.LoadFromFile('demo.gif');
editor.InsertPicture('Demo', gif, rvvaBaseLine);
```

For example, you can use free graphic classes:

- Anders Melander's TGifImage ***www.torry.net/vcl/graphics/gif/gifimage.exe*** (update for Delphi 6 and newer: ***www.trichview.com/resources/thirdparty/gifimage.zip***);
- TJvGifImage from JEDI's JVCL ***http://jvcl.sourceforge.net***;
- Gustavo Huffenbacher Daud's TPngObject ***www.trichview.com/resources/thirdparty/pngimage.zip***.

Starting from Delphi 2007, an advanced version of Anders Melander's TGifImage is included in VCL, so TGifImage becomes a standard graphic class.

Starting from Delphi 2009, TPngImage is included in VCL.

Gif animation

To support animation 90, assign TCustomRichView.AnimationMode 169 = *rvaniOnFormat*.

The following units must be included in your projects:

- RVGifAnimate for Anders Melander's TGifImage;
- RVJvGifAnimate for TJvGifImage from JEDI's JVCL;
- RVGifAnimate2007 for standard Delphi TGifImage (in Delphi 2007 or newer).

See also: animation in TRichView 90

Controlling automatic insertion of graphics when importing RTF

Sometimes images are inserted automatically (for example, when importing images referred from RTF).

Question: What graphic classes does TRichView use for loading non-standard graphic formats (such as GIF or PNG)?

Answer: RichView uses the graphic class which is registered for the given file extension.

If a third party creator has not associated his graphic class with the extension, you can do it yourself with TPicture.RegisterFileFormat:

```
TPicture.RegisterFileFormat('gif','Gif Image',TGifImage);
```

This registration is especially necessary if you define this graphic format as HTML graphic format 721.

Workaround for Delphi bug in constructors of graphic classes (Delphi 5 or older)

If you use Delphi 5/C++Builder 5 or older, use RVGraphicHandler 710.CreateGraphic (defined in RVFuncs unit) instead of direct call of graphic class constructor.

It is not necessary if you use Delphi 6/C++Builder 6 or newer.

Exporting graphics in HTML

By default, all images are converted to JPEGs. This is obviously undesirable for formats like GIF and PNG.

You can specify which graphic classes must not be converted.

Use RVGraphicHandler 710.RegisterHTMLGraphicFormat procedure, for example:

```
uses RVGrHandler;
...
// call this before the first html export
RVGraphicHandler.RegisterHTMLGraphicFormat(TMyGifImage);
```

This registration affects only images of the given format. Other formats will be converted to JPEGs.

TRichView registers the following classes as HTML graphic formats automatically:

- standard TPngImage (for Delphi 2009+);
- standard TGifImage (for Delphi 2007+), if you include RVGifAnimate2007 unit in your project;
- Anders Melander's TGifImage, if you include RVGifAnimate unit in your project;
- TJvGifImage from JEDI's JVCL, if you include RVJvGifAnimate unit in your project.

Specifying PNG class

Specify graphic class for PNG images using Use RVGraphicHandler 710.RegisterPngGraphic, for example:

```
uses RVFuncs;
...
RVGraphicHandler.RegisterPngGraphic(TPngObject).
```

It allows loading PNG images from RTF files and saving PNG images to RTF and DocX files without converting to bitmaps or metafiles.

In Delphi 2009+, TRichView automatically registers the standard TPngImage as a PNG class.

Specifying JPEG class

Specify graphic class for Jpeg images using Use RVGraphicHandler 710.RegisterJpegGraphic, for example:

```
uses RVFuncs;
...
RVGraphicHandler.RegisterJpegGraphic(TMyJpegImage).
```

If you do not call this method, TRichView uses TJPEGImage.

This class is used when loading JPEG images embedded in RTF.

In Delphi 2009+, TRichView automatically registers the standard TPngImage as a PNG class.

Using DevExpress graphic classes

If you uncomment RVUSEDXPNGIMAGE in RV_Defs.inc, TRichView supports DevExpress classes:

- TcxPngImage for Png
- TdxSmartImage for Bitmap, Jpeg, Tiff, Gif images.

TcxPngImage becomes a default Png class for TRichView, but TdxSmartImage does not become a default Jpeg class, unless you call RVGraphicHandler.RegisterJpegGraphic explicitly.

1.11.12 create a chat window

You can find the complete demo of a chat window on the support forum:
http://www.trichview.com/forums/viewtopic.php?t=63

This demo shows how to implement a chat window with the following features. The user types a plain text message in TEdit. This message can contain special commands, they are processed before displaying in TRichView window.

This demo supports:

- text commands;
- emoticons;
- URL detection.

Examples of text commands:

- **\b** to make text bold (toggle);
- **\cRRGGBB** to change text color.

Emoticons are easily customizable (their codes and graphics are stored in arrays). This demo uses bitmaps, but you can change them to Gif images (see how to use third-party graphic classes 720), including animations 90. For additional ideas about emoticons, see the demo in **Demos\Delphi\Assorted\Graphics\Emoticons**.

URL detection use RVIsURL and RVIsEmail 663 functions. You can use RVIsCustomURL 710 to define your own types of links.

1.11.13 remove text formatting

The code below removes text formatting from the whole document.

It converts indices of all text styles 437 and paragraph styles 431 to 0s. Optionally, it can keep hyperlinks (they are converted to the hypertext style with the lowest index).

Optionally, it can remove list markers 131 as well.

This code changes text items 120, tabulators 121 and list markers. All other item types remains unchanged.

Note 1: if the document contains both Unicode 99 and non-Unicode text, or non-Unicode text of different Charsets, some text information may be lost.

Note 2: this is not an editing operation. When called for TRichViewEdit, undo buffer must be cleared.

```
procedure RemoveFormatting(RVData: TCustomRVData 650; RemoveMarkers, KeepLinks: Boolean);
var i, HypertextStyleNo: Integer;
```

```
  {.....................................................}
  procedure DoRemoveFormatting(RVData: TCustomRVData[650]);
  var i, r, c, StyleNo, StyleNoTo: Integer;
      PB: Boolean;
      table: TRVTableItemInfo;
  begin
    for i := RVData.ItemCount[180]-1 downto 0 do
    begin
      RVData.GetItem[225](i).ParaNo := 0;
      StyleNo := RVData.GetItemStyle[229](i);
      case StyleNo of
        rvsTable[711]:
          begin
            table := TRVTableItemInfo[565](RVData.GetItem[225](i));
            for r := 0 to table.RowCount[580]-1 do
              for c := 0 to table.ColCount[575]-1 do
                if table.Cells[574][r,c]<>nil then
                  DoRemoveFormatting(table.Cells[574][r,c].GetRVData[613]);
          end;
        rvsListMarker[711]:
          if RemoveMarkers then
          begin
            PB := RVData.PageBreaksBeforeItems[184][i];
            RVData.DeleteItems[214](i, 1);
            if i<RVData.ItemCount[180] then begin
              RVData.GetItem[225](i).SameAsPrev := False;
              RVData.GetItem[225](i).PageBreakBefore := PB;
            end;
          end;
        rvsTab[711]:
          if RVData.GetRVStyle.TextStyles[437][
            TRVTabItemInfo[563](RVData.GetItem[225](i)).TextStyleNo].Jump[479] and KeepLinks then
            TRVTabItemInfo(RVData.GetItem[225](i)).TextStyleNo := HypertextStyleNo
          else
          begin
            TRVTabItemInfo(RVData.GetItem[225](i)).TextStyleNo := 0;
            RVData.SetItemTag[267](i, '');
          end;
        1..MaxInt:
          begin
            if KeepLinks and RVData.GetRVStyle.TextStyles[437][StyleNo].Jump[479] then
              StyleNoTo := HypertextStyleNo
            else
            begin
              StyleNoTo := 0;
              RVData.SetItemTag[267](i, '');
            end;
            RVData.GetItem[225](i).StyleNo := StyleNoTo;
          end;
      end;
    end;
  end;
  {.....................................................}
begin
  HypertextStyleNo := 0;
  if KeepLinks then
```

```
    for i := 0 to RVData.GetRVStyle.TextStyles[437].Count-1 do
      if RVData.GetRVStyle.TextStyles[437][i].Jump[479] then
      begin
        HypertextStyleNo := i;
        break;
      end;
  DoRemoveFormatting(RVData);
end;
```

How to use:

```
  RemoveFormatting(RichViewEdit1.RVData[187], True, True);
  NormalizeRichView(RichViewEdit1.RVData[187]); // from RVNormalize.pas, from RichViewActions
  RichViewEdit1.DeleteUnusedStyles[216](True, True, True);
  RichViewEdit1.ClearUndo[339];
  RichViewEdit1.Format[217];
```

More information: ***http://www.trichview.com/forums/viewtopic.php?t=2392***

Index

- A -

AcceptDragDropFormats 321
TRichViewEdit 321

AcceptPasteFormats 321
TRichViewEdit 321

Active 532
TRVVirtualPrinterProperties 532

ActualCurTextStyleNo 322
TCustomRichViewEdit 322

Add 203, 452, 457, 459, 506
TFontInfos 452
TParaInfos 457
TRichView 203
TRVListInfos 459
TRVListLevelCollection 506

AddBreak 196
TRichView 196

AddBreakEx 196
TRichView 196

AddBreakExTag 196
TRichView 196

AddBreakTag 196
TRichView 196

AddBullet 197
TRichView 197

AddBulletEx 197
TRichView 197

AddBulletExTag 197
TRichView 197

AddCheckpoint 202
TRichView 202

AddCheckpointTag 202
TRichView 202

AddControl 198
TRichView 198

AddControlEx 198
TRichView 198

AddControlExTag 198
TRichView 198

AddFmt 198
TRichView 198

AddFont 452
TFontInfos 452

AddFontEx 452
TFontInfos 452

AddFrom 503
TRVTabInfos 503

AddHotPicture 199
TRichView 199

AddHotPictureTag 199
TRichView 199

AddHotspot 200
TRichView 200

AddHotspotEx 200
TRichView 200

AddHotspotExTag 200
TRichView 200

AddItem 201
TRichView 201

AddNamedCheckpoint 202
TRichView 202

AddNamedCheckpointEx 202
TRichView 202

AddNamedCheckpointExTag 202
TRichView 202

AddNL 203
TRichView 203

AddNLATag 203
TRichView 203

AddNLTag 203
TRichView 203

AddNLWTag 203
TRichView 203

AddPicture 204
TRichView 204

AddPictureEx 204
TRichView 204

AddPictureExTag 204
TRichView 204

AddTab 205
TRichView 205

AddTag 203
TRichView 203

AddTextNL 206
TRichView 206

AddTextNLA 206
TRichView 206

AddTextNLW 206
TRichView 206

AddTextStyle 438
TRVStyle 438

AddXXX methods of TRichView 77

AdjustControlPlacement 330
TRichViewEdit 330

AdjustControlPlacement2 330
TRichViewEdit 330
Align 504
TRVTabInfo 504
Alignment 483, 618
TCustomRVParaInfo 483
TRVLabelItemInfo 618
AllNumbered 494
TRVListInfo 494
AllowSelection 169
TRichView 169
AllowTransparentDrawing 541
TRVReportHelper 541
Animation in TRichView 90
AnimationMode 169
TRichView 169
AppendFrom 207
TRichView 207
AppendRVFFromStream 208
TRichView 208
ApplyListStyle 331
TRichViewEdit 331
ApplyParaStyle 331
TRichViewEdit 331
ApplyParaStyleConversion 332
TRichViewEdit 332
ApplyParaStyleTemplate 332
TRichViewEdit 332
ApplyStyleConversion 333
TRichViewEdit 333
ApplyStyleTemplate 333
TRichViewEdit 333
ApplyTextStyle 334
TRichViewEdit 334
ApplyTextStyleTemplate 334
TRichViewEdit 334
ApplyToParaStyle 498
TRVStyleTemplate 498
ApplyToTextStyle 499
TRVStyleTemplate 499
Assign 477, 480, 489, 492
TCustomRVFontInfo 477
TCustomRVParaInfo 489
TFontInfo 480
TParaInfo 492
AssignDocParameters 522
TRVPrint 522
AssignProperties 584
TRVTableItemInfo 584
AssignSoftPageBreaks 209
TCustomRichView 209
AssignSource 522
TRVPrint 522
AssignStyleTemplates 461
TRVStyleTemplateCollection 461
AssignTo 453, 457, 459, 477
TCustomRVFontInfo 477
TFontInfos 453
TParaInfos 457
TRVListInfos 459
AssignToRVPrint 658
TRVReportHelperWithHeaderFooters 658
AssignToStrings 461
TRVStyleTemplateCollection 461
Author 311
TRVDocParameters 311
AutoDeleteUnusedStyles 412
TDBRichViewEdit 412
AutoDisplay 408, 412
TDBRichView 408
TDBRichViewEdit 412
AutoHideTableGridLines 304
TRVRTFReaderProperties 304

- B -

BackColor 467
TCustomRVFontInfo 467
Background 484, 635
TCustomRVParaInfo 484
TRVBoxProperties 635
BackgroundBitmap 169
TRichView 169
BackgroundImage 567, 605
TRVTableCellData 605
TRVTableItemInfo 567
BackgroundImageFileName 568, 606
TRVTableCellData 606
TRVTableItemInfo 568
BackgroundStyle 170, 568, 606
TRichView 170
TRVTableCellData 606
TRVTableItemInfo 568
BasePathLinks 305
TRVRTFReaderProperties 305
BeginItemModify 335
TRichViewEdit 335
BeginOleDrag 209
TRichView 209
BeginUndoCustomGroup 336

BeginUndoCustomGroup 336
TRichViewEdit 336
BeginUndoGroup 336
TRichViewEdit 336
BeginUpdate 336
TRichViewEdit 336
BestHeight 606
TRVTableCellData 606
BestWidth 568, 607
TRVTableCellData 607
TRVTableItemInfo 568
BiDiMode 100, 170, 467, 484
TCustomRVFontInfo 467
TCustomRVParaInfo 484
TRichView 170
Bidirectional text in TRichView 100
Border 485, 635
TCustomRVParaInfo 485
TRVBoxProperties 635
BorderColor 569, 607
TRVTableCellData 607
TRVTableItemInfo 569
BorderHSpacing 569
TRVTableItemInfo 569
BorderLightColor 569, 608
TRVTableCellData 608
TRVTableItemInfo 569
BorderOffsets 501, 502
TRVBackgroundRect 501
TRVBorder 502
Borders and background of paragraphs in TRichView 73
BorderStyle 545, 570
TRVScroller 545
TRVTableItemInfo 570
BorderVSpacing 570
TRVTableItemInfo 570
BorderWidth 571
TRVTableItemInfo 571
BottomMargin 171, 311
TRichView 171
TRVDocParameters 311
BoxPosition 634
TRVSidenoteItemInfo 634
BoxProperties 634
TRVSidenoteItemInfo 634
Breaks 125
Building TRichView document 77
Bullets 128
ButtonType 402
TRVSmartPopupProperties 402

- C -

CanChange 336
TRichViewEdit 336
CanMergeCells 584
TRVTableItemInfo 584
CanMergeSelectedCells 585
TRVTableItemInfo 585
CanPaste 337
TRichViewEdit 337
CanPasteRTF 337
TRichViewEdit 337
CanPasteRVF 337
TRichViewEdit 337
Cell Merging 152
CellBorderColor 571
TRVTableItemInfo 571
CellBorderLightColor 571
TRVTableItemInfo 571
CellBorderStyle 572
TRVTableItemInfo 572
CellBorderWidth 572
TRVTableItemInfo 572
CellHPadding 573
TRVTableItemInfo 573
CellHSpacing 573
TRVTableItemInfo 573
CellOverrideColor 573
TRVTableItemInfo 573
CellPadding 574
TRVTableItemInfo 574
Cells 574
TRVTableItemInfo 574
CellVPadding 574
TRVTableItemInfo 574
CellVSpacing 574
TRVTableItemInfo 574
CFRV_HTML 698
CFRV_RTF 698
CFRV_RVF 698
CFRV_URL 698
Change 337, 414
TDBRichViewEdit 414
TRichViewEdit 337
Changed 585
TRVTableItemInfo 585
ChangeListLevels 338
TRichViewEdit 338

ChangeStyleTemplates 338
TRichViewEdit 338
CharScale 467
TCustomRVFontInfo 467
Charset 468
TCustomRVFontInfo 468
CharsetForUnicode 305
TRVRTFReaderProperties 305
CharSpacing 468
TCustomRVFontInfo 468
CheckpointColor 420
TRVStyle 420
CheckpointEvColor 420
TRVStyle 420
Checkpoints in TRichView 65
Clear 210, 338, 553
TCustomRVPrint 553
TRichView 210
TRichViewEdit 338
ClearLeft 171
TRichView 171
ClearLiveSpellingResults 210
TRichView 210
ClearParaFormat 462
TRVStyleTemplateCollection 462
ClearRight 172
TRichView 172
ClearSoftPageBreaks 211
TRichView 211
ClearTextFlow 339
TRichViewEdit 339
ClearTextFormat 462
TRVStyleTemplateCollection 462
ClearUndo 339
TRichViewEdit 339
ClickMode 558
TCustomRVPrintPreview 558
ClientToDocument 211
TRichView 211
ClipMargins 518
TRVPrint 518
ColBandSize 575
TRVTableItemInfo 575
ColCount 575
TRVTableItemInfo 575
Color 173, 403, 421, 469, 501, 502, 558, 575, 608
TCustomRVFontInfo 469
TCustomRVPrintPreview 558
TRichView 173
TRVBackgroundRect 501
TRVBorder 502
TRVSmartPopupProperties 403
TRVStyle 421
TRVTableCellData 608
TRVTableItemInfo 575
ColorMode 549
TCustomRVPrint 549
Colors and layout of tables 148
ColSpan 608
TRVTableCellData 608
Comments 311
TRVDocParameters 311
Component Editor for TRichView 168
ContinuousPrint 523
TRVPrint 523
Controls 126
Controls - Documents - Items 104
ConvertDocToDifferentUnits 211
TRichView 211
ConvertDocToPixels 211
TRichView 211
ConvertDocToTwips 211
TRichView 211
ConvertHighlight 305
TRVRTFReaderProperties 305
ConvertToDifferentUnits 438
TRVStyle 438
ConvertToHotPicture 339
TRichViewEdit 339
ConvertToPicture 339
TRichViewEdit 339
ConvertToPixels 438
TRVStyle 438
ConvertToTwips 438
TRVStyle 438
ConvertToUnit 315
TRVDocParameters 315
ConvertToUnits 523
TRVPrint 523
Copy 212
TRichView 212
CopyDef 212
TRichView 212
CopyImage 212
TRichView 212
CopyRTF 213
TRichView 213
CopyRVF 213
TRichView 213
CopyText 213

CopyText 213
TRichView 213
CopyTextA 213
TRichView 213
CopyTextW 213
TRichView 213
Count 539
TRVOfficeCnvList 539
CPEventKind 173
TRichView 173
Create 214, 340, 409, 414, 438, 453, 457, 478, 481, 489, 492, 524, 536, 548, 562, 585, 620, 623, 626, 630, 632, 634, 643, 645, 647, 648, 658
TCustomRVFontInfo 478
TCustomRVParaInfo 489
TCustomRVPrintPreview 562
TDBRichView 409
TDBRichViewEdit 414
TFontInfo 481
TFontInfos 453
TParaInfo 492
TParaInfos 457
TRichView 214
TRichViewEdit 340
TRVEndnoteItemInfo 630
TRVFootnoteItemInfo 632
TRVLabelItemInfo 620
TRVMathItemInfo 623
TRVNoteReferenceItemInfo 645
TRVOfficeConverter 536
TRVPageCountItemInfo 648
TRVPageNumberItemInfo 647
TRVPrint 524
TRVReportHelperWithHeaderFooters 658
TRVScroller 548
TRVSeqItemInfo 626
TRVSidenoteItemInfo 634
TRVStyle 438
TRVTableItemInfo 585
TRVTextBoxItemInfo 643
CreateEx 586, 620, 627, 628, 630, 632, 634, 643, 646, 648, 649
TCustomRVNoteItemInfo 628
TRVEndnoteItemInfo 630
TRVFootnoteItemInfo 632
TRVLabelItemInfo 620
TRVNoteReferenceItemInfo 646
TRVPageCountItemInfo 649
TRVPageNumberItemInfo 648
TRVSeqItemInfo 627
TRVSidenoteItemInfo 634
TRVTableItemInfo 586
TRVTextBoxItemInfo 643
crJump 108
crRVZoomIn 108
crRVZoomOut 108
CurItemNo 322
TRichViewEdit 322
CurItemStyle 323
TRichViewEdit 323
CurParaStyleNo 323
TRichViewEdit 323
CurrentItemColor 421
TRVStyle 421
Cursor 174, 618
TRichView 174
TRVLabelItemInfo 618
Cursors in TRichView 108
CurTextStyleNo 324
TRichViewEdit 324
Custom drawing in TRichView 108
Custom item types in TRichView 144
CustomCaretInterval 324
TRichViewEdit 324
CutDef 340
TRichViewEdit 340

- D -

DataField 408, 413
TDBRichView 408
TDBRichViewEdit 413
DataSource 408, 413
TDBRichView 408
TDBRichViewEdit 413
DefaultPictureVAlign 325
TRichViewEdit 325
DefCodePage 421
TRVStyle 421
DefStyleName 460
TRVStyleTemplateCollection 460
DefStyleNo 491
TParaInfo 491
DefTabWidth 422
TRVStyle 422
DefUnicodeStyle 422
TRVStyle 422
DeleteCols 586
TRVTableItemInfo 586
DeleteEmptyCols 587
TRVTableItemInfo 587

DeleteEmptyRows 587
TRVTableItemInfo 587
DeleteItems 214
TRichView 214
DeleteList 504
TRVTabInfos 504
DeleteMarkedStyles 215
TRichView 215
DeleteParas 215
TRichView 215
DeleteRows 587
TRVTableItemInfo 587
DeleteSection 215
TRichView 215
DeleteSelectedCols 588
TRVTableItemInfo 588
DeleteSelectedRows 588
TRVTableItemInfo 588
DeleteSelection 340
TRichViewEdit 340
DeleteTextStyle 438
TRVStyle 438
DeleteUnusedStyles 216
TRichView 216
Delimiters 174
TRichView 174
Deselect 216, 588
TRichView 216
TRVTableItemInfo 588
Destroy 217, 340, 409, 415, 438, 536, 589
TDBRichView 409
TDBRichViewEdit 415
TRichView 217
TRichViewEdit 340
TRVOfficeConverter 536
TRVStyle 438
TRVTableItemInfo 589
DifferentUnitsToUnits 439
TRVStyle 439
DisabledFontColor 422
TRVStyle 422
DisplayInline 621
TRVMathItemInfo 621
DocObjects 174
TRichView 174
DocParameters 176
TRichView 176
DocProperties 176
TRichView 176
Document 628
TCustomRVNoteItemInfo 628
DocumentHeight 177
TRichView 177
DocumentToClient 211
TRichView 211
DoInPaletteMode 177
TRichView 177
Drag and drop in TRichView 89
Drag&Drop in TRichView 89
Draw 481
TFontInfo 481
DrawButton 660
DrawControl 660
DrawEdit 660
DrawMemo 660
DrawPage 524, 541
TRVPrint 524
TRVReportHelper 541
DrawPageAt 524, 542
TRVPrint 524
TRVReportHelper 542
DrawPanel 660
DrawPreview 524
TRVPrint 524

- E -

Edit 651
TCustomRVData 651
EditCell 589
TRVTableItemInfo 589
Editing Cells 151
EditorOptions 325
TRichViewEdit 325
EmptyWidth 469
TCustomRVFontInfo 469
EndAt 549
TCustomRVPrint 549
EndItemModify 341
TRichViewEdit 341
EndnoteNumbering 422
TRVStyle 422
Endnotes 135
EndUpdate 341
TRichViewEdit 341
ERichViewError 649
ErrorCode 306, 533
TRVOfficeConverter 533
TRVRTFReaderProperties 306
EvenColumnsColor 583

EvenColumnsColor 583
TRVTableItemInfo 583
EvenRowsColor 583
TRVTableItemInfo 583
Example
Converting table to text 159
Dividing table 160
Inserting controls what can modify themselves in DBRichViewEdit 417
Loading UTF-8 files 316
Moving caret to the beginning of paragraph 405
OnDropFiles example 405
Using viewer-style methods in DBRichViewEdit 416
ExcludeDocXExportConverter 534
TRVOfficeConverter 534
ExcludeDocXImportConverter 534
TRVOfficeConverter 534
ExcludeHTMLExportConverter 534
TRVOfficeConverter 534
ExcludeHTMLImportConverter 534
TRVOfficeConverter 534
Export of Tables 157
Export of TRichView contents to HTML 96
ExportConverters 535
TRVOfficeConverter 535
ExportRTF 536
TRVOfficeConverter 536
ExportRV 537
TRVOfficeConverter 537
ExtensionsInFilter 535
TRVOfficeConverter 535
ExtractMetafileBitmaps 306
TRVRTFReaderProperties 306

- F -

FacingPages 312, 518
TRVDocParameters 312
TRVPrint 518
Field 409, 413
TDBRichView 409
TDBRichViewEdit 413
FieldFormat 413
TDBRichViewEdit 413
FieldHighlightColor 423
TRVStyle 423
FieldHighlightType 423
TRVStyle 423
Filter 539
TRVOfficeConverterInfo 539
Find 504
TRVTabInfos 504
FindById 462
TRVStyleTemplateCollection 462
FindByName 462
TRVStyleTemplateCollection 462
FindCheckpointByName 217
TRichView 217
FindCheckpointByTag 217
TRichView 217
FindControlItemNo 217
TRichView 217
FindItemById 462
TRVStyleTemplateCollection 462
FindItemByName 462
TRVStyleTemplateCollection 462
FindParaStyle 440
TRVStyle 440
FindStyleWithAlignment 457
TParaInfos 457
FindStyleWithCharset 453
TFontInfos 453
FindStyleWithColor 453
TFontInfos 453
FindStyleWithFont 454
TFontInfos 454
FindStyleWithFontName 454
TFontInfos 454
FindStyleWithFontSize 454
TFontInfos 454
FindStyleWithFontStyle 455
TFontInfos 455
FindStyleWithLevels 459
TRVListInfos 459
FindSuchStyle 455, 458, 459
TFontInfos 455
TParaInfos 458
TRVListInfos 459
FindSuchStyleEx 455, 458
TFontInfos 455
TParaInfos 458
FindTextStyle 440
TRVStyle 440
Finished 542
TRVReportHelper 542
First 562
TCustomRVPrintPreview 562
FirstColumnColor 583
TRVTableItemInfo 583

FirstIndent 485, 507
TCustomRVParaInfo 485
TRVListLevel 507
FirstItemVisible 179
TRichView 179
FirstJumpNo 179
TRichView 179
FirstPageNo 550
TCustomRVPrint 550
FixMarginsMode 518
TRVPrint 518
FloatingLineColor 423
TRVStyle 423
Font 507
TRVListLevel 507
FontName 470, 622
TCustomRVFontInfo 470
TRVMathItemInfo 622
FontQuality 424
TRVStyle 424
FontSizeDouble 622
TRVMathItemInfo 622
FooterY 312, 519
TRVDocParameters 312
TRVPrint 519
FootnoteNumbering 422
TRVStyle 422
FootnotePageReset 425
TRVStyle 425
Footnotes 136
ForbiddenIds 461
TRVStyleTemplateCollection 461
ForceFieldHighlight 327
TRichViewEdit 327
Format 217
TRichView 217
FormatNextPage 542
TRVReportHelper 542
FormatPages 525
TRVPrint 525
FormatString 508, 625
TRVListLevel 508
TRVSeqItemInfo 625
FormatStringW 509
TRVListLevel 509
FormatTail 218
TRichView 218
FullRedraw 425
TRVStyle 425

- G -

GetAllText 665
GetAsDifferentUnits 439
TRVStyle 439
GetAsPixels 439
TRVStyle 439
GetAsRVUnits 439
TRVStyle 439
GetAsTwips 439
TRVStyle 439
GetBreakInfo 218
TRichView 218
GetBulletInfo 219
TRichView 219
GetCellAt 589
TRVTableItemInfo 589
GetCheckpointAtCaret 341
TRichViewEdit 341
GetCheckpointByNo 220
TRichView 220
GetCheckpointInfo 220
TRichView 220
GetCheckpointItemNo 221
TRichView 221
GetCheckpointNo 221
TRichView 221
GetCheckpointXY 221
TRichView 221
GetCheckpointY 221
TRichView 221
GetCheckpointYEx 222
TRichView 222
GetControlInfo 222
TRichView 222
GetCurrentBreakInfo 341
TRichViewEdit 341
GetCurrentBulletInfo 342
TRichViewEdit 342
GetCurrentChar 665
GetCurrentCheckpoint 343
TRichViewEdit 343
GetCurrentControlInfo 343
TRichViewEdit 343
GetCurrentHotspotInfo 344
TRichViewEdit 344
GetCurrentItem 345
TRichViewEdit 345
GetCurrentItemEx 345

GetCurrentItemEx 345
TRichViewEdit 345
GetCurrentItemExtraIntProperty 346
GetCurrentItemExtraIntPropertyEx 346
TRichViewEdit 346
GetCurrentItemExtraStrProperty 346
TRichViewEdit 346
GetCurrentItemExtraStrPropertyEx 346
TRichViewEdit 346
GetCurrentItemText 347
TRichViewEdit 347
GetCurrentItemTextA 347
TRichViewEdit 347
GetCurrentItemTextW 347
TRichViewEdit 347
GetCurrentItemVAlign 348
TRichViewEdit 348
GetCurrentLineCol 348
TRichViewEdit 348
GetCurrentLineText 665
GetCurrentMisspelling 348
TRichViewEdit 348
GetCurrentParaSectionText 665
GetCurrentParaText 665
GetCurrentPictureInfo 349
TRichViewEdit 349
GetCurrentTag 349
TRichViewEdit 349
GetCurrentTextInfo 350
TRichViewEdit 350
GetCurrentWord 665
GetEditedCell 590
TRVTableItemInfo 590
GetExportFilter 537
TRVOfficeConverter 537
GetFirstCheckpoint 223
TRichView 223
GetFirstItemOnPage 553
TCustomRVPrint 553
GetFocusedItem 223
TRichView 223
GetFooter 658
TRVReportHelperWithHeaderFooters 658
GetFooterRect 525
TRVPrint 525
GetHeader 659
TRVReportHelperWithHeaderFooters 659
GetHeaderRect 525
TRVPrint 525
GetHotspotInfo 224
TRichView 224
GetImportFilter 537
TRVOfficeConverter 537
GetItem 225
TRichView 225
GetItemAt 225
TCustomRichView 225
GetItemCheckpoint 226
TRichView 226
GetItemClientCoords 226
TRichView 226
GetItemCoords 226
TRichView 226
GetItemCoordsEx 227
TRichView 227
GetItemExtraIntProperty 227
TRichView 227
GetItemExtraIntPropertyEx 227
TRichView 227
GetItemExtraStrProperty 228
TCustomRichView 228
GetItemExtraStrPropertyEx 228
TCustomRichView 228
GetItemNo 229
TRichView 229
GetItemPara 229
TRichView 229
GetItemStyle 229
TRichView 229
GetItemTag 230
TRichView 230
GetItemText 230
TRichView 230
GetItemTextA 230
TRichView 230
GetItemTextW 230
TRichView 230
GetItemVAlign 230
TRichView 230
GetJumpPointItemNo 231
TRichView 231
GetJumpPointLocation 231
TRichView 231
GetJumpPointY 232
TRichView 232
GetLastCheckpoint 232
TRichView 232
GetLastPageHeight 543
TRVReportHelper 543
GetLineNo 233

GetLineNo 233
TRichView 233
GetListMarkerInfo 233
TRichView 233
GetListStyleClass 439
TRVStyle 439
GetMainCell 617
TRVTableRows 617
GetNextCheckpoint 234
TRichView 234
GetNextFileName 277
GetNormalizedSelectionBounds 590
TRVTableItemInfo 590
GetOffsAfterItem 235
TRichView 235
GetOffsBeforeItem 235
TRichView 235
GetOriginEx 653
TCustomRVFormattedData 653
GetPageNo 553
TCustomRVPrint 553
GetParaStyleClass 440
TRVStyle 440
GetPictureInfo 236
TRichView 236
GetPrevCheckpoint 237
TRichView 237
GetRVData 613, 652
TCustomRVData 652
TRVTableCellData 613
GetRVDataText 665
GetRVESearchOptions 660
GetRVSearchOptions 661
GetSelectedImage 237
TRichView 237
GetSelectionBounds 237, 591
TRichView 237
TRVTableItemInfo 591
GetSelText 238
TRichView 238
GetSelTextA 238
TRichView 238
GetSelTextW 238
TRichView 238
GetSourceRVData 652
TCustomRVData 652
GetTextInfo 239
TRichView 239
GetTextStyleClass 440
TRVStyle 440
GetVisibleText 665
GetWordAt 239
TRichView 239
GetWordAtA 239
TRichView 239
GetWordAtW 239
TRichView 239
GridColor 425
TRVStyle 425
GridReadOnlyColor 425
TRVStyle 425
GridReadOnlyStyle 425
TRVStyle 425
GridStyle 425
TRVStyle 425

- H -

HasNumbering 494, 513
TRVListInfo 494
TRVListLevel 513
HeaderY 312, 519
TRVDocParameters 312
TRVPrint 519
HeadingRowColor 583
TRVTableItemInfo 583
HeadingRowCount 576
TRVTableItemInfo 576
Height 636
TRVBoxProperties 636
HeightType 636
TRVBoxProperties 636
Hidden text in TRichView 76
Hint 403, 609
TRVSmartPopupProperties 403
TRVTableCellData 609
HorizontalAlignment 639
TRVBoxPosition 639
HorizontalAnchor 638
TRVBoxPosition 638
HorizontalOffset 639
TRVBoxPosition 639
HorizontalPositionKind 639
TRVBoxPosition 639
Hot-Pictures 125
Hotspots 129
HOutermostRule 576
TRVTableItemInfo 576
HoverBackColor 470
TCustomRVFontInfo 470

HoverColor 403, 426, 470
TCustomRVFontInfo 470
TRVSmartPopupProperties 403
TRVStyle 426
HoverEffects 471
TCustomRVFontInfo 471
HoverLineColor 403
TRVSmartPopupProperties 403
HoverUnderlineColor 471
TCustomRVFontInfo 471
How to
(List) 714
change page size and orientation for printing RichView 715
combine several TRichView documents in one 720
copy one TRichView to another 207
create a chat window 722
implement commands like "make bold", "apply font" 716
implement Find and Replace dialogs 716
implement headers and footers 715
implement smart indents 720
make a plain text editor 718
make Unicode editor 718
move a caret to the beginning or to the end of document in TRichViewEdit 717
remove text formatting 722
switch insert/overtype mode 717
use third-party graphic classes with RichView 720
HRuleColor 576
TRVTableItemInfo 576
HRuleWidth 577
TRVTableItemInfo 577
HScrollMax 545
TRVScroller 545
HScrollPos 545
TRVScroller 545
HScrollVisible 545
TRVScroller 545
HTML and TRichView 96
HTMLFormatName 698
Hypertext in TRichView 69

- I -

Id 495
TRVStyleTemplate 495
IgnoreBookmarks 306
TRVRTFReaderProperties 306
IgnoreEscape 414
TDBRichViewEdit 414
IgnoreFields 307
TRVRTFReaderProperties 307
IgnoreNotes 307
TRVRTFReaderProperties 307
IgnorePageBreaks 550
TCustomRVPrint 550
IgnorePictures 307
TRVRTFReaderProperties 307
IgnoreSequences 307
TRVRTFReaderProperties 307
IgnoreTables 307
TRVRTFReaderProperties 307
ImageIndex 404, 509
TRVListLevel 509
TRVSmartPopupProperties 404
ImageList 404, 509
TRVListLevel 509
TRVSmartPopupProperties 404
imgSaveNo 277
ImportConverters 535
TRVOfficeConverter 535
ImportRTF 538
TRVOfficeConverter 538
ImportRV 538
TRVOfficeConverter 538
InactiveSelColor 426
TRVStyle 426
InactiveSelTextColor 426
TRVStyle 426
Init 543
TRVReportHelper 543
InplaceEditor 546
TRVScroller 546
InsertBreak 350
TRichViewEdit 350
InsertBullet 351
TRichViewEdit 351
InsertCheckpoint 351
TRichViewEdit 351
InsertCols 591
TRVTableItemInfo 591
InsertColsLeft 592
TRVTableItemInfo 592
InsertColsRight 592
TRVTableItemInfo 592
InsertControl 352
TRichViewEdit 352
InsertFromRVST 463

InsertFromRVST 463
TRVStyleTemplateCollection 463
InsertHotPicture 352
TRichViewEdit 352
InsertHotspot 353
TRichViewEdit 353
InsertItem 353
TRichViewEdit 353
InsertOEMTextFromFile 354
TRichViewEdit 354
InsertPageBreak 355
TRichViewEdit 355
InsertPicture 355
TRichViewEdit 355
InsertRows 592
TRVTableItemInfo 592
InsertRowsAbove 593
TRVTableItemInfo 593
InsertRowsBelow 593
TRVTableItemInfo 593
InsertRTFFromFileEd 356
TRichViewEdit 356
InsertRTFFromStreamEd 357
TRichViewEdit 357
InsertRVFFromFileEd 357
TRichViewEdit 357
InsertRVFFromStream 240
TRichView 240
InsertRVFFromStreamEd 358
TRichViewEdit 358
InsertStringATag 359
TRichViewEdit 359
InsertStringTag 359
TRichViewEdit 359
InsertStringWTag 359
TRichViewEdit 359
InsertTab 359
TRichViewEdit 359
InsertText 360
TRichViewEdit 360
InsertTextA 360
TRichViewEdit 360
InsertTextFromFile 361
TRichViewEdit 361
InsertTextFromFileW 361
TRichViewEdit 361
InsertTextW 360
TRichViewEdit 360
InternalWidth 502
TRVBorder 502
Intersect 504
TRVTabInfos 504
InvalidItem 451, 456
TFontInfos 451
TParaInfos 456
InvalidPicture 427
TRVStyle 427
IsCellSelected 593
TRVTableItemInfo 593
IsEqual 478, 482, 490, 492
TCustomRVFontInfo 478
TCustomRVParaInfo 490
TFontInfo 482
TParaInfo 492
IsFromNewLine 242
TRichView 242
IsParaStart 242
TRichView 242
IsValidImporter 539
TRVOfficeConverter 539
Item types
Breaks 125
Bullets 128
Controls 126
Endnotes 135
Footnotes 136
Hot-Pictures 125
Hotspots 129
Labels 133
List Markers 131
Mathematical expressions 143
Numbered sequences 134
Page counts 142
Page numbers 141
Pictures 122
References to parent endnotes and footnotes 140
Sidenotes 137
Tables 130
Tabulators 121
Text 120
Text boxes 139
Item types in TRichView 118, 563
ItemCount 180
TRichView 180
Items 451, 456, 459, 461, 503, 506, 539, 615, 617
TFontInfos 451
TParaInfos 456
TRVListInfos 459
TRVListLevelCollection 506
TRVOfficeCnvList 539

Items 451, 456, 459, 461, 503, 506, 539, 615, 617
TRVStyleTemplateCollection 461
TRVTabInfos 503
TRVTableRow 615
TRVTableRows 617

- J -

Jump 479
TFontInfo 479
JumpCursor 427, 472
TCustomRVFontInfo 472
TRVStyle 427

- K -

KeepTogether 615
TRVTableRow 615
Kind 495
TRVStyleTemplate 495

- L -

Labels 133
Last 562
TCustomRVPrintPreview 562
LastColumnColor 583
TRVTableItemInfo 583
LastItemVisible 180
TRichView 180
LastLineAlignment 483
TCustomRVParaInfo 483
LastPageNo 550
TCustomRVPrint 550
LastRowColor 583
TRVTableItemInfo 583
Leader 505
TRVTabInfo 505
LeftIndent 485, 510
TCustomRVParaInfo 485
TRVListLevel 510
LeftMargin 180, 312
TRichView 180
TRVDocParameters 312
Levels 493
TRVListInfo 493
LinearToRichView 666
LineBreaksAsParagraphs 307
TRVRTFReaderProperties 307
LineColor 404
TRVSmartPopupProperties 404
LineCount 181
TRichView 181
LineSelectCursor 427
TRVStyle 427
LineSpacing 486
TCustomRVParaInfo 486
LineSpacingType 486
TCustomRVParaInfo 486
LineWrapMode 428
TRVStyle 428
List Markers 131
ListStyles 429
TRVStyle 429
ListType 510
TRVListLevel 510
Live spelling check in TRichView 101
LiveSpellingColor 430
TRVStyle 430
LiveSpellingMode 327
TRichViewEdit 327
LiveSpellingValidateWord 242
TRichView 242
LoadField 409, 415
TDBRichView 409
TDBRichViewEdit 415
LoadFromRVST 462
TRVStyleTemplateCollection 462
LoadFromStream 243, 593
TRichView 243
TRVTableItemInfo 593
LoadINI 440
TRVStyle 440
LoadReg 441
TRVStyle 441
LoadRTF 243
TRichView 243
LoadRTFFromStream 244
TRichView 244
LoadRVF 245
TRichView 245
LoadRVFFromStream 245
TRichView 245
LoadText 246
TRichView 246
LoadTextFromStream 247
TRichView 247
LoadTextFromStreamW 247
TRichView 247
LoadTextW 246

LoadTextW 246
TRichView 246

- M -

MainDoc 657
TRVReportHelperWithHeaderFooters 657
MainRVStyle 430
TRVStyle 430
MakePreview 526
TRVPrint 526
MakeScaledPreview 526
TRVPrint 526
Margins 520
TRVPrint 520
MarginsPen 559
TCustomRVPrintPreview 559
MarkerAlignment 511
TRVListLevel 511
MarkerIndent 512
TRVListLevel 512
MarkStylesInUse 247
TRichView 247
Mathematical expressions 143
MaxLength 181
TRichView 181
MaxPrintedItemNo 550
TCustomRVPrint 550
MaxTextWidth 181
TRichView 181
Menu 404
TRVSmartPopupProperties 404
MergeCells 594
TRVTableItemInfo 594
MergeSelectedCells 594
TRVTableItemInfo 594
MetafileCompatibility 551
TCustomRVPrint 551
MetafilePixelsPerInch 551
TCustomRVPrint 551
MinPrintedItemNo 551
TCustomRVPrint 551
MinTextWidth 181
TRichView 181
MinWidth 618
TRVLabelItemInfo 618
MirrorMargins 313, 521
TRVDocParameters 313
TRVPrint 521
Modified 327
TRichViewEdit 327
ModifiedProperties 480, 491
TFontInfo 480
TParaInfo 491
Modifying TRichView items 85

- N -

Name 496, 539
TRVOfficeConverterInfo 539
TRVStyleTemplate 496
Next 496, 563
TCustomRVPrintPreview 563
TRVStyleTemplate 496
NextId 496
TRVStyleTemplate 496
NextParaNo 492
TParaInfo 492
NextStyleNo 480
TFontInfo 480
NoMetafiles 551
TCustomRVPrint 551
NormalStyleTemplate 461
TRVStyleTemplateCollection 461
NoteText 182, 628
TCustomRVNoteItemInfo 628
TRichView 182
NoVScroll 546
TRVScroller 546
Numbered sequences 134
NumberType 625, 646
TCustomRVPageNumberItemInfo 646
TRVSeqItemInfo 625

- O -

Obtaining items of TRichView 84
OddColumnsColor 583
TRVTableItemInfo 583
OddRowsColor 583
TRVTableItemInfo 583
OffsetInCurItem 328
TRichViewEdit 328
OnAddStyle 274
TRichView 274
OnAfterApplyStyle 443
TRVStyle 443
OnAfterOleDrop 388
TRichViewEdit 388
OnApplyStyle 443

OnApplyStyle 443
TRVStyle 443
OnApplyStyleColor 444
TRVStyle 444
OnAssignImageFileName 274
TRichView 274
OnBeforeOleDrop 388
TRichViewEdit 388
OnCaretGetOut 389
TRichViewEdit 389
OnCaretMove 390
TRichViewEdit 390
OnCellEditing 602
TRVTableItemInfo 602
OnChange 390
TRichViewEdit 390
OnChanging 390
TRichViewEdit 390
OnCheckpointVisible 274
TRichView 274
OnCheckStickingItems 391
TRichViewEdit 391
OnCMHintShow 275
TRichView 275
OnControlAction 275
TRichView 275
OnConverting 539
TRVOfficeConverter 539
OnCopy 276
TRichView 276
OnCurParaStyleChanged 391
TRichViewEdit 391
OnCurTextStyleChanged 391
TRichViewEdit 391
OnDrawBackground 603
TRVTableItemInfo 603
OnDrawBorder 603
TRVTableItemInfo 603
OnDrawCheckpoint 444, 556
TCustomRVPrint 556
TRVStyle 444
OnDrawCustomCaret 391
TRichViewEdit 391
OnDrawHyperlink 556
TCustomRVPrint 556
OnDrawPageBreak 446
TRVStyle 446
OnDrawParaBack 447
TRVStyle 447
OnDrawStyleText 448
TRVStyle 448
OnDrawTextBack 449
TRVStyle 449
OnDropFile 392
TRichViewEdit 392
OnDropFiles 393
TRichViewEdit 393
OneLevelPreview 493
TRVListInfo 493
OnFormatting 529
TRVPrint 529
OnGetItemCursor 277
TRichView 277
OnGetPageNumber 543
TRVReportHelper 543
OnHScrolled 548
TRVScroller 548
OnHTMLSaveImage 277
TRichView 277
OnImportPicture 279
TRichView 279
OnItemAction 280
TRichView 280
OnItemHint 281
TRichView 281
OnItemResize 393
TRichViewEdit 393
OnItemTextEdit 394
TRichViewEdit 394
OnJump 282
TRichView 282
OnLoadCustomFormat 410, 415
TDBRichView 410
TDBRichViewEdit 415
OnLoadDocument 410, 415
TDBRichView 410
TDBRichViewEdit 415
OnMeasureCustomCaret 394
TRichViewEdit 394
OnNewDocument 410, 416
TDBRichView 410
TDBRichViewEdit 416
OnOleDragEnter 395
TRichViewEdit 395
OnOleDragLeave 395
TRichViewEdit 395
OnOleDragOver 396
TRichViewEdit 396
OnOleDrop 397
TRichViewEdit 397

OnPagePostpaint 530
TRVPrint 530
OnPagePrepaint 530
TRVPrint 530
OnPaint 283
TRichView 283
OnParaStyleConversion 397
TRichViewEdit 397
OnPaste 399
TRichViewEdit 399
OnPrintComponent 554
TCustomRVPrint 554
OnProgress 283
TRichView 283
OnReadHyperlink 284
TCustomRichView 284
OnReadMergeField 286
TCustomRichView 286
OnRVDblClick 287
TRichView 287
OnRVFControlNeeded 288
TRichView 288
OnRVFImageListNeeded 288
TRichView 288
OnRVFPictureNeeded 289
TRichView 289
OnRVMouseDown 289
TRichView 289
OnRVMouseMove 290
TRichView 290
OnRVMouseUp 291
TRichView 291
OnRVRightClick 291
TRichView 291
OnSaveComponentToFile 292
TRichView 292
OnSaveCustomFormat 416
TDBRichViewEdit 416
OnSaveDocXExtra 293
TRichView 293
OnSaveHTMLExtra 295
TRichView 295
OnSaveImage2 296
TRichView 296
OnSaveItemToFile 297
TRichView 297
OnSaveParaToHTML 298
TRichView 298
OnSaveRTFExtra 299
TRichView 299
OnSelect 301
TRichView 301
OnSendingToPrinter 531
TRVPrint 531
OnSmartPopupClick 400
TRichViewEdit 400
OnSpellingCheck 301
TRichView 301
OnStyleConversion 400
TRichViewEdit 400
OnStyleHoverSensitive 450
TRVStyle 450
OnStyleTemplatesChange 302
TRichView 302
OnVScrolled 549
TRVScroller 549
OnWriteHyperlink 302
TRichView 302
OnZoomChanged 563
TCustomRVPrintPreview 563
Opacity 501, 577, 609
TRVBackgroundRect 501
TRVTableCellData 609
TRVTableItemInfo 577
Options 182, 472, 487, 512, 577, 609
TCustomRVFontInfo 472
TCustomRVParaInfo 487
TRichView 182
TRVListLevel 512
TRVTableCellData 609
TRVTableItemInfo 577
Orientation 313
TRVDocParameters 313
OutlineLevel 488
TCustomRVParaInfo 488
Overview 63
Overview of item types 118
Overview of tables 144

- P -

Page counts 142
Page numbers 141
PageBorderColor 559
TCustomRVPrintPreview 559
PageBorderWidth 559
TCustomRVPrintPreview 559
PageBreakBefore 616
TRVTableRow 616
PageBreakColor 431

PageBreakColor 431
TRVStyle 431
PageBreaksBeforeItems 184
TRichView 184
PageHeight 313, 532
TRVDocParameters 313
TRVVirtualPrinterProperties 532
PageNo 559
TCustomRVPrintPreview 559
PageNoFromNumber 552
TCustomRVPrint 552
PagesCount 552
TCustomRVPrint 552
PageWidth 313, 532
TRVDocParameters 313
TRVVirtualPrinterProperties 532
Paragraphs in TRichView 72
ParaStyle 497
TRVStyleTemplate 497
ParaStyleMode 308
TRVRTFReaderProperties 308
ParaStyleNo 308
TRVRTFReaderProperties 308
ParaStyles 431
TRVStyle 431
ParaStyleTemplateId 473
TCustomRVFontInfo 473
Parent 497
TRVStyleTemplate 497
ParentId 497
TRVStyleTemplate 497
Paste 361
TRichViewEdit 361
PasteBitmap 362
TRichViewEdit 362
PasteGraphicFile 362
TRichViewEdit 362
PasteMetafile 363
TRichViewEdit 363
PasteRTF 363
TRichViewEdit 363
PasteRVF 364
TRichViewEdit 364
PasteText 364
TRichViewEdit 364
PasteTextA 364
TRichViewEdit 364
PasteTextW 364
TRichViewEdit 364
PasteURL 365
TRichViewEdit 365
Path 539
TRVOfficeConverterInfo 539
Picture 513
TRVListLevel 513
Pictures 122
PixelsPerInch 452
TFontInfos 452
PixelsToUnits 439
TRVStyle 439
Position 404, 505
TRVSmartPopupProperties 404
TRVTabInfo 505
PositionInText 640
TRVBoxPosition 640
Prev 563
TCustomRVPrintPreview 563
Preview100PercentHeight 521
TRVPrint 521
Preview100PercentWidth 521
TRVPrint 521
PreviewCorrection 552
TCustomRVPrint 552
PreviewMode 535
TRVOfficeConverter 535
Print 526
TRVPrint 526
PrintableAreaPen 560
TCustomRVPrintPreview 560
PrintOptions 579
TRVTableItemInfo 579
PrintPages 527
TRVPrint 527
Protection 473
TCustomRVFontInfo 473
ProtectTextStyleNo 619
TRVLabelItemInfo 619

- Q -

QuickAccess 497
TRVStyleTemplate 497

- R -

ReadDocParameters 308
TRVRTFReaderProperties 308
ReadOnly 328, 414
TDBRichViewEdit 414
TRichViewEdit 328

Redo 365
TRichViewEdit 365
RedoAction 366
TRichViewEdit 366
RedoName 366
TRichViewEdit 366
References to parent endnotes and footnotes 140
Reformat 248
TRichView 248
RefreshListMarkers 248
TRichView 248
RefreshSequences 248
TRichView 248
RelativeToCell 640
TRVBoxPosition 640
RemoveCheckpoint 248
TRichView 248
RemoveCheckpointAtCaret 366
TRichViewEdit 366
RemoveCheckpointEd 366
TRichViewEdit 366
RemoveCurrentCheckpoint 367
TRichViewEdit 367
RemoveCurrentPageBreak 367
TRichViewEdit 367
RemoveInternalLeading 619
TRVLabelItemInfo 619
RemoveLists 367
TRichViewEdit 367
RemoveSpaceBelow 619
TRVLabelItemInfo 619
ReplaceDocumentEd 628
TCustomRVNoteItemInfo 628
Reset 315, 626, 659
TRVDocParameters 315
TRVReportHelperWithHeaderFooters 659
TRVSeqItemInfo 626
ResetAnimation 249
TRichView 249
ResetLayout 315
TRVDocParameters 315
ResetNameCounter 463
TRVStyleTemplateCollection 463
ResizeControl 368
TRichViewEdit 368
ResizeCurrentControl 368
TRichViewEdit 368
RichView 27, 541
TRVReportHelper 541
RichView Format (RVF) overview 93
RichViewAllowCopyTableCells 700
RichViewAlternativePicPrint 700
RichViewApostropheInWord 700
RichViewAutoAssignNormalStyleTemplate 700
RichViewCompareStyleNames 701
RichViewDoNotCheckRVFStyleRefs 701
RichViewDoNotMergeNumbering 701
RichViewEditCaretHeightExtra 699
RichViewEditCaretPosition 699
RichViewEditCaretWidth 699
RichViewEditCaretWidthMode 699
RichViewEditCustomCaretSize 699
RichViewEditDefaultProportionalResize 702
RichViewEditEnterAllowsEmptyMarkeredLines 702
RichViewEditMaxCaretHeight 699
RichViewEditMinCaretHeight 699
RichViewJustifyBeforeLineBreak 702
RichViewMaxPictureCount 702
RichViewMaxThumbnailCount 703
RichViewPixelsPerInch 532, 703
RichViewResetStandardFlag 703
RichViewSaveDivInHTMLEx 698
RichViewSaveHyperlinksInDocXAsFields 704
RichViewSavePageBreaksInText 698
RichViewSavePInHTML 698
RichViewShowGhostSpaces 704
RichViewStringFormatMethod 704
RichViewTableAutoAddRow 705
RichViewTableDefaultRTFAutofit 705
RichViewToLinear 666
RichViewUnicodeInput 705
RightIndent 488
TCustomRVParaInfo 488
RightMargin 185, 313
TRichView 185
TRVDocParameters 313
Rotation 611
TRVTableCellData 611
RowBandSize 575
TRVTableItemInfo 575
RowCount 580
TRVTableItemInfo 580
Rows 580
TRVTableItemInfo 580
RowSpan 612
TRVTableCellData 612
RTFOptions 185
TRichView 185
RTFormatName 698

RTFReadProperties 187
TRichView 187
Rules of tables 148
RV_AfterImportGraphic 720
RV_CreateGraphics 720
RV_GetPrinterDC 661
RV_RegisterHTMLGraphicFormat 720
RV_RegisterPngGraphic 720
RV_TestFileUnicode 664
RV_TestStreamUnicode 664
RV_TestStringUnicode 664
RV_TwipsToUnits 667
RV_UnitsPerInch 667
RV_UnitsToPixels 667
RV_UnitsToTwips 667
RV_UnitsToUnits 667
RVAllFontInfoProperties 679
RVAllParaInfoProperties 684
RVApplyMathProperties 667
RVCharsPerLineBreak 666
RVData 187
TRichView 187
rvdoALL 671
rveipcAlignment 706, 709
rveipcBreakColor 706
rveipcBreakStyle 706
rveipcBreakWidth 706
rveipcCursor 706, 709
rveipcDisplayInline 706
rveipcFontSizeDouble 706
rveipcHotImageIndex 706
rveipcMinWidth 706, 709
rveipcPercentWidth 706
rveipcProtectTextStyleNo 706, 709
rveipcRemoveInternalLeading 706, 709
rveipcSeqNumberType 706, 709
rveipcSeqReset 706, 709
rveipcSeqStartFrom 706, 709
rveipcTextColor 706
RVEMPTYTAG 714
RVF 93, 109, 176
RVF Specification 109
RVFOptions 188
TRichView 188
RVFormatName 698
RVFParaStylesReadMode 190
TRichView 190
RVFTextStylesReadMode 191
TRichView 191
RVFWarnings 191
TRichView 191
RVGetFirstEndnote 662
RVGetFirstFootnote 662
RVGetFirstSidenote 662
RVGetLinearCaretPos 666
RVGetMathProperties 667
RVGetNextEndnote 662
RVGetNextFootnote 662
RVGetNextSidenote 662
RVGetNoteTextStyleNo 663
RVGetSelection 666
RVGetSelectionEx 666
RVGetTextLength 666
RVGetTextRange 666
RVGraphicHandler 710
RVIsCustomURL procedural 710
RVIsEmail functions 663
RVIsURL functions 663
RVNonTextCharacter 666
RVParagraphMarks 711
RVPrint 418
TRVPrintPreview 418
rvsBreak 711
rvsBullet 711
rvsComponent 711
rvsDefStyle 711
rvsEndnote 711
RVSetLinearCaretPos 666
RVSetMathProperties 667
RVSetSelection 666
RVSetSelectionEx 666
rvsFootnote 711
rvsHeading 711
rvsHotPicture 711
rvsHotspot 711
rvsJump1 711
rvsJump2 711
rvsKeyword 711
rvsLabel 711
rvsListMarker 711
rvsMathEquation 711
rvsNormal 711
rvsNoteReference 711
rvsPicture 711
rvsSequence 711
rvsSubheading 711
rvsTab 711
rvsTable 711

RVStyle.TRVColorMode 671
RVThumbnailMaker 713
RVU_AnsiToUnicode 664
RVU_AnsiToUTF8 664
RVU_UnicodeToAnsi 664
RVUnitsToUnits 439
  TRVStyle 439
RVVisibleSpecialCharacters 714

- S -

SaveCSS 441
  TRVStyle 441
SaveCSSToStream 442
  TRVStyle 442
SaveDocX 249
  TRichView 249
SaveDocXToStream 250
  TRichView 250
SaveHTML 250
  TRichView 250
SaveHTMLEx 251
  TRichView 251
SaveHTMLToStream 252
  TRichView 252
SaveHTMLToStreamEx 252
  TRichView 252
SaveINI 442
  TRVStyle 442
SavePicture 253
  TRichView 253
SaveReg 442
  TRVStyle 442
SaveRowsToStream 595
  TRVTableItemInfo 595
SaveRTF 253
  TRichView 253
SaveRTFToStream 254
  TRichView 254
SaveRVF 254
  TRichView 254
SaveRVFToStream 254
  TRichView 254
SaveText 255
  TRichView 255
SaveTextToStream 255
  TRichView 255
SaveTextToStreamW 255
  TRichView 255
SaveTextW 255
  TRichView 255
SaveToRVST 462
  TRVStyleTemplateCollection 462
SaveToStream 595
  TRVTableItemInfo 595
Scrolling in TRichView 80
ScrollTo 548
  TRVScroller 548
Searching and replacing in TRichView and TRichViewEdit 83
SearchText 256, 369
  TRichView 256
  TRichViewEdit 369
SearchTextA 256, 369
  TRichView 256
  TRichViewEdit 369
SearchTextW 256, 369
  TRichView 256
  TRichViewEdit 369
SelColor 426
  TRVStyle 426
Select 595
  TRVTableItemInfo 595
SelectAll 257
  TRichView 257
SelectCols 595
  TRVTableItemInfo 595
SelectControl 257
  TRichView 257
SelectCurrentLine 369
  TRichViewEdit 369
SelectCurrentWord 370
  TRichViewEdit 370
Selecting parts of TRichView document 81
Selection in Table 150
SelectionExists 258
  TRichView 258
SelectionHandleKind 431
  TRVStyle 431
SelectionHandlesVisible 192
  TRichView 192
SelectionMode 432
  TRVStyle 432
SelectionStyle 432
  TRVStyle 432
SelectRows 596
  TRVTableItemInfo 596
SelectWordAt 258
  TRichView 258
SelTextColor 426

SelTextColor 426
TRVStyle 426
Semitransparent Objects in TRichView 91
SeqName 626
TRVSeqItemInfo 626
SetAddParagraphMode 259
TRichView 259
SetBackgroundImageEd 371
TRichViewEdit 371
SetBreakInfo 260
TRichView 260
SetBreakInfoEd 371
TRichViewEdit 371
SetBulletInfo 261
TRichView 261
SetBulletInfoEd 372
TRichViewEdit 372
SetButtonState 405
TRVSmartPopupProperties 405
SetCellBackgroundImage 596
TRVTableItemInfo 596
SetCellBackgroundImageFileName 596
TRVTableItemInfo 596
SetCellBackgroundStyle 597
TRVTableItemInfo 597
SetCellBestHeight 597
TRVTableItemInfo 597
SetCellBestWidth 597
TRVTableItemInfo 597
SetCellBorderColor 597
TRVTableItemInfo 597
SetCellBorderLightColor 598
TRVTableItemInfo 598
SetCellColor 598
TRVTableItemInfo 598
SetCellHint 598
TRVTableItemInfo 598
SetCellOpacity 598
TRVTableItemInfo 598
SetCellOptions 599
TRVTableItemInfo 599
SetCellRotation 599
TRVTableItemInfo 599
SetCellTag 599
TRVTableItemInfo 599
SetCellVAlign 599
TRVTableItemInfo 599
SetCellVisibleBorders 600
TRVTableItemInfo 600
SetCheckpointInfo 261
TRichView 261
SetCheckpointInfoEd 373
TRichViewEdit 373
SetControlInfo 262
TRichView 262
SetControlInfoEd 374
TRichViewEdit 374
SetCurrentBreakInfo 375
TRichViewEdit 375
SetCurrentBulletInfo 375
TRichViewEdit 375
SetCurrentCheckpointInfo 376
TRichViewEdit 376
SetCurrentControlInfo 377
TRichViewEdit 377
SetCurrentHotspotInfo 377
TRichViewEdit 377
SetCurrentItemExtraIntProperty 378
TRichViewEdit 378
SetCurrentItemExtraIntPropertyEx 378
TRichViewEdit 378
SetCurrentItemExtraStrProperty 379
TRichViewEdit 379
SetCurrentItemExtraStrPropertyEx 379
TRichViewEdit 379
SetCurrentItemText 380
TRichViewEdit 380
SetCurrentItemTextA 380
TRichViewEdit 380
SetCurrentItemTextW 380
TRichViewEdit 380
SetCurrentItemVAlign 380
TRichViewEdit 380
SetCurrentPictureInfo 381
TRichViewEdit 381
SetCurrentTag 382
TRichViewEdit 382
SetFloatPropertyEd 370
TRichViewEdit 370
SetFooter 263, 528
TRichView 263
TRVPrint 528
SetHeader 264, 529
TRichView 264
TRVPrint 529
SetHotspotInfo 265
TRichView 265
SetHotspotInfoEd 382
TRichViewEdit 382
SetIntPropertyEd 370

SetIntPropertyEd 370
TRichViewEdit 370
SetItemExtraIntProperty 266
TRichView 266
SetItemExtraIntPropertyEd 383
TRichViewEdit 383
SetItemExtraIntPropertyEx 266
TRichView 266
SetItemExtraIntPropertyExEd 383
TRichViewEdit 383
SetItemExtraStrProperty 266
TRichView 266
SetItemExtraStrPropertyEd 384
TRichViewEdit 384
SetItemExtraStrPropertyEx 266
TRichView 266
SetItemExtraStrPropertyExEd 384
TRichViewEdit 384
SetItemTag 267
TRichView 267
SetItemTagEd 384
TRichViewEdit 384
SetItemText 267
TRichView 267
SetItemTextA 267
TRichView 267
SetItemTextEd 385
TRichViewEdit 385
SetItemTextEdA 385
TRichViewEdit 385
SetItemTextEdW 385
TRichViewEdit 385
SetItemTextW 267
TRichView 267
SetItemVAlign 268
TRichView 268
SetItemVAlignEd 386
TRichViewEdit 386
SetListMarkerInfo 269
TRichView 269
SetPictureInfo 269
TRichView 269
SetPictureInfoEd 386
TRichViewEdit 386
SetRowKeepTogether 600
TRVTableItemInfo 600
SetRowPageBreakBefore 600
TRVTableItemInfo 600
SetRowVAlign 600
TRVTableItemInfo 600
SetSelectionBounds 271
TRichView 271
SetStrPropertyEd 370
TRichViewEdit 370
SetTableVisibleBorders 600
TRVTableItemInfo 600
SetUndoGroupMode 387
TRichViewEdit 387
SetZoom 563
TCustomRVPrintPreview 563
ShadowColor 560
TCustomRVPrintPreview 560
ShadowWidth 560
TCustomRVPrintPreview 560
ShortCut 404
TRVSmartPopupProperties 404
SidenoteNumbering 422
TRVStyle 422
Sidenotes 137
SingleClick 192
TRichView 192
SingleSymbols 473
TCustomRVFontInfo 473
Size 475
TCustomRVFontInfo 475
SizeDouble 475
TCustomRVFontInfo 475
SkipHiddenText 309
TRVRTFReaderProperties 309
Smart Popups 402
SmartPopupProperties 329
TRichViewEdit 329
SmartPopupVisible 329
TRichViewEdit 329
Smooth image scaling in TRichView 91
SoftPageBreakColor 434
TRVStyle 434
SpaceAfter 489
TCustomRVParaInfo 489
SpaceBefore 489
TCustomRVParaInfo 489
SpacesInTab 435
TRVStyle 435
SpecialCharactersColor 435
TRVStyle 435
SplitSelectedCellsHorizontally 600
TRVTableItemInfo 600
SplitSelectedCellsVertically 601
TRVTableItemInfo 601
Standard 464

Standard 464
TCustomRVInfo 464
StartAnimation 272
TRichView 272
StartAt 552
TCustomRVPrint 552
StartFrom 513, 626
TRVListLevel 513
TRVSeqItemInfo 626
StartLiveSpelling 273
TRichView 273
StopAnimation 273
TRichView 273
Stream 536
TRVOfficeConverter 536
Style 193, 475, 502
TCustomRVFontInfo 475
TRichView 193
TRVBorder 502
style templates 491
StyleEx 475
TCustomRVFontInfo 475
StyleName 465
TCustomRVInfo 465
Styles and Style-Templates in TRichView 74
StyleTemplateId 465
TCustomRVFontOrParaInfo 465
StyleTemplateInsertMode 193
TRichView 193
StyleTemplates 435
TRVStyle 435
SubSuperScriptType 476
TCustomRVFontInfo 476

- T -

Table Cells 146
Table Operations 153
Table Resizing 158
Tables 130
Tables and Compatibility 145
Tables in RichView 145
Tables in TRichView 144
TabNavigation 194
TRichView 194
Tabs 489
TCustomRVParaInfo 489
Tabulators 121
Tag 612
TRVTableCellData 612
Tags of TRichView items 68
TBackgroundStyle 170
TCheckpointData 667
TConvertingEvent 539
TCPEventKind 173
TCustomRichView 162, 289
TCustomRichView.AssignSoftPageBreaks 209
TCustomRichView.GetItemAt 225
TCustomRichView.GetItemExtraStrProperty 228
TCustomRichView.GetItemExtraStrPropertyEx 228
TCustomRichView.OnReadHyperlink 284
TCustomRichView.OnReadMergeField 286
TCustomRichViewEdit 316, 667
TCustomRichViewEdit.ActualCurTextStyleNo 322
TCustomRVControl 544
TCustomRVData 650
TCustomRVData.Edit 651
TCustomRVData.GetRVData 652
TCustomRVData.GetSourceRVData 652
TCustomRVFieldItemInfo 646
TCustomRVFontInfo 466
TCustomRVFontInfo.Assign 477
TCustomRVFontInfo.AssignTo 477
TCustomRVFontInfo.BackColor 467
TCustomRVFontInfo.BiDiMode 467
TCustomRVFontInfo.CharScale 467
TCustomRVFontInfo.Charset 468
TCustomRVFontInfo.CharSpacing 468
TCustomRVFontInfo.Color 469
TCustomRVFontInfo.Create 478
TCustomRVFontInfo.EmptyWidth 469
TCustomRVFontInfo.FontName 470
TCustomRVFontInfo.HoverBackColor 470
TCustomRVFontInfo.HoverColor 470
TCustomRVFontInfo.HoverEffects 471
TCustomRVFontInfo.HoverUnderlineColor 471
TCustomRVFontInfo.IsEqual 478
TCustomRVFontInfo.JumpCursor 472
TCustomRVFontInfo.Options 472
TCustomRVFontInfo.ParaStyleTemplateId 473
TCustomRVFontInfo.Protection 473
TCustomRVFontInfo.SingleSymbols 473
TCustomRVFontInfo.Size 475
TCustomRVFontInfo.SizeDouble 475
TCustomRVFontInfo.Style 475
TCustomRVFontInfo.StyleEx 475
TCustomRVFontInfo.SubSuperScriptType 476
TCustomRVFontInfo.UnderlineColor 476
TCustomRVFontInfo.UnderlineType 476

TCustomRVFontInfo.VShift 477
TCustomRVFontOrParaInfo.StyleTemplateId 465
TCustomRVFormattedData 650
TCustomRVFormattedData.GetOriginEx 653
TCustomRVInfo 463
TCustomRVInfo.Standard 464
TCustomRVInfo.StyleName 465
TCustomRVIsURLFunction 710
TCustomRVItemInfo 563
TCustomRVNoteItemInfo 563, 627
TCustomRVNoteItemInfo.CreateEx 628
TCustomRVNoteItemInfo.Document 628
TCustomRVNoteItemInfo.NoteText 628
TCustomRVNoteItemInfo.ReplaceDocumentEd 628
TCustomRVPageNumberItemInfo 646
TCustomRVPageNumberItemInfo.NumberType 646
TCustomRVParaInfo 482
TCustomRVParaInfo.Alignment 483
TCustomRVParaInfo.Assign 489
TCustomRVParaInfo.Background 484
TCustomRVParaInfo.BiDiMode 484
TCustomRVParaInfo.Border 485
TCustomRVParaInfo.Create 489
TCustomRVParaInfo.FirstIndent 485
TCustomRVParaInfo.IsEqual 490
TCustomRVParaInfo.LastLineAlignment 483
TCustomRVParaInfo.LeftIndent 485
TCustomRVParaInfo.LineSpacing 486
TCustomRVParaInfo.LineSpacingType 486
TCustomRVParaInfo.Options 487
TCustomRVParaInfo.OutlineLevel 488
TCustomRVParaInfo.RightIndent 488
TCustomRVParaInfo.SpaceAfter 489
TCustomRVParaInfo.SpaceBefore 489
TCustomRVParaInfo.Tabs 489
TCustomRVPrint 549
TCustomRVPrint.Clear 553
TCustomRVPrint.ColorMode 549
TCustomRVPrint.EndAt 549
TCustomRVPrint.FirstPageNo 550
TCustomRVPrint.GetFirstItemOnPage 553
TCustomRVPrint.GetPageNo 553
TCustomRVPrint.IgnorePageBreaks 550
TCustomRVPrint.LastPageNo 550
TCustomRVPrint.MaxPrintedItemNo 550
TCustomRVPrint.MetafileCompatibility 551
TCustomRVPrint.MetafilePixelsPerInch 551
TCustomRVPrint.MinPrintedItemNo 551
TCustomRVPrint.NoMetafiles 551
TCustomRVPrint.OnDrawCheckpoint 556
TCustomRVPrint.OnDrawHyperlink 556
TCustomRVPrint.OnPrintComponent 554
TCustomRVPrint.PageNoFromNumber 552
TCustomRVPrint.PagesCount 552
TCustomRVPrint.PreviewCorrection 552
TCustomRVPrint.StartAt 552
TCustomRVPrint.TransparentBackground 553
TCustomRVPrint.UpdatePaletteInfo 554
TCustomRVPrintPreview 556
TCustomRVPrintPreview.ClickMode 558
TCustomRVPrintPreview.Color 558
TCustomRVPrintPreview.Create 562
TCustomRVPrintPreview.First 562
TCustomRVPrintPreview.Last 562
TCustomRVPrintPreview.MarginsPen 559
TCustomRVPrintPreview.Next 563
TCustomRVPrintPreview.OnZoomChanged 563
TCustomRVPrintPreview.PageBorderColor 559
TCustomRVPrintPreview.PageBorderWidth 559
TCustomRVPrintPreview.PageNo 559
TCustomRVPrintPreview.Prev 563
TCustomRVPrintPreview.PrintableAreaPen 560
TCustomRVPrintPreview.SetZoom 563
TCustomRVPrintPreview.ShadowColor 560
TCustomRVPrintPreview.ShadowWidth 560
TCustomRVPrintPreview.ZoomInCursor 560
TCustomRVPrintPreview.ZoomMode 561
TCustomRVPrintPreview.ZoomOutCursor 561
TCustomRVPrintPreview.ZoomPercent 562
TDBRichView 407
TDBRichView.AutoDisplay 408
TDBRichView.Create 409
TDBRichView.DataField 408
TDBRichView.DataSource 408
TDBRichView.Destroy 409
TDBRichView.Field 409
TDBRichView.LoadField 409
TDBRichView.OnLoadCustomFormat 410
TDBRichView.OnLoadDocument 410
TDBRichView.OnNewDocument 410
TDBRichViewEdit 410
Inserting controls what can modify themselves in DBRichViewEdit 417
Using viewer-style methods in DBRichViewEdit 416
TDBRichViewEdit.AutoDeleteUnusedStyles 412
TDBRichViewEdit.AutoDisplay 412
TDBRichViewEdit.Change 414

TDBRichViewEdit.Create 414
TDBRichViewEdit.DataField 413
TDBRichViewEdit.DataSource 413
TDBRichViewEdit.Destroy 415
TDBRichViewEdit.Field 413
TDBRichViewEdit.FieldFormat 413
TDBRichViewEdit.IgnoreEscape 414
TDBRichViewEdit.LoadField 415
TDBRichViewEdit.OnLoadCustomFormat 415
TDBRichViewEdit.OnLoadDocument 415
TDBRichViewEdit.OnNewDocument 416
TDBRichViewEdit.OnSaveCustomFormat 416
TDBRichViewEdit.ReadOnly 414
Text 619, 622
  TRVLabelItemInfo 619
  TRVMathItemInfo 622
Text boxes 139
Text Items 120
TextBackgroundKind 435
  TRVStyle 435
TextColor 623
  TRVMathItemInfo 623
TextColSeparator 581
  TRVTableItemInfo 581
TextEngine 436
  TRVStyle 436
TextRowSeparator 581
  TRVTableItemInfo 581
TextStyle 497
  TRVStyleTemplate 497
TextStyleMode 309
  TRVRTFReaderProperties 309
TextStyleNo 309, 620
  TRVLabelItemInfo 620
  TRVRTFReaderProperties 309
TextStyles 437
  TRVStyle 437
TFontInfo 478
TFontInfo.Assign 480
TFontInfo.Create 481
TFontInfo.Draw 481
TFontInfo.IsEqual 482
TFontInfo.Jump 479
TFontInfo.ModifiedProperties 480
TFontInfo.NextStyleNo 480
TFontInfo.Unicode 480
TFontInfos 451
TFontInfos.Add 452
TFontInfos.AddFont 452
TFontInfos.AddFontEx 452
TFontInfos.AssignTo 453
TFontInfos.Create 453
TFontInfos.FindStyleWithCharset 453
TFontInfos.FindStyleWithColor 453
TFontInfos.FindStyleWithFont 454
TFontInfos.FindStyleWithFontName 454
TFontInfos.FindStyleWithFontSize 454
TFontInfos.FindStyleWithFontStyle 455
TFontInfos.FindSuchStyle 455
TFontInfos.FindSuchStyleEx 455
TFontInfos.InvalidItem 451
TFontInfos.Items 451
TFontInfos.PixelsPerInch 452
Title 314
  TRVDocParameters 314
TitlePage 314, 521
  TRVDocParameters 314
  TRVPrint 521
TJumpEvent 282
TopLevelEditor 329
  TRichViewEdit 329
TopMargin 194, 314
  TRichView 194
  TRVDocParameters 314
TParaInfo 490
TParaInfo.Assign 492
TParaInfo.Create 492
TParaInfo.DefStyleNo 491
TParaInfo.IsEqual 492
TParaInfo.ModifiedProperties 491
TParaInfo.NextParaNo 492
TParaInfos 456
TParaInfos.Add 457
TParaInfos.AssignTo 457
TParaInfos.Create 457
TParaInfos.FindStyleWithAlignment 457
TParaInfos.FindSuchStyle 458
TParaInfos.FindSuchStyleEx 458
TParaInfos.InvalidItem 456
TParaInfos.Items 456
Tracking 546
  TRVScroller 546
TransparentBackground 553
  TCustomRVPrint 553
TRichView 162
  adding items 77
  Compatibility with older versions of RichView 62
  Component Editor 168

TRichView 162
live spelling check 101
Loading UTF-8 files 316
modifying items 85
New after version 17 29
New in 17 30
New in v1.2 60
New in v1.3 60
New in v1.4 58
New in v1.5 56
New in v1.6 55
New in v1.7 52
New in v1.8 50
New in v1.9 48
New in v10 44
New in v11 42
New in v12 41
New in v13 39
New in v14 36
New in v15 34
New in v16 32
obtaining items 84
selecting parts of document 81
Version history 29
TRichView Clipboard functions overview 82
TRichView constants and variables (List) 696
CFRV_HTML 698
CFRV_RTF 698
CFRV_RVF 698
CFRV_URL 698
HTMLFormatName 698
RichViewAllowCopyTableCells 700
RichViewAlternativePicPrint 700
RichViewAutoAssignNormalStyleTemplate 700
RichViewCompareStyleNames 701
RichViewDoNotCheckRVFStyleRefs 701
RichViewDoNotMergeNumbering 701
RichViewEditDefaultProportionalResize 702
RichViewEditEnterAllowsEmptyMarkeredLines 702
RichViewJustifyBeforeLineBreak 702
RichViewMaxPictureCount 702
RichViewMaxThumbnailCount 703
RichViewPixelsPerInch 703
RichViewResetStandardFlag 703
RichViewSaveDivInHTMLEx 698
RichViewSaveHyperlinksInDocXAsFields 704
RichViewSavePageBreaksInText 698
RichViewSavePInHTML 698
RichViewShowGhostSpaces 704
RichViewStringFormatMethod 704
RichViewTableAutoAddRow 705
RichViewTableDefaultRTFAutofit 705
RichViewUnicodeInput 705
RTFormatName 698
rveipcAlignment 706, 709
rveipcBreakColor 706
rveipcBreakStyle 706
rveipcBreakWidth 706
rveipcCursor 706, 709
rveipcHotImageIndex 706
rveipcImageIndex 706
rveipcMinWidth 706, 709
rveipcPercentWidth 706
rveipcProtectTextStyleNo 706, 709
rveipcRemoveInternalLeading 706, 709
rveipcSeqNumberType 706, 709
rveipcSeqReset 706, 709
rveipcSeqStartFrom 706, 709
RVFormatName 698
RVIsCustomURL 710
RVParagraphMarks 711
rvsBreak 711
rvsBullet 711
rvsComponent 711
rvsDefStyle 711
rvsEndnote 711
rvsFootnote 711
rvsHeading 711
rvsHotPicture 711
rvsHotspot 711
rvsJump1 711
rvsJump2 711
rvsKeyword 711
rvsLabel 711
rvsListMarker 711
rvsNormal 711
rvsNoteReference 711
rvsPicture 711
rvsSequence 711
rvsSubheading 711
rvsTab 711
rvsTable 711
RVVisibleSpecialCharacters 714
URLFormatName 698
TRichView hypertext 69
TRichView item types 64, 118, 563
TRichView items' "checkpoints" overview 65
TRichView items' "tags" overview 68
TRichView methods for saving and loading 92
TRichView overview 27, 63

TRichView paragraphs 72
TRichView procedures and functions
DrawButton 660
DrawControl 660
DrawEdit 660
DrawMemo 660
DrawPanel 660
GetRVESearchOptions 660
GetRVSearchOptions 661
RV_GetPrinterDC 661
RV_TestFileUnicode 664
RV_TestStreamUnicode 664
RV_TestStringUnicode 664
RVGetFirstEndnote 662
RVGetFirstFootnote 662
RVGetFirstSidenote 662
RVGetNextEndnote 662
RVGetNextFootnote 662
RVGetNextSidenote 662
RVGetNoteTextStyleNo 663
RVIsEmail 663
RVIsURL 663
RVU_AnsiToUnicode 664
RVU_AnsiToUTF8 664
RVU_GetRawUnicode 664
RVU_RawUnicodeToWideString 664
RVU_UnicodeToAnsi 664
TRichView scrolling 80
TRichView.Add 203
TRichView.AddBreak 196
TRichView.AddBreakEx 196
TRichView.AddBreakExTag 196
TRichView.AddBreakTag 196
TRichView.AddBullet 197
TRichView.AddBulletEx 197
TRichView.AddBulletExTag 197
TRichView.AddCheckpoint 202
TRichView.AddCheckpointTag 202
TRichView.AddControl 198
TRichView.AddControlEx 198
TRichView.AddControlExTag 198
TRichView.AddFmt 198
TRichView.AddHotPicture 199
TRichView.AddHotPictureTag 199
TRichView.AddHotspot 200
TRichView.AddHotspotEx 200
TRichView.AddHotspotExTag 200
TRichView.AddItem 201
TRichView.AddNamedCheckpoint 202
TRichView.AddNamedCheckpointEx 202
TRichView.AddNamedCheckpointExTag 202
TRichView.AddNL 203
TRichView.AddNLATag 203
TRichView.AddNLTag 203
TRichView.AddNLWTag 203
TRichView.AddPicture 204
TRichView.AddPictureEx 204
TRichView.AddPictureExTag 204
TRichView.AddTab 205
TRichView.AddTag 203
TRichView.AddTextNL 206
TRichView.AddTextNLA 206
TRichView.AddTextNLW 206
TRichView.AllowSelection 169
TRichView.AnimationMode 169
TRichView.AppendFrom 207
TRichView.AppendRVFFromStream 208
TRichView.BackgroundBitmap 169
TRichView.BackgroundStyle 170
TRichView.BeginOleDrag 209
TRichView.BiDiMode 170
TRichView.BottomMargin 171
TRichView.Clear 210
TRichView.ClearLeft 171
TRichView.ClearLiveSpellingResults 210
TRichView.ClearRight 172
TRichView.ClearSoftPageBreaks 211
TRichView.ClientToDocument 211
TRichView.Color 173
TRichView.ConvertDocToDifferentUnits 211
TRichView.ConvertDocToPixels 211
TRichView.ConvertDocToTwips 211
TRichView.Copy 212
TRichView.CopyDef 212
TRichView.CopyImage 212
TRichView.CopyRTF 213
TRichView.CopyRVF 213
TRichView.CopyText 213
TRichView.CopyTextA 213
TRichView.CopyTextW 213
TRichView.CPEventKind 173
TRichView.Create 214
TRichView.Cursor 174
TRichView.DeleteItems 214
TRichView.DeleteMarkedStyles 215
TRichView.DeleteParas 215
TRichView.DeleteSection 215
TRichView.DeleteUnusedStyles 216
TRichView.Delimiters 174

TRichView.Deselect 216
TRichView.Destroy 217
TRichView.DocObjects 174
TRichView.DocParameters 176
TRichView.DocProperties 176
TRichView.DocumentHeight 177
TRichView.DocumentToClient 211
TRichView.DoInPaletteMode 177
TRichView.FindCheckpointByName 217
TRichView.FindCheckpointByTag 217
TRichView.FindControlItemNo 217
TRichView.FirstItemVisible 179
TRichView.FirstJumpNo 179
TRichView.Format 217
TRichView.FormatTail 218
TRichView.GetBreakInfo 218
TRichView.GetBulletInfo 219
TRichView.GetCheckpointByNo 220
TRichView.GetCheckpointInfo 220
TRichView.GetCheckpointItemNo 221
TRichView.GetCheckpointNo 221
TRichView.GetCheckpointXY 221
TRichView.GetCheckpointY 221
TRichView.GetCheckpointYEx 222
TRichView.GetControlInfo 222
TRichView.GetFirstCheckpoint 223
TRichView.GetFocusedItem 223
TRichView.GetHotspotInfo 224
TRichView.GetItem 225
TRichView.GetItemCheckpoint 226
TRichView.GetItemClientCoords 226
TRichView.GetItemCoords 226
TRichView.GetItemCoordsEx 227
TRichView.GetItemExtraIntProperty 227
TRichView.GetItemExtraIntPropertyEx 227
TRichView.GetItemNo 229
TRichView.GetItemPara 229
TRichView.GetItemStyle 229
TRichView.GetItemTag 230
TRichView.GetItemText 230
TRichView.GetItemTextA 230
TRichView.GetItemTextW 230
TRichView.GetItemVAlign 230
TRichView.GetJumpPointItemNo 231
TRichView.GetJumpPointLocation 231
TRichView.GetJumpPointY 232
TRichView.GetLastCheckpoint 232
TRichView.GetLineNo 233
TRichView.GetListMarkerInfo 233
TRichView.GetNextCheckpoint 234
TRichView.GetOffsAfterItem 235
TRichView.GetOffsBeforeItem 235
TRichView.GetPictureInfo 236
TRichView.GetPrevCheckpoint 237
TRichView.GetSelectedImage 237
TRichView.GetSelectionBounds 237
TRichView.GetSelText 238
TRichView.GetSelTextA 238
TRichView.GetSelTextW 238
TRichView.GetTextInfo 239
TRichView.GetWordAt 239
TRichView.GetWordAtA 239
TRichView.GetWordAtW 239
TRichView.InsertRVFFromStream 240
TRichView.IsFromNewLine 242
TRichView.IsParaStart 242
TRichView.ItemCount 180
TRichView.LastItemVisible 180
TRichView.LeftMargin 180
TRichView.LineCount 181
TRichView.LiveSpellingValidateWord 242
TRichView.LoadFromStream 243
TRichView.LoadRTF 243
TRichView.LoadRTFFromStream 244
TRichView.LoadRVF 245
TRichView.LoadRVFFromStream 245
TRichView.LoadText 246
TRichView.LoadTextFromStream 247
TRichView.LoadTextFromStreamW 247
TRichView.LoadTextW 246
TRichView.MarkStylesInUse 247
TRichView.MaxLength 181
TRichView.MaxTextWidth 181
TRichView.MinTextWidth 181
TRichView.NoteText 182
TRichView.OnAddStyle 274
TRichView.OnAssignImageFileName 274
TRichView.OnCheckpointVisible 274
TRichView.OnCMHintShow 275
TRichView.OnControlAction 275
TRichView.OnCopy 276
TRichView.OnGetItemCursor 277
TRichView.OnHTMLSaveImage 277
TRichView.OnImportPicture 279
TRichView.OnItemAction 280
TRichView.OnItemHint 281
TRichView.OnJump 282
TRichView.OnPaint 283

TRichView.OnProgress 283
TRichView.OnRVDblClick 287
TRichView.OnRVFControlNeeded 288
TRichView.OnRVFImageListNeeded 288
TRichView.OnRVFPictureNeeded 289
TRichView.OnRVMouseDown 289
TRichView.OnRVMouseMove 290
TRichView.OnRVMouseUp 291
TRichView.OnRVRightClick 291
TRichView.OnSaveComponentToFile 292
TRichView.OnSaveDocXExtra 293
TRichView.OnSaveHTMLExtra 295
TRichView.OnSaveImage2 296
TRichView.OnSaveItemToFile 297
TRichView.OnSaveParaToHTML 298
TRichView.OnSaveRTFExtra 299
TRichView.OnSelect 301
TRichView.OnSpellingCheck 301
TRichView.OnStyleTemplatesChange 302
TRichView.OnWriteHyperlink 302
TRichView.Options 182
TRichView.PageBreaksBeforeItems 184
TRichView.Reformat 248
TRichView.RefreshListMarkers 248
TRichView.RefreshSequences 248
TRichView.RemoveCheckpoint 248
TRichView.ResetAnimation 249
TRichView.RightMargin 185
TRichView.RTFOptions 185
TRichView.RTFReadProperties 187
TRichView.RVData 187
TRichView.RVFOptions 188
TRichView.RVFParaStylesReadMode 190
TRichView.RVFTextStylesReadMode 191
TRichView.RVFWarnings 191
TRichView.SaveDocX 249
TRichView.SaveDocXToStream 250
TRichView.SaveHTML 250
TRichView.SaveHTMLEx 251
TRichView.SaveHTMLToStream 252
TRichView.SaveHTMLToStreamEx 252
TRichView.SavePicture 253
TRichView.SaveRTF 253
TRichView.SaveRTFToStream 254
TRichView.SaveRVF 254
TRichView.SaveRVFToStream 254
TRichView.SaveText 255
TRichView.SaveTextToStream 255
TRichView.SaveTextToStreamW 255
TRichView.SaveTextW 255
TRichView.SearchText 256
TRichView.SearchTextA 256
TRichView.SearchTextW 256
TRichView.SelectAll 257
TRichView.SelectControl 257
TRichView.SelectionExists 258
TRichView.SelectionHandlesVisible 192
TRichView.SelectWordAt 258
TRichView.SetAddParagraphMode 259
TRichView.SetBreakInfo 260
TRichView.SetBulletInfo 261
TRichView.SetCheckpointInfo 261
TRichView.SetControlInfo 262
TRichView.SetFooter 263
TRichView.SetHeader 264
TRichView.SetHotspotInfo 265
TRichView.SetItemExtraIntProperty 266
TRichView.SetItemExtraIntPropertyEx 266
TRichView.SetItemExtraStrProperty 266
TRichView.SetItemExtraStrPropertyEx 266
TRichView.SetItemTag 267
TRichView.SetItemText 267
TRichView.SetItemTextA 267
TRichView.SetItemTextW 267
TRichView.SetItemVAlign 268
TRichView.SetListMarkerInfo 269
TRichView.SetPictureInfo 269
TRichView.SetSelectionBounds 271
TRichView.SingleClick 192
TRichView.StartAnimation 272
TRichView.StartLiveSpelling 273
TRichView.StopAnimation 273
TRichView.Style 193
TRichView.StyleTemplateInsertMode 193
TRichView.TabNavigation 194
TRichView.TopMargin 194
TRichView.UpdatePaletteInfo 273
TRichView.UseStyleTemplates 195
TRichView.UseVCLThemes 195
TRichView.VAlign 195
TRichView.VSmallStep 196
TRichView.WordWrap 196
TRichViewEdit 316, 361
  Inserting items in position of caret 86
  InsertTextFromFile 361
  InsertTextFromFileW 361
  Moving caret to the beginning of paragraph 405
  OnDropFiles example 405

TRichViewEdit Undo 88
TRichViewEdit.AcceptDragDropFormats 321
TRichViewEdit.AcceptPasteFormats 321
TRichViewEdit.AdjustControlPlacement 330
TRichViewEdit.AdjustControlPlacement2 330
TRichViewEdit.ApplyListStyle 331
TRichViewEdit.ApplyParaStyle 331
TRichViewEdit.ApplyParaStyleConversion 332
TRichViewEdit.ApplyParaStyleTemplate 332
TRichViewEdit.ApplyStyleConversion 333
TRichViewEdit.ApplyStyleTemplate 333
TRichViewEdit.ApplyTextStyle 334
TRichViewEdit.ApplyTextStyleTemplate 334
TRichViewEdit.BeginItemModify 335
TRichViewEdit.BeginUndoCustomGroup 336
TRichViewEdit.BeginUndoGroup 336
TRichViewEdit.BeginUpdate 336
TRichViewEdit.CanChange 336
TRichViewEdit.CanPaste 337
TRichViewEdit.CanPasteRTF 337
TRichViewEdit.CanPasteRVF 337
TRichViewEdit.Change 337
TRichViewEdit.ChangeListLevels 338
TRichViewEdit.ChangeStyleTemplates 338
TRichViewEdit.Clear 338
TRichViewEdit.ClearTextFlow 339
TRichViewEdit.ClearUndo 339
TRichViewEdit.ConvertToHotPicture 339
TRichViewEdit.ConvertToPicture 339
TRichViewEdit.Create 340
TRichViewEdit.CurItemNo 322
TRichViewEdit.CurItemStyle 323
TRichViewEdit.CurParaStyleNo 323
TRichViewEdit.CurTextStyleNo 324
TRichViewEdit.CustomCaretInterval 324
TRichViewEdit.CutDef 340
TRichViewEdit.DefaultPictureVAlign 325
TRichViewEdit.DeleteSelection 340
TRichViewEdit.Destroy 340
TRichViewEdit.EditorOptions 325
TRichViewEdit.EndItemModify 341
TRichViewEdit.EndUpdate 341
TRichViewEdit.ForceFieldHighlight 327
TRichViewEdit.GetCheckpointAtCaret 341
TRichViewEdit.GetCurrentBreakInfo 341
TRichViewEdit.GetCurrentBulletInfo 342
TRichViewEdit.GetCurrentCheckpoint 343
TRichViewEdit.GetCurrentControlInfo 343
TRichViewEdit.GetCurrentHotspotInfo 344
TRichViewEdit.GetCurrentItem 345
TRichViewEdit.GetCurrentItemEx 345
TRichViewEdit.GetCurrentItemExtraIntProperty 346
TRichViewEdit.GetCurrentItemExtraIntPropertyEx 346
TRichViewEdit.GetCurrentItemExtraStrProperty 346
TRichViewEdit.GetCurrentItemExtraStrPropertyEx 346
TRichViewEdit.GetCurrentItemText 347
TRichViewEdit.GetCurrentItemTextA 347
TRichViewEdit.GetCurrentItemTextW 347
TRichViewEdit.GetCurrentItemVAlign 348
TRichViewEdit.GetCurrentLineCol 348
TRichViewEdit.GetCurrentMisspelling 348
TRichViewEdit.GetCurrentPictureInfo 349
TRichViewEdit.GetCurrentTag 349
TRichViewEdit.GetCurrentTextInfo 350
TRichViewEdit.InsertBreak 350
TRichViewEdit.InsertBullet 351
TRichViewEdit.InsertCheckpoint 351
TRichViewEdit.InsertControl 352
TRichViewEdit.InsertHotPicture 352
TRichViewEdit.InsertHotspot 353
TRichViewEdit.InsertItem 353
TRichViewEdit.InsertOEMTextFromFile 354
TRichViewEdit.InsertPageBreak 355
TRichViewEdit.InsertPicture 355
TRichViewEdit.InsertRTFFromFileEd 356
TRichViewEdit.InsertRTFFromStreamEd 357
TRichViewEdit.InsertRVFFromFileEd 357
TRichViewEdit.InsertRVFFromStreamEd 358
TRichViewEdit.InsertStringATag 359
TRichViewEdit.InsertStringTag 359
TRichViewEdit.InsertStringWTag 359
TRichViewEdit.InsertTab 359
TRichViewEdit.InsertText 360
TRichViewEdit.InsertTextA 360
TRichViewEdit.InsertTextW 360
TRichViewEdit.LiveSpellingMode 327
TRichViewEdit.Modified 327
TRichViewEdit.OffsetInCurItem 328
TRichViewEdit.OnAfterOleDrop 388
TRichViewEdit.OnBeforeOleDrop 388
TRichViewEdit.OnCaretGetOut 389
TRichViewEdit.OnCaretMove 390
TRichViewEdit.OnChange 390
TRichViewEdit.OnChanging 390
TRichViewEdit.OnCheckStickingItems 391
TRichViewEdit.OnCurParaStyleChanged 391

TRichViewEdit.OnCurTextStyleChanged 391
TRichViewEdit.OnDrawCustomCaret 391
TRichViewEdit.OnDropFile 392
TRichViewEdit.OnDropFiles 393
TRichViewEdit.OnItemResize 393
TRichViewEdit.OnItemTextEdit 394
TRichViewEdit.OnMeasureCustomCaret 394
TRichViewEdit.OnOleDragEnter 395
TRichViewEdit.OnOleDragLeave 395
TRichViewEdit.OnOleDragOver 396
TRichViewEdit.OnOleDrop 397
TRichViewEdit.OnParaStyleConversion 397
TRichViewEdit.OnPaste 399
TRichViewEdit.OnSmartPopupClick 400
TRichViewEdit.OnStyleConversion 400
TRichViewEdit.Paste 361
TRichViewEdit.PasteBitmap 362
TRichViewEdit.PasteGraphicFile 362
TRichViewEdit.PasteMetafile 363
TRichViewEdit.PasteRTF 363
TRichViewEdit.PasteRVF 364
TRichViewEdit.PasteText 364
TRichViewEdit.PasteTextA 364
TRichViewEdit.PasteTextW 364
TRichViewEdit.PasteURL 365
TRichViewEdit.ReadOnly 328
TRichViewEdit.Redo 365
TRichViewEdit.RedoAction 366
TRichViewEdit.RedoName 366
TRichViewEdit.RemoveCheckpointAtCaret 366
TRichViewEdit.RemoveCheckpointEd 366
TRichViewEdit.RemoveCurrentCheckpoint 367
TRichViewEdit.RemoveCurrentPageBreak 367
TRichViewEdit.RemoveLists 367
TRichViewEdit.ResizeControl 368
TRichViewEdit.ResizeCurrentControl 368
TRichViewEdit.SearchText 369
TRichViewEdit.SearchTextA 369
TRichViewEdit.SearchTextW 369
TRichViewEdit.SelectCurrentLine 369
TRichViewEdit.SelectCurrentWord 370
TRichViewEdit.SetBackgroundImageEd 371
TRichViewEdit.SetBreakInfoEd 371
TRichViewEdit.SetBulletInfoEd 372
TRichViewEdit.SetCheckpointInfoEd 373
TRichViewEdit.SetControlInfoEd 374
TRichViewEdit.SetCurrentBreakInfo 375
TRichViewEdit.SetCurrentBulletInfo 375
TRichViewEdit.SetCurrentCheckpointInfo 376
TRichViewEdit.SetCurrentControlInfo 377
TRichViewEdit.SetCurrentHotspotInfo 377
TRichViewEdit.SetCurrentItemExtraIntProperty 378
TRichViewEdit.SetCurrentItemExtraIntPropertyEx 378
TRichViewEdit.SetCurrentItemExtraStrProperty 379
TRichViewEdit.SetCurrentItemExtraStrPropertyEx 379
TRichViewEdit.SetCurrentItemText 380
TRichViewEdit.SetCurrentItemTextA 380
TRichViewEdit.SetCurrentItemTextW 380
TRichViewEdit.SetCurrentItemVAlign 380
TRichViewEdit.SetCurrentPictureInfo 381
TRichViewEdit.SetCurrentTag 382
TRichViewEdit.SetFloatPropertyEd 370
TRichViewEdit.SetHotspotInfoEd 382
TRichViewEdit.SetIntPropertyEd 370
TRichViewEdit.SetItemExtraIntPropertyEd 383
TRichViewEdit.SetItemExtraIntPropertyExEd 383
TRichViewEdit.SetItemExtraStrPropertyEd 384
TRichViewEdit.SetItemExtraStrPropertyExEd 384
TRichViewEdit.SetItemTagEd 384
TRichViewEdit.SetItemTextEd 385
TRichViewEdit.SetItemTextEdA 385
TRichViewEdit.SetItemTextEdW 385
TRichViewEdit.SetItemVAlignEd 386
TRichViewEdit.SetPictureInfoEd 386
TRichViewEdit.SetStrPropertyEd 370
TRichViewEdit.SetUndoGroupMode 387
TRichViewEdit.SmartPopupProperties 329
TRichViewEdit.SmartPopupVisible 329
TRichViewEdit.TopLevelEditor 329
TRichViewEdit.Undo 387
TRichViewEdit.UndoAction 388
TRichViewEdit.UndoLimit 329
TRichViewEdit.UndoName 388
TRichViewRVData 650
TRichViewUnicodeInput 705
TRV_AfterImportGraphicsProc 720
TRV_CreateGraphicsFunction 720
TRVAddStyleEvent 274
TRVAfterApplyStyleEvent 443
TRVAlignment 483
TRVAnimationMode 169
TRVAnsiString 668
TRVApplyStyleColorEvent 444
TRVApplyStyleEvent 443
TRVAssignImageFileNameEvent 274
TRVBackgroundRect 500

TRVBackgroundRect.BorderOffsets 501
TRVBackgroundRect.Color 501
TRVBackgroundRect.Opacity 501
TRVBiDiMode 668
TRVBooleanRect 653
TRVBorder 501
TRVBorder.BorderOffsets 502
TRVBorder.Color 502
TRVBorder.InternalWidth 502
TRVBorder.Style 502
TRVBorder.VisibleBorders 502
TRVBorder.Width 503
TRVBorderStyle 669
TRVBoxBackgroundRect 635
TRVBoxBorder 635
TRVBoxHeightType 636
TRVBoxPosition 637
TRVBoxPosition.HorizontalAlignment 639
TRVBoxPosition.HorizontalAnchor 638
TRVBoxPosition.HorizontalOffset 639
TRVBoxPosition.HorizontalPositionKind 639
TRVBoxPosition.PositionInText 640
TRVBoxPosition.RelativeToCell 640
TRVBoxPosition.VerticalAlignment 642
TRVBoxPosition.VerticalAnchor 641
TRVBoxPosition.VerticalOffset 642
TRVBoxPosition.VerticalPositionKind 642
TRVBoxProperties 635
TRVBoxProperties.Background 635
TRVBoxProperties.Border 635
TRVBoxProperties.Height 636
TRVBoxProperties.HeightType 636
TRVBoxProperties.VAlign 636
TRVBoxProperties.Width 636
TRVBoxProperties.WidthType 636
TRVBoxWidthType 636
TRVBreakItemInfo 563
TRVBreakStyle 669
TRVBulletItemInfo 563
TRVCaretWidthMode 699
TRVCellEditingEvent 602
TRVCellVAlign 670
TRVCheckPointVisibleEvent 274
TRVClickMode 558
TRVCMHintShowEvent 275
TRVCodePage 664, 671
TRVColorMode 549, 671
TRVControlAction 275
TRVControlActionEvent 275
TRVControlData 650
TRVControlItemInfo 563
TRVCustomFormatEvent 410, 415, 416
TRVDBFieldFormat 413
TRVDblClickEvent 287
TRVDeleteUnusedStylesData 653
TRVDisplayOption 671
TRVDisplayOptions 671
TRVDocObject 174
TRVDocObjectCollection 174
TRVDocParameters 176, 310
TRVDocParameters.Author 311
TRVDocParameters.BottomMargin 311
TRVDocParameters.Comments 311
TRVDocParameters.ConvertToUnit 315
TRVDocParameters.FacingPages 312
TRVDocParameters.FooterY 312
TRVDocParameters.HeaderY 312
TRVDocParameters.LeftMargin 312
TRVDocParameters.MirrorMargins 313
TRVDocParameters.Orientation 313
TRVDocParameters.PageHeight 313
TRVDocParameters.PageWidth 313
TRVDocParameters.Reset 315
TRVDocParameters.ResetLayout 315
TRVDocParameters.RightMargin 313
TRVDocParameters.Title 314
TRVDocParameters.TitlePage 314
TRVDocParameters.TopMargin 314
TRVDocParameters.Units 314
TRVDocParameters.UnitsPerInch 315
TRVDocParameters.ZoomMode 314
TRVDocParameters.ZoomPercent 315
TRVDocRotation 672
TRVDocXSaveArea 293
TRVDragDropFormat 321
TRVDragDropFormats 321
TRVDrawCheckpointEvent 444, 556
TRVDrawCustomCaretEvent 391
TRVDrawHyperlinkEvent 556
TRVDrawPageBreakEvent 446
TRVDrawParaRectEvent 447
TRVDrawStyleTextEvent 448
TRVDrawTextBackEvent 449
TRVDropFileAction 393
TRVDropFileEvent 392
TRVDropFilesEvent 393
TRVEditRVData 650
TRVEndnoteItemInfo 563, 629

TRVEndnoteItemInfo.Create 630
TRVEndnoteItemInfo.CreateEx 630
TRVESearchOption 672
TRVESearchOptions 672
TRVExtraItemProperty 673
TRVExtraItemStrProperty 676
TRVFControlNeededEvent 288
TRVFImageListNeededEvent 288
TRVFixMarginsMode 518
TRVFloatHorizontalAlignment 639
TRVFloatPosition 677
TRVFloatPositionInText 640
TRVFloatPositionKind 678
TRVFloatProperty 678
TRVFloatSize 678
TRVFloatVerticalAlignment 642
TRVFontInfoClass 440
TRVFontInfoProperties 679
TRVFontInfoProperty 679
TRVFontStyle 475
TRVFontStyles 475
TRVFootnoteItemInfo 563, 630, 632
TRVFootnoteItemInfo.Create 632
TRVFootnoteItemInfo.CreateEx 632
TRVFootOrEndnoteItemInfo 627
TRVFOption 188
TRVFOptions 188
TRVFPictureNeededEvent 289
TRVFReaderStyleMode 679
TRVFullLineItemInfo 563
TRVFWarning 191
TRVFWarnings 191
TRVGetItemCursorEvent 277
TRVGetOutDirection 389
TRVGetPageNumberEvent 543
TRVGraphicHandler 654
TRVGraphicItemInfo 563
TRVHFType 680
TRVHorizontalAnchor 638
TRVHorizontalCaretPosition 699
TRVHotGraphicItemInfo 563
TRVHotspotItemInfo 563
TRVHoverEffect 471
TRVHoverEffects 471
TRVHTMLLength 680
TRVHTMLSaveArea 295
TRVHTMLSaveImageEvent 277
TRVImportPictureEvent 279
TRVIntegerList 656
TRVIntProperty 681
TRVItemAction 280
TRVItemActionEvent 280
TRVItemBackgroundStyle 682
TRVItemFormattedData 650
TRVItemHintEvent 281
TRVItemResizeEvent 393
TRVItemTextEditEvent 394
TRVLabelItemInfo 563, 617
TRVLabelItemInfo.Alignment 618
TRVLabelItemInfo.Create 620
TRVLabelItemInfo.CreateEx 620
TRVLabelItemInfo.Cursor 618
TRVLabelItemInfo.MinWidth 618
TRVLabelItemInfo.ProtectTextStyleNo 619
TRVLabelItemInfo.RemoveInternalLeading 619
TRVLabelItemInfo.RemoveSpaceBelow 619
TRVLabelItemInfo.Text 619
TRVLabelItemInfo.TextStyleNo 620
TRVLastLineAlignment 483
TRVLength 682
TRVLineSpacingType 486
TRVLineSpacingValue 486
TRVLineWrapMode 428
TRVListInfo 493
TRVListInfo.AllNumbered 494
TRVListInfo.HasNumbering 494
TRVListInfo.Levels 493
TRVListInfo.OneLevelPreview 493
TRVListInfoClass 439
TRVListInfos 458
TRVListInfos.Add 459
TRVListInfos.AssignTo 459
TRVListInfos.FindStyleWithLevels 459
TRVListInfos.FindSuchStyle 459
TRVListInfos.Items 459
TRVListLevel 506
TRVListLevel.FirstIndent 507
TRVListLevel.Font 507
TRVListLevel.FormatString 508
TRVListLevel.FormatStringW 509
TRVListLevel.HasNumbering 513
TRVListLevel.ImageIndex 509
TRVListLevel.ImageList 509
TRVListLevel.LeftIndent 510
TRVListLevel.ListType 510
TRVListLevel.MarkerAlignment 511
TRVListLevel.MarkerIndent 512
TRVListLevel.Options 512

TRVListLevel.Picture 513
TRVListLevel.StartFrom 513
TRVListLevelCollection 506
TRVListLevelCollection.Add 506
TRVListLevelCollection.Items 506
TRVListLevelOption 512
TRVListLevelOptions 512
TRVListType 510
TRVLiveSpellingMode 327
TRVLoadFormat 668
TRVLongOperation 283
TRVMarginsPen 559
TRVMarkerAlignment 511
TRVMarkerItemInfo 563
TRVMathDocObject 656
TRVMathItemInfo 621
TRVMathItemInfo.Create 623
TRVMathItemInfo.DisplayInline 621
TRVMathItemInfo.FontName 622
TRVMathItemInfo.FontSizeDouble 622
TRVMathItemInfo.Text 622
TRVMathItemInfo.TextColor 623
TRVMeasureCustomCaretEvent 394
TRVMouseEvent 289, 291
TRVMouseMoveEvent 290
TRVNonTextItemInfo 563
TRVNoteData 650
TRVNoteReferenceItemInfo 563, 644
TRVNoteReferenceItemInfo.Create 645
TRVNoteReferenceItemInfo.CreateEx 646
TRVOfficeCnvList 539
TRVOfficeCnvList.Count 539
TRVOfficeCnvList.Items 539
TRVOfficeConverter 532
TRVOfficeConverter.Create 536
TRVOfficeConverter.Destroy 536
TRVOfficeConverter.ErrorCode 533
TRVOfficeConverter.ExcludeDocXExportConverter 534
TRVOfficeConverter.ExcludeDocXImportConverter 534
TRVOfficeConverter.ExcludeHTMLExportConverter 534
TRVOfficeConverter.ExcludeHTMLImportConverter 534
TRVOfficeConverter.ExportConverters 535
TRVOfficeConverter.ExportRTF 536
TRVOfficeConverter.ExportRV 537
TRVOfficeConverter.ExtensionsInFilter 535
TRVOfficeConverter.GetExportFilter 537
TRVOfficeConverter.GetImportFilter 537
TRVOfficeConverter.ImportConverters 535
TRVOfficeConverter.ImportRTF 538
TRVOfficeConverter.ImportRV 538
TRVOfficeConverter.IsValidImporter 539
TRVOfficeConverter.OnConverting 539
TRVOfficeConverter.PreviewMode 535
TRVOfficeConverter.Stream 536
TRVOfficeConverterInfo 539
TRVOfficeConverterInfo.Filter 539
TRVOfficeConverterInfo.Name 539
TRVOfficeConverterInfo.Path 539
TRVOleDragEnterEvent 395
TRVOleDragOverEvent 396
TRVOleDropEffect 682
TRVOleDropEffects 682
TRVOleDropEvent 397
TRVOnCaretGetOutEvent 389
TRVOpacity 683
TRVOption 182
TRVOptions 182
TRVPageBreakType 446
TRVPageCountItemInfo 648
TRVPageCountItemInfo.Create 648
TRVPageCountItemInfo.CreateEx 649
TRVPageNumberItemInfo 647
TRVPageNumberItemInfo.Create 647
TRVPageNumberItemInfo.CreateEx 648
TRVPageNumberType 683
TRVPagePrepaintEvent 530
TRVPageSet 683
TRVPaintEvent 283
TRVPaletteAction 177
TRVPAPen 560
TRVParagraphMarks 711
TRVParaInfoClass 440
TRVParaInfoProperties 684
TRVParaInfoProperty 684
TRVParaOption 487
TRVParaOptions 487
TRVPasteEvent 399
TRVPrint 513
TRVPrint.AssignDocParameters 522
TRVPrint.AssignSource 522
TRVPrint.ClipMargins 518
TRVPrint.ContinuousPrint 523
TRVPrint.ConvertToUnits 523
TRVPrint.Create 524

TRVPrint.DrawPage 524
TRVPrint.DrawPageAt 524
TRVPrint.DrawPreview 524
TRVPrint.FacingPages 518
TRVPrint.FixMarginsMode 518
TRVPrint.FooterY 519
TRVPrint.FormatPages 525
TRVPrint.GetFooterRect 525
TRVPrint.GetHeaderRect 525
TRVPrint.HeaderY 519
TRVPrint.MakePreview 526
TRVPrint.MakeScaledPreview 526
TRVPrint.Margins 520
TRVPrint.MirrorMargins 521
TRVPrint.OnFormatting 529
TRVPrint.OnPagePostpaint 530
TRVPrint.OnPagePrepaint 530
TRVPrint.OnSendingToPrinter 531
TRVPrint.Preview100PercentHeight 521
TRVPrint.Preview100PercentWidth 521
TRVPrint.Print 526
TRVPrint.PrintPages 527
TRVPrint.SetFooter 528
TRVPrint.SetHeader 529
TRVPrint.TitlePage 521
TRVPrint.Units 521
TRVPrint.VirtualPrinter 522
TRVPrintComponentEvent 554
TRVPrintingEvent 529, 531
TRVPrintingStep 684
TRVPrintPreview 417
TRVPrintPreview.RVPrint 418
TRVProgressEvent 283
TRVProgressStage 283
TRVProtectOption 473
TRVProtectOptions 473
TRVRawByteString 685
TRVReaderStyleMode 685
TRVReaderUnicode 310
TRVReadHyperlink 284
TRVReadMergeFieldEvent 286
TRVRect 657
TRVRectItemInfo 563
TRVReportHelper 540
TRVReportHelper.AllowTransparentDrawing 541
TRVReportHelper.DrawPage 541
TRVReportHelper.DrawPageAt 542
TRVReportHelper.Finished 542
TRVReportHelper.FormatNextPage 542
TRVReportHelper.GetLastPageHeight 543
TRVReportHelper.Init 543
TRVReportHelper.OnGetPageNumber 543
TRVReportHelper.RichView 541
TRVReportHelper.UpdatePageNumbers 543
TRVReportHelperWithHeaderFooters 657
TRVReportHelperWithHeaderFooters.AssignToRVPrint 658
TRVReportHelperWithHeaderFooters.Create 658
TRVReportHelperWithHeaderFooters.GetFooter 658
TRVReportHelperWithHeaderFooters.GetHeader 659
TRVReportHelperWithHeaderFooters.MainDoc 657
TRVReportHelperWithHeaderFooters.Reset 659
TRVReportHelperWithHeaderFooters.UseStyleTemplates 658
TRVRightClickEvent 291
TRVRTFErrorCode 306
TRVRTFHighlight 305
TRVRTFOption 185
TRVRTFOptions 185
TRVRTFReaderProperties 304
TRVRTFReaderProperties.AutoHideTableGridLines 304
TRVRTFReaderProperties.BasePathLinks 305
TRVRTFReaderProperties.CharsetForUnicode 305
TRVRTFReaderProperties.ConvertHighlight 305
TRVRTFReaderProperties.ErrorCode 306
TRVRTFReaderProperties.ExtractMetafileBitmaps 306
TRVRTFReaderProperties.IgnoreBookmarks 306
TRVRTFReaderProperties.IgnoreFields 307
TRVRTFReaderProperties.IgnoreNotes 307
TRVRTFReaderProperties.IgnorePictures 307
TRVRTFReaderProperties.IgnoreSequences 307
TRVRTFReaderProperties.IgnoreTables 307
TRVRTFReaderProperties.LineBreaksAsParagraphs 307
TRVRTFReaderProperties.ParaStyleMode 308
TRVRTFReaderProperties.ParaStyleNo 308
TRVRTFReaderProperties.ReadDocParameters 308
TRVRTFReaderProperties.SkipHiddenText 309
TRVRTFReaderProperties.TextStyleMode 309
TRVRTFReaderProperties.TextStyleNo 309
TRVRTFReaderProperties.UnicodeMode 310
TRVRTFReaderProperties.UseCharsetForUnicode 310
TRVRTFReaderProperties.UseHypertextStyles 310
TRVRTFReaderProperties.UseSingleCellPadding 310

TRVRTFSaveArea 299
TRVSaveComponentToFileEvent 292
TRVSaveCSSOption 441
TRVSaveCSSOptions 441
TRVSaveDocXExtraEvent 293
TRVSaveFormat 685
TRVSaveHTMLExtraEvent 295
TRVSaveImageEvent2 296
TRVSaveOption 686
TRVSaveOptions 686
TRVSaveParaToHTMLEvent 298
TRVSaveRTFExtraEvent 299
TRVScroller 544
TRVScroller.BorderStyle 545
TRVScroller.Create 548
TRVScroller.HScrollMax 545
TRVScroller.HScrollPos 545
TRVScroller.HScrollVisible 545
TRVScroller.InplaceEditor 546
TRVScroller.NoVScroll 546
TRVScroller.OnHScrolled 548
TRVScroller.OnVScrolled 549
TRVScroller.ScrollTo 548
TRVScroller.Tracking 546
TRVScroller.UseXPThemes 546
TRVScroller.VScrollMax 547
TRVScroller.VScrollPos 547
TRVScroller.VScrollVisible 547
TRVScroller.WheelStep 548
TRVSearchOption 689
TRVSearchOptions 689
TRVSelection 666
TRVSelectionHandleKind 431
TRVSelectionMode 432
TRVSelectionStyle 432
TRVSeqItemInfo 563, 624
TRVSeqItemInfo.Create 626
TRVSeqItemInfo.CreateEx 627
TRVSeqItemInfo.FormatString 625
TRVSeqItemInfo.NumberType 625
TRVSeqItemInfo.Reset 626
TRVSeqItemInfo.SeqName 626
TRVSeqItemInfo.StartFrom 626
TRVSeqType 689
TRVSidenoteItemInfo.BoxPosition 634
TRVSidenoteItemInfo.BoxProperties 634
TRVSidenoteItemInfo.Create 634
TRVSidenoteItemInfo.CreateEx 634
TRVSmartPopupClickEvent 400
TRVSmartPopupPosition 404
TRVSmartPopupProperties 402
TRVSmartPopupProperties.ButtonType 402
TRVSmartPopupProperties.Color 403
TRVSmartPopupProperties.Hint 403
TRVSmartPopupProperties.HoverColor 403
TRVSmartPopupProperties.HoverLineColor 403
TRVSmartPopupProperties.ImageIndex 404
TRVSmartPopupProperties.ImageList 404
TRVSmartPopupProperties.LineColor 404
TRVSmartPopupProperties.Menu 404
TRVSmartPopupProperties.Position 404
TRVSmartPopupProperties.SetButtonState 405
TRVSmartPopupProperties.ShortCut 404
TRVSpecialCharacter 714
TRVSpecialCharacters 714
TRVSpellingCheckEvent 301
TRVSTFontInfo 497
TRVStickingItemsEvent 391
TRVSTParaInfo 497
TRVStringFormatMethod 704
TRVStrProperty 690
TRVStyle 418
TRVStyle.AddTextStyle 438
TRVStyle.CheckpointColor 420
TRVStyle.CheckpointEvColor 420
TRVStyle.Color 421
TRVStyle.ConvertToDifferentUnits 438
TRVStyle.ConvertToPixels 438
TRVStyle.ConvertToTwips 438
TRVStyle.Create 438
TRVStyle.CurrentItemColor 421
TRVStyle.DefCodePage 421
TRVStyle.DefTabWidth 422
TRVStyle.DefUnicodeStyle 422
TRVStyle.DeleteTextStyle 438
TRVStyle.Destroy 438
TRVStyle.DifferentUnitsToUnits 439
TRVStyle.DisabledFontColor 422
TRVStyle.EndnoteNumbering 422
TRVStyle.FieldHighlightColor 423
TRVStyle.FieldHighlightType 423
TRVStyle.FindParaStyle 440
TRVStyle.FindTextStyle 440
TRVStyle.FloatingLineColor 423
TRVStyle.FontQuality 424
TRVStyle.FootnoteNumbering 422
TRVStyle.FootnotePageReset 425
TRVStyle.FullRedraw 425

TRVStyle.GetAsDifferentUnits 439
TRVStyle.GetAsPixels 439
TRVStyle.GetAsRVUnits 439
TRVStyle.GetAsTwips 439
TRVStyle.GetListStyleClass 439
TRVStyle.GetParaStyleClass 440
TRVStyle.GetTextStyleClass 440
TRVStyle.GridColor 425
TRVStyle.GridReadOnlyColor 425
TRVStyle.GridReadOnlyStyle 425
TRVStyle.GridStyle 425
TRVStyle.HoverColor 426
TRVStyle.InactiveSelColor 426
TRVStyle.InactiveSelTextColor 426
TRVStyle.InvalidPicture 427
TRVStyle.JumpCursor 427
TRVStyle.LineSelectCursor 427
TRVStyle.LineWrapMode 428
TRVStyle.ListStyles 429
TRVStyle.LiveSpellingColor 430
TRVStyle.LoadINI 440
TRVStyle.LoadReg 441
TRVStyle.MainRVStyle 430
TRVStyle.OnAfterApplyStyle 443
TRVStyle.OnApplyStyle 443
TRVStyle.OnApplyStyleColor 444
TRVStyle.OnDrawCheckpoint 444
TRVStyle.OnDrawPageBreak 446
TRVStyle.OnDrawParaBack 447
TRVStyle.OnDrawStyleText 448
TRVStyle.OnDrawTextBack 449
TRVStyle.OnStyleHoverSensitive 450
TRVStyle.PageBreakColor 431
TRVStyle.ParaStyles 431
TRVStyle.PixelsToUnits 439
TRVStyle.RVUnitsToUnits 439
TRVStyle.SaveCSS 441
TRVStyle.SaveCSSToStream 442
TRVStyle.SaveINI 442
TRVStyle.SaveReg 442
TRVStyle.SelColor 426
TRVStyle.SelectionHandleKind 431
TRVStyle.SelectionMode 432
TRVStyle.SelectionStyle 432
TRVStyle.SelTextColor 426
TRVStyle.SidenoteNumbering 422
TRVStyle.SoftPageBreakColor 434
TRVStyle.SpacesInTab 435
TRVStyle.SpecialCharactersColor 435
TRVStyle.StyleTemplates 435
TRVStyle.TextBackgroundKind 435
TRVStyle.TextEngine 436
TRVStyle.TextStyles 437
TRVStyle.TwipsToUnits 439
TRVStyle.Units 437
TRVStyle.UseSound 437
TRVStyleConversionEvent 397, 400
TRVStyleHoverSensitiveEvent 450
TRVStyleLength 690
TRVStyleTemplate 494
TRVStyleTemplate.ApplyToParaStyle 498
TRVStyleTemplate.ApplyToTextStyle 499
TRVStyleTemplate.Id 495
TRVStyleTemplate.Kind 495
TRVStyleTemplate.Name 496
TRVStyleTemplate.Next 496
TRVStyleTemplate.NextId 496
TRVStyleTemplate.ParaStyle 497
TRVStyleTemplate.Parent 497
TRVStyleTemplate.ParentId 497
TRVStyleTemplate.QuickAccess 497
TRVStyleTemplate.TextStyle 497
TRVStyleTemplate.UpdateModifiedParaStyleProperties 500
TRVStyleTemplate.UpdateModifiedTextStyleProperties 500
TRVStyleTemplate.ValidParaProperties 498
TRVStyleTemplate.ValidTextProperties 498
TRVStyleTemplateCollection 460
TRVStyleTemplateCollection.AssignStyleTemplates 461
TRVStyleTemplateCollection.AssignToStrings 461
TRVStyleTemplateCollection.ClearParaFormat 462
TRVStyleTemplateCollection.ClearTextFormat 462
TRVStyleTemplateCollection.DefStyleName 460
TRVStyleTemplateCollection.FindById 462
TRVStyleTemplateCollection.FindByName 462
TRVStyleTemplateCollection.FindItemById 462
TRVStyleTemplateCollection.FindItemByName 462
TRVStyleTemplateCollection.ForbiddenIds 461
TRVStyleTemplateCollection.InsertFromRVST 463
TRVStyleTemplateCollection.Items 461
TRVStyleTemplateCollection.LoadFromRVST 462
TRVStyleTemplateCollection.NormalStyleTemplate 461
TRVStyleTemplateCollection.ResetNameCounter 463
TRVStyleTemplateCollection.SaveToRVST 462

TRVStyleTemplateId 690
TRVStyleTemplateInsertMode 193
TRVStyleTemplateKind 495
TRVStyleTemplateName 691
TRVStyleUnits 691
TRVSubData 650
TRVSubSuperScriptType 476
TRVTabAlign 504
TRVTabInfo 504
TRVTabInfo.Align 504
TRVTabInfo.Leader 505
TRVTabInfo.Position 505
TRVTabInfos 503
TRVTabInfos.AddFrom 503
TRVTabInfos.DeleteList 504
TRVTabInfos.Find 504
TRVTabInfos.Intersect 504
TRVTabInfos.Items 503
TRVTabItemInfo 563
TRVTableBorderStyle 691
TRVTableCellData 605, 650
TRVTableCellData.BackgroundImage 605
TRVTableCellData.BackgroundImageFileName 606
TRVTableCellData.BackgroundStyle 606
TRVTableCellData.BestHeight 606
TRVTableCellData.BestWidth 607
TRVTableCellData.BorderColor 607
TRVTableCellData.BorderLightColor 608
TRVTableCellData.Color 608
TRVTableCellData.ColSpan 608
TRVTableCellData.GetRVData 613
TRVTableCellData.Hint 609
TRVTableCellData.Opacity 609
TRVTableCellData.Options 609
TRVTableCellData.properties 614
TRVTableCellData.Rotation 611
TRVTableCellData.RowSpan 612
TRVTableCellData.Tag 612
TRVTableCellData.VAlign 613
TRVTableCellData.VisibleBorders 613
TRVTableCellOption 609
TRVTableCellOptions 609
TRVTableDrawBackEvent 603
TRVTableDrawBorderEvent 603
TRVTableItemInfo 563, 565
TRVTableItemInfo.AssignProperties 584
TRVTableItemInfo.BackgroundImage 567
TRVTableItemInfo.BackgroundImageFileName 568
TRVTableItemInfo.BackgroundStyle 568
TRVTableItemInfo.BestWidth 568
TRVTableItemInfo.BorderColor 569
TRVTableItemInfo.BorderHSpacing 569
TRVTableItemInfo.BorderLightColor 569
TRVTableItemInfo.BorderStyle 570
TRVTableItemInfo.BorderVSpacing 570
TRVTableItemInfo.BorderWidth 571
TRVTableItemInfo.CanMergeCells 584
TRVTableItemInfo.CanMergeSelectedCells 585
TRVTableItemInfo.CellBorderColor 571
TRVTableItemInfo.CellBorderLightColor 571
TRVTableItemInfo.CellBorderStyle 572
TRVTableItemInfo.CellBorderWidth 572
TRVTableItemInfo.CellHPadding 573
TRVTableItemInfo.CellHSpacing 573
TRVTableItemInfo.CellOverrideColor 573
TRVTableItemInfo.CellPadding 574
TRVTableItemInfo.Cells 574
TRVTableItemInfo.CellVPadding 574
TRVTableItemInfo.CellVSpacing 574
TRVTableItemInfo.Changed 585
TRVTableItemInfo.ColBandSize 575
TRVTableItemInfo.ColCount 575
TRVTableItemInfo.Color 575
TRVTableItemInfo.Create 585
TRVTableItemInfo.CreateEx 586
TRVTableItemInfo.DeleteCols 586
TRVTableItemInfo.DeleteEmptyCols 587
TRVTableItemInfo.DeleteEmptyRows 587
TRVTableItemInfo.DeleteRows 587
TRVTableItemInfo.DeleteSelectedCols 588
TRVTableItemInfo.DeleteSelectedRows 588
TRVTableItemInfo.Deselect 588
TRVTableItemInfo.Destroy 589
TRVTableItemInfo.EditCell 589
TRVTableItemInfo.EvenColumnsColor 583
TRVTableItemInfo.EvenRowsColor 583
TRVTableItemInfo.FirstColumnColor 583
TRVTableItemInfo.GetCellAt 589
TRVTableItemInfo.GetEditedCell 590
TRVTableItemInfo.GetNormalizedSelectionBounds 590
TRVTableItemInfo.GetSelectionBounds 591
TRVTableItemInfo.HeadingRowColor 583
TRVTableItemInfo.HeadingRowCount 576
TRVTableItemInfo.HOutermostRule 576
TRVTableItemInfo.HRuleColor 576
TRVTableItemInfo.HRuleWidth 577
TRVTableItemInfo.InsertCols 591

TRVTableItemInfo.InsertColsLeft 592
TRVTableItemInfo.InsertColsRight 592
TRVTableItemInfo.InsertRows 592
TRVTableItemInfo.InsertRowsAbove 593
TRVTableItemInfo.InsertRowsBelow 593
TRVTableItemInfo.IsCellSelected 593
TRVTableItemInfo.LastColumnColor 583
TRVTableItemInfo.LastRowColor 583
TRVTableItemInfo.LoadFromStream 593
TRVTableItemInfo.MergeCells 594
TRVTableItemInfo.MergeSelectedCells 594
TRVTableItemInfo.OddColumnsColor 583
TRVTableItemInfo.OddRowsColor 583
TRVTableItemInfo.OnCellEditing 602
TRVTableItemInfo.OnDrawBackground 603
TRVTableItemInfo.OnDrawBorder 603
TRVTableItemInfo.Opacity 577
TRVTableItemInfo.Options 577
TRVTableItemInfo.PrintOptions 579
TRVTableItemInfo.RowBandSize 575
TRVTableItemInfo.RowCount 580
TRVTableItemInfo.Rows 580
TRVTableItemInfo.SaveRowsToStream 595
TRVTableItemInfo.SaveToStream 595
TRVTableItemInfo.Select 595
TRVTableItemInfo.SelectCols 595
TRVTableItemInfo.SelectRows 596
TRVTableItemInfo.SetCellBackgroundImage 596
TRVTableItemInfo.SetCellBackgroundImageFileName 596
TRVTableItemInfo.SetCellBackgroundStyle 597
TRVTableItemInfo.SetCellBestHeight 597
TRVTableItemInfo.SetCellBestWidth 597
TRVTableItemInfo.SetCellBorderColor 597
TRVTableItemInfo.SetCellBorderLightColor 598
TRVTableItemInfo.SetCellColor 598
TRVTableItemInfo.SetCellHint 598
TRVTableItemInfo.SetCellOpacity 598
TRVTableItemInfo.SetCellOptions 599
TRVTableItemInfo.SetCellRotation 599
TRVTableItemInfo.SetCellTag 599
TRVTableItemInfo.SetCellVAlign 599
TRVTableItemInfo.SetCellVisibleBorders 600
TRVTableItemInfo.SetRowKeepTogether 600
TRVTableItemInfo.SetRowPageBreakBefore 600
TRVTableItemInfo.SetRowVAlign 600
TRVTableItemInfo.SetTableVisibleBorders 600
TRVTableItemInfo.SplitSelectedCellsHorizontally 600
TRVTableItemInfo.SplitSelectedCellsVertically 601
TRVTableItemInfo.TextColSeparator 581
TRVTableItemInfo.TextRowSeparator 581
TRVTableItemInfo.UnmergeCells 601
TRVTableItemInfo.UnmergeSelectedCells 601
TRVTableItemInfo.VisibleBorders 581
TRVTableItemInfo.VOutermostRule 582
TRVTableItemInfo.VRuleColor 582
TRVTableItemInfo.VRuleWidth 582
TRVTableOption 577
TRVTableOptions 577
TRVTablePrintOption 579
TRVTablePrintOptions 579
TRVTableRow 615
TRVTableRow.Items 615
TRVTableRow.KeepTogether 615
TRVTableRow.PageBreakBefore 616
TRVTableRow.VAlign 616
TRVTableRows 616
TRVTableRows.GetMainCell 617
TRVTableRows.Items 617
TRVTabNavigationType 194
TRVTag 692
TRVTextBackgroundKind 435
TRVTextBoxItemInfo 643
TRVTextBoxItemInfo.Create 643
TRVTextBoxItemInfo.CreateEx 643
TRVTextDrawState 692
TRVTextDrawStates 692
TRVTextEngine 436
TRVTextItemInfo 563
TRVTextOption 472
TRVTextOptions 472
TRVThumbnailMaker 659
TRVTInplaceRVData 650
TRVUnderlineType 476
TRVUndoType 693
TRVUnicodeString 694
TRVUnicodeTestResult 664
TRVUnits 694
TRVUnitsRect 660
TRVVAlign 695
TRVVerticalAnchor 641
TRVVirtualPrinterProperties 532
TRVVirtualPrinterProperties.Active 532
TRVVirtualPrinterProperties.PageHeight 532
TRVVirtualPrinterProperties.PageWidth 532
TRVWriteHyperlink 302
TRVZoomMode 561

TwipsToUnits 439
TRVStyle 439

- U -

UnderlineColor 476
TCustomRVFontInfo 476
UnderlineType 476
TCustomRVFontInfo 476
Undo 387
TRichViewEdit 387
Undo in RichViewEdit 155
Undo in TRichViewEdit 88
Undo/redo in Tables 155
UndoAction 388
TRichViewEdit 388
UndoLimit 329
TRichViewEdit 329
UndoName 388
TRichViewEdit 388
Unicode 99, 480
TFontInfo 480
Unicode in TRichView 99
UnicodeMode 310
TRVRTFReaderProperties 310
Units 314, 437, 521
TRVDocParameters 314
TRVPrint 521
TRVStyle 437
Units of measurement 106
UnitsPerInch 315
TRVDocParameters 315
UnmergeCells 601
TRVTableItemInfo 601
UnmergeSelectedCells 601
TRVTableItemInfo 601
UpdateModifiedParaStyleProperties 500
TRVStyleTemplate 500
UpdateModifiedTextStyleProperties 500
TRVStyleTemplate 500
UpdatePageNumbers 543
TRVReportHelper 543
UpdatePaletteInfo 273, 554
TCustomRVPrint 554
TRichView 273
URLFormatName 698
UseCharsetForUnicode 310
TRVRTFReaderProperties 310
UseHypertextStyles 310
TRVRTFReaderProperties 310
UseSingleCellPadding 310
TRVRTFReaderProperties 310
UseSound 437
TRVStyle 437
UseStyleTemplates 195, 658
TRichView 195
TRVReportHelperWithHeaderFooters 658
UseVCLThemes 195
TRichView 195
UseXPThemes 546
TRVScroller 546

- V -

Valid documents 105
Valid TRichView documents 105
ValidParaProperties 498
TRVStyleTemplate 498
ValidTextProperties 498
TRVStyleTemplate 498
VAlign 195, 613, 616, 636
TRichView 195
TRVBoxProperties 636
TRVTableCellData 613
TRVTableRow 616
VerticalAlignment 642
TRVBoxPosition 642
VerticalAnchor 641
TRVBoxPosition 641
VerticalOffset 642
TRVBoxPosition 642
VerticalPositionKind 642
TRVBoxPosition 642
Viewer vs Editor 102
VirtualPrinter 522
TRVPrint 522
VisibleBorders 502, 581, 613
TRVBorder 502
TRVTableCellData 613
TRVTableItemInfo 581
VOutermostRule 582
TRVTableItemInfo 582
VRuleColor 582
TRVTableItemInfo 582
VRuleWidth 582
TRVTableItemInfo 582
VScrollMax 547
TRVScroller 547
VScrollPos 547
TRVScroller 547

VScrollVisible 547
TRVScroller 547

VShift 477
TCustomRVFontInfo 477

VSmallStep 196
TRichView 196

- W -

WheelStep 548
TRVScroller 548

Width 503, 636
TRVBorder 503
TRVBoxProperties 636

WidthType 636
TRVBoxProperties 636

WordWrap 196
TRichView 196

- Z -

ZoomInCursor 560
TCustomRVPrintPreview 560

ZoomMode 314, 561
TCustomRVPrintPreview 561
TRVDocParameters 314

ZoomOutCursor 561
TCustomRVPrintPreview 561

ZoomPercent 315, 562
TCustomRVPrintPreview 562
TRVDocParameters 315